MỤC LỤC

LỜI NÓI ĐẦU

Toán rời rạc là lĩnh vực nghiên cứu các đối tượng rời rạc để giải quyết các bài toán đếm các đối tượng, liệt kê các đối tượng rời rạc, nghiên cứu các mối quan hệ giữa các tập hợp, phân tích các quá trình hữu hạn. Việc cất giữ, lưu trữ và xử lý thông tin trên máy tính bản chất là quá trình rời rạc.

Boole là một trong những nhà khoa học tiên phong nghiên cứu cơ chế biểu diễn quá trình tư duy lôgic. Năm 1854 ông viết cuốn *Các qui luật tư duy*. Đóng góp lớn nhất của Boole là phát triển lý thuyết lôgic bằng *ký hiệu* thay cho từ ngữ và có thể sử dụng đại số Boole, hàm Boole để thiết kế mạch tổ hợp từ đó nghiên cứu mạch điện.

Đồ thị là một cấu trúc rời rạc gồm các *đỉnh* và các *cung* nối các đỉnh đó. Đây là công cụ hữu hiệu để mô hình hoá và giải quyết các bài toán trong nhiều lĩnh vực khoa học, kỹ thuật, kinh tế, xã hội, ... Chẳng hạn, đồ thị có thể sử dụng để xác định các mạch vòng trong các vấn đề giải tích mạch điện. Chúng ta có thể phân biệt các hợp chất hoá học hữu cơ khác nhau với cùng công thức phân tử nhưng khác nhau cấu trúc phân tử nhờ đồ thị. Chúng ta có thể xác định xem hai máy tính trong mạng có thể trao đổi thông tin được với nhau hay không nhờ mô hình đồ thị của mạng máy tính. Đồ thị có trọng số trên các cạnh có thể sử dụng để giải các bài toán như tìm đường đi ngắn nhất giữa hai thành phố trong mạng giao thông. Ngoài ra, chúng ta có thể xử dụng đồ thị để giải các bài toán về lập lịch, thời khoá biểu, và phân bố tầng số cho các trạm phát thanh và truyền hình. Những tư tưởng cơ bản của lý thuyết đồ thị được đề xuất vào những năm đầu của thế kỷ 18 bởi nhà toán học lỗi lạc người Thuỵ Sỹ Leonhard Euler. Chính ông là người đã xử dụng đồ thị để giải bài toán nổi tiếng về các cái cầu ở thành phố Konigsberg.

Giáo trình này nhằm giới thiệu kiến thức cơ bản trong ba lĩnh vực có nhiều ứng dụng của toán rời rạc. Lĩnh vực thức nhất tập trung nghiên cứu về thuật toán, độ phức tạp của thuật toán, công thức truy hồi, quy nạp toán học, lý thuyết tổ hợp, các nguyên lý trong tổ hợp, các bài toán đếm, các bài toán tồn tại và các bài toán liệt kê. Lĩnh vực thứ hai tập trung nghiên cứu về hàm đại số logic, đại số Boole, biểu diễn hàm Boole, xây dựng các mạch tổ hợp cũng như cực tiểu hoá các mạch tổ hợp để thiết kế được các mạch tổ hợp. Lĩnh vực thứ ba là lý thuyết đồ thị tập trung nghiên cứu về các khái niệm đồ thị, biểu diễn đồ thị, đồ thị Hamilton, đồ thị Euler, các bài toán về đường đi và các bài toán về cây phủ…

Chúng tôi xin chân thành cám ơn các đồng nghiệp, các nhà khoa học đã động viên và góp ý cho giáo trình Toán rời rạc này và lời cám ơn đặc biệt xin dành cho Trường Đại học Sư phạm, Khoa Tin học về sự giúp đỡ quý báu và tạo điều kiện thuận lợi cho việc xuất bản giáo trình này. Nhóm tác giả mong tiếp tục nhận được sự góp ý của các đồng nghiệp và độc giả về những thiếu sót khó tránh khỏi của cuốn giáo trình.

Tác giả

Nguyễn Đình Lầu

DANH MỤC HÌNH

DANH MỤC BẢNG

CHƯƠNG 1. CÁC KIẾN THỨC CƠ SỞ

Tóm tắt chương

Chương này, sẽ cung cấp các kiến thức cơ bản về thuật toán và đánh giá độ phức tạp thuật toán cũng như cách chứng minh bài toán bằng nguyên lý quy nạp và lập trình các bài toán bằng đệ quy để vận dụng vào các chương tiếp theo.

1.1 Thuật toán

1.1.1 Khái niệm

Thuật toán là tập hợp hữu hạn các thao tác dẫn đến lời giải cho một vấn đề hay bài toán nào đó trong thời gian hữu hạn.

➢ Các phương pháp biểu diễn thuật toán

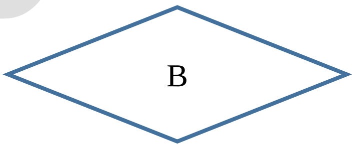
- Ngôn ngữ tự nhiên (liệt kê các bước)
- Ngôn ngữ lưu đồ (sơ đồ khối)
- Ngôn ngữ phỏng trình (mã giả): Khi biểu diễn thuật toán bằng mã giả chúng ta xây dựng dựng thuật toán gần với ngôn ngữ lập trình, từ đó tuỳ vào ngôn ngữ lập trình cụ thể mà viết thành chương trình hoàn thiện.

➢ Các nút chức năng của lưu đồ

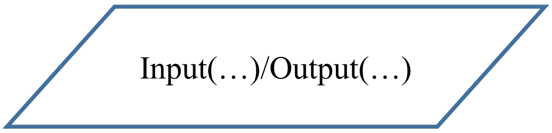
- Nút giới hạn:

- Nút thao tác:

- Nút điều kiện:



- Nút xuất/nhập dữ liệu:



- Đường đi của thuật toán: →

Ví dụ 1. Giải phương trình bậc 2. Cho phương trình $ax^2 + bx + c = 0$ ($a <> 0$), tìm x.

- Thuật toán được biểu diễn bằng cách liệt kê các bước:

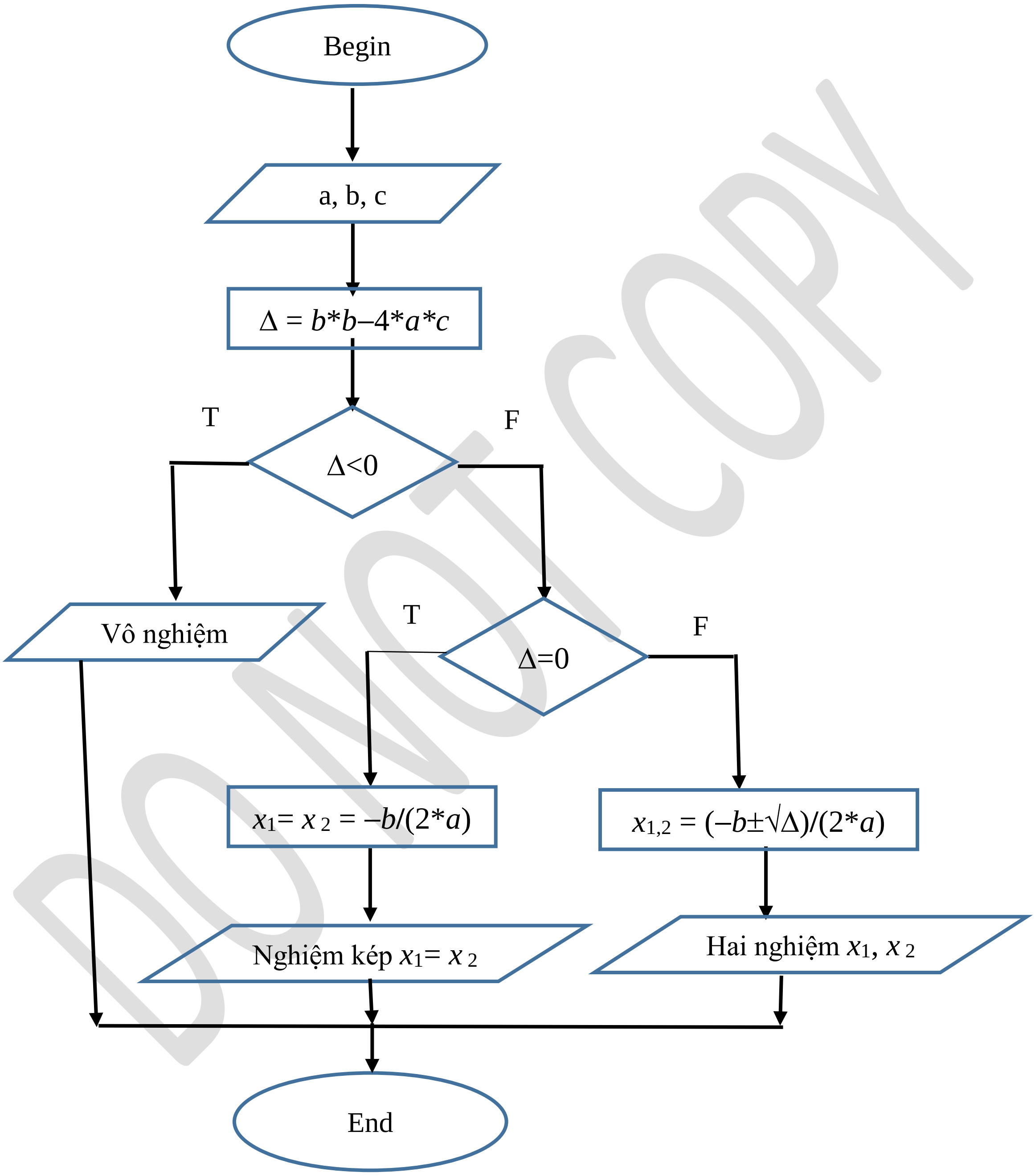
Bước 1. Nhập a, b, c và tính $\Delta = b*b-4*a*c$

Bước 2. Nếu $\Delta < 0$, thì kết luận: phương trình vô nghiệm. Kết thúc.

Ngược lại, Nếu $\Delta > 0$, thì nghiệm $x_{1,2} = (-b \pm \sqrt{\Delta})/(2*a)$. Kết thúc

Ngược lại, Tức $\Delta = 0$, thì nghiệm kép $x_1 = x_2 = -b/(2*a)$. Kết thúc.

- Thuật toán được biểu diễn bằng sơ đồ khối



Hình 1.1 Sơ đồ khối của thuật toán giải phương trình bậc 2

- Thuật toán được biểu diễn bằng mã giả

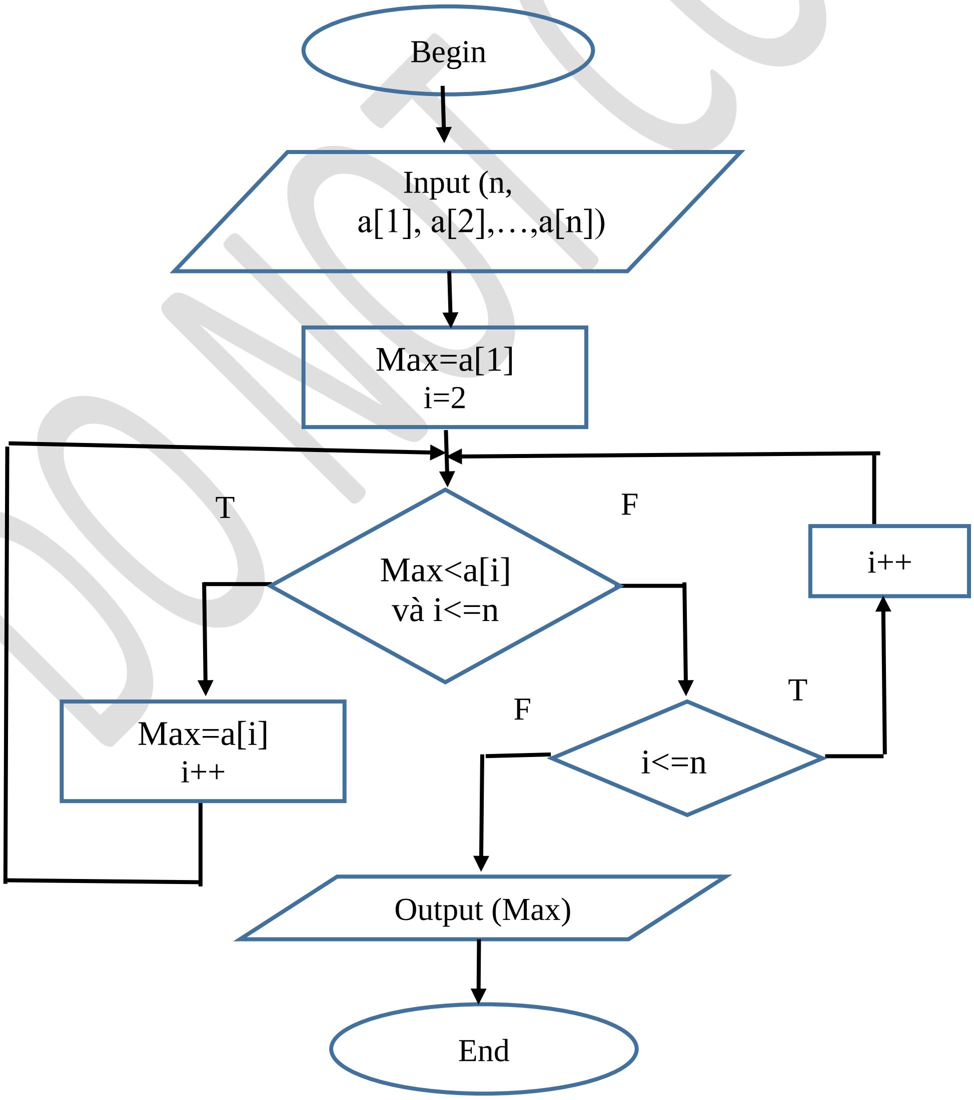
```
Begin
        Nhập a, b, c
        Δ = b*b–4*a*c
```

```
    if(Δ <0) "phương trình vô nghiệm"
    else
        if(Δ >0)
            Begin
                x1 = (–b+sqrt(Δ))/(2*a)
                x2 = (–b-sqrt(Δ))/(2*a)
                "Phương trình có 2 nghiệm phân biệt x1 và x2"
            End
        else
            Begin
                x1= x 2 = –b/(2*a)
                "Phương trinh có nghiệm kép"
            End
End.
```

Ví dụ 2.

Thuật toán tìm số lớn nhất của dãy số a[1], a[2],…,a[n]

- Thuật toán được biểu diễn bằng sơ đồ khối



Hình 1.4 Sơ đồ khối của thuật toán tìm số lớn nhất của dãy

- Thuật toán được biểu diễn bằng cách liệt kê các bước:

Bước 1: Nhập n, dãy a[1], a[2],…,a[n]

Bước 2: Gán max=a[1]

Bước 2: Kiểm tra điều kiện nếu max< a[i], i=2,…,n thì max=a[i]. Kết thúc

- Thuật toán được biểu diễn bằng mã giả

```
Begin
        Nhập n
        for (i=1,…,n) Nhập a[i]
        max=a[1]
        for (i=2,…,n)
                if(max<a[i]) max=a[i]
        Thông báo số lớn nhất là max
End.
```

Ví dụ 3.

Thuật toán tính tổng s=1+2+…+n

Bước 1: Nhập n

Bước 2: Gán s=0

Bước 3: Với i=1,…,n gán lệnh s=s+i

Bước 4: Thông báo tổng s

Ví dụ 4. Thuật toán giải phương trình ax + b = 0

Bước 1: Nhập các số a và b.

Bước 2: Nếu a= 0: chuyển sang bước 3; Ngược lại, chuyển sang bước 4.

Bước 3: Nếu b = 0: Phương trình có vô số nghiệm; Ngược lại: Phương trình vô nghiệm. Kết thúc.

Bước 4: Thông báo: nghiệm của phương trình là -b/a.

Ví dụ 5. Tìm phần tử nhỏ nhất trong ma trận $A=(a_{ij})_{mxn}$

Bước 1: Nhập m, n và các phần tử của ma trận A

Bước 2: Gán $Min=a_{11}$

Bước 3: Tính lại Min

For i=1,…,m $\Rightarrow O(n \times m)$

For j=1,…,n

Nếu $Min>a_{ij}$ *thì* $Min=a_{ij}$

Bước 4: In giá trị nhỏ nhất là Min

Dựa vào sơ đồ khối và mã giả của ví dụ 1 và ví dụ 2 vẽ sơ đồ khối và viết mã giả cho ví dụ 3, 4, 5 (xem như bài tập)

➢ ***Tính chất của thuật toán***

a. Tính đúng đắn

Thuật toán phải giải đúng bài toán. Thông thường để kiểm tra tính đúng đắn của thuật toán người ta cài đặt chương trình thể hiện thuật toán và chạy thử nghiệm với dữ liệu mẫu và so sánh với kết quả đã biết.

b. Tính hữu hạn

Thuật toán phải kết thúc sau một số bước hữu hạn và sau thời gian hữu hạn.

c. Tính tất định

Tại mỗi bước, với cùng một đầu vào thuật toán phải cho cùng một đầu ra.

d. Tính phổ quát

Thuật toán phải áp dụng được cho lớp nhiều bài toán.

e. Tính hiệu quả

- Thời gian thực hiện nhanh.
- Tài nguyên (bộ nhớ, CPU, mạng, …) sử dụng tiết kiệm.

1.1.2 Đánh giá độ phức tạp của thuật toán

Với một bài toán, có nhiều thuật toán được đề xuất. Chọn một thuật toán đưa tới kết quả nhanh là một yêu cầu thực tế và luôn được ưu tiên. Nhưng căn cứ vào đâu để có thể nói thuật toán này nhanh hơn hay chậm hơn thuật toán kia. Có thể thấy ngay, thời gian thực hiện một thuật toán phụ thuộc vào rất nhiều yếu tố. Một yếu tố cần chú ý trước tiên là kích thước dữ liệu đầu vào. Chẳng hạn sắp xếp một dãy số phải chịu ảnh hưởng của số lượng các số thuộc dãy số đó. Nếu gọi n là số lượng dữ liệu (kích thước) đầu vào, thì thời gian thực hiện T của một thuật toán phải được biểu diễn như một hàm của n: T(n).

Các kiểu lệnh và tốc độ xử lý của máy tính, ngôn ngữ lập trình và chương trình dịch đó đều ảnh hưởng tới thời gian thực hiện. Nhưng những yếu tố này không giống nhau trên các máy khác nhau, vì vậy không thể dựa vào chúng để xác lập T(n). Điều đó cũng có nghĩa là T(n) không biểu diễn được bằng giây, phút ...

Thời gian T(n) ở đây phải hiểu là cấp độ lớn của số lượng phép tính, phụ thuộc vào kích thước đầu vào và được gọi là độ phức tạp của thuật toán. Tuy nhiên, thời gian T(n) không chỉ phụ thuộc vào kích thước đầu vào mà còn phụ thuộc vào trạng thái dữ liệu đầu vào. Chẳng hạn nếu dãy cần sắp xếp đã được sắp xếp từ trước với mức độ nào đó, thì thời gian sắp xếp sẽ nhanh hơn nhiều so với dãy bất kỳ. Vì vậy chúng ta cần phân biệt các loại độ phức tạp thực hiện thuật toán sau:

- Độ phức tạp trung bình: $T_{tb}(n)$
- Độ phức tạp xấu nhất: $T_{max}(n)$
- Độ phức tạp tốt nhất: $T_{min}(n)$

Về lý thuyết, người ta thường đánh giá độ phức tạp T(n) trong trường hợp xấu nhất.

Độ phức tạp thường được biểu diễn thông qua các hàm đa thức, luỹ thừa, hàm mũ. Để so sánh cấp độ lớn giữa các hàm người ta dùng ký hiệu O.

- ➢ Ta quan tâm đến độ phức tạp về thời gian: Tức là quan tâm đến việc đánh giá thời gian cần thiết để thực hiện thuật toán.
- ➢ Rõ ràng thời gian tính toán của một thuật toán là hàm phụ thuộc dữ liệu đầu vào (kích thước đầu vào)
- ➢ Để tính thời gian tính toán của thuật toán ta sẽ đếm số câu lệnh mà nó thực hiện hoặc số phép toán số học, logic, gán… mà thuật toán đòi hỏi thực hiện.
- ➢ Không phụ thuộc vào người lập trình, ngôn ngữ lập trình và máy tính.

Ví dụ 1. T(n)= $3n^2+4n+7$ là tổng số phép toán cơ sở của một thuật toán.

Ta có định nghĩa độ phức tạp thuật toán như sau:

Định nghĩa. Cho n là số nguyên không âm, n_0, c là các hằng số dương. Giả sử f(n) và g(n) là các hàm số nguyên phụ thuộc vào n, ta nói f(n) có cấp g(n) và viết f(n)=O(g(n)) nếu f(n)≤ c.g(n), $\forall n \geq n_0$

- Nếu độ phức tạp T(n) của một thuật toán thoả mãn T(n)= O(g(n)), thì ta nói thuật toán có độ phức tạp O(g(n)).

Ví dụ 2. Tìm độ phức tạp của thuật toán có T(n)= $3n^2+4n+7$

Ta có T(n) $\leq 3n^2+4n^2+7n^2=14n^2$ 14=c, n_0=1 $\forall n \geq 1$ n_0

T(n) ≤14. g(n). Suy ra độ phức tạp là T(n)=O(g(n))=O(n^2)

Ví dụ 3. Tính độ phức tạp của thuật toán tính tổng s=1+2+…+n

Với i=1,…,n gán lệnh s=s+i (xem ví dụ 3 ở mục1.2.1) => f(n) = n ≤ 1n

Suy ra số phép toán tích cực T(n)=n. Vậy T(n)=O(n) $\forall n \geq 0$

Thông thường các hàm thể hiện độ phức tạp tính toán của thuật toán có dạng

HV: 17 = 10   10!   1 2 3 ... 10

1 2 3 ---10 y

Khó

$\log_2(\log_2 n)$, $\log_2(n)$, n, $n.\log_2(n)$, n^2, n^3, ..., n^m, 2^n, $n!$, n^n, ...

Các hàm như 2^n, $n!$, n^n gọi là hàm loại mũ. Một thuật toán có độ phức tạp hàm mũ thì tốc độ rất chậm. Các hàm $\log_2(\log_2 n)$, $\log_2(n)$, n, $n.\log_2(n)$, n^2, n^3, ..., n^m được gọi là hàm đa thức. Một thuật toán có độ phức tạp hàm đa thức thì tốc độ chấp nhận được.

Ta có các quy tắc tính toán độ phức tạp như: quy tắc cộng, quy tắc nhân, quy tắc tích cực và cách tính độ phức tạp cho các cấu trúc thuật toán được trình bày cụ thể hơn trong các học phần cấu trúc dữ liệu, phân tích và thiết kế giải thuật.

1.2 Quy nạp toán học và đệ quy

1.2.1 Quy nạp toán học

Quy nạp toán học là một hình thức chứng minh trực tiếp, thường được thực hiện theo hai bước. Bước thứ nhất được gọi là bước cơ sở, ở bước này ta chứng minh mệnh đề đưa ra là đúng với số tự nhiên đầu tiên. Bước thứ hai được gọi là bước quy nạp, ở bước này ta chứng minh rằng nếu mệnh đề được giả định là đúng cho bất kỳ số tự nhiên nào đó, thế thì nó cũng đúng cho số tự nhiên tiếp theo. Sau khi chứng minh hai bước này, ta khẳng định mệnh đề là đúng cho tất cả các số tự nhiên.

Ví dụ 1. Cho một dãy viên bi xếp theo thứ tự 1,2,..., n,... Giả sử viên bi thứ nhất có màu vàng và nếu với mọi k > 1, k-1 viên bi đầu màu vàng thì viên bi thứ k cũng màu vàng. Khi đó ta kết luận rằng tất cả viên bi đều màu vàng.

Từ đó ta có nguyên lý quy nạp toán học như sau:

Giả sử rằng với mỗi số nguyên dương n = 1,2,... ta có mệnh đề lôgic S(n) hoặc đúng hoặc sai. Giả thiết

a) Bước cơ sở: S(1) đúng.
b) Bước qui nạp: Nếu với mọi k > 1, S(i) đúng với mọi i < k ,thì S(k) đúng.

   Khi đó S(n) đúng với mọi n.

✓ Từ nguyên lý trên ta đưa ra phương pháp chứng minh P(n) đúng với mọi n như sau:

  Cho mệnh đề P(n) phụ thuộc vào n

  Để chứng minh P(n) đúng với mọi n ta thực hiện các bước sau:

  - Bước 1 (bước cơ sở): Kiểm tra rằng P(n) đúng với n=1.

  - Bước 2 (Bước quy nạp): Giả sử P(n) đúng với n=k (giả thiết quy nạp), ta chứng minh P(n) đúng với n=k+1.

  - Kết luận P(n) đúng với mọi n.

Ví dụ 2. Chứng minh rằng với mọi n nguyên dương ta có

$$2 + 5 + 8 + \cdots + (3n - 1) = n(3n + 1)/2 \qquad (1)$$

Chứng minh:

Bước 1 (bước cơ sở). Với n=1 ta có vế trái= 2, vế phải =2, suy ra đúng. Vậy (1) đúng với n =1.

Bước 2 (bước quy nạp)

Giả sử (1) đúng với n=k (k>1). Ta chứng minh (1) đúng với n=k+1.

Thật vậy ta có

$$2 + 5 + 8 + \cdots + (3k - 1) = k(3k + 1)/2 \qquad (1^*)$$

Ta chứng minh đúng với n=k+1 nghĩa là

$$2 + 5 + 8 + \cdots + (3k - 1) + [3(k + 1) - 1] = (k + 1)[3(k + 1) + 1)]/2 \quad \text{đúng}$$

$$2 + 5 + 8 + \cdots + (3k - 1) + 3k + 2 = (k + 1)(3k + 4)/2.$$

Thay (1*) vào vế trái ta có vế trái= $k(3k + 1)/2 + 3k + 2 = (3k^2 + 7k + 4)/2$ =Vế phải

Vậy đúng với n=k+1

Kết luận (1) đúng với mọi n (điều phải chứng minh)

Ví dụ 3. Chứng minh rằng với mọi n nguyên dương ta có

$$1 + 3 + 5 + \ldots + (2n - 1) = n^2 \ (1)$$

Chứng minh:

Bước 1 (bước cơ sở). Với n=1 ta có vế trái= 1, vế phải =1, suy ra đúng. Vậy (1) đúng với n =1.

Bước 2 (bước quy nạp)

Giả sử (1) đúng với n=k (k>1). Ta chứng minh (1) đúng với n=k+1.

Thật vậy ta có

$$1 + 3 + 5 + \cdots + (2k - 1) = k^2 \qquad (1^*)$$

Ta chứng minh đúng với n=k+1 nghĩa là chứng minh

$$1 + 3 + 5 + \cdots + (2k - 1) + [2(k + 1) - 1] = (k + 1)^2 \quad \text{đúng}$$

$$1 + 3 + 5 + \cdots + (2k - 1) + 2k + 1 = (k + 1)^2$$

Thay (1*) vào vế trái ta có vế trái= $k^2 + 2k + 1 =$ Vế phải

Vậy đúng với n=k+1

Kết luận (1) đúng với mọi n (điều phải chứng minh)

1.2.2 Giải thuật đệ quy

Khái niệm đệ quy

Đệ quy là phương pháp lập trình cho phép một hàm có thể gọi lại chính nó (thường nhỏ hơn nó) trực tiếp hoặc gián tiếp

Một chương trình đệ quy hoặc một hàm đệ quy không thể gọi đến chính nó mãi mãi mà phải có điểm dừng đến một trường hợp đặc biệt nào đó và được gọi là trường hợp suy biến.

Như vậy hàm đệ quy gồm 2 phần chính

- Phần cơ sở: điều kiện thoát khỏi đệ quy (điểm dừng)

- Phần đệ quy: trong phần thân chương trình có lời gọi đến chính nó (với giá trị mới của tham số nhỏ hơn giá trị ban đầu)

Ví dụ 1. Tính n!

- Phần cơ sở: n=0 thì giaithua=1
- Phần đệ quy: n>0 thì giaithua=n*giaithua(n-1)
- Hàm:

```
long giaithua(int n)
    {
        if  (n == 0)
            return  1;
        else
            return  n*giaithua(n-1);
    }
```

Ví dụ 2. Tính số Fibonacci f(n) = f(n–1) + f(n–2)

- Phần cơ sở: Nếu n = 0, thì f(0) = 0, nếu n = 1, thì f(1) = 1.
- Phần đệ quy: Nếu n > 1, thì f(n) = f(n–1) + f(n–2)
- Hàm

```
long fibonacciDQ(int n)
    {
        if (n<2)
            return n;
        else
        return fibonacciDQ(n-1)+fibonacciDQ(n-2);
    }
```

Ví dụ 3. Tính tổng n phần tử của mảng a (a[0], a[1], …, a[n-1])

- Phần cơ sở: Nếu n = 1, thì tongmangdq = a[0].
- Phần đệ quy: Nếu n > 1, thì tongmangdq= a[n-1]+tongmangdq(a,n-1)
-
-

- Hàm

```
int tongmangdq(int a[],int n)
	{
		if (n==1) return a[0];
		else return a[n-1]+tongmangdq(a,n-1);
	}
```

Ví dụ 4. Cho C(n, k)=C(n-1,k-1)+C(n-1,k), n>=k>0

C(n,0)=C(n,n)=1, với mọi n>=1

- Phần cơ sở: Nếu k = 0 hoặc k=n, thì C(n, k)= 1.
- Phần đệ quy: Ngược lại, thì C(n, k)=C(n-1,k-1)+C(n-1,k)
- Hàm

```
int C(int n, int k)
{
    if (k==0 or k==n) return 1;
    else return C(n-1,k-1)+C(n-1,k);
}
```

Bài tập chương 1

1.1 Xây dựng thuật toán để tính diện tích các hình tròn, hình chữ nhật

1.2 Xây dựng thuật toán nhập số tự nhiên, rồi hiển thị một trong các thông báo sau: "Số nguyên tố", "Không phải số nguyên tố".

1.3 Xây dựng thuật toán tìm số bé nhất của dãy số a[0], a[1],…,a[n-1]

1.4 Xây dựng thuật toán đọc số nguyên, rồi hiển thị số đảo ngược (ví dụ 649 đảo thành 946)

1.5 Tính độ phức tạp của thuật toán tìm số lớn nhất của dãy số a[1], a[2],…,a[n]

1.6 Tính độ phức tạp của của thuật toán tìm phần tử nhỏ nhất trong ma trận $A=(a_{ij})_{mxn}$

1.7 Tính độ phức tạp của khối lệnh sau:

```
for (i= 1;i<=n;i++)
    for (j= 1;j<=m;j++)
        for (k= 1;k<=x;k++)
            //lệnh
```

1.8 $1^2 + 2^2 + ... + n^2 = n(n+1)(2n + 1)/6$

If(n==1) → THN(1,"A","B","C")
ELSE (n-1,'A','C','B')
(1, A, B, C)
(n-1, B, A, C)

1.9 $1(1!) + 2(2!) + ... + n(n!) = (n + 1)! – 1$

1.10 Viết chương trình (đệ quy) giải bài toán tháp Hà Nội.

1.11 Viết chương trình nhập vào 2 số nguyên và dùng thủ tục đệ quy tìm ước số chung lớn nhất của chúng.

1.12 Viết chương trình nhập số tự nhiên n > 0 từ bàn phím và dùng thủ tục đệ quy chuyển sang dạng nhị phân.

1.13 Viết chương trình nhập số tự nhiên n > 0 từ bàn phím và dùng thủ tục đệ quy tính 1 +1/2 + 1/3 + ...+ 1/n.

1.14 Viết chương trình nhập số tự nhiên n > 0 từ bàn phím và dùng thủ tục đệ quy tính 1 + 1/2! + 1/3! + ... + 1/n!.

1.15 Viết chương trình (đệ quy) đọc số nguyên, rồi hiển thị số đảo ngược (ví dụ 649 đảo thành 946)

Chương 2. BÀI TOÁN ĐẾM

Tóm tắt chương

Chương này, sẽ tập trung trả lời câu hỏi có bao nhiêu cấu hình thỏa mãn điều kiện đã nêu? Để trả lời câu hỏi đó, cần dựa vào một số nguyên lý cơ bản và một số kết quả đếm các cấu hình đơn giản và một số phép tính trong tập hợp, các công thức tổ hợp và hệ thức truy hồi.

2.1 Tập hợp

2.1.1 Các khái niệm cơ bản

- Tập hợp được coi là kết hợp các đối tượng có cùng bản chất (thuộc tính, dấu hiệu) chung nào đó.

- Biểu diễn tập hợp

• Liệt kê các phần tử

$A = \{ 1, 2, 3, 4 \}$

$X = \{x_1, x_2, \dots, x_n\}$

• Biểu diễn tập hợp bằng cách mô tả tính chất:

$C = \{n |\ n \text{ là số lẻ}\}$

$Y = \{x | x \text{ là nghiệm của phương trình } 3x^2+7x+4=0\}$

- Lực lượng tập hợp: là số phần tử của A, ký hiệu là |A| hoặc card(A), gọi là lực lượng của tập A. Nếu $|A| < \infty$, ta nói A là tập hữu hạn, nếu $|A| = \infty$, ta nói A là tập vô hạn.

Quan hệ bao hàm: Cho hai tập A, B.

- Nếu mỗi phần tử thuộc A cũng thuộc B ta nói A là tập con của B (hoặc A bao hàm trong B) và ký hiệu $A \subset B$

- Nếu A không phải tập con của B ta ký hiệu $A \not\subset B$

- Nếu $A \subset B$ và $B \subset A$ ta nói A bằng B và ký hiệu A = B

Tập tất cả tập con của A ký hiệu là P(A)

Định lý 1. Nếu $|A| = n$, thì $|P(A)| = 2^n$

Chứng minh

Bước 1: n=1, |A|=1, thì |P(A)|=2 hiển nhiên đúng

Bước 2: Giả sử đúng với n=k, tức là |A|=k thì $|P(A)|=2^k$

Ta chứng minh đúng với n=k+1, nghĩa là ta chứng minh |A|=k+1 thì $|P(A)|=2^{k+1}$

con của

➢ Ta chia tập A có |A|=k+1 làm 2 tập là tập chứa {x} P(A_1) và tập không chứa {x} P(A_2). Ta luôn có tập chứa {x} có số phần tử là k và tập không chứa {x} cũng có số phần tử là k, nghĩa là $|P(A)|= |P(A_1)| +|P(A_2)|=2^k+2^k=2.2^k=2^{k+1}$

- Suy ra đúng với mọi n

Theo nguyên lý quy nạp ta có đpcm

Định lý 2*. Quan hệ bao hàm có các tính chất sau đây.*

- *Phản xạ*: $\forall$ A : A $\subset$ A

- *Phản đối xứng*: $\forall$A, B : A $\subset$ B & B $\subset$ A $\Rightarrow$ A = B

- *Bắc cầu*: $\forall$A, B, C : A $\subset$ B & B $\subset$ C $\Rightarrow$ A $\subset$ C

2.1.2 Các phép toán trên tập hợp

Cho các tập A và B. Ta định nghĩa các phép toán sau.

- *Phép hiệu*: hiệu của A và B, ký hiệu A \ B là tập:

$$A \setminus B = \{\, x | x \in A \;\&\; x \notin B \,\}$$

- *Phần bù*: cho tập X và A $\subset$ X. Phần bù của A (trong X) là tập

$$\overline{A}_X = X \setminus A$$

- Phép hợp: hợp của A và B, ký hiệu A $\cup$ B là tập

$$A \cup B = \{\, x \,|x \in A \text{ hoặc } x \in B \,\}$$

- Phép giao: giao của A và B, ký hiệu A $\cap$ B là tập

$$A \cap B = \{\, x |\, x \in A \;\&\; x \in B \,\}$$

- Phân hoạch:

Nếu A $\cap$ B = $\varnothing$, ta nói A và B rời nhau.

Nếu các tập $X_1, X_2, \dots, X_n$ thoả $A = X_1 \cup X_2 \cup \dots \cup X_n$ và chúng rời nhau từng đôi một, ta nói $\{ X_1, X_2, \dots, X_n \}$ là một phân hoạch của tập hợp A.

2.2 Các nguyên lý đếm cơ bản

2.2.1 Nguyên lý cộng

Giả sử $\{X_1, X_2, \dots, X_n\}$ là một phân hoạch của tập S.

Khi đó $|S| = |X_1| + |X_2| + \dots + |X_n|$

Hệ quả : $|A \cup B| = |A| + |B| - |A \cap B|$

Các luật

a) Luật kết hợp:

$$(A \cup B) \cup C = A \cup (B \cup C)$$

$$(A \cap B) \cap C = A \cap (B \cap C)$$

b) Luật giao hoán:

$A \cup B = B \cup A$

$A \cap B = B \cap A$

c) Luật phân bố:

$A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$

$A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$

d) Luật bù kép

$\bar{\bar{A}} = A$

e) Luật đối ngẫu De Morgan:

$\overline{A \cup B} = \bar{A} \cap \bar{B}$ & $\overline{A \cap B} = \bar{A} \cup \bar{B}$

$\overline{A_1 \cup A_2 \cup \ldots \cup A_n} = \overline{A}_1 \cap \overline{A}_2 \cap \ldots \cap \overline{A}_n$

2.2.2 Nguyên lý nhân

Giả sử có cấu hình tổ hợp xây dựng k bước, bước 1 có thể thực hiện qua n_1 cách, bước 2 có thể thực hiện qua n_2 cách, …, bước k có thể thực hiện qua n_k cách. Khi đó số cấu hình là: $n_1. n_2. \ldots . n_k$

2.2.3 Nguyên lý bù trừ

Cho 2 tập X_1, X_2

Ta có nguyên lý cộng trên 2 tập

$$|X_1 \cup X_2| = |X_1| + |X_1| - |X_1 \cap X_2|$$

Tổng quát lên n tập X_1 đến X_n

$$|X_1 \cup X_2 \cup \ldots \cup X_n| = S_1 - S_2 + \cdots . + (-1)^{n-1} S_n$$
$$= \sum_{k=1}^{n} (-1)^{k-1} S_k$$

S_k là tổng phần tử của tất cả các giao của k (k=1,…,n) tập lấy từ n tập

Giả sử X_1, X_2, X_3 là các tập hợp, khi đó ta có

$|X_1 \cup X_2 \cup X_3|$ X0 + 140 + 120 (− 50 − 20 − 30 + 10

$= |X_1| + |X_1| + |X_3| - |X_1 \cap X_2| - |X_1 \cap X_3| - |X_2 \cap X_3|$
$+ |X_1 \cap X_2 \cap X_3|$

Bây giờ ta cho các tính chất $\alpha_1, \ldots, \alpha_n$ trên tập X. Xét bài toán:

Đếm số phần tử trong X không thoả mãn một tính chất α_k nào cả.

Giải.

Với mọi k = 1, ..., n, ta ký hiệu:

$X_k = \{x \in X | \text{ x thoả mãn } \alpha_k\}$

Như vậy phần bù của X_k là $\overline{X}_k$ ={ x∈ X| x không thoả mãn α_k}

Ký hiệu N là số cần đếm, ta có :

$$N=|\overline{X}_1 \cap \overline{X}_2 \cap \ldots \cap \overline{X}_n| = |\overline{X_1 \cup X_2 \cup \ldots \cup X_n}| = |X| - |X_1 \cup X_2 \cup \ldots \cup X_n|$$

Suy ra $N=|X|+\sum_{k=1}^{n}(-1)^k S_k$

$$N=\sum_{k=0}^{n}(-1)^k S_k \ (S_0=|X|)$$

Ví dụ 1. Công đoàn Khoa CNTT cử đoàn vận động viên đi thi bơi lội và thi cầu lông cho trường. Nam có 12 người. Số vận động viên thi bơi lội có 16 người. Số nữ vận động viên thi cầu lông bằng số nam vận động viên thi bơi lội. Hỏi toàn đoàn có bao nhiêu người?

Giải: đoàn có 2 thành phần nam và nữ, nữ tham gia bơi lội và cầu lông mà số nữ thi cầu lông bằng số nam thi bơi lội nên số nữ bằng 16, suy ra tổng đoàn 28 vận động viên (theo nguyên lý cộng)

Ví dụ 2. Có 50 đề tài về lĩnh vực lập trình Web, 30 đề tài về lập trình di động và 20 đề tài về thiết kế hướng đối tượng. Hỏi một sinh viên có bao nhiêu khả năng để chọn đề tài trong 3 lĩnh vực trên.

Giải: sinh viên có thể chọn ở lĩnh vực 1 với 50 cách, lĩnh vực 2 với 30 cách và lĩnh vực 3 với 20 cách. Vậy có 50+30+20= 100 cách chọn

Ví dụ 3. y có giá trị bằng bao nhiêu khi nhập đầu vào m=20, n=30, k=10, ~~y=0~~

```
for (i= 1;i<=m;i++)     y=y+1;
for (j= 1;j<=n;j++)    y=y+1;
for (k= 1;k<=x;k++)  y=y+1;
```

Giải: có 3 vòng lặp for độc lập nên mỗi lần lặp y tăng lên 1 đơn vị, vậy giá trị của y=20+30+10=60

Ví dụ 4. Có bao nhiêu chuỗi 8 bit bắt đầu bằng 10 hoặc 11?

Giải: Theo nguyên lý nhân có 2^6 chuỗi bắt đầu bằng 10 và có 2^6 chuỗi bắt đầu bằng 11. Vì hai loại chuỗi này khác nhau nên ta có

$$2 . 2^6 = 128$$

chuỗi 8 bit bắt đầu bằng 10 hoặc 11.

Ví dụ 5. Có bao nhiêu cách xếp 5 người đứng thành 1 hàng ngang sao cho A không đứng cạnh B

Giải: theo nguyên lý nhân có 1.2.3.4.5=5! cách xếp 5 người thành hàng ngang

A đứng cạnh B có 2.4! cách xếp, nên ta có 5!-2.4! cách xếp

Ví dụ 6. Một đợt phát hành sổ số với các số vé gồm 2 phần: phần chữ và phần số. Phần chữ gồm 2 chữ cái từ A đến Z, phần số gồm 4 chữ số từ 0 đến 9. Hỏi xác xuất để trúng giải độc đắc là bao nhiêu?

Giải: phần chữ gồm 26.26 cách chọn. Phần số có 10^4 cách chọn. Vậy theo nguyên lý nhân có $m=26.26.10^4$ tờ vé số khác nhau. Nếu chỉ có 1 tờ vé trúng giải độc đắc thì xác xuất là 1/m.

Ví dụ 7. Đếm số cách chọn 2 quyển sách chuyên ngành khác nhau từ 6 quyển thuật toán khác nhau, 4 quyển lập trình web khác nhau, và 3 quyển lập trình C khác nhau.

Giải

Theo nguyên lý nhân ta có:

6x4 = 24 cách chọn 1 quyển thuật toán, 1 quyển lập trình web

6x3 = 18 cách chọn 1 quyển thuật toán, 1 quyển lập trình C

4x3 = 12 cách chọn 1 quyển lập trình web, 1 quyển lập trình C

Theo nguyên lý cộng ta có: $24 + 18 + 12 = 54$ cách chọn sách

Ví dụ 8. Có bao nhiêu xâu nhị phân có độ dài bằng 8 hoặc là bắt đầu bởi 00 hoặc là kết thúc bởi 01

Giải

Có $2^6 = 64$ xâu nhị phân độ dài 8 bắt đầu bởi 00 và $2^6 =64$ xâu nhị phân độ dài 8 kết thúc bởi 01. Số xâu nhị độ dài 8 bắt đầu 00 và kết thúc 01 là $2^4 =16$

Vậy theo nguyên lý bù trừ suy ra số xâu nhị phân có độ dài bằng 8 hoặc là bắt đầu bởi 00 hoặc là kết thúc bởi 01 là: 2.64-16=112 cách

2.3 Giải tích tổ hợp

2.3.1 Chỉnh hợp lặp

Định nghĩa 1. Một *chỉnh hợp lặp chập k của n phần tử* là một bộ có thứ tự gồm k thành phần lấy từ n phần tử đã cho. Các thành phần có thể được lặp lại.

Một chỉnh hợp lặp chập k của n có thể xem như một phần tử của tích Đề-các X^k, với X là tập n phần tử. Như vậy số tất cả các chỉnh hợp lặp chập k của n là n^k

Ví dụ 1. Tính số ánh xạ từ tập X có k phần tử đến tập Y có n phần tử. Mỗi ánh xạ từ X vào Y tương ứng với một bộ có thứ tự k thành phần của n phần tử của Y, các phần tử có thể lặp lại. Như vậy số ánh xạ từ X vào Y là n^k

Ví dụ 2. Tính số dãy nhị phân độ dài n

Giải

Mỗi dãy nhị phân độ dài n là bộ gồm n thành phần, trong đó mỗi thành phần lấy từ 2 giá trị 0 hoặc 1, suy ra 2^n dãy nhị phân có độ dài n.

Ví dụ 3. Tập X={x_1, x_2, ….,x_n} có n phần tử, hãy tìm số tập con của tập X

Giải

Biểu diễn mỗi tập con A của tập X bằng một dãy nhị phân có độ dài n là b=(b_1, b_2, …., b_n). Trong đó nếu b_i=1 thì $x_i \in A$ còn nếu b_i=0 thì $x_i \notin A$. Như trong ví dụ 2 ta có số dãy nhị phân là 2^n nên số tập con A cũng bằng 2^n

b) Chỉnh hợp không lặp

Định nghĩa 2. *Một chỉnh hợp không lặp chập k của n phần tử* là một bộ có tự gồm *k* thành phần lấy từ *n* phần tử đã cho. Các thành phần không được lặp lại.

Một chỉnh hợp không lặp chập *k* của *n* có thể được xây dựng qua k bước kế tiếp như sau:

Chọn thành phần đầu: có n khả năng.

Chọn thành phần thứ hai: có n - 1 khả năng.

...

Chọn thành phần thứ k: có n - k + 1 khả năng.

Như vậy, theo nguyên lý nhân, số tất cả chỉnh hợp không lặp chập k của n phần tử là:

$$A(n,k) = n.(n-1).....(n-k+1) = \frac{n!}{(n-k)!}$$

Ví dụ 4. Tính số đơn ánh từ tập X có k phần tử đến tập Y có n phần tử (k≤n).

Thật vậy, ta có mỗi hàm đơn ánh từ X vào Y tương ứng với một chỉnh không lặp chập k của n phần tử của Y. Như vậy, số đơn ánh cần tìm là A= *n.(n-1).....(n-k+1).*

2.3.2 Hoán vị

Định nghĩa 3. Một *hoán vị* của n phần tử là một cách sắp xếp thứ tự các phần tử đó.

Hoán vị có thể coi như trường hợp riêng của chỉnh hợp không lặp chập k của n trong đó k = n. Ta có số hoán vị là *P(n)* = *n!*.

Ví dụ 5. Có 10 người xếp thành hàng ngang để chụp ảnh. Hỏi có thể bố trí bao nhiêu kiểu khác nhau?

Giải

Mỗi kiểu ảnh là một hoán vị của 10 người. Vậy số kiểu ảnh là 10!

2.3.3 Tổ hợp

Định nghĩa 4. Một *tổ hợp chập k của n phần tử* là một bộ không kể thứ tự gồm *k* thành phần khác nhau lấy từ *n* phần tử đã cho.

Nói cách khác, ta có thể coi một tổ hợp chập *k* của *n* phần tử là một tập con có *k* phần tử từ *n* phần tử đã cho.

Gọi số tổ hợp chập k của n phần tử là C(n, k) ta có : *A(n,k) = C(n,k) * k! (k! là số hoán vị)*

Suy ra

$$C(n,k) = \frac{n!}{k!\,(n-k)!}$$

Ví dụ 6. Có *n* đội bóng thi đấu vòng tròn. Phải tổ chức bao nhiêu trận đấu bóng tất cả?

Ta có, mỗi trận ứng với một tổ hợp chập 2 của n. Vậy có *C(n,2)* trận đấu.

2.3.4 Hoán vị lặp

Định nghĩa 5. *Hoán vị lặp* là hoán vị trong đó mỗi phần tử được ấn định một số lần lặp lại cho trước.

Định lý 1: Giả sử tập S có n phần tử, trong đó có n_1 kiểu 1, n_2 kiểu 2, ..., n_k kiểu k ($n_1+n_2+...+n_k=n$). Khi đó số các hoán vị lặp n phần tử của S là

$$C_n(n_1, n_2, \ldots, n_k) = \frac{n!}{n_1!.\,n_2!.\,\ldots.\,n_k!}$$

Ví dụ 7. Cho biết có thể nhận bao nhiêu xâu ký tự khác nhau bằng cách sắp xếp lại các chữ cái của từ MISSISSIPPI.

+

- Có 11! Cách xếp 11 ký tự khác nhau
- Trong xâu trên có 4 ký tự S nên có 4!, 4 ký tự I nên có 4!, 2 ký tự P nên có 2! cách xếp, 1 ký tự M nên có 1! Cách xếp
- Vậy có 11!/(4!.4!.2!.1!) cách sắp xếp

2.3.5 Tổ hợp lặp

Định nghĩa 6. *Tổ hợp lặp* chập *k* từ *n* phần tử là một nhóm không phân biệt thứ tự gồm *k* phần tử trích từ *n* phần tử đã cho, trong đó các phần tử có thể được lặp lại.

Định lý 2. Giả sử X có *n* phần tử. Khi đó, số tổ hợp lặp chập *k* từ *n* phần tử của X là

$$CR(n, k) = C(k + n - 1, n - 1) = C(k + n - 1, k).$$

Ví dụ 8. Phương trình

$$x_1 + x_2 + x_3 + x_4 = 10$$

có bao nhiêu bộ nghiệm nguyên không âm?

Giải

Ta có, mỗi bội nghiệm nguyên không âm của phương trình tương ứng 1-1 với một cách chọn 10 phần tử, trong đó phần tử kiểu i lặp lại x_i lần, $i=1,...,4$. Vậy số bộ nghiệm là số tổ hợp lặp chập 10 của 4. Vậy ta có số nghiệm là

$$CR(4,10) = C(10 + 4 -1, 4 -1) = C(13, 3) = 286$$

Ví dụ 9. Phương trình

$$x_1 + x_2 + x_3 + x_4 = 20$$

có bao nhiêu bộ nghiệm nguyên dương?

Giải

Phương trình

$$x_1 + x_2 + x_3 + x_4 = 20$$

có bao nhiêu bộ nghiệm nguyên với điều kiện $x_1 \geq 1$, $x_2 \geq 1$, $x_3 \geq 1$, $x_4 \geq 1$ *hay* $x_1 - 1 \geq 0$, $x_2 - 1 \geq 0$, $x_3 - 1 \geq 0$, $x_4 - 1 \geq 0$.

đặt $a = x_1 - 1$, $b = x_2 - 1$, $c = x_3 - 1$, $d = x_4 - 1$

Ta có, phương trình a+b+c+d=16 có bao nhiêu bộ nghiệm nguyên không âm?

Ta có, mỗi bội nghiệm nguyên không âm của phương trình tương ứng 1-1 với một cách chọn 16 phần tử, trong đó phần tử kiểu i lặp lại x_i lần, $i=1,...,4$. Vậy số bộ nghiệm là số tổ hợp lặp chập 16 của 4. Vậy ta có số nghiệm là

$$CR(4,16) = C(16 + 4 -1, 4 -1) = C(19, 3)$$

2.4 Hệ thức truy hồi

2.4.1 Công thức truy hồi

Công thức truy hồi của dãy s(0), s(1), s(2),... là phương trình xác định s(n) bằng các phần tử s(0), s(1), s(2), ..., s(n–1) trước nó.

$$s(n) = F(s(0), s(1), s(2),..., s(n-1))$$

Điều kiện ban đầu là các giá trị gán cho một số hữu hạn các phần tử đầu. Trong ví dụ trên ta có điều kiện ban đầu là $s(0) = 1$.

Ví dụ 10. Xét bài toán đếm số tập con P(X) của tập X. Gọi s(n) là số tập con của tập có n phần tử. Cho *x* là phần tử thuộc X. Tách P(X) ra làm hai nhóm, nhóm tập con chứa *x* và nhóm tập con không chứa *x*. Ta có công thức

$$s(n) = 2.s(n-1) \quad \forall n$$

Đây là một công thức truy hồi.

2.4.2 Giải công thức truy hồi bằng phương pháp lặp

Nội dung của phương pháp này là thay thế liên tiếp công thức truy hồi vào chính nó, mỗi lần thay bậc *n* giảm ít nhất 1 đơn vị, cho đến khi đạt giá trị ban đầu.

Ví dụ 11. Đếm số tập con của tập X.

Ta có

$s(n) = 2.s(n-1)$ & $s(0) = 1$

Ta có

$s(n) = 2.s(n-1) = 2.2.s(n-2) = ... = 2.2.....2.s(0) = 2^n$

Ví dụ 12. *Bài toán tháp Hà nội*

Có 3 cọc, cọc thứ nhất có *n* đĩa kích thước khác nhau xếp chồng nhau, đĩa nhỏ nằm trên đĩa lớn. Hãy chuyển các đĩa từ cọc thứ nhất sang cọc thứ ba, sử dụng cọc trung gian thứ hai, sao cho luôn đảm bảo đĩa nhỏ trên đĩa lớn. Hãy đếm số lần di chuyển đĩa. Phương pháp di chuyển các đĩa như sau:

Chuyển n–1 đĩa từ cọc 1 sang cọc 2, chuyển đĩa lớn nhất từ cọc 1 sang cọc 3, và cuối cùng chuyển n–1 đĩa từ cọc 2 sang cọc 3.

Như vậy, nếu s(n) là số lần di chuyển n đĩa, thì ta có công thức truy hồi tính số lần di chuyển đĩa

$s(n) = 2.s(n-1) + 1$ & $s(1) = 1$

Ta có $s(n) = 2.s(n-1) + 1$

$= 2(2.s(n-2) + 1) + 1$

$= 2^2.s(n-2) + 2 + 1$

$= 2^2(2.s(n-3) + 1) + 2 + 1$

$= 2^3.s(n-3) + 2^2 + 2 + 1$

$2^4+2^3+2^2+2^1+2$

$= 2^5$

...

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

$= 2^{n-1} + 2^{n-2} + 2^{n-3} + ... + 2 + 1$

$= 2^{n-1} + 2^{n-2} + 2^{n-3} + ... + 2^1 + 2 - 1 = 2^n - 1$

2.4.3 Giải công thức truy hồi bằng phương trình đặc trưng

Công thức truy hồi *tuyến tính thuần nhất bậc k hệ số hằng* có dạng

$$s(n) = c_1.s(n-1) + c_2.s(n-2) + ... + c_k.s(n-k),\ \ c_k \neq 0 \qquad (1)$$

trong đó c_i, i=1, ..., k , là các hằng số.

Điều kiện ban đầu là

$$s(0) = C_0\,,\ s(1) = C_1\,,\ \ldots\,,\ s(k-1) = C_{k-1}. \qquad (2)$$

6

Phương trình đặc trưng của công thức truy hồi (1) là

$$t^k - c_1.t^{k-1} - c_2.t^{k-2} - \ldots - c_k \ = \ 0 \qquad (3)$$

?

Định lý 1. Nếu s_1 và s_2 là nghiệm của công thức (1) thì

$$a.s_1 + b.s_2$$

cũng là nghiệm của (1) với mọi hằng a, b.

Định lý 2. Nếu r là nghiệm bội m của phương trình đặc trưng (3) thì

$$r^n\,,\ n.r^n\,,\ \ldots\,,\ n^{m-1}.r^n$$

là các nghiệm của (1)

Định lý 3. Nếu k = 2 và α và β là hai nghiệm phân biệt của (3) thì tồn tại hằng a, b , c

k = 3

sao cho

$$s(n) = a.\alpha^n + b.\beta^n + c.\gamma^n$$

là nghiệm của (1) và thoả mãn điều kiện ban đầu (2).

Định lý 4. Nếu k = 2 và α là nghiệm kép của (3) thì tồn tại hằng a, b sao cho

$$s(n) = a.\alpha^n + b.n.\alpha^n$$

là nghiệm của (1) và thoả mãn điều kiện ban đầu (2).

Ví dụ 13. Giải công thức truy hồi sau

a(n)=2a(n-1)+5a(n-2)-6a(n-3) ;a(0)=0, a(1)=-4, a(2)=8

c_1

$$\begin{cases} x+y+z=0 \\ x+3y-2z=-4 \\ x+9y+4z=8 \end{cases}$$

Giải:

Bước 1. Lập phương trình đặc trưng t^3-2t^2-5t+6=0

Bước 2. Giải được phương trình đặc trưng t1=1, t2=3, t3=-2

Bước 3. Tìm dạng tổng quát a(n)=x.t1^n+y.t2^n+z.t3^n

Bước 4. Dựa vào các điều kiện đầu tìm được $\boxed{x= -24/15,\ y = 1/5,\ z=22/15}$

-2 2/5 8/5

Bước 5. Thay x, y, z và t1, t2, t3 ta tìm được a(n).

$a(n) = -2 + \frac{2}{5}3^n + \frac{8}{5}(-2)^n$

Ví dụ 14. Giải công thức truy hồi sau

a(n) = 6.a(n–1) – 8.a(n–2) với n ≥ 2, a(0) = 1, a(1) = 0

$c_1 = 6 \quad c_2 = -8$

Bước 1. Lập phương trình đặc trưng t^2-6t+8=0

Bước 2. Giải được phương trình đặc trưng t1=2, t2=4

Bước 3. Tìm dạng tổng quát a(n)=x.t1^n+y.t2^n

Bước 4. Dựa vào các điều kiện đề bài để tìm được x,y ⇐ $\begin{cases} x+y=1 \\ 2x+4y=0 \end{cases}$

Bước 5. Thay x, y và t1, t2 ta tìm được a(n). =

Bài tập chương 2

2.1 Có bao nhiêu biến trong ngôn ngữ C độ dài 8 chỉ chứa 3 chữ cái A, B, C bắt đầu bởi AA hoặc AB

2.2 Với đầu vào n, m, k. Hãy tính số lệnh thực hiện được theo n, m, k

```
for (i= 1;i<=n;i++)
        for (j= 1;j<=m;j++)
                for (k= 1;k<=x;k++)
                        //lệnh
```

2.3 Trung tâm ngoại ngữ ABC có 110 em học cấp độ B1, 140 em học cấp độ B2, 120 em học cấp độ C1, 50 em học cả B1 và B2, 30 em học cả B1 và C1, 20 em học cả B2 và C1 và 10 em học cả 3 cấp độ B1, B2, C1 cùng thời điểm. Hỏi có tất cả bao nhiêu học viên của trung tâm ABC, biết rằng học viên nào cũng phải học ít nhất 01 cấp độ.

2.4 Xét 3 chuỗi ký tự trên tập mẫu tự {a, b, c} (với a < b < c) : t1 = ac, t2 = aacb, t3 = aba.

a. Hãy sắp xếp chúng theo thứ tự tăng đối với thứ tự từ điển.

b. Cho biết giữa t1 và t3 có bao nhiêu chuỗi ký tự có chiều dài 5.

2.5 Hỏi trong tập X={1,2,…,10000} có bao nhiêu số không chia hết cho bất cứ số nào trong các số 3,4,7.

2.6 Giả sử ta có 3 đầu sách: Toán, Tin, Lý và mỗi đầu sách có ít nhất 6 quyển. Hỏi có bao nhiêu cách chọn ra 6 quyển.

2.7 Giả sử có 3 viên bi đỏ, 2 viên bi xanh và 4 viên bi trắng. Hỏi có bao nhiêu cách sắp các viên bi trên theo hàng ngang.

2.8 Có bao nhiêu cách xếp các chữ cái A, B, C, D, E, F, G, H chứa xâu FGH?

2.9 Có bao nhiêu cách xếp các chữ cái A, B, C, D, E, F, G, H chứa các chữ F, G, H kề nhau

2.10 Một tổ sinh viên có 10 nam và 6 nữ xếp thành hàng dọc. Hỏi có bao nhiêu cách xếp hàng để không có hai sinh viên nữ đứng gần nhau?

2.11 Có bao nhiêu cách xếp *n* bit 0 và *m* bit 1 trên hàng ngang sao cho không có 2 bit 0 kề nhau

2.12 Xét biển số xe: A1A2A3N1N2N3N4

A_i(i=1..3): A->Z;

N_j(j=1..4): 0->9

a. Hỏi có bao nhiêu biển số khác nhau?

b. Hỏi có bao nhiêu biển số thỏa điều kiện: ba mẫu tự khác nhau đôi một và trong biển số có đúng 1 chữ số 4 và 1 chữ số 6?

c. Hỏi có bao nhiêu biển số thỏa điều kiện: trong biển số có ít nhất 1 chữ số 4 và 1 chữ số 6?

2.13 Cho biết có thể nhận bao nhiêu xâu ký tự khác nhau bằng cách sắp xếp lại các chữ cái của từ SUCCESS.

$\frac{7!}{3!\cdot 2!} = 7.5 \quad 420$

2.14 Giả sử chúng ta có 5 viên bi giống nhau và 3 chiếc túi khác màu là xanh, vàng và đỏ. Cho biết có bao nhiêu cách bỏ bi vào các túi? Ví dụ: cách 1 -> túi xanh 5 viên, túi vàng và túi đỏ không có bi; cách 2 -> túi xanh 3 viên, túi vàng và túi đỏ mỗi túi 1 viên, …

2.15 Giả sử chúng ta có 5 viên bi (2 bi sắt, 2 bi chai và 1 bi đất) và 3 chiếc túi màu xanh, vàng và đỏ. Cho biết có bao nhiêu cách bỏ bi vào các túi? Ví dụ: Cách 1 túi xanh chứa 2 bi sắt, túi vàng 2 bi chai và túi đỏ 1 bi đất; cách 2 -> túi xanh 1 bi sắt, túi vàng 2 bi chai + 1 bi sắt và túi đỏ 1 bi đất, …

2.16 Giả sử chúng ta có 5 viên bi (2 bi sắt, 2 bi chai và 1 bi đất). Cho biết có bao nhiêu cách sắp chúng thành hàng? Ví dụ: sắt sắt chai chai đất, sắt chai sắt chai đất,…

2.17 Phương trình

$$x_1 + x_2 + x_3 + x_4 = 20$$

có bao nhiêu bộ nghiệm nguyên với điều kiện $x_1 \geq 6$, $x_2 \geq 3$, $x_3 \geq 9$, $x_4 \geq -2$

35.

2.18 Giải công thức truy hồi sau

$a(n) = 2.n.a(n-1)$ với $n \geq 1$, $a(0) = 1$

2.19 Giải công thức truy hồi sau

$a(n) = a(n-1) + n$ với $n \geq 1$, $a(0) = 0$

2.20 Giải công thức truy hồi sau

$a(n) = 7.a(n-1) - 10.a(n-2)$ với $n \geq 2$, $a(0) = 5$, $a(1) = 16$

2.21 Giải công thức truy hồi sau

$a(n) = a(n-1) + 6.a(n-2)$ với $n \geq 2$, $a(0) = 3$, $a(1) = 6$

2.22 Giải công thức truy hồi sau $\cancel{x_1} = x_2 = 3 = t_1 = t_2$

$a(n) = 6.a(n-1) - 9.a(n-2)$ với $n \geq 2$, $a(0) = 1$, $a(1) = 1$

$a(n) = x.n.3^n + y.3^n$

$= 3^n - \frac{2}{3}n.3^n$

$\begin{cases} y = 1 \\ 3x + 3y = 1 \end{cases} \Rightarrow \boxed{\begin{cases} y = 1 \\ x = \frac{-2}{3} \end{cases}}$

Chương 3. BÀI TOÁN TỒN TẠI

Tóm tắt chương

Trong nhiều trường hợp việc xác định sự tồn tại một cấu hình thoả mãn tính chất nào đó cũng có ý nghĩa quan trọng về mặt lý thuyết cũng như thực tế. Vì thế, một dạng bài toán tiếp theo trong tổ hợp là bài toán tồn tại đó là bài toán xét sự tồn tại các cấu hình tổ hợp thoả mãn các tính chất cho trước.

Bài toán tồn tại được nghiên cứu từ rất lâu và góp phần đáng kể thúc đẩy sự phát triển của lý thuyết tổ hợp cũng như nhiều ngành toán học khác.

3.1 Giới thiệu một số bài toán tồn tại

Bài toán 36 sĩ quan

Bài toán này do nhà toán học *Euler* đưa ra, nội dung của nó như sau. Người ta triệu tập từ 6 trung đoàn, mỗi trung đoàn 6 sĩ quan có 6 cấp bậc khác nhau: thiếu uý, trung uý, thượng uý, đại uý, thiếu tá và trung tá. Hỏi có thể xếp 36 sĩ quan này thành hình vuông 6×6 sao cho trong mỗi hàng ngang cũng như hàng dọc đều có đại diện của 6 trung đoàn và có cả 6 cấp bậc khác nhau?

Để đơn giản ta dùng các chữ cái lớn A, B, C, D, E, F để chỉ 6 trung đoàn và các chữ cái nhỏ a, b, c, d, e, f để chỉ 6 cấp bậc. Bài toán có thể tổng quát hoá bằng cách thay số 6 bằng n.

Trường hợp n = 4 ta có một lời giải sau:

Bảng 3.1 Bài toán 36 sĩ quan với n=4

Ab	Dd	Ba	Cc
Bc	Ca	Ad	Db
Cd	Bd	Dc	Aa
Da	Ac	Cb	Bd

Trường hợp n = 5 ta có một lời giải sau:

Bảng 3.2 Bài toán 36 sĩ quan với n=5

Aa	Bd	Cc	Dd	Ee
Cd	De	Ea	Ab	Bc
Eb	Ac	Bd	Ce	Da
Be	Ca	Db	Ec	Ad
Dc	Ed	Ae	Ba	Cb

Do lời giải của bài toán có thể biểu diễn bởi hình vuông với các chữ cái hoa và thường xếp cạnh nhau nên còn có tên gọi là *bài toán hình vuông la tinh trực giao*.

Euler đã mất nhiều công sức để tìm lời giải cho bài toán nhưng không thành công. Vì vậy ông đưa ra giả thuyết rằng cách xếp như vậy không tồn tại. Giả thuyết này được nhà toán học Pháp *Tarri* chứng minh năm 1901 bằng cách duyệt tất cả khả năng xếp. Dựa trên giả thuyết không tồn tại lời giải cho n = 2 và n =6 Euler còn đưa ra giả thuyết tổng quát hơn là: Không tồn tại hình vuông la tinh trực giao cấp 4k + 2. Giả thuyết này tồn tại suốt 2 thế kỷ. Mãi đến năm 1960 ba nhà toán học Mỹ là *Boce*, *Parker*, *Srikanda* mới chỉ ra một lời giải với n = 10 và sau đó đưa ra phương pháp xây dựng hình vuông la tinh trực giao cấp 4k + 2 với k >1.

Bài toán có nhiều ứng dụng trong quy hoạch thực nghiệm, hình học xạ ảnh, sắp xếp các lịch thi đấu trong các giải cờ quốc tế ...

Bài toán 2n điểm trên lưới n× n điểm

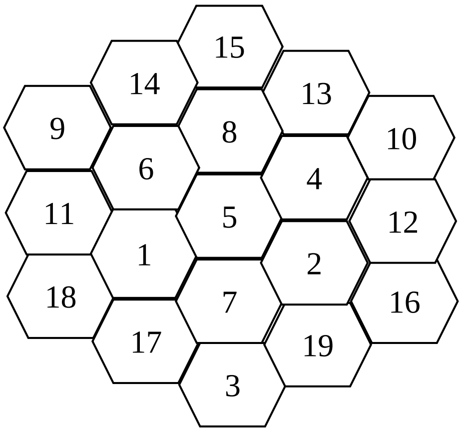
Cho lưới ô vuông gồm n × n điểm. Hỏi có thể chọn trong chúng 2n điểm sao cho không có 3 điểm nào thẳng hàng hay không? Hiện nay người ta mới biết lời giải đối với n ≤ 15. Dưới đây là lời giải với n = 12.

Bảng 3.3 *Bài toán 2n điểm (với n=12)*

	•				•						
		•							•		
						•		•			
	•	•									
			•	•							
										•	•
						•			•		
•											•
				•						•	
					•		•				
			•				•				
•								•			

Bài toán hình lục giác thần bí

Năm 1910 Clifford Adams đề ra bài toán hình lục giác thần bí như sau trên 19 ô lục giác, hãy điền các số từ 1 đến 19 sao cho tổng theo 6 hướng của lục giác đều bằng nhau và bằng 38



Hình 3.1 Bài toán 19 hình lục giác thần bí

Sau 47 năm trời kiên nhẫn cuối cùng ông ta đã tìm được lời giải. Sau đó vì sơ ý đánh mất bản thảo nên ông ta đã tốn thêm 5 năm để khôi phục lại. Năm 1962 Adams đã công bố lời giải đó.

Thật không thể ngờ là đó là lời giải duy nhất (nếu không tính đến các lời giải sai khác nhau bởi phép biến hình đơn giản).

Bài toán 4 màu

Bài toán có thể phát biểu trực quan như sau: Chứng minh rằng mọi bản đồ trên mặt phẳng đều có thể tô bằng 4 màu sao cho không có hai nước láng giềng nào lại bị tô bởi cùng một màu. Chú ý rằng, ta xem như mỗi nước là một vùng liên thông và hai nước được gọi là láng giềng nếu chúng có chung biên giới là một đường liên tục.

Con số 4 không phải là ngẫu nhiên. Người ta đã chứng minh được rằng mọi bản đồ đều được tô với số màu lớn hơn 4, còn với số mầu ít hơn 4 thì không tô được. Chẳng hạn bản đồ gồm 4 nước không thể tô được với số màu ít hơn 4.

Bài toán này xuất hiện vào khoảng những năm 1950-1952 từ một nhà buôn người Anh là Gazri khi tô bản đồ hành chính nước Anh đã cố gắng chứng minh rằng nó có thể tô bằng 4 màu. Sau đó, 1952 ông đã viết bức thư cho De Morgan để thông báo về giả thiết này. Keli trong một bài báo đăng ở truyển tập các công trình của hội toán học Anh có hỏi rằng bài toán này đã được giải quyết hay chưa. Từ đó bài toán trở thành nổi tiếng, và trong suốt hơn một thế kỷ qua đã có rất nhiều người làm toán nghiệp dư cũng như chuyên nghiệp đã cố gắng chứng minh giả thiết này. Tuy nhiên, mãi đến năm 1976, K. Appel và W. Haken mới chứng minh được giả thiết này bằng máy tính điện tử. Tất nhiên một chứng minh với sự giúp đỡ của máy tính điện tử không thật sự thoả mãn được nhu cầu của công chúng muốn kiểm tra tính đúng đắn của cách chứng minh. Vì vậy, chính hai tác giả trên vào cuối những năm 1990 đã cho công bố một cuốn sách trình bày về phương pháp chứng minh của mình (cuốn sách dày trên 800). Cũng vào những năm cuối thế kỹ 20, một nhóm các nhà toán học Mỹ đã đưa ra một chứng minh có thể kiểm tra bằng tay! Rất tiết là chứng minh này cũng không phải là đơn giản. Cho đến nay các nhà toán học vẫn đang nỗ lực nghiên cứu để tìm ra một cách chứng minh dễ hiểu như bản thân nội dung của bài toán

3.2 Nguyên lý Dirichlet

a) Nguyên lý Dirichlet

Nếu xếp nhiều hơn k đối tượng vào k cái hộp thì tồn tại hộp chứa ít nhất 2 đối tượng.

Ví dụ 1. Trong 367 người bao giờ cũng có hai người trùng ngày sinh nhật, bởi vì trong năm chỉ có nhiều nhất 366 ngày.

Ví dụ 2. Mười người có họ Trần, Lê, Nguyễn và tên là Hùng, Hưng, Hai. Khi đó sẽ có ít nhất 2 người trùng họ và tên, bởi vì chỉ có 9 cặp họ tên khác nhau.

Ví dụ 3. Trong kỳ thi cuối kỳ điểm bài thi được đánh giá bởi số nguyên từ 0 đến 10. Hỏi rằng ít nhất phải có bao nhiêu sinh viên dự thi để cho chắc chắn ít nhất phải có 2 sinh viên có kết quả thi như nhau

Giải: Số học sinh cần tìm là 12 vì có 11 kết quả điểm thi khác nhau

b) Nguyên lý Dirichlet tổng quát

Nếu xếp N đối tượng vào k cái hộp thì tồn tại hộp chứa ít nhất $\lceil N/k \rceil$ *đối tượng* ($\lceil x \rceil$ *là số nguyên nhỏ nhất* $\geq x$ *gọi là trần nguyên của x).*

Ví dụ 4. Có 6 loại học bổng khác nhau. Hỏi rằng phải có ít nhất bao nhiêu sinh viên để chắc chắn rằng có ít ra 7 người cùng nhận học bổng như nhau

Chứng minh: ta phải có $\lceil N/k \rceil > 6$,

Tức là $\lceil N/6 \rceil > 6$, vậy N=36+1=37

Như vậy 37 là số lượng sinh viên nhỏ nhất đảm bảo chắc chắn có 7 sinh viên cùng nhận một loại học bổng

Ví dụ 5. Trong 100 người bao giờ cũng có ít nhất 9 người trùng tháng sinh.

Giải.

Thật vậy, xếp những người cùng tháng sinh vào 1 nhóm. Có 12 nhóm tất cả. Ta có $\lceil 100/12 \rceil = 9$. Vậy theo nguyên lý Dirichlet tổng quát tồn tại nhóm có ít nhất 9 người.

Ví dụ 6. Trong hội nghị có *n* người bao giờ cũng có 2 người có số người quen bằng nhau.

Chứng minh.

Thật vậy, xếp những người có số người quen bằng nhau và bằng i vào cùng 1 nhóm, kí hiệu *nhóm i*, $0 \le i \le n - 1$. Ta phân 2 trường hợp:

(i) Có 1 người không quen ai cả. Trong trường hợp này không có ai quen (n−1) người. Vì vậy ta có tối đa (n−1) nhóm 0, 1, 2, ..., n−3, n−2. Như vậy theo nguyên lý Dirichlet phải tồn tại nhóm có ít nhất 2 người.

(ii) Ai cũng có người quen. Ta có tối đa (n−1) nhóm 1,2,...,n−2,n−1. Như vậy theo nguyên lý Dirichlet phải tồn tại nhóm có ít nhất 2 người.

***Ví dụ** 7*. Trong mặt phẳng cho 6 điểm, trong đó không có ba điểm thẳng hàng. Các điểm được nối với nhau từng cặp một bởi các cạnh xanh hoặc đỏ. Chứng minh rằng luôn tồn tại tam giác có các cạnh cùng màu.

Chứng minh.

Chọn điểm P bất kỳ từ 6 điểm. Từ P có 5 cạnh nối đến 5 điểm còn lại. Như vậy theo nguyên lý Dirichlet sẽ có $3 = \rceil 5/2 \lceil$ cạnh cùng màu PP_1, PP_2, PP_3. Giả sử đó là màu xanh. Nếu các cạnh P_1P_2, P_2P_3, P_3P_1 màu đỏ thì ta có $\Delta P_1P_2P_3$ cùng màu. Nếu một trong các cạnh P_1P_2, P_2P_3, P_3P_1 có màu xanh, giả sử đó là P_iP_j thì ΔPP_iP_j cùng màu xanh.

Bài tập chương 3

3.1 Một trung tâm máy tính có 151 máy vi tính. Các máy của trung tâm được đặt tên bởi một số nguyên dương trong khoảng từ 1 đến 300 sao cho không có hai máy nào được đặt tên trùng nhau. Chứng minh rằng luôn tìm được 2 máy có tên là các số nguyên liên tiếp.

3.2 Có 12 cầu thủ bóng rổ đeo áo với số từ 1 đến 12 đứng tập trung thành mọt vòng tròn giữa sân. Chứng minh rằng luôn tìm được 3 người liên tiếp có tổng các số trên áo là lớn hơn hoặc bằng 20.

3.3 Các học sinh của một lớp học gồm 45 *nam* và 35 *n*ữ được xếp ra thành một hàng ngang. Chứng minh rằng, trong hàng đó luôn tìm được hai học sinh nam mà ở giữa họ có 8 người đứng xen vào.

3.4 Chứng minh rằng trong nhóm 10 người có 2 người có tổng hoặc hiệu của tuổi chia hết 16.

3.5 Cho 5 điểm trên mặt phẳng có toạ độ nguyên. Chứng minh rằng tồn tại hai điểm có trung độ nguyên.

3.6 Một đội bóng thi đấu liên tục trong 30 ngày của tháng 9, mỗi ngày ít nhất 1 trận và tổng số trận đấu không quá 45 trận. Chứng minh rằng tìm được số ngày liên tục nào đó trong tháng 9 mà đội chơi đúng 14 trận.

3.7 Cho X={0..15}. Chứng tỏ rằng nếu S là một tập con gồm 9 phần tử của X thì có ít nhất 2 phần tử của S có tổng bằng 15.

3.8 Trong mặt phẳng cho 6 điểm phân biệt nối nhau từng đôi một bởi các đoạn thẳng màu xanh hoặc đỏ. Chứng tỏ rằng có 3 điểm nối nhau bởi các đoạn thẳng cùng màu.

3.9 Một võ sĩ quyền anh thi đấu giành chức vô địch trong 75 giờ. Mỗi giờ đấu ít nhất một trận, nhưng toàn bộ không quá 125 trận. Chứng tỏ rằng có những giờ liên tiếp đã đấu 24 trận.

3.10 Trong một mặt phẳng có 17 điểm phân biệt được nối với nhau từng đôi một bởi các đoạn thẳng màu xanh, hoặc màu đỏ, hoặc màu vàng. Chứng minh rằng luôn tồn tại ba điểm nối với nhau bởi các đoạn thẳng cùng màu

Chương 4. BÀI TOÁN LIỆT KÊ

Tóm tắt chương

Bài toán kiệt kê quan tâm đến tất cả cấu hình có thể có được, vì thế lời giải của nó cần được biểu diễn dưới dạng "vét cạn" tất cả các cấu hình. Máy tính sẽ giải quyết lời giải theo thuật toán đã nêu. Chương này, chủ yếu tập trung đến hai phương pháp để liệt kê đó là phương pháp sinh và phương phương pháp quay lui. Các bài toán liệt kê sẽ làm nền cho nhiều bài toán khác.

4.1 Phát biểu bài toán

Giả sử có một bảng video có thể ghi được C giây. Ta có n phim video với thời gian tương ứng là $t_1, t_2, \ldots, t_n$. Ta phải chọn k phim $i_1, i_2, \ldots, i_k$ sao cho tổng.

$$\sum_{i \in \{i_1, i_2, \ldots, i_k\}} t_i$$

không vượt quá C và lớn nhất. Một cách giải chân phương là liệt kê tất cả các tập con $\{i_1, i_2, \ldots, i_k\} \subset \{1, 2, \ldots, n\}$ và chọn tập có tổng trên là lớn nhất không vượt quá C.

Vì vậy, trong nhiều trường hợp, khi không có thuật toán hiệu quả để giải quyết những bài toán như trên, thì phương pháp liệt kê với sự trợ giúp của máy tính vẫn là giải pháp khả dĩ.

Như vậy, bài toán liệt kê được xác định thuật toán xây dựng lần lượt cấu hình quan tâm. Thuật toán cần đảm bảo các yêu cầu sau:

- Không lặp lại cấu hình
- Không bỏ sót cấu hình

4.2 Phương pháp sinh

4.2.1 Thứ tự từ điển

Cho $\alpha = s_1 s_2 \ldots s_p$ và $\beta = t_1 t_2 t_q$ là các dãy số hoặc ký tự. Ta nói rằng α nhỏ hơn β (theo kiểu từ điển), ký hiệu $\alpha < \beta$, nếu hoặc

(i) $p < q$ và $s_i = t_i$ với mọi $i = 1,2,\ldots,p$

hoặc

(ii) Tồn tại $k \le \min\{p, q\}$ sao cho $s_i = t_i$ với mọi $i = 1, 2, \ldots, k-1$ và $s_k < t_k$

Ví dụ 1.

BANANA < BANDIT

AN < ANH

"1324" < "1324567"

"134612398" < "1352"

Trong các thuật toán liệt kê tiếp theo sẽ liệt kê các cấu hình theo thứ tự từ điển.

4.2.2 Phương pháp sinh

Để có thể liệt kê tất cả cấu hình ta cần sắp xếp các cấu hình theo thứ tự nào đó, sau đó liệt kê chúng theo thứ tự từ nhỏ đến lớn thì đảm bảo hai yêu cầu của bài toán liệt kê. Một loại thứ tự hay được dùng là thứ tự từ điển. Phương pháp tạo ra một cấu hình tổ hợp liền kề ngay sau một cấu hình đã có theo thứ tự từ điển. Đó chính là ý tưởng phương pháp sinh.

Thuật toán sinh tổng quát

Kí hiệu s là cấu hình hiện hành, s_0 là cấu hình đầu tiên (theo thứ tự từ điển).

Bước 1. Khởi tạo, gán s = s_0

Bước 2. Kết xuất s.

Bước 3. Kiểm tra tiêu chuẩn kết thúc.

Nếu s là cấu hình cuối cùng thì kết thúc, ngược lại sang bước 4.

Bước 4. Tìm cấu hình t đứng kề sau s theo thứ tự từ điển.

Gán s= t và quay lại bước 2.

- *Lưu ý*. Tuỳ theo bài toán cụ thể có thể gộp bước 3 và 4 thành 1 bước để tăng hiệu quả thuật toán. áp dụng thuật toán tổng quát cho các bài toán cụ thể, ta chỉ cần xác định cấu hình đầu tiên *s* và phương pháp tìm cấu hình *t* kế tiếp sau cấu hình *s*.

4.3 Các thuật toán về phương pháp sinh

4.3.1 Liệt kê dãy nhị phân

• ***Phát biểu bài toán***. Cho $n \in N$. Hãy liệt kê, theo thứ tự từ điển, tất cả các dãy nhị phân độ dài *n*, tức là các dãy $[b_1, ..., b_n]$, trong đó $b_i \in \{0, 1\}$ $\forall$ i=1, ..., n.

Số dãy nhị phân là 2^n và dãy đầu tiên $s_0 = [0, 0, ..., 0]$. Phương pháp tìm dãy kế tiếp như sau.

- *Phương pháp tìm dãy kế tiếp*

Cho dãy nhị phân $s = [s_1, s_2, ..., s_n]$, ta tìm dãy tiếp theo $t = [t_1, t_2, ..., t_n]$. Xuất phát từ s_n, đi từ phải sang trái, ta tìm phần tử đầu tiên s_m, $1 \le m \le n$, thoả $s_m = 0$.

Nếu không tìm thấy thì s = [1, 1, ..., 1] là dãy cuối cùng, kết thúc tìm kiếm. Nếu tìm thấy ta xây dựng dãy t = [t_1, t_2, ..., t_n] như sau:

$t_i = s_i$ với mọi $i = 1,2,...,m-1$

$t_m = 1$

$t_i = 0$ với mọi $i = m+1,m+2,...,n$

Từ đó suy ra thuật toán liệt kê dãy nhị phân sau.

- *Thuật toán*

- Đầu vào: n

- Đầu ra: Danh sách tất cả dãy nhị phân độ dài n theo thứ tự từ điển tăng dần.

- Các bước:

1. Khởi tạo dãy xuất phát:

Gán $s_i = 0$ với mọi $i = 1,2,..., n$.

2. Kết xuất s.

3. Tìm m thoả

$m = \max\{i \mid s_i = 0\}$

Nếu không tìm thấy thì s = [1, 1, ..., 1] là dãy cuối cùng, *kết thúc*.

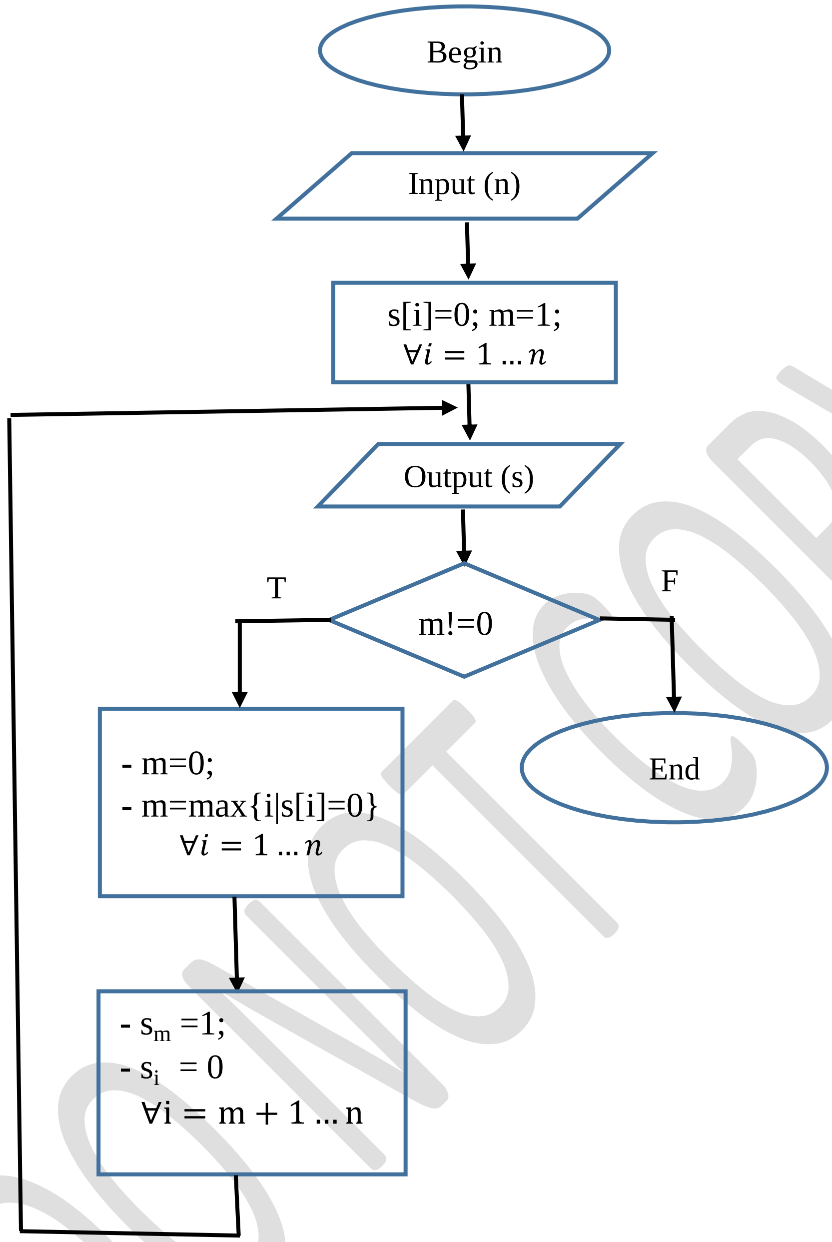
Nếu tìm thấy ta đặt

$s_m = 1$

$s_i = 0$ với mọi $i = m+1, m+2, ..., n$

Quay lại bước 2.

Thuật toán bằng sơ đồ khối



Hình 4.1 Sơ đồ khối thuật toán liệt kê dãy kế tiếp

Thuật toán trên được viết bằng ngôn ngữ C như sau

- Lệnh 4 là nhập dữ liệu đầu vào
- Lệnh 5 khởi tạo ở bước 1,
- Lệnh 6 mượn biến m và khởi tạo m=1 để làm điều kiện thoát
- Lệnh 7 (while (m!=0)) là điều kiện để lặp lại bước 2 và bước 3: ban đầu m=1, như vậy các lệnh trong while thực hiện ít nhất 1 lần, trong quá trình thực hiện m sẽ gán bằng 0 (lệnh 12) và sẽ tìm được m mới từ lệnh 13 đến lệnh 18. Trong trường hợp không tìm được m mới nghĩa là s là cấu hình cuối cùng thì m sẽ bằng 0 và sẽ thoát khỏi while
- Lệnh 9, 10 là kết xuất s bước 2

- Lệnh 13-18 tính lại m ở bước 3

- Lệnh 19 gán s[m]=1 ở bước 3

- Lệnh 20, 21 gán lại s[k]=0 k=m+1,…,n ở bước 3

```
1.  main()
2.  {
3.      int n;
4.      cout<<"n="; cin>>n;
5.      for (int i=1;i<=n;i++) s[i]=0;
6.      int m=1;
7.      while (m!=0)
8.      {
9.      for (int i=1;i<=n;i++)
10.             cout<<s[i]<<" ";
11.     cout<<endl;
12.      m=0;
13.     for (int i=n;i>=1;i--)
14.             if (s[i]==0)
15.                     {
16.                     m=i;
17.                     break;
18.                     }
19.     s[m]=1;
20.     for (int k=m+1;k<=n;k++)
21.             s[k]=0;
22.     }
23. }
```

Kết quả với n=4
0 0 0 0
0 0 0 1
0 0 1 0
0 0 1 1
0 1 0 0
0 1 0 1
0 1 1 0
0 1 1 1
1 0 0 0
1 0 0 1
1 0 1 0
1 0 1 1
1 1 0 0
1 1 0 1
1 1 1 0
1 1 1 1
Số dãy nhị phân 16

4.3.2 Liệt kê tổ hợp chập r từ n phần tử

Xét bài toán liệt kê tất cả tổ hợp chập *r* từ *n* phần tử $\{1,2,...,n\}$. Vì tổ hợp là tập hợp các phần tử, không kể thứ tự, nên ta quy ước mỗi tổ hợp sẽ được biểu diễn bằng danh sách [s_1, s_2 ,... , s_r] với $s_1 < s_2 <... < s_r$. Như vậy, theo thứ tự từ điển, tổ hợp đầu tiên là [*1, 2,..., r*] và tổ hợp cuối cùng là [*n–r+1, n–r+2,..., n*].

Xét tổ hợp chập 5 của [1,2,3,4,5,6,7]. Tổ hợp đầu là [1,2,3,4,5]. Tổ hợp tiếp theo là [1,2,3,4,6] và [1,2,3,4,7]. Tổ hợp liền sau sẽ là [1,2,3,5,6] và [1,2,3,5,7]. Tổ hợp cuối là [3,4,5,6,7].

Tìm tổ hợp đi sau [1,3,4,6,7]. Không dãy 5 số nào bắt đầu bằng 1,3,4 vượt qua [1,3,4,6,7]. Vì vậy dãy phải tìm bắt buộc bắt đầu bằng 1,3,5. Vì vậy tổ hợp tiếp theo là [1,3,5,6,7].

- *Phương pháp tìm tổ hợp kế tiếp:*

Cho tổ hợp $\alpha = \{s_1 , s_2 ,... ,s_r \}$, ta tìm tổ hợp tiếp theo $\beta = \{t_1 , t_2 ,... ,t_r\}$. Trước hết ta nhận xét thấy rằng thành phần thứ *i* trong tổ hợp không thể vượt quá $n-r+i$. Giá trị này gọi là trị cực đại của thành phần thứ *i* . Ta tìm

$$m = \max\{i \mid s_i < n-r+i \}$$

Sau đó ta đặt

$t_i = s_i$ với mọi $i = 1, 2,..., m-1$

$t_m = s_m + 1$

$t_m+i = s_m + i + 1$ với mọi $i =1, 2, ..., r-m$

- *Thuật toán:*

- Đầu vào: *r, n*

- Đầu ra : Danh sách tất cả tổ hợp chập r của [*1,2,...,n*] theo thứ tự từ điển tăng dần.

1. Khởi tạo dãy xuất phát :

Gán $s_i := i$ với mọi $i = 1,2,...,r$.

2. Kết xuất *s*.

3. Nếu thoả điều kiện kết thúc $s_1 = n - r+1$

thuật toán kết thúc.

Ngược lại sang bước 4.

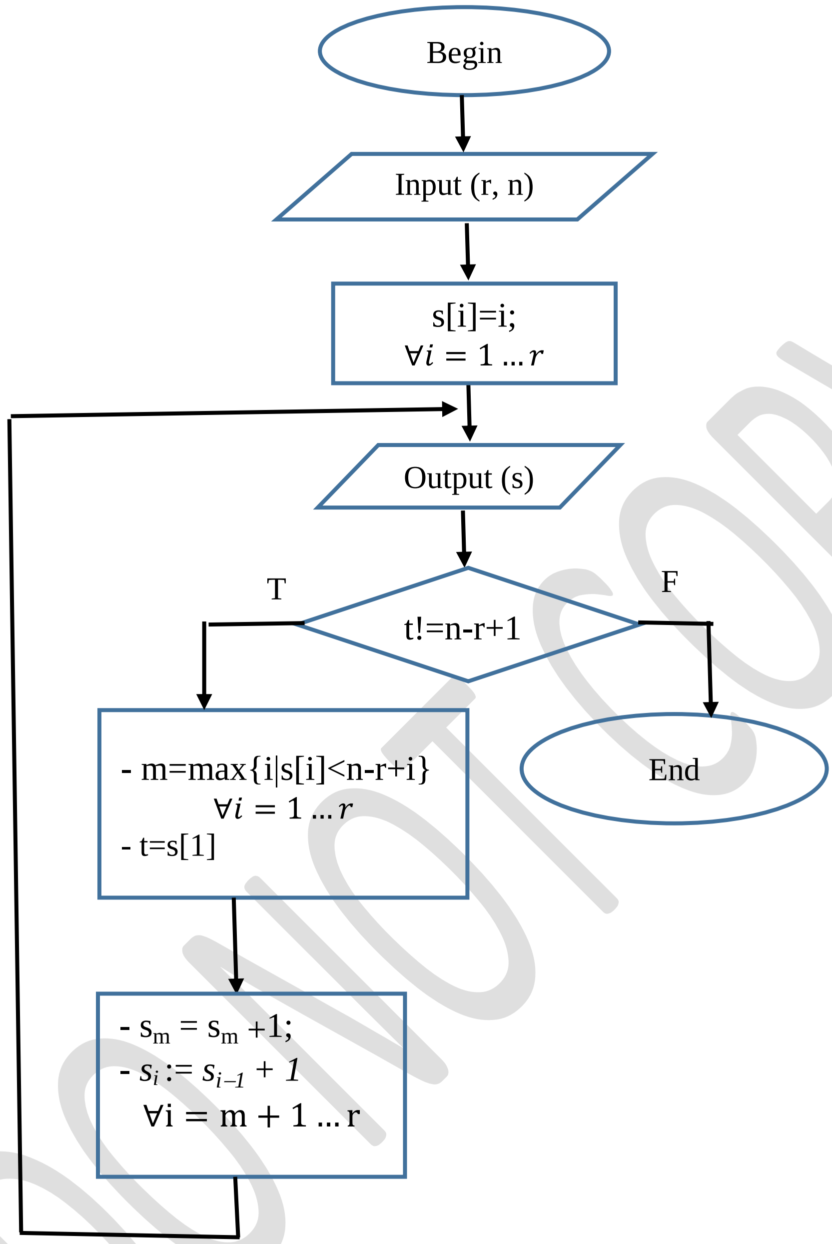
4. Tìm *m* thoả

$$m = \max\{i \mid s_i < n-r+i \}$$

Đặt $s_m := s_m + 1$

$s_i := s_{i-1} + 1$ với mọi $i = m+1,m+2,...,r$Quay lại bước 2.

Thuật toán bằng sơ đồ khối



Hình 4.2 Sơ đồ khối liệt kê tổ hợp chập r của n phần tử

Thuật toán trên được viết bằng ngôn ngữ C như sau

```
int dem=0;
int t=0;
main()
	{
		cout<<"nhap n="; cin>>n;
		cout<<"nhap r="; cin>>r;
		for(int i=1;i<=r;i++)
			s[i]=i;
	while(t!=n-r+1)// Kiểm tra điều kiện
```

```
10.	{
11.		dem++;
12.		for(int i=1;i<=r;i++)
13.			cout<<s[i]<<" ";
14.		cout<<endl;
15.		for(int i=r;i>=1;i--)
16.			if(s[i]<n-r+i)
17.				{
18.					m=i;
19.					break;
20.				}
21.		t=s[1];
22.		s[m]=s[m]+1;
23.		for(int i=m+1;i<=r;i++)
24.			s[i]=s[i-1]+1;
25.	}//end whlie
26.	cout<<" So day="<<dem;
27.	}
```

- Lệnh 7,8 khởi tạo bước 1

- Lệnh 12, 13 bước 2 kết xuất s

- Lệnh 9 kiểm tra điều kiện lặp ở bước 3

- Lệnh 15-20 tìm m ở bước 4

- Lệnh 21 gán lại giá trị t để kiểm tra điều kiện lặp

- Lệnh 22- 24 gán lại các giá trị của s ở bước 4

Kết quả n=7, r=5
1 2 3 4 5
1 2 3 4 6
1 2 3 4 7
1 2 3 5 6
1 2 3 5 7
1 2 3 6 7
1 2 4 5 6
1 2 4 5 7
1 2 4 6 7
1 2 5 6 7
1 3 4 5 6
1 3 4 5 7
1 3 4 6 7
1 3 5 6 7
1 4 5 6 7
2 3 4 5 6
2 3 4 5 7
2 3 4 6 7
2 3 5 6 7
2 4 5 6 7
3 4 5 6 7

Số dãy tổ hợp=21

4.3.3 Liệt kê hoán vị

Xét bài toán liệt kê tất cả hoán vị của n phần tử {*1,2,...,n*}. Mỗi hoán vị sẽ được biểu diễn như là dãy s_1, s_2 ,... , s_n. Như vậy hoán vị đầu tiên là [*1,2,...,n*] và hoán vị cuối cùng là [*n,n-1,...,1*].

Giả sử ta phải tìm hoán vị $t = [t_1 , t_2 , t_3 , t_4 , t_5 , t_6]$ của {1, 2, 3, 4, 5, 6} tiếp theo sau hoán vị s = [1, 6, 3, 5, 4, 2].

Liệu 4 số đầu của *t* có thể là 1, 6, 3, 5 được không? Không! Bởi vì trong các hoán vị bắt đầu bằng 1, 6, 3, 5 (chỉ có s và [1, 6, 3, 5, 2, 4]) thì s có thứ tự lớn nhất.

Liệu 3 số đầu của t có thể là 1, 6, 3 được không? Không! Thật vậy, các hoán vị bắt đầu bằng 1, 6, 3 gồm: s=[1, 6, 3, 5, 4, 2], [1, 6, 3, 5, 2, 4], [1, 6, 3, 4, 2, 5], [1, 6, 3, 4, 5, 2], [1, 6, 3, 2, 5, 4], [1, 6, 3, 2, 4, 5]

Và hiển nhiên là trong các hoán vị đó s là hoán vị có số thứ tự lớn nhất.

Lý do việc t không thể bắt đầu bởi 1, 6, 3, 5 hoặc 1, 6, 3 là vì trong cả hai trường hợp này các số còn lại của s tạo thành dãy giảm dần.

Như vậy xuất phát từ bên phải ta tìm số đầu tiên *d* nhỏ hơn số bên phải nó. Trong ví dụ này đó là số 3. Và t bắt đầu bằng 1, 6. Số tiếp theo của t phải lớn hơn 3. Vì ta muốn t là hoán vị nhỏ nhất trong các hoán vị lớn hơn s, số tiếp theo phải là số nhỏ nhất trong các số còn lại (tức trừ 1 và 6) lớn hơn 3. Số đó phải là 4. Ba số còn lại phải tăng dần. Như vậy ta có t = [1, 6, 4, 2, 3, 5].

- *Phương pháp tìm hoán vị kế tiếp*:

Cho hoán vị $s = [s_1 , s_2 ,... ,s_n]$, ta tìm hoán vị tiếp theo $t = [t_1 , t_2 ,... ,t_n]$. Đi từ phải sang trái ta tìm phần tử đầu tiên s_m thoả $s_m < s_{m+1}$. Sau đó tìm chỉ số k (k > m) lớn nhất thoả $s_m < s_k$.

Sau đó ta đặt

$t_i = s_i$ với mọi i = 1,2,...,m–1

$t_m = s_k$

n–m phần tử tiếp theo là các số còn lại sắp xếp theo thứ tự tăng dần.

- *Thuật toán:*

- Đầu vào: n

- Đầu ra : Danh sách tất cả hoán vị của {1,2,...,n} theo thứ tự tăng dần.

1. Khởi tạo dãy xuất phát :

Gán s_i = i với mọi i = 1, 2,..., n.

2. Kết xuất s

3. Nếu thoả điều kiện s = [n, n−1,..., 2,1] thuật toán kết thúc. Ngược lại sang bước 4.

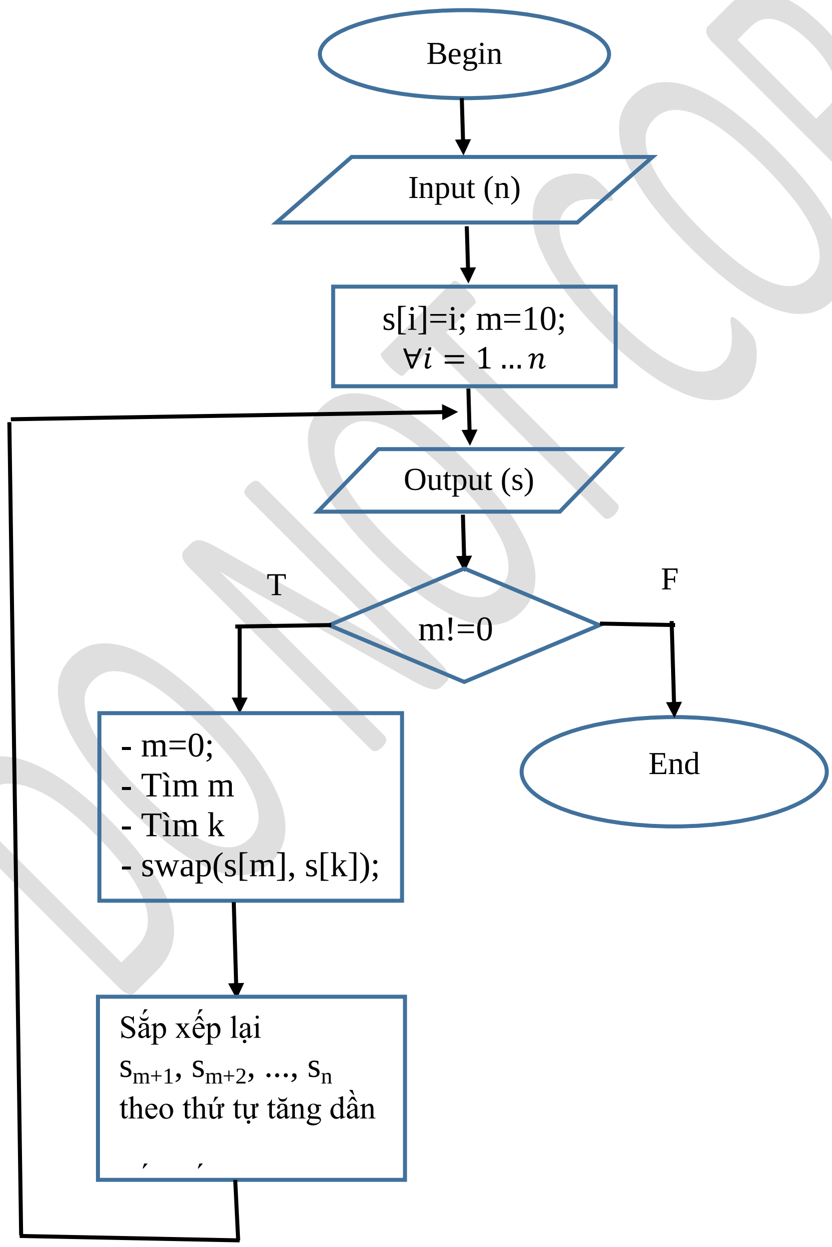
4. Tìm m là chỉ số lớn nhất thoả $s_m < s_{m+1}$.

Tìm k là chỉ số lớn nhất thoả $s_m < s_k$.

Hoán vị s_m và s_k

Sắp xếp lại s_{m+1}, s_{m+2}, ..., s_n theo thứ tự tăng dần (lưu ý rằng dãy s_{m+1}, s_{m+2}, ..., s_n đã được sắp xếp giảm dần). Quay lại bước 2.

Thuật toán bằng sơ đồ khối



Hình 4.3 Sơ đồ khối liệt kê hoán vị

Thuật toán trên được viết bằng ngôn ngữ C như sau

- Lệnh 5, 6, Khởi tạo s ở bước 1
- Lệnh 7 khởi tạo m để làm điều kiện thoát
- Lệnh 8 điều kiện lặp ở bước 3
- Lệnh 10, 11 kết xuất s ở bước 2
- Lệnh 14 gán lại giá trị m để thoát vòng lặp
- Lệnh 15- 20 tìm m ở bước 4
- Lệnh 22-27 tìm k ở bước 4
- Lệnh 28 hoán vị giữa s[m] và s[k] ở bước 4
- Lệnh 29-34 sắp xếp lại s_{m+1}, s_{m+2}, ..., s_n theo thứ tự tăng dần ở bước 4

```
1. main()
2. {
3.      cout<<"Nhap n=";cin>>n;
4.      //buoc 1
5.      for(int i=1; i<=n;i++)
6.              s[i]=i;
7.      int m=10;
8.      while(m!=0)
9.      {//begin while
10.     dem++;
11.     for(int i=1; i<=n;i++)
12              cout<<s[i]<<" ";
13.     cout<<endl;
14.     m=0;
15.     for(int i=n; i>=1;i--)
16.             if(s[i]>s[i-1])
17.             {
18.                     m=i-1;
19.                     break;
20.             }
21.     cout<<m;
22.     for(int i=n; i>=1;i--)
23.             if(s[m]<s[i])
24.             {
25.                     k=i;
26.                     break;
27.             }
28.     swap(s[m], s[k]);
29.     int j=n;
30.     for(int i=1;i<j;i++)
31.                     {
32.                             swap(s[i], s[j]);
33.                             j--;
```

```
                    }
    }//end while
    cout<<"So day Hoan vi "<<n;
}
```

Kết quả n=4
1 2 3 4
1 2 4 3
1 3 2 4
1 3 4 2
1 4 2 3
1 4 3 2
2 1 3 4
2 1 4 3
2 3 1 4
2 3 4 1
2 4 1 3
2 4 3 1
3 1 2 4
3 1 4 2
3 2 1 4
3 2 4 1
3 4 1 2
3 4 2 1
4 1 2 3
4 1 3 2
4 2 1 3
4 2 3 1
4 3 1 2
4 3 2 1

Số dãy hoán vị=24

4.3.4 Kiệt kê dãy tập con

Cho tập con A={x_1, x_2,….,x_n} liệt kê tất cả các tập con của tập A

Giải

Số dãy tập con của A là 2^n đây cũng chính là số dãy nhị phân có độ dài n

Ta gọi dãy nhị phân là {$b_1b_2…b_n$} với b_i=0 hoặc 1 (i=1,…,n). Suy ra dãy tập con có được từ dãy nhị phân bằng cách nếu b_i=1 thì ta thay b_i bằng x_i còn b_i=0 thì x_i không xuất hiện.

Dựa vào thuật toán sinh dãy nhị phân ta xây dựng được thuật toán kiệt kê tất cả các tập con của tập A.

```
#include <iostream>
using namespace std;
int s[100];
char A[100];
```

```
main()
{
        int n;
        cout<<"n=";cin>>n;
        cout<<"Nhap tap co "<<n<<" phan tu A=";
        for (int i=1;i<=n;i++)
                cin>>A[i];
        cout<<"Cac tap con cua tap A nhu sau:"<<endl;
        for (int i=1;i<=n;i++)
                s[i]=0;
        int t=1; int d=0;
        while (t!=0)
        {
                d++;
                cout<<"{";
        for (int i=1;i<=n;i++)
                if (s[i]!=0) cout<<A[i]<<" ";
                cout<<"}";
        cout<<endl;
        int j=0;
        for (int i=n;i>=1;i--)
                if (s[i]==0)
                        {
                        j=i;
                        break;
                        }
        s[j]=1;
        for (int k=j+1;k<=n;k++)
                s[k]=0;
        t=j;
        }
        cout<<"Tong cac tap con cua A ="<<d;
}
```

n=5

Nhập tập có 5 phần tử A={a, b, c, d, e}

Các tập con của tập A như sau:

{}

{e }

{d }

{d e }

{c }

{c e }

{c d }

{c d e }

{b }

{b e }

{b d }

{b d e }

{b c }

{b c e }

{b c d }

{b c d e }

{a }

{a e }

{a d }

{a d e }

{a c }

{a c e }

{a c d }

{a c d e }

{a b }

{a b e }

{a b d }

{a b d e }

{a b c }

{a b c e }

{a b c d }

{a b c d e }

Tổng các tập con của A =32

4.3.5 Liệt kê dãy bị chặn

Phép thế, nghịch thế

Khái niệm về *phép thế*, *chuyển trí* và *nghịch thế* như sau :

Giả sử tập hợp $X_n= \{1, 2, 3,..., n\}$, ($n \geq 1$). Một song ánh $\sigma : X_n \rightarrow X_n$ được gọi là một *phép thế* trên tập X_n. Song ánh đồng nhất được gọi là phép thế đồng nhất.

Một phép thế τ trên tập X_n được gọi là một *chuyển trí* hai phần tử i, j thuộc X_n *nếu*

$\tau(i) = j$, $\tau(j) = i$ và $\tau(k) = k$, với mọi $k \in X_n$,$k \neq i, j$. Nói một cách đơn giản, một chuyển trí chỉ hoán vị hai phần tử nào đó của X_n , còn giữ nguyên mọi phần tử khác. Tập hợp tất cả các phép thế trên tập X_n được kí hiệu bởi S_n

Phép thế $\sigma : X_n \to X_n$ được biểu diễn như sau:

$$\sigma = \begin{pmatrix} 1 & 2 & 3 & \dots\dots\dots\dots.n \\ \sigma(1) & \sigma(2) & \sigma(3) & \dots\dots\dots.\sigma(n) \end{pmatrix}$$

trong đó s(i) là ảnh của phần tử i $\in X_n$ được viết ở dòng dưới, trong cùng một cột với i.

Ví dụ 1.

$$\sigma = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 3 & 2 & 4 & 1 \end{pmatrix}$$

là phép thế trên tập $X_4 = \{1, 2, 3, 4\}$ xác định bởi: $\sigma(1) = 3$, $\sigma(2) = 2$, $\sigma(3) = 4$, $\sigma(4) = 1$.

$\tau = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 1 & 4 & 3 & 2 \end{pmatrix}$là một chuyển trí hoán vị hai số 2 và 4. Nó được viết gọn là $\tau = (2, 4)$.

Như vậy: Ảnh của các phần tử của tập X_n qua mỗi phép thế cho ta một hoán vị trên tập X_n. Nguợc lại, mỗi hoán vị lại xác định một phép thế, (chẳng hạn, hoán vị (3, 4, 1, 2) xác định phép thế $\mu = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 3 & 4 & 1 & 2 \end{pmatrix}$ trên tập X_4). Vì thế số các phép thế trên tập X_n bằng số các hoán vị trên tập ấy; nghĩa là bằng n!. Như vậy, tập S_n có n! phần tử.

Ví dụ 2. S_3 có 3! = 6 phần tử. Đó là những phép thế sau:

$$\sigma_1 = \begin{pmatrix} 1 & 2 & 3 \\ 1 & 2 & 3 \end{pmatrix}, \sigma_2 = \begin{pmatrix} 1 & 2 & 3 \\ 2 & 3 & 1 \end{pmatrix}, \sigma_3 = \begin{pmatrix} 1 & 2 & 3 \\ 3 & 1 & 2 \end{pmatrix}$$

$$\tau_1 = \begin{pmatrix} 1 & 2 & 3 \\ 1 & 3 & 2 \end{pmatrix}, \tau_2 = \begin{pmatrix} 1 & 2 & 3 \\ 3 & 2 & 1 \end{pmatrix}, \tau_3 = \begin{pmatrix} 1 & 2 & 3 \\ 2 & 1 & 3 \end{pmatrix}$$

Giả sử một phép thế trên tập X_n. Với $i,j \in X_n$, $i \neq j$, ta nói cặp $(\sigma(i), \sigma(j))$ là một nghịch thế của σ nếu $i < j$ nhưng $\sigma(i) > \sigma(j)$.

Trên X_3, phép thế $\sigma_2 = \begin{pmatrix} 1 & 2 & 3 \\ 2 & 3 & 1 \end{pmatrix}$có *2* nghịch thế là: (2, 1), (3, 1). Phép thế $\tau_2 = \begin{pmatrix} 1 & 2 & 3 \\ 3 & 2 & 1 \end{pmatrix}$ có 3 nghịch thế là: (3, 2), (3, 1), (2, 1).

Trên tập X_n có n! hoán vị và có n! phép thế. Do đó ta gọi dãy nghịch thế trên mỗi phép thế như sau: xem phần tử 1 có bao nhiêu nghịch thế trong phép thế rồi lấy giá trị đó gán cho dãy nghịch thế, Xem phần tử 2 có bao nhiêu nghịch thế trong phép thế rồi lấy giá trị nó gán cho dãy nghịch thế. Cứ tiếp tục như vậy cho phần tử n. Sau đây là dãy nghịch thế của các phép thế với n=3.

Bảng 4.1 Phép thế, dãy nghịch thế và dãy nghịch thế ngược với n=3

STT	Hoán vị (Ảnh Phép thế)	Dãy nghịch thế	Dãy nghịch thế ngược
1	1 2 3	0 0 0	0 0 0
2	2 1 3	1 0 0	0 0 1
3	2 3 1	2 0 0	0 0 2
4	1 3 2	0 1 0	0 1 0
5	3 1 2	1 1 0	0 1 1
6	3 2 1	2 1 0	0 1 2

Dựa vào bảng 4.1, ta nhận thấy rằng ứng với 1 hoán vị luôn luôn tìm được một nghịch thế ngược và ngược lại ứng với một nghịch thế ngược ta luôn luôn tìm được một hoán vị. Như vậy, bài toán đặt ra là thay vì đi tìm hoán vị của n phần tử theo các phương khác nhau như: phương pháp thứ tự từ điển, phương pháp từ điển ngược, phương pháp thêm bớt một phần tử, phương pháp đổi chỗ. Ta đi tìm hoán vị bằng cách tìm dãy nghịch thế ngược. Việc tìm dãy nghịch thế ngược là đơn giản hơn và tốn ít thời gian hơn thuật toán tuần tự. Đồng thời việc phân chia dãy nghịch thế ngược ra thành nhiều đoạn nghiệm con là đơn giản và khả thi hơn khi phân chia các hoán vị thành các đoạn nghiệm con. Dưới đây, ta sinh dãy nghịch thế ngược được bắt đầu bằng dãy 0 0 0 và kết thúc bằng 0 1 2 ứng với n=3. Còn với n=4 thì dãy nghịch thế đầu là 0 0 0 0 và kết thúc bằng 0 1 2 3.

Thuật toán sinh dãy bị chặn

Ký hiệu Z là tập các số nguyên. Cho n là một sô nguyên dương nào đó, giả sử p và q là 2 dãy số nguyên độ dài n và ký hiệu như sau:

$p=(p_1p_2\ldots p_n)$, $q=(q_1q_2\ldots q_n)|\ p_i, q_i \in Z, \forall i \in 1,\ldots,n$

Ta có định nghĩa sau:

1) p $\leq$q khi và chỉ khi $p_i \leq q_i\ \forall i \in 1,\ldots,n$

2) p < q khi và chỉ khi $\exists j \in (1 \ldots n)$: $p_j<q_j$ và $p_i \leq q_i : \forall i \in 1,\ldots,n)$ và $i \neq j$

Bài toán dãy bị chặn được phát biểu như sau:

Cho hai dãy số nguyên *s* và *g* có độ dài *n*, sao cho *s*<*g*, hãy tìm tất cả các dãy số t độ dài n sao cho $s \leq t \leq g$

Giả sử $s=(s_1s_2\ldots s_n)$ và $g=(g_1g_2\ldots g_n)$, hai dãy này được gọi là các dãy biên. Các dãy cần tìm $t=(t_1t_2...t_n)$ phải thỏa mãn :

$$t_i \in Z \wedge s_i \le t_i \le g_i \forall i \in (1 \dots n) \qquad (3)$$

Cho s=(2 1 0 3), g=(3 1 1 5) là hai dãy biên, các dãy số nguyên t thỏa $s \le t \le g$ và cho s'=(0 0 0), g'=(0 1 2) là hai dãy biên các dãy số nguyên t' thỏa $s' \le t' \le g'$. Như vậy, t và t' được sắp xếp theo thứ tự từ điển tăng dần như trong bảng sau:

Bảng 4.2 Biểu diễn các dãy bị chặn t và $\boldsymbol{t'}$

STT	**Dãy bị chặn t**	**STT**	**Dãy bị chặn t**	**STT**	**Dãy bị chặn t'**
1	2 1 0 3	7	3 1 0 3	1	0 0 0
2	2 1 0 4	8	3 1 0 4	2	0 0 1
3	2 1 0 5	9	3 1 0 5	3	0 0 2
4	2 1 1 3	10	3 1 1 3	4	0 1 0
5	2 1 1 4	11	3 1 1 4	5	0 1 1
6	2 1 1 5	12	3 1 1 5	6	0 1 2

Định lý 1. Cho hai dãy biên s=(0 …0) (có n phần tử 0) và g=(0 1 2 …n-1). Các dãy số bị chặn t thỏa $s \le t \le g$ là dãy nghịch thế ngược của tập X_n= {1, 2, 3,..., n}, (n ≥ 1). và số dãy t bằng n! và nghịch thế ngược s=(0 …0) tương ứng với hoán vị (1 2 …n) và nghịch thế ngược g=(0 1 2 …n-1) tương ứng với hoán vị (n n-1 … 1).

Chứng minh:

Cho X_n= {1, 2, 3,..., n}, (n ≥ 1). Hoán vị đầu tiên là (1 2 3 …n). Ta nhận thấy rằng trước 1 không có gặp ngịch thế nào nên số nghịch thế của 1 bằng 0, trước 2 không có gặp ngịch thế nào nên số nghịch thế của 2 bằng 0, tương tự như vậy số nghịch thế của 3, 4,…,n đều bằng 0. Suy ra, dãy nghịch thế ngược ứng với hoán vị đầu tiên là (0…0) (có n phần tử 0).

Hoán vị cuối cùng của X_n theo thứ tự từ điển tăng dần là (n n-1….1). Ta nhận thấy rằng trước n không có gặp ngịch thế nào nên số nghịch thế của n bằng 0, trước n-1 có 1 gặp ngịch thế nên số nghịch thế của n-1 bằng 1, tương tự như vậy ta có số nghịch thế của (n-2 n-3…1) tăng dần lên một đơn vị. (Trong đó số cặp nghịch thế của 1 là n-1). Suy ra, dãy nghịch thế ứng với hoán vị (n n-1 …1) là (n-1…0). Từ đó ta tiếp tục suy ra dãy nghịch thế ngược ứng với hoán vị (n n-1 … 1) là 0 1 …n-1.

Mà từ hoán vị (1 2 …n) đến (n n-1 … 1) có n! hoán vị nên từ dãy nghịch thế ngược (0…0) đến (0 1 … n-1) cũng có n! dãy bị chặn t. □

Định lý 2. Cho s=($s_1 s_2 \dots s_n$) và g=($g_1 g_2 \dots g_n$), hai dãy biên. Các dãy t=($t_1 t_2 \dots t_n$) là dãy bị chặn. gọi C là số lượng các dãy t. Khi đó ta có:

$$C=\prod_{i=1}^{n}(g_i - s_i + 1) \qquad (4)$$

Chứng minh:

Ta có: t_1 có thể chọn một trong các giá trị s_1, s_1+1, s_1+2, …, g_1. Như vậy, thành phần t_1 có g_1-s_1+1 cách chọn. Tương tự như vậy, thành phần t_2 có g_2-s_2+1 cách chọn, thành phần n có g_n-s_n+1 cách chọn.

Vậy theo nguyên lý nhân số lượng các dãy của t là $C=(g_1-s_1+1)x(g_2-s_2+1)x\ldots x(g_n-s_n+1)= \prod_{i=1}^{n}(g_i - s_i + 1)$ □

Sau đây là thuật toán sinh dãy bị chặn.

Từ dãy nhỏ nhất $s=(s_1s_2\ldots s_n)$, dãy lớn nhất là $g=(g_1g_2\ldots g_n)$, ta xây dựng một vòng lặp để sinh các dãy còn lại. Giả sử rằng $t=(t_1t_2...t_n)$ là một dãy nào đó thỏa mãn (3). Ta phải tìm dãy $t' = (t'_1 t'_2 \ldots t'_n)$ kế tiếp sau dãy *t* trong dãy đã được sắp xếp theo thứ tự từ điển.Theo thứ tự từ điển, dãy *t'* được kế thừa phần bên trái nhiều nhất có thể của dãy *t* từ vị trí 1 đến vị trí thứ *i*-1, với: $i = \max\{ j \mid t_j < g_j\}$

Khi đó, từ vị trí đầu tiên đến vi trí thứ i-1 không thay đổi: $t'_j = t_j, j = 1, \ldots, i-1$

Các vị trí còn lại được xác định như sau: $t'_i = t_i + 1$ và $t'_j = s_j, j = i+1,\ i+2 \ldots, n$

Bước sinh các dãy kết thúc khi tất cả các dãy số nguyên thỏa mãn (3), tức là sinh xong dãy kế tiếp là $g=(g_1g_2\ldots g_n)$. Khi đó thì: $i = 0$

Void sinh_day_bi _chan(s[], g[n])

```
{
Nhập n, s[i], g[i], i=1,…,n //s, g là hai biên. Tức là dãy nhỏ nhất và dãy lớn nhất
t[i]=s[i], i=1,…,n
do
    Print t[i], i=1,…,n
    i=n;
    while (t[i] ==g[i])
        {
            t[i]=s[i];
            i=i-1;
        }
    if (i>=1)  t[i]=t[i]+1;
while (i!=0)
}
```

4.4 Phương pháp quay lui

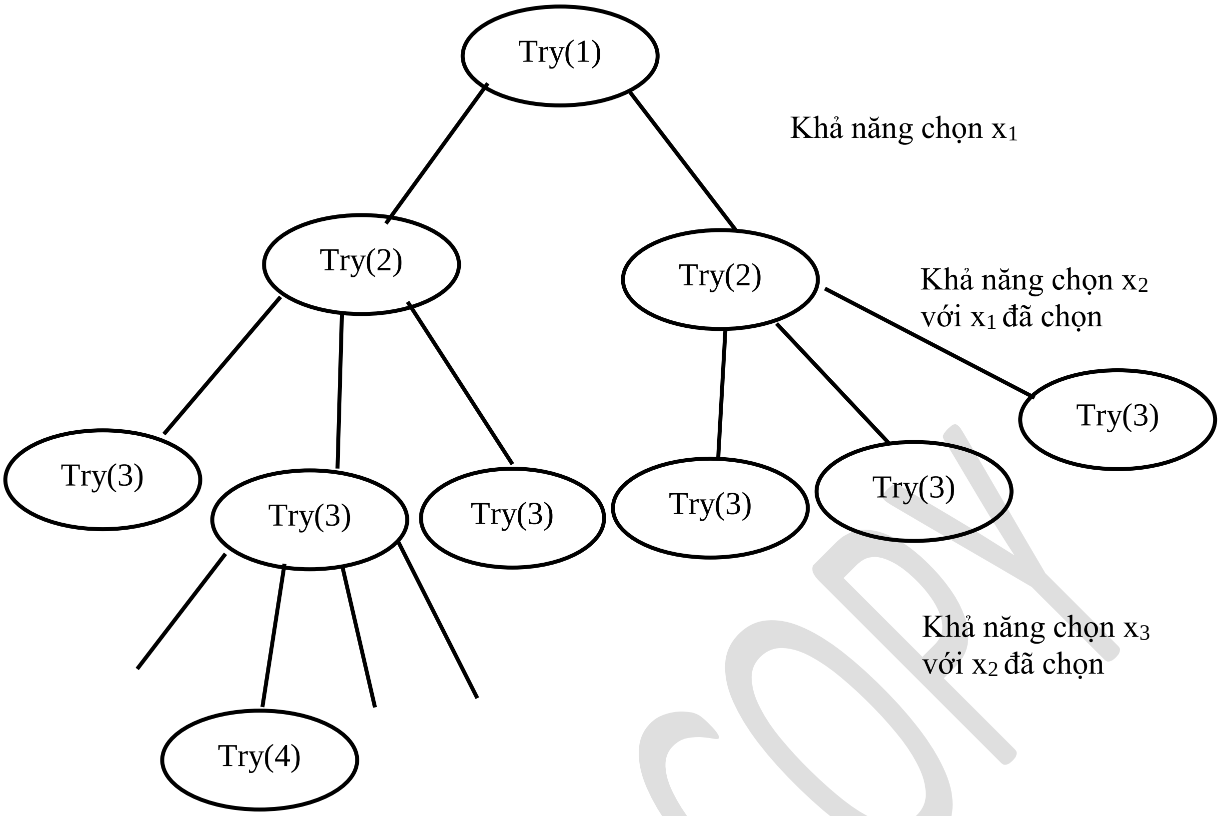
Ý tưởng chính của thuật toán này là xây dựng dần các thành phần của cấu hình bằng cách thử tất cả các khả năng. Giả thiết cấu hình cần được mô tả bằng một bộ gồm *n* thành phần $x_1, x_2, ..., x_n$. Giả sử đã xác định được i−1 thành phần $x_1, x_2, ..., x_{i-1}$. Ta xác định thành phần thứ i bằng cách duyệt tất cả khả năng có thể đề cử cho nó trong miền D_i (có thể đánh số các khả năng từ 1 đến n_i). Với mỗi khả năng j, kiểm tra xem khả năng j có chấp nhận được không. Có thể xảy ra 2 trường hợp:

- Nếu chấp nhận j thì xác định x_i theo j, sau đó nếu i = n, thì ta được một cấu hình, còn trái lại ta tiến hành xác định x_{i+1}.
- Nếu thử tất cả khả năng mà không khả năng nào được chấp nhận thì quay lại bước trước để xác định lại x_{i-1}.

Điều quan trọng của thuật toán là phải ghi nhớ tại mỗi bước đã đi qua những khả năng đã thử để tránh trùng lặp. Rõ ràng những thông tin này cần được lưu trữ theo cơ cấu ngăn xếp (stack - vào sau ra trước). Vì thế các bài toán đệ quy rất phù hợp với thuật toán này

```
void Try(int i);
    int j;
    {
        for j ∈ Di do
         if <chấp nhận j> then
         {
             <xác định xi theo j> [Ghi nhớ trạng thái mới]
             if i == n then <ghi nhận cấu hình>
             else  Try(i+1);
             [trả lại trạng thái cũ]
           }
    }
```

Quá trình tìm kiếm lời giải theo thuật toán quay lui có thể mô tả bằng cây sau:



Hình 4.4 Mô tả thuật toán quay lui

Phần quan trọng nhất trong hàm trên là việc đưa ra được một danh sách các khả năng đề cử và việc xác định giá trị của biểu thức logic <*chấp nhận j*>. Thông thường giá trị này ngoài việc phụ thuộc j còn phụ thuộc vào khả năng được chọn ở các bước trước. Vì thế cần ghi nhớ trạng thái mới của quá trình tìm kiếm sau khi <*xác định x_i theo j*> và trả lại trạng thái cũ sau lời gọi *Try(i+1)*. Các trạng thái này được ghi nhận nhờ một số biến toàn cục (global) gọi là biến trạng thái.

Sau khi xây dựng thủ tục đệ qui *Try*, chương trình chính giải bài toán liệt kê có dạng:

```
int main()
{
    Init; Try(1) ;
  }
```

trong đó *Init* là thủ tục khởi tạo các giá trị ban đầu (nhập các giá trị tham số của bài toán, khởi gán các biến trạng thái, biến đếm ...).

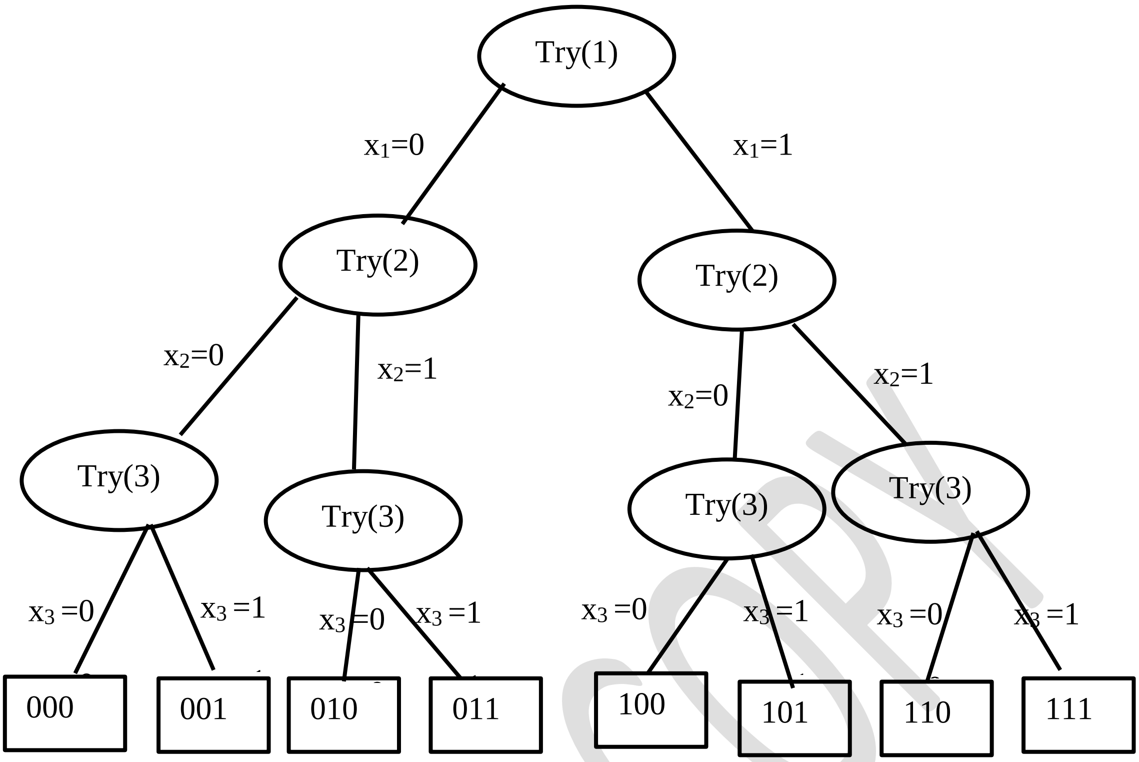
4.5 Các thuật toán về phương pháp quay lui

4.5.1 Liệt kê các dãy nhị phân có độ dài n

Ta biểu diễn dãy nhị phân dưới dạng x_1, x_2, ..., x_n, trong đó $x_i \in \{0,1\}$. Thủ tục *Try(i)* xác định $x_i \in \{0,1\}$. Các giá trị này được mặc nhiên chấp nhận mà không cần phải thoả mãn điều kiện gì (vì thế bài toán không cần biến trạng thái).

```
#include <iostream>
using namespace std;
int X[100]; int n; int dem=0;
void Try(int i)
{
	for(int j=0;j<=1;j++)
		{
			X[i]=j;
			if (i==n)
				{
					for (int k=1; k<=n;k++)
						cout<<X[k]<<" ";
						cout<<endl;
				}
			else Try(i+1);
		}
}
int main()
	{
		cout<<"Nhap n=";
		cin>>n;
		Try(1);
		return 0;
	}
```

Với n =3 ta có kết quả như sau:



Hình 4.5 Quay lui liệt kê dãy nhị phân n=3

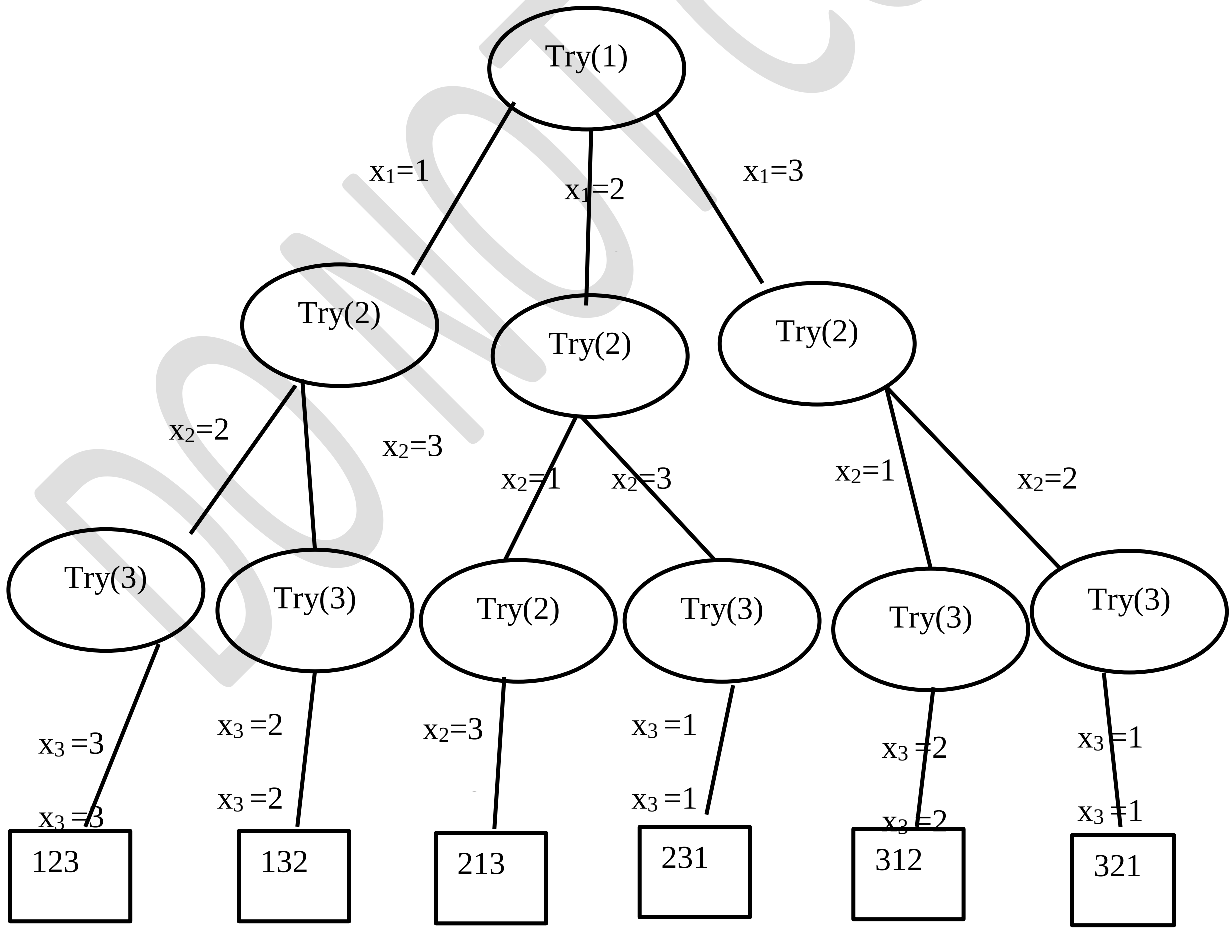
4.5.2 Liệt kê các hoán vị

Biểu diễn hoán vị dưới dạng x_1, x_2, ..., x_n, trong đó $x_i \in \{1,2,...,n\}$ và $x_i \neq x_j$ với $i \neq j$. Các giá trị j chạy từ 1 đến n được lần lượt đề cử cho x_i và j được chấp nhận nếu nó chưa được dùng. Vì thế, cần ghi nhớ đối với mỗi giá trị j xem nó đã được dùng hay chưa. Điều này được thực hiện nhờ dãy biến B[j] , trong đó B[j] = 0 nếu j chưa được dùng. Các biến này cần phải được gán giá trị *true*. Sau khi gán j cho x_i cần gán *1* cho b[j] và gán lại 0 khi thực hiện xong hàm *Try(i+1)*.

```
#include <iostream>
using namespace std;
int x[10];int B[10]={0};
int n;
void Try(int i)
{
     for (int j=1;j<=n;j++)
      {
           if (B[j]==0)
           {
                x[i]=j;
                B[j]=1;
                if (i==n)
```

```
                        {
                                for (int k=1;k<=n;k++)
                                        cout<<x[k]<<" ";
                                cout<<endl;
                         }
                        else
                                Try(i+1);
                        B[j]=0;
                }
        }
}
int main()
{
    cout<<"nhap n=";
    cin>>n;
    Try(1);
    return 0;
}
```

Với n =3 ta có kết quả như sau:



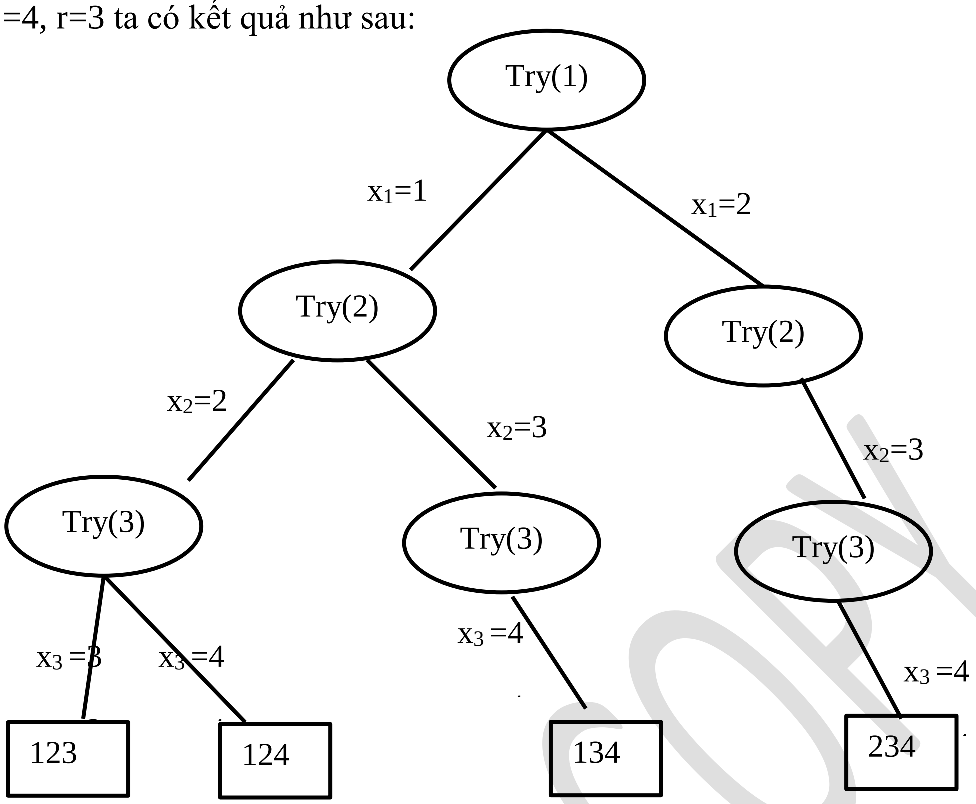
Hình 4.6 Mô tả thuật toán quay lui liệt kê dãy hoán vị với n=3

4.5.3 Tổ hợp chập r từ n phần tử

Xét bài toán liệt kê tất cả tổ hợp chập *r* từ *n* phần tử {1,2,...,n}. Mỗi tổ hợp sẽ được biểu diễn như là dãy $[x_1, x_2 ,... , x_r]$ với $x_1 < x_2 <... < x_r$. Như vậy các giá trị đề cử cho x_i là từ $x_{i-1} + 1$ đến $n-r+i$. Để điều này đúng cho cả trường hợp i = 1 ta thêm vào x_0 với $x_0 = 0$.

```
int X[100];
int n,r; int dem=0;
void Try(int i)
{
    for(int j=X[i-1]+1 ;j<=n-r+i;j++)//x0,x1, x2, ...,xr
        {
            X[i]=j;
            if (i==r)
                {
                for (int k=1; k<=r;k++)
                    cout<<X[k]<<" ";
                    cout<<endl;
                }
            else Try(i+1);
        }
}
int main()
    {
        cout<<"Nhap r=";
        cin>>r;
        cout<<"Nhap n=";
        cin>>n;
        Try(1);
        return 0;
    }
```

Với n =4, r=3 ta có kết quả như sau:



Hình 4.7 Mô tả thuật toán quay lui liệt kê dãy tổ hợp chập r=3 của n=4

Bài Tập chương 4

4.1 Viết chương trình (dùng hàm) liệt kê tất cả các tập con của tập A có n phần tử theo phương pháp sinh

4.2 Viết chương trình liệt kê tất cả các tập con của tập A có n phần tử theo phương pháp quay lui

4.3 Viết chương trình liệt kê dãy bị chặn theo phương pháp sinh

4.4 Viết chương trình liệt kê dãy bị chặn theo phương pháp quay lui

4.5 Viết các hàm liệt kê tổ hợp chập r từ n phần tử theo phương pháp sinh.

4.6 Viết các hàm liệt kê hoán vị của n phần tử theo phương pháp sinh.

4.7 Viết các hàm liệt kê các dãy nhị phân độ dài n theo phương pháp sinh.

4.8 Viết chương trình liệt kê các xâu nhị phân độ dài 5 không chứa 2 số 0 liên tiếp

4.9 Cho tập X có n phần tử. Viết chương trình liệt kê tất cả các phân hoạch tập X

4.10 Chứng minh rằng số dãy nhị phân của n phần tử là 2^n

4.11 Tìm số tập con của tập A gồm n phần tử

4.12 Dựa vào thuật toán sinh dãy bị chặn, hãy tính số lượng của dãy bị chặn
với s=3 2 1 4 5, g=5 4 1 2 3 và lập bảng để biết cấu hình của các dãy trên

Chương 5. TỐI ƯU MẠCH TỔ HỢP

Tóm tắt chương

Các mạch điện trong máy tính và các dụng cụ điện tử khác đều có các đầu vào, mỗi đầu vào là các số 0 và 1, và các đầu ra cũng là các số 0 và 1. Các mạch điện đều có thể được xây dựng bằng cách dùng bất kỳ phần tử cơ bản nào có hai trạng thái khác nhau. Chúng bao gồm các chuyển mạch có thể ở hai vị trí mở và đóng, hoặc các dụng cụ quang học có thể sáng hoặc tối. Đại số Boole dùng để phát triển lý thuyết lôgic bằng *ký hiệu* thay cho từ ngữ và có thể sử dụng đại số Boole để nghiên cứu mạch điện.

Trong chương này, chúng ta sẽ nghiên cứu các tính chất cơ bản của đại số Boole và sử dụng công cụ đại số Boole để thiết kế mạch tổ hợp.

5.1 Đại số Boole

Định nghĩa 1. Đại số Boole là hệ thống $\{S, +, \cdot, \bar{}, 0, 1\}$, trong đó tập S chứa phần tử 0 và 1, phép lấy tổng Boole (+) và phép lấy tích Boole (·) là các phép toán 2 ngôi trên S và phép bù Boole (¯) là phép toán 1 ngôi trên S thoả mãn các tính chất sau đây.

(1) Luật kết hợp

$$\forall x, y, z \in S: (x + y) + z = x + (y + z) \quad \& \; (x \cdot y) \cdot z = x \cdot (y \cdot z)$$

(2) Luật giao hoán

$$\forall x, y \in S: \quad x + y = y + x \; \& \quad x \cdot y = y \cdot x$$

(3) Luật phân phối

$$\forall x, y, z \in S: x \cdot (y + z) = (x \cdot y) + (x \cdot z) \; \&$$
$$x + (y \cdot z) = (x + y) \cdot (x + z)$$

(4) Luật đồng nhất

$$\forall x \in S: x + 0 = x \; \& \quad x \cdot 1 = x$$

(5) Luật bù trừ $\forall x \in S \; \exists \bar{x} \in S: x + \bar{x} = 1 \quad \& \quad x . \bar{x} = 0$

Ví dụ 1.

Cho U là tập vũ trụ và S là tập tất cả tập con của U. Ta định nghĩa các phép toán trên S như sau

$$\forall X, Y \in S: X + Y = X \cup Y \;\&\; X \cdot Y = X \cap Y \;\&\; \bar{X} = U \setminus X$$

Tập rỗng đóng vai trò phần tử 0 và tập U đóng vai trò phần tử 1. Khi đó hệ thống $\{S, \cup, \cap, \bar{}, \varnothing, U\}$ là đại số Boole.

Ví dụ 2.

Cho tập B = {0, 1} với các phép toán sau

Phép bù $\bar{}$: $\bar{0}= 1$, $\bar{1}= 0$

Phép +: 1 + 1 = 1, 1 + 0 = 1, 0 + 1 = 1, 0 + 0 = 0

Phép ·: 1 · 1 = 1, 1 · 0 = 0, 0 · 1 = 0, 0 · 0 = 0

là một đại số Boole.

Định lý 1. Trong đại số Boole, phần tử bù là duy nhất. Đặc biệt, nếu x và y thỏa

$$x + y = 1 \ \& \ x \cdot y = 0 \text{ thì } y = \bar{x}$$

Chứng minh:

$$\begin{aligned} y &= y \cdot 1 \\ &= y \cdot (x + \bar{x}) \\ &= y \cdot x + y \cdot \bar{x} \\ &= x \cdot y + y \cdot \bar{x} \\ &= 0 + y \cdot \bar{x} \\ &= x \cdot \bar{x} + y \cdot \bar{x} \\ &= \bar{x}(x+y) \\ &= \bar{x} \cdot 1 \\ &= \bar{x} \end{aligned}$$

Định nghĩa 2. Phần tử $\bar{x}$ gọi là *phần bù của x*

Định lý 2. Cho Đại số Boole {S, +, · , $\bar{}$, 0, 1}. Khi đó ta có các tính chất sau

(6) Luật luỹ đẳng (Idempotent)

$$\forall x \in S: x + x = x \ \& \ x \cdot x = x$$

(7) Luật giới nội (Bound)

$$\forall x \in S: x + 1 = 1 \ \& \ x \cdot 0 = 0$$

(8) Luật hấp thụ (Absortion)

$$\forall x, y \in S: x + x \cdot y = x \ \& \ x \cdot (x + y) = x$$

(9) Luật bù kép (Involution)

$$\forall x \in S: x = \bar{\bar{x}}$$

(10) Luật 0 và 1

$$\bar{0} = 1 \ \& \ \bar{1} = 0$$

(11) *Luật de Morgan*

$$\forall x, y \in S: \overline{x + y} = \bar{x}.\bar{y} \ \& \ \overline{x.y} = \bar{x} + \bar{y}$$

Chứng minh

(6), (7), (11) chứng minh như bài tập

(8) $x + x.y = x.1 + x.y$, luật đồng nhất (4)

$= x.(1 + y)$, luật phân phối (3)

$= x.1$, luật giới nội (7)

$= x$, luật đồng nhất (4)

$x.(x+y) = x.x + x.y$, luật phân phối (3)

$= x + x.y$, luật luỹ đẳng (6)

$= x$, đẳng thức trên

(9) $\overline{\overline{x}} = \overline{\overline{x}} + 0$, luật đồng nhất (4)

$= \overline{\overline{x}} + (x . \overline{x})$, luật bù trừ (5)

$= (\overline{\overline{x}} + \overline{x}) . (\overline{\overline{x}} + x)$, luật phân phối (3)

$= 1.(\overline{\overline{x}} + x)$, luật bù trừ (5)

$= (x + \overline{\overline{x}}).(x + \overline{x})$, luật bù trừ (5), giao hoán (2)

$= x + (\overline{\overline{x}} . \overline{x})$, luật phân phối (3)

$= x$, luật bù trừ (5)

(10) $\overline{0} = \overline{0} + 0$, luật đồng nhất (4)

$= 1$, luật bù trừ (5)

$\overline{1} = \overline{1} . 1$, luật đồng nhất (4)

$= 0$, luật bù trừ (5)

(11) chứng minh tương tự (xem như bài tập)

5.2 Biểu diễn hàm Boole

5.2.1 Hàm Boole

Định nghĩa 3. Cho B = {0, 1}. Biến *x* gọi là *biến Boole*, nếu nó chỉ nhận các giá trị trong B.

Một hàm từ B^n vào B, $f(x_1, ..., x_n)$, gọi là *hàm Boole bậc n*.

Các hàm *Boole* thường được biểu diễn bằng bảng.

Ví dụ 3. Hàm f: $B^2 \rightarrow B$, $f(x,y) = 1$ khi $x = 1$, $y = 0$ và $f(x,y) = 0$ trong các trường hợp khác được biểu diễn bằng bảng sau

Bảng 5.1 Biểu diễn hàm f

	x	y	$f(x,y)$
1	*1*	*1*	*0*
2	*1*	*0*	*1*
3	*0*	*1*	*0*
4	*0*	*0*	*0*

Các hàm *Boole* cũng có thể được biểu diễn bởi các *biểu thức Boole*

Định nghĩa 4. Cho B = {0, 1} với các phép toán bù, tổng và tích Boole. *Biểu thức Boole* với các biến $x_1, ..., x_n$, được định nghĩa đệ quy như sau

(1) 0, 1, $x_1, ..., x_n$ là biểu thức Boole

(2) Nếu E là biểu thức Boole, thì $\bar{E}$ cũng là biểu thức Boole

(3) Nếu E_1 và E_2 là các biểu thức Boole, thì $E_1 + E_2$ và $E_1 \cdot E_2$ cũng là biểu thức Boole.

Ví dụ 4.

Tìm bảng giá trị của hàm f(x,y,z) cho bởi biểu thức sau

$$f(x,y,z) = x \cdot y + \bar{z} \qquad 2^3 = 8$$

Giải

Các giá trị của hàm cho bởi bảng sau

Bảng 5.2 Bảng giá trị của hàm f(x,y,z)

x	y	z	$x \cdot y$	$\bar{z}$	$f(x,y,z) = x \cdot y + \bar{z}$
1	1	1	1	0	1
1	1	0	1	1	1
1	0	1	0	0	0
1	0	0	0	1	1
0	1	1	0	0	0
0	1	0	0	1	1
0	0	1	0	0	0
0	0	0	0	1	1

Ví dụ 5.

Tìm bảng giá trị của hàm f(x,y,z,t) cho bởi biểu thức sau

$$f(x,y,z) = \bar{x}.\,y + \bar{z}\,.t$$

Bảng 5.3 Bảng giá trị của hàm f(x,y,z,t)

x	y	z	t	$\bar{x}$	$\bar{x}\cdot y$	$\bar{z}$	$\bar{z}\,.t$	$f(x,y,z,t) = \bar{x}.\,y + \bar{z}\,.t$
1	1	1	1	0	0	0	0	0
1	1	1	0	0	0	0	0	0
1	1	0	1	0	0	1	1	1
1	1	0	0	0	0	1	0	0
1	0	1	1	0	0	0	0	0
1	0	1	0	0	0	0	0	0
1	0	0	1	0	0	1	1	1
1	0	0	0	0	0	1	0	0
0	1	1	1	1	1	0	0	1
0	1	1	0	1	1	0	0	1
0	1	0	1	1	1	1	1	1
0	1	0	0	1	1	1	0	1
0	0	1	1	1	0	0	0	0
0	0	1	0	1	0	0	0	0
0	0	0	1	1	0	1	1	1
0	0	0	0	1	0	1	0	0

5.2.2 Biểu diễn hàm Boole

Hai bài toán quan trọng trong đại số Boole sẽ được nghiên cứu trong bài này. *Bài toán thứ nhất* là: cho các giá trị hàm Boole, làm thế nào tìm được biểu thức Boole biểu diễn hàm đó. Bài toán này được giải bằng cách chứng minh rằng mọi hàm Boole đều có thể được biểu diễn bằng tổng các tích Boole của các biến và phần bù của chúng. Lời giải bài toán này chứng tỏ rằng mọi hàm Boole đều có thể được biểu diễn bằng cách dùng ba toán tử Boole là tích (·), tổng (+) và bù (¯)

Bài toán thứ hai là: liệu có thể dùng một tập toán tử nhỏ hơn để biểu diễn các hàm Boole hay không. Ta sẽ thấy rằng mọi hàm Boole đều có thể được biểu diễn bằng cách dùng chỉ một toán tử. Các bài toán này có tầm quan trọng thực tiễn trong việc thiết kế các mạch.

Các dạng chuẩn tắc

Ví dụ 6. Tìm các biểu thức Boole biểu diễn các hàm f(x,y,z) và g(x,y,z) có các giá trị cho trong bảng sau

Bảng 5.4 Biểu diễn các hàm f(x,y,z) và g(x,y,z)

$g = x.y.\bar{z} + \bar{x}.y.\bar{z}$

$f = x.\bar{y}.z$

x	*y*	*z*	*f*	*g*
1	*1*	*1*	*0*	*0*
1	*1*	*0*	*0*	*1*
1	*0*	*1*	*1*	*0*
1	*0*	*0*	*0*	*0*
0	*1*	*1*	*0*	*0*
0	*1*	*0*	*0*	*1*
0	*0*	*1*	*0*	*0*
0	*0*	*0*	*0*	*0*

Giải

Cần phải lập biểu thức có giá trị 1 khi

$$x = z = 1 \ \& \ y = 0,$$

và có giá trị 0 trong các trường hợp còn lại, để biểu diễn hàm *f*. Biểu thức

$$x.\bar{y}.z$$

thoả mãn yêu cầu này.

Để biểu diễn hàm *g* ta cần biểu thức bằng 1 khi

$$x = y = 1 \ \& \ z = 0 \text{ hoặc } x = z = 0 \ \& \ y = 1$$

và có giá trị 0 trong các trường hợp còn lại. Biểu thức

$$x.y.\bar{z} + \bar{x}.y.\bar{z}$$

thoả mãn yêu cầu này.

Ví dụ 7.

Tìm các biểu thức Boole biểu diễn hàm f(x,y,z,t) có các giá trị cho trong bảng sau:

Bảng 5.5 Biểu diễn giá trị hàm f(x,y,z,t)

x	*y*	*z*	*t*	*f(x,y,z,t)* = 1
1	1	1	1	0
1	1	1	0	0
1	1	0	1	1
1	1	0	0	0
1	0	1	1	0
1	0	1	0	0
1	0	0	1	1
1	0	0	0	0
0	1	1	1	0
0	1	1	0	0
0	1	0	1	1
0	1	0	0	0
0	0	1	1	0
0	0	1	0	0
0	0	0	1	1
0	0	0	0	0

$f(A_1) = a_1\ a_2\ a_3\ a_4$, $f(A_1) = 1$, $f(A_2) = 1$, $f(A_3) = 1$, $f(A_4) = 1$

$m_1 = x.y.\bar{z}.t$

$m_2 = x.\bar{y}.\bar{z}.t$

$k = 4$

$m_3 = \bar{x}.y.\bar{z}.t$

$m_4 = \bar{x}.\bar{y}.\bar{z}.t$

$A_i = (a_1, \dots, a_n)$

$\bar{x}.\bar{y}.\bar{z}.t +$

Các giá trị f =1 tại các dòng 3, 7, 11, 15 nên ta có

$$f(x,y,z,t)=x.y.\,\bar{z}.t+x.\,\bar{y}.\,\bar{z}.t+\ \bar{x}.y.\,\bar{z}.t+\,\bar{x}.\,\bar{y}.\,\bar{z}.t$$

Định nghĩa 5. Một biến Boole hoặc bù của nó gọi là một *tục biến*. Tích Boole

$$y_1.y_2.....y_n$$

trong đó

$$y_i = x_i \ \text{ hoặc } y_i = \bar{x}_i \ \forall\ i = 1, \dots, n$$

với x_i là các biến Boole, được gọi là một *tiểu hạng* (minterm)

Một tiểu hạng có giá trị 1 chỉ khi

$$y_i = 1\ \forall\ i = 1, \dots, n$$

tức là

$$x_i = 1, \text{ nếu } y_i = x_i \ \ \&\ x_i = 0, \text{ nếu } y_i = \bar{x}_i \ \ \forall\ i = 1, \dots, n$$

Ví dụ 8. Tìm tiểu hạng có giá trị bằng 1 nếu $x_1 = x_3 = 0$ và $x_2 = x_4 = x_5 = 1$ và bằng 0 trong các trường hợp còn lại.

$\bar{x}_1 . x_2 . \bar{x}_3 . x_4 . x_5$

Giải

Theo trên, tiểu hạng cần tìm là

$$\bar{x}_1 \,.\, x_2 \,.\, \bar{x}_3 \,.\, x_4 \,.\, x_5$$

Bằng cách lấy tổng Boole của các tiểu hạng phân biệt ta có thể lập được biểu thức Boole với tập các giá trị cho trước.

Định lý sau khẳng định mọi hàm Boole đều có thể biểu diễn bằng tổng các tiểu hạng.

Định lý 3. Cho hàm Boole cấp n $f(x_1, ..., x_n) \neq 0$. Giả sử $A_1, ..., A_k \in B^n$, B = {0, 1}, là các bộ giá trị thoả $f(A_i) = 1\ \forall i=1, ..., k$. Với mỗi $A_i = (a_1, ..., a_n)$, ta đặt

$$m_i = y_1 \,.\, ... \,.\, y_n$$

trong đó

$$y_j = \begin{cases} x_j, a_j = 1 \\ \bar{x}_j, a_j = 0 \end{cases}, \forall i = 1, \dots, n$$

Khi đó $f(x_1, ..., x_n) = m_1 + m_2 + ... + m_k$

Chứng minh

Với mọi $i = 1, ..., k$, ta ký hiệu $m_i(a_1, ..., a_n)$ giá trị của m_i sau khi thay x_j bằng a_j với mỗi $j = 1, ..., n$. Ta có

$$m_i(A) = \begin{cases} 1, A = A_i \\ 0, A \neq A_i \end{cases}, \forall i = 1, \dots, k$$

Khi đó với mọi $A \in B^n$ ta có

$$m_1(A) + m_2(A) + ... + m_k(A) = 1, \text{ nếu } \exists i: A = A_i$$

và

$$m_1(A) + m_2(A) + ... + m_k(A) = 0, \text{ nếu } \forall i: A \neq A_i$$

Từ đó suy ra định lý.

***Định nghĩa* 6.**

Biểu diễn $f(x1, ..., xn) = m1 + m2 + ... + mk$ *ở định lý 3* gọi là *dạng tuyển chuẩn tắc* của hàm Boole *f*.

Ví dụ 9. Tìm dạng tuyển chuẩn tắc của hàm $f(x,y,z) = (x + y).\bar{z}$

Giải.

Trước tiên ta tìm các giá trị của hàm f. Các giá trị cho ở bảng sau

Bảng 5.6 Giá trị của hàm f= (x + y).$\bar{z}$

x	y	z	$x + y$	$\bar{z}$	$(x + y).\bar{z}$
1	1	1	1	0	0
1	1	0	1	1	1
1	0	1	1	0	0
1	0	0	1	1	1
0	1	1	1	0	0
0	1	0	1	1	1
0	0	1	0	0	0
0	0	0	0	1	0

$x.y.\bar{z}$

$x.\bar{y}.\bar{z}$

$\bar{x}.y.\bar{z}$

$m_1 = \bar{z}$

$m_2 : \bar{z}$

$m_3 , \bar{z}$

Tiếp theo ta tìm các tiểu hạng tương ứng với các bộ giá trị cho biểu thức giá trị

1. Ta có tiểu hạng m như sau:

$m_1 = x.y.\bar{z}$, ứng với hàng thứ 2

$m_2 = x.\bar{y}.\bar{z}$, ứng với hàng thứ 4

$m_3 = \bar{x}.y.\bar{z}$, ứng với hàng thứ 6

Vậy $f(x,y,z)= m_1+m_2+m_3= x.y.\bar{z} + x.\bar{y}.\bar{z} + \bar{x}.y.\bar{z}$

Ví dụ 10.

Tìm dạng tuyển chuẩn tắc của hàm $f(x,y,z,t) = \bar{x}.y + \bar{z}.t$

Giải ta có bảng giá trị của f sau:

Bảng 5.7 Giá trị của hàm $f(x,y,z,t) = \bar{\mathbf{x}}.y + \bar{\mathbf{z}}.t$

x	y	z	t	$\bar{x}$	$\bar{x}\cdot y$	$\bar{z}$	$\bar{z}.t$	$f(x,y,z,t) = \bar{x}.y + \bar{z}.t$
1	1	1	1	0	0	0	0	0
1	1	1	0	0	0	0	0	0
1	1	0	1	0	0	1	1	1
1	1	0	0	0	0	1	0	0
1	0	1	1	0	0	0	0	0
1	0	1	0	0	0	0	0	0
1	0	0	1	0	0	1	1	1
1	0	0	0	0	0	1	0	0
0	1	1	1	1	1	0	0	1
0	1	1	0	1	1	0	0	1

0	1	0	1	1	1	1	1	1
0	1	0	0	1	1	1	0	1
0	0	1	1	1	0	0	0	0
0	0	1	0	1	0	0	0	0
0	0	0	1	1	0	1	1	1
0	0	0	0	1	0	1	0	0

Tiếp theo ta tìm các tiểu hạng tương ứng với các bộ giá trị cho biểu thức giá trị 1. Ta có các tiểu hạng m như sau:

$m_1 = x.y.\bar{z}.t$, ứng với hàng thứ 3

$m_2 = x.\bar{y}.\bar{z}.t$, ứng với hàng thứ 7

$m_3 = \bar{x}.y.z.t$, ứng với hàng thứ 9

$m_4 = \bar{x}.y.z.\bar{t}$, ứng với hàng thứ 10

$m_5 = \bar{x}.y.\bar{z}.t$, ứng với hàng thứ 11

$m_6 = \bar{x}.y.\bar{z}.\bar{t}$, ứng với hàng thứ 12

$m_7 = \bar{x}.\bar{y}.\bar{z}.t$, ứng với hàng thứ 15

Vậy $f(x,y,z)= m_1+m_2+m_3+ m_4 +m_5+m_6+m_7= x.y.\bar{z}.t+ x.\bar{y}.\bar{z}.t+\bar{x}.y.z.t+\bar{x}.y.\bar{z}.t+ \bar{x}.y.\bar{z}.t +\bar{x}.y.\bar{z}.\bar{t}+\bar{x}.\bar{y}.\bar{z}.t$

Định nghĩa 7. Tổng Boole $y_1 + y_2 + ... + y_n$

trong đó

$$y_i = x_i \text{ hoặc } y_i = \bar{x}_i \ \forall i = 1, ..., n$$

với x_i là các biến Boole, được gọi là một *đại hạng* (maxterm)

Một đại hạng có giá trị 0 chỉ khi

$$y_i = 0 \ \forall i = 1, ..., n$$

tức là

$x_i = 0$, nếu $y_i = x_i$ & $x_i = 1$, nếu $y_i = \bar{x}_i$ $\forall i = 1, ..., n$

Định lý sau khẳng định mọi hàm Boole đều có thể biểu diễn bằng tích các đại hạng.

Định lý 4. Cho hàm Boole cấp n $f(x_1, ..., x_n) \neq 1$. Giả sử $A_1, ..., A_k \in B^n$, B = {0, 1}, là các bộ giá trị thoả $f(A_i) = 0$ $\forall i=1, ..., k$. Với mỗi $A_i = (a_1, ..., a_n)$, ta đặt

$$M_i = y_1 + y_2 + \ldots + y_n$$

$$y_j = \begin{cases} x_j, a_j = 0 \\ \overline{x}_j, a_j = 1 \end{cases}, \forall i = 1, \ldots, n$$

Khi đó

$$f(x_1, \ldots, x_n) = M_1 \, . \, M_2 \, . \, \ldots \, . \, M_k$$

Chứng minh

Với mọi $i = 1, \ldots, k$, ta ký hiệu $M_i(a_1, \ldots, a_n)$ giá trị của M_i sau khi thay x_j bằng a_j với mỗi $j = 1, \ldots, n$. Ta có

$$M_i(A) = \begin{cases} 0, A = A_i \\ 1, A \neq A_i \end{cases}, \forall i = 1, \ldots, k$$

Khi đó với mọi A $\in B^n$ ta có

$$M_1(A) \, . \, M_2(A) \, . \, \ldots \, . \, M_k(A) = 0, \text{ nếu } \exists i\text{: } A = A_i$$

và

$$M_1(A) \, . \, M_2(A) \, . \, \ldots \, . \, M_k(A) = 1, \text{ nếu } \forall i\text{: } A \neq A_i$$

Từ đó suy ra định lý.

Định nghĩa 8.

Biểu diễn $f(x_1, \ldots, x_n) = M_1 \, . \, M_2 \, . \, \ldots \, . \, M_k$ ở định lý 2 gọi là *dạng hội chuẩn tắc* của hàm Boole *f*.

Ví dụ 11. Tìm dạng hội chuẩn tắc của hàm f(x, y, z) = (x + y). $\bar{z}$.

Giải.

Trước tiên ta tìm các giá trị của hàm f. Các giá trị cho ở bảng sau

Bảng 5.8 Giá trị của hàm f(x, y, z) = (x + y). $\bar{z}$

x	*y*	*z*	*x + y*	$\bar{z}$	*(x + y).* $\bar{z}$
1	1	1	1	0	0
1	1	0	1	1	1
1	0	1	1	0	0
1	0	0	1	1	1
0	1	1	1	0	0
0	1	0	1	1	1
0	0	1	0	0	0
0	0	0	0	1	0

$M_1 = (\bar{x} + \bar{y} + \bar{z})$.
$M_2 = (\bar{x} + y + \bar{z})$.
$M_3 = (x + \bar{y} + \bar{z})$.
$M_4 = (x + y + \bar{z})$
$M_5 = (x + y + z)$

Tiếp theo ta tìm các đại hạng tương ứng với các bộ giá trị cho biểu thức giá trị 0.

Ta có các đại hạng M như sau:

Ta có $M_1 = (\bar{x} + \bar{y} + \bar{z})$, ứng với hàng thứ 1

$M_2 = (\bar{x} + y + \bar{z})$, ứng với hàng thứ 3

$M_3 = (x + \bar{y} + \bar{z})$, ứng với hàng thứ 5

$M_4 = (x + y + \bar{z})$, ứng với hàng thứ 7

$M_5 = (x + y + z)$, ứng với hàng thứ 8

$f(x, y, z) = M_1.M_2.M_3.M_4$

Ví dụ 12.

Tìm dạng tuyển chuẩn tắc của hàm $f(x,y,z,t) = \bar{x}.\ y + \bar{z}\ .t$

Giải ta có bảng giá trị của f sau:

Bảng 5.9 Giá trị của hàm $f(x,y,z,t) = \bar{\mathbf{x}}.\ y + \bar{\mathbf{z}}\ .t$

x	y	z	t	$\bar{x}$	$\bar{x} \cdot y$	$\bar{z}$	$\bar{z}\ .t$	$f(x,y,z,t) = \bar{x}.\ y + \bar{z}\ .t$
1	1	1	1	0	0	0	0	0
1	1	1	0	0	0	0	0	0
1	1	0	1	0	0	1	1	1
1	1	0	0	0	0	1	0	0
1	0	1	1	0	0	0	0	0
1	0	1	0	0	0	0	0	0
1	0	0	1	0	0	1	1	1
1	0	0	0	0	0	1	0	0
0	1	1	1	1	1	0	0	1
0	1	1	0	1	1	0	0	1
0	1	0	1	1	1	1	1	1
0	1	0	0	1	1	1	0	1
0	0	1	1	1	0	0	0	0
0	0	1	0	1	0	0	0	0
0	0	0	1	1	0	1	1	1
0	0	0	0	1	0	1	0	0

Tiếp theo ta tìm các tiểu hạng tương ứng với các bộ giá trị cho biểu thức giá trị 0. Ta có các số 0 ở dòng 1, 2, 4, 5, 6, 8, 13, 14, 16

$M_1 = \bar{x}+\bar{y}+\bar{z}+\bar{t}$, ứng với hàng thứ 1

$M_2 = \bar{x}+\bar{y}+\bar{z}+t$, ứng với hàng thứ 2

$M_3 = \bar{x}+\bar{y}+z+t$, ứng với hàng thứ 4

$M_4 = \bar{x}+y+\bar{z}+\bar{t}$, ứng với hàng thứ 5

$M_5 = \bar{x}+y+\bar{z}+t$, ứng với hàng thứ 6

$M_6 = \bar{x}+y+z+t$, ứng với hàng thứ 8

$M_7 = x+y+\bar{z}+\bar{t}$, ứng với hàng thứ 13

$M_8 = x+y+\bar{z}+t$, ứng với hàng thứ 14

$M_9 = x+y+z+t$, ứng với hàng thứ 16

Vậy $f(x, y, z) = M_1.M_2.M_3.M_4.M_5.M_6.M_7.M_8.M_9 =$ $(\bar{x}+\bar{y}+\bar{z}+\bar{t}).(\bar{x}+\bar{y}+\bar{z}+t).(\bar{x}+\bar{y}+z+t).(\bar{x}+y+\bar{z}+\bar{t}).(\bar{x}+y+\bar{z}+t).(\bar{x}+y+z+t).(x+y+\bar{z}+\bar{t}).(x+y+\bar{z}+t).(x+y+z+t)$

5.3 Mạch tổ hợp

Trong máy tính số chỉ có hai khả năng, viết dạng 0 và 1, cho đối tượng nguyên tử, không thể phân chia được nữa. Mọi chương trình và dữ liệu bắt buộc phải quy về tổ hợp các *bit*. Nhiều thiết bị được sử dụng để lưu trữ *bit*. Các mạch điện tử cho phép các thiết bị lưu trữ liên lạc với nhau. Một *bit* từ một bộ phận của mạch được chuyển tới bộ phận khác dưới dạng điện thế. Như vậy phải có hai mức điện thế khác nhau, chẳng hạn điện thế cao có thể biểu diễn bằng bit 1 và điện thế thấp có thể biểu diễn bằng bit 0.

Trong mục này chúng ta sẽ nghiên cứu *mạch tổ hợp*. Đầu ra của mạch tổ hợp được xác định duy nhất cho từng tổ hợp đầu vào. Mạch tổ hợp không có bộ nhớ; đầu vào trước đó và trạng thái hệ thống không ảnh hưởng đến đầu ra của mạch.

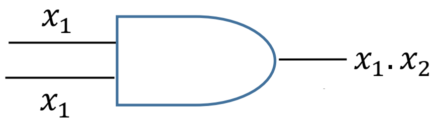
Mạch tổ hợp có thể được xây dựng bằng các thiết bị có trạng thái cơ bản gọi là *cổng*, có khả năng chuyển đổi các mức điện thế. Chúng ta sẽ bắt đầu với các cổng AND, OR và NOT.

Định nghĩa 9.

Cổng AND nhận đầu vào là các bit x_1 và x_2 và xuất đầu ra là

$$x_1.x_2 = \begin{cases} 1, & (x_1, x_2) = (1,1) \\ 0, & (x_1, x_2) \neq (1,1) \end{cases}$$

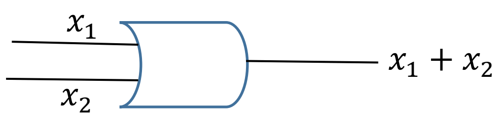
Cổng AND được biểu diễn bằng sơ đồ sau



Cổng OR nhận đầu vào là các bit x_1 và x_2 và xuất đầu ra là

$$x_1 + x_2 = \begin{cases} 0, & (x_1, x_2) = (0,0) \\ 1, & (x_1, x_2) \neq (0,0) \end{cases}$$

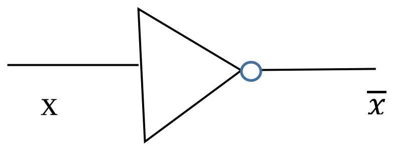
Cổng OR được biểu diễn bằng sơ đồ sau



Cổng NOT (bộ đảo) nhận đầu vào là bit x và xuất đầu ra là

$$\overline{x} = \begin{cases} 0, & x = 1 \\ 1, & x = 0 \end{cases}$$

Cổng NOT được biểu diễn bằng sơ đồ sau



◊ *Ghi chú*. Cổng AND và cổng OR có thể mở rộng cho trường hợp n đầu vào như sau

Tổ hợp các cổng

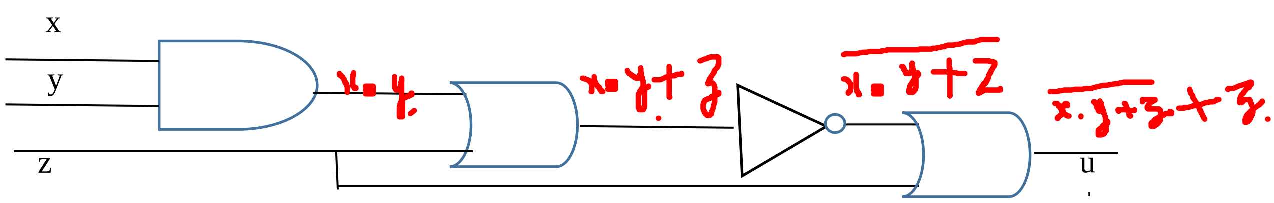
Các mạch tổ hợp có thể được xây dựng bằng cách tổ hợp các cổng NOT, AND và OR.

Biểu thức Boole đầu ra của mạch tổ hợp được xác định bằng cách tổ hợp các biểu thức cơ bản của các cổng.

Ngược lại, có thể thiết kế mạch tổ hợp cho mỗi biểu thức Boole bằng cách thiết kế dần các mạch con cho tới khi xác định được mạch toàn bộ.

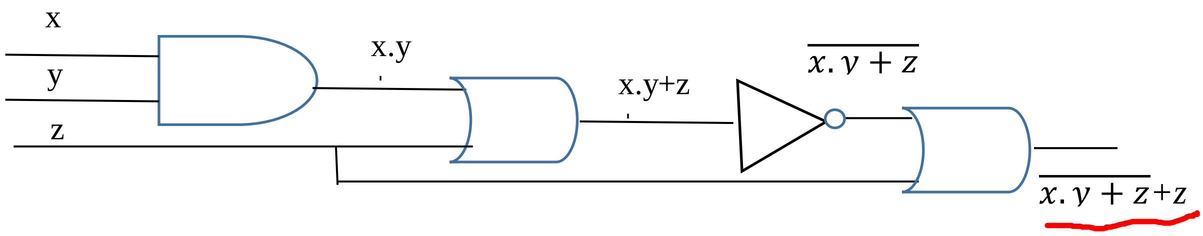
Khi thiết kế tổ hợp các mạch một số cổng có thể dùng chung đầu vào hoặc có thể tách riêng đầu vào.

Ví dụ 13. Xác định biểu thức Boole đầu ra u của mạch tổ hợp sau



Hình 5.1 Mạch tổ hợp u chưa xác định

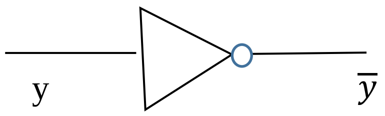
Ta lần lượt tính các biểu thức đầu ra trung gian cho tới khi xác định được đầu ra *u* như sau



Hình 5.2 Mạch tổ hợp u đã xác định

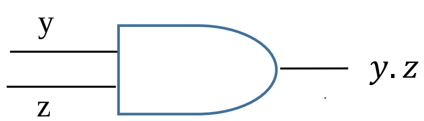
Ví dụ 14. Thiết kế mạch tổ hợp cho biểu thức Boole (x. ($\bar{y}$ + z)) + y.z. Các mạch con được thiết kế tuần tự theo trình tự sau

(1)



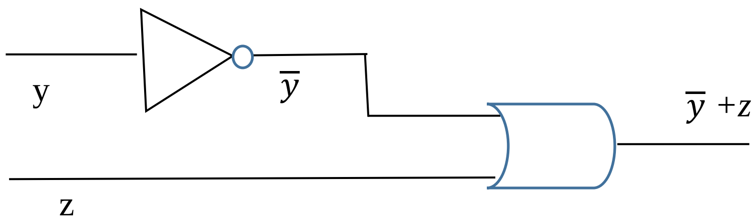
Hình 5.3 Mạch tổ hợp bước 1

(2)



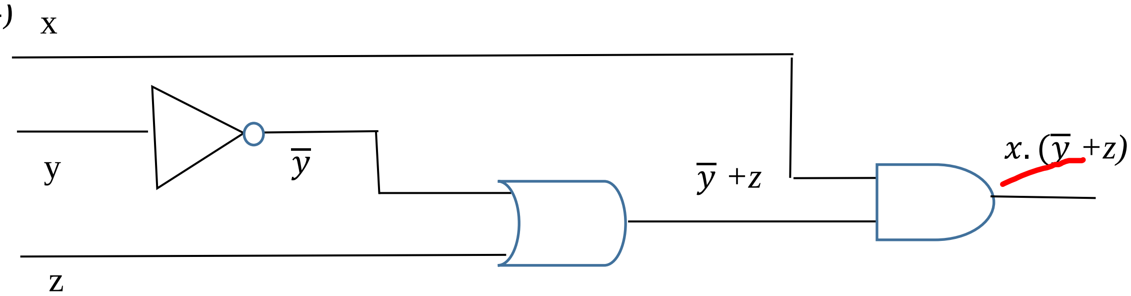
Hình 5.4 Mạch tổ hợp bước 2

(3)



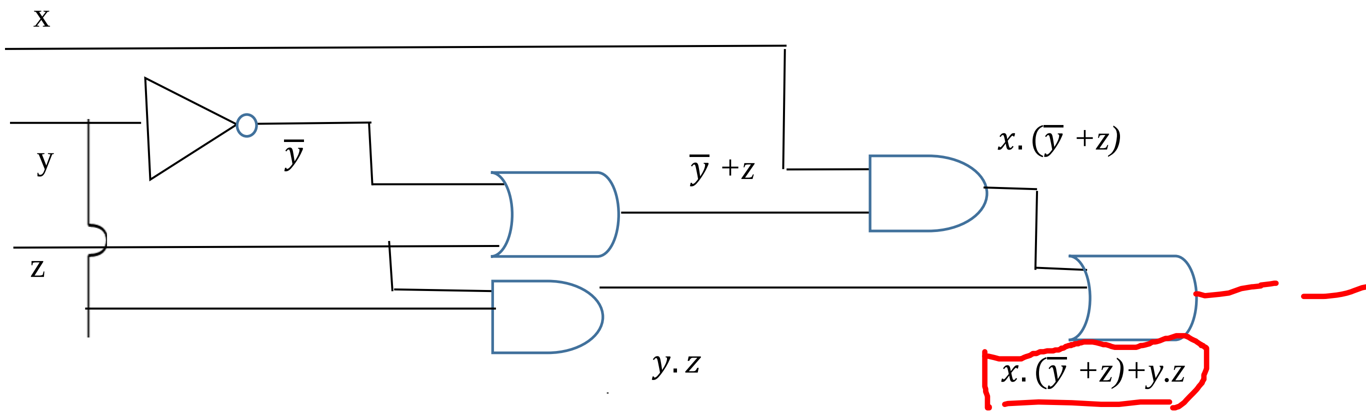
Hình 5.5 Mạch tổ hợp bước 3

(4)



Hình 5.6 Mạch tổ hợp bước 4

(5)



Hình 5.7 Mạch tổ hợp bước 5

5.4 Cực tiểu hóa mạch tổ hợp

5.4.1 Bài toán cực tiểu hoá mạch

Ví dụ 15. Xét bài toán thiết lập mạch cho hàm Boole *f* cho bởi bảng giá trị sau

Bảng 5.10 Bảng chân trị của f

x	*y*	*z*	*f*
1	1	1	1
1	1	0	1
1	0	1	0
1	0	0	1
0	1	1	1
0	1	0	0
0	0	1	0
0	0	0	0

Ta tìm các tiểu hạng tương ứng với các bộ giá trị cho biểu thức giá trị 1.

Ta có các tiểu hạng m sau:

$m_1 = x.y.z$, ứng với hàng thứ 1

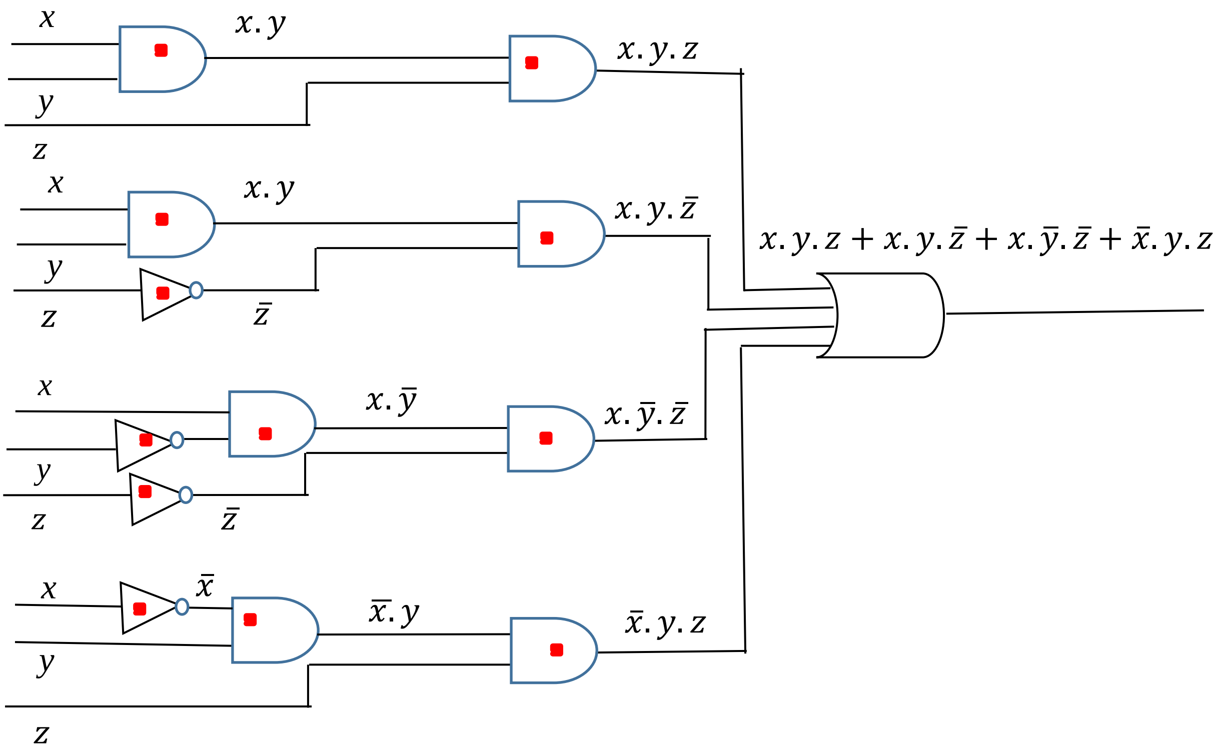
$m_2 = x.y.\bar{z}$, ứng với hàng thứ 2

$m_3 = x.\bar{y}.\bar{z}$, ứng với hàng thứ 4

$m_4 = \bar{x}.y.z$, ứng với hàng thứ 4 5

Vậy f(*x*,*y*,*z*) = $m_1 + m_2 + m_3 + m_4$

Mạch tổ hợp tương ứng có dạng sau



Hình 5.8 Mạch tổ hợp ứng với lời giải

Mạch trên có tới 13 cổng, đây là mạch tương đối phức tạp. Có tồn tại thiết kế mạch tương đương đơn giản hơn không? Câu trả lời là tồn tại.

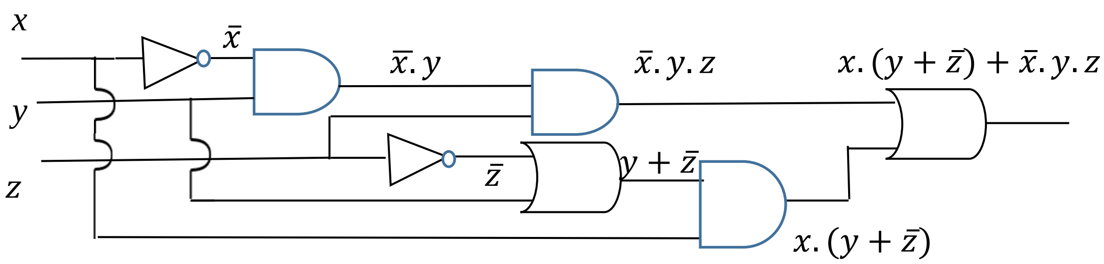
Sử dụng các đẳng thức

$$1.a + 1.\bar{a} = 1 \qquad \& \quad 1 = 1 + 1.a\ \forall a$$

$x.y(\overbrace{z+\bar{z}}^{1})$

$$\begin{aligned} f(x,y,z) &= x.y.z + x.y.\bar{z} + x.\bar{y}.\bar{z} + \bar{x}.y.z \\ &= x.y + x.\bar{y}.\bar{z} + \bar{x}.y.z \\ &= x.y(1+\bar{z}) + x.\bar{y}.\bar{z} + \bar{x}.y.z \\ &= x.y + x.y.\bar{z} + x.\bar{y}.\bar{z} + \bar{x}.y.z \\ &= x.y + x.\bar{z} + \bar{x}.y.z \\ &= x.(y+\bar{z}) + \bar{x}.y.z \end{aligned}$$

Mạch tổ hợp tương đương chỉ sử dụng 7 cổng như sau



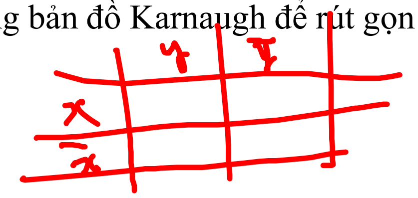
Hình 5.9 Mạch tổ hợp rút gọn

Bài toán tìm mạch tương đương có số cổng ít nhất gọi là bài toán cực tiểu hoá mạch tổ hợp.

5.4.2 Phương pháp bản đồ Karnaugh

Để làm giảm số các số hạng trong biểu thức Boole biểu diễn mạch, ta cần tìm các số hạng để tổ hợp lại. Phương pháp bản đồ Karnaugh được Maurice Karnaugh đưa ra năm 1953 dựa trên công trình trước đó của E.W. Veitch. Đây là phương pháp trực quan để rút gọn khai triển tổng các tích.

Trước hết chúng ta sẽ minh hoạ cách dùng bản đồ Karnaugh để rút gọn biểu thức Boole 2 biến.



5.4.3 Rút gọn biểu thức Boole 2 biến

Có 4 tiểu hạng khả dĩ trong khai triển tổng các tích của một hàm Boole 2 biến x và y. Một bản đồ Karnaugh đối với một hàm Boole 2 biến gồm 4 ô vuông trong đó hình vuông biểu diễn tiểu hạng có mặt trong khai triển được ghi số 1. Các ô gọi là kề nhau nếu các tiểu hạng mà chúng biểu diễn chỉ khác nhau một tục biến. Chẳng hạn ô biểu diễn $\bar{x}$.y kề với các ô x.y và $\bar{x}.\bar{y}$. Bốn ô vuông biểu diễn các tiểu hạng cho ở bảng dưới

Bảng 5.11 Bản đồ Karnaugh 2 biến

	y	$\bar{y}$
x	1	1
$\bar{x}$	1	1

Ví dụ 16. Tìm các bản đồ Karnaugh cho các biểu thức sau

(a) $x.y + \bar{x}.y$ (b) $x.\bar{y} + \bar{x}.y$ (c) $x.\bar{y} + \bar{x}.y + \bar{x}.\bar{y}$

Giải

Các bản đồ Karnaugh tương ứng như sau

(a)

Bảng 5.12 Bản đồ Karnaugh a

	y	$\bar{y}$
x	1	
$\bar{x}$	1	

(b)

Bảng 5.13 Bản đồ Karnaugh b

	y		$\bar{y}$
x			1
$\bar{x}$	1		

(c)

Bảng 5.14 Bản đồ Karnaugh c

	y	$\bar{y}$
x		1
$\bar{x}$	1	1

Ta nhận thấy tổng các tiểu hạng kề nhau có thể rút gọn thành 1 biến tương ứng với cột hoặc hàng của các tiểu hạng đó. Do đó để rút gọn biểu thức, ta khoanh tất cả các cặp ô kề nhau có thể, sau đó thế chúng bằng biến tương ứng.

Ví dụ 17. Rút gọn các biểu thức ở ví dụ 16.

(a)

Bảng 5.15 Bản đồ Karnaugh đánh dấu a

	y	$\bar{y}$
x	1	
$\bar{x}$	1	

(b)

Bảng 5.16 Bản đồ Karnaugh đánh dấu b

	y	$\bar{y}$
x		1
$\bar{x}$	1	

(c)

Bảng 5.17 Bản đồ Karnaugh đánh dấu c

	y	$\bar{y}$
x		1
$\bar{x}$	1	1

	yz	$y\bar{z}$	$\bar{y}\bar{z}$	$\bar{y}z$
x				
$\bar{x}$				

Khai triển cực tiểu các biểu thức tương ứng là

a)y b) $\bar{x}.y + x.\bar{y}$ c) $\bar{x} + \bar{y}$

5.4.4 Rút gọn biểu thức Boole 3 biến

Có 8 tiểu hạng khả dĩ trong khai triển tổng các tích của một hàm Boole 3 biến x, y và z. Một bản đồ Karnaugh đối với một hàm Boole 3 biến gồm 8 ô vuông trong đó hình vuông biểu diễn tiểu hạng có mặt trong khai triển được ghi số 1. Các ô gọi là kề nhau nếu các tiểu hạng mà chúng biểu diễn chỉ khác nhau một tục biến. Do đó, mỗi ô kề với 3 ô khác. Chẳng hạn ô biểu diễn $\bar{x}$ .y.z kề với các ô x.y.z, $\bar{x}$ .y. $\bar{z}$ và $\bar{x}$. $\bar{y}$ .z. 8 ô vuông biểu diễn các tiểu hạng cho ở bảng dưới

Bảng 5.18 Bản đồ Karnaugh 4 biến

	y.z	$y.\bar{z}$	$\bar{y}.\bar{z}$	$\bar{y}.z$
x	1	1	1	1
$\bar{x}$	1	1	1	1

$f = \dots = 1$

Ta có thể coi các ô được vẽ trên mặt trụ sao cho các ô biên kề nhau

Trên bản đồ Karnaugh 3 biến, ta có thể nhóm các ô kề nhau thành các khối 2, 4, 8 ô. Các bảng sau là một số ví dụ biểu diễn các khối 1 x 2, 2 x 1, 2 x 2, 4 x 1 và 4 x 2 và các tích tương ứng

(a) $\bar{y}.\bar{z} = x.\bar{y}.\bar{z} + \bar{x}.\bar{y}.\bar{z}$

Bảng 5.19 Bản đồ Karnaugh 3 biến a

	y.z	$y.\bar{z}$	$\bar{y}.\bar{z}$	$\bar{y}.z$
x			1	
$\bar{x}$			1	

(b) $\bar{x}.z = \bar{x}.y.z + \bar{x}.\bar{y}.z$

Bảng 5.20 Bản đồ Karnaugh 3 biến b

	y.z	$y.\bar{z}$	$\bar{y}.\bar{z}$	$\bar{y}.z$
x				
$\bar{x}$	1			1

(c) $\bar{z} = x.y.\bar{z} + \bar{x}.y.\bar{z} + x.\bar{y}.\bar{z} + \bar{x}.\bar{y}.\bar{z}$

Bảng 5.21 Bản đồ Karnaugh 3 biến c

	y.z	$y.\bar{z}$	$\bar{y}.\bar{z}$	$\bar{y}.z$
x		1	1	
$\bar{x}$		1	1	

(d) $\bar{x} = \bar{x}.y.z + \bar{x}.y.\bar{z} + \bar{x}.\bar{y}.\bar{z} + \bar{x}.\bar{y}.z$

Bảng 5.22 Bản đồ Karnaugh 3 biến d

	y.z	$y.\bar{z}$	$\bar{y}.\bar{z}$	$\bar{y}.z$
x				
$\bar{x}$	1	1	1	1

(e) 1 $= x.y.z + x.y.\bar{z} + x.\bar{y}.\bar{z} + x.\bar{y}.z + \bar{x}.y.z + \bar{x}.y.\bar{z} + \bar{x}.\bar{y}.\bar{z}+\bar{x}.\bar{y}.z$

Bảng 5.23 Bản đồ Karnaugh 3 biến e

	y.z	$y.\bar{z}$	$\bar{y}.\bar{z}$	$\bar{y}.z$
x	1	1	1	1
$\bar{x}$	1	1	1	1

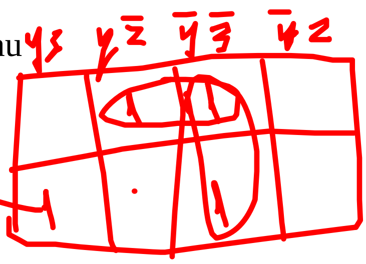
Ta nhận thấy tổng các tiểu hạng trong một khối có thể rút gọn thành 1 tiểu hạng tương ứng. Do đó để rút gọn biểu thức, ta khoanh tất cả các khối ô lớn nhất có thể, sau đó thế chúng bằng tiểu hạng tương ứng.

Ví dụ 18. Dùng bản đồ Karnaugh ba biến để rút gọn các tổng các tích sau

(a) $x.y.\bar{z} + x.\bar{y}.\bar{z} + \bar{x}.y.z + \bar{x}.\bar{y}.\bar{z}$

(b) $x.\bar{y}.z + x.\bar{y}.\bar{z} + \bar{x}.y.z + \bar{x}.\bar{y}.z + \bar{x}.\bar{y}.\bar{z}$

(c) $x.y.z + x.y.\bar{z} + x.\bar{y}.z + x.\bar{y}.\bar{z} + \bar{x}.y.z + \bar{x}.\bar{y}.z + \bar{x}.\bar{y}.\bar{z}$

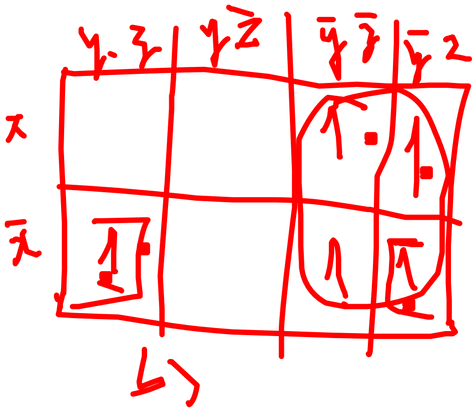


Giải: Bản đồ Karnaugh của các biểu thức trên và các khối tương ứng như sau

(a)

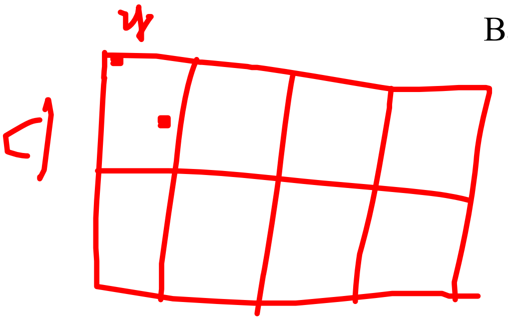
a) $\bar{x}yz + \bar{y}\bar{z} + x\bar{z}$

Bảng 5.24 Bản đồ Karnaugh 3 biến đánh dấu a



	y.z	$y.\bar{z}$	$\bar{y}.\bar{z}$	$\bar{y}.z$
x		1	1	
$\bar{x}$	1		1	

(b)



Bảng 5.25 Bản đồ Karnaugh 3 biến đánh dấu b

	y.z	$y.\bar{z}$	$\bar{y}.\bar{z}$	$\bar{y}.z$
x			1	1
$\bar{x}$	1		1	1

(c)

Bảng 5.26 Bản đồ Karnaugh 3 biến đánh dấu c

	y.z	$y.\bar{z}$	$\bar{y}.\bar{z}$	$\bar{y}.z$
x	1	1	1	1
$\bar{x}$	1		1	1

$x + \bar{y} + z$

Từ đó ta suy ra các khai triển cực tiểu là

(a) $\bar{x}$ .y.z + x. $\bar{z}$ +$\bar{y}.\bar{z}$

(b) $\bar{x}$.z + $\bar{y}$

(c) x + $\bar{y}$ + z

$f = xy + x\bar{y} + \bar{x}y + \bar{x}\bar{y}$

$f = \bar{y}$

Bài tập chương 5

5.1 Tìm khai triển tổng các tích của các hàm Boole hai biến x, y

a) $\bar{x}$ + y b) x. $\bar{y}$ c) 1 d) $\bar{y}$

$m_1 = x.y$

$m_2 = \bar{x}.y$

$m_3 = \bar{x}.\bar{y}$

5.2 Tìm khai triển tổng các tích của các hàm Boole ba biến x, y, z

a) x + y + z b) (x + z).y c) x d) x. $\bar{y}$

5.3 Tìm khai triển tổng các tích của các hàm Boole F(x, y, z) biết F = 1 nếu và chỉ nếu

$\Rightarrow f$

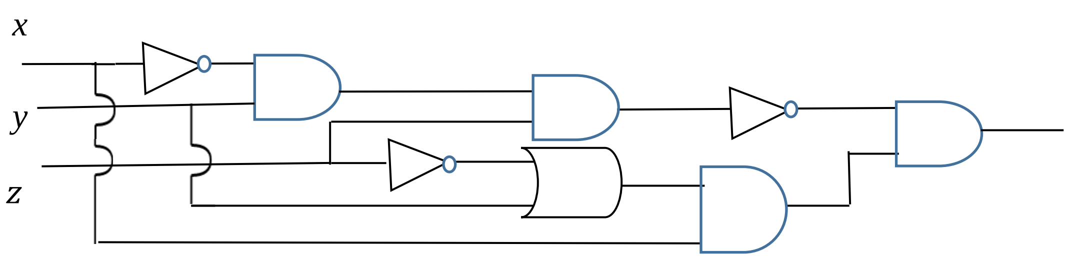
a) x = 0 b) x.y = 0 c) x + y = 0 d) x.y.z = 0

5.4 Tìm khai triển tổng các tích của các hàm Boole F(w, x, y, z) biết F = 1 nếu và chỉ nếu một số lẻ của w, x, y, z có giá trị 1.

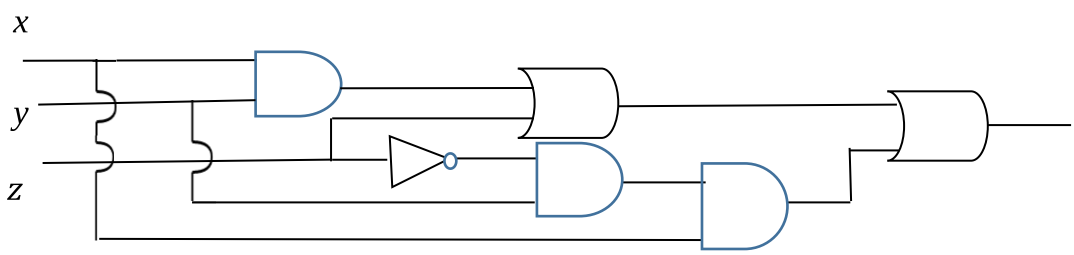
5.5 Tìm khai triển tổng các tích của các hàm Boole F(v, w, x, y, z) biết F = 1 nếu và chỉ nếu số biến có giá trị 1 lớn hơn hoặc bằng 3.

5.6 Xác định hàm đầu ra của các mạch sau

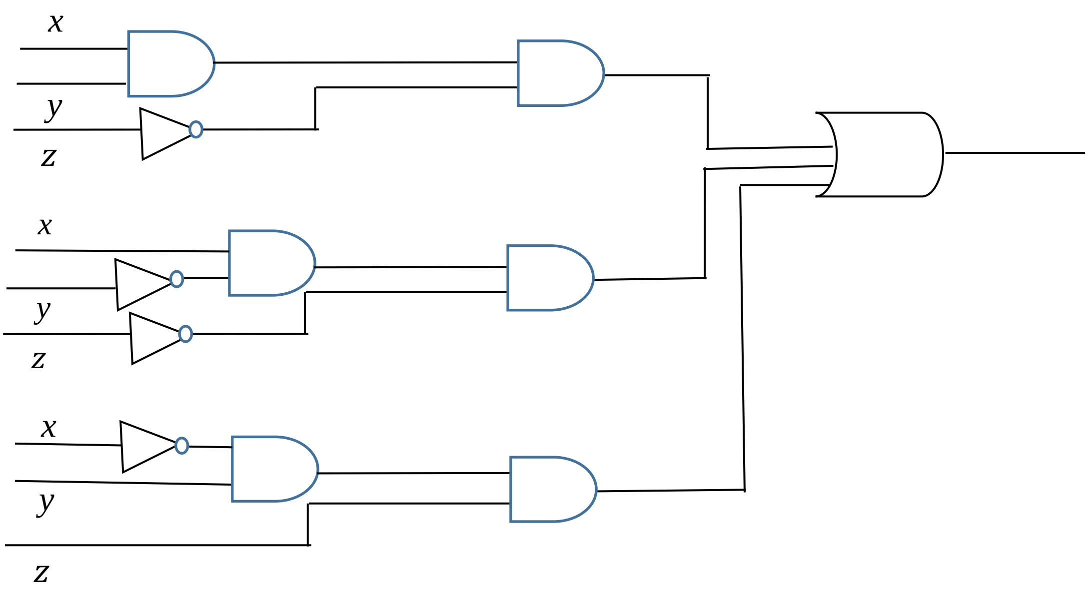
a)



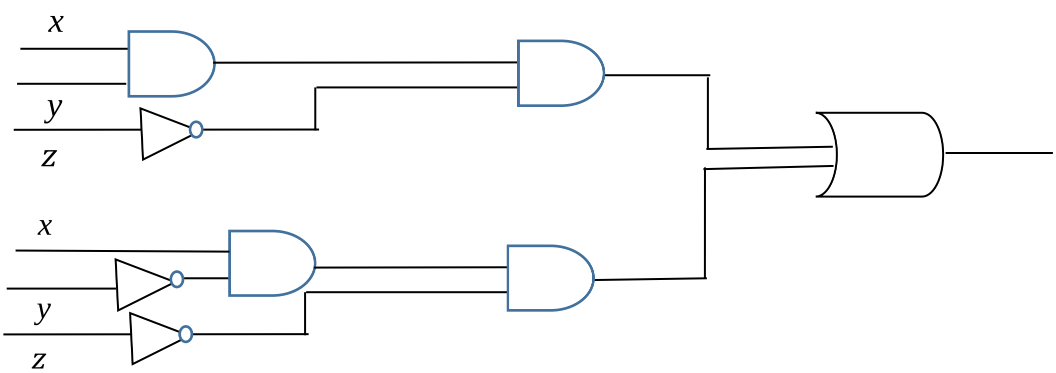
b)



c)



d)



5.7 Dựng các mạch gồm các bộ đảo, các cổng AND và OR để tạo các đầu ra sau

a) $\bar{x}$ .y.z + x. $\bar{z}$ +$\bar{y}.\bar{z}$

b) $\bar{x}$.z + $\bar{y}$

c) x + $\bar{y}$ + z

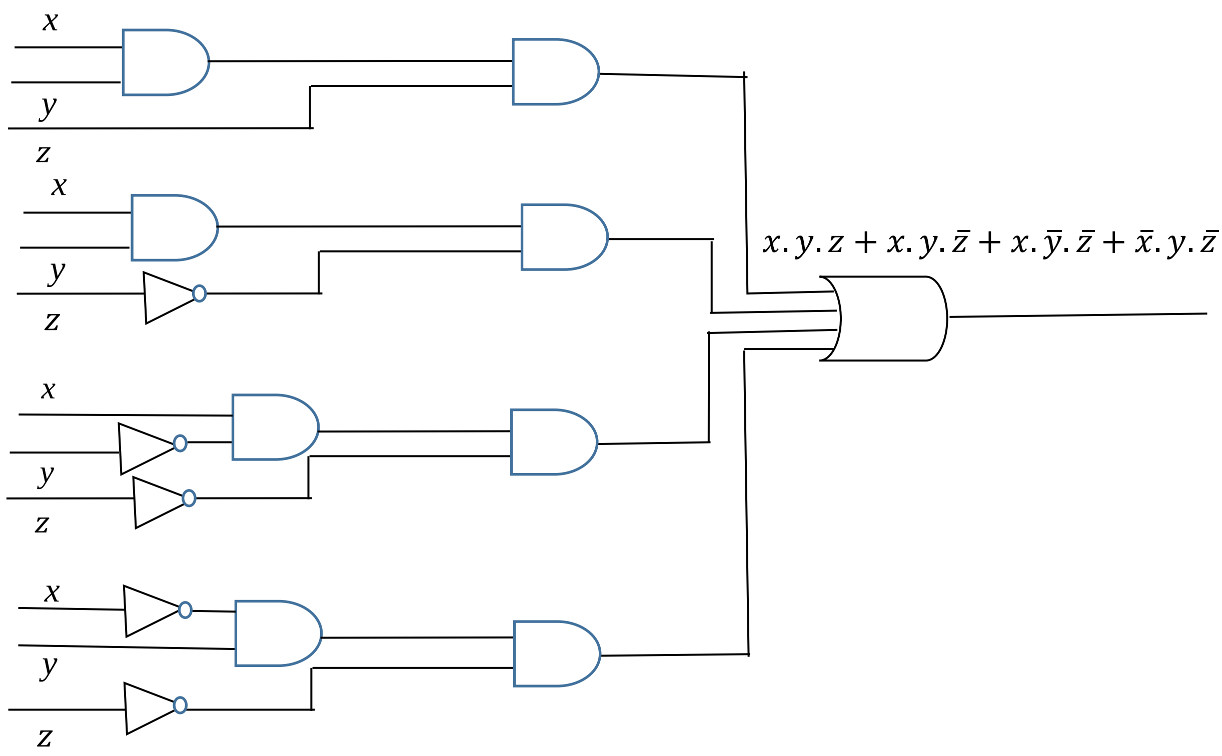
d) $\bar{x}$ .y.z + $x.\bar{y}.\bar{z}$

e) $\overline{(x+y)}$.z + x. $\bar{z}$ +$\bar{y}.\bar{z}$

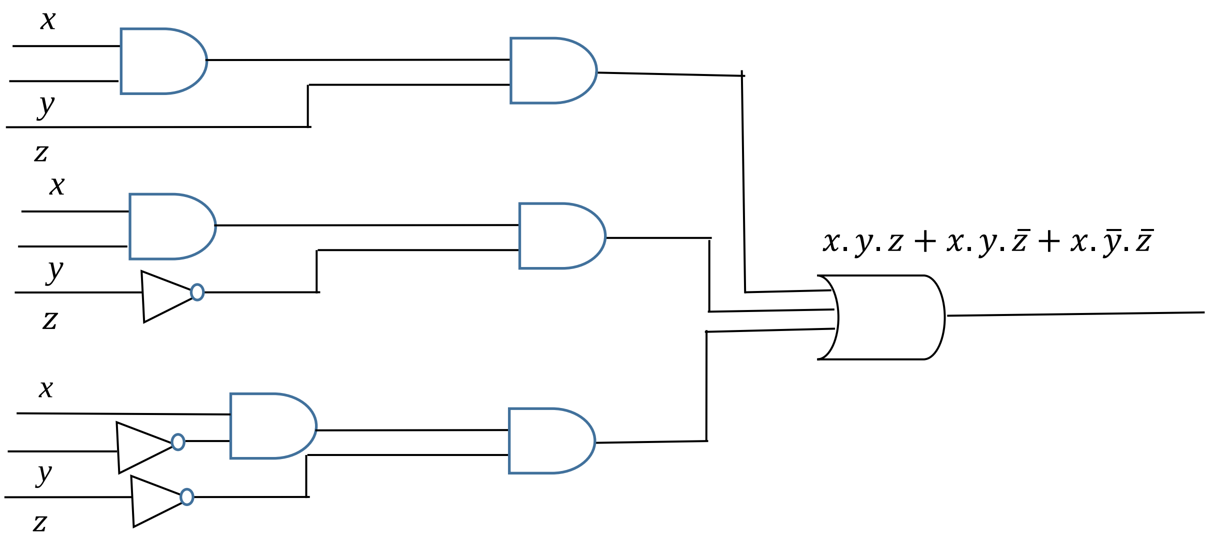
5.8 Dùng bản đồ Karnaugh để tìm khai triển cực tiểu của các hàm hai biến sau

a) $\bar{x}$ .y + $\bar{x}.\bar{y}$ b) x.y + x. $\bar{y}$ c) x.y + $\bar{x}.y + x.\bar{y}$ + $\bar{x}.\bar{y}$

5.9 Dùng bản đồ Karnaugh để tìm các mạch đơn giản hơn có cùng đầu ra đối với các mạch sau a)



b)



5.10 Dùng bản đồ Karnaugh để tìm khai triển cực tiểu của các hàm ba biến sau

a) $\bar{x}.y.z + \bar{x}.\bar{y}.z$

b) $x.y.z + x.\bar{y}.z$

c) $x.y.\bar{z} + x.\bar{y}.z + x.\bar{y}.\bar{z} + \bar{x}.\bar{y}.z$

Chương 6. ĐẠI CƯƠNG VỀ ĐỒ THỊ

Tóm tắt chương

Đồ thị là một cấu trúc rời rạc gồm các *đỉnh* và các *cạnh* hoặc *cung* (cạnh có hướng) nối các đỉnh đó. Đây là công cụ hữu hiệu để mô hình hoá và giải quyết các bài toán trong nhiều lĩnh vực khoa học, kỹ thuật, kinh tế, xã hội, ... Trong chương này sẽ nghiên cứu các khái niệm, cách biểu diễn đồ thị và một số loại đồ thị

6.1 Các khái niệm cơ bản

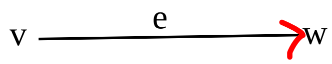
6.1.1. Đồ thị, đỉnh, cạnh, cung

a) Đồ thị vô hướng và đồ thị có hướng

Định nghĩa 1.

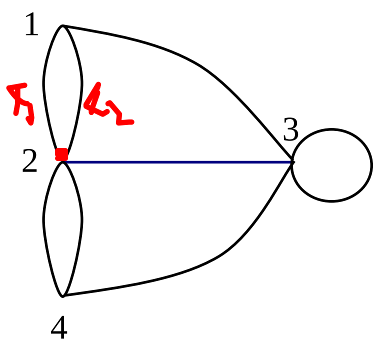
Đồ thị vô hướng G = (V, E) gồm một tập V các đỉnh và tập E các cạnh.

Mỗi cạnh e ∈ E được liên kết với một cặp đỉnh v, w (không kể thứ tự) như hình sau.



Hình 6.1 Đồ thị 2 đỉnh 1 cạnh

Ví dụ 1. Đồ thị bên dưới là đồ thị có 4 đỉnh và 7 cạnh

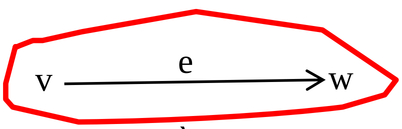


Hình 6.2 Đồ thị 4 đỉnh 7 cạnh

Định nghĩa 2.

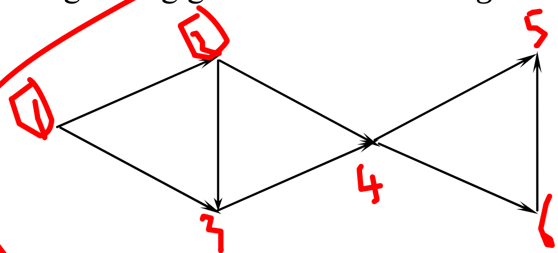
Đồ thị có hướng G = (V, E) gồm một tập V các đỉnh và tập E các cạnh có hướng gọi là cung.

Mỗi cung e ∈ E được liên kết với một cặp đỉnh (v, w) có thứ tự như hình sau.



Hình 6.3 Đồ thị có hướng 2 đỉnh 1 cung

Ví dụ 2. Đồ thị có hướng hướng gồm 6 đỉnh và 8 cung.



e = (1,4)

8

Hình 6.4 Đồ thị có hướng 6 đỉnh 8 cung

Cho đồ thị (có hướng hoặc vô hướng) G = (V, E).

Nếu cạnh e liên kết đỉnh v, w thì ta nói cạnh e *liên thuộc* đỉnh v, w, các đỉnh v, w *liên thuộc* cạnh e, các đỉnh v, w là các *đỉnh biên* của cạnh e và đỉnh v *kề* đỉnh w.

Nếu chỉ có duy nhất một cạnh e liên kết với cặp đỉnh v, w, ta viết e=(v, w). Nếu e là cung thì v gọi là *đỉnh đầu* và w gọi là *đỉnh cuối* của cung e.

Nếu có nhiều cạnh liên kết với cùng một cặp đỉnh thì ta nói đó là các cạnh *song song*.

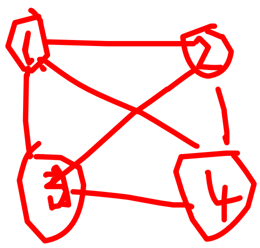
Cạnh có hai đỉnh liên kết trùng nhau gọi là *khuyên*. Đỉnh không kề với đỉnh khác gọi là *đỉnh cô lập*.

Số đỉnh của đồ thị gọi là *bậc* của đồ thị, số cạnh hoặc số cung của đồ thị gọi là *cỡ* của đồ thị.

Đồ thị hữu hạn là đồ thị có bậc và cỡ hữu hạn.

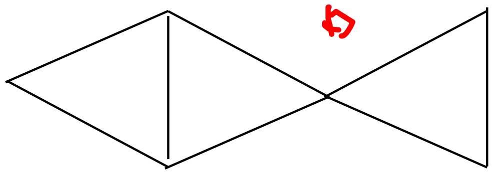
Đồ thị đơn là đồ thị không có khuyên và không có cạnh song song.

Đồ thị vô hướng đủ là đồ thị mà mọi cặp đỉnh đều kề nhau.



Ví dụ 3. Đồ thị sau là đồ thị đơn, nhưng không phải đồ thị đủ.

$C_4^2 = \frac{4!}{2!\,2!} = \frac{2.3.4}{4} = 6$



5

Hình 6.5 Đồ thị đơn

b) Bậc, nửa bậc vào, nửa bậc ra

Cho đồ thị G = (V, E).

Định nghĩa 3.

Giả sử đỉnh $v \in V$ có *p* khuyên và *q* cạnh liên thuộc (không phải khuyên). Khi đó *bậc* của đỉnh *v* là 2*p*+*q* và ký hiệu là $deg_G(v)$ hoặc đơn giản *deg*(*v*).

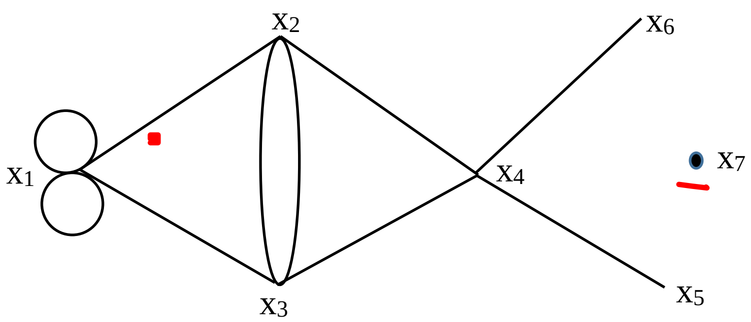
Số bậc đỉnh lớn nhất của G ký hiệu là Δ(G), số bậc đỉnh nhỏ nhất của G ký hiệu là δ(G).

Từ định nghĩa suy ra *đỉnh cô lập* trong đồ thị đơn là đỉnh có bậc bằng 0. Đỉnh có bậc bằng 1 gọi là *đỉnh treo*.

Định nghĩa 4.

Cho G= (V, E) là đồ thị có hướng, $v \in V$. *Nửa bậc ra* của đỉnh *v*, ký hiệu là $deg_O(v)$, là số cung đi ra từ đỉnh *v* (*v* là đỉnh đầu), và *nửa bậc vào* của đỉnh $v \in V$, ký hiệu là $deg_I(v)$, là số cung đi tới đỉnh *v* (*v* là đỉnh cuối).

Ví dụ 4.



Hình 6.6 Đồ thị vô hướng 7 đỉnh

Trong đồ thị này ta có

$deg(x_1) = 6$; $deg(x_2) = deg(x_3) = 4$; $deg(x_4) = 4$; $deg(x_5) = 1$; $deg(x_6) = 1$, $deg(x_7) = 0$

$\Rightarrow \Delta(G) = 6, \delta(G) = 0$.

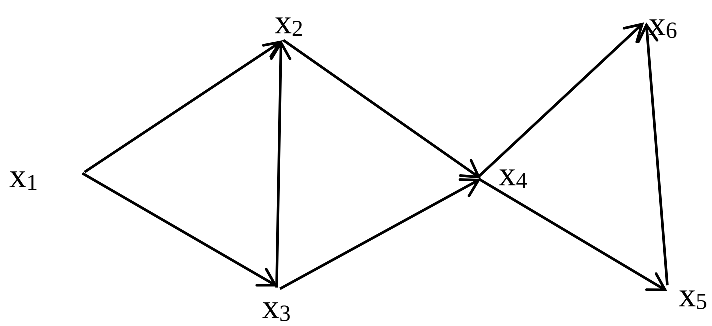
Đỉnh x_1 có hai khuyên liên thuộc.

Có hai cạnh song song liên thuộc đỉnh x_2 và x_3. Đỉnh x_7 là đỉnh cô lập.

Đỉnh x_5, x_6 là 2 đỉnh treo.

Đồ thị này không phải là đồ thị đơn, không phải là đồ thị đủ.

Ví dụ 5. Xét đồ thị có hướng sau



Hình 6.7 Đồ thị có hướng 6 đỉnh

Trong đồ thị có hướng này ta có

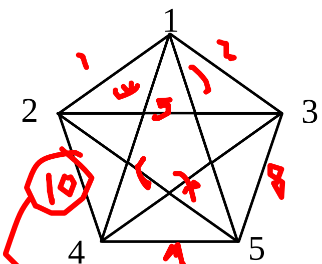
$deg_I(x_1) = 0$; $deg_O(x_1) = 2$; $deg_I(x_2) = 2$; $deg_O(x_2) = 1$;
$deg_I(x_3) = 1$; $deg_O(x_3) = 2$; $deg_I(x_4) = 2$; $deg_O(x_4) = 2$;
$deg_I(x_5) = 1$; $deg_O(x_5) = 1$; $deg_I(x_6) = 2$; $deg_O(x_6) = 0$;

Định nghĩa 5.

Đồ thị $\boldsymbol{K_n}$ là đồ thị đơn, đủ n đỉnh (mỗi cặp đỉnh đều có duy nhất một cạnh liên kết).

Ví dụ 6.

Hình sau đây là đồ thị $\boldsymbol{K_5}$



$C_5^2 = \frac{5!}{2!\,.3!} = \frac{2.3.4.5}{1.2.3} = 10$

Hình 6.8 Đồ thị K_5

Mọi đỉnh của đồ thị K_n có bậc $n-1$ và K_n có $n(n-1)/2$ cạnh.

6.1.2. Đường đi, chu trình, liên thông

Định nghĩa 6. Cho đồ thị G=(V, E).

Dây μ từ đỉnh *v* đến đỉnh *w* là tập hợp các đỉnh và cạnh nối tiếp nhau bắt đầu từ đỉnh *v* và kết thúc tại đỉnh *w*. Số cạnh trên dây μ gọi là *độ dài* của dây μ.

Dây μ từ đỉnh *v* đến đỉnh *w* độ dài *k* được biểu diễn như sau

$$\mu = (v, e_1, v_1, e_2, v_2, \dots, v_{k-1}, e_k, w)$$

trong đó v_i (i = 1, ..., k–1) là các đỉnh trên dây và e_i (i=1,...,k) là các cạnh trên dây liên thuộc đỉnh kề trước và sau nó. Các đỉnh và cạnh trên dây có thể lặp lại.

Đường đi từ đỉnh *v* đến đỉnh *w* là dây từ đỉnh v đến đỉnh w, trong đó các cạnh không lặp lại.

Đường đi sơ cấp là đường đi không đi qua một đỉnh quá 1 lần.

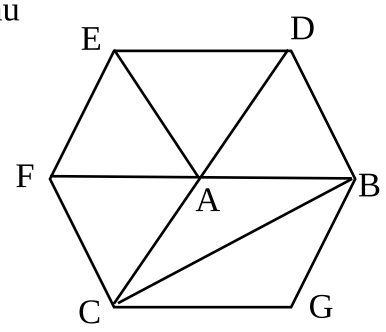
◊ *Ghi chú.* Trong đồ thị *n* đỉnh, đường đi sơ cấp giữa hai đỉnh khác nhau có nhiều nhất *n*–1 cạnh.

Vòng là dây có đỉnh đầu và đỉnh cuối trùng nhau.

Chu trình là đường đi có đỉnh đầu và đỉnh cuối trùng nhau. C*hu trình sơ cấp* là chu trình không đi qua một đỉnh quá 1 lần.

◊ *Ghi chú.* Trong đồ thị n đỉnh, chu trình sơ cấp có nhiều nhất n cạnh.

Ví dụ 7. Cho đồ thị sau



Hình 6.9 Đồ thị vô hướng 7 đỉnh

Ta có một số dây từ B đên E

μ_1 = (B,G,C,A,F,C,A,E), μ_2 = (B,A,C,F,A,E),

μ_3 = (B,A,C,F,E), μ_4 = (B,A,E).

μ_1 không phải đường đi, vì có cạnh CA lặp lại. Độ dài của μ_1 là 7.

μ_2 là đường đi, nhưng không sơ cấp, vì có đỉnh A lặp lại. Độ dài của μ_2 là 5.

μ_3 và μ_4 là các đường đi sơ cấp. Ta có các vòng sau:

η_1 = (A,B,C,G,B,C,A), η_2 = (A,B,C,G,B,D,A), η_3 = (A,B,C,A).

η_1 không phải chu trình, vì có cạnh BC lặp lại.

η_2 là chu trình, nhưng không sơ cấp, vì có đỉnh B lặp lại.

η_3 là chu trình sơ cấp.

Dây có hướng trong đồ thị có hướng là dãy các đỉnh và cung nối tiếp nhau (e_1, e_2, ..., e_k) thoả mãn đỉnh cuối của cung e_i là đỉnh đầu của cung e_{i+1} , i=1, ..., k-1.

Đường đi có hướng trong đồ thị có hướng là dây có hướng, trong đó các cung không lặp lại.

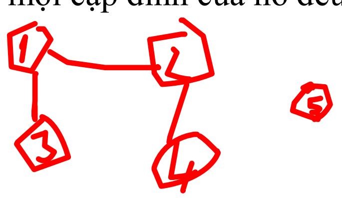
Đường đi có hướng sơ cấp là đường đi có hướng không đi qua một đỉnh quá 1 lần.

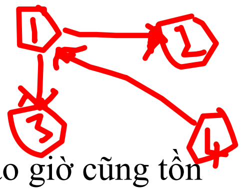
Vòng có hướng là dây có hướng có đỉnh đầu và đỉnh cuối trùng nhau.

Chu trình có hướng là đường đi có hướng có đỉnh đầu và đỉnh cuối trùng nhau.

Chu trình có hướng sơ cấp là chu trình có hướng không đi qua một đỉnh quá 1 lần.

Đồ thị vô hướng gọi là *liên thông*, nếu mọi cặp đỉnh của nó đều có đường đi nối chúng với nhau.





Đồ thị có hướng gọi là *liên thông mạnh*, nếu mọi cặp đỉnh (*u*,*v*) bao giờ cũng tồn tại đường đi có hướng từ *u* đến *v* và từ *v* đến *u*.

Đồ thị có hướng gọi là *bán liên thông*, nếu với mọi cặp đỉnh (*u*,*v*) bao giờ cũng tồn tại đường đi có hướng từ *u* đến *v* hoặc từ *v* đến *u*.

Đồ thị có hướng gọi là *liên thông yếu*, nếu đồ thị lót (vô hướng) của nó liên thông.

◊ *Ghi chú.* Đồ thị *liên thông mạnh* ⇒ Đồ thị *bán liên thông* ⇒ Đồ thị *liên thông yếu.*

Định lý 1.

(i). Trong đồ thị vô hướng mỗi dây từ đỉnh *v* đến *w* chứa đường đi sơ cấp từ *v* đến *w*.

(ii) Trong đồ thị có hướng mỗi dây có hướng từ đỉnh *v* đến *w* chứa đường đi có hướng sơ cấp từ *v* đến *w*.

Chứng minh

(i) Cho $\mu = (v, e_1, v_1, e_2, v_2, \ldots, v_{n-1}, e_n, w)$ là dây từ v đến w. Nếu $v_1, \ldots, v_{n-1}$ khác nhau thì μ là đường đi sơ cấp. Ngược lại tồn tại i, j, $0 < i < j < n$, thoả $v_i = v_j$. Ta loại các đỉnh $v_{i+1}, \ldots, v_j$ khỏi dây μ và nhận được dây từ v đến w có số cạnh ít hơn. Vì số cạnh hữu hạn, nên tiếp tục quá trình trên, đến lúc nào đó dây sẽ không có đỉnh lặp nữa, và ta nhận được đường đi sơ cấp từ v đến w.

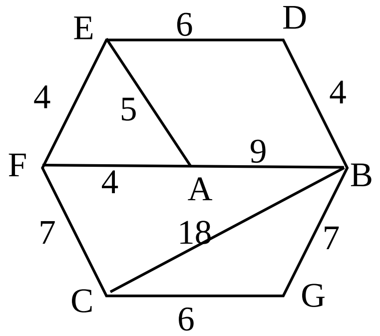
(ii) Chứng minh tương tự như (i).

Định nghĩa 7. *Trọng đồ* (có hướng) là đồ thị (có hướng) mà mỗi cạnh (cung) của nó được gán một số.

Trọng đồ được biểu diễn bởi G=(V, E, w), trong đó V là tập các đỉnh, E là tập các cạnh (cung) và $w:E \rightarrow R$ là hàm số trên E, w(e) là trọng số của cạnh (cung) e với mọi $e \in E$.

Trong trọng đồ *độ dài trọng số của đường đi* μ là tổng các trọng số của các cạnh (cung) trên đường đi đó.

Ví dụ 8: Mô phỏng trọng đồ bên dưới



Hình 6.10 Đồ thị gán trọng số

Định nghĩa 8.

Cho đồ thị G=(V, E). Đồ thị G'= (V', E') gọi là *đồ thị con* của G nếu

$V' \subset V$ & $E' \subset E$ Nếu V' = V, thì G' gọi là *đồ thị con phủ* của G.

Nếu $F \subset E$, thì ký hiệu G–F là đồ thị con (V, E–F) của G gồm tập đỉnh V và tập cạnh (cung) E – F.

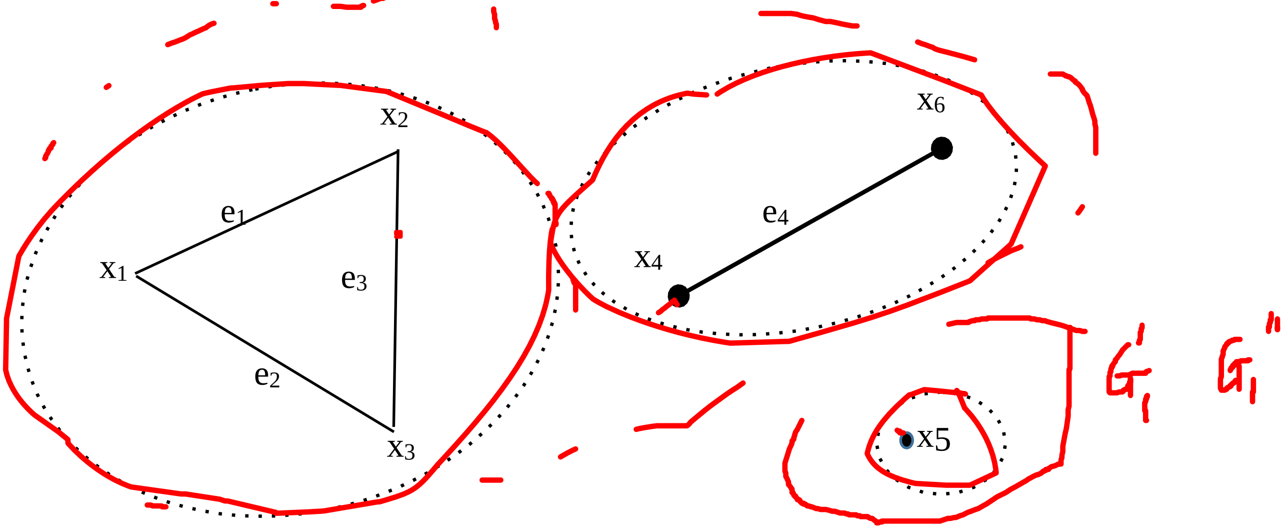
Nếu $U \subset V$, thì ký hiệu G–U là đồ thị con của G thu được từ G sau khi loại bỏ các đỉnh trong U và các cạnh liên thuộc chúng.

Cho $U \subset V$. Đồ thị con của G *sinh bởi* U, ký hiệu <U>, là đồ thị (U, E_U) với $E_U = \{e \in E \mid e$ liên thuộc đỉnh trong $U\}$

Đồ thị con G'=(V',E') của đồ thị (có hướng) G=(V,E) gọi là *thành phần liên thông* (*mạnh*) của đồ thị G, nếu nó là đồ thị con liên thông (mạnh) tối đại của G, tức là không tồn tại đồ thị con liên thông (mạnh) G"=(V",E") ≠ G' của G thoả $V' \subset V"$, $E' \subset E"$.

Ví dụ 9. Xét đồ thị G=(V,E).

G1



Hình 6.11 Các thành phần liên thông

Đồ thị $G_0 = (V_0, E_0)$, với $V_0 = \{x_1, x_2, x_3\}$ và $E_0 = \{e_1, e_2\}$ là đồ thị con liên thông của đồ thị G, nhưng không phải thành phần liên thông của G.

Đồ thị $G_1 = (V_1, E_1)$, với $V_1 = \{x_1, x_2, x_3\}$ và $E_1 = \{e_1, e_2, e_3\}$ là thành phần liên thông của G.

Đồ thị $G_2 = (V_2, E_2)$, với $V_2 = \{x_4, x_6\}$ và $E_2 = \{e_4\}$ là thành phần liên thông của G.

Đồ thị $G_3 = (\{x_5\}, \varnothing)$ là thành phần liên thông của G. Đồ thị G có 3 thành phần liên thông.

Ghi chú: Đồ thị liên thông khi và chỉ khi số thành phần liên thông của nó bằng 1.

Định nghĩa 9.

Cho đồ thị G=(V, E, *w*). Ta định nghĩa *khoảng cách* từ *u* đến *v*, $\forall u, v \in$ V, là độ dài đường đi ngắn nhất từ *u* đến *v* và ký hiệu là $d(u, v)$.

Đại lượng

$$e(v) = max\{d(v, w) \mid w \in V\}$$ gọi là *độ lệch tâm* của đỉnh *v*, $\forall v \in$ V.

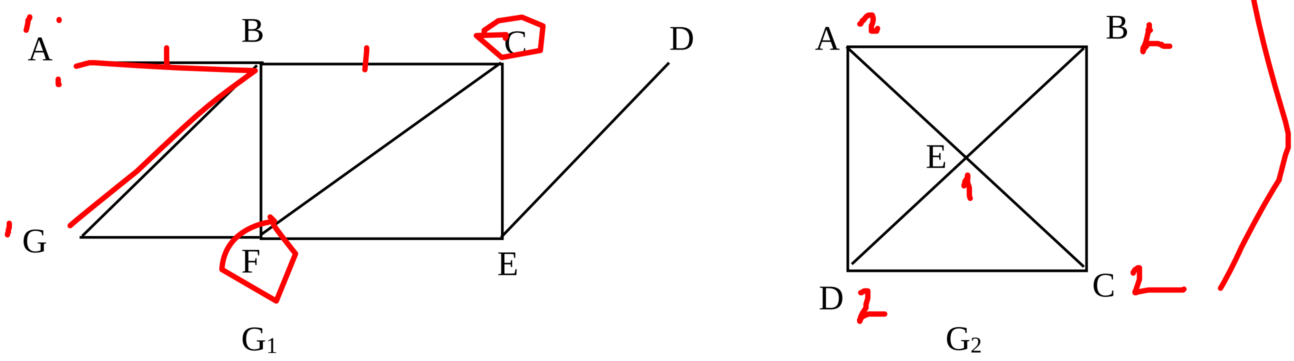
Bán kính của đồ thị G, ký hiệu $r(G)$, là độ lệch tâm nhỏ nhất

$$r(\mathrm{G}) = min\{e(v) \mid v \in V\}$$

Đỉnh $v \in$ V gọi là *đỉnh tâm* nếu $e(v) = r(\mathrm{G})$. Tập hợp tất cả các đỉnh tâm gọi là *tâm* của đồ thị và ký hiệu là C(G).

Ví dụ 10.

Xét các đồ thị sau (trọng số các cạnh bằng 1)



Hình 6.12 Đồ thị G_1 và G_2 có trọng số bằng 1

$e(A) = max\ (d(A, w = \{B, C, D, G, F, E)$

Độ lệch tâm các đỉnh A, B, C, D, E, F, G của đồ thị G_1 tương ứng là 4, 3, 2, 4, 3, 2, 3. Suy ra bán kính $r(G_1)$ = 2, các đỉnh tâm là C và F, và tâm $C(G_1)$ ={C,F}.

Độ lệch tâm các đỉnh A, B, C, D, E của đồ thị G_2 tương ứng là 2, 2, 2, 2, 1.

Suy ra bán kính $r(G_2)$ = 1, đỉnh tâm duy nhất là E, và tâm $C(G_2)$ = {E}.

6.2 Biểu diễn đồ thị

6.2.1 Ma trận kề

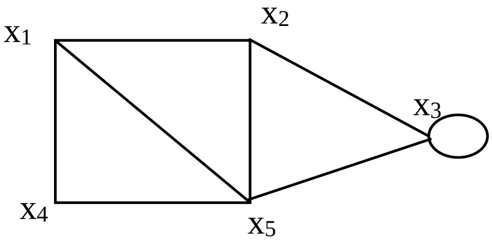
$= max\{1, 2, 4, 2, 3, 3\} = 4$

a) Đồ thị vô hướng

Định nghĩa 10. Cho đồ thị vô hướng G=(V,E) có *n* đỉnh theo thứ tự $v_1, v_2, \ldots, v_n$. *Ma trận kề* của đồ thị G là ma trận vuông $A=(a_{ij})_{n\times n}$, trong đó a_{ij} là số cạnh (khuyên) nối v_i với v_j. Lưu ý rằng khi tính bậc của đỉnh mỗi khuyên được tính hai bậc.

Từ định nghĩa suy ra rằng ma trận kề của đồ thị vô hướng luôn *đối xứng* qua đường chéo chính.

Ví dụ 11. Cho đồ thị như sau



Hình 6.13 Đồ thị vô hướng 5 đỉnh

Ma trận kề là

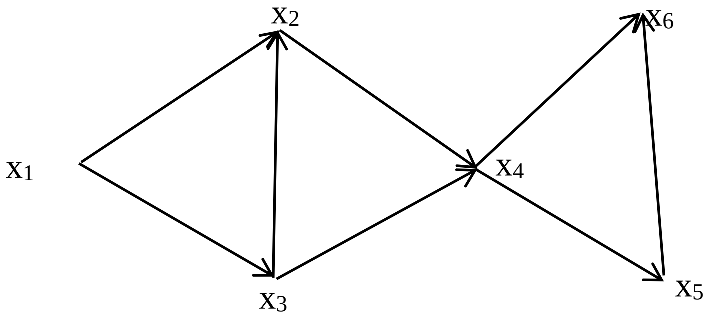
Bảng 6.1 Ma trận kề của đồ thị vô hướng

	x_1	x_2	x_3	x_4	x_5
x_1	0	1	0	1	1
x_2	1	0	1	0	1
x_3	0	1	1	0	1
x_4	1	0	0	0	1
v_5	1	1	1	1	0

b) Đồ thị có hướng

Định nghĩa 11. Cho đồ thị có hướng G=(V,E) có *n* đỉnh theo thứ tự v_1, v_2,..., v_n. *Ma trận kề* của đồ thị G là ma trận vuông A=$(a_{ij})_{n\times n}$, trong đó a_{ij} là số cung đi từ v_i tới v_j.

Ví dụ 12. Xét đồ thị có hướng sau



Hình 6.14 Đồ thị có hướng 6 đỉnh

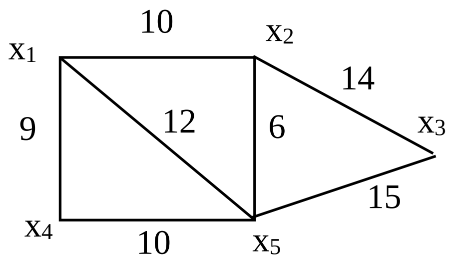
Đồ thị trên có ma trận kề là

Bảng 6.2 Ma trận kề của đồ thị có hướng

	x_1	x_2	x_3	x_4	x_5	x_6
x_1	0	1	1	0	0	0
x_2	0	0	0	1	0	0
x_3	0	1	0	1	0	0
x_4	0	0	0	0	1	1
x_5	0	0	0	0	0	1
x_6	0	0	0	0	0	0

Ghi chú: Nếu đồ thị G có trọng số cạnh (cung) $(i,j) \in G$ là w_{ij}, thì có thể thay ma trận kề bằng *ma trận trọng số* $(a_{ij})_{n\times n}$, trong đó $a_{ij} = w_{ij}$ nếu $(i,j)\in G$ và $a_{ij} = +\infty$ hoặc $-\infty$ (tùy theo bài toán cụ thể), nếu không tồn tại cạnh (cung) (i,j).

Ví dụ 13. Đồ thị có trọng số



Hình 6.15 Đồ thị vô hướng có trọng số

có ma trận trọng số là

Bảng 6.3 Ma trận trọng số của đồ thị

	x_1	x_2	x_3	x_4	x_5
x_1	∞	10	∞	9	12
x_2	10	∞	14	∞	6
x_3	∞	14	∞	∞	15
x_4	9	∞	∞	∞	10
x_5	12	6	15	10	∞

6.2.2 Ma trận liên thuộc

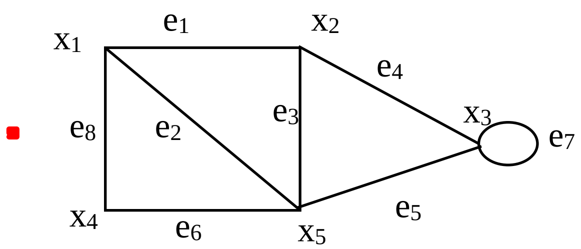
a) Đồ thị vô hướng

Định nghĩa 12. Cho đồ thị G=(V, E) có n đỉnh, $V=\{v_1, v_2, ..., v_n\}$ và m cạnh $E=\{e_1, e_2, ..., e_m\}$. *Ma trận liên thuộc* của đồ thị G là ma trận $A=(a_{ij})_{n\times m}$ thoả mãn

$a_{ij} = 1$, nếu đỉnh v_i liên thuộc cạnh e_j.

$a_{ij} = 0$, nếu đỉnh v_i không liên thuộc cạnh e_j

Ví dụ 14. Đồ thị



Hình 6.16 Đồ thị vô hướng có cạnh liên thuộc

Có ma trận liên thuộc là

Bảng 6.4 Ma trận liên thuộc của đồ thị vô hướng

	e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8
x_1	1	1	0	0	0	0	0	1
x_2	1	0	1	1	0	0	0	0
x_3	0	0	0	1	1	0	1	0
x_4	0	0	0	0	0	1	0	1
x_5	0	1	1	0	1	1	0	0

b) Đồ thị có hướng

Định nghĩa 13. Cho đồ thị có hướng không khuyên G=(V, E) có n đỉnh, V={v_1, v_2, ..., v_n} và m cung E={e_1, e_2, ..., e_m}. *Ma trận liên thuộc* của đồ thị G là ma trận $A=(a_{ij})_{nxm}$ thoả mãn

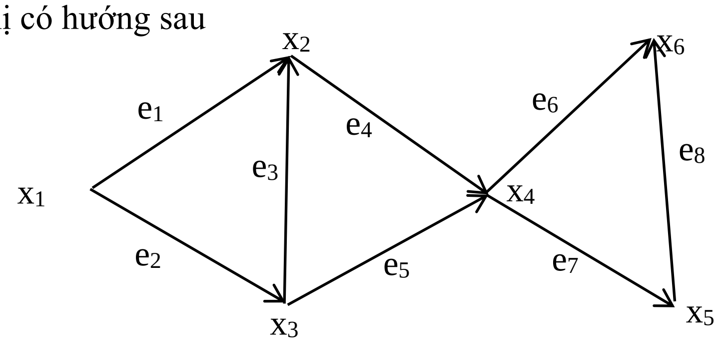
a_{ij} = 1, nếu đỉnh v_i là đỉnh đầu của cung e_j

a_{ij} = -1, nếu đỉnh v_i là đỉnh cuối của cung e_j

a_{ij} =0, nếu đỉnh v_i không liên thuộc cung e_j

Ví dụ 15.

Xét đồ thị có hướng sau



Hình 6.17 Đồ thị có hướng có cung liên thuộc

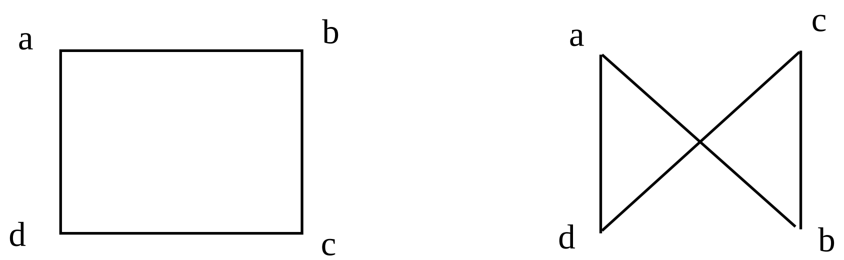
Đồ thị có ma trận liên thuộc là

Bảng 6.5 Ma trận liên thuộc của đồ thị có hướng

	e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8
x_1	1	1	0	0	0	0	0	0
x_2	-1	0	-1	1	0	0	0	0
x_3	0	-1	1	0	1	0	0	0
x_4	0	0	0	-1	-1	1	1	0
x_5	0	0	0	0	0	0	-1	1
x_6	0	0	0	0	0	-1	0	-1

6.3. Đồ thị đẳng cấu

Giả sử hai người nhận được yêu cầu sau: Cho 4 điểm a, b, c, d. Nối a với b, b với c, c với d và d với a. Rất có thể 2 người sẽ vẽ 2 đồ thị như sau



Hình 6.18 Hai đồ thị đẳng cấu

Hai đồ thị trên trông bề ngoài khác nhau, nhưng chúng thực chất là một. Đó là các đồ thị đẳng cấu.

Định nghĩa 14. Hai đồ thị $G_1 =(V_1, E_1)$ và $G_2 =(V_2, E_2)$ gọi là *đẳng cấu* với nhau, nếu tồn tại song ánh $f: V_1 \rightarrow V_2$ và $g: E_1 \rightarrow E_2$ thoả mãn

$$\forall e \in E_1: e = (v, w) \Leftrightarrow g(e)=(f(v), f(w))$$

Cặp ánh xạ (f, g) gọi là một *đẳng cấu* từ G_1 đến G_2. Từ định nghĩa suy ra

Mệnh đề 1. Hai đơn đồ thị $G_1 =(V_1, E_1)$ và $G_2 =(V_2, E_2)$ *đẳng cấu* với nhau nếu tồn tại song ánh $f:V_1 \rightarrow V_2$ thoả mãn

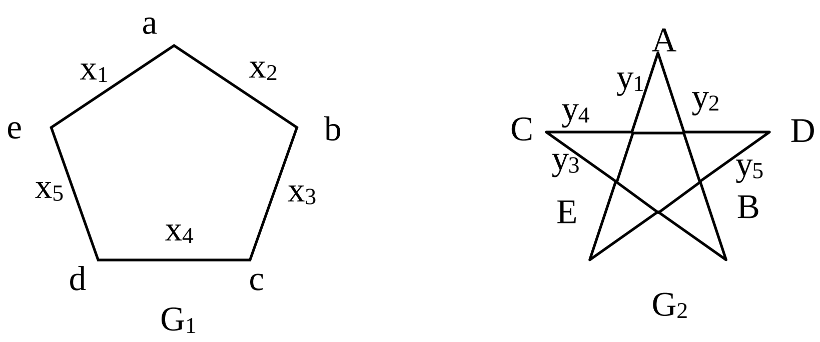
$$\forall v, w \in V_1 : v \text{ kề } w \Leftrightarrow f(v) \text{ kề } f(w) \text{ trong } G_2$$

Chứng minh

Ánh xạ $g: E_1 \rightarrow E_2$, $(u, v) \mapsto (f(u), f(v))$, cùng với ánh xạ f tạo thành một đẳng cấu từ G_1 vào G_2.

Trong trường hợp G_1 và G_2 là đơn đồ thị, hàm f trong mệnh đề 1 gọi là một *đẳng cấu* từ G_1 đến G_2.

Ví dụ 16. Cho hai đồ thị



Hình 6.19 Hai đồ thị đẳng cấu G_1 và G_2

Một đẳng cấu f từ G_1 vào G_2 là f(a)=A, f(b)=B, f(c)=C, f(d)=D, f(e)=E vì

$(a,b) \in G_1 \Leftrightarrow (f(a),f(b)) = (A,B) \in G_2$

$(b,c) \in G_1 \Leftrightarrow (f(b),f(c)) = (B,C) \in G_2$

$(c,d) \in G_1 \Leftrightarrow (f(c),f(d)) = (C,D) \in G_2$

$(d,e) \in G_1 \Leftrightarrow (f(d),f(e)) = (D,E) \in G_2$

$(a,e) \in G_1 \Leftrightarrow (f(a),f(e)) = (A,E) \in G_2$

6.4 Đồ thị phẳng

Định nghĩa 15.

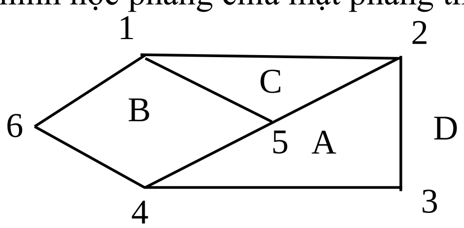
Một đồ thị gọi là *đồ thị hình học phẳng*, nếu nó được biểu diễn trên mặt phẳng sao cho các cạnh không cắt nhau.

Một đồ thị gọi là *đồ thị phẳng*, nếu nó đẳng cấu với đồ thị hình học phẳng.

Với một đồ thị hình học phẳng liên thông, mặt phẳng được chia làm các miền con gọi là *mặt*. Mỗi mặt được giới hạn bởi chu trình gọi là *biên* của mặt. Số cạnh trên biên của mặt *f* được gọi là *bậc* của mặt, ký hiệu *deg*(*f*). Bậc nhỏ nhất gọi là *đai* của đồ thị.

Ví dụ 17.

Đồ thị sau là đồ thị hình học phẳng chia mặt phẳng thành các miền A, B, C, D



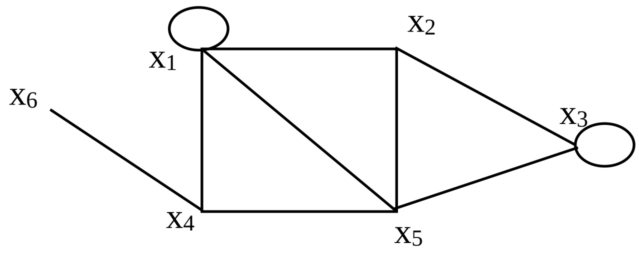
Hình 6.20 Đồ thị phẳng

Miền A có bậc 4, miền B có bậc 4, miền C có bậc 3, miền D có bậc 5. Đai của đồ thị là 3.

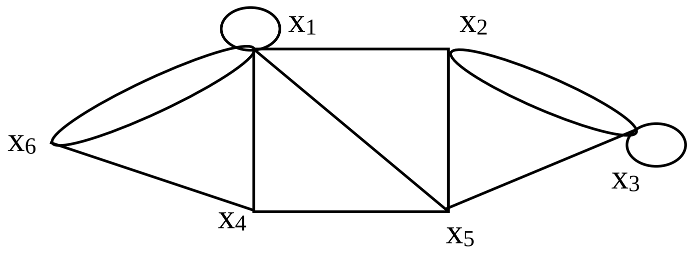
Bài tập chương 6

6.1 Hãy xác định số đỉnh, số cạnh, bậc của các đỉnh của những đồ thị vô hướng sau. Hãy chỉ ra những đỉnh cô lập, đỉnh treo, cạnh song song, khuyên (nếu có).

a)



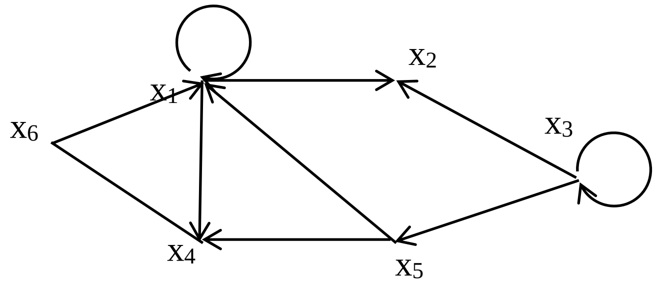
b)



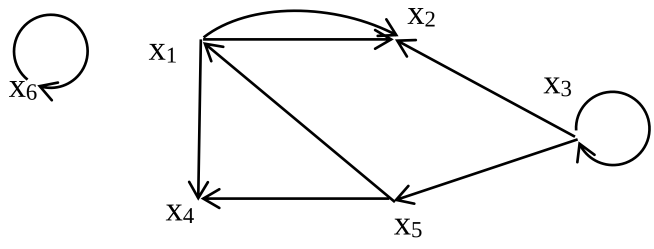
6.2 Tính tổng bậc các đỉnh trên các đồ thị bài tập **1**. Hãy kiểm tra xem tổng bậc các đỉnh có bằng hai lần số cạnh không.

6.3 Hãy xác định số đỉnh, số cung, nửa bậc vào và nửa bậc ra của các đỉnh của những đồ thị có hướng sau. Hãy chỉ ra những đỉnh cô lập, đỉnh treo, cung song song, khuyên (nếu có).

a)



b)



6.4 Tính tổng nửa bậc vào, tổng nửa bậc ra các đỉnh trên các đồ thị bài tập 3. Hãy kiểm tra xem các tổng nửa bậc các đỉnh có bằng số cạnh không.

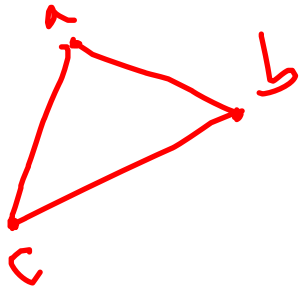
6.5 Hãy lập ma trận kề, ma trận liên thuộc của các đồ thị vô hướng cho ở bài tập 1

6.6 Hãy lập ma trận kề, ma trận liên thuộc của các đồ thị có hướng không khuyên cho ở bài tập 3

6.7 Hãy vẽ đồ thị có các ma trận kề sau

a)

	a	b	c
a	0	1	1
b	1	0	1
c	1	1	0



b)

0	2	1
2	0	3
1	3	0

c)

0	1	2	1
2	0	1	3
3	1	2	2
1	3	1	0

d)

0	3	0	1
3	0	2	0
0	2	1	1
1	0	1	0

6.8 Viết chương trình tính bậc (nửa bậc) của các đỉnh theo ma trận kề

6.9 Viết chương trình tính số cạnh của đồ thị vô hướng

6.10 Viết chương trình tính số cạnh của đồ thị có hướng

Chương 7. CÁC BÀI TOÁN VỀ ĐƯỜNG ĐI

Tóm tắt chương

Các bài toán về đường đi trên đồ thì có rất nhiều ứng dụng trong thực tế. Việc tìm các phương pháp giải các bài toán về đường đi là rất quan trọng được nhiều người quan tâm.

Chương này trình bày các chi tiết về đường đi Euler, các bài toán tìm đường đi ngắn nhất và đường đi chu trình Hamiton từ định nghĩa, định lý và các ví dụ cũng như các thuật toán liên quan về các đường đi trên đồ thị.

7.1 Đồ thị euler

7.1.1 Định nghĩa

Cho đồ thị G= (V, E).

Chu trình (đường đi) Euler là chu trình (đường đi) qua mọi cạnh và mọi đỉnh đồ thị, mỗi cạnh không quá một lần.

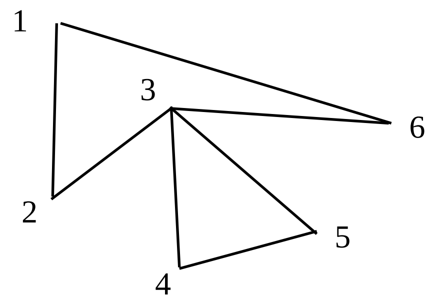
Cho đồ thị có hướng G= (V, E).

Chu trình có hướng Euler là chu trình có hướng qua mọi cung và mọi đỉnh đồ thị, mỗi cung không quá một lần.

Đường đi có hướng Euler là đường đi có hướng qua mọi cung và mọi đỉnh đồ thị, mỗi cung không quá một lần.

Đồ thị chứa chu trình Euler gọi là Đồ thị Euler.

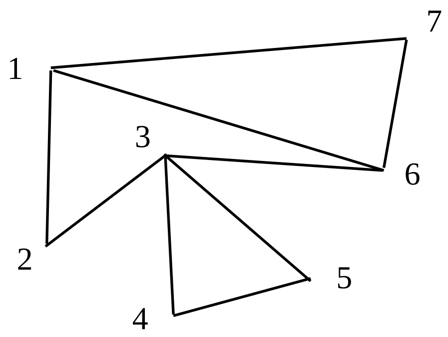
Ví dụ 1.



Hình 7.1 Biểu diễn chu trình Euler

Đồ thị trên có chu trình Euler là: 1-2-3-4-5-3-6-1

Ví dụ 2.

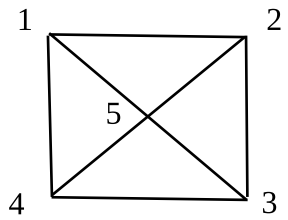


Hình 7.2 Biểu diễn đường đi Euler

Đồ thị trên có đường đi Euler là: 1-2-3-4-5-3-6-1-7-6

Nhưng không có chu trình Euler

Ví dụ 3.



Hình 7.3 Đồ thị không có chu trình và đường đi Euler

Đồ thị trên không có chu trình Euler và đường đi Euler

7.1.2 Điều kiện cần và đủ

Định lý 1 (**Định lý Euler*****)***

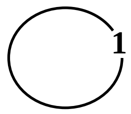
Đồ thị G có chu trình Euler khi và chỉ khi G liên thông và mọi đỉnh có bậc chẵn lớn hơn 0.

Chứng minh

(i) Chứng minh điều kiện cần (⇒): Giả sử G có chu trình Euler và v là đỉnh bất kỳ của G. Khi đó chu trình Euler đến v theo cạnh e vì theo định nghĩa chu trình Euler không được lặp lại cạnh nên chu trình này đi ra khỏi v bằng cạnh e’ thì e’ ≠ e. Do đó bậc của v phải là số chẵn. Vì chu trình Euler qua mọi đỉnh của G vì vậy G có duy nhất một thành phần liên thông suy ra G liên thông.

(ii) Chứng minh điều kiện đủ (⇐): Giả sử G liên thông và mọi đỉnh có bậc chẵn khác 0. Ta chứng minh G có chu trình Euler quy nạp theo số cạnh m của G.

* m = 1: Vì G liên thông và mọi đỉnh bậc chẵn nên G chỉ có 1 đỉnh và 1 khuyên. Khuyên đó cũng tạo thành chu trình Euler.



Hình 7.4 Đồ thị 1 đỉnh và 1 khuyên

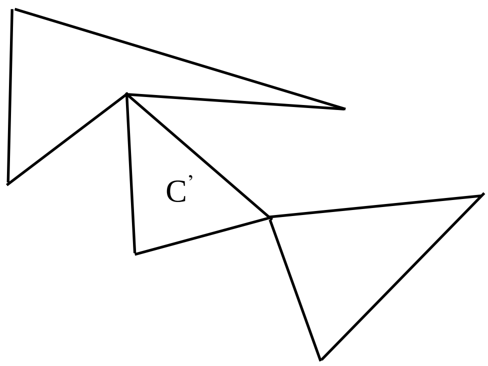
*

Giả sử G có m cạnh, số đỉnh n > 0 và mọi đồ thị liên thông có số cạnh nhỏ hơn m với mọi đỉnh bậc chẵn đều có chu trình euler.

+ Trường hợp n = 1 hoặc 2 mà số bậc chẵn nên hiển nhiên tồn tại chu trình Euler.

+ Trường hợp n > 2. Vì bậc của các đỉnh chẵn ≥ 2, bao giờ cũng chọn được một chu trình C’ trong G

- Xét đồ thị H thu được từ G bằng cách loại bỏ C’.



Hình 7.5 Đồ thị 8 đỉnh

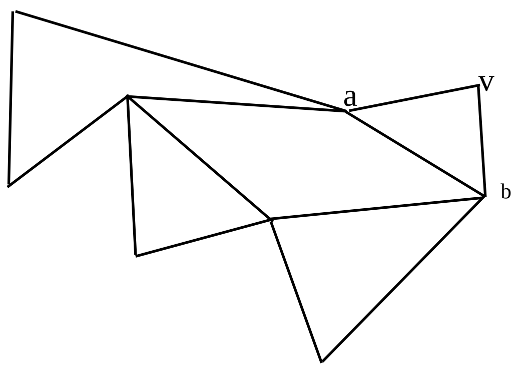
H có thể có nhiều thành phần liên thông

Vì số cạnh của H nhỏ hơn m (do đã loại đi số cạnh trong C’) và các đỉnh vẫn có bậc chẵn nên theo giả thiết quy nạp (số cạnh nhỏ hơn m là đúng) thì tồn tại chu trình Euler C’’ của mỗi thành phần liên thông trong H. Do G là đồ thị liên thông nên mỗi thành phần trong H có ít nhất 1 điểm chung với C’. Vì vậy ta có thể xây dựng Euler C trong G như sau: Xuất phát từ một đỉnh bất kỳ trong C’ đi theo các cạnh trong C’ cho đến khi nào gặp đỉnh chung của thành phần liên thông H thì đi theo các cạnh của chu trình Euler trong thành phần liên thông H chứa đỉnh đó. Sau đó lại tiếp tục đi theo cạnh của C’ cho đến khi gặp đỉnh chung với thành phần liên thông mới của H thi đi theo chu trình Euler ứng với thành phần liên thông đó. Cứ tiếp tục như vậy cho đến khi trở về điểm xuất phát ta thu được chu trình Euler C trong G (C’ được minh hoạ như hình trên)

Định lý 2. Đồ thị liên thông G có đường đi Euler khi và chỉ khi G có 2 đỉnh bậc lẻ.

Chứng minh:

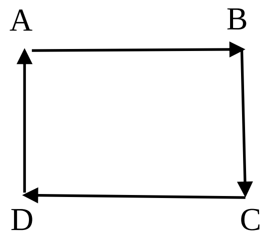
Thật vậy, nếu G có 2 đỉnh bậc lẻ a, b. Gọi G1 là đồ thị thu được từ G bằng cách thêm đỉnh v và 2 cạnh (v, a) và (v, b) (minh hoạ hình dưới). Lúc đó G1 đều có đỉnh bậc chẵn, theo định lý 1 ta có G1 đều có đỉnh bậc chẵn nên có chu trình Euler C. Xoá bỏ khỏi chu trình này đỉnh v và 2 cạnh kề với v ta thu được đường đi Euler.



Hình 7.6 Đồ thị 9 đỉnh

Định lý 3. Đồ thị có hướng G có chu trình có hướng Euler khi và chỉ khi G liên thông yếu và mọi đỉnh có nửa bậc vào bằng nửa bậc ra

Chứng minh: (Tương tự định lý 1) Bài tập.



Hình 7.7 Đồ thị có hướng 4 đỉnh

Đồ thị trên là đồ thị có hướng và liên thông yếu, đồ thị có chu trình Euler A, B, C, D, A

7.1.3 Các thuật toán tìm chu trình Euler

a) Dựa vào cách chứng minh định lý 1 ta có thuật toán sau tìm chu trình Euler

- Đầu vào. Đồ thị G ≠ ∅, không có đỉnh cô lập.
- Đầu ra. Chu trình (CT) Euler C của G, hoặc kết luận G không có chu trình Euler.
- Phương pháp

Bước 1. Khởi tạo: Đặt H = G, k = 1, C = ∅, v ∈ G.

Bước 2. Xuất phát từ v, xây dựng chu trình bất kỳ C_k trong H.

Nếu tồn tại C_k, nối C_k vào C, C := C $\cup C_k$ và sang bước (3).

Nếu không tồn tại C_k, thì kết luận không có chu trình Euler, kết thúc.

Bước 3. Loại chu trình C_k khỏi H. Nếu H chứa các đỉnh cô lập, thì loại chúng khỏi H. Sang bước (4).

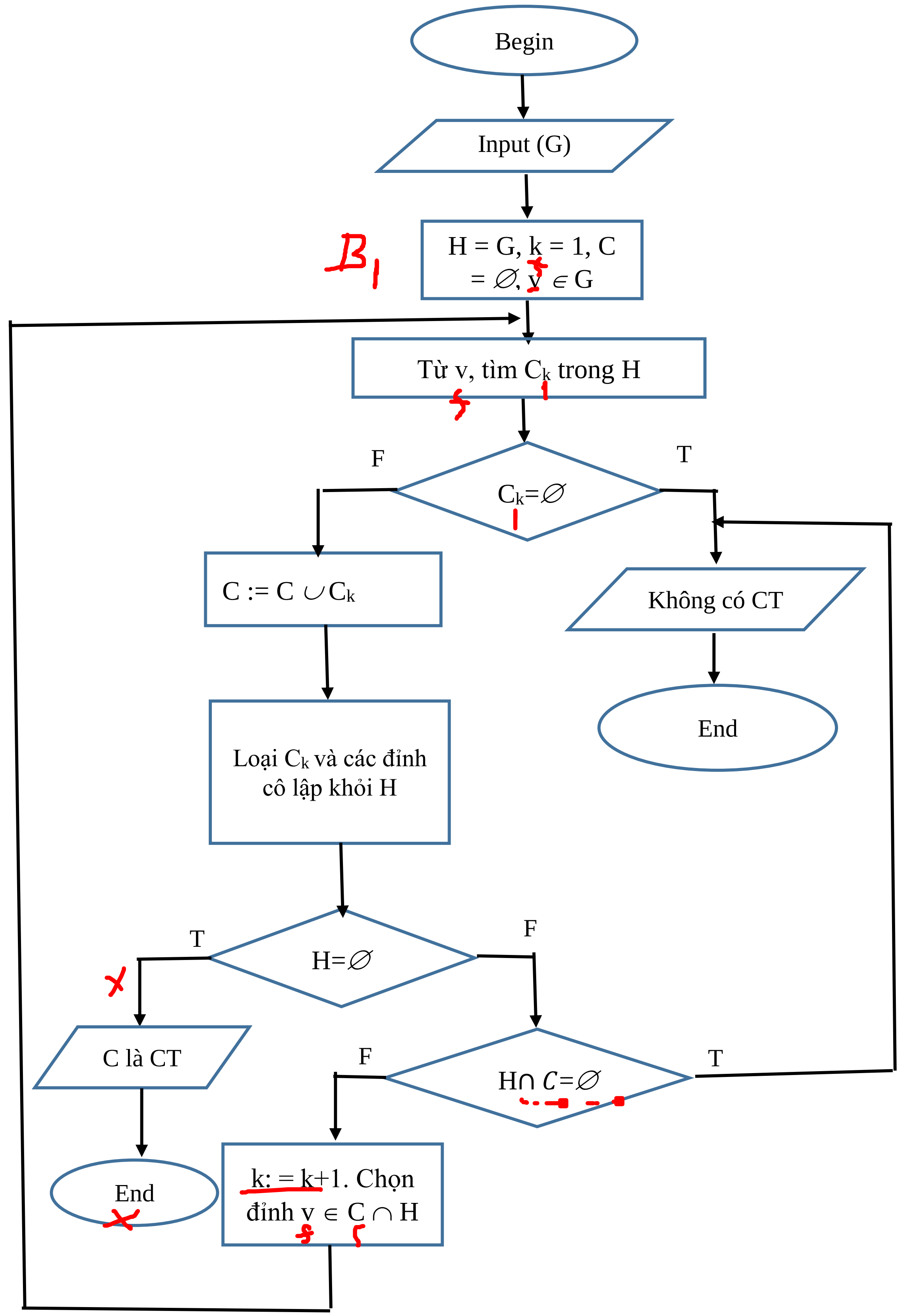
Bước 4. Nếu H = $\varnothing$, thì kết luận C là chu trình Euler, kết thúc. Ngược lại, sang bước (5).

Bước 5. Nếu H và C không có đỉnh chung, thì kết luận không có chu trình Euler, kết thúc.

Nếu H và C có đỉnh chung. Đặt k: = k+1. Chọn đỉnh v $\in$ C $\cap$ H bất kỳ.

Quay lại bước (2).

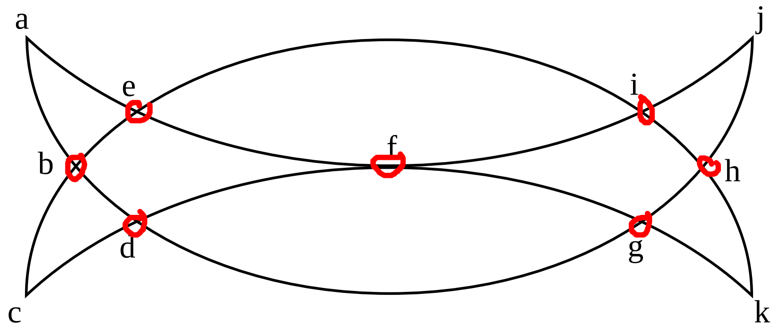
Thuật toán bằng sơ đồ khối



Hình 7.8 Sơ đồ khối cho thuật toán tìm chu trình Euler

Thuật toán được trình bày mã giả

Ví dụ 4. Cho G là đồ thị Thanh mã tấu Mohammed



Hình 7.9 Đồ thị thanh mã tấu Mohammed

Ta áp dụng thuật toán trên để tìm chu trình Euler.

Bước 1: Đặt H := G, k := 1, C := ∅, *v* := f.

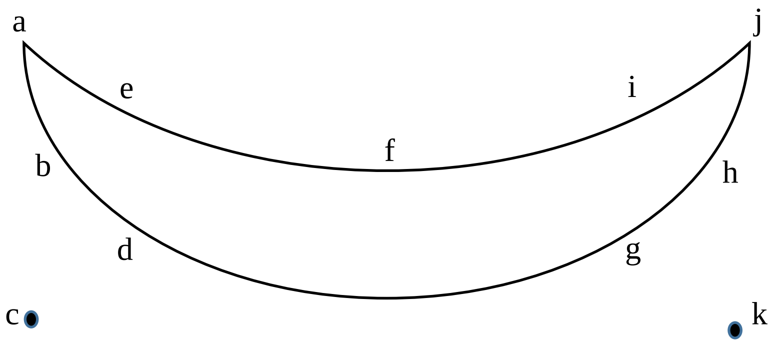
Bước 2: Ta xây dựng chu trình C1 trong H:

C1: = (f-g-k-h-i-e-b-c-d-f)

Đặt

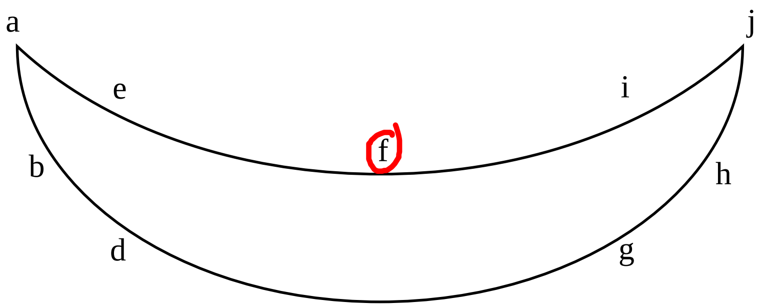
C := C ∪ C1 = (f-g-k-h-i-e-b-c-d-f)

Bước 3: Loại C1 ra khỏi H, ta được đồ thị H như sau



Hình 7.10 Đồ thị thanh mã tấu Mohammed đã xoá chu trình

Các đỉnh c và k là các đỉnh cô lập, vì thế ta loại chúng ra khỏi H và nhận được đồ thị H sau



Hình 7.11 Đồ thị thanh mã tấu Mohammed đã xoá đỉnh

Bước 5: Chọn đỉnh chung của H và C là *v*: = f. Đặt k: = k+1 = 2. Quay lại bước (2)

Bước 2: Ta xây dựng chu trình C2 trong H:

C2: = (f-i-j-h-g-d-b-a-e-f)

Nối C2 vào C ta được chu trình C sau

C := C ∪ C2 = (f-g-k-h-i-e-b-c-d-f) ∪ (f-i-j-h-g-d-b-a-e-f)

= (f-g-k-h-i-e-b-c-d-f-i-j-h-g-d-b-a-e-f)

Bước 3: Loại C2 ra khỏi H, ta được đồ thị H gồm toàn các đỉnh cô lập. Loại nốt các đỉnh cô lập ta có H = ∅.

Bước 4: Vì H = ∅, ta kết luận *C là chu trình Euler,* kết thúc.

b) Thuật toán Fleury

Xuất phát từ một đỉnh u nào đó của G ta đi theo các cạnh của nó một cách tuỳ ý chỉ cần tuân thủ 2 quy tắc sau:

i) Xoá bỏ cạnh đã đi qua và đồng thời xoá cả những đỉnh cô lập tạo thành.

ii) Ở mỗi bước ta chỉ đi qua cầu khi không còn cách lựa chọn nào khác

- Đầu vào. Đồ thị G ≠ ∅, không có đỉnh cô lập.
- Đầu ra. Chu trình Euler C của G, hoặc kết luận G không có chu trình Euler.
- Phương pháp.

Bước 1. Chọn đỉnh xuất phát bất kỳ v_0. Đặt v_1: = v_0, C:= {v_0}. H := G.

Bước 2. Nếu H = ∅, thì kết luận C là chu trình Euler, kết thúc. Ngược lại sang bước 3.

Bước 3. Chọn cạnh đi tiếp:

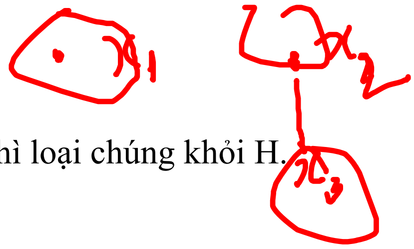
- Trường hợp đỉnh v_1 là đỉnh treo: Tồn tại duy nhất đỉnh v_2 kề v_1 .

Chọn cạnh (v_1 , v_2). Sang bước 4.

- Trường hợp đỉnh v_1 không là đỉnh treo:

Nếu mọi cạnh liên thuộc v_1 là cầu, thì không có chu trình Euler, kết thúc.

Ngược lại, chọn cạnh (v_1 , v_2) bất kỳ không phải là cầu trong H. Thêm vào đường đi C đỉnh v_2. Sang bước 4.



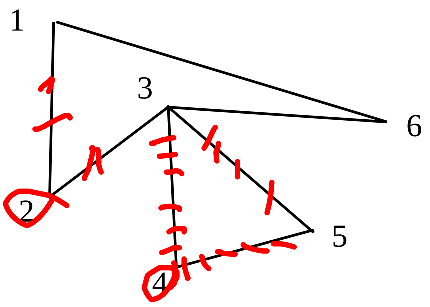
Bước 4. Xoá cạnh vừa đi qua, và xoá đỉnh cô lập:

Loại khỏi H cạnh (v_1, v_2). Nếu H có đỉnh cô lập, thì loại chúng khỏi H.

Đặt v_1:= v_2. Quay lại bước 2.

Ghi chú: Cạnh e được gọi là cầu nếu việc loại bỏ nó khỏi đồ thị làm tăng số thành phần liên thông của đồ thị (đỉnh cô lập k° được hiểu là TPLT)

Ví dụ 5. Tìm chu trình Euler bằng thuật toán Fleury với đồ thị sau



Hình 7.12 Đồ thị 6 đỉnh

Lượt 1

Bước 1: Chọn v_0=1, v_1=1, C={1}, H={1, 2, 3, 4, 5, 6; (12), (23), (34), (45), (53), (36), (16)}

Bước 2: H≠ ∅ sang bước 3

Bước 3: Chọn (1, 2) vì cạnh này không phải là cầu, C={1, 2}

Bước 4: H={1, 2, 3, 4, 5, 6; (23), (34), (45), (53), (36), (16)}, v_1=2. Quay lại bước 2

Lượt 2

Bước 2: vì H≠ ∅ sang bước 3

Bước 3: Chọn cạnh (2, 3) vì cạnh này không phải là cầu, C={1, 2, 3}

Bước 4: H={1, 3, 4, 5, 6; (34), (45), (53), (36), (16)}, v_1=3. Quay lại bước 2

Lượt 3

Bước 2: H≠ ∅ sang bước 3

Bước 3: Chọn cạnh (3, 4) vì cạnh này không phải là cầu, C={1, 2, 3, 4}

Bước 4: H={1, 3, 4, 5, 6; (45), (53), (36), (16)}, v_1=4. Quay lại bước 2

Lượt 4

Bước 2: H≠ ∅ sang bước 3

Bước 3: Chọn cạnh (4, 5) vì cạnh này không phải là cầu, C={1, 2, 3, 4, 5}

Bước 4: H={1, 3, 5, 6; (53), (36), (16)}, v_1=6. Quay lại bước 2

Lượt 5

Bước 2: H≠ ∅ sang bước 3

Bước 3: Chọn cạnh (5,3) vì cạnh này không phải là cầu, C={1, 2, 3, 4, 5,3}

Bước 4: H={1, 3, 6; (36), (16)}, v_1=3. Quay lại bước 2

Lượt 6:

Bước 2: H≠ ∅ sang bước 3

Bước 3: Chọn cạnh (3,6) vì cạnh này không phải là cầu, C={1, 2, 3, 4, 5,3,6}

Bước 4: H={1, 6; (16)}, v_1=6. Quay lại bước 2

Lượt 7:

Bước 2: H≠ ∅ sang bước 3

Bước 3: Chọn cạnh (6,1) vì cạnh này không phải là cầu, C={1, 2, 3, 4, 5,3,6,1}

Bước 4: H={∅}. Quay lại bước 2 (Vì H= ∅ nên kết thúc. C là chu trình Euler

7.2 Tìm đường đi ngắn nhất

7.2.1 Phát biểu bài toán

Bài toán tìm đường đi ngắn nhất trân đồ thị là một bài toán rất kinh điển từ trước đến nay. Có một số thuật toán kinh điển để giải quyết bài toán này, trong phần này ta sẽ tìm hiểu 3 thuật toán đó là thuật toán *Dijkstra, thuật toán Floyd và thuật toán Floyd mở rộng*. Riêng thuật toán Dijkstra có hai trường hợp, trường hợp 1 là tìm đường đi ngắn ngất giữa 2 đỉnh trên đồ thị và trường hợp 2 là tìm đường đi ngắn nhất từ đỉnh 1 đến tất cả các đỉnh còn lại trên đồ thị.

7.2.2 Thuật toán Dijkstra

Thuật toán Dijkstra tìm đường đi ngắn nhất từ 1 đỉnh đến tất cả các đỉnh còn lại trong đồ thị được thực hiện như sau:

Đầu vào: Đồ thị liên thông $G(V, E, w)$, $w(i, j) \geq 0 \ \forall (i, j) \in E$, đỉnh nguồn *a, đỉnh đích z*.

Đầu ra: *L(z) chiều dài đường đi ngắn nhất từ a đến z, và đường đi ngắn nhất* (nếu $L(z) < +\infty$).

Bước 1. Khởi tạo

Gán $L(a):=0$. *Với mọi đỉnh* $x \neq a$ *gán* $L(x) := \infty$. *Đặt* $T := V$.

Gán P(*x*) := ∅, ∀*x*∈*V* (*P(x)* *là đỉnh trước đỉnh x trên đường đi ngắn nhất* từ *a đến x).*

Bước 2. Tính *m* := *min*{*L(u)* | *u* ∈ *T*}.

Nếu *m* = + ∞, *kết luận không tồn tại đường đi từ a đến z. Kết thúc.*

Ngược lại, nếu m < + ∞, *chọn v* ∈ *T sao cho L(v) = m, và đặt* T := T - {*v*} Sang bước 3.

Bước 3. Kiểm tra điều kiện

- Nếu *z* = *v*,

+*L(z) là chiều dài đường đi ngắn nhất từ a đến z.*

+ Từ *z lần ngược theo đỉnh được ghi nhớ trong hàm P(x) ta tìm được đường* đi ngắn nhất như sau:

Đặt z_1 = P(z), z_2= P(z_1), …, z_k= P(z_{k-1}), *a* = *P*(z_k). Suy ra đường đi ngắn nhất là $a \rightarrow z_k \rightarrow z_{k-1} \rightarrow \ldots \rightarrow z_1 \rightarrow z$ *Kết thúc.*

- Ngược lại, nếu *z*≠*v, sang bước 4.*

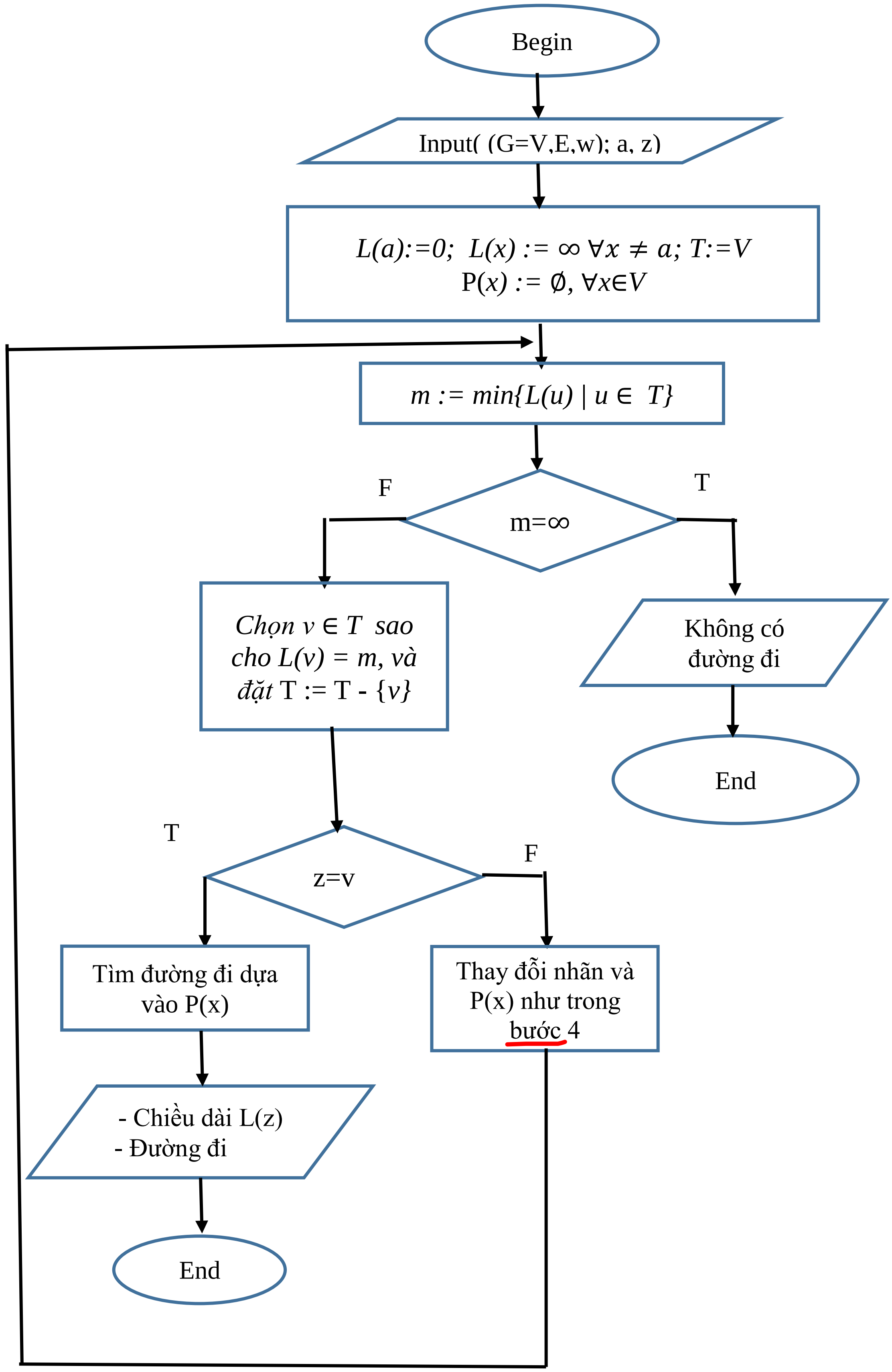
Bước 4. Thay đổi nhãn

Với mỗi *x*∈*T kề (kề sau đối với đồ thị có hướng) v, tức là tồn tại cạnh* (cung đối với đồ thị có hướng) (*v,x), nếu*

Nếu (L(*x*) > *L(v)+w(v,x)*) thì L(*x*) := *L(v)+w(v,x)*

và ghi nhớ đỉnh *v cạnh đỉnh x, gán P(x) := v (để sau này xây dựng đường* đi ngắn nhất). Quay về bước (2).

Thuật toán bằng sơ đồ khối



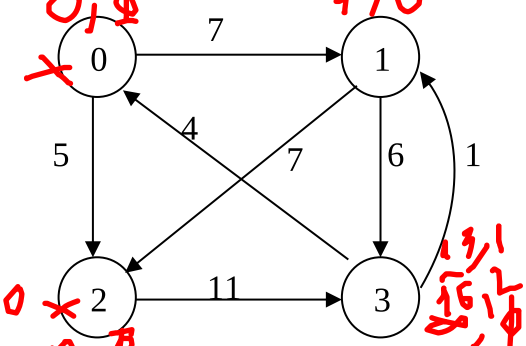
Hình 7.13 Sơ đồ thuật toán Dijkstra

Các bước trên được thực hiện bằng giả lệnh sau đây:

```
for vϵV
        L[v]= vô cùng;
        Truoc[v]=chưa tồn tại (=-1);
        v đỉnh chưa xét
Đặt T= V
L[a]= 0
while T chưa rỗng:
        chọn v= nút trong T chưa xét với khoảng cách nhỏ nhất(L[v] nhỏ nhất)
        T=T\{v}
        Đánh dấu đỉnh v đã xét
          for x kề với v:
         if L(x)>L(v)+w (v, x)
              {L(x)= L[v] + w(v, x)
              Truoc[x]= v}
End while
```

Độ phức tạp của thuật toán Dijktra là $O(n^2)$. Vì để chọn v có giá trị L(v) nhỏ nhất ta có số phép tính n và để gán lại nhãn trong bước 4 thì số phép tính cũng n. Vậy tổng số phép toán tích cực của thuật toán là n. $n=n^2$. Suy ra độ phức tạp là $O(n^2)$

Ví dụ 6. Tìm đường đi ngắn nhất của đỉnh 0 đến tất cả các đỉnh của đồ thị dưới đây:



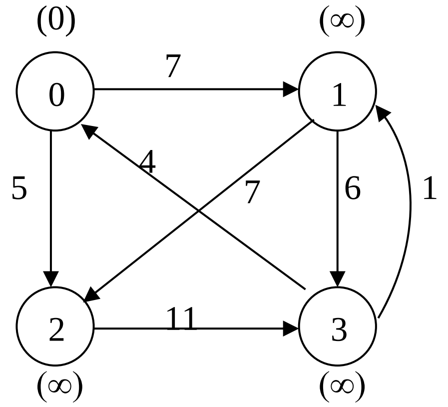
Hình 7.14 Đồ thị có hướng 4 đỉnh

- Thực hiện bước 1: 0 → 3 L(3) = 13 16 ≤ 7+6 = 13

Đặt: *T:={0, 1, 2, 3}*, đánh dấu các đỉnh chưa xét. *L(0):=0; L(1)=L(2)=L(3)=∞.*

3 → 1 → 0 ⇒ 0 → 1 → 3

Các tham số được biểu diễn trên đồ thị như sau:



Hình 7.15 Đồ thị gán nhãn khởi tạo

Các số trong ngoặc là $L(x)$, $x \in T$.

- Thực hiện bước 2:

$L(0)=min\{L(x) \mid x \in T,\ x$ chưa xét$\} = 0$.

Suy ra $T=T-\{0\}=\{1, 2, 3\}$ đánh dấu đỉnh 0 đã xét.

- Thực hiện bước 3: Vì $T \neq \phi$ nên sang *Bước 4*.
- Thực hiện bước 4:

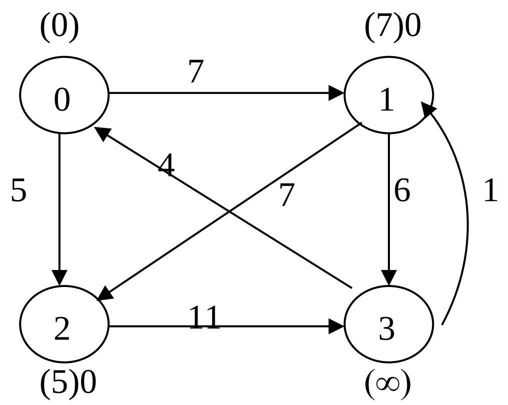
Đỉnh 1 và 2 kề đỉnh 0, ta có:

$L(1) =min\{L(1), L(0)+w(0, 1)\}=min\{ \infty, 0+7\} = 7.$

$L(2) =min\{L(2), L(0)+w(0, 2)\}=min\{ \infty, 0+5\} = 5.$

$L(1)$ và $L(2)$ thay đổi ghi nhớ đỉnh 0 vào đỉnh 1 và 2.

Các đỉnh khác không thay đổi, đồ thị có các nhãn như sau:



Hình 7.16 Đồ thị gán nhãn bước 1

- Thực hiện bước 2:

$L(2)=min\{L(x) \mid x \in T$, x chưa xét $\} = 5$, suy ra $T=T-\{2\}=\{1, 3\}$.

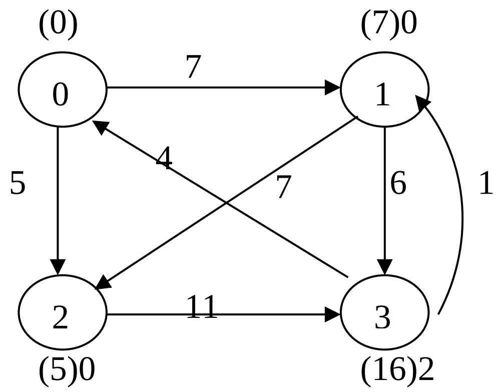
- Thực hiện bước 3: Vì $T \neq \phi$ nên sang *Bước 4*.

- Thực hiện bước 4: Đỉnh 3 kề đỉnh 2, ta có:

  $L(3)=min\{L(3), L(2)+w(2, 3)\}=min\{\infty, 5+11\} = 16$.

  L(3) thay đổi ghi nhớ đỉnh 2 vào đỉnh 3.

  Các đỉnh khác không thay đổi, đồ thị có các nhãn như sau:



Hình 7.17 Đồ thị gán nhãn bước 2

- Thực hiện bước 2: $L(1)=min\{L(x) \mid x \in T$, x chưa xét$\} = 7$, suy ra $T=T-\{1\}=\{3\}$.

- Thực hiện bước 3: Vì $T \neq \phi$ nên sang *Bước 4*.

- Thực hiện bước 4:

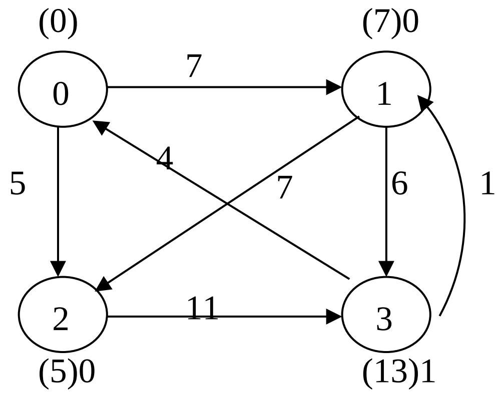
  Đỉnh 2, 3 kề đỉnh 1, ta có:

  $L(2)=min\{L(2), L(1)+w(1, 2)\}=min\{5, 7+7\} = 5$ (không thay đổi).

  $L(3)= min\{L(3), L(1)+w(1, 3)\}=min\{16, 7+6\} = 13$.

  L(3) thay đổi ghi nhớ đỉnh 1 vào đỉnh 3.

  Các đỉnh khác không thay đổi, đồ thị có các nhãn như sau:



Hình 7.18 Đồ thị gán nhãn bước 3

- Thực hiện bước 2:

$L(3)=min\{L(x) \mid x \in T,\ x$ chưa xét$\} = 13$, suy ra $T=T-\{4\}=\phi$.

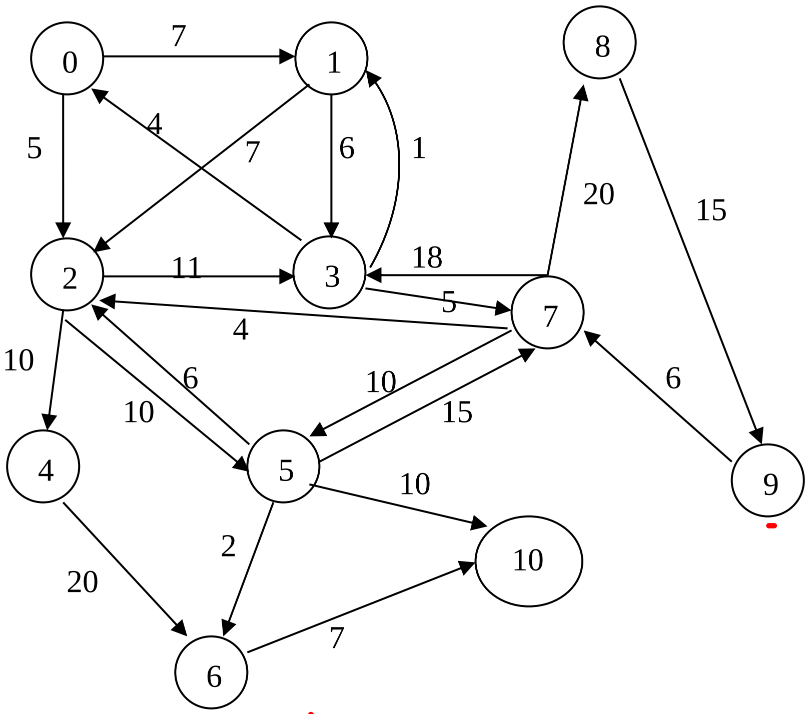
- Thực hiện bước 3 $T=\phi$, kết thúc.

Kết luận:

- Cặp *(0, 3) L(3) =13* là độ dài đường đi ngắn nhất từ đỉnh *0* đến đỉnh *3*. Từ đỉnh *3* ta đi ngược lại theo các đỉnh đã được ghi nhớ *3→1→0* ta suy ra đường đi ngắn nhất là *0→1→3*.

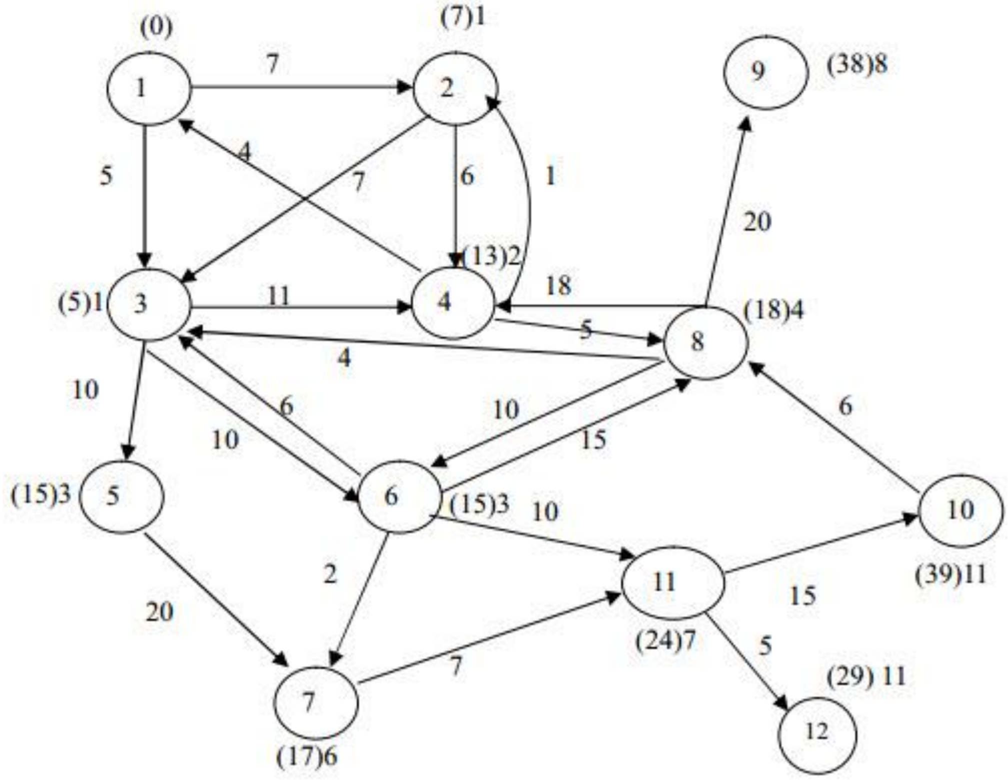
- Cặp *(0, 1) L(1)=7, 0→1*. Cặp *(0, 2), L(2)=5, 0→2*.

Ví dụ 7. Hãy gán nhãn theo thuật toán Dijktra cho tất cả các đỉnh của đồ thị dưới đây



Hình 7.19 Đồ thị có hướng 11 đỉnh

Khi thức hiện các thuật toán Dijktra thì các bước được gán nhãn như đồ thị sau:

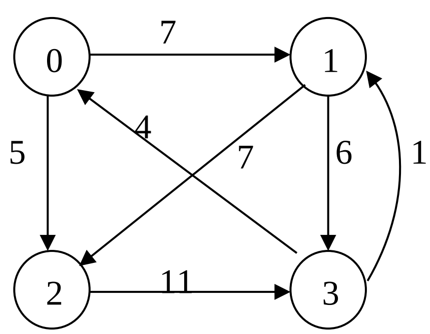


Hình 7.20 Đồ thị có hướng 12 đỉnh đã gán nhãn

Ghi chú: ta có thể dùng bảng để biểu diễn thực toán Dijktra như sau

- ✓ Ta lập bảng tính toán các nhãn gồm các cột ứng với các đỉnh, và các hàng ứng với các lần tính nhãn ở bước (4).
- ✓ Các nhãn gạch dưới ứng với nhãn nhỏ nhất ở bước (2), và đỉnh bị loại được ghi bên phải.
- ✓ Sau khi đỉnh z bị loại, từ z ta lần ngược về đỉnh a theo nhãn ghi trên bảng.
- ✓ Các đỉnh trên đường đi được gạch dưới (trên cột các đỉnh loại).
- ✓ Cuối cùng theo thứ tự ngược lại ta nhận được đường đi ngắn nhất.

Ví dụ 8. *Cho đồ thị sau*



Hình 7.21 Đồ thị dùng lập bảng

Hãy lập bảng để tính đường đi ngắn nhất từ 0 đến 3

Bảng 7.1 Tính đường đi ngắn nhất

Vòng	0	1	2	3	Đỉnh loại
1	0	∞	∞	∞	0
2	|	7(0)	5(0)	∞	2
3	|	7	|	16(2)	1
4	|	|	|	13 (1)	3 (Stop)

Vậy chiều dài đường đi ngắn nhất từ 0 đến 3 là 13. Đường di là 0 --> 1-->3

7.2.3 Thuật toán Floyd

Đầu vào: Đồ thị $G=(V, E, w)$, $V=\{1, 2,\dots,n\}$, có trọng số $w(i, j)$ với mọi cung (i, j)

Đầu ra: Ma trận $D=[d(i,j)]$, trong đó $d(i,j)$ là chiều dài đường đi ngắn nhất từ *i* đến *j* với mọi cặp (i,j).

Phương pháp:

Bước 1. (bước khởi tạo) ký hiệu D_0 là ma trận xuất phát.

- $D_0 = [d_0(i,j)]$. Trong đó $d_0(i,j)=w(i,j)$ nếu tồn tại cung (i,j) và $d_0(i,j)=+\infty$ nếu không tồn tại cung (i,j) (đặc biệt nếu không có khuyên tại i thì $d_0(i,j)= +\infty$).

- Gán $k=0$

Bước 2. Kiểm tra kết thúc. for k = 0 → n

- Nếu $k=n$, kết thúc. $D=D_n$ là ma trận độ dài đường đi ngắn nhất

- Ngược lại tăng *k* lên *1* đơn vị ($k:=k+1$) và sang (*3*).

Bước 3. Tính ma trận D_k theo D_{k-1}.

Với mọi cặp (i,j), $i=1,\dots,n$, $j=1\dots,n$ thực hiện:

- Nếu $d_{k-1}(i,j) > d_{k-1}(i,k)+ d_{k-1}(k,j)$ thì đặt

$$d_k(i,j):= d_{k-1}(i,k)+ d_{k-1}(k,j);$$

- Ngược lại đặt

$$d_k(i,j):= d_{k-1}(i,j);$$

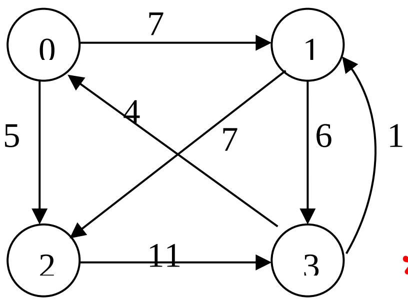
Quay lại bước (2).

Các bước trên được viết trong 3 vòng lặp for lồng nhau

```
for (k=0…n)
        for(i=0…n)
                for(j=0…n)
                        D[i][j]=min(D[i][j], D[i][k]+D[k][j]);
```

Độ phức tạp của thuật toán là $O(n^3)$. Vì ở bước 2 khi gán k ta có n phép tính. Ở bước 3 là tính lại các phần tử của ma trận nên có số phép tính là n^2. Vậy tổng số phép tính tích cực là n^3, suy ra độ phức tạp của thuật toán $O(n^3)$

Ví dụ 9. Tìm chiều dài của tất cả các cặp đỉnh theo thuật toán Floyd



Hình 7.22 Đồ thị biểu diễn thuật toán Floyd

Giải:

Ta có ma trận D_0

Bảng 7.2 Ma trận D_0 cho thuật toán Floyd

k=0

Đỉnh	0	1	2	3
0	vc	7	5	vc
1	vc	vc	7	6
2	vc	vc	vc	11
3	4	1	vc	vc

VC = +∞

$d(0,0) \boxed{\geq} d_{00} + d_{00}$

$d_{01} \boxed{\geq} d_{00} + d_{01}$

$7 > [\infty + 7]$

Ma trận D_1

Bảng 7.3 Ma trận D_1 cho thuật toán Floyd

Đỉnh	0	1	2	3
0	vc	7	5	vc
1	vc	vc	7	6
2	vc	vc	vc	11
3	4	1	9	vc

vc >

$d_{32}^{k=0} \boxed{} \geq d_{30} + d_{02}$

Ma trận D_2

Bảng 7.4 Ma trận D_2 cho thuật toán Floyd

Đỉnh	0	1	2	3
0	vc	7	5	13
1	vc	vc	7	6
2	vc	vc	vc	11
3	4	1	8	7

Ma trận D_3

Bảng 7.5 Ma trận D_3 cho thuật toán Floyd

Đỉnh	0	1	2	3
0	vc	7	5	13
1	vc	vc	7	6
2	vc	vc	vc	11
3	4	1	8	7

Ma trận D_4

Bảng 7.6 Ma trận D_4 cho thuật toán Floyd

Đỉnh	0	1	2	3
0	17	7	5	13
1	10	7	7	6
2	15	12	19	11
3	4	1	8	7

X

7.3.4 Thuật toán Floyd mở rộng (Floyd-Warshall)

Thuật toán Floyd-Warshall khác với thuật toán Floyd là trong thuật toán Floyd-Warshall có them ma trận P để tình đường đi

Đầu vào: Đồ thị *G=(V, E, w), V={1, 2,…,n}*, có trọng số *w(i, j)* với mọi cung *(i, j)*

Đầu ra: Ma trận *D=[d(i,j)]*, trong đó *d(i,j)* là chiều dài đường đi ngắn nhất từ *i* đến *j* với mọi cặp *(i,j)*. Ma trận *P=[p(i,j)]*, dùng để xác định đường đi ngắn nhất.

Phương pháp:

Bước 1. (bước khởi tạo) Ký hiệu D_0 là ma trận xuất phát.

- $D_0 = [d_0(i,j)]$. Trong đó $d_0(i,j)=w(i,j)$ nếu tồn tại cung (i,j) và $d_0(i,j)=+\infty$ nếu không tồn tại cung (i,j) (đặc biệt nếu không có khuyên tại i thì $d_0(i,j)= +\infty$).

- $P_0 = [p_0(i,j)]$. Trong đó $p_0(i,j)=j$ nếu có cung đi từ *i* đến *j* và $p_0(i,j)$ không xác định nếu không có cung đi từ *i* đến *j*.

- Gán $k=0$

Bước 2. Kiểm tra kết thúc.

- Nếu $k=n$, kết thúc. $D=D_n$ là ma trận độ dài đường đi ngắn nhất, $P=P_n$.

- Ngược lại tăng *k* lên *1* đơn vị ($k:=k+1$) và sang (*3*).

Bước 3. Tính ma trận D_k và P_k theo D_{k-1} và P_{k-1}.

Với mọi cặp (i,j), $i=1,\ldots,n$, $j=1\ldots,n$ thực hiện:

- Nếu $d_{k-1}(i,j) > d_{k-1}(i,k)+ d_{k-1}(k,j)$thì đặt

```
{
        dk(i,j):= dk-1(i,k)+ dk-1(k,j);
    pk(i,j):=pk-1(i,k);
}
```

- Ngược lại đặt

```
{
        dk(i,j):= dk-1(i,j);
    pk(i,j):=pk-1(i,j);
}
```

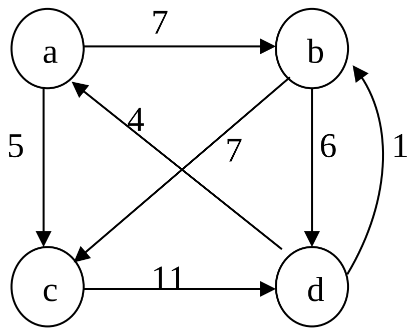
Quay lại bước (2).

Các bước trên được viết trong 3 vòng lặp for lồng nhau

```
for(int k=0;k<=n;k++)
        for(int i=0;i<=n;i++)
                for(int j=0;j<=n;j++)
                        if(D[i][j]>D[i][k]+D[k][j])
                                {
                                   D[i][j]=D[i][k]+D[k][j];
                                   P[i][j]=P[i][k];
                                }
```

Phương pháp xác định đường đi ngắn nhất từ đỉnh i đến đỉnh j, đường đi ngắn nhất từ i đến j gồm dãy các đỉnh. $i, i_1, i_2, i_3, \ldots, i_k, i_{k+1}, \ldots, i_m, j$ thỏa mãn: $i_1=p(i,j)$, $i_2=p(i_1,j)$, …, $i_{k+1}=p(i_k,j)$, …, $p(i_m,j)=j$.

Ví dụ 10. Cho đồ thị G như sau



$p(i,j)=j$

Hình 7.23 Đồ thị biểu diễn thuật toán Floyd mở rộng

Ta có ma trận D_0, P_0 (với 2 đỉnh i, j không có cạnh thì d(i, j)=vc và P(i,j)=null (n))

Bảng 7.7 Ma trận D_0, P_0 cho thuật toán Floyd mở rộng

Đỉnh	a	b	c	d
a	vc	7	5	vc
b	vc	vc	7	6
c	vc	vc	vc	11
d	4	1	vc	vc

$P_0=$

Đỉnh	a	b	c	d
a	n	b	c	n
b	n	n	c	d
c	n	n	n	d
d	a	b	n	n

P_0

$P(i,j)=P(i,k)$

Ma trận D_1, P_1

Bảng 7.8 Ma trận D_1, P_1 cho thuật toán Floyd mở rộng

Đỉnh	a	b	c	d
a	vc	7	5	vc
b	vc	vc	7	6
c	vc	vc	vc	11
d	4	1	9	vc

$P_1=$

Đỉnh	a	b	c	d
a	n	b	c	n
b	n	n	c	d
c	n	n	n	d
d	a	b	a	n

$p(4,3) = P_{01}$

$P(3,2) = P(3,0) = a$

Ma trận D_2, P_2

Bảng 7.9 Ma trận D_2, P_2 cho thuật toán Floyd mở rộng

Đỉnh	a	b	c	d
a	vc	7	5	13
b	vc	vc	7	6
c	vc	vc	vc	11
d	4	1	8	7

P_2=

Đỉnh	a	b	c	d
a	n	b	c	b
b	n	n	c	d
c	n	n	n	d
d	a	b	b	b

Ma trận D_3, P_3

Bảng 7.10 Ma trận D_3, P_3 cho thuật toán Floyd mở rộng

Đỉnh	a	b	c	d
a	vc	7	5	13
b	vc	vc	7	6
c	vc	vc	vc	11
d	4	1	8	7

P_3=

Đỉnh	a	b	c	d
a	n	b	c	b
b	n	n	c	d
c	n	n	n	d
d	a	b	b	b

Ma trận D_4, P_4

Bảng 7.11 Ma trận D_4, P_4 cho thuật toán Floyd mở rộng

Đỉnh	a	b	c	D
a	17	7	5	13
b	10	7	7	6
c	15	12	19	11
d	4	1	8	7

P_4=

Đỉnh	a	b	c	d
a	b	b	c	b
b	d	d	c	d
c	d	d	d	d
d	a	b	b	b

Truy ra đường đi: từ a đến d: P(a,d)=b; P(b,d)=d

Vậy đường đi từ a đến d là: a-->b-->d và có độ dài 13

```
#include <iostream>
using namespace std;
int vc=1000;
int D[4][4]={{vc,7,5,vc},{vc,vc,7,6},{vc,vc,vc,11},{4,1,vc,vc}};
char P[4][4]={{'n','2','3','n'},{'n','n','3','4'},{'n','n','n','4'},{'1','2','n','n'}};
void inmatran(int D[4][4])
{
	int n=3;
```

```
    for(int i=0;i<=n;i++)
            {
                    for(int j=0;j<=n;j++)
                            if(D[i][j]==vc)cout<<"vc ";
                            else if(D[i][j]<<10) cout<<D[i][j]<<"  ";
                            else cout<<D[i][j]<<" ";
                     cout<<endl;
            }
}
void inmatran(char P[4][4])
{
    int n=3;
    for(int i=0;i<=n;i++)
                    {
                            for(int j=0;j<=n;j++)
                                    cout<<P[i][j]<<" ";
                     cout<<endl;
                    }
}
void floydw(int n)
{
    for(int k=0;k<=n;k++)
    {
            for(int i=0;i<=n;i++)
                    for(int j=0;j<=n;j++)
                            if(D[i][j]>D[i][k]+D[k][j])
                                    {
                                       D[i][j]=D[i][k]+D[k][j];
                                            P[i][j]=P[i][k];
                                    }
    cout<<"........Ma tran D"<<k+1<<endl;
    inmatran(D);
    cout<<"........Ma tran P"<<k+1<<endl;
    inmatran(P);
    }
}
main()
    {
    cout<<"........Ma tran D"<<0<<endl;
    inmatran(D);
    cout<<"........Ma tran P"<<0<<endl;
    inmatran(P);
    Floydw(3);
    }
```

Ví dụ 11.

Cho đồ thị với ma trận trọng số sau, tìm ma trận D_6 và P_6

$$\begin{bmatrix} vc\ 7\ \ vc\ 2\ \ vc\ vc \\ vc\ vc\ 4\ \ vc\ 1\ \ vc \\ vc\ vc\ vc\ vc\ vc\ 3 \\ vc\ 4\ \ vc\ vc\ vc\ vc \\ 2\ \ vc\ 2\ \ vc\ vc\ vc \\ vc\ 1\ \ vc\ vc\ vc\ vc \end{bmatrix}$$

--------------------Ma tran P0--------------------

n 2 n 4 n n

n n 3 n 4 n

n n n n n 4

n 2 n n n n

1 n 3 n n n

n 2 n n n n

--------------------Ma tran D1--------------------

vc 7 vc 2 vc vc

vc vc 4 vc 1 vc

vc vc vc vc vc 3

vc 4 vc vc vc vc

2 9 2 4 vc vc

vc 1 vc vc vc vc

--------------------Ma tran P1--------------------

n 2 n 4 n n

n n 3 n 4 n

n n n n n 4

n 2 n n n n

1 1 3 1 n n

n 2 n n n n

--------------------Ma tran D2--------------------

vc 7 11 2 8 vc

vc vc 4 vc 1 vc

vc vc vc vc vc 3

vc 4 8 vc 5 vc

2 9 2 4 10 vc

vc 1 5 vc 2 vc

--------------------Ma tran P2--------------------

n 2 2 4 2 n

n n 3 n 4 n

n n n n n 4

n 2 2 n 2 n

1 1 3 1 1 n

n 2 2 n 2 n

--------------------Ma tran D3--------------------

vc 7 11 2 8 14

vc vc 4 vc 1 7

vc vc vc vc vc 3

vc 4 8 vc 5 11

2 9 2 4 10 5

vc 1 5 vc 2 8

--------------------Ma tran P3--------------------

n 2 2 4 2 2

n n 3 n 4 3

n n n n n 4

n 2 2 n 2 2

1 1 3 1 1 3

n 2 2 n 2 2

--------------------Ma tran D4--------------------

vc 6 10 2 7 13

vc vc 4 vc 1 7

vc vc vc vc vc 3

vc 4 8 vc 5 11

2 8 2 4 9 5

vc 1 5 vc 2 8

--------------------Ma tran P4--------------------

n 4 4 4 4 4

n n 3 n 4 3

n n n n n 4

n 2 2 n 2 2

1 1 3 1 1 3

n 2 2 n 2 2

--------------------Ma tran D5--------------------

9 6 9 2 7 12

3 9 3 5 1 6

vc vc vc vc vc 3

7 4 7 9 5 10

2 8 2 4 9 5

4 1 4 6 2 7

--------------------Ma tran P5--------------------

4 4 4 4 4 4

4 4 4 4 4 4

n n n n n 4

2 2 2 2 2 2

1 1 3 1 1 3

2 2 2 2 2 2

--------------------Ma tran D6--------------------

9 6 9 2 7 12

3 7 3 5 1 6

7 4 7 9 5 3

7 4 7 9 5 10

2 6 2 4 7 5

4 1 4 6 2 7

--------------------Ma tran P6--------------------

4 4 4 4 4 4

4 4 4 4 4 4

4 4 4 4 4 4

2 2 2 2 2 2

1 3 3 1 3 3

2 2 2 2 2 2

7.3 Đồ thị Hamilton

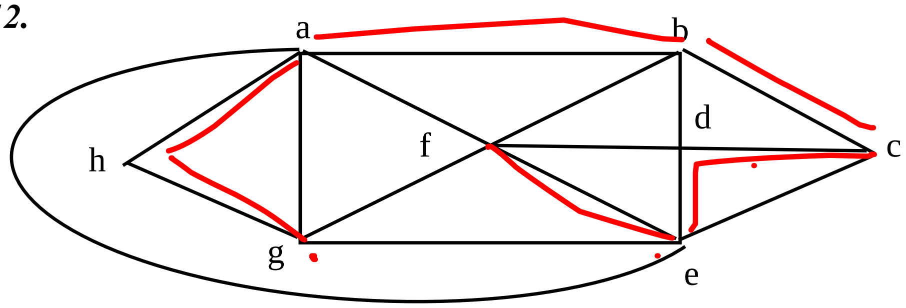
7.3.1 Định nghĩa

Cho đồ thị (có hướng) G=(V,E).

- Chu trình (có hướng) Hamilton là chu trình (có hướng) sơ cấp qua mọi đỉnh đồ thị.
- Đường đi (có hướng) Hamilton là đường đi (có hướng) sơ cấp qua mọi đỉnh đồ thị.
- Như vậy mọi chu trình Hamilton có độ dài bằng số đỉnh, và mọi đường đi Hamilton có độ dài bằng số đỉnh trừ 1.
- Đồ thị chứa chu trình (có hướng) Hamilton gọi là đồ thị Hamilton.

Ghi chú: Đường đi (chu trình) sơ cấp là đường đi (chu trình) không đi qua một đỉnh quá 1 lần

Ví dụ 12.



Hình 7.24 Đồ thị biểu diễn chu trình Hamilton

Đồ thị trên có 4 đỉnh a, c, e, f là các đỉnh bậc lẻ nên không có đường đi Euler nhưng có chu trình Hamilton: a-->b-->c-->d-->e-->f-->g-->h-->a

7.3.2 Điều kiện cần

Định lý 4.

Giả sử đồ thị G có chu trình Hamilton C. Khi đó

(i) Đồ thị G liên thông.

(ii) Mọi đỉnh của G có bậc lớn hơn hoặc bằng 2, và có đúng hai cạnh liên thuộc nằm trên chu trình C.

(iii) Nếu xóa đi *k* đỉnh bất kỳ cùng các cạnh liên thuộc chúng, thì đồ thị còn lại sẽ có tối đa *k* thành phần liên thông.

Chứng minh

- Mệnh đề (i) vì đồ thi G có chu trình Hamilton C nên C đi qua mọi đỉnh của đồ thị nên G liên thông
- Mệnh đề (ii) vì đồ thi G có chu trình Hamilton C mà chu trình này qua mỗi đỉnh đúng 1 lần nên có đúng 2 cạnh nằm trên chu trình C.

- Mệnh đề (iii) suy ra từ thực tế là khi xóa đi k đỉnh bất kỳ cùng các cạnh liên thuộc chúng thì chu trình C bị chia ra thành nhiều nhất k phần.

Hệ quả 1.

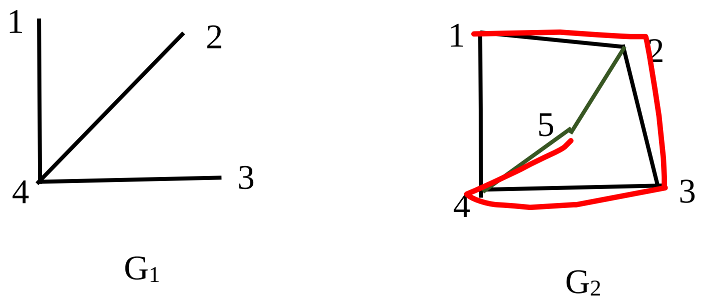
Giả sử đồ thị G có *n* đỉnh có đường đi Hamilton P. Khi đó

(i) Đồ thị G liên thông.

(ii) Có ít nhất *n*–2 đỉnh bậc ≥ 2, và mỗi đỉnh đó có đúng hai cạnh liên thuộc nằm trên đường đi P.

(iii) Nếu xóa đi *k* đỉnh bất kỳ cùng các cạnh liên thuộc chúng, thì đồ thị còn lại sẽ có tối đa *k*+1 thành phần liên thông.

Ví dụ 13. Đồ thị G_1 không có đường đi và chu trình Hamilton



Hình 7.25 Đồ thị G_1 và G_2

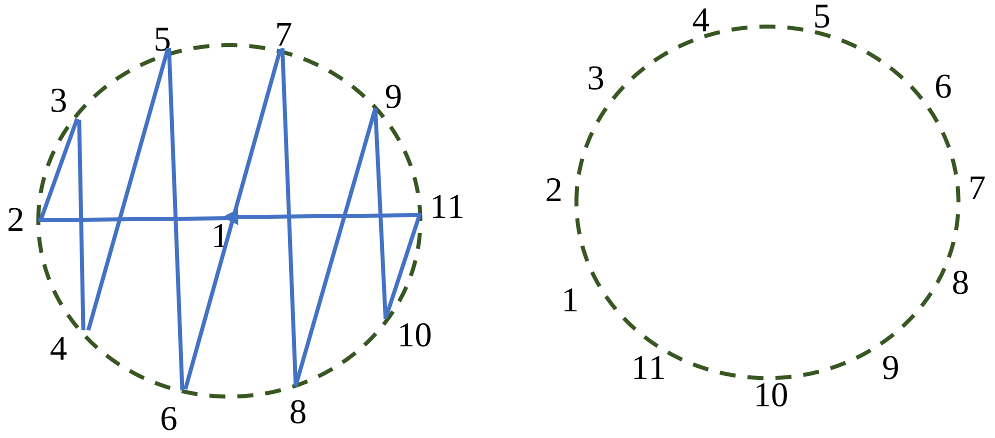
Đồ thị G_2 có đường đi: 1-2-3-4-5 nhưng không có chu trình vì đồ thị G_2 có số đỉnh bằng 5 mà theo định nghĩa nếu đồ thị G_2 có chu trình Hamilton C thì số cạnh của C bằng số đỉnh và phải bằng 5. Hơn nữa mỗi đỉnh trong C có đúng 2 cạnh thuộc C, nhưng vì đỉnh 2 và 4 có bậc 3 nên phải có 1 cạnh liên thuộc đỉnh 4 và 1 cạnh liên thuộc đỉnh 2 không thuộc C mà 2 cạnh này không trùng nhau nên đồ thị G_2 còn lại 4 cạnh thuộc C. Điều này mâu thuẫn với định nghĩa nên G_2 không có chu trình Hamilton C.

Hoặc ta có thể áp dụng ý (iii) của định lý 4 "Nếu xóa đi *k* đỉnh bất kỳ cùng các cạnh liên thuộc chúng, thì đồ thị còn lại sẽ có tối đa *k* thành phần liên thông". Thật vậy, nếu ta xoá đi đỉnh 2 và 4 cùng với các cạnh liên thuộc 2 đỉnh này thì ta thu được đồ thị G_2 còn lại 3 đỉnh cô lập nên G_2 có 3 thành phần liên thông. Suy ra G_2 không có chu trình Hamilton.

Ví dụ 14.

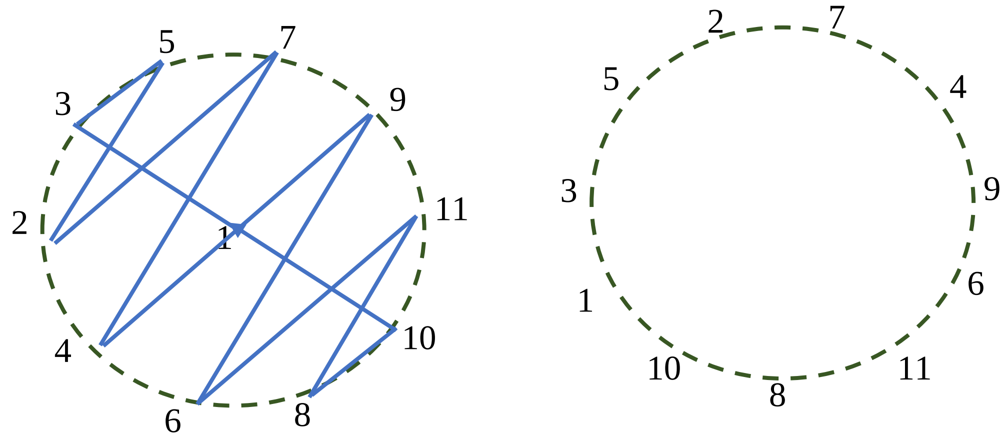
(Bài toán xếp chỗ ngồi) 11 người bạn cùng ngồi ăn trong bàn tròn 5 lần. Mỗi lần họ được xếp ngồi theo một thứ tự. Hãy thay đổi chỗ ngồi mỗi lần sao cho không có 2 người ngồi gần nhau hơn 1 lần.

Giải: Ta lập đồ thị 11 đỉnh 1, 2, ...,11, đỉnh i chỉ người i. Ta đặt đỉnh 1 tại tâm và các đỉnh còn lại trên đường tròn như hình vẽ. Mỗi cách xếp là một chu trình Hamilton của đồ thị. Chu trình thứ 1 như hình vẽ là: 1-2-3-4-5-6-7-8-9-10-11-1



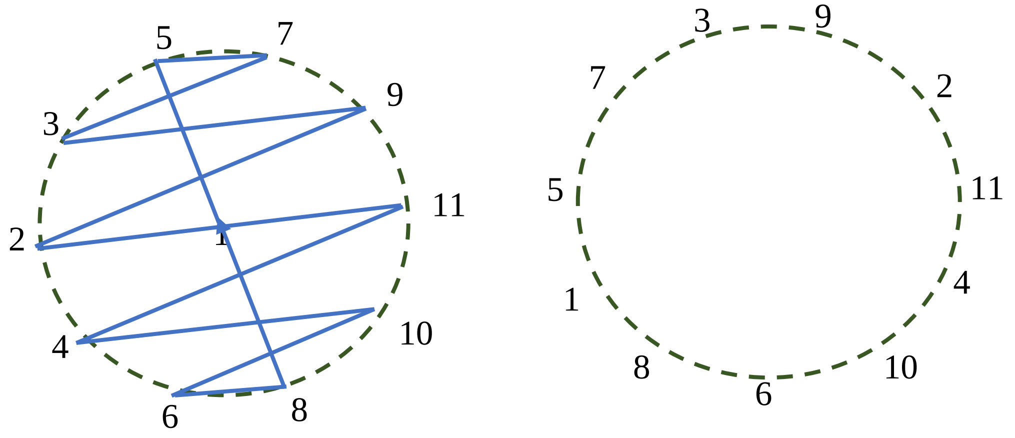
Hình 7.26 Đồ thị xếp chỗ ngồi lần 1

c chu trình, cũng là các cách xếp, chu trình thứ 2 như sau: 1-3-5-2-7-4-9-6-11-8-10-1



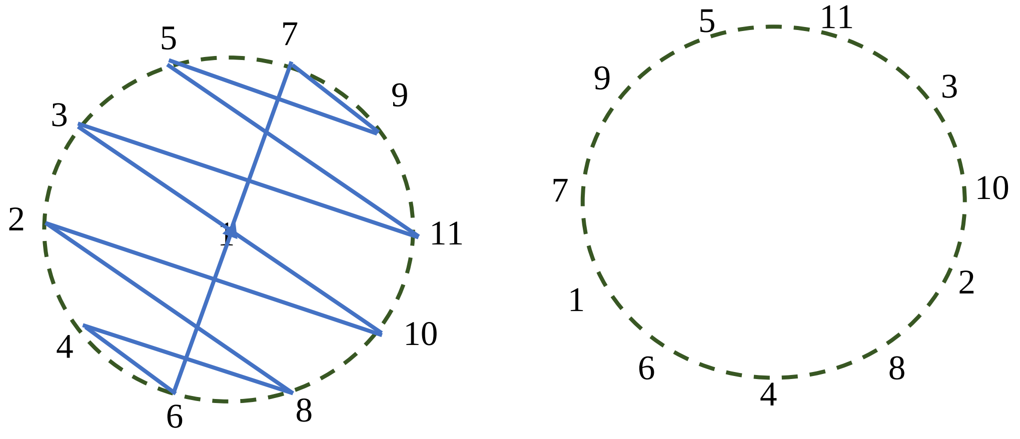
Hình 7.27 Đồ thị xếp chỗ ngồi lần 2

chu trình thứ 3 như sau: 1-5-7-3-9-2-11-4-10-6-8-1



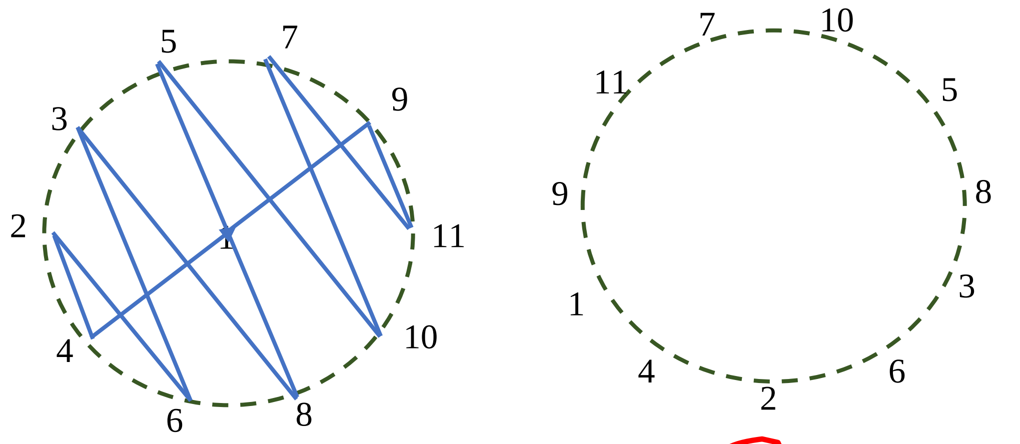
Hình 7.28 Đồ thị xếp chỗ ngồi lần 3

chu trình thứ 4 như sau: 1-7-9-5-11-3-10-2-8-4-6-1



Hình 7.29 Đồ thị xếp chỗ ngồi lần 4

Chu trình thứ 5 như sau: 1-9-11-7-10-5-8-3-6-2-4-1



Hình 7.30 Đồ thị xếp chỗ ngồi lần 5

7.3.3 Điều kiện đủ

Định lý 5.

Đồ thị đủ K_n với n lẻ ($n \geq 3$) có $(n-1)/2$ chu trình Hamilton từng đôi một không giao nhau (tức là không có cạnh chung).

Chứng minh

Tương tự như lời giải bài toán xếp 11 người trên bàn tròn, ta xây dựng cách xếp theo chu trình Hamilton trên đồ thị có n=2k+1:

Với chu trình đầu tiên là: $1 \to 2 \to 3 \to ... \to 2k \to 2k+1 \to 1$

Với ví dụ xếp 11 người trên bàn tròn thì n=2k+1 (k=5) tương đương với 5 cách xếp. Tổng quát ta xoay chu trình $1 \to 2 \to 3 \to ... \to 2k \to 2k+1 \to 1$ lần lượt một góc π/k theo chiều kim đồng hồ ta nhận được k chu trình.

***Định lý 6.* (**Dirak 1952)

Cho G là đơn đồ thị n đỉnh (n>2). Nếu bậc $\deg(v) \geq n/2$ với mọi đỉnh v của G, thì G có chu trình Hamilton.

Định lý 7 sau đây là tổng quát hoá của định lý 6 cho độ thị có hướng

Định lý 7.

Giả sử G có n đỉnh là đồ thị có hướng liên thông mạnh. Nếu $\deg_+(v) \geq n/2$, $\deg_-(v) \geq n/2$ $\forall v$ thì G có chu trình Hamilton

Định lý 8 (Konig).

Mọi đồ thị có hướng đủ đều có đường đi có hướng Hamilton.

Định lý 9 (Camion).

Đồ thị có hướng đủ có chu trình có hướng Hamilton khi và chỉ khi nó liên thông mạnh.

7.3.4 Mã Gray

Định nghĩa.

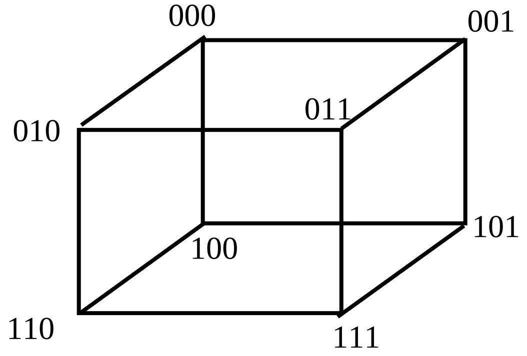
Dãy 2^n chuỗi n bit khác nhau $s_1, s_2, \ldots s_{2^n}$ gọi là mã Gray, nếu thỏa mãn

(i) s_i và s_{i+1} khác nhau đúng 1 bit, $i=1, 2, ..., 2^n-1$

(ii) s_1 và s_{2^n} khác nhau đúng 1 bit

Mã Gray liên quan mật thiết với một loại đồ thị gọi là siêu khối. Siêu khối cấp n là đồ thị có 2^n đỉnh được gán nhãn $0, 1, 2, ..., 2^n - 1$, trong đó hai đỉnh kề nhau nếu biểu diễn nhị phân n bit của chúng chỉ khác nhau đúng 1 bit.

Ví dụ 15. Siêu khối cấp 3 được biểu diễn như sau



Hình 7.31 Khối cấp 3

Siêu khối được dùng để mô hình hóa máy tính song song và thuật toán song song. Ta dễ dàng thấy rằng mỗi mã Gray $s_1, s_2, \ldots s_{2^n}$ tương ứng với một chu trình Hamilton trong siêu khối cấp *n* và ngược lại.

Ngoài ra, mã Gray còn được nghiên cứu ứng dụng trong việc chuyển tính hiệu analog sang tín hiệu số.

Định lý 10.

Ký hiệu G_1 là dãy 0, 1. Ta xây dựng truy hồi dãy G_n theo G_{n-1} (n=2,3,...) như sau:

(i) Ký hiệu H_{n-1} là dãy G_{n-1} được viết theo thứ tự đảo ngược.

(ii) Ký hiệu G'_n là dãy các chuỗi *n* bit có được bằng cách thêm bit 0 vào trước các chuỗi của G_{n-1}.

(iii) Ký hiệu G''_n là dãy các chuỗi *n* bit có được bằng cách thêm bit 1 vào trước các chuỗi của H_{n-1}.

(iv) Ký hiệu G_n là dãy các phần tử của G'_n và G''_n nối tiếp nhau. Khi đó G_n chính là mã Gray với mọi *n*.

Hệ quả 2. Mọi siêu khối cấp $n \geq 2$ có chu trình Hamilton.

Ví dụ 16. Xây dựng truy hồi dãy G_4

Ta có H_1= 1, 0

G'_2=00, 01

G''_2=11, 10

Suy ra G_2= 00, 01, 11, 10

Ta có H_2= 10, 11, 01, 00

G'_3=000, 001, 011, 010

G''_3=110, 111, 101, 100

Suy ra G_3= 000, 001, 011, 010, 110, 111, 101, 100

Ta có H_3= 100, 101, 111, 110, 010, 011, 001, 000

G'_4=0000, 0001, 0011, 0010, 0110, 0111, 0101, 0100

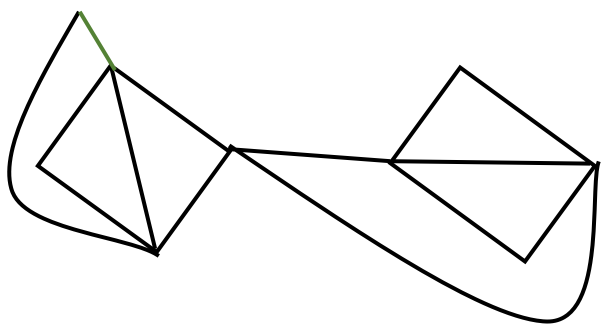
G''_4=1000, 1001, 1011, 1010, 1110, 1111, 1101, 1100

Suy ra G_4= 0000, 0001, 0011, 0010, 0110, 0111, 0101, 0100, 1000, 1001, 1011, 1010, 1110, 1111, 1101, 1100

Bài tập chương 7

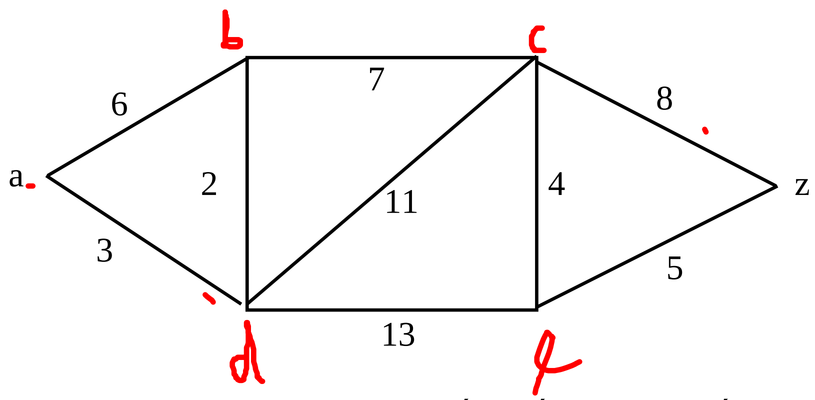
7.1 Xác định điều kiện của tham số n để đồ thị K_n có chu trình Euler, đường đi

7.2 Với mỗi đồ thị sau dùng thuật toán Fleury để đưa ra chu trình Euler của đồ thị hoặc không có chu trình Euler.

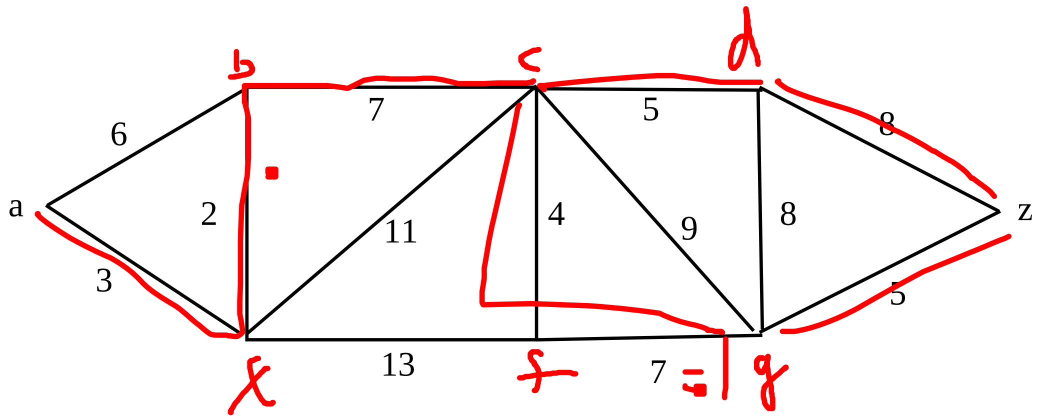


7.3 Viết chương trình tìm đường đi Euler theo thuật toán Fleury

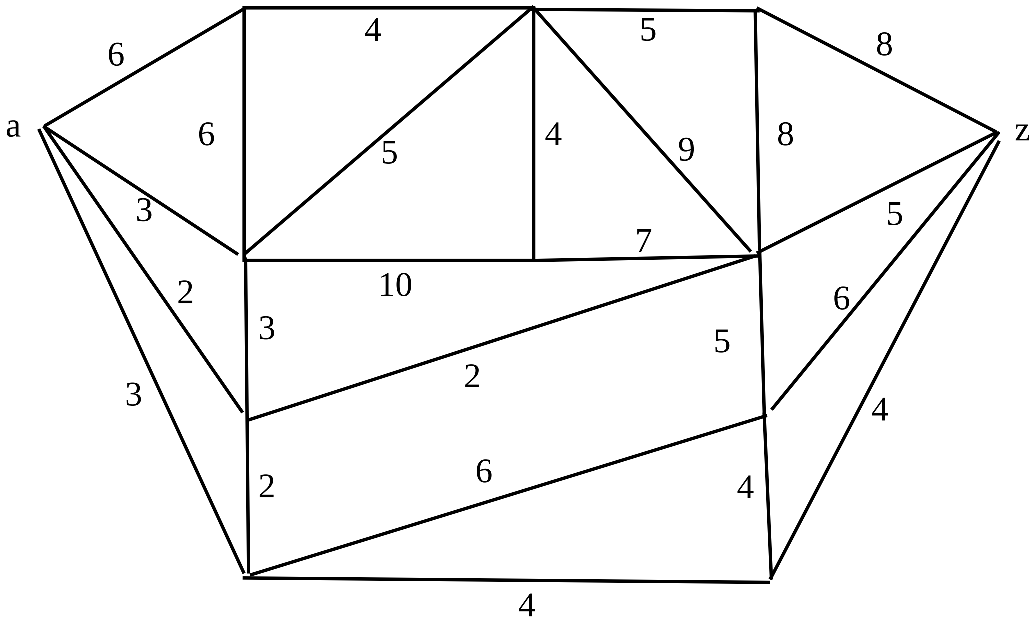
7.4 Dùng giải thuật Dijkstra tìm đường đi ngắn nhất từ đỉnh a đến đỉnh z trong đồ thị có trọng số sau



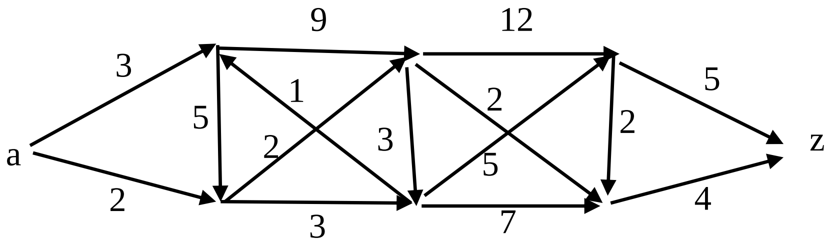
7.5 Dùng giải thuật Dijkstra tìm đường đi ngắn nhất từ đỉnh a đến đỉnh z trong đồ thị có trọng số sau



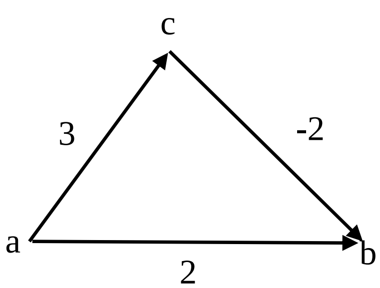
7.6 Dùng giải thuật Dijkstra tìm đường đi ngắn nhất từ đỉnh a đến đỉnh z trong đồ thị có trọng số sau



7.7 Dùng giải thuật Dijkstra tìm đường đi ngắn nhất từ đỉnh a đến đỉnh z trong đồ thị có trọng số sau



7.8 Dùng giải thuật Dijkstra tìm đường đi ngắn nhất từ đỉnh a đến đỉnh b trong đồ thị có trọng số sau



7.9 Thuật toán Dijkstra có thể không cho kết quả đúng nếu có trọng số âm, Thuật toán Bellman- Ford khắc phục hạn chế trên và cho phép tìm đường đi ngắn nhất từ một đỉnh đến các đỉnh khác mà không cần điều kiện trọng số dương.

Thuật toán Bellman- Ford

◊ *Đầu vào*. Đồ thị có trọng số (có hướng) G=(V,E,w), $s \in$V.

◊ *Đầu ra*. Danh sách L(v), độ dài đường đi ngắn nhất từ s đến v, và danh sách P(v), đỉnh kề trước v trên đường đi ngắn nhất từ s đến v, v $\in$ V, hoặc kết luận đồ thị có chu trình âm qua đỉnh khả nối với s.

◊ *Phương pháp*

(1) Khởi tạo: L(s) = 0; L(v) = $+\infty$, $\forall v \neq$ s; P(v) = $\varnothing$, $\forall v \in$ V.

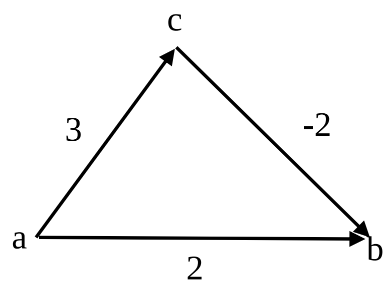
(2) For i $\in$[1 … n–1] do (n = |V| số phần tử của V)

```
    For (u,v) ∈ E do
        if (L(v) > L(u) + w(u,v))
            {
                L(v) = L(u) + w(u,v);
                P(v) = u;
            }
```

(3) Nếu tồn tại (u,v) $\in$ E thỏa L(v) > L(u) + w(u,v), thì kết luận đồ thị có chu trình âm khả nối với s. Ngược lại, L(v) là độ dài đường đi (có hướng) ngắn nhất từ s đến v, và danh sách P(v), đỉnh kề trước v trên đường đi ngắn nhất từ s đến v, v $\in$ V

Ghi chú: Cách xác đường đi dựa vào P giống thuật toán Dijkstra

Dùng giải thuật Dijkstra tìm đường đi ngắn nhất từ đỉnh a đến đỉnh b trong đồ thị có trọng số sau



7.10 Cài đặt thuật toán Dijkstra

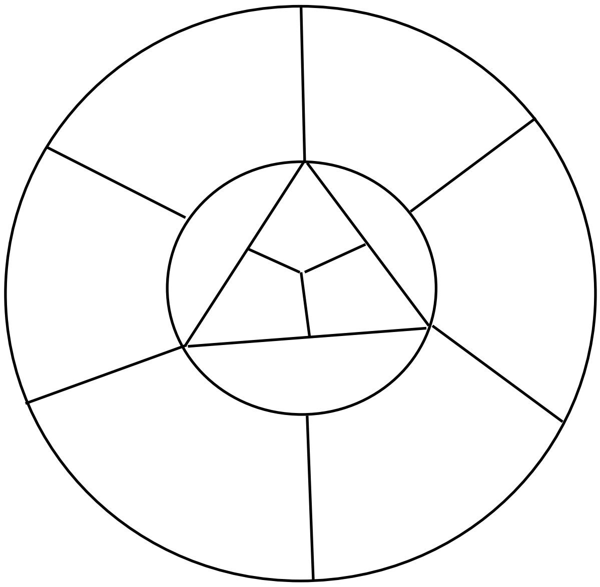
7.11 Cài đặt thuật toán Bellman- Ford

7.12 Dùng giải thuật Floyd-Warshall tìm đường đi ngắn nhất giữa các đỉnh trong các đồ thị có trọng số sau:

$$\begin{bmatrix} 9 & 7 & vc & 2 & vc & 6 \\ 3 & vc & 4 & vc & 1 & 2 \\ vc & 1 & 5 & vc & vc & 3 \\ vc & 4 & vc & 6 & 4 & vc \\ 2 & vc & 2 & 4 & vc & 7 \\ vc & 1 & vc & 7 & 9 & vc \end{bmatrix}$$

7.13 Dùng giải thuật Floyd-Warshall tìm đường đi ngắn nhất giữa các đỉnh trong các đồ thị có trọng số của các đồ thị ở bài tập 4, 5, 6, 7

7.14 Cho đồ thị G với 16 đỉnh dưới đây, chứng minh rằng G không có đường đi Himilton



7.15 (Bài toán xếp chỗ ngồi) 9 người bạn cùng ngồi ăn trong bàn tròn. Mỗi lần họ được xếp ngồi theo một thứ tự. Hãy thay đổi chỗ ngồi mỗi lần sao cho không có 2 người ngồi gần nhau hơn 1 lần và có bao nhiêu lần như vậy?

7.16 Có n người, trong đó mỗi người quen ít nhất $\lceil n/2 \rceil$ ($\lceil x \rceil$ là số nguyên nhỏ nhất $\geq$ x) người. Hỏi có thể xếp n người này vào bàn tròn sao cho mỗi người ngồi cạnh hai người quen?

7.17 Có n nam và n nữ, trong đó mỗi người quen ít *nhất* $\lceil n/2 +1 \rceil$ *người* khác giới. Hỏi có thể xếp xen kẽ nam nữ 2n người này vào bàn tròn sao cho mỗi người ngồi cạnh hai người quen?

7.18 Xây dựng truy hồi tìm siêu khối G_5 với G_4 = 0000, 0001, 0011, 0010, 0110, 0111, 0101, 0100, 1000, 1001, 1011, 1010, 1110, 1111, 1101, 1100

7.19 Viết chương trình cài đặt thuật toán xây dựng mã Gray

CHƯƠNG 8. CÂY

Tóm tắt chương

Trong chương này, sẽ trình bày các khái niệm cơ bản định nghĩa và định lý về cây, cây phủ và cây phủ nhỏ nhất. Hơn nữa bài toán tìm cây phủ nhỏ nhất có rất nhiều ứng dụng vì vậy trong chương này cũng trình bày các thuật toán tìm cây phủ nhỏ nhất. Phần cuối chương sẽ đề cập đến cây nhị phân tìm kiếm và các phương pháp duyệt cây.

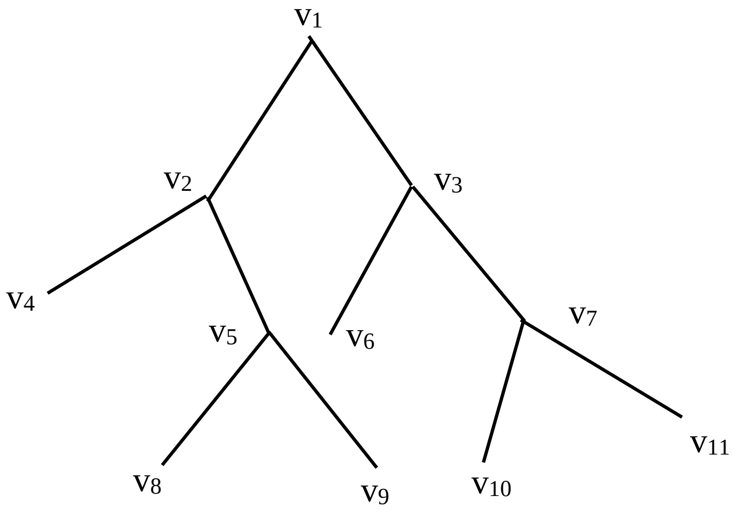
8.1 Các khái niệm cơ bản

8.1.1 Định nghĩa

Định nghĩa 1.

Cây là đồ thị liên thông không chứa chu trình

Ví dụ 1. Đồ thị sau đây là cây



Hình 8.1 Cây có 11 đỉnh

Vì cây trên là đô thị liên thông và không chứa chu trình

- Gốc của cây là đỉnh trên cùng.
- Mức của đỉnh là độ dài đường đi từ gốc đến đỉnh đó (số cạnh của đường đi).
- Độ cao của cây là mức lớn nhất của cây (tức mức của đỉnh cách xa gốc nhất).

Trong ví dụ trên nếu ta chọn v_1 là gốc thì v_2, v_3 là những đỉnh mức 1, các đỉnh v_4, v_5, v_6, v_7 có mức 2, các đỉnh v_8, v_9, v_{10}, v_{11} có mức 3 và độ cao của cây là 3.

Định nghĩa 2.

Cho T là cây có gốc v_0. Giả sử x, y, z là các đỉnh của T và (v_0, v_1, ..., v_n) là đường đi từ v_0 đến v_n trong T.

Khi đó ta gọi v_{n-1} là cha của v_n

$v_0,\dots,v_{n-1}$ là tiền bối của v_n

v_n là con của v_{n-1}

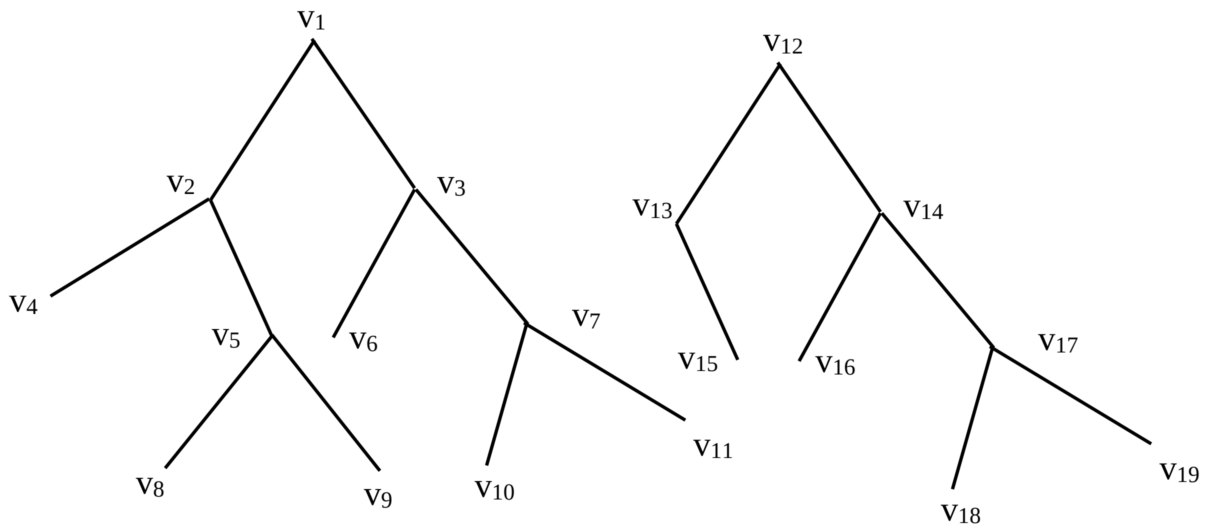
y là hậu thế của x, nếu x là tiền bối của y

y và z là anh em nếu chúng đều là con của đỉnh x

x là đỉnh lá nếu nó không có con

x là đỉnh trong của cây nếu nó có con.

Định nghĩa 3. Rừng là đồ thị mà mỗi thành phần liên thông là cây



Hình 8.2 Cây có 2 thành phần liên thông (rừng)

Đồ thị trên có 2 thành phần liên thông và mỗi thành phần liên thông là cây nên đồ thị trên gọi là rừng

8.1.2 Định lý tương đương (Định lý 1)

Cho T là đồ thị *n* đỉnh. Các mệnh đề sau tương đương

(i) T là cây

(ii) T không chứa chu trình và có $n-1$ cạnh.

(iii) T liên thông và có $n-1$ cạnh

(iv) T liên thông và mỗi cạnh là cầu

(v) Hai đỉnh bất kỳ của T được nối với nhau bằng một đường đi duy nhất

(vi) T không chứa chu trình và nếu thêm một cạnh nối hai đỉnh thì ta thu được đúng một chu trình

(vii) T liên thông và nếu thêm một cạnh nối hai đỉnh thì ta thu được đúng một chu trình

Chứng minh:

- Từ (i) suy ra (ii), ta chứng minh bằng qui nạp:

Với n=1 hiển nhiên đúng.

Ta giả sử đúng với $n \leq k$ ta chứng minh đúng với $n=k+1$

Thật vậy, e là cạnh thuộc T loại e khỏi T có có độ thị T gồm 2 thành phần liên thông T_1 và T_2. Gọi n_1, m_1 là số đỉnh và số cạnh của T_1. n2, m_2 là số đỉnh và số cạnh của T_2. Theo giả thiết qui nạp ta có $m_1=n_1-1$, $m_2=n_2-1$. Mà số cạnh của T khi e chưa bị loại nên số cạnh T tính theo m_1, m_2 là $m_1 + m_2 +1$. Thay $m_1=n_1-1$, $m_2=n_2-1$ ta có

$$m_1 + m_2 +1= n_1-1+ n_2-1+1= n_1+ n_2-1=n-1 \text{ (đpcm)}$$

- Từ (ii) suy ra (iii), ta chứng minh bằng phản chứng. Giả sử T không liên thông và có k thành phần liên thông ($k \geq 2$) $T_1, T_2, ..., T_k$. Vì T không chứa chu trình nên các thành phần liên thông cũng không chứa chu trình. Theo định nghĩa $T_1, T_2, ..., T_k$ cũng là cây. Gọi n_i , m_i là số đỉnh và số cạnh tương ứng với T_i, $i=1,...,k$. Theo (ii) ta có $m_i = n_i - 1$, $i=1,...,k$. Suy ra số cạnh m của T là $m = \sum m_i = \sum (n_i - 1) = n - k$ mâu thuẫn với (ii) $(m=n-1)$.
- Từ (iii) suy ra (iv), Việc loại bỏ cạnh bất kỳ của T dẫn đến một đồ thị n đỉnh và $n-2$ cạnh. Đồ thị này không thể liên thông được. Như vậy khi loại bỏ 1 cạnh bất kỳ số thành phần liên thông sẽ tăng lên 1. Vậy mỗi cạnh của T là cầu.
- Từ (iv) suy ra (v), Do T liên thông nên hai đỉnh bất kỳ của nó được nối với nhau bằng một dây đường đi. Nếu có cặp đỉnh được nối với nhau bằng 2 đường đi thì 2 đường đi này tạo thành chu trình, suy ra đồ thị chứa chu trình và các cạnh trên chu trình đó không thể là cầu được, mâu thuẫn với mỗi cạnh của đồ thị là cầu.
- Từ (v) suy ra (vi), Hiển nhiên T không chứa chu trình. Bây giờ ta thêm cạnh $e=(u,v)$. Khi đó cạnh e cùng đường đi nối u và v tạo nên chu trình duy nhất của T.
- Từ (vi) suy ra (vii). Hiển nhiên
- Từ (vii) suy ra (i), T liên thông, thêm một cạnh nối hai đỉnh thì ta thu được đúng một chu trình, suy ra ban đầu T không thể có chu chu trình, vậy T là cây

8.1.3 Cây m-phân

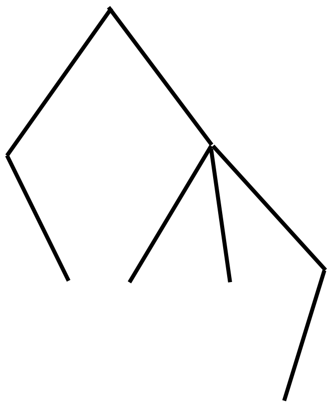
Định nghĩa 4.

Cây m-phân ($m \in N^*$) là cây mà mọi đỉnh trong có tối đa m con và có ít nhất một đỉnh có m con.

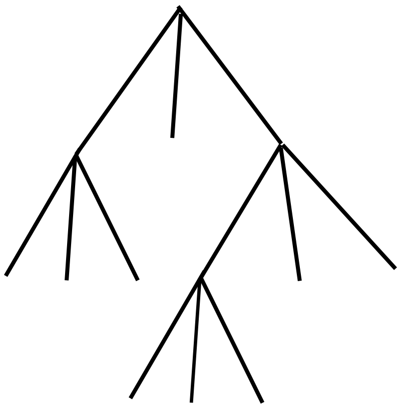
Cây m-phân đầy đủ là cây mà mọi đỉnh trong có đúng m con.

Cây cân bằng là cây mà mọi đỉnh lá có mức là h hay $h-1$, trong đó h là chiều cao của cây.

Ví dụ 2.

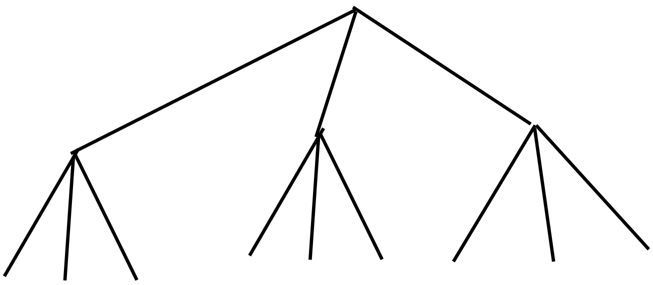


a) Cây tam không đầy đủ không cân bằng



b) Cây tam phân đầy đủ không cân bằng

Hình 8.3 Cây tam phân



Hình 8.4 Cây tam phân đầy đủ cân bằng

Định lý 2. Nếu cây m-phân đầy đủ có i đỉnh trong, thì nó có m.i+1 đỉnh.

Chứng minh

Mỗi đỉnh trong có m đỉnh con suy ra cây có m.i đỉnh con. Thêm đỉnh gốc không phải là đỉnh con của đỉnh khác ta có tất cả m.i+1 đỉnh.

Hệ quả. Cho T là cây m-phân đầy đủ có i đỉnh trong, l đỉnh lá và n đỉnh. Khi đó:

(i) $l = (m-1).i + 1$

(ii) $i = \frac{l-1}{m-1}$ & $n = \frac{m.l-1}{m-1}$

(iii) $i = \frac{n-1}{m}$

Chứng minh. Suy ra trực tiếp từ định lý

Ví dụ 3. Xét trò chơi viết thư dây chuyền. Ban đầu có một người nhận được một bức thư. Mỗi người khi nhận được thư hoặc sẽ viết thư cho 5 người chưa nhận thư hoặc sẽ không viết thư 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 đúng 17 người nhận thư mà không viết tiếp cho người khác.

Giải:

Trò chơi có thể biểu diễn bằng cây ngũ phân đầy đủ. Các đỉnh tương ứng với những người gửi thư cho người khác, còn lá là những người nhận thư mà không viết tiếp. Vì vậy số lá là l = 17

Như vậy số người nhận thư, tức số đỉnh,

$$n = \frac{5.17-1}{5-1} = 21$$

Số người viết thư, tức số đỉnh trong, là i=n-l=21-17=4

8.2 Cây phủ

8.2.1 Định nghĩa và tính chất

Định nghĩa 5.

Cho đồ thị G= (V, E). Cây T gọi là *cây phủ* hay *cây bao trùm* của G, nếu T là đồ thị con phủ của G

Tính chất tương đương giữa cây phủ và đồ thị liên thông thông qua định lý 3 sau đây

Định lý 3.

Đồ thị G= (V, E) có cây phủ khi và chỉ khi G liên thông.

Chứng minh.

($\Rightarrow$): Nếu G có cây phủ, thì hiển nhiên G liên thông.

($\Leftarrow$): Giả sử G là đồ thị liên thông. Nếu G không có chu trình, thì theo định nghĩa G là cây phủ của chính nó. Nếu G có chu trình, thì ta bỏ 1 cạnh (không bỏ đỉnh) trên chu trình đó. Đồ thị con thu được vẫn liên thông. Nếu vẫn còn chu trình, ta lại loại bớt 1 cạnh. Cứ tiếp tục quá trình trên cho đến khi không còn chu trình thì ta thu được cây phủ của đồ thị G.

8.2.2 Các thuật toán tìm cây phủ

a) Thuật toán tìm theo chiều ngang

Trong thuật giải này ta ký hiệu Q là hàng đợi (queue) các đỉnh, Ke(x) là danh sách các đỉnh kề đỉnh *x*.

- *Đầu vào*. Đồ thị G= (V, E).

- *Đầu ra*. Cây phủ T hoặc kết luận đồ thị không liên thông.

- *Các bước*.

Bước 1. Khởi tạo:

Chọn đỉnh v_1 bất kỳ. Khởi tạo hàng đợi Q := [v_1], T là đồ thị gồm 1 đỉnh v_1 và không có cạnh, v_1 là gốc.

Bước 2. Thêm cạnh:

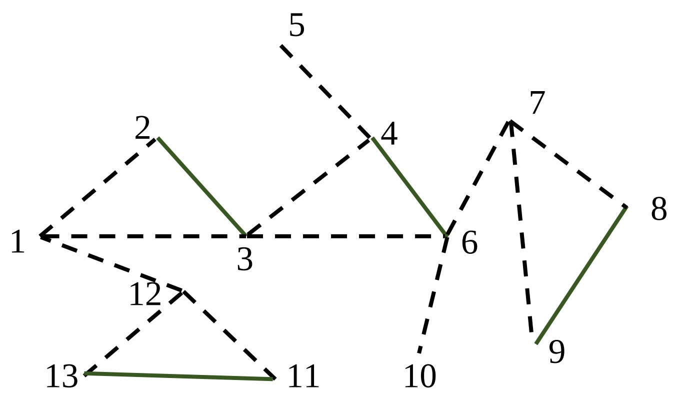
Rút đỉnh $x \in$Q. Duyệt các đỉnh y$\in$Ke(x), nếu $y \notin$ T, thì thêm cạnh (x,y) và đỉnh y vào T và đẩy y vào hàng đợi Q.

Nếu Q $\neq \varnothing$ và T chưa phủ hết các đỉnh, thì quay lại bước (2), ngược lại sang bước (3).

Bước 3. Kết luận:

Nếu T phủ hết các đỉnh của đồ thị, thì T là cây phủ, ngược lại, Q=$\varnothing$, đồ thị không liên thông. Kết thúc.

Ví dụ 4. Tìm cây phủ sau theo thuật toán tìm theo chiều ngang



Hình 8.5 Cây có 11 đỉnh

Giải:

Ta xếp thứ tự các đỉnh là 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13.

Bước 1: Khởi tạo: Q := [1]. T := ({1}, $\varnothing$). Chọn đỉnh 1 làm gốc.

Bước 2: Thêm cạnh:

Rút đỉnh 1 khỏi hàng đợi Q. Thêm các cạnh (1,2), (1,3), (1,12) và các đỉnh 2, 3, 12 vào cây T.

T = **(**{1,2,3,12}, {(1,2), (1,3), (1,2)}**)**

Đẩy các đỉnh 2,3,12 vào hàng đợi Q

Q = [2,3,12]

Bước 2: Thêm cạnh:

Rút đỉnh 2 khỏi hàng đợi Q. đỉnh 3 nằm trong T nên không them vào T

T = **(**{1,2,3,12}, {(1,2), (1,3), (1,2)}**)**

Q = [3,12]

Bước 2: Thêm cạnh:

Rút đỉnh 3 khỏi hàng đợi Q. Thêm các cạnh (3,4), (3,6) và các đỉnh 4, 6 vào cây T.

T = **(**{1,2,3,12,4,6}, {(1,2), (1,3), (1,2), (3,4), (3,6) }**)**

Đẩy các đỉnh 4,6 vào hàng đợi Q

Q = [12,4,6]

Bước 2: Thêm cạnh:

Rút đỉnh 12 khỏi hàng đợi Q. Thêm các cạnh (12,11), (12,13) và các đỉnh 11, 13 vào cây T.

T = **(**{1,2,3,12,4,6, 11, 13}, {(1,2), (1,3), (1,2), (3,4), (3,6), (12,11), (12,13) }**)**

Đẩy các đỉnh 11,13 vào hàng đợi Q

Q = [4,6, 11,13]

Bước 2: Thêm cạnh:

Rút đỉnh 4 khỏi hàng đợi Q. Thêm các cạnh (4,5) và các đỉnh 5 vào cây T.

T = **(**{1,2,3,12,4,6,11,13,5}, {(1,2), (1,3), (1,2), (3,4), (3,6), (12,11), (12,13), (4,5), }**)**

Đẩy các đỉnh 5 vào hàng đợi Q

Q = [6, 11,13,5]

Bước 2: Thêm cạnh:

Rút đỉnh 6 khỏi hàng đợi Q. Thêm các cạnh (6,7), (6,10) và các đỉnh 7, 10 vào cây T.

T = **(**{1,2,3,12,4,6,11,13,5, 7,10}, {(1,2), (1,3), (1,2), (3,4), (3,6), (12,11), (12,13), (4,5), (6,7), (6,10)}**)**

Đẩy các đỉnh 5 vào hàng đợi Q

Q = [11,13,5, 7, 10]

Bước 2: Thêm cạnh:

Rút đỉnh 11 khỏi hàng đợi Q. Cây T không thay đổi.

T = **(**{1,2,3,12,4,6,11,13,5, 7,10}, {(1,2), (1,3), (1,2), (3,4), (3,6), (12,11), (12,13), (4,5), (6,7), (6,10)}**)**

Q = [13,5, 7, 10]

Bước 2: Thêm cạnh:

Rút đỉnh 13 khỏi hàng đợi Q. Cây T không thay đổi.

T = **(**{1,2,3,12,4,6,11,13,5, 7,10}, {(1,2), (1,3), (1,2), (3,4), (3,6), (12,11), (12,13), (4,5), (6,7), (6,10)}**)**

Q = [5, 7, 10]

Bước 2: Thêm cạnh:

Rút đỉnh 5 khỏi hàng đợi Q. Cây T không thay đổi.

T = **(**{1,2,3,12,4,6,11,13,5, 7,10}, {(1,2), (1,3), (1,2), (3,4), (3,6), (12,11), (12,13), (4,5), (6,7), (6,10)}**)**

Q = [7, 10]

Bước 2: Thêm cạnh:

Rút đỉnh 7 khỏi hàng đợi Q. Thêm các cạnh (7,8), (7,9) và các đỉnh 8, 9 vào cây T.

T = **(**{1,2,3,12,4,6,11,13,5,7,10, 8,9}, {(1,2), (1,3), (1,2), (3,4), (3,6), (12,11), (12,13), (4,5), (6,7), (6,10), (7,8), (7,9) }**)**

Q = [10]

Vì T phủ hết các đỉnh nên sang bước 3

Bước 3: Kết luận T là cây phủ

(T các tất cả các đỉnh còn các cạnh chính là các cạnh nét đứt)

b) Thuật toán tìm theo chiều sâu

Trong thuật giải này ta ký hiệu S là ngăn xếp (stack) các đỉnh, Ke(x) là danh sách các đỉnh kề đỉnh x.

- Đầu vào. Đồ thị G=(V,E).
- Đầu ra. Cây phủ T hoặc kết luận đồ thị không liên thông.
- Các bước.

Bước 1. Khởi tạo:

Chọn đỉnh v_1 bất kỳ. Khởi tạo ngăn xếp S: = [v_1], T là đồ thị cây gồm 1 đỉnh v_1 và không có cạnh, v_1 là gốc của T.

Bước 2. Thêm cạnh:

Xét đỉnh w∈S. Tìm đỉnh v∈Ke(w) thỏa v ∉ T. Nếu tồn tại đỉnh v, thì Thêm cạnh (w,v) và đỉnh v vào T và đẩy v vào ngăn xếp S.

Nếu T phủ hết các đỉnh, T là cây phủ, kết thúc;

Nếu S ≠ ∅ và T chưa phủ hết các đỉnh, thì quay lại bước (2).

Nếu không tồn tại đỉnh v ∈ Ke(w) thỏa v∉T, thì đến bước (3).

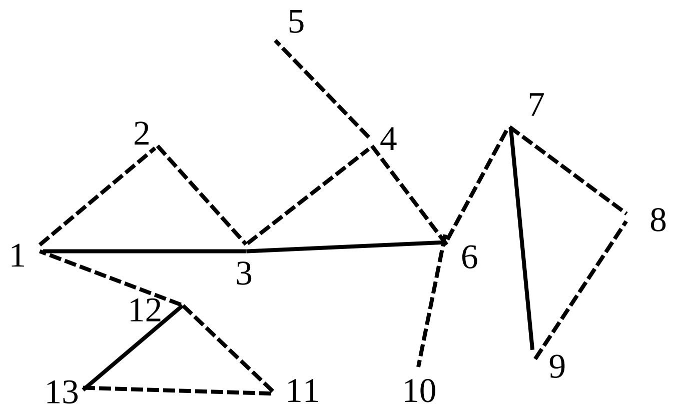
Bước 3. Kiểm tra điều kiện kết thúc:

Nếu w = v_1, kết luận đồ thị không liên thông. Kết thúc. Nếu w ≠ v_1 sang bước (4)

Bước 4. Quay lui:

Đẩy đỉnh w ra khỏi ngăn xếp S. Quay lại bước (2).

Ví dụ 5. Tìm cây phủ của đồ thị sau theo thuật toán chiều sâu



Hình 8.6 Cây có 11 đỉnh tìm theo chiều sâu

Giải:

Ta xếp thứ tự các đỉnh là 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13.

Bước 1: *Khởi tạo*: v_1:=1; S := [1]. T = **(**{1}, ∅**)**. Chọn đỉnh 1 làm gốc.

Bước 2: *Thêm cạnh*:

Xét đỉnh w = 1 trên ngăn xếp S. Thêm cạnh (1,2) và đỉnh 2 vào cây T.

T = **(**{1,2}, {(1,2)}**)**

Đẩy đỉnh 2 vào ngăn xếp S

S = [2,1]

Bước 2: *Thêm cạnh*:

Xét đỉnh w = 2 trên ngăn xếp S. Thêm cạnh (2,3) và đỉnh 3 vào cây T.

T = **(**{1,2,3}, {(1,2), (2,3) }**)**

Đẩy đỉnh 3 vào ngăn xếp S

S = [3, 2,1]

Bước 2: *Thêm cạnh*:

Xét đỉnh w = 3 trên ngăn xếp S. Thêm cạnh (3,4) và đỉnh 4 vào cây T.

T = **(**{1,2,3,4}, {(1,2), (2,3), (3,4) }**)**

Đẩy đỉnh 4 vào ngăn xếp S

S = [4, 3, 2,1]

Bước 2: *Thêm cạnh*:

Xét đỉnh w = 4 trên ngăn xếp S. Thêm cạnh (4,5) và đỉnh 5 vào cây T.

T = **(**{1,2,3,4,5}, {(1,2), (2,3), (3,4), (4,5) }**)**

Đẩy đỉnh 5 vào ngăn xếp S

S = [5, 4, 3, 2,1]

Bước 2: *Thêm cạnh*:

Xét đỉnh w = 5 trên ngăn xếp S. Không có đỉnh nào kề 5 mà$\notin$T nên sang bước 3

Bước 3: Vì 5 $\neq$ 1 nên sang bước 4

Bước 4: Quay lui:

Đẩy đỉnh 5 ra khỏi ngăn xếp S, S = [4, 3, 2,1]. Quay lại bước (2).

Bước 2: *Thêm cạnh*:

Xét đỉnh w = 4 trên ngăn xếp S. Thêm cạnh (4,6) và đỉnh 6 vào cây T.

T = **(**{1,2,3,4,5,6}, {(1,2), (2,3), (3,4), (4,5), (4,6)}**)**

Đẩy đỉnh 5 vào ngăn xếp S

S = [6,5, 4, 3, 2,1]

Bước 2: *Thêm cạnh*:

Xét đỉnh w = 6 trên ngăn xếp S. Thêm cạnh (6,7) và đỉnh 7 vào cây T.

T = **(**{1,2,3,4,5,6,7}, {(1,2), (2,3), (3,4), (4,5), (4,6), (6,7) }**)**

Đẩy đỉnh 7 vào ngăn xếp S

S = [7,6,5, 4, 3, 2,1]

Bước 2: *Thêm cạnh*:

Xét đỉnh w = 7 trên ngăn xếp S. Thêm cạnh (7,8) và đỉnh 8 vào cây T.

T = **(**{1,2,3,4,5,6,7,8}, {(1,2), (2,3), (3,4), (4,5), (4,6), (6,7), (7,8)}**)**

Đẩy đỉnh 8 vào ngăn xếp S

S = [8,7,6,5, 4, 3, 2,1]

Bước 2: *Thêm cạnh*:

Xét đỉnh w = 8 trên ngăn xếp S. Thêm cạnh (8,9) và đỉnh 9 vào cây T.

T = **(**{1,2,3,4,5,6,7,8,9}, {(1,2), (2,3), (3,4), (4,5), (4,6), (6,7), (7,8),(8,9)}**)**

Đẩy đỉnh 9 vào ngăn xếp S

S = [9,8,7,6,5, 4, 3, 2,1]

Bước 2: *Thêm cạnh*:

Xét đỉnh w = 9 trên ngăn xếp S. Không có đỉnh nào kề với đỉnh 9 mà không thuộc T nên sang bước 3.

Bước 3: Vì 9 ≠ 1 nên sang bước 4

Bước 4: Quay lui:

Đẩy đỉnh 9 ra khỏi ngăn xếp S, S = [8,7,6,5, 4, 3, 2,1]. Quay lại bước (2).

Bước 2: *Thêm cạnh*:

Xét đỉnh w = 8 trên ngăn xếp S. Không có đỉnh nào kề với đỉnh 8 mà không thuộc T nên sang bước 3.

Bước 3: Vì 8 ≠ 1 nên sang bước 4

Bước 4: Quay lui:

Đẩy đỉnh 8 ra khỏi ngăn xếp S, S = [7,6,5, 4, 3, 2,1]. Quay lại bước (2).

Bước 2: *Thêm cạnh*:

Xét đỉnh w = 7 trên ngăn xếp S. Không có đỉnh nào kề với đỉnh 7 mà không thuộc T nên sang bước 3.

Bước 3: Vì 7 ≠ 1 nên sang bước 4

Bước 4: Quay lui:

Đẩy đỉnh 7 ra khỏi ngăn xếp S, S = [6,5, 4, 3, 2,1]. Quay lại bước (2).

Bước 2: *Thêm cạnh*:

Xét đỉnh w = 6 trên ngăn xếp S. Thêm cạnh (6,10) và đỉnh 10 vào cây T.

T = **(**{1,2,3,4,5,6,7,8,9,10}, {(1,2), (2,3), (3,4), (4,5), (4,6), (6,7), (7,8),(8,9), (6,10) }**)**

S = [10,6,5, 4, 3, 2,1]. Quay lại bước (2).

Bước 2: *Thêm cạnh*:

Xét đỉnh w = 10 trên ngăn xếp S. Không có đỉnh nào kề với đỉnh 10 mà không thuộc T nên sang bước 3.

Bước 3: Vì 10 ≠ 1 nên sang bước 4

Bước 4: Quay lui:

Đẩy đỉnh 10 ra khỏi ngăn xếp S, S = [6,5, 4, 3, 2,1]. Quay lại bước 2

Bước 2: *Thêm cạnh*:

Xét đỉnh w = 6 trên ngăn xếp S. Không có đỉnh nào kề với đỉnh 6 mà không thuộc T nên sang bước 3.

Bước 3: Vì 6 ≠ 1 nên sang bước 4

Bước 4: Quay lui:

Đẩy đỉnh 6 ra khỏi ngăn xếp S, S = [5, 4, 3, 2,1]. Quay lại bước (2).

Bước 2: *Thêm cạnh*:

Xét đỉnh w = 5 trên ngăn xếp S. Không có đỉnh nào kề với đỉnh 5 mà không thuộc T nên sang bước 3.

Bước 3: Vì 5 ≠ 1 nên sang bước 4

Bước 4: Quay lui:

Đẩy đỉnh 5 ra khỏi ngăn xếp S, S = [4, 3, 2,1]. Quay lại bước (2).

Bước 2: *Thêm cạnh*:

Xét đỉnh w = 4 trên ngăn xếp S. Không có đỉnh nào kề với đỉnh 4 mà không thuộc T nên sang bước 3.

Bước 3: Vì 4 ≠ 1 nên sang bước 4

Bước 4: Quay lui:

Đẩy đỉnh 4 ra khỏi ngăn xếp S, S = [3, 2,1]. Quay lại bước (2).

Bước 2: *Thêm cạnh*:

Xét đỉnh w = 3 trên ngăn xếp S. Không có đỉnh nào kề với đỉnh 3 mà không thuộc T nên sang bước 3.

Bước 3: Vì 3 ≠ 1 nên sang bước 4

Bước 4: Quay lui:

Đẩy đỉnh 3 ra khỏi ngăn xếp S, S = [2,1]. Quay lại bước (2).

Bước 2: *Thêm cạnh*:

Xét đỉnh w = 2 trên ngăn xếp S. Không có đỉnh nào kề với đỉnh 2 mà không thuộc T nên sang bước 3.

Bước 3: Vì 2 ≠ 1 nên sang bước 4

Bước 4: Quay lui:

Đẩy đỉnh 2 ra khỏi ngăn xếp S, S = [1]. Quay lại bước (2).

Bước 2: *Thêm cạnh*:

Xét đỉnh w = 1 trên ngăn xếp S. Thêm cạnh (1,12) và đỉnh 12 vào cây T.

T = **(**{1,2,3,4,5,6,7,8,9,10,12}, {(1,2), (2,3), (3,4), (4,5), (4,6), (6,7), (7,8),(8,9), (6,10), (1,12) }**)**

Đẩy đỉnh 12 vào ngăn xếp S, S = [12,1]

Bước 2: *Thêm cạnh*:

Xét đỉnh w = 12 trên ngăn xếp S. Thêm cạnh (12,11) và đỉnh 11 vào cây T.

T = **(**{1,2,3,4,5,6,7,8,9,10, 12,11}, {(1,2), (2,3), (3,4), (4,5), (4,6), (6,7), (7,8),(8,9), (6,10), (1,12), (12,11) }**)**

Đẩy đỉnh 11 vào ngăn xếp S, S = [11,12,1]

Bước 2: *Thêm cạnh*:

Xét đỉnh w = 11 trên ngăn xếp S. Thêm cạnh (11,13) và đỉnh 13 vào cây T.

T = **(**{1,2,3,4,5,6,7,8,9, 10,12,11,13}, {(1,2), (2,3), (3,4), (4,5), (4,6), (6,7), (7,8),(8,9), (1,12), (12,11), (11,13) }**)**

Đẩy đỉnh 13 vào ngăn xếp S, S = [13,11,12,1]

Đến đây T đã phủ hết các đỉnh vậy kết luận T là cây phủ với T là cây được minh hoạ bởi các nét đứt

8.3 Cây phủ nhỏ nhất

8.3.1 Phát biểu bài toán

Cho G= (V, E, (c_{ij})) là trọng đồ liên thông có trọng số (c_{ij}). Tìm cây phủ T của G có tổng trọng số d(T) là nhỏ nhất.

$$d(T) = \sum_{(i,j)\in T} c_{ij}$$

8.3.2 Thuật toán Prim tìm cây phủ nhỏ nhất

- Đầu vào. Đồ thị G= (V, E) có n đỉnh. Trọng số của cạnh (i,j), (i,j)∈E, ký hiệu là c_{ij}.
- Đầu ra. Cây phủ nhỏ nhất T, hoặc kết luận đồ thị không liên thông.
- Các bước.

Bước 1. Khởi tạo

T là đồ thị gồm một đỉnh và không có cạnh.

Bước 2. Kiểm tra điều kiện kết thúc

Nếu T có $n-1$ cạnh, kết luận: T là cây phủ nhỏ nhất, kết thúc. Ngược lại sang bước 3.

Bước 3. Thêm cạnh

Ký hiệu M là tập $M = \{ (i,j)\in E \,/\, i\in T \;\&\; j \notin T \}$

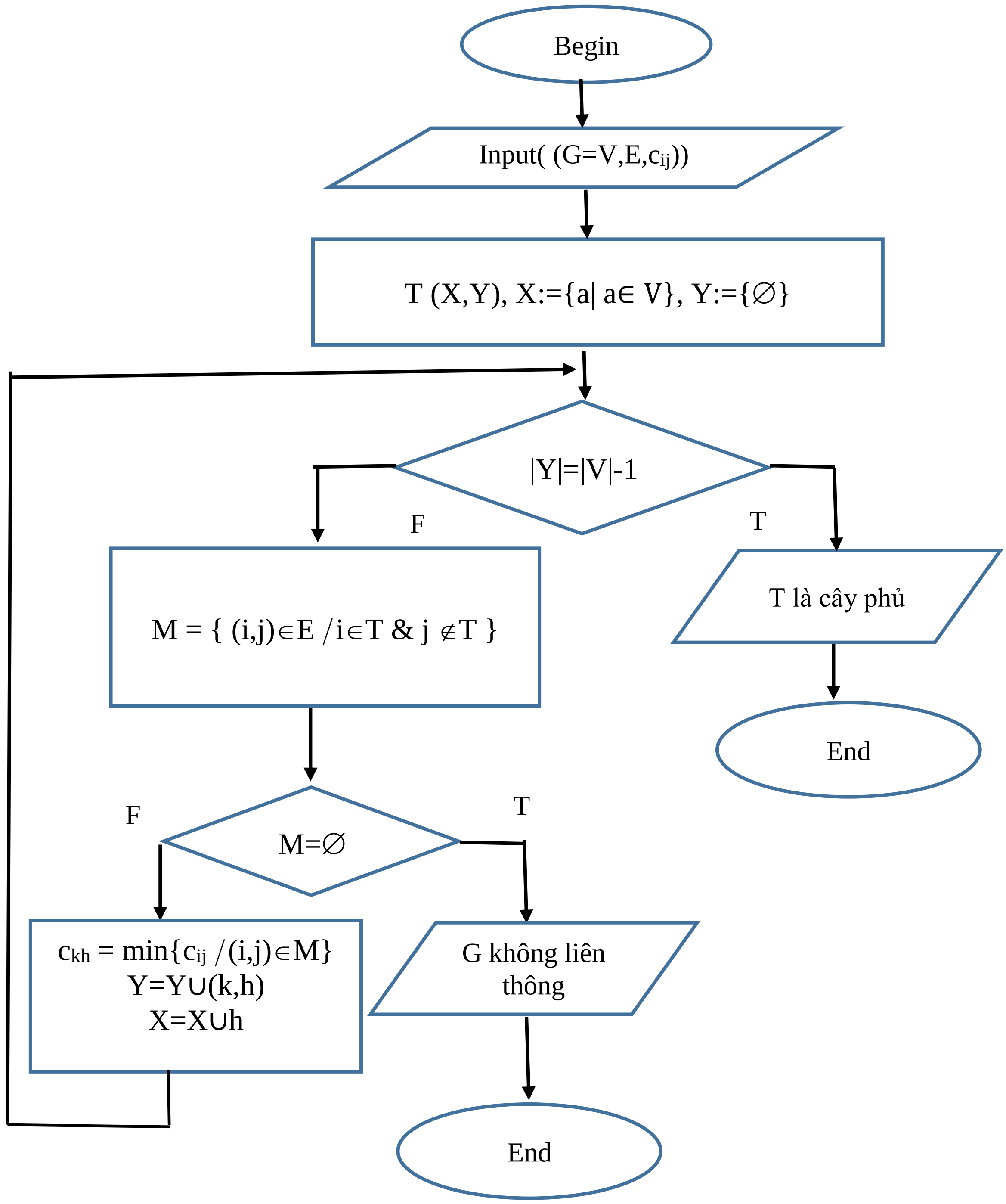
Nếu $M = \emptyset$, kết luận đồ thị G không liên thông, kết thúc.

Ngược lại, nếu $M \neq \emptyset$, tìm cạnh $(k,h)\in M$ sao cho

$c_{kh} = \min\{c_{ij} \,/\, (i,j)\in M\}$

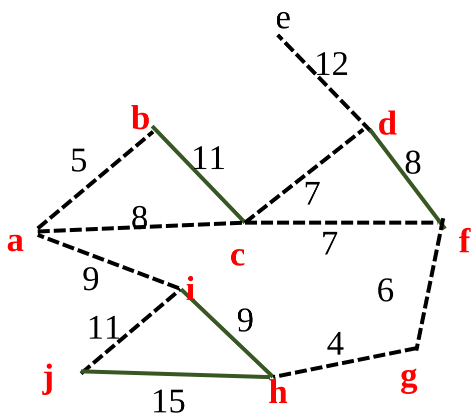
Thêm cạnh (k,h) và đỉnh h vào T, quay lại bước 2

Thuật toán bằng sơ đồ khối



Hình 8.7 Sơ đồ khối thuật toán Prim

Ví dụ 6. Cho đồ thị có trọng số sau đây, tìm cây phủ nhỏ nhất theo thuật toán Prim



Hình 8.8 Đồ thị biểu diễn thuật toán Prim

Giải:

Bước 1: Khởi tạo: T chỉ có đỉnh a, không có cạnh:

$T := (\{a\},\varnothing)$, $d(T) := 0$

Bước 2: Kiểm tra: Số cạnh của T là 0 < 9 (n-1), sang bước (3)

Bước 3: Thêm cạnh: Tập M là M = {(a,b), (a,c), (a,i)}. Ta có

$c_{ab} = 5 = \min\{c_{ij} \mid (i,j)\in M\}$

Thêm đỉnh b và cạnh (a,b) vào T,

$T: = (\{a,b\},\{(a,b)\})$, $d(T) := d(T) + c_{ab} = 5$

Bước 2: Kiểm tra: Số cạnh của T là 1 < 9 (n-1), sang bước (3)

Bước 3: Thêm cạnh: Tập M là M = {(a,c), (a,i), (b,c)}. Ta có

$c_{ac} = 8 = \min\{c_{ij} \mid (i,j)\in M\}$

Thêm đỉnh c và cạnh (a,c) vào T,

$T: = (\{a,b,c\},\{(a,b), (a,c)\})$, $d(T) := d(T) + c_{ac} = 5+8=13$

Bước 2: Kiểm tra: Số cạnh của T là 2 < 9 (n-1), sang bước (3)

Bước 3: Thêm cạnh: Tập M là M = {(a,i), (c,d), (c,f)}. Ta có

$C_{cd} = 7 = \min\{c_{ij} \mid (i,j)\in M\}$

Thêm đỉnh d và cạnh (c,d) vào T,

$T: = (\{a,b,c,d\},\{(a,b), (a,c), (c,d)\})$, $d(T) := d(T) + c_{cd} = 13+7=20$

Bước 2: Kiểm tra: Số cạnh của T là 3 < 9 (n-1), sang bước (3)

Bước 3: Thêm cạnh: Tập M là M = {(a,i), (c,f), (d,e), (d,f)}. Ta có

$C_{cf} = 7 = \min\{c_{ij} \mid (i,j)\in M\}$

Thêm đỉnh f và cạnh (c,f) vào T,

T: = ({a,b,c,d,f},{(a,b), (a,c), (c,d), (c,f)}), d(T) := d(T) + c_{cf} = 20+7=27

Bước 2: Kiểm tra: Số cạnh của T là 4 < 9 (n-1), sang bước (3)

Bước 3: Thêm cạnh: Tập M là M = {(a,i), (d,e), (f,g)}. Ta có

$C_{fg} = 6 = \min\{c_{ij} \mid (i,j)\in M\}$

Thêm đỉnh g và cạnh (f,g) vào T,

T: = ({a,b,c,d,f,g},{(a,b), (a,c), (c,d), (c,f), (f,g)}), d(T) := d(T) + c_{fg} = 27+6=33

Bước 2: Kiểm tra: Số cạnh của T là 5 < 9 (n-1), sang bước (3)

Bước 3: Thêm cạnh: Tập M là M = {(a,i), (d,e), (g,h)}. Ta có

$C_{gh} = 4 = \min\{c_{ij} \mid (i,j)\in M\}$

Thêm đỉnh h và cạnh (g,h) vào T,

T: = ({a,b,c,d,f,g,h},{(a,b), (a,c), (c,d), (c,f), (f,g), (g,h)}), d(T) := d(T) + c_{gh} = 33+4=37

Bước 2: Kiểm tra: Số cạnh của T là 6 < 9 (n-1), sang bước (3)

Bước 3: Thêm cạnh: Tập M là M = {(a,i), (d,e), (h,i), (h,j)}. Ta có

$C_{ai} = 9 = \min\{c_{ij} \mid (i,j)\in M\}$

Thêm đỉnh i và cạnh (a,i) vào T,

T: = ({a,b,c,d,f,g,h,i},{(a,b), (a,c), (c,d), (c,f), (f,g), (g,h), (a,i)}), d(T) := d(T) + c_{ai} = 37+9=46

Bước 2: Kiểm tra: Số cạnh của T là 7< 9 (n-1), sang bước (3)

Bước 3: Thêm cạnh: Tập M là M = {(d,e), (h,j), (i,j)}. Ta có

$C_{ij} = 11 = \min\{c_{ij} \mid (i,j)\in M\}$

Thêm đỉnh j và cạnh (i,j) vào T,

T: = ({a,b,c,d,f,g,h,i,j},{(a,b), (a,c), (c,d), (c,f), (f,g), (g,h), (a,i), (i,j)}), d(T) := d(T) + c_{ij} = 46+11=57

Bước 2: Kiểm tra: Số cạnh của T là 8< 9 (n-1), sang bước (3)

Bước 3: Thêm cạnh: Tập M là M = {(d,e)}. Ta có

$C_{de} = 12 = \min\{c_{ij} \mid (i,j)\in M\}$

Thêm đỉnh e và cạnh (d,e) vào T,

T: = ({a,b,c,d,f,g,h,i,j,e},{(a,b), (a,c), (c,d), (c,f), (f,g), (g,h), (a,i), (i,j), (d,e)}), d(T) := d(T) + c_{de} = 57+12=69

Bước 2: Kiểm tra: Số cạnh của T là 9= n-1, nên kết thúc, Trọng số của cây T là 69

Cây qua các cạnh nét đứt trên đồ thị

8.3.3 Thuật toán Kruskal tìm cây phủ nhỏ nhất

- Đầu vào: Đồ thị G= (V, E) có n đỉnh. Trọng số của cạnh (i,j), (i,j)$\in$E, ký hiệu là c_{ij}
- Đầu ra: Cây phủ nhỏ nhất T, hoặc kết luận đồ thị không liên thông.
- Các bước.

Bước 1: Khởi tạo

T := (V, $\emptyset$) là đồ thị gồm các đỉnh của G và không có cạnh.

Bước 2: Kiểm tra điều kiện kết thúc

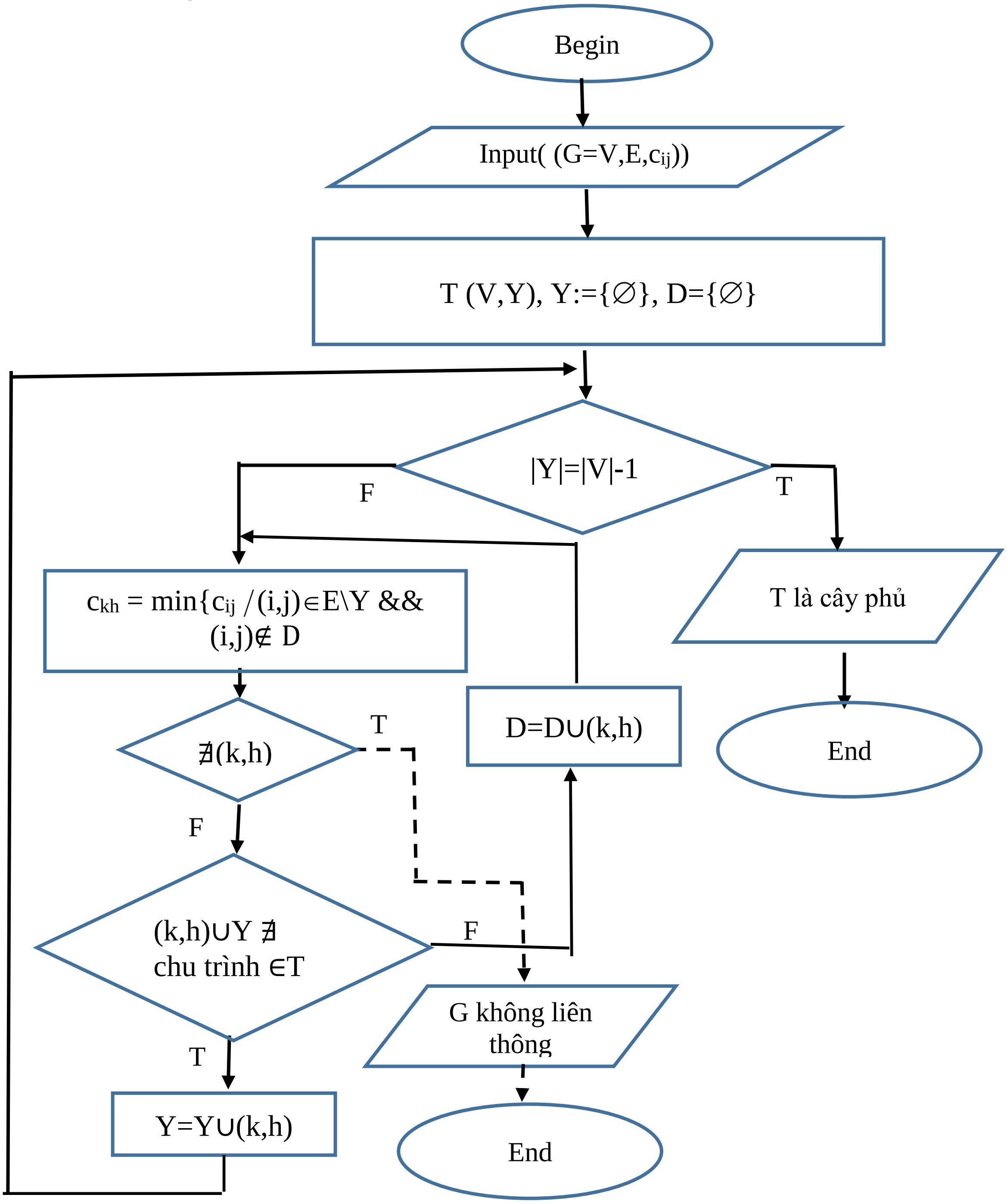
Nếu T có n$-$1 cạnh, kết luận: T là cây phủ nhỏ nhất, kết thúc. Ngược lại sang bước 3.

Bước 3: Thêm cạnh

Chọn cạnh có trọng số nhỏ nhất không thuộc T sao cho khi thêm cạnh này vào T thì không tạo ra chu trình trong T. Quay lại bước 2.

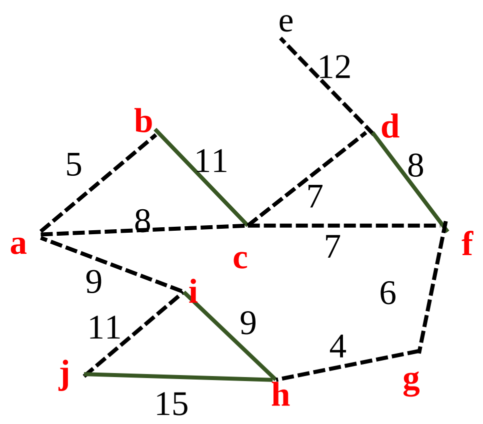
Ngược lại, nếu không chọn được cạnh thêm vào T, thì kết luận đồ thị G không liên thông, kết thúc.

Thuật toán bằng sơ đồ khối



Hình 8.9 Sơ đồ khối thuật toán Kruskal

Ví dụ 7. Cho đồ thị có trọng số sau đây, tìm cây phủ nhỏ nhất theo thuật toán Kruskal



Hình 8.10 Đồ thị biểu diễn thuật toán Kruskal

Giải:

Bước 1: Khởi tạo: T := ({a,b,c,d,e,f,g,h,i,j}, ∅), không có cạnh, số cạnh là 0, trọng số d(T):=0.

Bước 2. Kiểm tra:

Số cạnh của T là 0 < 9 (n-1), sang bước 3.

Bước 3. Thêm cạnh:

Trong các cạnh không thuộc T cạnh (g,h) có độ dài nhỏ nhất là 4.

Thêm cạnh (g,h) vào T. Ta có

T = ({a,b,c,d,e,f,g,h,i,j}, {(g,h)}) , d(T):=d(T)+4=4 và số cạnh là 1.

Bước 2. Kiểm tra:

Số cạnh của T là 1 < 9 (n-1), sang bước 3.

Bước 3. Thêm cạnh:

Trong các cạnh không thuộc T cạnh (a,b) có độ dài nhỏ nhất là 5.

Thêm cạnh (a,b) vào T. Ta có

T = ({a,b,c,d,e,f,g,h,i,j}, {(g,h), (a,b)}) , d(T):=d(T)+5=9 và số cạnh là 2.

Bước 2. Kiểm tra:

Số cạnh của T là 2 < 9 (n-1), sang bước 3.

Bước 3. Thêm cạnh:

Trong các cạnh không thuộc T cạnh (f,g) có độ dài nhỏ nhất là 6.

Thêm cạnh (f,g) vào T. Ta có

T = ({a,b,c,d,e,f,g,h,i,j}, {(g,h), (a,b), (f,g)}) , d(T):=d(T)+6=15 và số cạnh là 3.

Bước 2. Kiểm tra:

Số cạnh của T là 2 < 9 (n-1), sang bước 3.

Bước 3. Thêm cạnh:

Trong các cạnh không thuộc T cạnh (f,g) có độ dài nhỏ nhất là 6.

Thêm cạnh (f,g) vào T. Ta có

T = ({a,b,c,d,e,f,g,h,i,j}, {(g,h), (a,b), (f,g)}) , d(T):=d(T)+6=15 và số cạnh là 3.

Bước 2. Kiểm tra:

Số cạnh của T là 3 < 9 (n-1), sang bước 3.

Bước 3. Thêm cạnh:

Trong các cạnh không thuộc T cạnh (c,d) có độ dài nhỏ nhất là 7.

Thêm cạnh (c,d) vào T. Ta có

T = ({a,b,c,d,e,f,g,h,i,j}, {(g,h), (a,b), (f,g), (c,d)}) , d(T):=d(T)+7=22 và số cạnh là 4.

Bước 2. Kiểm tra:

Số cạnh của T là 4< 9 (n-1), sang bước 3.

Bước 3. Thêm cạnh:

Trong các cạnh không thuộc T cạnh (c,f) có độ dài nhỏ nhất là 7.

Thêm cạnh (c,f) vào T. Ta có

T = ({a,b,c,d,e,f,g,h,i,j}, {(g,h), (a,b), (f,g), (c,d), (c,f) }) , d(T):=d(T)+7=29 và số cạnh là 5.

Bước 2. Kiểm tra:

Số cạnh của T là 5 < 9 (n-1), sang bước 3.

Bước 3. Thêm cạnh:

Trong các cạnh không thuộc T cạnh (a,c) có độ dài nhỏ nhất là 8.

Thêm cạnh (a,c) vào T. Ta có

T = ({a,b,c,d,e,f,g,h,i,j}, {(g,h), (a,b), (f,g), (c,d), (c,f), (a,c)}) ,
d(T):=d(T)+8=37 và số cạnh là 6.

Bước 2. Kiểm tra:

Số cạnh của T là 6 < 9 (n-1), sang bước 3.

Bước 3. Thêm cạnh:

Trong các cạnh không thuộc T cạnh (a,i) có độ dài nhỏ nhất là 9 (cạnh (d,f)=8 không chọn vì tạo thành chu trình).

Thêm cạnh (a,c) vào T. Ta có

T = ({a,b,c,d,e,f,g,h,i,j}, {(g,h), (a,b), (f,g), (c,d), (c,f), (a,c), (a,i)}) ,

d(T):=d(T)+9=45 và số cạnh là 7.

Bước 2. Kiểm tra:

Số cạnh của T là 7 < 9 (n-1), sang bước 3.

Bước 3. Thêm cạnh:

Trong các cạnh không thuộc T cạnh (i,j) có độ dài nhỏ nhất là 11 (cạnh (i,h)=9 không chọn vì tạo thành chu trình).

Thêm cạnh (i,j) vào T. Ta có

T = ({a,b,c,d,e,f,g,h,i,j}, {(g,h), (a,b), (f,g), (c,d), (c,f), (a,c), (a,i), (i,j)}) ,

d(T):=d(T)+11=56 và số cạnh là 8.

Bước 2. Kiểm tra:

Số cạnh của T là 8 < 9 (n-1), sang bước 3.

Bước 3. Thêm cạnh:

Trong các cạnh không thuộc T cạnh (d,e) có độ dài nhỏ nhất là 12 (cạnh (b,c)=11 không chọn vì tạo thành chu trình).

Thêm cạnh (d,e) vào T. Ta có

T = ({a,b,c,d,e,f,g,h,i,j}, {(g,h), (a,b), (f,g), (c,d), (c,f), (a,c), (a,i), (i,j), (d,e)}) ,

d(T):=d(T)+12=68 và số cạnh là 9.

Bước 2. Kiểm tra:

Số cạnh của T là 9= n-1, kết luận T là cây phủ nhỏ nhất, các cạnh của T là các cạnh nét đứt và có d(T)=68.

8.3.4 Ứng dụng

a) Xây dựng một hệ thống đường sắt

Giả sử ta muốn xây dựng một hệ thống đường sắt nối n thành phố sao cho giữa hai thành phố bất kỳ đều có thể thông thương với nhau. Nếu biết trước chi phí xây dựng các tuyến đường thì bài toán đặt ra là phải xây dựng những tuyến đường nào để tổng chi phí xây dựng là ít nhất. Nếu ta coi các thành phố là đỉnh và các tuyến đường nối các thành phố là cạnh kèm theo chi phí xây dựng thì ta có trọng đồ. Khi đó bài toán quy về tìm cây phủ bé nhất của đồ thị.

b) Bài toán cây phủ lớn nhất

Cho G = (V, E, (c_{ij})) là đồ thị liên thông có trọng số (c_{ij}). Tìm cây phủ T của

G có tổng trọng số d(T) là lớn nhất.

$$d(T) = \sum_{(i,j)\in T} c_{ij}$$

Xét trọng đồ G’ = (V, E, (−cij)). áp dụng các thuật toán tìm cây phủ nhỏ nhất Prim hoặc Kruskal (vì chúng không phụ thuộc dấu của trọng số) ta tìm cây phủ nhỏ nhất T* của đồ thị G’ thỏa. Như vậy cây T* cũng là cây phủ lớn nhất của đồ thị G

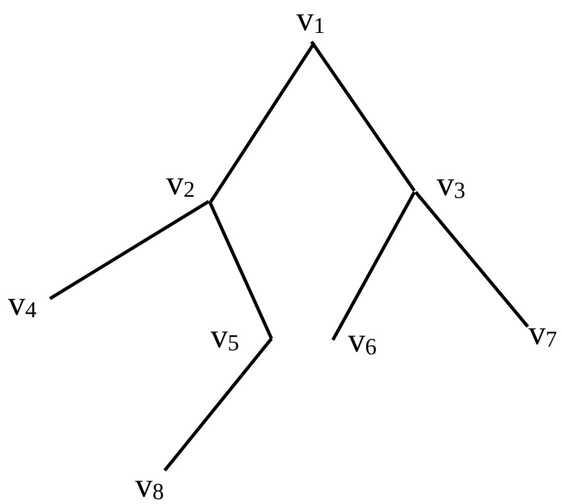
8.4 Cây nhị phân tìm kiếm

8.4.1 Cây nhị phân

Cây nhị phân là một dạng cấu trúc cây quan trọng, mỗi nút của nó chỉ có tối đa hai nút con.

Với mỗi nút trên cây nhị phân, cây con xuất phát từ nút con trái gọi là cây con trái và cây con xuất phát từ nút con phải gọi là cây con phải của nó.

Ví dụ 8. Cây sau là cây nhị phân



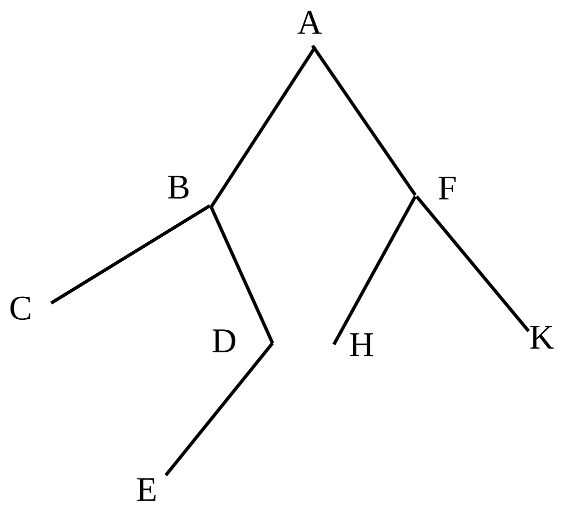
Hình 8.11 Cây nhị phân

8.4.2 Duyệt cây

Phép duyệt cây là lần lượt đi qua tất cả các nút và mỗi nút chỉ qua 1 lần. Có 6 cách duyệt cây dựa vào thứ tự duyệt của các nút bên trái (L), nút gốc (N) và các nút bên phải (R) là:

- Thứ tự Preorder: NLR, NRL (nút gốc trước)
- Thứ tự Inorder: LNR, RNL (nút gốc giữa)
- Thứ tự Postorder: LRN, RLN (nút gốc sau)

Ví dụ 9. Cây sau là cây nhị phân. Duyệt cây nhị phân theo thứ tự trên



Hình 8.12 Cây nhị phân có các nút là các chữ cái

Thứ tự Preorder: **NLR**: ABCDEFHK, **NRL**: AFKHBDEC

Thứ tự Inorder: **LNR**: CBEDAHFK, **RNL**: KFHADEBC

Thứ tự Postorder: **LRN**: CEDBHKFA, **RLN**: KHFEDCBA

Nhận xét:

- **NLR** đối xứng với **RLN**
- **LNR** đối xứng với **RNL**
- **LRN** đối xứng với **NRL**

Do đó ta chỉ xét ba phép duyệt cơ bản là NLR, LNR, LRN

a) Thuật toán duyệt cây theo thứ tự NLR - Preorder (Giải thuật đệ quy)

- Đầu vào. *Gốc r của cây nhị phân*
- Đầu ra. *Danh sách dữ liệu ở các nút*
- Phương pháp.

    1. Duyệt gốc

    2. Duyệt cây bên trái

    3. Duyệt cây bên phải

```
void Traverse_NLR (pointer *r)
    {
        if (r <> NULL)
        {
            cout<<r->Info;
```

```
            Traverse_NLR(r->Left);
            Traverse_NLR(r->Right);
        }
    }
```

b) Thuật toán duyệt cây theo thứ tự LNR - Inorder (giải thuật đệ quy)

- Đầu vào. *Gốc r của cây nhị phân*
- Đầu ra. *Danh sách dữ liệu ở các nút*
- Phương pháp.

    1. Duyệt cây bên trái

    2. Duyệt gốc

    3. Duyệt cây bên phải

```
void Traverse_LNR ( pointer *r)
    {
        if (r <> NULL)
        {
            Traverse_LNR(r->Left);
            cout<<r->Info;
            Traverse_LNR(r->Right);
        }
    }
```

c) Thuật toán 3.4.3. Duyệt cây theo thứ tự LRN - Postorder: (giải thuật đệ quy)

- Đầu vào. *Gốc r của cây nhị phân*
- Đầu ra. *Danh sách dữ liệu ở các nút*
- Phương pháp.

    1. Duyệt cây bên trái

    2. Duyệt cây bên phải

    3. Duyệt gốc

```
void Traverse_LRN (pointer *r)
    {
        if (r <> NULL)
        {
```

```
                Traverse_LRN(r->Left);
                Traverse_LRN(r->Right);
                cout<<r->Info;
            }
        }
```

8.4.3 Cây nhị phân tìm kiếm (binary search tree)

Cây nhị phân tìm kiếm là cây nhị phân trong đó dữ liệu được gán với các nút và dữ liệu được sắp xếp theo khóa sao cho khóa tại mỗi nút của cây lớn hơn khóa của các nút cây con bên trái và nhỏ hơn hoặc bằng khóa của các nút cây con bên phải.

Gọi:

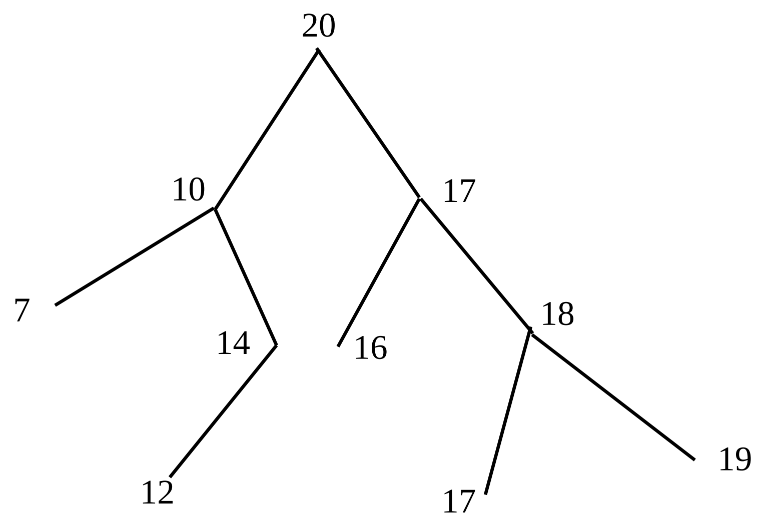
K_n khoá nút xét

K_l khoá nút con bên trái

K_r khoá nút con bên phải

Khi đó ta có $K_l < K_n \leq K_r$

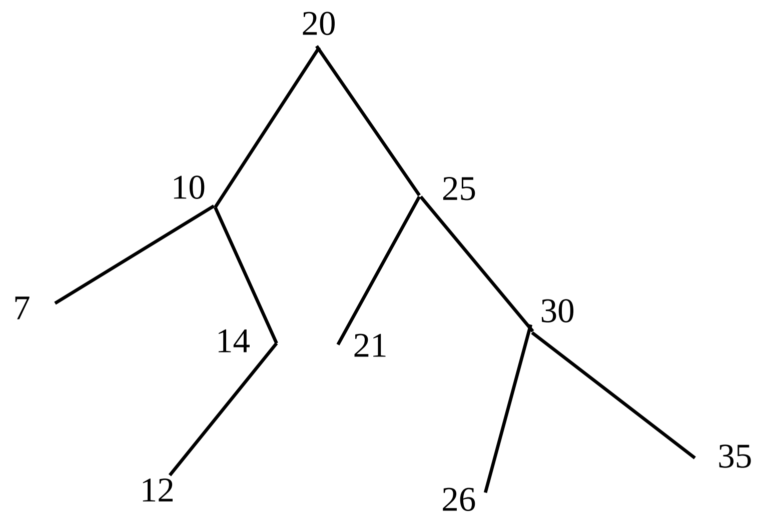
Ví dụ 10. Cây sau không là cây nhị phân tìm kiếm



Hình 8.13 Cây nhị phân có các nút là các số

Nếu duyệt theo LNR: 7, 10, 12, 14, 20, 16, 17, 17, 18, 19

***Ví dụ 11*:** Cây sau là cây nhị phân tìm kiếm



Hình 8.14 Cây nhị phân tìm kiếm

Nếu duyệt theo LNR: 7, 10, 12, 14, 20, 21, 25, 26, 30, 35

Cây nhị phân tìm kiếm được sử dụng để sắp xếp và tìm kiếm dữ liệu. Như trên đã thấy, nếu dữ liệu được lưu trong cây nhị phân tìm kiếm thì duyệt cây theo phương pháp **LNR** ta sẽ được danh sách dữ liệu sắp xếp tăng dần theo khóa.

Bài tập chương 8

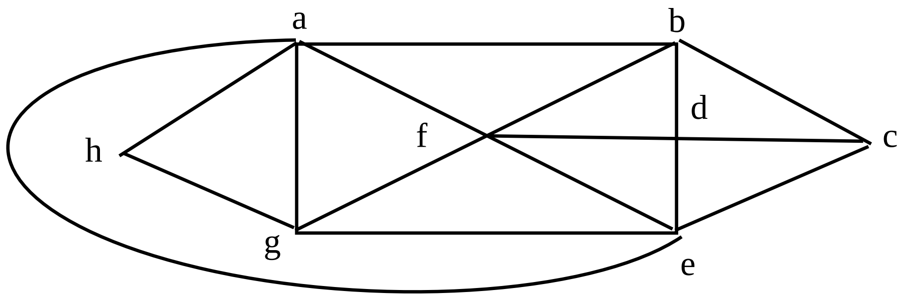
8.1 Cây nhị phân đầy đủ với 1000 đỉnh trong có bao nhiêu cạnh tất cả?

8.2 Cây tam phân đầy đủ với 100 đỉnh trong có bao nhiêu lá tất cả?

8.3 Giả sử có 512 đội bóng tham gia một giải với phương thức loại trực tiếp (1 trận thua sẽ bị loại), cho đến khi tìm được 1 đội vô địch. Hãy lập mô hình các trận đấu để xác định số trận đấu phải tổ chức.

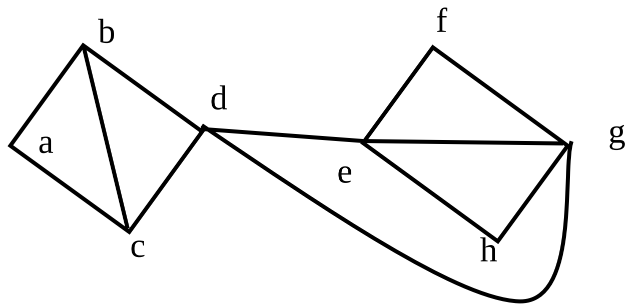
8.4 Giả sử có 612 đội bóng tham gia một giải với phương thức loại trực tiếp (1 trận thua sẽ bị loại), cho đến khi tìm được 1 đội vô địch. Hãy lập mô hình các trận đấu để xác định số trận đấu phải tổ chức.

8.5 Tìm cây phủ theo chiều rộng của đồ thị cho dưới đây



8.6 Tìm cây phủ theo chiều sâu của đồ thị cho dưới đây với đồ thị cho ở bài tập 5

8.7 Tìm cây phủ theo chiều rộng của đồ thị cho dưới đây



8.8 Tìm cây phủ theo chiều sâu của đồ thị cho dưới đây với đồ thị cho ở bài tập 7

8.9 Cho đơn đồ thị đầy đủ K_5 với V={1,2,3,4,5} trọng số của các cạnh (i,j) ký hiệu là d(i,j)=|i-j|. Tìm cây phủ nhỏ nhất bằng thuật toán Prim

8.10 Cho đơn đồ thị đầy đủ K_5 với V={1,2,3,4,5} trọng số của các cạnh (i,j) ký hiệu là d(i,j)=|i-j|. Tìm cây phủ nhỏ nhất bằng thuật toán Kruskal

8.11 Cho đơn đồ thị đầy đủ K_5 với V={1,2,3,4,5} trọng số của các cạnh (i,j) ký hiệu là d(i,j)=i+j. Tìm cây phủ nhỏ nhất bằng thuật toán Prim

8.12 Cho đơn đồ thị đầy đủ K_5 với V={1,2,3,4,5} trọng số của các cạnh (i,j) ký hiệu là d(i,j)=i+j. Tìm cây phủ nhỏ nhất bằng thuật toán Kruskal

TÀI LIỆU THAM KHẢO

[1]. Trần Quốc Chiến, *Giáo trình toán rời rạc,* lưu hành nội bộ tại trường Đại học Sư phạm, Đại học Đà Nẵng, 2007.

[2]. Trần Quốc Chiến, *Giáo trình lý thuyết đồ thị*, lưu hành nội bộ tại trường Đại học Sư phạm, Đại học Đà Nẵng, 2007.

[3]. Nguyễn Đình Lầu, *Song song hóa các thuật toán trên mạng đồ thị*, luận án tiến sĩ, Đại học Đà Nẵng, 2015.

[4]. Nguyễn Đức Nghĩa, Nguyễn Tô Thành, *Giáo trình toán rời rạc*, nhà xuất bản Khoa học Kỹ thuật, 2009.

[5]. Nguyễn Xuân Quỳnh, *Toán rời rạc cho kỹ thuật số*, nhà xuất bản Khoa học Kỹ thuật, 2010.

[6]. Nguyễn Cam, Chu Đức Khánh, *Lý thuyết đồ thị*, nhà xuất bản Đại học Quốc Gia TP.HCM, 2008.

[7]. Douglas West, *Introduction to Graph Theory*, Pearson, ISBN: 978-0131437371, 2017

[8]. Richard J Trudeau, *Introduction to Graph Theory*, Dover Publications, ISBN: 978-0486678702, 2017

[9] Reinhard Diestel , Graph Theory, Springer, ISBN: 978-3662575604, 2017

[10]. Kenneth H. Rosen, *Discrete Mathematics and Its Applications,* McGraw Hill, ISBN: 978-0073383095, 2018