BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC NÔNG NGHIỆP HÀ NỘI

VŨ KIM THÀNH

TOÁN RỜI RẠC

(Giáo trình dành cho sinh viên ngành công nghệ thông tin)

Hà nội 2008

MỤC LỤC

LỜI NÓI ĐẦU

Toán Rời rạc (Discrete mathematics) là môn toán học nghiên cứu các đối tượng rời rạc. Nó được ứng dụng trong nhiều ngành khoa học khác nhau, đặc biệt là trong tin học bởi quá trình xử lý thông tin trên máy tính thực chất là một quá trình rời rạc.

Phạm vi nghiên cứu của Toán Rời rạc rất rộng, có thể chia thành các môn học khác nhau. Theo quy định của chương trình môn học, giáo trình này đề cập đến các lĩnh vực: Thuật toán và bài toán đếm; Lý thuyết đồ thị; Đại số Logic và được chia thành 8 chương:

- Chương 1 đề cập đến một trong các vấn đề cơ bản nhất của *Thuật toán* đó là độ phức tạp về thời gian của thuật toán.
- Chương 2 nói về các nguyên lý cơ bản của *Bài toán đếm.*
- Các chương 3, 4, 5 và 6 trình bày về *Lý thuyết đồ thị và các ứng dụng*. Đây là phần chiếm tỷ trọng nhiều nhất của giáo trình. Trong đó có các chương về các khái niệm cơ bản của đồ thị, các đồ thị đặc biệt như đồ thị Euler, đồ thị Hamilton, đồ thị phẳng, Cây cùng các ứng dụng của các đồ thi đặc biệt này. Riêng chương 6 dành cho một vấn đề trọng là một số bài toán tối ưu trên đồ thị hoặc bài toán tối ưu được giải bằng cách ứng dụng lý thuyết đồ thị.
- Chương 7 là các kiến thức cơ bản về *Đại số Boole,* một công cụ hữu hiệu trong việc thiết kế các mạch điện, điện tử.

Cuối giáo trình là phụ chương: *Những khái niệm cơ bản về toán Logic* để người học có thể tự nghiên cứu thêm về Toán Logic.

Trong mỗi chương chúng tôi cố gắng trình bày các kiến thức cơ bản nhất của chương đó cùng các thí dụ minh họa cụ thể. Vì khuôn khổ số tiết học nên chúng tôi lược bỏ một số chứng minh phức tạp. Cuối mỗi chương đều có các bài tập để người học ứng dụng, kiểm chứng các lý thuyết đã học, đồng thời cũng cung cấp một số đáp số của các bài tập đã cho.

Cũng cần nói thêm rằng toán Rời rạc không chỉ được ứng dụng trong tin học mà còn được ứng dụng trong nhiều ngành khoa học khác. Bởi vậy giáo trình cũng có ích cho những ai cần quan tâm đến các ứng dụng khác của môn học này.

Tác giả xin chân thành cảm ơn các bạn đồng nghiệp đã động viên và góp ý cho việc biên soạn giáo trình này. Đặc biệt chúng tôi xin cảm ơn Nhà giáo ưu tú Nguyễn Đình Hiền đã hiệu đính và cho nhiều ý kiến đóng góp bổ ích và thiết thực.

Vì trình độ có hạn và giáo trình được biên soạn lần đầu nên không tránh khỏi các thiếu sót. Tác giả rất mong nhận được các ý kiến đóng góp của các đồng nghiệp và bạn đọc về các khiếm khuyết của cuốn sách.

TÁC GIẢ

CHƯƠNG1.

THUẬT TOÁN

1. Định nghĩa.
2. Mô tả thuật toán bằng lưu đồ.
3. Mô tả thuật toán bằng ngôn ngữ phỏng Pascal.
   - 3.1. Câu lệnh Procedure (thủ tục) hoặc Function (hàm).
   - 3.2. Câu lệnh gán.
   - 3.3. Khối câu lệnh tuần tự.
   - 3.4. Câu lệnh diều kiện.
   - 3.5. Các câu lệnh lặp.
4. Độ phức tạp của thuật toán.
   - 4.1. Khái niệm độ tăng của hàm.
   - 4.2. Độ tăng của tổ hợp các hàm.
   - 4.3. Độ phức tạp của thuật toán.
5. Thuật toán tìm kiếm
   - 5.1. Thuật toán tìm kiếm tuyến tính (còn gọi là thuật toán tìm kiếm tuần tự).
   - 5.2. Thuật toán tìm kiếm nhị phân.
6. Thuật toán đệ quy.
   - 6.1. Công thức truy hồi.
   - 6.2. Thuật toán đệ quy.
   - 6.3. Đệ quy và lặp
7. Một số thuật toán về số nguyên.
   - 7.1. Biểu diễn các số nguyên.
   - 7.2. Cộng và nhân trong hệ nhị phân.

1. Định nghĩa

Thuật toán (algorithm) là một dãy các quy tắc nhằm xác định một dãy các thao tác trên các đối tượng sao cho **sau một số hữu hạn** bước thực hiện sẽ **đạt được mục tiêu đặt ra.**

Từ định nghĩa của thuật toán cho thấy các đặc trưng (tính chất) cơ bản của thuật toán là:

a. Yếu tố vào, ra:

- Đầu vào (Input): Mỗi thuật toán có một giá trị hoặc một bộ giá trị đầu vào từ một tập xác định đã được chỉ rõ.
- Đầu ra (Output): Từ các giá trị đầu vào, thuật toán cho ra các giá trị cần tìm gọi là kết quả của bài toán.

b. Tính dừng:

Sau một số hữu hạn bước thao tác thuật toán phải kết thúc và cho kết quả .

c. Tính xác định:

Các thao tác phải rõ ràng, cho cùng một kết quả dù được xử lý trên các bộ xử lý khác nhau (hai bộ xử lý trong cùng một điều kiện không thể cho hai kết quả khác nhau).

d. Tính hiệu quả

Sau khi đưa dữ liệu vào cho thuật toán hoạt động phải đưa ra kết quả như ý muốn.

e. Tính tổng quát

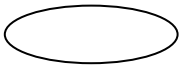
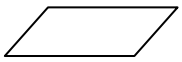
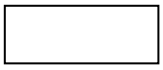
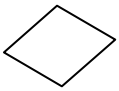
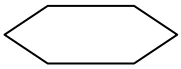
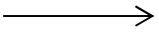
Thuật toán phải được áp dụng cho mọi bài toán cùng dạng chứ không phải chỉ cho một tập đặc biệt các giá trị đầu vào.

Có nhiều cách mô tả thuật toán như: Dùng ngôn ngữ tự nhiên; dùng lưu đồ (sơ đồ khối); dùng một ngôn ngữ lập trình nào đó (trong giáo trình này dùng loại ngôn ngữ mô tả gần như ngôn ngữ lập trình Pascal và gọi là phỏng Pascal); …

2. Mô tả thuật toán bằng lưu đồ

Sau khi có thuật toán để giải bài toán, trước khi chuyển sang ngôn ngữ lập trình, người ta thường phải thể hiện thuật toán dưới dạng sơ đồ. Sơ đồ đó gọi là lưu đồ của thuật toán. Các ký hiệu quy ước dùng trong lưu đồ được trình bày trong bảng 1.

Bảng 1. Các ký hiệu quy ước dung trong lưu đồ thuật toán

Tên ký hiệu	Ký hiệu	Vai trò của ký hiệu
Khối mở đầu hoặc kết thúc		Dùng để mở đầu hoặc kết thúc thuật toán
Khối vào ra		Đưa dữ liệu vào và in kết quả
Khối tính toán		Biểu diễn các công thức tính toán và thay đổi giá trị các đối tượng
Khối điều kiện		Kiểm tra các điều kiện phân nhánh
Chương trình con		Gọi các chương trình con
Hướng đi của thuật toán		Hướng chuyển thông tin, liên hệ giữa các khối

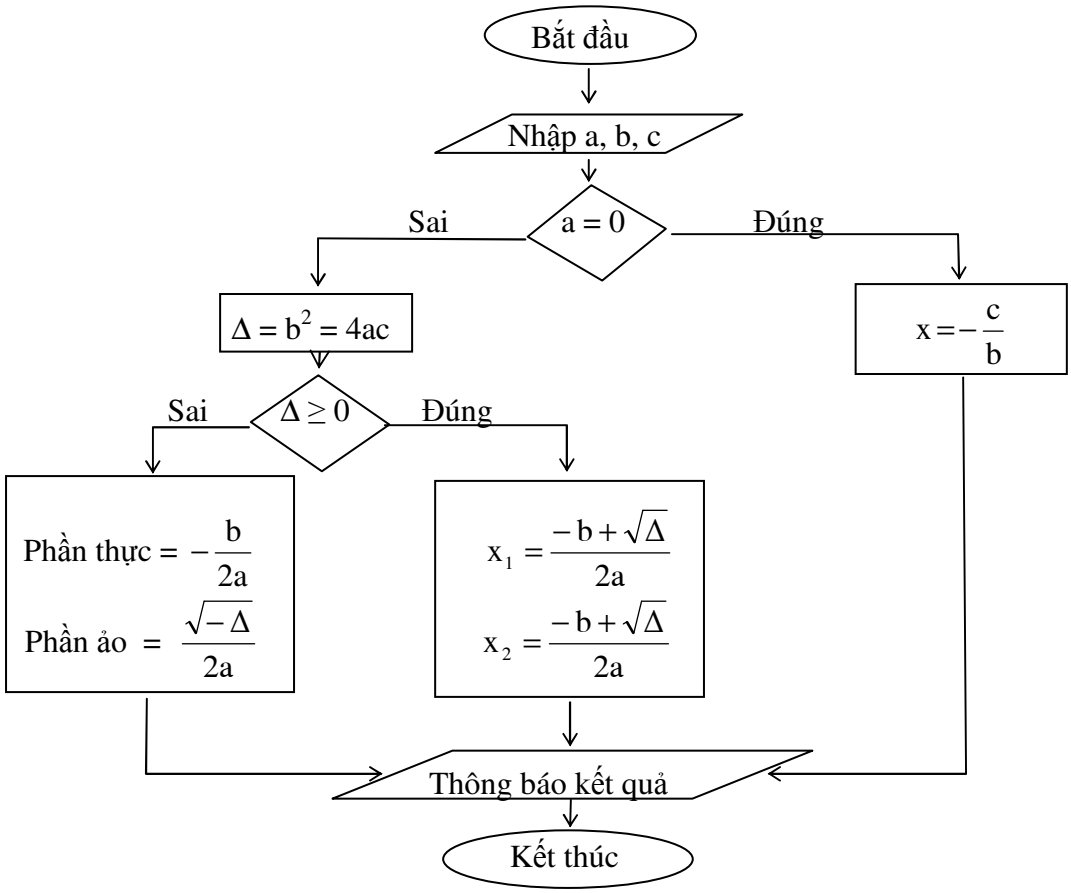
Thí dụ: Thuật toán giải phương trình bậc hai $ax^2 + bx + c = 0$ gồm các bước sau:

1) Xác định các hệ số a, b, c (thông tin đầu vào)
2) Kiểm tra hệ số a:

- Nếu a = 0: Phương trình đã cho là phương trình bậc nhất, nghiệm là: $x = -\frac{c}{b}$.
- Nếu $a \neq 0$: Chuyển sang bước 3.

3) Tính biệt thức $\Delta = b^2 - 4ac$.
4) Kiểm tra dấu của biệt thức Δ
   - Nếu $\Delta \geq 0$: Phương trình có nghiệm thực
   - Nếu $\Delta < 0$: Phương trình có nghiệm phức
5) In kết quả

Lưu đồ của thuật toán được trình bày trong hình 1



Hình 1. Lưu đồ giải phương trình bậc hai

3. Mô tả thuật toán bằng ngôn ngữ phỏng Pascal

Để giải bài toán trên máy tính điện tử phải viết chương trình theo một ngôn ngữ lập trình nào đó (Pascal, C, Basic, ...). Mỗi ngôn ngữ lập trình có một quy tắc cấu trúc riêng. Để thay việc mô tả thuật toán bằng lời, có thể mô tả thuật toán bằng các cấu trúc lệnh tương tự như ngôn ngữ lập trình Pascal và gọi là ngôn ngữ phỏng Pascal.

Các câu lệnh chính dùng để mô tả thuật toán gồm có: Procedure hoặc Function; câu lệnh gán; các câu lệnh điều kiện; các vòng lặp. Ngoài ra khi cần giải thích các câu lệnh bằng lời, có thể để các lời giải thích trong dấu (* ... *) hoặc {…}.

Nghĩa là ngôn ngữ phỏng Pascal hoàn toàn tương tự ngôn ngữ lập trình Pascal, nhưng không có phần khai báo. Tuy nhiên, phải nêu rõ đầu vào (Input) và đầu ra (output) của thuật toán.

3.1. Câu lệnh Procedure (thủ tục) hoặc Function (hàm)

Đứng ngay sau câu lệnh này là tên của thủ tục hoặc tên hàm. Các bước thực hiện của thuật toán được mô tả trong thủ tục (hàm) được bắt đầu bởi từ khóa begin và kết thúc bởi từ khóa end.

Thí dụ 1

```
Function Max(a, b, c)  (* Hàm tìm số lớn nhất trong 3 số a, b, c *)
Begin
 (* thân hàm*)
End;
```

Thí dụ 2

```
Procedure Giai_phuong_trình_bac_hai (* Thủ tục giải phương trình bậc hai *)
 Begin
 (* thân thủ tục *)
 End;
```

Chú ý rằng, khi mô tả thuật toán bằng function, trước khi kết thúc (end) thuật toán phải trả về (ghi nhận) giá trị của function đó.

3.2. Câu lệnh gán

Dùng để gán giá trị cho các biến. Cách viết:

Tên biến := giá trị gán

Thí dụ:

```
x := a;       (*biến x được gán giá trị a*)
max := b;   (*biến max được gán giá trị b*)
```

3.3. Khối câu lệnh tuần tự

Được mở đầu bằng từ khóa **begin** và kết thúc bằng **end** như sau:

```
begin
   Câu lệnh 1;
   Câu lệnh 2;
   ... .....
   Câu lệnh n;
end;
```

Các lệnh được thực hiện tuần tự từ câu lệnh thứ nhất đến câu lệnh cuối cùng.

3.4. Câu lệnh điều kiện

Có hai dạng: dạng đơn giản và dạng lựa chọn.

a. Dạng đơn giản: Cách viết:

if <điều kiện> **then** câu lệnh hoặc khối câu lệnh;

Khi thực hiện, điều kiện được kiểm tra, nếu điều kiện thỏa mãn thì câu lệnh (khối câu lệnh) được thực hiện, nếu điều kiện không thỏa mãn thì lệnh bị bỏ qua.

b. Dạng lựa chọn: Cách viết:

if <điều kiện> **then** câu lệnh hoặc khối câu lệnh 1 **else** câu lệnh hoặc khối câu lệnh 2;

Khi thực hiện, điều kiện được kiểm tra, nếu điều kiện thỏa mãn thì câu lệnh (khối câu lệnh) 1 được thực hiện, nếu điều kiện không được thỏa mãn thì câu lệnh (khối câu lệnh) 2 được thực hiện.

Thí dụ 1. Thuật toán tìm số lớn nhất trong 3 số thực a, b, c.

- Đầu tiên cho max = a;
- So sánh max với b, nếu b > max thì max = b;
- So sánh max với c, nếu c > max thì max = c.

```
Function  max(a,b,c)
 Input:    3 số thực a,b,c;
 Output:  Số lớn nhất trong 3 số đã nhập;
   Begin
      x := a;
      if  b > x  then  x:= b;
      if  c > x  then  x:= c;
      max := x;
    End;
```

Thí dụ 2. Thuật toán giải phương trình bậc hai $ax^2 + bx + c = 0$

```
Procedure  Giai_phuong_trinh_bac2;
Input:    Các hệ số a, b, c;
Output:  Nghiệm của phương trình;
          begin
             if  a = 0  then  x := -c/b;
             if  a ≠ 0 then
                begin
                   Δ := b^2 – 4ac;
                   if  Δ ≥ 0  then
                                begin
                                   x1 = (– b – √Δ )/2a ;
                                   x2 = (– b + √Δ )/2a;
                                end
                              else
                                begin
                                   Phần thực := -b/2a;
                                   Phần ảo := ( √(– Δ) )/2a;
                                 end;
                 end;
          end;
```

3.5. Các câu lệnh lặp

Có hai loại: Loại có bước lặp xác định và loại có bước lặp không xác định.

a. Loại có bước lặp xác định: Cách viết như sau:

for biến điều khiển := giá trị đầu **to** giá trị cuối **do** câu lệnh hoặc khối câu lệnh;

Khi thực hiện, biến điều khiển được kiểm tra, nếu biến điều khiển nhỏ hơn hoặc bằng giá trị cuối thì câu lệnh (khối câu lệnh) được thực hiện. Tiếp đó biến điều khiển sẽ tăng thêm 1 đơn vị và quá trình được lặp lại cho đến khi biến điều khiển lớn hơn giá trị cuối thì vòng lặp dừng và cho kết quả. Như vậy hết vòng lặp for số bước lặp là giá trị cuối (của biến điều khiển) trừ giá trị đầu cộng một.

Thí dụ: Tìm giá trị lớn nhất của dãy số $a_1, a_2, \ldots, a_n$.

Thuật toán: Đầu tiên cho giá trị lớn nhất (max) bằng a_1, sau đó lần lượt so sánh max với các số a_i (i = 2, 3, …, n), nếu max < a_i thì max bằng a_i, nếu max > a_i thì max không đổi.

```
Function  max_day_so;
   Input:    Dãy số a1, a2, … ,an;
   Output:  Giá trị lớn nhất (max) của dãy số đã nhập;
   begin
      max := a1 ;
      for  i:= 2  to  n  do
           if ai > max  then  max := ai ;
        max_day_so := max;
   end;
```

Chú thích: Vòng lặp for còn cách viết lùi biến điều khiển như sau:

for biến điều khiển := giá trị cuối **downto** giá trị đầu **do** câu lệnh hoặc khối câu lệnh;

Việc thực hiện câu lệnh này tương tự như khi viết biến điều khiển tăng dần.

b. Loại có bước lặp không xác định: Có hai cách viết

Cách thứ nhất: ***while*** điều kiện ***do*** câu lệnh hoặc khối câu lệnh;

Khi thực hiện, điều kiện được kiểm tra, nếu điều kiện được thoả mãn thì câu lệnh (khối câu lệnh) được thực hiện. Nếu điều kiện không thoả mãn thì vòng lặp dừng và cho kết quả.

Thí dụ: Kiểm tra xem số nguyên dương m đã cho có phải là số nguyên tố không?

Thuật toán như sau: Số m là số nguyên tố nếu nó không chia hết cho bất cứ số nguyên dương khác 1 nào nhỏ hơn hoặc bằng $\sqrt{m}$.

Thật vậy, nếu m là một hợp số (không phải là số nguyên tố), nghĩa là tồn tại các số nguyên dương a, b sao cho:

$$m = a.b \quad \Rightarrow \quad a \le \sqrt{m} \text{ hoặc } b \le \sqrt{m}$$

Vậy, nếu m là số nguyên tố thì m không chia hết cho mọi số nguyên dương i, $2 \le i \le \sqrt{m}$

```
Procedure nguyento(m);
   Input:   Số nguyên dương m;
   Output:  True, nếu m là số nguyên tố;   False, nếu m không phải là số nguyên tố;
      begin
            i := 2;
            while  i  ≤ √m   do
             begin
                if  m mod i  = 0  then  nguyento := false
                                  else  nguyento := true;
              i := i+1;
             end;
        end;
```

Cách thứ hai: ***repeat*** câu lệnh hoặc khối câu lệnh ***until*** điều kiện;

Khi thực hiện, câu lệnh (khối câu lệnh) được thực hiện, sau đó điều kiện được kiểm tra, nếu điều kiện sai thì vòng lặp được thực hiện, nếu điều kiện đúng thì vòng lặp dừng và cho kết quả.

Thí dụ: Thuật toán Ơ-clit tìm ước số chung lớn nhất của hai số nguyên dương a, b như sau: Giả sử a > b và a chia cho b được thương là q và số dư là r, trong đó a, b, q, r là các số nguyên dương:

$$a = bq + r$$

suy ra: ƯCLN(a, b) = ƯCLN(b, r)

và số dư cuối cùng khác không là ước số chung lớn nhất của a và b.

Thật vậy: Giả sử d là ước số chung của hai số nguyên dương a và b, khi đó: r = a – bq chia hết cho d. Vậy d là ước chung của b và r.

Ngược lại, nếu d là ước số chung của b và r, khi đó do bq + r = a cũng chia hết cho d. Vậy d là ước số chung của a và b.

Chẳng hạn, muốn tìm ước số chung lớn nhất của 111 và 201 ta làm như sau:

$$201 = 1.\ 111 + 90$$
$$111 = 1.\ 90 + 21$$
$$90 = 4.\ 21 + 6$$
$$21 = 3.\ 6 + 3$$
$$6 = 2.\ 3$$

Vậy ƯCLN(111, 201) = 3 (3 là số dư cuối cùng khác 0).

```
Function   UCLN(a, b)
     Input:   a, b là 2 số nguyên dương;
     Output:  UCLN(a, b);
     begin
```

```
x := a;
   y : = b;
  repeat
   begin
     r := x mod y;  (* r là phần dư khi chia x cho y *)
     x : = y;    y := r ;
          if  y ≠ 0 then uc := y;
   end;
  until  y =  0;
 UCLN := uc;
end  ;
```

Chú ý: Khi giải các bài toán phức tạp thì thường phải phân chia bài toán đó thành các bài toán nhỏ hơn gọi là các bài toán con. Khi đó phải xây dựng các thủ tục hoặc hàm để giải các bài toán con đó, sau đó tập hợp các bài toán con này để giải bài toán ban đầu đã đặt ra. Thuật ngữ tin học gọi các chương trình giải bài toán con đó là các chương trình con.

Thí dụ: Tìm số nguyên tố nhỏ nhất lớn hơn số nguyên dương m đã cho.

```
Procedure  So_nguyen_to_lon_hon(m);
Input:     Số nguyên dương m;
Output:  n là số nguyên tố nhỏ nhất lớn hơn m;
begin
    n := m + 1;
    while  nguyento(n) = false  do  n := n + 1;
end;
```

4. Độ phức tạp của thuật toán

Có hai lý do làm cho một thuật toán đúng có thể không thực hiện được trên máy tính. Đó là do máy tính không đủ bộ nhớ để thực hiện hoặc thời gian tính toán quá dài. Tương ứng với hai lý do trên người ta đưa ra hai khái niệm:

- **Độ phức tạp không gian của thuật toán,** độ phức tạp này gắn liền với các cấu trúc dữ liệu được sử dụng. Vấn đề này không thuộc phạm vi của môn học này.
- **Độ phức tạp thời gian của thuật toán,** độ phức tạp này được thể hiện qua số lượng các câu lệnh về các phép gán, các phép tính số học, phép so sánh, … được sử dụng trong thuật toán khi các giá trị đầu vào có kích thước đã cho.

4.1. Khái niệm độ tăng của hàm

Trước hết xét thí dụ: Giả sử thời gian tính toán của một thuật toán phụ thuộc vào kích thước n của đầu vào theo công thức:

$$t(n) = 60n^2 + 9n + 9$$

Bảng sau cho thấy khi n lớn, t(n) xấp xỉ số hạng $60n^2$:

n	$t(n) = 60n^2 + 9n + 9$	$60n^2$
10	9099	6000
100	600 909	600 000
1 000	60 009 009	60 000 000
10 000	6 000 090 009	6 000 000 000

Định nghĩa: Cho f(x) và g(x) là hai hàm từ tập số nguyên hoặc tập số thực vào tập các số thực. Người ta nói f(x) là O(g(x)) hay f(x) có quan hệ big-O với g(x), ký hiệu f(x) = O(g(x)), nếu tồn tại hai hằng số C và k sao cho:

$$|f(x)| \le C.|g(x)|, \quad \forall x \ge k.$$

Thí dụ 1. $t(n) = 60n^2 + 9n + 9 = O(n^2)$

Thật vậy: $\forall n \ge 1$, ta có: $|60n^2 + 9n + 9| = 60n^2 + 9n + 9$

$$= n^2\left(60 + \frac{9}{n} + \frac{9}{n^2}\right)$$

$$\le n^2 (60 + 9 + 9) = 78n^2.$$

Vậy $C = 78$; $k = 1$.

Tương tự ta có thể chứng minh:

$$P_n(x) = a_0x^n + a_1x^{n-1} + ... + a_n = O(x^n), \ x \in R.$$

Thí dụ 2. $f(n) = 1 + 2 + 3 + ... + n < n + n + n + ... + n = n.n = n^2$.

Vậy $f(n) = O(n^2)$.

Thí dụ 3. Ta có: $n! < n^n$, suy ra: $n! = O(n^n)$; $(C = k = 1)$

Thí dụ 4. $\forall n$: $n! < n^n$, $\lg(n!) < n \lg n$,

suy ra $\lg(n!) = O(n\lg n)$; $(C = k = 1)$

Thí dụ 5. Ta có: $\lg n < n$, suy ra $\lg n = O(n)$; $(C = k = 1)$

Có thể hiểu đơn giản quan hệ f(x) = O(g(x)) là f(x) và g(x) là "cùng cấp", tuy nhiên g(x) là hàm đơn giản nhất có thể đại diện cho f(x) về độ lớn cũng như tốc độ biến thiên.

4.2. Độ tăng của tổ hợp các hàm

Định lý: Nếu $f_1(x) = O(g_1(x))$ và $f_2(x) = O(g_2(x))$

Thì: 1) $(f_1 + f_2)(x) = O(\max\{g_1(x), g_2(x)\})$.

2) $(f_1.f_2)(x) = O(g_1(x).g_2(x))$

Chứng minh: Theo giả thiết, ta có: $|f_1(x)| \le C_1|g_1(x))|$, $\forall x > k_1$

$|f_2(x)| \le C_2|g_2(x))|$, $\forall x > k_2$

Chọn $k = \max(k_1; k_2)$ thì cả hai bất đẳng thức đều thoả mãn. Do đó:

1) $|(f_1 + f_2)(x)| = |f_1(x) + f_2(x)| \le |f_1(x)| + |f_2(x)| \le$

$$\le C_1|g_1(x)| + C_2|g_2(x)| \le (C_1 + C_2)g(x)$$

ở đây $g(x) = \max\{|g_1(x)|, |g_2(x)|\}$.

2) $|(f_1.f_2)(x)| = |f_1(x)|.|f_2(x)| \le C_1C_2\,|g_1(x)|.|g_2(x)| = C_1C_2|g_1(x)g_2(x)|$

Hệ quả: Nếu $f_1(x) = O(g(x))$, $f_2(x) = O(g(x))$ thì $(f_1+f_2)(x) = O(g(x))$

Thí dụ. Cho đánh giá O của các hàm:

1/ $f(n) = 2n\lg(n!) + (n^3 + 3)\lg n$, $n \in N$.

2/ $f(x) = (x + 1)\lg(x^2 + 1) + 3x^2$, $x \in R$.

Giải: 1) Ta có: $\lg(n!) = O(n\lg n) \Rightarrow 2n\lg(n!) = O(n^2\lg n)$

$(n^3 + 3)\lg n = O(n^3\lg n)$

Vậy $f(n) = O(n^3\lg n)$

2) Ta có: $\lg(x^2 + 1) \le \lg 2x^2 = \lg 2 + 2\lg x \le 3\lg x$, $\forall x > 1$

$\Rightarrow \lg(x^2 + 1) = O(\lg x) \Rightarrow (x + 1)\lg(x^2 + 1) = O(x\lg x)$

Mặt khác: $3x^2 = O(x^2)$ và $\max\{x\lg x; x^2\} = x^2$.

Vậy $f(x) = O(x^2)$.

4.3. Độ phức tạp của thuật toán

Như đã nói ở phần đầu của mục 4, chúng ta chỉ đề cập đến độ phức tạp về thời gian của thuật toán. Độ phức tạp về thời gian của thuật toán được đánh giá qua số lượng các phép toán mà thuật toán sử dụng. Vì vậy phải đếm các phép toán có trong thuật toán. Các phép toán phải đếm là:

- Các phép so sánh các số nguyên hoặc số thực;
- Các phép tính số học: cộng, trừ, nhân, chia;
- Các phép gán;
- Và bất kỳ một phép tính sơ cấp nào khác xuất hiện trong quá trình tính toán.

Giả sử số các phép toán của thuật toán là f(n), trong đó n là kích thước đầu vào, khi đó người ta thường quy độ phức tạp về thời gian của thuật toán về các mức:

- Độ phức tạp O(1), gọi là độ phức tạp hằng số, nếu f(n) = O(1).
- Độ phức tạp $O(\log_a n)$, gọi là độ phức tạp logarit, nếu $f(n) = O(\log_a n)$. (Điều kiện $a > 1$)
- Độ phức tạp O(n), gọi là độ phức tạp tuyến tính, nếu f(n) = O(n).
- Độ phức tạp $O(n\log_a n)$, gọi là độ phức tạp nlogarit nếu $f(n) = O(\log_a n)$. (Điều kiện $a > 1$)
- Độ phức tạp $O(n^k)$, gọi là độ phức tạp đa thức, nếu $f(n) = O(n^k)$.
- Độ phức tạp $O(a^n)$, gọi là độ phức tạp mũ, nếu $f(n) = O(a^n)$. (Điều kiện $a > 1$)
- Độ phức tạp O(n!), gọi là độ phức tạp giai thừa, nếu f(n) = O(n!).

Thí dụ 1. Tìm độ phức tạp của thuật toán để giải bài toán: Tìm số lớn nhất trong dãy n số nguyên $a_1, a_2, \ldots, a_n$ đã cho:

```
Procedure  max(a1, a2, …, an);
Input:    Dãy số  a1, a2, ... , an;
Output:  Số lớn nhất (max) của dãy số đã cho;
 begin
  max := a1;
  for i := 2 to  n  do
```

```
        if  a_i > max   then  max := a_i;
    end;
```

Mỗi bước của vòng lặp for phải thực hiện nhiều nhất 3 phép toán: phép gán biến điều khiển i, phép so sánh a_i với max và có thể là phép gán a_i vào max; vòng lặp có (n – 1) bước (i = 2, 3, …, n) do đó nhiều nhất có cả thảy 3(n - 1) phép toán phải thực hiện. Ngoài ra thuật toán còn phải thực hiện phép gán đầu tiên max := a_1.

Vậy số phép toán nhiều nhất mà thuật toán phải thực hiện là:

$$3(n – 1) + 1 = 3n – 2 = O(n)$$

Độ phức tạp về thời gian của thuật toán là độ phức tạp tuyến tính.

Thí dụ 2. Độ phức tạp của thuật toán nhân ma trận.

```
Procedure   nhân_matran;
   Input:  2 ma trận  A = (a_ij)_m x p và  B = (b_ij)_p x n;
   Output:   ma trận tích  AB = (c_ij)_m x n ;
   Begin
   for  i:=1 to  m  do
     for  j:=1 to  n  do
        begin
        c_ij := 0;
        for  k:=1  to  p  do
           c_ij := c_ij + a_ik b_kj ;
        end;
   End.
```

Số phép cộng và số phép nhân trong thuật toán trên là: Với mỗi phần tử c_{ij} phải thực hiện p phép nhân và p phép cộng. Có tất cả m.n phần tử c_{ij}, vậy phải thực hiện 2mnp phép cộng và phép nhân.

Để xác định độ phức tạp của thuật toán, ta giả sử A, B là hai ma trận vuông cấp n, nghĩa là m = n = p. như vậy phải cần $2n^3$ phép cộng và phép nhân. Vậy độ phức tạp của thuật toán là $O(n^3)$ – độ phức tạp đa thức.

Một điều thú vị là, khi nhân từ 3 ma trận trở lên thì số phép tính cộng và nhân phụ thuộc vào thứ tự nhân các ma trận ấy. Chẳng hạn A, B, C là các ma trận có kích thước tương ứng là 30×20, 20×40, 40×10. Khi đó:

Nếu thực hiện theo thứ tự ABC =A(BC) thì tích BC là ma trận kích thước 20×10 và cần thực hiện 20.40.10 = 8000 phép tính cộng và nhân. Ma trận A(BC) có kích thước 30×10 và cần thực hiện 30.20.10 = 6000 phép cộng và nhân. Từ đó suy ra cần thực hiện 8000+6000 = 14000 phép tính cộng và nhân để hoàn thành tích ABC.

Tương tự, nếu thực hiện theo thứ tự ABC = (AB)C thì cần thực hiện 30.20.40 phép tính cộng và nhân để thực hiện tích AB và 30.40.10 phép cộng và nhân để thực hiên tích (AB)C. Do đó số các phép tính cộng và nhân phải thực hiện để hoàn thành tích ABC là 24000+12000 = 36000 phép tính.

Rõ ràng hai cách nhân cho kết quả về số lượng các phép tính phải thực hiện là khác nhau.

5. Thuật toán tìm kiếm

Bài toán tìm kiếm được phát biểu như sau: Tìm trong dãy số $a_1, a_2, \ldots, a_n$ một phần tử có giá trị bằng số a cho trước và ghi lại vị trí của phần tử tìm được.

Bài toán này có nhiều ứng dụng trong thực tế. Chẳng hạn việc tìm kiếm từ trong từ điển, việc kiểm tra lỗi chính tả của một đoạn văn bản, …

Có hai thuật toán cơ bản để giải bài toán này: Thuật toán tìm kiếm tuyến tính và thuật toán tìm kiếm nhị phân. Chúng ta lần lượt xét các thuật toán này.

5.1. Thuật toán tìm kiếm tuyến tính (còn gọi là thuật toán tìm kiếm tuần tự).

Đem so sánh a lần lượt với a_i (i = 1, 2, …, n) nếu gặp một giá trị a_i = a thì ghi lại vị trí của a_i, nếu không gặp giá trị a_i = a nào ($a_i \neq a. \forall i$) thì trong dãy không có số nào bằng a.

```
Procedure  Tim_tuyen_tinh_phan_tu_bang_a;
Input:     a và dãy số a1, a2, ... , an;
Output:  Vị trí phần tử của dãy có giá trị bằng a, hoặc là số 0 nếu không tìm
                thấy a trong dãy;
begin
   i := 1;
   while  (i ≤ n  and ai ≠ a)  do    i := i + 1;
      if  i ≤ n  then   vitri := i    else   vitri := 0;
end;
```

Như vậy nếu a được tìm thấy ở vị trí thứ i của dãy (a_i = a) thì câu lệnh i := i + 1 trong vòng lặp while được thực hiện i lần (i = 1, 2, …, n). Nếu a không được tìm thấy, câu lệnh phải thực hiện n lần. Vậy số phép toán trung bình mà thuật toán phải thực hiện là:

$$\frac{1+2+\ldots+n}{n} = \frac{n(n+1)}{2n} = \frac{n+1}{2} = O(n)$$

Vậy, độ phức tạp của thuật toán tìm kiếm tuyến tính là độ phức tạp tuyến tính.

5.2. Thuật toán tìm kiếm nhị phân.

Giả thiết rằng các phần tử của dãy được xếp theo thứ tự tăng dần. Khi đó so sánh a với số ở giữa dãy, nếu $a < a_m$ với $m = \left[\frac{1+n}{2}\right]$ (cần nhắc lại rằng phần nguyên của x: [x] là số nguyên nhỏ nhất có trong x) thì tìm a trong dãy $a_1, \ldots, a_m$, nếu $a > a_m$ thì tìm a trong dãy $a_{m+1}, \ldots, a_n$. Đối với mỗi dãy con (một nửa của dãy đã cho) được làm tương tự để chỉ phải tìm phần tử có giá trị bằng a ở một nửa dãy con đó. Quá trình tìm kiếm kết thúc khi tìm thấy vị trí của phần tử có giá trị bằng a hoặc khi dãy con chỉ còn 1 phần tử.

Chẳng hạn việc tìm số 8 trong dãy số 5, 6, 8, 9, 11, 12, 13, 15, 16, 17, 18, 19, 20, 22 được tiến hành như sau:

Dãy đã cho gồm 14 số hạng, chia dãy thành 2 dãy con:

5, 6, 8, 9, 11, 12, 13 và 15, 16, 17, 18, 19, 20, 22

Vì 8 < 13 nên tiếp theo chỉ cần tìm ở dãy đầu tiên. Tiếp tục chia đôi thành 2 dãy: 5, 6, 8, 9 và 11, 12, 13; vì 8 < 9 nên lại chỉ phải tìm ở dãy 5, 6, 8, 9. Lại chia đôi dãy này thành 2 dãy con 5, 6 và 8, 9 thấy ngay 8 thuộc về dãy con 8, 9 và quá trình tìm kiếm kết thúc, vị trí của số 8 trong dãy đã cho là thứ ba

```
Procedure  Tim_nhi_phan_phan_tu_bang_a;
Input:     a và dãy số a1, a2, ... , an đã xếp theo thứ tự tăng;
Output:  Vị trí phần tử của dãy có giá trị bằng a, hoặc là số 0 nếu không tìm thấy
                trong dãy;
Begin
 i := 1;  (* i là điểm mút trái của khoảng tìm kiếm*)
 j := n;  (* j là điểm mút phải của khoảng tìm kiếm*)
 while i < j do
   begin
      m := ⌈(1 + j)/2⌉;
     if a > am then i := m+1
          else j := m;
   end;
         if  a = ai  then   vitri := i    else   vitri := 0;
end;
```

Độ phức tạp của thuật toán tìm kiếm nhị phân được đánh giá như sau: Không giảm tổng quát có thể giả sử độ dài của dãy $a_1, a_2, \ldots, a_n$ là $n = 2^k$ với k là số nguyên dương. (Nếu n không phải là lũy thừa của 2, luôn tìm được số k sao cho $2^{k-1} < n < 2^k$ do đó có thể xem dãy đã cho là một phần của dãy có 2^k phần tử). Như vậy phải thực hiện nhiều nhất k lần chia đôi các dãy số (mỗi nửa dãy của lần chia đôi thứ nhất có 2^{k-1} phần tử, của lần chia đôi thứ hai có 2^{k-2} phần tử, …, và của lần chia đôi thứ k là $2^{k-k} = 2^0 = 1$ phần tử). Nói cách khác là nhiều nhất có k vòng lặp while được thực hiện trong thuật toán tìm kiếm nhị phân. Trong mỗi vòng lặp while phải thực hiện hai phép so sánh, và vòng lặp cuối cùng khi chỉ còn 1 phần tử phải thực hiện 1 phép so sánh để biết không còn 1 phần tử nào thêm nữa và 1 phép so sánh để biết a có phải là phần tử đó hay không. Từ đó thấy rằng thuật toán phải thực hiện nhiều nhất $2k + 2 = 2[\log_2 n] + 2 = O(\log n)$ phép so sánh.

Vậy, độ phức tạp của thuật toán tìm kiếm nhị phân là độ phức tạp logarit.

6. Thuật toán đệ quy

6.1. Công thức truy hồi

Đôi khi rất khó định nghĩa một đối tượng nào đó một cách tường minh, nhưng có thể định nghĩa đối tượng đó qua chính nó với đầu vào nhỏ hơn. Cách định nghĩa như vậy gọi là cách định nghĩa bằng truy hồi hoặc định nghĩa bằng đệ quy và nó cho một công thức gọi là công thức truy hồi.

Định nghĩa: Định nghĩa bằng truy hồi bao gồm các quy tắc để xác định các đối tượng, trong đó có một số quy tắc dùng để xác định các đối tượng ban đầu gọi là các điều kiện ban đầu; còn các quy tắc khác dùng để xác định các đối tượng tiếp theo gọi là công thức truy hồi.

Thí dụ 1. Dãy số a_n được định nghĩa bằng đệ quy như sau:

$$a_0 = 3;\ a_n = a_{n-1} + 3.$$

Trong đó $a_0 = 3$ là điều kiện ban đầu, còn $a_n = a_{n-1} + 3$ là công thức truy hồi.

Thí dụ 2. Định nghĩa bằng đệ quy giai thừa của số tự nhiên n là:

$$GT(0) = 1; \quad GT(n) = n.GT(n-1).$$

Vì GT(n) = n! = n(n-1)(n-2)…1 = n.GT(n-1). Trong đó GT(0) = 1 là điều kiện ban đầu, còn GT(n) = n.GT(n – 1) là công thức truy hồi.

Thí dụ 3. Dãy số $F_0, F_1, F_2, \ldots, F_n$ được định nghĩa:

$$F_0 = 0;\ F_1 = 1;\ F_n = F_{n-1} + F_{n-2}.$$

Đó chính là định nghĩa bằng đệ quy của dãy số có tên là dãy Fibonacci. Trong đó $F_0 = 0$, $F_1 = 1$ là các điều kiện ban đầu, còn $F_n = F_{n-1} + F_{n-2}$ là công thức đệ quy.

Dễ thấy một số số hạng đầu tiên của dãy là: 0; 1; 1; 2; 3; 5; 8; 13; 21; …

6.2. Thuật toán đệ quy.

Nhiều khi việc giải bài toán với đầu vào xác định có thể đưa về việc giải bài toán đó với giá trị đầu vào nhỏ hơn. Chẳng hạn:

$$n! = n\ .\ (n-1)! \quad \text{hay} \quad UCLN(a, b) = UCLN(a \bmod b, b),\ a > b$$

Định nghĩa: Một thuật toán gọi là đệ quy nếu thuật toán đó giải bài toán bằng cách rút gọn liên tiếp bài toán ban đầu tới bài toán cũng như vậy nhưng với dữ liệu đầu vào nhỏ hơn.

Dễ thấy cơ sở của thuật toán là công thức truy hồi.

Thí dụ 1. Tính giai thừa của số tự nhiên n bằng đệ quy.

```
Function   GT(n);
   Input:    Số tự nhiên n;
Output:   Giá trị của n!;
Begin
 if  n = 0  then  GT(0) := 1
          else   GT(n) := n*GT(n – 1);
End;
```

Thí dụ 2. Tính số hạng của dãy Fibonacci bằng đệ quy.

```
Function   Fibonacci(n);
Input:    Vị trí thứ n của dãy Fibonacci;
Output:  Giá trị  Fn của dãy Fibonaci;
Begin
```

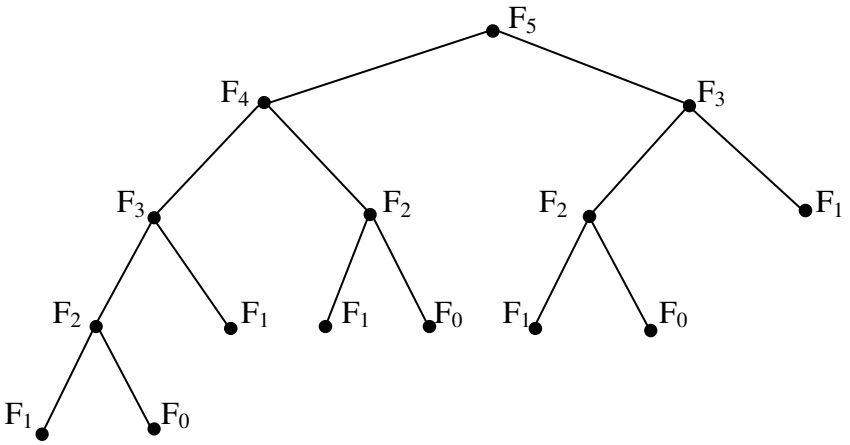
```
    if  n = 0  then  Fibonacci(0) := 0
          else  if  n = 1  then  Fibonacci(1) := 1
                else   Fibonacci(n) := Fibonacci(n-1) + Fibonacci(n-2);
End;
```

Thí dụ 3. Thuật toán đệ quy tìm UCLN(a, b).

```
Function  UCLN(a, b);
Input:    Hai số nguyên dương a và b;
Output: Ước số chung lớn nhất của a và b;
Begin
  if  b = 0  then  UCLN(a, b) := a
        else  if a > b  then  UCLN(a, b) := UCLN(a mod b, b)
                              else  UCLN(a, b) := UCLN(b mod a, a);
End;
```

Bây giờ chúng ta thử tìm độ phức tạp về thời gian của một vài thuật toán viết bằng đệ quy. Chẳng hạn xét thuật toán đệ quy tính số hạng của dãy Fibonacci, để tính F_n ta biểu diễn $F_n = F_{n-1} + F_{n-2}$, sau đó thay thế cả hai số này bằng tổng của hai số Fibonacci bậc thấp hơn. Quá trình tiếp tục như vậy cho đến khi F_0 và F_1 xuất hiện thì được thay bằng các giá trị của chúng trong định nghĩa.

Mỗi bước đệ quy cho tới khi F_0 và F_1 xuất hiện, các số Fibonacci được tính hai lần. Chẳng hạn giản đồ cây ở hình 2 cho ta hình dung cách tính F_5 theo thuật toán đệ quy. Từ đó có thể thấy rằng để tính F_n cần thực hiện $F_{n+1} - 1$ phép cộng.



Hình 2. Lược đồ tính F_5 bằng đệ quy

Độ phức tạp của thuật toán đệ quy tìm ước số chung lớn nhất của hai số nguyên dương a, b (thí dụ 3): UCLN(a,b) = UCLN(a mod b,b), nếu $a \geq b$ (a mod b là phần dư khi chia a cho b) được đánh giá bằng cách ứng dụng dãy Fibonacci.

Trước hết bằng quy nạp toán học chúng ta chứng minh số hạng tổng quát của dãy Fibonacci thỏa mãn:

$$F_n > \alpha^{n-2}, \quad \forall n \geq 3, \text{ trong đó } \quad \alpha = \frac{1+\sqrt{5}}{2} \ . \qquad (1)$$

Thật vậy:

Ta có: $\alpha < 2 = F_3$, nghĩa là (1) đúng với n = 3.

Giả sử $F_n > \alpha^{n-2}$ đúng với n, xét với n+1. Dễ thấy α là một nghiệm của phương trình $x^2 - x - 1 = 0$ nên suy ra $\alpha^2 = \alpha + 1$. Từ đó:

$$\alpha^{n-1} = \alpha^2\alpha^{n-3} = (\alpha + 1)\alpha^{n-3} = \alpha^{n-2} + \alpha^{n-3} \ .$$

Theo giả thiết quy nạp, nếu $n \geq 4$, ta có $F_{n-1} > \alpha^{n-3}$ và $F_n > \alpha^{n-2}$. Thay vào định nghĩa của dãy Fibonacci:

$$F_{n+1} = F_n + F_{n-1} > \alpha^{n-2} + \alpha^{n-3} = \alpha^{n-1}$$

Vậy $F_n > \alpha^{n-2}$, $\forall n \geq 3$.

Công thức (1) được chứng minh.

Trở lại thuật toán đệ quy tìm ước số chung lớn nhất của hai số nguyên dương a, b ($a \geq b$). Độ phức tạp của thuật toán được đánh giá qua số lượng các phép chia dùng trong thuật toán này.

Đặt $r_0 = a$, $r_1 = b$, ta có:

$$\begin{aligned}
&r_0 = r_1q_1 + r_2; && 0 \leq r_2 < r_1,\\
&r_1 = r_2q_2 + r_3; && 0 \leq r_3 < r_2,\\
&\ldots\ldots\\
&r_{n-2} = r_{n-1}q_{n-1} + r_n; && 0 \leq r_n < r_{n-1},\\
&r_{n-1} = r_nq_n ;
\end{aligned}$$

Như vậy phải dùng n phép chia để tìm r_n = UCLN(a,b). Các thương $q_1, q_2, \ldots, q_{n-1}$ luôn lớn hơn hoặc bằng 1, còn $q_n \geq 2$. Từ đó suy ra:

$$\begin{aligned}
&r_n \geq 1 = F_2,\\
&r_{n-1} \geq 2r_n = 2F_2 = F_3,\\
&r_{n-2} \geq r_{n-1} + r_n \geq F_3 + F_2 = F_4,\\
&\ldots\\
&r_2 \geq r_3 + r_4 \geq F_{n-1} + F_{n-2} = F_n,\\
&b = r_1 \geq r_2 + r_3 \geq F_n + F_{n-1} = F_{n+1}.
\end{aligned}$$

trong đó F_n là số hạng thứ n trong dãy Fibonacci.

Vậy nếu n là số các phép chia trong thuật toán Ơ-clit tìm ước số chung lớn nhất của hai số nguyên dương a, b thì $b \geq F_{n+1}$, trong đó F_n là số Fibonacci thứ n.

Do $F_{n+1} > \alpha^{n-1}$ với $n > 2$ và $\alpha = \frac{1+\sqrt{5}}{2}$ nên $b > \alpha^{n-1}$.

Từ đó: $\lg b > (n-1)\lg\alpha > \frac{n-1}{5}$ (vì $\lg\alpha \approx 0{,}208 > \frac{1}{5}$).

Vậy: $n - 1 < 5\lg b \Rightarrow n < 5\lg b + 1 = O(\lg b)$.

Nghĩa là độ phức tạp của thuật toán Ơ-clit viết theo đệ quy là O(lgb).

Qua đây có thể thấy trong nhiều trường hợp, việc đánh giá độ phức tạp của một thuật toán quả là không dễ dàng.

6.3. Đệ quy và lặp

Hầu như các bài toán giải được bằng lặp thì cũng giải được bằng đệ quy. Thông thường, nếu dùng phương pháp lặp thì số phép tính sẽ ít hơn; chúng ta minh họa điều này bằng thủ tục tính số Fibonacci bằng thuật toán đệ quy và thuật toán lặp.

Trong thí dụ 2, mục 6.2. chúng ta đã tính số phép cộng theo thuật toán đệ quy để tính số Fibonacci thứ n là $F_{n+1} - 1$. Bây giờ ta xét thủ tục lặp để tính số Fibonacci:

```
Function   Lap_Fibonacci(n);
 Input:    Vị trí thứ  n + 1 của dãy Fibonacci;
 Output:  Giá trị  f_n của dãy;
 Begin
     if  n = 0  then  y := 0
     else
     begin
      x := 0;  y:=1
      for i := 1  to n – 1
      begin
          z := x + y;
          x :=y;   y := z
      end;
     end;
    Lap_Fibonacci := z;
 End.
```

Thuật toán này khởi tạo x như là $F_0 = 0$ và y như là $F_1 = 1$. Qua mỗi bước lặp tổng của x và y được gán cho biến phụ z. Sau đó x được gán giá trị của y và y được gán giá trị của z. Vậy qua vòng lặp thứ nhất, ta có $x = F_1$ và $y = F_2$. Khi qua vòng lặp thứ n – 1 thì $x = F_{n-1}$. Vậy chỉ có n – 1 phép cộng được thực hiện để tính F_n. Rõ ràng số lượng các phép tính này nhỏ hơn khi tính bằng đệ quy.

Tuy số lượng các phép tính khi dùng đệ quy nhiều hơn khi dùng lặp, nhưng nhiều khi người ta vẫn thích sử dụng đệ quy hơn. Có lẽ lý do là ở chỗ chương trình đệ quy thường gọn hơn, mặt khác có những bài toán chỉ giải được bằng đệ quy mà không giải được bằng lặp.

7. Một vài thuật toán về số nguyên

7.1. Biểu diễn các số nguyên

Định lý: Cho b là một số nguyên dương lớn hơn 1, khi đó nếu n là số nguyên dương tùy ý thì n có thể biểu diễn duy nhất dưới dạng:

$$n = a_{k-1} b^{k-1} + a_{k-2} b^{k-2} + \ldots + a_1 b + a_0. \qquad (1)$$

trong đó k là số nguyên không âm, a_0, a_1, …, a_{k-1} là các số nguyên không âm và nhỏ hơn b đồng thời $a_{k-1} \neq 0$.

Chúng ta không chứng minh định lý này. Biểu diễn của số n theo công thức (1) gọi là **khai triển cơ số b của n** và được ký hiệu là $(a_{k-1}\, a_{k-2} \ldots a_0)_b$. Đặc biệt, nếu là cơ số 10 (hệ thập phân, b = 10), người ta quy ước không cần viết cơ số kèm theo. Ngoài hệ thập phân còn có hệ nhị phân (cơ số 2), hệ bát phân (cơ số 8) và hệ thập lục phân (cơ số 16) là hay được dùng trong tin học.

Thí dụ 1: Xác định khai triển thập phân của số nguyên từ các hệ cơ số khác:

$(245)_8 = 2.8^2 + 4.8 + 5 = 165$

$(10101111)_2 = 1.2^7 + 0.2^6 + 1.2^5 + 0.2^4 + 1.2^3 + 1.2^2 + 1.2 + 1 = 175$

Hệ thập lục phân có 16 chữ số 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F, trong đó A, B, C, D, E, F tương ứng với các số 10, 11, 12, 13, 14, 15 trong hệ thập phân. chẳng hạn:

$(2AE0B)_{16} = 2.16^4 + 10.16^3 + 14.16^2 + 0.16 + 11 = 175\,627$

Bây giờ xét bài toán ngược, xác định khai triển theo cơ số b của số tự nhiên n cho trong hệ thập phân. Trước hết chia n cho b được số dư là a_0:

$$n = bq_0 + a_0, \quad 0 \leq a_0 \leq b \quad (a_0 = n \bmod b)$$

khi đó a_0 là chính là chữ số cuối cùng trong hệ cơ số b. Tiếp theo chia q_0 cho b được số dư a_1 chính là số đứng trước a_0 trong hệ đếm cơ số b:

$$q_0 = bq_1 + a_1, \quad 0 \leq a_1 \leq b \quad (a_1 = q_0 \bmod b)$$

Quá trình tiếp tục cho đến khi nhận được thương bằng 0.

Thí dụ 2: Tìm khai triển cơ số 8 của 12345. Ta có:

$12345 = 8.1543 + 1$

$1543 = 8.192 + 7$

$192 = 8.24 + 0$

$24 = 8.3 + 0$

$3 = 8.0 + 3$

Vậy: $12345 = (30071)_8$.

Thuật toán tìm khai triển cơ số b của số tự nhiên n như sau:

```
Procedure Khai_trien_co_so_b;
Input:    Số tự nhiên n và cơ số b;
Output: Khai triển n theo cơ số b (a_k a_{k-1} … a_0)_b;
Begin
q := n;
k := 0;
while q ≠ 0 do
begin
    a_k := q mod b;
```

$q := \left[\frac{q}{b}\right]$;

k := k + 1;

end;

End;

7.2. Cộng và nhân trong hệ nhị phân

Các thuật toán thực hiện các phép tính với các số nguyên ở hệ nhị phân có vai trò quan trọng trong tin học. Bởi vậy phần này mô tả hai thuật toán thực hiện phép cộng và phép nhân – hai phép toán số học cơ bản nhất – các số nguyên trong hệ nhị phân.

Giả sử có hai số nguyên a và b dưới dạng nhị phân:

$$a = (a_{n-1}\, a_{n-2} \ldots a_1\, a_0)_2 \quad \text{và} \quad b = (b_{n-1}\, b_{n-2} \ldots b_1\, b_0)_2 .$$

(Nếu cần thiết có thể đặt thêm các bít 0 lên phần đầu của các khai triển nhị phân của a hoặc b)

a. Phép cộng

Để cộng a và b, trước hết cộng hai bit cuối:

$$a_0 + b_0 = c_0.2 + s_0.$$

ở đây s_0 là bit cuối trong khai triển nhị phân của tổng a + b và c_0 là số nhớ. Sau đó là cộng hai bit tiếp theo và số nhớ:

$$a_1 + b_1 + c_0 = c_1.2 + s_1.$$

ở đây s_1 là bit tiếp theo trong khai triển nhị phân của tổng a + b và c_1 là số nhớ. Tiếp tục như vậy cho đến:

$$a_{n-1} + b_{n-1} + c_{n-2} = c_{n-1}.2 + s_{n-1} .$$

Đặt $s_n = c_{n-1}$ ta được khai triển nhị phân của tổng a + b:

$$a + b = (s_n\, s_{n-1} \ldots s_1\, s_0)_2.$$

Thuật toán được mô tả bằng ngôn ngữ phỏng Pascal như sau:

Procedure Cong_hai_so_nguyen_duong_dang_nhi_phan;
Input: Hai số nguyên dương a và b dưới dạng nhị phân
$(a_{n-1} \ldots a_1\, a_0)_2$ và $(b_{n-1} \ldots b_1\, b_0)_2$;
Output: Tổng a + b dưới dạng nhị phân $(s_n\, s_{n-1} \ldots s_1\, s_0)_2$;
Begin
c := 0
for j:=1 to n do
begin

$d := \left[\frac{a_j + b_j + c}{2}\right]$;

```
        sj := aj + bj + c – 2d;
        c := d;
      end;
  sn := c;
End;
```

Có thể hiểu là phép cộng trong hệ nhị phân cũng được tiến hành như ở hệ thập phân, nhưng lưu ý rằng:

$$0 + 0 = 0; \quad 0 + 1 = 1 + 0 = 1; \quad 1 + 1 = 10$$

Chẳng hạn tính $(1110)_2 + (10111)_2$ là:

$$\begin{array}{rr} & 0\ 1110 \\ + & 1\ 0111 \\ \hline & 10\ 0101 \end{array}$$

Độ phức tạp của thuật toán được đánh giá như sau: Mỗi bit trong khai triển nhị phân của tổng a + b được tính bằng cách cộng một cặp bit và có thể phải cộng thêm bit nhớ. Như vậy để có mỗi bit của tổng cần nhiều nhất 3 phép cộng. Mỗi khai triển nhị phân có n bit. Vậy phải thực hiện tối đa 3n phép cộng hai số nguyên n bit, nghĩa là thuật toán toán có độ phức tạp tuyến tính O(n).

b. Phép nhân

Theo luật phân phối, ta có:

$$a.b = a\sum_{j=0}^{n-1} b_j 2^j = \sum_{j=0}^{n-1} (ab_j)2^j$$

Dễ thấy:

$$ab_j = \begin{cases} a & , \text{ nếu } b_j = 1 \\ 0 & , \text{ nếu } b_j = 0 \end{cases}$$

và mỗi lần nhân một khai triển nhị phân với 2 là dịch chuyển khai triển đó một vị trí về bên trái bằng cách thêm một số 0 vào cuối khai triển đó. Do đó có thể nhận được tích $(ab_j)2^j$ bằng cách dịch khai triển nhị phân của ab_j về bên trái j vị trí, nói cách khác là thêm j bit 0 vào cuối khai triển nhị phân ab_j. Cuối cùng nhận được tích ab bằng cách cộng n số nguyên dương ab_j2^j với j = 0, 1, …, n – 1.

Thí dụ: Tìm tích của $a = (110)_2$ và $b = (101)_2$.

$ab_0.2^0 = (110)_2.1.2^0 = (110)_2$

$ab_1.2^1 = (110)_2.0.2^1 = (0000)_2$

$ab_2.2^2 = (110)_2.1.2^2 = (11000)_2$

và, cuối cùng ta cộng $(110)_2$, $(0000)_2$ và $(11000)_2$ với nhau được $ab = (11110)_2$.

$$\begin{array}{rr} & 110 \\ \times & 101 \\ \hline & 110 \\ + & 000 \\ & 110 \\ \hline & 11110 \end{array}$$

Dễ thấy cách thực hiện phép nhân trong hệ nhị phân cũng giống như trong hệ thập phân.

Thuật toán được mô tả bằng ngôn ngữ phỏng Pascal như sau:

Procedure Nhan_hai_so_nguyen_dang_nhi_phan;
Input: Hai số nguyên dương a và b dưới dạng nhị phân
$(a_{n-1} \ldots a_1\ a_0)_2$ và $(b_{n-1} \ldots b_1\ b_0)_2$;
Output: Tích ab dưới dạng nhị phân;
Begin
for j := 0 to n do
if $b_j = 1$ then $c_j := a$ được dịch sang trái j vị trí
else $c_j := 0$;
(* $c_0, c_1, \ldots, c_{n-1}$ là các tích riêng phần *)
p := 0;
for j := 0 to n – 1 do p := p + c_j ; (* p là giá trị của tích ab *)
End;

Độ phức tạp của thuật toán nhân 2 số nguyên dạng nhị phân bằng cách tính số các phép cộng bit và dịch bít của thuật toán được đánh giá như sau:

Các tích riêng $c_j = ab_j2^j$ không cần dịch chuyển khi $b_j = 0$, vì khi đó $c_j = 0$ và nó phải dịch chuyển j vị trí khi $b_j = 1$. Vì thế với tất cả n tích riêng c_j cần thực hiện tối đa:

$$0 + 1 + 2 + \ldots + (n-1) = \frac{n(n-1)}{2}$$

phép dịch chỗ. Vậy số phép dịch chỗ tối đa là $O(n^2)$.

Sau khi dich chỗ số nguyên $c_{n-1} = ab_{n-1}2^{n-1}$ có 2 n bit, theo thuật toán cộng hai số nguyên sẽ có tối đa O(n) số phép cộng bit để cộng tất cả n các tích riêng c_j.

Vậy độ phức tạp của thuật toán nhân 2 số nguyên dạng nhị phân là $O(n^2)$.

BÀI TẬP CHƯƠNG 1

1.1. Chứng minh rằng:

a) $\frac{x^2+1}{x+1} = O(x)$; b) $2^n + 17 = O(3^n)$.

1.2. Tìm một số nguyên n nhỏ nhất sao cho $f(x) = O(x^n)$, nếu:

a) $f(x) = 2x^3 + x^2\lg x$; b) $f(x) = 3x^5 + (\lg x)^4$;

c) $f(x) = \frac{x^4 + x^2 + 1}{x^3 + 1}$; d) $f(x) = \frac{x^3 + 5\lg x}{x^4 + 1}$.

1.3. Chứng minh rằng $x^2 + 4x + 17$ là $O(x^3)$, nhưng x^3 không là $O(x^2 + 4x + 17)$.

1.4. Hãy cho một đánh giá big-O tốt nhất có thể được đối với các hàm sau:

a) $(n^2 + 8)(n + 1)$; b) $(n\lg n + n^2)(n^3 + 2)$;

c) $(n! + 2^n)(n^3 + \lg(n^2 + 1))$; d) $(2^n + n^2)(n^3 + 3^n)$;

e) $(x^2 + x(\lg x)^3)(2^x + x^3)$.

1.5. Lập và mô tả thuật toán bằng ngôn ngữ phỏng Pascal cho các bài toán sau:

a) Tính x^n với x là số thực và n là số nguyên. (Gợi ý: trước hết lập thủ tục tính x^n với n là số nguyên dương, sau đó mở rộng thủ tục cho n là số nguyên âm $x^{-n} = \frac{1}{x^n}$, $n \in N^+$)

b) Chỉ dùng các lệnh gán để biến bộ ba số (x, y, z) thành (y, z, x). Số tối thiểu các lệnh gán là bao nhiêu?

c) Chèn 1 số nguyên x vào vị trí thích hợp trong dãy các số nguyên $a_1, a_2, ..., a_n$ đã được xếp theo thứ tự tăng dần.

d) Tìm số nhỏ nhất và số lớn nhất trong dãy gồm n số nguyên đã cho.

1.6. Xác định số các phép nhân được dùng để tính x^{2^k} bắt đầu với x rồi liên tiếp bình phương (để tìm x^2, x^4, …). Cách này có hiệu quả hơn cách nhân x với chính nó một số lần thích hợp không?

1.7. Lập và mô tả thuật toán bằng ngôn ngữ phỏng Pascal cho bài toán sau: Đếm các bit 1 trong một xâu bit bằng cách kiểm tra từng bit của xâu xem có phải là bit 1 không. Cho đánh giá big-O đối với các phép so sánh được dùng trong thuật toán.

1.8. Thuật toán tính giá trị của đa thức $P(x) = a_n x^n + a_{n-1} x^{n-1} + ... + a_1 x + a_0$ tại x = c như sau:

```
function  Dathuc(c, a0, a1, ..., an);
  Đầu vào:  c, a0, a1, ..., an ;
  Đầu ra:  y = P(c);
  begin
     power := 1;
     y : = a0 ;
```

```
    for   i := 1 to   n    do
      begin
         power := power *c ;
         y := y + a_i*power ;
      end;
    Dathuc := y;
end;
```

a) Tìm P(2) nếu $P(x) = 3x^2 + x + 1$ bằng cách thực hiện từng bước thuật toán trên.

b) Có bao nhiêu phép nhân và phép cộng dùng để tăng biến trong vòng lặp?

1.9. Thuật toán Horner để tìm giá trị P(c) của đa thức $P(x) = a_n x^n + a_{n-1}x^{n-1} + \ldots + a_1 x + a_0$ như sau:

```
function   Horner(c, a_0, a_1, ..., a_n);
Đầu vào:  c, a_0, a_1, ..., a_n ;
Đầu ra:  y = P(c);
begin
    y := a_n ;
    for   i := 1 to  n  do   y := y*c + a_{n-i} ;
    Horner :=  y;
end;
```

Các câu hỏi như trong bài tập 1.8.

1.10. Lập thuật toán tìm số hạng đầu tiên trong một dãy các số nguyên cho trước sao cho số hạng tìm thấy bằng một số hạng nào đó đứng trước nó. Tìm độ phức tạp về thời gian của thuật toán đã lập.

1.11. Lập thuật toán tìm trong dãy số nguyên dương cho trước số hạng đầu tiên nhỏ hơn số hạng đứng ngay trước nó. Xác định độ phức tạp của thuật toán đã lập.

1.12. *Ma trận lôgic*

Ma trận mà các phần tử là 0 hoặc 1 được gọi là ma trận lôgic.

1) Cho hai ma trận lôgic kích thước m×n: $A = [a_{ij}]_{m,n}$ và $B = [b_{ij}]_{m,n}$, ta có:

- $A \vee B$ là ma trận lôgic hợp giữa A và B có các phần tử ở hàng i, cột j là:

$$a_{ij} \vee b_{ij} = \begin{cases} 1 & \text{nếu } a_{ij} = 1 \text{ hoặc } b_{ij} = 1 \\ 0 & \text{nếu } a_{ij} = b_{ij} = 0 \end{cases}$$

- $A \wedge B$ là ma trận lôgic giao giữa A và B có các phần tử ở hàng i cột j là

$$a_{ij} \wedge b_{ij} = \begin{cases} 1 & \text{nếu } a_{ij} = b_{ij} = 1 \\ 0 & \text{nếu } a_{ij} = 0 \text{ hoặc } b_{ij} = 0 \end{cases}$$

2) Cho hai ma trận lôgic $A = [a_{ij}]_{m,p}$ kích thước m×p và $B = [b_{ij}]_{p,n}$ kích thước p×n. Tích boole của A với B, ký hiệu $A \otimes B$, là ma trận lôgic kích thước m×n mà phần tử c_{ij} (phần tử ở hàng i, cột j) được xác định như sau:

$$c_{ij} = (a_{i1} \wedge b_{1j}) \vee (a_{i2} \wedge b_{2j}) \vee \ldots \vee (a_{ip} \wedge b_{pj})$$

a) Viết thuật toán dạng phỏng Pascal cho các phép toán của các ma trận lôgic như các định nghĩa nêu ở trên.

b) Có bao nhiêu phép toán bit được dùng để tính A⊗B với A, B là các ma trận lôgic vuông cấp n.

c) Chứng minh rằng: A⊗(B⊗C) = (A⊗B)⊗C với giả thiết các phép toán trong biểu thức đã cho là thực hiện được,

1.13. Tìm và mô tả thuật toán đệ quy cho các bài toán sau:

a) Tìm tổng của n số nguyên dương lẻ đầu tiên.

b) Tìm số cực tiểu của tập hữu hạn các số nguyên.

1.14. Hãy đưa ra thuật toán đệ quy tìm n! **mod** m, tromg đó m, n, là các số nguyên dương.

1.15. Tìm thuật toán đệ quy tính a^{2^n}, trong đó a là một số thực và n là một số nguyên dương (Gợi ý: dùng đẳng thức $a^{2^{n+1}} = \left(a^{2^n}\right)^2$).

1.16. Tìm thuật toán đệ qui tìm ước số chung lớn nhất của 2 số nguyên không âm a, b (a < b) biết rằng UCLN(a, b) = UCLN(a, b – a).

ĐÁP SỐ

1.2. a) 3; b) 5; c) 1; d) 0

1.4. a) $O(n^3)$; b) $O(n^5)$; c) $O(n^3n!)$; d) $O(6^n)$; e) $O(x^2 2^x)$.

1.5.

c) Procedure insert (x, a_1, a_2, …,a_n : integer);
{dãy xếp theo thứ tự tăng: $a_1 \le a_2 \le \ldots \le a_n$}
$a_n := x + 1$;
i := 1;
while $x > a_i$ do
i := i + 1;
for j := 0 to n-i do
$a_{n-j+1} := a_{n-j}$;
$a_j := x$;
{x đã được chèn vào vị trí đúng}

1.7. Procedure ones (a: xâu bit, $a = a_1a_2 \ldots a_n$);
begin
ones := 0;
for i:= 1 to n do
if $a_i = 1$ then ones := ones + 1;
end; {ones là số các bít 1 trong xâu a}

1.10. Procedure find duplicate (a_1a_2 ... a_n : integer);

location := 0;

i := 2;

while i ≤ n and location = 0 do

begin

j := 1;

while j < i and location := 0 do

if $a_i = a_j$ then location := i else j := j +1

i := i +1;

end; {location là chỉ số của giá trị đầu tiên lặp lại giá trị trước trong dãy}

$O(n^2)$

1.11. Procedure find decrease (a_1 a_2 ... a_n nguyên dương);

location := 0;

i := 2;

while i ≤ n and location = 0 do

if $a_i < a_{i-1}$ then location := i else i : = i + 1;

end; {location là chỉ số của giá trị đầu tiên nhỏ hơn giá trị ngay trước nó}

Độ phức tạp của thuật toán là O(n).

1.13.

a) Procedure sum of odds (n : nguyên dương);

if n = 1 then sum of odds := 1

else sum of odds : = sum of odds(n – 1) + 2n – 1;

b) Procedure smalless (a_1, a_2, …, a_n : nguyên dương);

if n = 1 then smalless (a_1, a_2, …, a_n) := a_1

else smalless (a_1, a_2, …, a_n) := min(smalless (a_1, a_2, …, a_{n-1}), a_n);

1.14. Procedure modfactorial (n, m : nguyên dương);

if n = 0 then modfactorial (n, m) : = 1

else modfactorial (n, m) := (n * modfactorial (n-1, m)) **mod** m;

CÂU HỎI ÔN TẬP CHƯƠNG 1

1. Phát biểu định nghĩa và các đặc trưng của thuật toán. Các phương pháp mô tả thuật toán.

2. Trình bày quan hệ big-O của các hàm số và khái niệm độ phức tạp về thời gian của thuật toán. Đánh giá độ phức tạp của thuật toán nhân ma trận

3. Mô tả các thuật toán tìm kiếm và đánh giá độ phức tạp của chúng.

4. Định nghĩa bằng đệ quy là gì? Thế nào là công thức truy hồi? Trình bày thuật toán đệ quy. Mối liên hệ giữa đệ quy và lặp.

5. Mô tả các thuật toán lặp và đệ quy để: Tìm tổng của một dãy số; Tìm ước số chung nhỏ nhất của hai số nguyên dương; Tìm số Fibonacci

6. Nêu thuật toán chuyển đổi cơ số cơ số của số tự nhiên. Cách cộng và nhân các số nguyên trong hệ nhị phân?

CHƯƠNG 2

BÀI TOÁN ĐẾM

Lý thuyết tổ hợp là một phần rất quan trọng của toán rời rạc, nó chuyên nghiên cứu sự sắp xếp các đối tượng (phần tử của tập hợp) theo một quy tắc nào đó. Mỗi kết quả của một cách sắp xếp các phần tử của một tập hợp theo một quy tắc xác định đã cho được gọi là một cấu hình tổ hợp. Đếm các phần tử của một tập hợp, hoặc các cấu hình tổ hợp là nội dung của bài toán đếm. Chúng ta cũng cần có một quan niệm rộng rãi về khái niệm đếm, chẳng hạn nếu nói có 100đ thì đó là một cách đếm, nhưng có thể thay cách nói đó bằng cách nói có 10 tờ 5đ và 25 tờ 2đ. Nói cách khác, liệt kê các cấu hình tổ hợp là một cách đếm.

Các tập hợp nói đến trong chương này là các tập hợp có hữu hạn các phần tử. Số lượng các phần tử của tập hợp A, ký hiệu là N(A), được gọi là lực lượng (hay bản số) của tập hợp A.

1. Nguyên lý cộng và nguyên lý nhân.

1.1. Nguyên lý cộng

Nếu A và B là 2 tập hợp rời nhau ($A \cap B = \varnothing$) thì $N(A \cup B) = N(A) + N(B)$.

Nguyên lý này có thể mở rộng như sau: Giả sử $A_1, A_2, \ldots, A_n$ là các tập con của một tập X nào đó thoả mãn:

- Các A_i đôi một rời nhau ($A_i \cap A_j = \varnothing$, khi $i \neq j$)
- $X = A_1 \cup A_2 \cup \ldots \cup A_n$.

(Các tập $A_1, A_2, \ldots, A_n$ thoả mãn 2 tính chất trên được gọi là một phân hoạch của tập X)

Khi đó: $N(X) = N(A_1) + N(A_2) + \ldots + N(A_n)$

Đặc biệt, ta có $N(A) = N(X) - N(\overline{A})$ trong đó $\overline{A}$ là tập bù của tập A trong X.

Thí dụ 1: Một khoá học có 3 danh sách lựa chọn các bài thực hành: Danh sách thứ nhất có 10 bài; danh sách thứ hai có 15 bài; danh sách thứ ba có 25 bài. Mỗi sinh viên được chọn 1 bài trong 3 danh sách trên để làm thực hành. Hỏi mỗi sinh viên có bao nhiêu cách lựa chọn bài thực hành.

Giải: Có 10 cách lựa chọn trong danh sách thứ nhất, 15 cách lựa chọn trong danh sách thứ hai và 25 cách lựa chọn trong danh sách thứ ba. Vậy có 10 + 15 + 25 = 50 cách lựa chọn cho mỗi sinh viên.

Thí dụ 2: Một đoàn vận động viên của một địa phương được cử đi thi đấu ở 2 môn: bắn súng và bắn cung. Trong đoàn có 10 nam và số vận động viên bắn súng là 14 (kể cả nam và nữ). Số nữ vận động viên bắn cung bằng số nam vận động viên bắn súng. Hỏi đoàn có bao nhiêu người.

Giải: Gọi A, B tương ứng là các tập vận động viên nam, vận động viên nữ, ta có $N(A) = 10$

A_1, A_2 tương ứng là các tập vận động viên nam bắn súng, vận động viên nam bắn cung, ta có $N(A_1) + N(A_2) = 10$

B_1, B_2 tương ứng là các tập vận động viên nữ bắn súng, vận động viên nữ bắn cung, ta có $N(B_2) = N(A_1)$ và $N(B_1) + N(A_1) = 14$

Từ đó $N(B_1) + N(B_2) = 14$, nghĩa là đoàn có 14 vận động viên nữ.

Vậy toàn đoàn có 10 + 14 = 24 vận động viên.

Thí du 3: Xét đoạn chương trình phỏng Pascal sau

```
k := 0;
for  i1:= 1 to n1 do  k := k + 1;
for  i2:= 1 to n2 do  k := k + 1;
for  i3:= 1 to n3 do  k := k + 1;
```

Hỏi sau khi thực hiện xong đoạn chương trình trên, giá trị của k bằng bao nhiêu?

Giải: Giá trị ban đầu gán cho k là 0. Khối lệnh gồm 3 vòng lặp tuần tự, sau mỗi bước lặp của từng vòng lặp giá trị của k được tăng thêm 1, vòng lặp thứ i (i = 1, 2, 3) có n_i bước nên sau vòng lặp thứ i giá trị của k được tăng thêm n_i. Do các vòng lặp là tuần tự nên sau cả 3 vòng lặp thì giá trị của k là: $k = n_1 + n_2 + n_3$.

1.2. Nguyên lý nhân

Nếu mỗi thành phần thứ i (tức là a_i) của bộ có thứ tự gồm k thành phần ($a_1, a_2, \ldots, a_k$) có n_i cách chọn (i = 1, 2, …, k) thì số lượng các bộ k thành phần đó sẽ là $n_1 n_2 \ldots n_k$ (tích của số các cách chọn của mỗi thành phần).

Có thể phát biểu nguyên lý nhân dưới dạng số phần tử của tích Đề-các của các tập hợp:

$$N(A_1 \times A_2 \times \ldots \times A_k) = N(A_1).\ N(A_2)\ \ldots\ N(A_k)$$

Thí dụ 1: Có 4 phương tiện đi từ Hà nội vào thành phố Hồ Chí Minh là: Ô tô, Tàu hoả, Tàu thuỷ và Máy bay. Có 2 phương tiện đi từ thành phố Hồ chí Minh ra Côn đảo là Máy bay và Tàu thuỷ. Hỏi có mấy cách đi từ Hà nội đến Côn đảo nếu bắt buộc phải đi qua thành phố Hồ Chí Minh?

Giải: Mỗi cách đi từ Hà nội vào thành phố Hồ Chí Minh có 2 cách đi ra Côn đảo. Vậy với 4 cách đi từ Hà nội vào thành phố Hồ Chí Minh có 4 . 2 = 8 cách đi từ Hà nội qua thành phố Hồ Chí Minh rồi đến Côn đảo.

Thí dụ 2: Một xâu nhị phân là một dãy liên tiếp các chữ số 0 hoặc 1 (gọi là bit 0 và bit 1). Độ dài của xâu là số các chữ số 0, 1 có mặt trong xâu. Hai xâu cùng độ dài được gọi là khác nhau nếu có ít nhất một vị trí tại đó các bit là khác nhau. Hỏi có bao nhiêu xâu nhị phân có độ dài 8 ?

Giải: Tại mỗi một trong 8 vị trí có 2 cách lựa chọn là bit 0 hoặc bit 1. Do đó theo quy tắc nhân cho thấy có $2^8 = 256$ xâu nhị phân khác nhau có độ dài 8.

Thí dụ 3: Một biển số xe máy của một địa phương được cấu tạo gồm hai nhóm. Nhóm đầu gồm 2 ký tự: đầu tiên là một chữ cái sau đó là một chữ số. Nhóm thứ 2 gồm một dãy 4 chữ số liên tiếp (kể cả các số trùng nhau). Hỏi có thể phát hành bao nhiêu biển số xe tại địa phương đó.

Giải: Có tất cả 24 cách chọn cho vị trí chữ cái đầu tiên. Năm vị trí tiếp theo, mỗi vị trí có 10 cách chọn. Theo quy tắc nhân, số biển số đăng ký xe máy có thể là:

$$24.10.10.10.10.10 = 24.10^5 = 2\ 400\ 000$$

Thí dụ 4: Giá trị của biến k bằng bao nhiêu sau khi đoạn chương trình sau được thực hiện?

```
k := 0;
for i1 := 1 to n1 do
    for i2 := 1 to n2 do
        for i3:= 1 to n3 do
            k := k + 1;
```

Giải: Đầu tiên k được gán giá trị bằng 0. Có 3 vòng lặp lồng nhau. Sau mỗi lần lặp của vòng for trong cùng, giá trị của k tăng thêm 1. Với mỗi giá trị của i_1, vòng for thứ hai thực hiện n_2 lần và với mỗi giá trị của i_2 vòng for thứ 3 thực hiện n_3 lần. Vậy theo nguyên lý nhân, sau khi cả 3 vòng lặp kết thúc thì vòng lặp trong cùng (i_3) thực hiện $n_1.n_2.n_3$ lần, nghĩa là $k = n_1.n_2.n_3$.

Chú ý rằng nhiều bài toán đếm phải giải bằng cách kết hợp cả nguyên lý cộng và nguyên lý nhân.

Thí dụ: Tên biến trong ngôn ngữ lập trình Pascal là một xâu gồm chữ cái tiếng Anh (không phân biệt chữ cái thường và chữ cái hoa) và chữ số, trong đó không được bắt đầu bằng chữ số. Hỏi có thể đặt được bao nhiêu tên biến khác nhau có độ dài không quá 2 ký tự, biết rằng có 10 từ khoá có độ dài 2 (chẳng hạn như to, do, go, if, or, in, …)

Giải: Gọi A là tập tên biến có 1 ký tự, thì N(A) = 26 (có 26 tên biến ứng với 26 chữ cái tiến Anh)

Gọi B là tập tên biến có 2 ký tự. Ký tự đầu có 26 cách lựa chọn, còn ký tự thứ hai có 36 cách lựa chọn (26 chữ cái và 10 chữ số). Vậy N(B) = 26.36 = 936. Do có 10 xâu bị "cấm" nên tổng số tên biến có độ dài không quá 2 ký tự là:

$$26 + 936 - 10 = 952$$

2. Chỉnh hợp. Hoán vị. Tổ hợp.

2.1. Chỉnh hợp

Định nghĩa: Một chỉnh hợp chập k của n phần tử là một bộ có thứ tự gồm k phần tử lấy từ n phần tử đã cho ($k \le n$).

Số các chỉnh hợp chập k của n phần tử, ký hiệu là A_n^k , được tính theo công thức:

$$A_n^k = n(n-1)(n-2)\ldots(n-k+1) = \frac{n!}{(n-k)!}$$

Thật vậy, vì có n phần tử đã cho nên có n cách chọn phần tử đứng đầu, tiếp theo có n – 1 cách chọn phần tử đứng thứ hai, n – 2 phần tử đứng thứ ba, ..., n – k + 1 cách chọn phần tử thứ k. Theo nguyên lý nhân được công thức cần chứng minh.

Thí dụ 1. Có 10 vận động viên thi chạy. Hỏi có bao nhiêu cách dự đoán các vận động viên về nhất, nhì, ba? Biết rằng các vận động viên đều có cùng khả năng như nhau.

Giải: Số cách dự đoán là số cách chọn có thứ tự 3 trong 10 vận động viên, tức là:

$$A_{10}^3 = 10.\ 9.\ 8 = 720 \text{ cách dự đoán.}$$

Thí dụ 2. Có thể lập được $9A_9^3 = 9.9.8.7 = 4536$ số nguyên có 4 chữ số khác nhau.

Thí dụ 3. Có bao nhiêu đơn ánh từ tập hợp A có k phần tử vào tập hợp B có n phần tử ($k \le n$)?

Giải : Giả sử các phần tử của tập hợp A tương ứng với các số 1, 2, ..., k; khi đó mỗi đơn ánh chính là một bộ có thứ tự các ảnh của các phần tử của tập hợp A. Vậy có A_n^k các đơn ánh từ A vào B.

2.2. Hoán vị:

Định nghĩa: Một hoán vị của n phần tử đã cho là một cách sắp xếp có thứ tự của n phần tử đó.

Dễ nhận ra rằng, một hoán vị của n phần tử chính là một chỉnh hợp chập n của n. Do đó:

Số các hoán vị của n phần tử , ký hiệu P_n , là:

$$P_n = A_n^n = n(n-1)\ldots 1 = n!$$

Thí dụ 1. Mỗi cách xếp hàng (ngang hoặc dọc) của 5 người là một hoán vị của 5 phần tử. Vậy có $P_5 = 5! = 120$ cách xếp hàng của 5 người.

Thí dụ 2. Một người phải đi kiểm tra 8 địa điểm khác nhau theo bất kỳ thứ tự nào. Hỏi người đó có bao nhiêu cách đi nếu bắt đầu từ một trong 8 địa điểm đó sao cho mỗi địa điểm đều được kiểm tra đúng một lần.

Giải: Người đó có $P_7 = 7! = 5040$ cách đi.

2.3. Tổ hợp

Định nghĩa: Một tổ hợp chập k của n phần tử là một cách chọn không kể thứ tự k phần tử từ n phần tử đã cho ($n \geq k$).

Số các tổ hợp chập k của n phần tử , ký hiệu là C_n^k , được tính như sau:

Với mỗi bộ k phần tử không sắp thứ tự tương ứng với k! cách sắp xếp có thứ tự. Nói cách khác mỗi tổ hợp chập k của n tương ứng với k! chỉnh hợp chập k của n. Từ đó suy ra:

$$A_n^k = k!\, C_n^k \quad \Rightarrow \quad C_n^k = \frac{A_n^k}{k!} = \frac{n!}{k!(n-k)!}$$

Thí dụ 1. Có 6 đội bóng thi đấu vòng tròn một lượt. Hỏi phải tổ chức bao nhiêu trận đấu?

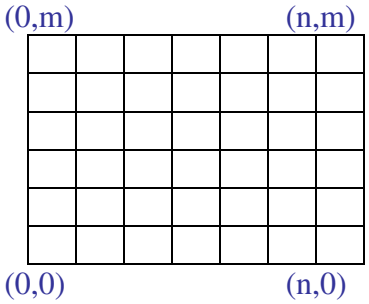
Giải: Cứ 2 đội gặp nhau một trận suy ra số trận đấu là $C_6^2 = 15$ trận

Thí dụ 2. Tính số giao điểm của các đường chéo nằm phía trong một đa giác lồi n cạnh ($n \geq 4$). Biết rằng không có 3 đường chéo nào đồng quy tại 1 điểm ở phía trong đa giác đó.

Giải: Cứ 4 đỉnh của đa giác cho ta một giao điểm thoả mãn bài toán. Vậy số điểm cần đếm là:

$$C_n^4 = \frac{n(n-1)(n-2)(n-3)}{24}$$

Thí dụ 3. Cho một lưới có m.n ô chữ nhật. Các nút được đánh số từ 0 đến n theo chiều từ trái sang phải và từ 0 đến m theo chiều từ dưới lên trên (xem hình vẽ). Hỏi có bao nhiêu đường đi khác nhau từ nút (0,0) đến nút (n,m), biết rằng mọi bước đi là sự dịch chuyển hoặc sang phải hoặc lên trên của một ô hình chữ nhật (không dịch chuyển sang trái hoặc xuống dưới)



Giải: Một đường đi như vậy được xem là gồm n+m bước, mỗi bước chỉ được chọn một trong hai giá trị là 0 nếu đi lên và 1 nếu đi sang phải. Như vậy mỗi đường đi tương ứng với một dãy nhị phân có độ dài m+n trong đó có đúng m giá trị 1 (và tất nhiên có đúng n giá trị 0).

Bài toán dẫn đến việc tìm xem có bao nhiêu dãy nhị phân có độ dài m+n trong đó có đúng m thành phần 1. Nói cách khác là số cách chọn m vị trí trong m+n vị trí để xếp số 1. Vậy số đường đi sẽ là C_{m+n}^{m} (hoặc C_{m+n}^{n})

Thí dụ 4. Xét đoạn chương trình phỏng Pascal sau:

```
for i := 1 to n – 1 do
    for j := i + 1 to n do
    if ai > aj then
        begin    {đổi chỗ ai, aj }
            t : = ai ;
            ai := aj ;
            aj := t ;
        end;
```

(Đây là một trong các thuật toán để xếp dãy số $a_1, a_2, ..., a_n$ theo thứ tự tăng dần)

Hãy đếm số phép so sánh các a_i được thực hiện trong đoạn chương trình trên.

Giải: Tại mỗi vòng lặp i = k thì vòng lặp j phải thực hiện n – k + 1 bước. Nghĩa là với mỗi i = k phải thực hiện n – k +1 phép so sánh (k = 2, 3, ..., n). Từ đó theo nguyên lý cộng, số các phép so sánh là:

$$(n-1)+(n-2)+\ldots+1=\frac{n(n-1)}{2}$$

Cũng có thể giải như sau: Vì mọi cặp a_i, a_j ($i \neq j$) đều phải so sánh với nhau, do đó tổng các phép so sánh là C_n^2.

Một vài tính chất

Từ công thức tính C_n^k, dễ dàng chứng minh được:

1) $C_n^0 = C_n^n = 1$;
2) $C_n^k = C_n^{n-k}$;
3) $C_n^k = C_{n-1}^k + C_{n-1}^{k-1}$;

Tính chất 3 chính là công thức truy hồi để tính C_n^k. Ta có:

n\ k	0	1	2	3	4	5
0	1					
1	1	1				
2	1	2	1			
3	1	3	3	1		
4	1	4	6	4	1	
5	1	5	10	10	5	1
… …						

Hay:

C_0^0

$C_1^0 \quad C_1^1$

$C_2^0 \quad C_2^1 \quad C_2^2$

. . .

$C_n^0 \quad C_n^1 \quad C_n^2 \quad \ldots \quad C_n^n$

Mỗi số hạng của hàng dưới, trừ số hạng đầu và số hạng cuối bằng 1, bằng tổng 2 số hạng ở hàng trên cùng cột và cột trước của số hạng cần tính, người ta thường gọi chúng là các tam giác Pascal.

Bằng lý luận tổ hợp có thể chứng minh được:

$$(x+1)^n = C_n^0 x^0 + C_n^1 x + C_n^2 x^2 + \ldots + C_n^n x^n .$$

Thật vậy, hệ số của x^k ở vế phải của công thức có được bằng cách nhân k số hạng x trong k nhân tử (x + 1) với n – k số 1 trong n – k nhân tử (x + 1) còn lại ở vế trái. Điều đó tương ứng với số cách chọn k nhân tử (x + 1) trong n nhân tử ở vế trái, số cách chọn đó chính là C_n^k. Vậy công thức được chứng minh.

Từ công thức trên, ta có công thức nhị thức Niu-tơn:

$$(x+y)^n = y^n\left(\frac{x}{y}+1\right)^n = y^n \sum_{k=0}^{n} C_n^k \left(\frac{x}{y}\right)^k = \sum_{k=0}^{n} C_n^k y^n \frac{x^k}{y^k} = \sum_{k=0}^{n} C_n^k x^k y^{n-k}$$

Chú ý 1. Nếu chọn x = y = 1, ta có:

$$C_n^0 + C_n^1 + \ldots + C_n^n = 2^2 .$$

Tuy nhiên, C_n^k chính là số các tập con có k phần tử của tập hợp có n phần tử . Vậy, tổng số tập con (kể cả tập rỗng và chính tập hợp đó) của một tập hợp có n phần tử là 2^n.

Chú ý 2. Nếu chọn x = 1, y = – 1, ta có:

$$C_n^0 - C_n^1 + C_n^2 + \ldots + (-1)^n C_n^n = 0 .$$

2.4. Chỉnh hợp và Tổ hợp suy rộng

a. Chỉnh hợp lặp

Định nghĩa: Một bộ có thứ tự gồm k phần tử lấy từ n phần tử đã cho, trong đó mỗi phần tử có thể được lấy nhiều lần được gọi là một chỉnh hợp lặp chập k từ n phần tử đã cho.

Chú ý rằng do mỗi phần tử có thể được lấy nhiều lần nên rất có thể k > n.

Số các chỉnh hợp lặp chập k của n được ký kiệu là $\overline{A}_n^k$ và được tính theo công thức:

$$\overline{A}_n^k = n^k$$

Thật vậy, vì có n cách chọn cho mỗi phần tử thứ i (i = 1, 2, ..., k); nên theo quy tắc nhân được công thức cần chứng minh.

Thí dụ 1. Tính số các dãy nhị phân có độ dài n.

Mỗi dãy nhị phân có độ dài n là một bộ có thứ tự gồm n thành phần được chọn trong tập {0, 1}. Do đó số dãy nhị phân có độ dài n là 2^n.

Thí dụ 2. Có thể lập được $9.\overline{A}_{10}^{3} = 9.10^3 = 9000$ số nguyên dương có 4 chữ số.

Thí dụ 3. Có bao nhiêu ánh xạ từ tập hợp A có k phần tử vào tập hợp B có n phần tử ?

Lý luận tương tự như thí dụ tính số đơn ánh trong mục chỉnh hợp; do mỗi phần tử của tập hợp B có thể là ảnh của nhiều phần tử thuộc A nên số ánh xạ có thể có là n^k.

Thí dụ 4. Trong ngôn ngữ Pascal chuẩn quy định tên biến không quá 8 ký tự (mỗi ký tự là một trong 26 chữ cái tiếng Anh hoặc một trong 10 chữ số) và phải bắt đầu bằng một chữ cái, ký tự là chữ không phân biệt chữ hoa và chữ thường. Hỏi có thể định nghĩa được bao nhiêu biến khác nhau?

Giải: Tất cả có 8 loại biến: 1 ký tự, 2 ký tự, ..., 8 ký tự. Theo quy tắc nhân thì số biến có k ký tự là 26.36^{k-1}, (k = 1, 2, ..., 8). Từ đó áp dụng quy tắc cộng được số các biến khác nhau trong ngôn ngữ Pascal là:

$$26(1 + 36 + 36^2 + ... + 36^7) = 2\ 095\ 681\ 645\ 538 \approx 2.10^{12}.$$

Đây là một con số rất lớn, không thể nào duyệt hết được chúng. Hiện tượng các số tổ hợp tăng như vậy gọi là hiện tượng bùng nổ tổ hợp.

Số ký tự	1	2	3	4	5	6	7
Số biến	26	962	34 658	1 247 714	44 917 730	1 617 038 306	58 213 379 042

b. Tổ hợp lặp

Định nghĩa: Một tổ hợp lặp chập k của n phần tử là một cách chọn không kể thứ tự k phần tử từ n phần tử đã cho, trong đó mỗi phần tử có thể được chọn nhiều lần.

Chú ý rằng do mỗi phần tử có thể được lấy nhiều lần nên rất có thể k > n.

Có rất nhiều bài toán đếm phải sử dụng khái niệm tổ hợp lặp. Chẳng hạn, trong hộp có 3 loại bi: bi đỏ, bi xanh và bi trắng; mỗi loại có ít nhất 4 viên. Hỏi có bao nhiêu cách lấy ra 4 viên bi từ hộp đó, nếu không phân biệt thứ tự lấy và các viên bi lấy ra có thể cùng 1 loại.

Giải: Có thể liệt kê các cách lấy như sau:

4 bi đỏ
4 bi xanh
4 bi trắng
3 bi đỏ, 1 bi xanh
3 bi đỏ, 1 bi xanh
3 bi xanh, 1 bi đỏ
3 bi xanh, 1 bi trắng
3 bi trắng, 1 bi đỏ
3 bi trắng, 1 bi xanh
2 bi đỏ, 2 bi xanh
2 bi đỏ, 2 bi trắng
2 bi xanh, 2 bi trắng
2 bi đỏ, 1 bi xanh, 1 bi trắng
2 bi xanh, 1 bi đỏ, 1 bi trắng
2 bi trắng, 1 bi đỏ, 1 bi xanh

Có tất cả 15 cách lấy 4 bi từ hộp đã cho. Mỗi cách lấy như vậy là một tổ hợp lặp chập 4 từ 3 phần tử đã cho.

Chúng ta không thể tính số tổ hợp lặp chập k từ n phần tử một cách thủ công như vậy, bởi không phải lúc nào cũng liệt kê hết các tổ hợp đó được.

Giả sử một hộp có 3 ngăn để đựng 3 loại bi khác nhau:

Việc chọn 4 bi tương ứng với việc đặt 4 bi vào trong 3 ngăn đã cho. Loại trừ 2 vách ngăn ngoài cùng là cố định, còn 2 vách ngăn phía trong và 4 viên bi có thể đổi chỗ cho nhau. Như vậy số cách chọn 4 bi tương ứng với số cách chọn 4 trong 6 vị trí để đặt 4 bi. Nói cách khác số cách chọn 4 bi là $C_6^4 = 15$

Khái quát có thể lập luận như sau: Mỗi tổ hợp lặp chập k của n phần tử có thể xem như cách xếp k dấu * vào một hộp có n ngăn, trong đó các vách ngăn hộp và các dấu * có thể dịch chuyển được trừ hai vách ngăn ở hai đầu hộp là cố định. Như vậy mỗi tổ hợp lặp chập k của n phần tử chính là số cách chọn k vị trí trong n + k – 1 (gồm k vị trí đặt dấu * và n – 1 vách ngăn) để đặt k dấu *. Vậy, ký hiệu số các tổ hợp lặp chập k của n phần tử là $\overline{C}_n^k$, ta có:

$$\overline{C}_n^k = C_{n+k-1}^k$$

Thí dụ 1. Một cửa hàng có 4 loại bánh hộp khác nhau. Hỏi có bao nhiêu cách chọn 6 hộp bánh?

Giải: Theo khái niệm tổ hợp lặp, ta có $\overline{C}_4^6 = C_{4+6-1}^6 = C_9^6 = 84$ cách chọn.

Thí dụ 2. Có bao nhiêu cách chọn 5 tờ tiền từ 7 loại tiền giấy 1000đ, 2000đ, 5000đ, 10 000đ, 20 000đ, 50 000đ và 100 000đ?

Giải: Số cách chọn là $\overline{C}_7^5 = C_{11}^5 = 462$.

Thí dụ 3. Phương trình x + y + z = 11 có bao nhiêu nghiệm nguyên không âm?

Giải: Mỗi nghiệm của phương trình ứng với một cách phân chia 11 phần tử (11 số 1) đã cho thành ba loại, sao cho có x phần tử loại 1, y phần tử loại 2 và z phần tử loại 3. Vì vậy số nghiệm của phương trình bằng số tổ hợp lặp chập 11 từ tập có 3 loại phần tử , nghĩa là có:

$$\overline{C}_3^{11} = C_{3+11-1}^{11} = 78 \text{ nghiệm.}$$

Có thể tìm được nghiệm của phương trình đã cho với các điều kiện ràng buộc của các ẩn số. Chẳng hạn tìm số nghiệm nguyên không âm của phương trình đã cho ở trên thoả mãn điều kiện: $x \geq 1$; $y \geq 2$; $z \geq 3$. Khi đó một nghiệm của phương trình ứng với một cách phân chia 11 phần tử đã cho thành 3 loại, sao cho có x phần tử loại 1, y phần tử loại 2 và z phần tử loại 3 trong đó có ít nhất 1 phần tử loại 1 (x đã có sẵn một số 1), có ít nhất 2 phần tử loại 2 và có ít nhất 3 phần tử loại 3. Vì thế trước hết chọn 1 phần tử loại 1, 2 phần tử loại 2 và 3 phần tử loại 3; còn lại 5 phần tử được chia tiếp cho ba loại. Do đó số nghiệm của phương trình là số tổ hợp lặp chập 5 từ 3 loại phần tử đã cho:

$$\overline{C}_3^5 = C_7^5 = 21 \text{ (nghiệm)}$$

c. Hoán vị của các tập hợp có các phần tử giống nhau

Trong các bài toán đếm, một số phần tử của tập hợp đã cho có thể giống nhau. Khi đó phải cẩn thận kẻo ta đếm chúng nhiều hơn một lần.

Thí dụ: Có thể lập được bao nhiêu xâu khác nhau bằng cách sắp xếp lại các chữ cái trong cụm từ NHANDANANHHUNG (nhân dân anh hùng)?

Giải: Trong từ đã cho có 14 chữ cái gồm 5 chữ N, 3 chữ H, 3 chữ A và các chữ D, U, G. Lẽ ra có 14! cách sắp xếp, nhưng trong mỗi cách sắp xếp nếu đổi chỗ 5 chữ N (có 5! cách đổi chỗ), đổi chỗ 3 chữ H (có 3! cách đổi chỗ) và đổi chỗ 3 chữ A (có 3! cách đổi chỗ), thì xâu thu được là không đổi. Vậy nếu gọi số xâu thu được là m thì theo nguyên lý nhân, ta có:

$$m.5!.3!.3! = 14!$$

Từ đó số các xâu khác nhau có được là: $m = \dfrac{14!}{5!.3!.3!} = 28\,180\,160$

Bằng lý luận tương tự ta có: Số hoán vị của n phần tử trong đó có n_1 phần tử giống nhau thuộc loại một, n_2 phần tử giống nhau thuộc loại hai, …, n_k phần tử giống nhau thuộc loại k ($n_1 + n_2 + \ldots + n_k \le n$) là:

$$\frac{n!}{n_1!\, n_2! \ldots n_k!}$$

Chú ý: Có nhiều cách để suy ra công thức tính số hoán vị lặp. Chẳng hạn, trong n vị trí ta chọn n_1 vị trí để xếp các phần tử loại một, như vậy có $C_n^{n_1}$ cách chọn; sau đó chọn n_2 vị trí trong $n - n_1$ vị trí còn lại để xếp các phần tử loại hai, tức là có $C_{n-n_1}^{n_2}$; v.v … Từ đó dùng nguyên lý nhân và công thức tính số tổ hợp được kết quả như đã biết.

d. Sự phân chia các tập hợp thành các tập con khác nhau

Trước hết xét thí dụ: Có bao nhiêu cách chia một cỗ bài 52 quân cho 4 người sao cho một người được 10 quân, những người khác lần lượt được 8, 8, 7 quân?

Giải: Người đầu tiên nhận 10 quân bài có C_{52}^{10} cách nhận khác nhau, còn lại 42 quân nên người tiếp theo có C_{42}^{8} cách nhận 8 quân bài, còn lại 34 quân bài nên người thứ ba có C_{34}^{8} cách nhận 8 quân bài, và người cuối cùng có C_{26}^{7} cách nhận 7 quân bài. Theo nguyên lý nhân thì số cách chia các quân bài thỏa mãn đầu bài là:

$$C_{52}^{10}C_{42}^{8}C_{34}^{8}C_{26}^{7} = \frac{52!}{10!42!}\,\frac{42!}{8!34!}\,\frac{34!}{8!26!}\,\frac{26!}{7!19!} = \frac{52!}{10!8!8!7!19!} \approx 2{,}23.10^{31}.$$

Tổng quát, ta có: Số cách phân chia một tập hợp gồm n phần tử thành k tập sao cho tập con thứ i có n_i phần tử ($n_1 + n_2 + \ldots + n_k = n$) là:

$$\frac{n!}{n_1!\, n_2! \ldots n_k!}$$

Chú ý là công thức vừa lập được cho thấy thứ tự các tập con không ảnh hưởng đến kết quả phân chia tập hợp đã cho.

3. Nguyên lý bù trừ.

Nếu không có giả thiết A, B là 2 tập hợp rời nhau, ta có nguyên lý cộng tổng quát:

$$N(A \cup B) = N(A) + N(B) - N(A \cap B) \qquad (1)$$

Rõ ràng, nếu A, B rời nhau thì $A \cap B = \varnothing$ nên $N(A \cap B) = 0$ và được nguyên lý cộng đơn giản (mục 1.1):

$$N(A \cup B) = N(A) + N(B)$$

Công thức (1) còn gọi là nguyên lý bù trừ cho 2 tập hợp. Tổng quát hơn, ta có:

Định lý: Giả sử $A_1, A_2, ..., A_n$ là n tập hợp hữu hạn. Khi đó:

$$N(A_1 \cup A_2 \cup \ldots \cup A_n) =$$

$$= \sum_{i=1}^{n} N(A_i) - \sum_{i \neq j} N(A_i \cap A_j) + \sum_{i \neq j \neq k} N(A_i \cap A_j \cap A_k) + \ldots + (-1)^{n-1} N(A_1 \cap A_2 \cap \ldots \cap A_n) \qquad (2)$$

Chứng minh: Chúng ta biết rằng số các tập giao $A_i \cap A_j$ là C_n^2, số các tập giao $A_i \cap A_j \cap A_k$ là C_n^3, ... Tổng quát số các tập giao của r tập là C_n^r, r = 1, 2, ...,n.

Công thức (2) được chứng minh bằng cách chỉ ra rằng, mỗi phần tử của tập hợp $A_1 \cup A_2 \cup ... \cup A_n$ được đếm đúng một lần ở vế phải của (2).

Xét phần tử tuỳ ý x của tập hợp $A_1 \cup A_2 \cup ... \cup A_n$. Giả sử x có mặt trong r tập của $A_1, A_2, ..., A_n$ trong đó r = 1, 2, ..., n. Phần tử x này được đếm C_r^1 lần trong tổng $\Sigma N(A_i)$, đếm C_r^2 trong tổng $\Sigma N(A_i \cap A_j)$, ... Bởi vậy tổng các lần đếm của x ở vế phải của (2) là:

$$C_r^1 - C_r^2 + C_r^3 - \ldots + (-1)^r C_r^r = 1 - [1 - C_r^1 + C_r^2 - C_r^3 + \ldots + (-1)^r C_r^r]$$

$$= 1 - (1 - 1)^r = 1$$

Vậy x được đếm đúng một lần trong vế phải của (2).

Định lý được chứng minh.

Có thể viết công thức (2) dưới dạng khác: Đồng nhất tập A_i với tính chất A_i cho trên một tập hợp X nào đó và phải đếm xem có bao nhiêu phần tử của X không thoả mãn bất cứ tính chất A_i nào (i = 1, 2, ..., n)

Gọi N^* là số cần đếm; N = N(X) (là số phần tử của X), ta có:

$$N^* = N - N(A_1 \cup A_2 \cup \ldots \cup A_n) = N - N_1 + N_2 - \ldots + (-1)^n N_n . \qquad (3)$$

trong đó N_k là tổng các phần tử của X thoả mãn k tính chất lấy từ n tính chất đã cho.

Thí dụ 1: Trong một lớp học có 50 sinh viên thì có 30 người biết tiếng Anh, 20 người biết tiếng Pháp, trong số sinh viên biết ngoại ngữ có 10 người biết cả tiếng Anh và tiếng Pháp. Hỏi trong lớp có bao nhiêu sinh viên không biết cả tiếng Anh và tiếng Pháp?

Giải: Theo công thức (1), ta có số sinh viên biết ít nhất 1 trong 2 thứ tiếng Anh, Pháp là: $30 + 20 - 10 = 40 \Rightarrow$ số sinh viên không biết cả tiếng Anh và tiếng Pháp là 10 người.

Thí dụ 2. Có bao nhiêu xâu nhị phân có độ dài 8 hoặc bắt đầu bằng 1 hoặc kết thúc bằng 00?

Giải: Do vị trí đầu tiên là cố định, chỉ có 1 cách chọn duy nhất, mỗi một trong 7 vị trí tiếp theo có 2 cách chọn (0 hoặc 1), suy ra số xâu nhị phân có độ dài 8 bắt đầu bằng 1 là $2^7 = 128$ xâu. Tương tự số xâu nhị phân có độ dài 8 kết thúc bằng 00 là $2^6 = 64$ và số xâu nhị phân có độ dài 8 bắt đầu bằng 1 và kết thúc 00 là $2^5 = 32$.

Vậy, số xâu nhị phân có độ dài 8 hoặc bắt đầu bằng 1 hoặc kết thúc bằng 00 là

$$128 + 64 - 32 = 160 \text{ (xâu)}$$

Thí dụ 3. Có bao nhiêu số nguyên dương không lớn hơn 100 hoặc chia hết cho 4 hoặc chia hết cho 6 nhưng không đồng thời chia hết cho cả 4 và 6?

Giải: Số các số chia hết cho 4 là: $\left[\frac{100}{4}\right] = 25$,

Số các số chia hết cho 6 là: $\left[\frac{100}{6}\right] = 16$,

Số các số đồng thời chia hết cho cả 4 và 6 là: $\left[\frac{100}{12}\right] = 8$

Vậy số cần tìm là: $25 + 16 - 8 = 33$.

Thí dụ 4. Có bao nhiêu số nguyên nhỏ hơn hoặc bằng 1000 không chia hết cho bất cứ số nào trong các số 3, 4, 5?

Giải: Gọi X là các số nguyên dương nhỏ hơn hoặc bằng 1000 thì $N(X) = 1000$.

$$A_i = \{x \in X \mid x \text{ chia hết cho } i\}, \quad i = 3, 4, 5.$$

Khi đó $A_3 \cup A_4 \cup A_5$ là tập các số trong X chia hết cho ít nhất một trong các số 3, 4, 5 và:

$$N(A_3 \cup A_4 \cup A_5) = N_1 - N_2 + N_3 .$$

Ta có: $N_1 = N(A_3) + N(A_4) + N(A_5) =$

$$= \left[\frac{1000}{3}\right] + \left[\frac{1000}{4}\right] + \left[\frac{1000}{5}\right] = 333 + 250 + 200 = 783$$

$N_2 = N(A_3 \cap A_4) + N(A_3 \cap A_5) + N(A_4 \cap A_5) =$

$$= \left[\frac{1000}{3.4}\right] + \left[\frac{1000}{3.5}\right] + \left[\frac{1000}{4.5}\right] = 83 + 66 + 50 = 199$$

$$N_3 = N(A_3 \cap A_4 \cap A_5) = \left[\frac{1000}{3.4.5}\right] = 16$$

Từ đó có đáp số của bài toán là: $1000 - N_1 + N_2 - N_3 = 400$.

Thí dụ 5. Bài toán bỏ thư

Có n phong bì ghi sẵn địa chỉ và n lá thư viết cho n địa chỉ đó. Bỏ ngẫu nhiên các lá thư vào các phong bì (mỗi lá thư cho vào 1 phong bì). Hãy tìm xem có bao nhiêu cách bỏ thư sao cho không có lá thư nào được bỏ đúng địa chỉ.

Giải: Số cách bỏ thư vào phong bì là n!. Nói cách khác, nếu gọi X là tất cả các cách bỏ thư vào phong bì thì N(X) = n! .

Gọi A_k là tính chất có k lá thư bỏ đúng địa chỉ.

Gọi N^* là số cách bỏ thư sao cho không có lá thư nào đúng địa chỉ.

Theo công thức (3), ta có: $N^* = n! - N_1 + N_2 - ... + (-1)^n N_n$,

trong đó N_k là số tất cả các cách bỏ thư sao cho có k lá thư bỏ đúng địa chỉ.

Để tính số N_k ta lý luận như sau: Có C_n^k cách lấy k lá thư, mỗi cách lấy k lá thư có (n – k)! cách bỏ k lá thư này đúng địa chỉ (vì khi đó n – k lá thư còn lại được bỏ tùy ý), do đó:

$$N_k = C_n^k (n-k)! = \frac{n!}{k!}$$

Từ đó: $$N^* = n!\left(1 - \frac{1}{1!} + \frac{1}{2!} - \Lambda + \frac{(-1)^n}{n!}\right)$$

Số N^* trong bài toán bỏ thư được gọi là số mất thứ tự, ký hiệu là D_n. Dưới đây là một vài giá trị của D_n:

n	2	3	4	5	6	7	8	9	10	11
D_n	1	2	9	44	265	1854	14833	133 496	1 334 961	4 890 741

Số mất thứ tự D_n tăng rất nhanh, người ta gọi đó là hiện tượng bùng nổ tổ hợp.

4. Giải các hệ thức truy hồi

Thường các bài toán đếm phụ thuộc vào tham số đầu vào là số tự nhiên n, chẳng hạn như số tổ hợp, số chỉnh hợp, số mất thứ tự, ... Việc biểu diễn tường minh các kết quả này như một hàm của số tự nhiên n gọi là công thức trực tiếp. Trong nhiều trường hợp việc tìm công thức trực tiếp như thế là khó khăn. Có một phương pháp biểu diễn kết quả đếm được ứng với giá trị đầu vào thông qua các giá trị đầu vào nhỏ hơn và công thức tính thu được gọi là công thức truy hồi. Các giá trị đầu tiên mà kể từ đó công thức truy hồi có hiệu lực gọi là giá trị ban đầu của công thức truy hồi (xem lại định nghĩa trong 6.1. chương 1).

Thí dụ: Có thể tính số tổ hợp C_n^k như sau: Chọn một phần tử a cố định trong n phần tử đang xét. Khi đó, nếu a được chọn vào tổ hợp thì phải chọn thêm k – 1 phần tử nữa từ n – 1 phần tử còn lại, số cách chọn trong trường hợp này là C_{n-1}^{k-1}; còn nếu a không được chọn vào tổ hợp thì phải chọn cả k phần tử trong n – 1 phần tử còn lại, số cách chọn trong trường hợp này là C_n^k. Theo quy tắc cộng ta có:

$$C_n^k = C_{n-1}^{k-1} + C_{n-1}^k$$

đó là công thức truy hồi tính số C_n^k. Các giá trị ban đầu của công thức này là $C_1^0 = C_1^1 = 1$.

Dưới đây chúng ta xét một số phương pháp đưa công thức truy hồi về công thức trực tiếp.

4.1. Phương pháp khử để tìm công thức trực tiếp từ công thức truy hồi

Phương pháp này được xét qua một số thí dụ sau:

Thí dụ 1: Bài toán chia mặt phẳng.

Trên mặt phẳng, kẻ n đường thẳng sao cho không có 2 đường nào song song và không có quá 2 đường thẳng đồng quy tại một điểm. Hỏi mặt phẳng được chia thành bao nhiêu phần?

Giải: Gọi số phần của mặt phẳng được chia bởi n đường thẳng như vậy là S_n. Giả sử đã kẻ được n – 1 đường thẳng, kẻ thêm đường thẳng thứ n thì số phần được thêm sẽ bằng số giao điểm được thêm cộng 1. Số giao điểm được thêm là số giao điểm mà đường thẳng vừa kẻ cắt n – 1 đường thẳng kẻ trước đó, nghĩa là bằng n – 1. Từ đó nhận được công thức truy hồi:

$$S_n = S_{n-1} + n \; ; \; n \geq 1; \quad S_0 = 1.$$

(S_0 là giá trị ban đầu)

Từ công thức trên, dễ dàng tính được:

$S_1 = 1 + 1 = 2$, $S_2 = 2 + 2 = 4$, $S_3 = 4 + 3 = 7$

$S_4 = 7 + 4 = 11$, $S_5 = 11 + 5 = 16$, $S_6 = 16 + 6 = 22$, ...

Để tìm công thức trực tiếp, ta có:

$S_0 = 1$

$S_1 = S_0 + 1$

$S_2 = S_1 + 2$

$S_3 = S_2 + 3$

. . . . . .

$S_n = S_{n-1} + n$

Cộng vế với vế của các đẳng thức trên được:

$$S_n = S_0 + 1 + 2 + 3 + ... + n = 1 + \frac{n(n-1)}{2} = \frac{n^2 + n + 2}{2}$$

Thí dụ 2: Bài toán lãi kép.

Một người gửi tiết kiệm 10 triệu đồng với lãi suất hàng năm là 10%. Hết một năm gửi tiền, nếu không rút tiền ra người đó được cộng số lãi vào gốc để tính lãi cho năm tiếp theo (lãi kép). Hỏi sau 20 năm gửi không rút ra lần nào thì số tiền của người đó là bao nhiêu?

Giải: Gọi P_n là số tiền của người đó sau n năm gửi, khi đó số tiền của người đó bằng số tiền của năm thứ n – 1 cộng với lãi suất của năm thứ n. Như vậy:

$$P_n = P_{n-1} + 0{,}1\, P_{n-1} = 1{,}1\, P_{n-1}.$$

Đó là công thức truy hồi tính số tiền sau n năm gửi của người đó. Giá trị ban đầu của bài toán là $P_0 = 10$ (triệu).

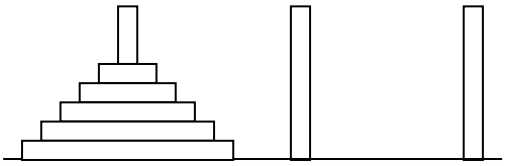
Thế lần lượt $P_{n-1} = 1{,}1\ P_{n-2}$ vào P_n, $P_{n-2} = 1{,}1\ P_{n-3}$ vào P_{n-2}; ... Cuối cùng ta có:

$$P_n = 1{,}1^n\ P_0.$$

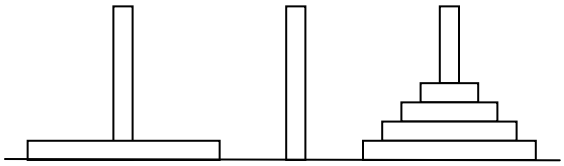
Sau 20 năm số tiền của người đó là: $P_{20} = 1{,}1^{20}.10 \approx 6{,}7275.\ 10 = 67{,}275$ triệu đồng.

Thí dụ 3: Bài toán tháp Hà nội.

Tương truyền rằng, tại một ngôi chùa ở Hà nội có một tấm đế bằng đồng, trên đó có 3 cái cọc bằng kim cương. Lúc khai thiên lập địa, trên một trong 3 cái cọc đó Đức Phật đã để 64 chiếc đĩa bằng vàng với đường kính nhỏ dần (hình 1). Ngày đêm các nhà sư phải dịch chuyển đĩa sang một cọc khác theo nguyên tắc: mỗi lần chỉ được chuyển 1 đĩa từ cọc này sang cọc khác bất kỳ nhưng không được đặt một chiếc đĩa to lên trên một chiếc đĩa khác nhỏ hơn. Hỏi phải mất bao lâu các nhà sư mới chuyển hết số đĩa sang một cọc khác, biết rằng mỗi lần chuyển 1 đĩa hết 1 giây?



Hình 1. Trạng thái ban đầu



Hình 2. Trạng thái sau H_{n-1} lần chuyển

Giải: Gọi H_n là số lần dịch chuyển đĩa nếu cọc ban đầu có n đĩa.

Giả sử lúc đầu n đĩa ở trên cọc 1 (hình 1). Sau H_{n-1} lần dịch chuyển các nhà sư đã chuyển được n – 1 đĩa sang cọc 3 (hình 2). Từ đây phải có 1 lần chuyển chiếc đĩa lớn nhất từ cọc 1 sang cọc 2 và H_{n-1} lần dịch chuyển n – 1 chiếc đĩa từ cọc 3 sang cọc 2. Như vậy, ta có:

$$H_n = 2\ H_{n-1} + 1$$

đó là công thức truy hồi để tính số lần dịch chuyển n đĩa từ cọc này sang cọc khác. Dễ thấy giá trị ban đầu là $H_1 = 1$ (cọc chỉ có 1 đĩa).

Chúng ta tìm công thức trực tiếp tính H_n. Ta có:

$$H_n = 2\ H_{n-1} + 1 = 2\ (2\ H_{n-2} + 1) + 1 = 2^2\ H_{n-2} + 2 + 1 =$$
$$= 2^2\ (2\ H_{n-3} + 1) + 2 + 1 = 2^3\ H_{n-3} + 2^2 + 2 + 1 =$$
$$. \quad . \quad . \quad . \quad .$$
$$= 2^{n-1}\ H_1 + 2^{n-2} + ... + 2 + 1 =$$
$$= 2^n - 1.$$

Với 64 chiếc đĩa đã cho, các nhà sư phải dịch chuyển:

$$H_{64} = 2^{64} - 1 = 18{,}4467.10^{18} \text{ (hơn 18 tỷ tỷ lần)}$$

Nếu mỗi lần dịch chuyển hết 1 giây, các nhà sư cần khoảng 580 tỷ năm mới hoàn thành việc chuyển 64 đĩa từ cọc này sang cọc khác.

Thí dụ 4: Lập công thức truy hồi tính số mất thứ tự D_n.

Giải: Đánh số thư và phong bì từ 1 đến n, thư thứ i đúng với địa chỉ phong bì i . Một cách bỏ thư đồng nhất với một hoán vị $(a_1, a_2, ..., a_n)$ của (1, 2, ..., n). Một cách mất thứ tự là một hoán vị $(a_1, a_2, ..., a_n)$ sao cho $a_i \neq i$, $\forall i$. Thành phần a_1 của hoán vị có thể nhận n – 1 giá trị ngoài số 1. Với mỗi giá trị k của a_1 ($k \neq 1$) có thể xảy ra 2 trường hợp:

1) $a_k = 1$, khi đó các thành phần còn lại được xác định như số cách mất thứ tự của n – 2 phần tử , số này chính là D_{n-2} .
2) $a_k \neq 1$, khi đó các thành phần còn lại được xác định như số cách mất thứ tự của n – 1 phần tử (xem giá trị 1 như là giá trị k), số này chính là D_{n-1}.

Từ đó có công thức truy hồi:

$$D_n = (n-1)(D_{n-2} + D_{n-1}) , \quad n \geq 3 .$$

Các giá trị ban đầu có thể tính trực tiếp từ định nghĩa và được $D_1 = 0$, $D_2 = 1$.

Nếu coi $D_0 = 1$ thì công thức truy hồi trên đúng $\forall n \geq 2$.

Ta có: $D_3 = (3-1)(0+1) = 2$, $D_4 = (4-1)(1+2) = 9$,

$D_5 = (5-1)(2+9) = 44$, $D_6 = (6-1)(9+44) = 265$, ...

Có thể đưa công thức truy hồi trên về công thức trực tiếp như sau:

$$D_n = (n-1)(D_{n-2} + D_{n-1}) \quad \Leftrightarrow \quad D_n - nD_{n-1} = -[D_{n-1} - (n-1)D_{n-2}]$$

Đặt $I_n = D_n - nD_{n-1}$. Ta có:

$$D_n - n D_{n-1} = I_n = -I_{n-1} = I_{n-2} = ... = (-1)^{n-1} I_1 = (-1)^n.$$

Chia cả 2 vế cho n! , được:

$$\frac{D_n}{n!} - \frac{D_{n-1}}{(n-1)!} = \frac{(-1)^n}{n!}, \qquad \forall n \geq 2$$

Cộng vế với vế của hệ thức trên được :

$$\frac{D_n}{n!} = 1 - \frac{1}{1!} + \frac{2}{2!} + ... + \frac{(-1)^n}{n!} \quad \Rightarrow \quad D_n = n!\left(1 - \frac{1}{1!} + \frac{1}{2!} + ... + \frac{(-1)^n}{n!}\right)$$

Công thức thu được là công thức đã đã biết trong thí dụ 5, mục 3 (Nguyên lý bù trừ).

4.2. Giải các hệ thức truy hồi tuyến tính.

Trong phần này xét cách tìm công thức trực tiếp từ công thức truy hồi có dạng đặc biệt gọi là dạng tuyến tính.

a. Các khái niệm

Định nghĩa: Công thức truy hồi tuyến tính thuần nhất bậc k có hệ số hằng số là công thức truy hồi có dạng:

$$a_n = c_1 a_{n-1} + c_2 a_{n-2} + ... + c_k a_{n-k} \qquad (1)$$

trong đó $c_1, c_2, ..., c_k$ là các số thực (hằng số) và $c_k \neq 0$.

Sở dĩ có thuật ngữ "công thức truy hồi bậc k" vì số hạng a_n được tính qua k số hạng trước nó. Các giá trị ban đầu của công thức truy hồi trên gồm k giá trị: $a_0, a_1, ..., a_{k-1}$.

Có thể tìm công thức trực tiếp tính a_n dưới dạng:

$$a_n = r^n . \qquad (2)$$

Trong đó r là một số thực cần xác định.

Dãy $\{ a_n \}$ xác định theo (2) sẽ thoả mãn (1) nếu r là nghiệm của phương trình:

$$r^n = c_1 r^{n-1} + c_2 r^{n-2} + ... + c_k r^{n-k} .$$

hay: $$r^n - c_1 r^{n-1} - c_2 r^{n-2} - ... - c_k r^{n-k} = 0. \qquad (3)$$

Phương trình (3) gọi là phương trình đặc trưng của công thức (1), và nghiệm của (3) gọi là nghiệm đặc trưng.

Trước hết xét với k = 2, tức là công thức truy hồi tuyến tính bậc hai:

$$a_n = c_1 a_{n-1} + c_2 a_{n-2} .$$

Phương trình đặc trưng của công thức truy hồi tuyến tính bậc hai là:

$$r^2 - c_1 r - c_2 = 0$$

b. Giải công thức truy hồi tuyến tính bậc hai khi phương trình đặc trưng có 2 nghiệm phân biệt.

Định lý 1: Cho c_1, c_2 là hai số thực. Nếu phương trình đặc trưng $r^2 - c_1 r - c_2 = 0$ có hai nghiệm thực phân biệt r_1, r_2 thì điều kiện cần và đủ để dãy số $\{a_n\}$ là nghiệm của công thức truy hồi:

$$a_n = c_1 a_{n-1} + c_2 a_{n-2}$$

là: $$a_n = \alpha_1 r_1^n + \alpha_2 r_2^n , \quad n = 1, 2, ...$$

trong đó α_1, α_2 là các số thực có thể xác định được nhờ các điều kiện ban đầu của công thức truy hồi.

Chứng minh:

Điều kiện cần: Nếu r_1, r_2 là 2 nghiệm của phương trình đặc trưng, ta phải chứng minh

$$a_n = \alpha_1 r_1^n + \alpha_2 r_2^n .$$

với α_1, α_2 là các hằng số thoả mãn công thức truy hồi $a_n = c_1 a_{n-1} + c_2 a_{n-2}$.

Thật vậy: Ta có: $r_1^2 - c_1 r_1 - c_2 = 0$ và $r_2^2 - c_1 r_2 - c_2 = 0$

hay: $r_1^2 = c_1 r_1 + c_2$ và $r_2^2 = c_1 r_2 + c_2$

Từ đó: $a_n = c_1 a_{n-1} + c_2 a_{n-2} = c_1(\alpha_1 r_1^{n-1} + \alpha_2 r_2^{n-1}) + c_2(\alpha_1 r_1^{n-2} + \alpha_2 r_2^{n-2}) =$

$$= \alpha_1 r_1^{n-2}(c_1 r_1 + c_2) + \alpha_2 r_2^{n-2}(c_1 r_2 + c_2) =$$

$$= \alpha_1 r_1^{n-2} r_1^2 + \alpha_2 r_2^{n-2} r_2^2 =$$

$$= \alpha_1 r_1^n + \alpha_2 r_2^n .$$

Điều kiện đủ:

Giả sử $\{ a_n \}$ là một dãy bất kỳ thoả mãn hệ thức truy hồi $a_n = c_1 a_{n-1} + c_2 a_{n-2}$. Ta phải chứng minh rằng sẽ chọn được các số α_1 , α_2 sao cho $a_n = \alpha_1 r_1^n + \alpha_2 r_2^n$.

Thật vậy, giả sử a_0 và a_1 là 2 giá trị ban đầu của hệ thức truy hồi. Ta có:

$$\begin{cases} a_0 = \alpha_1 + \alpha_2 \\ a_1 = \alpha_1 r_1 + \alpha_2 r_2 \end{cases}$$

Đây là hệ phương trình tuyến tính 2 ẩn Cramer, nó có nghiệm duy nhất:

$$\alpha_1 = \frac{a_1 - r_2 a_0}{r_1 - r_2}; \qquad \alpha_2 = \frac{a_0 r_1 - a_1}{r_1 - r_2}$$

Vậy $a_n = \alpha_1 r_1^n + \alpha_2 r_2^n$ được xác định.

Thí dụ: Dãy Fibonacci, như đã biết được xác định bởi công thức truy hồi:

$$F_n = F_{n-1} + F_{n-2}; \quad F_0 = 0; \quad F_1 = 1.$$

Đó là công thức truy hồi tuyến tính bậc hai có hệ số không đổi. Ta đi tìm công thức trực tiếp tính F_n.

Giải: Phương trình đặc trưng của công thức là:

$$r^2 - r - 1 = 0 \quad \Leftrightarrow \quad r_{1,2} = \frac{1 \pm \sqrt{5}}{2}$$

Từ đó, theo định lý 1, ta có:

$$F_n = \alpha_1 \left(\frac{1+\sqrt{5}}{2}\right)^n + \alpha_2 \left(\frac{1-\sqrt{5}}{2}\right)^n$$

Thay các điều kiện ban đầu $F_0 = 0$, $F_1 = 1$ vào được hệ phương trình:

$$\begin{cases} \alpha_1 + \alpha_2 = 0 \\ \frac{1+\sqrt{5}}{2}\alpha_1 + \frac{1+\sqrt{5}}{2}\alpha_2 = 1 \end{cases}$$

Giải hệ phương trình này được nghiệm: $\alpha_1 = \frac{1}{\sqrt{5}}$; $\alpha_2 = -\frac{1}{\sqrt{5}}$

Vậy: $$F_n = \frac{1}{\sqrt{5}}\left(\frac{1+\sqrt{5}}{2}\right)^n - \frac{1}{\sqrt{5}}\left(\frac{1-\sqrt{5}}{2}\right)^n.$$

c. Giải công thức truy hồi tuyến tính bậc hai khi phương trình đặc trưng có nghiệm kép

Định lý 2: Cho c_1, c_2 là hai số thực. Nếu phương trình đặc trưng $r^2 - c_1 r - c_2 = 0$ có nghiệm kép r_0 thì điều kiện cần và đủ để dãy số $\{a_n\}$ là nghiệm của công thức truy hồi:

$$a_n = c_1 a_{n-1} + c_2 a_{n-2}$$

là: $$a_n = \alpha_1 r_0^n + \alpha_2 n r_0^n, \quad n = 1, 2, ...$$

trong đó α_1, α_2 là các số thực có thể xác định được nhờ các điều kiện ban đầu của công thức truy hồi.

Việc chứng minh định lý 2 tương tự chứng minh định lý 1.

Thí dụ: Tìm nghiệm của công thức truy hồi:

$$a_n = 6a_{n-1} - 9a_{n-2}; \quad a_0 = 1,\ a_1 = 6.$$

Giải: Phương trình đặc trưng $r^2 - 6r + 9 = 0$ có nghiệm kép $r = 3$.

Vậy a_n có dạng: $a_n = \alpha_1 3^n + \alpha_2 n 3^n$.

Thay $n = 0$, $n = 1$, ta có:

$$\begin{cases} \alpha_1 = a_0 = 1 \\ 3\alpha_1 + 3\alpha_2 = a_1 = 6 \end{cases} \Leftrightarrow \quad \alpha_1 = \alpha_2 = 1$$

Vậy: $a_n = 3^n + n3^n = (n+1)3^n$.

d. Chú thích

Có thể mở rộng định lý 1 cho hệ thức truy hồi tuyến tính bậc k như sau:

Cho $c_1, c_2, \ldots, c_k$ là các số thực. Giả sử phương trình đặc trưng:

$$r^k - c_1 r^{k-1} - \ldots - c_{k-1} r - c_k = 0$$

có k nghiệm phân biệt $r_1, r_2, \ldots, r_k$. Khi đó điều kiện cần và đủ để dãy $\{a_n\}$ là nghiệm của công thức truy hồi:

$$a_n = c_1 a_{n-1} + c_2 a_{n-2} + \ldots + c_k a_{n-k}$$

là: $a_n = \alpha_1 r_1^n + \alpha_2 r_2^n + \ldots + \alpha_k r_k^n$

trong đó $\alpha_1, \alpha_2, \ldots, \alpha_k$ là các số thực có thể xác định được nhờ các điều kiện ban đầu của công thức truy hồi đã cho.

Thí dụ: Tìm nghiệm của công thức truy hồi:

$$a_n = 6a_{n-1} - 1a_{n-2} + 6a_{n-3}; \quad a_0 = 2,\ a_1 = 5,\ a_2 = 15.$$

Giải: Phương trình đặc trưng của công thức truy hồi đã cho là:

$$r^3 - 6r^2 + 11r - 6 = 0$$

Các nghiệm của phương trình đặc trưng là: $r = 1$, $r = 2$, $r = 3$. Vậy công thức trực tiếp tính a_n có dạng:

$$a_n = \alpha_1 1^n + \alpha_2 2^n + \alpha_3 3^n.$$

Thay $n = 0$, $n = 1$, $n = 2$, $n = 3$ vào công thức tính a_n, được hệ phương trình tuyến tính theo $\alpha_1, \alpha_2, \alpha_3$:

$$\begin{cases} \alpha_1 + \alpha_2 + \alpha_3 = 2 \\ \alpha_1 + 2\alpha_2 + 3\alpha_3 = 5 \\ \alpha_1 + 4\alpha_2 + 9\alpha_3 = 15 \end{cases}$$

Giải hệ phương trình được nghiệm: $\alpha_1 = 1$, $\alpha_2 = -1$, $\alpha_3 = 2$. Vậy dạng hiện của công thức truy hồi đã cho là:

$$a_n = 1 - 2^n + 2.3^n.$$

5. Bài toán liệt kê.

Có những bài toán đếm các cấu hình của tổ hợp, ngoài việc đếm số lượng các cấu hình thoả mãn còn phải liệt kê tất cả các cấu hình cần đếm. Đó là các bài toán phải tìm các cấu hình tổ hợp thoả mãn thêm một hoặc một số điều kiện nào đó. Thí dụ như phải tìm các tập con của một tập hợp có 5 số sao cho tổng của các phần tử của tập con đó bằng 50, khi đó phải kiểm tra tổng các số trong mỗi tập của $2^5 = 32$ tập con của tập đã cho để tìm các tập thoả mãn điều kiện đã nêu. Muốn vậy, phải liệt kê các phần tử của mỗi tập trong 32 tập con đó.

Việc liệt kê các cấu hình phải thoả mãn các nguyên tắc sau:

- Không được lặp lại một cấu hình đã đếm.
- Không được để sót một cấu hình.

Sau đây chúng ta tìm hiểu một số thuật toán liệt kê thường gặp.

5.1. Phương pháp sinh

Để thực hiện phương pháp này, bài toán cần thoả mãn 2 điều kiện:

1- Có thể xác định được một thứ tự trên tập các cấu hình cần đếm. Từ đó có thể xác định được cấu hình đầu tiên và cấu hình cuối cùng theo thứ tự đã xác định.

2- Xây dựng được thuật toán để từ một cấu hình chưa phải là cuối cùng đang có có thể đưa ra một cấu hình kế tiếp theo thứ tự đã xác định.

Đặt tên thuật toán theo điều kiện 2 là thuật toán Sinh_kế_tiếp. Có thể mô phỏng thuật toán sinh như sau:

```
Agorithm:  Tổng quát về thuật toán sinh.
Procedue   Phương pháp sinh;
   Begin
      < Xây dựng cấu hình ban đầu >
     stop := false;
     while  not stop  do
    begin
        < đưa ra cấu hình đang có >
        Sinh_kế_tiếp ;
    end;
   End;
```

Biến stop là một biến Boole dùng để dừng chương trình, khi đến cấu hình cuối cùng thì stop nhận giá trị true và chương trình dừng.

Sinh_kế_tiếp là một thủ tục con nhằm xây dựng cấu hình kế tiếp của cấu hình đang có.

Chú ý rằng thứ tự các cấu hình xác định trong điều kiện 1 cần lựa chọn sao cho có thể xây dựng được thuật toán sinh_kế_tiếp.

Xét 3 bài toán cụ thể sử dụng phương pháp sinh.

a. Liệt kê các xâu nhị phân có độ dài n

Mỗi xâu nhị phân $b = b_{n-1} b_{n-2} \dots b_0$, trong đó $b_i \in \{0, 1\}$ có thể xem là biểu diễn nhị phân của 1 số nguyên P(b) nào đó. Vậy có thể lấy thứ tự tăng của số nguyên để xác định thứ tự các xâu nhị phân.

Xâu nhị phân $b = b_{n-1} b_{n-2} \dots b_0$ được gọi là đi trước xâu $a = a_{n-1} a_{n-2} \dots a_0$, (hay a kế tiếp b) nếu P(b) < P(a), ký hiệu là b < a. Cách xác định thứ tự như vậy gọi là thứ tự tự nhiên hay còn gọi là *thứ tự từ điển*.

Chẳng hạn với n = 3, có thứ tự từ điển của các xâu như sau:

Thứ tự của xâu	0	1	2	3	4	5	6	7
Xâu cụ thể	000	001	010	011	100	101	110	111

Như vậy xâu đầu tiên là gồm toàn số 0 và xâu cuối cùng gồm toàn số 1. Từ đó suy ra quy tắc sinh xâu nhị phân kế tiếp như sau:

- Xâu đầu tiên gồm toàn số 0: 00 ... 0
- Giả sử đang có xâu $b_{n-1} b_{n-2} \dots b_0$, khi đó:

  - Tìm từ phải sang, tức là từ b_0, b_1,... đến khi gặp bít 0 đầu tiên, chẳng hạn đó là b_i,
  - Thay $b_i = 1$ và $b_j = 0$ $\forall j < i$ xâu mới thu được là xâu kế tiếp của xâu đang có.

Thí dụ: Tìm xâu kế tiếp của xâu 1 000 100 111 (n =10).

Đi từ bên phải sang, bit đầu tiên bằng 0 là bit thứ 4, tức là $b_3 = 0$. Vậy thay $b_3 = 1$, $b_2 = b_1 = b_0 = 0$ được xâu kế tiếp của xâu đã cho là 1 000 101 000.

(Rõ ràng là: $1\,000\,100\,111 = 2^9 + 2^5 + 2^2 + 2^1 + 2^0 = 541$

và $1\,000\,101\,000 = 2^9 + 2^5 + 2^3 = 542$)

Thuật toán vừa trình bày có thể mô tả như sau:

Algorithm: Sinh xâu nhị phân kế tiếp.
Procedure Xâu nhị phân kế tiếp;
Input: Xâu nhị phân $b_{n-1} b_{n-2} \dots b_0$, $\exists b_i = 0$;
Output: Xâu nhị phân kế tiếp $a_{n-1} a_{n-2} \dots a_0$;

```
Begin
 i := 0;
 while bi = 1 do
     begin
      bi := 0;
           i := i + 1 ;
      end;
 bi := 1;
End;
```

b. Liệt kê các tổ hợp chập k của một tập hợp có n phần tử

Xét bài toán liệt kê tất cả các tổ hợp chập k của tập hợp X = {1, 2, ..., n}

Vì mỗi tổ hợp chập k của n phần tử đã cho là một tập con gồm k phần tử lấy từ n phần tử đó nên phải xếp thứ tự các tập con đó. Trước hết biểu diễn mỗi tập con k phần tử thành một bộ có thứ tự:

$$a = (a_1, a_2, ...,a_k) ;\ 1 \le a_1 < a_2 < ... < a_k \le n$$

và định nghĩa: Tập con $a = (a_1, a_2, ...,a_k)$ gọi là đi trước tập con $a' = (a'_1, a'_2, ...,a'_k)$ theo thứ tự từ điển, ký hiệu $a \propto a'$, nếu tìm được một chỉ số i $(1 \le i \le k)$ sao cho:

$$a_1 = a'_1 ,\ \ a_2 = a'_2 , ... ,\ \ a_{i-1} = a'_{i-1} ,\ \ a_i < a'_i .$$

Thí dụ: Cho X = {1, 2, 3, 4, 5}. Các tổ hợp chập 3 của X được liệt kê theo thứ tự từ điển là:

{1, 2, 3} ⇒ {1, 2, 4} ⇒ {1, 2, 5} ⇒
{1, 3, 4} ⇒ {1, 3, 5} ⇒ {1, 4, 5} ⇒
{2, 3, 4} ⇒ {2, 3, 5} ⇒ {2, 4, 5} ⇒ {3, 4, 5}

Như vậy tập con đầu tiên là {1, 2, ..., k} và tập con cuối cùng là {n – k+1, n–k+2, ..., n}.

Từ đó suy ra quy tắc sinh tập con kế tiếp như sau:

- Tìm từ phải qua trái của dãy phần tử đầu tiên thoả mãn $a_i \ne n - k + i$,
- Thay a_i bằng $a_i + 1$,
- Thay a_j bằng $a_i + j - i$, với $j = i + 1, i + 2, ..., k$.

Thí dụ: Tìm tổ hợp chập 4 từ tập {1, 2, 3, 4, 5, 6} kế tiếp tổ hợp {1, 2, 5, 6}

Giải: Lùi từ phải sang trái thấy $a_2 = 2$ là số hạng đầu tiên thoả mãn:

$$a_2 = 2 \ne 6 - 4 + 2 = 4.$$

Như vậy tổ hợp kế tiếp là:

$$a_1 = 1, a_2 = 2 + 1 = 3, a_3 = 3 + 3 - 2 = 4, a_4 = 3 + 4 - 2 = 5,$$

tức là tập {1, 3, 4, 5}

Algorithm: Sinh tổ hợp kế tiếp.
Procedure Tổ hợp kế tiếp;
Input: Tổ hợp $\{a_1, a_2, ...,a_k\}$ không trùng với $\{n - k + 1, ... , n\}$;
Output: Tổ hợp kế tiếp $\{a'_1, a'_2, ...,a'_k\}$;
Begin
i := k ;
while $a_i = n - k + i$ do
i := i – 1;
$a_i := a_i + 1$;
for j := i + 1 to k do $a_j = a_i + j - i$;
End;

c. Liệt kê các hoán vị của tập hợp có n phần tử

Bài toán đặt ra là: Liệt kê tất cả các hoán vị của tập hợp X = {1, 2, ..., n}

Mỗi hoán vị của X có thể biểu diễn bởi bộ có thứ tự n thành phần

$$a = (a_1, a_2, ...,a_n) ; \quad a_i \neq a_j \text{ khi } i \neq j$$

Định nghĩa thứ tự từ điển của các hoán vị như sau: Hoán vị $a = (a_1, a_2, ...,a_n)$ gọi là đi trước hoán vị $b = (b_1, b_2, ...,b_n)$, ký hiệu $a \propto b$, nếu tìm được một chỉ số k ($1 \leq k \leq n$) sao cho:

$$a_1 = b_1 , \quad a_2 = b_2 , ... , \quad a_{k-1} = b_{k-1} , \quad a_k < b_k .$$

Thí dụ: Các hoán vị của {1, 2, 3} được liệt kê theo thứ tự từ điển là:

(1 2 3) (1 3 2) (2 1 3) (2 3 1) (3 1 2) (3 2 1)

Như vậy hoán vị đầu tiên là (1, 2, ..., n) và hoán vị cuối cùng là (n, n – 1, ...,1).

Từ đó có thể chỉ ra quy tắc sinh hoán vị kế tiếp hoán vị $a = (a_1, a_2, ...,a_n) \neq (n, n–1,..., 1)$ như sau:

- Tìm từ phải qua trái của hoán vị đang xét có chỉ số i đầu tiên thoả mãn $a_i < a_{i+1}$.
- Tìm a_k ở bên phải a_i (tức là $k > i$) theo điều kiện a_k là số nhỏ nhất trong các số được chọn sao cho $a_k > a_i$, đổi chỗ a_i, a_k.
- Lật ngược đoạn từ a_{i+1} đến a_n (xếp lại các số từ a_{i+1} đến a_n theo thứ tự tăng dần)

Thí dụ: Tìm hoán vị kế tiếp của hoán vị (3, 6, 2, 5, 4, 1)

Số đầu tiên tính từ cuối lên mà nhỏ hơn số sau nó là $a_3 = 2 < a_4 = 5$.

Số nhỏ nhất trong các số đứng sau $a_3 = 2$ mà lớn hơn a_3 là $a_5 = 4$.

Đổi chỗ $a_3 = 2$ và $a_5 = 4$ ta được (3, 6, 4, 5, 2, 1).

Lật ngược $(a_4, a_5, a_6) = (5, 2, 1)$ được hoán vị kế tiếp là (3, 6, 4, 1, 2, 5).

Thuật toán được mô phỏng như sau:

```
Algorithm  Sinh hoán vị kế tiếp.
Procedure  Hoán vị kế tiếp;
    Input:    Hoán vị  (a1, a2, ...,an) ≠ (n, n – 1, ..., 1);
    Output:  Hoán vị kế tiếp  (b1, b2, ...,bn);
    Begin
          (* Tìm chỉ số j lớn nhất thoả mãn aj < aj + 1*)
      j := n – 1;
      while  aj > aj + 1  do j:= j – 1;
        (* Tìm ak là số nhỏ nhất còn lớn hơn aj ở bên phải aj*)
      k := n;
      while aj > ak  do  k := k – 1;
      Đổi chỗ (aj ,ak )     (* Đổi chỗ aj với  ak *)
         (*Lật ngược đoạn từ aj + 1 đến  an *)
      r := n;
      s := j + 1;
      while   r > s   do
     begin
```

```
        đổi chỗ (ar, as);
        r := r – 1;
        s := s + 1;
      end;
    End;
  (* Đổichỗ (x, y) là thủ tục con nhằm đổi chỗ x với y:   z := x;   x := y;   y := z *)
```

Nói chung, phương pháp sinh có nhiều hạn chế vì không phải cấu hình nào cũng sinh được cấu hình kế tiếp một cách đơn giản. Mặt khác, ngay cả cấu hình đầu tiên cũng như cấu hình cuối cùng cũng không dễ xác định được. Để giải bài toán liệt kê các cấu hình tổ hợp, người ta thường dùng thuật toán quay lui dưới đây.

5.2. Thuật toán quay lui

Nội dung của thuật toán là tìm dần các thành phần của cấu hình bằng cách thử tất cả các khả năng có thể được. Cụ thể như sau:

Giả sử cần tìm cấu hình được biểu diễn bằng một bộ n thành phần (x_1, x_2, ..., x_n) và đã tìm được k – 1 thành phần của cấu hình đó là x_1, x_2, ..., x_{k-1}. Khi đó thành phần x_k của cấu hình được xác định theo cách sau:

Gọi T_k là tập hợp tất cả các thành phần mà x_k có thể nhận được. Tập T_k là hữu hạn. Giả sử $N(T_k) = n_k$, do đó lần lượt duyệt tất cả các phần tử (thành phần) của T_k. Để việc duyệt được thuận lợi, cần đánh số cho các phần tử của T_k. Với mỗi khả năng j, tức là chọn phần tử x_j ($1 \le j \le n_k$) phải thử xem x_j có chấp nhận được không. Khi đó xảy ra 2 tình huống:

- Nếu x_j chấp nhận được thì đưa x_j vào cấu hình đang xét, tức là thành phần x_k của cấu hình là x_j. Nếu k = n thì cấu hình cần liệt kê được xác định, nếu k < n thì tiến hành xác định thành phần x_{k+1}.
- Nếu x_j không chấp nhận được thì quay lại bước trước để xác định lại x_k.

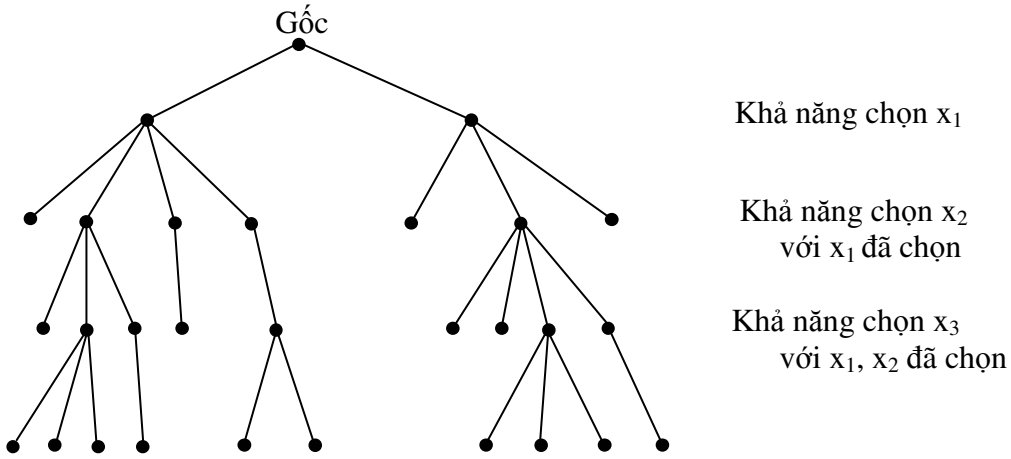
Thủ tục đệ quy cho thuật toán quay lui như sau:

```
Algorithm  Thuật toán quay lui.
Procedure  Thử (k: integer);
    var  j: integer;
    Begin
  for  j := 1  to  n  do
  if  <chấp nhận j >  then
      begin
        < Xác định xk theo j >;
        if   k = n  then  < ghi nhận một cấu hình >
                  else  Thử (k + 1);
      end;
    End;
```

Điều cần quan tâm trong thủ tục trên là:

1) Làm thế nào để xác định được tập T_k là tập tất cả các thành phần mà x_k có thể nhận.
2) Xác định giá trị biểu thức logic < chấp nhận j >. Giá trị này ngoài việc phụ thuộc vào j còn phụ thuộc vào các thành phần đã chọn tại k – 1 bước trước. Như vậy cần ghi nhớ trạng thái cũ sau lời gọi Thử(k+1). Các trạng thái này được ghi nhận nhờ một biến logic (biến Boole) và gọi là biến trạng thái.

Thuật toán quay lui có thể mô tả bằng cây tìm kiếm như sau:



Hình 3. Cây liệt kê lới giải theo thuật toán quay lui

Ngoài thủ tục đệ quy Thử(k), chương trình cần có thêm các thủ tục Khởi tạo và In kết quả. Chương trình chính giải bài toán liệt kê có dạng:

```
Begin
    Khoi_tao;
    Thu;
End.
```

Thí dụ 1: Liệt kê tất cả các hoán vị của n số tự nhiên dương đầu tiên.

Mỗi hoán vị có thể được biểu diễn dưới dạng x[1], x[2], ..., x[n] trong đó x[i] nhận giá trị từ 1 đến n và x[i] ≠ x[j]. Các giá trị từ 1 đến n lần lượt đề cử cho x[k], trong đó giá trị j được chấp nhận nếu nó chưa được dùng. Nói cách khác, ta có $T_1 = N = \{1, 2, ..., n\}$ và nếu đã xác định được x[1], x[2], ..., x[k – 1] thì $T_k = N \setminus \{x[1], x[2], ..., x[k–1]\}$. Vì thế cần phải ghi nhớ xem giá trị j đã được dùng hay chưa. Điều này được thực hiện nhờ một dãy biến logic b[j], trong đó b[j] là true nếu j chưa được dùng. Do đó chúng được gán giá trị true cho mọi b[j] trong thủ tục khởi tạo và sau khi gán giá trị j cho x_k thì b[j] nhận giá trị false, cuối cùng lại phải gán true cho b[j] sau khi thực hiện xong bước in kết quả hay thực hiện xong thủ tục thu(k+1) để sử dụng cho bước sau.

```
Program Liet_ke_hoan_vi;
  uses crt;
  var  n: integer;
```

```
    x: array[1..20] of integer;
    b: array[1..20] of boolean;
    dem: word;
  Procedure khoitao;
    var i: integer;
    begin
      write (' So phan tu n = '); readln(n);
      for i:=1 to n do b[i] := true;
      dem := 0;
    end;
  Procedure in_kq;
    var i: integer;
    begin
      dem := dem+1;
      write (' Hoan vi thu',dem:3,':');
      for i:=1 to n do write(x[i]:4);
      writeln;
    end;
  Procedure thu(k: integer);
    var j: integer;
    begin
      for j:=1 to n do
      if b[j] then
        begin
          x[k] := j;
          b[j] := false;  {chuyển trạng thái mới}
          if k=n then in_kq
                 else thu(k+1);
          b[j] := true;  {trả lại trạng thái cũ}
        end;
   end;
BEGIN {main}
  clrscr;
  khoitao;
  thu(1);
  readln;
END.
```

Thí dụ 2: Liệt kê tất cả các chuỗi nhị phân có độ dài n

Mỗi chuỗi nhị phân độ dài n có dạng x[1] x[2] ... x[n] với x[i] ∈ B; B = {0, 1}. Giả sử đã xác định được x[1] x[2] ... x[k–1], khi đó x[k] được chọn một trong 2 phần tử của B. Vậy có chương trình như sau:

```
Program Lietke_Day_Nhiphan;
    uses crt;
    var  n: integer;
         b: array[1..20] of 0..1;
         dem: word;
    Procedure  Khoi_tao;
       begin
         write (' Do dai cua chuoi la: n = '); readln(n);
         dem := 0;
       end;
    Procedure In_kq;
       var  i: integer;
       begin
         dem := dem+1;
         if dem mod 20 = 0 then readln;
         write ( 'Chuoi thu',dem:5,':');
         for i:=1 to n do write(b[i]:2);
         writeln;
       end;
    Procedure Thu(k:integer);
       var j: integer;
       begin
         for j:=0  to 1 do
           begin
              b[k] := j;
              if k = n then In_kq  else Thu(k+1);
           end;
       end;
    BEGIN
       clrscr;
       Khoi_tao;
       Thu(1);
```

```
        write(' Tong so co', dem:4,'chuoi nhi phan');
        readln;
    END.
```

Thí dụ 3: Bài toán xếp hậu.

Liệt kê tất cả các cách xếp n con hậu trên bàn cờ n×n sao cho chúng không ăn được nhau. Tức là không có 2 con hậu nào cùng hàng, cùng cột, cùng đường chéo chính hoặc đường chéo phụ.

Giải: Giả sử n hàng, n cột của bàn cờ được đánh số từ 1 đến n. Chúng ta sử dụng chỉ số k để chỉ hàng và chỉ số j để chỉ cột ($1 \le k, j \le n$). Quy ước x[k] = j để chỉ vị trí con hậu ở hàng k, cột j. Mỗi hàng được xếp đúng một con hậu, vấn đề còn lại là xem mỗi con hậu được xếp vào cột nào.

Dễ thấy $T_1 = N = \{1, 2, ..., n\}$.

Giả sử đã xác định được x[1], x[2], ..., x[k–1], cần phải xếp tiếp x[k] thoả mãn yêu cầu bài toán.

Việc kiểm soát theo chiều ngang là không cần thiết vì mỗi hàng được xếp đúng một con hậu. Việc kiểm soát theo chiều dọc được thực hiện nhờ dãy biến logic a[j] với quy ước a[j] bằng true nếu cột j còn trống

Bàn cờ có 2 đường chéo: đường chéo chính và các đường song song đường chéo chính có k – j vị trí ($1–n \le k–j \le n–1$) được kiểm soát nhờ dãy biến logic c[j]. Đường chéo phụ có k + j vị trí ($2 \le k+j \le 2n$) được kiểm soát nhờ dãy biến logic b[j]. Cũng như biến a[j], các biến b[j] và c[j] sẽ bằng true nếu ô tương ứng còn trống. Các biến trạng thái a[j], b[j], c[j] được khởi gán giá trị true trong thủ tục khởi tạo. Như vậy giá trị j được chấp nhận khi và chỉ khi cả 3 biến a[j], b[k+j] và c[k–j] có giá trị true, các biến này cần được gán lại false khi xếp xong quân hậu thứ j và trả lại true sau khi gọi thu(i+1) hay sau khi in kết quả.

```
Program Xep_hau;
  uses  crt;
  var   n: integer;
        x: array[1..20] of integer;
        a: array[1..20] of boolean;
        b: array[2..40] of boolean;
        c: array[–19..19] of boolean;
        dem: word;
  Procedure Khoi_tao;
     var i: integer;
     begin
        write(' Kich thuoc ban co, n = '); readln(n);
        for i:=1 to n do a[i] := true;
        for i:=2 to 2*n do b[i] := true;
        for i:=1–n to n–1 do c[i] := true;
```

```
        dem := 0;
    end;
Procedure In_kq;
    var i: integer;
    begin
        dem := dem+1;
        if dem mod 20 = 0 then readln;
        write(' Phuong an', dem:3,' :');
        for i:=1 to n do              write(x[i]:4);
        writeln;
    end;
Procedure Thu(k: integer);
    var j: integer;
    begin
        for j:=1 to n do
        if a[j] and b[k+j] and c[k–j] then
            begin
                x[k] := j;
                a[j] := false;
                b[k+j] := false;
                c[k–j] := false;
                if k=n then in_kq else thu(k+1);
                a[j] := true;
                b[k+j] := true;
                c[k–j] := true;
            end;
    end;
BEGIN
    clrscr;
    Khoi_tao;
    Thu(1);
    readln;
    write(' Tong so co:',dem:3,' phuong an');
    readln;
END.
```

6. Bài toán tồn tại

Trong các bài toán trên chúng ta đã đưa ra các thuật toán đếm các cấu hình tổ hợp. Trong các bài toán đó, sự tồn tại của các cấu hình là có thật và chúng ta chỉ việc đếm chúng. Tuy nhiên, có những bài toán ngay cả việc chỉ ra sự tồn tại của cấu hình thoả mãn điều kiện đã cho cũng là một khó khăn. Như vậy xuất hiện bài toán: Khẳng định có hay không có ít ra là một cấu hình thoả mãn các điều kiện đã cho. Các bài toán như vậy được gọi là bài toán tồn tại.

6.1. Nguyên lý Di-ric-le (Dirichlet)

a. Định lý 1. Nguyên lý lồng chim bồ câu

Nếu có nhiều hơn n đối tượng cần xếp vào n cái hộp thì có ít nhất một hộp có chứa nhiều hơn 1 đối tượng.

Nguyên lý này xuất phát từ bài toán: Có một đàn chim bồ câu bay về tổ của chúng, nếu số ngăn chuồng ít hơn số chim thì ít nhất có một ngăn có nhiều hơn 1 con chim.

Định lý được chứng minh bằng phản chứng. Giả sử không có một hộp nào có nhiều hơn một đối tượng, khi đó tổng số đối tượng được xếp trong các hộp không vượt quá số hộp, nghĩa là số đối tượng là ít hơn số hộp. Điều bày trái với giả thiết số đối tượng nhiều hơn số hộp.

Thí dụ: 1. Trong số 367 người khác nhau luôn luôn tìm được 2 người có cùng ngày sinh. (Vì 1 năm chỉ có nhiều nhất là 366 ngày)

2. Trong kỳ thi học sinh giỏi, thang điểm cho là từ 0 đến 20, và điểm thi được làm tròn đến 0,5 điểm. Hỏi phải có ít nhất bao nhiêu học sinh dự thi để chắc chắn có ít nhất 2 học sinh có cùng điểm số.

Giải: Có 41 loại điểm số khác nhau. Vậy cần phải có ít nhất 42 học sinh dự thi.

***b*. Định lý 2.** Nguyên lý Di-ric-le tổng quát

Nếu xếp n đối tượng vào k cái hộp, thì luôn luôn tìm được một hộp có không ít hơn $\left[\frac{n}{k}\right]+1$ đối tượng.

Chứng minh: Giả sử không có hộp nào trong k hộp có chứa nhiều hơn $\left[\frac{n}{k}\right]$ đối tượng. Khi đó tổng số đối tượng được chứa trong k cái hộp nhiều nhất là:

$$k\left[\frac{n}{k}\right] \quad < \quad k\,\frac{n}{k} \quad = \quad n$$

Điều này trái với giả thiết.

Thí dụ 1: Trong một tập thể 100 người có ít nhất $\left[\frac{100}{12}\right]+1=9$ người có cùng tháng sinh.

Thí dụ 2: Biển số xe máy gồm 8 ký tự: XX-XXXX-NX. Trong đó 2 ký tự đầu là mã địa danh, 4 ký tự tiếp theo là số xe và 2 ký tự cuối là mã đăng ký. Mỗi ký tự X là một trong

10 chữ số, mỗi ký tự N là 1 trong 26 chữ cái tiếng Anh. Hỏi để dăng ký 3 triệu xe cần phải có bao nhiêu mã địa danh?

Giải: Với mỗi mã địa danh có thể đăng ký được 10^4. 26. 10 = 26.10^5 xe

Vậy cần ít nhất $\left[\frac{3.10^6}{26.10^5}\right]+1=\left[\frac{30}{26}\right]+1=2$ mã địa danh.

6.2. Bài bài toán ứng dụng nguyên lý Di-ric-lê.

Việc ứng dụng nguyên lý Di-ric-lê cần phải vận dụng để lựa chọn đối tượng và hộp cho phù hợp.

Bài toán 1. Trong số n người dự hội nghị, mỗi người quen ít nhất một người khác. Hãy chứng tỏ rằng trong phòng họp có ít nhất 2 người có số người quen như nhau.

Giải: Chia số người dự họp thành các nhóm, mọi người trong cùng một nhóm có số người quen là bằng nhau. Như vậy có nhiều nhất là n – 1 nhóm. Do có n người xếp vào (n – 1) nhóm nên phải có ít nhất 1 nhóm có 2 người, nghĩa là ít nhất có 2 người có số người quen bằng nhau.

Có thể phát biểu bài toán này dưới dạng hình học: Có n điểm phân biệt, biết rằng mỗi điểm được nối với ít nhất một điểm khác bằng một đoạn thẳng và không có điểm nào được nối với mọi điểm còn lại. Khi đó sẽ có ít nhất 2 điểm tại đó có cùng số cạnh xuất phát để nối với các điểm khác.

Bài toán 2. Trong một tháng có 30 ngày, một đội bóng chuyền phải thi đấu ít nhất một ngày một trận. Tổng số trận mà đội phải chơi là 45 trận. Hãy chứng tỏ rằng có một số ngày liên tiếp trong tháng đó đội phải chơi đúng 14 trận.

Giải: Gọi a_i là số trận đội phải chơi từ ngày thứ nhất đến ngày thứ i. Ta thấy a_1, a_2, ..., a_{30} là dãy các số nguyên dương tăng ($a_i < a_{i+1}$) và $1 \le a_i \le 45$. Từ đó suy ra dãy số a_1+14, a_2+14,..., $a_{30}+14$ cũng là dãy số nguyên dương, không giảm và $15 \le a_i+14 \le 59$.

Tất cả có 60 số nguyên dương a_1, a_2, ..., a_{30} , a_i+14, ... , $a_{30}+14$ luôn luôn nhỏ hơn hoặc bằng 59. Theo nguyên lý Di-ric-lê phải có ít nhất 2 trong số 60 số này bằng nhau. Do $a_i \ne a_j$, $a_i+14 \ne a_j+14$, nên phải tìm được 2 chỉ số k và n sao cho $a_n = a_k+14$.

Điều này có nghĩa là từ ngày k+1 đến ngày n đội phải chơi đúng 14 trận.

Bài toán *3*. Trong mặt phẳng cho 6 điểm nối với nhau từng đôi bởi các cung màu đỏ hoặc màu xanh. Chứng minh rằng luôn luôn tìm được 3 điểm sao cho các cung nối giữa chúng có cùng màu (người ta nói rằng 3 điểm đó tạo thành một tam giác xanh hay tam giác đỏ).

Giải: Chọn điểm P tuỳ ý trong 6 điểm. Từ P có 5 cung nối với 5 điểm còn lại. Theo nguyên lý Di-ric-lê phải có 3 trong số 5 cung đó cùng màu, chẳng hạn màu xanh. Giả sử đó là các cung PA, PB, PC, nếu một hoặc hai trong 3 cung AB, AC, BC có màu xanh thì nó cùng với 2 trong 3 cung PA, PB, PC tạo thành tam giác xanh. Trong trường hợp ngược lại được tam giác đỏ.

6.3. Phương pháp phản chứng.

Một trong các phương pháp khác để giải bài toán tồn tại là dùng lập luận phản chứng. Thực ra việc chứng minh nguyên lý Di-ríc-lê là đã dùng phản chứng, sau đây chúng ta xét thêm một số thí dụ khác.

Thí dụ 1. Cho 7 đoạn thẳng có độ dài lớn hơn 10 và nhỏ hơn 100. Chứng minh rằng luôn luôn tìm được 3 đoạn để có thể ghép thành một tam giác.

Giải: Gọi độ dài các đoạn thẳng theo thứ tự tăng dần:

$$a_1, a_2, ..., a_7; \quad a_i < a_{i+1}, \ 10 < a_i < 100, \ \forall i.$$

Để chứng minh có 3 đoạn ghép thành một tam giác thì cần và đủ là trong 3 số liên tiếp của dãy số đã xếp ở trên có tổng 2 đoạn đầu lớn hơn đoạn thứ ba. Giả sử điều đó không xảy ra, nghĩa là:

$$a_1 + a_2 \le a_3 ;$$
$$a_2 + a_3 \le a_4 ;$$
$$a_3 + a_4 \le a_5 ;$$
$$a_4 + a_5 \le a_6 ;$$
$$a_5 + a_6 \le a_7 ;$$

Vì $a_1 > 10, a_2 > 10 \Rightarrow a_3 > 20 \Rightarrow a_4 > 30 \Rightarrow a_5 > 50 \Rightarrow a_6 > 80 \Rightarrow a_7 > 130.$ Điều này trái với giả thiết của bài toán. Vậy bài toán được chứng minh.

Thí dụ 2. Các đỉnh của một thập giác lồi được gán các số nguyên 0, 1, 2, ..., 9 một cách ngẫu nhiên; mỗi đỉnh là một số khác nhau. Chứng minh rằng luôn luôn tìm được 3 đỉnh liên tiếp sao cho tổng các số được gán lớn hơn 13.

Giải: Gọi $x_1, x_2, ..., x_{10}$ là các số được gán cho các đỉnh thứ nhất, thứ hai, ..., thứ mười của thập giác. Giả sử không tìm được 3 đỉnh liên tiếp nào có tổng các số gán lớn hơn 13, nghĩa là:

$$k_1 = x_1 + x_2 + x_3 \le 13 ;$$
$$k_2 = x_2 + x_3 + x_4 \le 13 ;$$
$$. \ . \ . \ . \ . \ . \ . \ . \ . \ . \ .$$
$$k_9 = x_9 + x_{10} + x_1 \le 13 ;$$
$$k_{10} = x_{10} + x_1 + x_2 \le 13 ;$$

Suy ra: $k_1 + k_2 + ... + k_{10} \le 130.$

Mặt khác: $k_1 + k_2 + ... + k_{10} = 3(x_1 + x_2 + ... + x_{10}) = 3(0 + 1 + ... + 9) = 135$

Từ đó có: $135 = k_1 + k_2 + ... + k_{10} \le 130.$ Vô lý.

Vậy phải tồn tại 3 đỉnh liên tiếp để tổng các số được gán là lớn hơn 13.

BÀI TẬP CHƯƠNG 2

Tổ hợp, Chỉnh hợp, Hoán vị

2.1. Một nhóm sinh viên gồm n nam và n nữ. Có bao nhiêu cách xếp thành một hàng ngang sao cho nam nữ đứng xen kẽ nhau?

2.2. Một tập hợp có 10 phần tử có bao nhiêu tập con với số phần tử lẻ?

2.3. Một tổ có 13 người.

a) Có bao nhiêu cách chọn 10 người đi làm một công việc định sẵn? Biết rằng mọi người đều làm được công việc đó.

b) Có bao nhiêu cách chọn 10 người để làm 10 việc khác nhau? Biết rằng mọi người đều làm được mọi việc như nhau.

c) Có bao nhiêu cách chọn 10 người sao cho có ít nhất 1 nữ? Biết trong tổ có 3 nữ.

2.4. Có bao nhiêu xâu nhị phân có độ dài 10 nếu trong xâu có:

a) Đúng 3 số 0.

b) Ít nhất 3 số 1.

c) Số các số 0 bằng số các số 1.

d) Có ít nhất 3 số 0 và ít nhất 3 số 1.

2.5. Có bao nhiêu biển đăng ký xe. Biết mỗi biển gồm 3 chữ cái tiếng Anh khác nhau và tiếp đến là 3 chữ số khác nhau?

2.6. Chứng minh rằng (Đẳng thức Van-dec-non)

a) $C^r_{m+n} = \sum_{k=0}^{r} C^{r-k}_m . C^k_n$; b) $\sum_{k=0}^{r} C^k_{n+k} = C^r_{n+r+1}$

Gợi ý: a) Xét 2 tập hợp có tương ứng m, n phần tử. Lấy từ 2 tập hợp đó ra r phần tử theo hai cách: Cách I là trộn lẫn 2 tập hợp với nhau; Cách II là lấy ở tập hợp này k phần tử, còn tập hợp kia r – k phần tử (k = 0, 1, 2, ..., r)

b) Chứng minh bằng quy nạp theo r , xuất phát từ $C^0_n = C^0_{n+1} = 1$.

2.7. Có bao nhiêu cách phân công 3 công việc cho 5 người làm, nếu 1 người có thể làm được cả 3 việc.

2.8. Một "con lợn tiết kiệm" có 50 đồng xu. Có thể có bao nhiêu tổ hợp khác nhau các đồng xu: 200đ, 500đ, 1000đ, 2000đ, 5000đ trong đó?

2.9. Có 100 viên bi hoàn toàn giống nhau. Có bao nhiêu cách để xếp các viên bi vào 3 cái hộp? biết mọi cái hộp đều có thể chứa đến 100 viên bi.

2.10. Phương trình: $x_1 + x_2 + x_3 + x_4 = 17$ có bao nhiêu nghiệm nguyên không âm?

2.11. Phương trình: $x_1 + x_2 + x_3 + x_4 + x_5 = 21$ có bao nhiêu nghiệm nguyên không âm sao cho:

a) $x_1 \geq 1$; b) $x_i \geq 2$ với $i = 1, 2, 3, 4, 5$

c) $0 \leq x_1 \leq 10$; d) $0 \leq x_1 \leq 3$ và $1 \leq x_2 \leq 4$ và $x_3 \geq 15$

2.12. Có bao nhiêu xâu có đủ các chữ cái trong từ MISSISSIPPI ?

2.13. Có bao nhiêu xâu khác nhau có 7 hoặc nhiều hơn các ký tự có thể lập được từ các chữ cái của từ NONGNGHIEP ?

2.14. Có bao nhiêu xâu nhị phân khác nhau gồm 6 chữ số 0 và 8 chữ số 1?

2.15. Một đề thi có 10 câu hỏi. Có bao nhiêu cách gán điểm cho các câu hỏi nếu tổng số điểm là 100 và mỗi câu được ít nhất 5 điểm?

2.16. Bất phương trình: $x_1 + x_2 + x_3 \leq 11$ có bao nhiêu nghiệm nguyên không âm?

Gợi ý: Bổ sung thêm biến phụ x_4 sao cho: $x_1 + x_2 + x_3 + x_4 = 11$.

2.17. Trong không gian Oxyz một con bọ di chuyển bằng cách nhảy từng bước, mỗi bước 1 đơn vị theo hướng hoặc truc Ox, hoặc trục Oy, hoặc trục Oz và không được nhảy giật lùi. Tính số cách để con bọ đó có thể di chuyển từ gốc tọa độ (0, 0, 0) đến điểm (4, 3, 5).

Nguyên lý bù trừ

2.18. Hãy tìm số phần tử của $A \cup B \cup C$ nếu mỗi tập A, B, C đều có 100 phần tử và nếu các tập hợp A, B, C thỏa mãn:

a) Đôi một rời nhau.

b) Có 50 phần tử chung của mỗi cặp tập và không có phần tử nào chung của cả 3 tập hợp.

c) Có 50 phần tử chung của mỗi cặp tập và 25 có phần tử chung của cả ba tập hợp.

2.19. Kiểm tra 270 sinh viên mới tốt nghiệp của một trường đại học. Kết quả cho thấy có 64 sinh viên thành thạo tiếng Anh, 94 sinh viên thành thạo tiếng Pháp, 108 sinh viên thành thạo vi tính văn phòng, 26 sinh viên vừa thành thạo tiếng Pháp vừa thành thạo vi tính, 28 sinh viên vừa thành thạo tiếng Anh vừa thành thạo vi tính, 22 sinh viên vừa thành thạo tiếng Pháp vừa thành thạo tiếng Anh và 11 sinh viên thành thạo cả 3 thứ (tiếng Anh, tiếng Pháp và vi tính). Hỏi trong số 270 sinh viên đó có bao nhiêu người không biết cả 3 thứ kể trên?

2.20. Có bao nhiêu xâu nhị phân có độ dài 8 và không chứa 6 số 0 liên tiếp?

2.21. Có bao nhiêu số nguyên dương không vượt quá 100 và hoặc là số lẻ hoặc là số chính phương.

2.22. Có bao nhiêu hoán vị của 10 chữ số nguyên dương đầu tiên hoặc là bắt đầu bằng ba số 987, hoặc là chứa các số 45 ở vị trí thứ 5 và thứ 6, hoặc là kết thúc bằng 3 số 123.

Giải các hệ thức truy hồi.

2.23. Số lượng một loài sinh vật sẽ tăng gấp ba sau mỗi giờ.

a) Lập công thức truy hồi tính số lượng vi sinh vật đó sau n giờ.

b) Nếu lúc đầu có 100 con thì sau 10 giờ sẽ có bao nhiêu con?

2.24. a) Tìm hệ thức truy hồi tính số cách đi lên n bậc cầu thang của một người. Biết người đó có thể bước mỗi lần một hoặc hai bậc.

b) Có bao nhiêu cách để người đó đi lên một cầu thang có 8 bậc?

2.25. Chứng minh rằng các số Fibonacci thoả mãn hệ thức truy hồi:

$$f_n = 5f_{n-4} + 3f_{n-5}, \quad n \geq 5; \quad f_0 = 0,\ f_1 = 1,\ f_2 = 1,\ f_3 = 2,\ f_4 = 3.$$

Dùng hệ thức này chứng minh rằng f_{5n} chia hết cho 5 với $n = 1, 2, 3, ...$

2.26. Tìm hệ thức truy hồi tính số xâu nhị phân có độ dài n và:

a) Không có 2 số 0 liên tiếp.

b) Có 2 số 0 liên tiếp.

Tính cụ thể số xâu nhị phân thoả mãn điều kiện a hoặc b nếu n = 5

2.27. Một người đầu tư 100 triệu đồng vào một cơ sở sản xuất. Sau 1 năm người đó được hai khoản lãi: Khoản thứ nhất là 20% tổng số tiền trong năm cuối; Khoản thứ hai là 13% của tổng số tiền có trong năm trước đó.

a) Tìm công thức truy hồi tính $\{P_n\}$, trong đó P_n là tổng số tiền có vào cuối năm thứ n.

b) Tìm công thức trực tiếp tính P_n.

Giả thiết rằng người đó không rút bất cứ khoản tiền nào trong cả n năm đó.

2.28. Với các tấm lát kích thước 1×2 và 2×2 có thể lát một bảng 2×n bằng bao nhiêu cách?

2.29. Giải các hệ thức truy hồi sau:

a) $a_n = 5a_{n-1} - 6a_{n-2}, \quad n \geq 2; \quad a_0 = 1,\ a_1 = 0.$

b) $a_n = 4a_{n-1} - 4a_{n-2}, \quad n \geq 2; \quad a_0 = 6,\ a_1 = 8.$

c) $a_n = \frac{a_{n-2}}{4}, \quad n \geq 2; \quad a_0 = 1,\ a_1 = 0.$

d) $a_n = 7a_{n-2} + 6a_{n-3}, \quad a_0 = 9,\ a_1 = 10,\ a_2 = 32.$

e) $a_n = 2a_{n-1} + 5a_{n-2} - 6a_{n-3}, \ a_0 = 7,\ a_1 = -4,\ a_2 = 8.$

2.30. Xét ma trận $A = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}$

a) Chứng minh rằng: $A^n = \begin{pmatrix} F_{n-1} & F_n \\ F_n & F_{n+1} \end{pmatrix}$, trong đó F_n là số hạng thứ n của dãy Fibonacci.

b) Tính $\det(A^n)$. từ đó suy ra công thức: $F_{n+1} F_{n-1} - (F_n)^2 = (-1)^n$.

Bài toán tồn tại.

2.31. Cho n là một số nguyên dương. Chứng minh rằng trong mọi tập hợp có n số nguyên dương liên tiếp có đúng một số chia hết cho n.

2.32. Cho (x_i, y_i, z_i), i = 1, 2, ..., 9 là một tập hợp gồm 9 điểm khác nhau có toạ độ nguyên trong không gian Oxyz. Chứng minh rằng các điểm giữa của đường nối của 2 trong 9 điểm đó có ít nhất một điểm có toạ độ nguyên.

2.33. Trong một hộp kín có 10 viên bi màu đỏ và 10 bi màu xanh. Một người lấy ngẫu nhiên các viên bi trong bóng tối. Hỏi người đó cần lấy ít nhất bao nhiêu viên bi để chắc chắn lấy được ít nhất 2 viên bi cùng màu.

2.34. Chứng tỏ rằng trong 5 số chọn từ 8 số nguyên dương đầu tiên chắc chắn có một cặp có tổng bằng 9.

2.35. a) Trong phòng có 6 máy tính. Mỗi máy được nối trực tiếp hoặc không nối với các máy khác. Hãy chứng tỏ rằng có ít nhất hai máy có cùng số kết nối với các máy khác hoặc có ít nhất hai máy không được nối với máy khác.

b) Một mạng máy tính có 6 máy. Mỗi máy được nối trực tiếp với ít nhất một máy khác. Hãy chứng tỏ rằng có ít nhất hai máy có cùng số kết nối với các máy khác.

2.36. Có 12 cầu thủ bóng đá mặc áo mang số từ 1 đến 12 đứng thành một vòng tròn giữa sân. Chứng minh rằng luôn luôn tìm được 3 người đứng liên tiếp nhau có tổng các số trên áo ít nhất bằng 20.

ĐÁP SỐ

2.1. $2(n!)^2$.

2.2. 2^9.

2.3. a) 286; b) 1 716; c) 1 037 836 800.

2.4. a) 120; b) 968; c) 252; d) 912.

2.5. 12 232 000.

2.7. 60.

2.8. 316 251.

2.9. 5 151.

2.10. 1 140.

2.11. a) 10 626; b) 1 365; c) 11 649; d) 106.

2.12. 34 650.

2.13. 838 320.

2.14. 3 003.

2.15. 12 565 671 261.

2.16. 364.

2.17. 27 720.

2.18. a) 300; b) 150; c) 175.

2.19. 69.

2.20. 248.

2.21. 55.

2.22. 50 138.

2.23. a) $a_n = 3a_{n-1}$, $a_0 = 100$; b) 5 904 900.

2.24. a) $a_n = a_{n-1} + a_{n-2}$, $a_0 = 1, a_1 = 1$; b) 34.

2.26. a) $a_n = a_{n-1} + a_{n-2}$, $a_1 = 2$, $a_2 = 3$; $\Rightarrow$ $a_5 = 13$;

b) $a_n = a_{n-1} + a_{n-2} + 2^{n-2}$, $a_0 = 0$, $a_2 = 1$; $\Rightarrow$ $a_5 = 19$.

2.27. a) $P_n = 1{,}2P_{n-1} + 0{,}13P_{n-2}$, $P_0 = 100$; $P_1 = 120$.

b) $P_n = \frac{100}{14}\left(13.(1{,}3)^3 + (-0{,}1)^n\right) \approx 92{,}857(1{,}3)^n + 7{,}143(-0{,}1)^n$

2.28. $a_n = a_{n-1} + 2a_{n-2}$, $a_1 = 1$, $a_2 = 3$; $a_n = \frac{1}{3}\left(2^{n+1} + (-1)^n\right)$.

2.29. a) $a_n = 3.2^n - 2\,.\,3^n$; b) $a_n = (6 - 2n)2^n$; c) $a_n = \left(\frac{1}{2}\right)^{n+1} - \left(-\frac{1}{2}\right)^{n+1}$

d) $a_n = 8(-1)^n - 3(-2)^n + 4.\,3^n$; d) $a_n = 5 + 3(-2)^n - 3^n$.

2.33. 11.

CÂU HỎI ÔN TẬP CHƯƠNG 2

1. Phát biểu các nguyên lý cơ bản của phép đếm: Nguyên lý cộng; Nguyên lý nhân; Cho các thí dụ minh hoạ.

2. Nêu sự khác nhau giữa tổ hợp và chỉnh hợp chập k từ n phần tử. Mối quan hệ giữa tổ hợp và chỉnh hợp? Mối quan hệ giữa chỉnh hợp và hoán vị?

3. Hãy giải thích các tìm công thức tính số cách chọn có lặp k phần tử từ n phần tử đã cho? Giải thích tại sao có thể dùng tổ lặp để tính số nghiệm nguyên không âm của phương trình $x_1 + x_2 + \ldots + x_n = k$, trong đó k là số nguyên dương.

4. Hãy lập công thức tính số hoán vị của các tập hợp có các phần tử giống nhau và số cách phân chia một tập hợp có n phần tử thành k tập con. Cho thí dụ minh hoạ.

5. Phát biểu nguyên lý bù trừ. Ứng dụng nguyên lý bù trừ tìm số nghiệm nguyên không âm của phương trình $x_1 + x_2 + x_3 + x_4 = 20$ thoả mãn $x_1 < 7$, $x_2 < 5$ và $x_3 < 4$.

7. Trình bày các phương pháp để giải các hệ thức truy hồi. Cho thí dụ minh hoạ

8. Phát biểu nguyên lý Di-ric-lê. Ứng dụng nguyên lý Di-ric-lê chứng minh rằng trong bất kỳ 91 số nguyên nào cũng có ít nhất 10 số có cùng chữ số cuối cùng.

9. Trình bày thuật toán quay lui để giải bài toán liệt kê. Cho thí dụ minh hoạ.

10. Cho một vài thí dụ giải bài toán tồn tại bằng phản chứng.

CHƯƠNG 3.

CÁC KHÁI NIỆM CƠ BẢN VỀ ĐỒ THỊ

1. Các định nghĩa về đồ thị và biểu diễn hình học của đồ thị
   - 1.1. Các định nghĩa và các loại đồ thị
   - 1.2. Bậc của đỉnh của đồ thị
   - 1.3. Một số dạng đồ thị đặc biệt
   - 1.4. Một vài ứng dụng của các đồ thị đặc biệt
2. Biểu diễn đồ thị bằng đại số
   - 2.1. Danh sách kề
   - 2.2. Danh sách cạnh
   - 2.3. Ma trận kề
   - 2.4. Ma trận liên thuộc
3. Sự đẳng cấu của các đồ thị
4. Tính liên thông trong đồ thị
   - 4.1. Đường đi, chu trình
   - 4.2. Đồ thị liên thông
5. Số ổn định trong, số ổn định ngoài và nhân của đồ thị.
   - 5.1. Số ổn định trong
   - 5.2. Số ổn định ngoài
   - 5.3. Nhân của đồ thị
6. Sắc số của đồ thị
   - 6.1. Định nghĩa
   - 6.2. Một số định lý về sắc số
   - 6.3. Vài thí dụ ứng dụng sắc số

Lý thuyết đồ thị ra đời từ thế kỷ 18 bởi nhà toán học Leonhard Euler (1707 – 1783) gốc Thụy sĩ nhưng chủ yếu sống và làm việc ở Nga và Đức. Nó được ứng dụng để giải các bài toán trong nhiều lĩnh vực khác nhau, chẳng hạn lý thuyết mạng điện, mạng giao thông, cấu trúc hoá học, …, và đặc biệt trong lý thuyết thông tin và điều khiển học.

1. Các định nghĩa về đồ thị và biểu diễn hình học của đồ thị

Đồ thị là một cấu trúc rời rạc gồm các đỉnh và các cạnh nối các đỉnh. Người ta phân loại đồ thị theo đặc tính và số các cạnh nối giữa các đỉnh.

1.1. Các định nghĩa và các loại đồ thị

a. Định nghĩa đồ thị

Định nghĩa: Cho X là một tập hợp không rỗng các đối tượng nào đó và $U \subseteq X^2$. Bộ G = (X, U) được gọi là một đồ thị. Nếu X là tập hữu hạn thì gọi là đồ thị hữu hạn. Nếu X là tập vô hạn thì gọi là đồ thị vô hạn.

Giáo trình này chỉ xét đồ thị hữu hạn và gọi tắt là đồ thị.

Mỗi phần tử $x \in X$ được gọi là một đỉnh của đồ thị.Tập X gọi là tập các đỉnh của đồ thị.

Nếu mỗi phần tử $u = (x, y) \in U$ là không phân biệt thứ tự thì đồ thị được gọi là **đồ thị vô hướng**. Mỗi phần tử $u \in U$ gọi là một cạnh của đồ thị. Tập U gọi là tập các cạnh của đồ thị.

Nếu mọi phần tử $u = (x, y) \in U$ của đồ thị được sắp thứ tự (x trước, y sau) thì đồ thị được gọi là **đồ thị có hướng**. Mỗi phần tử $u \in U$ của đồ thị có hướng được gọi là một cung hay một cạnh có hướng. Tập U được gọi là tập các cung của đồ thị.

Đồ thị vừa có cạnh vừa có cung được gọi là **đồ thị hỗn hợp.**

Nếu cạnh (cung) u = (x, x) thì u gọi là một khuyên và đỉnh x gọi là đỉnh có khuyên.

Nếu u = (x, y) thì cạnh (cung) u gọi là kề hay thuộc đỉnh x và đỉnh y, đồng thời cũng nói đỉnh x kề hay thuộc cạnh (cung) u, đỉnh y kề hay thuộc cạnh (cung) u.

Hai đỉnh x, y được gọi là kề nhau nếu chúng thuộc cùng một cạnh (cung).

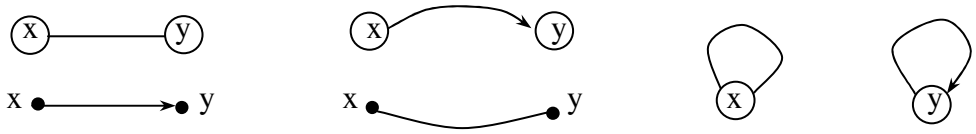
Hai cạnh (cung) u, v được gọi là kề nhau nếu chúng có chung đỉnh.

Một cách trực quan nhất là biểu diễn đồ thị bằng hình vẽ:

Cho đồ thị G = (X, U). Tập đỉnh X được biểu diễn bằng các điểm trong mặt phẳng hoặc trong không gian. Mỗi đỉnh có thể đặt một tên gọi tương ứng, chẳng hạn x, x_1, x_2, …hay y, y_1, y_2, … hay a, b, c, …

Tập cạnh U được biểu diễn như sau:

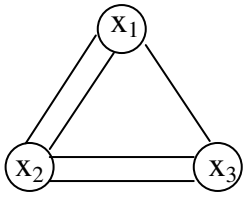
- Nếu $u = (x, y) \in U$ là cạnh vô hướng, thì 2 đỉnh x, y được nối với nhau bằng một đoạn thẳng hay cung cong.
- Nếu $u = (x, y) \in U$ là cạnh có hướng, thì 2 đỉnh x, y được nối với nhau bằng một đoạn thẳng hay cung cong có mũi tên chỉ hướng từ đỉnh đầu x đến đỉnh cuối y.
- Nếu u = (x, x) là một khuyên thì a được biểu diễn bằng một vòng xuyến từ x đến chính nó.



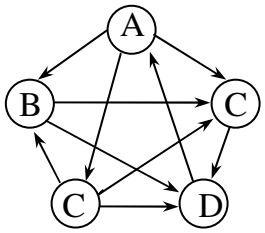
Hình 1. Biểu diễn đỉnh, cạnh, cung và khuyên của đồ thị

Thí dụ 1: Giữa ba thành phố Hà Nội, Hải Phòng và Hạ Long có các đường giao thông như sau: Hà Nội - Hải Phòng có đường bộ và đường sắt; Hải Phòng - Hạ Long có đường bộ và đường thuỷ; Hà Nội - Hạ Long có đường bộ.

Gọi x_1 là Hà Nội, x_2 là Hải Phòng, x_3 là Hạ Long là các đỉnh và biểu diễn mỗi đường giao thông bằng một cạnh được đồ thị vô hướng ở hình 2.



Hình 2 Mạng giao thông



Hình 3. Thi đấu vòng tròn

Thí dụ 2: Mô hình đồ thị thi đấu vòng tròn. Thi đấu vòng tròn giữa các đội bóng là mỗi đội đều thi đấu với tất cả các đội còn lại. Giả sử có 5 đội thi đấu vòng tròn, khi đó có thể biểu diễn kết quả thi đấu theo sơ đồ như hình 3. Trong sơ đồ đó A, B, C, D, E là tên các đội bóng và mũi tên chỉ đội ở gốc mũi tên thắng đội ở ngọn mũi tên. Đó là một đồ thị có hướng gồm 5 đỉnh

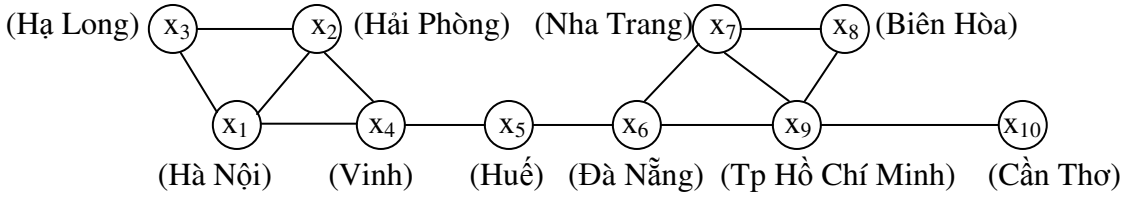
b. Phân loại đồ thị

Ngoài cách phân chia thành đồ thị vô hướng và đồ thị có hướng, người ta còn phân loại đồ thị dựa trên số cạnh (cung) kề với các cặp đỉnh. Để tiện việc phân biệt các loại đồ thị, chúng ta nghiên cứu chúng qua các thí dụ về mạng máy tính.

Giả sử các máy chủ của một công ty cung cấp dịch vụ internet (ISP) được đặt ở các thành phố lớn: Hà Nội (x_1), Hải Phòng (x_2), Hạ Long (x_3), Vinh (x_4), Huế (x_4), Đà Nẵng (x_6), Nha Trang (x_7), Biên Hoà (x_8), Thành phố Hồ Chí Minh (x_9), và Cần Thơ (x_{10}). Các máy này được nối với nhau bằng các đường cáp quang theo sơ đồ ở hình 4, trong đó các đường cáp này truyền dữ liệu theo cả hai chiều. Mạng máy tính này có thể mô hình bằng một đồ thị gọi là **đơn đồ thị** có 10 đỉnh (hình 4)

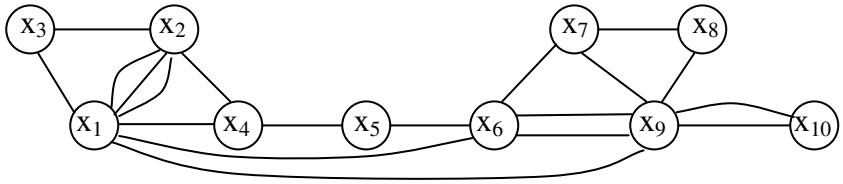
Định nghĩa 1: Một ***đơn đồ thị*** là một đồ thị mà mọi cặp đỉnh được nối với nhau nhiều nhất là một cạnh (cung) và không có khuyên.

Hình 3 là một thí dụ về đơn đồ thị có hướng. Hình 4 là một đơn đồ thị vô hướng.



Hình 4. Mạng máy tính đơn

Cũng trong mạng máy tính nói trên, do yêu cầu của việc truyền dữ liệu, có thể có một số máy chủ được nối với nhau bằng hơn 1 đường cáp quang. Chẳng hạn giữa Hà nội (x_1) và Hải phòng (x_2) có 3 đường truyền, Hà nội và Đà nẵng (x_6), Hà nội và thành phố Hồ Chí Minh có thêm đường truyền trực tiếp, …(Hình 5). Đó là một mô hình **đa đồ thị.**

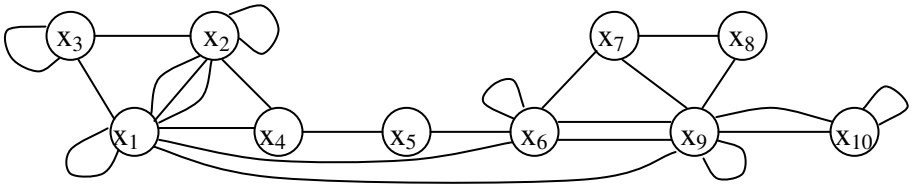


Hình 5. Mạng máy tính đa kênh

Định nghĩa 2: Một **đa đồ thị** là một đồ thị trong đó có ít nhất một cặp đỉnh được nối với nhau nhiều hơn một cạnh (cung) và không có khuyên. Các cặp đỉnh có nhiều cạnh thì các cạnh tương ứng gọi là các cạnh bội hay các cạnh song song.

Hình 2 và hình 5 là các thí dụ về đa đồ thị.

Có thể trong mạng máy tính có những máy có đường truyền với chính nó, khi đó tại đỉnh tương ứng của mô hình đồ thị sẽ tồn tại một khuyên. Những đồ thị như vậy được gọi là **giả đồ thị.** Giả đồ thị là dạng đồ thị tổng quát nhất.



Hình 6. Mạng máy tính có các đường truyền nội bộ

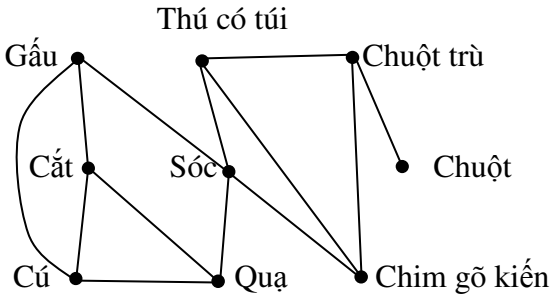
Định nghĩa 3: Giả đồ thị là đồ thị (đơn hoặc đa đồ thị) có khuyên.

c. Một vài mô hình đồ thị khác

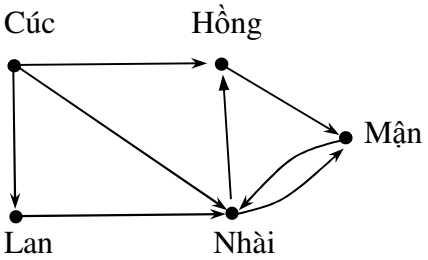
Thí dụ 1: Đồ thị cạnh tranh trong sinh thái học là một mô hình đồ thị biểu hiện sự cạnh tranh của các loài trong một hệ sinh thái nào đó. Mỗi loài vật được biểu diễn bằng một đỉnh. Mỗi cạnh vô hướng nối hai đỉnh nếu hai loài được biểu diễn bằng hai đỉnh này có sự cạnh tranh với nhau (chẳng hạn chúng cùng chung nguồn thức ăn hoặc cùng chung chỗ ở).

Hình 7 là mô hình hệ sinh thái rừng. Ta thấy Gấu và Cú, Sóc và Quạ, … cạnh tranh nhau; còn Gấu và Chim gõ kiến, Sóc và Cú, … là không cạnh tranh nhau.

Thí dụ 2: Đồ thị ảnh hưởng. Khi nghiên cứu tính cách của một nhóm người thì thấy một số người này có thể ảnh hưởng lên suy nghĩ của một số người khác. Có thể dùng một đồ thị có hướng để biểu diễn điều đó, trong đó mỗi người là một đỉnh của đồ thị và nếu người A ảnh hưởng đến người B thì có một cung đi từ A đến B. Hình 8 cho thấy, chẳng hạn Cúc có ảnh hưởng đến Hồng, Lan và Nhài … Không có ai ảnh hưởng tới Cúc. Nhài và Mận ảnh hưởng lẫn nhau.



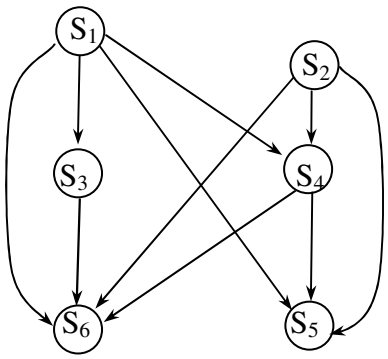
Hình 7. Đồ thị lấn tổ



Hình 8. Đồ thị ảnh hưởng

Thí dụ 3: Đồ thị ưu tiên trước sau. Các chương trình máy tính có thể chạy nhanh hơn bằng cách thực hiện đồng thời một số câu lệnh nào đó. Tuy nhiên không được thực hiện câu lệnh A nếu A đòi hỏi kết quả của câu lệnh B khi B chưa được thực hiện. Sự phụ thuộc giữa các câu lệnh có thể được biểu diễn bằng một đồ thị có hướng. Trong đồ thị đó các đỉnh là các câu lệnh và có cung mối từ đỉnh B đến đỉnh A nếu câu lệnh A chưa thực hiện được khi câu lệnh B chưa được thực hiện. Đồ thị này gọi là đồ thị ưu tiên trước sau. Hình 9 thể hiện một đoạn chương trình và đồ thị tương ứng với đoạn chương trình đó. Ta thấy câu lệnh S_5 không thể thực hiện trước khi các câu lệnh S_1, S_2 và S_4 được thực hiện. Cũng vậy, S_6 chưa thể thực hiện nếu chưa thực hiện S_1, S_2, S_3 và S_4.

S_1 a := 0;
S_2 b := 1
S_3 c := a+1
S_4 d := b+a
S_5 e := d+1
S_6 f := c+d



Hình 9. Đồ thị ưu tiên trước sau

1.2. Bậc của đỉnh của đồ thị

a. Định nghĩa

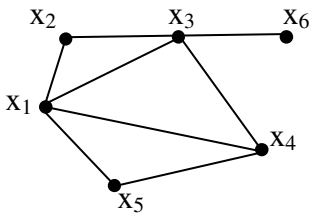
Số cạnh thuộc đỉnh x của đồ thị vô hướng gọi là bậc của đỉnh x, ký hiệu là deg(x).

Số cung đi ra từ đỉnh x của đồ thị có hướng gọi là bán bậc ra của đỉnh x, ký hiệu là $deg^+(x)$.

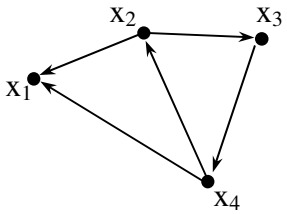
Số cung đi vào đỉnh x của đồ thị có hướng gọi là bán bậc vào của đỉnh x, ký hiệu là $deg^-(x)$.

Các đỉnh có bậc (hoặc tổng bậc đối với đồ thị có hướng) bằng 0 gọi là đỉnh **biệt lập** hay đỉnh **cô lập**; còn nếu bằng 1 gọi là đỉnh **treo**, khi đó các cạnh (cung) tương ứng gọi là cạnh (cung) treo.

Thí dụ Xem hình 10 và hình 11



Hình 10



Hình 11

$\deg(x_1) = 4$; $\deg(x_2) = 2$
$\deg(x_3) = 4$; $\deg(x_4) = 3$
$\deg(x_5) = 2$; $\deg(x_6) = 1$

$\deg^+(x_1) = 0$; $\deg^-(x_1) = 2$
$\deg^+(x_2) = 2$; $\deg^-(x_2) = 1$
$\deg^+(x_3) = 1$; $\deg^-(x_3) = 1$
$\deg^+(x_4) = 2$; $\deg^-(x_4) = 1$

b. Tính chất

Tính chất 1: Trong đồ thị vô hướng G = (X, U) tổng số bậc của các đỉnh là gấp đôi số cạnh:

$$\sum_{x\in X}\deg(x) = 2N(U);$$

còn trong đồ thị có hướng G = (X, U) thì: $\sum_{x\in X}\deg^-(x) = \sum_{x\in X}\deg^+(x) = N(U)$

Chứng minh: Hiển nhiên.

Thí dụ: Có bao nhiêu cạnh trong một đồ thị có 10 đỉnh, mỗi đỉnh đều có bậc bằng 6?

Giải: Tổng số bậc của đồ thị là 10.6 = 60. Do đó 2N(U) = 60, suy ra số cạnh của đồ thị là N(U) = 30.

Tính chất 2: Trong đồ thị vô hướng, số đỉnh bậc lẻ là một số chẵn (đỉnh bậc lẻ là đỉnh có bậc là số lẻ).

Chứng minh: Gọi L và C tương ứng là tập các đỉnh bậc lẻ và tập các đỉnh bậc chẵn, khi đó $\sum_{x\in C}\deg(x)$ là một số chẵn vì nó là tổng các số chẵn. Mặt khác theo tính chất 1, ta có:

$$\sum_{x\in C}\deg(x) + \sum_{x\in L}\deg(x) = \sum_{x\in X}\deg(x) = 2N(U) \text{ là một số chẵn}$$

Suy ra $\sum_{x\in L}\deg(x)$ là số chẵn, do mọi số hạng của $\sum_{x\in L}\deg(x)$ là lẻ nên tổng này phải có một số chẵn các số hạng, nghĩa là N(L) là một số chẵn.

Tính chất 3: Trong một đơn đồ thị vô hướng luôn luôn tồn tại ít nhất hai đỉnh cùng bậc.

Chứng minh: Giả sử đồ thị G = (X, U) có n đỉnh (N(X) = n, n ≥ 2).

Nếu đồ thị có đỉnh bậc 0, khi đó đồ thị không có đỉnh nào kề với tất cả các đỉnh còn lại. Do đó mỗi đỉnh của đồ thị có bậc là một trong n – 1 số nguyên 0, 1, 2, …, n – 2.

Nếu đồ thị có đỉnh bậc n – 1 thì đồ thị không có đỉnh nào có bậc 0. Do đó bậc của mỗi đỉnh của đồ thị là một trong n – 1 số nguyên 1, 2, …, n – 1.

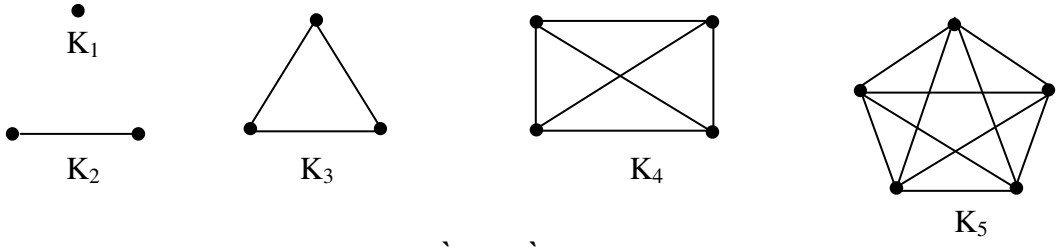
Vậy, trong mọi trường hợp, với n đỉnh chỉ có n – 1 loại bậc đỉnh. Bởi vậy, theo nguyên lý Di-ric-lê phải có ít nhất hai đỉnh có cùng bậc.

1.3. Một số dạng đồ thị đặc biệt

a. Đồ thị đầy đủ: Đồ thị đầy đủ n đỉnh, ký hiệu K_n, là đồ thị mà mọi cặp đỉnh bất kỳ luôn luôn kề nhau.

Dễ thấy đồ thị đầy đủ K_n có $\frac{n(n-1)}{2}$ cạnh và mọi đỉnh đều có bậc n – 1.

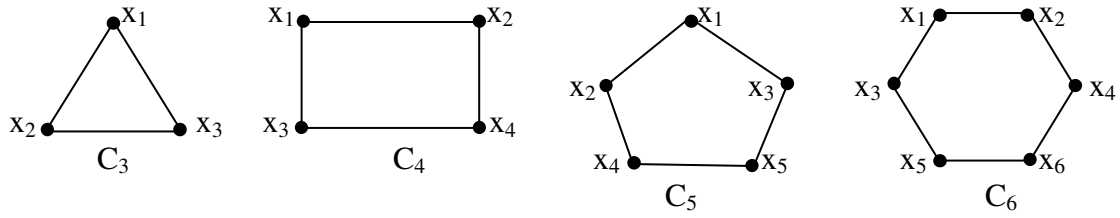
Hình 12 là các đồ thị K_1, K_2, K_3, K_4, K_5.



Hình 12. Các đồ thị đầy đủ K_1, K_2, K_3, K_4, K_5.

b. Đồ thị vòng: Đồ thị vòng n đỉnh ($n \geq 3$), ký hiệu C_n, là đồ thị có các đỉnh x_1, x_2, …, x_n và các cạnh (x_1, x_2), (x_2, x_3), …, (x_{n-1}, x_n), (x_n, x_1). Xem hình 13

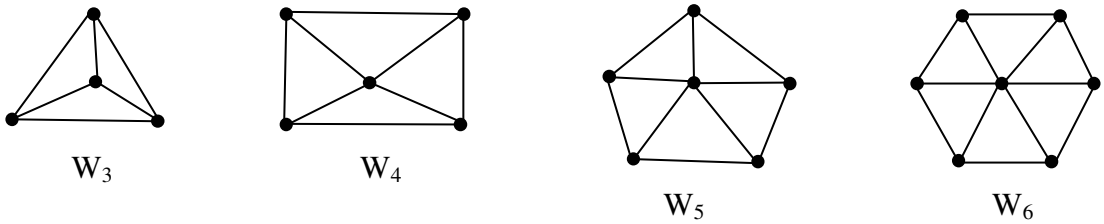
Đồ thị vòng C_n có n cạnh, và $\deg(x_i) = 2$, $\forall i = 1, 2, \ldots n$.



Hình 13. Các đồ thị vòng C_3, C_4, C_5, C_6.

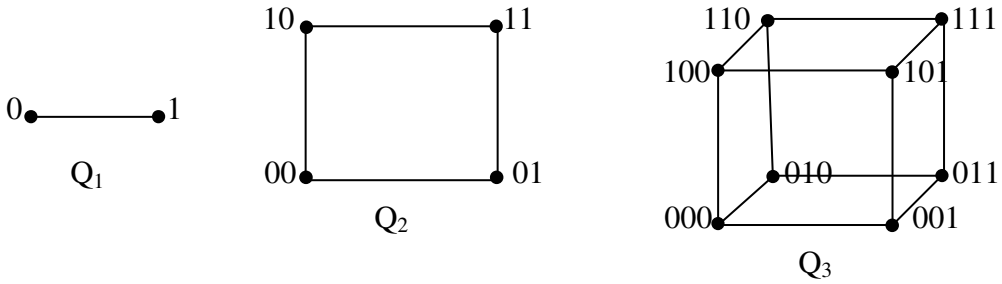
c. Đồ thị bánh xe: Khi thêm một đỉnh vào đồ thị vòng C_n và nối đỉnh này với tất cả các đỉnh của C_n được một đồ thị mới gọi là đồ thị bánh xe, ký hiệu W_n (hình 14).

Đồ thị bánh xe W_n có n + 1 đỉnh, 2n cạnh, một đỉnh bậc n và n đỉnh bậc 3.



Hình 14. Các đồ thị bánh xe W_3, W_4, W_5, W_6.

d. Đồ thị lập phương: Đồ thị lập phương được ký hiệu Q_n, là đồ thị có 2^n đỉnh và mỗi đỉnh được ký hiệu bằng một xâu nhị phân có độ dài n. Hai đỉnh của Q_n là kề nhau khi và chỉ khi các xâu nhị phân tương ứng khác nhau đúng một bit (Hình 15).

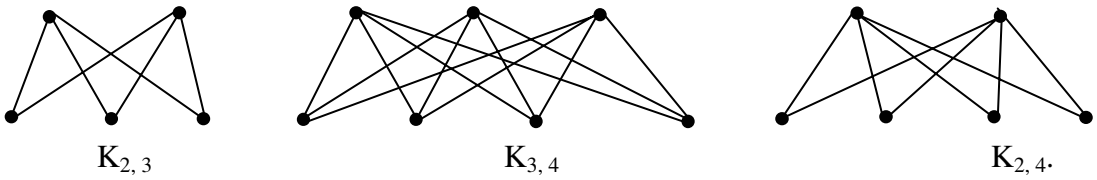


Hình 15. Các đồ thị lập phương Q_1, Q_2, Q_3.

Dễ thấy các đỉnh đồ thị lập phương Q_n đều có bậc n, và do đó nó có $n.2^{n-1}$ cạnh (vì số cạnh bằng một nửa tổng số bậc của tất cả các đỉnh).

e. Đồ thị phân đôi (đồ thị hai phía): Đơn đồ thị G = (X, U) được gọi là đồ thị phân đôi nếu tập các đỉnh X được phân chia thành hai tập không rỗng, rời nhau X_1, X_2 ($X_1 \cup X_2 = X$, $X_1 \neq \emptyset$, $X_2 \neq \emptyset$ và $X_1 \cap X_2 = \emptyset$) sao cho mỗi cạnh của đồ thị nối một đỉnh của X_1 với một đỉnh của X_2.

Đồ thị phân đôi G = ($X_1 \cup X_2$, U) có $|X_1| = m$, $|X_2| = n$ được gọi là đồ thị phân đôi đầy đủ, ký hiệu $K_{m, n}$ nếu mỗi đỉnh của X_1 đều được nối với mọi đỉnh trong X_2.



Hình 16. Một số đồ thị phân đôi đầy đủ

Có thể áp dụng thuật toán sau để nhận biết đồ thị G = (X,U) có là đồ thị phân đôi hay không.

Đặt: 1/ $X_1 = \{ x \mid x \in X, x \text{ lấy tùy ý}\}$

2/ $X_2 = \{ y \mid y \in X, y \text{ kề với các đỉnh của } X_1\}$

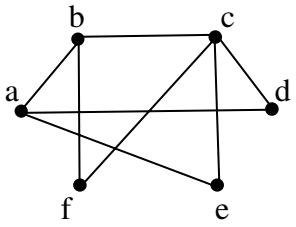
3/ $T = \{ z \mid z \in X, z \text{ kề với các đỉnh của } X_2\}$

- Nếu $T \cap X_2 \neq \emptyset$ thì G không phải là đồ thị phân đôi.
- Nếu $T \cap X_2 = \emptyset$ và $X_1 \cup X_2 \subset X$ thì lặp lại các bước 1/ – 3/ với $X_1 = X_1 \cup T$.
- Nếu $T \cap X_2 = \emptyset$ và $X_1 \cup X_2 = X$ thì đã phân hoạch được X và G là đồ thị phân đôi.

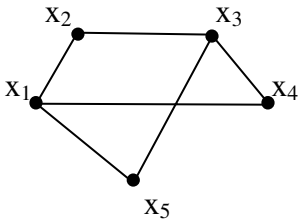
Sau một số hữu hạn bước, hoặc là chứng minh được G không phải là đồ thị phân đôi, hoặc là phân hoạch được tập đỉnh X nếu G là đồ thị phân đôi.

Thí dụ 1: Xét đồ thị trên hình 17. Chọn $X_1 = \{a\}$, ta có $X_2 = \{b, d, e\}$, từ đó $T = \{a, c\}$, rõ ràng $T \cap X_2 = \emptyset$ và có $X_1 = \{a\} \cup \{a, c\} = \{a, c\}$. Tiếp theo ta có $X_2 = \{b,d,e,f\}$, từ đó $T = \{a,c,f\}$. Rõ ràng $T \cap X_2 \neq \emptyset$. Vậy đồ thị đã cho không phải là đồ thị phân đôi.

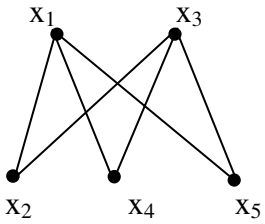
Thí dụ 2: Xét đồ thị trên hình 18a. Chọn $X_1 = \{x_1\}$, ta có $X_2 = \{x_2, x_4, x_5\}$, từ đó $T = \{x_1,x_3\}$. Rõ ràng $T \cap X_2 = \varnothing$, và có $X_1 = X_1 \cup T_1 = \{x_1, x_3\}$. Dễ nhận thấy $X_1 \cap X_2 = \varnothing$ và $X_1 \cup X_2 = X$. Vậy đồ thị đã cho là đồ thị phân đôi. Vẽ lại đồ thị ở dạng phân đôi được hình 18b.



Hình 17



Hình 18a



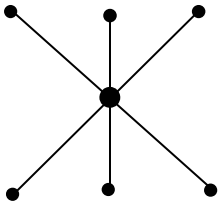
Hình 18b

1.4. Một vài ứng dụng của các đồ thị đặc biệt:

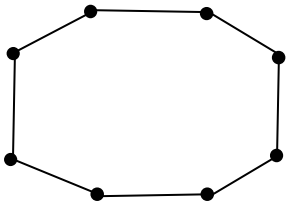
a. Các mạng cục bộ (LAN): Một số mạng cục bộ dùng **cấu trúc hình sao**, trong đó tất cả các thiết bị được nối với thiết bị điều khiển trung tâm. Mạng cục bộ kiểu này có thể biểu diễn bằng một đồ thị phân đôi đầy đủ $K_{1,n}$. Các thông báo gửi từ thiết bị này tới thiết bị khác đều phải qua thiết bị điều khiển trung tâm (hình 19a).

Mạng cục bộ cũng có thể có **cấu trúc vòng tròn**, trong đó mỗi thiết bị nối với đúng hai thiết bị khác. Mạng cục bộ kiểu này có thể biểu diễn bằng một đồ thị vòng C_n. Thông báo gửi từ thiết bị này tới thiết bị khác được truyền đi theo vòng tròn cho tới khi đến nơi nhận (hình 19b).

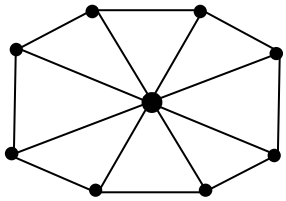
Cuối cùng, một số mạng cục bộ dùng **cấu trúc hỗn hợp** của hai cấu trúc trên. Các thông báo được truyền vòng quanh theo vòng tròn hoặc có thể qua thiết bị trung tâm. Sự dư thừa này có thể làm cho mạng đáng tin cậy hơn. Mạng cục bộ kiểu này có thể biểu diễn bằng một đồ thị bánh xe W_n (hình 19c).



a. Mạng hình sao



b. Mạng vòng tròn



c. Mạng hỗn hợp

Hình 19. Các cấu trúc mạng cục bộ

b. Xử lý song song: Các thuật toán để giải các bài toán có thể được thiết kế để thực hiện các phép toán một cách tuần tự, đó là thuật toán nối tiếp. Tuy nhiên có nhiều bài toán với số lượng tính toán rất lớn như bài toán mô phỏng thời tiết, hay phân tích mật mã, v.v… không thể giải được trong một khoảng thời gian hợp lý nếu dùng thuật toán nối tiếp ngay cả khi dùng các siêu máy tính. Ngoài ra, do những giới hạn về mặt vật lý đối với tốc độ thực hiện

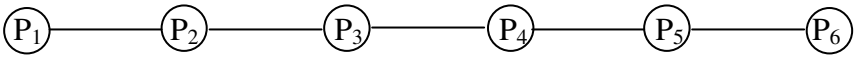
các phép toán cơ sở, nên thường gặp các bài toán không thể giải trong khoảng thời gian hợp lý bằng các thao tác nối tiếp. Vì vậy, người ta phải nghĩ đến kiểu xử lý song song.

Khi xử lý song song, người ta dùng các máy tính có nhiều bộ xử lý riêng biệt, mỗi bộ xử lý có bộ nhớ riêng, nhờ đó có thể khắc phục được những hạn chế của các máy nối tiếp. Các thuật toán song song phân chia bài toán chính thành một số bài toán con sao cho có thể giải đồng thời được. Do vậy, bằng các thuật toán song song và nhờ việc sử dụng các máy tính có bộ đa xử lý, người ta hy vọng có thể giải nhanh các bài toán phức tạp. Trong thuật toán song song có một dãy các chỉ thị theo dõi việc thực hiện thuật toán, gửi các bài toán con tới các bộ xử lý khác nhau, truyền các thông tin vào, thông tin ra tới các bộ xử lý thích hợp.

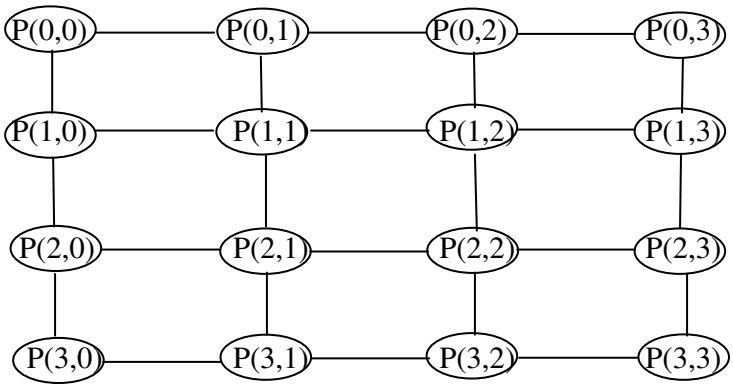
Khi dùng cách xử lý song song, mỗi bộ xử lý có thể cần các thông tin ra của các bộ xử lý khác. Do đó chúng cần phải được kết nối với nhau. Có thể dùng loại đồ thị thích hợp để biểu diễn mạng kết nối các bộ xử lý trong một máy tính có nhiều bộ xử lý. Kiểu mạng kết nối dùng để thực hiện một thuật toán song song cụ thể phụ thuộc vào những yêu cầu với việc trao đổi dữ liệu giữa các bộ xử lý, phụ thuộc vào tốc độ mong muốn và tất nhiên vào phần cứng hiện có.

Mạng kết nối các bộ xử lý song song đơn giản nhất và cũng đắt nhất là có các liên kết hai chiều giữa mỗi cặp bộ xử lý. Các mạng này có thể mô hình bằng đồ thị đầy đủ K_n, trong đó n là số bộ xử lý. Tuy nhiên, các mạng liên kết kiểu này có số kết nối quá nhiều mà trong thực tế số kết nối cần phải có giới hạn.

Các bộ xử lý có thể kết nối đơn giản là sắp xếp chúng theo một **mảng một chiều**. Ưu điểm của mảng một chiều là mỗi bộ xử lý có nhiều nhất 2 đường nối trực tiếp với các bộ xử lý khác. Nhược điểm là nhiều khi cần có rất nhiều các kết nối trung gian để các bộ xử lý trao đổi thông tin với nhau. Hình 20 là mảng một chiều với 6 bộ sử lý.



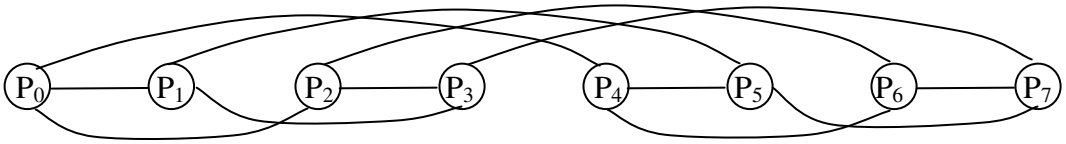
Hình 20. Mảng một chiều kết nối 6 bộ xử lý



Hình 21. Mảng kiểu lưới có 16 bộ xử lý

Mạng kiểu lưới (hoặc **mảng hai chiều**) rất hay được dùng cho các mạng liên kết. Trong một mạng như thế, số các bộ xử lý là một số chính phương, $n = m^2$. Các bộ xử lý được gán nhãn P(i, j), $0 \le i, j \le m-1$. Các kết nối hai chiều sẽ nối bộ xử lý P(i,j) với bốn bộ xử lý bên cạnh, tức là với P(i, j ±1) và P(i ±1, j) chừng nào các bộ xử lý còn ở trong lưới (Hình 21).

Mạng kết nối quan trọng nhất là mạng kiểu **siêu khối**. Với các mạng loại này số các bộ xử lý là luỹ thừa của 2, $n = 2^m$. Các bộ xử lý được gán nhãn là $P_0, P_1, ..., P_{n-1}$. Mỗi bộ xử lý có liên kết hai chiều với m bộ xử lý khác. Bộ xử lý P_i nối với bộ xử lý có chỉ số biểu diễn bằng dãy nhị phân khác với dãy nhị phân biểu diễn P_i đúng một bit. Mạng kiểu siêu khối cân bằng số các kết nối trực tiếp của mỗi bộ xử lý và số các kết nối gián tiếp sao cho các bộ xử lý có thể truyền thông được. Nhiều máy tính đã chế tạo theo mạng kiểu siêu khối và nhiều thuật toán đã được thiết kế để sử dụng mạng kiểu siêu khối. Đồ thị lập phương Q_m biểu diễn mạng kiểu siêu khối có 2^m bộ xử lý. Mạng siêu khối có 8 bộ vi sử lý được mô tả trong hình 22 nó cũng là cách vẽ khác của Q_3 so với hình 15.



Hình 22. Mạng kiểu siêu khối

2. Biểu diễn đồ thị bằng đại số.

Trong phần trên đồ thị được biểu diễn bằng hình vẽ, đó là cách biểu diễn rất trực quan nhưng cũng có nhiều bất tiện, đặc biệt khi sử lý đồ thị trên máy tính. Phần này trình bày các cách biểu diễn đồ thị không cần hình vẽ, điều đó đặc biệt thuận lợi khi sử lý đồ thị trên máy tính.

2.1. Danh sách kề.

Với mỗi $x \in X$ của đồ thị G = (X, U) không có cạnh bội, đặt:

$$G(x) = \{ y \mid y \in X, (x,y) \in U \}$$

Nghĩa là G(x) là tập đỉnh kề với x nếu G là đồ thị vô hướng, hoặc G(x) là tập đỉnh có cung đi từ x đến các đỉnh đó.

Tập hợp $\{G(x_i) \mid x_i \in X\}$ gọi là danh sách kề của đồ thị.

Danh sách kề chính là sự liệt kê các cạnh (cung) của đồ thị.

Thí dụ 1: Xét đồ thị ở hình 23, ta có:

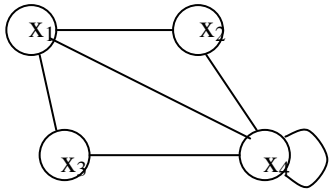
$G(x_1) = \{x_2, x_3, x_4\}$; $G(x_2) = \{x_1, x_4\}$;

$G(x_3) = \{x_1, x_4\}$; $G(x_4) = \{x_1, x_2, x_3, x_4\}$

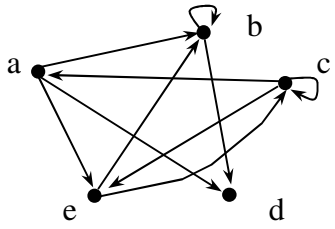
Thí dụ 2: Với đồ thị có hướng trong hình 24, ta có:

$G(a) = \{ b, d, e\}$ $G(b) = \{ b, d\}$

$G(c) = \{ c, a, e\}$ $G(d) = \varnothing$ $G(e) = \{b, c\}$



Hình 23



Hình 24

2.2. Danh sách cạnh

Là danh sách liệt kê các cạnh của đồ thị bằng một bảng chỉ rõ các đỉnh kề của từng cạnh có trong đồ thị đã cho. Chẳng hạn danh sách cạnh của các đồ thị trong hình 23, 24 là:

Danh sách cạnh của đồ thị hình 23	
Đỉnh đầu	Đỉnh cuối
x_1	x_2
x_1	x_3
x_1	x_4
x_2	x_4
x_3	x_4
x_4	x_4

Danh sách cạnh của đồ thị hình 24	
Đỉnh đầu	Đỉnh cuối
a	b
a	d
a	e
b	b
b	d
c	c
c	a
c	e
e	b
e	c

Việc biểu diễn đồ thị bằng danh sách kề hoặc danh sách cạnh, trong nhiều trường hợp dẫn đến việc thực hiện một số thuật toán khá cồng kềnh nhưng sẽ là đơn giản hơn nếu dùng dạng biểu diễn đồ thị bằng ma trận. Có hai kiểu biểu diễn đồ thị bằng ma trận, đó là ma trận kề và ma trận liên thuộc.

2.3. Ma trận kề

Định nghĩa 1: Cho G = (X, U) là đồ thị vô hướng với $X = \{x_1, x_2, \ldots, x_n\}$. Khi đó có thể biểu diễn đồ thị G bằng một ma trận vuông $A = (a_{ij})$ cấp n, trong đó:

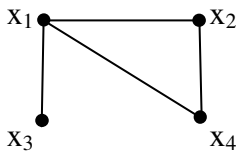
$$a_{ij} = \begin{cases} d, & \text{nếu } x_i \text{ nối với } x_j \text{ bằng d cạnh} \\ 0, & \text{nếu không có cạnh nối giữa } x_i \text{ và } x_j \end{cases}$$

Ma trận A gọi là ma trận kề của đồ thị G.

Chú ý rằng ma trận kề của một đồ thị tùy thuộc vào thứ tự liệt kê các đỉnh. Điều đó có nghĩa là với một đồ thị n đỉnh có thể viết được n! ma trận kề tương ứng.

Thí dụ 1: Với đồ thị trong hình 25, ta có:

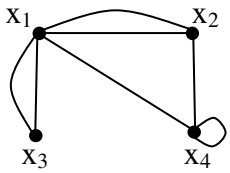
$$A = \begin{array}{c} \begin{matrix} x_1 & x_2 & x_3 & x_4 \end{matrix} \\ \begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \end{pmatrix} \end{array} \begin{matrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{matrix}$$



Hình 25

Thí dụ 2: Ma trận kề của đồ thị ở hình 26 viết theo thứ tự các đỉnh x_1, x_2, x_3, x_4 là:

$$A = \begin{pmatrix} 0 & 2 & 2 & 1 \\ 2 & 0 & 0 & 1 \\ 2 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 \end{pmatrix}$$



Hình 26

Một vài tính chất của ma trận kề của đồ thị vô hướng:

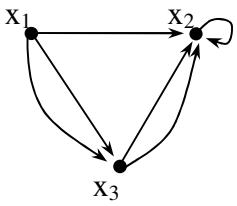
- Là ma trận đối xứng, phần tử trên đường chéo chính là khác 0 khi và chỉ khi tại đỉnh tương ứng có khuyên.
- Là ma trận ma trận không – một (ma trận có các phần tử là 0 hoặc 1) khi và chỉ khi G là đơn đồ thị.
- Tổng các phần tử của một hàng (cột) bằng bậc của đỉnh tương ứng với hàng (cột) đó.

Định nghĩa 2: Nếu G = (X, U) là đồ thị có hướng với $X = \{x_1, x_2, \ldots, x_n\}$ thì ma trận kề cũng là ma trận vuông $A = (a_{ij})$ cấp n , trong đó:

$$a_{ij} = \begin{cases} d, & \text{nếu có d cung đi từ } x_i \text{ đến } x_j \\ 0, & \text{nếu không có cung nào đi từ } x_i \text{ đến } x_j \end{cases}$$

Thí dụ: Ma trận kề của đồ thị hình 27 theo thứ tự các đỉnh x_1, x_2, x_3 là:

$$A = \begin{pmatrix} 0 & 1 & 2 \\ 0 & 1 & 0 \\ 0 & 2 & 0 \end{pmatrix}$$



Hình 27

Ma trận kề của đồ thị có hướng có tính chất sau:

- Là ma trận không đối xứng, phần tử trên đường chéo chính là khác 0 khi và chỉ khi tại đỉnh tương ứng có khuyên. Là ma trận không – một nếu G là đơn đồ thị.
- Tổng các phần tử của một hàng (cột) bằng bán bậc ra (bán bậc vào) của đỉnh tương ứng với hàng (cột) đó.

2.4. Ma trận liên thuộc

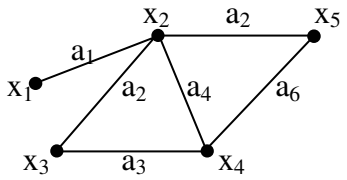
Cho G = (X, U) là đồ thị vô hướng với $X = \{x_1, x_2, \ldots, x_n\}$ và $U = \{a_1, a_2, \ldots, a_m\}$. Khi đó có thể biểu diễn G bằng ma trận $M = (m_{ij})_{n \times m}$ trong đó:

$$m_{ij} = \begin{cases} 1 , & \text{nếu đỉnh } x_i \text{ thuộc cạnh } a_j \\ 0 , & \text{nếu đỉnh } x_i \text{ không thuộc cạnh } a_j \end{cases}$$

Ma trận M gọi là ma trận liên thuộc của đồ thị G.

Thí dụ 1: Ma trận liên thuộc ứng với đồ thị trong hình 28 là:

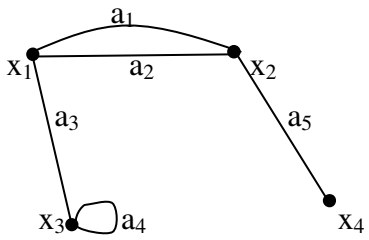
$$M = \begin{array}{c} \begin{matrix} a_1 & a_2 & a_3 & a_4 & a_5 & a_6 \end{matrix} \\ \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix} \end{array} \begin{matrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{matrix}$$



Hình 28

Thí dụ 2: Ma trận liên thuộc của đồ thị hình 29 là:

$$M = \begin{array}{c} \begin{matrix} a_1 & a_2 & a_3 & a_4 & a_5 \end{matrix} \\ \begin{pmatrix} 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \end{array} \begin{matrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{matrix}$$



Hình 29

Như vậy ma trận liên thuộc là ma trận không – một. Nếu một cạnh là khuyên thì cột tương ứng có đúng một số 1. Nếu có cạnh bội thì các cột tương ứng của các cạnh này giống hệt nhau.

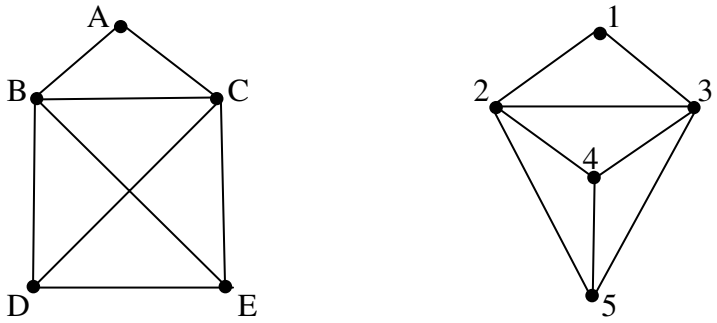
3. Sự đẳng cấu của các đồ thị

Với mỗi cách biểu diễn một đồ thị đã cho luôn luôn tồn tại nhiều dạng khác nhau, kể cả cách biểu diễn bằng hình vẽ. Những dạng biểu diễn khác nhau của một đồ thị dễ làm cho ta lầm tưởng đó là các đồ thị khác nhau. Trong phần này đề cập đến cách nhận biết một đồ thị với các dạng biểu diễn khác nhau.

Trong nhiều trường hợp cần phải xem xét hai đồ thị đã vẽ có phải là một hay không? Chẳng hạn, các hợp chất trong hóa học có thể cùng một công thức phân tử nhưng cấu trúc hóa học có thể khác nhau; vì thế khi tìm ra một chất cần phải vẽ cấu trúc hóa học (tức là đồ thị) của nó để xem chất đó có trùng với các chất đã tìm ra trước nó hay không. Vấn đề được giải quyết nhờ khái niệm đẳng cấu của đồ thị.

Định nghĩa: Hai đồ thị $G_1 = (X_1, E_1)$ và $G_2 = (X_2, E_2)$ được gọi là đẳng cấu với nhau nếu tồn tại một song ánh $f : X_1 \to X_2$ sao cho các đỉnh x và y là kề nhau trong G_1 khi và chỉ khi f(x) và f(y) là kề nhau trong G_2.

Thí dụ 1: Hai đồ thị trong hình 30 là đẳng cấu qua song ánh:
f(A) = 1; f(B) = 2;f(C) = 3 ;f(D) = 4; f(E) = 5

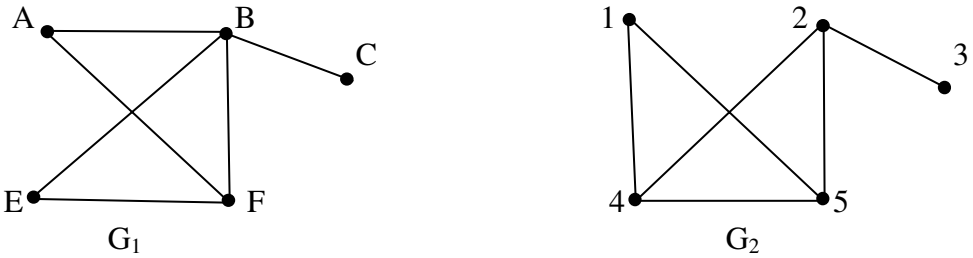


Hình 30. Hai đồ thị đẳng cấu

Việc xác định xem hai đồ thị có phải là đẳng cấu hay không là một việc làm rất khó khăn. Vì với hai đồ thị n đỉnh chúng có tới n! phép tương ứng 1–1 khác nhau giữa hai tập đỉnh, do đó sẽ mất rất nhiều thời gian để tìm được một phép tương ứng bảo toàn tính liền kề của các đỉnh.

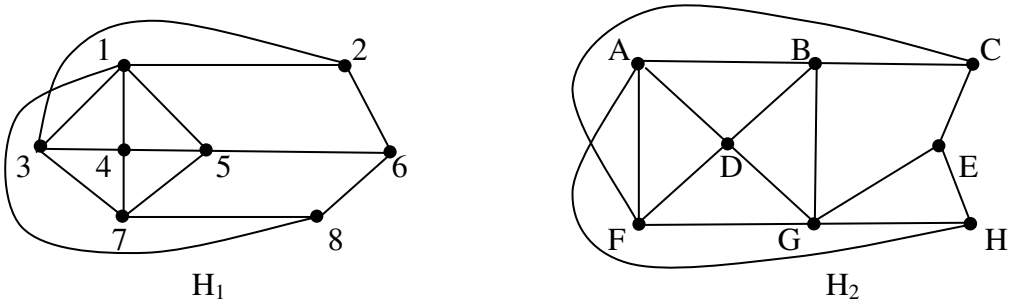
Việc xác định hai đơn đồ thị là không đẳng cấu đơn giản hơn việc xác định chúng là đẳng cấu, vì khi đó chỉ cần chỉ ra hai đồ thị đó không có chung một tính chất nào đó mà hai đồ thị đẳng cấu phải có như: **Cùng số đỉnh; cùng số đỉnh bậc k** (với mọi k nguyên không âm); **cùng số cạnh; v.v…**

Thí dụ 2: Hai đồ thị G_1, G_2 trong hình 31 là không đẳng cấu vì G_1 có một đỉnh bậc 4 còn G_2 thì không có đỉnh bậc 4 nào.



Hình 31

Thí dụ 3: Hai đồ thị H_1 và H_2 trong hình 32 cũng không đẳng cấu vì H_1 có 3 đỉnh bậc ba (các đỉnh 2, 6, 8), mỗi đỉnh bậc 3 này đều kề với một đỉnh bậc 4; nhưng trong 3 đỉnh bậc ba của H_2 (các đỉnh C, E, H) có đỉnh E không kề với đỉnh bậc 4 nào cả.



Hình 32

Dễ thấy là hai đồ thị đẳng cấu thì hai ma trận kề (theo một thứ tự nào đó của các đỉnh) tương ứng là bằng nhau.

4. Tính liên thông trong đồ thị

4.1. Đường đi, chu trình

Định nghĩa 1: Đường đi từ đỉnh x_0 đến đỉnh x_k của đồ thị G = (X, U) (G có thể là đồ thị vô hướng, có hướng hoặc đồ thị hỗn hợp) là dãy các đỉnh $x_0x_1 \ldots x_k$, trong đó (x_i, x_{i+1}), i = 0, 1,…, k–1, là các cạnh hay cung của đồ thị, nghĩa là $(x_i, x_{i+1}) \in U$.

Đỉnh x_0 gọi là đỉnh đầu, đỉnh x_k gọi là đỉnh cuối của đường đi.

Số các cạnh có trong một đường đi gọi là độ dài của đường đi đó.

Định nghĩa 2: Đường đi có đỉnh đầu trùng với đỉnh cuối gọi là chu trình.

Đường đi hay chu trình gọi là *đơn* nếu nó không đi qua cạnh nào quá một lần.

Đường đi hay chu trình gọi là *sơ cấp* nếu nó không đi qua đỉnh nào quá một lần.

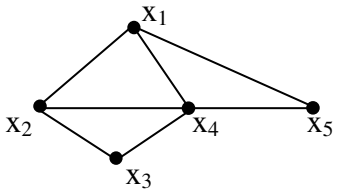
Cũng có thể biểu diễn đường đi từ đỉnh x_0 đến đỉnh x_k bằng dãy các cạnh: (x_0, x_1), (x_1, x_2), …, (x_{k-1}, x_k).

Thí dụ 1: Xét đồ thị trong hình 33, ta có:

$x_1\ x_2\ x_4\ x_5$ là một đường đi đơn từ x_1 đến x_5 có độ dài bằng 3.

$x_1\ x_4\ x_5$ là một đường đi đơn từ x_1 đến x_5 có độ dài bằng 2.

$x_4\ x_1\ x_2\ x_4\ x_1\ x_5$ là một đường đi từ x_4 đến x_5 có độ dài 5 nhưng không phải là đường đi đơn vì cung (x_4,x_1) đi qua 2 lần.



Hình 33

$x_1\ x_4\ x_3\ x_2\ x_4\ x_5\ x_1$ là chu trình có độ dài 6 (không phải là chu trình sơ cấp).

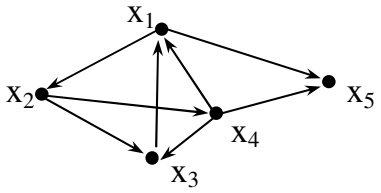
Thí dụ 2:

$x_1\ x_2\ x_4\ x_5$ là một đường đi đơn từ x_1 đến x_5 có độ dài bằng 3.

$x_1\ x_2\ x_3\ x_1$ là một chu trình đơn có độ dài 3.

$x_1\ x_2\ x_3\ x_1\ x_5$ là một đường đi đơn từ x_1 đến x_2 nhưng không sơ cấp.

$x_1\ x_2\ x_3\ x_4$ không phải là một đường đi vì (x_3, x_4) không phải là một cung.



Hình 34

Định nghĩa 3: Hai đỉnh x và y của đồ thị G = (X,U) được gọi là liên thông với nhau nếu tồn tại ít nhất một đường đi từ x đến y hoặc từ y đến x.

Định lý 1: Trong một đồ thị vô hướng G = (X, U) có n đỉnh ($n \geq 3$), nếu mọi đỉnh đều có bậc không nhỏ hơn 2 ($\deg(x) \geq 2, \forall x \in X$) thì trong G luôn luôn tồn tại một chu trình đơn.

Chứng minh: Xét tất cả các đường đi đơn có trong G. Vì số đỉnh của G là hữu hạn nên số đường đi đơn có trong G cũng là hữu hạn. Xét đường đi p có độ dài lớn nhất, chẳng hạn

$$p = x_0\ x_1\ \ldots\ x_p\ .$$

Vì deg(x_0) ≥ 2 nên ngoài cạnh (x_0, x_1) còn có ít nhất là một cạnh nữa thuộc x_0, gọi cạnh đó là (x_0, y). Nếu y không trùng với bất kỳ đỉnh nào trong p thì đường đi $p_1 = y\ x_0\ x_1 \dots x_p$ có độ dài lớn hơn p vì có thêm cạnh (y, x_0), điều này trái với giả thiết p có độ dài lớn nhất. Vậy y phải trùng với một đỉnh nào đó của p, chẳng hạn $y \equiv x_k$. Khi đó $x_0\ x_1 \dots x_k\ x_0$ là một chu trình đơn.

Định lý được chứng minh.

Bài toán ứng dụng: Trong một hội nghị có n người (n ≥ 3), mỗi người đều quen với ít nhất 2 người khác. Chứng tỏ rằng có thể chọn ra một số người để khi xếp ngồi quanh một bàn tròn thì mỗi người đều ngồi giữa 2 người quen.

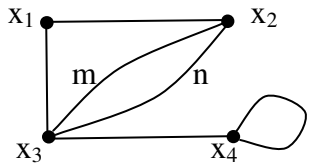
Giải: Coi mỗi người là một đỉnh của đồ thị, 2 người quen nhau được nối với nhau bằng một cạnh được một đồ thị vô hướng thoả mãn định lý trên.

Định lý 2: Trong đồ thị vô hướng có n đỉnh (n≥ 4), nếu mọi đỉnh đều có bậc không nhỏ hơn 3 thì trong đồ thị luôn luôn có chu trình đơn có độ dài chẵn.

Định lý được chứng minh tương tự định lý 1.

Định lý 3: Cho G là một đồ thị liên thông có ma trận kề là A theo thứ tự các đỉnh x_1, x_2, …, x_n (với các cạnh vô hướng, có hướng hay cạnh bội, có thể có khuyên). Số các đường đi có độ dài k từ x_i đến x_j (k là số nguyên dương) bằng giá trị của phần tử (i,j) trong ma trận A^k.

Chúng ta thừa nhận, không chứng minh định lý này và minh hoạ bằng thí dụ sau:



Hình 35

Ta có:

$$A = \begin{pmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 2 & 0 \\ 1 & 2 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{pmatrix} \Rightarrow A^2 = \begin{pmatrix} 2 & 2 & 2 & 1 \\ 2 & 5 & 1 & 2 \\ 2 & 1 & 6 & 1 \\ 1 & 2 & 1 & 2 \end{pmatrix} \Rightarrow A^3 = \begin{pmatrix} 4 & 6 & 7 & 3 \\ 6 & 4 & 14 & 3 \\ 7 & 14 & 5 & 7 \\ 3 & 3 & 7 & 3 \end{pmatrix}$$

Như vậy có 3 đường đi có độ dài 3 từ x_1 đến x_4, đó là:

$x_1\ x_2\ m\ x_3\ x_4$; $x_1\ x_2\ n\ x_3\ x_4$; $x_1\ x_3\ x_4\ x_4$.

Có 6 đường đi có độ dài 3 từ x_1 đến x_2, …

4.2. Đồ thị liên thông

a-) Đồ thị con, đồ thị bộ phận

Định nghĩa: Cho đồ thị G = (X, U).

Nếu bỏ đi một số đỉnh cùng với các cạnh (cung) xuất phát từ các đỉnh bỏ đi đó thì được đồ thị mới gọi là đồ thị con của đồ thị G.

Nếu giữ nguyên các đỉnh và bỏ đi một số cạnh thì được đồ thị mới gọi là đồ thị bộ phận của đồ thị G.

Nói cách khác:

Đồ thị con của đồ thị G = (X, U) là đồ thị H = (A, E), trong đó $A \subseteq X$ và $E \subseteq U$.

Đồ thị bộ phận của đồ thị G = (X, U) là đồ thị K = (X, F), trong đó $F \subseteq U$.

Thí dụ: Nếu G là mạng giao thông trong toàn quốc với ba phương tiện: Máy bay, Tàu hoả, Ô tô. Khi ấy nếu chỉ xét mạng giao thông ở miền Bắc với cả 3 phương tiện thì có một đồ thị con. Còn nếu xét mạng giao thông trên toàn quốc với 2 trong 3 phương tiện đã nêu thì có một đồ thị bộ phận.

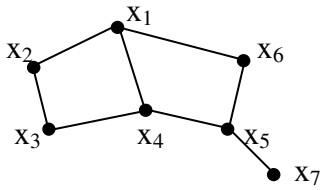
b-) Đồ thị vô hướng liên thông

Định nghĩa: Đồ thị vô hướng G = (X, U) được gọi là liên thông, nếu luôn luôn tìm được một đường đi giữa 2 đỉnh bất kỳ của đồ thị.

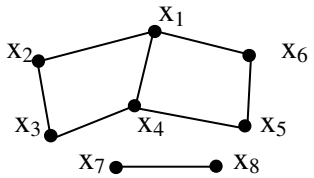
Nếu đồ thị vô hướng G = (X, U) không liên thông, nhưng có đồ thị con liên thông thì đồ thị con đó gọi là thành phần liên thông của đồ thị G.

Dễ thấy đồ thị G là liên thông khi và chỉ khi nó chỉ có một thành phần liên thông duy nhất.

Thí dụ:



Hình 36. Đồ thị vô hướng liên thông



Hình 37. Đồ thị vô hướng có 2 thành phần liên thông

Định lý 1: Nếu bậc của mọi đỉnh của đồ thị vô hướng G = (X, U) không nhỏ hơn một nửa số đỉnh thì đồ thị đó liên thông. (Nếu $\deg(x) \geqslant \frac{N(X)}{2}$, $\forall x \in X$ thì G = (X,U) liên thông).

Chứng minh: Dùng phản chứng để chứng minh định lý.

Giả sử đồ thị G thoả mãn điều kiện của định lý nhưng không liên thông. Khi đó G phải có ít nhất hai thành phần liên thông. Gọi $G_1 = (X_1, U_1)$ và $G_2 = (X_2, U_2)$ là hai thành phần liên thông của đồ thị. Khi đó:

$$X_1 \cap X_2 = \varnothing; \quad X_1 \cup X_2 \subseteq X \quad \Rightarrow \quad N(X_1) + N(X_2) \leq N(X)$$

Từ đó: hoặc $N(X_1) \leqslant \frac{N(X)}{2}$, hoặc $N(X_2) \leqslant \frac{N(X)}{2}$

Không giảm tổng quát, giả sử rằng $N(X_1) \leqslant \frac{N(X)}{2}$

Suy ra $\forall y \in X_1$ $\deg(y) \leqslant \frac{N(X)}{2} - 1$ (Vì mọi đỉnh của G_1 chỉ có thể nối nhiều nhất với $\frac{N(X)}{2} - 1$ đỉnh còn lại).

Điều này trái với giả thiết $\deg(x) \geqslant \frac{N(X)}{2}$, $\forall x \in X$. Vậy G phải liên thông.

Định lý 2: Đồ thị vô hướng có tổng bậc của hai đỉnh tùy ý không nhỏ hơn số đỉnh là đồ thị liên thông.

Chứng minh: Chứng minh định lý bằng phản chứng.

Giả sử đồ thị G có n đỉnh và x, y là hai đỉnh tùy ý của G có $\deg(x) + \deg(y) \geq n$ nhưng x, y không liên thông. Khi đó G có ít nhất hai thành phần liên thông. Gọi G_1 là thành phần liên thông chứa x và G_2 là thành phần liên thông chứa y. Giả sử G_1 có n_1 đỉnh và G_2 có n_2 đỉnh, ta có:

$$n_1 + n_2 \leq n \text{ và } \deg(x) + \deg(y) \leq (n_1 - 1) + (n_2 - 1) = n_1 + n_2 - 2 < n$$

Điều đó mâu thuẫn với giả thiết. Vậy x, y phải liên thông, nghĩa là G liên thông,

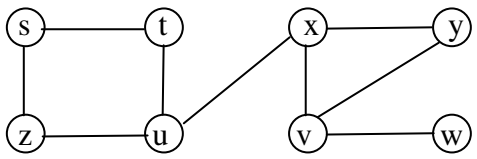
c-) Đỉnh cắt và cầu

Định nghĩa: Một đỉnh của đồ thị được gọi là đỉnh cắt hay điểm khớp nếu khi xóa đỉnh đó và tất cả các cạnh liên thuộc với nó thì nhận được đồ thị con có số thành phần liên thông nhiều hơn so với đồ thị đã cho.

Một cạnh của đồ thị được gọi là cạnh cắt hay cầu nếu khi loại cạnh đó ra khỏi đồ thị đã cho thì nhận được đồ thị bộ phận có số thành phần liên thông nhiều hơn đồ thị xuất phát.

Rõ ràng việc xóa đỉnh cắt hay cầu khỏi một đồ thị liên thông thì đồ thị nhận được là không liên thông

Thí dụ: Xét đồ thị trên hình 38, các đỉnh x, u, v là các đỉnh cắt. Các cạnh (u,x), (v,w) là các cầu (cạnh cắt).



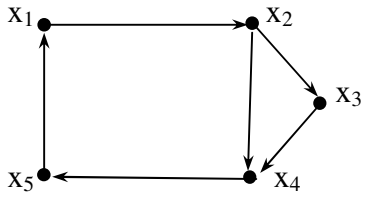
Hình 38

d-) Đồ thị có hướng liên thông

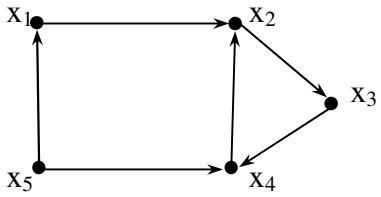
Định nghĩa 1: Đồ thị có hướng G = (X, U) gọi là liên thông mạnh nếu luôn luôn tìm được đường đi giữa 2 đỉnh bất kỳ của nó.

Định nghĩa 2: Đồ thị có hướng G = (X, U) gọi là liên thông yếu nếu đồ thị vô hướng tương ứng (tức đồ thị đã cho được thay cung bằng cạnh) với nó là đồ thị liên thông.

Thí dụ:



Hình 39. Đồ thị có hướng liên thông mạnh



Hình 40. Đồ thị có hướng liên thông yếu

Từ một đồ thị vô hướng, nếu định hướng cho các cạnh được đồ thị có hướng. Một đồ thị vô hướng liên thông, có thể định hướng được thành đồ thị liên thông mạnh gọi là đồ thị định hướng được.

Định lý: Đồ thị vô hướng liên thông là định hướng được khi và chỉ khi mỗi cạnh của nó nằm trên ít nhất một chu trình.

Chứng minh:

Điều kiện cần: Giả sử (x, y) là một cạnh của đồ thị. Từ sự tồn tại một đường đi có hướng từ x đến y và ngược lại suy ra (x, y) phải nằm trên một chu trình.

Điều kiện đủ: Tiến hành gắn hướng cho các cạnh theo nguyên tắc sau:

Giả sử C là một chu trình nào đó trong đồ thị vô hướng G có mỗi cạnh nằm trên ít nhất một chu trình, tiến hành định hướng các cạnh của C theo hướng đi của chu trình này. Nếu mọi cạnh của G được định hướng, bài toán được giải quyết xong. Nếu còn một cạnh nào đó chưa được định hướng thì cạnh này phải nằm trên một chu trình C' nào đó, tiếp tục định hướng cho các cạnh của C' (trừ các cạnh đã được định hướng trong C, nếu có). Lặp lại quá trình trên nếu còn cạnh chưa được định hướng. Vì số chu trình của G là hữu hạn nên việc định hướng các cạnh sẽ kết thúc sau một số hữu hạn bước và được một đồ thị có hướng liên thông mạnh.

5. Số ổn định trong, số ổn định ngoài và nhân của đồ thị.

Trong phần này chúng ta xét đồ thị vô hướng hoặc có hướng G = (X,U).

5.1. Số ổn định trong

Định nghĩa 1:

Tập con A của X ($A \subseteq X$) được gọi là tập ổn định trong của G nếu 2 đỉnh tuỳ ý của A là không kề nhau.

Dễ thấy, nếu A là tập ổn định trong của một đồ thị thì khi bớt đi một số đỉnh của A được tập mới vẫn là tập ổn định trong của đồ thị đó.

Thí dụ: Bài toán về 8 quân hậu: Cần xếp 8 quân hậu trên bàn cờ vua sao cho chúng không ăn được nhau.

Biểu diễn 64 ô của bàn cờ bằng 64 đỉnh của một đồ thị, hai đỉnh x và y sẽ có cạnh nối với nhau nếu đó là 2 vị trí mà 2 quân hậu có thể ăn lẫn nhau. Các vị trí cần tìm chính là tập đỉnh ổn định trong có 8 đỉnh của đồ thị.

Định nghĩa 2: Tập $A \subseteq X$ gọi là tập ổn định trong cực đại của G nếu nó thoả mãn:

- A là tập ổn định trong.
- Nếu thêm vào A một đỉnh tuỳ ý được tập mới mất tính ổn định trong.

Định nghĩa 3: Gọi L là họ tất cả các tập ổn định trong của G. Khi ấy số:

$$\alpha(G) = \max \{ N(A) \mid A \in L \}$$

gọi là số ổn định trong của đồ thị G.

5.2. Số ổn định ngoài

a. Các định nghĩa.

Định nghĩa 1: Tập con B của X ($B \subseteq X$) được gọi là tập ổn định ngoài của G nếu với mỗi đỉnh $y \in X \setminus B$ đều có một đỉnh $x \in B$ sao cho $(y,x) \in U$, nghĩa là có cạnh nối giữa x và y hoặc (y,x) là một cung của đồ thị.

Tương tự tập ổn định trong, nếu thêm vào tập ổn định ngoài một số đỉnh bất kỳ thì tập mới vẫn là tập ổn định ngoài.

Thí dụ: Cần xếp 5 quân hậu trên bàn cờ vua sao cho chúng kiểm soát được toàn bộ bàn cờ.

Biểu diễn bàn cờ như thí dụ trong 5.1. Vị trí cần tìm chính là một tập ổn định ngoài gồm 5 đỉnh.

Định nghĩa 2: Tập $B \subseteq X$ gọi là tập ổn định ngoài cực tiểu của G nếu nó thoả mãn:

- B là tập ổn định ngoài.
- Nếu bớt đi một đỉnh bất kỳ của B được tập mới mất tính ổn định ngoài.

Định nghĩa 3: Gọi M là họ tất cả các tập ổn định ngoài của G. Khi ấy số:

$$\beta(G) = \min \{ N(B) \mid B \in M \}$$

gọi là số ổn định ngoài của đồ thị G.

b. Thuật toán tìm số ổn định ngoài

Thuật toán sau đây cho phép tìm các tập ổn định ngoài có số phần tử ít nhất của đồ thị G = (X, U). Thuật toán có tên là thuật toán tìm vị trí đặt vọng gác.

Bước 1. Xây dựng ánh xạ $\Delta: X \to X$ theo cách sau:

Với mỗi $x_i \in X$ thì $\Delta(x_i) \subseteq X$, trong đó $y \in \Delta(x_i)$ nếu:

- $y = x_i$ hoặc y kề với x_i, nếu G là đồ thị vô hướng.
- $y = x_i$ hoặc y có cung đi đến x_i, nếu G là đồ thị có hướng.

Bước 2. Xét các tập $\Delta(x_1), \Delta(x_2), \ldots, \Delta(x_n)$ đã được xác định trong bước 1. Từ đó tìm số ít nhất các tập $\Delta(x_i)$ sao cho:

$$\Delta(x_{i_1}) \cup \Delta(x_{i_2}) \cup \ldots \cup \Delta(x_{i_k}) = X$$

khi ấy $B = \{x_{i_1}, x_{i_2}, \ldots, x_{i_k}\}$ là tập ổn định ngoài cực tiểu của đồ thị và số ổn định ngoài là $\beta(G) = N(B)$.

Thí dụ: Tìm số ổn định ngoài của đồ thị trong hình 41.

Ta có $X = \{x_1, x_2, x_3, x_4, x_5, x_6, x_7\}$

Bước 1. Xác định ánh xạ Δ:

$\Delta(x_1) = \{x_1, x_2, x_3, x_4\}$;

$\Delta(x_2) = \{x_1, x_2, x_3\}$;

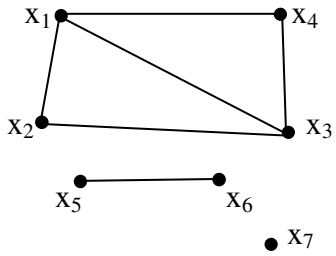
$\Delta(x_3) = \{x_1, x_2, x_3, x_4\}$;

$\Delta(x_4) = \{x_1, x_3, x_4\}$;

$\Delta(x_5) = \{x_5, x_6\}$;

$\Delta(x_6) = \{x_5, x_6\}$;

$\Delta(x_7) = \{x_7\}$.



Hình 41

Bước 2. Từ các tập $\Delta(x_i)$ trên, ta có:

$\Delta(x_1) \cup \Delta(x_5) \cup \Delta(x_7) = X$; $\Delta(x_1) \cup \Delta(x_6) \cup \Delta(x_7) = X$;

$\Delta(x_3) \cup \Delta(x_5) \cup \Delta(x_7) = X$; $\Delta(x_3) \cup \Delta(x_6) \cup \Delta(x_7) = X$

Vậy có 4 tập hợp:

$B_1 = \{x_1, x_5, x_7\}$; $B_2 = \{x_1, x_6, x_7\}$; $B_3 = \{x_3, x_5, x_7\}$; $B_4 = \{x_3, x_6, x_7\}$

là các tập ổn định ngoài có số phần tử ít nhất. Từ đó có số ổn định ngoài là $\beta(G) = 3$.

5.3. Nhân của đồ thị

Định nghĩa: Tập con C các đỉnh của đồ thị G = (X, U) được gọi là nhân của G nếu C vừa là tập ổn định trong vừa là tập ổn định ngoài của đồ thị G.

Trong đồ thị người ta quan tâm đến các nhân có số phần tử ít nhất.

Thí dụ 1: Trở lại thí dụ trong 5.2.b. Dễ thấy các tập hợp:

$B_1 = \{x_1, x_5, x_7\}$; $B_2 = \{x_1, x_6, x_7\}$; $B_3 = \{x_3, x_5, x_7\}$; $B_4 = \{x_3, x_6, x_7\}$

cũng là các tập ổn định trong.

Vậy đồ thị có 4 nhân có số phần tử ít nhất là các tập B_1, B_2, B_3, B_4.

Đồ thị có các tập ổn định trong có 4 phần tử là:

$A_{10} = \{x_2, x_4, x_5, x_7\}$; $A_{11} = \{x_2, x_4, x_6, x_7\}$

và không có tập ổn định trong có trên 4 phần tử.

Vậy số ổn định trong là $\alpha(G) = 4$.

Thí dụ 2: Đồ thị trong hình 42 có tập đỉnh là:

$X = \{x_1, x_2, x_3, x_4, x_5, x_6\}$

Ta có: $\Delta(x_1) = \{x_1\}$;

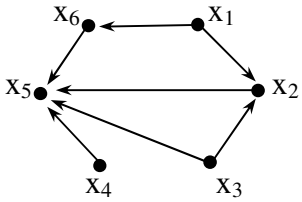
$\Delta(x_2) = \{x_1, x_2, x_3\}$;

$\Delta(x_3) = \{x_3\}$;

$\Delta(x_4) = \{x_4\}$;

$\Delta(x_5) = \{x_2, x_3, x_4, x_5, x_6\}$;

$\Delta(x_6) = \{x_1, x_6\}$;



Hình 42

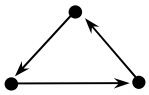
Từ đó có: $\Delta(x_1) \cup \Delta(x_5) = X$, $\Delta(x_2) \cup \Delta(x_5) = X$, $\Delta(x_5) \cup \Delta(x_6) = X$

Vậy các tập ổn định ngoài có số phần tử ít nhất là $B_1 = \{x_1, x_5\}$, $B_2 = \{x_2, x_5\}$ và $B_3 = \{x_5, x_6\}$ và số ổn định ngoài của đồ thị là $\beta(G) = 2$.

Tập B_1 cũng là tập ổn định trong, do đó nó cũng là nhân có ít phần tử nhất của đồ thị.

Các tập ổn định trong cực đại có: $A_1 = \{x_1, x_3, x_4\}$; $A_1 = \{x_2, x_4, x_6\}$ do đó số ổn định trong của đồ thị là $\alpha(G) = 3$.

Chú ý rằng không phải đồ thị nào cũng có nhân, chẳng hạn đồ thị trong hình 43 có ba tập ổn định trong đều có 1 đỉnh và ba tập ổn định ngoài đều có 2 đỉnh nên đồ thị không có nhân.



Hình 43

6. Sắc số của đồ thị

6.1. Định nghĩa

Sắc số của đồ thị là số màu tối thiểu cần dùng để tô màu các đỉnh của đồ thị sao cho 2 đỉnh kề nhau có màu khác nhau.

Sắc số của đồ thị G được ký hiệu là $\lambda(G)$.

Hiện nay chưa có một thuật toán hữu hiệu nào để tìm sắc số của một đồ thị. Người ta mới chỉ chứng minh được một số định lý sau.

6.2. Một số định lý về sắc số

Định lý 1: Đối với đồ thị đầy đủ, sắc số luôn luôn bằng số đỉnh của đồ thị.

Chứng minh: Định lý được chứng minh bằng qui nạp theo số đỉnh của đồ thị.

Với K_1 – đồ thị đầy đủ có 1 đỉnh, định lý hiển nhiên đúng.

Giả sử định lý đúng với đồ thị đầy đủ n đỉnh K_n, nghĩa là $\lambda(K_n) = n$.

Xét đồ thị đầy đủ K_{n+1}, bớt đi 1 đỉnh bất kỳ cùng các cạnh liên thuộc với nó được K_n có $\lambda(K_n) = n$. Thêm đỉnh bị loại và các cạnh bị loại thì đỉnh mới không thể tô bằng n màu đã có nên phải thêm một màu. Vậy $\lambda(K_{n+1}) = n + 1$.

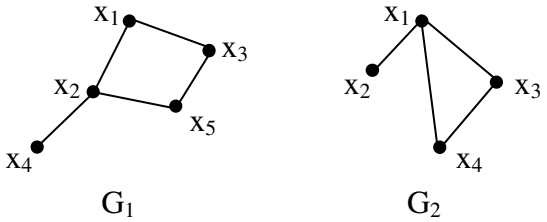
Định lý được chứng minh .

Định lý 2: Đồ thị vô hướng G = (X, U) có $\lambda(G) = 2$ khi và chỉ khi trong G không có chu trình có độ dài lẻ.

Chúng ta thừa nhận, không chứng minh định lý này.

Thí dụ: Xét 2 đồ thị G_1 và G_2 ở hình 44.

Trong G_1 không có chu trình có độ dài lẻ, có thể tô x_1 bằng một màu (đỏ chẳng hạn), khi đó x_2, x_3 được tô bằng cùng một màu khác (xanh chẳng hạn) và x_4, x_5 có thể tô trở lại màu đỏ. Nghĩa là $\lambda(G_1) = 2$.



Hình 44

Trong G_2 có một chu trình có độ dài lẻ (x_1 x_3 x_4). Nếu tô x_3 bằng màu đỏ thì x_1, x_4 không thể tô bằng cùng một màu (khác màu đỏ), nghĩa là $\lambda(G_2) > 2$ (Dễ thấy $\lambda(G_2) = 3$).

Định lý 3: Trong đồ thị vô hướng G với n đỉnh luôn luôn có $\lambda(G)\alpha(G) \geq n$. Trong đó $\alpha(G)$ là số ổn định trong và $\lambda(G)$ là sắc số của đồ thị.

Chứng minh: Giả sử $\lambda(G) = k$ ($k \le n$), như vậy các đỉnh của đồ thị được tô bằng k màu. Gọi X_i là tập đỉnh được tô bằng màu i, ta có $X_i \cap X_j = \varnothing$ khi $i \ne j$. Mặt khác $X_1 \cup X_2 \cup \ldots \cup X_k = X$. Theo nguyên lý cộng ta có:

$$N(X) = N(X_1) + N(X_2) + \ldots + N(X_k) = n \qquad (1)$$

Do 2 đỉnh tuỳ ý trong X_i (i = 1, 2, …, k) là không kề nhau (suy từ định nghĩa sắc số) do đó $\alpha(G) \ge N(X_i)$, $\forall i = 1, 2, \ldots, k$. Từ đó và do (1) được:

$$n \le k\alpha(G) = \lambda(G)\alpha(G)$$

Định lý được chứng minh.

Định lý 4: (Định lý bốn màu) Sắc số của đồ thị phẳng là không lớn hơn 4. (Nếu G là đồ thị phẳng thì $\lambda(G) \le 4$).

Định lý này được hai nhà toán học Mỹ là Keneth Appel và Wollgang Haken chứng minh năm 1976 trên máy tính.

Đồ thị phẳng là đồ thị đồ thị có ít nhất một cách biểu diễn hình học trên mặt phẳng sao cho các cạnh của nó chỉ cắt nhau tại các đỉnh của đồ thị.

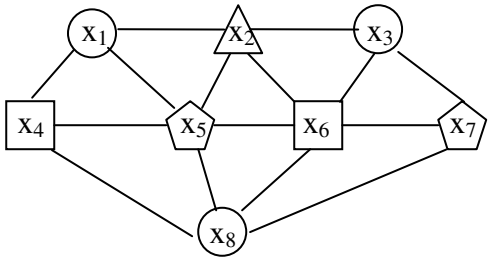
Việc tìm sắc số của đồ thị thường phải kết hợp các định lý trên với cách tô màu cụ thể.

Thí dụ: Xét đồ thị ở hình 45

Đồ thị là phẳng $\Rightarrow$ $\lambda(G) \le 4$

Đồ thị có chu trình lẻ $\Rightarrow$ $\lambda(G) > 2$

Tiến hành tô màu cụ thể được $\lambda(G) = 4$ (mỗi loại hình trong đồ thị thể hiện một màu)



Hình 45

Chú ý: Có thể dùng thuật toán sau để tô màu một đồ thị đơn (tức là tìm sắc số của đồ thị đơn):

- Liệt kê các đỉnh $x_1, x_2, \ldots, x_n$ theo thứ tự bậc giảm dần ($\deg(x_1) \ge \deg(x_2) \ge \ldots$
- Gán màu 1 cho đỉnh x_1 và các đỉnh không kề với x_1.
- Tiếp tục gán màu 2 cho đỉnh chưa được gán màu có bậc cao nhất và các đỉnh không kề với đỉnh đã gán màu 2.
- Thủ tục gán các màu tiếp theo được tiến hành tương tự.

6.3. Vài thí dụ ứng dụng sắc số

Bài toán 1: Bài toán lập lịch thi.

Hãy lập lịch thi n môn học cho sinh viên sao cho không có sinh viên nào phải thi 2 môn cùng một thời điểm.

Giải: Lập một đồ thị có các đỉnh là các môn thi. Nếu có sinh viên phải thi 2 môn thì 2 đỉnh tương ứng được nối với nhau bằng một cạnh. Số đợt thi là sắc số của đồ thị đã lập.

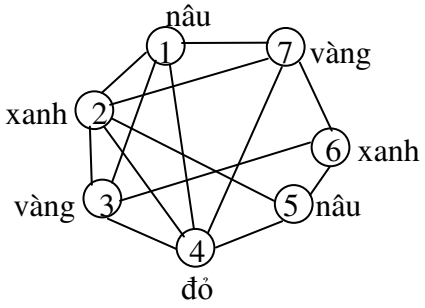
Thí dụ: Có 7 môn thi đánh số từ 1 đến 7 và các cặp môn thi sau là môn thi chung của ít nhất hai sinh viên:

(1,2); (1.3); (1,4); (1,7)

(2,3); (2,4); (2,5); (2,7)
(3,4); (3,6);
(4,5); (4,7);
(5,6); (6,7)

Ta có: deg(2) = deg(4) >
> deg(1) = deg(3) = deg(7) >
> deg(5) = deg(6)

Sau khi tô màu các đỉnh (hình 46), sắc số của đồ thị bằng 4. Vậy phải bố trí 4 đợt thi:



Hình 46

Đợt 1 thi các môn 2 và 6; Đợt 2 thi môn 4;
Đợt 3 thi các môn 3 và 7; Đợt 4 thi các môn 1 và 5

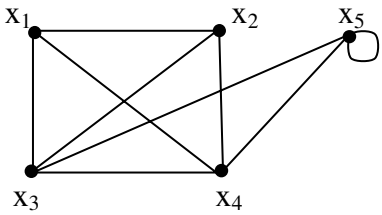
Bài toán 2: Bài toán phân chia kênh truyền hình.

Giả sử mỗi kênh truyền hình phủ sóng trong phạm vi bán kính 50 km. Đồng bằng Bắc bộ có tất cả 25 đài phát truyền hình. Phải phân chia các kênh cho các đài phát như thế nào sao cho không có 2 đài phát nào bị trùng kênh trong vùng phủ sóng của chúng.

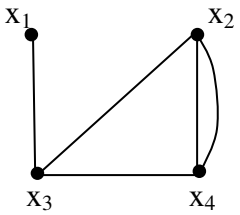
Giải: Xét đồ thị gồm 25 đỉnh, mỗi đỉnh ứng với một đài phát. Hai đài có khoảng cách từ 100 km trở xuống được nối với nhau bằng 1 cạnh. Tô màu cho đồ thị, các đỉnh cùng màu được phép phát cùng một kênh.

BÀI TẬP CHƯƠNG 3

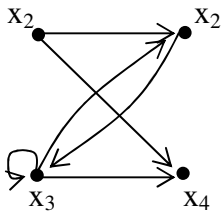
3.1. Cho các đồ thị:



G_1



G_2



G_3

a) Lập danh sách cạnh của các đồ thị.
b) Tìm ma trận kề của các đồ thị.
c) Tìm ma trận liên thuộc của các đồ thị.

3.2. Biểu diễn bằng hình học các đồ thị có ma trận kề sau:

$$A=\begin{pmatrix}0&1&0\\1&0&1\\0&1&0\end{pmatrix};\quad B=\begin{pmatrix}1&2&3\\1&0&0\\2&2&0\end{pmatrix};\quad C=\begin{pmatrix}1&2&0&1\\2&0&3&0\\0&3&1&1\\1&0&1&0\end{pmatrix};\quad D=\begin{pmatrix}0&1&2&1\\1&2&2&0\\1&1&1&3\\0&1&3&2\end{pmatrix}$$

3.3. Tính số bậc các đỉnh của các đồ thị cho trong bài tập 3.2.

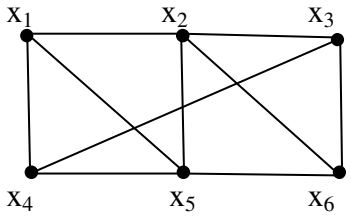
3.4. Có bao nhiêu cạnh trong một đồ thị 10 đỉnh,mỗi đỉnh đều có bậc bằng 6?

3.5. Hãy tìm trong đồ thị ở hình bên các đường đi đơn từ x_1 đến x_6 có độ dài:

a) 2; b) 3;

c) 4; d) 5;

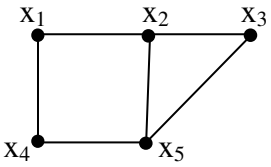
e) 6; f) 7;



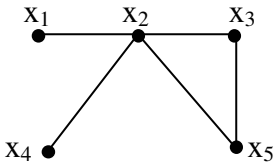
3.6. Hợp của 2 đồ thị đơn $G_1 = (X_1, U_1)$ và $G_2 = (X_2, U_2)$ là một đồ thị, ký hiệu $G_1 \cup G_2 = (X, U)$ với $X = X_1 \cup X_2$; $U = U_1 \cup U_2$.

Tìm hợp của các cặp đồ thị sau:

a)

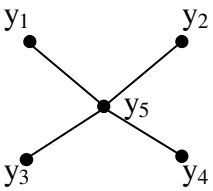


G_1

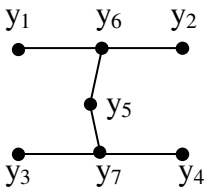


G_2

b)



G_1



G_2

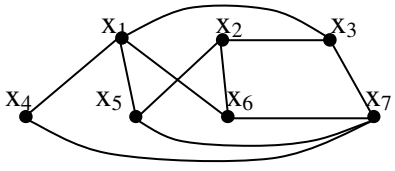
3.7. Đồ thị phần bù $\overline{G}$ của đơn đồ thị G có cùng số đỉnh số đỉnh như đồ thi G. Hai đỉnh kiền kề trong $\overline{G}$ nếu chúng không liền kề trong G.

a) Tìm $\overline{K_n}$ (K_n là đồ thị đầy đủ n đỉnh).

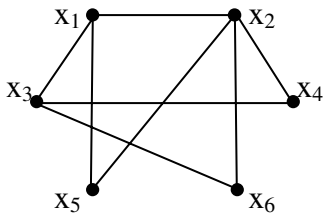
b) Nếu G có n đỉnh và m cạnh thì $\overline{G}$ có bao nhiêu cạnh.

c) Chứng tỏ rằng nếu G là đồ thị đơn có n đỉnh thì $G \cup \overline{G} = K_n$.

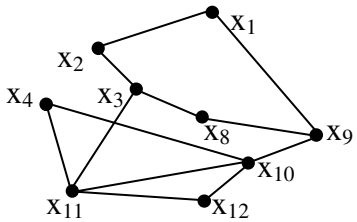
3.8. Các đồ thị trong hình dưới đây, đồ thị nào là đồ thị phân đôi? Nếu là đồ thị phân đôi hãy phân hoạch tập đỉnh thành 2 tập thuộc 2 phía và vẽ lại đồ thị đó.



G_1



G_2



G_3

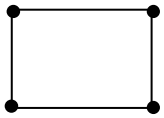
3.9. a) Chứng minh rằng nếu G là đơn đồ thị vô hướng có n đỉnh, m cạnh và k thành phần liên thông thì:

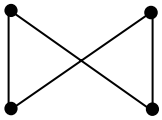
$$m \leq \frac{1}{2}(n-k)(n-k+1)$$

b) Chứng minh rằng đơn đồ thị có n đỉnh với số cạnh lớn hơn $\frac{1}{2}(n-1)(n-2)$ là liên thông.

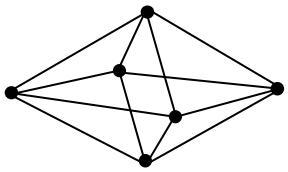
3.10. Hãy xét xem các cặp đồ thị sau có đẳng cấu không?

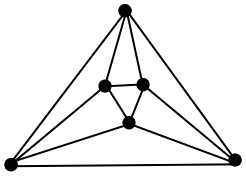
a)



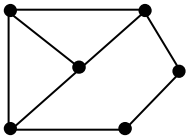


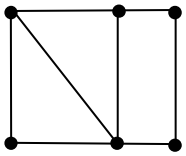
b)



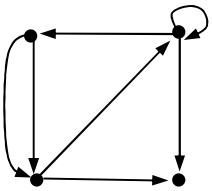


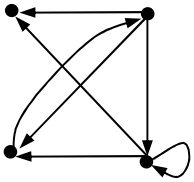
c)



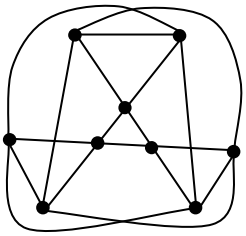


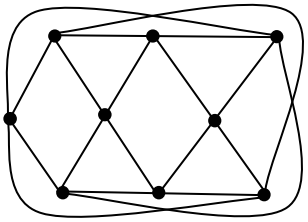
d)





e)





3.11. Các cặp đồ thị cho bằng ma trận kề sau có đẳng cấu không?

a) $\begin{pmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix};$ b) $\begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 \end{pmatrix}; \begin{pmatrix} 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 \end{pmatrix}$

3.12. Cho $X = \{2, 3, 4, 5, 6, 7, 8\}$ và

$U = \{(x,y) \mid x \in X, y \in X, x < y$ và x, y nguyên tố cùng nhau$\}$

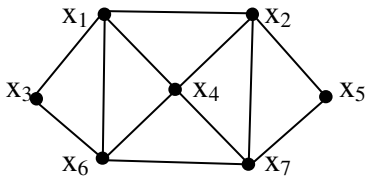
Hãy vẽ đồ thị G = (X,U) và tìm các đường đi đơn phân biệt có độ dài 3 từ đỉnh 2 tới đỉnh 8. (Xét hai trường hợp: đồ thị vô hướng và đồ thị có hướng).

3.13. a) Một cuộc họp có ít nhất ba đại biểu đến dự. Mỗi đại biểu quen với ít nhất hai đại biểu khác. Chứng minh rằng có thể xếp một số các đại biểu ngồi quanh một bàn tròn sao cho mỗi đại biểu đều ngồi giữa hai người quen.

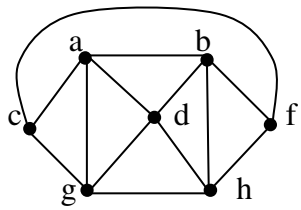
b) Một lớp học có ít nhất 4 sinh viên. Mỗi sinh viên thân với ít nhất 3 sinh viên khác. Chứng minh rằng có thể chọn xếp một số chẵn sinh viên ngồi quanh một cái bàn tròn sao cho mỗi sinh viên đều ngồi giữa hai bạn thân của mình.

3.14. Tìm số ổn định trong, số ổn định ngoài, nhân có số phần tử ít nhất và sắc số của các đồ thị sau:

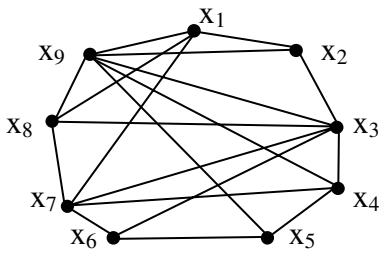
a)



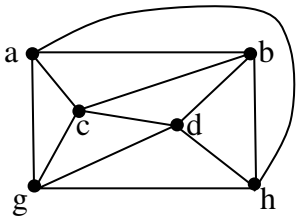
b)



c)

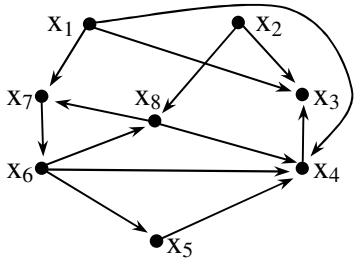


d)

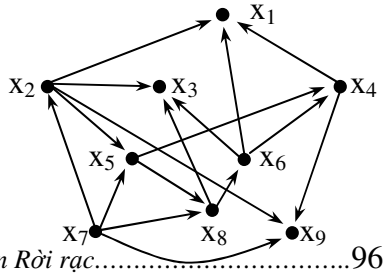


3.15. Tìm số ổn định trong, số ổn định ngoài, nhân có số phần tử ít nhất các đồ thị sau:

a)

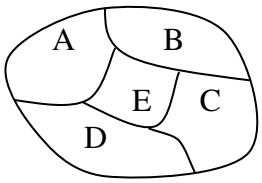


b)

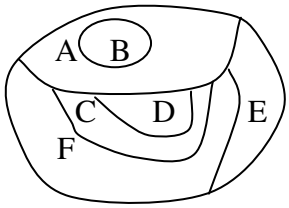


3.16. Tìm số màu cần thiết để tô 2 bản đồ sau sao cho số các vùng có chung đường biên được tô bởi các màu khác nhau.

a)



b)



3.17. Bảng dưới đây cho khoảng cách tính theo km của 6 đài truyền hình (đánh số từ 1 đến 6). Hỏi phải cần bao nhiêu kênh khác nhau để phát sóng, biết rằng mỗi đài phủ sóng trong phạm vi bán kính 50km. Hãy lập kế hoạch cụ thể để phân chia các kênh.

	1	2	3	4	5	6
1		85	75	100	50	100
2	85		125	65	100	120
3	75	125		100	90	150
4	100	65	100		45	65
5	50	100	90	45		55
6	100	120	150	65	55	

3.18. Lập lịch thi cho tình huống sau: Có 7 môn thi A, B, C, D, E, F, H và các môn có sinh viên đăng ký cùng thi là: (A, B, C); (B, C, F); (E, F); (A, D, E); (A, E, F); (B, E); (C, H); (A, C, E, F).

ĐÁP SỐ

3.8. G_1 phân đôi, G_2, G_3 không phân đôi.

3.9. a) Gợi ý: Gọi n_i, m_i tương ứng là số đỉnh và số cạnh của thành phần liên thông thứ i.

Ta có: $m = \sum_{i=1}^{k} m_i \le \sum_{i=1}^{k} C_{n_i}^2$, từ đó suy ra công thức cần chứng minh.

b) Gợi ý: Dùng phản chứng, bằng cách xét đồ thị n đỉnh không liên thông nhưng có thành phần liên thông với n_1 đỉnh ($n_1 \le n$)

3.10. a, b, e: đẳng cấu ; c, d: không đẳng cấu.

3.11. a, b: Không đẳng cấu.

3.14. a) $\alpha = 3$, $\beta = 2$, $\lambda = 3$, có 5 nhân có 2 phần tử mỗi nhân.

b) $\alpha = 2$, $\beta = 2$, $\lambda = 4$, có 7 nhân có 2 phần tử mỗi nhân.

c) $\alpha = 4$, $\beta = 2$, $\lambda = 3$, có 3 nhân có 2 phần tử mỗi nhân.

d) $\alpha = 2$ $\beta = 2$, $\lambda = 3$, có 3 nhân có 2 phần tử mỗi nhân.

3.16. a) 3 màu, b) 3 màu

3.17. 4 kênh.

CÂU HỎI ÔN TẬP CHƯƠNG 3

1. Định nghĩa đơn đồ thị, đa đồ thị, giả đồ thị, đồ thị vô hướng, đồ thị có hướng, đồ thị hỗn hợp. Cho thí dụ về cách dùng đồ thị để mô hình hoá các bài toán thực tế.

2. Định nghĩa và các tính chất của bậc của đỉnh của đồ thị.

3. Phát biểu định nghĩa và tính chất (số cạnh, bậc của các đỉnh, …) của các đồ thị: K_n, $K_{m,n}$, C_n, W_n, Q_n.

4. Định nghĩa và thuật toán nhận biết đồ thị phân đôi? Cho thí dụ.

5. Trình bày các cách biểu diễn đồ thị bằng đại số (danh sách kề, danh sách cạnh, ma trận kề, ma trận liên thuộc).

6. Thế nào là hai đồ thị đẳng cấu? Bất biến với phép đẳng cấu giữa hai đơn đồ thị là gi? Hãy cho thí dụ về hai đơn đồ thị có cùng số đỉnh, cùng số cạnh và bậc của các đỉnh là như nhau nhưng không đẳng cấu.

7. Định nghĩa và phân loại đường đi và chu trình trong đồ thị. Định nghĩa đồ thị con, đồ thị bộ phận. Khái niệm và các định lý nhận biết đồ thị liên thông, cho thí dụ để thấy các định lý chỉ là điều kiện đủ mà không phải là điều kiện cần.

8. Định nghĩa tập ổn định trong, tập ổn định ngoài và nhân của đồ thị. Nêu thuật toán tìm tập ổn định ngoài cực tiểu của đồ thị. Định nghĩa số ổn định trong, số ổn định ngoài của đồ thị.

9. Định nghĩa sắc số của đồ thị. Trình bày các định lý về sắc số và thuật toán tìm sắc số của đồ thị. Nêu một vài bài toán thực tế áp dụng sắc số.

CHƯƠNG 4

ĐỒ THỊ EULER. ĐỒ THỊ HAMILTON. ĐỒ THỊ PHẲNG

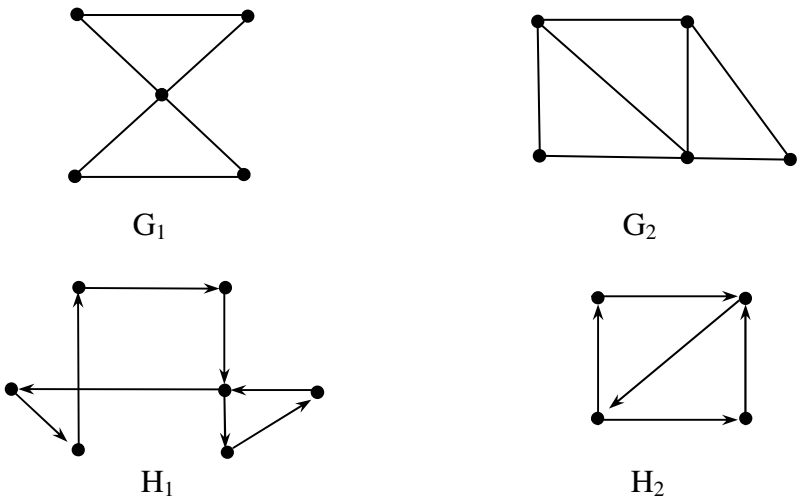
1. Đồ thị Euler

1.1. Định nghĩa: Xét đồ thị vô hướng hoặc có hướng G = (X, U).

Chu trình đơn (chu trình chỉ đi qua mỗi cạnh đúng một lần) đi qua tất cả các cạnh của đồ thị gọi là chu trình Euler. Đường đi Euler là đường đi đơn đi qua tất cả các cạnh của đồ thị.

Đồ thị có chu trình Euler gọi là đồ thị Euler. Đồ thị có đường đi Euler, nhưng không có chu trình Euler gọi là đồ thị nửa Euler.

Thí dụ: Các đồ thị G_1 và H_1 là các đồ thị Euler. Các đồ thị G_2 và H_2 là các đồ thị nửa Euler (Hình 1).



Hình 1. Các đồ thị Euler (G_1, H_1) và nửa Euler (G_2, H_2)

1.2. Nhận biết đồ thị Euler

Định lý 1: Điều kiện cần và đủ để đồ thị vô hướng liên thông G là đồ thị Euler là mọi đỉnh của G đều có bậc chẵn.

Chứng minh:

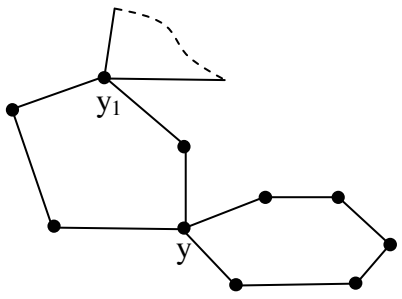
Điều kiện cần

Giả sử G là đồ thị Euler, tức là G có chu trình Euler. Vì cứ mỗi lần chu trình Euler đi qua một đỉnh thì số bậc của đỉnh đó tăng thêm 2. Hơn nữa chu trình Euler là đi qua tất cả các cạnh đúng một lần. Vậy mọi đỉnh của G đều có bậc chẵn.

Điều kiện đủ

Theo định lý 1 trong 4.1 chương 3, nếu đồ thị G có $\deg(x) \geq 2$ với mọi đỉnh x thì trong G có chu trình đơn. Gọi chu trình đơn đó là C, nếu C đi qua mọi cạnh của đồ thị thì C là chu trình Euler, định lý được chứng minh. Nếu C không đi qua tất cả các cạnh, khi đó loại khỏi G tất cả các cạnh của chu trình C và các đỉnh biệt lập nếu nó xuất hiện sẽ được đồ thị con G_1 của đồ thị G (G_1 có thể có nhiều thành phần liên thông). Đồ thị G_1 có các tính chất sau:

- G_1 có đỉnh chung với chu trình C vừa bị loại. Điều này là hiển nhiên vì đồ thị G liên thông. Giả sử đỉnh chung đó là y.
- Mọi đỉnh của G_1 đều có bậc chẵn. Vì, nếu đỉnh của G_1 không phải là đỉnh chung với chu trình C thì bậc của nó là chẵn theo giả thiết của định lý, nếu đỉnh của G_1 là đỉnh chung với C thì vì đã loại khỏi đỉnh đó 2 cạnh liên thuộc nên bậc của đỉnh đó giảm đi 2 do đó nó vẫn là đỉnh bậc chẵn.



Hình 2

Do đó trong G_1 lại tồn tại một chu trình đơn C_1, chu trình này được nối với chu trình C qua đỉnh chung y thành một chu trình đơn lớn hơn.

Nếu chu trình lớn này đi qua tất các cạnh của G thì nó là chu trình Euler, định lý được chứng minh. Nếu chu trình lớn này chưa phải là chu trình Euler thì lặp lại quá trình loại bỏ như trên.Vì G là đồ thị hữu hạn nên sau một số hữu hạn bước sẽ tìm được chu trình Euler của đồ thị G.

Hình 2 minh hoạ quá trình chứng minh trên.

Từ cách chứng minh định lý suy ra thuật toán (thuật toán Fleury) tìm chu trình Euler của đồ thị G = (X, U) như sau:

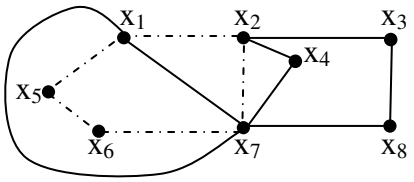
Bước 1. Kiểm tra tính liên thông của đồ thị. Nếu đồ thị không liên thông thì kết luận đồ thị không có chu trình Euler.

Bước 2. Tính deg(x), $\forall x \in X$. Nếu có deg(x) lẻ thì đồ thị không có chu trình Euler.

Bước 3. Xuất phát từ một đỉnh bất kỳ của đồ thị và đi theo các cạnh để tìm một chu trình đơn, khi đi qua một cạnh nào đó thì xoá cạnh đó và các đỉnh cô lập (nếu có) khỏi đồ thị, chú ý là không đi qua một cầu trừ khi không còn cách nào khác. Nếu chưa hết các cạnh thì tiến hành tìm tiếp trên phần đồ thị còn lại sau đó ghép các chu trình đó lại với nhau.

Có thể gọi thuật toán trên là thuật toán vừa đi vừa xoá.

Thí dụ 1: Xét đồ thị ở hình 3.



Hình 3

Kiểm tra tính liên thông và bậc các đỉnh là thoả mãn.

Chẳng hạn lần đầu tìm được chu trình đơn $x_1\ x_2\ x_7\ x_6\ x_5\ x_1$ (các cạnh gạch - chấm).

Tiếp tục tìm lần hai được chu trình đơn $x_1\ x_7\ x_8\ x_3\ x_2\ x_4\ x_7\ x_1$.

Nối hai chu trình tìm được chu trình Euler: $x_1x_2x_7x_6x_5x_1x_7x_8x_3x_2x_4x_7x_1$.

Thí dụ 2: Tìm chu trình Euler (nếu có) của đồ thị cho bằng ma trận kề:

$$\begin{array}{c} x \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{array} \begin{array}{c} \begin{array}{cccccc} 1 & x_2 & x_3 & x_4 & x_5 & x_6 \end{array} \\ \begin{pmatrix} 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 \end{pmatrix} \end{array}$$

Ta có: $\deg(x_1) = \deg(x_4) = \deg(x_5) = \deg(x_6) = 2$; $\deg(x_2) = \deg(x_3) = 4$. Do đó đồ thị đã cho là đồ thị Euler.

Áp dụng thuật toán Fleury, vừa đi vừa xóa được chu trình Euler: $x_1x_2x_3x_4x_2x_5x_6x_3x_1$.

1.3. Nhận biết đồ thị nửa Euler

Định lý 2: Điều kiện cần và đủ để một đồ thị vô hướng liên thông là đồ thị nửa Euler là đồ thị đó có đúng 2 đỉnh bậc lẻ.

Chứng minh: Giả sử x, y là 2 đỉnh bậc lẻ của đồ thị đã cho. Thêm vào đồ thị cạnh (x, y) được đồ thị mới có bậc của tất cả các đỉnh đều chẵn. Theo định lý 1 đồ thị mới này có chu trình Euler, bỏ cạnh (x, y) mới thêm được đường đi Euler. Định lý được chứng minh.

Chú ý: 1- Theo cách chứng minh định lý 2, đường đi Euler bao giờ cũng bắt đầu từ một trong hai đỉnh bậc lẻ và kết thúc ở đỉnh bậc lẻ còn lại.

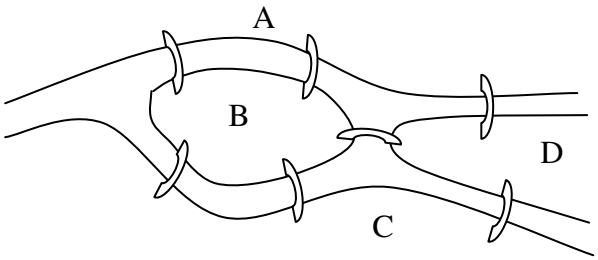
2- Từ các định lý 1 và 2 suy ra cách nhận biết một hình vẽ có thể vẽ được bằng một nét bút. Đó là các hình khi mô hình hoá thành đồ thị thì có không quá 2 đỉnh có bậc lẻ.

Thí dụ: Bài toán bảy cây cầu ở Koenigsberg.

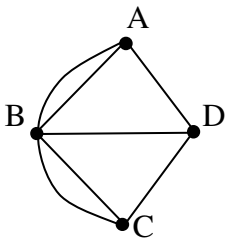
Con sông Pregel ở thành phố Koenigsberg (thuộc nước Đức, thế kỷ 18) chia thành phố thành 4 vùng như hình 4a. Người ta đã xây dựng 7 cây cầu nối các vùng đất của thành phố với nhau. Hỏi có thể đi một lượt qua hết cả 7 cây cầu đó rồi trở về điểm xuất phát sao cho mỗi cây cầu chỉ đi qua đúng một lần?

Bài toán này được giải lần đầu tiên vào năm 1766 bởi Euler. Ông đã chứng minh là không có cách đi nào thoả mãn yêu cầu của bài toán.

Có thể mô hình hoá các vùng đất là các đỉnh và các cây cầu là các cạnh nối các đỉnh và được đồ thị như hình 4b. Đồ thị này có cả 4 đỉnh đều có bậc lẻ, vì thế không có cách đi nào thoả mãn đòi hỏi của bài toán.



Hình 4a



Hình 4b

1.4. Bài toán người đưa thư Trung hoa

Một bưu tá nhận thư ở bưu điện và phải đi qua một số phố để phát thư rồi quay về bưu điện. Bưu tá đó phải đi theo một hành trình như thế nào để đường đi là ngắn nhất? (Giả thiết rằng mọi con phố người đó phải đi qua đều dài như nhau)

Bài toán này được nhà Toán học Trung quốc Guan nêu lên lần đầu tiên vào năm 1960. Bởi vậy nó có tên là "Bài toán người đưa thư Trung hoa". Có thể phát biểu bài toán dưới dạng đồ thị:

Cho một đơn đồ thị vô hướng liên thông G. Một chu trình đi qua mọi cạnh của G được gọi là một hành trình. Hãy tìm một hành trình ngắn nhất, tức là hành trình đi qua ít cạnh nhất.

Rõ ràng rằng nếu G là đồ thị Euler (mọi đỉnh đều có bậc chẵn) thì chu trình Euler trong G (qua mỗi cạnh của G đúng một lần) là hành trình ngắn nhất cần tìm.

Chỉ còn phải xét trường hợp G có một số đỉnh bậc lẻ (số đỉnh bậc lẻ là một số chẵn). Khi đó, mọi hành trình trong G phải đi qua ít nhất hai lần một số cạnh nào đó.

Dễ thấy rằng một hành trình qua một cạnh (x,y) nào đó quá hai lần thì không phải là hành trình ngắn nhất trong G. Vì vậy, chỉ cần xét những hành trình T đi qua hai lần một số cạnh nào đó của G.

Quy ước rằng mỗi hành trình T trong G là một chu trình Euler trong đồ thị Euler G_T, trong đó G_T có được từ G bằng cách vẽ thêm một số cạnh song song đối với những cạnh mà T đi qua hai lần. Bài toán đặt ra được đưa về bài toán sau:

Trong các đồ thị Euler G_T của một đơn đồ thị G, tìm đồ thị có số cạnh ít nhất (khi đó chu trình Euler trong đồ thị này là hành trình ngắn nhất).

Chúng ta thừa nhận định lý sau:

Định lý: (Goodman và Hedetniemi, 1973) Nếu G là một đồ thị liên thông có m cạnh thì hành trình ngắn nhất trong G có chiều dài là:

$$m + m(G)$$

trong đó m(G) là số ít nhất các cạnh mà hành trình đi qua hai lần và được xác định như sau:

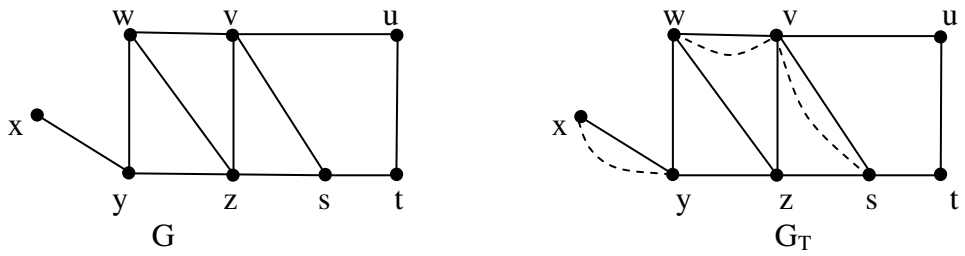
Gọi $X_0(G)$ là tập hợp các đỉnh bậc lẻ của G, khi đó tập $X_0(G)$ có 2k phần tử. Tiến hành phân hoạch tập $X_0(G)$ thành k cặp, mỗi phân hoạch như vậy gọi là một phân hoạch cặp của $X_0(G)$. Gọi P là tập hợp mọi phân hoạch cặp có thể của $X_0(G)$.

Ký hiệu độ dài đường đi ngắn nhất giữa đỉnh x và đỉnh y là d(x,y), nghĩa là từ đỉnh x đến đỉnh y có d cạnh. Với mọi phân hoạch cặp $P_i \in P$, đặt:

$$d(P_i) = \sum_{(x,y)\in P_i} d(x,y)$$

Khi đó: $$m(G) = \min \{d(P_i) \mid P_i \in P\}$$

Thí dụ: Giải bài toán người đưa thư Trung Hoa cho trong đồ thị G ở hình 5.



Hình 5

Đồ thị có 8 đỉnh 11 cạnh (n = 8; m = 11). Tập hợp các đỉnh bậc lẻ $X_0(G) = \{x, y, s, w\}$ và tập hợp các phân hoạch cặp là $P = \{P_1, P_2, P_3\}$, trong đó:

$P_1 = \{(x, y), (s, w)\} \rightarrow d(P_1) = d(x, y) + d(s, w) = 1 + 2 = 3,$

$P_2 = \{(x, s), (y, w)\} \rightarrow d(P_2) = d(x, s) + d(y, w) = 3 + 1 = 4,$

$P_3 = \{(x, w), (y, s)\} \rightarrow d(P_3) = d(x, w) + d(y, t) = 2 + 2 = 4.$

$m(G) = \min\{d(P_1), d(P_2), d(P_3)\} = d(p_1) = 3.$

Do đó G_T có được từ G bằng cách thêm vào 3 cạnh (x, y), (s, v), (v, w) và G_T là đồ thị Euler. Một trong các hành trình ngắn nhất cần tìm là:

x, y, z, s, v, u, t, s, v, w, z, v, w, y, x.

Tổng các cạnh người đưa thư phải đi (chiều dài hành trình) là 11 + 3 = 14.

Chú ý: Cũng có thể thêm vào G các cạnh (x,y), (w,z), (z,s) để được đồ thị Euler G_T.

1.5. Chú thích

Đối với đồ thị Euler có hướng, có các kết luận hoàn toàn tương tự:

- Đồ thị có hướng liên thông G là đồ thị Euler khi và chỉ khi mọi đỉnh của G đều có bậc vào bằng bậc ra.
- Đồ thị có hướng liên thông G là nửa Euler (mà không là Euler) khi và chỉ khi tồn tại một đỉnh có bậc vào lớn hơn bậc ra một đơn vị và một đỉnh có bậc ra lớn hơn bậc vào một đơn vị. Nghĩa là tồn tại hai đỉnh x và y sao cho :

$\deg^+(x) = \deg^-(x)+1$, $\deg^-(y) = \deg^+(y)+1$, $\deg^+(v) = \deg^-(v)$, $\forall v \in V$, $v \neq x$, $v \neq y$.

2. Đồ thị Hamilton

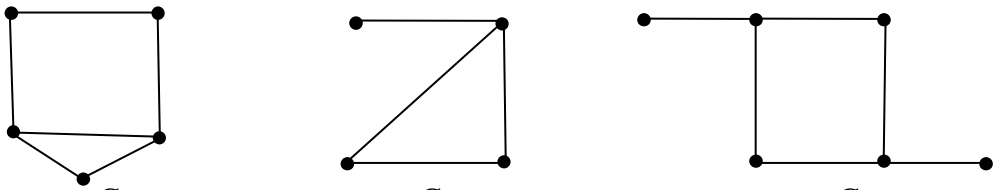
2.1. Định nghĩa: Xét đồ thị vô hướng hoặc có hướng G = (X, U).

Đường đi đơn đi qua tất cả các đỉnh của đồ thị, mỗi đỉnh đúng một lần gọi là đường đi Hamilton.

Chu trình đơn đi qua tất cả các đỉnh của đồ thị, mỗi đỉnh đúng một lần gọi là chu trình Hamilton.

Đồ thị có chu trình Hamilton gọi là đồ thị Hamilton. Đồ thị không có chu trình Hamilton nhưng có đường đi Hamilton gọi là đồ thị nửa Hamilton.

Thí dụ: G_1 là đồ thị Hamilton, G_2 là đồ thị nửa Hamilton, G_3 không phải là đồ thị Hamilton cũng không phải là đồ thị nửa Hamilton (hình 6).



Hình 6

Dễ nhận thấy, mọi đồ thị có đỉnh treo (đỉnh có bậc bằng 1) đều không phải là đồ thị Hamilton.

2.2. Nhận biết đồ thị Hamilton

Hiện tại chưa có tiêu chuẩn chặt chẽ nào để nhận biết đồ thị Hamilton. Người ta mới chỉ chứng minh được một vài định lý với các điều kiện quá rộng để nhận biết đồ thị Hamilton. Sau đây là một số định lý như vậy.

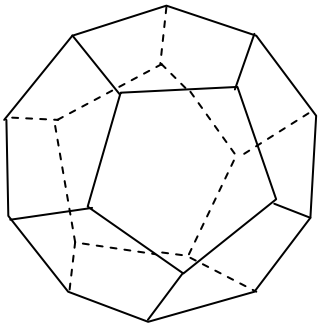
Định lý 1: Đơn đồ thị vô hướng liên thông mà mọi đỉnh đều có bậc không nhỏ hơn nửa số đỉnh là đồ thị Hamilton.

Hệ quả: Đơn đồ thị vô hướng liên thông G có n đỉnh, nếu mọi đỉnh của đồ thị đều có bậc không nhỏ hơn $\frac{n-1}{2}$ thì G là đồ thị nửa Hamilton.

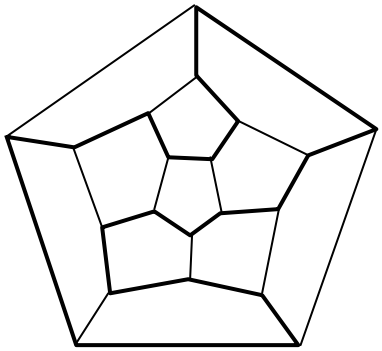
Thật vậy, thêm vào G một đỉnh x và nối x với mọi đỉnh của G nhận được đồ thị G' có n+1 đỉnh và bậc của mỗi đỉnh không nhỏ hơn $\frac{n-1}{2}+1=\frac{n+1}{2}$. Theo định lý 1, G' là đồ thi Hamilton, loại đỉnh x ra khỏi chu trình Hamilton trong G' được đường Hamilton trong G.

Có thể thấy các điều kiện đã nêu trong định lý trên là quá rộng. Đồ thị G_1 trong hình 6 là một minh chứng.

Xét thêm một thí dụ nữa gọi là bài toán du lịch vòng quanh thế giới do chính Hamilton đặt ra (Hamilton là nhà toán học Ailen): Có một khách muốn đi du lịch qua 20 thành phố trên thế giới. Các thành phố này được biểu diễn bằng các đỉnh của một khối 12 mặt đều, mỗi mặt là một ngũ giác đều, các cạnh của khối đa diện biểu hiện đường đi giữa các thành phố (hình 7a). Liệu có cách nào để người khách đó xuất phát từ một thành phố bất kỳ và đi một lượt hết cả 20 thành phố, mỗi thành phố đi qua đúng một lần rồi trở về thành phố xuất phát?



Hình 7a



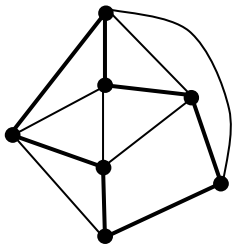
Hình 7b

Có thể mô hình hoá bài toán bằng một đồ thị phẳng 20 đỉnh (Hình 7b), một trong các hành trình của người khách là chu trình Hamilton như đã chỉ trong hình 7b bằng nét đậm.

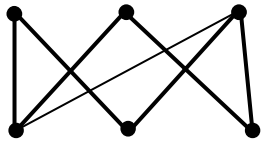
Trong thí dụ này, mỗi đỉnh của đồ thị đều có bậc bằng 3, trong khi đó nửa số đỉnh của đồ thị là 10.

Định lý 2: Nếu G là một đơn đồ thị vô hướng có tổng số bậc của hai đỉnh tùy ý không kề nhau không nhỏ hơn số đỉnh của đồ thị thì G là đồ thị Hamilton.

Thí dụ: Đồ thị ở hình 8 có 5 đỉnh bậc 4 và 2 đỉnh bậc 3 kề nhau nên tổng số bậc của hai đỉnh bất kỳ không kề nhau là bằng 7 hoặc 8, do đó nó là đồ thị Hamilton.



Hình 8



Hình 9

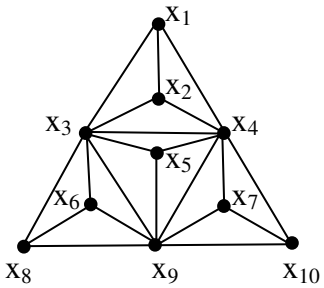
Định lý 3: Đồ thị phân đôi với hai tập đỉnh X_1 và X_2, mỗi tập đỉnh đều có số đỉnh là n ($n \geq 2$) và bậc của mỗi đỉnh đều lớn hơn $\frac{n}{2}$ là đồ thị Hamilton.

Đồ thị ở hình 9 là một thí dụ minh họa cho định lý này.

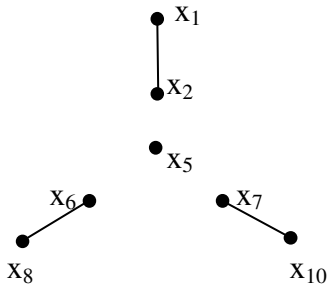
Định lý 4: Nếu có thể xóa đi k đỉnh cùng các cạnh liên thuộc của một đơn đồ thị G liên thông mà được đồ thị con có nhiều hơn k thành phần liên thông thì đồ thị G không phải là đồ thị Hamilton.

Chứng minh: Điều này là hiển nhiên, vì khi xóa đi k đỉnh cùng các cạnh liên thuộc trong một chu trình thì chu trình đó bị chia thành nhiều nhất là k thành phần liên thông.

Thí dụ: Xét đồ thị trong hình 10a. Xóa các đỉnh x_3, x_4, x_9 cùng các cạnh liên thuộc được đồ thị con có 4 thành phần liên thông (hình 10b). Vậy đồ thị đã cho không phải là đồ thị Hamilton.



Hình 10a.



Hình 10b.

Định lý 5: Đơn đồ thị có hướng liên thông mà mọi đỉnh có bán bậc ra và bán bậc vào đều không nhỏ hơn nửa số đỉnh là đồ thị Hamilton.

Một đồ thị có hướng được gọi là đồ thị có hướng đầy đủ nếu với hai đỉnh phân biệt bất kỳ x và y phải có một và chỉ một trong 2 cung (x,y) hoặc (y,x).

Định lý 6: Đồ thị có hướng đầy đủ là đồ thị nửa Hamilton.

Thí dụ: Chứng tỏ rằng khi tổng kết cuộc thi đấu bóng bàn của n đấu thủ ($n \geq 2$) theo thể thức vòng tròn một lượt, luôn luôn có thể xếp n đấu thủ đó thành một hàng dọc sao cho người đứng trước thắng người đứng ngay sau.

Theo mô hình đồ thị thi đấu vòng tròn (thí dụ 2, mục 1.1, chương 3) chúng ta có một đồ thị có hướng đầy đủ n đỉnh ứng với n đấu thủ, theo định lý 6 có thể xếp n đấu thủ này thành một đường Hamilton, nghĩa là bài toán được giải.

2.3. Quy tắc tìm chu trình Hamilton.

Nhiều khi việc vận dụng các định lý đã nêu trong mục 2.2 không đủ cho phép kết luận một đồ thị đã cho có phải là đồ thị Hamilton hay nửa Hamilton hay không. Khi đó nên trực tiếp tìm chu trình (đường) Hamilton theo các quy tắc nêu sau đây.

Vì mỗi đỉnh của chu trình Hamilton đều liền kề với đúng 2 cạnh nên suy ra:

1. Nếu đồ thị G = (X,U) có dù chỉ một đỉnh có bậc không lớn hơn 1 ($\exists x \in X, \deg(x) \leq 1$) thì G không phải là đồ thị Hamilton.
2. Nếu đỉnh x có bậc là 2 thì cả hai cạnh kề với x đều thuộc chu trình Hamilton.

3. Trong khi xây dựng chu trình Hamilton, sau khi đã lấy 2 cạnh kề với một đỉnh x nào đó để đặt vào chu trình Hamilton thì không thể lấy thêm bất kỳ cạnh nào kề với x nữa, vì thế có thể xóa mọi cạnh còn lại kề với x.
4. Chu trình Hamilton không chứa bất kỳ chu trình con nào.

Thí dụ: Đồ thị trong hình 11a có phải là đồ thị Hamilton hay không?

Giả sử đồ thị là đồ thị Hamilton, gọi C chu trình Hamilton của đồ thị.

Vì deg(a) = deg(g) = 2 nên các cạnh (a,b), (a,c), (g,e), (g,i) phải thuộc C (quy tắc 2).

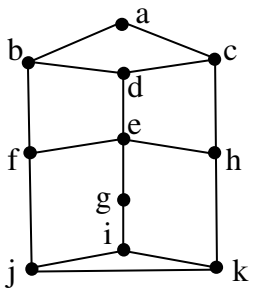
Xét đỉnh i, do (i,g) đã được chọn nên chỉ có thể chọn một trong hai cạnh (i,j) hoặc (i,k) vào chu trình C. Do tính đối xứng của đồ thị nên chọn cạnh nào cũng được, chẳng hạn chọn cạnh (i,k), vậy phải xóa cạnh (i,j) (quy tắc 3) còn lại đồ thị ở hình 11b.

Bây giờ đỉnh j có bậc bằng 2, vậy (k,j)∈C và (f,j)∈C (quy tắc 2).

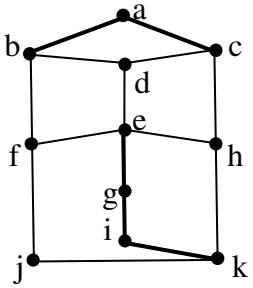
Xóa tiếp các cạnh (k,h) (theo quy tắc 3) và (e,f) (quy tắc 4). Hình 11c là đồ thị còn lại. Dễ thấy (f,b)∈C (quy tắc 2). Còn các cạnh (b,d), (c,d), (c,h), (e,d), (e,h) không có cách nào chọn vào C mà không tạo thành chu trình con.

Vậy đồ thị đã cho không có chu trình Hamilton.

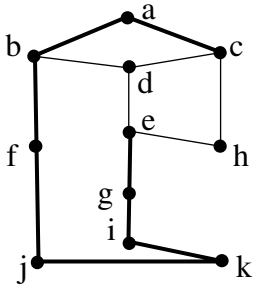
Chú ý rằng đồ thị có đường Hamilton, đó là đường đi qua các đỉnh dcabfjkigeh.



Hình 11a



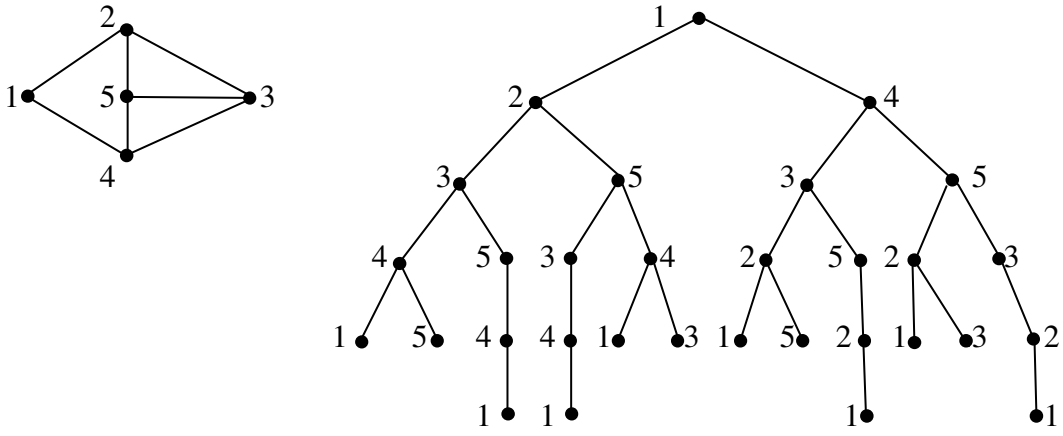
Hình 11b



Hình 11c

2.4. Cây liệt kê chu trình Hamilton

Có thể liệt kê các chu trình Hamilton của một đồ thị đã cho nhờ thuật toán quay lui. Hình 12 là cây liệt kê các chu trình Hamilton của đồ thị đã cho.



Hình 12

Đồ thị đã cho có 4 chu trình Hamilton, đó là: 1, 2, 3, 5, 4, 1; 1, 2, 5, 3, 4, 1; 1, 4, 3, 5, 2, 1 và 1, 4, 5, 3, 2, 1.

Khi dùng cây liệt kê chu trình Hamilton nên chọn đỉnh có bậc thấp nhất làm gốc của cây.

Với các đồ thị không có quá nhiều cạnh, dùng cây liệt kê có thể nhận biết được đồ thị đã cho có phải là đồ thị Hamilton hay không.

2.5. Bài toán sắp xếp chỗ ngồi:

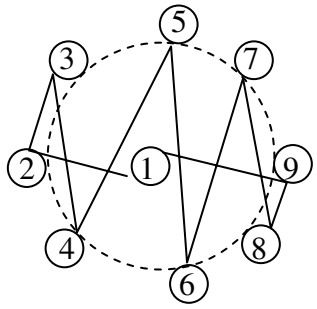
Có n đại biểu từ n nước đến dự một hội nghị quốc tế. Mỗi ngày họp một lần ngồi quanh một bàn tròn. Hỏi phải bố trí bao nhiêu ngày và bố trí như thế nào sao cho trong mỗi ngày, mỗi người có hai người kế bên là bạn mới. Lưu ý rằng n người đều muốn làm quen với nhau.

Xét đồ thị gồm n đỉnh, mỗi đỉnh ứng với mỗi người dự hội nghị, hai đỉnh kề nhau khi hai đại biểu tương ứng muốn làm quen với nhau. Đồ thị đầy đủ này là Hamilton và rõ ràng mỗi chu trình Hamilton là một cách sắp xếp thoả mãn yêu cầu đặt ra. Bài toán lúc này trở thành bài toán tìm các chu trình Hamilton phân biệt của đồ thị đầy đủ K_n (hai chu trình Hamilton gọi là phân biệt nếu chúng không có cạnh chung).

Định lý: Đồ thị đầy đủ K_n với n lẻ và $n \geq 3$ có đúng $\frac{n-1}{2}$ chu trình Hamilton phân biệt.

Chứng minh: K_n có $\frac{n(n-1)}{2}$ cạnh và mỗi chu trình Hamilton có n cạnh, nên số chu trình Hamilton phân biệt nhiều nhất là $\frac{n-1}{2}$.

Giả sử các đỉnh của K_n là 1, 2, ..., n. Đặt đỉnh 1 tại tâm của một đường tròn và các đỉnh 2, ..., n đặt cách đều nhau trên đường tròn (mỗi cung là $\frac{360^0}{n-1}$ sao cho đỉnh lẻ nằm ở một nửa đường tròn và đỉnh chẵn nằm ở nửa đường tròn còn lại. Khi đó chu trình Hamilton đầu tiên là 1, 2, ..., n, 1. Các đỉnh được giữ cố định, xoay khung theo chiều kim đồng hồ với các góc quay:



Hình 13

$$\frac{360^0}{n-1},\quad 2\frac{360^0}{n-1},\quad 3\frac{360^0}{n-1},\ ...,\quad \frac{n-3}{2}\frac{360^0}{n-1}.$$

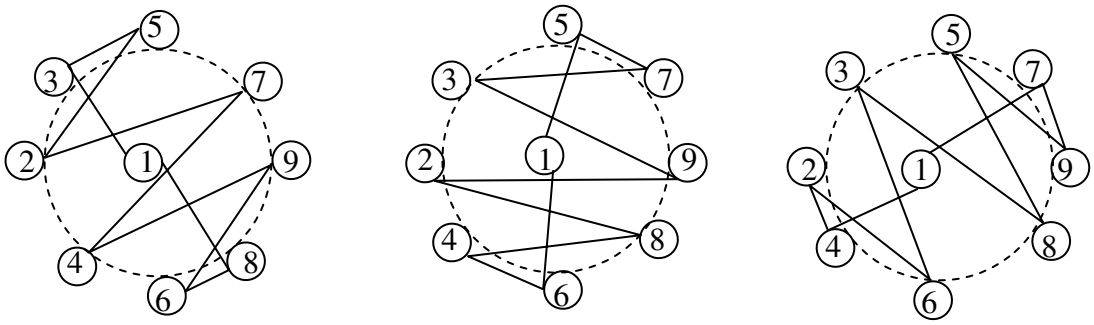
ta nhận được $\frac{n-3}{2}$ khung phân biệt với khung đầu tiên. Do đó có đúng $\frac{n-1}{2}$ chu trình Hamilton phân biệt.

Thí dụ: Giải bài toán sắp xếp chỗ ngồi với n = 9.

Có $\frac{9-1}{2}=4$ cách sắp xếp chỗ ngồi phân biệt như sau (hình 13 và hình 14):

1 2 3 4 5 6 7 8 9 1
1 3 5 2 7 4 9 6 8 1

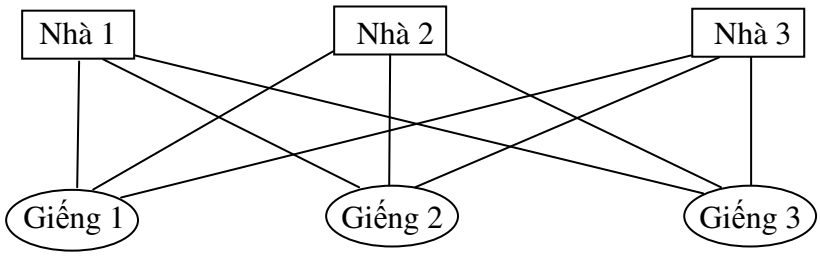
1 5 7 3 9 2 8 4 6 1
1 7 9 5 8 3 6 2 4 1



Hình 14

3. Đồ thị phẳng:

Trong phần này chỉ xét các đồ thị vô hướng. Trước hết xét bài toán cổ "ba nhà, ba giếng" như sau: Có 3 ngôi nhà và 3 cái giếng. Mỗi nhà đều có quyền dùng nước ở cả 3 giếng và có quyền làm đường riêng để đi đến từng giếng (hình 15). Có cách làm nào sao cho các đường đi không cắt nhau?



Hình 15. Bài toán "3 nhà, 3 giếng"

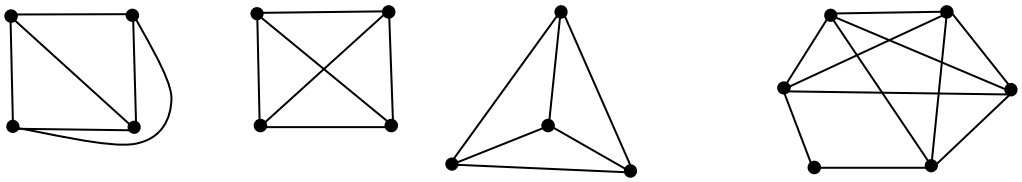
Bài toán được mô hình hóa bằng một đồ thị đầy đủ $K_{3,3}$. Và câu hỏi được đặt ra là: liệu có cách nào vẽ đồ thị đầy đủ $K_{3,3}$ trên mặt phẳng sao cho mọi cặp cạnh bất kỳ của đồ thị không cắt nhau ngoại trừ tại đỉnh của đồ thị.

Để giải quyết các bài toán như vậy, người ta đưa ra khái niệm "đồ thị phẳng".

3.1. Định nghĩa

Đồ thị G = (X, U) được gọi là đồ thị phẳng nếu tồn tại ít nhất một dạng biểu diễn hình học của nó trên mặt phẳng sao cho các cạnh chỉ cắt nhau ở đỉnh. Cách biểu diễn như vậy được gọi là **biểu diễn phẳng** của đồ thị.

Thí dụ:



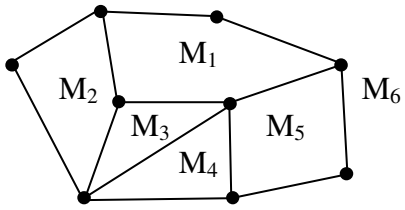
Đồ thị phẳng (ba cách vẽ của đồ thị K_4) Đồ thị không phẳng

Hình 16.

Đồ thị phẳng có vai trò rất lớn trong thiết kế các mạch in.

3.2 Công thức Euler

Biểu diễn phẳng của một đồ thị phẳng sẽ chia mặt phẳng thành các miền khác nhau, trong đó có một miền không bị chặn. Thí dụ đồ thị ở hình 17 chia mặt phẳng thành 5 miền M_1, M_2, M_3, M_4, M_5, M_6, trong đó M_6 là miền vô hạn.



Hình 17

Định lý 1 (Định lý Euler):

Nếu một đồ thị phẳng liên thông có n đỉnh, m cạnh và f miền thì:

$$f = m - n + 2$$

Công thức nêu trong kết luận của định lý được gọi là công thức Euler.

Chứng minh: Định lý được chứng minh bằng quy nạp.

Trước hết xây dựng một dãy các đồ thị con $G_1, G_2, \ldots, G_m$ của một đồ thị phẳng, liên thông G bằng cách lấy G_1 là một cạnh tùy ý của G. G_k nhận được từ G_{k-1} bằng cách thêm một cạnh tùy ý liên thuộc với một đỉnh của G_{k-1}, đỉnh còn lại của cạnh mới thêm có thể là một đỉnh của G_{k-1} hoặc là một đỉnh mới nếu nó chưa có trong G_{k-1}. Điều này luôn luôn thực hiện được vì G là liên thông và cuối cùng nhận được G khi đã ghép đủ m cạnh, nghĩa là $G_m \equiv G$. Gọi n_k, m_k và f_k tương ứng là số đỉnh, số cạnh và số miền của G_k.

Với k = 1, khi đó G_1 có: $n_1 = 2$, $m_1 = 1$, $f_1 = 1$. Rõ ràng: $m_1 - n_1 + 2 = 1 - 2 + 2 = f_1$. Công thức Euler đúng với k = 1.

Giả sử công thức đúng với k, xét với k+1, có hai trường hợp xảy ra:

a) Cả hai đỉnh x_{k+1} và y_{k+1} đã thuộc G_k (hình 18a). Khi đó cả hai đỉnh này phải nằm trên biên chung của một miền M nào đó, vì nếu không thì không thể gộp cạnh (x_{k+1}, y_{k+1}) vào G_k mà không có các cạnh cắt nhau (G_{k+1} là đồ thị phẳng).

Cạnh mới sẽ chia miền M thành 2 miền con. Do đó $f_{k+1} = f_k + 1$, $n_{k+1} = n_k$, $m_{k+1} = m_k + 1$ và:

$$m_{k+1} - n_{k+1} + 2 = (m_k + 1) - n_k + 2 = (m_k - n_k + 2) + 1 = f_k + 1 = f_{k+1}.$$

Vậy công thức đúng với k+1.

a) b)

Hình 18. Thêm một cạnh vào G_k để được G_{k+1}

b) Một trong hai đỉnh của cạnh mới chưa thuộc G_k. Khi đó đỉnh không thuộc G_k phải nằm trong miền nào đó của G_k và đỉnh thuộc G_k phải nằm trên biên của G_k (hình 18b). Như vậy số miền không tăng, nghĩa là $f_{k+1} = f_k$ còn $m_{k+1} = m_k+1$, $n_{k+1} = n_k +1$ do đó:

$$m_{k+1} – n_{k+1} + 2 = (m_k+1) – (n_k +1) + 2 = m_k – n_k +2 = f_k = f_{k+1}.$$

Vậy công thức đúng với k+1.

Định lý được chứng minh.

Thí dụ: Một đồ thị phẳng có 20 đỉnh và bậc của mỗi đỉnh đều bằng 3. Hỏi biểu diễn phẳng của đồ thị sẽ chia mặt phẳng thành bao nhiêu miền?

Trước hết tính số cạnh của đồ thị, theo tính chất bậc của các đỉnh đồ thị (mục 1.2, chương 3), ta có:

$$2m = 20.3 = 60 \Rightarrow m = 30$$

Vậy, theo công thức Euler, số miền của mặt phẳng được tạo thành bởi đồ thị là:

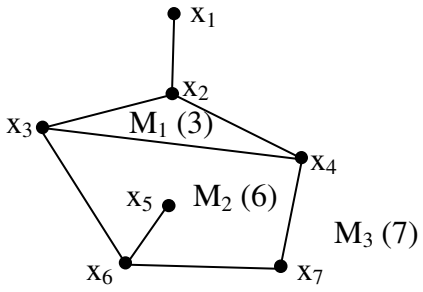
$$f = 30 – 20 + 2 = 12.$$

Hệ quả 1: Nếu G là một đơn đồ thị phẳng liên thông với m cạnh, n đỉnh trong đó $n \geq 3$ thì:

$$m \leq 3n – 6$$

Chứng minh:

Trước hết tìm hiểu khái niệm **Bậc của miền**. Người ta định nghĩa như sau: bậc của miền là số cạnh trên biên (biên của miền là chu trình bao quanh miền đó) của miền ấy. Nếu một cạnh được vẽ hai lần khi vẽ biên nghĩa là cạnh đó xuất hiện 2 lần nên nó được tính 2 đơn vị vào bậc của miền đó. Ký hiệu bậc của miền M là deg(M). Hình 19 cho biết bậc của các miền tương ứng(số ghi cạnh tên miền để trong dấu ngoặc đơn).



Hình 19

Bây giờ chứng minh hệ quả. Giả sử đơn đồ thị phẳng đã cho chia mặt phẳng thành f miền.Bậc của mỗi miền ít nhất bằng 3 (vì đơn đồ thị không có cạnh bội nên không có miền bậc 2 và không có khuyên nên không có miền bậc 1). Bậc của miền vô hạn cũng không nhỏ hơn 3 vì đang xét các đồ thị có ít nhất 3 đỉnh. Từ định nghĩa bậc của miền suy ra tổng

số bậc của các miền trong đơn đồ thị bằng hai lần số cạnh của đồ thị (mỗi cạnh được tính 2 lần cho 2 miền liền kề nhau). Vì mỗi miền có bậc nhỏ nhất là 3 nên:

$$2m = \sum_i \deg(M_i) \geq 3f \qquad \Rightarrow \qquad \frac{2m}{3} \geq f$$

Theo công thức Euler: f = m – n + 2, thay vào công thức trên được:

$$m - n + 2 \leq \frac{2m}{3} \qquad \Rightarrow \quad m \leq 3n - 6$$

Hệ quả được chứng minh.

Từ hệ quả này dễ thấy **đồ thị đầy đủ K_5 là đồ thị không phẳng**. Thật vậy K_5 có: n = 5, m = 10 do đó 3n – 6 = 9 < m. Vậy K_5 là không phẳng.

Hệ quả 2: Nếu một đơn đồ thị phẳng liên thông có m cạnh, n đỉnh trong đó n ≥ 3 và không có chu trình có độ dài 3 thì:

$$m \leq 2n - 4$$

Hệ quả 2 được chứng minh tương tự hệ quả 1, nhưng chú ý rằng do không có chu trình có độ dài 3 nên bậc của các miền ít nhất là bằng 4.

Từ hệ quả 2 suy ra **đồ thị phân đôi đầy đủ $K_{3,3}$ là không phẳng** vì $K_{3,3}$ không có chu trình có độ dài 3, có 6 đỉnh (n = 6) và 9 cạnh (m = 9) do đó 2n – 4 = 8 < m không thỏa mãn hệ quả 2. Như vậy không có cách nào làm các đường không cắt nhau trong bài toán "ba nhà, ba giếng".

Hệ quả 3: Trong một đơn đồ thị phẳng liên thông luôn luôn tồn tại ít nhất một đỉnh có bậc không vượt quá 5.

Thật vậy, theo chứng minh hệ quả 1, ta có:

$$3f \leq 2m \qquad (1)$$

Giả sử đơn đồ thị phẳng liên thông có tất cả bậc của các đỉnh đều lớn hơn hoặc bằng 6 thì theo tính chất về bậc của các đỉnh đồ thị (tính chất 1, mục 1.2, chương 3) ta có:

$$6n \leq 2m \quad \Rightarrow \quad 3n \leq m \qquad (2)$$

Từ (1) và (2) suy ra:

$$3n + 3f \leq m + 2m \quad \Rightarrow \quad n + f \leq m \quad \Rightarrow \quad f \leq m - n$$

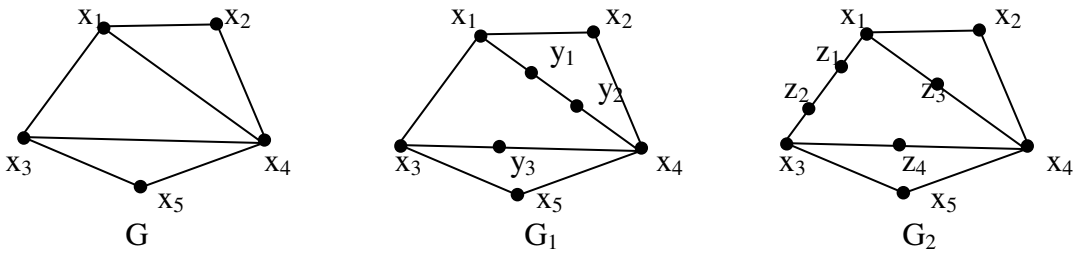
Điều này trái với định lý Euler (f = m – n + 2 ≥ m – n).

3.3. Nhận biết đồ thị không phẳng

Chúng ta đã thấy đồ thị $K_{3,3}$ và đồ thị K_5 là không phẳng, do đó mọi đồ thị chứa $K_{3,3}$ koặc K_5 như một đồ thị con là đồ thị không phẳng. Hơn thế, có một phép biến đổi sao cho một đồ thị không phẳng thành một đồ thị có chứa thị con là $K_{3,3}$ hoặc K_5. Trước hết chúng ta nghiên cứu phép biến đổi đó.

Cho đồ thị G, nếu bỏ đi cạnh (x,y) của G và thêm vào đỉnh z cùng hai cạnh (x,z) và (z,y) được một đồ thị mới G'. Phép biến đổi G thành G' như trên gọi là **phép chia cạnh** (còn gọi là phép **phân chia sơ cấp**) đồ thị.

Các đồ thị $G_1 = (X_1,U_1)$ và $G_2 = (X_2, U_2)$ được gọi là **đồng phôi** nếu có thể nhận được chúng từ cùng một đồ thị bằng một dãy các phép chia cạnh. Chẳng hạn ba đồ thị trên hình 20 là các đồ thị đồng phôi.



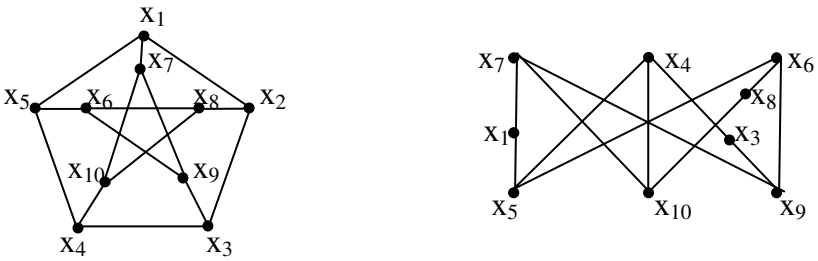
Hình 20. Các đồ thị đồng phôi

Định lý 2 (Định lý Kuratowski): Đồ thị là không phẳng khi và chỉ khi nó chứa một đồ thị con đồng phôi với đồ thị $K_{3,3}$ hoặc K_5.

Rõ ràng đồ thị chứa đồ thị con đồng phôi với $K_{3,3}$ hoặc K_5 là đồ thị không phẳng. Chúng ta thừa nhận điều ngược lại vì việc chứng minh nó khá phức tạp.

Thí dụ: Đồ thị Petersen (hình 21a) có phải là đồ thị phẳng không?

Giải: Bỏ đỉnh x_2 và 3 cạnh liên thuộc với x_2 và vẽ lại đồ thị con nhận được như hình 21b được đồ thị đồng phôi với đồ thị $K_{3,3}$. Vậy đồ thị Petersen là không phẳng.

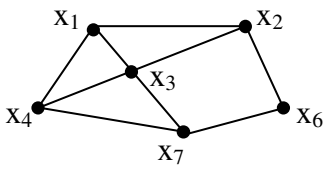


Hình 21

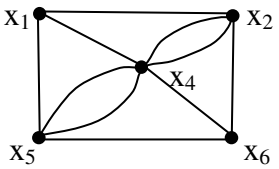
BÀI TẬP CHƯƠNG 4

4.1. Hãy xét xem các đồ thị sau, đồ thị nào là đồ thị Euler hoặc nửa Euler và tìm chu trình Euler hoặc đường Euler, nếu có.

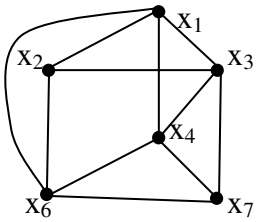
a)



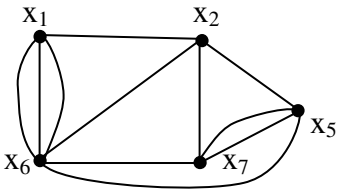
b)



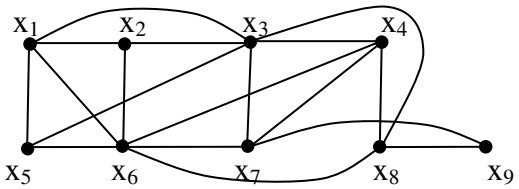
c)



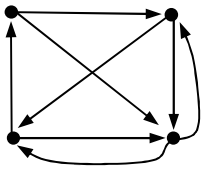
d)

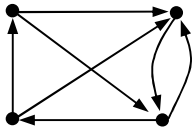


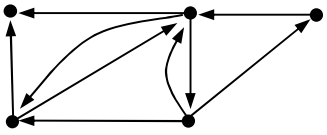
e)



4.2. Hãy xét xem các đồ thị sau, đồ thị nào là đồ thị Euler, đồ thị nào là nửa Euler và chỉ ra chu trình Euler hoặc đường Euler, nếu có.







4.3. Với giá trị nào của n thì các đồ thị sau là đồ thị Euler?

a) K_n ; b) W_n ; c) Q_n .

4.4. Với giá trị nào của m và n thì đồ thị phân đôi đầy đủ $K_{m,n}$ là đồ thị Euler, nửa Euler ?

4.5. Hãy xét xem các đồ thị cho bằng ma trận kề sau, đồ thị nào là đồ thị Euler hoặc nửa Euler và tìm chu trình Euler hoặc đường Euler, nếu có.

a) Đồ thị vô hướng

	x_1	x_2	x_3	x_4	x_5	x_6	x_7
x_1	0	1	1	1	1	0	0
x_2	1	0	1	1	1	0	0
x_3	1	1	0	1	0	1	1
x_4	1	1	1	0	1	1	1
x_5	1	1	0	1	0	1	1
x_6	0	0	1	1	1	0	1
x_7	0	0	1	1	1	1	0

b) Đồ thị vô hướng

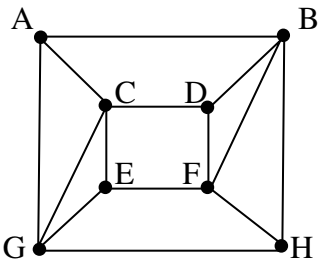
	A	B	C	D	E	F	G
A	0	1	1	0	1	1	0
B	1	0	1	0	0	1	1
C	1	1	0	1	0	1	0
D	0	0	1	0	2	0	1
E	1	0	0	2	0	1	0
F	1	1	1	0	1	0	0
G	0	1	0	1	0	0	0

c) Đồ thị có hướng

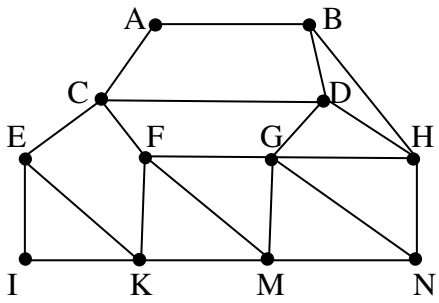
	1	2	3	4	5	6	7	8
1	1	0	0	0	1	1	0	0
2	1	0	0	0	1	0	0	0
3	0	1	0	0	0	1	0	0
4	0	1	0	0	0	1	0	0
5	1	0	1	0	0	0	1	0
6	0	0	1	0	0	0	1	1
7	0	0	0	1	0	0	0	1
8	0	0	0	1	1	0	0	1

4.6. Giải bài toán người đưa thư Trung hoa với đồ thị cho trong các hình sau:

a)

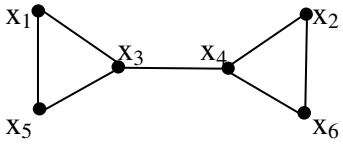


b)

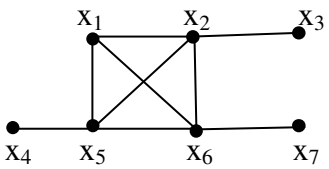


4.7. Hãy xét xem các đồ thị sau, đồ thị nào là đồ thị Hamilton hoặc nửa Hamilton và tìm chu trình Hamilton hoặc đường Hamilton, nếu có.

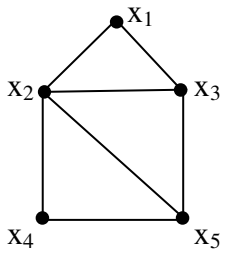
a)



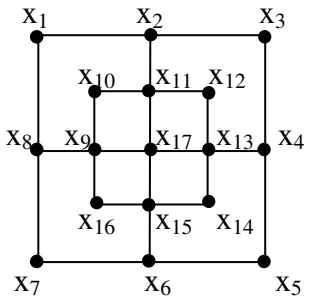
b)



c)

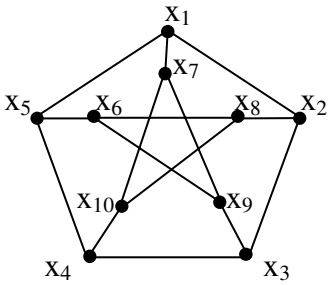


d)



4.8. a) Với giá trị nào của m và n thì đồ thị phân đôi đầy đủ $K_{m, n}$ là đồ thị Hamilton.
b) Vẽ cây liệt kê các chu trình Hamilton của đồ thị lập phương Q_3.

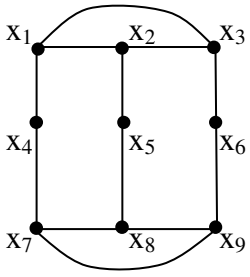
4.9. Hãy chỉ ra rằng đồ thị Petesen (xem hình vẽ) không có chu trình Hamilton, nhưng nếu xoá đi một đỉnh bất kỳ cùng các cạnh liên thuộc sẽ được một đồ thị Hamilton.



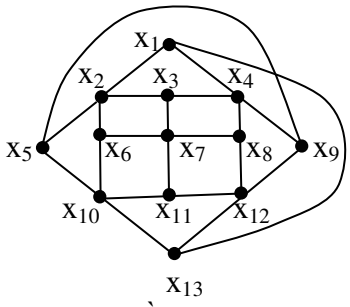
Đồ thị Petesen

4.10. Xây dựng một đồ thị thỏa mãn:
a) Vừa có chu trình Euler, vừa có chu trình Hamilton.
b) Có 6 đỉnh, là đồ thị Euler, nhưng không phải là đồ thị Hamilton.
c) Có 6 đỉnh, là đồ thị Hamilton nhưng không phải là đồ thị Euler.

4.11. Chứng minh rằng các đồ thị sau không có chu trình Hamilton nhưng có đường Hamilton.



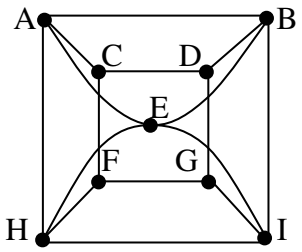
Đồ thị G_1



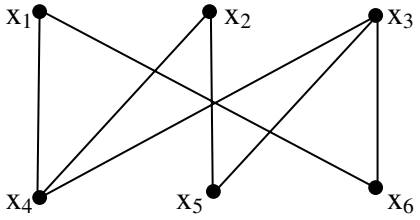
Đồ thị G_2

4.12. Trong các đồ thị sau, đồ thị nào là đồ thị phẳng, đồ thị nào là đồ thị không phẳng? Nếu là đồ thị phẳng hãy tìm dạng biểu diễn phẳng của nó.

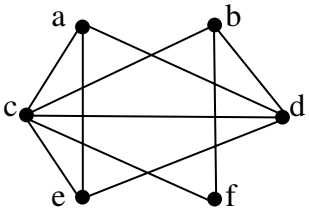
a)



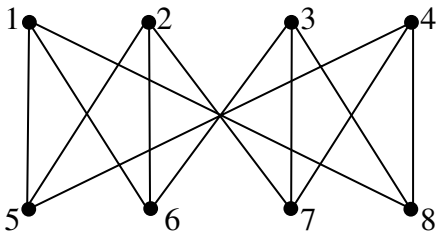
b)



c)

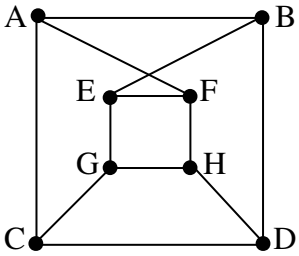


d)

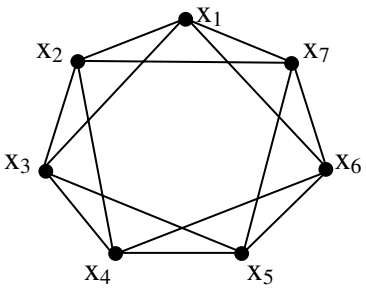


4.13. Dùng định lý Kuratowski chứng tỏ các đồ thị sau là không phẳng:

a)



b)



4.14. a) Một đơn đồ thị phẳng liên thông có 10 mặt, tất cả các đỉnh đều có bậc bằng 4. Tìm số đỉnh của đồ thị.

b) Một đơn đồ thị phẳng liên thông có 9 đỉnh, bậc của các đỉnh là 2, 2, 2, 3, 3, 3, 4, 4, 5. Tìm số cạnh và số mặt của đồ thị.

ĐÁP SỐ

4.1. a) không; b), c), e/ nửa Euler; d) Euler.
4.2. b), c) không; a) nửa Euler.
4.3. a) n lẻ; b) không; c) n chẵn.
4.5. a) nửa Euler; b), c) Euler
4.7. a) nửa Hamilton; b), d) không; c) Hamilton.
4.12. cả 4 đồ thị đều là đồ thị phẳng.
4.14. a) 8 đỉnh; b) 14 cạnh, 7 mặt.

CÂU HỎI ÔN TẬP CHƯƠNG 4

1. Phát biểu định nghĩa đồ thị Euler, đồ thị nửa Euler. Phát biểu và chứng minh định lý nhận biết. Phát biểu bài toán người đưa thư Trung hoa, cách giải.

2. Phát biểu định nghĩa đồ thị Hamilton, đồ thị nửa Hamilton. Phát biểu các định lý nhận biết đồ thị Hamilton, cho thí dụ chứng tỏ rằng các điều kiện của định lý là quá rộng rãi. Phát biểu các quy tắc tìm chu trình Hamilton.

3. Định nghĩa đồ thị phẳng. Phát biểu và chứng minh định lý và các hệ quả của công thức Euler. Khái niệm đồ thị đồng phôi và phát biểu định lý Kuratowski để nhận biết đồ thị không phẳng.

CHƯƠNG 5.

CÂY VÀ MỘT SỐ ỨNG DỤNG CỦA CÂY

1. Cây và các tính chất cơ bản của cây
    1.1. Định nghĩa
    1.2. Các tính chất cơ bản của cây
    1.3. Cây có gốc
2. Cây nhị phân và phép duyệt cây
    2.1. Các khái niệm
    2.2. Các thuật toán duyệt cây nhị phân
    2.3. Ký pháp nghịch đảo Ba lan (RPN)
3. Một vài ứng dụng của cây
    3.1. Cây và bài toán liệt kê
    3.2. Các mã tiền tố
    3.3. Mã tiền tố tối ưu
4. Cây khung (cây bao trùm) của đồ thị
    4.1. Định nghĩa
    4.2. Điều kiện đồ thị có cây khung
    4.3. Các thuật toán tìm cây khung của đồ thị
5. Hệ chu trình độc lập
6. Cây khung nhỏ nhất
    6.1. Định nghĩa
    6.2. Thuật toán Kruskal
    6.3. Tính đúng đắn của thuật toán Kruskal
    6.4. Thuật toán Prim
    6.5. Tính đúng đắn của thuật toán Prim

Cây là một khái niệm gắn liền với lý thuyết đồ thị, nó ra đời năm 1857 bởi nhà toán học Cayley người Anh khi ông dùng cây để xác định các cấu trúc phân tử khác nhau của hợp chất hữu cơ.

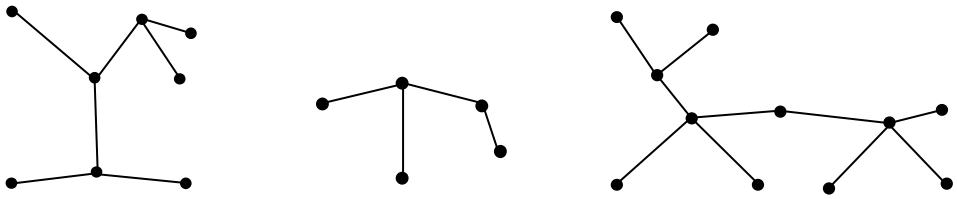
Cây có nhiều ứng dụng trong tin học. Chẳng hạn, cây được dùng để xây dựng các thuật toán, đặc biệt là trong các bài toán sắp xếp, liệt kê. Cách tổ chức các thư mục trong một hệ điều hành máy tính là một ứng dụng cụ thể của cây. Cây còn được dùng để xây dựng các mạng máy tính và nhiều ứng dụng khác trong tin học.

1. Cây và các tính chất cơ bản của cây

1.1. Định nghĩa

Cây là một đồ thị vô hướng liên thông, có ít nhất 2 đỉnh và không có chu trình.

Rừng là một đồ thị vô hướng, không liên thông và không có chu trình. Nói cách khác, rừng là một tập hợp mà mỗi phần tử là một cây riêng biệt. Chẳng hạn hình 1 là một rừng gồm 3 cây



Hình 1. Rừng có 3 cây

1.2. Các tính chất cơ bản của cây

a. Bổ đề: Mọi cây đều có ít nhất 2 đỉnh treo.

Thật vậy, gọi $\alpha = x_1\, x_2 \ldots x_k$ là đường đi đơn dài nhất có trong cây T. Khi đó x_1 và x_k là các đỉnh treo vì đỉnh x_1 chỉ có cạnh nối với x_2 và không có cạnh nối với bất cứ đỉnh x_i (i = 3, 4, …, k) nào khác, vì nếu không thì T có chu trình. Đồng thời x_1 cũng không có cạnh nối với bất cứ đỉnh nào khác ngoài tập $\{x_2, x_3, x_4, \ldots, x_k\}$, vì nếu không thì α không phải là đường đi dài nhất. Vậy x_1 là đỉnh treo. Tương tự x_k cũng là đỉnh treo.

b. Định lý: Cho T = (X, U) là một đồ thị vô hướng có n đỉnh ($n \geq 2$). Các tính chất sau đây là tương đương:

1-/ T liên thông và không có chu trình (T là cây).

2-/ T không có chu trình và có n – 1 cạnh.

3-/ T liên thông và có n – 1 cạnh.

4-/ T liên thông và nếu bỏ đi một cạnh bất kỳ thì đồ thị nhận được là không liên thông, nghĩa là mọi cạnh của T đều là cầu.

5-/ Mọi cặp đỉnh trong T được nối với nhau bằng một đường đi đơn duy nhất.

6-/ T không có chu trình và nếu thêm vào 1 cạnh nối 2 đỉnh bất kỳ không kề nhau thì trong T xuất hiện chu trình và chỉ một chu trình mà thôi.

Chứng minh

Định lý được chứng minh theo sơ đồ sau:

$$1 \Rightarrow 2 \Rightarrow 3 \Rightarrow 4 \Rightarrow 5 \Rightarrow 6 \Rightarrow 1$$

1 $\Rightarrow$ 2: Dùng quy nạp theo số đỉnh của cây.

Rõ ràng khi n = 2, $1 \Rightarrow 2$ là đúng.

Giả sử $1 \Rightarrow 2$ đúng với T_k có k đỉnh khi đó T_k có k – 1 cạnh. Xét cây T_{k+1} có k+1 đỉnh, loại bỏ đỉnh treo và cạnh treo khỏi T_{k+1} được cây T_k có k đỉnh, k – 1 cạnh. Vậy T_{k+1} có k cạnh.

Khẳng định được chứng minh.

2 $\Rightarrow$ 3: Dùng phản chứng.

Giả sử T không liên thông, khi đó T phải có k thành phần liên thông: T_1, T_2, …, T_k ($k \geq 2$). Do T không có chu trình nên các T_i không có chu trình, nghĩa là T_i là các cây.

Gọi $T_i = (X_i, U_i)$, ta có: $X_i \cap X_j = \varnothing$; $X_1 \cup \ldots \cup X_k = X$

$U_i \cap U_j = \varnothing$; $U_1 \cup \ldots \cup U_k = U$

Theo 2-/ ta có: $N(U_i) = N(X_i) - 1 \Rightarrow \sum_{i=1}^{k} N(U_i) = \sum_{i=1}^{k} [N(X_i) - 1] = n - k$

Mặt khác theo 2-/ thì T có n – 1 cạnh, nghĩa là: $\sum_{i=1}^{k} N(U_i) = n - 1$

Điều đó là mâu thuẫn. Vậy T liên thông.

3 $\Rightarrow$ 4: Nếu bỏ đi một cạnh được đồ thị có n đỉnh với n – 2 cạnh, hiển nhiên đồ thị đó không liên thông.

4 $\Rightarrow$ 5: Dùng phản chứng.

Giả sử tồn tại cặp đỉnh x, y không có đường đi đơn nối giữa chúng, như vậy x và y không liên thông, điều này trái với giả thiết nêu trong 4-/.

Giả sử tồn tại cặp đỉnh x, y có 2 đường đi nối giữa chúng, khi đó loại bỏ 1 cạnh của một trong hai đường đi đó thì đồ thị T vẫn liên thông. Điều này trái với 4-/ là nếu loại bỏ một cạnh thì T mất tính liên thông.

5 $\Rightarrow$ 6: T không có chu trình, vì nếu có chu trình thì mọi cặp đỉnh trên chu trình đó được nối với nhau bởi 2 đường đi đơn.

Nếu thêm vào T một cạnh nối 2 đỉnh x, y không kề nhau nào đó thì cạnh này cùng đường đi đơn từ x sang y tạo thành một chu trình. Chu trình này là duy nhất bởi nếu không thì hoặc giữa x và y phải có 2 đường đi đơn hoặc trước đó trong T phải có sẵn một chu trình.

6 $\Rightarrow$ 1: Dùng phản chứng

Giả sử T không liên thông, khi đó T có ít nhất 2 thành phần liên thông và cả 2 thành phần liên thông đều không có chu trình. Khi đó thêm vào T một cạnh nối giữa 2 đỉnh của 2 thành phần liên thông đó thì trong T không xuất hiện một chu trình nào cả, điều này trái với giả thiết 6-/

1.3. Cây có gốc

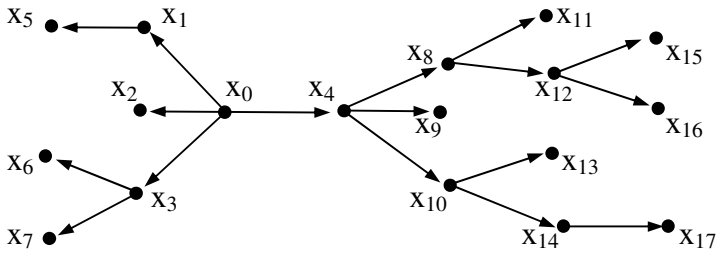
a. Định nghĩa:

Cây có hướng là đồ thị có hướng mà đồ thị vô hướng tương ứng của nó là một cây.

Cây có gốc là cây có hướng, trong đó có một đỉnh đặc biệt, gọi là gốc, từ gốc có đường đi đến mọi đỉnh khác của cây.

Trong cây có gốc thì gốc x_0 có bậc vào bằng 0, còn tất cả các đỉnh khác đều có bậc vào bằng 1 (hình 2).

Thí dụ:



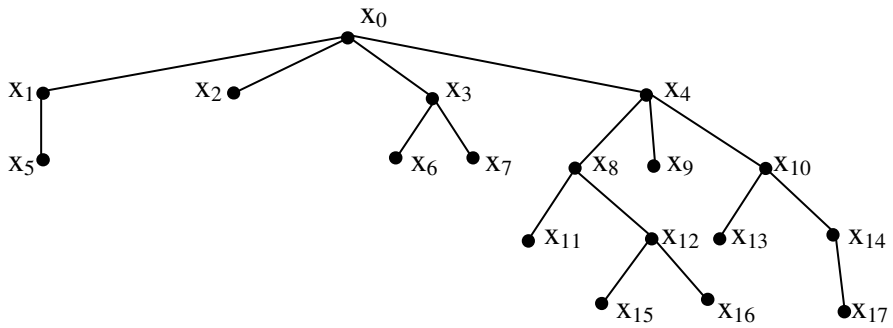
Hình 2. Cây với gốc x_0

Một cây có gốc thường được vẽ với gốc x_0 ở trên cùng và cây phát triển từ trên xuống (hình 3). Gốc của cây (đỉnh x_0) gọi là đỉnh mức 0. Các đỉnh kề với gốc được xếp ở phía dưới và gọi là đỉnh mức 1. Đỉnh ngay dưới đỉnh mức k là đỉnh mức k +1.

Tổng quát, trong một cây có gốc x_0 thì đỉnh x của cây là đỉnh có mức k khi và chỉ khi đường đi từ x_0 đến x có độ dài bằng k.

Mức lớn nhất của một đỉnh có trong cây gọi là chiều cao của cây.

Cây có gốc ở hình 2 thường được vẽ như trong hình 3 để làm rõ mức của các đỉnh.



Hình 3. Cách vẽ thông thường của cây có gốc

Trong cây có gốc, mọi cung đều có hướng từ trên xuống, vì vậy vẽ mũi tên để chỉ hướng đi là không cần thiết.

Xét một đường đi xuất phát từ gốc của cây có gốc. Ta có các khái niệm sau:

- Đỉnh có mức k được gọi là cha của các đỉnh mức k+1, đỉnh ở mức k+1 gọi là con của đỉnh ở mức k.
- Các đỉnh có cùng cha được gọi là anh em.
- Các đỉnh có mức nhỏ hơn k gọi là tổ tiên của đỉnh ở mức k.
- Đỉnh ở mức lớn nhất của đường đi đó gọi là lá. Lá là đỉnh treo và không có con.
- Mọi đỉnh không phải là lá được gọi là đỉnh trong.

Cây có gốc mà mức của mọi lá sai khác nhau không quá 1 đơn vị gọi là cây cân đối (hay cây cân bằng). Nói cách khác cây cân đối là cây mà mọi lá đều có mức h hoặc h – 1 trong đó h là chiều cao của cây.

Chú ý rằng với một cây tùy ý, nếu chọn một đỉnh nào đó làm gốc được cây có gốc tương ứng. Do chọn gốc bất kỳ nên với cây có n đỉnh sẽ có n cây có gốc tương ứng khác

nhau. Có thể chứng minh được rằng, nếu chọn tâm của cây là gốc thì được cây có gốc có độ cao nhỏ nhất (xem khái niệm tâm của đồ thị ở chương 6).

b. Cây m- phân

Định nghĩa: Một cây có gốc T được gọi là cây m-phân nếu mỗi đỉnh trong của T có nhiều nhất là m con.

Cây có gốc T được gọi là một cây m-phân đầy đủ nếu mỗi đỉnh trong của T đều có m con.

Định lý 1: Một cây m-phân đầy đủ có k đỉnh trong thì có m.k + 1 đỉnh và có (m−1)k + 1 lá.

Chứng minh: Mọi đỉnh trong của cây m-phân đầy đủ đều có bậc ra là m, còn lá có bậc ra là 0. Vậy tổng số bậc ra của cây là m.k, từ đó số cung của cây là m.k (theo tính chất 1 bậc của đỉnh, mục 1.2 chương 3). Và do đó số đỉnh của cây là m.k + 1. Gọi s là số lá thì ta có s + k = m.k + 1, nên s = (m−1)k + 1.

Định lý 2:

1) Một cây m-phân có chiều cao h thì có nhiều nhất là m^h lá.

2) Một cây m-phân có s lá thì có chiều cao h ≥ $[\log_m s]$.

3) Một cây m-phân đầy đủ và cân đối có s lá thì có chiều cao h = $[\log_m s]$

Chứng minh:

1) Mệnh đề được chứng minh bằng quy nạp theo h.

Mệnh đề hiển nhiên đúng khi h = 1.

Giả sử mọi cây m-phân có chiều cao h = k đều có nhiều nhất m^k lá (với h ≥ 2). Xét cây m-phân (cây T) có chiều cao h = k + 1. Bỏ gốc khỏi cây được một rừng gồm không quá m cây con, mỗi cây con này có chiều cao không quá k. Do giả thiết quy nạp, mỗi cây con này có nhiều nhất là m^k lá. Do lá của những cây con này cũng là lá của T, nên T có nhiều nhất là $m.m^k = m^{k+1}$ lá.

Theo nguyên lý quy nạp, mệnh đề đúng đối với mọi cây có chiều cao tuỳ ý.

2) $s \le m^h \Leftrightarrow h \ge [\log_m s]$.

3) Do cây là cân đối nên các lá có mức h hoặc h –1, vì có ít nhất một lá ở mức h nên số lá phải nhiều hơn m^{h-1}. Vậy $m^{h-1} < s \le m^h \Leftrightarrow h - 1 < \log_m s \le h \Leftrightarrow h = [\log_m s]$.

Thí dụ: Xét trò chơi gửi thư dây truyền. Ban đầu có một người nhận được một lá thư và giả sử rằng mỗi người khi nhận được một thư hoặc sẽ sao gửi thư đó cho 4 người khác hoặc là không chuyển cho ai cả. Hỏi có bao nhiêu người nhận được thư kể cả người đầu tiên nếu không có ai nhận được nhiều hơn một bức thư và trò chơi kết thúc khi có 100 người nhận được thư mà không viết cho ai cả.

Giải: Biểu diễn trò chơi bằng một cây tứ phân đầy đủ, các đỉnh trong là những người nhận được thư và sao gửi cho người khác còn lá là những người nhận được thư mà không gửi cho ai cả. Vì có 100 người không viết thư cho ai nên số lá của cây là s = 100, theo định lý 1 ta có 100 = (4 – 1)k + 1 ⇒ số đỉnh trong của cây là: k = 33 ⇒ số đỉnh của của cây là: n = 100 + 33 = 133. Vậy có 133 người nhận được thư.

2. Cây nhị phân và phép duyệt cây

2.1. Các khái niệm

Cây nhị phân là cây có gốc và mỗi đỉnh có nhiều nhất là 2 con.

Trong một cây nhị phân, mỗi con được chỉ rõ là con bên trái (được vẽ về phía dưới bên trái) hay con bên phải (được vẽ phía dưới bên phải) của cha.

Cây nhị phân có gốc x được ký hiệu là T(x). Giả sử x có con bên trái là y và con bên phải là z. Nếu loại x và các cạnh liên thuộc ra khỏi T(x) thì được hai cây nhị phân T(y) và T(z) "nhỏ hơn" T(x), khi ấy T(y) được gọi là cây con bên trái còn T(z) là cây con bên phải của T(x).

Duyệt cây (hay đọc cây) là đưa ra danh sách liệt kê tất cả các đỉnh của cây, mỗi đỉnh một lần.

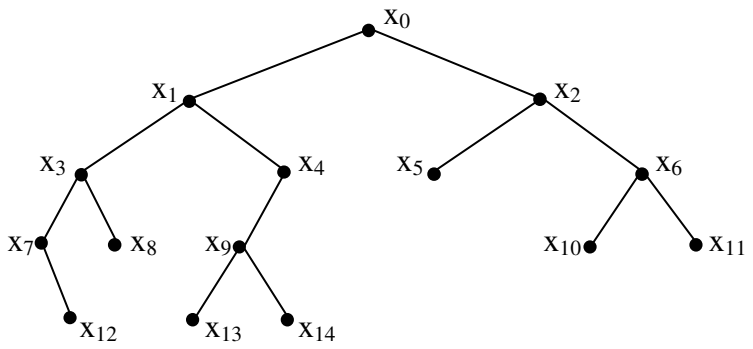
2.2. Các thuật toán duyệt cây nhị phân

Để tránh bỏ sót các đỉnh không được liệt kê và tránh lặp lại các đỉnh đã liệt kê khi duyệt cây cần phải có các thuật toán duyệt cây. Các thuật toán dưới đây được trình bày đệ qui và khi thuật ngữ duyệt x_i được dùng có nghĩa là đỉnh x_i được đưa vào danh sách liệt kê.

a) Thuật toán tiền thứ tự:

1. Duyệt gốc x.
2. Duyệt cây con bên trái của T(x) theo tiền thứ tự.
3. Duyệt cây con bên phải của T(x) theo tiền thứ tự.

Thí dụ: Duyệt cây nhị phân $T(x_0)$ trong hình 4 theo tiền thứ tự:



Hình 4

1. Duyệt x_0
2. Duyệt $T(x_1)$
   - 2.1. Duyệt x_1
   - 2.2. Duyệt $T(x_3)$
     - 2.2.1. Duyệt x_3
     - 2.2.2. Duyệt $T(x_7)$
       - 2.2.2.1. Duyệt x_7

2.2.2.3. Duyệt T(x_{12}): Duyệt x_{12}

2.2.3. Duyệt T(x_8): Duyệt x_8

2.3. Duyệt T(x_4)

2.3.1. Duyệt x_4

2.3.2. Duyệt T(x_9)

2.3.2.1. Duyệt x_9

2.3.2.2. Duyệt T(x_{13}): Duyệt x_{13}

2.3.2.3. Duyệt T(x_{14}): Duyệt x_{14}

3. Duyệt T(x_2)

3.1. Duyệt x_2

3.2. Duyệt T(x_5): Duyệt x_5

3.3. Duyệt T(x_6)

3.3.1. Duyệt x_6

3.3.2. Duyệt T(x_{10}): Duyệt x_{10}

3.3.3. Duyệt T(x_{11}): Duyệt x_{11}

Kết quả duyệt cây T(x_0) theo tiền thứ tự là:

x_0 x_1 x_3 x_7 x_{12} x_8 x_4 x_9 x_{13} x_{14} x_2 x_5 x_6 x_{10} x_{11}

b) Thuật toán trung thứ tự:

1. Duyệt cây con bên trái của T(x) theo trung thứ tự.
2. Duyệt gốc x.
3. Duyệt cây con bên phải của T(x) theo trung thứ tự.

Thí dụ: Duyệt cây nhị phân T(x_0) trong hình 4 theo trung thứ tự:

1. Duyệt T(x_1)

1.1. Duyệt T(x_3)

1.1.1. Duyệt T(x_7)

1.1.1.2. Duyệt x_7

1.1.1.3. Duyệt T(x_{12}): Duyệt x_{12}

1.1.2. Duyệt x_3

1.1.3. Duyệt T(x_8): Duyệt x_8

1.2. Duyệt x_1

1.3. Duyệt T(x_4)

1.3.1. Duyệt T(x_9)

1.3.1.1. Duyệt T(x_{13}): Duyệt x_{13}

1.3.1.2. Duyệt x_9

1.3.1.3. Duyệt T(x_{14}): Duyệt x_{14}

1.3.2. Duyệt x_4

2. Duyệt x_0

3. Duyệt $T(x_2)$

3.1. Duyệt $T(x_5)$: Duyệt x_5

3.2. Duyệt x_2

3.3. Duyệt $T(x_6)$

3.3.1. Duyệt $T(x_{10})$: Duyệt x_{10}

3.3.2. Duyệt x_6

3.3.3. Duyệt $T(x_{11})$: Duyệt x_{11}

3.3. Duyệt x_2

Kết quả duyệt cây $T(x_0)$ theo trung thứ tự là:

$$x_7\ x_{12}\ x_3\ x_8\ x_1\ x_{13}\ x_9\ x_{14}\ x_4\ x_0\ x_5\ x_2\ x_{10}\ x_6\ x_{11}.$$

c) Thuật toán hậu thứ tự:

1. Duyệt cây con bên trái của T(x) theo hậu thứ tự.
2. Duyệt cây con bên phải của T(x) theo hậu thứ tự.
3. Duyệt gốc x.

Thí dụ: Duyệt cây nhị phân $T(x_0)$ trong hình 4 theo hậu thứ tự:

1. Duyệt $T(x_1)$

1.1. Duyệt $T(x_3)$

1.1.1. Duyệt $T(x_7)$

1.1.1.2. Duyệt $T(x_{12})$: Duyệt x_{12}

1.1.1.3. Duyệt x_7

1.1.2. Duyệt $T(x_8)$: Duyệt x_8

1.1.3. Duyệt x_3

1.2. Duyệt $T(x_4)$

1.2.1. Duyệt $T(x_9)$

1.2.1.1. Duyệt $T(x_{13})$: Duyệt x_{13}

1.2.1.2. Duyệt $T(x_{14})$: Duyệt x_{14}

1.2.1.3. Duyệt x_9

1.2.3. Duyệt x_4

1.3. Duyệt x_1

2. Duyệt $T(x_2)$

2.1. Duyệt $T(x_5)$: Duyệt x_5

2.2. Duyệt $T(x_6)$

2.2.1. Duyệt $T(x_{10})$: Duyệt x_{10}

2.2.2. Duyệt $T(x_{11})$: Duyệt x_{11}

2.2.3. Duyệt x_6

2.3. Duyệt x_2

3. Duyệt x_0

Kết quả duyệt cây $T(x_0)$ trong hình 4 theo hậu thứ tự là:

$$x_{12}\ x_7\ x_8\ x_3\ x_{13}\ x_{14}\ x_9\ x_4\ x_1\ x_5\ x_{10}\ x_{11}\ x_6\ x_2\ x_0$$

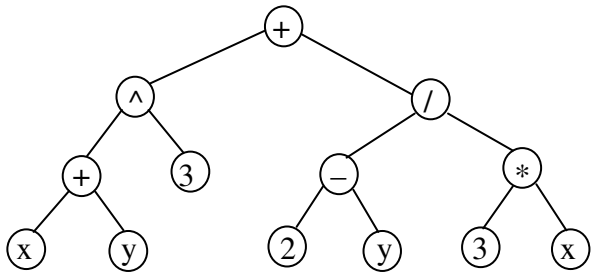
Chú thích: Các thuật toán duyệt cây nhị phân có thể mở rộng để duyệt cây m-phân bằng cách thay thứ tự cây con bên trái, cây con bên phải bằng thứ tự các cây con từ trái sang phải.

2.3. Ký pháp nghịch đảo Ba lan (RPN)

Xét biểu thức đại số sau:

$$A = (x + y)^3 + \frac{2 - y}{3x}$$

Ký hiệu ^ là phép lũy thừa, * là phép nhân, / là phép chia. Các phép toán trong biểu thức trên đều là các phép toán hai ngôi nên có thể biểu diễn cách tính A bằng một cây nhị phân có các đỉnh trong là ký hiệu của các phép toán số học, còn các lá là các số hoặc chữ đại diện cho các số (hình 5).



Hình 5

Duyệt cây ở hình 5 theo hậu thứ tự được:

x y + 3 ^ 2 y – 3 x * / + (1)

Rõ ràng có thể tính A theo trình tự của biểu thức (1) mà không cần các dấu ngoặc và không sợ nhầm lẫn.

Biểu thức dạng (1) được gọi là ký pháp nghịch đảo Balan, viết tắt là RPN (Reverse Polish Notation) do nhà toán học và logic học Ba lan Lukasiewicz (1878 – 1956) đặt ra.

Ngược lại, nếu cho một biểu thức RPN có thể tính được giá trị của biểu thức đó. Chẳng hạn, giá trị của biểu thức RPN 8 2 3 * – 4 ^ 9 3 / + được tính như sau:

8 2 3 * – 4 ^ 9 3 / + = 8 6 – 4 ^ 9 3 / + =
(1 2 3: 2.3=6; 8–6=2)

2 4 ^ 9 3 / + = 16 9 3 / + = 16 3 + = 19
(1 2 3: 2^4=16; 9:3=3; 1 2 3: 16+3=19)

Chú thích: Nếu duyệt cây ở hình 5 theo tiền thứ tự được:

+ ^ + x y 3 / – 2 y * 3 x (2)

Biểu thức (2) được gọi là ký pháp Ba lan ít dùng hơn RPN.

3. Một vài ứng dụng của cây

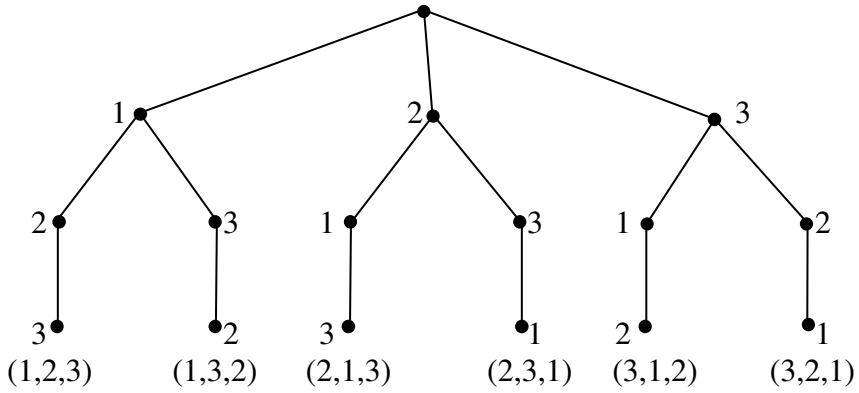
3.1. Cây và bài toán liệt kê

Bài toán liệt kê đã được trình bày trong chương 2. Có thể ứng dụng cây có gốc để giải bài toán liệt kê bằng thuật toán quay lui.

Giả sử cần liệt kê các cấu hình nào đó gồm k phần tử lấy từ tập hợp có n phần tử $N = \{x_1, x_2, \ldots, x_n\}$ đã cho. Chọn một điểm bất kỳ làm gốc và các đỉnh ở mức 1 tương ứng với các phần tử của tập N thỏa mãn cấu hình cần tìm. Đó là các phần tử thứ nhất trong các cấu hình phải liệt kê. Như vậy gốc của cây có không quá n con. Tuỳ theo yêu cầu của cấu hình, phần tử thứ hai của cấu hình được chọn từ các phần tử thích hợp trong tập N hoặc trong tập $N_1 = N \setminus \{$phần tử đã chọn làm phần tử thứ nhất của cấu hình$\}$. Quá trình tiếp tục cho đến khi chọn được các cấu hình thỏa mãn bài toán đã cho. Cây được xây dựng như đã trình bày gọi là cây liệt kê. Các cấu hình cần liệt kê chính là các đỉnh nằm trên đường đi từ đỉnh ở mức 1 đến các lá của cây liệt kê..

Sau đây là một vài thí dụ.

Thí dụ 1: Xây dựng cây liệt kê các hoán vị của tập $N = \{1, 2, 3\}$.



Hình 6. Cây liệt kê hoán vị của {1, 2, 3}

Chọn điểm tùy ý làm gốc, cả 3 phần tử của N đều được chọn làm phần tử thứ nhất của các hoán vị nên gốc có ba con: 1, 2 và 3.

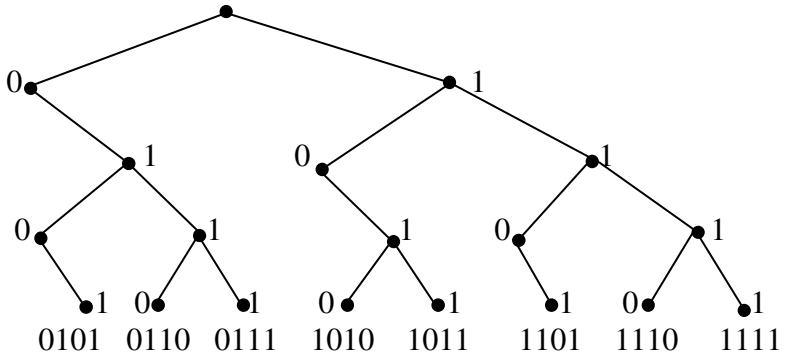
Xét đỉnh 1, vì 1 đã được chọn do đó các con của đỉnh 1 chỉ còn chọn trong {2, 3}. Cả 2 và 3 đều được chọn vậy đỉnh 1 có hai con là 2 và 3. Tương tự đối với các đỉnh 2 và 3 ở mức 1.

Xét đỉnh 2 là con của đỉnh 1, vì 2 và 1 đã được chọn do đó con của 2 chỉ còn duy nhất phần tử 3. Hoàn toàn tương tự đối với các đỉnh ở mức 2 khác.

Tất cả các phần tử của N đã được chọn. Vậy cây tìm kiếm được kết thúc, ta có 6 hoán vị như trong hình 6.

Thí dụ 2: Dùng cây liệt kê các xâu nhị phân có độ dài 4 sao cho không có hai số 0 liên tiếp.

Hình 7 là lời giải của bài toán. Chú ý là người ta qui ước trong cây nhị phân biểu diễn xâu nhi phân thì con bên trái là bít 0 và con bên phải là bít 1. Các con trong mọi bước đều được chọn từ tập $B = \{0,1\}$. Có 8 xâu thỏa mãn, đó là các xâu đã liệt kê trong hình 7.

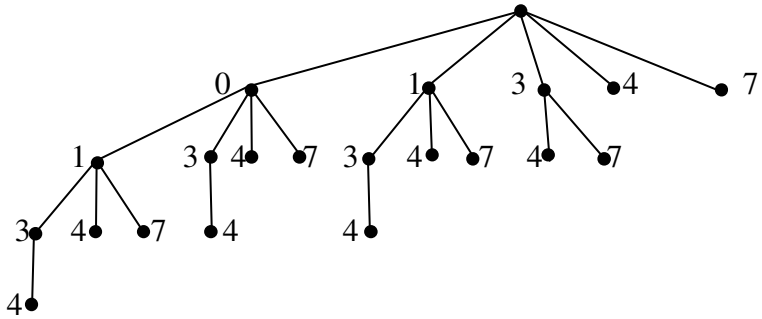


Hình 7

Thí dụ 3: Liệt kê các tập con của tập hợp A = {0, 1, 3, 4, 7, 11,12} sao cho tổng các phần tử của tập con tìm được không lớn hơn 10.

Lời giải của bài toán được trình bày trong hình 8. Các tập hợp con có k phần tử là các đỉnh trên các đường đi từ gốc đến đỉnh bậc k, trừ đỉnh gốc của cây. Cụ thể là:

- Các tập con có 1 phần tử: {0}, {1}, {3}, {4}, {7}.
- Các tập con có 2 phần tử: {0,1}, {0,3}, {0,4}, {0,7}, {1,3}, {1,4}, {1,7}, {3,4}, {3,7}.
- Các tập con có 3 phần tử: {0,1,3}, {0,1,4}, {0,1,7}, {0,3,4}, {1,3,4}.
- Các tập con có 4 phần tử: {0,1,3,4}



Hình 8. Cây liệt kê các tập hợp con

3.2. Các mã tiền tố

Trong tin học, mỗi chữ cái được mã hoá bằng một xâu nhị phân. Nếu không phân biệt chữ hoa với chữ thường thì với các chữ cái tiếng Việt hoặc tiếng Anh, mỗi chữ cái chỉ cần dùng các xâu có độ dài 5 là đủ (vì có $2^5 = 32$ xâu nhị phân có độ dài 5). Tuy nhiên có thể dùng ít hơn các bit để mã hoá các chữ cái nếu dùng các xâu có độ dài khác nhau để mã hoá. Khi đó các chữ cái xuất hiện thường xuyên hơn được mã hoá bằng các xâu có độ dài ngắn hơn.

Vấn đề đặt ra là phải mã hoá như thế nào để có thể phân biệt được một chữ cái được bắt đầu và kết thúc tại đâu. Chẳng hạn nếu ta mã hoá theo bảng mã sau:

Ký tự	e	a	t
Mã	0	1	01

Khi đó xâu 0101 có thể ứng với các từ eat, eaea hoặc tt.

Để đảm bảo không có xâu nhị phân nào ứng với hơn một dãy các chữ cái thì cần phải mã hoá sao cho *xâu nhị phân ứng với một chữ cái nào đó không bao giờ xuất hiện như là phần đầu của xâu nhị phân ứng với một chữ cái khác.* Các mã có tính chất như vậy gọi là **mã tiền tố**. Thí dụ, bảng mã:

Ký tự	e	a	t
Mã	0	10	11

là một mã tiền tố của ba chữ cái: e, a, t.

Với các mã tiền tố thì một xâu nhị phân chỉ ứng với duy nhất một dãy chữ cái, chẳng hạn với thí dụ đã nêu thì xâu 01011 là mã của chữ eat.

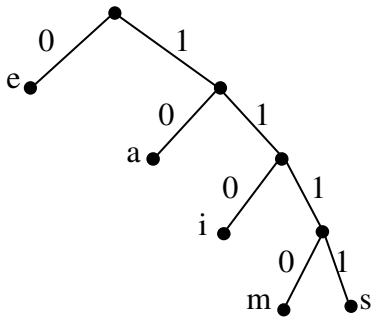
Các mã tiền tố được xây dựng bằng cây nhị phân và được gọi là cây mã, trong đó các chữ cái là nhãn các lá. Xâu nhị phân gồm các bít trên đường đi từ gốc đến lá là mã của chữ cái là nhãn của lá đó.

Thí dụ: Xét cây nhị phân trong hình 9.

Khi đó ta có bảng mã:

Ký tự	e	a	i	m	s
Mã	0	10	110	1110	1111

Từ đó xâu 11111011100 là mã của từ same



Hình 9

Ngược lại để kiểm tra xem một cách mã có phải là mã tiền tố hay không, cần phải biểu diễn các mã đó thành cây nhị phân. Nếu các lá có nhãn là các chữ cái có trong bảng mã thì đó là mã tiền tố, còn nếu tồn tại đỉnh trong có nhãn là chữ cái thì đó không phải là mã tiền tố.

Thí dụ: Hãy xét xem các cách mã hóa sau có phải là mã tiền tố không?

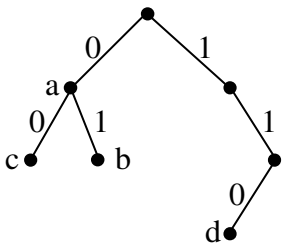
a)

Ký tự	a	b	c	d
Mã	0	01	00	110

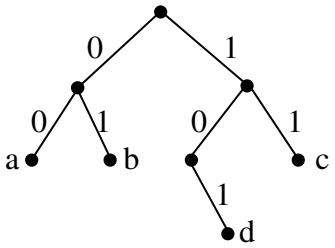
b)

Ký tự	a	b	c	d
Mã	00	01	11	101

Biểu diễn các mã thành cây nhị phân (hình 10a và hình 10b) ta có kết luận.



Hình 10a
Cách a) không phải là mã tiền tố



Hình 10b
Cách b) là mã tiền

3.3. Mã tiền tố tối ưu

Để triệt để tiết kiệm số bít dùng cho một bản tin thì việc chọn mã tiền tố cho các chữ cái có độ dài ngắn khác nhau còn phụ thuộc vào tần suất các chữ cái được dùng trong bản tin đó. Chẳng hạn một bản tin gồm 10^5 ký tự lấy trong tập các chữ cái X = {a,b,c,d,e,f} với tần suất sau:

Ký tự	a	b	c	d	e	f
Tần suất (%)	45	10	12	3	20	10

Khi ấy, nếu dùng bảng mã tiền tố:

Ký tự	a	b	c	d	e	f
Mã	00	01	111	101	100	110

để mã hóa bản tin thì số bit cần dùng là:

$$10^5(2.45\% + 2.10\% + 3.12\% + 3.3\% + 3.20\% + 3.10\%) = 245000$$

Nếu dùng bảng mã tiền tố:

Ký tự	a	b	c	d	e	f
Mã	0	100	101	110	1110	1111

để mã hóa bản tin thì số bít cần dùng là:

$$10^5(1.45\% + 3.10\% + 3.12\% + 3.3\% + 4.20\% + 5.10\%) = 250000$$

Rõ ràng, với các cách mã hóa khác nhau dẫn đến độ dài ngắn khác nhau của cùng một bản tin. **Thuật toán Huffman** sau đây cho mã tiền tố tối ưu đối với một bản tin sử dụng các ký tự trong tập ký tự X.

Trước hết, với cây mã T thì với mỗi lá được gán nhãn $x \in X$, trong đó X là tập ký tự cần mã hóa. Mức $\Delta(x)$ của x trong cây mã T chính là chiều dài của xâu nhị phân là mã của x. Gọi f(x) là tần suất xuất hiện của ký tự x trong bản tin thì số các bít dùng để mã hóa toàn bộ các ký tự có trong bản tin là $n\sum_{x\in X}\Delta(x).f(x)$, trong đó n là số ký tự có trong bản tin.

Cây mã T là tối ưu khi $E(T) = \sum_{x\in X}\Delta(x)f(x)$ đạt giá trị nhỏ nhất theo nghĩa chuỗi bít mã hóa bản tin có độ dài ngắn nhất.

Thuật toán Huffman gồm các bước:

1. Xây dựng tập đỉnh, mỗi đỉnh tương ứng với một ký tự có trong tập X và sẽ là lá trong cây mã sau này. Gán cho mỗi đỉnh 2 nhãn, một nhãn là ký tự $x \in X$ và một nhãn là tần suất f(x) tương ứng. Gọi tập đỉnh đã gán nhãn là X.

2. Chọn đỉnh x có nhãn f(x) nhỏ nhất và y có nhãn f(y) nhỏ tiếp theo, nối x và y với một đỉnh mới z được cây nhị phân gốc z, nhãn của z là f(z) = f(x) + f(y).

3. Nếu tất cả các đỉnh của bước 1 đã là lá của cây thì dừng và cây thu được là cây mã tối ưu. Nếu không, lặp lại từ bước 2 với tập đỉnh $X = X \setminus \{x,y\} \cup \{z\}$.

Người ta gọi cây mã xây dựng được theo thuật toán Huffman là **cây mã Huffman** và bảng mã thu được là **mã Huffman.**

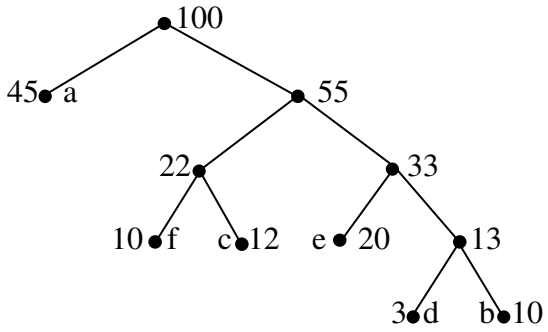
Trở lại thí dụ ở đầu mục này, áp dụng thuật toán Huffman được cây mã Huffman như hình 11.

Từ đó có bảng mã tiền tố tối ưu cho bản tin là:

Ký tự	a	b	c	d	e	f
Mã	0	1111	101	1110	110	100

Số bít cần dùng để mã hóa bản tin theo mã Huffman là:

$10^5(1.45\% + 4.10\% + 3.12\% + 4.3\% + 3.20\% + 3.10\%) = 223000$



Hình 11. Cây mã Huffman

Định lý: Cây mã Huffman là cây mã tối ưu.

Để chứng minh định lý, trước hết xét bổ đề sau:

Bổ đề: Gọi X là tập hợp các ký hiệu cần mã hóa và x, y là hai ký hiệu có tần suất nhỏ nhất. Khi đó có một cây mã tối ưu T cho X sao cho x, y là hai anh em trong T.

Chứng minh bổ đề: Gọi T_0 là cây mã tối ưu cho X. Nếu x, y không là anh em, gọi u là lá có mức lớn nhất trong T_0 thì phải có một lá v là anh em với u. Thật vậy, nếu u không có anh em, gọi w là cha của u, xóa đỉnh u và coi w là lá tương ứng với ký hiệu u sẽ nhận được cây mã mới T_0' với $E(T_0') < E(T_0)$, điều này là vô lý vì T_0 là cây mã tối ưu.

Bây giờ, nếu x không cùng mức u, v thì $\Delta(x) < \Delta(u)$ và bằng cách hoán đổi giữa x và u nhận được cây mã T_1 với:

$$E(T_1) = E(T_0) - \Delta(x)f(x) - \Delta(u)f(u) + \Delta(x)f(u) + \Delta(u)f(x) =$$
$$= E(T_0) - [\Delta(u) - \Delta(x)][f(u) - f(x)] < E(T_0)$$

Điều này trái với tính tối ưu của T_0. Vậy x có cùng mức với u và y có cùng mức với v. Bằng cách hoán đổi x với u và y với v sẽ nhận được cây mã tối ưu T vì $E(T) = E(T_0)$.

Chứng minh định lý: Dùng quy nạp theo n = |X|.

Khi n = 2, định lý hiển nhiên đúng.

Giả sử định lý đúng với k ($k \geq 2$). Xét với k +1. Coi $X = \{x_1, x_2, \ldots, x_k, x_{k+1}\}$. Không mất tính tổng quát, có thể giả thiết x_k, x_{k+1} là hai ký tự có tần suất nhỏ nhất. Gọi T là cây mã tối ưu cho X sao cho x_k và x_{k+1} là anh em trong T (T tồn tại do bổ đề). Gọi H là cây mã Huffman (cây mã có được từ việc áp dụng thuật toán Hufman) và x_k, x_{k+1} cũng là anh em trong H. Gọi H' là cây mã nhận được từ H bằng cách xóa x_k, x_{k+1} và gán cho đỉnh cha của x_k, x_{k+1} ký tự y với tần suất $f(y) = f(x_k) + f(x_{k+1})$. Dễ thấy H' là cây mã Huffman cho tập $Y= \{x_1, \ldots, x_{k-1}, y\}$. Tương tự, gọi T' là cây mã nhận được từ T bằng cách xóa x_k, x_{k+1} và gán cho đỉnh cha của x_k, x_{k+1} ký tự z với tần suất $f(z) = f(x_k) + f(x_{k+1})$ thì T' là một cây mã của tập Y. Do giả thiết quy nạp, ta có: $E(H') \leq E(T')$. Vậy:

$$E(H) = E(H') + \Delta(x_k)f(x_k) + \Delta(x_{k+1})f(x_{k+1}) \leq$$
$$\leq E(T') + \Delta(x_k)f(x_k) + \Delta(x_{k+1})f(x_{k+1}) = E(T).$$

Vì cây mã T tối ưu nên E(H) = E(T). Vậy cây mã H là tối ưu.

4. Cây khung (Cây bao trùm) của đồ thị

4.1. Định nghĩa:

Cho đồ thị vô hướng G = (X, U) với N(X) > 1. Đồ thị bộ phận T (là đồ thị giữ nguyên đỉnh và bỏ bớt một số cạnh) của đồ thị G được gọi là một cây khung của G nếu T là một cây. Hình 12 là 3 cây khung của một đồ thị (đồ thị G).

4.2. Điều kiện đồ thị có cây khung

Định lý: Điều kiện cần và đủ để một đồ thị vô hướng G với 2 đỉnh trở lên có cây khung là đồ thị đó liên thông.

Chứng minh:

Điều kiện đủ: Dùng phản chứng

Giả sử G có cây khung T nhưng G không liên thông, khi đó tồn tại cặp đỉnh $x,y \in G$ không có đường đi giữa chúng. Do x,y cũng thuộc T nên T cũng không liên thông, vậy T không phải là cây. Điều này trái với giả thiết đã nêu.

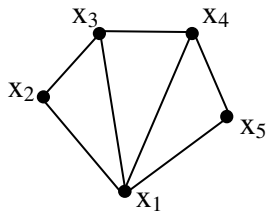
Điều kiện cần:

Giả sử G liên thông. Khi đó:

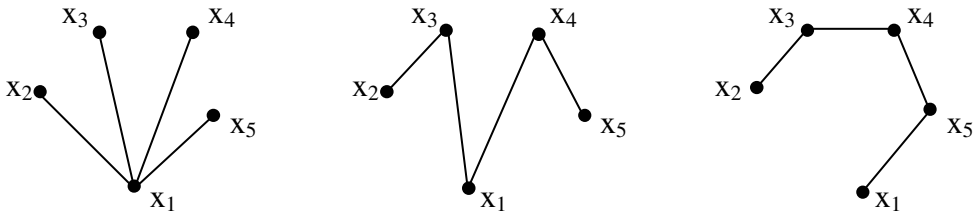
a-/ Nếu G không có chu trình thì G là một cây và là cây khung của chính nó.

b-/ Nếu G có k chu trình thì mỗi chu trình bỏ đi một cạnh được cây khung của đồ thị.

Thí dụ:



Đồ thị G



Ba cây khung của G

Hình 12

4.3. Các thuật toán tìm cây khung của đồ thị

Một đồ thị liên thông có rất nhiều cây khung, đôi khi số cây khung của một đồ thị rất lớn, chẳng hạn đồ thị đầy đủ K_n có n! cây khung.

Các thuật toán sau cho phép tìm một trong các cây khung của đồ thị liên thông G = (X,U).

a- Tìm kiếm ưu tiên chiều sâu

Trước kết tìm một đỉnh tuỳ ý của G làm gốc của cây khung. Từ đỉnh này xây dựng một đường đi bằng cách ghép lần lượt các cạnh vào sao cho cạnh mới ghép nối tiếp với cạnh đã ghép trước đó. Tiến hành ghép đến khi nào không ghép được nữa thì thôi. Chú ý khi ghép không để tạo thành chu trình.

Nếu đường đi vừa lập đi qua tất cả các đỉnh của đồ thị G thì đã được cây khung của G.

Nếu đường đi vừa lập chưa đi qua tất cả các đỉnh của đồ thị G thì từ đỉnh cuối cùng của đường đi này quay lại đỉnh trước nó và, từ đỉnh này xây dựng một đường đi mới xuất phát từ đỉnh đó và đi qua các đỉnh chưa được ghép vào đường đi trước. Nếu tại đỉnh đó không tìm được đường đi nào hoặc đã tìm hết một đường đi dài nhất có thể thì lại lui thêm một đỉnh nữa và tiếp tục xây dựng một đường đi đơn dài nhất có thể từ đỉnh này. Quá trình tiếp tục cho đến khi lui về đến đỉnh gốc. Vì đồ thị là hữu hạn nên thuật toán sẽ kết thúc sau một số hữu hạn bước.

Thuật toán trên còn gọi là *thuật toán quay lui*.

b- Tìm kiếm ưu tiên chiều rộng

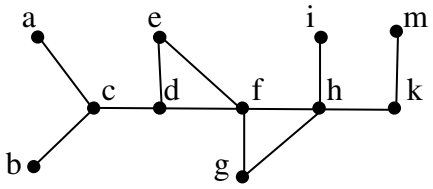
Chọn một đỉnh tuỳ ý của đồ thị G làm gốc của cây khung.

Ghép tất cả các cạnh liên thuộc của đỉnh đã chọn được các đỉnh thuộc mức 1.

Với mỗi đỉnh ở mức 1 ta lại làm như đỉnh gốc, tức là ghép tất cả các cạnh liên thuộc với đỉnh ở mức 1 sao cho không tạo thành chu trình (nên ghép theo một thứ tự nào đó để khỏi lẫn) được các đỉnh ở mức 2.

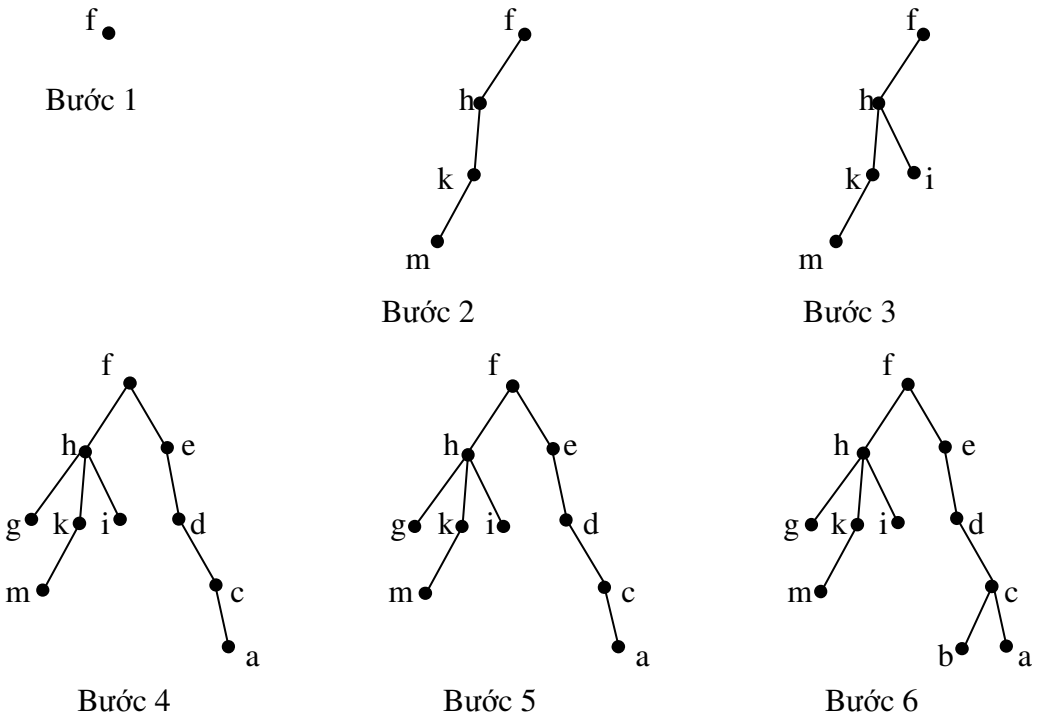
Tiếp tục ghép đối với đỉnh thuộc mức 2, … Thủ tục kết thúc khi tất cả các đỉnh được ghép vào cây

Thí dụ: Tìm cây khung của đồ thị ở hình 13:



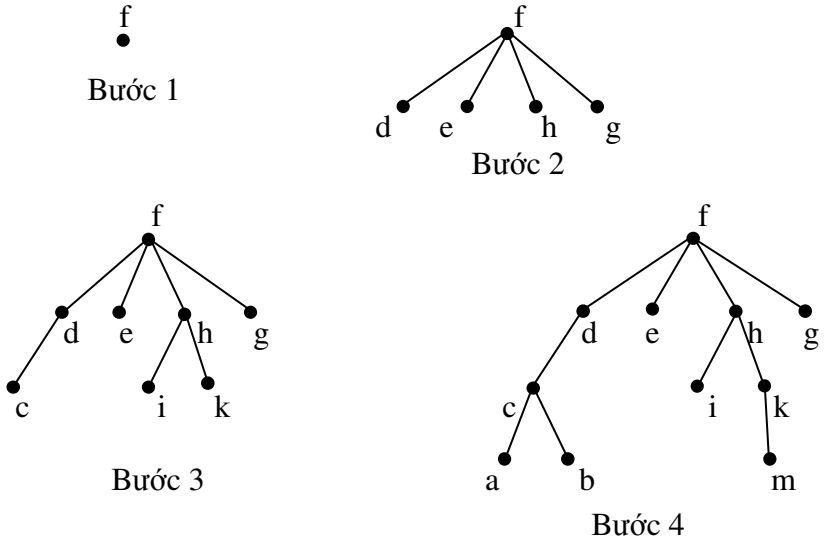
Hình 13

Chọn đỉnh f làm gốc, các bước quay lui được trình bày trong hình 14.



Hình 14. Các bước tìm cây khung theo thuật toán ưu tiên chiều sâu

Cũng chọn đỉnh f làm gốc, các bước áp dụng thuật toán tìm kiếm ưu tiên chiều rộng được trình bày trong hình 15.



Hình 15. Các bước tìm cây khung theo thuật toán ưu tiên chiều

5. Hệ chu trình độc lập

Theo tính chất 6 của cây, nếu T = (X,V) là cây khung của đồ thị liên thông G = (X,U) thì khi thêm một cạnh $u \in U \setminus V$ vào T được một chu trình đơn duy nhất C.

Định nghĩa: Tập các chu trình:

$$\Omega = \{C \mid C = C(X, V \cup \{u\}, u \in U \setminus V\}$$

gọi là hệ chu trình độc lập hay hệ chu trình cơ bản của đồ thị liên thông G = (X,U), trong đó T = (X,V) là cây khung của đồ thị G.

Nếu G = (X, U) là đồ thị vô hướng liên thông có N(X) = n và N(U) = m, khi đó cây khung T = (X, V) có N(V) = n – 1, nghĩa là khi xây dựng cây khung T đã loại m – n + 1 cạnh ra khỏi đồ thị G. Cứ thêm mỗi cạnh bị loại vào T sẽ có thêm 1 chu trình. Vậy, nếu ký hiệu số các chu trình độc lập của đồ thị G là $\mu(G)$, ta có:

$$\mu(G) = m - n + 1$$

Từ đó ta suy ra rằng, nếu G = (X, U) là đồ thị vô hướng có k thành phần liên thông thì:

$$\mu(G) = m - n + k$$

(vì mỗi thành phần liên thông của G có một cây khung).

Số $\mu(G)$ gọi là chu số của đồ thị.

Từ các diễn giải trên suy ra thuật toán tìm hệ chu trình độc lập của một đồ thị vô hướng liên thông như sau:

Bước 1: - Tìm cây khung T của đồ thị.

- Xác định tập các cạnh của đồ thị không nằm trong cây khung T:

$$W = \{u_1, u_2, \ldots, u_{m-n+1}\}$$

Bước 2: - Thêm cạnh u_1 vào T được chu trình thứ nhất.

- Loại cạnh u_1 vừa thêm vào T, thêm cạnh u_2 vào được chu trình thứ hai.
- Quá trình tiếp tục đến khi được m – n +1 chu trình.

Procedure He_chu_trinh_doc_lap;

Input: Đồ thị liên thông G = (X,U) có n đỉnh, m cạnh;

Ouput: Tập các chu trình độc lập;

Begin

T := (X,V) là cây khung của G;

W := U \ V;

Ω := ∅;

for i := 1 to m – n +1 do

begin

e := cạnh có trong W;

C := T ∪ {e};

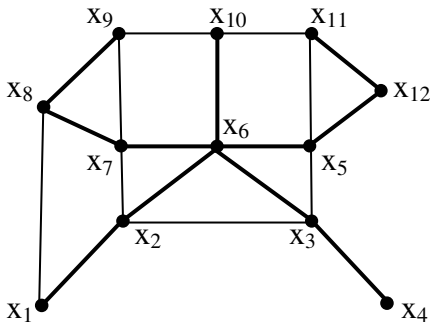
Ω := Ω ∪ C;

C := C \ {e};

end;

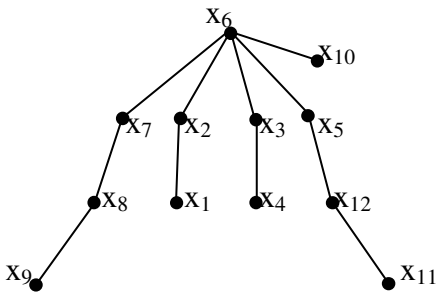
End;

Thí dụ: Tìm hệ chu trình độc lập của đồ thị ở hình 16a



Hình 16a

⇒



Hình 16b

Ta có: Số cạnh: m = 19;

Số đỉnh: n = 12;

Số chu trình độc lập là:

μ = 19 – 12 + 1 = 8;

Bước 1. Tìm cây khung T (hình 16b):

Các cạnh bị loại là: (x_1,x_8); (x_2,x_3); (x_2,x_7); (x_7,x_9); (x_9,x_{10}); (x_{10},x_{11}); (x_5,x_{11}); (x_3,x_5).

Bước 2. Tìm các chu trình cơ bản:

Thêm cạnh (x_1,x_8) được chu trình $C_1 = x_6\ x_2\ x_1\ x_8\ x_7\ x_6$.

Thêm cạnh (x_2,x_3) được chu trình $C_2 = x_6\ x_2\ x_3\ x_6$.
Thêm cạnh (x_2,x_7) được chu trình $C_3 = x_6\ x_2\ x_7\ x_6$.
Thêm cạnh (x_7,x_9) được chu trình $C_4 = x_7\ x_8\ x_9\ x_7$.
Thêm cạnh (x_9,x_{10}) được chu trình $C_5 = x_6\ x_7\ x_8\ x_9\ x_{10}\ x_6$.
Thêm cạnh (x_{10},x_{11}) được chu trình $C_6 = x_6\ x_5\ x_{12}\ x_{11}\ x_{10}\ x_6$.
Thêm cạnh (x_5,x_{11}) được chu trình $C_7 = x_5\ x_{11}\ x_{12}\ x_5$.
Thêm cạnh (x_3,x_5) được chu trình $C_8 = x_6\ x_3\ x_5\ x_6$.

6. Cây khung nhỏ nhất

Một trong các ứng dụng của lý thuyết đồ thị là giải các bài toán tối ưu. Chẳng hạn muốn xây dựng một mạng đường sắt nối n thành phố với nhau sao cho từ một thành phố nào đó có thể đi đến mọi thành phố khác bằng mạng đường sắt đó, thì vấn đề đặt ra là phải xây dựng sao cho khi khai thác mạng sẽ đạt lợi nhuận cao nhất. Hoặc để xây dựng một mạng máy tính gồm n máy ở các địa điểm khác nhau thì cần phải trải cáp như thế nào để tổng chi phí là rẻ nhất.

Các bài toán như vậy thường được giải nhờ khái niệm cây khung nhỏ nhất.

6.1. Định nghĩa:

Cho G = (X,U) là đồ thị vô hướng liên thông trong đó mỗi cạnh $u \in U$ được gán một số thực không âm l(u) gọi là trọng số của cạnh u. Cây khung T = (X,V) của đồ thị G có tổng trọng số các cạnh nhỏ nhất gọi là cây khung nhỏ nhất.

Tổng trọng số các cạnh của cây khung T gọi là trọng số của cây khung đó và ký hiệu là l(T), nghĩa là:

$$l(T) = \sum_{u \in V} l(u)$$

Cây khung T_0 là cây khung nhỏ nhất của đồ thị G nếu:

$$l(T_0) = \min\{\ l(T) \mid\ T \in\ T(G)\ \}$$

trong đó T(G) là tập các cây khung của G.

Sau đây là hai thuật toán cho phép tìm một cây khung nhỏ nhất: Thuật toán Kruskal và thuật toán Prim.

6.2. Thuật toán Kruskal:

Thuật toán sẽ xây dựng tập cạnh V của cây khung nhỏ nhất T = (X,V) theo từng bước. Trước hết sắp xếp các cạnh của đồ thị G theo thứ tự không giảm của trọng số. Bắt đầu từ V = ∅, sau đó ở mỗi bước lần lượt duyệt trong danh sách cạnh đã sắp xếp, từ cạnh có độ dài nhỏ đến cạnh có độ dài lớn hơn, để tìm ra cạnh mà việc bổ sung nó vào tập V không tạo thành chu trình trong tập này. Thuật toán sẽ kết thúc khi tập V có đủ n−1 cạnh. Cụ thể có thể mô tả thuật toán như sau:

Bước 1. Đặt T = (X, ∅).

Bước 2. Sắp xếp các cạnh của G theo thứ tự không giảm của trọng số.

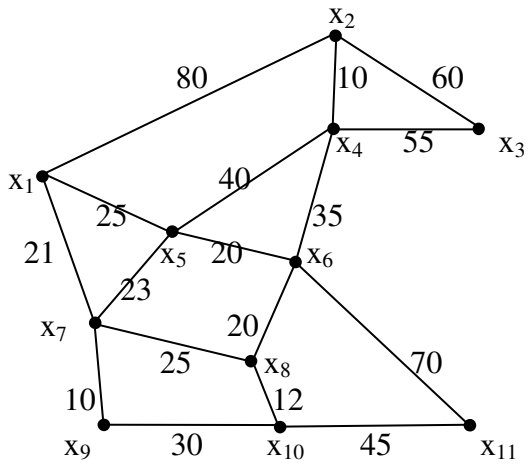
Bước 3. Bắt đầu từ cạnh đầu tiên của dãy này, thêm dần các cạnh của dãy đã được xếp vào T sao cho cạnh thêm vào không tạo thành chu trình trong T.

Bước 4. Lặp lại bước 3 cho đến khi nào số cạnh trong T bằng n −1 là được cây khung nhỏ nhất cần tìm.

```
Procedure  Kruskal;
Input:    Đồ thị G liên thông có n đỉnh và có trọng số;
Output: Cây khung nhỏ nhất T;
Begin
    T := đồ thị rỗng có n đỉnh là các đỉnh của G;
    for  i := 1  to  n –1  do
        begin
         u := cạnh có trọng số nhỏ nhất trong G và không tạo thành chu trình trong T
                  khi ghép vào T;
        T := T ∪ {u};
        end;
end.
```

Thí dụ 1: Tìm cây khung nhỏ nhất của đồ thị cho trong hình 17. Đồ thị có 11 đỉnh, 17 cạnh như vậy cây khung có 10 cạnh.

Hình 18 minh hoạ các bước thực hiện thuật toán.



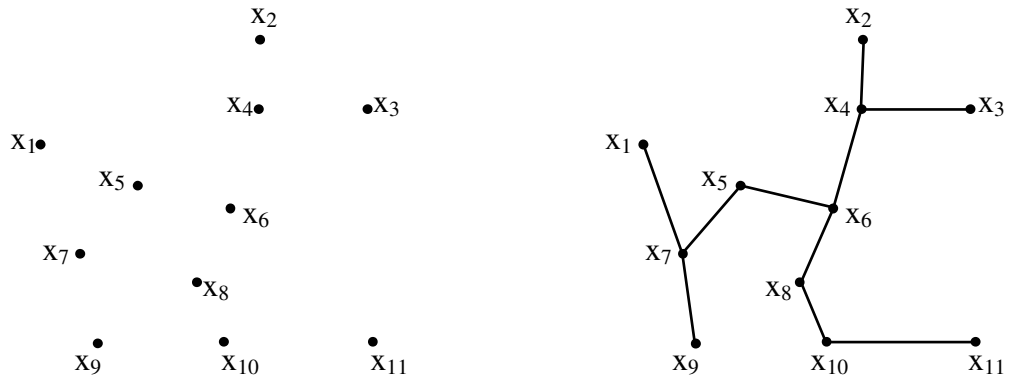
Hình 17

Bắt đầu từ đồ thị rỗng T có 11 đỉnh.

Bước sắp thứ tự các cạnh			Bước chọn các cạnh cho cây khung nhỏ nhất		
Thứ tự	Cạnh	Trọng số	Bước chọn	Cạnh	Trọng số
1	$x_2\ x_4$	10	1	$x_2\ x_4$	10
2	$x_7\ x_9$	10	2	$x_7\ x_9$	10

3	x_8 x_{10}	12	3	x_8 x_{10}	12
4	x_5 x_6	20	4	x_5 x_6	20
5	x_6 x_8	20	5	x_6 x_8	20
6	x_1 x_7	21	6	x_1 x_7	21
7	x_5 x_7	23	7	x_5 x_7	23
8	x_1 x_5	25	8	x_4 x_6	35
9	x_7 x_8	25	9	x_{10} x_{11}	45
10	x_9 x_{10}	30	10	x_3 x_4	55
11	x_4 x_6	35	Tổng trọng số các cạnh		251
12	x_4 x_5	40	Ghi chú Sau bước chọn 7 ta không thể chọn các cạnh (x_1,x_5), (x_7,x_8) và (x_9, x_{10}) vì các cạnh này tạo thành chu trình với các cạnh đã chọn. Tình huống tương tự nếu chọn cạnh (x_4,x_5) ở bước chọn 9.		
13	x_{10} x_{11}	45			
14	x_3 x_4	55			
15	x_2 x_3	60			
16	x_6 x_{11}	70			
17	x_1 x_2	80			

Hình 18a là bước đầu tiên của thuật toán. Kết thúc thuật toán được cây khung nhỏ nhất là cây trong hình 18b. Trọng số của cây khung nhỏ nhất thu được là l(T) = 251.



a) Bước 1: T = (X, ∅) b) Cây khung nhỏ nhất

Hình 18

6.3. Tính đúng đắn của thuật toán Kruskal.

Rõ ràng đồ thị thu được theo thuật toán có n đỉnh, n −1 cạnh và không có chu trình. Vì vậy theo định lý trong 1.2 nó là cây và là cây khung của đồ thị G. Như vậy chỉ còn phải chỉ ra rằng T có trọng số nhỏ nhất. Giả sử tồn tại cây khung S của đồ thị mà l(S) < l(T). Ký hiệu u_k là cạnh đầu tiên trong dãy các cạnh của T xây dựng theo thuật toán vừa mô tả không thuộc S. Khi đó đồ thị bộ phận của G sinh bởi cây S được bổ sung cạnh u_k sẽ chứa một chu trình duy nhất C đi qua u_k. Do chu trình C phải chứa cạnh u thuộc S và thuộc T nên đồ thị bộ phận S' thu được từ S bằng cách thay cạnh u của nó bằng u_k là cây khung.

Theo cách xây dựng, $l(u_k) \le l(u)$, do đó $l(S') \le l(S)$, đồng thời số cạnh chung của S' và T đã tăng thêm một so với số cạnh chung của S và T. Lặp lại quá trình trên từng bước một có thể biến đổi S thành T và trong mỗi bước tổng độ dài không tăng, tức là $l(T) \le l(S)$. Mâu thuẫn này chứng tỏ T là cây khung nhỏ nhất của G.

Độ phức tạp của thuật toán Kruskal được đánh giá như sau. Trước tiên, việc sắp xếp các cạnh của G theo thứ tự trọng số tăng dần có độ phức tạp $O(m^2)$, với m là số cạnh của G. Người ta chứng minh được rằng việc chọn cạnh u_{k+1} không tạo nên chu trình với k cạnh đã chọn trước đó có độ phức tạp là $O(n^2)$. Do $m \le \frac{n(n-1)}{2}$, thuật toán Kruskal có độ phức tạp là $O(n^2)$.

6.4. Thuật toán Prim tìm cây khung nhỏ nhất

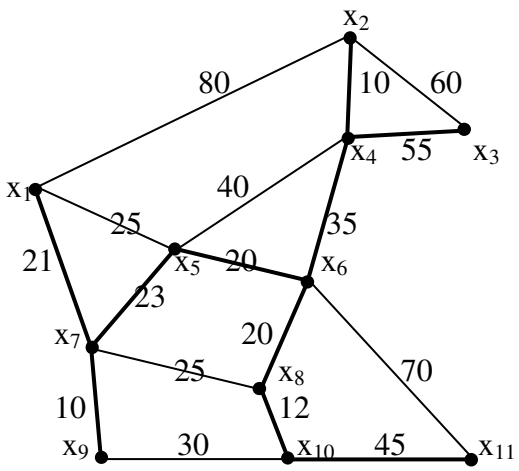
Thuật toán Kruskal làm việc kém hiệu quả đối với những đồ thị dày (đồ thị có số cạnh $m \approx \frac{n(n-1)}{2}$). Trong trường hợp đó, thuật toán Prim tỏ ra hiệu quả hơn. Thuật toán Prim còn được gọi là phương pháp lân cận gần nhất.

Chọn trong đồ thị đã cho cạnh có trọng số nhỏ nhất và đặt nó vào cây khung. Lần lượt ghép vào cây các cạnh có trọng số nhỏ nhất trong các cạnh liên thuộc với một trong các đỉnh của cây và không tạo ra chu trình trong cây. Thuật toán dừng khi có n – 1 cạnh được chọn.

```
Procedure  Prim;
Input:    Đồ thị liên thông  n đỉnh có trọng số;
Ouput:  Cây khung nhỏ nhất T của đồ thị;
Begin
    T := Cạnh có trọng số nhỏ nhất cùng 2 đỉnh liên thuộc;
    for  i :=1  to  n – 2  do
     Begin
              e := Cạnh có trọng số nhỏ nhất liên thuộc với một trong các đỉnh của T
                   và không tạo ra chu trình trong T;
              T :=  T ∪ e;
     end;   {Cây khung nhỏ nhất}
End.
```

Thí dụ 1: Trở lại thí dụ trong thuật toán Kruskal (đồ thị ở hình 17). Áp dụng thuật toán Prim có các bước như sau:

Bước	Cạnh	Trọng số
1	(x_7,x_9)	10
2	(x_7,x_1)	21
3	(x_7,x_5)	23
4	(x_5,x_6)	20
5	(x_6,x_8)	20
6	(x_8,x_{10})	12
7	(x_6,x_4)	35
8	(x_4,x_2)	10
9	(x_{10},x_{11})	45
10	(x_4,x_3)	55
l(T)		251



Hình 19

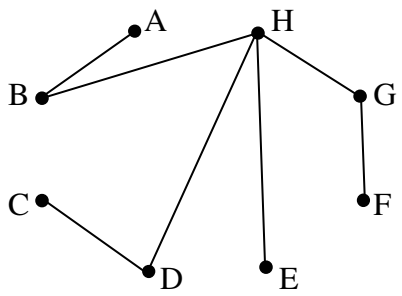
Cây khung nhỏ nhất thu được thể hiện bằng nét đậm trên hình 19 và trọng số của nó là l(T) = 251.

Thí dụ 3: Tìm cây khung nhỏ nhất của đơn đồ thị gồm các đỉnh A, B, C, D, E, F, G, H được cho bởi ma trận trọng số sau (ma trận trọng số giống như ma trận kề, nhưng thay số cạnh bằng trọng số của cạnh; các ô trống là không có cạnh nối):

	A	B	C	D	E	F	G	H
A	.	15	16	19	23	20	32	18
B	15	.	33	13	34	19	20	12
C	16	33	.	13	29	21	20	19
D	19	13	13	.	22	30	21	11
E	23	34	29	22	.	34	23	21
F	20	19	21	30	34	.	17	18
G	32	20	20	21	23	17	.	14
H	18	12	19	11	21	18	14	.

Đây là một đồ thị đầy đủ có 8 đỉnh với 28 cạnh, bởi vậy nên sử dụng thuật toán Prim. Do ma trận là đối xứng nên chỉ cần xét nửa trên hoặc nửa dưới của đường chéo là đủ.

Bước	Cạnh	Trọng số
1	(H, D)	11
2	(H, B)	12
3	(D, C)	13
4	(H, G)	14
5	(B, A)	15
6	(G, F)	17
7	(H, E)	21
l(T) =		103



Hình 20. Cây khung nhỏ nhất của đồ thị trong thí dụ 3.

6.5. Tính đúng đắn của thuật toán Prim

Xét đồ thị G = (X,U) và T = (X,V) là cây khung nhỏ nhất thu được theo thuật toán Prim.

Trước hết theo tính chất 2 của cây thì T là một cây và nó là cây khung của G vì nó đi qua mọi đỉnh của G. Vậy chỉ còn phải chứng minh T là cây khung nhỏ nhất của G.

Thật vậy, dùng phản chứng, giả sử T không phải là cây khung nhỏ nhất, khi đó phải tồn tại cây khung $T_1 = (X,V_1)$ là cây khung nhỏ nhất, nghĩa là:

$l(T_1) = \min\{ l(T) \mid T \in T(G) \}$, trong đó T(G) là tập các cây khung của G.

Chúng ta sẽ chỉ ra sự tồn tại cây khung T_2 có $l(T_2) < l(T_1)$.

Vì $l(T_1) < l(T)$ nên $V_1 \neq V$.

Giả sử cạnh $u_k \in V$, nhưng $u_k \notin V_1$, khi đó trong đồ thị $(X, V_1 \cup \{u_k\})$ có một chu trình duy nhất (theo tính chất 6 của cây). Gọi ω là chu trình có trong đồ thị $(X, V_1 \cup \{u_k\})$, vì trong T = (X,V) không có chu trình mà $u_k \in V$ nên tồn tại cạnh $u_0 \notin V$ và $u_0 \in \omega$.

Xét đồ thị $T_2 = (X, V_2)$, trong đó $V_2 = V_1 \cup \{u_k\} \setminus \{u_0\}$.

Rõ ràng T_2 không có chu trình vì ω là chu trình duy nhất có trong $(X, V_1 \cup \{u_k\})$ và $u_0 \in \omega$. Do T_1 là cây khung của G nên $N(V_1) = n - 1$ (theo tính chất 2 của cây), từ đó suy ra $N(V_2) = n - 1$. Vậy T_2 là cây khung của G. Mặt khác do $u_0 \notin V$ và $u_k \in V$ nên $l(u_0) > l(u_k)$. Ta có:

$$l(T_2) = \sum_{u \in V_2} l(u) = \sum_{u \in V_1} l(u) + l(u_k) - l(u_0) < \sum_{x \in V_1} l(u) = l(T_1)$$

Điều này trái với giả thiết T_1 là cây khung nhỏ nhất.

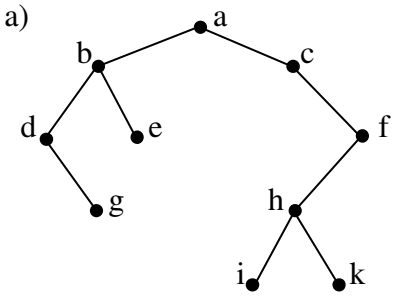
BÀI TẬP CHƯƠNG 5

5.1. Tìm số đỉnh tối đa của một cây m-phân có chiều cao h.

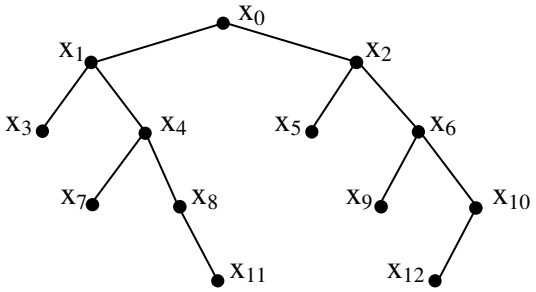
5.2. Đồ thị G là một rừng có k cây, n đỉnh. Tìm số cạnh của G.

5.3. Duyệt các cây sau theo các thuật toán tiền thứ tự, trung thứ tự và hậu thứ tự:

a)



b)



5.4. Một cây nhị phân có danh sách các đỉnh khi duyệt theo thuật toán tiền thứ tự là A, B, D, E, C, F, G và khi duyệt theo thuật toán trung thứ tự là D, B, E, A, F, G, C. Hãy vẽ cây nhị phân đó và duyệt cây này theo thuật toán hậu thứ tự.

5.5. Viết các biểu thức sau dưới dạng RPN:

a) ((A – B).C) + (D.E);

b) (a + (b.(c – d))) + e;

c) $x(y+\frac{z}{t^2}-u)$;

d) $\frac{(2+3)(4+1)}{(3-2)4+1}+\frac{2^2+3.2}{4^3-3.2}$.

5.6. Tính giá trị các biểu thức RPN sau:

a) 5 3 + 12 4 + * 6 2 * +;

b) 4 2 ^ 4 + 10 / 16 +;

c) 3 6 * 5 2 ^ + 2 3 + – ;

d) 3 2 * 2 ^ 5 3 – 8 4 / * – .

5.7. Ứng dụng cây giải các bài toán đếm sau bằng cách liệt kê tất cả các cấu hình có thể.

a) Hai đội bóng chuyền A và B thi đấu trong 5 ván. Đội nào thắng 3 trong 5 ván là thắng toàn trận. Hỏi cuộc thi có thể diễn ra theo bao nhiêu cách?

b) Có bao nhiêu xâu nhị phân có độ dài 4 không có 3 số 0 liên tiếp ?

c) Có bao nhiêu hoán vị 3 chữ cái a, b, c sao cho chữ b không đi liền sau chữ a?

d) Có bao nhiêu tập con của tập {3, 7, 9, 11, 24} sao cho tổng các phần tử trong tập con đó nhỏ hơn 28?

5.8. Xác định xem trong các bảng mã sau, mã nào là mã tiền tố:

a)

Ký tự	a	e	t	s
Mã	11	00	10	01

b)

Ký tự	a	e	t	s
Mã	0	1	01	001

c)

Ký tự	a	e	t	s	n
Mã	101	11	001	011	010

d)

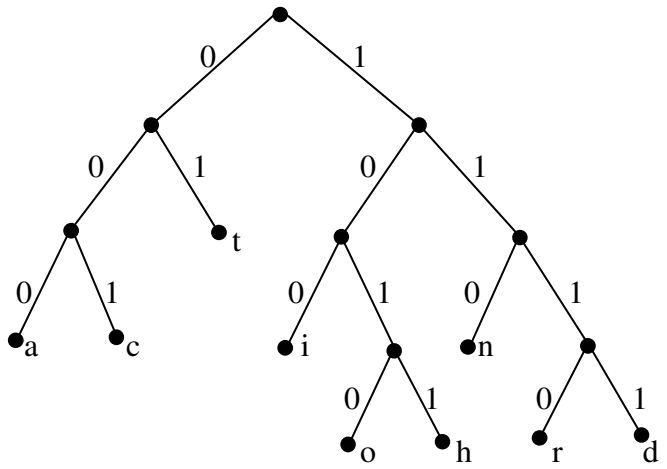
Ký tự	a	e	t	s	n	i
Mã	010	11	011	1011	1001	10101

5.9. Cho cây nhị phân biểu diễn mã tiền tố như hình vẽ.

a) Xác định mã của các chữ cái là nhãn các lá.

b) Xác định mã của các từ: hoc; toan; tin; dothi.

c) Xác định từ ứng với mã: 01 1010 000 110 1110 1010 100 1110 000 001.



Hình vẽ cho bài tập 5.9

5.10. Xây dựng cây mã Huffman cho tập ký tự sau:

a)

Ký tự	A	B	C	D	E	F	G	H
Tần suất (%)	6	25	20	8	10	19	3	9

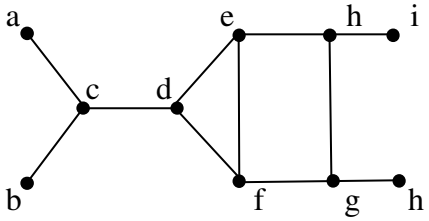
b)

Ký tự	a	c	d	i	h	o	n	r	t
Tần suất (%)	12	9	4	11	10	18	15	7	14

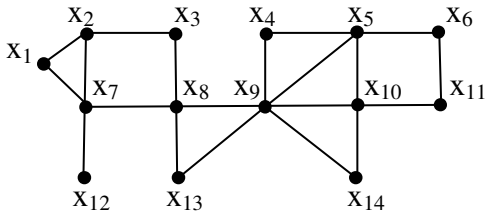
5.11. Tìm cây khung của các đồ thị sau theo:

a) Thuật toán ưu tiên chiều sâu.

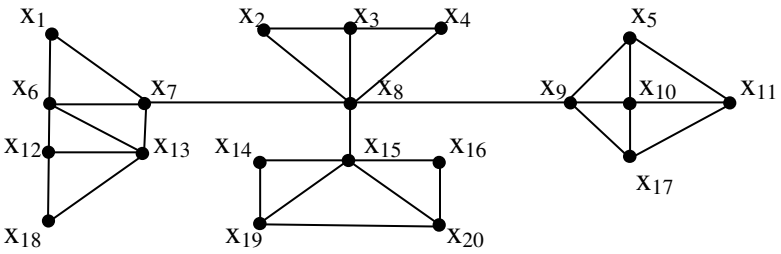
b) Thuật toán ưu tiên chiều rộng.



Đồ thị G_1



Đồ thị G_2



Đồ thị G_3

5.12. Câu hỏi như bài 5.11 đối với các đồ thị cho bằng ma trận kề (các ô trống là 0) sau:

a)

	A	B	C	D	E	F	G
A	.	1	.	.	1	.	.
B	1	.	1	.	.	.	1
C	.	1	.	1	.	1	.
D	.	.	1	.	1	.	1
E	1	.	.	1	.	.	.
F	.	.	1	.	.	.	.
G	.	1	.	1	.	.	.

b)

	1	2	3	4	5	6	7	8	9	10
1	.	1	.	.	1	1	.	.	.	.
2	1	.	1	.	.	.	1	.	.	.
3	.	1	.	1	.	.	.	1	.	.
4	.	.	1	.	1	.	.	.	1	.
5	1	.	.	1	.	1	.	.	.	1
6	1	.	.	.	.	.	.	1	1	.
7	.	1	.	.	.	.	.	.	1	1
8	.	1	.	.	.	1	.	.	.	1
9	.	.	.	1	.	1	1	.	.	.
10	.	.	.	.	1	.	1	1	.	.

5.13. Đồ thị có ma trận kề sau có liên thông không?

	1	2	3	4	5	6	7
1	.	.	1	.	1	.	.
2	.	.	.	1	.	1	.
3	1	.	.	.	1	.	.
4	.	1	.	.	.	1	1
5	1	.	1	.	.	.	1
6	.	1	.	1	.	.	1
7	.	.	.	1	1	1	.

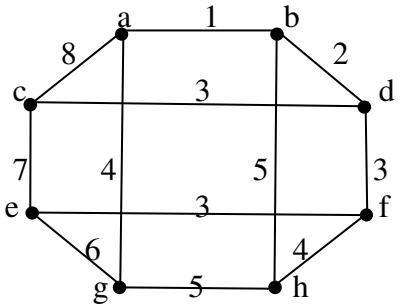
5.14. Cho đồ thị G = (X,U) với X = {x_2, x_3, …, x_{25}}, các đỉnh x_i, x_j ($2 \le i, j \le 25$) có cạnh nối nếu i, j không nguyên tố cùng nhau.

a) Đồ thi G có bao nhiêu thành phần liên thông?

b) Tìm cây bao trùm của mỗi thành phần liên thông của G.

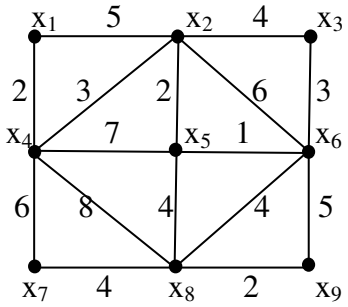
5.15. Tìm cây khung nhỏ nhất của các đồ thị sau bằng thuật toán Kruskal và sau đó bằng thuật toán Prim:

a)



Đồ thị G_1

b)



Đồ thị G_2

5.16. Tìm hệ chu trình độc lập của các đồ thị cho trong bài tập 5.15

5.17. Tìm cây khung nhỏ nhất của đồ thị có ma trận trọng số sau.

	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8
x_1	.	16	15	23	19	18	32	20
x_2	16	.	13	33	24	20	19	11
x_3	15	13	.	13	29	21	20	19
x_4	23	33	13	.	22	30	21	12
x_5	19	24	29	22	.	34	23	21
x_6	18	20	21	30	34	.	17	14
x_7	32	19	20	21	23	17	.	18
x_8	20	11	19	12	21	14	18	.

5.18. Cho đồ thị đầy đủ G có n đỉnh là 0, 1, 2, …, n – 1. Tìm cây khung nhỏ nhất của G nếu trọng số của các cạnh là:

a) l(i,j) = |i – j|; b) l(i,j) = i + j

Hãy vẽ cụ thể các cây khung tìm được.

ĐÁP SỐ

5.1. Số đỉnh tối đa ở mức i là m^i ($0 \le i \le h$) $\Rightarrow$ số đỉnh tối đa của cây m-phân có chiều cao h là: $\sum_{i=0}^{h} m^i = \frac{m^{h+1}-1}{m-1}$.

5.2. n – k .

5.3. a) a b d g e c f h i k; d g b e a c i h k f; g d e b i k h f c a.

5.4. duyệt theo hậu thứ tự: D E B G F C A.

5.5. a) A B – C * D E * + ; b) a b c d – * + e + ; c) x y z t 2 ^ / + u - * ;

d) 2 3 + 4 1 + * 3 2 – 4 * 1 + * / 2 2 ^ 3 2 * + 4 3 ^ 3 2 * – / + .

5.6. a) 140; b) 18; c) 38; d) 32

5.8. a), c), d): Là mã tiền tố; b) Không phải mã tiền tố

5.10. a) Một trong các mã là: a: 0010; b: 10; c: 01; d: 1100;
e: 000; f: 111; g: 0011; h: 1101.

5.13. Là đồ thị liên thông.

5.14. Có 5 thành phần liên thông, trong đó có 4 thành phần, mỗi thành phần chỉ gồm một đỉnh duy nhất, đó là : 13: 17: 19; 23.

5.18. a) Cây khung là đường nối lần lượt các đỉnh 0, 1, 2, …, n – 1.

b) Cây khung là đường nối đỉnh 0 với các đỉnh 1, 2, …, n – 1.

CÂU HỎI ÔN TẬP CHƯƠNG 5

1. Định nghĩa cây và rừng. Phát biểu và chứng minh các tính chất của cây.

2. Định nghĩa cây có gốc. Thế nào là đỉnh trong? là lá? mức của đỉnh? chiều cao của cây?. Nêu mối quan hệ giữa các đỉnh của cây có gốc.

3. Định nghĩa cây m-phân, cây m-phân đầy đủ. Trình bày các thuật toán duyệt cây nhị phân và ứng dụng để xây dựng ký pháp nghịch đảo Ba lan.

4. Mã tiền tố là gi? Cách xây dựng và kiểm tra mã tiền tố? Trình bày thuật toán Huffman để xây dựng mã tiền tố tối ưu.

5. Thế nào là cây khung của một đơn đồ thị? Điều kiện cần và đủ để một đơn đồ thị có cây khung? Các thuật toán tìm cây khung của một đơn đồ thị.

6. Thế nào là cây khung nhỏ nhất của một đơn đồ thị? Trình bày thuật toán Prim và thuật toán Kruskal tìm cây khung của một đơn đồ thị.

CHƯƠNG 6

MỘT SỐ BÀI TOÁN TỐI ƯU TRÊN ĐỒ THỊ

Một trong các ứng dụng của đồ thị là giải các bài toán tối ưu. Trong các chương trước chúng ta đã đề cập đến một số bài toán như vậy, chẳng hạn bài toán mã Huffman hay bài toán cây khung nhỏ nhất. Chương này đề cập đến một số bài toán tối ưu khác được giải nhờ lý thuyết đồ thị.

1. Đường đi ngắn nhất trong đồ thị

Có nhiều bài toán được mô hình hoá bằng **đồ thị có trọng số**. Đó là các đồ thị mà mỗi cạnh của nó được gán một số thực. Chẳng hạn một mạng giao thông đường bộ, trên đó mỗi cung đường (cạnh của đồ thị) được gán số chỉ khoảng cách giữa 2 địa điểm (2 đỉnh của đồ thị). Hoặc một mạng máy tính, giữa các máy có thể ghi thời gian truyền thông tin chẳng hạn. Những đồ thị có trọng số như vậy cần quan tâm đến các đường đi từ đỉnh x đến đỉnh y sao cho tổng các trọng số của đường này là nhỏ nhất, người ta gọi đó là đường đi ngắn nhất từ x đến y.

Nếu các cạnh của đồ thị đều có trọng số bằng 1 thì không cần ghi trọng số này lên các cạnh, và để cho tiện, các đồ thị như vậy được gọi là đồ thị không trọng số. Khi đó đường đi ngắn nhất chính là tổng số cạnh trên đường đi đó.

Các thuật toán tìm đường đi ngắn nhất trong đồ thị đều dựa trên cơ sở gán nhãn cho các đỉnh của đồ thị

1.1. Đường đi ngắn nhất trên đồ thị không trọng số

Giả sử cần tìm đường đi ngắn nhất từ đỉnh x cho trước đến mọi đỉnh còn lại của đồ thị G = (X, U) vô hướng, có hướng hoặc hỗn hợp, không trọng số. Khi đó cần thực hiện các bước sau:

Bước1. Bước cơ sở – Gán nhãn (đánh số) cho các đỉnh của đồ thị.

- Đánh số 0 cho đỉnh x.
- Nếu đỉnh x đã được gán nhãn là k thì gán nhãn k+1 cho mọi đỉnh y chưa được gán nhãn mà (x,y) là một cạnh hoặc một cung tuỳ theo đồ thị là vô hướng hay có hướng.

Như vậy, gọi A(k) là tập hợp các đỉnh được gán nhãn k (k = 0,1,2,…) thì:

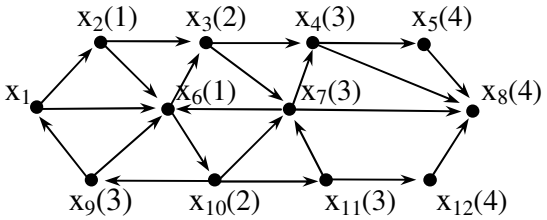
$$A(k+1) = \{ y \in X \mid y \notin A(i), i = 0,1,...,k \text{ và } \exists x \in A(k) \text{ sao cho } (x, y) \in U \}$$

Do đồ thị là hữu hạn nên sau một số hữu hạn bước mọi đỉnh của đồ thị được gán nhãn. Việc gán nhãn cho các đỉnh của đồ thị được hoàn thành.

Bước 2. Xác định đường đi ngắn nhất.

Giả sử đỉnh y nào đó của đồ thị được gán nhãn m, khi đó có đường đi gồm m cung từ x đến y, (và do đó m là độ dài đường đi ngắn nhất từ x đến y). Tất cả các đường đi ngắn nhất này là dãy các đỉnh $x, y_{m-1}, y_{m-2}, \ldots, y_1, y$; trong đó $y_{m-i} \in A(m-i)$, $i = 1, 2, \ldots, m-1$.

Thí dụ: Tìm đường đi ngắn nhất từ x_1 đến x_8 của đồ thị ở hình 1.



Hình 1

Bước 1. Ta có:

$A(0) = \{x_1\}$

$A(1) = \{x_2, x_6\}$

$A(2) = \{x_3, x_{10}\}$

$A(3) = \{x_4, x_9, x_{11}\}$

$A(4) = \{x_5, x_8, x_{12}\}$

Bước 2. Đi ngược từ A(4) về A(0), ta có các đường đi cần tìm là:

$x_1\ x_2\ x_3\ x_4\ x_8$; $x_1\ x_6\ x_3\ x_4\ x_8$; $x_1\ x_6\ x_{10}\ x_7\ x_8$; $x_1\ x_2\ x_3\ x_7\ x_8$; …

Tất cả các đường đi này đều có độ dài bằng 4.

1.2. Thuật toán Dijkstra tìm đường đi ngắn nhất trong đồ thị có trọng số

Xét đồ thị đơn vô hướng, có hướng hoặc hỗn hợp liên thông G = (X, U). Mỗi cạnh (cung) $u \in U$ được gán tương ứng một số không âm d(u) gọi là trọng số của u.

Giả sử x, y là 2 đỉnh của đồ thị G. Đặt D(x,y) là tập hợp tất cả các đường đi từ x đến y và $\forall \alpha \in D(x,y)$, đại lượng $l(\alpha) = \sum_{u \in \alpha} d(u)$ gọi là trọng số hay độ dài của đường α.

Đường đi α_0 thỏa mãn:

$$l(\alpha_0) = \min \{ l(\alpha) \mid \alpha \in D(x,y) \}$$

được gọi là đường đi ngắn nhất từ x đến y.

Thuật toán Dijkstra sau đây cho phép tìm các đường đi ngắn nhất từ một đỉnh x cho trước đến mọi đỉnh của đồ thị có trọng số G = (X,U).

Bước 1. Bước khởi gán trọng số cho các đỉnh. (Ký hiệu m(x) là trọng số của đỉnh x).

Đỉnh x được gán trọng số bằng 0, m(x) := 0. Các đỉnh khác của đồ thị được gán trọng số là một số dương đủ lớn sao cho nó lớn hơn tất cả các trọng số của mọi đường đi từ x đến đỉnh đó, cụ thể có thể gán $m(x_i) := \infty$, $\forall x_i \in X$, $x_i \neq x$.

Bước 2. Các bước lặp để giảm trọng số các đỉnh.

Giả sử đỉnh v được gán trọng số m(v).

- Nếu tìm được đỉnh z kề với v có trọng số m(z) thoả mãn $m(v) + d(z,v) < m(z)$ thì thay trọng số m(z) bởi $m'(z) = m(v) + d(z,v)$.
- Nếu không tìm được đỉnh z kề với v thoả mãn $m(v) + d(v,z) < m(z)$ thì trọng số m(z) được giữ nguyên.

Quá trình được tiếp tục cho tới khi trọng số của tất cả các đỉnh trong đồ thị G đạt cực tiểu, tức là $\forall v \in X$, $\bar{\exists} z \in X$, z kề với v sao cho $m(v) + d(z,v) < m(z)$.

Theo thuật toán này, trọng số của đường đi từ x đến y chính là trọng số của đỉnh y và nó là độ dài đường đi ngắn nhất từ x đến y.

Thật vậy, xuất phát từ đỉnh y, tìm tất cả các đỉnh kề với y, chẳng hạn đó là đỉnh y_k thoả mãn:

$$d(y_k, y) = m(y) - m(y_k)$$

Từ đỉnh y_k, tìm các đỉnh y_{k-1} kề với y_k thoả mãn:

$$d(y_{k-1}, y_k) = m(y_k) - m(y_{k-1})$$

Qúa trình tiếp tục cho đến khi về tới đỉnh y_1 kề với đỉnh x và ta có:

$$d(x, y_1) = m(y_1) - m(x), \text{ trong đó } m(x) = 0;$$

Đường đi theo thuật toán là:

$$\alpha = (x, y_1, y_2, \ldots, y_{k-1}, y_k, y)$$

và: $l(\alpha) = d(x,y_1) + d(y_1,y_2) + \ldots + d(y_{k-1},y_k) + d(y_k,y) =$

$= [m(y_1) - m(x)] + [m(y_2) - m(y_1)] + \ldots + [m(y_k) - m(y_{k-1})] + [m(y) - m(y_k)] =$

$= m(y) - m(x) = m(y).$

Giả sử có đường đi khác từ x đến y: $\beta = (x, z_1, z_2, \ldots, z_{r-1}, z_r, y)$.

Khi đó theo bước 2 của thuật toán ta có các bất đẳng thức:

$$d(x,z_1) \geq m(z_1) - m(x) = m(z_1)$$
$$d(z_1,z_2) \geq m(z_2) - m(z_1)$$
$$\ldots$$
$$d(z_{r-1},z_r) \geq m(z_r) - m(z_{r-1})$$
$$d(z_r,y) \geq m(b) - m(z_r)$$

Cộng vế với vế của các bất đẳng thức trên được:

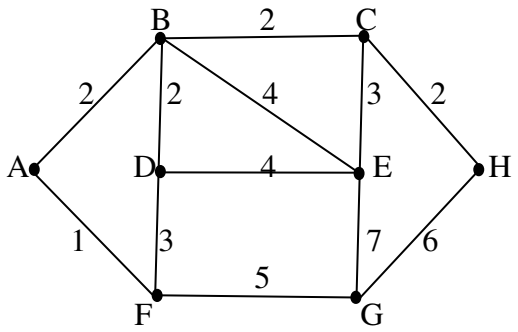
$$l(\beta) = d(x,z_1) + d(z_1,z_2) + \ldots + d(z_r,y) \geq m(y) = l(\alpha)$$

Vậy α là đường đi ngắn nhất từ x đến y.

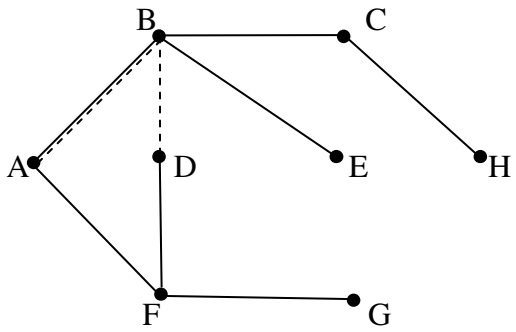
Có thể tóm tắt thuật toán Dijkstra như sau:

```
Procedure   Dijkstra;
Input:  G = (X,U) đồ thị liên thông, G có các đỉnh x, x1, …, xn và trọng số d(xi,xj)≥0, x
          là đỉnh xuất phát;
Output: Các đường đi ngắn nhất từ x đến các đỉnh x1, x2, …, xn và độ dài các đường đi
          đó;
 Begin
        {Bước khởi tạo trọng số các đỉnh}
 for i:=1 to n do
     begin
           m(xi) :=∞;      m(x) := 0;
           Truoc(xi) := x;
             {Truoc(xi) là đỉnh đứng ngay trước đỉnh xi trong đường đi tìm được}
        end;
     S := X \ {x};   {S là tập đỉnh có nhãn tạm thời}
     Truoc(x) := x;
              {Bước lặp, giảm trọng số các đỉnh}
     while  S ≠ ∅ do
      begin
       y := đỉnh thuộc S có trọng số nhỏ nhất;
       S := S \{y};  {Cố định nhãn của đỉnh y}
       for tất cả các đỉnh z  thuộc S   {Gán nhãn lại cho các đỉnh trong S}
           if  m(v) + d(v,z) < m(z)  then
                          begin
                              m(z) := m(v) + d(v,z);
                              trươc(z) := v;
                          end;
     end;
  End;
```

Thí dụ 1: Tìm đường đi ngắn nhất từ đỉnh A đến các đỉnh còn lại của đồ thị cho trong hình 2a.



Hình 2a



Hình 2b

Kết quả tính toán được trình bày thành bảng. Trong bảng, quy ước viết nhãn của đỉnh y có hai thành phần là m(y), trước(y) trong đó trước(y) là đỉnh đứng ngay trước đỉnh y trong đường đi đang tìm kiếm. Trong mỗi bước lặp đỉnh đánh dấu * là đỉnh được cố định nhãn trong bước đó.

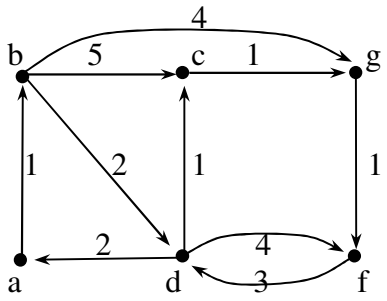
Bước lặp	Đỉnh A	Đỉnh B	Đỉnh C	Đỉnh D	Đỉnh E	Đỉnh F	Đỉnh G	Đỉnh H
Khởi tạo	0,A*	∞,A	∞,A	∞,A	∞,A	∞,A	∞,A	∞,A
1	-	2,A	∞,A	∞,A	∞,A	1,A*	∞,A	∞,A
2	-	2,A*	4,B	4,B∪F	6,B	-	6,F	∞,A
3	-	-	4,B	4,B∪F*	6,B	-	6,F	6,C
4	-	-	4,B*	-	6.B	-	6,F	6,C
5	-	-	-	-	6,B*	-	6,F	6,C
6	-	-	-	-	-	-	6,F*	6,C
7	-	-	-	-	-	-	-	6,C*

Từ đó các đường đi ngắn nhất từ đỉnh A đến các đỉnh là:

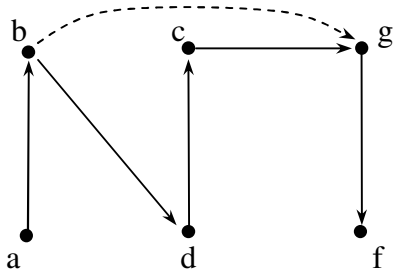
AB (độ dài 2); ABC (độ dài 4); ABD hoặc AFD (độ dài 4);
ABE (độ dài 6); AF (độ dài 1) ; AFG (độ dài 6);
ABCH (độ dài 6).

Có thể minh họa các đường đi này bằng hình vẽ (Hình 2b),

Thí dụ 2: Tìm đường đi ngắn nhất từ đỉnh a đến các đỉnh còn lại của đồ thị cho trong hình 3a.



Hình 3a



Hình 3b

Cũng như thí dụ 1 các kết quả tính toán được trình bày trong bảng sau:

Bước lặp	Đỉnh a	Đỉnh b	Đỉnh c	Đỉnh d	Đỉnh f	Đỉnh g
Khởi tạo	0,a	∞,a	∞,a	∞,a	∞,a	∞,a
1	-	1,a*	∞,a	∞,a	∞,a	∞,a
2	-	-	6,b	3,b*	∞,a	5,b
3	-	-	4,d*	-	6,g	5,b
4	-	-	-	-	6,g	5,b∪c*
5	-	-	-	-	6,g*	-

Các đường đi cần tìm là:

ab (độ dài 1); abdc (độ dài 4); abd (độ dài 3);

abdcgf hoặc abgf (độ dài 6); abg hoặc abdcg (độ dài 5); (Hình 3b).

Độ phức tạp của thuật toán Dijkstra qua các phép cộng và phép so sánh là $O(n^2)$.

Thật vậy. Trước hết thuật toán dùng không quá n – 1 bước lặp để thay đổi trọng số các đỉnh tức là để bớt các đỉnh trong tập đỉnh S. Số phép toán trong mỗi bước lặp không quá n – 1 phép so sánh và không quá n – 1 phép cộng để thay đổi trọng số các đỉnh thuộc tập S. Vậy mỗi bước lặp dùng không quá 2(n – 1) phép cộng và phép so sánh. Với n – 1 bước lặp thì số phép cộng và phép so sánh là không quá $(n – 1)2(n – 1) = O(n^2)$. Độ phức tạp của thuật toán là độ phức tạp đa thức bậc hai.

2. Tâm, Bán kính, Đường kính của đồ thị

Xét đồ thị vô hướng có trọng số dương G = (X, U). Với mọi x, y∈X, gọi D(x,y) là tập hợp tất cả các đường đi đơn nối giữa x và y. Đặt:

$$d(x,y) = \min \{ l(\beta) \mid \beta \in D(x,y)\}$$

Đồng thời quy ước:

$d(x,x) = 0$;

$d(x,y) = \infty$, nếu giữa x và y không liên thông (không có đường đi giữa x và y).

Đại lượng d(x,y) gọi là khoảng cách hay độ lệch giữa đỉnh x và đỉnh y.

Dễ thấy các tính chất của khoảng cách d(x,y):

- $d(x,y) \geq 0, \ \forall x,y \in X; \quad d(x,y) = 0 \Leftrightarrow x \equiv y.$
- $d(x,y) = d(y,x), \ \forall x,y \in X.$
- Thoả mãn bất đẳng thức tam giác: $d(x,y) + d(y,z) \geq d(x,z), \forall x,y,z \in X$.

Độ lệch tại đỉnh $x_0 \in X$, ký hiệu $\alpha(x_0)$ được định nghĩa là:

$$\alpha(x_0) = \max_{y \in X}\{d(x_0, y)\}$$

Dễ thấy:

$$\alpha(x_0) = \max_{y \in X}\left\{ \min_{\beta \in D(x_0,y)} \{l(\beta)\} \right\}$$

Định nghĩa:

Độ lệch nhỏ nhất của các đỉnh của đồ thị gọi là bán kính của đồ thị. Ký hiệu bán kính của đồ thị G là r(G), ta có:

$$r(G) = \min \{\alpha(x) \mid x \in X\}$$

Các đỉnh tại đó độ lệch đạt tối thiểu gọi là tâm của đồ thị.

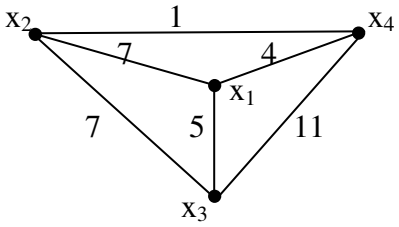
Độ dài của đường đi đơn dài nhất có trong đồ thị gọi là đường kính của đồ thị.

Nói cách khác: Đỉnh y_0 gọi là tâm của đồ thị nếu $\alpha(y_0) = \min_{x \in X}\{\alpha(x)\}$. Khi đó số $\alpha(y_0)$ gọi là bán kính của đồ thị và số $d = \max_{x,y \in X}\{l(\beta) \mid \beta \in D(x,y)\}$ gọi là đường kính của đồ thị.

Chú ý :

- Nếu đồ thị có đỉnh biệt lập thì nó không có tâm.
- Những đỉnh có độ lệch đối với tâm đạt tối đa được gọi là các đỉnh rìa của đồ thị.

Thí dụ: Xét đồ thị trong hình 4.



Hình 4

Sử dụng thuật toán Dijkstra tìm được đường đi ngắn nhất từ đỉnh x_1 đến các đỉnh còn lại là:

$l(x_1,x_2) = 5$; $l(x_1,x_3) = 5$; $l(x_1,x_4) = 4$

Vậy độ lệch của đỉnh x_1 là: $\alpha(x_1) = 5$;

Tương tự, độ lệch các đỉnh còn lại là:

$\alpha(x_2) = 7$; $\alpha(x_3) = 8$; $\alpha(x_4) = 8$.

Từ đó thấy rằng x_1 là tâm của đồ thị và bán kính của đồ thị bằng 5 .

Đường kính của đồ thị: $d_0 = 25$ (đó là độ dài của đường đi $x_1x_2x_3x_4$).

3. Mạng và Luồng

3.1. Các khái niệm

a Mạng

Mạng (còn gọi là mạng vận tải) là một đồ thị G = (X,U) có hướng, không có khuyên, trong đó có duy nhất một đỉnh không có cung đi vào gọi là đỉnh vào (còn gọi là điểm phát) và có duy nhất một đỉnh không có cung đi ra gọi là đỉnh ra (còn gọi là điểm thu), đồng thời trên mỗi cung $u = (x,y) \in U$ có gán một số không âm $c(u) = c(x,y)$ gọi là khả năng thông qua của cung u.

Sau đây ký hiệu đỉnh vào của mạng là x_0 và đỉnh ra là z.

Và quy ước: Nếu không có cung từ x đến y thì $c(x,y) = 0$.

Với mọi $x \in X$, ký hiệu:

Tập cung đi vào x là $U^-(x)$, tức là tập cung có điểm cuối là x.

Tập cung đi ra khỏi x là $U^+(x)$, tức là tập cung có điểm đầu là x.

b. Luồng

Xét mạng G = (X,U), người ta định nghĩa:

Hàm $\varphi : U \to R$ gán cho mỗi cung $u \in U$ một số thực không âm $\varphi(u)$ được gọi là luồng (còn gọi là luồng vận tải) của mạng G nếu nó thoả mãn các tính chất sau:

1) Lượng vật chất trên mỗi cung không vượt quá khả năng thông qua của cung đó:

$$0 \le \varphi(u) \le c(u), \quad \forall u \in U$$

2) Lượng vật chất đưa vào mỗi đỉnh bằng lượng vật chất đưa ra khỏi đỉnh đó:

$$\sum_{u \in U^-(x)} \varphi(u) = \sum_{v \in U^+(x)} \varphi(v), \quad \forall x \in X,\ x \ne x_0,\ x \ne z.$$

Tính chất này được gọi là điều kiện cân bằng luồng.

3) Từ tính chất 2 ta suy ra:

$$\sum_{u \in U^+(x_0)} \varphi(u) = \sum_{v \in U^-(z)} \varphi(v) = val(\varphi)$$

Đại lượng val(φ) gọi là giá trị luồng φ (còn gọi là luồng qua mạng hay cường độ luồng φ).

Giả sử $A \subset X$ và ký hiệu:

$U^-(A)$ là tập cung có điểm cuối thuộc A nhưng điểm đầu không thuộc A và gọi là tập cung đi vào A

$U^+(A)$ là tập cung có điểm đầu thuộc A nhưng điểm cuối không thuộc A, gọi là tập cung đi ra khỏi A.

Đối với tập cung $M \subseteq U$, đại lượng $\varphi(M) = \sum_{u \in M} \varphi(u)$ được gọi là luồng của tập cung M.

Từ điều kiện 2 của định nghĩa luồng, suy ra:

Nếu φ là luồng của mạng G = (X,U), thì đối với mọi $A \subset (X \setminus \{x_0, z\})$ đều có:

$$\varphi(U^-(A)) = \varphi(U^+(A))$$

3.2. Bài toán luồng cực đại

Phát biểu bài toán: Cho mạng G = (X,U). Hãy tìm luồng φ sao cho giá trị luồng val(φ) đạt cực đại.

Để giải bài toán này, trước hết cần tìm hiểu một số khái niệm.

a. Lát cắt

Lát cắt (A, A') là một phân hoạch tập đỉnh X của mạng thành hai tập A và A' = X \ A, trong đó đỉnh vào $x_0 \in A$ và đỉnh ra $z \in A'$. Đại lượng:

$$c(A,A') = \sum_{u \in U^-(A')} c(u) = \sum_{\substack{x \in A \\ y \in A'}} c(x,y) = \sum_{v \in U^+(A)} c(v)$$

được gọi là khả năng thông qua lát cắt.

Thiết diện là tên gọi khác của lát cắt.

Lát cắt có khả năng thông qua nhỏ nhất gọi là lát cắt hẹp nhất.

Từ định nghĩa lát cắt và khả năng thông qua của nó nhận thấy rằng: Mỗi đơn vị hàng hoá được chuyển từ đỉnh vào x_0 đến đỉnh ra z ít nhất cũng phải một lần thông qua một cung nào đó của lát cắt (A,A'). Vì vậy dù luồng φ và thiết diện (A,A') như thế nào đi nữa thì vẫn thoả mãn quan hệ:

$$val(\varphi) \leq c(A,A')$$

Từ đó thấy rằng: *Giá trị luồng cực đại trong mạng không vượt quá khả năng thông qua của lát cắt hẹp nhất.*

Người ta đã chứng minh được rằng: *Giá trị luồng cực đại bằng khả năng thông qua của lát cắt hẹp nhất:*

$$val(\varphi_{max}) = c(B, B')$$ với (B, B') là lát cắt hẹp nhất.

b. Luồng đầy

Cung u trong mạng vận tải G với luồng φ được gọi là cung bão hoà nếu φ(u)=c(u).

Luồng φ của mạng G được gọi là luồng đầy nếu mọi đường đi từ đỉnh vào x_0 đến đỉnh ra z đều chứa ít nhất một cung bão hoà. Ngược lại nếu tồn tại ít nhất một đường đi từ đỉnh vào x_0 đến đỉnh ra z mà mọi cung của đường đi đó đều chưa bão hòa thì luồng tương ứng gọi là luồng chưa đầy.

Nếu luồng φ trong mạng G chưa đầy thì nhất định tìm được đường đi α từ đỉnh vào x_0 đến đỉnh ra z không chứa cung bão hoà. Khi đó có thể tăng luồng φ thành luồng $\varphi_1(u)$ với:

$$\varphi_1(u) = \varphi(u) + \rho, \quad \text{nếu } u \in \alpha$$
$$\varphi_1(u) = \varphi(u), \quad \text{nếu } u \notin \alpha$$

trong đó $\rho = \min_{u \in \alpha}\{c(u) - \phi(u)\}$.

Dễ thấy φ_1 cũng là một luồng và giá trị của nó là: $val(\varphi_1) = val(\varphi) + \rho > val(\varphi)$.

Như vậy, đối với luồng không đầy có thể nâng giá trị của nó lên và nâng cho tới khi nhận được luồng đầy.

Tuy nhiên, thực tế cho thấy có thể có luồng đầy nhưng chưa đạt đến giá trị cực đại. Bởi vậy cần phải áp dụng thuật toán Ford – Fulkerson sau đây để tìm của luồng cực đại.

3.3. Thuật toán Ford-Fulkerson tìm luồng cực đại

Trước hết xét khái niệm xích: **Xích** trong một đồ thị có hướng là một đường đi trong đồ thị vô hướng tương ứng với đồ thị đã cho. (Đồ thị vô hướng có được từ đồ thị có hướng bằng cách bỏ đi chiều của các cung được gọi là đồ thị vô hướng tương ứng với đồ thị có hướng đã cho).

Thuật toán gồm các bước:

Bước 1. Gán nhãn cho các đỉnh của mạng.

Đỉnh vào x_0 được gán nhãn là 0.

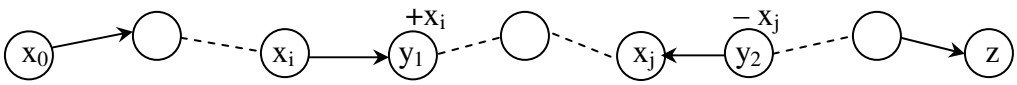
1) Nếu đỉnh x_i đã được gán nhãn thì dùng chỉ số $+x_i$ để gán nhãn cho mọi đỉnh y chưa được gán nhãn nếu có cung từ x_i đến y và cung này chưa bão hoà, nghĩa là đánh dấu cho mọi đỉnh y thoả mãn:

$$(x_i,y) \in U \text{ và } \varphi(x_i,y) < c(x_i,y)$$

2) Nếu đỉnh x_i đã được gán nhãn thì dùng chỉ số $-x_i$ để gán nhãn cho mọi đỉnh y chưa được gán nhãn nếu có cung từ y đến x_i và luồng của cung này dương, nghĩa là gán nhãn cho mọi đỉnh y thoả mãn:

$$(y, x_i) \in U \text{ và } \varphi(y, x_i) > 0$$

Nếu với cách gán nhãn này mà gán nhãn được đỉnh ra z thì trong mạng G tồn tại một xích α từ x_0 đến z có mọi đỉnh được gán nhãn theo chỉ số của đỉnh trước đó (chỉ khác nhau về dấu). Khi đó chắc chắn nâng được giá trị của luồng. Gọi xích α tìm được theo thuật toán là xích tăng luồng (hình 5).



Hình 5. Xích tăng luồng

Để thuận tiện cho việc sử dụng thuật toán. Chúng ta gọi các cung của xích α cùng chiều với đường đi từ đỉnh vào x_0 đến đỉnh ra z là cung thuận, còn các cung ngược chiều là cung nghịch. Chẳng hạn trên hình 5, cung (x_i,y_1) là cung thuận, cung (y_2,x_j) là cung nghịch.

Chú ý rằng đường tăng luồng được định nghĩa trong mục luồng đầy cũng là một xích tăng luồng gồm toàn các cung thuận.

Bước 2. Nếu tìm được xích tăng luồng α thì có thể nâng giá trị của luồng φ thành luồng φ_1 như sau:

- Với mỗi $u \in \alpha$, đặt:

$$m(u) = \begin{cases} c(u) - \varphi(u), & \text{nếu u là cung thuận} \\ \varphi(u), & \text{nếu u là cung nghịch} \end{cases}$$

và: $\rho = \min \{ m(u) \mid u \in \alpha \}$.

Khi đó:

- Nếu $u \notin \alpha$ thì đặt: $\varphi_1(u) = \varphi(u)$;
- Nếu $u \in \alpha$ và u là cung thuận thì đặt: $\varphi_1(u) = \varphi(u) + \rho$;
- Nếu $u \in \alpha$ và u là cung nghịch thì đặt: $\varphi_1(u) = \varphi(u) - \rho$.

Đối với φ_1 cả 3 điều kiện của luồng đều thoả mãn nên φ_1 cũng là một luồng và ta có:

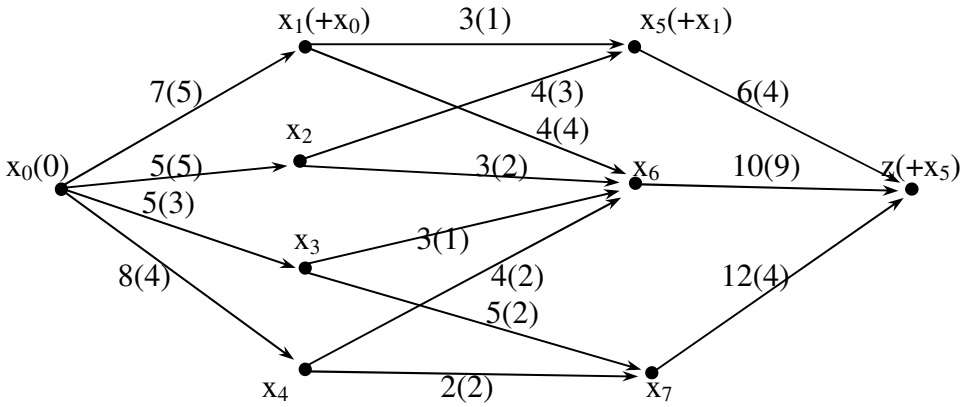
$$val(\varphi_1) = val(\varphi) + \rho$$

Như vậy ta đã nâng giá trị của luồng thêm ρ đơn vị.

Bước 3. Lặp lại các bước 1 và 2 đến khi được luồng φ_n không thể gán nhãn được đỉnh ra z thì quá trình nâng giá trị luồng kết thúc và φ_n là luồng cực đại đồng thời $val(\varphi_n)$ là giá trị của luồng cực đại. Vì khả năng thông qua của các cung là hữu hạn nên quá trình phải dừng sau một số hữu hạn bước.

Gọi A là tập đỉnh đã được gán nhãn (trong trường hợp không thể gán nhãn được đỉnh ra z, tức là quá trình nâng luồng kết thúc) và A' = X \ A thì (A,A') là một trong các lát cắt hẹp nhất của mạng.

Thí dụ: Cho mạng vận tải như hình 6 với khả năng thông qua của cung được ghi trên các cung và luồng ban đầu được ghi trong dấu ngoặc đơn .



Hình 6

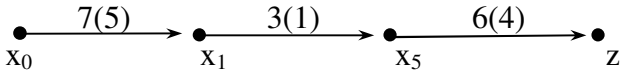
Luồng ban đầu là:

$\varphi = \{\varphi(x_0,x_1), \varphi(x_0,x_2), \varphi(x_0,x_3), \varphi(x_0,x_4), \varphi(x_1,x_5), \varphi(x_1,x_6), \varphi(x_2,x_5), \varphi(x_2,x_6), \varphi(x_3,x_6), \varphi(x_3,x_7), \varphi(x_4,x_6), \varphi(x_4,x_7), \varphi(x_5,z), \varphi(x_6,z), \varphi(x_7,z)\} =$

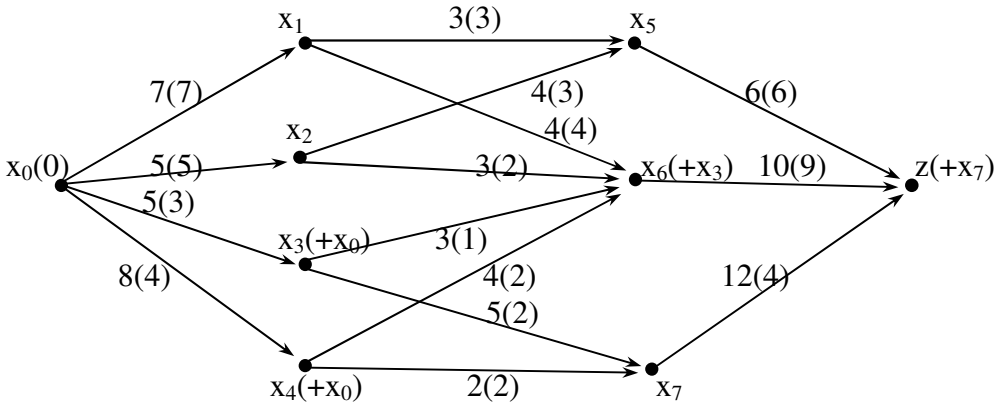
$= \{5, 5, 3, 4, 1, 4, 3, 2, 1, 2, 2, 2, 4, 9, 4\}.$

$$val(\varphi) = \sum_{u \in U^+(x_0)} \varphi(u) = \varphi(x_0,x_1) + \varphi(x_0,x_2) + \varphi(x_0,x_3) + \varphi(x_0,x_4) = 5+5+3+4 = 17$$

Bước lặp 1: Đỉnh x_0 được gán nhãn 0, các đỉnh x_1 được gán nhãn $+x_0$ vì các cung (x_0,x_1) chưa đầy. Tiếp theo cung (x_1,x_5) chưa đầy nên đỉnh x_5 được gán nhãn $+x_1$. Đỉnh z kề với x_5 và cung (x_5,z) chưa đầy nên z được gán nhãn $+x_5$ (hình 6). Do đỉnh ra đã được gán nhãn nên có thể kết thúc việc gán nhãn cho các đỉnh và được xích tăng luồng α_1: $x_0x_1x_5z$ gồm toàn các cung thuận nên nó cũng là đường tăng luồng (hình 7).



Hình 7

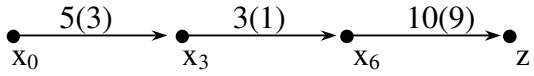


Hình 8

Ta có $\rho_1 = \min\{7 - 5, 3 - 1, 6 - 4\} = 2$, vì thế có thể nâng luồng φ của các cung của đường này thêm 2 đơn vị và được luồng φ_1 (hình 8):

$$\varphi_1 = \{7, 5, 3, 4, 3, 3, 4, 2, 1, 2, 2, 2, 6, 9, 4\};\ \ val(\varphi_1) = val(\varphi) + 2 = 19$$

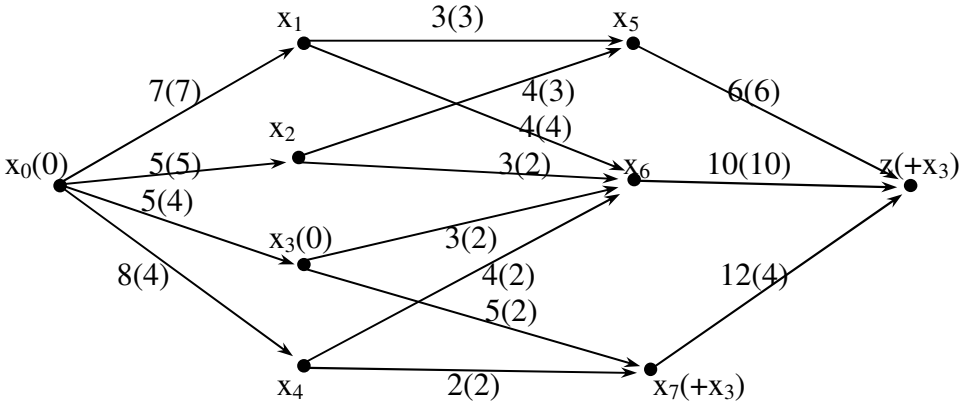
Bước lặp 2: Xóa các nhãn đã gán, gán nhãn mới cho các đỉnh (hình 8) được xích tăng luồng α_2: $x_0x_3x_7$. Xích α_2 cũng toàn cung các cung thuận (hình 9).



Hình 9

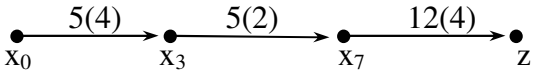
Từ đó $\rho_2 = \min\{5 - 3, 3 - 1, 10 - 9\} = 1$. Luồng φ_1 được nâng thêm 1 đơn vị thành luồng φ_2 (hình 10).

$$\varphi_2 = \{7, 5, 4, 4, 3, 3, 4, 2, 2, 2, 2, 2, 6, 10, 4\};\ \ val(\varphi_2) = val(\varphi_1) + 1 = 20$$



Hình 10

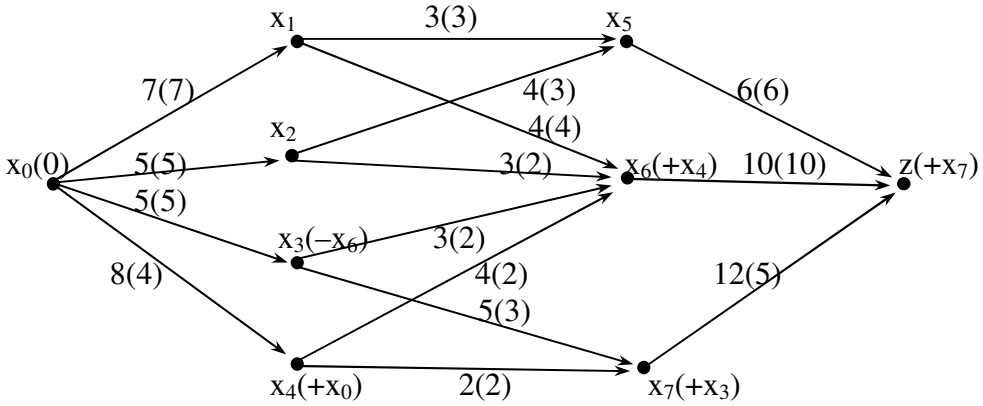
Bước lặp 3: Xóa các nhãn đã gán và gán lại nhãn cho các đỉnh (hình 10) được xích tăng luồng α_3: $x_0x_3x_7z$ gồm toàn cung thuận (hình 11).



Hình 11

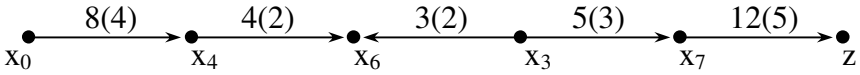
Xích α_3 cho phép nâng luồng φ_2 thêm 1 đơn vị ($\rho_3 = 1$) và được luồng φ_3 (hình 12):

$$\varphi_3 = \{7, 5, 5, 4, 3, 3, 4, 2, 2, 2, 3, 2, 6, 10, 5\};\ \ val(\varphi_3) = val(\varphi_2) + 1 = 21$$



Hình 12

Bước lặp 4: Xóa các nhãn đã gán và gán lại nhãn cho các đỉnh (hình 12) được xích tăng luồng α_4: $x_0x_4x_6x_3x_7z$, trong đó cung (hình 13).

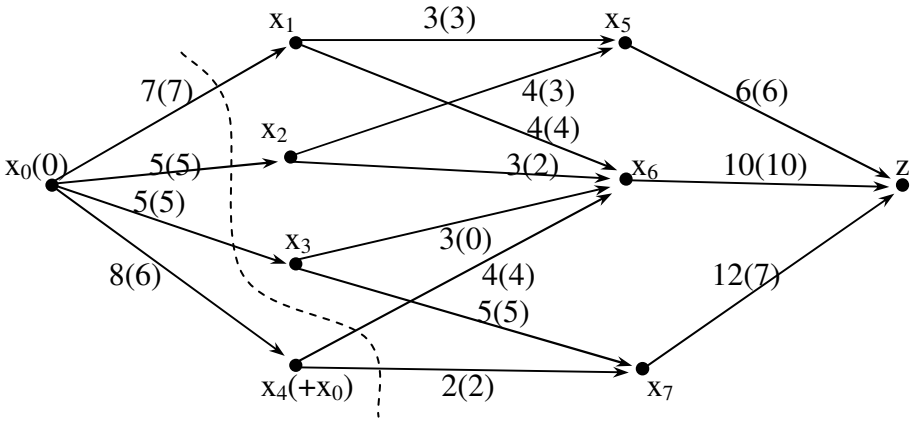


Hình 13

Ta có: $\rho_4 = \min\{8-4, 4-2, 2, 5-2, 12-5\} = 2$.

Xích α_4 cho phép nâng luồng φ_3 thêm 2 đơn vị và được luồng φ_4 (hình 12):

$$\varphi_4 = \{7, 5, 5, 6, 3, 3, 4, 2, 0, 4, 5, 2, 6, 10, 7\};\ val(\varphi_4) = val(\varphi_3) + 2 = 23$$



Hình 14

Tiếp theo chỉ có thể gán nhãn được đỉnh x_0 và đỉnh x_4 vì các cung khác đi ra khỏi x_0 và mọi cung đi ra khỏi x_4 đã bão hoà. Vậy quá trình nâng luồng kết thúc, và được luồng cực đại:

$$\varphi_{max} = \varphi_4;\ val(\varphi_{max}) = val(\varphi_4) = 23.$$

Vì bước cuối cùng chỉ gán nhãn được cho các đỉnh x_0 và x_4 nên lát cắt hẹp nhất là (A,A') với A={x_0,x_4} và A' = {x_1,x_2,x_3,x_5,x_6,x_7,z}. Rõ ràng $c(A,A') = c(x_0,x_1)+c(x_0,x_2) + c(x_0,x_3) + c(x_4,x_6) + c(x_4,x_7) = 7+5+5+4+2 = 23$, nghĩa là $val(\varphi_{max}) = c(A,A')$.

Chú thích: Bài toán mạng và luồng có thể có nhiều lát cắt hẹp nhất, chẳng hạn lát cắt (B,B') với B = $\{x_0,x_1,x_2,x_3,x_4,x_5,x_6\}$ cũng là lát cắt hẹp nhất trong thí dụ trên.

4. Bài toán du lịch

4.1. Phát biểu bài toán

Một người xuất phát từ một thành nào đó trong n thành phố được đánh số 1, 2, …, n để đi thăm n –1 thành phố còn lại, mỗi thành phố đúng một lần rồi trở về thành phố xuất phát. Hỏi phải chọn hành trình theo thứ tự nào để tổng chi phí là ít nhất, biết chi phí đi từ thành phố i đến thành phố j là c_{ij} (i, j = 1, 2, …, n; c_{ij} và c_{ji} không nhất thiết bằng nhau).

Rõ ràng mỗi hành trình như vậy là một hoán vị ($\alpha(1), \alpha(2), \ldots, \alpha(n)$) của n số tự nhiên dương đầu tiên. Như vậy phải chọn lựa trong n! hành trình khác nhau đó ra một hành trình tối ưu theo nghĩa tổng chi phí là nhỏ nhất. (Chú ý rằng có (n –1)! hành trình thực sự khác nhau trong n! hành trình, vì có thể cố định một thành phố để xuất phát).

Mô hình hóa bài toán bằng đồ thị G = (X,U) đầy đủ có hướng hai chiều n đỉnh và có trọng số. Trong đó X = {1, 2, …, n} và c_{ij} là trọng số của cung (i, j) với c_{ij} và c_{ji} không nhất thiết bằng nhau. Như vậy ma trận trọng số của đồ thị là ma trận vuông M= (c_{ij}) cấp n không đối xứng. Ma trận M được gọi là ma trận chi phí. Mỗi hành trình của bài toán du lịch tương ứng với một chu trình Hamilton trong đồ thị G, và phải tìm hành trình Hamilton ngắn nhất.

Gọi H là tập hợp mọi chu trình Hamilton (hành trình) có trong đồ thị. Với mọi $h \in H$ ta có $f(h) = \sum_{(i,j)\in h} c_{ij}$ là độ dài của chu trình h và cũng chính là tổng chi phí của hành trình tương ứng. Hàm f(h) gọi là hàm mục tiêu và phải tìm chu trình h_0 sao cho:

$$f(h_0) = \min \{f(h) \mid h \in H\} \qquad (1)$$

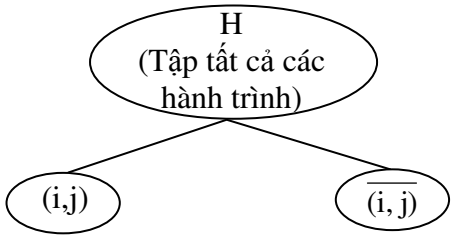
Hành trình h_0 thỏa mãn (1) được gọi là phương án tối ưu của bài toán.

Dễ thấy f(h) là tổng của n số hạng, trong đó mỗi hàng và mỗi cột của ma trận chi phí M có đúng một phần tử là số hạng để tính f(h).

Bài toán này được giải theo thuật toán nhánh cận dưới đây.

4.2. Thuật toán nhánh cận giải bài toán du lịch

Ý tưởng cơ bản của thuật toán là tìm kiếm lời giải của bài toán (phương án tối ưu h_0) bằng cách phân chia tập tất cả các hành trình thành hai tập con: một tập gồm các hành trình chứa một cung (i, j) nào đó còn tập kia là các hành trình không chứa cung này. Việc làm này gọi là **phân nhánh** tập hành trình H, mỗi tập con là một nhánh hay một nút của cây tìm kiếm. Để cho tiện, tập hành trình chứa cung (i, j) được ký hiệu là (i,j) còn tập hành trình không chứa cung (i,j) được ký hiệu là $\overline{(i, j)}$. Hình 15 là minh họa nút đầu tiên của cây tìm kiếm.



Hình 15

Sau khi phân nhánh thì tìm **cận dưới** của hàm mục tiêu cho mỗi tập. Việc tìm kiếm được tiếp tục bằng cách phân nhánh tập con có giá trị hàm mục tiêu nhỏ hơn. Thủ tục này được lặp lại cho đến khi tìm được hành trình tối ưu h_0 và nó chắc chắn sẽ kết thúc vì số hành trình là hữu hạn (tập H là tập hữu hạn).

Từ đây cho đến hết thuật toán, chúng ta ký hiệu cận dưới hàm mục tiêu của tập (i,j) là $f_{min}(i,j)$ và của tập $\overline{(i,j)}$ là $f_{\overline{min}}(i,j)$.

Như vậy, kỹ thuật cơ bản của thuật toán là phân nhánh và tìm cận dưới của hàm mục tiêu của các hành trình.

Thuật toán được tiến hành bằng thủ tục rút gọn ma trận trọng số $M = (c_{ij})$ của đồ thị được trình bày sau đây.

a. Rút gọn ma trận trọng số

Cho ma trận vuông A, nếu trừ các phần tử mỗi hàng (hoặc cột) của A cho phần tử nhỏ nhất của hàng (cột) đó thì trong hàng (cột) đó có ít nhất một phần tử bằng không. Phần tử nhỏ nhất được chọn của hàng (cột) gọi là phần tử rút gọn của hàng (cột) đó. Sau khi thực hiện các phép rút gọn đối với tất cả các hàng (cột) của ma trận A, nếu còn cột (hàng) nào có tất cả các phần tử đều khác không thì tiếp tục rút gọn cột (hàng) đó. Cuối cùng được một ma trận A' mà mỗi hàng, mỗi cột có ít nhất một phần tử bằng không. Ma trận A' gọi là ma trận rút gọn của ma trận A và tổng tất cả các hằng số rút gọn của các hàng và của các cột được gọi chung là hằng số rút gọn ma trận A. Sau đây ta ký hiệu S_A để chỉ hằng số rút gọn ma trận A.

Thí dụ:

$$A = \begin{pmatrix} 5 & 2 & 3 \\ 8 & 4 & 1 \\ 6 & 4 & 3 \end{pmatrix} \begin{matrix} 2 \\ 1 \\ 3 \end{matrix} \rightarrow \begin{pmatrix} 3 & 0 & 1 \\ 7 & 3 & 0 \\ 3 & 1 & 0 \end{pmatrix} \begin{matrix} \\ \\ \\ 3 \end{matrix} \begin{matrix} \\ \\ \\ 0 \end{matrix} \begin{matrix} \\ \\ \\ 0 \end{matrix} \rightarrow A' = \begin{pmatrix} 0 & 0 & 1 \\ 4 & 3 & 0 \\ 0 & 1 & 0 \end{pmatrix}$$

Hằng số rút gọn ma trận là $S_A = 2+1+3+3 = 9$.

Do mỗi hàng và mỗi cột của ma trận chi phí M chỉ có một phần tử nằm trong một hành trình du lịch nên nếu rút gọn ma trận M thì hàm mục tiêu của mọi hành trình đều cùng giảm đi một đại lượng đúng bằng hằng số rút gọn S_M của M và:

$$f(h) \geq S_M \, , \forall h \in H$$

Vậy $f_{min} \geq S_M$ (S_M là cận dưới của mọi hành trình).

Chú ý đặt $c_{ii} = \infty$ trong ma trận chi phí ban đầu.

b. Thủ tục phân nhánh

Như đã trình bày ở trên, phân nhánh là phân tập các hành trình thành hai tập con (i,j) là tập các hành trình chứa cung (i, j) và $\overline{(i,j)}$ là tập các hành trình không chứa cung (i, j). Điều này có nghĩa là sự phân nhánh diễn ra tại một trong hai đỉnh i hoặc j. Bởi vậy phải chọn cặp thành phố nào có chi phí nhỏ nhất để phân nhánh. Trên ma trận chi phí rút gọn

M' =(c'_{ij}) các cặp thành phố đó tương ứng với $c'_{ij} = 0$, và các hành trình chứa cung (i, j) đều có triển vọng tốt. Nói cách khác các tập hành trình thuộc tập con (i,j) sẽ chứa cung (i, j) và các hành trình không chứa cung (i, j) sẽ thuộc tập $\overline{(i, j)}$.

Các hành trình thuộc tập $\overline{(i, j)}$ không được chứa cung (i, j) nên có thể cấm việc đi qua cung này bằng cách đặt $c'_{ij} = \infty$ và được ma trận chi phí $M_{\overline{ij}}$ nếu chọn cung tiếp theo của hành trình trong tập này. Cận dưới của các hành trình theo hướng này là $f_{\overline{min}} \geq S_M + S_{M_{\overline{ij}}}$.

Các hành trình thuộc tập (i,j) luôn luôn chứa cung (i, j) bởi vậy có thể xóa hàng i cột j của ma trận rút gọn M' để được ma trận vuông giảm một cấp so với ma trận M (vì chỉ được phép chọn ở mỗi hàng mỗi cột của M một phần tử). Hơn nữa khi đã đi theo cung (i, j) thì không được phép đi theo cung (j, i) nữa nên phải đặt $c'_{ji} = \infty$ để cấm đi theo cung (j, i). Cuối cùng được ma trận vuông M_{ij} cấp n – 1 là ma trận chi phí cho các cung được chọn tiếp theo. Thủ tục phân nhánh được tiếp tục khi rút gọn ma trận chi phí M_{ij}, và cận dưới của các hành trình chứa cung (i, j) là $f_{min} \geq S_M + S_{M_{ij}}$.

Thủ tục phân nhánh kết thúc khi cấp của ma trận M_{ij} bằng 2. Hai cung cuối cùng được chọn trực tiếp từ ma trận M_{ij} cuối cùng.

Vấn đề đặt ra là thường có nhiều phần tử $c'_{ij} = 0$ nên cần phải chọn phần tử nào tức là chọn cung (i, j) nào để việc rẽ nhánh tương ứng nhanh dẫn đến phương án tối ưu. Lẽ đương nhiên là nên chọn cung nào để có cận dưới tăng nhanh nhất. Do cận dưới là tổng các hằng số rút gọn của ma trận rút gọn nên phải chọn phần tử $c'_{ij} = 0$ có tổng phần tử nhỏ nhất của hàng i với phần tử nhỏ nhất của cột j, trừ phần tử c'_{ij}, là lớn nhất.

c. Thủ tục ngăn chặn hành trình con

Như đã biết, chu trình Hamilton là không chứa chu trình con, vì nếu không sẽ có đỉnh đi qua hai lần. Bởi vậy phải loại bỏ hành trình con trong quá trình tìm kiếm hành trình tối ưu. Phương pháp tốt nhất là trong từng bước của thuật toán nhánh cận cần tìm cách ngăn chặn không để cho các hành trình con có khả năng xuất hiện.

Vì chu trình con chỉ xuất hiện khi hành trình có từ ba cạnh trở lên, nên khi đã chọn được hai cung (i, j), (j, k) thì không được chọn cung (k, i) vào hành trình, nếu không sẽ có hành trình con (i, j, k, i). Điều này được thực hiện bằng cách đặt $c'_{ki} = \infty$.

Một cách tổng quát nếu đã chọn được các đỉnh $i_1, i_2, \ldots, i_k$ vào hành trình và các đỉnh này theo thứ tự tạo thành một đường đi thì không được chọn tiếp cung (i_k, i_s) với s = 1, 2, …, k – 1 bằng cách đặt $c'_{ks} = \infty$ trừ khi (i_k, i_s) là cung cuối cùng được chọn vào hành trình.

d. Tóm tắt các bước của thuật toán

1. Rút gọn ma trận chi phí M =(c_{ij}) được ma trận rút gọn M' = (c'_{ij}). Tính cận dưới của mọi hành trình.

$$f_{min} \geq S_M = \text{Hằng số rút gọn ma trận M}$$

Chú ý: Trong bước lặp đầu tiên thì $f_{\overline{min}} = f_{min}$ và đặt $c_{ii} = \infty$.

2. Với mỗi phần tử $c'_{ij} = 0$ đặt:

σ_{ij} = Tổng phần tử nhỏ nhất của hàng i và phần tử nhỏ nhất của cột j (trừ c'_{ij}).

Chọn cung (i, j) có σ_{ij} lớn nhất để rẽ nhánh.

a) Đặt $c'_{ij} = \infty$ được ma trận chi phí $M_{\overline{ij}}$ cho các hành trình không chứa cung (i, j), tính $f_{\overline{min}} \geq f_{\overline{min}}$ + Hằng số rút gọn ma trận $M_{\overline{ij}}$.

b) Xóa hàng i cột j của ma trận M', đặt $c'_{ji} = \infty$ và đặt $c'_{ks} = \infty$ (nếu chọn cung (k, s) sẽ tạo thành chu trình con) được ma trận chi phí M_{ij} cho các chặng tiếp theo của các hành trình chứa cung (i, j).

$$f_{min} \geq f_{min} + \text{Hằng số rút gọn ma trận } M_{ij}$$

3. Lặp lại từ bước 1 đối với ma trận chi phí M_{ij} cho đến khi cấp của ma trận M_{ij} bằng 2. Hai cung cuối của hành trình được chọn trực tiếp từ ma trận M_{ij} cuối cùng. Khi kết thúc, ta có $f_{min} = f_{min}$ + Hằng số rút gọn của ma trận cấp 2 cuối.

Thí dụ: Giải bài toán du lịch với ma trận chi phí sau (phần tử để trống là không có cung tương ứng, có thể xem phần tử đó bằng 0):

$$M = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{array} \begin{array}{c} \begin{array}{cccccc} 1 & 2 & 3 & 4 & 5 & 6 \end{array} \\ \begin{pmatrix} . & 5 & 7 & 9 & 12 & 15 \\ 8 & . & 8 & 10 & 13 & 17 \\ 15 & 12 & . & 9 & 7 & 3 \\ 8 & 18 & 25 & . & 12 & 9 \\ 10 & 12 & 14 & 16 & . & 18 \\ 20 & 16 & 14 & 12 & 10 & . \end{pmatrix} \end{array}$$

Giải: Thay $c_{ii} = \infty$, i = 1, 2, …, 6 và rút gọn ma trận chi phí được:

$$M' = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{array} \begin{array}{c} \begin{array}{cccccc} 1 & 2 & 3 & 4 & 5 & 6 \end{array} \\ \begin{pmatrix} \infty & 0 & 2 & 2 & 7 & 10 \\ 0 & \infty & 0 & 0 & 5 & 9 \\ 12 & 9 & \infty & 4 & 4 & 0 \\ 0 & 10 & 17 & \infty & 4 & 1 \\ 0 & 2 & 4 & 4 & \infty & 8 \\ 10 & 6 & 4 & 0 & 0 & \infty \end{pmatrix} \end{array}$$

Hằng số rút gọn ma trận M là: $S_M = 5 + 8 + 3 + 8 + 10 + 10 + 2 = 46$.

Vậy, cận dưới của hàm mục tiêu của mọi hành trình là 46 ($f_{min} = f_{\overline{min}} \geq 46$).

Có 9 phần tử bằng 0, đó là c'_{12}, c'_{21}, c'_{23}, c'_{24}, c'_{36}, c'_{41}, c'_{51}, c'_{64} và c'_{65}. Ta có:

$\sigma_{12} = 2+2 = 4$, $\sigma_{21} = 0+0 = 0$, $\sigma_{23} = 2$, $\sigma_{24} = 0$, $\sigma_{36} = 5$, $\sigma_{41} = 1$, $\sigma_{51} = 2$, $\sigma_{64} = 2$, $\sigma_{65} = 4$

Vậy cung (3, 6) được chọn để phân nhánh. Đặt $c_{36} = \infty$ được ma trận $M_{\overline{36}}$. Đặt $c_{63} = \infty$, xóa hàng 3 và cột 6 được ma trận M_{36}:

$$M_{\overline{36}} = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{array} \begin{array}{c} \begin{array}{cccccc} 1 & 2 & 3 & 4 & 5 & 6 \end{array} \\ \begin{pmatrix} \infty & 0 & 2 & 2 & 7 & 10 \\ 0 & \infty & 0 & 0 & 5 & 9 \\ 12 & 9 & \infty & 4 & 4 & \infty \\ 0 & 10 & 17 & \infty & 4 & 1 \\ 0 & 2 & 4 & 4 & \infty & 8 \\ 10 & 6 & 4 & 0 & 0 & \infty \end{pmatrix} \end{array}; \quad M_{36} = \begin{array}{c} \\ 1 \\ 2 \\ 4 \\ 5 \\ 6 \end{array} \begin{array}{c} \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{pmatrix} \infty & 0 & 2 & 2 & 7 \\ 0 & \infty & 0 & 0 & 5 \\ 0 & 10 & 17 & \infty & 4 \\ 0 & 2 & 4 & 4 & \infty \\ 10 & 6 & \infty & 0 & 0 \end{pmatrix} \end{array}$$

Hằng số rút gọn ma trận $M_{\overline{36}}$ là: $4 + 1 = 5$. Vậy $f_{\overline{\min}}(3,6) \geq 46 + 5 = 51$.

Ma trận M_{36} không phải rút gọn vì nó đã là ma trận rút gọn. Vậy f_{min} không tăng, nghĩa là $f_{min}(3,6) \geq 46$.

$$M'_{36} = M_{36} = \begin{array}{c} \\ 1 \\ 2 \\ 4 \\ 5 \\ 6 \end{array} \begin{array}{c} \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{pmatrix} \infty & 0 & 2 & 2 & 7 \\ 0 & \infty & 0 & 0 & 5 \\ 0 & 10 & 17 & \infty & 4 \\ 0 & 2 & 4 & 4 & \infty \\ 10 & 6 & \infty & 0 & 0 \end{pmatrix} \end{array}$$

Có 8 phần tử bằng 0 trong ma trận M'_{36} là c_{12}, c_{21}, c_{23}, c_{24}, c_{41}, c_{51}, c_{64} và c_{65}. Ta có:

$$\sigma_{12} = 4, \sigma_{21} = 0, \sigma_{23} = 2, \sigma_{24} = 0, \sigma_{41} = 4, \sigma_{51} = 2, \sigma_{64} = 2, \sigma_{65} = 4.$$

Có hai giá trị σ_{41} và σ_{65} cùng lớn nhất trong các σ vừa tính nên có thể chọn một trong 2 cung (4, 1) hoặc (6,5) để rẽ nhánh. Chọn cung (6, 5). Thay c_{65} trong M'_{36} bằng ∞ được ma trận $M_{\overline{65}}$. Cũng trong M'_{36} đặt $c_{56} = \infty$, xóa hàng 6 và cột 5 được ma trận M_{65}:

$$M_{\overline{65}} = \begin{array}{c} \\ 1 \\ 2 \\ 4 \\ 5 \\ 6 \end{array} \begin{array}{c} \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{pmatrix} \infty & 0 & 2 & 2 & 7 \\ 0 & \infty & 0 & 0 & 5 \\ 0 & 10 & 17 & \infty & 4 \\ 0 & 2 & 4 & 4 & \infty \\ 10 & 6 & \infty & 0 & \infty \end{pmatrix} \end{array}; \quad M_{65} = \begin{array}{c} \\ 1 \\ 2 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{pmatrix} \infty & 0 & 2 & 2 \\ 0 & \infty & 0 & 0 \\ 0 & 10 & 17 & \infty \\ 0 & 2 & 4 & 4 \end{pmatrix} \end{array}$$

Hằng số rút gọn của $M_{\overline{65}}$ là 4 (số nhỏ nhất cột 5). Vậy $f_{\overline{\min}}(6,5) \geq 51 + 4 = 55$

Ma trận M_{65} không phải rút gọn do vậy $M'_{65} = M_{65}$ và f_{min} là không đổi, nghĩa là $f_{min}(6,5) \geq 46$.

Có 6 phần tử bằng 0 trong ma trận M'_{65} là c_{12}, c_{21}, c_{23}, c_{24}, c_{41} và c'_{51}. Ta có:

$$\sigma_{12} = 4,\ \sigma_{21} = 0,\ \sigma_{23} = 2,\ \sigma_{24} = 2,\ \sigma_{41} = 10 \text{ và } \sigma_{51} = 2$$

Vậy cung (4, 1) được chọn để rẽ nhánh. Thay c_{41} trong M_{65} bằng ∞ được ma trận $M_{\overline{41}}$. Cũng trong ma trận M_{65} đặt $c'_{14} = \infty$, xóa hàng 4 và cột 1, do các cung (3, 6) và (6,5) đã được chọn nên không thể chọn cung (5, 3) (nếu chọn cung (5, 3) sẽ tạo thành chu trình con), vậy phải đặt $c_{53} = \infty$ được ma trận M_{41}:

$$M_{\overline{41}} = \begin{array}{c} \\ 1 \\ 2 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{pmatrix} \infty & 0 & 2 & 2 \\ 0 & \infty & 0 & 0 \\ \infty & 10 & 17 & \infty \\ 0 & 2 & 4 & 4 \end{pmatrix} \end{array}; \qquad M_{41} = \begin{array}{c} \\ 1 \\ 2 \\ 5 \end{array} \begin{array}{c} \begin{array}{ccc} 2 & 3 & 4 \end{array} \\ \begin{pmatrix} 0 & 2 & \infty \\ \infty & 0 & 0 \\ 2 & \infty & 4 \end{pmatrix} \end{array}$$

Hằng số rút gọn của M_{41} là 2. Vậy $f_{min}(4,1) \geq 46 + 2 = 48$.

Hằng số rút gọn của $M_{\overline{41}}$ là 10. Vậy $f_{\overline{min}}(4,1) \geq 55 + 10 = 65$

Rút gọn ma trận M_{41} được:

$$M'_{41} = \begin{array}{c} \\ 1 \\ 2 \\ 5 \end{array} \begin{array}{c} \begin{array}{ccc} 2 & 3 & 4 \end{array} \\ \begin{pmatrix} 0 & 2 & \infty \\ \infty & 0 & 0 \\ 0 & \infty & 2 \end{pmatrix} \end{array}$$

Có 4 phần tử bằng 0 trong M'_{41} là c_{12}, c_{23}, c_{24} và c_{52}. Ta có:

$$\sigma_{12} = 2,\ \sigma_{23} = 2,\ \sigma_{24} = 2 \text{ và } \sigma_{52} = 2$$

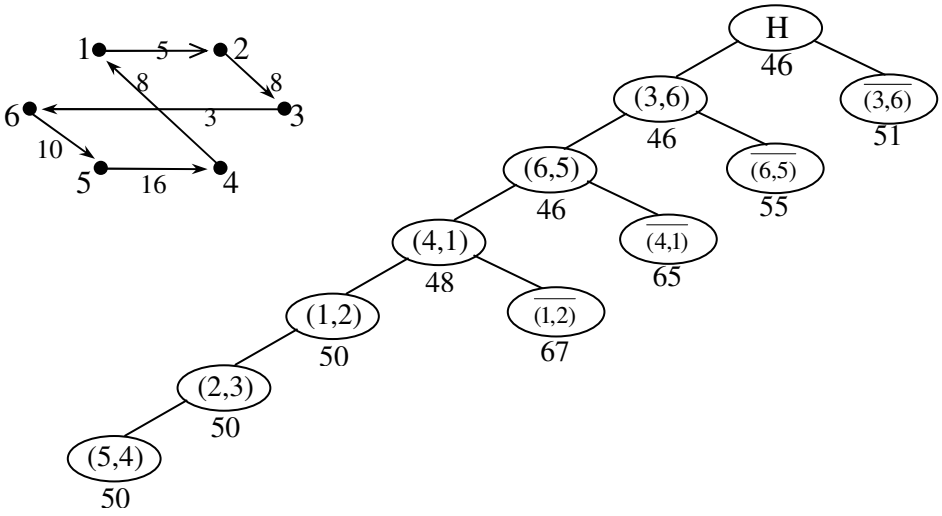
Vậy cả 4 cung (1,2), (2,3), (2,4), (5,2) đều có thể được chọn để rẽ nhánh. Chọn (1,2). Trong M'_{41}, đặt $c_{12} = \infty$ được $M_{\overline{12}}$. Loại hàng 1 và cột 2 ra khỏi M'_{41} được M_{12}:

$$M_{\overline{12}} = \begin{array}{c} \\ 1 \\ 2 \\ 5 \end{array} \begin{array}{c} \begin{array}{ccc} 2 & 3 & 4 \end{array} \\ \begin{pmatrix} \infty & 2 & \infty \\ \infty & 0 & 0 \\ 0 & \infty & 2 \end{pmatrix} \end{array}; \qquad M_{12} = \begin{array}{c} \\ 2 \\ 5 \end{array} \begin{array}{c} \begin{array}{cc} 3 & 4 \end{array} \\ \begin{pmatrix} 0 & 0 \\ \infty & 2 \end{pmatrix} \end{array}$$

Hằng số rút gọn của $M_{\overline{12}}$ là 2. Vậy $f_{\overline{min}}(1,2) \geq 65 + 2 = 67$

M_{12} là ma trận vuông cấp 2 nên chọn trực tiếp hai cung cuối cùng của hành trình ứng với 2 phần tử có giá trị nhỏ nhất của M_{12}, đó là 2 cung (2,3) và (5,4). Chi phí cực tiểu là $f_{min} = 48 + 0 + 2 = 50$.

Vậy các cung được chọn vào hành trình là: (3.6), (6,5), (4,1), (1,2), (2,3) và (5,4). Sắp xếp lại ta có hành trình đi qua các thành phố 3,6,5,4,1,2,3 với chi phí cực tiểu là 50. Hành trình tối ưu và cây tìm kiếm hành trình tối ưu được vẽ trong hình 16.

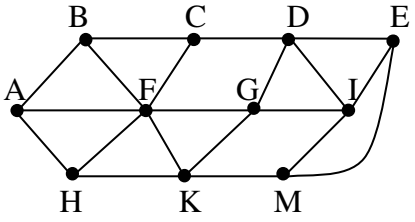


Hình 16. Hành trình tối ưu và cây phân

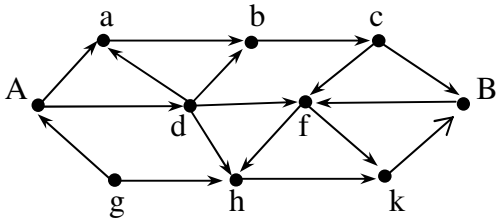
BÀI TẬP CHƯƠNG 6

6.1. Tìm đường đi ngắn nhất từ đỉnh A đến các đỉnh còn lại trong các đồ thị sau:

a)

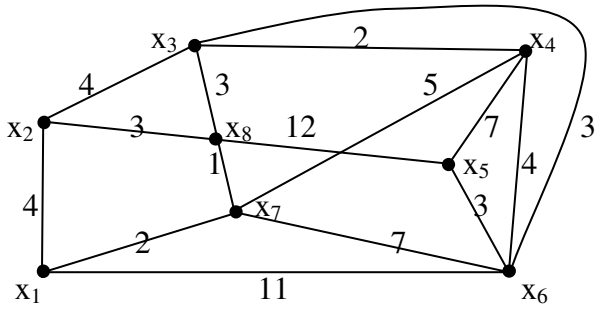


b)

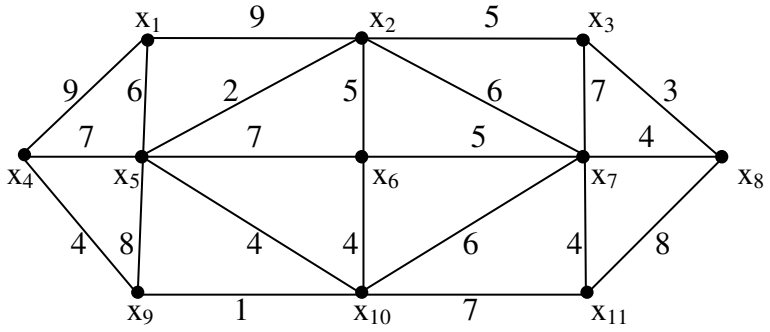


6.2. Dùng thuật toán Dijkstra tìm đường đi ngắn nhất từ đỉnh x_1 tới các đỉnh còn lại trong các đồ thị sau:

a)

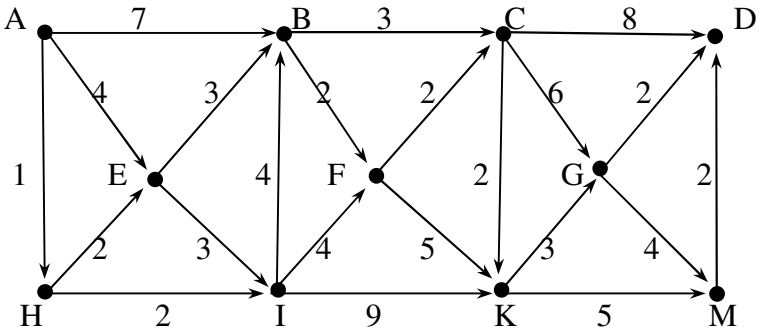


b)

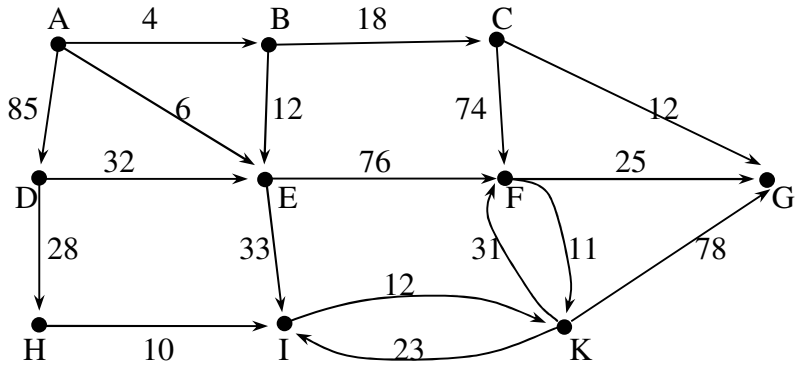


6.3. Tìm đường đi ngắn nhất từ đỉnh A tới các đỉnh còn lại trong đồ thị sau:

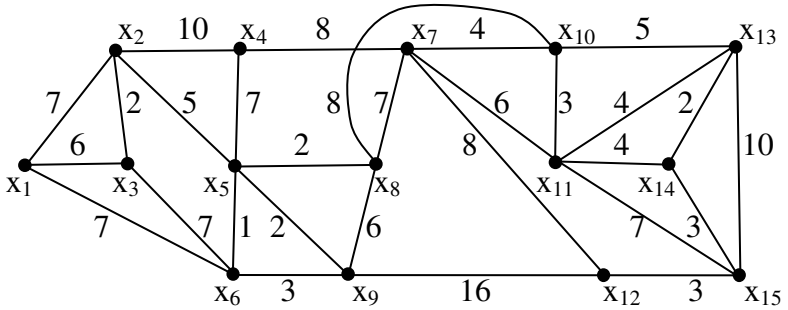
a)



b)



6.4. Cho đồ thị



a) Tìm đường đi ngắn nhất từ x_1 đến x_{14}.

a) Tìm đường đi ngắn nhất từ x_1 đến x_{14} có chứa cạnh x_8x_9.

a) Tìm đường đi ngắn nhất từ x_1 đến x_{14} có chứa đỉnh x_7.

6.5. Tìm đường đi ngắn nhất từ x_2 đến các đỉnh còn lại trong các đồ thị sau:

a) (Đồ thị vô hướng)

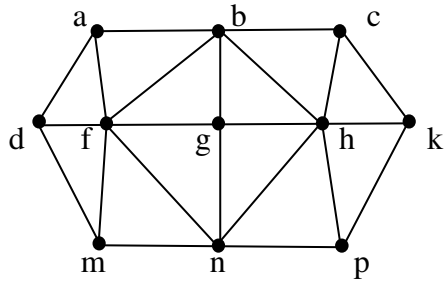
	x_1	x_2	x_3	x_4	x_5	x_6	x_7
x_1	.	3	6	.	.	.	.
x_2	3	.	2	4	.	.	.
x_3	6	2	.	1	4	2	.
x_4	.	4	1	.	2	.	4
x_5	.	.	4	2	.	2	1
x_6	.	.	2	.	2	.	4
x_7	.	.	.	4	1	4	.

b) (Đồ thị có hướng)

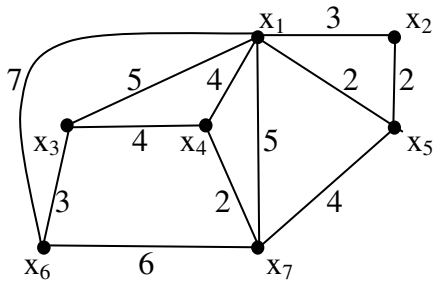
	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8
x_1	.	.	.	.	.	.	12	13
x_2	31	.	.	.	4	.	.	.
x_3	.	.	.	.	.	.	4	1
x_4	.	.	3	.	.	.	8	.
x_5	.	.	.	.	.	3	.	.
x_6	2	.	.	8	.	.	5	.
x_7	.	.	.	.	.	.	.	6
x_8	2	.	.	.	.	.	.	.

6.6. Tìm tâm, bán kính, đường kính của các đồ thị:

a)

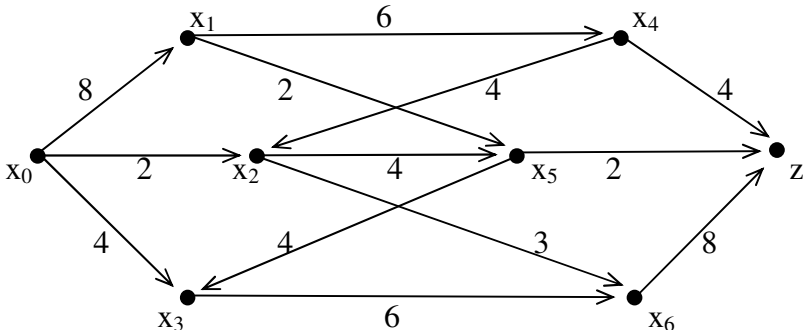


b)

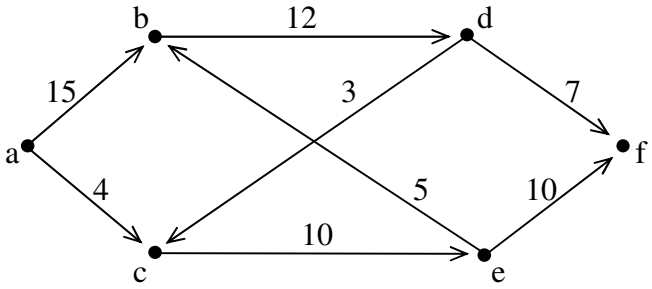


6.7. Bằng thuật toán Ford-Fulkerson hãy tìm luồng cực đại và lát cắt hẹp nhất của các mạng sau, biết luồng ban đầu bằng 0 và khả năng thông qua được ghi trên các cung:

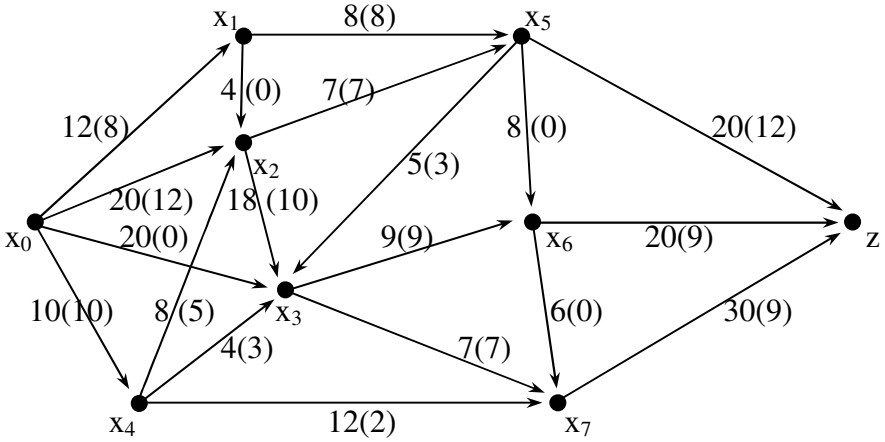
a)



b)

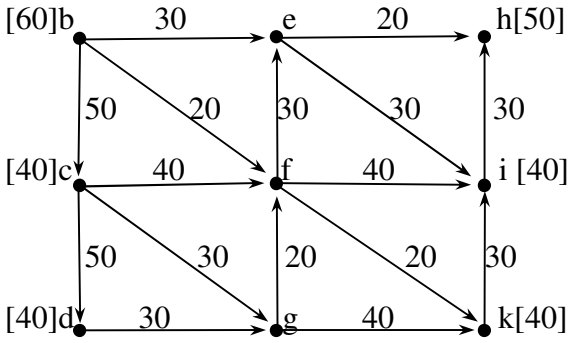


6.8. Bằng thuật toán Ford-Fulkerson hãy tìm luồng cực đại và lát cắt hẹp nhất của mạng sau, biết khả năng thông qua được ghi trên cung và luồng ban đầu được ghi trong dấu ngoặc đơn:



6.9. *Mạng với nhiều điểm thu, phát.*

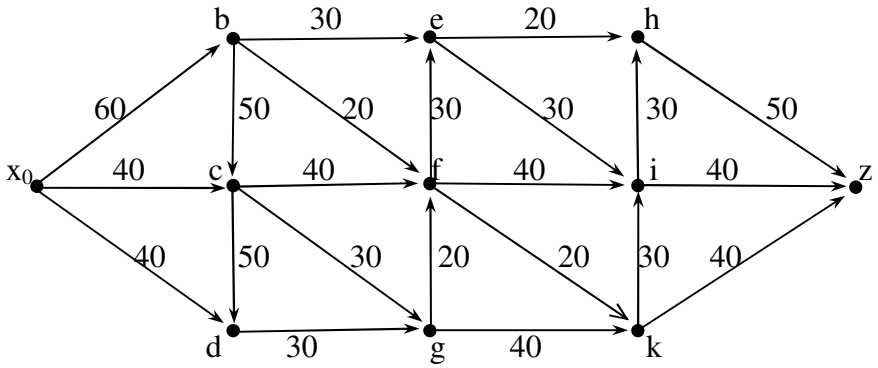
Xét mạng G sau:



Mạng G

Mạng G có ba đỉnh b, c, d với khả năng cung cấp lần lượt là 60, 40, 40 và có ba đỉnh thu h, i, k với khả năng tiếp nhận tương ứng là 50, 40,40.

Để tìm luồng vận tải cực đại trong mạng, ta bổ sung vào G một đỉnh phát a và một đỉnh thu z để được mạng G' quen thuộc như hình sau:



Mạng G'

Bằng thuật toán Ford-Fulkerson hãy tìm luồng cực đại trên mạng G' (cũng là luồng cực đại trên G) với luồng ban đầu bằng 0.

6.10. Giải các bài toán du lịch với các ma trận chi phí sau:

a)

	1	2	3	4
1	.	3	9	7
2	3	.	6	5
3	5	6	.	6
4	9	7	4	.

b)

	A	B	C	D
A	.	9	12	6
B	4	.	8	8
C	6	15	.	10
D	5	8	11	.

c)

	A	B	C	D	E
A	.	12	9	8	8
B	14	.	.	10	15
C	7	12	.	9	.
D	20	9	15	.	8
E	16	15	14	13	.

d)

	1	2	3	4	5	6
1	.	3	4	2	6	3
2	5	.	3	4	4	5
3	4	1	.	5	3	4
4	4	2	6	.	4	5
5	3	3	3	5	.	4
6	7	4	5	6	7	.

ĐÁP SỐ

6.2. a) $l(x_1x_2) = 4$; $l(x_1x_7x_8x_3) = 6$; $l(x_1x_7x_4) = 7$; $l(x_1x_7x_6x_5) = 12$; $l(x_1x_7x_8x_3x_6) = 9$ hoặc $(x_1x_7x_6) = 9$; $l(x_1x_7) = 2$; $l(x_1x_7x_8) = 3$.

b) $l(x_1x_5x_2) = 8$; $l(x_1x_5x_2x_3) = 13$; $l(x_1x_4) = 9$; $l(x_1x_5) = 6$; $l(x_1x_5x_6) = 13$; $l(x_1x_5x_2x_7) = 14$; $l(x_1x_5x_2x_3x_8) = 16$; $l(x_1x_5x_{10}x_9) = 11$; $l(x_1x_5x_{10}) = 10$; $l(x_1x_5x_{10}x_{11}) = 17$.

6.3. a) l(AHEB) = 6; l(AHIFC) = 9; l(AHIFCKGD) = 16; l(AHE) = 3;
l(AHIF) = 7; l(AHIFCKG) = 14; l(AH) = 1; l(AHI) = 3;
l(AHIFCK) = 11; l(AHIFCKM) = 16.

b) l(AB) = 4; l(ABC) = 22; L(AD) = 85; l(AE) = 6; l(AEF) =82;
l(ABCG) = 34; l(ADH) = 113; l(AEI) = 39; l(AEIK) = 51.

6.4. a) $x_1x_6x_5x_8x_{10}x_{11}x_{14}$, hoặc $x_1x_6x_5x_8x_{10}x_{13}x_{14}$: l = 25;

b) $x_1x_6x_5x_9x_8x_{10}x_{11}x_{14}$, hoặc $x_1x_6x_5x_9x_8x_{10}x_{13}x_{14}$,
hoặc $x_1x_6x_9x_8x_{10}x_{11}x_{14}$, hoặc $x_1x_6x_9x_8x_{10}x_{13}x_{14}$: l = 31;

c) $x_1x_6x_5x_8x_7x_{10}x_{11}x_{14}$: l = 27.

6.5. a) $l(x_2x_1) = 3$; $l(x_2x_3) = 2$; $l(x_2x_3x_4) = 3$; $l(x_2x_3x_4x_5) = 5$;
$l(x_2x_3x_6) = 4$; $l(x_2x_3x_4x_5x_7) = 6$.

b) $l(x_2x_5x_6x_1) = 9$; $l(x_2x_5x_6x_4x_3) = 18$; $l(x_2x_5x_6x_4) = 15$; $l(x_2x_5) = 4$;
$l(x_2x_5x_6) = 7$; $l(x_2x_5x_6x_7) = 12$; $l(x_2x_5x_6x_7x_8) = 18$.

6.6. a) r = 2; b, g, n là các tâm

b) r = 6; x_7 là tâm; d = 22.

6.7. a) $val(\varphi_{max}) = 14$; Lát cắt hẹp nhất với $A = \{x_0\}$.

b) $val(\varphi_{max}) = 14$; Lát cắt hẹp nhất với A = {a, b, d}.

6.8. $val(\varphi_{max}) = 41$; Lát cắt hẹp nhất với $A = \{x_0, x_1, x_2, x_3\}$.

6.9. $val(\varphi_{max}) = 130$; Lát cắt hẹp nhất với $A = \{x_0, d\}$.

6.10. a) 1, 2, 4, 3, 1. Chi phí cực tiểu là 17.

b) A, D, B, C, A. Chi phí cực tiểu là 28.

c) A, B, D, E, C, A. Chi phí cực tiểu là 51.

d) Có 5 hành trình: 1,4,6,2,3,5,1; 1,4,6,3,2,5,1; 1,4,2,6,3,5,1;
1,6,3,2,4,5,1; 1,6,4,2,3,5,1.
Chi phí cực tiểu là 20. (Chỉ cần tìm một trong năm hành trình).

CÂU HỎI ÔN TẬP CHƯƠNG 6

1. Định nghĩa đường đi ngắn nhất trong đơn đồ thị không trọng số và trong đơn đồ thị có trong số dương. Trình bày thuật toán tìm đường đi ngắn nhất trong đồ thị không trọng số và thuật toán Dijstra để tìm đwong đi ngắn nhất trong đồ thị có trọng số dương?

2. Phát biểu các định nghĩa về: Khoảng cách giữa các đỉnh, Tâm, Bán kính, Đường kính của một đơn đồ thị vô hướng.

3. Phát biểu định nghĩa và các tính chất của Mạng và Luồng. Trình bày thuật toán Ford-Fulkerson tìm luồng cực đại.

4. Phát biểu bài toán du lịch. Trình bày thuật toán nhánh, cận để giải bài toán du lịch.

CHƯƠNG 7

ĐẠI SỐ BOOLE

1. Hàm Boole

1.1. Mở đầu

Mỗi mạch điện tử hoặc mạch quang học có thể làm việc theo các quy tắc nhất định và chúng chỉ có 2 trạng thái: đóng và mở hoặc sáng và tối. Có thể gán các trạng thái đó với các số 1 và 0. Đại số Boole ra đời nhằm mô hình hoá hoạt động của các mạch điện tử và mạch quang học.

Đại số Boole đưa ra các phép toán và các hàm trên tập {0, 1}. Biểu thức Boole là sự tổ hợp các phép toán của đại số Boole. Các mạch điện tử và quang học có thể được nghiên cứu bằng các phép biến đổi hàm Boole.

Ba phép toán cơ bản của đại số Boole là:

- **Phần bù** (còn gọi là phủ định) của một phần tử, ký hiệu bằng gạch ngang trên đầu phần tử đó và được định nghĩa: $\overline{1} = 0;\ \overline{0} = 1$.

- **Tổng Boole** (còn gọi là tuyển) của 2 phần tử, ký hiệu + (hoặc OR, hoặc $\vee$), được định nghĩa như sau: $1 + 1 = 1$; $1 + 0 = 1$; $0 + 1 = 1$; $0 + 0 = 0$.
- **Tích Boole** (còn gọi là hội) của 2 phần tử, ký hiệu bằng dấu chấm (.) (hoặc AND, hoặc $\wedge$), được định nghĩa như sau: $1 . 1 = 1$; $1. 0 = 0$; $0 . 1 = 0$; $0 . 0 = 0$

Thường dùng các dấu ngoặc để chỉ thứ tự thực hiện các phép tính. Trong trường hợp không có dấu ngoặc thì thứ tự thực hiện các phép tính là: Phần bù, tích, sau cùng là tổng.

Thí dụ: $1.0 + \overline{(1+0)} = 0 + \overline{1} = 0 + 0 = 0$.

Phép lấy phần bù, lấy tổng và lấy tích Boole tương ứng với các toán tử logic: phủ định ($\bar{}$), tuyển ($\vee$) và hội ($\wedge$) trong đó 0 ứng với F và 1 ứng với T. Các kết quả của đại số Boole có thể áp dụng trực tiếp cho đại số các mệnh đề. Ngược lại các kết quả về đại số mệnh đề có thể áp dụng trực tiếp cho đại số Boole. Và, do vậy đại số Boole còn được gọi là đại số Logic.

1.2. Hàm Boole

Cho tập hợp B = {0;1}. Biến x được gọi là biến Boole nếu nó nhận giá trị 0, 1 trong B.

Định nghĩa: Hàm Boole n biến là một ánh xạ f: $B^n \to B$, trong đó B^n là tích Đề các n lần của B ($B^n = \{(x_1, x_2, ..., x_n) \mid x_i \in B, i = 1, 2, ..., n\}$).

Hàm Boole n biến được viết là $f = f(x_1, x_2, ..., x_n)$

Các giá trị của hàm Boole có thể cho dưới dạng bảng.

Thí dụ: Hàm Boole 2 biến f(x,y) nhận giá trị 1 khi x =1 và y = 0; nhận giá trị 0 trong các trường hợp còn lại được cho trong bảng sau:

x	y	f(x,y)
0	0	0
0	1	0
1	0	1
1	1	0

Hai hàm Boole $f = f(x_1,x_2, ...,x_n)$ và $g = g(x_1,x_2, ...,x_n)$ được gọi là bằng nhau, ký hiệu f = g, nếu chúng nhận giá trị giống nhau ứng với mỗi bộ giá trị của bộ biến $x_1,x_2, ...,x_n$.

Cho 2 hàm Boole $f = f(x_1,x_2, ...,x_n)$ và $g = g(x_1,x_2, ...,x_n)$, người ta định nghĩa:

- Phần bù (hay phủ định) của hàm Boole f, ký hiệu $\bar{f}$, được định nghĩa:

$$\bar{f}(x_1, x_2,..., x_n) = \overline{f(x_1, x_2,..., x_n)}$$

- Tổng (hay tuyển) của 2 hàm f và g, ký hiệu f + g (hay f$\vee$g), được định nghĩa:

$$(f+g)(x_1,x_2, ...,x_n) = f(x_1,x_2, ...,x_n) + g(x_1,x_2, ...,x_n) .$$

- Tích (hay hội) của 2 hàm f và g, ký hiệu f.g (hay f$\wedge$g), được định nghĩa:

$$(f.g)(x_1,x_2, ...,x_n) = f(x_1,x_2, ...,x_n) . g(x_1,x_2, ...,x_n)$$

Có 4 hàm Boole 1 biến :

x	f_1	f_2	f_3	f_4
1	0	0	1	1
0	0	1	0	1

f_1 là hàm hằng 0; f_4 là hàm hằng 1; f_3 là hàm lặp lại giá trị của x; f_4 là hàm phủ định của x; dễ thấy $f_1 = \overline{f_4}$; $f_2 = \overline{f_3}$.

Có 16 hàm Boole 2 biến:

x	y	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}	f_{11}	f_{12}	f_{13}	f_{14}	f_{15}	f_{16}
0	0	0	1	0	0	0	1	1	1	1	1	0	0	1	0	1	0
0	1	0	1	1	0	1	1	0	0	1	0	0	1	0	0	1	1
1	0	0	1	1	0	1	0	0	0	1	1	1	0	1	1	0	0
1	1	0	1	1	1	0	1	1	0	0	1	1	1	0	0	0	0

- f_1 được gọi là hàm hằng 0, ký hiệu là f(x,y) = 0
- f_2 được gọi là hàm hằng 1, ký hiệu là f(x,y) = 1
- f_3 được gọi là hàm tuyển của x và y, ký hiệu là f(x,y) = x + y hay f(x,y) = x∨y
- f_4 được gọi là hàm hội của x và y, ký hiệu là f(x,y) = xy hay f(x,y) = x∧y
- f_5 được gọi là hàm tuyển loại của x và y, ký hiệu là f(x,y) = x ⊕ y
- f_6 được gọi là hàm kéo theo, ký hiệu là f(x,y) = x → y
- f_7 được gọi là hàm tương đương, ký hiệu là f(x,y) = x ⇔ y
- f_8 được gọi là hàm Vebb của x và y, ký hiệu là f(x,y) = x ↓ y
- f_9 được gọi là hàm Sheffer của x và y, ký hiệu là f(x,y) = x↑y

Tổng quát có 2^{2^n} hàm Boole n biến. Thật vậy: Với n biến Boole $x_1, x_2, \ldots, x_n$ và mỗi biến nhận 2 giá trị 0 và 1 nên có 2^n bộ giá trị của n biến đó. Với mỗi bộ giá trị của n biến hàm Boole n biến nhận 2 giá trị do đó phải có 2^{2^n} hàm Boole n biến.

Chú ý rằng hàm Boole còn được gọi là hàm đại số Logic.

2. Biểu thức Boole

2.1. Định nghĩa

Biểu thức Boole đối với các biến $x_1, x_2, ..., x_n$ được định nghĩa đệ quy như sau:

- 0, 1, $x_1, x_2, ..., x_n$ là các biểu thức Boole.
- Nếu A_1 và A_2 là các biểu thức Boole thì $\overline{A_1}$, $A_1.A_2$ và A_1+A_2 cũng là các biểu thức Boole.

2.2. Các hằng đẳng thức của đại số Boole (các tính chất của các phép toán Boole)

Với mọi x, y, z là các biến Boole luôn luôn có:

- Luật phần bù kép: $\overline{\overline{x}} = x$
- Luật luỹ đẳng: $x + x = x$
  $x.x = x$

- Luật đồng nhất: $x + 0 = x$
  $x.1 = x$
- Luật nuốt: $x + 1 = 1$
  $x . 0 = 0$
- Luật giao hoán: $x + y = y + x$
  $x . y = y . x$
- Luật kết hợp: $x + (y + z) = (x + y) + z$
  $x. (yz) = (xy)z$
- Luật phân phối: $x + yz = (x + y)(x + z)$
  $x(y + z) = xy + xz$
- Luật Đơ-Moocgan: $\overline{(x + y)} = \overline{x}.\overline{y}$
  $\overline{(x.y)} = \overline{x} + \overline{y}$

Có thể chứng minh các tính chất trên bằng cách lập bảng giá trị. Chẳng hạn, chứng minh luật phân phối $x + yz = (x + y)(x + z)$:

x	y	z	yz	**x + yz**	x + y	x + z	**(x+y)(x+z)**
0	0	0	0	**0**	0	0	**0**
0	0	1	0	**0**	0	1	**0**
0	1	0	0	**0**	1	0	**0**
0	1	1	1	**1**	1	1	**1**
1	0	0	0	**1**	1	1	**1**
1	0	1	0	**1**	1	1	**1**
1	1	0	0	**1**	1	1	**1**
1	1	1	1	**1**	1	1	**1**

Nhìn vào cột x + yz và (x+y)(y+z) ta thấy chúng cùng giá trị với cùng bộ giá trị của x, y, z. Vậy x + yz = (x+y)(x+z).

Các hằng đẳng thức trên có thể dùng để chứng minh các hằng đẳng thức khác.

Thí dụ: Chứng minh **luật hút thu:** $x(x + y) = x$

Thật vậy: $x(x + y) = (x + 0)(x + y)$ (Luật đồng nhất)
$= x + 0.y$ (Luật phân phối)
$= x + y. 0$ (Luật giao hoán)
$= x + 0$ (Luật nuốt)
$= x$ (Luật đồng nhất)

2.3. Tính đối ngẫu

Các tính chất nêu trong 2.2 đều xuất hiện từng cặp, trừ tính chất phần bù kép. Hơn nữa, trong mỗi cặp đó, nếu biết được một đẳng thức thì có thể suy ra đẳng thức còn lại bằng cách thay tích Boole bằng tổng Boole và ngược lại, thay 0 bằng 1 và thay 1 bằng 0.

Việc thay đổi các phép toán Boole và các hằng Boole như vậy gọi là lấy đối ngẫu một đẳng thức Boole.

Định nghĩa: Đối ngẫu của một biểu thức Boole là một biểu thức Boole có được từ biểu thức Boole đã cho bằng cách thay tích Boole bằng tổng Boole, thay tổng Boole bằng tích Boole, thay 1 bằng 0 và thay 0 bằng 1.

Thí dụ 1: 1-/ Đối ngẫu của $x.\bar{y} + z$ là $(x + \bar{y}).z$

2-/ Đối ngẫu của $x.1 + (y + \bar{z})$ là $(x + 0).(y.\bar{z})$

Nguyên lý đối ngẫu: Một hằng đẳng thức Boole vẫn đúng nếu lấy đối ngẫu cả hai vế của nó.

Thí dụ 2: Lấy đối ngẫu 2 vế của luật hút thu $x(x + y) = x$ được $x + xy = x$, hằng đẳng thức mới này cũng được gọi là luật hút thu.

2.4. Quy tắc thay thế

Chúng ta thừa nhận quy tắc thay thế sau: Một hằng đẳng thức Boole vẫn đúng nếu thay một biến của đẳng thức bằng một biểu thức Boole nào đó.

Thí dụ: Ta đã biết $\overline{(x+y)} = \bar{x}.\bar{y}$ (luật Đơ-Moocgan). Nếu thay y bằng y+z được:

$$\overline{(x+(y+z)} = \overline{x+y+z} = \bar{x}.\overline{y+z} = \bar{x}.\bar{y}.\bar{z}$$

3. Định nghĩa đại số Boole theo tiên đề

Một cách tổng quát có thể định nghĩa đại số Boole theo tiên đề hay còn gọi là định nghĩa trừu tượng đại số Boole như sau.

Định nghĩa: Cho một tập hợp E không rỗng. Xác định trên E hai phép toán hai ngôi gọi là phép cộng (ký hiệu +) và phép nhân (ký hiệu .), cùng phép toán một ngôi gọi là phép bù (ký hiệu $\bar{}$). Tập E cùng ba phép toán trên được gọi là một đại số Boole nếu các tiên đề sau được thỏa mãn với mọi $x, y, z \in E$:

- Luật giao hoán: $x + y = y + x$
  $x . y = y . x$
- Luật kết hợp: $(x + y) + z = x + (y + z)$
  $(x . y) . z = x . (y . z)$
- Luật phân phối: $(x + y) . z = (x . z) + (y . z)$
  $x + (y . z) = (x + y) . (x + z)$
- Tồn tại phần tử trung hòa: Phần tử trung hòa của phép cộng được ký hiệu là 0 và phần tử trung hòa của phép nhân được ký hiệu là 1 sao cho:
  $x + 0 = 0 + x = x$
  $x . 1 = 1 . x = x$
- Tồn tại phần tử bù: Với mỗi $x \in E$ tồn tại duy nhất một phần tử $\bar{x} \in E$ sao cho:

$$x + \bar{x} = \bar{x} + x = 1$$
$$x.\bar{x} = \bar{x}.x = 0$$

$\bar{x}$ được gọi là phần tử bù của phần tử x.

Qua các phần đã trình bày, thấy tập B = {0, 1} cùng 3 phép toán cộng (+), nhân(.) và phần bù ($\bar{}$) được định nghĩa trong 1.1 thoả mãn các tiên đề đã nêu trong định nghĩa theo tiên đề trên.

Ngược lại, từ các tính chất đã nêu trong định nghĩa trên có thể chứng minh được các tính chất còn lại đã nêu trong mục 2.2.

4. Biểu diễn các hàm Boole

Trong phần trước hàm Boole được cho dưới dạng bảng, điều đó gây nhiều bất tiện. Phần này trình bày tìm cách biểu diễn các hàm Boole dưới dạng biểu thức được tạo bởi các biến, các hằng và các phép toán Boole.

4.1. Biểu diễn hàm Boole bằng biểu thức Boole

Mỗi hàm Boole được biểu diễn bằng một biểu thức Boole. Các giá trị của hàm Boole nhận được bằng cách gán 0 hoặc 1 cho các biến có trong biểu thức đó.

Thí dụ: Các giá trị của hàm Boole $f(x,y,z) = xy + \bar{z}$ được cho trong bảng dưới đây:

x	y	z	xy	$\bar{z}$	$f(xyz) = xy + \bar{z}$
0	0	0	0	1	1
0	0	1	0	0	0
0	1	0	0	1	1
0	1	1	0	0	0
1	0	0	0	1	1
1	0	1	0	0	0
1	1	0	1	1	1
1	1	1	1	0	1

Hai biểu thức Boole cùng biểu diễn một hàm Boole được gọi là 2 biểu thức tương đương nhau.

4.2. Dạng tuyển chuẩn tắc của hàm Boole

a. Hội sơ cấp và tuyển sơ cấp

Có hai loại biểu thức Boole thường gặp để biểu diễn các hàm Boole, đó là các tuyển sơ cấp và hội sơ cấp, chúng được định nghĩa như sau:

Cho x là một biến Boole và $\sigma \in \{0, 1\}$, ký hiệu:

$$x^{\sigma} = \begin{cases} x, \text{nếu } \sigma = 1 \\ \bar{x}, \text{nếu } \sigma = 0 \end{cases}$$

Để tiện lợi, ta gọi x^σ là tục biến của biến x. Dễ thấy:

$$x^\sigma = 1 \quad \Leftrightarrow \quad x = \sigma$$

Định nghĩa: Giả sử $x_1, x_2, \ldots, x_n$ là các biến Boole.

- Biểu thức $x_{i_1}^{\sigma_1} x_{i_2}^{\sigma_2} \ldots x_{i_k}^{\sigma_k}$, trong đó $1 \le i_1 < i_2 < \ldots < i_k \le n$ và $\sigma_1,\sigma_2, \ldots,\sigma_k \in \{0, 1\}$ được gọi là một hội sơ cấp hạng k của n biến $x_1,x_2,\ldots,x_n$.
- Biểu thức $x_{i_1}^{\sigma_1} + x_{i_2}^{\sigma_2} + \ldots + x_{i_k}^{\sigma_k}$, trong đó $1 \le i_1 < i_2 < \ldots < i_k \le n$ và $\sigma_1,\sigma_2, \ldots,\sigma_k \in \{0, 1\}$ được gọi là một tuyển sơ cấp hạng k của n biến $x_1,x_2,\ldots,x_n$.

Nói cách khác, một tích gồm k tục biến của n biến Boole đã cho được gọi là một hội sơ cấp hạng k của n biến đó. Và, một tổng gồm k tục biến của n biến Boole đã cho được gọi là một tuyển sơ cấp hạng k của n biến đó.

Chú thích: Hội sơ cấp còn gọi là minterm. Tuyển sơ cấp còn gọi là maxterm

Dễ thấy một hội sơ cấp có giá trị bằng 1 khi và chỉ khi mọi $x_{i_j}^{\sigma_j} = 1$, nghĩa là khi và chỉ khi mọi $x_{i_j} = \sigma_j$. Và, một tuyển sơ cấp có giá trị bằng 0 khi và chỉ khi mọi $x_{i_j}^{\sigma_j} = 0$.

Thí dụ: Hội sơ cấp $\bar{x}_1 x_2 x_3 \bar{x}_4 x_5$ có giá trị bằng 1 nếu $x_1 = x_4 = 0$; $x_2 = x_3 = x_5 = 1$ và có giá trị bằng 0 trong mọi trường hợp còn lại.

b. Định nghĩa dạng tuyển chuẩn tắc của hàm Boole.

Định nghĩa:

- **Dạng tuyển chuẩn tắc** của hàm Boole là biểu diễn hàm Boole dưới dạng tổng các hội sơ cấp.
- Nếu mọi hội sơ cấp trong dạng tuyển chuẩn tắc của hàm Boole n biến đều có hạng bằng n thì dạng tuyển chuẩn tắc đó gọi là **dạng tuyển chuẩn tắc hoàn toàn** của hàm đã cho.

Nói chung, cả hai dạng nêu trong định nghĩa trên có thể gọi chung là **dạng tuyển chuẩn tắc** hay là **khai triển tổng các tích** của hàm Boole.

Có thể tìm dạng tuyển chuẩn tắc hoàn toàn của hàm Boole từ bảng giá trị của hàm đó.

Do hội sơ cấp có giá trị bằng 1 khi và chỉ khi mọi tục biến đều bằng 1 và do đó nó bằng 0 trong mọi trường hợp còn lại. Vì vậy bằng cách lấy tổng Boole của các hội sơ cấp phân biệt có giá trị bằng 1 thì được biểu thức Boole dạng tuyển chuẩn tắc hoàn toàn từ tập giá trị cho trước.

Từ đây đến hết chương, nếu không có chú thích thì thuật ngữ "dạng tuyển chuẩn tắc" được hiểu là dạng tuyển chuẩn tắc hoàn toàn.

Thí dụ 1: Tìm dạng tuyển chuẩn tắc của các hàm Boole f = f(x,y,z) và g = g(x,y,z) được cho trong bảng sau:

x	y	z	f	g
0	0	0	0	0
0	0	1	1	0

0	1	0	0	1
0	1	1	0	0
1	0	0	0	1
1	0	1	1	0
1	1	0	0	1
1	1	1	0	0

Trước hết ta xét hàm f(x,y,z)

Ta thấy f(x,y,z) = 1 khi x = 1, y = 0, z = 1 hoặc x = y = 0, z = 1

Hội sơ cấp ứng với bộ giá trị x = y = 0, z = 1 là $\bar{x}\,\bar{y}z$

Hội sơ cấp ứng với bộ giá trị x = z = 1, y = 0 là $x\bar{y}z$

Vậy dạng tuyển chuẩn tắc của hàm f là:

$$f(x,y,z) = \bar{x}\,\bar{y}z + x\bar{y}z$$

Tương tự, có dạng tuyển chuẩn tắc của hàm g là:

$$g(x,y,z) = \bar{x}\,y\bar{z} + x\bar{y}\bar{z} + x\,y\bar{z}$$

Chú thích: Từ bảng giá trị chân lý trên có dạng tuyển chuẩn tắc của hàm phần bù của hàm f và hàm g là:

$$\bar{f}(x,y,z) = x\,y\,z + x\,y\bar{z} + x\bar{y}\bar{z} + \bar{x}\,y\,z + \bar{x}\,y\bar{z} + \bar{x}\bar{y}\bar{z}$$

$$\bar{g}(x,y,z) = x\,y\,z + x\bar{y}z + \bar{x}\,y\,z + \bar{x}\bar{y}z + \bar{x}\bar{y}\bar{z}$$

Vì tổng Boole của các hội sơ cấp có giá trị bằng 1 chỉ khi một trong các hội sơ cấp bằng 1. Do đó với hàm Boole cho bằng biểu thức chưa phải là dạng tuyển chuẩn tắc thì có thể tìm dạng tuyển chuẩn tắc từ bảng giá trị chân lý của nó.

Thí dụ 2: Tìm dạng tuyển chuẩn tắc của hàm $f(x,y,z) = (x+y)\bar{z}$:

Trước hết ta lập bảng giá trị của hàm f:

x	y	z	x + y	$\bar{z}$	$f(x,y,z) = (x+y)\bar{z}$
0	0	0	0	1	0
0	0	1	0	0	0
0	1	0	1	1	1
0	1	1	1	0	0
1	0	0	1	1	1
1	0	1	1	0	0
1	1	0	1	1	1
1	1	1	1	0	0

Phân tích tương tự thí dụ 1 được dạng tuyển chuẩn tắc của hàm f là:

$$f = \bar{x}\,y\bar{z} + x\,\bar{y}\bar{z} + x\,y\bar{z}$$

Thí dụ 3: Từ định nghĩa các hàm Boole hai biến trong mục 1.2, ta có:

$\bar{x}\,y + x\,\bar{y}$ là dạng tuyển chuẩn tắc của hàm tuyển loại $f(x,y) = x \oplus y$

$\bar{x}\,\bar{y} + \bar{x}\,y + x\,\bar{y}$ là dạng tuyển chuẩn tắc của hàm Sheffer $f(x,y) = x \uparrow y$

(Dạng tuyển chuẩn tắc của các hàm 2 biến còn lại được dành cho bạn đọc như là một bài tập)

c. Định lý triển khai

Định lý: Mọi hàm Boole đều có thể biểu diễn được dưới dạng:

$$f(x_1,x_2,\ldots,x_n) = \sum_{(\sigma_1,\sigma_2,\ldots,\sigma_i)\in B^i} x_1^{\sigma_1} K\, x_i^{\sigma_i} f(\sigma_1,\ldots,\sigma_i,x_{i+1},\ldots,x_n), \qquad (1)$$

Trong đó i là số tự nhiên bất kỳ thoả mãn $1 \le i \le n$.

Chứng minh:

Đặt: $T_f = \{(x_1,x_2,\ldots,x_n)\in B^n \mid f(x_1,x_2,\ldots,x_n) = 1\}$

T_f được gọi là tập đặc trưng của hàm f. Ta có:

$$T_{\bar{f}} = \overline{T_f}, \quad T_{f+g} = T_f \cup T_g, \quad T_{f.g} = T_f \cap T_g$$

Gọi g là hàm Boole ở vế phải của (1).

Cho $(x_1,x_2,\ldots,x_n)\in T_f$, khi đó số hạng ứng với bộ giá trị $\sigma_1 = x_1$, $\sigma_2 = x_2$, …,$\sigma_i = x_i$ trong tổng ở vế phải của (1) bằng 1, do đó $(x_1,x_2,\ldots,x_n) \in T_g$.

Đảo lại, nếu $(x_1,x_2,\ldots,x_n) \in T_g$ tức là vế phải của (1) bằng 1. Điều này có nghĩa là tồn tại một số hạng nào ở vế phải của (1) bằng 1, chẳng hạn đó là số hạng ứng với bộ giá trị $(\sigma_1, \ldots,\sigma_i)$, khi đó $x_1 = \sigma_1$, …, $\sigma_i = x_i$ và $f(\sigma_1,\ldots,\sigma_i, x_{i+1},\ldots,x_n) = 1$ hay $(x_1,x_2,\ldots,x_n) \in T_f$.

Vậy $T_f = T_g$ hay f = g. Định lý được chứng minh.

Cho i = 1 và nhận xét rằng vai trò của mọi biến x_i là như nhau, từ định lý được hệ quả sau:

Hệ quả 1. Mọi hàm Boole $f(x_1,\ldots,x_n)$ đều có thể khai triển theo một biến x_i:

$$f(x_1,\ldots,x_n) = \bar{x}_i\, f(x_1,\ldots,x_{i-1},0,x_{i+1},\ldots,x_n) + x_i\, f(x_1,\ldots,x_{i-1},1,x_{i+1},\ldots,x_n)$$

Cho i = n và bỏ đi các nhân tử bằng 1, từ định lý được hệ quả sau:

Hệ quả 2. Mọi hàm Boole $f(x_1,\ldots,x_n)$ đều có thể khai triển dưới dạng:

$$f(x_1,\ldots,x_n) = \sum x_1^{\sigma_1} \ldots x_n^{\sigma_n}$$

Nói cách khác, mọi hàm Boole đều có thể biểu diễn được dưới dạng tuyển chuẩn tắc hoàn toàn.

Chú thích: Bằng cách lấy đối ngẫu dạng tuyển chuẩn tắc của một hàm Boole thì được biểu diễn hàm Boole đó dưới dạng tích các tuyển sơ cấp của các biến. Dạng biểu diễn hàm Boole dưới dạng tích các tuyến sơ cấp được gọi là dạng **hội chuẩn tắc** của hàm Boole đó.

d. Biểu diễn hàm Boole bằng số nguyên

Mỗi biến Boole chỉ nhận hai giá trị 0 và 1 nên nếu thay mỗi biến Boole bằng số 1 và phần bù của mỗi biến Boole bằng số 0 thì mỗi hội sơ cấp các biến Boole có thể xem như một số nguyên được biểu diễn dưới dạng khai triển nhị phân của số đó. Bảng sau cho tương ứng hội sơ cấp của không quá 3 biến Boole với các số nguyên:

Số biến	Hội sơ cấp	Xâu nhị phân tương ứng	Số nguyên tương ứng
2	$\bar{x}\bar{y}$	00	0
	$\bar{x}y$	01	1
	$x\bar{y}$	10	2
	xy	11	3
3	$\bar{x}\bar{y}\bar{z}$	000	0
	$\bar{x}\bar{y}z$	001	1
	$\bar{x}y\bar{z}$	010	2
	$\bar{x}yz$	011	3
	$x\bar{y}\bar{z}$	100	4
	$x\bar{y}z$	101	5
	$xy\bar{z}$	110	6
	xyz	111	7

Với cách cho tương ứng các hội sơ cấp với số nguyên như vậy có thể biểu diễn dạng tuyển chuẩn tắc hoàn toàn của hàm Boole dưới dạng dấu tổng bằng cách thay hội sơ cấp bằng số nguyên tương ứng. Các thí dụ sau cho thấy cách biểu diễn như vậy.

Thí dụ 1: Xét hàm Boole 2 biến $f(x,y) = \bar{x}y + xy$ là tuyển của hai hội sơ cấp ứng với các số nguyên 1 và 2 do vậy có thể viết:

$$f(x,y) = \Sigma(1,2).$$

Thí dụ 2: Hàm Boole 3 biến $f(x,y,z) = xy\bar{z} + x\bar{y}\bar{z} + \bar{x}y\bar{z}$ là tuyển của ba hội sơ cấp ứng với các số nguyên 6, 4 và 2 do vậy có thể viết:

$$f(x,y,z) = \Sigma(2,4,6).$$

Thí dụ 3: Hàm Boole 4 biến:

$$f(x,y,z,u) = \bar{x}\bar{y}\bar{z}u + \bar{x}\bar{y}zu + \bar{x}y\bar{z}u + xy\bar{z}u + x\bar{y}zu$$

là tuyển của 5 hội sơ cấp:

$\bar{x}\bar{y}\bar{z}u$ tương ứng với xâu 0001 biểu diễn số 1;

$\bar{x}\bar{y}zu$ tương ứng với xâu 0011 biểu diễn số 3;

$\bar{x}y\bar{z}\bar{u}$ tương ứng với xâu 0010 biểu diễn số 2;

$x\bar{y}\bar{z}\bar{u}$ tương ứng với xâu 1000 biểu diễn số 8;

$x\bar{y}zu$ tương ứng với xâu 1011 biểu diễn số 11;

$xy\bar{z}\bar{u}$ tương ứng với xâu 1010 biểu diễn số 10;

Vậy có thể viết: $f(x,y,z,u) = \Sigma(1,2,3,8,10,11)$.

Thí dụ 4: Chuyển hàm Boole $f(x,y,z,u) = \Sigma(1,3,5.10,12)$ thành dạng tuyển chuẩn tắc.

Ta có: Số 1 tương ứng với xâu 0001, đó là hội sơ cấp $\bar{x}\bar{y}\bar{z}u$;

Số 3 tương ứng với xâu 0011, đó là hội sơ cấp $\bar{x}\bar{y}zu$;

Số 5 tương ứng với xâu 0101, đó là hội sơ cấp $\bar{x}y\bar{z}u$;

Số 10 tương ứng với xâu 1010, đó là hội sơ cấp $xy\bar{z}\bar{u}$;

Số 12 tương ứng với xâu 1100, đó là hội sơ cấp $xyz\bar{u}$;

Vậy: $f(x,y,z,u) = \bar{x}\bar{y}\bar{z}u + \bar{x}\bar{y}zu + \bar{x}y\bar{z}u + xy\bar{z}u + xy\bar{z}\bar{u} + xyz\bar{u}$.

Với các hàm nhiều biến hơn, việc chuyển đổi giữa dạng tuyển chuẩn tắc và dạng số nguyên của các hàm Boole được tiến hành tương tự.

4.3. Tính đầy đủ của đại số Boole

Mọi hàm Boole đều có thể biểu diễn được dưới dạng tuyển chuẩn tắc. Nói cách khác, mọi hàm Boole đều có thể biểu diễn được dưới dạng tổng Boole của các hội sơ cấp. Mỗi hội sơ cấp là tích Boole của các biến và phần bù của các biến. Vậy có thể nói, mọi hàm Boole đều có thể biểu diễn bằng một biểu thức Boole thông qua các phép toán Boole . , + và $\bar{}$. Người ta gọi tập hợp {. , +, $\bar{}$ } là đầy đủ đối với đại số Boole.

Định nghĩa: Tập đầy đủ của đại số Boole là tập hợp các phép toán Boole mà qua các phép toán đó ta có thể biểu diễn được mọi hàm Boole.

Ngoài tập hợp {. , +, $\bar{}$ } là tập đầy đủ, đại số Boole còn có các tập đầy đủ khác có ít phép toán hơn.

1) Tập hợp {. , $\bar{}$ } và tập hợp { +, $\bar{}$ } là các tập đầy đủ.

Chúng ta chứng minh cho tập hợp {. , $\bar{}$ }. Thật vậy, theo luật Đơ-Moocgan, có thể loại tất cả các tổng Boole bằng cách dùng hằng đẳng thức:

$$x + y = \overline{\bar{x} \wedge \bar{y}} \qquad (\text{do } \overline{\bar{x} \wedge \bar{y}} = \bar{\bar{x}} + \bar{\bar{y}} = x + y)$$

Đối với tập hợp { +, $\bar{}$ } được chứng minh tương tự.

2) Trong đại số Boole, người ta còn đưa thêm hai phép toán hai ngôi: Phép NAND, ký hiệu $\uparrow$, và phép NOR, ký hiệu $\downarrow$, và được định nghĩa như sau:

- Phép NAND: $1 \uparrow 1 = 0, \quad 1 \uparrow 0 = 1, \quad 0 \uparrow 1 = 1, \quad 0 \uparrow 0 = 1$
- Phép NOR: $1 \downarrow 1 = 0, \quad 1 \downarrow 0 = 0, \quad 0 \downarrow 1 = 0, \quad 0 \downarrow 0 = 1$

Cả hai tập hợp { $\uparrow$ } và { $\downarrow$ } là các tập đầy đủ.

Thật vậy, Dễ dàng chứng minh được:

$\bar{x} = x \uparrow x$ (từ định nghĩa 2 phép toán $\bar{}$ và $\uparrow$)

$x.y = (x \uparrow y) \mid (x \uparrow y)$ (dùng bảng giá trị)

Nói cách khác, có thể thay thế hai phép toán . và $\bar{}$ bằng một phép toán $\uparrow$. Tập hợp { . , $\bar{}$ } là tập đầy đủ nên tập hợp { $\uparrow$ } là tập đầy đủ.

Việc chứng minh tập hợp {$\downarrow$ } là tập đầy đủ hoàn toàn tương tự.

5. Các cổng logic

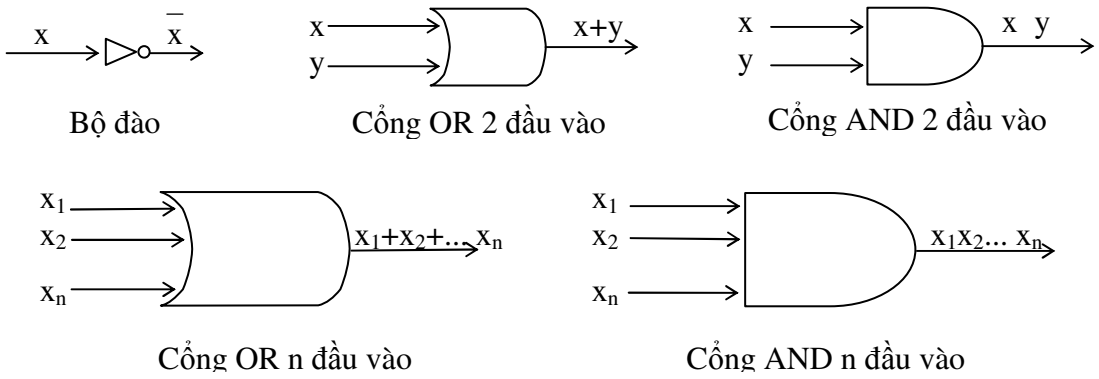
5.1. Khái niệm về các cổng logic

Đại số Boole được dùng để mô hình hoá sơ đồ các mạch trong các dụng cụ điện, điện tử. Mỗi một đầu vào và mỗi một đầu ra của một dụng cụ điện tử được xem là một phần tử của tập {0, 1}. Một máy tính cũng như một dụng cụ điện tử được tạo bởi nhiều mạch. Mỗi một mạch có thể được thiết kế bằng cách dùng các quy tắc của đại số Boole.

Các phần tử cơ bản của mạch gọi là các cổng. Mỗi loại cổng thực hiện một phép toán Boole.

Nhờ các quy tắc của đại số Boole có thể thiết kế được các mạch từ các cổng. Ở đây chỉ đề cập đến các mạch mà đầu ra chỉ phụ thuộc vào đầu vào chứ không phụ thuộc vào trạng thái hiện thời của mạch. Những mạch như vậy gọi là mạch logic tổ hợp.

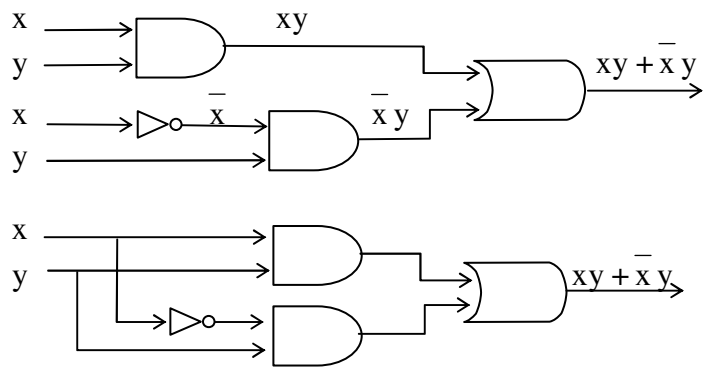
Có 3 loại cổng logic: Bộ đảo, Cổng OR, Cổng AND



Hình 1. Các cổng lôgic

5.2. Tổ hợp các cổng

Các mạch tổ hợp có thể được thiết kế bằng cách dùng tổ hợp các cổng OR, cổng AND, bộ đảo. Khi tổ hợp các mạch, một số cổng có thể dùng chung đầu vào, chẳng hạn hai cách vẽ trong hình 2 đều chỉ cùng một đầu ra là $x\,y + \bar{x}\,y$.



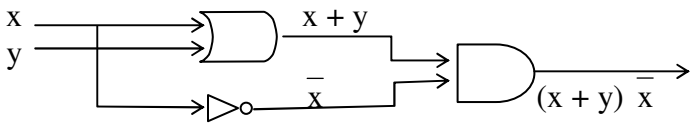
Hình 2

Thí dụ 1: Dựng các mạch tạo các đầu ra sau:

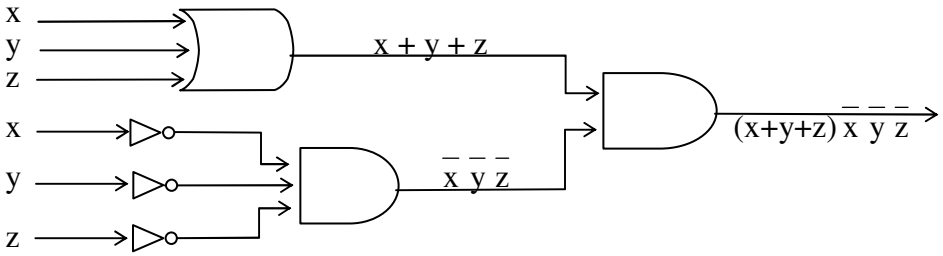
a) $(x + y)\,\bar{x}$ b) $(x + y + z)\,\bar{x}\,\bar{y}\,\bar{z}$

Giải:

a)



b)



Thí dụ 2: Nhiều khi hệ thống đèn cô định được điều khiển bởi nhiều công tăc sao cho khi bật hoặc tắt một công tắc bất kỳ đèn đang sáng sẽ tắt hoặc đang tắt sẽ sáng. Hãy thiết kế một mạch như vậy khi có hai công tắc.

Giải: Gọi 2 công tắc là x và y. Khi x = 1 là công tắc thứ nhất đóng và x = 0 là công tắc thứ nhất mở. Tương tự đối với công tắc thứ hai.

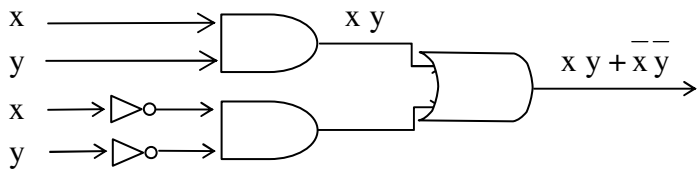
Giả sử f(x,y) = 1 là đèn sáng và f(x,y) = 0 là đèn tắt. Chúng ta có thể hoàn toàn tự chọn để đèn sáng khi cả 2 công tắc đều đóng, tức là f(1, 1) = 1, điều này sẽ dẫn tới việc xác định các giá trị khác của hàm f theo giá trị các biến. Khi một trong hai công tắc mở đèn sẽ tắt, tức là f(1, 0) = f(0, 1) = 0, và khi cả 2 công tắc đều mở thì đèn lại sáng, tức là f(0, 0) = 1. Lập bảng các giá trị của hàm f:

x	y	f
1	1	1
1	0	0
0	1	0
0	0	1

Từ đó biểu diễn được hàm là:

$$f(x,y) = x\,y + \bar{x}\,\bar{y}$$

Và, do đó thiết kế được mạch theo yêu cầu (hình 3)



Hình 3. Mạch điều khiển bằng 2 công tắc

6. Tối thiểu hoá hàm Boole

Như đã biết, việc thiết kế các mạch dựa trên biểu thức Boole biểu diễn hàm Boole cho đầu ra của mạch, các mạch càng đơn giản nếu biểu thức Boole biểu diễn hàm Boole đã cho càng đơn giản. Bởi vậy cần phải tìm biểu thức đơn giản nhất để biểu diễn hàm Boole đó sao cho đầu ra là không đổi so với trước khi đơn giản hoá biểu thức đó. Công việc này gọi là tối thiểu hoá hàm Boole.

Có ba phương pháp tối thiểu hoá hàm Boole.

6.1. Phương pháp biến đổi đại số

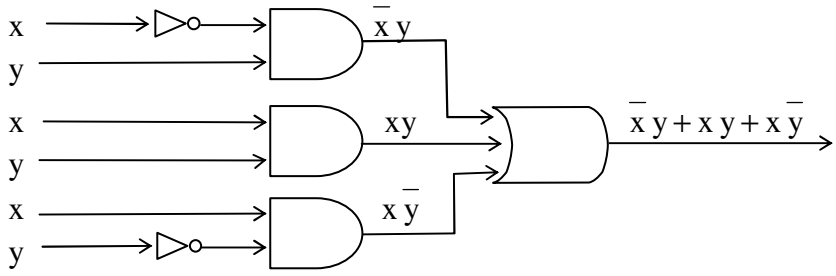
Phương pháp này dựa vào các hằng đẳng thức của đại số Boole (các luật được trình bày trong 2.2). Sau đây là một số thí du:

Thí dụ 1: Tối thiểu hoá hàm Boole sau: $f(x,y) = \bar{x}\,y + x\,y + x\,\bar{y}$.

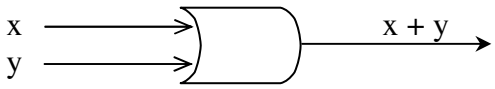
Giải: Ta có:

$$\begin{aligned} f(x,y) = \bar{x}\,y + x\,y + x\,\bar{y} &= \bar{x}\,y + x(y + \bar{y}) && \text{(Luật phân phối)} \\ &= \bar{x}\,y + x.1 && \text{(Luật nuốt)} \\ &= (\bar{x} + x)\,(y + x) && \text{(Luật phân phối)} \\ &= 1.\,(y + x) = y + x && \text{(Luật nuốt)} \end{aligned}$$

Hình 4a và hình 4b là biểu diễn mạch tổ hợp có đầu ra là f(x,y) tương ứng với trước và sau khi tối thiểu hoá.



Hình 4a



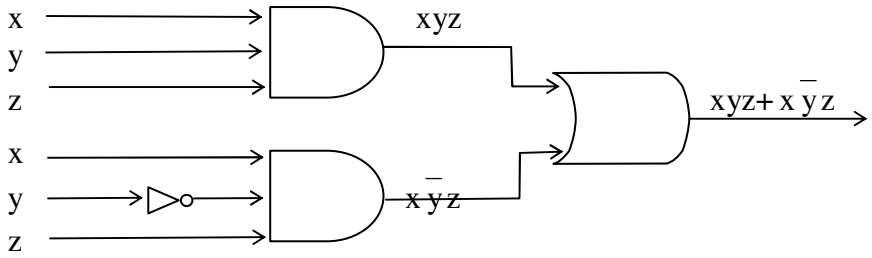
Hình 4b

Thí dụ 2: Tối thiểu hoá hàm Boole $f(x,y,z) = x\,y\,z + x\,\bar{y}\,z$

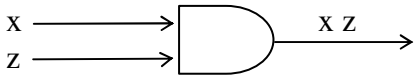
Giải: Áp dụng các luật giao hoán, nuốt, kết hợp. Ta có:

$$f(x,y,z) = x\,y\,z + x\,\bar{y}z = y\,x\,z + \bar{y}\,x\,z = (y+\bar{y})\,x\,z = 1.(xz) = xz$$

Các hình 5a và 5b là tương ứng với các mạch tổ hợp của trước và sau tối thiểu hoá hàm Boole đã cho.



Hình 5a



Hình 5b

6.2. Phương pháp lập bảng Karnaugh

Bảng Karnaugh (còn gọi là bản đồ Karnaugh) cho chúng ta một phương pháp trực quan để rút gọn dạng tuyển chuẩn tắc hoàn toàn của các hàm Boole. Phương pháp chỉ được áp dụng với các hàm Boole có không quá 6 biến, tuy nhiên với trên 4 biến nó đã rất khó sử dụng.

a. Bảng Karnaugh của hàm 2 biến.

Đối với hàm Boole 2 biến, ta có $2^2 = 4$ hội sơ cấp có thể là: $x\,y$, $x\,\bar{y}$, $\bar{x}\,y$ và $\bar{x}\,\bar{y}$. Vì vậy bảng Karnaugh đối với hàm Boole 2 biến có 4 ô để biểu diễn 4 hội sơ cấp có thể đó.

	$\bar{y}$	y
$\bar{x}$	$\bar{x}\,\bar{y}$	$\bar{x}\,y$
x	$x\,\bar{y}$	$x\,\bar{y}$

Hai ô của bảng Karnaugh được gọi là kề nhau nếu các hội sơ cấp của chúng chỉ khác nhau một tục biến.

Nếu hội sơ cấp nào có mặt trong biểu thức biểu diễn hàm Boole thì ô tương ứng trong bảng Karnaugh được ghi số 1, còn nếu không có mặt thì bỏ trống. Bất cứ 2 ô kề nhau nào

được ghi số 1 thì 2 hội sơ cấp tương ứng 2 ô đó được rút gọn chỉ còn lại 1 tục biến. Chẳng hạn $x\bar{y}$ và $\bar{x}\bar{y}$ được rút gọn thành $\bar{y}$ vì $x\bar{y}+\bar{x}\bar{y}=(x+\bar{x})\bar{y}=\bar{y}$.

Thí dụ 1: Cực tiểu hoá các hàm sau bằng bảng Karnaugh:

a) $f(x,y)=x\bar{y}+\bar{x}y$; b) $g(x,y)=x\bar{y}+\bar{x}y+\bar{x}\bar{y}$

Giải:

a)

	$\bar{y}$	y
$\bar{x}$		1
x		1

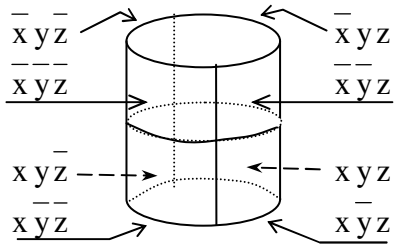
b)

	$\bar{y}$	y
$\bar{x}$	1	1
x	1	

Từ đó: $f(x,y)=y$ $g(x,y)=\bar{x}+\bar{y}$

b. Bảng Karnaugh của hàm 3 biến

Bảng Karnaugh cho hàm 3 biến cần $2^3 = 8$ ô để biểu diễn 8 hội sơ cấp có thể của 3 biến x, y, z. Hai ô được gọi là kề nhau nếu hội sơ cấp tương ứng với chúng chỉ khác nhau một tục biến. Bản đồ Karnaugh 3 biến được xem như nằm trên mặt trụ (hình 6).



Hình 6

Dàn phẳng mặt trụ ở hình 6 ta được bảng Karnaugh ba biến:

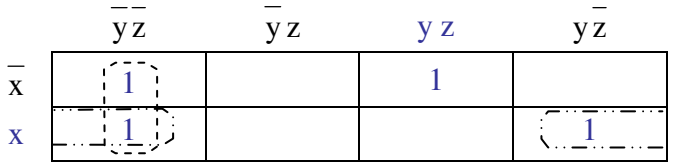
	$\bar{y}\bar{z}$	$\bar{y}z$	yz	$y\bar{z}$
$\bar{x}$	$\bar{x}\bar{y}\bar{z}$	$\bar{x}\bar{y}z$	$\bar{x}yz$	$\bar{x}y\bar{z}$
x	$x\bar{y}\bar{z}$	$x\bar{y}z$	xyz	$xy\bar{z}$

Nếu hội sơ cấp nào có mặt trong biểu thức biểu diễn hàm Boole thì ô tương ứng trong bảng Karnaugh được ghi số 1, còn nếu không có mặt thì bỏ trống. Bất cứ hai ô kề nhau nào được ghi số 1 thì hai hội sơ cấp tương ứng hai ô đó được rút gọn thành một hội sơ cấp có 2 tục biến. Các khối gồm bốn ô kiểu 2 x 2 hoặc 4 x 1 có thể rút gọn thành hội sơ cấp chỉ còn 1 tục biến. Khối cả tám ô có thể rút gọn về hằng là 1.

Nguyên tắc rút gọn là bắt đầu từ khối lớn nhất kề nhau có chứa số 1.

Thí dụ 2: Cực tiểu hoá các hàm Boole sau bằng bảng Karnaugh:

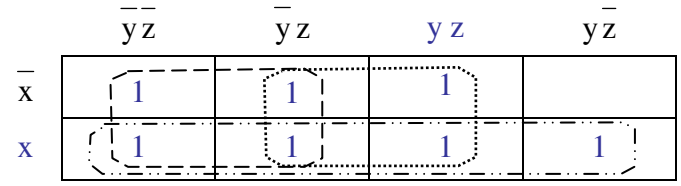
a) $f(x,y,z)=xy\bar{z}+x\bar{y}\bar{z}+\bar{x}yz+\bar{x}\bar{y}\bar{z}$



Ta có: $\bar{x}\bar{y}\bar{z} + x\bar{y}\bar{z} = \bar{y}\bar{z}$; $x\bar{y}\bar{z} + xy\bar{z} = x\bar{z}$

Vậy: $f = \bar{y}\bar{z} + x\bar{z} + \bar{x}yz$

b) $g(x,y,z) = xyz + xy\bar{z} + x\bar{y}z + x\bar{y}\bar{z} + \bar{x}yz + \bar{x}\bar{y}z + \bar{x}\bar{y}\bar{z}$



Ta có: $\bar{x}\bar{y}\bar{z} + \bar{x}\bar{y}z + x\bar{y}\bar{z} + x\bar{y}z = \bar{x}\bar{y} + x\bar{y} = \bar{y}$

$\bar{x}\bar{y}z + \bar{x}yz + x\bar{y}z + xyz = \bar{x}z + xz = z$

$x\bar{y}\bar{z} + x\bar{y}z + xyz + xy\bar{z} = x\bar{y} + xy = x$

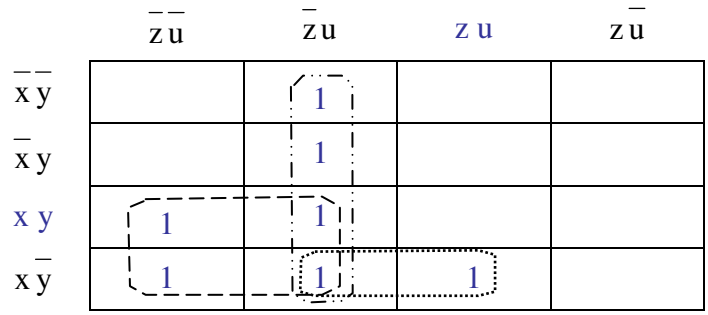
Vậy: $g = x + \bar{y} + z$

c. Bảng Karnaugh của hàm 4 biến

Với bảng Karnaugh của hàm Boole 4 biến gồm 16 ô chia là 4 hàng và 4 cột. Cứ 2 ô kề nhau có ghi số 1 được rút gọn thành hội sơ cấp 3 tục biến; 4 ô kề nhau có ghi số 1 được rút gọn thành hội sơ cấp có 2 tục biến; 8 ô kề nhau rút gọn thành hội sơ cấp 2 tục biến và các 16 ô kề nhau rút gọn thành hằng 1.

Thí dụ 3: Dùng bảng Karnaugh rút gọn hàm Boole sau:

$f(x,y,z,u) = x\bar{y}\bar{z}\bar{u} + x\bar{y}zu + x\bar{y}z\bar{u} + \bar{x}y\bar{z}u + \bar{x}yz\bar{u} + \bar{x}\bar{y}zu + \bar{x}\bar{y}z\bar{u}$



Ta có: $\bar{x}\bar{y}zu + \bar{x}yzu + x\bar{y}zu + xyzu = \bar{x}zu + xzu = zu$

$x\bar{y}\bar{z}\bar{u} + x\bar{y}z\bar{u} + xy\bar{z}\bar{u} + xyz\bar{u} = x\bar{y}z + xy\bar{z} = x\bar{z}$

$x\bar{y}\bar{z}u + x\bar{y}zu = x\bar{y}u$

Vậy: $f = x\bar{z} + \bar{z}u + x\bar{y}u$

Thí dụ 4: Một hội đồng có 3 thành viên nhóm họp nhằm thông qua một vấn đề nào đó theo nguyên tắc đa số. Mỗi thành viên bỏ phiếu tán thành hoặc không tán thành cho vấn đề được đặt ra. Vấn đề được thông qua nếu có ít nhất hai thành viên hội đồng bỏ phiếu tán thành. Hãy thiết kế một mạch logic cho phép xác định vấn đề được thông qua hay không?

Giải: Gọi x, y, z là ba thành viên của hội đồng. Mỗi biến x, y, z nhận giá trị 1 khi thành viên tương ứng tán thành và nhận giá trị 0 khi không tán thành, ta có bảng giá trị chân lý của bài toán:

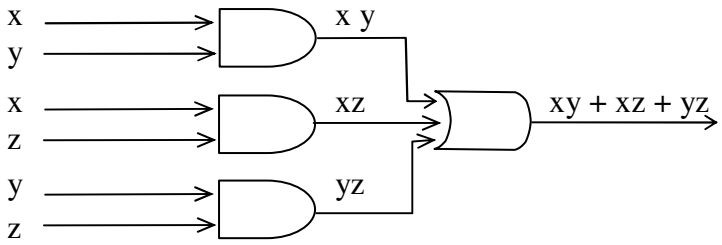
x	y	z	f
1	1	1	1
1	1	0	1
1	0	1	1
1	0	0	0
0	1	1	1
0	1	0	0
0	0	1	0
0	0	0	0

Vậy $f(x,y,z) = x\,y\,z + x\,y\,\bar{z} + x\,\bar{y}\,z + \bar{x}\,y\,z$

Tối thiểu hóa hàm thu được bằng bảng Karnaugh:

	$\bar{y}\,\bar{z}$	$\bar{y}\,z$	$y\,z$	$y\,\bar{z}$
$\bar{x}$			1	
x		1	1	1

được: $f(x,y,z) = xy + xz + yz$. Vậy ta có mạch logic sau:

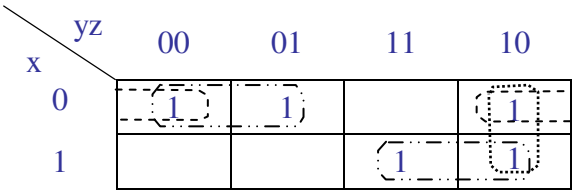


Hình 7. Mạch bỏ phiếu theo đa số của 3 thành viên

Thí dụ 5: Cực tiểu hóa các hàm Boole sau bằng bảng Karnaugh:

a) $f(x,y,z) = \Sigma(0, 1, 2, 6, 7)$.

Để thiết lập bảng Karnaugh cho hàm Boole cho theo dạng số nguyên nên thay ký hiệu biến bằng số 1 và phủ định của biến bằng số 0 và được dạng bảng sau:



Ta có:

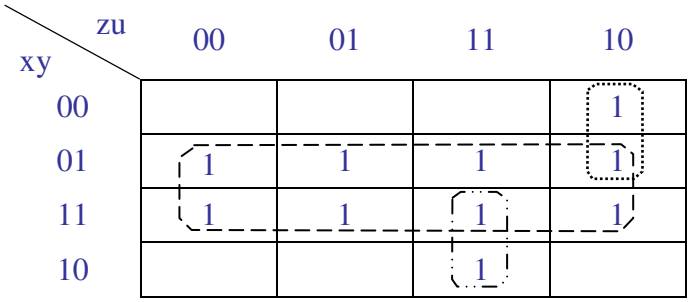
$\bar{x}\bar{y}\bar{z} + \bar{x}y\bar{z} = \bar{x}\bar{z}$; $\bar{x}\bar{y}z + \bar{x}yz = \bar{x}y$; $\bar{x}yz + x\bar{y}z = y\bar{z}$; $xyz + xy\bar{z} = xy$

Chú ý rằng có thể viết 4 tổ hợp trên tương ứng như sau:

$\Sigma(0,2) = \bar{x}\bar{z}$; $\Sigma(0,1) = \bar{x}\bar{y}$; $\Sigma(2,6) = y\bar{z}$; $\Sigma(6,7) = xy$

Vậy: $f = \bar{x}\bar{y} + \bar{x}\bar{z} + y\bar{z} + xy$

b) g(x,y,z,u) = Σ(2, 4, 5, 6, 7, 11, 12, 13, 14, 15).



Ta có: $\bar{x}y\bar{z}\bar{u} + \bar{x}y\bar{z}u + \bar{x}yzu + \bar{x}yz\bar{u} + xy\bar{z}\bar{u} + xy\bar{z}u + xyzu + xyz\bar{u} = y$;

$\bar{x}\bar{y}z\bar{u} + \bar{x}yz\bar{u} = \bar{x}z\bar{u}$; $xyzu + x\bar{y}zu = xzu$.

Hay: $\Sigma(4,5,6,7,12,13,14,15) = y$; $\Sigma(2,6) = \bar{x}z\bar{u}$ $\Sigma(11,15) = xyz$.

Vậy: $f = y + \bar{x}z\bar{u} + xyz$.

Bảng Karnaugh của hàm Boole n biến có 2^n ô gồm $2^{\frac{n}{2}}$ cột và $2^{\frac{n}{2}}$ hàng nếu n chẵn; còn nếu n lẻ có $2^{\frac{n+1}{2}}$ cột và $2^{\frac{n-1}{2}}$ hàng; và cứ 2^k ($1 \le k \le n$) ô kề nhau được rút gọn thành hội sơ cấp có n – k tục biến. Như vậy bảng Karnaugh của hàm Boole 7 biến gồm 16 cột, 8 hàng do đó không có khả năng quan sát hết các ô. Trong thực tế bảng Karnaugh 5 biến đã rất khó sử dụng. Phương pháp Quine – Mc Cluskey sau đây sẽ khắc phục được tình trạng đó.

6.3. Phương pháp xâu bit (Phương pháp Quine – Mc Cluskey)

Phương pháp Quine – Mc Cluskey ra đời năm 1950 có thể áp dụng để rút gọn hàm Boole với số biến tuỳ ý. Có thể chia phương pháp này thành 3 bước:

Bước 1: Gán xâu bit cho các hội sơ cấp của hàm đã cho bằng cách thay biến bằng bit 1 và phần bù của biến bằng bit 0.

Bước 2: Tổ hợp các cặp xâu bit thành xâu bít ngắn hơn. Bước này được lặp cho các cặp xâu bit ngắn hơn thu được ở bước lặp trước cho đến khi không tổ hợp được nữa thì

dừng. Cuối cùng được các xâu bít là các ứng viên để đưa vào dạng tuyển chuẩn tắc tối thiểu.

*Bước 3:*Xác định xem trong các ứng viên đã chọn thì ứng viên nào thực sự dùng được.

Phương pháp được mô tả cụ thể qua các thí dụ sau:

Thí dụ 1: Tối thiểu hoá hàm Boole sau:

$$f(x,y,z) = x\,y\,z + x\,\bar{y}\,z + \bar{x}\,y\,z + \bar{x}\,\bar{y}\,z + \bar{x}\,\bar{y}\,\bar{z}$$

Bước 1. Trước hết biểu diễn các hội sơ cấp có mặt trong hàm f bằng các xâu bit theo nguyên tắc sau: Các biến Boole không có dấu phủ định được gán số 1, và có dấu phủ định thì gán số 0. Có thể trình bày bước này trong bảng sau:

T.T	Hội sơ cấp	Xâu bit	Số các số 1
1	$x\,y\,z$	1 1 1	3
2	$x\,\bar{y}\,z$	1 0 1	2
3	$\bar{x}\,y\,z$	0 1 1	2
4	$\bar{x}\,\bar{y}\,z$	0 0 1	1
5	$\bar{x}\,\bar{y}\,\bar{z}$	0 0 0	0

Chú ý rằng trong bảng vừa lập, cần phải liệt kê các hội sơ cấp theo thứ tự giảm dần của số các bit 1 của xâu bit tương ứng.

Bước 2. Tiếp theo, 2 hội sơ cấp có thể tổ hợp được với nhau nếu chúng chỉ khác nhau một tục biến, nghĩa là 2 xâu bit tương ứng chỉ khác nhau một bit ở cùng một vị trí. Khi tổ hợp 2 xâu bit như vậy, hai bit khác nhau ở cùng vị trí của 2 xâu đó được loại bỏ trong xâu bít tổ hợp và được thay bằng một dấu gạch ngang còn các bít khác được giữ nguyên. Trong thí dụ đang xét, các cặp xâu bít (1, 2) ; (1, 3) ; (2, 4) ; (3, 4) và (4, 5) có thể tổ hợp được với nhau. Cặp (1, 2) là $x\,y\,z + x\,\bar{y}\,z = x\,z$ và như vậy cặp xâu bít (1, 2) là 2 xâu 111 và 101 được tổ hợp thành xâu 1-1. Tương tự với các cặp còn lại và tiếp tục như vậy trong các bước lặp tiếp theo. Có thể trình bày bước này trong bảng:

Khởi đầu			Tổ hợp lần I		Tổ hợp lần II	
TT	Hội sơ cấp	Xâu bit	Hội sơ cấp	Xâu bit	Hội sơ cấp	Xâu bit
1	$x\,y\,z$	1 1 1	(1,2) $x\,z$	1 - 1	(1,2,3,4) z	- - 1
2	$x\,\bar{y}\,z$	1 0 1	(1,3) $y\,z$	- 1 1		
3	$\bar{x}\,y\,z$	0 1 1	(2,4) $\bar{y}\,z$	- 0 1		
4	$\bar{x}\,\bar{y}\,z$	0 0 1	(3,4) $\bar{x}\,z$	0 - 1		
5	$\bar{x}\,\bar{y}\,\bar{z}$	0 0 0	(4,5) $\bar{x}\,\bar{y}$	0 0 -		

Các ứng viên để đưa vào dạng tuyển chuẩn tắc tối thiểu là các hội sơ cấp chưa được dùng đến để tổ hợp thành các hội sơ cấp có số tục biến ít hơn (trong thí dụ trên là z và $\bar{x}\,\bar{y}$). Các

ứng viên này không nhất thiết phải có mặt trong dạng tuyển chuẩn tắc tối thiểu của hàm Boole.

Bước 3. Cuối cùng là kiểm tra xem các ứng viên có phủ hết các hội sơ cấp ban đầu của hàm Boole đã cho không. Khái niệm phủ có nghĩa là ứng viên đó có mặt trong hội sơ cấp ban đầu thì nó thay thế được cho hội sơ cấp đó. Nếu ứng viên có mặt trong hội sơ cấp ban đầu nào thì ta đánh dấu × cho hội sơ cấp đó. Bảng sau đây làm công việc này:

Ứng viên	$x\,y\,z$	$x\,\bar{y}\,z$	$\bar{x}\,y\,z$	$\bar{x}\,\bar{y}\,z$	$\bar{x}\,\bar{y}\,\bar{z}$
z	×	×	×	×	
$\bar{x}\,\bar{y}$				×	×

Từ bảng trên thấy z có mặt trong 4 hội sơ cấp của hàm f còn $\bar{x}\,\bar{y}$ có mặt trong hai hội sơ cấp cuối của hàm f. Như vậy hai ứng viên z và $\bar{x}\,\bar{y}$ là phủ hết các hội sơ cấp ban đầu của hàm f.

Vậy dạng tối thiểu của hàm f(x,y,z) là: $f(x,y,z) = z + \bar{x}\,\bar{y}$

Thí dụ 2: Tối thiểu hoá hàm Boole sau:

$$f(x,y,z,u) = x\,y\,\bar{z}\,u + \bar{x}\,y\,z\,u + x\,y\,z\,\bar{u} + \bar{x}\,y\,\bar{z}\,u + x\,\bar{y}\,\bar{z}\,\bar{u} + \bar{x}\,\bar{y}\,z\,\bar{u} + \bar{x}\,\bar{y}\,\bar{z}\,\bar{u}$$

Bước 1: Gán xâu bit cho các hội sơ cấp:

T.T	Hội sơ cấp	Xâu bit	Số các số 1
1	$x\,y\,\bar{z}\,u$	1 1 0 1	3
2	$\bar{x}\,y\,z\,u$	0 1 1 1	3
3	$x\,y\,z\,\bar{u}$	1 1 1 0	3
4	$\bar{x}\,y\,\bar{z}\,u$	0 1 0 1	2
5	$x\,\bar{y}\,\bar{z}\,\bar{u}$	1 0 1 0	2
6	$\bar{x}\,y\,z\,\bar{u}$	0 1 1 0	2
7	$\bar{x}\,\bar{y}\,\bar{z}\,\bar{u}$	0 0 1 0	1

Bước 2: Tổ hợp các hội sơ cấp:

Khởi đầu			Tổ hợp lần I		TTổ hợp lân II	
TT	Hội sơ cấp	Xâu bit	Hội sơ cấp	Xâu bit	Hội sơ cấp	Xâu bít
1	$x\,y\,\bar{z}\,u$	1 1 0 1	(1,4) $y\,\bar{z}\,u$	- 1 0 1	(3,5,6,7) $z\,\bar{u}$	- - 1 0
2	$\bar{x}\,y\,z\,u$	0 1 1 1	(2,4) $\bar{x}\,y\,u$	0 1 - 1		
3	$x\,y\,z\,\bar{u}$	1 1 1 0	(2,6) $\bar{x}\,y\,z$	0 1 1 -		

4	$\bar{x}y\bar{z}u$	0 1 0 1	(3,5)	$xz\bar{u}$	1 - 1 0		
5	$x\bar{y}z\bar{u}$	1 0 1 0	(3,6)	$yz\bar{u}$	- 1 1 0		
6	$\bar{x}yz\bar{u}$	0 1 1 0	(5,7)	$\bar{y}z\bar{u}$	- 0 1 0		
7	$\bar{x}\bar{y}z\bar{u}$	0 0 1 0	(6,7)	$\bar{x}z\bar{u}$	0 - 1 0		

Các ứng viên (các hội sơ cấp không tổ hợp được tiếp) có thể được sử dụng trong biểu thức tối thiểu của hàm Boole là: $y\bar{z}u$; $\bar{x}yu$; $\bar{x}yz$; $z\bar{u}$.

Bước 3: Xét sự có mặt của các ứng viên trong các hội sơ cấp ban đầu

Ứng viên	$xy\bar{z}u$	$\bar{x}yzu$	$xyz\bar{u}$	$\bar{x}y\bar{z}u$	$x\bar{y}z\bar{u}$	$\bar{x}yz\bar{u}$	$\bar{x}\bar{y}z\bar{u}$
$y\bar{z}u$	×			×			
$\bar{x}yu$		×		×			
$\bar{x}yz$		×				×	
$z\bar{u}$			×		×	×	×

Các hội sơ cấp: $y\bar{z}u$; $\bar{x}yu$; $z\bar{u}$ phủ hết các hội sơ cấp ban đầu. Vì vậy dạng tối thiểu của hàm f là:

$$f(x,y,z,u) = y\bar{z}u + \bar{x}yu + z\bar{u}$$

Các hội sơ cấp: $y\bar{z}u$; $\bar{x}yz$; $z\bar{u}$; cũng phủ hết các hội sơ cấp ban đầu, do đó có dạng tối thiểu hoá thứ hai của hàm f: $f(x,y,z,u) = y\bar{z}u + \bar{x}yz + z\bar{u}$

Bài toán có hai đáp số.

BÀI TẬP CHƯƠNG 7

Hàm Boole và Biểu diễn các hàm Boole

7.1 Bằng cách lập bảng giá trị, hãy chứng minh rằng:

a) $x\bar{y} + y\bar{z} + \bar{x}z = \bar{x}y + \bar{y}z + x\bar{z}$

b) Các luật giao hoán, kết hợp và Đơ-Moocgan.

7.2. Phép toán XOR, ký hiệu $\oplus$, được định nghĩa như sau:

$$1 \oplus 1 = 0, \quad 1 \oplus 0 = 1, \quad 0 \oplus 1 = 1, \quad 0 \oplus 0 = 0$$

a) Rút gọn các biểu thức sau: $x \oplus 0$; $x \oplus 1$; $x \oplus x$; $x \oplus \bar{x}$

b) Chứng minh rằng: $x \oplus y = (x + y)\overline{(xy)}$

$$x \oplus y = x\bar{y} + \bar{x}y$$

7.3. Tìm đối ngẫu của các biểu thức sau:

a) $x\,y\,z + \bar{x}\bar{y}\bar{z}$; b) $x\bar{z} + x.0 + \bar{x}.1$

7.4. Tìm dạng tuyển chuẩn tắc của các hàm Boole sau:

a) $f(x,y) = x\bar{y}$; b) $f(x,y) = 1$; c) $f(x,y,z) = x + y + z$

d) $f(x,y,z) = (x + z)\,y$; e) $f(x,y,z) = x$; f) $f(x,y,z) = x\bar{y}$

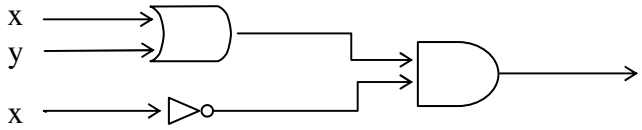
7.5. Tìm dạng tuyển chuẩn tắc của các hàm Boole 3 biến, biết f(x,y,z) bằng 1 nếu và chỉ nếu:

a) $x = 0$; b) $xy = 0$; c) $x + y = 0$; d) $xyz = 0$

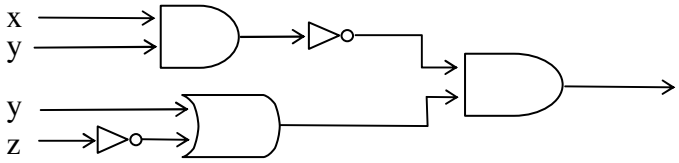
Các cổng logic

7.6. Tìm đầu ra cho các mạch sau:

a)



b)



7.7. Dựng các mạch gồm bộ đảo, các cổng OR, AND để tạo các đầu ra sau:

a) $\bar{x}\bar{y}$; b) $(x+y)\,x$ c) $x\,y\,z + \bar{x}\bar{y}\bar{z}$

7.8. Thiết kế một mạch thực hiện việc điều khiển một bóng đèn bằng 3 công tắc, sao cho khi thay đổi trạng thái của bất kỳ một công tắc nào thì đèn đang sáng sẽ tắt và ngược lại.

7.9. Xây dựng một mạch để so sánh hai số nguyên hai bit $x = (x_1x_0)_2$ và $y = (y_1y_0)_2$ và cho đầu ra bằng 1 nếu $x > y$ và bằng 0 trong các trường hợp còn lại.

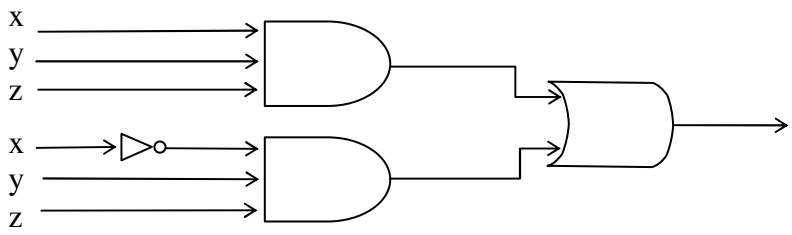
Tối thiểu hoá hàm Boole

7.10. Dùng bảng Karnaugh tối thiểu hoá các hàm Boole sau:

a) $f(x,y) = x\,y + x\bar{y}$; b) $f(x,y) = x\,y + x\bar{y} + \bar{x}\,y + \bar{x}\bar{y}$;

c) $f(x,y,z) = \bar{x}\,y\,z + \bar{x}\bar{y}\,z$; d) $f(x,y,z) = x\,y\,z + x\,y\bar{z} + \bar{x}\,y\,z + \bar{x}\,y\,\bar{z}$;

7.11. Dùng bảng Karnaugh tìm mạch đơn giản hơn có cùng đầu ra như mạch trong hình dưới đây.



7.12. Thiết kế một mạch logic thực hiện việc bỏ phiếu theo đa số cho một hội đồng gồm năm thành viên.

7.13. Cực tiểu hoá các hàm trong bài 7.10 bằng phương pháp xâu bit.

7.14. Cực tiểu hoá các hàm sau bằng hai phương pháp: xâu bit và Karnaugh:

a) $f(x,y,z) = x y \bar{z} + x \bar{y} z + x \bar{y} \bar{z} + \bar{x} y z + \bar{x} \bar{y} z$

b) $f(x,y,z) = x y z + x \bar{y} z + x \bar{y} \bar{z} + \bar{x} y z + \bar{x} y \bar{z} + \bar{x} \bar{y} z$

c) $f(x,y,z,u) = x y z u + x \bar{y} \bar{z} u + x \bar{y} z u + \bar{x} y \bar{z} u + \bar{x} \bar{y} z u$

d) $f(x,y,z,u) = x y \bar{z} u + \bar{x} \bar{y} z u + \bar{x} y z u + x \bar{y} \bar{z} \bar{u} + \bar{x} y \bar{z} u + \bar{x} \bar{y} z \bar{u}$

e) $f(x,y,z,u) = x y z u + x y \bar{z} u + x \bar{y} z u + \bar{x} y z u + \bar{x} y \bar{z} u + x y z \bar{u} + \bar{x} y z \bar{u} + \bar{x} y \bar{z} u + \bar{x} \bar{y} z \bar{u}$

7.15. Cực tiểu hoá các hàm sau bằng hai phương pháp: xâu bit và bảng Karnaugh:

a) $f(x,y,z) = \Sigma(0,2,4,6,7)$; b) $f(x,y,z) = \Sigma(0,1,2,3,4,5,6)$;

c) $f(x,y,z,u) = \Sigma(0,4,6,8,10,12,14)$; d) $f(x,y,z,u) = \Sigma(1,3,8,9,10,11,14,15)$

ĐÁP SỐ

7.2. a) $x \oplus 0 = x$; $x \oplus 1 = \bar{x}$; $x \oplus x = 0$; $x \oplus \bar{x} = 1$

7.4. a) $x \bar{y}$; b) $x y + \bar{x} y + \bar{x} \bar{y} + x \bar{y}$;

c) $\bar{x} \bar{y} \bar{z} + x \bar{y} \bar{z} + \bar{x} y \bar{z} + \bar{x} \bar{y} z + x y \bar{z} + x \bar{y} z + \bar{x} y z$; d) $x y z + x y \bar{z} + \bar{x} y z$

e) $x y z + x y \bar{z} + x \bar{y} z + x \bar{y} \bar{z}$; f) $x \bar{y} z + x \bar{y} \bar{z}$

7.5. a) $\bar{x} y z + \bar{x} y \bar{z} + \bar{x} \bar{y} z + \bar{x} \bar{y} \bar{z}$; c) $\bar{x} \bar{y} \bar{z} + \bar{x} \bar{y} z$;

d) $\bar{x} \bar{y} \bar{z} + x \bar{y} \bar{z} + \bar{x} y \bar{z} + \bar{x} \bar{y} z + x y \bar{z} + x \bar{y} z + \bar{x} y z$

7.8. Vẽ mạch logic có đầu ra là $x y z + \bar{x} y \bar{z} + \bar{x} \bar{y} z + \bar{x} y \bar{z}$

7.9. Vẽ mạch logic có đầu ra là $x_1 \overline{y_1} + x_0 \overline{y_0} \left(x_1 y_1 + \overline{x_1} \overline{x_2} \right)$

7.10. a) x; b) 1; c) $\bar{x} y$; d) y

7.12. Vẽ mạch là tổng các tích của 3 trong 5 biến x, y, z, u, t

7.14. a) $\bar{x}z + x\bar{y} + \bar{y}z + x\bar{z}$; b) $\bar{z} + \bar{x}y + x\bar{y}$;

c) $xyz + x\bar{y}u + \bar{x}yzu + \bar{x}\bar{y}zu$; d) $\bar{x}\bar{y}z + \bar{x}zu + \bar{x}y\bar{z}u + xyzu + xy\bar{z}\bar{u}$

e) $\bar{x}y + yu + yz + xzu + \bar{x}z\bar{u}$

7.15. a) $xz + y\bar{z} + xz$; b) $y + z$; c) $\bar{z}\bar{u} + y\bar{z} + x\bar{y}\bar{u} + xz\bar{u}$; d) $x\bar{y} + xz + \bar{x}\bar{y}z$

CÂU HỎI ÔN TẬP CHƯƠNG 7

1. Phát biểu định nghĩa đại số Boole, hàm Boole và các tính chất của biểu thức Boole. Thế nào là đối ngẫu của biểu thức Boole? Nguyên lý đối ngẫu là gì? Thế nào là quy tắc thay thế?

2. Các phương pháp cho hàm Boole? Thế nào là dạng tuyển chuẩn tắc của hàm Boole? Trình bày cách đưa một hàm Boole về dạng tuyển chuẩn tắc.

3. Thế nào là tập đầy đủ các phép toán của đại số Boole? Hãy chỉ ra các tập đầy đủ gồm ba phép toán, tập đầy đủ gồm hai phép toán và tập đầy đủ chỉ một phép toán của đại số Boole.

4. Định nghĩa các cổng Lôgic và cách tổ hợp các cổng Lôgic. Xây dựng mạch Lôgíc biểu quyết của một hội đồng có 5 thành viên, biết mọi vấn đề đem biểu quyết được quyết định theo đa số. Rút gọn hàm Boole thu được.

5. Trình bày phương pháp bảng Karnaugh để rút gọn các một hàm Boole 2, 3 hoặc 4 biến được cho dưới dạng tuyển chuẩn tắc.

6. Trình bày phương pháp Quine – McCluskey (phương pháp xâu bit) để rút gọn một hàm Boole được cho dưới dạng tuyển chuẩn tắc.

PHỤ CHƯƠNG .

ĐẠI CƯƠNG VỀ TOÁN LOGIC

Các quy tắc của logic cho ý nghĩa chính xác của một mệnh đề. Các quy tắc này được sử dụng để phân biệt giữa các lập luận đúng và không đúng.

1. Lôgic mệnh đề

1.1. Khái niệm mệnh đề

Mệnh đề là cơ sở của toán logic. Ở đây chỉ đề cập đến các mệnh đề hoặc đúng, hoặc sại. Chẳng hạn "Trường Đại học Nông nghiệp I có khoa Công nghệ thông tin" là mệnh đề đúng; còn "5 là nghiệm của phương trình $x^2 + 1 = 0$" là mệnh đề sai.

Lôgic mệnh đề không quan tâm tới các mệnh đề không đủ nghĩa để khẳng định nó là đúng hay sai, chẳng hạn "Bây giờ là mấy giờ"; "Tôi nói dối" hay "x + 1 = 2".

Định nghĩa 1: Các mệnh đề hoặc đúng, hoặc sai được gọi là các mệnh đề sơ cấp hay còn gọi là biến mệnh đề, và được ký hiệu bằng các chữ cái thường: p, q, r, s,…

Mệnh đề chỉ nhận một trong hai giá trị là đúng, ký hiệu T (viết tắt của từ True trong tiếng Anh) và sai, ký hiệu F (viết tắt của từ False). T và F gọi là các giá trị chân lý của mệnh đề.

Định nghĩa 2: Hai mệnh đề sơ cấp được gọi là tương đương nếu nó cùng đúng hoặc cùng sai.

Hai mệnh đề p và q tương đương với nhau được ký hiệu là $p \leftrightarrow q$.

Thí dụ 1: "Số 3 là số nguyên tố" và "Trường Đại học Nông nghiệp có khoa Nông học" là hai mệnh đề tương đương.

Thí dụ 2: "1 + 1 = 3" và "6 chia hết cho 3" là hai mệnh đề không tương đương.

1.2. Các phép toán mệnh đề

Từ các mệnh đề sơ cấp có thể tổ hợp thành các mệnh đề phức hợp nhờ các toán tử logic. Đó là các phép toán: tuyển, hội, phủ định, … được định nghĩa trên tập hợp các mệnh đề sơ cấp.

Phép tuyển: Cho p, q là hai mệnh logic. Mệnh đề "p hoặc q", ký hiệu $p \vee q$, là một mệnh đề mới, nó sai khi cả p và q sai và đúng trong các trường hợp còn lại.

Mệnh đề $p \vee q$ gọi là tuyển của hai mệnh đề p và q.

Phép hội: Giả sử p và q là hai mệnh đề logic. Mệnh đề "p và q" là một mệnh đề mới, ký hiệu $p \wedge q$, nó đúng khi cả p và q đúng và sai trong các trường hợp còn lại.

Mệnh đề $p \wedge q$ gọi là hội của hai mệnh đề p và q.

Thí dụ: Cho các mệnh đề p: "Hôm nay là thứ bảy",
q: "Hôm nay trời mưa".

Ta có:

$p \vee q$ là mệnh đề "Hôm nay là thứ bảy hoặc hôm nay trời mưa". Mệnh đề này đúng vào bất kỳ ngày nào là thứ bảy hoặc bất kỳ ngày nào có mưa kể cả ngày thứ bảy có mưa. Nó chỉ sai vào ngày không phải là thứ bảy và ngày đó trời không mưa.

$p \wedge q$ là mệnh đề "Hôm nay là thứ bảy và trời mưa". Mệnh đề này chỉ đúng khi ngày thứ bảy nào đó có mưa.

Chú ý rằng liên từ "hoặc" trong cách nói thông thường đôi khi được sử dụng như một sự lựa chọn loại trừ. Chẳng hạn câu nói: " Khi trúng thưởng có thể chọn giải thưởng là tiền hoặc hiện vật" phải hiểu là chỉ được lấy tiền hoặc lấy hiện vật chứ không thể lấy cả hai. Điều này không được dùng trong logic mệnh đề.

Phép tuyển loại: Cho p và q là hai mệnh đề logic. Mệnh đề tuyển loại của p và q được ký hiệu là $p \oplus q$ là một mệnh đề đúng khi chỉ một trong p hoặc q đúng và sai trong các trường hợp còn lại.

Phép phủ định: Giả sử p là một mệnh đề logic, khi đó "Không phải p" là một mệnh đề khác được gọi là mệnh đề phủ định của mệnh đề p, ký hiệu là $\overline{p}$.

T*hí dụ:* Mệnh đề p là: "Hôm nay là thứ sáu", khi đó $\overline{p}$ là: "Hôm nay không phải là thứ sáu".

Phép kéo theo: Cho p và q là hai mệnh đề logic. Khi đó "p kéo theo q" là một mệnh đề, ký hiệu $p \rightarrow q$, nó chỉ sai khi p đúng và q sai, và đúng trong mọi trường hợp còn lại.

Trong phép kéo theo nói trên thì p được gọi là giả thiết, còn q là kết luận.

Có nhiều cách nói tương đương với "p kéo theo q", các cách nói tương đương thường gặp là:

- "p suy ra q"
- "Nếu p thì q"
- "p là điều kiện đủ của q"
- "q là điều kiện cần của p"

Ngoài ra còn có một số khái niệm liên quan đến phép kéo theo. Giả sử có mệnh đề $p \to q$ (gọi là mệnh đề thuận), khi đó $q \to p$ gọi là mệnh đề đảo, và $\overline{q} \to \overline{p}$ là mệnh đề phản đảo.

Thí dụ 1:

"Nếu tam giác ABC là tam giác đều thì ba góc A, B, C bằng nhau" là mệnh đề thuận. Ta có:

"Nếu ba góc A, B, C bằng nhau thì tam giác ABC là tam giác đều" là mệnh đề đảo.

"Nếu ba góc A, B, C không bằng nhau thì tam giác ABC không là tam giác đều" là mệnh đề phản đảo.

Chú ý rằng khái niệm kéo theo $p \to q$ chỉ sai khi p đúng và q sai, như vậy nó đúng trong các trường hợp: cả p và q đều đúng và khi p sai (bất kể q đúng hay sai). Điều này hoàn toàn khác với mối quan hệ nhân – quả giữa giả thiết và kết luận.

Chẳng hạn mệnh đề: "Nếu hôm nay trời nắng, chúng tôi sẽ đi chơi" là một phép kéo theo. Nếu dùng theo ngôn ngữ thông thường thì vì nó có mối quan hệ nhân quả giữa giả thiết "hôm nay trời nắng" và kết luận "chúng tôi đi chơi" nên mệnh đề là đúng trừ trường hợp hôm nay trời nắng nhưng chúng tôi không đi chơi và trường hợp hôm nay trời không nắng nhưng chúng tôi có đi chơi. Còn nếu hiểu theo kéo theo trong logic thì mệnh đề đó đúng cả khi hôm nay trời không nắng nhưng chúng tôi có đi chơi hoặc không đi chơi.

Thí dụ 2: Phép kéo theo: "Nếu hôm nay trời nắng thì 2 + 3 = 5" là luôn luôn đúng vì mệnh đề "2 + 3 = 5" luôn luôn đúng (giá trị chân lý của "Hôm nay trời nắng" là không quan trọng).

Thí dụ 3: Phép kéo theo: "Nếu hôm nay là thứ sáu thì 2 + 3 = 6" là đúng trong mọi ngày, trừ ngày thứ sáu.

Giá trị chân lý của các phép toán mệnh đề được tổng kết trong bảng 1

Bảng 1. Bảng giá trị chân lý của các phép toán mệnh đề

p	q	$p \vee q$	$p \wedge q$	$p \oplus q$	$\overline{p}$	$p \to q$	$p \leftrightarrow q$
T	T	T	T	F	F	T	T
T	F	T	F	T	F	F	F
F	T	T	F	T	T	T	F
F	F	F	F	F	T	T	T

1.3. Dịch các câu thông thường

Có thể tạo ra các mệnh đề phức hợp bằng cách sử dụng các toán tử đã định nghĩa ở trên. Các dấu ngoặc sẽ được dùng để chỉ thứ tự thực hiện các toán tử. Tuy nhiên để đơn

giản người ta quy định toán tử phủ định không cần để trong dấu ngoặc, chẳng hạn $(p \vee q) \wedge (\bar{r})$ là hội của p∨q với $\bar{r}$ có thể viết là $(p \vee q) \wedge \bar{r}$.

Nói chung, các câu thông thường đều không rõ ràng. Dịch các câu thông thường ra các biểu thức logic là làm mất đi tính không rõ ràng đó. Muốn vậy, phải tách câu nói thông thường thành các mệnh đề sơ cấp (mệnh đề hoặc đúng, hoặc sai) dựa trên ý nghĩa hàm định của câu đó, sau đó dùng các toán tử logic để liên kết chúng lại.

Thí dụ: Xét câu: "Bạn không được lái xe máy nếu bạn chưa cao tới 1,5m trừ khi bạn đã đủ 18 tuổi". Để dịch câu này ta có thể đặt:

p: "Bạn được lái xe máy"

q: "Bạn cao dưới 1,5m"

r: "Bạn đủ 18 tuổi"

Khi đó câu đã cho có thể viết thành biểu thức logic: $q \wedge \bar{r} \rightarrow \bar{p}$

1.4. Mối liên hệ giữa các phép toán logic và các phép toán bit

Máy tính dùng các bit để biểu diễn thông tin (bit là viết tắt của từ tiếng Anh binary digit: số nhị phân). Một bit có hai trạng thái là 0 và 1 (có thể xem hai trạng thái này là các giá trị có thể có của bit).

Bit cũng được dùng để biểu diễn các giá trị chân lý: 1 để biểu diễn giá trị True, còn 0 để biểu diễn giá trị False.

Trong tin học còn có biến Boole (Boolean variable) cũng chỉ nhận có hai giá trị là True và False. Do đó bit cũng dùng để biểu diễn một biến Boole.

Các phép toán cơ bản về bit dùng trong máy tính tương ứng với các phép toán logic. Các phép toán bit gồm có: OR, AND, XOR tương ứng với các phép toán ∨, ∧, ⊕. Nếu thay False bằng 0 và True bằng 1 được bảng các kết quả của các phép toán bit như trong bảng 3.

Bảng 2. Các kết quả của các phép toán bit

OR	0	1
0	0	1
1	1	1

AND	0	1
0	0	0
1	0	1

XOR	0	1
0	0	1
1	1	0

Thông tin trong máy tính được biểu diễn bằng các xâu bit. Xâu bit là dãy các số 0 và 1. Độ dài của một xâu bit là số các số 0 và số 1 có trong xâu. Với hai xâu bit cùng độ dài có thể thực hiện các phép toán OR, AND, XOR cho hai xâu bit đó bằng cách thực hiện các phép toán này đối với từng bit tương ứng.

Thí dụ:

OR
01101 00101
11000 10110
11101 10111

AND
01101 00101
11000 10110
01000 00100

XOR
01101 00101
11000 10110
10101 10011

2. Công thức đồng nhất đúng và công thức đồng nhất bằng nhau trong lôgic mệnh đề

Trong các lập luận toán học thường phải thay một mệnh đề phức hợp này bằng một mệnh đề phức hợp khác có cùng giá trị chân lý. Vì thế các phương pháp tạo ra các mệnh đề có cùng giá trị chân lý là rất quan trọng trong các lập luận toán học.

2.1. Các khái niệm về công thức trong lôgic mệnh đề

Công thức trong lôgic mệnh đề được định nghĩa bằng đệ quy như sau:

Định nghĩa 1: Định nghĩa công thức.

- Mỗi mệnh đề sơ cấp p, q, r, …, và T, F là một công thức.
- Nếu A, B là hai công thức thì các ký hiệu $(A \vee B), (A \wedge B), (\overline{A}), (A \to B)$ cũng là các công thức.

Chú ý: Mỗi công thức chỉ chứa các ký hiệu p, q, r, T, F; các phép toán $\vee$, $\wedge$, $\overline{\ }$, $\to$ và các ký hiệu mở ngoặc "(" , ký hiệu đóng ngoặc ")". Bản thân $A \vee B$, $A \wedge B$, $\overline{A}$, $A \to B$ không phải là các công thức, tuy nhiên để cho gọn thường viết $A \vee B$ thay cho $(A \vee B)$. v.v… Khi không dùng dấu ngoặc thì thứ tự thực hiện các phép tính trong một công thức là $\overline{\ }$, $\to$, $\wedge$ và cuối cùng là $\vee$.

Định nghĩa 2: Định nghĩa công thức đồng nhất đúng và công thức đồng nhất sai

- Công thức A được gọi là đồng nhất đúng (còn gọi là hằng đúng), ký hiệu $\models A$, khi và chỉ khi A luôn luôn nhận giá trị đúng (True) với mọi giá trị có thể của các mệnh đề sơ cấp thành phần có trong A.
- Công thức A được gọi là đồng nhất sai (còn gọi là mâu thuẫn), khi và chỉ khi A luôn luôn nhận giá trị sai (False) với mọi giá trị có thể của các mệnh đề sơ cấp thành phần có trong A..
- Công thức A không phải là hằng đúng, cũng không phải là mâu thuẫn được gọi công thức thực hiện được. Nghĩa là tồn tại ít ra là một bộ giá trị của các mệnh đề sơ cấp có trong công thức để công thức đó nhận giá trị đúng.

Dễ thấy phủ định của công thức đồng nhất đúng là công thức đồng nhất sai và ngược lại.

Sau đây là một số công thức đồng nhất đúng thường gặp:

1. $\models A \to (B \to A)$
2. $\models (A \to (B \to C)) \to ((A \to B) \to (A \to C)$
3. $\models (A \wedge B) \to A$
4. $\models (A \wedge B) \to B$
5. $\models (A \to B) \to ((A \to C) \to (A \to (B \to C)))$
6. $\models A \to (A \vee B)$
7. $\models B \to (A \vee B)$
8. $\models (A \to C) \to ((B \to C) \to ((A \vee B) \to C))$
9. $\models (A \to B) \to (\overline{B} \to \overline{A})$

10. $\models A \to \overline{\overline{A}}$

11. $\models \overline{\overline{A}} \to A$

Định nghĩa 3: Định nghĩa công thức đồng nhất bằng nhau

Hai công thức A và B được gọi là đồng nhất bằng nhau, ký hiệu $A \equiv B$ hoặc $A \Leftrightarrow B$, khi và chỉ khi A, B nhận cùng giá trị chân lý đối với mọi bộ giá trị có thể của các mệnh đề sơ cấp có trong chúng.

Nói cách khác, $A \equiv B$ khi và chỉ khi $A \leftrightarrow B$ là hằng đúng

Hai công thức đồng nhất bằng nhau còn được gọi là hai công thức tương đương lôgic.

Các công thức đồng nhất bằng nhau thường gặp được cho trong bảng 3.

Bảng 3. Các công thức tương đương lôgic thường gặp

Số TT	Tên gọi	Công thức
1.	Quy tắc phủ định kép	$\overline{\overline{A}} \equiv A$
2.	Tính chất giao hoán	$A \vee B \equiv B \vee A$ $A \wedge B \equiv B \wedge A$
3.	Tính chất kết hợp	$(A \vee B) \vee C \equiv A \vee (B \vee C)$ $(A \wedge B) \wedge C \equiv A \wedge (B \wedge C)$
4.	Tính chất phân bố	$A \vee (B \wedge C) \equiv (A \vee B) \wedge (A \vee C)$ $A \wedge (B \vee C) \equiv (A \wedge B) \vee (A \wedge C)$
5.	Quy tắc Đơ-Moocgan	$\overline{A \vee B} \equiv \overline{A} \wedge \overline{B}$ $\overline{A \wedge B} \equiv \overline{A} \vee \overline{B}$
6.	Tính lũy đẳng	$A \vee A \equiv A$ $A \wedge A \equiv A$
7.	Tính đồng nhất	$A \wedge T \equiv A$ $A \vee F \equiv A$
8.	Tính nuốt	$A \vee T \equiv T$ $A \wedge F \equiv F$
9.	Một số tiện ích	$A \to B \equiv \overline{A} \vee B$ $\overline{A} \vee A \equiv T$ $\overline{A} \wedge A \equiv F$

Để chứng minh các công thức trên, có thể dùng phương pháp lập bảng giá trị chân lý của chúng.

Thí dụ 1: Chứng minh rằng: $\models A \to (B \to A)$ (Công thức 1 sau định nghĩa 2).

Ta lập bảng giá trị chân lý:

A	B	$B \to A$	$A \to (B \to A)$
T	T	T	T
T	F	T	T
F	T	F	T
F	F	T	T

Vậy $A \to (B \to A)$ là công thức đồng nhất đúng, bởi cột cuối nói lên điều đó.

Thí dụ 2: Chứng minh rằng $\overline{A \vee B} \equiv \overline{A} \wedge \overline{B}$ (Công thức 5, quy tắc Đơ-Moocgan sau định nghĩa 3).

Lập bảng giá trị chân lý của cả hai vế:

A	B	$A \wedge B$	$\overline{A \wedge B}$	$\overline{A}$	$\overline{B}$	$\overline{A} \vee \overline{B}$
T	T	T	F	F	F	F
T	F	F	T	F	T	T
F	T	F	T	T	F	T
F	F	F	T	T	T	T

Quan sát các giá trị chân lý ở hai cột $\overline{A \wedge B}$ và $\overline{A} \vee \overline{B}$ thấy chúng giống nhau. Vậy

$$\overline{A \wedge B} \equiv \overline{A} \vee \overline{B}$$

Chú ý rằng, nếu trong công thức có 3 thành phần thì bảng các giá trị chân lý sẽ có $2^3 = 8$ hàng (Tổng quát là bảng 2^n hàng đối với công thức có n thành phần). Chẳng hạn chứng minh tính chất kết hợp của phép hợp:

$$(A \vee B) \vee C \equiv A \vee (B \vee C)$$

Bảng giá trị chân lý của công thức là:

A	B	C	$A \vee B$	$(A \vee B) \vee C$	$B \vee C$	$A \vee (B \vee C)$
T	T	T	T	T	T	T
T	T	F	T	T	T	T
T	F	T	T	T	T	T
T	F	F	T	T	F	T
F	T	T	T	T	T	T
F	T	F	T	T	T	T
F	F	T	F	T	T	T
F	F	F	F	F	F	F

Một phương pháp chứng minh khác là dùng các công thức thường gặp được trình bày sau các định nghĩa 2 và định nghĩa 3 để biến đổi công thức đã cho.

Thí dụ 4: Chứng minh rằng: $\overline{A \vee (\overline{A} \wedge B)} \equiv \overline{A} \wedge \overline{B}$.

Ta có:

$$\begin{aligned}
\overline{A \vee (\overline{A} \wedge B)} &\equiv \overline{A} \wedge \overline{(\overline{A} \wedge B)} && \text{Theo quy tắc Đơ-Moocgan}\\
&\equiv \overline{A} \wedge \left(\overline{\overline{A}} \vee \overline{B}\right) && \text{Theo quy tắc Đơ-Moocgan}\\
&\equiv \overline{A} \wedge (A \vee \overline{B}) && \text{Theo quy tắc phủ định kép}\\
&\equiv (\overline{A} \wedge A) \vee (\overline{A} \wedge B) && \text{Theo tính phân phối}\\
&\equiv F \vee (\overline{A} \wedge \overline{B}) && \text{Vì } \overline{A} \wedge A \equiv F\\
&\equiv (\overline{A} \wedge \overline{B}) \vee F && \text{Theo tính giao hoán}\\
&\equiv (\overline{A} \wedge \overline{B}) && \text{Theo tính đồng nhất}
\end{aligned}$$

Thí dụ 5: Chứng minh rằng: $\models (A \wedge B) \rightarrow (A \vee B)$.

Trước hết dùng bảng giá trị chân lý chứng minh $(A \rightarrow B) \equiv (\overline{A} \vee B)$.

A	B	$\overline{A}$	$\overline{A} \vee B$	$A \rightarrow B$
T	T	F	T	T
T	F	F	F	F
F	T	T	T	T
F	F	T	T	T

Từ đó:

$$\begin{aligned}
(A \wedge B) \rightarrow (A \vee B) &\equiv \overline{(A \wedge B)} \vee (A \vee B)\\
&\equiv (\overline{A} \vee \overline{B}) \vee (A \vee B) && \text{Theo quy tắc Đơ-Moocgan}\\
&\equiv (\overline{A} \vee A) \vee (\overline{B} \vee B) && \text{Theo tính giao hoán của phép tuyển}\\
&\equiv T \vee T \equiv T
\end{aligned}$$

2.2. Luật đối ngẫu.

Giả sử A là một công thức chỉ chứa các phép toán tuyển, hội và phủ định mà không chứa các phép toán $\oplus$, $\rightarrow$. Nếu trong A chúng ta đổi mỗi $\vee$ thành $\wedge$, mỗi $\wedge$ đổi thành $\vee$, mỗi T thành F và mỗi F thành T thì được một công thức mới, ký hiệu là A*. Công thức A* gọi là công thức đối ngẫu của công thức A.

Thí dụ: Đối ngẫu của $A \equiv (p \vee q) \wedge \overline{r}$ là $A^* = (p \wedge q) \vee \overline{r}$.

Đối ngẫu của $B \equiv (p \wedge T) \vee (r \wedge \overline{q}) \vee F$ là $B^* = (p \vee F) \wedge (r \vee \overline{q}) \wedge T$.

Định lý 1: Giả sử $A \equiv A(p_1, p_2, \ldots, p_n)$ là một công thức, trong đó p_i (i = 1, 2, …,n) là các mệnh đề sơ cấp có trong A. Khi đó luôn luôn có:

$$A^* \equiv \overline{A}(\overline{p_1}, \overline{p_2}, \ldots, \overline{p_n})$$

Chứng minh: Sử dụng định nghĩa đệ quy về công thức trong logic mệnh đề.

Để cho gọn, chúng ta viết A(X) thay cho $A(p_1, p_2, \ldots, p_n)$

và $\overline{A}(\overline{X})$ thay cho $\overline{A}(\overline{p_1}, \overline{p_2}, \ldots, \overline{p_n})$

1- Nếu A = p, ở đây p là mệnh đề sơ cấp nên hiển nhiên ta có:

$$\overline{A}(\overline{X}) \equiv \overline{\overline{X}} \equiv X \quad \text{hay} \quad A^* \equiv \overline{A}(\overline{X})$$

2- Giả sử đã chứng minh được cho các công thức A và B, nghĩa là:

$$A^* \equiv \overline{A}(\overline{X}) \text{ và } B^* \equiv \overline{B}(\overline{X})$$

Chúng ta phải chứng minh định lý cũng đúng cho (A∨B), (A∧B) và ($\overline{A}$)

Xét công thức (A∨B), ta có:

$$(A \vee B)^* \equiv A^* \wedge B^* \equiv \overline{A}(\overline{X}) \wedge \overline{B}(\overline{X}) \equiv \overline{(A \vee B)}(\overline{X})$$

Xét công thức (A∧B), ta có:

$$(A \wedge B)^* \equiv A^* \vee B^* \equiv \overline{A}(\overline{X}) \vee \overline{B}(\overline{X}) \equiv \overline{(A \wedge B)}(\overline{X})$$

Xét công thức $\left(\overline{A}\right)$, ta có:

$$\left(\overline{A}\right)^* \equiv \overline{A^*} \equiv \overline{\overline{A}}(\overline{X})$$

Định lý được chứng minh.

Thí dụ: Cho $A \equiv (p \vee q) \wedge \overline{r}$

Theo định nghĩa ta có: $A^* = (p \wedge q) \vee \overline{r}$

Theo định lý thì:

$$\overline{A}(\overline{X}) \equiv \overline{(\overline{p} \vee \overline{q}) \wedge \overline{r}} \equiv \overline{(\overline{p} \vee \overline{q})} \vee \overline{\overline{r}} \equiv (\overline{\overline{p}} \wedge \overline{\overline{q}}) \vee \overline{r} \equiv (p \wedge q) \vee \overline{r} \equiv A^*$$

Nguyên lý đối ngẫu: Nếu $A \equiv B$ thì $A^* \equiv B^*$, trong đó A*, B* là công thức đối ngẫu tương ứng của các công thức A, B. (Tất cả các cặp công thức 2–8 trong bảng 3 đều thỏa mãn nguyên lý này).

2.3. Luật thay thế

Định lý 2: Giả sử A là công thức chứa mệnh đề sơ cấp p thì khi thay p bởi một công thức E nào đó được công thức mới ký hiệu là B. Khi đó: Nếu ⊨A thì ⊨B.

Định lý được suy ra từ định nghĩa công thức đồng nhất đúng.

2.4. Luật kết luận

Định lý 3: Nếu A và (A → B) là các công thức đồng nhất đúng thì B cũng là công thức đồng nhất đúng. (Nếu ⊨A và ⊨(A→B) thì ⊨B)

Chứng minh: Định lý được chứng minh bằng phản chứng.

Giả sử ⊨A và ⊨(A→B) nhưng B không đồng nhất đúng, khi đó có một mệnh đề sơ cấp p để B(p) là sai, nhưng do A→B là đồng nhất đúng nên A(p) là sai vậy A không thể là đồng nhất đúng, điều này trái với giả thiết ⊨A. Định lý được chứng minh.

3. Điều kiện đồng nhất đúng trong lôgic mệnh đề

Phương pháp dùng bảng giá trị chân lý để nhận biết một công thức trong lôgic mệnh đề là đồng nhất đúng (đồng nhất sai) sẽ gặp khó khăn khi trong công thức có nhiều mệnh đề thành phần. Phần này xét thuật toán để khắc phục tình trạng đó.

3.1. Tuyển sơ cấp và hội sơ cấp

Định nghĩa:

- Biểu thức lôgic là tuyển của các mệnh đề sơ cấp hoặc phủ định của nó được gọi là một tuyển sơ cấp (TSC).
- Biểu thức lôgic là hội của các mệnh đề sơ cấp hoặc phủ định của nó được gọi là một hội sơ cấp (HSC).

Thí dụ: $p \vee \bar{q} \vee r$ là một tuyển sơ cấp; $\bar{q} \wedge \bar{r} \wedge p$ là một hội sơ cấp.

Định lý: Định lý có hai phần:

- Điều kiện cần và đủ để một tuyển sơ cấp là đồng nhất đúng là trong tuyển sơ cấp đó có chứa đồng thời một mệnh đề sơ cấp cùng với phủ định của nó.
- Điều kiện cần và đủ để một hội sơ cấp là đồng nhất sai là trong hội sơ cấp đó có chứa đồng thời một mệnh đề sơ cấp cùng với phủ định của nó.

Chứng minh: Chứng minh phần thứ nhất của định lý.

Điều kiện cần: Giả sử TSC là đồng nhất đúng, phải chỉ ra rằng tuyển sơ cấp đó chứa một mệnh đề sơ cấp cùng với phủ định của nó.

Giả sử ngược lại trong TSC đồng nhất đúng đó không có mệnh đề sơ cấp cùng với phủ định của nó. Khi đó nếu cho các mệnh đề sơ cấp không có dấu phủ định có trong TSC nhận giá trị F, còn các mệnh đề sơ cấp có dấu phủ định nhận giá trị T thì TSC nhận giá trị F. Điều này trái với giả thiết đồng nhất đúng của TSC. Vậy TSC đồng nhất đúng phải chứa một mệnh đề sơ cấp cùng với phủ định của nó.

Điều kiện đủ: Giả sử một TSC có chứa một mệnh đề sơ cấp cùng với phủ định của nó, chẳng hạn:

$$TSC = p \vee \bar{p} \vee \ldots \vee r$$

Khi đó vì $p \vee \bar{p}$ là đồng nhất đúng nên TSC là đồng nhất đúng.

Phần hai của định lý được chứng minh tương tự.

3.2. Dạng tuyển chuẩn tắc và dạng hội chuẩn tắc

Định nghĩa: Giả sử A là một công thức trong lôgic mệnh đề.

- Nếu $A \equiv A'$, trong đó A' là tuyển của các hội sơ cấp thì A' gọi là dạng tuyển chuẩn tắc (TCT) của A. Nghĩa là:
$$A \equiv A' \equiv (HSC)_1 \vee (HSC)_2 \vee \ldots \vee (HSC)_n .$$
- Nếu $A \equiv A'$, trong đó A' là hội của các tuyển sơ cấp thì A' gọi là dạng hội chuẩn tắc (HCT) của A. Nghĩa là:
$$A \equiv A' \equiv (TSC)_1 \wedge (TSC)_2 \wedge \ldots \vee (TSC)_n .$$

Thí dụ: Cho: $A \equiv p \wedge (q \rightarrow q)$, ta có:

$A' \equiv p \wedge (\bar{p} \vee q)$ là dạng HCT của A.

$A' \equiv (p \wedge \bar{p}) \vee (p \wedge q)$ là dạng TCT của A.

Định lý 1: Mọi công thức trong lôgic mệnh đề đều có dạng tuyển chuẩn tắc và dạng hội chuẩn tắc.

Chứng minh: Giả sử A là một công thức bất kỳ trong lôgic mệnh đề.

Nếu A có chứa phép toán $\rightarrow$ thì có thể khử phép toán $\rightarrow$ bằng công thức đồng nhất bằng nhau $p \rightarrow q \equiv \overline{p} \vee q$. Vì vậy có thể giả thiết A chỉ chứa các phép toán $\vee$, $\wedge$ và $\overline{\ }$.

Nếu phép phủ định còn chưa trực tiếp đối với các mệnh đề sơ cấp trong A thì sử dụng các công thức đồng nhất bằng nhau $\overline{p \vee q} \equiv \overline{p} \wedge \overline{q}$ hoặc $\overline{p \wedge q} \equiv \overline{p} \vee \overline{q}$ (Quy tắc Đơ-Moocgan).

Tiếp theo đưa A về dạng HCT và dạng TCT nhờ các công thức đồng nhất bằng nhau $p \vee (q \wedge r) \equiv (p \vee q) \wedge (p \vee r)$ hoặc $p \wedge (q \vee r) \equiv (p \wedge q) \vee (p \wedge r)$ (Tính phân bố).

Định lý được chứng minh.

Thí dụ: Tìm dạng TCT và dạng HCT của công thức $A \equiv p \rightarrow (q \rightarrow r)$

Ta có: $p \rightarrow (q \rightarrow r) \equiv \overline{p} \vee (q \rightarrow r) \equiv \overline{p} \vee \overline{q} \vee r$

Công thức $A' \equiv \overline{p} \vee \overline{q} \vee r$ là dạng HCT đồng thời cũng là TCT của các HSC $\overline{p}$, $\overline{q}$ và r.

Để thuận tiện trong việc chuyển một công thức về dạng tuyển chuẩn tắc hoặc hội chuẩn tắc có thể sử dụng định lý sau gọi là định lý khai triển:

Định lý 2: Định lý có hai phần:

- Điều kiện cần và đủ để công thức A đồng nhất đúng là mọi tuyển sơ cấp trong dạng HCT của A đều chứa một mệnh đề sơ cấp cùng với phủ định của nó,
- Điều kiện cần và đủ để công thức A đồng nhất sai là mọi hội sơ cấp trong dạng TCT của A đều chứa một mệnh đề sơ cấp cùng với phủ định của nó.

Chứng minh: Chứng minh phần thứ nhất của định lý.

Điều kiện cần: Giả sử A là công thức đồng nhất đúng. Theo định lý 1 thì A có dạng hội chuẩn tắc:

$$A' \equiv (TSC)_1 \wedge (TSC)_2 \wedge \ldots \wedge (TSC)_n .$$

Vì A đồng nhất đúng nên $(TSC)_i$, i = 1, 2, …, n là đồng nhất đúng. Theo định lý trong 3.1 thì trong mỗi tuyển sơ cấp $(TSC)_i$, i = 1, 2, …, n có chứa một mệnh đề sơ cấp cùng với phủ định của nó.

Điều kiện đủ: Giả sử $A' \equiv (TSC)_1 \wedge (TSC)_2 \wedge \ldots \wedge (TSC)_n$ là dạng HCT của A, trong đó mỗi $(TSC)_i$, i = 1, 2, …, n có chứa một mệnh đề sơ cấp cùng phủ định của nó. Theo định lý trong 3.1 thì mỗi tuyển sơ cấp $(TSC)_i$, i = 1, 2, …, n là đồng nhất đúng do đó A là công thức đồng nhất đúng.

Phần hai của định lý được chứng minh tương tự.

3.3. Thuật toán nhận biết một công thức là đồng nhất đúng hay đồng nhất sai

Từ các định lý trên suy ra thuật toán nhận biết công thức A là đồng nhất đúng hay đồng nhất sai như sau:

Bước 1: Tìm dạng hội chuẩn tắc A' và dạng tuyển chuẩn tắc A" của A.

Bước 2: Kết luận:

- Nếu mỗi tuyển sơ cấp trong A' đều chứa một mệnh đề sơ cấp cùng phủ định của nó thì A là đồng nhất đúng.

- Nếu mỗi hội sơ cấp trong A" đều chứa một mệnh đề sơ cấp cùng phủ định của nó thì A là đồng nhất sai.

- Nếu có một tuyển sơ cấp trong A' (hoặc một hội sơ cấp trong A") không có một mệnh đề sơ cấp nào cùng phủ định của nó thì A không phải là công thức đồng nhất đúng cũng không phải là công thức đồng nhất sai.

Thí dụ: Chứng minh rằng: $\models (A \wedge B) \rightarrow A$

Ta có: $A \wedge B \rightarrow A \equiv \overline{(A \wedge B)} \vee A \equiv (\overline{A} \vee \overline{B}) \vee A \equiv \overline{A} \vee A \vee \overline{B}$

Vậy $(A \wedge B) \rightarrow A$ là công thức đồng nhất đúng vì tuyển sơ cấp trong dạng hội chuẩn tắc của nó chứa A và $\overline{A}$

4. logic vị từ

4.1. Vị từ

Trong toán học cũng như trong các chương trình máy tính, chúng ta thường gặp các câu như: "x > 3", "x + y = 3", "x – y = z", … Các câu này không đúng cũng không sai chừng nào các biến x, y, z còn chưa được cho các giá trị cụ thể.

Phần này trình bày cách tạo ra các mệnh đề từ các câu như đã nêu.

Câu "x > 3" có hai bộ phận: bộ phận thứ nhất là biến x đóng vai trò chủ ngữ trong câu; bộ phận thứ hai "lớn hơn 3" đóng vai trò vị ngữ của câu, nó cho biết tính chất mà chủ ngữ có thể có. Có thể ký hiệu câu "x lớn hơn 3" là P(x) với P là ký hiệu vị ngữ "lớn hơn 3" và x là biến. Người ta cũng gọi P(x) là giá trị của hàm mệnh đề P tại x. Xét trong tập hợp các số thực, một khi biến x được gán một giá trị cụ thể thì câu P(x) sẽ có giá trị chân lý. Chẳng hạn P(4) là đúng còn P(2,5) là sai. Chú ý rằng hàm mệnh đề P(x) cũng có thể xét trong tập hợp các số nguyên hoặc tập hợp các số tự nhiên, …Bởi vậy có định nghĩa:

Định nghĩa 1: Giả sử $\mathcal{M}$ là một tập hợp các phần tử nào đó. Thành lập trên $\mathcal{M}$ các mệnh đề P(x), trong đó x nhận các giá trị trong $\mathcal{M}$ và P(x) nhận giá trị trong tập hợp {True, False} thì P(x) được gọi là một hàm mệnh đề xác định trên $\mathcal{M}$ và $\mathcal{M}$ gọi là không gian của hàm mệmh đề P.

Hàm mệnh đề P(x) còn được gọi là vị từ một ngôi xác định trên không gian $\mathcal{M}$.

Xét câu lệnh "if x < 0 then x := x + 1" thường gặp trong các chương trình máy tính. Khi thực hiện câu lệnh này, giá trị của biến x tại thời điểm nào đó được đặt vào câu P(x) = "x < 0". Nếu P(x) là True đối với giá trị này thì lệnh gán x := x + 1 được thực hiện và x tăng thêm 1. Nếu P(x) là False đối với giá trị này thì lệnh gán x := x + 1 không được thực hiện và giá trị của x không đổi.

Vấn đề đặt ra hoàn toàn tương tự với các câu nhiều biến hơn, chẳng hạn Q(x,y) là ký hiệu câu "x + y = 3", và R(x,y,z) là câu "x – y = z", ở đây x, y, z là các biến.

Định nghĩa 2: Cho $\mathcal{M}^n$ là tích Đề-các của n tập hợp $\mathcal{M}_i$ với i = 1, 2, …, n. Thành lập trên $\mathcal{M}^n$ các mệnh đề $P(x_1, x_2, \ldots, x_n)$, trong đó $x_i \in \mathcal{M}_i$ và $P(x_1, x_2, \ldots, x_n) \in$ {True, False} thì $P(x_1, x_2, \ldots, x_n)$ được gọi là một hàm mệnh đề xác định trên $\mathcal{M}^n$ và $\mathcal{M}^n$ gọi là không gian của hàm mệnh đề P.

Hàm mệnh đề $P(x_1, x_2, \ldots, x_n)$ còn được gọi là vị từ n ngôi xác định trên không gian $\mathcal{M}^n$.

Việc nghiên cứu các vị từ được gọi là logic vị từ. Chú ý rằng, cũng như trong lôgic mệnh đề, trong logic vị từ cũng chỉ quan tâm đến các giá trị của vị từ hoặc đúng, hoặc sai mà không xét đến các giá trị không đúng cũng không sai.

4.2. Lượng từ

Trong một vị từ có thể xảy ra các điều sau: vị từ đã cho đúng với mọi phần tử trong không gian xác định của nó; cũng có thể vị từ đó chỉ đúng với một số phần tử nào đó trong không gian xác định của nó. Người ta gọi đó là sự lượng hóa hay lượng từ các hàm mệnh đề.

Định nghĩa: Giả sử P(x) là một vị từ xác định trong không gian $\mathcal{M}$.

1/ Biểu thức $\forall x P(x)$ được gọi là lượng từ với mọi x của P(x); $\forall x P(x)$ là mệnh đề "P(x) đúng với mọi phần tử x trong không gian $\mathcal{M}$ ".

2/ Biểu thức $\exists x P(x)$ được gọi là lượng từ tồn tại x của P(x); $\exists x P(x)$ là mệnh đề "Có phần tử x trong không gian $\mathcal{M}$ để P(x) đúng".

Ký hiệu $\forall$ đọc là "với mọi" và ký hiệu $\exists$ đọc là "tồn tại" hay "có".

Giá trị chân lý của các lượng từ "với mọi" và "tồn tại" được cho trong bảng 4.

Bảng 4. Ý nghĩa của lượng từ "với mọi" và lượng từ "tồn tai"

Mệnh đề	Khi nào đúng (Nhận giá trị True)	Khi nào sai (Nhận giá trị False)
$\forall x P(x)$	P(x) là đúng với mọi phần tử x	Có ít nhất 1 phần tử x để P(x) là sai
$\exists x P(x)$	Có ít nhất 1 phần tử x để P(x) đúng	P(x) là sai với mọi phần tử x

Thí dụ 1: Xét trong không gian các số thực, ta có:

1/ Cho P(x):= "x+1 > x", khi đó có thể viết: $\forall x P(x)$.

2/ Cho P(x):= "2x = x+1", khi đó có thể viết: $\exists x P(x)$.

Thí dụ 2: Cho vị từ P(x):= "x tự học ở nhà hơn 4 giờ mỗi ngày", ở đây không gian là tập hợp các sinh viên. Khi đó để diễn đạt mệnh đề "Có ít nhất một sinh viên đã tự học hơn 4 giờ mỗi ngày" chỉ cần viết $\exists x P(x)$.

Từ bảng 4 có thể thấy rằng để chứng minh $\forall x P(x)$ là sai chỉ cần chỉ ra một phần tử của không gian đang xét để khi thay vào thì P(x) là sai. Chẳng hạn trong không gian các số thực, mệnh đề $\forall x$"2x > x" là mệnh đề sai bởi vì chỉ cần lấy x = – 1 thay vào mệnh đề là đủ thấy sai.

Còn việc chứng minh $\exists x P(x)$ là sai sẽ khó khăn hơn rất nhiều vì phải chỉ ra P(x) là đúng mọi phần tử của không gian.

Chú ý rằng nếu tất các phần tử của không gian $\mathcal{M}$ có thể liệt kê ra được, chẳng hạn $\mathcal{M} = \{x_1, x_2, \ldots, x_n\}$ thì ta có:

$$\forall x P(x) \equiv P(x_1) \wedge P(x_2) \wedge \ldots \wedge P(x_n)$$
$$\exists x P(x) \equiv P(x_1) \vee P(x_2) \vee \ldots \vee P(x_n)$$

Đối với các vị từ nhiều ngôi cũng các khái niệm tương tự. Chẳng hạn bảng 5 cho ý nghĩa các lượng từ đối với vị từ hai ngôi P(x,y) xác định trên $\mathcal{M}^2 = \mathcal{M}_1 \times \mathcal{M}_2$.

Bảng 5. Ý nghĩa của các lượng từ đối với vị từ hai ngôi

Mệnh đề	Khi nào đúng (Nhận giá trị True)	Khi nào sai (Nhận giá trị False)
$\forall x \forall y P(x,y)$ $\forall y \forall x P(x,y)$	P(x,y) đúng với mọi phần tử $(x,y) \in \mathcal{M}^2$	Có ít nhất 1 cặp phần tử $(x,y) \in \mathcal{M}^2$ sao cho P(x,y) là sai
$\forall x \exists y P(x,y)$	Với mọi phần tử $x \in \mathcal{M}_1$ có một $y \in \mathcal{M}_2$ sao cho P(x,y) là đúng	Có ít nhất 1 phần tử $x \in \mathcal{M}_1$ sao cho P(x,y) là sai với mọi $y \in \mathcal{M}_2$
$\exists x \forall y P(x,y)$	Có ít nhất 1 phần tử $x \in \mathcal{M}_1$ sao cho P(x,y) là đúng với mọi $y \in \mathcal{M}_2$	Với mọi phần tử $x \in \mathcal{M}_1$ có một $y \in \mathcal{M}_2$ sao cho P(x,y) là sai
$\exists x \exists y P(x,y)$ $\exists x \exists y P(x,y)$	Có 1 phần tử $(x,y) \in \mathcal{M}^2$ sao cho P(x,y) là đúng	P(x,y) là sai với mọi $(x,y) \in \mathcal{M}^2$

Thí dụ: Cho P(x,y) := "x + 2y = 2x – y" trong không gian là các số nguyên, khi đó ta có:

Mệnh đề P(3,1) có giá trị chân lý là True, còn P(1,0) có giá trị chân lý là False.

Mệnh đề $\forall x P(x,1)$ có giá trị chân lý là False.

Mệnh đề $\forall x \exists y P(x,y)$ là False vì, chẳng hạn x = 1 thì không có giá trị y nguyên nào thỏa mãn (dễ thấy $y = \frac{x}{3}$).

Mệnh đề $\forall y \exists x P(x,y)$ là True vì chỉ cần lấy x = 3y thì P(x,y) luôn thỏa mãn.

Chú ý rằng thứ tự các lượng từ trong vị từ nhiều ngôi là quan trọng. Nhiều khi đổi thứ tự các lượng từ dẫn đến việc một mệnh đề đang đúng trở thành sai hoặc ngược lại. Chẳng hạn, xét vị từ hai ngôi:

P(x,y) := "x + y = 0; x,y là các số thực"

Ta có: Mệnh đề $\forall x \exists y$ P(x,y) là mệnh đề đúng (cụ thể y = – x), nhưng:

Mệnh đề $\exists y \forall x P(x,y)$ là mệnh đề sai vì không có một giá trị nào của y để x+y=0 với mọi giá trị của x được.

Đối với vị từ nhiều ngôi hơn nữa, chẳng hạn vị từ ba ngôi P(x,y,z), vấn đề càng phức tạp hơn.

Thí dụ: Trong toán giải tích, định nghĩa giới hạn của hàm số:

$$\lim_{x \to a} f(x) = b$$

là: "Với mọi số thực $\varepsilon > 0$ đều tồn tại một số thực $\delta > 0$ sao cho $|f(x) - b| < \varepsilon$ khi $0 < |x - a| < \delta$".

Trong logic vị từ định nghĩa đó được diễn đạt như sau:

$$\forall \varepsilon \, \exists \delta \, \forall x \, (0 < |x - a| < \delta \rightarrow |f(x) - b| < \varepsilon)$$

ở đây không gian của ε và δ là các số thực dương, còn không gian của x là tập các số thực.

Cuối cùng ta xem xét vấn đề phủ định các lượng từ và vị từ. Điều này được tóm tắt trong bảng 6.

Bảng 6. Phủ định của vị từ và lượng từ

Phủ định	Cách viết tương đương	Khi nào đúng	Khi nào sai
$\overline{\exists}xP(x)$	$\forall x\overline{P(x)}$	P(x) sai với mọi x	Có một x để P(x) là đúng
$\overline{\forall}P(x)$	$\exists x\overline{P(x)}$	Có một x để P(x) là sai	P(x) đúng với mọi x

Thí dụ: Giả sử P(x):= "x đã thi đạt môn giải tích" với không gian là sinh viên của lớp thì để diễn tả mệnh đề "không phải mọi sinh viên của lớp đều đã thi đạt môn giải tích" có thể viết $\overline{\forall}xP(x)$ hoặc $\exists x\overline{P(x)}$.

4.3. Khái niệm công thức trong lôgic vị từ

Định nghĩa 1: Công thức trong logic vị từ được định nghĩa đệ quy như sau:

- Mỗi mệnh đề là một công thức. Mỗi vị từ là một công thức.
- Nếu A, B là công thức thì $(A\vee B)$, $(A\wedge B)$, $(A \to B)$, $(\overline{A})$ là các công thức.
- Nếu A là công thức thì biểu thức $\forall xA$, $\exists xA$ là các công thức.

Chú thích 1: Trong các công thức $\forall xA$ và $\exists xA$ thì công thức A gọi là miền tác dụng của lượng từ $\forall$ và $\exists$.

Chú thích 2: Nếu trong các công thức A của công thức $\forall xA$ hoặc $\exists xA$ chứa biến x và có thể có một số biến khác không nằm dưới dấu $\forall$, $\exists$ thì biến x gọi là biến ràng buộc còn các biến khác gọi là biến tự do hay biến độc lập.

Chú thích 3: Khác với công thức trong lôgic mệnh đề chỉ có một loại biến đó là biến mệnh đề. Trong công thức trong lôgic vị từ có thể có 2 loại biến khác nhau đó là:

- Mỗi mệnh đề là một biến và được gọi là biến mệnh đề.
- Mỗi vị từ là một biến và được gọi là biến vị từ.

Ngoài ra trong biến vị từ còn phải phân biệt biến ràng buộc và biến tự do.

Thí dụ 1: $\forall xA(x) \to F(x)$ là một công thức và nó cũng là nột vị từ mà miền tác dụng là toàn thể A(x), x có mặt trong A là biến ràng buộc. Còn x trong F là biến tự do, vì nó không bị ràng buộc bởi $\forall$.

Thí dụ 2: $A(x,y) \to \forall xB(x)$ là một công thức, B(x) là miền tác dụng của $\forall$, biến x trong B là biến ràng buộc, còn biến x trong A là biến tự do, biến y trong A cũng là biến tự do.

Qua các thí dụ trên thấy rằng một biến trong một công thức lôgic có thể vừa là biến tự do vừa là biến ràng buộc.

Định nghĩa 2: Định nghĩa công thức đồng nhất bằng nhau.

- Hai công thức A và B được gọi là đồng nhất bằng nhau trên không gian $\mathcal{M}$, ký hiệu $A \Leftrightarrow B$, hay $A \equiv B$ nếu chúng nhận giá trị đúng sai như nhau khi các biến vị từ được thay bởi các mệnh đề cụ thể trong không gian $\mathcal{M}$.

- Hai công thức A và B được gọi là đồng nhất bằng nhau nếu nó đồng nhất bằng nhau trên mọi không gian.

Định nghĩa 3: Định nghĩa công thức đồng nhất đúng.

- Công thức A gọi là đồng nhất đúng trên không gian $\mathcal{M}$ nếu nó luôn luôn nhận giá trị đúng khi các biến vị từ được thay bởi các mệnh đề cụ thể trong không gian $\mathcal{M}$.
- Công thức A gọi là đồng nhất đúng nếu nó đồng nhất đúng trên mọi không gian.

Chú ý rằng các công thức đồng nhất bằng nhau trong lôgic mệnh đề cũng là công thức đồng nhất bằng nhau trong lôgic vị từ. Chẳng hạn:

$$A \to B \equiv \overline{A} \vee B \qquad (1)$$

$$\overline{A \wedge B} \equiv \overline{A} \vee \overline{B} \qquad (2)$$

$$\overline{A \vee B} \equiv \overline{A} \wedge \overline{B} \qquad (3)$$

Ngoài ra trong lôgic vị từ còn các công thức đồng nhất bằng nhau sau:

$$\overline{\forall x\, A} \equiv \exists x\, \overline{A} \qquad (4)$$

$$\overline{\exists x\, A} \equiv \forall x\, \overline{A} \qquad (5)$$

Định nghĩa 4: Định nghĩa công thức sơ cấp và công thức rút gọn.

- Một công thức mà trong đó chỉ chứa các phép toán hội, tuyển và phủ định được gọi là công thức sơ cấp.
- Một công thức sơ cấp mà phép toán phủ định không thực hiện với các lượng từ $\forall$ và $\exists$ gọi là công thức rút gọn.

Có thể sử dụng các công thức đồng nhất bằng nhau (1) – (5) để đưa một công thức đã cho về công thức rút gọn.

Thí dụ:

Ta có:
$$\begin{aligned}\overline{\exists x\big(A(x) \to \forall y B(y)\big)} &\equiv \overline{\exists x\big(\overline{A(x)} \vee \forall y B(y)\big)} \\ &\equiv \forall x\big(\overline{\overline{A(x)}} \wedge \exists y \overline{B(y)}\big) \\ &\equiv \forall x\big(A(x) \wedge \exists y \overline{B(y)}\big)\end{aligned}$$

Vậy: Công thức $\forall x\big(A(x) \wedge \exists y \overline{B(y)}\big)$ là công thức rút gọn của công thức $\overline{\exists x\big(A(x) \to \forall y B(y)\big)}$

Khác với lôgíc mệnh đề, trong lôgic vị từ chưa có thuật toán chung để nhận biết một công thức là đồng nhất đúng hoặc đồng nhất sai. Bài toán chỉ được giải quyết trong một số lớp công thức của lôgic vị từ. Ở đây chúng ta không đề cập đến vấn đề này.

BÀI TẬP

8.1. Cho p và q là hai mệnh đề :

p: Tôi di xe máy với tốc độ trên 60 km/h.

q: Tôi bị phạt vì vượt quá tốc độ cho phép.

Hãy viết các mệnh đề sau bằng cách dùng p và q và các phép toán logic:

a) Tôi không đi xe máy với tốc độ trên 60 km/h.

b) Tôi sẽ bị phạt vì vượt quá tốc độ cho phép nếu tôi di xe máy với tốc độ trên 60 km/h.

c) Nếu tôi không đi xe máy với tốc độ trên 60 km/h thì tôi sẽ không bị phạt vì vượt quá tốc độ cho phép.

d) Tôi đi xe máy với tốc độ trên 60 km/h là đủ để bị phạt vì vượt quá tốc độ cho phép.

e) Mỗi lần tôi bị phạt vì vượt quá tốc độ cho phép là tôi đã đi xe máy với tốc độ trên 60 km/h.

8.2. Cho p, q, r là các mệnh đề:

p: Tôi được điểm giỏi trong kỳ thi hết môn.

q: Tôi làm hết các bài tập của môn học.

r: Tôi được công nhận là học sinh giỏi của lớp.

Hãy dùng p, q, r và các phép toán logic để viết các mệnh đề:

a) Tôi được công nhận là học sinh giỏi của lớp nhưng tôi không làm hết các bài tập của môn học.

b) Để được công nhận là học sinh giỏi của lớp tôi cần phải được điểm giỏi ở kỳ thi hết môn.

c) Nhận được điểm giỏi trong kỳ thi hết môn và làm hết các bài tập của môn học là đủ để được công nhận là học sinh giỏi của lớp.

d) Tôi được công nhận là học sinh giỏi của lớp, nếu và chỉ nếu tôi làm hết các bài tập của môn học hoặc nhận điểm giỏi ở kỳ thi hết môn.

8.3. Lập bảng giá trị chân lý cho các mệnh đề sau:

a) $p \wedge \bar{p}$ b) $(p \vee \bar{p}) \rightarrow q$ c) $p \oplus \bar{p}$

d) $(p \oplus q) \wedge (p \oplus \bar{p})$ e) $(p \vee q) \wedge r$ f) $p \rightarrow (q \rightarrow r)$

8.4. Tìm các OR bit, AND bit và XOR bit của các xâu bít sau:

a) 10 11110 và 01 00001 b) 111 10000 và 101 01010

8.5. Xác định các biểu thức sau:

a) $11000 \wedge (11011 \vee 11011)$ b) $(01010 \oplus 11011) \oplus 01000$

8.6. Để chứng minh các đẳng thức tập hợp ngoài cách chứng minh theo định nghĩa:

$$A = B \Leftrightarrow x \in A \Rightarrow x \in B \text{ và } x \in B \Rightarrow x \in A$$

có thể dùng bảng **tính thuộc** của một tập hợp, trong bảng tính thuộc để chỉ một phần tử thuộc một tập hợp ta dùng số 1 và không thuộc tập hợp ta dùng số 0 và sau đó dùng các phép toán bit để chứng minh.

Thí dụ: Để chứng minh quy tắc Đơ-Moocgan $\overline{A \cap B} = \overline{A} \cup \overline{B}$ với A, B là hai tập hợp đã cho, có thể lập bảng tính thuộc như sau:

A	B	A∩B	$\overline{A \cap B}$	$\overline{A}$	$\overline{B}$	$\overline{A} \cup \overline{B}$
1	1	1	0	0	0	0
1	0	0	1	0	1	1
0	1	0	1	1	0	1
0	0	0	1	1	1	1

Từ đó: $\overline{A \cap B} = \overline{A} \cup \overline{B}$

Dùng bảng tính thuộc chứng minh rằng:

a) $\overline{A \cup B \cup C} = \overline{A} \cap \overline{B} \cap \overline{C}$; b) $A \setminus B = A \cap B$

c) $(A \setminus C) \cap (C \setminus B) = \varnothing$; d) $(B \setminus A) \cup (C \setminus A) = (B \cup C) \setminus A$

8.7. Chứng minh các công thức sau là đồng nhất đúng bằng cách lập bảng giá trị chân lý:

a) $(p \to (q \to r)) \to ((p \to q) \to (p \to r))$;

b) $(p \to q) \to ((p \to r) \to (p \to ((q \wedge r)))$.

8.8. Dùng bảng chân lý chứng minh các tương đương lôgic (đồng nhất bằng nhau) được gọi là **luật hấp thu** sau:

a) $p \vee (p \wedge q) \Leftrightarrow p$; b) $p \wedge (p \vee q) \Leftrightarrow p$

8.9. Chứng minh rằng: a) $(p \leftrightarrow q) \Leftrightarrow (p \wedge q) \vee (\overline{p} \wedge \overline{q})$

b) $\overline{(p \oplus q)} \Leftrightarrow (p \leftrightarrow q)$

c) $p \oplus q \Leftrightarrow (p \vee \overline{q}) \wedge \overline{p \wedge q}$

8.10. Tìm đối ngẫu của các công thức sau:

a) $(p \wedge q \wedge r) \vee s$; b) $(p \vee T) \wedge (q \vee T)$

8.11. Tìm dạng hội chuẩn tắc của các công thức trong bài 7. Từ đó suy ra các công thức đã cho là đồng nhất đúng mà không cần lập bảng giá trị chân lý.

8.12. Tìm dạng tuyển chuẩn tắc của các công thức trong bài 7.

8.13. Bằng phương pháp quy nạp chứng minh rằng:

a) $p \vee (q_1 \wedge q_2 \wedge \ldots \wedge q_n) \Leftrightarrow (p \vee q_1) \wedge (p \vee q_2) \wedge \ldots \wedge (p \vee q_n)$

b) $\overline{(p_1 \wedge p_2 \wedge \ldots \wedge p_n)} \Leftrightarrow \overline{p_1} \vee \overline{p_2} \vee \ldots \vee \overline{p_n}$

8.14. Một tập hợp các phép toán lôgic được gọi là đầy đủ nếu mọi công thức đều tương đương lôgic với công thức chỉ chứa các phép toán đó.

a) Chứng minh rằng các phép toán $\vee$, $\wedge$ và $\overline{\ }$ lập thành một tập hợp đầy đủ của các phép toán lôgic.

b) Chứng minh rằng các phép toán $\vee$ và $\overline{\ }$ lập thành một tập hợp đầy đủ của các phép toán lôgic.

c) Chứng minh rằng các phép toán $\wedge$ và $\overline{\ }$ lập thành một tập hợp đầy đủ của các phép toán lôgic.

8.15. Ngoài các phép toán đã định nghĩa trong mục 1.2. người ta còn định nghĩa thêm hai phép toán lôgic NAND (not and) và NOR (not or) và được định nghĩa như sau:

p	p	p NAND q	p NOR q
T	T	F	F
T	F	T	F
F	T	T	F
F	F	T	T

Phép toán p NAND q được ký hiệu là $p \uparrow q$, còn p NOR q được ký hiệu $p \downarrow q$

Hãy chứng minh rằng:

a) $p \uparrow q \Leftrightarrow \overline{p \wedge q}$; b) $p \downarrow q \Leftrightarrow \overline{p \vee q}$

c) $p \downarrow p \Leftrightarrow \overline{p}$; d) $(p \downarrow q) \downarrow (p \downarrow q) \Leftrightarrow p \vee q$

8.16. Chứng minh rằng $p \uparrow (q \uparrow r)$ và $(p \uparrow q) \uparrow r$ là không tương đương lôgic (do đó phép toán NAND là không có tính kết hợp).

8.17. Cho vị từ hai ngôi P(x,y) là câu "x là thủ phủ của y" và vị từ một ngôi Q(x) là câu "Từ x chứa chữ cái a". Hãy xác định giá trị chân lý của các mệnh đề sau:

a) P(Hà nội, Việt nam); b) P(Băng cốc, Lào);

c) Q(Quýt); d) Q(Cam).

8.18. Hãy xác định giá trị chân lý các lượng từ sau:

a) Cho vị từ P(x): "x + 2 > x +1", hãy xác định giá trị chân lý của $\forall x P(x)$ trong không gian các số thực.

b) Cho vị từ Q(x,y): " x + y = 0", hãy xác định giá trị chân lý của $\exists y \forall x Q(x,y)$ và $\forall x \exists y Q(x,y)$ trong không gian các số thực.

c) Cho vị từ R(x,y,z): "x + y = z", hãy xác định giá trị chân lý của $\forall x \forall y \exists z\, R(x,y,z)$ và $\exists z \forall x \forall y\, R(x,y,z)$ trong không gian các số thực.

8.19. Chứng tỏ rằng các mệnh đề: $\exists x \forall y\, P(x,y)$ và $\forall x \exists y \overline{P(x,y)}$ có cùng giá trị chân lý

8.20. $\exists!x\, P(x)$ là ký hiệu của mệnh đề "tồn tại duy nhất một x sao cho P(x) là đúng".

a) Trong không gian là tập các số thực, hãy xác định giá trị chân lý của các lượng từ sau:

α) $\exists!x\ (x > 1)$; β) $\exists!x\ (x^2 = 1)$; γ) $\exists!x\ (x + 3 = 2x)$

b) Xác định giá trị chân lý của các mệnh đề sau:

α) $\exists!x\ P(x) \rightarrow \exists x\ P(x)$; β) $\forall x\ P(x) \rightarrow \exists!x\ P(x)$.

MỘT SỐ BÀI TẬP LÀM TRÊN MÁY TÍNH

Bằng ngôn ngữ lập trình Pascal hoặc ngôn ngữ lập trình tự chọn, hãy lập trình giải các bài toán sau.

Yêu cầu phân tich rõ các bước thực hiện trong quá trình lập trình.

Các bài toán lập trình về thuật toán và bài toán đếm:

Bằng thuật toán quay lui, lập trình giải các bài toán (kết quả in ra màn hình hoặc ghi thành tệp dạng văn bản):

1. Liệt kê tất cả dãy nhị phân $(x_1, x_2, \ldots, x_n)$, $x_i \in \{0,1\}$ thoả mãn phương trình:

$$a_1x_1 + a_2x_2 + \ldots + a_nx_n = b$$

Trong đó $a_1, a_2, \ldots, a_n$, b là các số nguyên dương cho trước được vào từ bàn phím.

2. Liệt kê tất cả các nghiệm nguyên không âm (nghĩa là các bộ giá trị $(x_1, x_2, \ldots, x_n)$, trong đó x_i nguyên, không âm) của phương trình:

$$a_1x_1 + a_2x_2 + \ldots + a_nx_n = b$$

Trong đó $a_1, a_2, \ldots, a_n$, b là các số nguyên dương cho trước được vào từ bàn phím.

3. Cho số nguyên dương N. Hãy liệt kê tất cả các cách biểu diễn N dưới dạng tổng của một số các số nguyên dương. N vào từ bàn phím.

4. Hình vuông thần bí bậc n (Ma phương bậc n) là ma trận vuông cấp n với các phần tử là các số tự nhiên từ 1 đến n^2 thoả mãn tính chất: *"Tổng các phần tử trên mỗi dòng, mỗi cột và mỗi đường chéo đều bằng nhau"*. Chẳng hạn sau đây là một ma phương bậc 3:

$$\begin{bmatrix} 8 & 1 & 6 \\ 3 & 5 & 7 \\ 4 & 9 & 2 \end{bmatrix}$$

Dễ thấy tổng mỗi hàng, mỗi cột và mỗi đường chéo của ma trận đều bằng 15.

Hãy lập trình liệt kê tất cả các ma phương bậc 3, bậc 4 không sai khác nhau bởi các phép biến đổi đơn giản như phép quay, phép đối xứng.

Chú ý: Hiện tại bài toán mới chỉ giải được với các bậc 3, 4, 5.

Các bài toán lập trình về đồ thị

Các bài toán sau được lập trình với:

Input: Đồ thị n đỉnh, m cạnh được cho dưới dạng hoặc danh sách kề, hoặc danh sách cạnh, hoặc ma trận kề từ bàn phím hoặc từ tệp văn bản. Cần nói rõ input dưới dạng nào.

Output: Kết quả có thể in ra màn hình hoặc ghi vào tệp dạng văn bản.

Bằng các thuật toán đã học, hãy lập trình giải các bài toán sau:

5. Chuyển đổi một đồ thị cho từ một trong ba dạng: danh sách kề, danh sách cạnh, ma trận kề sang các dạng còn lại. Sau đó tìm bậc của các đỉnh.

6. Kiểm tra tính liên thông của một đồ thị. Nếu đồ thị không liên thông, hãy liệt kê các thành phần liên thông của nó.

7. Cho hai đơn đồ thị, mỗi đồ thị có không quả 6 đỉnh. Hãy xác định xem hai đồ thị đó có đẳng cấu với nhau không.

8. Tô màu một đồ thị vô hướng, từ đó suy ra sắc số của đồ thị.

9. Tìm nhân của một đồ thị vô hướng.

10. Nhận biết một đồ thị là đồ thị Euler hoặc là nửa Euler. Liệt kê các chu trình (đường đi) Euler, nếu nó là đồ thị Euler (nửa Euler).

11. Nhận biết một đồ thị là đồ thị Hamilton hoặc là nửa Hamilton. Liệt kê các chu trình (đường đi) Hamilton, nếu nó là đồ thị Hamilton (nửa Hamilton).

12. Duyệt cây nhị phân.

13. Tìm cây khung nhỏ nhất của một đơn đồ thị liên thông theo thuật toán Kruskal.

14. Tìm cây khung nhỏ nhất của một đơn đồ thị liên thông theo thuật toán Prim.

15. Tìm đương đi ngắn nhất từ đỉnh x_i đến đỉnh x_j theo thuật toán Dijkstra.

16. Tìm luồng cực đại từ mạng đã cho theo thuật toán Ford-Fulkerson.

17. Giải bài toán du lịch theo thuật toán nhánh cận.

18. Giải bài toán du lịch bằng các so sánh tất cả các hành trình có thể bằng cách xuất phát từ đỉnh x_i tuỳ ý.

MỘT SỐ THUẬT NGỮ DÙNG TRONG GIÁO TRÌNH

Thuật ngữ tiếng Việt	Tiếng Anh tương ứng	Trang
Bậc (của đỉnh của đồ thị)	Degree	61
Bán bậc ra	Out-degree	61
Bán bậc vào	In-degree	61
Bán kính (của đồ thị)	Radius	129
Cạnh (của đồ thị)	Edge	58
Cạnh bội	Paralleges	60
Cạnh treo	Pendant edge	62
Cây	Tree	100
Cây có gốc	Rooted tree	101
Cây khung nhỏ nhất	Minimal Spanning Tree (MST)	115
Cây khung/ Cây bao trùm	Spanning tree	112
Cây m phân đầy đủ	Complete m-ary tree	103
Cây m-phân	m-ary tree	103
Cây nhị phân	Binary tree	104
Chiều cao (của cây có gốc)	Height	102
Chỉnh hợp	Arrangement	29
Chỉnh hợp lặp	Arrangement with repetition	32
Chu trình	Cycle/ Circuit	70
Chu trình Euler	Eulerian cycle	83
Chu trình Hamilton	Hamiltonian cycle	87
Đa đồ thị	Multi graph	60
Đẳng cấu (đồ thị)	Isomorphie	69
Danh sách cạnh	Incidence list	66
Danh sách kề	Adjacency list	66
Đệ quy	Recursion	17
Đỉnh (của đồ thị)	Vertex	58
Đỉnh cắt	Cut vertex/ Cut point	73
Đỉnh cô lập	Isolated vertex	62
Đỉnh cuối	Initial vertex	59
Đỉnh đầu	Terminal vertex	59
Đỉnh phát	Source vertex	130
Đỉnh thu	Sink vertex	130
Đỉnh treo	Pendant vertex	62
Đỉnh trong (của cây có gốc)	Internal vertex	103

Độ phức tạp của thuật toán	Computational complexity theory	12
Độ phức tạp đa thức	Polynominal complexity	14
Độ phức tạp giai thừa	Factorial complexity	14
Độ phức tạp hằng số	Constant complexity	14
Độ phức tạp logarit	Logarithmic complexity	14
Độ phức tạp nlogarit	Linearithmic complexity	14
Độ phức tạp mũ	exponential complexity	14
Độ phức tạp tuyến tính	Linear complexity	14
Đồ thị bánh xe	Wheel graph	63
Đồ thị có hướng	Directed graph	58
Đồ thị có trọng số	Weighted graph	125
Đồ thị con	Subgraph	72
Đồ thị đầy đủ	Complete graph	63
Đồ thị hữu hạn	Finite graph	58
Đồ thị lập phương	Cube graph	63
Đồ thị phân đôi	Bipartite graph	64
Đồ thị phân đôi đầy đủ	Complete bipartite graph	64
Đồ thị phẳng	Planar graph	91
Đồ thị vô hạn	Infinite graph	58
Đồ thị vô hướng	Undirected graph	58
Đồ thị vòng	Circular graph	63
Đơn đồ thị	Simple graph	60
Đồng phôi	Heneomorphie	94
Đường đi (trong đồ thị)	Path	70
Đường đi đơn	Simple path	70
Đường đi Euler	Eulerian path	83
Đường đi Hamilton	Hamiltonian path	87
Duyệt cây	Tree searching	104
Hoán vị	Permutation	30
Khuyên	Loop	59
Ký pháp nghịch đảo Ba lan (RPN)	Reverse Polish Notation	106
Lá (của cây có gốc)	Leaved	103
Liên thông (đồ thị)	Connected	72
Liên thông mạnh	Strongly connected	74
Liên thông yếu	Weakly connected	74
Luồng	Flow function	130

Luồng cực đại	Max Flow funtion	131
Lý thuyết đồ thị	Graph theory	58
Ma trận kề	Adjacency matrix	67
Ma trận liên thuộc	Incidence matrix	68
Mạng	Network	130
Miền (trong đồ thị phẳng)	Face/Region	92
Mức (trong cây có gốc)	Level	102
Sắc số (của đồ thị)	Chromatic number	76
Tâm (của đồ thị)	Center	129
Thành phần liên thông	Connected component	72
Thuật toán	Algorithm	6
Thuật toán nhánh cận	Branch and Bound algorithm	136
Thuật toán quay lui	Backtracking algorithm	46
Thuật toán tìm kiếm	Searaching algorithm	15
Tìm kiếm nhị phân	Binary search	16
Tìm kiếm tuyến tính	Linear search	15
Tìm kiếm ưu tiên chiều rộng	Breadth-First Search (BFS)	113
Tìm kiếm ưu tiên chiều sâu	Depth-First Search (DFS)	112
Tổ hợp	Combination	30
Tổ hợp lặp	Combination with repetition	33

TÀI LIỆU THAM KHẢO

1. Chu Đức Khánh. *Lý thuyết đồ thị*. Nhà xuất bản Đại học Quốc gia Thành phố Hồ Chí Minh - 2002.
2. Đại học Huế. *Giáo trình Toán Rời rạc*. Tài liệu trên mạng Internet.
3. Đặng Huy Ruận. *Lý thuyết đồ thị và ứng dụng*. Nhà xuất bản Khoa học và Kỹ thuật. Hà Nội - 2000.
4. Đỗ Đức Giáo. *Toán Rời rạc*. Nhà xuất bản Đại học Quốc gia Hà Nội - 2000
5. Kenneth H.Rosen. *Toán học Rời rạc ứng dụng trong tin học*. Bản dịch từ tiếng Anh. Nhà xuất bản Khoa học và Kỹ thuật. Hà Nội - 2000.
6. P.X. Novikop. *Đại cương về Lôgic toán*. Bản dịch từ tiếng Nga của Nguyễn Hữu Ngự – Đặng Huy Ruận. Nhà xuất bản Khoa học và Kỹ thuật. Hà Nội - 1971.
7. Nguyễn Đức Nghĩa – Nguyễn Tô Thành. *Toán Rời rạc*. Nhà xuất bản Đại học Quốc gia Hà Nội - 2003.