



# **Giáo trình lý thuyết đồ thị**

**Biên tập bởi:**

Thạc sĩ Nguyễn Thanh Hùng

# **Giáo trình lý thuyết đồ thị**

**Biên tập bởi:**

Thạc sĩ Nguyễn Thanh Hùng

**Các tác giả:**

Thạc sĩ Nguyễn Thanh Hùng

Phiên bản trực tuyến:

<http://voer.edu.vn/c/9c021e14>

# MỤC LỤC

1. Các khái niệm cơ bản của lý thuyết đồ thị
  2. Các thuật ngữ cơ bản
  3. Đường đi, chu trình, độ liên thông
  4. Một số dạng đồ thị đặc biệt
  5. Biểu diễn đồ thị trên máy vi tính
  6. Danh sách cạnh(cung)
  7. Danh sách kề
  8. Các thuật toán tìm kiếm trên đồ thị và ứng dụng
  9. Tìm kiếm theo chiều rộng trên đồ thị
  10. Tìm đường đi và kiểm tra tính liên thông
  11. Đồ thị Euler và đồ thị Hamiton
  12. Đồ thị Hamilton
  13. Cây và cây khung của đồ thị
  14. Cây khung của đồ thị
  15. Xây dựng tập các chu trình cơ bản của đồ thị
  16. Bài toán cây khung nhỏ nhất
  17. Bài toán đường đi ngắn nhất
  18. Đường đi ngắn nhất xuất phát từ một đỉnh
  19. Trường hợp ma trận trọng số không âm-Thuật toán Dijkstra
  20. Đồ thị trong đồ thị không có chu trình
  21. Đường đi ngắn nhất giữa tất cả các cặp đỉnh
  22. Bài toán luồng cực đại trong mạng
  23. Lát cắt.Đường tăng luồng.Định lý Ford\_Fulkerson
  24. Thuật toán tìm luồng cực đại
  25. Một số bài toán luồng tổng quát
  26. Một số ứng dụng trong tổ hợp
  27. Bài tập chương 3
  28. Bài tập chương 4
  29. Bài tập chương 5
  30. Bài tập chương 6
  31. Bài tập chương 7
  32. Các bài tập khác
- Tham gia đóng góp

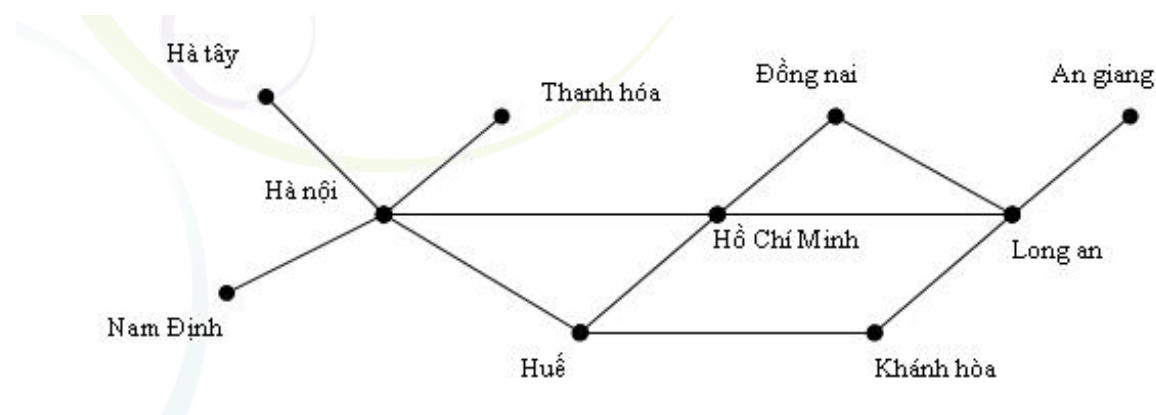
# Các khái niệm cơ bản của lý thuyết đồ thị

Lý thuyết đồ thị là một lĩnh vực đã có từ lâu và có nhiều ứng dụng hiện đại. 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 Thụy Sĩ Lenhard Euler. Chính ông là người đã sử dụng đồ thị để giải bài toán nổi tiếng về các cái cầu ở thành phố Königsberg.

Đồ thị được sử dụng để giải các bài toán trong nhiều lĩnh vực khác nhau. Chẳng hạn, đồ thị có thể sử dụng để xác định các mạch vòng trong vấn đề giải tích mạch điện. Chúng ta có thể phân biệt các hợp chất hóa học hữu cơ khác nhau với cùng công thức phân tử nhưng khác nhau về cấu trúc phân tử nhờ đồ thị. Chúng ta có thể xác định 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. Chúng ta cũng còn sử dụng đồ thị để giải các bài toán về lập lịch, thời khóa biểu, và phân bố tần số cho các trạm phát thanh và truyền hình...

## ĐỊNH NGHĨA ĐỒ THỊ

Đồ thị là một cấu trúc rời rạc bao gồm các đỉnh và các cạnh nối các đỉnh này. Chúng ta phân biệt các loại đồ thị khác nhau bởi *kiểu* và *số lượng* cạnh nối hai đỉnh nào đó của đồ thị. Để có thể hình dung được tại sao lại cần đến các loại đồ thị khác nhau, chúng ta sẽ nêu ví dụ sử dụng chúng để mô tả một mạng máy tính. Giả sử ta có một mạng gồm các máy tính và các kênh điện thoại (gọi tắt là kênh thoại) nối các máy tính này. Chúng ta có thể biểu diễn các vị trí đặt máy tính bởi các điểm và các kênh thoại nối chúng bởi các đoạn nối, xem hình 1.



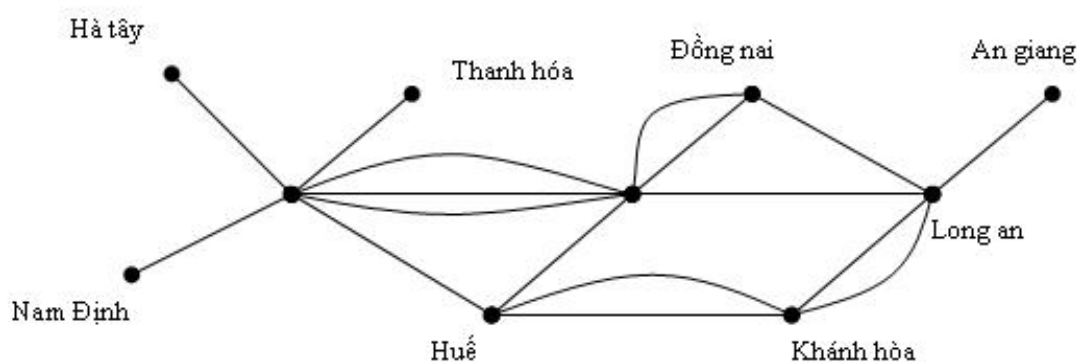
Hình 1. Sơ đồ mạng máy tính.

Nhận thấy rằng trong mạng ở hình 1, giữa hai máy bất kỳ chỉ có nhiều nhất là một kênh thoại nối chúng, kênh thoại này cho phép liên lạc cả hai chiều và không có máy tính nào lại được nối với chính nó. Sơ đồ mạng máy cho trong hình 1 được gọi là *đơn đồ thị vô hướng*. Ta đi đến định nghĩa sau

### Định nghĩa 1.

Đơn đồ thị vô hướng  $G = (V, E)$  bao gồm  $V$  là tập các đỉnh, và  $E$  là tập các cặp không có thứ tự gồm hai phần tử khác nhau của  $V$  gọi là các cạnh.

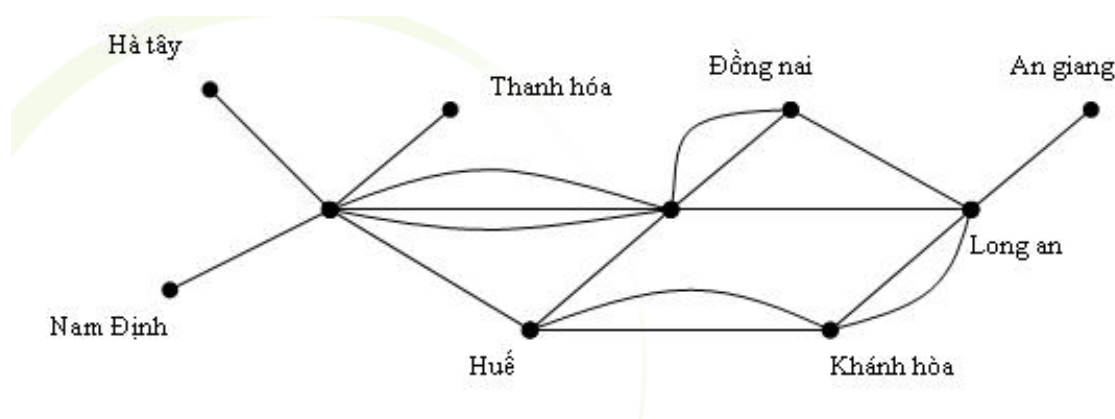
Trong trường hợp giữa hai máy tính nào đó thường xuyên phải truyền tải nhiều thông tin người ta phải nối hai máy này bởi nhiều kênh thoại. Mạng với đa kênh thoại giữa các máy được cho trong hình 2.



**Hình 2.** Sơ đồ mạng máy tính với đa kênh thoại.

### Định nghĩa 2.

Đa đồ thị vô hướng  $G = (V, E)$  bao gồm  $V$  là tập các đỉnh, và  $E$  là tập các cặp không có thứ tự gồm hai phần tử khác nhau của  $V$  gọi là các cạnh. Hai cạnh  $e_1$  và  $e_2$  được gọi là cạnh lặp nếu chúng cùng tương ứng với một cặp đỉnh.

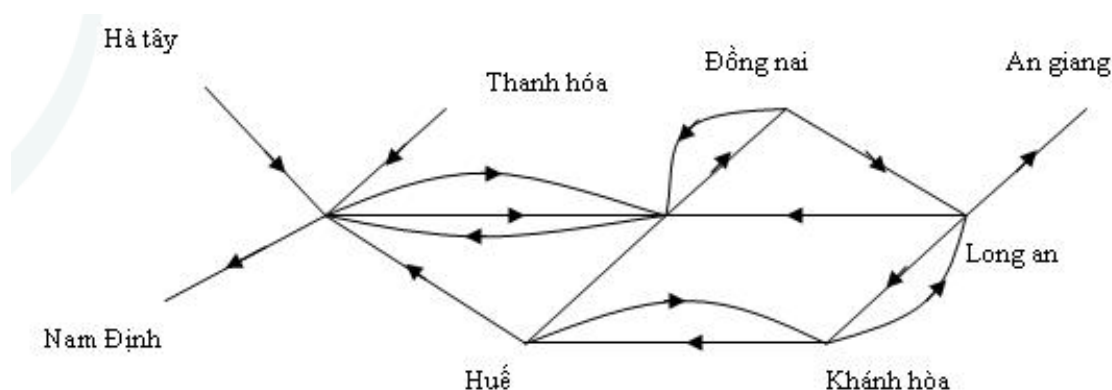


**Hình 3.** Sơ đồ mạng máy tính với kênh thoại thông báo.

Rõ ràng mỗi đơn đồ thị đều là đa đồ thị, nhưng không phải đa đồ thị nào cũng là đơn đồ thị, vì trong đa đồ thị có thể có hai (hoặc nhiều hơn) cạnh nối một cặp đỉnh nào đó. Trong mạng máy tính có thể có những kênh thoại nối một máy nào đó với chính nó (chẳng hạn với mục đích thông báo). Mạng như vậy được cho trong hình 3. Khi đó đa đồ thị không thể mô tả được mạng như vậy, bởi vì có những khuyên (cạnh nối một đỉnh với chính nó). Trong trường hợp này chúng ta cần sử dụng đến khái niệm giả đồ thị vô hướng, được định nghĩa như sau:

### Định nghĩa 3.

Giả đồ thị vô hướng  $G = (V, E)$  bao gồm  $V$  là tập các đỉnh và  $E$  là tập các cặp không có thứ tự gồm hai phần tử (không nhất thiết phải khác nhau) của  $V$  gọi là cạnh. Cạnh  $e$  được gọi là khuyên nếu nó có dạng  $e = (u, u)$ .



**Hình 4.** Mạng máy tính với kênh thoại một chiều

Các kênh thoại trong mạng máy tính có thể chỉ cho phép truyền tin theo một chiều. Chẳng hạn, trong hình 4 máy chủ ở Hà Nội chỉ có thể nhận tin từ các máy ở địa phương, có một số máy chỉ có thể gửi tin đi, còn các kênh thoại cho phép truyền tin theo cả hai chiều được thay thế bởi hai cạnh có hướng ngược chiều nhau.

Ta đi đến định nghĩa sau.

### Định nghĩa 4.

Đơn đồ thị có hướng  $G = (V, E)$  bao gồm  $V$  là tập các đỉnh và  $E$  là tập các cặp có thứ tự gồm hai phần tử khác nhau của  $V$  gọi là các cung.

Nếu trong mạng có thể có đa kênh thoại một chiều, ta sẽ phải sử dụng đến khái niệm đa đồ thị có hướng

### **Định nghĩa 5.**

Đồ thị có hướng  $G = (V, E)$  bao gồm  $V$  là tập các đỉnh và  $E$  là tập các cặp có thứ tự gồm hai phần tử khác nhau của  $V$  gọi là các cung. Hai cung  $e_1, e_2$  tương ứng với cùng một cặp đỉnh được gọi là cung lặp.

Trong các phần tiếp theo chủ yếu chúng ta sẽ làm việc với đơn đồ thị vô hướng và đơn đồ thị có hướng. Vì vậy, để cho ngắn gọn, ta sẽ bỏ qua tính từ **đơn** khi nhắc đến chúng.

# Các thuật ngữ cơ bản

Trong mục này chúng ta sẽ trình bày một số thuật ngữ cơ bản của lý thuyết đồ thị. Trước tiên, ta xét các thuật ngữ mô tả các đỉnh và cạnh của đồ thị vô hướng.

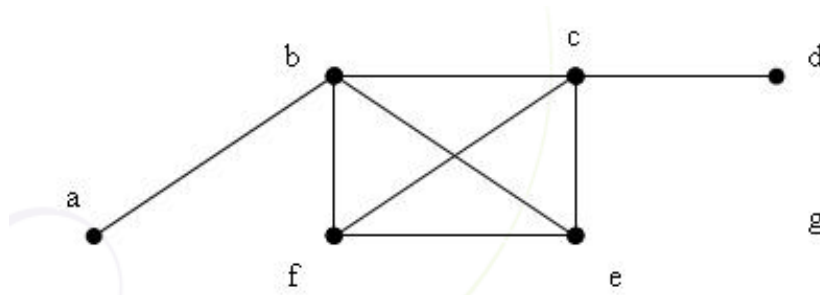
## Định nghĩa 1.

Hai đỉnh  $u$  và  $v$  của đồ thị vô hướng  $G$  được gọi là kề nhau nếu  $(u,v)$  là cạnh của đồ thị  $G$ . Nếu  $e = (u, v)$  là cạnh của đồ thị ta nói cạnh này là liên thuộc với hai đỉnh  $u$  và  $v$ , hoặc cũng nói là nối đỉnh  $u$  và đỉnh  $v$ , đồng thời các đỉnh  $u$  và  $v$  sẽ được gọi là các đỉnh đầu của cạnh  $(u, v)$ .

Để có thể biết có bao nhiêu cạnh liên thuộc với một đỉnh, ta đưa vào định nghĩa sau

## Định nghĩa 2.

Ta gọi bậc của đỉnh  $v$  trong đồ thị vô hướng là số cạnh liên thuộc với nó và sẽ ký hiệu là  $\deg(v)$ .



**Hình 1.** Đồ thị vô hướng

**Thí dụ 1.** Xét đồ thị cho trong hình 1, ta có

$$\deg(a) = 1, \deg(b) = 4, \deg(c) = 4, \deg(f) = 3,$$

$$\deg(d) = 1, \deg(e) = 3, \deg(g) = 0$$

Đỉnh bậc 0 gọi là *đỉnh cô lập*. Đỉnh bậc 1 được gọi là *đỉnh treo*. Trong ví dụ trên đỉnh  $g$  là đỉnh cô lập,  $a$  và  $d$  là các đỉnh treo. Bậc của đỉnh có tính chất sau:

**Định lý 1.** Giả sử  $G = (V, E)$  là đồ thị vô hướng với  $m$  cạnh. Khi đó tổng bậc của tất cả các đỉnh bằng hai lần số cung.



**Chứng minh.** Rõ ràng mỗi cạnh  $e = (u, v)$  được tính một lần trong  $\deg(u)$  và một lần trong  $\deg(v)$ . Từ đó suy ra tổng tất cả các bậc của các đỉnh bằng hai lần số cạnh.

**Thí dụ 2.** Đồ thị với  $n$  đỉnh có bậc là 6 có bao nhiêu cạnh?

**Giải:** Theo định lý 1 ta có  $2m = 6n$ . Từ đó suy ra tổng các cạnh của đồ thị là  $3n$ .

**Hệ quả.** Trong đồ thị vô hướng, số đỉnh bậc lẻ (nghĩa là có bậc là số lẻ) là một số chẵn.

**Chứng minh.** Thực vậy, gọi  $O$  và  $U$  tương ứng là tập đỉnh bậc lẻ và tập đỉnh bậc chẵn của đồ thị. Ta có

$$2m = \sum_{v \in U} \deg(v) + \sum_{v \in O} \deg(v)$$

Do  $\deg(v)$  là chẵn với  $v$  là đỉnh trong  $U$  nên tổng thứ nhất ở trên là số chẵn. Từ đó suy ra tổng thứ hai (chính là tổng bậc của các đỉnh bậc lẻ) cũng phải là số chẵn, do tất cả các số hạng của nó là số lẻ, nên tổng này phải gồm một số chẵn các số hạng. Vì vậy, số đỉnh bậc lẻ phải là số chẵn.

Ta xét các thuật ngữ tương tự cho đồ thị vô hướng.

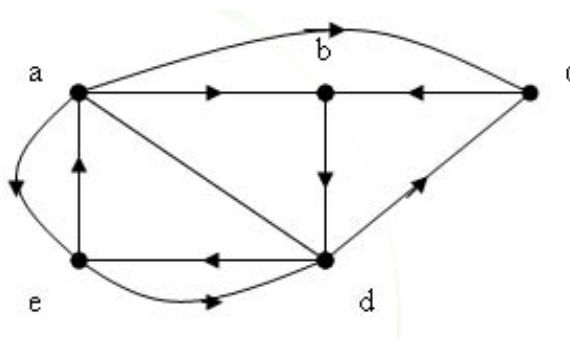
### Định nghĩa 3.

Nếu  $e = (u, v)$  là cung của đồ thị có hướng  $G$  thì ta nói hai đỉnh  $u$  và  $v$  là kề nhau, và nói cung  $(u, v)$  nối đỉnh  $u$  với đỉnh  $v$  hoặc cũng nói cung này là đi ra khỏi đỉnh  $u$  và vào đỉnh  $v$ . Đỉnh  $u(v)$  sẽ được gọi là đỉnh đầu (cuối) của cung  $(u, v)$ .

Tương tự như khái niệm bậc, đối với đồ thị có hướng ta có khái niệm bán bậc ra và bán bậc vào của một đỉnh.

### Định nghĩa 4.

Ta gọi bán bậc ra (bán bậc vào) của đỉnh  $v$  trong đồ thị có hướng là số cung của đồ thị đi ra khỏi nó (đi vào nó) và ký hiệu là  $\deg^+(v)$  ( $\deg^-(v)$ )



## Hình 2. Đồ thị có hướng

**Thí dụ 3.** Xét đồ thị cho trong hình 2. Ta có

$$\deg^-(a)=1, \deg^-(b)=2, \deg^-(c)=2, \deg^-(d)=2, \deg^-(e)=2.$$

$$\deg^+(a)=3, \deg^+(b)=1, \deg^+(c)=1, \deg^+(d)=2, \deg^+(e)=2.$$

Do mỗi cung  $(u, v)$  sẽ được tính một lần trong bán bậc vào của đỉnh  $v$  và một lần trong bán bậc ra của đỉnh  $u$  nên ta có:

**Định lý 2.** Giả sử  $G = (V, E)$  là đồ thị có hướng. Khi đó

$$2m = \sum_{v \in V} \deg^+(v) + \sum_{v \in V} \deg^-(v)$$

$$\sum_{v \in V} \deg^+(v) = \sum_{v \in V} \deg^-(v)$$

Rất nhiều tính chất của đồ thị có hướng không phụ thuộc vào hướng trên các cung của nó. Vì vậy, trong nhiều trường hợp sẽ thuận tiện hơn nếu ta bỏ qua hướng trên các cung của đồ thị. Đồ thị vô hướng thu được bằng cách bỏ qua hướng trên các cung được gọi là *đồ thị vô hướng tương ứng* với đồ thị có hướng đã cho.

# Đường đi, chu trình, độ liên thông

## Định nghĩa 1.

Đường đi độ dài  $n$  từ đỉnh  $u$  đến đỉnh  $v$ , trong đó  $n$  là số nguyên dương, trên đồ thị vô hướng  $G = (V, E)$  là dãy

$$x_0, x_1, \dots, x_{n-1}, x_n$$

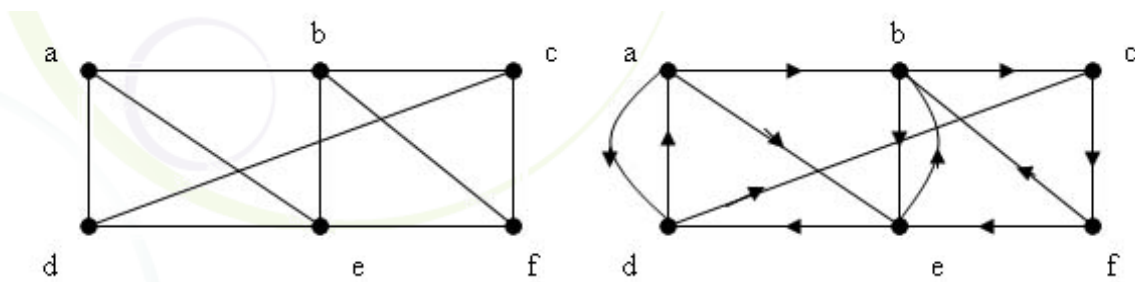
trong đó  $u = x_0, v = x_n, (x_i, x_{i+1}) \in E, i = 0, 1, 2, \dots, n-1$ .

Đường đi nói trên còn có thể biểu diễn dưới dạng dãy các cạnh:

$$(x_0, x_1), (x_1, x_2), \dots, (x_{n-1}, x_n)$$

Đỉnh  $u$  gọi là đỉnh đầu, còn đỉnh  $v$  gọi là đỉnh cuối của đường đi. Đường đi có đỉnh đầu trùng với đỉnh cuối (tức là  $u = v$ ) được gọi là chu trình. Đường đi hay chu trình được gọi là đơn nếu như không có cạnh nào bị lặp lại.

**Thí dụ 1.** Trên đồ thị vô hướng cho trong hình 1: a, d, c, f, e là đường đi đơn độ dài 4. Còn d, e, c, a không là đường đi, do (c,e) không phải là cạnh của đồ thị. Dãy b, c, f, e, b là chu trình độ dài 4. Đường đi a, b, e, d, a, b có độ dài là 5 không phải là đường đi đơn, do cạnh (a, b) có mặt trong nó 2 lần.



**Hình 1.** Đường đi trên đồ thị

Khái niệm đường đi và chu trình trên đồ thị có hướng được định nghĩa hoàn toàn tương tự như trong trường hợp đồ thị vô hướng, chỉ khác là ta có chú ý đến hướng trên các cung.

## Định nghĩa 2.

Đường đi độ dài  $n$  từ đỉnh  $u$  đến đỉnh  $v$ , trong đó,  $n$  là số nguyên dương, trên đồ thị có hướng  $G = (V, A)$  là dãy

$x_0, x_1, \dots, x_{n-1}, x_n$

trong đó  $u = x_0, v = x_n, (x_i, x_{i+1}) \in E, i = 0, 1, 2, \dots, n-1$ .

Đường đi nói trên còn có thể biểu diễn dưới dạng dãy các cung:

$(x_0, x_1), (x_1, x_2), \dots, (x_{n-1}, x_n)$

*Đỉnh  $u$  gọi là đỉnh đầu, còn đỉnh  $v$  gọi là đỉnh cuối của đường đi. Đường đi có đỉnh đầu trùng với đỉnh cuối (tức là  $u = v$ ) được gọi là chu trình. Đường đi hay chu trình được gọi là đơn nếu như không có cạnh nào bị lặp lại.*

**Thí dụ 2.** Trên đồ thị có hướng cho trong hình 1: a, d, c, f, e là đường đi đơn độ dài 4. Còn d, e, c, a không là đường đi, do (c,e) không phải là cạnh của đồ thị. Dãy b, c, f, e, b là chu trình độ dài 4. Đường đi a, b, e, d, a, b có độ dài là 5 không phải là đường đi đơn, do cạnh (a, b) có mặt trong nó 2 lần.

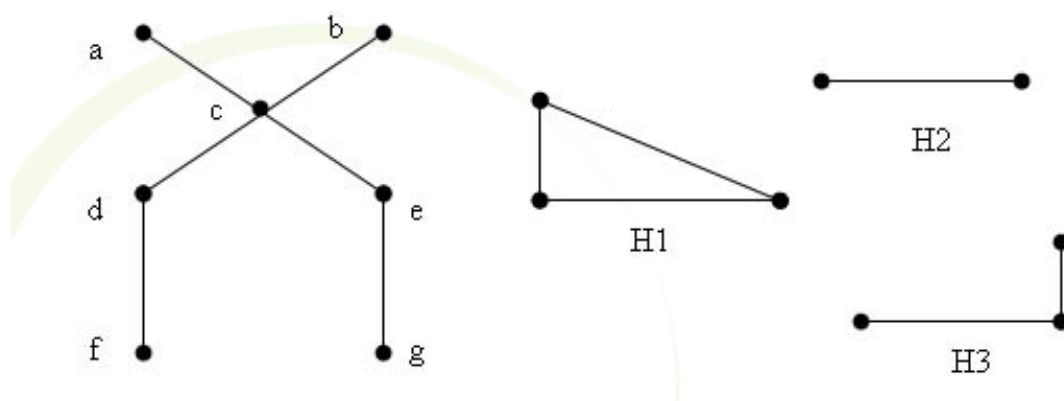
Xét một mạng máy tính. Một câu hỏi đặt ra là hai máy tính bất kỳ trong mạng này có thể trao đổi thông tin được với nhau hoặc là trực tiếp qua kênh nối chúng hoặc thông qua một hoặc vài máy tính trung gian trong mạng? Nếu sử dụng đồ thị để biểu diễn mạng máy tính này (trong đó các đỉnh của đồ thị tương ứng với các máy tính, còn các cạnh tương ứng với các kênh nối) câu hỏi đó được phát biểu trong ngôn ngữ đồ thị như sau: Tồn tại hay không đường đi giữa mọi cặp đỉnh của đồ thị.

### Định nghĩa 3.

Đồ thị vô hướng  $G = (V, E)$  được gọi là liên thông nếu luôn tìm được đường đi giữa hai đỉnh bất kỳ của nó.

Như vậy hai máy tính bất kỳ trong mạng có thể trao đổi thông tin được với nhau khi và chỉ khi đồ thị tương ứng với mạng này là đồ thị liên thông.

**Thí dụ 3.** Trong hình 2 : Đồ thị G là liên thông, còn đồ thị H là không liên thông.



## Hình 2. Đồ thị G và H

### Định nghĩa 4.

Ta gọi đồ thị con của đồ thị  $G = (V, E)$  là đồ thị  $H = (W, F)$ , trong đó  $W \subseteq V$  và  $F \subseteq E$ .

Trong trường hợp đồ thị là không liên thông, nó sẽ rã ra thành một số đồ thị con liên thông đôi một không có đỉnh chung. Những đồ thị con liên thông như vậy ta sẽ gọi là các thành phần liên thông của đồ thị.

**Thí dụ 4.** Đồ thị H trong hình 2 gồm 3 thành phần liên thông  $H_1, H_2, H_3$ .

Trong mạng máy tính có thể có những máy (Những kênh nối) mà sự hỏng hóc của nó sẽ ảnh hưởng đến việc trao đổi thông tin trong mạng. Các khái niệm tương ứng với tình huống này được đưa ra trong định nghĩa sau.

### Định nghĩa 5.

Đỉnh  $v$  được gọi là đỉnh rẽ nhánh nếu việc loại bỏ  $v$  cùng với các cạnh liên thuộc với nó khỏi đồ thị làm tăng số thành phần liên thông của đồ thị. Cạnh  $e$  được gọi là cầu nếu việc loại bỏ nó khỏi đồ thị làm tăng số thành phần liên thông của đồ thị.

**Thí dụ 5.** Trong đồ thị G ở hình 2, đỉnh d và e là đỉnh rẽ nhánh, còn các cạnh (d,g) và (e,f) là cầu.

Đối với đồ thị có hướng có hai khái niệm liên thông phụ thuộc vào việc ta có xét đến hướng trên các cung hay không.

### Định nghĩa 6.

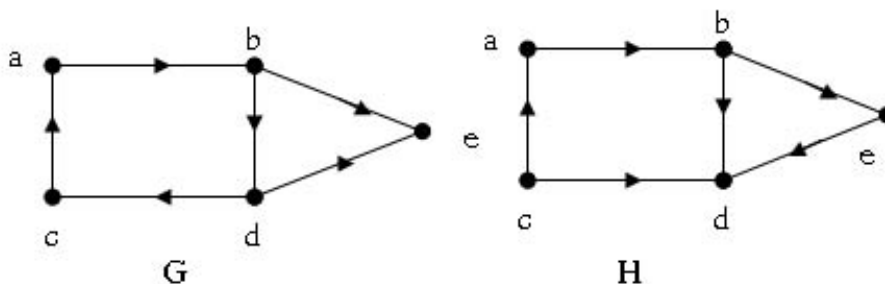
Đồ thị có hướng  $G = (V, A)$  được gọi là liên thông mạnh nếu luôn tìm được đường đi giữa hai đỉnh bất kỳ của nó.

### Định nghĩa 7.

Đồ thị có hướng  $G = (V, A)$  được gọi là liên thông yếu nếu đồ thị vô hướng tương ứng với nó là vô hướng liên thông.

Rõ ràng nếu đồ thị là liên thông mạnh thì nó cũng là liên thông yếu, nhưng điều ngược lại là không luôn đúng, như chỉ ra trong ví dụ dưới đây.

**Thí dụ 6.** Trong hình 3 đồ thị G là liên thông mạnh, còn H là liên thông yếu nhưng không là liên thông mạnh.



Hình 3. Đồ thị liên thông mạnh G và đồ thị liên thông yếu H

Một câu hỏi đặt ra là khi nào có thể định hướng các cạnh của một đồ thị vô hướng liên thông để có thể thu được đồ thị có hướng liên thông mạnh? Ta sẽ gọi đồ thị như vậy là đồ thị định hướng được. Định lý dưới đây cho ta tiêu chuẩn nhận biết một đồ thị có là định hướng được hay không.

### Định lý 1.

Đồ thị vô hướng liên thông là định hướng được khi và chỉ khi mỗi cạnh của nó nằm trên ít nhất một chu trình.

### Chứng minh.

*Điều kiện cần.* Giả sử  $(u, v)$  là một cạnh của một đồ thị. Từ sự tồn tại đường đi có hướng từ  $u$  đến  $v$  và ngược lại suy ra  $(u, v)$  phải nằm trên ít nhất một chu trình.

*Điều kiện đủ.* Thủ tục sau đây cho phép định hướng các cạnh của đồ thị để thu được đồ thị có hướng liên thông mạnh. Giả sử  $C$  là một chu trình nào đó trong đồ thị. Định hướng các cạnh trên chu trình này theo một hướng đi vòng theo nó. Nếu tất cả các cạnh của đồ thị là đã được định hướng thì kết thúc thủ tục. Ngược lại, chọn  $e$  là một cạnh chưa định hướng có chung đỉnh với ít nhất một trong số các cạnh đã định hướng. Theo giả thiết tìm được chu trình  $C'$  chứa cạnh  $e$ . Định hướng các cạnh chưa được định hướng của  $C'$  theo một hướng dọc theo chu trình này (không định hướng lại các cạnh đã có định hướng). Thủ tục trên sẽ được lặp lại cho đến khi tất cả các cạnh của đồ thị được định hướng. Khi đó ta thu được đồ thị có hướng liên thông mạnh.

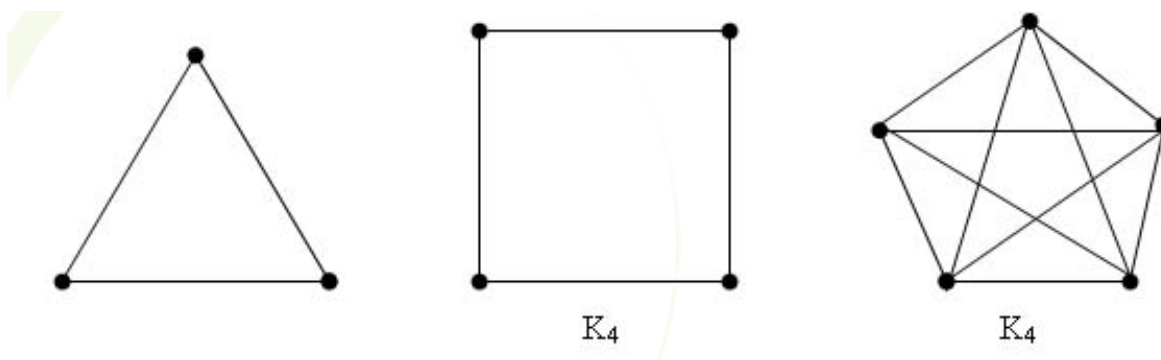
# Một số dạng đồ thị đặc biệt

Trong mục này ta xét một số đơn đồ thị vô hướng dạng đặc biệt xuất hiện trong nhiều vấn đề ứng dụng thực tế.

## Đồ thị đầy đủ.

Đồ thị đầy đủ  $n$  đỉnh, ký hiệu bởi  $K_n$ , là đơn đồ thị vô hướng mà giữa hai đỉnh bất kỳ của nó luôn có cạnh nối.

Các đồ thị  $K_3$ ,  $K_4$ ,  $K_5$  cho trong hình dưới đây.



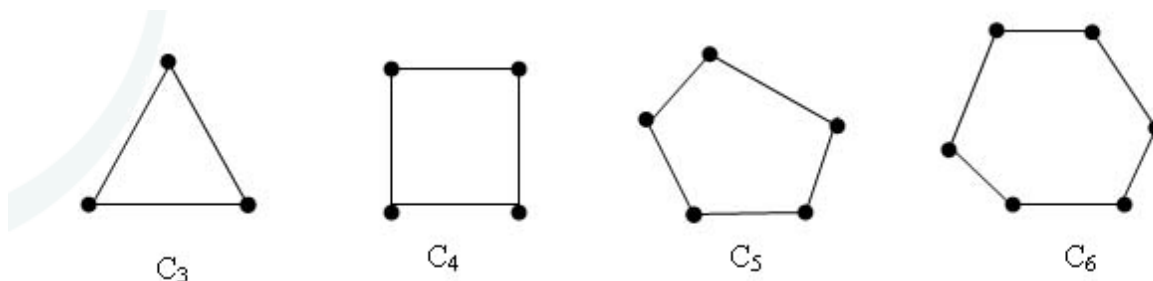
**Hình 1.** Đồ thị đầy đủ

Đồ thị đầy đủ  $K_n$  có tất cả  $n(n-1)/2$  cạnh, nó là đơn đồ thị có nhiều cạnh nhất.

## Đồ thị vòng.

Đồ thị vòng  $C_n$ ,  $n \geq 3$ , gồm  $n$  đỉnh  $v_1, v_2, \dots, v_n$  và các cạnh  $(v_1, v_2), (v_2, v_3), \dots, (v_{n-1}, v_n), (v_n, v_1)$ .

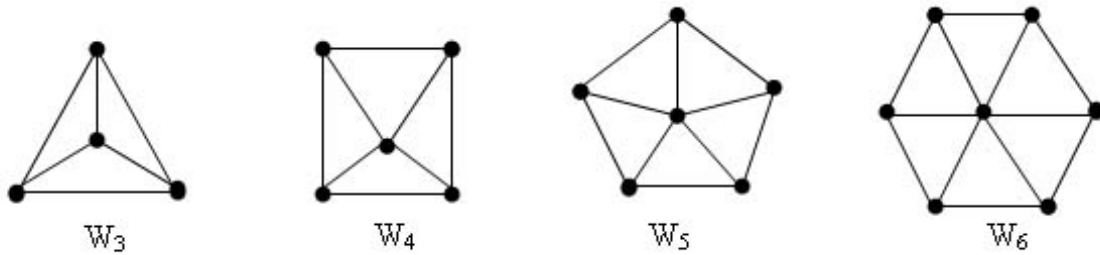
Đồ thị vòng  $C_3$ ,  $C_4$ ,  $C_5$ ,  $C_6$  cho trong hình 2.



**Hình 2.** Đồ thị vòng  $C_3, C_4, C_5, C_6$

### Đồ thị bánh xe.

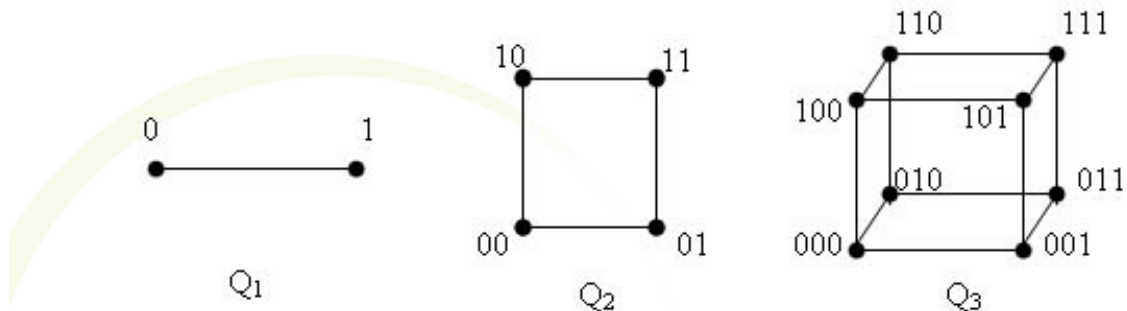
Đồ thị  $W_n$  thu được từ  $C_n$  bằng cách bổ sung vào một đỉnh mới nối với tất cả các đỉnh của  $C_n$  (xem hình 3).



**Hình 3.** Đồ thị bánh xe  $W_3, W_4, W_5, W_6$

### Đồ thị lập phương.

Đồ thị lập phương  $n$  đỉnh  $Q_n$  là đồ thị với các đỉnh biểu diễn  $2^n$  xâu nhị phân độ dài  $n$ . Hai đỉnh của nó gọi là kề nhau nếu như hai xâu nhị phân tương ứng chỉ khác nhau 1 bit. Hình 4 cho thấy  $Q_n$  với  $n=1,2,3$ .



**Hình 4.** Đồ thị lập phương  $Q_1, Q_2, Q_3$

### Đồ thị hai phía.

Đơn đồ thị  $G=(V,E)$  được gọi là hai phía nếu như tập đỉnh  $V$  của nó có thể phân hoạch thành hai tập  $X$  và  $Y$  sao cho mỗi cạnh của đồ thị chỉ nối một đỉnh nào đó trong  $X$  với một đỉnh nào đó trong  $Y$ . Khi đó ta sẽ sử dụng ký hiệu  $G=(X? Y, E)$  để chỉ đồ thị hai phía với tập đỉnh  $X? Y$ .

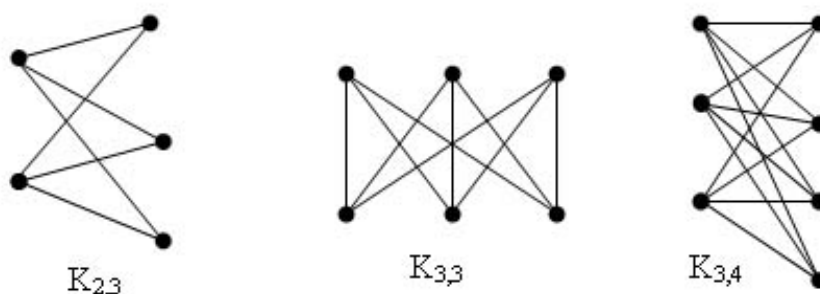


Định lý sau đây cho phép nhận biết một đơn đồ thị có phải là hai phía hay không.

**Định lý 1.** Đơn đồ thị là đồ thị hai phía khi và chỉ khi nó không chứa chu trình độ dài lẻ.

Để kiểm tra xem một đồ thị liên thông có phải là hai phía hay không có thể áp dụng thủ tục sau. Cho  $v$  là một đỉnh bất kỳ của đồ thị. Đặt  $X = \{v\}$ , còn  $Y$  là tập các đỉnh kề của  $v$ . Khi đó các đỉnh kề của các đỉnh trong  $Y$  phải thuộc vào  $X$ . Ký hiệu tập các đỉnh như vậy là  $T$ . Vì thế nếu phát hiện  $T \cap Y \neq \emptyset$  thì đồ thị không phải là hai phía, kết thúc. ngược lại, đặt  $X = X \cup T$ . Tiếp tục xét như vậy đối với  $T'$  là tập các đỉnh kề của  $T$ , . . .

Đồ thị hai phía  $G = (X \cup Y, E)$  với  $|X| = m, |Y| = n$  được gọi là đồ thị hai phía đầy đủ và ký hiệu là  $K_{2,3}, K_{3,3}, K_{3,4}$  được cho trong hình 5. Khi  $E \dots$



**Hình 5.** Đồ thị hai phía

### Đồ thị phẳng.

Đồ thị được gọi là đồ thị phẳng nếu ta có thể vẽ nó trên mặt phẳng sao cho các cạnh của nó không cắt nhau ngoài ở đỉnh. Cách vẽ như vậy sẽ được gọi là biểu diễn phẳng của đồ thị.

Thí dụ đồ thị  $K_4$  là phẳng, vì có thể vẽ nó trên mặt phẳng sao cho các cạnh của nó không cắt nhau ngoài ở đỉnh (xem hình 6).



**Hình 6.** Đồ thị  $K_4$  là đồ thị phẳng

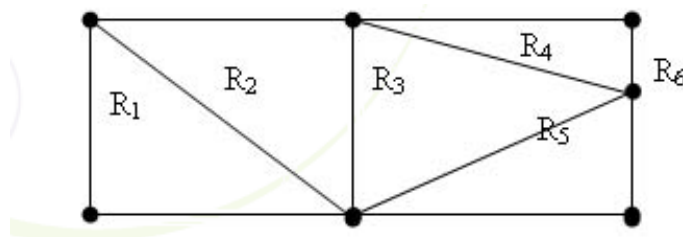
Một điều đáng lưu ý nếu đồ thị là phẳng thì luôn có thể vẽ nó trên mặt phẳng với các cạnh nối là các đoạn thẳng không cắt nhau ngoài ở đỉnh (ví dụ xem cách vẽ  $K_4$  trong hình 6).

Để nhận biết xem một đồ thị có phải là đồ thị phẳng có thể sử dụng định lý Kuratovski, mà để phát biểu nó ta cần một số khái niệm sau: Ta gọi một phép chia cạnh  $(u,v)$  của đồ thị là việc loại bỏ cạnh này khỏi đồ thị và thêm vào đồ thị một đỉnh mới  $w$  cùng với hai cạnh  $(u,w)$ ,  $(w,v)$ . Hai đồ thị  $G(V,E)$  và  $H(W,F)$  được gọi là đồng cấu nếu chúng có thể thu được từ cùng một đồ thị nào đó nhờ phép chia cạnh.

**Định lý 2 (Kuratovski).** Đồ thị là phẳng khi và chỉ khi nó không chứa đồ thị con đồng cấu với  $K_{3,3}$  hoặc  $K_5$ .

Trong trường hợp riêng, đồ thị  $K_{3,3}$  hoặc  $K_5$  không phải là đồ thị phẳng. Bài toán về tính phẳng của đồ thị  $K_{3,3}$  là bài toán nổi tiếng về ba căn hộ và ba hệ thống cung cấp năng lượng cho chúng: Cần xây dựng hệ thống đường cung cấp năng lượng với mỗi một căn hộ nói trên sao cho chúng không cắt nhau. Đồ thị phẳng còn tìm được những ứng dụng quan trọng trong công nghệ chế tạo mạch in.

Biểu diễn phẳng của đồ thị sẽ chia mặt phẳng ra thành các miền, trong đó có thể có cả miền không bị chặn. Thí dụ, biểu diễn phẳng của đồ thị cho trong hình 7 chia mặt phẳng ra thành 6 miền  $R_1, R_2, \dots, R_6$ .



**Hình 7.** Các miền tương ứng với biểu diễn phẳng của đồ thị

Euler đã chứng minh được rằng các cách biểu diễn phẳng khác nhau của một đồ thị đều chia mặt phẳng ra thành cùng một số miền. Để chứng minh điều đó, Euler đã tìm được mối liên hệ giữa số miền, số đỉnh của đồ thị và số cạnh của đồ thị phẳng sau đây.

**Định lý 3 (Công thức Euler).** Giả sử  $G$  là đồ thị phẳng liên thông với  $n$  đỉnh,  $m$  cạnh. Gọi  $r$  là số miền của mặt phẳng bị chia bởi biểu diễn phẳng của  $G$ . Khi đó

$$r = m - n + 2$$

Có thể chứng minh định lý bằng qui nạp. Xét thí dụ minh họa cho áp dụng công thức Euler.

**Thí dụ.** Cho  $G$  là đồ thị phẳng liên thông với 20 đỉnh, mỗi đỉnh đều có bậc là 3. Hỏi mặt phẳng bị chia làm bao nhiêu phần bởi biểu diễn phẳng của đồ thị  $G$ ?

**Giải.** Do mỗi đỉnh của đồ thị đều có bậc là 3, nên tổng bậc của các đỉnh là  $3 \times 20 = 60$ . Từ đó suy ra số cạnh của đồ thị  $m = 60/2 = 30$ . Vì vậy, theo công thức Euler, số miền cần tìm là  $r = 30 - 20 + 2 = 12$ .

# Biểu diễn đồ thị trên máy vi tính

Để lưu trữ đồ thị và thực hiện các thuật toán khác nhau với đồ thị trên máy tính cần phải tìm những cấu trúc dữ liệu thích hợp để mô tả đồ thị. Việc chọn cấu trúc dữ liệu nào để biểu diễn đồ thị có tác động rất lớn đến hiệu quả của thuật toán. Vì vậy, việc chọn lựa cấu trúc dữ liệu để biểu diễn đồ thị phụ thuộc vào từng tình huống cụ thể (bài toán và thuật toán cụ thể). Trong mục này chúng ta sẽ xét một số phương pháp cơ bản được sử dụng để biểu diễn đồ thị trên máy tính, đồng thời cũng phân tích một cách ngắn gọn những ưu điểm cũng như những nhược điểm của chúng.

## MA TRẬN KỀ. MA TRẬN TRỌNG SỐ

Xét đơn đồ thị vô hướng  $G=(V,E)$ , với tập đỉnh  $V=\{ 1, 2, \dots, n \}$ , tập cạnh  $E=\{ e_1, e_2, \dots, e_m \}$ . Ta gọi ma trận kề của đồ thị  $G$  là ma trận.

$$A=\{ a_{i,j} : i,j=1, 2, \dots, n \}$$

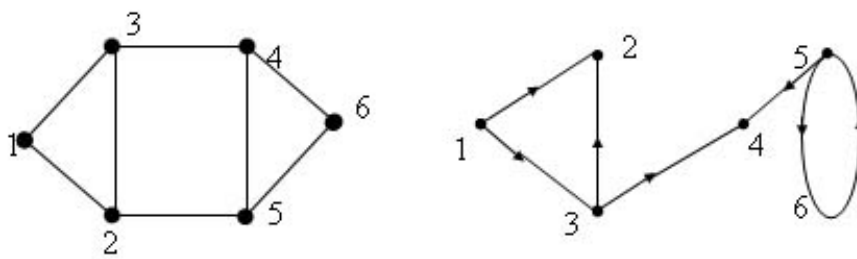
Với các phần tử được xác định theo qui tắc sau đây:

$$a_{i,j} = 0, \text{ nếu } (i,j) \notin E \text{ và}$$

$$a_{i,j} = 1, \text{ nếu } (i,j) \in E, i, j=1, 2, \dots, n.$$

**Thí dụ 1.** Ma trận kề của đồ thị vô hướng cho trong hình 1 là:

	1	2	3	4	5	6
1	0	1	1	0	1	0
2	1	0	1	0	1	0
3	1	1	0	1	0	0
4	0	0	1	0	1	1
5	1	1	0	1	0	1
6	0	0	0	1	1	0



Hình 1. Đồ thị vô hướng  $G$  và Đồ thị có hướng  $G1$

### Các tính chất của ma trận kề:

1) Rõ ràng ma trận kề của đồ thị vô hướng là ma trận đối xứng, tức là

$$a_{ij} = a_{ji}, \quad i, j = 1, 2, \dots, n.$$

ngược lại, mỗi  $(0,1)$ -ma trận đối xứng cấp  $n$  sẽ tương ứng, chính xác đến cách đánh số đỉnh (còn nói là: chính xác đến đẳng cấu), với một đơn đồ thị vô hướng  $n$  đỉnh.

2) Tổng các phần tử trên dòng  $i$  (cột  $j$ ) của ma trận kề chính bằng bậc của đỉnh  $i$  (đỉnh  $j$ ).

3) nếu ký hiệu

$$a_{ij}^p, \quad i, j = 1, 2, \dots, n$$

là phần tử của ma trận

$$A^p = A.A. \dots A$$

p thừa số

Khi đó

$$a_{ij}^p, i, j=1, 2, \dots, n$$

cho ta số đường đi khác nhau từ đỉnh i đến đỉnh j qua p-1 đỉnh trung gian.

Ma trận kề của đồ thị có hướng được định nghĩa một cách hoàn toàn tương tự.

**Thí dụ 2.** Đồ thị có hướng  $G_1$  cho trong hình 1 có ma trận kề là ma trận sau:

	1	2	3	4	5	6
1	0	1	1	0	0	0
2	0	0	0	0	0	0
3	0	1	0	1	0	0
4	0	0	0	0	0	0
5	0	0	0	1	0	1
6	0	0	0	0	1	0

Lưu ý rằng ma trận kề của đồ thị có hướng không phải là ma trận đối xứng.

*Chú ý:* Trên đây chúng ta chỉ xét đơn đồ thị. Ma trận kề của đa đồ thị có thể xây dựng hoàn toàn tương tự, chỉ khác là thay vì ghi 1 vào vị trí  $a[i,j]$  nếu  $(i,j)$  là cạnh của đồ thị, chúng ta sẽ ghi k là số cạnh nối hai đỉnh i, j.

Trong rất nhiều vấn đề ứng dụng của lý thuyết đồ thị, mỗi cạnh  $e=(u,v)$  của đồ thị được gán với một con số  $c(e)$  (còn viết là  $c(u,v)$ ) gọi là trọng số của cạnh e. Đồ thị trong trường hợp như vậy được gọi là đồ thị có trọng số. Trong trường hợp đồ thị có trọng số, thay vì ma trận kề, để biểu diễn đồ thị ta sử dụng ma trận trọng số.

$$C = \{c[i,j], i, j=1, 2, \dots, n\}$$

với

$$c[i,j]=c(i,j) \text{ nếu } (i,j) \in E$$

$$\text{và } c[i,j]=0 \text{ nếu } (i,j) \notin E$$

trong đó số  $\theta$  , tùy từng trường hợp cụ thể, có thể được đặt bằng một trong các giá trị sau:  $0, +\infty, -\infty$  .

Ưu điểm lớn nhất của phương pháp biểu diễn đồ thị bằng ma trận kề (hoặc ma trận trọng số) là để trả lời câu hỏi: Hai đỉnh  $u, v$  có kề nhau trên đồ thị hay không, chúng ta chỉ phải thực hiện một phép so sánh. nhược điểm lớn nhất của phương pháp này là: không phụ thuộc vào số cạnh của đồ thị, ta luôn phải sử dụng  $n^2$  đơn vị bộ nhớ để lưu trữ ma trận kề của nó.


## DANH SÁCH CẠNH (CUNG)

Trong trường hợp đồ thị thưa (đồ thị có số cạnh  $m$  thoả mãn bất đẳng thức:  $m \leq 6n$ ) người ta thường dùng cách biểu diễn đồ thị dưới dạng danh sách cạnh.

Trong cách biểu diễn đồ thị bởi danh sách cạnh (cung) chúng ta sẽ lưu trữ danh sách tất cả các cạnh (cung) của đồ thị vô hướng (có hướng). Một cạnh (cung)  $e=(x,y)$  của đồ thị sẽ tương ứng với hai biến  $Dau[e]$ ,  $Cuoi[e]$ . như vậy, để lưu trữ đồ thị ta cần sử dụng  $2m$  đơn vị bộ nhớ. Nhược điểm của cách biểu diễn này là để xác định những đỉnh nào của đồ thị là kề với một đỉnh cho trước chúng ta phải làm cỡ  $m$  phép so sánh (khi duyệt qua danh sách tất cả các cạnh của đồ thị).

*Chú ý:* Trong trường hợp đồ thị có trọng số ta cần thêm  $m$  đơn vị bộ nhớ để lưu trữ trọng số của các cạnh.

**Thí dụ 3.** Danh sách cạnh (cung) của đồ thị  $G$  ( $G_1$ ) cho trong hình 1 là:



Dau	Cuoi
1	2
1	3
1	5
2	3
2	5
3	4
4	5
4	6
5	6

Danh sách cạnh của G

Dau	Cuoi
1	2
1	3
3	2
3	4
5	4
5	6
6	5

Danh sách cung của G1

## DANH SÁCH KÊ

Trong rất nhiều vấn đề ứng dụng của lý thuyết đồ thị, cách biểu diễn đồ thị dưới dạng danh sách kê là cách biểu diễn thích hợp nhất được sử dụng.

Trong cách biểu diễn này, với mỗi đỉnh  $v$  của đồ thị chúng ta lưu trữ danh sách các đỉnh kề với nó, mà ta sẽ ký hiệu là

$$Ke(v) = \{ u \in V : (v, u) \in E \}$$

Khi đó vòng lặp thực hiện với mỗi một phần tử trong danh sách này theo thứ tự các phần tử được sắp xếp trong nó sẽ được viết như sau:

for  $u \in Ke(v)$  do . . .

Chẳng hạn, trên PASCAL có thể mô tả danh sách này như sau (Gọi là cấu trúc **Forward Star**):



Const

$m=1000$ ; ?  $m$ -số cạnh ?

$n=100$ ; ?  $n$ -số đỉnh ?

var

Ke:array[1..m] of integer;

Tro:array[1..n+1] of integer;

Trong đó Tro[i] ghi nhận vị trí bắt đầu của danh sách kề của đỉnh i,  $i=1, 2, \dots, n$ ,  
Tro[n+1]=2m+1.

Khi đó dòng lệnh qui ước

*for*  $u \in Ke(v)$  *do*

begin

.....

end.

Có thể thay thế bởi cấu trúc lệnh cụ thể trên PASCAL như sau

For i:=Tro[v] to Tro[v+1]-1 do

Begin

U:=Ke[i];

.....

End;

Trong rất nhiều thuật toán làm việc với đồ thị chúng ta thường xuyên phải thực hiện các thao tác: Thêm hoặc bớt một số cạnh. Trong trường hợp này cấu trúc dữ liệu dùng ở trên là không thuận tiện. Khi đó nên chuyển sang sử dụng danh sách kề liên kết (Linked Adjacency List) như mô tả trong chương trình nhập danh sách kề của đồ thị từ bàn phím và đưa danh sách đó ra màn hình sau đây:

Program AdjList;

Const

maxV=100;

Type

link=^node;

node=record

v:integer;

next:link;

End;

Var

j,x,y,m,n,u,v:integer;

t:link;

Ke:array[1..Vmax] of link;

Begin

Write('Cho so canh va dinh cua do thi:'); readln(m,n);

(\*Khoi tao\*)

for j:=1 to n do Ke[j]:=nil;

for j:=1 to m do

begin

write('Cho dinh dau va cuoi cua canh ',j,':');

readln(x,y);

new(t); t^.v:=x, t^.next:=Ke[y]; Ke[y]:=t;

new(t); t^.v:=y, t^.next:=Ke[x]; Ke[x]:=t;

```

end;

writeln('Danh sach ke cua cac dinh cua do thi:');

for J:=1 to m do

begin

writeln('Danh sachcac dinh ke cua dinh ' ,j,':');

t:=Ke[j];

while t^.next<>nil do

begin

write(t^.v:4);

t:=t^.next;

end;

end;

readln;

End.

```

**Thí dụ 4.** Danh sách kề của các đồ thị trong hình 1 được mô tả trong hình sau:

Đỉnh đầu



## Danh sách cạnh(cung)

Trong trường hợp đồ thị thưa (đồ thị có số cạnh  $m$  thoả mãn bất đẳng thức:  $m < 6n$ ) người ta thường dùng cách biểu diễn đồ thị dưới dạng danh sách cạnh.

Trong cách biểu diễn đồ thị bởi danh sách cạnh (cung) chúng ta sẽ lưu trữ danh sách tất cả các cạnh (cung) của đồ thị vô hướng (có hướng). Một cạnh (cung)  $e=(x,y)$  của đồ thị sẽ tương ứng với hai biến  $Dau[e]$ ,  $Cuoi[e]$ . như vậy, để lưu trữ đồ thị ta cần sử dụng  $2m$  đơn vị bộ nhớ. Nhược điểm của cách biểu diễn này là để xác định những đỉnh nào của đồ thị là kề với một đỉnh cho trước chúng ta phải làm cỡ  $m$  phép so sánh (khi duyệt qua danh sách tất cả các cạnh của đồ thị).

*Chú ý:* Trong trường hợp đồ thị có trọng số ta cần thêm  $m$  đơn vị bộ nhớ để lưu trữ trọng số của các cạnh.

**Thí dụ 3.** Danh sách cạnh (cung) của đồ thị  $G$  ( $G_1$ ) cho trong hình 1 là:

Dau	Cuoi		Dau	Cuoi
1	2		1	2
1	3		1	3
1	5		3	2
2	3		3	4
2	5		5	4
3	4		5	6
4	5		6	5
4	6			
5	6			

Danh sách cạnh của  $G$

Danh sách cung của  $G_1$

# Danh sách kề

Trong rất nhiều vấn đề ứng dụng của lý thuyết đồ thị, cách biểu diễn đồ thị dưới dạng danh sách kề là cách biểu diễn thích hợp nhất được sử dụng.

Trong cách biểu diễn này, với mỗi đỉnh  $v$  của đồ thị chúng ta lưu trữ danh sách các đỉnh kề với nó, mà ta sẽ ký hiệu là

$$Ke(v) = \{ u \in V : (v, u) \in E \}$$

Khi đó vòng lặp thực hiện với mỗi một phần tử trong danh sách này theo thứ tự các phần tử được sắp xếp trong nó sẽ được viết như sau:

for  $u \in Ke(v)$  do . . .

Chẳng hạn, trên PASCAL có thể mô tả danh sách này như sau (Gọi là cấu trúc **Forward Star**):

Const

$m=1000$ ; ?  $m$ -số cạnh ?

$n=100$ ; ?  $n$ -số đỉnh ?

var

Ke:array[1..m] of integer;

Tro:array[1..n+1] of integer;

Trong đó Tro[i] ghi nhận vị trí bắt đầu của danh sách kề của đỉnh  $i$ ,  $i=1, 2, \dots, n$ , Tro[n+1]=2m+1.

Khi đó dòng lệnh qui ước

for  $u \in Ke(v)$  do

begin

....

end.

Có thể thay thế bởi cấu trúc lệnh cụ thể trên PASCAL như sau

```
For i:=Tro[v] to Tro[v+1]-1 do
```

```
Begin
```

```
U:=Ke[i];
```

```
.....
```

```
End;
```

Trong rất nhiều thuật toán làm việc với đồ thị chúng ta thường xuyên phải thực hiện các thao tác: Thêm hoặc bớt một số cạnh. Trong trường hợp này cấu trúc dữ liệu dùng ở trên là không thuận tiện. Khi đó nên chuyển sang sử dụng danh sách kề liên kết (Linked Adjacency List) như mô tả trong chương trình nhập danh sách kề của đồ thị từ bàn phím và đưa danh sách đó ra màn hình sau đây:

```
Program AdjList;
```

```
Const
```

```
maxV=100;
```

```
Type
```

```
link=^node;
```

```
node=record
```

```
  v:integer;
```

```
  next:link;
```

```
End;
```

```
Var
```

```
  j,x,y,m,n,u,v:integer;
```

```
  t:link;
```

```
  Ke:array[1..Vmax] of link;
```

```

Begin

Write('Cho so canh va dinh cua do thi:'); readln(m,n);

(*Khoi tao*)

for j:=1 to n do Ke[j]:=nil;

for j:=1 to m do

begin

write('Cho dinh dau va cuoi cua canh ' ,j,':');

readln(x,y);

new(t); t^.v:=x, t^.next:=Ke[y]; Ke[y]:=t;

new(t); t^.v:=y, t^.next:=Ke[x]; Ke[x]:=t;

end;

writeln('Danh sach ke cua cac dinh cua do thi:');

for J:=1 to m do

begin

writeln('Danh sachcac dinh ke cua dinh ' ,j,':');

t:=Ke[j];

while t^.next<>nil do

begin

write(t^.v:4);

t:=t^.next;

end;

end;

```





# Các thuật toán tìm kiếm trên đồ thị và ứng dụng

Rất nhiều thuật toán trên đồ thị được xây dựng trên cơ sở duyệt tất cả các đỉnh của đồ thị sao cho mỗi đỉnh của nó được viếng thăm đúng một lần. Vì vậy, việc xây dựng những thuật toán cho phép duyệt một cách hệ thống tất cả các đỉnh của đồ thị là một vấn đề quan trọng thu hút sự quan tâm nghiên cứu của nhiều tác giả. Những thuật toán như vậy chúng ta sẽ gọi là thuật toán tìm kiếm trên đồ thị. Trong mục này chúng ta sẽ giới thiệu hai thuật toán tìm kiếm cơ bản trên đồ thị: *Thuật toán tìm kiếm theo chiều sâu (Depth First Search)* và *Thuật toán tìm kiếm theo chiều rộng (Breadth First Search)* và ứng dụng của chúng vào việc giải một số bài toán trên đồ thị.

Trong mục này chúng ta sẽ xét đồ thị vô hướng  $G=(V,E)$ , với đỉnh  $n$  và  $m$  cạnh.

Chúng ta sẽ quan tâm đến việc đánh giá hiệu quả của các thuật toán trên đồ thị, mà một trong những đặc trưng quan trọng nhất là độ phức tạp tính toán, tức là số phép toán mà thuật toán cần phải thực hiện trong tình huống xấu nhất được biểu diễn như hàm của kích thước đầu vào của bài toán. Trong các thuật toán trên đồ thị, đầu vào là đồ thị  $G=(V,E)$ , vì vậy, kích thước của bài toán là số đỉnh  $n$  và số cạnh  $m$  của đồ thị. Khi đó độ phức tạp tính toán của thuật toán sẽ được biểu diễn như là hàm của hai biến số  $f(n,m)$  là số phép toán nhiều nhất cần phải thực hiện theo thuật toán đối với mọi đồ thị  $n$  đỉnh và  $m$  cạnh. Khi so sánh tốc độ tăng của hai hàm nhận giá trị không âm  $f(n)$  và  $g(n)$  chúng ta sẽ sử dụng ký hiệu sau:

$$f(n)=O(g(n))$$

$\Leftrightarrow$  tìm được các hằng số  $C, N \geq 0$  sao cho

$$f(n) \leq C g(n) \text{ với mọi } n \geq N.$$

Tương tự như vậy nếu  $f(n_1, n_2, \dots, n_k), g(n_1, n_2, \dots, n_k)$  là các hàm nhiều biến ta viết

$$f(n_1, n_2, \dots, n_k) = O(g(n_1, n_2, \dots, n_k))$$

$\Leftrightarrow$  tìm được các hằng số  $C, N > 0$  sao cho

$$f(n_1, n_2, \dots, n_k) \leq C g(n_1, n_2, \dots, n_k) \text{ với mọi } n_1, n_2, \dots, n_k \geq N.$$

Nếu độ phức tạp tính toán của thuật toán là  $O(g(n))$  thì ta sẽ còn nói là nó đòi hỏi thời gian tính cỡ  $O(g(n))$ .

## TÌM KIẾM THEO CHIỀU SÂU TRÊN ĐỒ THỊ

Ý tưởng chính của thuật toán có thể trình bày như sau. Ta sẽ bắt đầu tìm kiếm từ một đỉnh  $v_0$  nào đó của đồ thị. Sau đó chọn  $u$  là một đỉnh tùy ý kề với  $v_0$  và lặp lại quá trình đối với  $u$ . Ở bước tổng quát, giả sử ta đang xét đỉnh  $v$ . Nếu như trong số các đỉnh kề với  $v$  tìm được đỉnh  $w$  là chưa được xét thì ta sẽ xét đỉnh này (nó sẽ trở thành đã xét) và bắt đầu từ nó ta sẽ bắt đầu quá trình tìm kiếm còn nếu như không còn đỉnh nào kề với  $v$  là chưa xét thì ta nói rằng đỉnh này đã duyệt xong và quay trở lại tiếp tục tìm kiếm từ đỉnh mà trước đó ta đến được đỉnh  $v$  (nếu  $v=v_0$ , thì kết thúc tìm kiếm). Có thể nói nôm na là tìm kiếm theo chiều sâu bắt đầu từ đỉnh  $v$  được thực hiện trên cơ sở tìm kiếm theo chiều sâu từ tất cả các đỉnh chưa xét kề với  $v$ . Quá trình này có thể mô tả bởi thủ tục đệ qui sau đây:

Procedure DFS( $v$ );

(\*tìm kiếm theo chiều sâu bắt đầu từ đỉnh  $v$ ; các biến Chuaxet, Ke là biến toàn cục\*)

Begin

Tham\_dinh( $v$ );

Chuaxet[ $v$ ]:=false;

For  $u \in Ke(v)$  do

If Chuaxet[ $u$ ] then DFS( $u$ );

End; (\*đỉnh  $v$  đã duyệt xong\*)

Khi đó, tìm kiếm theo chiều sâu trên đồ thị được thực hiện nhờ thuật toán sau:

Begin

(\*Initialization\*)

for  $v \in V$  do Chuaxet[ $v$ ]:=true;

for  $v \in V$  do

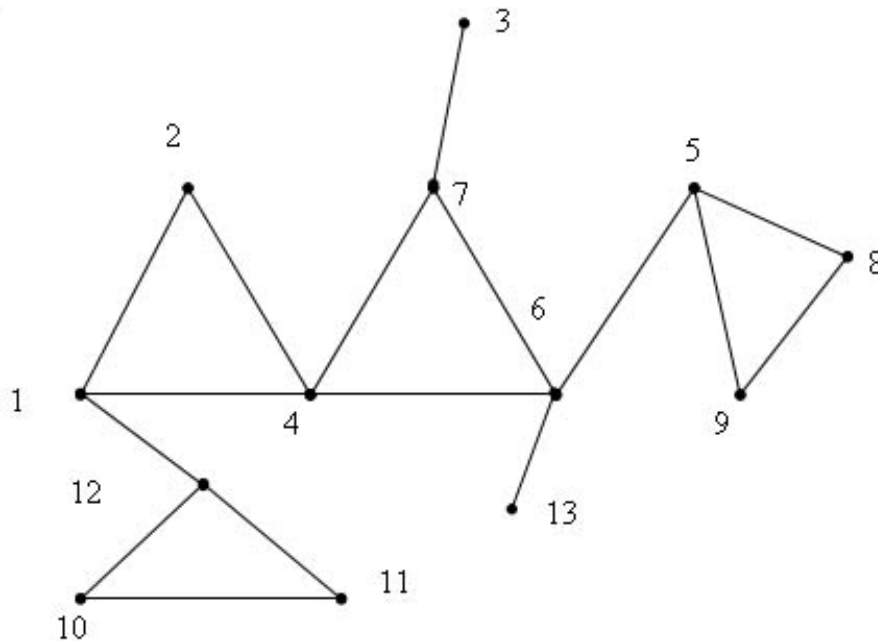
if Chuaxet[v] then DFS(v);

End.

Rõ ràng lệnh gọi SFS(v) sẽ cho phép đến thăm tất cả các đỉnh thuộc cùng thành phần liên thông với đỉnh v, bởi vì sau khi thăm đỉnh là lệnh gọi đến thủ tục DFS đối với tất cả các đỉnh kề với nó. Mặt khác, do mỗi khi thăm đỉnh v xong, biến Chuaxet[v] được đặt lại giá trị false nên mỗi đỉnh sẽ được thăm đúng một lần. Thuật toán lần lượt sẽ tiến hành tìm kiếm từ các đỉnh chưa được thăm, vì vậy, nó sẽ xét qua tất cả các đỉnh của đồ thị (không nhất thiết phải là liên thông).

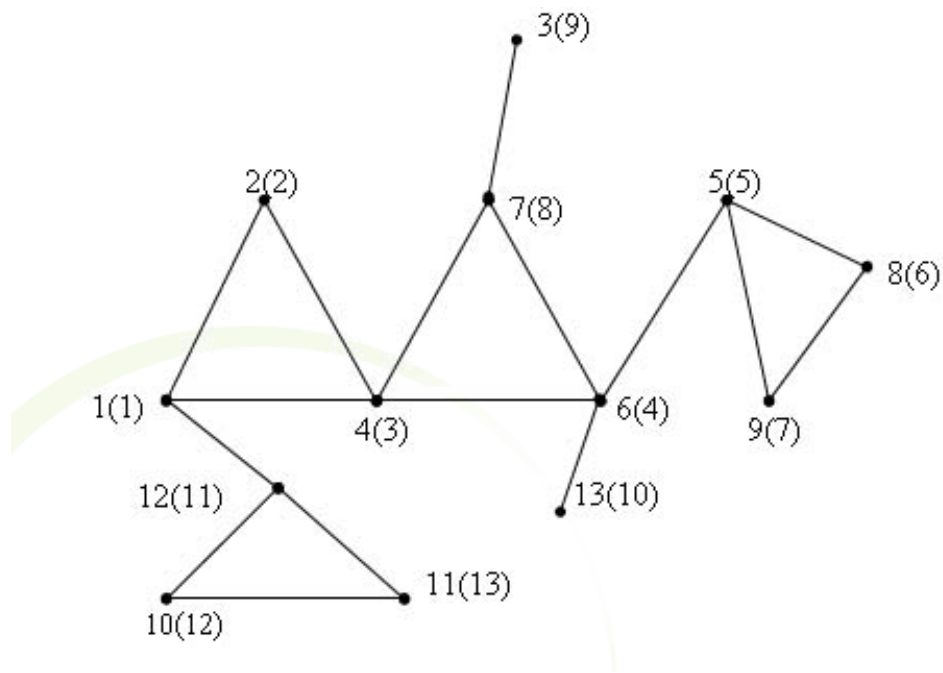
Để đánh giá độ phức tạp tính toán của thủ tục, trước hết nhận thấy rằng số phép toán cần thực hiện trong hai chu trình của thuật toán (hai vòng for ở chương trình chính) là cỡ n. Thủ tục DFS phải thực hiện không quá n lần. Tổng số phép toán cần phải thực hiện trong các thủ tục này là  $O(n+m)$ , do trong các thủ tục này ta phải xét qua tất cả các cạnh và các đỉnh của đồ thị. Vậy độ phức tạp tính toán của thuật toán là  $O(n+m)$ .

**Thí dụ 1.** Xét đồ thị cho trong hình 1 gồm 13 đỉnh, các đỉnh được đánh số từ 1 đến 13 như sau:



Hình 1

Khi đó các đỉnh của đồ thị được đánh số lại theo thứ tự chúng được thăm theo thủ tục tìm kiếm theo chiều sâu mô tả ở trên như hình 2. Giả thiết rằng các đỉnh trong danh sách kề của đỉnh v ( $Ke(v)$ ) được sắp xếp theo thứ tự tăng dần của chỉ số.



Hình 2. Chỉ số mới (trong ngoặc) của các đỉnh được đánh lại theo thứ tự chúng được thăm trong thuật toán tìm kiếm theo chiều sâu.

Thuật toán tìm kiếm theo chiều sâu trên đồ thị vô hướng trình bày ở trên dễ dàng có thể mô tả lại cho đồ thị có hướng. Trong trường hợp đồ thị có hướng, thủ tục DFS(v) sẽ cho phép thăm tất cả các đỉnh u nào mà từ v có đường đi đến u. Độ phức tạp tính toán của thuật toán là  $O(n+m)$ .

# Tìm kiếm theo chiều rộng trên đồ thị

Đề ý rằng trong thuật toán tìm kiếm theo chiều sâu đỉnh được thăm càng muộn sẽ càng sớm trở thành đã duyệt xong. Điều đó là hệ quả tất yếu của việc các đỉnh được thăm sẽ được kết nạp vào trong ngăn xếp (STACK). Tìm kiếm theo chiều rộng trên đồ thị, nếu nói một cách ngắn gọn, được xây dựng trên cơ sở thay thế ngăn xếp (STACK) bởi hàng đợi (QUEUE). Với sự cải biên như vậy, đỉnh được thăm càng sớm sẽ càng sớm trở thành đã duyệt xong (tức là càng sớm rời khỏi hàng đợi). Một đỉnh sẽ trở thành đã duyệt xong ngay sau khi ta xét xong tất cả các đỉnh kề (chưa được thăm) với nó. Thủ tục có thể mô tả như sau:

Procedure BFS(v);

(\*Tìm kiếm theo chiều rộng bắt đầu từ đỉnh v, các biến Chuaxet, Ke là biến cục bộ\*)

begin

$QUEUE := \emptyset;$

$QUEUE \leftarrow v; (*kết qua nạp vào QUEUE*)$

$Chuaxet[v] := false;$

While  $QUEUE \neq \emptyset$  do

Begin

$p \leftarrow QUEUE;$  (\*lấy p từ QUEUE.\*)

Tham\_dinh(p);

For  $u \in Ke(v)$  do

If  $Chuaxet[u]$  then

Begin

$QUEUE \leftarrow u;$

$Chuaxet[u] := false;$

End;

End;

end;

Khi đó, tìm kiếm theo chiều rộng trên đồ thị được thực hiện nhờ thuật toán sau:

Begin

(\*Initialization\*)

for  $f \in V$  do Chuaxet[v] := true;

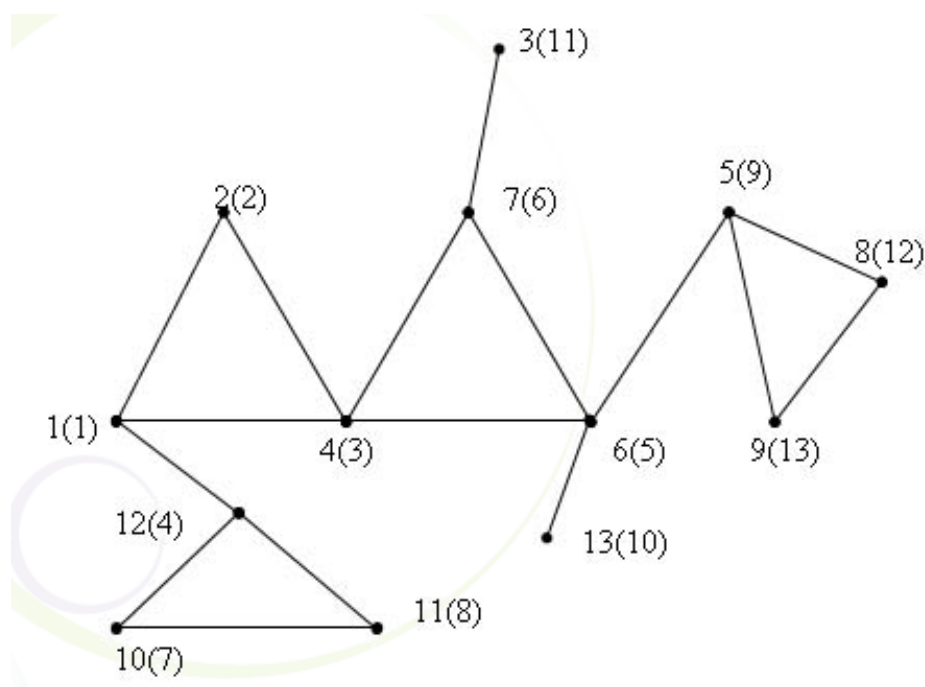
for  $v \in V$  do

if Chuaxet[v] then BFS(v);

End.

Lập luận tương tự như trong thủ tục tìm kiếm theo chiều sâu, có thể chỉ ra được rằng lệnh gọi BFS(v) sẽ cho phép thăm đến tất cả các đỉnh thuộc cùng thành phần liên thông với đỉnh v, và mỗi đỉnh của đồ thị sẽ được thăm đúng một lần. Độ phức tạp tính toán của thuật toán là  $O(m+n)$ .

**Thí dụ 2.** Xét đồ thị xét trong hình 1. Thứ tự thăm đỉnh của đồ thị theo thuật toán tìm kiếm theo chiều rộng được ghi trong ngoặc.



Hình3. Chỉ số mới (trong ngoặc) của các đỉnh được đánh lại theo thứ tự chúng được thăm trong thuật toán tìm kiếm theo chiều sâu.



# Tìm đường đi và kiểm tra tính liên thông

Trong mục này ta xét ứng dụng các thuật toán tìm kiếm mô tả trong các mục trước vào việc giải bài toán cơ bản trên đồ thị: bài toán về tìm đường đi và bài toán về xác định tính liên thông của đồ thị.<sup>7</sup>

## Bài toán tìm đường đi giữa hai đỉnh:

Giả sử  $s$  và  $t$  là hai đỉnh nào đó của đồ thị. Hãy tìm đường đi từ  $s$  đến  $t$ . Như trên đã phân tích, thủ tục DFS( $s$ ) (BS( $s$ )) sẽ cho thăm tất cả các đỉnh thuộc cùng một thành phần liên thông với  $s$ . vì vậy, sau khi thực hiện xong thủ tục, nếu Chuaxet[ $t$ ]=true, thì điều đó có nghĩa là không có đường đi từ  $s$  đến  $t$ , còn nếu Chuaxet[ $t$ ]=false thì  $t$  thuộc cùng thành phần liên thông với  $s$ , hay nói một cách khác: tồn tại đường đi từ  $s$  đến  $t$ . Trong trường hợp tồn tại đường đi, để ghi nhận đường đi, ta dùng thêm biểu thức Truoc[ $v$ ] để ghi nhận đỉnh đi trước đỉnh  $v$  trong đường đi tìm kiếm từ  $s$  đến  $v$ . Khi đó, đối với thủ tục DFS( $v$ ) cần sửa đổi câu lệnh if trong nó như sau:

If Chuaxet[ $u$ ] then

Begin

Truoc[ $u$ ]:=v;

DFS( $u$ );

End;

Còn đối với thủ tục BFS( $v$ ) cần sửa đổi câu lệnh if trong nó như sau:

*If Chuaxet [ $u$ ] then*

Begin

*QUEUE  $\leftarrow u$ ;*

Chuaxet[ $u$ ]:=false;

Truoc[ $u$ ]:=p;

End;

*Chú ý:* Đường đi tìm được theo thuật toán tìm kiếm theo chiều rộng là đường đi ngắn nhất (theo số cạnh) từ s đến t. Điều này suy trực tiếp từ thứ tự thăm đỉnh theo thuật toán tìm kiếm theo chiều rộng.

### **Tìm các thành phần liên thông của đồ thị:**

Hãy cho biết đồ thị gồm bao nhiêu thành phần liên thông và từng thành phần liên thông của nó là gồm những đỉnh nào.

Do thủ tục DFS(v) (BFS(s)) cho phép thăm tất cả các đỉnh thuộc cùng một thành phần liên thông với s, nên số thành phần liên thông của đồ thị bằng số lần gọi đến thủ tục này. Vấn đề còn lại là cách ghi nhận các đỉnh trong từng thành phần liên thông. Ta dùng thêm biến Index[v] để ghi nhận chỉ số của thành phần liên thông chứa đỉnh v, và dùng thêm biến Inconnect để đếm số thành phần liên thông (biến này cần khởi tạo giá trị 0). Thủ tục Tham\_dinh(v) trong các thủ tục DFS(v) và BFS(v) có nhiệm vụ gán: Index[v]:=connect, còn câu lệnh if trong các chương trình chính gọi đến các thủ tục này cần được sửa lại như sau:

Inconnect:=0;

If Chuaxet[v] then

Begin

Inconnect:=Inconnect+1;

DFS(v); (\*BFS(v)\*)

End;

Kết thúc vòng lặp thứ hai trong chương trình chính, Inconnect cho số thành phần liên thông của đồ thị, còn mảng Index[v],  $v \in V$  cho phép liệt kê các đỉnh thuộc cùng một thành phần liên thông.

Chương trình PASCAL giải bài toán trên có thể viết như sau:

? CHUONG TRINH TIM DUONG DI VA KIEM TRA TINH LIEN THONG THEO  
CAC THUAT TOAN TIM KIEM TREN DO THI?

uses crt;

var

a:array[1..20,1..20] of byte;

*QUEUE, Chuaxet, Truoc: array[1..20] of byte;*

i,j,n,solt,k,s,t: integer;

Stop: boolean;

Ch: char;

Procedure Nhapsolieu;

Begin

Write('Cho so dinh cua do thi:'); readln(n);

Writeln('Nhap so lieu ma tran ke:');

For i:= 1 to n do

Begin

For j:= i+1 to n do

Begin

Write('a[' ,i , ' , ' ,j , ']='); readln(a[i,j]);

End;

a[i,j]:=0; writeln;

End;

End;

{=====}

Procedure readfile;

Var f:text; fn:string;

Begin

Write(' Cho ten file du lieu:'); readln (fn);

Assign(fnfn); reset(f); readln(f,n);

Writeln('Nhap so lieu ma tran ke:');

For i:= 1 to n do

For j:=1 to n do read(f, a[i,j]);

Close(f);

End;

{=====}

Procedure Insolieu;

Begin

Writeln('Ma tran ke:');

For i:= 1 to n do

Begin

For j:=1 to n do write(a[i,j]:3);

Writeln;

End;

End;

{=====}

Procedure Ketqualienthon;

Begin

Insolieu;

If solt=1 then writeln('Do thi la lien thong')

Else

Begin

Writeln('Thanh phan lien thon thu ',i,' gom cac dinh:');

For j:=1 to n do if Chuaxet[j]=i then write(j:3); writeln;

End;

Write('Go Enter de tiep tuc...'#7); readln;

End;

{=====}

Procedure BFS(i:integer);

(\*tim kiem theo chieu rong bat dau tu dinh i\*);

var u, dauQ, CuoiQ,: integer;

begin

dauQ:=1; cuoiQ:=1;

QUEUE[cuoiQ]:=i; Chuaxet[i]:=Solt;

While dauQ<=cuoiQ do

Begin

U:= QUEUE[sauQ]; dauQ:=dauQ+1;

For j:=1 to n do

If a[u,j]=1) and (Chuaxet[j]=0) then

Begin

cuoiQ:=cuoiQ+1; QUEUE:[cuoiQ]:=j;

Chuaxet[j]:=Solt; Truoc[j]:=u;

End;

End;

*End; ? of procedure BFS ?*

{=====}

Procedure DFS(v:integer);

(\*Tim kiem theo chieu sau bat dau tu dinh v\*);

var U: integer;

begin

Chuaxet[v]:=solt;

For u:=1 to n do

If (a[v,u]=1) and (Chuaxet[u]=0) then

Begin

Truoc[u]:=v;

DFS9(u);

End;

End;

{=====}

Procedure Lienthong;

Begin

*? Khoi toa so lieu ?*

for j:=1 to n do Chuaxet[j]:=0;

solt:=0;

for i:=1 to n do

if Chuaxet[i]=0 then

begin

solt:=solt+1;

? *BFS(i)*; ? *DFS(i)*;

end;

Ketqualienthong;

End;

{=====}

Procedure Ketquaduongdi;

Begin

If Truoc[t]=0 then writeln('Khong co duong di tu ', s, ' den ', t)

Else

Begin

Writeln('Duong di tu ', s, ' den ', t, ' la:');

J:=t;

Write(t, '<==');

While Truoc[j]<>s do

Begin

Write(Truoc[g], '<==');

J:=Truoc[j];

End;

Writeln(s);

End;

Write('Go Enter de tiep tục...'#7); readln;

End;

{=====}

Procedure duongdi;

Begin

Insolieu;

Write('Tìm duon đi tu dinh:'); readln(s);

Write(' den dinh:'); readln(t);

*For j:=1 to n do ? Khoi tao so lieu ?*

Begin

Truoc[j]:=0;

Chuaxet[j]:=0;

End;

*Silt:=1; BFS(s); ? DFS(s); ?*

Ketquaduondi;

End;

{=====}

Procedure menu;

Begin

Clrscr;

Writeln('TIM DUONG DI VA KIEM TRA TINH LIEN THONG');



```

Writeln('CUA DO THIJ THEO THUAT TOAN TIM KIEM TREN DO THI');

Writeln('=====');

Writeln(' 1. Nhap so lieu tu ban phim');

Writeln(' 2. Nhap so lieu tu file');

Writeln(' 3. Kiem tra tinh lien thong');

Writeln(' 4. Tim duong di giua hai dinh');

Writeln(' 5. Thoat');

Writeln('-----');

Write('Hay go phim so de chon chuc nang...#7);

Ch:=readkey;

Writeln(ch);

End;

{=====}

? Main program ?

Begin

repeat

menu;

case ch of

'1':Nhapsolieu;

'2':Readfile;

'3':Lienthong;

'4':Duongdi;

```

until (ch='5') or (upcase (ch)='Q');

End.

# Đồ thị Euler và đồ thị Hamiton

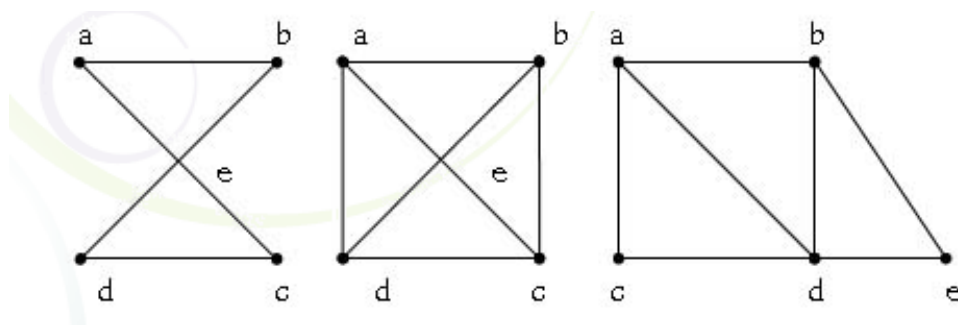
Trong chương này chúng ta sẽ nghiên cứu hai dạng đồ thị đặc biệt là đồ thị Euler và đồ thị Hamilton. Dưới đây, nếu không có giải thích bổ sung, thuật ngữ đồ thị được dùng để chỉ chung đa đồ thị vô hướng và có hướng, và thuật ngữ cạnh sẽ dùng để chỉ chung cạnh của đồ thị vô hướng cũng như cung của đồ thị có hướng.

## ĐỒ THỊ EULER

**Định nghĩa 1.** Chu trình đơn trong đồ thị  $G$  đi qua mỗi cạnh của nó một lần được gọi là chu trình Euler. Đường đi đơn trong  $G$  đi qua mỗi cạnh của nó một lần được gọi là đường đi Euler. Đồ thị được gọi là đồ thị Euler nếu nó có chu trình Euler, và gọi là đồ thị nửa Euler nếu nó có đường đi Euler.

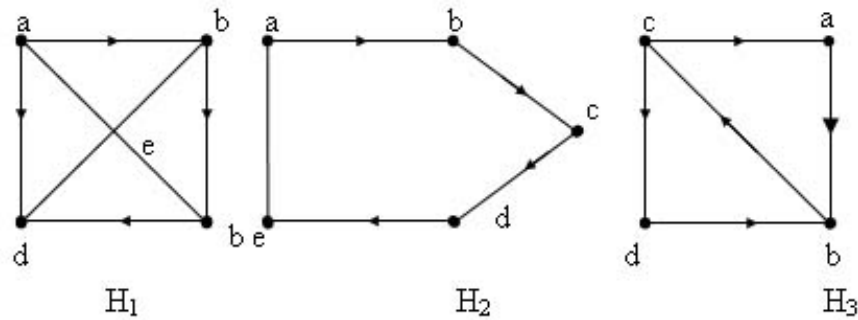
Rõ ràng mọi đồ thị Euler luôn là nửa Euler, nhưng điều ngược lại không luôn đúng.

**Thí dụ 1.** Đồ thị  $G_1$  trong hình 1 là đồ thị Euler vì nó có chu trình Euler  $a, e, c, d, e, b, a$ . Đồ thị  $G_3$  không có chu trình Euler nhưng nó có đường đi Euler  $a, c, d, e, b, d, a, b$ , vì thế  $G_3$  là đồ thị nửa Euler. Đồ thị  $G_2$  không có chu trình cũng như đường đi Euler.



Hình 1. Đồ thị  $G_1$ ,  $G_2$ ,  $G_3$

**Thí dụ 2.** Đồ thị  $H_2$  trong hình 2 là đồ thị Euler vì nó có chu trình Euler  $a, b, c, d, e, a$ . Đồ thị  $H_3$  không có chu trình Euler nhưng nó có đường đi Euler  $c, a, b, c, d, b$  vì thế  $H_3$  là đồ thị nửa Euler. Đồ thị  $H_1$  không có chu trình cũng như đường đi Euler.



Hình 2. Đồ thị  $H_1$ ,  $H_2$ ,  $H_3$

Điều kiện cần và đủ để một đồ thị là một đồ thị Euler được Euler tìm ra vào năm 1736 khi ông giải quyết bài toán học búa nổi tiếng thế giới thời đó về bảy cái cầu ở thành phố Königsberg và đây là định lý đầu tiên của lý thuyết đồ thị.

**Định lý 1 (Euler).** Đồ thị vô hướng liên thông  $G$  là đồ thị Euler khi và chỉ khi mọi đỉnh của  $G$  đều có bậc chẵn.

Để chứng minh định lý trước hết ta chứng minh bổ đề:

**Bổ đề.** Nếu bậc của mỗi đỉnh của đồ thị  $G$  không nhỏ hơn 2 thì  $G$  chứa chu trình.

**Chứng minh.**

Nếu  $G$  có cạnh lặp thì khẳng định của bổ đề là hiển nhiên. Vì vậy giả sử  $G$  là đơn đồ thị. Gọi  $v$  là một đỉnh nào đó của  $G$ . Ta sẽ xây dựng theo qui nạp đường đi

$$v \rightarrow v_1 \rightarrow v_2 \rightarrow \dots$$

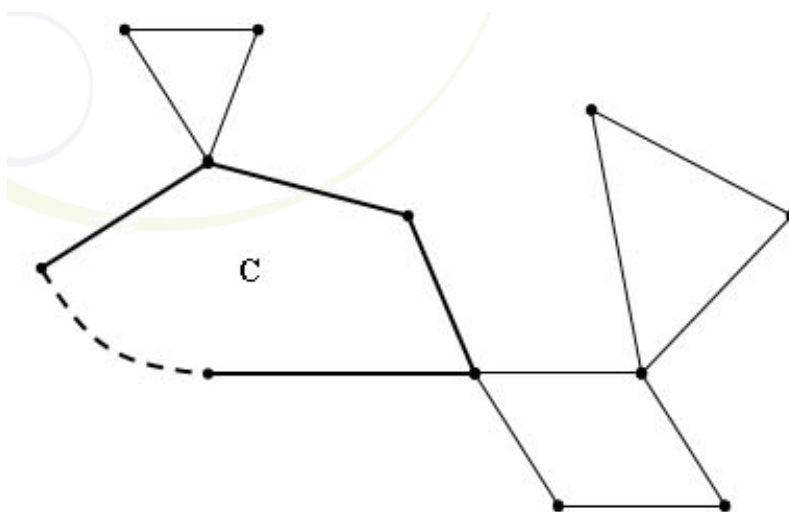
trong đó  $v_1$  là đỉnh kề với  $v$ , còn với  $i \geq 1$  chọn  $v_{i+1} \neq v_{i-1}$  (có thể chọn  $v_{i+1}$  như vậy là vì  $\deg(v_i) \geq 2$ ). Do tập đỉnh của  $G$  là hữu hạn, nên sau một số hữu hạn bước ta phải quay lại một đỉnh đã xuất hiện trước đó. Gọi đỉnh đầu tiên như thế là  $v_k$ . Khi đó, đoạn của đường đi xây dựng nằm giữa hai đỉnh  $v_k$  là 1 chu trình cần tìm.

**Chứng minh định lý:**

**Cần.** Giả sử  $G$  là đồ thị Euler tức là tồn tại chu trình Euler  $P$  trong  $G$ . Khi đó cứ mỗi lần chu trình  $P$  đi qua một đỉnh nào đó của  $G$  bậc của đỉnh đó tăng lên 2. Mặt khác mỗi cạnh của đồ thị xuất hiện trong  $P$  đúng một lần, suy ra mỗi đỉnh của đồ thị đều có bậc chẵn.

**Đủ.** Quy nạp theo số đỉnh và số cạnh của  $G$ . Do  $G$  liên thông và  $\deg(v)$  là số chẵn nên bậc của mỗi đỉnh của nó không nhỏ hơn 2. Từ đó theo bổ đề  $G$  phải chứa chu trình  $C$ . Nếu  $C$  đi qua tất cả các cạnh của  $G$  thì nó chính là chu trình Euler. Giả sử  $C$  không đi

qua tất cả các cạnh của  $G$ . Khi đó loại bỏ khỏi  $G$  tất cả các cạnh thuộc  $C$  ta thu được một đồ thị mới  $H$  vẫn có bậc là chẵn. Theo giả thiết qui nạp, trong mỗi thành phần liên thông của  $H$  đều tìm được chu trình Euler. Do  $G$  là liên thông nên trong mỗi thành phần của  $H$  có ít nhất một đỉnh chung với chu trình  $C$ . Vì vậy, ta có thể xây dựng chu trình Euler trong  $G$  như sau: bắt đầu từ một đỉnh nào đó của chu trình  $C$ , đi theo các cạnh của  $C$  chừng nào chưa gặp phải đỉnh không cô lập của  $H$ . Nếu gặp phải đỉnh như vậy ta sẽ đi theo chu trình Euler của thành phần liên thông của  $H$  chứa đỉnh đó. Sau đó lại tiếp tục đi theo cạnh của  $C$  cho đến khi gặp phải đỉnh không cô lập của  $H$  thì lại theo chu trình Euler của thành phần liên thông tương ứng trong  $H$ .v... (xem hình 3). Quá trình sẽ kết thúc khi ta trở về đỉnh xuất phát, tức là thu được chu trình đi qua mỗi cạnh của đồ thị đúng một lần.



Hình 3. Minh hoạ cho chứng minh định lý 1.

**Hệ quả 2.** Đồ thị vô hướng liên thông  $G$  là nửa Euler khi và chỉ khi nó có không quá 2 đỉnh bậc lẻ.

**Chứng minh.** Thực vậy, nếu  $G$  có không quá 2 đỉnh bậc lẻ thì số đỉnh bậc lẻ của nó chỉ có thể là 0 hoặc 2. Nếu  $G$  không có đỉnh bậc lẻ thì theo định lý 1, nó là đồ thị Euler. Giả sử  $G$  có 2 đỉnh bậc lẻ là  $u$  và  $v$ . Gọi  $H$  là đồ thị thu được từ  $G$  bằng cách thêm vào  $G$  một đỉnh mới  $w$  và hai cạnh  $(w,u)$  và  $(w,v)$ . Khi đó tất cả các đỉnh của  $H$  đều có bậc chẵn, vì thế theo định lý 1, nó có chu trình Euler  $C$ . Xoá bỏ khỏi chu trình này đỉnh  $w$  và hai cạnh kề nó ta thu được đường đi Euler trong đồ thị  $G$ .

Giả sử  $G$  là đồ thị Euler, từ chứng minh định lý ta có thủ tục sau để tìm chu trình Euler trong  $G$ .

Procedure Euler\_Cycle;

Begin

$STACK := \emptyset; CE := \emptyset;$

Chọn  $u$  là một đỉnh nào đó của đồ thị;

$STACK \leftarrow u;$

*While*  $STACK \neq \emptyset$  *do*

Begin

$X := \text{top}(STACK);$  (\*  $x$  là phần tử đầu  $STACK$  \*)

*If*  $Ke(x) \neq \emptyset$  *then*

Begin

$Y :=$  đỉnh đầu tiên trong danh sách  $Ke(x);$

$STACK \leftarrow y;$

(\* loại bỏ cạnh  $(x,y)$  khỏi đồ thị \*)

$Ke(x) := Ke(x) \setminus \{y\};$

$Ke(y) := Ke(y) \setminus \{x\};$

End

Else

Begin

$x \leftarrow STACK; CE \leftarrow x;$

End;

End;

End;

Giả sử  $G$  là đồ thị Euler, thuật toán đơn giản sau đây cho phép xác định chu trình Euler khi làm bằng tay.

## Thuật toán Flor

Xuất phát từ một đỉnh  $u$  nào đó của  $G$  ta đi theo các cạnh của nó một cách tùy ý chỉ cần tuân thủ 2 qui tắc sau:

- (1) Xoá bỏ cạnh đã đi qua đồng thời xoá bỏ cả những đỉnh cô lập tạo thành.
- (2) Ở mỗi bước ta chỉ đi qua cầu khi không còn cách lựa chọn nào khác.

### Chứng minh tính đúng đắn của thuật toán.

Trước tiên ta chỉ ra rằng thủ tục trên có thể thực hiện ở mỗi bước. Giả sử ta đi đến một đỉnh  $v$  nào đó, khi đó nếu  $v \neq u$  thì đồ thị còn lại  $H$  là liên thông và chứa đúng hai đỉnh bậc lẻ là  $v$  và  $u$ . Theo hệ quả trong  $H$  có đường đi Euler  $P$  từ  $v$  tới  $u$ . Do việc xoá bỏ cạnh đầu tiên của đường đi  $P$  không làm mất tính liên thông của  $H$ , từ đó suy ra thủ tục có thể thực hiện ở mỗi bước. Nếu  $v = u$  thì lập luận ở trên sẽ vẫn đúng chừng nào vẫn còn cạnh kề với  $u$ .

Như vậy chỉ còn phải chỉ ra thủ tục trên dẫn đến đường đi Euler. Thực vậy trong  $G$  không thể còn cạnh chưa đi qua khi mà ta sử dụng cạnh cuối cùng kề với  $u$  (trong trường hợp ngược lại, việc loại bỏ một cạnh nào đó kề với một trong số những cạnh còn lại chưa đi qua sẽ dẫn đến một đồ thị không liên thông, và điều đó là mâu thuẫn với giả thiết ii).

Chứng minh tương tự như trong định lý 1 ta thu được kết quả sau đây cho đồ thị có hướng.

**Định lý 2.** Đồ thị có hướng liên thông mạnh là đồ thị Euler khi và chỉ khi

$$\text{Deg}^+(v) = \text{deg}^-(v), \forall v \in V.$$

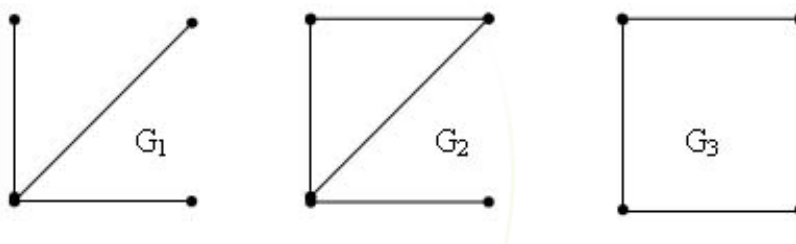
# Đồ thị Hamilton

Trong mục này chúng ta xét bài toán tương tự như trong mục trước chỉ khác là ta quan tâm đến đường đi qua tất cả các đỉnh của đồ thị, mỗi đỉnh đúng một lần. Sự thay đổi tưởng chừng như là không đáng kể này trên thực tế đã dẫn đến sự phức tạp hoá vấn đề cần giải quyết.

**Định nghĩa 2.** Đường đi qua tất cả các đỉnh của đồ thị mỗi đỉnh đúng một lần được gọi là đường đi Hamilton. Chu trình bắt đầu từ một đỉnh  $v$  nào đó qua tất cả các đỉnh còn lại mỗi đỉnh đúng một lần rồi quay trở về  $v$  được gọi là chu trình Hamilton. Đồ thị  $G$  được gọi là đồ thị Hamilton nếu nó chứa chu trình Hamilton và gọi là đồ thị nửa Hamilton nếu nó có đường đi Hamilton.

Rõ ràng đồ thị Hamilton là nửa Hamilton, nhưng điều ngược lại không còn đúng.

**Thí dụ 3.** Trong hình 4:  $G_3$  là Hamilton,  $G_2$  là nửa Hamilton còn  $G_1$  không là nửa Hamilton.



Hình 4. Đồ thị Hamilton  $G_3$ , nửa Hamilton  $G_2$ , và  $G_1$ .

Cho đến nay việc tìm một tiêu chuẩn nhận biết đồ thị Hamilton vẫn còn là mở, mặc dù đây là một vấn đề trung tâm của lý thuyết đồ thị. Hơn thế nữa, cho đến nay cũng chưa có thuật toán hiệu quả để kiểm tra một đồ thị có là Hamilton hay không. Các kết quả thu được phần lớn là điều kiện đủ để một đồ thị là đồ thị Hamilton. Phần lớn chúng điều có dạng "nếu  $G$  có số cạnh đủ lớn thì  $G$  là Hamilton". Một kết quả như vậy được phát biểu trong định lý sau đây.

**Định lý 3 (Dirak 1952).** Đơn đồ thị vô hướng  $G$  với  $n > 2$  đỉnh, mỗi đỉnh có bậc không nhỏ hơn  $n/2$  là đồ thị Hamilton.

Chứng minh:

Thêm vào đồ thị  $G$   $k$  đỉnh mới và nối chúng với tất cả các đỉnh của  $G$ . giả sử  $k$  là số nhỏ nhất các đỉnh cần thêm vào để cho đồ thị thu được  $G'$  là đồ thị Hamilton. Ta sẽ chỉ



ra rằng  $k=0$ . Thực vậy, giả sử ngược lại là  $k > 0$ . Ký hiệu  $v, p, w, \dots$ ,  $v$  là chu trình Hamilton trong  $G'$ , trong đó  $v, w$  là đỉnh của  $G$  còn  $p$  là một trong số các đỉnh mới. Khi đó  $w$  không kề với  $v$  vì nếu ngược lại, ta không cần sử dụng  $p$  và điều đó là mâu thuẫn với giả thiết  $k$  nhỏ nhất. Hơn thế nữa đỉnh ( $w'$  chẳng hạn) kề với  $w$  không thể đi liền sau đỉnh  $v'$  (kề với  $v$ ) vì rằng khi đó có thể thay

$$v \rightarrow p \rightarrow w \rightarrow \dots \rightarrow v' \rightarrow w' \rightarrow \dots \rightarrow v$$

$$\text{bởi } v \rightarrow v' \rightarrow \dots \rightarrow w \rightarrow w' \rightarrow \dots \rightarrow v$$

bằng cách đảo ngược đoạn của chu trình nằm giữa  $w$  và  $v'$ . Từ đó suy ra là số đỉnh của đồ thị  $G'$  không kề với  $w$  là không nhỏ hơn số đỉnh kề với  $v$  (tức là ít nhất cũng là bằng  $n/2+k$ ), đồng thời số đỉnh của  $G'$  kề với  $w$  ít ra là phải bằng  $n/2+k$ . Do không có đỉnh nào của  $G'$  vừa không kề, lại vừa kề với  $w$ , cho nên tổng số đỉnh của đồ thị  $G'$  ( $G'$  có  $n+k$  đỉnh) không ít hơn  $n+2k$ . Mâu thuẫn thu được đã chứng minh định lý.

Định lý sau là tổng quát hoá của định lý Dirak cho đồ thị có hướng:

#### **Định lý 4.**

Giả sử  $G$  là đồ có hướng liên thông với  $n$  đỉnh. Nếu

$$\deg^+(v) \geq n/2, \deg^-(v) \geq n/2, \forall v$$

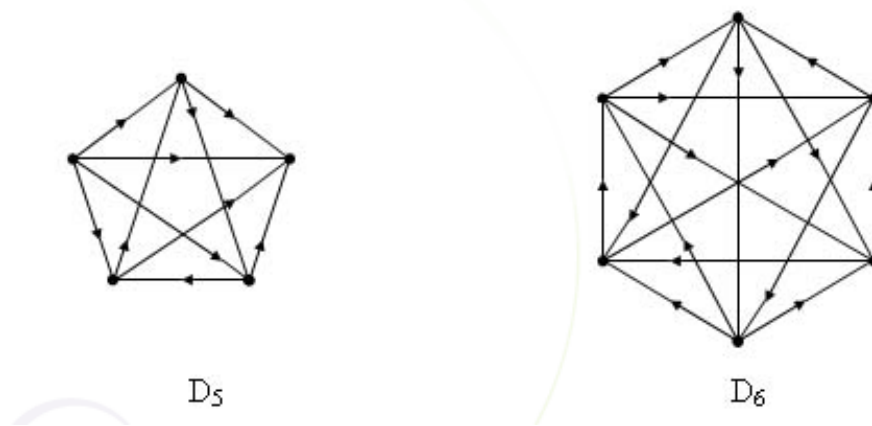
thì  $G$  là Hamilton.

Có một số dạng đồ thị mà ta có thể biết khi nào là đồ thị Hamilton. Một ví dụ như vậy là đồ thị đấu loại. Đồ thị đấu loại là đồ thị có hướng mà trong đó hai đỉnh bất kỳ của nó được nối với nhau bởi đúng một cung. Tên đấu loại xuất hiện như vậy vì đồ thị như vậy có thể dùng để biểu diễn kết quả thi đấu bóng chuyền, bóng bàn hay bất cứ một trò chơi nào mà không cho phép hoà. Ta có định lý sau:

#### **Định lý 5.**

- i) Mọi đồ thị đấu loại là nửa Hamilton.
- ii) Mọi đồ thị đấu loại liên thông mạnh là Hamilton.

**Thí dụ 4.** Đồ thị đấu loại  $D_5, D_6$  được cho trong hình 5.



Hình 5. Đồ thị đầu loại  $D_5$ , đầu loại liên thông mạnh  $D_6$

Thuật toán liệt kê tất cả các chu trình Hamilton của đồ thị

Thuật toán sau đây được xây dựng dựa trên cơ sở thuật toán quay lui cho phép liệt kê tất cả các chu trình Hamilton của đồ thị.

Procedure Hamilton(k);

*(\* liệt kê các chu trình Hamilton thu được bằng việc phát triển dãy đỉnh  $(X[1], \dots, X[k-1])$  của đồ thị  $G=(V,E)$  cho bởi danh sách kề:  $Ke(v), v \in V$  \*)*

begin

for  $y \in Ke(X[k-1])$  do

if  $(k = N+1)$  and  $(y=v_0)$  then Ghinhan( $X[1], \dots, X[n], v_0$ )

else

if Chuaxet[y] then

begin

$X[k] := y;$

$Chuaxet[y] := false;$

Hamilton(k+1);

$Chuaxet[y] := true;$



# Cây và cây khung của đồ thị

Đồ thị vô hướng liên thông không có chu trình gọi là cây. Khái niệm cây lần đầu tiên được Cayley đưa ra vào năm 1857, khi ông sử dụng chúng để đếm một dạng cấu trúc phân tử của các hợp chất hoá học trong hoá học hữu cơ. Cây còn được sử dụng rộng rãi trong rất nhiều lĩnh vực khác nhau, đặc biệt trong tin học, cây được sử dụng để xây dựng các thuật toán tổ chức các thư mục, các thuật toán cắt giử, truyền dữ liệu và tìm kiếm...

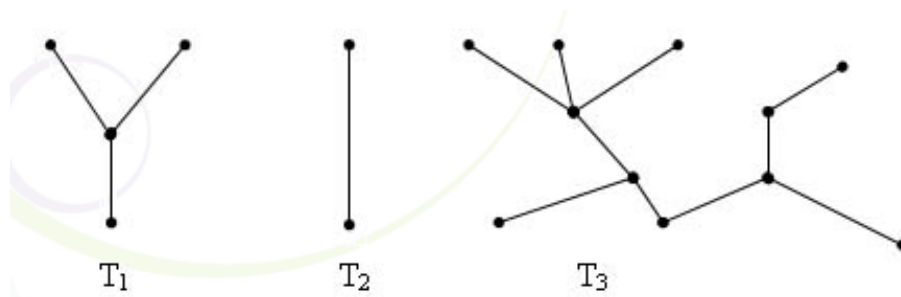
## CÂY VÀ CÁC TÍNH CHẤT CƠ BẢN CỦA CÂY

### Định nghĩa1.

Ta gọi cây là đồ thị vô hướng liên thông không có chu trình. Đồ thị không có chu trình được gọi là rừng.

Như vậy, rừng là đồ thị mà mỗi thành phần liên thông của nó là một cây.

**Thí dụ 1.** Trong hình 1 là một rừng gồm 3 cây  $T_1$ ,  $T_2$ ,  $T_3$ .



**Hình 1.** Rừng gồm 3 cây  $T_1$ ,  $T_2$ ,  $T_3$ .

Có thể nói cây là đồ thị vô hướng đơn giản nhất. Định lý sau đây cho ta một số tính chất của cây.

**Định lý 1.** Giả sử  $G=(V,E)$  là đồ thị vô hướng  $n$  đỉnh. Khi đó các mệnh đề sau đây là tương đương:

- (1)  $T$  là cây;
- (2)  $T$  không chứa chu trình và có  $n-1$  cạnh;
- (3)  $T$  liên thông và có  $n-1$  cạnh;

(4) T liên thông và mỗi cạnh của nó đều là cầu;

(5) Hai đỉnh bất kỳ của T được nối với nhau bởi đúng một đường đi đơn;

(6) T không chứa chu trình nhưng hề cứ thêm vào một cạnh ta thu được đúng một chu trình.

**Chứng minh.** Ta sẽ chứng minh định lý theo sơ đồ sau:

$(1) \Rightarrow (2) \Rightarrow (3) \Rightarrow (4) \Rightarrow (5) \Rightarrow (6) \Rightarrow (1)$

$(1) \Rightarrow (2)$  Theo định nghĩa T không chứa chu trình. Ta sẽ chứng minh bằng qui nạp theo số đỉnh n cho khẳng định: Số cạnh của cây với n đỉnh là n-1. Rõ ràng khẳng định đúng với n=1. Giả sử n>1. Trước hết nhận rằng trong mọi cây T có n đỉnh đều tìm được ít nhất một đỉnh là đỉnh treo (tức là đỉnh có bậc là 1). Thực vậy, gọi  $v_1, v_2, \dots, v_k$  là đường đi dài nhất (theo số cạnh) trong T. Khi đó rõ ràng  $v_1$  và  $v_k$  là các đỉnh treo, vì từ  $v_1$  ( $v_k$ ) không có cạnh nối với bất cứ đỉnh nào trong số các đỉnh  $v_2, v_3, \dots, v_k$  (do đồ thị không chứa chu trình), cũng như với bất cứ đỉnh nào khác của đồ thị (do đường đi đang xét dài nhất). Loại bỏ  $v_1$  và cạnh  $(v_1, v_2)$  khỏi T ta thu được cây  $T_1$  với n-1 đỉnh, mà theo giả thiết qui nạp có n-2 cạnh. Vậy cây T có  $n-2+1 = n-1$  cạnh.

$(2) \Rightarrow (3)$  Ta chứng minh bằng phản chứng. Giả sử T không liên thông. Khi đó T phân rã thành  $k \geq 2$  phần liên thông  $T_1, T_2, \dots, T_k$ . Do T không chứa chu trình nên mỗi  $T_i$  ( $i=1, 2, \dots, k$ ) cũng không chứa chu trình, vì thế mỗi  $T_i$  là cây. Do đó nếu gọi  $n(T_i)$  và  $e(T_i)$  theo thứ tự là số đỉnh và cạnh của  $T_i$ , ta có:

$$e(T_i) = n(T_i) - 1, i = 1, 2, \dots, k,$$

suy ra

$$n-1 = e(T) = e(T_1) + \dots + e(T_k)$$

$$= n(T_1) + \dots + n(T_k) - k$$

$$= n(T) - k < n-1$$

Mâu thuẫn thu được chứng tỏ là T liên thông.

$(3) \Rightarrow (4)$  Việc loại bỏ một cạnh bất kỳ khỏi T dẫn đến đồ thị với n đỉnh và n-2 cạnh rõ ràng là đồ thị không liên thông. Vậy mọi cạnh trong T đều là cầu.

(4)  $\Rightarrow$  (5) Do  $T$  là liên thông nên hai đỉnh bất kỳ của nó được nối với nhau bởi một đường đi đơn. Nếu có cặp đỉnh nào của  $T$  có hai đường đi đơn khác nhau nối chúng, thì từ đó suy ra đồ thị chứa chu trình, và vì thế các cạnh trên chu trình này không phải là cầu.

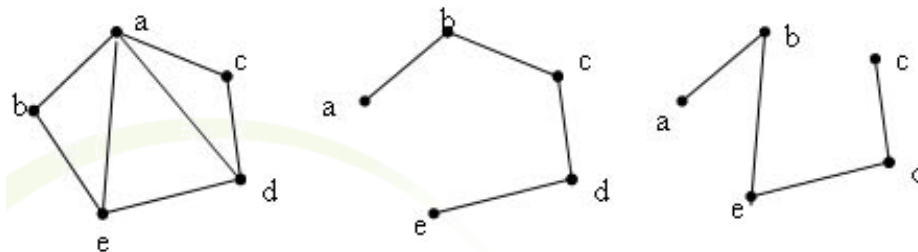
(5)  $\Rightarrow$  (6)  $T$  không chứa chu trình, bởi vì thế nếu có chu trình thì hoá ra tìm được cặp đỉnh của  $T$  được nối với nhau bởi hai đường đi đơn. Bây giờ, nếu thêm vào  $T$  một cạnh  $e$  nối hai đỉnh  $u$  và  $v$  nào đó của  $T$ . Khi đó cạnh này cùng với đường đi đơn nối  $u$  với  $v$  sẽ tạo thành chu trình trong  $T$ . Chu trình thu được này là duy nhất, vì nếu thu được nhiều hơn một chu trình thì suy ra trong  $T$  trước đó phải có sẵn chu trình.

(6)  $\Rightarrow$  (1) Giả sử  $T$  không liên thông. Khi đó gồm ít ra là 2 thành phần liên thông. Vì vậy, nếu thêm vào  $T$  một cạnh nối hai đỉnh thuộc hai thành phần liên thông khác nhau ta không thu được thêm một chu trình nào cả. Điều đó mâu thuẫn với giả thiết (6).

Định lý được chứng minh.

# Cây khung của đồ thị

**Định nghĩa 2.** Đồ thị  $G$  và cây khung của nó được cho trong hình 2



Hình 2. Đồ thị và các cây khung của nó

Định lý sau đây cho biết số lượng cây khung của đồ thị đầy đủ  $K_n$ :

**Định lý 2 (Cayley).** Số lượng cây khung của đồ thị  $K_n$  là  $n^{n-2}$ .

Định lý 2 cho thấy số lượng cây khung của đồ thị là một số rất lớn. Bây giờ ta xét áp dụng của thuật toán tìm kiếm theo chiều sâu và theo chiều rộng trên đồ thị để xây dựng cây khung của đồ thị vô hướng liên thông. Trong cả hai trường hợp mỗi khi ta đến được đỉnh mới  $u$  (tức  $\text{Chuaxet}[u]=\text{true}$ ) từ đỉnh  $v$  thì cạnh  $(v, u)$  sẽ được kết nạp vào cây khung. Hai thuật toán tương ứng được trình bày trong hai thủ tục sau đây.

Procedure  $\text{stree\_DFS}(v)$ ;

*(\* tìm kiếm theo chiều sâu áp dụng vào tìm tập cạnh của cây khung  $T$  của đồ thị vô hướng liên thông  $G$  cho bởi danh sách kề. Các biến  $\text{Chuaxet}$ ,  $Ke$ ,  $T$  là toàn cục\*)*

begin

$\text{Chuaxet}[v] := \text{false}$ ;

For  $u \in Ke(v)$  do

If  $\text{Chuaxet}[u]$  then

Begin

$T := T \cup (u, v)$ ;

$\text{STREE\_DFS}(u)$ ;

```

End;

end;

(* Main Program *)

begin

(* Initialition *)

for  $u \in V$  do Chuaxet[u]:=true;

 $T := \emptyset$ ; (* T la tap canh cua cay khung *)

STREE_DFS(root); ( root la dinh nao do cua do thi *)

end.

Procedure Stree_BFS(v);

(* tim kiem theo chieu rong ap dung tim tap canh cua cau khung T cua do thi vo huong
lien thong G cho boi danh sach Ke *)

begin

Queue:=  $\emptyset$ ;

Queue  $\leftarrow r$ ;

Chuaxet[r]:=false;

While queue  $\neq \emptyset$  do

Begin

V  $\leftarrow$  queue;

For  $r \in Ke(v)$  do

If Chuaxet[u] then

Begin

Queue  $\leftarrow u$ ;

```



```

Chuaxet[u]:=false;

T:= T ? (u,v);

End;

End;

end;

(* Main Program *);

begin

for u ∈ V do Chuaxet[u]:=true;

T := ∅; (* T là tập cạnh của cây khung *)

Stree_BFS(root); (* root là một đỉnh tùy ý của đồ thị *)

end.

```

Chú ý:

1. Lập luận tương tự như trong phần trước có thể chỉ ra được rằng các thuật toán mô tả ở trên có độ phức tạp tính toán  $O(m+n)$ .
2. Cây khung tìm được theo thủ tục Stree\_BFS() là cây đường đi ngắn nhất từ gốc r đến tất cả các đỉnh còn lại của đồ thị.

# Xây dựng tập các chu trình cơ bản của đồ thị

Bài toán xây dựng cây khung của đồ thị liên quan chặt chẽ đến một số bài toán ứng dụng khác của lý thuyết đồ thị: bài toán xây dựng tập các chu trình cơ bản của đồ thị mà ta sẽ xét trong mục này.

Giả sử  $G=(V,E)$  là đơn đồ thị vô hướng liên thông,  $H=(V,T)$  là cây khung của nó. Các cạnh của đồ thị thuộc cây khung ta sẽ gọi là các cạnh trong, còn các cạnh còn lại sẽ gọi là cạnh ngoài.

## Định nghĩa 3.

Nếu thêm một cạnh ngoài  $e \in E \setminus T$  vào cây khung  $H$  chúng ta sẽ thu được đúng một chu trình trong  $H$ , ký hiệu chu trình này là  $C_e$ . Tập các chu trình

$$\mathcal{C} = \{ C_e : e \in E \setminus T \}$$

được gọi là tập các chu trình cơ bản của đồ thị  $G$ .

Giả sử  $A$  và  $B$  là hai tập hợp, ta đưa vào phép toán sau

$$A \oplus B = (A \cup B) \setminus (A \cap B).$$

Tập  $A \oplus B$  được gọi là hiệu đối xứng của hai tập  $A$  và  $B$ .

Tên gọi chu trình cơ bản gắn liền với sự kiện là mỗi chu trình của đồ thị đều có thể thu được từ các chu trình cơ bản như chỉ ra trong định lý sau đây:

## Định lý 3.

Giả sử  $G=(V,E)$  là đồ thị vô hướng liên thông,  $H=(V,T)$  là cây khung của nó. Khi đó mọi chu trình của đồ thị  $G$  đều có thể biểu diễn như là hiệu đối xứng của một số các chu trình cơ bản.

Việc tìm tập hợp chu trình cơ bản giữ một vai trò quan trọng trong vấn đề giải tích mạng điện. Cụ thể hơn, theo mỗi chu trình cơ bản của đồ thị tương ứng với mạng điện cần phân tích ta sẽ thiết lập được một phương trình tuyến tính theo định luật Kirchoff: tổng hiệu điện thế dọc theo một mạch vòng là bằng không. Hệ thống phương trình tuyến tính thu được cho phép tính toán hiệu điện thế trên mọi đường dây của lưới điện.

Ta sẽ xây dựng thuật toán xây dựng các chu trình cơ bản dựa trên thủ tục tìm kiếm theo chiều sâu trên đồ thị. Thuật toán có cấu trúc tương tự như thuật toán xây dựng cây khung theo thủ tục tìm kiếm theo chiều sâu mô tả trong mục trước.

Thuật toán xây dựng tập các chu trình cơ bản.

Giả thiết rằng đồ thị  $G=(V,E)$  được mô tả bằng danh sách  $Ke(v), v \in V$ .

Procedure Cycle(v);

(\* tìm kiếm các chu trình cơ bản của thành phần liên thông chưa đỉnh v; các biến d, num, stack, index là biến toàn cục \*)

begin

d:=d+1; stack[d]:=v; num:=num+1; index[v]:=num;

*for*  $u \in Ke(v)$  *do*

if index[u]=0 then cycle(u)

else

if  $(u \triangleleft stack[d-1])$  and  $(index[v] > index[u])$  then

<Ghi nhận chu trình với các đỉnh:

stack[d], stack[d-1], . . . , stack[c], với stack[c]=u>

d:=d-1;

end;

(\* Main Program \*)

begin

*for*  $v \in V$  *do* Index[v]:=0;

num:=0; d:=0; stack[0]:=0;

*for*  $v \in V$  *do*

if Index[v]=0 then cycle(v);

end.

*Chú ý:* Độ phức tạp tính toán của thuật toán vừa mô tả là  $O(E + V)$ .

# Bài toán cây khung nhỏ nhất

Bài toán cây khung nhỏ nhất của đồ thị là một trong số những bài toán tối ưu trên đồ thị tìm được ứng dụng trong nhiều lĩnh vực khác nhau của đời sống. Trong mục này chúng ta trình bày những thuật toán cơ bản để giải bài toán nào. Trước hết chúng ta phát biểu nội dung bài toán.

Cho  $G=(V,E)$  là đồ thị vô hướng liên thông với tập đỉnh  $V=\{1, 2, \dots, n\}$  và tập cạnh  $E$  gồm  $m$  cạnh. Mỗi cạnh  $E$  của đồ thị  $G$  được gán với một số không âm  $c(e)$ , gọi là độ dài của nó. Giả sử  $H=(V,T)$  là cây khung của đồ thị  $G$ . Ta gọi độ dài  $c(H)$  của cây khung  $H$  là tổng độ dài các cạnh của nó:

$$C(H) = \sum_{e \in T} c(e).$$

Bài toán đặt ra là trong tất cả cây khung của đồ thị  $G$  hãy tìm cây khung với độ dài nhỏ nhất. Cây khung như vậy được gọi là cây khung nhỏ nhất của đồ thị và bài toán đặt ra được gọi là bài toán cây khung nhỏ nhất.

Để minh họa cho những ứng dụng bài toán cây khung nhỏ nhất, dưới đây, ta phát biểu hai mô hình thực tế tiêu biểu của nó.

*Bài toán xây dựng hệ thống đường sắt.* Giả sử ta muốn xây dựng một hệ thống đường sắt nối  $n$  thành phố sao cho hành khách có thể đi từ bất kỳ một thành phố nào đến bất kỳ một trong các thành phố còn lại. Mặt khác trên quan điểm kinh tế đòi hỏi là chi phí xây dựng hệ thống đường phải nhỏ nhất. Rõ ràng đồ thị mà đỉnh là các thành phố còn các cạnh là các tuyến đường sắt nối các thành phố tương ứng với phương án xây dựng tối ưu phải là cây. Vì vậy, bài toán đặt ra dẫn về bài toán tìm cây khung nhỏ nhất trên đồ thị đầy đủ  $n$  đỉnh, mỗi đỉnh tương ứng với một thành phố, với độ dài trên các cạnh chính là chi phí xây dựng đường ray nối hai thành phố tương ứng (chú ý là trong bài toán này ta giả thiết là không xây dựng tuyến đường sắt có các nhà ga phân tuyến nằm ngoài các thành phố).

*Bài toán nối mạng máy tính.* Cần nối mạng một hệ thống gồm  $n$  máy tính đánh số từ 1 đến  $n$ . Biết chi phí nối máy  $i$  với máy  $j$  là  $c[i,j]$ ,  $i,j = 1, 2, \dots, n$  (thông thường chi phí này phụ thuộc vào độ dài cáp nối cần sử dụng). Hãy tìm cách nối mạng sao cho tổng chi phí nối mạng là nhỏ nhất.

Để giải bài toán cây khung nhỏ nhất, tất nhiên có thể liệt kê tất cả các cây khung của đồ thị và chọn trong số cây khung ấy cây khung nhỏ nhất. Phương pháp như vậy, trong trường hợp đồ thị đầy đủ, sẽ đòi hỏi thời gian cỡ  $nn^2$ , và rõ ràng không thể thực hiện được ngay cả với những đồ thị với số đỉnh cỡ hàng chục. Rất may là đối với bài toán

cây khung nhỏ nhất chúng ta đã có những thuật toán rất hiệu quả để giải chúng. Chúng ta xét hai trong số những thuật toán như vậy: Thuật toán Kruskal và Thuật toán Prim.

## Thuật toán Kruskal

Thuật toán sẽ xây dựng tập cạnh  $T$  của cây khung nhỏ nhất  $H=(V,T)$  theo từng bước. Trước hết sắp xếp các cạnh của đồ thị  $G$  theo thứ tự không giảm của độ dài. Bắt đầu từ tập  $T=\emptyset$ , ở mỗi bước ta sẽ lần lượt duyệt trong danh sách cạnh đã sắp xếp, từ cạnh có độ dài nhỏ đến cạnh có độ dài lớn hơn, để tìm ra cạnh mà việc bổ sung nó vào tập  $T$  gồm  $n-1$  cạnh. Cụ thể, thuật toán có thể mô tả như sau:

Procedure Kruskal;

Begin

$T := \emptyset$ ;

While  $|T| < (n-1)$  and  $(E \setminus T \neq \emptyset)$  do

Begin

$E := E \setminus T$ ;

if  $(T \cup e$  không chứa chu trình) then  $T := T \cup e$ ;

End;

if  $(|T| < n-1)$  then Đồ thị không liên thông;

End;

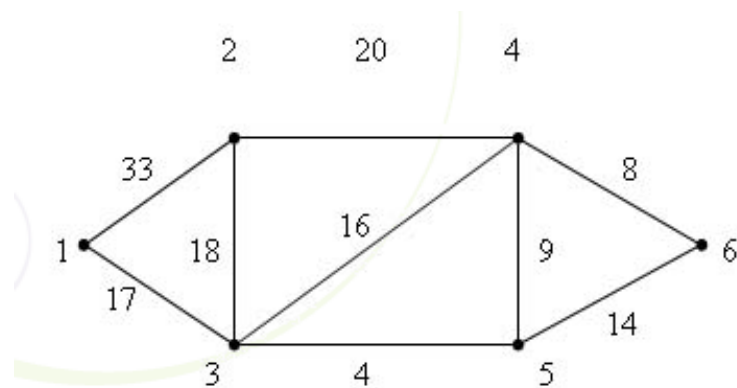
**Thí dụ 3.** Tìm cây khung nhỏ nhất của đồ thị cho trong hình 3 dưới.

Bước khởi tạo. Đặt  $T := \emptyset$ . Sắp xếp các cạnh của đồ thị theo thứ tự không giảm của độ dài ta có dãy:

$(3,5)$ ,  $(4,6)$ ,  $(4,5)$ ,  $(5,6)$ ,  $(3,4)$ ,  $(1,3)$ ,  $(2,3)$ ,  $(2,4)$ ,  $(1,2)$

dãy độ dài tương ứng của chúng

4, 8, 9, 14, 16, 17, 18, 20, 23.



Hình 3. Đồ thị và cây khung nhỏ nhất

Ở ba lần gặp đầu tiên ta lần lượt bổ sung vào tập  $T$  các cạnh  $(3,5)$ ,  $(4,6)$ ,  $(4,5)$ . Rõ ràng nếu thêm cạnh  $(5,6)$  vào  $T$  thì sẽ tạo thành 2 cạnh  $(4,5)$ ,  $(4,6)$  đã có trong  $T$  chu trình. Tình huống tương tự cũng xảy ra đối với cạnh  $(3,4)$  là cạnh tiếp theo của dãy. Tiếp theo ta bổ sung cạnh  $(1,3)$ ,  $(2,3)$  vào  $T$  và thu được tập  $T$  gồm 5 cạnh:

$T = \{ (3,5), (4,6), (4,5), (1,3), (2,3) \}$  ?

Chính là tập cạnh của cây khung nhỏ nhất cần tìm.

Chứng minh tính đúng đắn của thuật toán.

Rõ ràng đồ thị thu được theo thuật toán có  $n-1$  cạnh và không có chu trình, vì vậy theo định lý 1 nó là cây khung của đồ thị  $G$ . Như vậy, chỉ còn phải chỉ ra rằng  $T$  có độ dài nhỏ nhất. Giả sử tồn tại cây  $S$  của đồ thị  $G$  mà  $c(S) < c(T)$ . Ký hiệu  $e_k$  là cạnh đầu tiên trong dãy các cạnh của  $T$  xây dựng theo thuật toán vừa mô tả không thuộc  $S$ . Khi đó đồ thị con của  $G$  sinh bởi cây  $S$  được bổ sung cạnh  $e_k$  sẽ chứa một chu trình  $C$  duy nhất đi qua  $e_k$ . Do chu trình  $C$  phải chứa cạnh  $e$  thuộc  $S$  nhưng không thuộc  $T$  nên đồ thị con thu được từ  $S$  bằng cách thay cạnh  $e$  của nó bởi cạnh  $e_k$  (ký hiệu đồ thị là  $S'$ ) sẽ là cây khung. Theo cách xây dựng  $c(e_k) \leq c(e)$  do đó  $c(S') \leq c(S)$ , đồng thời số cạnh chung của  $S'$  và  $T$  đã tăng thêm 1 so với số cạnh chung của  $S$  và  $T$ . Lặp lại quá trình trên từng bước một ta có thể biến đổi  $S$  thành  $T$  và trong mỗi bước tổng độ dài không tăng, tức là  $c(T) \leq c(S)$ . Mâu thuẫn thu được chứng tỏ  $T$  là cây khung nhỏ nhất.

Về việc lập trình thực hiện thuật toán.

Khối lượng tính toán nhiều nhất của thuật toán chính là ở bước sắp xếp các cạnh của đồ thị theo thứ tự không giảm của độ dài để lựa chọn cạnh bổ sung. Đối với đồ thị  $m$  cạnh cần phải thực hiện  $m \log m$  phép toán để sắp xếp các cạnh của đồ thị thành dãy không giảm theo độ dài. Tuy nhiên, để xây dựng cây khung nhỏ nhất với  $n-1$  cạnh, nói chung ta không cần phải sắp thứ tự toàn bộ các cạnh mà chỉ cần xét phần trên của dãy đó chứa  $r < m$  cạnh. Để làm việc đó ta có thể sử dụng các thủ tục sắp xếp dạng Vun đống (Heap

Sort). Trong thủ tục này, để tạo đồng đầu tiên ta mất cỡ  $O(m)$  phép toán, mỗi phần tử tiếp theo trong đồng có thể tìm sau thời gian  $O(\log m)$ . Vì vậy, với cải tiến này thuật toán sẽ mất thời gian cỡ  $O(m+p) \log m$  cho việc sắp xếp các cạnh. Trong thực tế tính toán số  $p$  nhỏ hơn rất nhiều so với  $m$ .

Vấn đề thứ hai trong việc thể hiện thuật toán Kruskal là việc lựa chọn cạnh để bổ sung đòi hỏi phải có một thủ tục hiệu quả kiểm tra tập cạnh  $T \cup \{e\}$  có chứa chu trình hay không. Để ý rằng, các cạnh trong  $T$  ở các bước lặp trung gian sẽ tạo thành một rừng. Cạnh  $e$  cần khảo sát sẽ tạo thành chu trình với các cạnh trong  $T$  khi và chỉ khi cả hai đỉnh đầu của nó thuộc vào cùng một cây con của rừng nói trên. Do đó, nếu cạnh  $e$  không tạo thành chu trình với các cạnh trong  $T$ , thì nó phải nối hai cây khác nhau trong  $T$ . vì thế, để kiểm tra xem có thể bổ sung cạnh  $e$  vào  $T$  ta chỉ cần kiểm tra xem nó có nối hai cây khác nhau trong  $T$  hay không. Một trong các phương pháp hiệu quả để thực hiện việc kiểm tra này là ta sẽ phân hoạch tập các đỉnh của đồ thị ra thành các tập con không giao nhau, mỗi tập xác định bởi một cây con trong  $T$  (được hình thành ở các bước do việc bổ sung cạnh vào  $T$ ). chẳng hạn, đối với đồ thị trong ví dụ 3, đầu tiên ta có sáu tập con 1 phần tử:  $\{1\}, \{2\}, \{3\}, \{4\}, \{5\}, \{6\}$ . Sau khi bổ sung cạnh  $(3,5)$ , ta có các tập con  $\{1\}, \{2\}, \{3,5\}, \{4\}, \{6\}$ . Ở bước thứ 3, ta chọn cạnh  $(4,5)$ , khi đó hai tập con được nối lại và danh sách các tập con là  $\{1\}, \{2\}, \{3,4,5,6\}$ . Cạnh có độ dài tiếp theo là  $(4,6)$ , do hai đầu của nó thuộc vào cùng một tập con  $\{3,4,5,6\}$ , nên nó sẽ tạo thành chu trình trong tập này. Vì vậy cạnh này không được chọn. Và thuật toán sẽ tiếp tục chọn cạnh tiếp theo để khảo sát ...

Như vậy, để giải quyết vấn đề thứ hai này ta phải xây dựng hai thủ tục: Kiểm tra xem hai đầu  $u, v$  của cạnh  $e=(u,v)$  có thuộc vào hai tập con khác nhau hay không, và trong trường hợp câu trả lời là khẳng định, nối hai tập con tương ứng thành một tập. Chú ý rằng mỗi tập con trong phân hoạch có thể lưu trữ như là một cây có gốc, và khi đó mỗi gốc sẽ được sử dụng làm nhãn nhận biết tập con tương ứng. Chương trình trên Pascal thực hiện thuật toán Kruskal với những nhận xét vừa nêu có thể viết như sau:

(\* TÌM CÂY KHUNG NHỎ NHẤT THEO THUẬT TOÁN KRUSKAL CỦA ĐỒ THỊ CHO BỞI DANH SÁCH CẠNH \*)

```
uses crt;
```

```
type
```

```
arrn=array[1..50] of integer;
```

```
arrm= array[1...50] of integer;
```

```
var
```

```
n,m, minL:integer;
```



```

Dau, cuoi, W:array;

DauT, CuoiT, Father:array;

Connect:boolean;

Procedure Nhapdl;

Var

i:integer;

Fanme:string;

F:text;

Begin

Write('Cho ten file du lieu:') readln(fname);

Assign(f,fname); reset(f);

Readln(f,n,m);

For i:=1 to m do readln(f, Dau[i], Cuoi[i], W[i]);

Close(f);

End;

Procedure Indulieu;

Var i:integer;

Begin

Writeln('So dinh ',n,'. So canh ',m);

Writeln('Dinh dau Dinh cuoi Do dai');

For i:=1 to m do

Writeln(Dau[i]:4, Cuoi[i]:10, W[i]:12);

```

```

End;

Procedure Heap(First, Last:integer);

Var j,k,t1,t2,t3 : integer;

Begin

J:=first;

While (j<=trunc(last/2)) do

Begin

If (2*j<last) and W[2*j+1]<W[2*j]) then K:= 2*j+1

Else k:=2*j;

If W[k]<W[j] then

Begin

T1:=Dau[j]; t1:=Cuoi[j]; t3:=W[j];

Dau[j]:=Dau[k];Cuoi[j]:=Cuoi[k]; W[j]:=W[k];

Dau[k]:=t1; Cuoi[k]:=t2; W[k]:=t3;

J:=k;

End

Else j:=Last;

End;

End;

Function Find(i:integer):integer;

Var Tro:integer;

Begin

```

```

Tro:=i;

While Father[Tro]>0 do Tro:=Father[Tro];

Find:=Tro;

End;

Procedure Union(i,j:integer);

Var x:integer;

Begin

x:=father[i]+father[j];

if father[i]>father[j] then

begin

father[i]:=f;

father[j]:=x;

end

else

begin

father[j]:=i;

father[i]:=x;

end;

End;

Procedure Kruskal;

Var

I, Last, u,v,r1,r2, Ncanh, Ndinh:integer;

```

Begin

(\* Khoi tao mang Father danh dau cay con va khoi tao Heap \*)

for i:= 1 to n do father[i]:=-1;

for i:=trunc(m/2) downto 1 do Heap(i,m);

last:=m; Ncanh:=0; Ndinh:=0;

MinL:=0;

Connect:=true;

While (Ndinh<n-1) and (Ncanh<m) do

Begin

Ncanh:=Ncanh+1;

u:=dau[1];

v:=Cuoi[1];

(\* Kiem tra u va v co thuoc cung mot cay con \*)

r1:=find(u);

r2:=find(v);

if  $r1 \neq r2$  then

begin

(\* Ket nap canh (u,v) vao cay khung \*)

Ndinh:=Ndinh+1; Union(r1, r2);

DauT[Ndinh]:=u;

CuoiT[Ndinh]:=v;

MinL:=MinL+W[1];

```

end;

(* To chuc lai Heap *)

Dau[1]:=Dau[Last];

Cuoi[1]:=Cuoi[Last];

W[1]:=W[Last];

Last:=Last-1;

End;

If Ndinh<>n-1 then Connect:=false;

End;

Procedure Inketqua;

Var i:integer;

Begin

Writeln('*****');

Writeln('***** Ket qua tinh toan *****');

Writeln('*****');

Writeln('Do dai cua cay khung nho nhat: ',MinL);

Writeln('Cac canh cua cay khung nho nhat');

For i:=1 to n-1 do

    Writeln((' ',DauT[i]:2,', ',CuoiT[i]:2,','));

Writeln('*****');

End;

Begin

```

```
Clrscr;  
Nhapdl;'  
Indulieu;  
Kruskal;  
If connect then Inketqua  
Else  
Writeln(' Do thi khong lien thong');  
Readln;  
End.
```

File dữ liệu của bài toán trong ví dụ 3 có dạng sau:

```
7 9  
3 5 4  
4 6 8  
4 5 9  
5 6 14  
3 4 16  
1 3 17  
2 3 18  
2 4 20  
1 2 23
```

## Thuật toán Prim

Thuật toán Kruskal làm việc kém hiệu quả với những đồ thị dày (đồ thị với số cạnh  $m \approx n(n-1)/2$ ). Trong trường hợp đó thuật toán Prim tỏ ra hiệu quả hơn. Thuật toán Prim còn được gọi là phương pháp lân cận gần nhất. Trong phương pháp này bắt đầu từ một đỉnh tùy ý của đồ thị, đầu tiên ta nối  $s$  với đỉnh lân cận gần nó nhất, chẳng hạn là đỉnh  $y$ . Nghĩa là trong số các cạnh kề của đỉnh  $s$ , cạnh  $(s,y)$  có độ dài nhỏ nhất. Tiếp theo trong số các cạnh kề với hai đỉnh  $s$  hoặc  $y$  ta tìm cạnh có độ dài nhỏ nhất, cạnh này dẫn đến đỉnh thứ ba  $z$ , và ta thu được cây bộ phận gồm 3 đỉnh và 2 cạnh. Quá trình này sẽ tiếp tục cho đến khi ta thu được cây gồm  $n$  đỉnh và  $n-1$  cạnh sẽ chính là cây khung nhỏ nhất cần tìm.

Giả sử đồ thị cho bởi ma trận trọng số  $C = [c[i,j]]$ ,  $i, j = 1, 2, \dots, n$ . trong quá trình thực hiện thuật toán, ở mỗi bước để có thể nhanh chóng chọn đỉnh và cạnh cần bổ sung vào cây khung, các đỉnh của đồ thị sẽ được gán cho các nhãn. Nhãn của một đỉnh  $v$  sẽ gồm hai phần và có dạng  $[d[v], \text{near}[v]]$ , trong đó  $d[v]$  dùng để ghi nhận độ dài của cạnh có độ dài nhỏ nhất trong số các cạnh nối với đỉnh  $v$  với các đỉnh của cây khung đang xây dựng (ta sẽ gọi là khoảng cách từ đỉnh  $v$  đến tập đỉnh của cây khung), nói một cách chính xác

$$d[v] := \min \{ c[v,w] : w \in V_H \} \quad (= c[v,z]),$$

còn  $\text{near}[v]$  ghi nhận đỉnh của cây khung gần  $v$  nhất ( $\text{near}[v] := z$ ).

Thuật toán Prim được mô tả đầy đủ trong thủ tục sau:

Procedur Prim;

Begin

(\* buoc khoi tao \*)

chon  $s$  la mot dinh nao do cua do thi;

$V_H := \{ s \} ; T := \emptyset ; d[s] := 0 ; \text{near}[s] := s.$

For  $v \in V \setminus V_H$  do

Begin

$D[v] := c[s,v];$

$\text{near}[v] := s;$

```

End;

(* buoc lap *)

stop:=false;

while not stop do

begin

tim  $u \in V \setminus VH$  thoa man:

 $d[u] = \min \{ d[v] : u \in V \setminus VH \}$  ;

 $VH := VH \cup u$  ;  $T := T \cup (u, near[u])$  ;

If  $|VH| = n$  then

Begin

H=( VH,T) la cay khung nho nhat cua do thi;

Stop:=true;

End

Else

For  $v \in V \setminus VH$  do

If  $d[v] > c[u,v]$  then

Begin

 $d[v] := c[u,v]$ ;

 $near[v] := u$ ;

End;

end;

End;

```



**Thí dụ 4.** Tìm cây khung nhỏ nhất cho đồ thị xét trong ví dụ 3 theo thuật toán Prim. Ma trận trọng số của đồ thị có dạng

		1	2	3	4	5	6
	1	0	33	17	$\infty$	$\infty$	$\infty$
	2	33	0	18	20	$\infty$	$\infty$
<b>C =</b>	3	17	18	0	16	4	$\infty$
	4	$\infty$	20	16	0	9	8
	5	$\infty$	$\infty$	4	9	0	14
	6	$\infty$	$\infty$	$\infty$	8	14	0

Bảng dưới đây ghi nhận của các đỉnh trong các bước lặp của thuật toán, đỉnh đánh dấu \* là đỉnh được chọn để bổ sung vào cây khung (khi đó nhãn của nó không còn bị biến đổi trong các bước lặp tiếp theo, vì vậy ta đánh dấu – để ghi nhận điều đó):

Bước lặp	Đỉnh 1	Đỉnh 2	Đỉnh 3	Đỉnh 4	Đỉnh 5	Đỉnh 6	VH	T
Khởi tạo	[0,1]	[33,1]	[17,1]*	[ $\infty$ ,1]	[ $\infty$ ,1]	[ $\infty$ ,1]	1	$\emptyset$
1	-	[18,3]	-	[16,3]	[4,3]*	[ $\infty$ ,1]	1,3	(3,1)
2	-	[18,3]	-	[9,5]	-	[14,5]	1,3,5	(3,1), (5,3)
3	-	[18,3]	-	-	-	[8,4]	1,3,5,4	(3,1), (5,3), (4,5)
4	-	[18,3]*	-	-	-	-	1,3,5,4,6	(3,1), (5,3), (4,5), (6,4)
5	-	-	-	-	-	-	1,3,5,4,6,2	(3,1), (5,3), (4,5), (6,4), (2,3)

## Một số bài toán dẫn về bài toán cây khung nhỏ nhất

Trong mục này ta nêu hai bài toán có thể dẫn về bài toán cây khung nhỏ nhất, và vì thế có thể giải được nhờ các thuật toán mô tả trong mục trước

a) Bài toán cây khung lớn nhất. Dễ dàng nhận thấy rằng trong các thuật toán trên ta không cần sử dụng đến đòi hỏi về dấu của độ dài. Vì thế có thể áp dụng chúng đối với

đồ thị có các cạnh với độ dài có dấu tùy ý. Vì vậy, giả sử ta phải tìm cây lớn nhất (tức là có độ dài  $c(H)$  lớn nhất) thì chỉ cần đổi dấu tất cả các độ đo và áp dụng một trong hai thuật toán vừa mô tả.

b) Bài toán tìm mạng điện với độ tin cậy lớn nhất. Cho lưới điện có  $n$  nút. Đường dây nối nút  $i$  với nút  $j$  có độ tin cậy  $1 > p[i, j] > 0$ ,  $i, j = 1, 2, \dots, n$ . Gọi  $G = (V, E)$  là đồ thị tương ứng với lưới điện này. Hãy tìm cây khung  $H$  của đồ thị  $G$  với độ tin cậy

$$\prod_{e \in H} p(e)$$

lớn nhất.

Bài toán này dẫn về bài toán tìm cây khung với tổng độ dài nhỏ nhất trên đồ thị  $G$  với độ dài của mỗi cạnh  $e \in E$  là  $-\log p(e)$ . Thực vậy, giả sử  $H$  là cây khung nhỏ nhất trên đồ thị với độ dài  $-\log p(e)$ , ta có

$$-\sum_{e \in H} \log p(e) \leq -\sum_{e \in H'} \log p(e) \quad e \in H'$$

với mọi cây khung  $H'$  của đồ thị  $G$ . Từ đó suy ra

$$\sum_{e \in H} \log p(e) \geq \sum_{e \in H'} \log p(e) \quad e \in H'$$

do đó

$$\log \prod_{e \in H} p(e) \geq \log \prod_{e \in H'} p(e) \quad e \in H'$$

hay là

$$\prod_{e \in H} p(e) \geq \prod_{e \in H'} p(e) \quad e \in H'$$

với mọi cây khung  $H'$ . Vậy  $H$  là cây khung có độ tin cậy lớn nhất.

# Bài toán đường đi ngắn nhất

Trong các ứng dụng thực tế, bài toán tìm đường đi ngắn nhất giữa hai đỉnh của một đồ thị liên thông có một ý nghĩa to lớn. Có thể dẫn về bài toán như vậy nhiều bài toán thực tế quan trọng. Ví dụ, bài toán chọn một hành trình tiết kiệm nhất (theo tiêu chuẩn hoặc khoảng cách hoặc thời gian hoặc chi phí) trên một mạng giao thông đường bộ, đường thủy hoặc đường không; bài toán chọn một phương pháp tiết kiệm nhất để đưa ra một hệ thống động lực từ trạng thái xuất phát đến trạng thái đích, bài toán lập lịch thi công các công đoạn trong một công trình thi công lớn, bài toán lựa chọn đường truyền tin với chi phí nhỏ nhất trong mạng thông tin, v.v... Hiện nay có rất nhiều phương pháp để giải các bài toán như vậy. Thế nhưng, thông thường, các thuật toán được xây dựng dựa trên cơ sở lý thuyết đồ thị tỏ ra là các thuật toán có hiệu quả cao nhất. Trong chương này chúng ta sẽ xét một số thuật toán như vậy.

## CÁC KHÁI NIỆM MỞ ĐẦU

Trong chương này chúng ta chỉ xét đồ thị có hướng  $G=(V,E)$ ,  $|V|=n$ ,  $|E|=m$  với các cung được gán trọng số, nghĩa là, mỗi cung  $(u,v) \in E$  của nó được đặt tương ứng với một số thực  $a(u,v)$  gọi là trọng số của nó. Chúng ta sẽ đặt  $a(u,v) = \infty$ , nếu  $(u,v) \notin E$ . Nếu dãy

$v_0, v_1, \dots, v_p$

là một đường đi trên  $G$ , thì độ dài của nó được định nghĩa là tổng sau

$$p \sum_{i=1}^p a(v_{i-1}, v_i).$$

tức là, độ dài của đường đi chính là tổng của các trọng số trên các cung của nó. (Chú ý rằng nếu chúng ta gán trọng số cho tất cả cung đều bằng 1, thì ta thu được định nghĩa độ dài của đường đi như là số cung của đường đi giống như trong các chương trước đã xét).

Bài toán tìm đường đi ngắn nhất trên đồ thị dưới dạng tổng quát có thể phát biểu như sau: tìm đường đi có độ dài nhỏ nhất từ một đỉnh xuất phát  $s \in V$  đến đỉnh cuối (đích)  $t \in V$ . Đường đi như vậy ta sẽ gọi là đường đi ngắn nhất từ  $s$  đến  $t$  còn độ dài của nó ta sẽ ký hiệu là  $d(s,t)$  và còn gọi là khoảng cách từ  $s$  đến  $t$  (khoảng cách định nghĩa như vậy có thể là số âm). Nếu như không tồn tại đường đi từ  $s$  đến  $t$  thì ta sẽ đặt  $d(s,t)=\infty$ . Rõ ràng, nếu như mỗi chu trình trong đồ thị đều có độ dài dương, trong đường đi ngắn nhất không có đỉnh nào bị lặp lại (đường đi không có đỉnh lặp lại sẽ gọi là đường đi cơ bản). Mặt khác nếu trong đồ thị có chu trình với độ dài âm (chu trình như vậy để gọi ngắn gọn

ta gọi là chu trình âm) thì khoảng cách giữa một số cặp đỉnh nào đó của đồ thị có thể là không xác định, bởi vì, bằng cách đi vòng theo chu trình này một số đủ lớn lần, ta có thể chỉ ra đường đi giữa các đỉnh này có độ dài nhỏ hơn bất cứ số thực cho trước nào. Trong những trường hợp như vậy, có thể đặt vấn đề tìm đường đi cơ bản ngắn nhất, tuy nhiên bài toán đặt ra sẽ trở nên phức tạp hơn rất nhiều, bởi vì nó chứa bài toán xét sự tồn tại đường đi Hamilton trong đồ thị như là một trường hợp riêng.

Trước hết cần chú ý rằng nếu biết khoảng cách từ  $s$  đến  $t$ , thì đường đi ngắn nhất từ  $s$  đến  $t$ , trong trường hợp trọng số không âm, có thể tìm được một cách dễ dàng. Để tìm đường đi, chỉ cần để ý là đối với cặp đỉnh  $s, t \in V$  tùy ý ( $s \neq t$ ) luôn tìm được đỉnh  $v$  sao cho

$$d(s,t) = d(s,v) + a(v,t).$$

Thực vậy, đỉnh  $v$  như vậy chính là đỉnh đi trước đỉnh  $t$  trong đường đi ngắn nhất từ  $s$  đến  $t$ . Tiếp theo ta lại có thể tìm được đỉnh  $u$  sao cho  $d(s,v) = d(s,u) + a(u,v)$ , . . . Từ giả thiết về tính không âm của các trọng số dễ dàng suy ra rằng dãy  $t, v, u, \dots$  không chứa đỉnh lặp lại và kết thúc ở đỉnh  $s$ . Rõ ràng dãy thu được xác định (nếu lật ngược thứ tự các đỉnh trong nó) đường đi ngắn nhất từ  $s$  đến  $t$ . Từ đó ta có thuật toán sau đây để tìm đường đi ngắn nhất từ  $s$  đến  $t$  khi biết độ dài của nó.

Procedure Find\_Path;

(\*

Đầu vào:

$D[v]$  - khoảng cách từ đỉnh  $s$  đến tất cả các đỉnh còn lại  $v \in V$ ;

- đỉnh đích;

$a[u,v], u, v \in V$  - ma trận trọng số trên các cung.

Đầu ra:

Mảng Stack chứa dãy đỉnh xác định đường đi ngắn nhất từ  $s$  đến  $t$

\*)

begin

$stack := \emptyset; stack \leftarrow t; v := t;$

while  $v \neq s$  do

begin

$u := \text{đỉnh thỏa mãn } d[v] = d[u] + a[u, v];$

$stack \leftarrow u;$

$v := u;$

end;

end;

Chú ý rằng độ phức tạp tính toán của thuật toán là  $O(n^2)$ , do để tìm đỉnh  $u$  ta phải xét qua tất cả các đỉnh của đồ thị. Tất nhiên, ta cũng có thể sử dụng kỹ thuật ghi nhận đường đi đã trình bày trong chương 3: dùng biến mảng  $Truoc[v]$ ,  $v \in V$ , để ghi nhớ đỉnh đi trước  $v$  trong đường đi tìm kiếm.

Cũng cần lưu ý thêm là trong trường hợp trọng số trên các cạnh là không âm, bài toán tìm đường đi ngắn nhất trên đồ thị vô hướng có thể dẫn về bài toán trên đồ thị có hướng, bằng cách thay đổi mỗi cạnh của nó bởi nó bởi hai cung có hướng ngược chiều nhau với cùng trọng số là trọng số của các cạnh tương ứng. Tuy nhiên, trong trường hợp có trọng số âm, việc thay như vậy có thể dẫn đến chu trình âm.

# Đường đi ngắn nhất xuất phát từ một đỉnh

Phần lớn các thuật toán tìm khoảng cách giữa hai đỉnh  $s$  và  $t$  được xây dựng nhờ kỹ thuật tính toán mà ta có thể mô tả đại thể như sau: từ ma trận trọng số  $a[u,v]$ ,  $u,v \in V$ , ta tính cận trên  $d[v]$  của khoảng cách từ  $s$  đến tất cả các đỉnh  $v \in V$ . Mỗi khi phát hiện

$$d[u] + a[u,v] < d[v] \quad (1)$$

cận trên  $d[v]$  sẽ được làm tốt lên:  $d[v] + a[u,v]$ .

Quá trình đó sẽ kết thúc khi nào chúng ta không làm tốt thêm được bất kỳ cận trên nào. Khi đó, rõ ràng giá trị của mỗi  $d[v]$  sẽ cho khoảng cách từ đỉnh  $s$  đến đỉnh  $v$ . Khi thể hiện kỹ thuật tính toán này trên máy tính, cận trên  $d[v]$  sẽ được gọi là nhãn của đỉnh  $v$ , còn việc tính lại các cận này sẽ được gọi là thủ tục gán. Nhận thấy rằng để tính khoảng cách từ  $s$  đến  $t$ , ở đây, ta phải tính khoảng cách từ  $s$  đến tất cả các đỉnh còn lại của đồ thị. Hiện nay vẫn chưa biết thuật toán nào cho phép tìm đường đi ngắn nhất giữa hai đỉnh làm việc thực sự hiệu quả hơn những thuật toán tìm đường đi ngắn nhất từ một đỉnh đến tất cả các đỉnh còn lại.

Sơ đồ tính toán mà ta vừa mô tả còn chưa xác định, bởi vì còn phải chỉ ra thứ tự các đỉnh  $u$  và  $v$  để kiểm tra điều kiện (1). Thứ tự chọn này có ảnh hưởng rất lớn đến hiệu quả của thuật toán.

Bây giờ ta sẽ mô tả thuật toán Ford-Bellman tìm đường đi ngắn nhất từ đỉnh  $s$  đến tất cả các đỉnh còn lại của đồ thị. Thuật toán làm việc trong trường hợp trọng số của các cung là tùy ý, nhưng giả thiết rằng trong đồ thị không có chu trình âm.

Procedure Ford\_Bellman

(\*

Đầu vào:

Đồ thị có hướng  $G=(V,E)$  với  $n$  đỉnh,

$S \in V$  là đỉnh xuất phát,  $A[u,v]$ ,  $u, v \in V$ , ma trận trọng số;

Giả thiết: Đồ thị không có chu trình âm.

Đầu ra:

Khoảng cách từ đỉnh  $s$  đến tất cả các đỉnh còn lại  $d[v]$ ,  $v \in V$ .

*Truoc[v],  $v \in V$ , ghi nhận đỉnh đi trước  $v$  trong đường đi ngắn nhất từ  $s$  đến  $v$ .*

\*)

begin

(\* Khởi tạo \*)

*for  $v \in V$  do*

begin

$d[v] := a[s, v];$

*Truoc[v] := s;*

end;

$d[s] := 0;$

for  $k := 1$  to  $n-2$  do

*for  $v \in V \setminus \{s\}$  do*

*for  $u \in V$  do*

if  $d[v] > d[u] + a[u, v]$  then

begin

$d[v] := d[u] + a[u, v];$

*Truoc[v] := u;*

end;

end;

Tính đúng đắn của thuật toán có thể chứng minh trên cơ sở trên nguyên lý tối ưu của quy hoạch động. Rõ ràng là độ phức tạp tính toán của thuật toán là  $O(n^3)$ . Lưu ý rằng chúng ta có thể chấm dứt vòng lặp theo  $k$  khi phát hiện trong quá trình thực hiện hai vòng lặp trong không có biến  $d[v]$  nào bị đổi giá trị. Việc này có thể xảy ra đối với  $k < n-2$ , và điều đó làm tăng hiệu quả của thuật toán trong việc giải các bài toán thực tế. Tuy nhiên, cải tiến đó không thực sự cải thiện được đánh giá độ phức tạp của bản thân thuật toán. Đối

với đồ thị thưa tốt hơn là sử dụng danh sách kề  $Ke(v)$ ,  $v \in V$ , để biểu diễn đồ thị, khi đó vòng lặp theo  $u$  cần viết lại dưới dạng

For  $u \in Ke(v)$  do

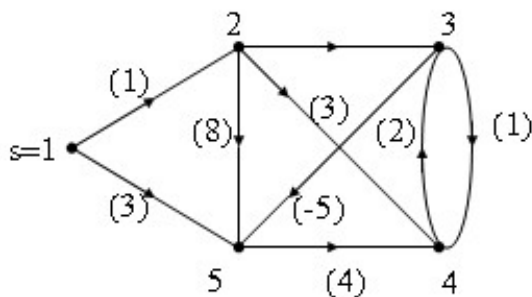
If  $d[v] > d[u] + a[u,v]$  thenBegin

$D[v] := d[u] + a[u,v]$ ;  $Truoc[v] := u$ ;

End;

Trong trường hợp này ta thu được thuật toán với độ phức tạp  $O(n,m)$ .

**Thí dụ 1.** xét đồ thị trong hình 1. Các kết quả tính toán theo thuật toán được mô tả trong bảng dưới đây



	$\infty$	2	$\infty$	$\infty$	$\infty$
	$\infty$	$\infty$	3	3	8
A=	$\infty$	$\infty$	$\infty$	1	-5
	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$
	$\infty$	$\infty$	$\infty$	4	$\infty$

**Hình 1.** Minh họa thuật toán Ford\_Bellman



k	d[1] Truoc[1]	d[2] Truoc[2]	d[3] Truoc[3]	d[4] Truoc[4]	d[5] Truoc[5]
	0,1	1,1	$\infty$ ,1	$\infty$ ,1	3,1
1	0,1	1,1	4,2	4,2	-1,3
2	0,1	1,1	4,2	3,5	-1,3
3	0,1	1,1	4,2	3,5	S

Bảng kết quả tính toán theo thuật toán Ford\_Bellman

Trong các mục tiếp theo chúng ta sẽ xét một số trường hợp riêng của bài toán tìm đường đi ngắn nhất mà để giải chúng có thể xây dựng những thuật toán hiệu quả hơn thuật toán Ford\_Bellman. Đó là khi trọng số của tất cả các cung là các số không âm hoặc là khi đồ thị không có chu trình.

# Trường hợp ma trận trọng số không âm- Thuật toán Dijkstra

Trong trường hợp trọng số trên các cung là không âm thuật toán do Dijkstra đề nghị làm việc hữu hiệu hơn rất nhiều so với thuật toán trình bày trong mục trước. Thuật toán được xây dựng dựa trên cơ sở gán cho các đỉnh các nhãn tạm thời. Nhãn của mỗi đỉnh cho biết cận của độ dài đường đi ngắn nhất từ  $s$  đến nó. Các nhãn này sẽ được biến đổi theo một thủ tục lặp, mà ở mỗi bước lặp có một nhãn tạm thời trở thành nhãn cố định. Nếu nhãn của một đỉnh nào đó trở thành một nhãn cố định thì nó sẽ cho ta không phải là cận trên mà là độ dài của đường đi ngắn nhất từ đỉnh  $s$  đến nó. Thuật toán được mô tả cụ thể như sau.

Procedure Dijkstra;

(\*

Đầu vào:

Đồ thị có hướng  $G=(V,E)$  với  $n$  đỉnh,

$s \in V$  là đỉnh xuất phát,  $a[u,v]$ ,  $u,v \in V$ , ma trận trọng số;

Giả thiết:  $a[u,v] \geq 0$ ,  $u,v \in V$ .

Đầu ra:

Khoảng cách từ đỉnh  $s$  đến tất cả các đỉnh còn lại  $d[v]$ ,  $v \in V$ .

Truoc[v],  $v \in V$ , ghi nhận đỉnh đi trước  $v$  trong đường đi ngắn nhất từ  $s$  đến  $v$

\*)

Begin

(\* Khởi tạo \*)

for  $v \in V$  do

begin

$d[v]:=a[s,v]$ ;

```

Truoc[v]:=s;

end;

d[s]:=0; T:=V \ { s } ; (* T là tập các đỉnh cá nhãn tạm thời *)

(* Bước lặp *)

while T <> ∅ do

begin

    tìm đỉnh u ∈ T thoả mãn d[u]=min { d[z]:z ∈ T } ;

    T:=T \ { u } ; (* Cố định nhãn của đỉnh u *)

    For v ∈ T do

        If d[v]>d[u]+a[u,v] then

            Begin

                d[v]:=d[u]+a[u,v];

                Truoc[v]:=u;

            End;

        end;

    End;

end;

```

**Định lý 1.** Thuật toán Dijkstra tìm được đường đi ngắn nhất trên đồ thị sau thời gian cỡ  $O(n^2)$ .

### **Chứng minh.**

Trước hết ta chứng minh là thuật toán tìm được đường đi ngắn nhất từ đỉnh s đến các đỉnh còn lại của đồ thị. Giả sử ở một bước lặp nào đó các nhãn cố định cho ta độ dài các đường đi ngắn nhất từ s đến các đỉnh có nhãn cố định, ta sẽ chứng minh rằng ở lần gặp tiếp theo nếu đỉnh  $u^*$  thu được nhãn cố định  $d(u^*)$  chính là độ dài đường đi ngắn nhất từ s đến  $u^*$ .

Ký hiệu  $S_1$  là tập hợp các đỉnh có nhãn cố định còn  $S_2$  là tập các đỉnh có nhãn tạm thời ở bước lặp đang xét. Kết thúc mỗi bước lặp nhãn tạm thời  $d(u^*)$  cho ta độ dài của đường đi ngắn nhất từ  $s$  đến  $u^*$  không nằm trong tập  $S_1$ , tức là nó đi qua ít nhất một đỉnh của tập  $S_2$ . Gọi  $z \in S_2$  là đỉnh đầu tiên như vậy trên đường đi này. Do trọng số trên các cung là không âm, nên đoạn đường từ  $z$  đến  $u^*$  có độ dài  $L > 0$  và

$$d(z) < d(u^*) - L < d(u^*).$$

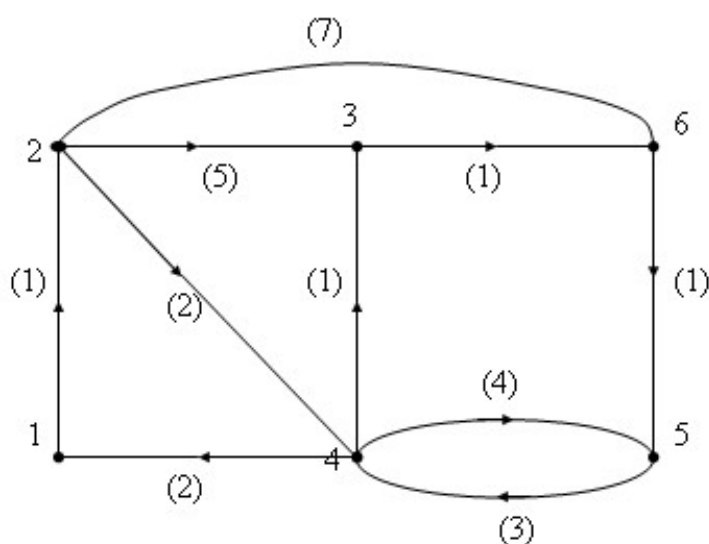
Bất đẳng thức này là mâu thuẫn với cách xác định đỉnh  $u^*$  là đỉnh có nhãn tạm thời nhỏ nhất. Vậy đường đi ngắn nhất từ  $s$  đến  $u^*$  phải nằm trọn trong  $S_1$ , và vì thế,  $d[u^*]$  là độ dài của nó. Do ở lần lặp đầu tiên  $S_1 = \{s\}$  và sau mỗi lần lặp ta chỉ thêm vào một đỉnh  $u^*$  nên giả thiết là  $d(v)$  cho độ dài đường đi ngắn nhất từ  $s$  đến  $v$  với mọi  $v \in S_1$  là đúng với bước lặp đầu tiên. Theo qui nạp suy ra thuật toán cho ta đường đi ngắn nhất từ  $s$  đến mọi đỉnh của đồ thị.

Bây giờ ta sẽ đánh giá số phép toán cần thực hiện theo thuật toán. Ở mỗi bước lặp để tìm ra đỉnh  $u$  cần phải thực hiện  $O(n)$  phép toán, và để gán nhãn lại cũng cần thực hiện một số lượng phép toán cũng là  $O(n)$ . Thuật toán phải thực hiện  $n-1$  bước lặp, vì vậy thời gian tính toán của thuật toán cỡ  $O(n^2)$ .

Định lý được chứng minh.

Khi tìm được độ dài của đường đi ngắn nhất  $d[v]$  thì đường đi này có thể tìm dựa vào nhãn  $Truoc[v]$ ,  $v \in V$ , theo qui tắc giống như chúng ta đã xét trong chương 3.

**Thí dụ 2.** Tìm đường đi ngắn nhất từ 1 đến các đỉnh còn lại của đồ thị ở hình 2.



**Hình 2.** Minh họa thuật toán Dijkstra

Kết quả tính toán theo thuật toán được trình bày theo bảng dưới đây. Qui ước viết hai thành phần của nhãn theo thứ tự: d[v]. Đỉnh được đánh dấu \* là đỉnh được chọn để cố định nhãn ở bước lặp đang xét, nhãn của nó không biến đổi ở các bước tiếp theo, vì thế ta đánh dấu -.

Bước lặp	Đỉnh 1	Đỉnh 2	Đỉnh 3	Đỉnh 4	Đỉnh 5	Đỉnh 6
Khởi tạo	0,1	1,1*	$\infty$ ,1	$\infty$ ,1	$\infty$ ,1	$\infty$ ,1
1	-	-	6,2	3,2*	$\infty$ ,1	8,2
2	-	-	4,4*	-	7,4	8,2
3	-	-		-	7,4	5,3*
4	-	-		-	6,6*	-
5						

Chú ý:

Nếu chỉ cần tìm đường đi ngắn nhất từ s đến một đỉnh t nào đó thì có thể kết thúc thuật toán khi đỉnh t trở thành có nhãn cố định.

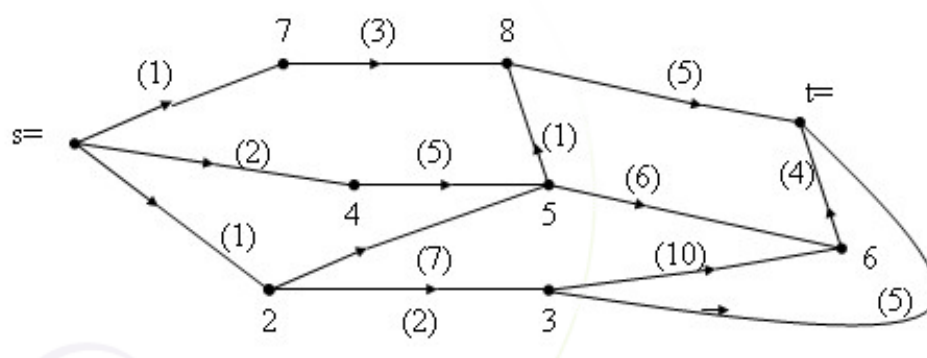
Tương tự như trong mục 2, dễ dàng mô tả thuật toán trong trường hợp đồ thị cho bởi danh sách kề. Để có thể giảm bớt khối lượng tính toán trong việc xác định đỉnh u ở mỗi bước lặp, có thể sử dụng thuật toán Heasort (tương tự như trong chương 5 khi thể hiện thuật toán Kruskal). Khi đó có thể thu được thuật toán với độ phức tạp tính toán là  $O(m \log n)$ .

# Đồ thị trong đồ thị không có chu trình

Bây giờ ta xét trường hợp riêng thứ hai của bài toán đường đi ngắn nhất, mà để giải nó có thể xây dựng thuật toán với độ phức tạp tính toán  $O(n^2)$ , đó là khi đồ thị không có chu trình (còn trọng số trên các cung có thể là các số thực tùy ý). Trước hết ta chứng minh định lý sau.

**Định lý 2.** Giả sử  $G$  là đồ thị không có chu trình. Khi đó các đỉnh của nó có thể đánh số sao cho mỗi cung của đồ thị chỉ hướng từ đỉnh có chỉ số nhỏ hơn đến đỉnh có chỉ số lớn hơn, nghĩa là mỗi cung của nó có sự biểu diễn dưới dạng  $(v[i], v[j])$ , trong đó  $i < j$ .

**Thí dụ 3.** Đồ thị trong hình 3 có các đỉnh số thỏa mãn điều kiện nêu trong định lý.



Hình 3. Đồ thị không có chu trình

Để chứng minh định lý ta mô tả thuật toán sau đây, cho phép tìm ra cách đánh số thỏa mãn điều kiện định lý.

Procudure Numbering;

(\* Đầu vào: Đồ thị có hướng  $G=(V,E)$  với  $n$  đỉnh không chứa chu trình được cho bởi danh sách kề  $Ke(v)$ ,  $v \in V$ .

Đầu ra:

Với mỗi đỉnh  $v \in V$  chỉ số  $NR[v]$  thỏa mãn:

Với mọi cung  $(u,v)$  của đồ thị ta đều có  $NR[u] < NR[v]$  \*)

Begin

For  $v \in V$  do  $Vao[v] := 0$ ;

(\* Tính  $Vao[v] = deg - (v)$  \*)

for  $u \in V$  do

for  $v \in Ke(u)$  do  $Vao[v] := Vao[v] + 1$ ;

Queue :=  $\emptyset$ ;

For  $v \in V$  do

if  $Vao[v] = 0$  then Queue  $\leftarrow v$ ;

num := 0;

while queue  $\neq \emptyset$  do

begin

$u \leftarrow queue$ ;

num := num + 1; NR[u] := num;

for  $v \in Ke(u)$  do

begin

$Vao[v] := Vao[v] - 1$ ;

If  $Vao[v] = 0$  then queue  $\leftarrow v$ ;

end;

end;

End;

Thuật toán được xây dựng dựa trên ý tưởng rất đơn giản sau: rõ ràng trong đồ thị không có chu trình bao giờ cũng tìm được đỉnh có bán bậc vào bằng 0 (không có cung đi vào). Thực vậy, bắt đầu từ đỉnh  $v_1$  nếu có cung đi vào nó từ  $v_2$  thì ta lại chuyển sang xét đỉnh  $v_2$ . Nếu có cung từ  $v_3$  đi vào  $v_2$ , thì ta lại chuyển sang xét đỉnh  $v_3$ . . . Do đồ thị không có chu trình nên sau một số hữu hạn lần chuyển như vậy ta phải đi đến đỉnh không có cung đi vào. Thoạt tiên, tìm các đỉnh như vậy của đồ thị. Rõ ràng ta có thể đánh số chúng theo thứ tự tùy ý bắt đầu từ 1. Tiếp theo, loại bỏ khỏi đồ thị những đỉnh đã được đánh số cùng các cung đi ra khỏi chúng, ta thu được đồ thị mới cũng không có chu trình, và

thủ tục được lặp với đồ thị mới này. Quá trình đó sẽ được tiếp tục cho đến khi tất cả các đỉnh của đồ thị được đánh số.

Chú ý:

Rõ ràng trong bước khởi tạo ra phải duyệt qua tất cả các cung của đồ thị khi tính bán bậc vào của các đỉnh, vì vậy ở đó ta tốn cỡ  $O(m)$  phép toán, trong đó  $m$  là số cung của đồ thị. Tiếp theo, mỗi lần đánh số một đỉnh, để thực hiện việc loại bỏ đỉnh đã đánh số cùng với các cung đi ra khỏi nó, chúng ta lại duyệt qua tất cả các cung này. Suy ra để đánh số tất cả các đỉnh của đồ thị chúng ta sẽ phải duyệt qua tất cả các cung của đồ thị một lần nữa. Vậy độ phức tạp của thuật toán là  $O(m)$ .

Thuật toán có thể áp dụng để kiểm tra xem đồ thị có chứa chu trình hay không? Thực vậy, nếu kết thúc thuật toán vẫn còn có đỉnh chưa được đánh số ( $\text{num} < n$ ) thì điều đó có nghĩa là đồ thị chứa chu trình.

Do có thuật toán đánh số trên, nên khi xét đồ thị không có chu trình ta có thể giả thiết là các đỉnh của nó được đánh số sao cho mỗi cung chỉ đi từ đỉnh có chỉ số nhỏ đến đỉnh có chỉ số lớn hơn. Thuật toán tìm đường đi ngắn nhất trên đồ thị không có chu trình được mô tả trong sơ đồ sau đây.

Procedure Critical\_Path;

(\* Tìm đường đi ngắn nhất từ đỉnh nguồn đến tất cả các đỉnh còn lại trên đồ thị không có chu trình \*)

Đầu vào:

Đồ thị  $G=(V,E)$ , trong đó  $V = \{v[1], v[2], \dots, v[n]\}$ .

Đối với mỗi cung  $(v[i], v[j]) \in E$ , ta có  $i < j$ .

Đồ thị được cho bởi danh sách kề  $Ke(v)$ ,  $v \in V$ .

Đầu ra:

Khoảng cách từ  $v[1]$  đến tất cả các đỉnh còn lại được ghi trong mảng  $d[v[i]]$ ,  $i = 2, 3, \dots, n$  \*)

Begin

$d[v[1]] := 0$ ;

for  $j := 2$  to  $n$  do  $d[v[j]] := a[v[1], v[j]]$ ;



for  $j:=2$  to  $n$  do

*for*  $v \in Ke[v[j]]$  *do*  $d[v] := \min(d[v], d[v[j]] + a[v[j], v]);$

End;

Độ phức tạp tính toán của thuật toán là  $O(m)$ , do mỗi cung của đồ thị phải xét qua đúng một lần.

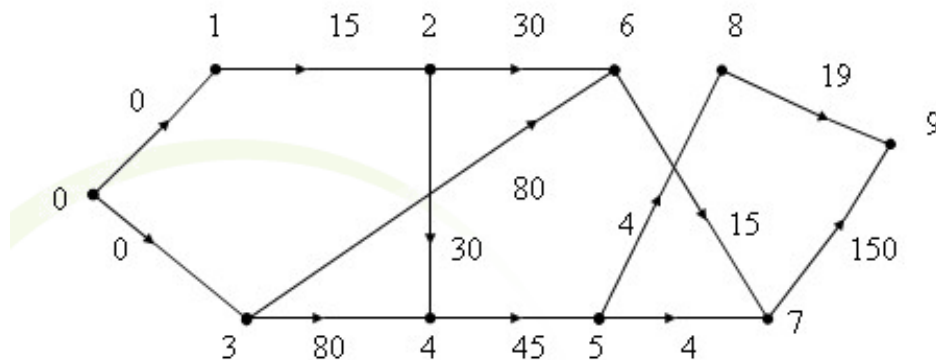
Các thuật toán được mô tả ở trên thường được ứng dụng vào việc xây dựng những phương pháp giải bài toán điều khiển việc thực hiện những dự án lớn, gọi tắt là PERT (Project Evaluation and Review Technique) hay CDM (Critical path Method). Một thí dụ đơn giản cho ứng dụng này được mô tả trong thí dụ dưới đây.

Công đoạn	$t[i]$	Các công đoạn phải được hoàn thành trước nó
1	15	Không có
2	30	1
3	80	Không có
4	45	2, 3
5	124	4
6	15	2, 3
7	15	5, 6
8	19	5

**Thí dụ 4.** Việc thi công một công trình lớn được chia thành  $n$  công đoạn, đánh số từ 1 đến  $n$ . Có một số công đoạn mà việc thực hiện nó chỉ được tiến hành sau khi một số công đoạn nào đó đã hoàn thành. Đối với mỗi công đoạn  $i$  biết  $t[i]$  là thời gian cần thiết để hoàn thành nó ( $i=1, 2, \dots, n$ ). Dữ liệu với  $n=8$  được cho trong bảng dưới đây

Giả sử thời điểm bắt đầu tiến hành thi công công trình là 0. Hãy tìm tiến độ thi công công trình (chỉ rõ mỗi công đoạn phải được bắt đầu thực hiện vào thời điểm nào) cho công trình được hoàn thành xong trong thời điểm sớm nhất có thể được.

Ta có thể xây dựng đồ thị có hướng  $n$  đỉnh biểu diễn hạn chế về trình tự thực hiện các công việc như sau: Mỗi đỉnh của đồ thị tương ứng với một công việc, nếu công việc  $i$  phải được thực hiện trước công đoạn  $j$  thì trên đồ thị có cung  $(i,j)$ , trọng số trên cung này được gán bằng  $t[i]$ , xem hình 4 dưới đây.



Hình 4. Đồ thị minh họa PERT

Thêm vào đồ thị hai đỉnh 0 và  $n+1$  tương ứng với hai sự kiện đặc biệt: đỉnh 0 tương ứng với công đoạn lễ khởi công, nó phải được thực hiện trước tất cả các công đoạn khác, và đỉnh  $n+1$  tương ứng với công đoạn cắt băng khánh thành công trình, nó phải được thực hiện sau các công đoạn, với  $t[0]=t[n+1]=0$  (trên thực tế chỉ cần nối đỉnh 0 với tất cả các đỉnh có bán bậc bằng 0 và nối tất cả các đỉnh có bán bậc ra bằng 0 với đỉnh  $n+1$ ). Gọi đồ thị thu được là  $G$ . Rõ ràng bài toán đặt ra dẫn về bài toán tìm đường đi ngắn nhất từ đỉnh 0 đến tất cả các đỉnh còn lại trên đồ thị  $G$ . Do đồ thị  $G$  rõ ràng là không chứa chu trình, nên để giải bài toán đặt ra có thể áp dụng các thuật toán mô tả trên, chỉ cần đổi dấu tất cả các trọng số trên các cung thành dấu ngược lại, hoặc đơn giản hơn chỉ cần đổi toán tử Min trong thuật toán Critical\_Path thành toán tử Max. Kết thúc thuật toán, chúng ta thu được  $d[v]$  là độ dài đường đi dài nhất từ đỉnh 0 đến đỉnh  $v$ . Khi đó  $d[v]$  cho ta thời điểm sớm nhất có thể bắt đầu thực hiện công đoạn  $v$ , nói riêng  $d[n+1]$  là thời điểm sớm nhất có thể cắt băng khánh thành, tức là thời điểm sớm nhất có thể hoàn thành toàn bộ công trình.

Cây đường đi dài nhất của bài toán trong thí dụ 4 tìm được theo thuật toán được chỉ ra trong hình 4.

# Đường đi ngắn nhất giữa tất cả các cặp đỉnh

Rõ ràng ta có thể giải bài toán tìm đường đi ngắn nhất giữa tất cả các cặp đỉnh của đồ thị bằng cách sử dụng  $n$  lần thuật toán mô tả ở mục trước, trong đó ta sẽ chọn  $s$  lần lượt là các đỉnh của đồ thị. Rõ ràng, khi đó ta thu được thuật toán với độ phức tạp  $O(n^4)$  (nếu sử dụng thuật toán Ford\_Bellman) hoặc  $O(n^3)$  đối với trường hợp trọng số không âm hoặc đồ thị không có chu trình. Trong trường hợp tổng quát, sử dụng thuật toán Ford\_Bellman  $n$  lần không phải là cách làm tốt nhất. Ở đây ta sẽ mô tả một thuật toán giải bài toán trên với độ phức tạp tính toán  $O(n^3)$ : thuật toán Floyd. Thuật toán được mô tả trong thủ tục sau đây.

Procedure Floyd;

(\* Tìm đường đi ngắn nhất giữa tất cả các cặp đỉnh

Đầu vào: Đồ thị cho bởi ma trận trọng số  $a[i,j]$ ,  $i, j = 1, 2, \dots, n$ .

*Đầu ra:*

Ma trận đường đi ngắn nhất giữa các cặp đỉnh

$d[i,j]$ ,  $i, j = 1, 2, \dots, n$ ,

trong đó  $d[i,j]$  cho độ dài đường đi ngắn nhất từ đỉnh  $i$  đến đỉnh  $j$ .

Ma trận ghi nhận đường đi

$p[i,j]$ ,  $i, j = 1, 2, \dots, n$ ,

trong đó  $p[i,j]$  ghi nhận đỉnh đi trước đỉnh  $j$  trong đường đi ngắn nhất từ  $i$  đến  $j$ .

\*)

begin

(\* Khởi tạo \*)

for  $i:=1$  to  $n$  do

for  $j:=1$  to  $n$  do

```

begin
d[i,j]:=a[i,j];
p[i,j]:=i;
end;

(* Bước lặp *)
for k:=1 to n do
for i:=1 to n do
for j:=1 to n do
if d[i,j]>d[i,k]+d[k,j] then
begin
d[i,j]:=d[i,k]+d[k,j];
p[i,j]:=p[k,j];
end;
end;
end;

```

Rõ ràng độ phức tạp tính toán của thuật toán là  $O(n^3)$ .

Kết thúc chương này chúng ta trình bày một cách thể hiện thuật toán Dijkstra trên ngôn ngữ Pascal:

*(\* CHƯƠNG TRÌNH TÌM ĐƯỜNG ĐI NGẮN NHẤT TỪ ĐỈNH S ĐẾN ĐỈNH T THEO THUẬT TOÁN DIJKSTRA \*)*

```

uses crt;

const max=50;

var

```

```

n, s, t:integer;

chon:char;

Truoc:array[1..max] of byte;

d: array[1..max] of integer;

a: array[1..max,1..max] of integer;

final: array[1..max] of boolean;

procedure Nhapsolieu;

var

f:text;

fname:string;

i,j:integer;

begin

write('Vao ten file du lieu can doc:');

readln(fname);

assign(f,fname);

reset(f);

readln(f,n);

for i:=1 to n do

for j:=1 to n do read(f, a[i,j]);

close(f);

end;

procedure Insolieu;

```

```

var
i,j:integer;

begin
writeln('So dinh cua do thi:',n);
writeln('Ma tran khoang cach:');
for i:=1 to n do
begin
for j:=1 to n do write(a[i,j]:3,' ');
writeln;
end;
end;

```

Procedure Inketqua;

```

Var
i,j:integer;

begin
writeln('Duong di ngan nhat tu 's,' den ',t);
write(t,' ← ');
while i<> s do
begin
i:=Truoc[i];
write(i,' ← ');
end;

```

```

end;

Procedure Dijkstra;

Var

U,v,minp:integer;

Begin

Write('Tim duon di tu s=');Readln(s);Write(' den t=');Readln(t);

For v:=1 to n doBegin

d[v]:=a[s,v];

Truoc[v]:=s;

Filal[v]:=false;

End;

Truoc[s]:=0;D[s]:=0;Final[s]:=true;

While not final[t] do (* Buoc lap *)Begin

Tim u la dinh co nhan tam thoi nho nhat

minp:=maxint;

for v:=1 to n do

if (not final[v]) and minp>d[v]) then

begin

u:=v;

minp:=d[v];

end;

final[u]:=true;

```

```

if not final[t] then
  for v:=1 to n do
    if (not final[v]) and (d[u]+a[u,v]<d[v]) then
      begin
        d[v]:=d[u]+a[u,v];
        Truoc[v]:=u;
      end;
    End;
  end;
end;

Procedure Menu;

Begin
  Clrscr;

  Writeln('1. Nhap du lieu tu file');
  Writeln('2. Giai bai toan');
  Writeln('3. Ket thuc');
  Writeln('-----');
  Write('Hay chon chuc nang:');

  End;

(* Chuong trinh chinh *)

Begin
  Repeat
    Menu;

```



Chon:=readkey;

Writeln(chon);

Case chon of

‘1’:Nhapsolieu;

‘2’: begin

Insolieu;

Dijkstra;

Inketqua;

‘3’:exit;

end;

Until false;

End.

# Bài toán luồng cực đại trong mạng

Bài toán luồng cực đại trong mạng là một trong số bài toán tối ưu trên đồ thị tìm được những ứng dụng rộng rãi trong thực tế cũng như những ứng dụng thú vị trong lý thuyết tổ hợp. Bài toán được đề xuất vào đầu năm 1950, và gắn liền với tên tuổi của hai nhà toán học Mỹ là Ford và Fulkerson. Trong chương này chúng ta sẽ trình bày thuật toán Ford và Fulkerson để giải bài toán đặt ra và nêu một số ứng dụng của bài toán.

## MẠNG. LUỒNG TRONG MẠNG. BÀI TOÁN LUỒNG CỰC ĐẠI

**Định nghĩa 1.** Ta gọi mạng là đồ thị có hướng  $G=(V,E)$ , trong đó duy nhất một đỉnh  $s$  không có cung đi vào gọi là đỉnh phát, duy nhất một đỉnh  $t$  không có cung đi ra gọi là điểm thu và mỗi cung  $e=(v,w) \in E$  được gán với một số không âm  $c(e)=c(v,w)$  gọi là khả năng thông qua của cung  $e$ .

Để thuận tiện cho việc trình bày ta sẽ qui ước rằng nếu không có cung  $(v,w)$  thì khả năng thông qua  $c(v,w)$  được gán bằng 0.

**Định nghĩa 2.** Giả sử cho mạng  $G=(V,E)$ . Ta gọi mạng  $f$  trong mạng  $G=(V,E)$  ; là ánh xạ  $f: E \rightarrow \mathbb{R}_+$  gán cho mỗi cung  $e=(v,w) \in E$  một số thực không âm  $f(e)=f(v,w)$ , gọi là luồng trên cung  $e$ , thỏa mãn các điều kiện sau:

Luồng trên cung  $e \in E$  không vượt quá khả năng thông qua của nó:

$$0 \leq f(e) \leq c(e),$$

Điều kiện cân bằng luồng trên mỗi đỉnh của mạng: Tổng luồng trên các cung đi vào đỉnh  $v$  bằng tổng luồng trên các cung đi ra khỏi đỉnh  $v$ , nếu  $v \neq s, t$ :

$$\text{Div } f(v) = \sum_{w \in \Gamma^-(v)} f(w,v) - \sum_{w \in \Gamma^+(v)} f(v,w) = 0$$

trong đó  $\Gamma^-(v)$  – tập các đỉnh của mạng mà từ đó có cung đến  $v$ ,  $\Gamma^+(v)$  – tập các đỉnh của mạng mà từ  $v$  có cung đến nó:

$$\Gamma^-(v) = \{ w \in V : (w,v) \in E \},$$

$$\Gamma^+(v) = \{ w \in V : (v,w) \in E \}.$$

Giá trị của luồng  $f$  là số

$$\text{Val}(f) = \sum_{w \in \Gamma^+(s)} f(s, w) = \sum_{w \in \Gamma^-(t)} f(w, t).$$

$$w \in \Gamma^+(s) \quad w \in \Gamma^-(t)$$

### **Bài toán luồng cực đại trong mạng:**

Cho mạng  $G(V, E)$ . Hãy tìm luồng  $f^*$  trong mạng với giá trị luồng  $\text{val}(f^*)$  là lớn nhất. Luồng như vậy ta sẽ gọi là luồng cực đại trong mạng.

Bài toán như vậy có thể xuất hiện trong rất nhiều ứng dụng thực tế. Chẳng hạn khi cần xác định cường độ lớn nhất của dòng vận tải giữa hai nút của một bản đồ giao thông. Trong ví dụ này lời giải của bài toán luồng cực đại sẽ chỉ cho ta các đoạn đường đông xe nhất và chúng tạo thành "chỗ hẹp" tương ứng với dòng giao thông xét theo hai nút được chọn. Một ví dụ khác là nếu xét đồ thị tương ứng với một hệ thống đường ống dẫn dầu. Trong đó các ống tương ứng với các cung, điểm phát có thể coi là tàu chở dầu, điểm thu là bể chứa, còn những điểm nối giữa các ống là các nút của đồ thị. Khả năng thông qua của các cung tương ứng với tiết diện của các ống. Cần phải tìm luồng dầu lớn nhất có thể bơm từ tàu chở dầu vào bể chứa.

# Lát cắt. Đường tăng luồng. Định lý Ford\_Fulkerson

**Định nghĩa 3.** Ta gọi lát cắt  $(X, X^*)$  là một cách phân hoạch tập đỉnh  $V$  của mạng ra thành hai tập  $X$  và  $X^* = V \setminus X$ , trong đó  $s \in X$ ,  $t \in X^*$ . Khả năng thông qua của lát cắt  $(X, X^*)$  là số

$$c(X, X^*) = \sum_{v \in X, w \in X^*} c(v, w)$$

Lát cắt với khả năng thông qua nhỏ nhất được gọi là lát cắt hẹp nhất.

## Bổ đề 1.

Giá trị của luồng  $f$  trong mạng luôn nhỏ hơn hoặc bằng khả năng thông qua của lát cắt  $(X, X^*)$  bất kỳ trong nó:  $\text{val}(f) \leq c(X, X^*)$ .

**Chứng minh.** Cộng các điều kiện cân bằng luồng  $\text{Divf}(v) = 0$  với mọi  $v \in X$ . Khi đó ta có

$$\sum_{w \in \Gamma^-(v)} f(w, v) - \sum_{w \in \Gamma^+(v)} f(v, w) = -\text{val}(f) \quad v \in X$$

tổng này sẽ gồm các số hạng dạng  $f(u, v)$  với dấu cộng hoặc dấu trừ mà trong đó có ít nhất một trong hai đỉnh  $u, v$  phải thuộc tập  $X$ . Nếu cả hai đỉnh  $u, v$  đều trong tập  $X$ , thì  $f(u, v)$  xuất hiện với dấu cộng trong  $\text{Divf}(v)$  và với dấu trừ trong  $\text{Divf}(u)$ , vì thế, chúng triệt tiêu lẫn nhau. Do đó, sau khi giản ước các số hạng như vậy ở vế trái, ta thu được

$$-\sum_{v \in X, w \in X^*} f(v, w) + \sum_{v \in X^*, w \in X} f(v, w) = -\text{val}(f), \quad v \in X, w \in X^*$$

hay là

$$\text{val}(f) = \sum_{v \in X, w \in X^*} f(v, w) - \sum_{v \in X^*, w \in X} f(v, w)$$

Mặt khác, từ điều kiện 1 rõ ràng là

$$\sum_{v \in X, w \in X^*} f(v, w) \leq \sum_{v \in X, w \in X^*} c(v, w)$$

Còn

$$- \sum f(v,w) \leq 0 \quad v \in X^* \quad w \in X$$

suy ra  $\text{val}(f) \leq c(X, X^*)$ . Bổ đề được chứng minh.

**Hệ quả 1.** Giá trị luồng cực đại trong mạng không vượt quá khả năng thông qua của lát cắt hẹp nhất trong mạng. Ford và Fulkerson đã chứng minh rằng giá trị luồng cực đại trong mạng đúng bằng khả năng thông qua của lát cắt hẹp nhất. Để có thể phát biểu và chứng minh kết quả này chúng ta sẽ cần thêm một số khái niệm. Giả sử  $f$  là một luồng trong mạng  $G = (V, E)$ . Từ mạng  $G = (V, E)$  ta xây dựng đồ thị có trọng số trên cung  $G_f = (V, E_f)$ , với tập cung  $E_f$  và trọng số trên các cung được xác định theo qui tắc sau:

Nếu  $e = (v, w) \in E$  với  $f(v, w) = 0$ , thì  $(v, w) \in E_f$  với trọng số  $c(v, w)$ ;

Nếu  $e = (v, w) \in E$  với  $f(v, w) = c(v, w)$ , thì  $(w, v) \in E_f$  với trọng số  $f(v, w)$ ;

Nếu  $e = (v, w) \in E$  với  $0 < f(v, w) < c(v, w)$ , thì  $(v, w) \in E_f$  với trọng số  $c(v, w) - f(v, w)$  và  $(w, v) \in E_f$  với trọng số  $f(v, w)$ .

Các cung của  $G_f$  đồng thời cũng là cung của  $G$  được gọi là cung thuận, các cung còn lại được gọi là cung nghịch. Đồ thị  $G_f$  được gọi là đồ thị tăng luồng.

**Thí dụ:** Các số viết cạnh các cung của  $G$  ở hình 1 theo thứ tự là khả năng thông qua và luồng trên cung.

Dễ dàng kiểm tra được rằng  $f'$  được xây dựng như trên là luồng trong mạng và  $\text{val}(f') = \text{val}(f) + \delta$ . Ta sẽ gọi thủ tục biến đổi luồng vừa nêu là tăng luồng dọc theo đường  $P$ .

**Định nghĩa 4.** Ta gọi đường tăng luồng  $f$  là mọi đường đi từ  $s$  đến  $t$  trên đồ thị tăng luồng  $G(f)$ .

**Định lý 1.** Các mệnh đề dưới đây là tương đương:

- (i)  $f$  là luồng cực đại trong mạng;
- (ii) không tìm được đường tăng luồng  $f$ ;
- (iii)  $\text{val}(f) = c(X, X^*)$  với một lát cắt  $(X, X^*)$  nào đó.

Chứng minh.

(i)  $\Rightarrow$  (ii). Giả sử ngược lại, tìm được đường tăng luồng P. Khi đó ta có thể tăng giá trị luồng bằng cách tăng luồng dọc theo đường P. Điều đó mâu thuẫn với tính cực đại của luồng f.

(ii)  $\Rightarrow$  (iii). Giả sử không tìm được đường tăng luồng. Ký hiệu X là tập tất cả các đỉnh có thể đến được từ đỉnh s trong đồ thị Gf, và đặt  $X^* = V \setminus X$ . Khi đó  $(X, X^*)$  là lát cắt, và  $f(v, w) = 0$  với mọi  $v \in X^*, w \in X$  nên

$$\text{val}(f) = \sum_{v \in X} \sum_{w \in X^*} f(v, w) - \sum_{v \in X^*} \sum_{w \in X} f(v, w) = \sum_{v \in X} \sum_{w \in X^*} f(v, w) \quad v \in X \quad v \in X^* \quad v \in X \quad w \in X^*$$

Với  $v \in X, w \in X^*$ , do  $(v, w) \notin Gf$ , nên  $f(v, w) = c(v, w)$ . Vậy

$$\text{val}(f) = \sum_{v \in X} \sum_{w \in X^*} f(v, w) = \sum_{v \in X} \sum_{w \in X^*} c(v, w) = c(X, X^*) \quad v \in X \quad v \in X^* \quad w \in X^* \quad w \in X^*$$

(iii)  $\Rightarrow$  (i). Theo bổ đề 1,  $\text{val}(f) \leq c(X, X^*)$  với mọi luồng f và với mọi lát cắt  $(X, X^*)$ . Vì vậy, từ đẳng thức  $\text{val}(f) = c(X, X^*)$  suy ra luồng f là luồng cực đại trong mạng.

# Thuật toán tìm luồng cực đại

Định lý 1 là cơ sở để xây dựng thuật toán lặp sau đây để tìm luồng cực đại trong mạng: Bắt đầu từ luồng với luồng trên tất cả các cung bằng 0 (ta sẽ gọi luồng như vậy là luồng không), và lặp lại bước lặp sau đây cho đến khi thu được luồng mà đối với nó không còn đường tăng:

Bước lặp tăng luồng (Ford-Fulkerson): Tìm đường tăng P đối với luồng hiện có. Tăng luồng dọc theo đường P.

Khi đã có luồng cực đại, lát cắt hẹp nhất có thể theo thủ tục mô tả trong chứng minh định lý 1. Sơ đồ của thuật toán Ford-Fulkerson có thể mô tả trong thủ tục sau đây:

Procedure Max\_Flow;

(\* Thuật toán Ford-Fulkerson \*)

begin

(\* Khởi tạo: Bắt đầu từ luồng với giá trị 0 \*)

for  $u \in V$  do

for  $v \in V$  do  $f(u,v) := 0$ ;

stop := false;

while not stop do

if <Tìm được đường tăng luồng P> then

<Tăng luồng dọc theo P>

else stop := true;

end;

Để tăng luồng trong Gf có thể tìm kiếm thuật toán theo chiều rộng (hay tìm kiếm theo chiều sâu) bắt đầu từ đỉnh s, trong đó không cần xây dựng đồ thị tương minh Gf. Ford\_Fulkerson đề nghị thuật toán gán nhãn chi tiết sau đây để giải bài toán luồng cực đại trong mạng (có thể bắt đầu từ luồng không), sau đó ta sẽ tăng luồng bằng cách tìm các đường tăng luồng. Để tìm đường tăng luồng ta sẽ áp dụng phương pháp gán nhãn

cho các đỉnh. Mỗi đỉnh trong quá trình thực hiện thuật toán sẽ ở một trong ba trạng thái: chưa có nhãn, có nhãn chưa xét, có nhãn đã xét. Nhãn của một đỉnh  $v$  gồm 2 phần và có một trong 2 dạng sau:  $[+p(v), \epsilon(v)]$  hoặc  $[-p(v), \epsilon(v)]$ . Phần thứ nhất  $+p(v)$  ( $-p(v)$ ) chỉ ra là cần tăng (giảm) luồng theo cung  $(p(v), v)$  (cung  $v, p(v)$ ) còn phần thứ hai  $\epsilon(v)$  chỉ ra lượng lớn nhất có thể tăng hoặc giảm luồng trên các cung của đường tăng luồng từ  $s$  tới  $v$ . Đầu tiên chỉ có đỉnh  $s$  được khởi tạo và nhãn của nó là chưa xét, còn tất cả các đỉnh còn lại là đều chưa có nhãn. Từ  $s$  ta gán nhãn cho tất cả các đỉnh kề với nó và nhãn của đỉnh  $s$  sẽ trở thành đã xét. Tiếp theo, từ mỗi đỉnh  $v$  có nhãn chưa xét ta lại gán nhãn cho tất cả các đỉnh chưa có nhãn kề nó và nhãn của đỉnh  $v$  trở thành đã xét. Quá trình sẽ lặp lại cho đến khi hoặc là đỉnh  $t$  trở thành có nhãn hoặc là nhãn của tất cả các đỉnh có nhãn đều là đã xét nhưng đỉnh  $t$  vẫn không có nhãn. Trong trường hợp thứ nhất ta tìm được đường tăng luồng, còn trong trường hợp thứ hai đối với luồng đang xét không tồn tại đường tăng luồng (tức là luồng đã là cực đại). Mỗi khi ta tìm được đường tăng luồng, ta lại tăng luồng theo đường tìm được, sau đó xoá tất cả nhãn và đối với luồng mới thu được lại sử dụng phép gán nhãn các đỉnh để tìm đường tăng luồng. Thuật toán sẽ kết thúc khi nào đối với luồng đang có trong mạng không tìm được đường tăng luồng.

Hai thủ tục tìm đường tăng luồng và tăng luồng có thể mô tả như sau.

Procedure Find\_Path;

(\* Thủ tục gán nhãn tìm đường tăng luồng

$p[v], \epsilon[v]$  là nhãn của đỉnh  $v$ ;

VT – danh sách các đỉnh có nhãn nhưng chưa xét;

$c[u,v]$  – khả năng thông qua của cung  $(u,v)$ ,  $u, v \in V$ ;

$f[u, v]$  – luồng trên cung  $(u, v)$ ,  $u, v \in V$ . \*)

begin

$p[s] := s$ ;

$\epsilon[s] := +\infty$  ;

VT = {  $s$  } ;

PathFound := true;

While VT  $\neq \emptyset$  do

Begin



```

U  $\leftarrow$  VT; (* Lấy u từ VV *)

For v  $\in$  VTdo

begin

If (f[u,v] < c[u,v] then

Begin

p[v] := u;

 $\varepsilon[v] := \min \{ \varepsilon[u], c[u,v] - f[u,v] \}$ ;

VT = VT  $\cup$  { v } ; (* Nạp v vào danh sách đỉnh có nhãn *)

If v = t then exit;

Else

If (f[v,u] > 0) then

Begin

p[v] := u;

 $\varepsilon[v] := \min \{ \varepsilon[u], f[v,u] \}$  ;

VT = VT  $\cup$  { v } ; (* Nạp v vào danh sách đỉnh có nhãn *)

If v = t then exit;

End;

End;

End;

PathFound := false;

end;

Procedure Inc_Flow;

```

(\* Tăng luồng theo đường tăng \*)

begin

$v := p[t]$ ;  $u := t$ ;  $tang := \epsilon[t]$ ;

while  $u \neq s$  do

begin

if  $v > 0$  then  $f[v, u] := f[v, u] + tang$ ; else begin

$v := -v$ ;  $f[u, v] := f[u, v] - tang$ ;

end;  $u := v$ ;  $v := p[u]$ ;

end;

end;

Thuật toán Ford\_Fulkerson được thực hiện nhờ thủ tục:

Procedure Max\_Flow;

(\*Thuật toán Ford\_Fulkerson \*)

begin

(\* Khởi tạo: Bắt đầu từ luồng với giá trị 0 \*)

for  $u \in V$  do

    for  $v \in V$  do  $f[u, v] := 0$ ;

stop := false;

while not stop do

begin

find\_path;

if pathFound then Inc\_Flow

else stop:=true;

end;

<Luồng cực đại trong mạng là  $f[u,v]$ ,  $u, v \in V$  >

<Lát cắt hẹp nhất là  $(VT, V \setminus VT)$ >

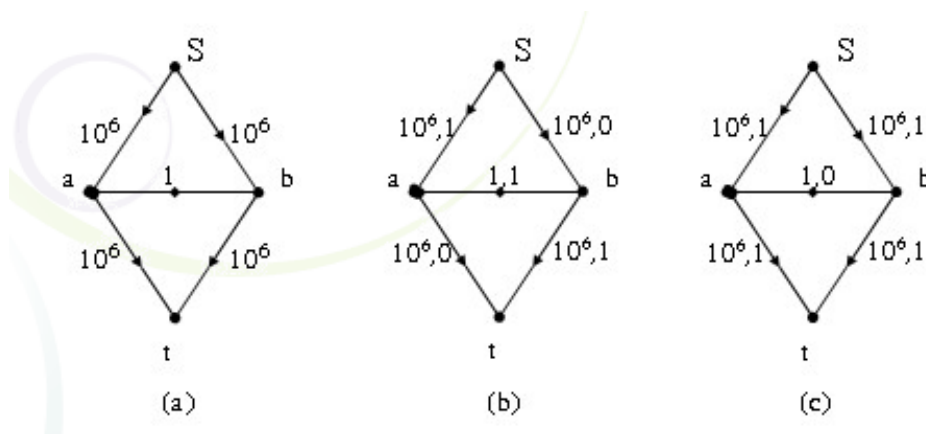
end;

Giả sử là khả năng thông qua của tất cả các khung của đồ thị là các số nguyên. Khi đó sau mỗi lần tăng luồng, giá trị luồng sẽ tăng lên ít nhất là 1. Từ đó suy ra thuật toán Ford\_Fulkerson sẽ dừng sau không quá  $val(f^*)$  lần tăng luồng và cho ta luồng cực đại trong mạng. Đồng thời, rõ ràng  $f^*(u,v)$  sẽ là số nguyên đối với mỗi cung  $(u,v) \in E$ . Từ đó ra có các kết quả sau:

**Định lý 2** (Định lý về luồng cực đại trong mạng và lát cắt hẹp nhất). Luồng cực đại trong mạng bằng khả năng thông qua của lát cắt hẹp nhất.

**Định lý 3** (Định lý về tính nguyên). Nếu tất cả các khả năng thông qua là các số nguyên thì luồng tìm được cực đại với luồng trên các cung là các số nguyên.

Tuy nhiên, nếu các khả năng thông qua là các số rất lớn thì giá trị của luồng cực đại cũng có thể là rất lớn và khi đó thuật toán mô tả ở trên sẽ đòi hỏi thực hiện rất nhiều bước tăng luồng. Thí dụ trong hình 2 sẽ minh họa cho điều này. Hình 2(a) mô tả mạng cần xét với các khả năng thông qua trên các cung. Hình 2(b) mô tả luồng trên các cung (số thứ hai bên cạnh cung) sau khi thực hiện tăng luồng dọc theo đường tăng luồng  $(s, a, b, t)$ . Hình 2(c) mô tả luồng trên các cung sau khi thực hiện tăng luồng dọc theo đường tăng luồng  $(s, b, a, t)$ . Rõ ràng, sau  $2 \cdot 10^6$  lần tăng luồng theo đường  $(s, a, b, t)$  và  $(s, b, a, t)$  một cách luân phiên ta thu được luồng cực đại.



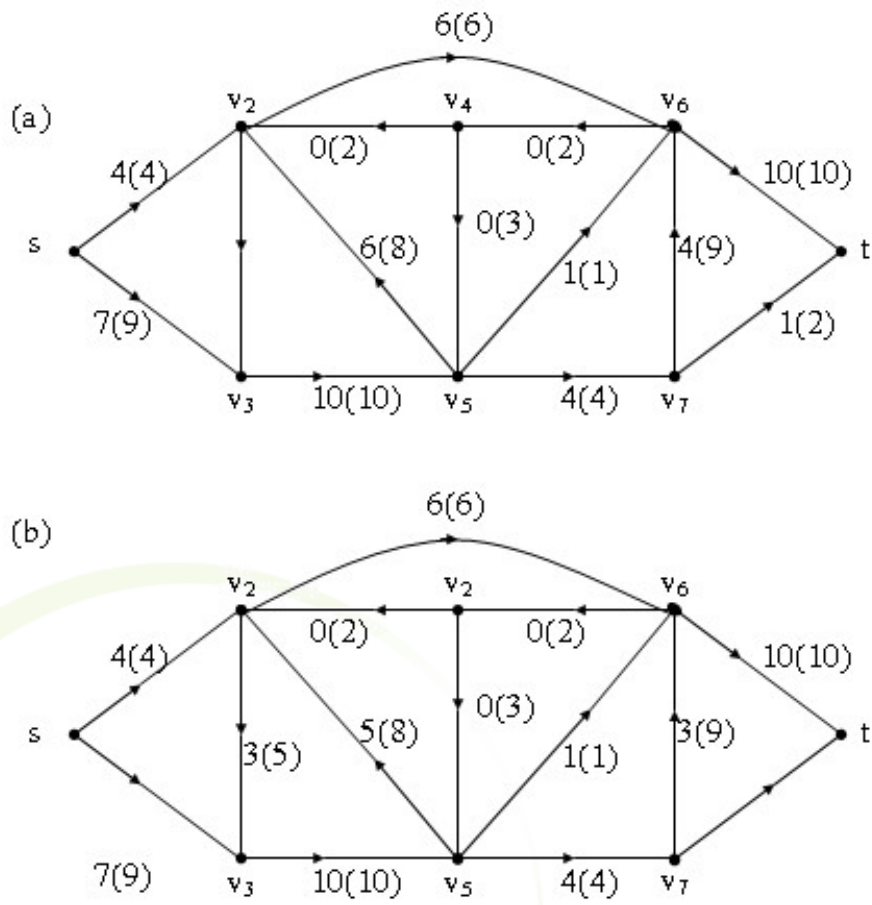
Hình 2. Ví dụ tồi tệ đối với thuật toán Ford\_Fulkerson.

Hơn thế nữa, nếu các khả năng thông qua là các số vô tỉ, người ta còn xây dựng được ví dụ để cho thuật toán không dừng, và tệ hơn nếu dãy các giá trị luồng xây dựng theo thuật toán hội tụ thì nó còn không hội tụ đến giá trị luồng cực đại. Như vậy, muốn thuật toán làm việc hiệu quả, việc lựa chọn đường tăng luồng cần được tiến hành hết sức cẩn thận.

Edmonds và Karp chỉ ra rằng nếu đường tăng luồng được chọn là đường ngắn nhất từ  $s$  đến  $t$  trên đồ thị  $G_f$ . Điều đó có thể thực hiện, nếu trong thủ tục tìm đường tăng  $\text{Find\_Path}$  mô tả ở trên, danh sách VT được tổ chức dưới dạng QUEUE (nghĩa là ta thực hiện tìm đường tăng bởi thủ tục tìm kiếm theo chiều rộng) thì thuật toán sẽ kết thúc sau không quá  $mn/2$  lần sử dụng đường tăng luồng. Nếu để ý rằng, tìm kiếm theo chiều rộng trên đồ thị đòi hỏi thời gian  $O(m+n)$ , thì thuật toán thu được sẽ có độ phức tạp tính toán là  $O(nm^2)$ .

Nhờ cách tổ chức tìm đường tăng khéo léo hơn, người ta đã xây dựng được thuật toán với độ phức tạp tính toán tốt hơn như:  $O(n^2m)$  (Dinic, 1970).  $O(n^3)$  (Karzanov, 1974),  $O(n^2m^2)$ , (Cherkasky, 1977),  $O(nm \log n)$  (Sleator, - Tarjan, 1980).

Ta kết thúc mục này bởi ví dụ minh họa cho thuật toán Ford\_Fulkerson sau đây. Hình 3(a) cho mạng  $G$  cùng với thông qua của tất cả các cung và luồng giá trị 10 trong nó. Hai số viết bên cạnh mỗi cung là khả năng thông qua của cung (số trong ngoặc) và luồng trên cung. Đường tăng luồng có dạng  $(s, v_3, v_2, v_6, v_7, t)$ . Ta tính ước  $\varepsilon(t) = 1$ , giá trị luồng tăng từ 10 lên 11. Hình 3 (b) cho luồng thu được sau khi tăng luồng theo đường tăng tìm được.



Hình 3. Tăng luồng dọc theo đường tăng

Luồng trong hình 3 (b) đã là cực đại. Lát cắt hẹp nhất

$$X = \{s, v_2, v_3, v_5\}, X^c = \{v_4, v_6, v_7, t\}.$$

Giá trị luồng cực đại là 11.

# Một số bài toán luồng tổng quát

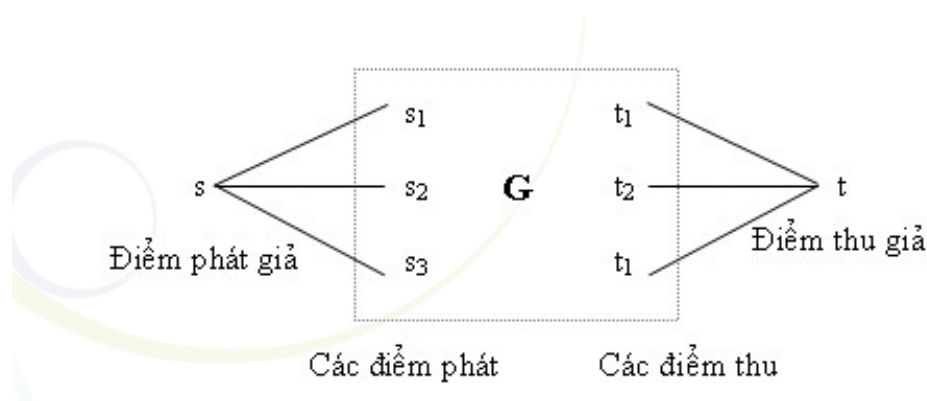
Trong phần này ta nêu ra một số dạng bài toán về luồng tổng quát mà việc giải chúng có thể dẫn về bài toán luồng cực đại trình bày ở trên.

## Mạng với nhiều điểm phát và điểm thu.

Xét mạng  $G$  với  $p$  điểm phát  $s_1, s_2, \dots, s_p$  và  $q$  điểm thu  $t_1, t_2, \dots, t_q$ . Giả sử rằng luồng có thể đi từ một điểm phát bất kỳ đến tất cả các điểm thu. Bài toán tìm luồng cực đại từ các điểm phát đến các điểm thu có thể đưa về bài toán với một điểm phát và một điểm thu bằng cách đưa vào một điểm phát giả  $s$  và một điểm thu giả  $t$  và cạnh nối  $s$  với tất cả các điểm phát và cạnh nối các điểm thu với  $t$ .

Hình 4 minh họa cho cách đưa mạng với nhiều điểm phát và nhiều điểm thu về mạng chỉ có một điểm phát và một điểm thu. Khả năng thông qua của cung nối  $s$  với điểm phát  $s_k$  sẽ bằng  $+\infty$  nếu không có hạn chế về lượng phát của điểm phát  $s_k$ , và nếu lượng phát của  $s_k$  bị hạn chế bởi  $b_k$  thì cung  $(s, s_k)$  có khả năng thông qua là  $b_k$ . Cũng như vậy, đối với các cung nối  $t_k$  với điểm thu  $t$ , giả sử khả năng thông qua của  $(t_k, t)$  sẽ là giới hạn hoặc không giới hạn tùy theo lượng thu của điểm thu này có bị giới hạn hay không.

Trường hợp một số điểm thu chỉ nhận "hàng" từ một số điểm phát ta có bài toán nhiều luồng là một bài toán phức tạp hơn rất nhiều so với bài toán luồng cực đại giữa điểm phát  $s$  và điểm thu  $t$ .



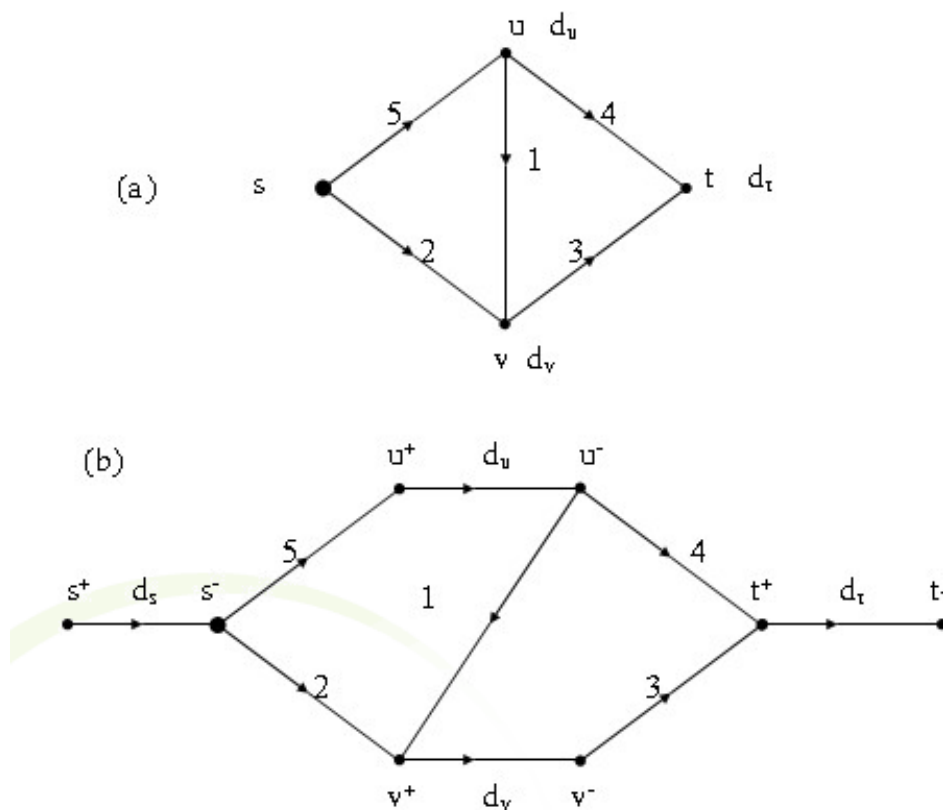
**Hình 4.** Mạng với nhiều điểm phát và thu.

## Bài toán với khả năng thông qua của các cung và các đỉnh.

Giả sử trong đồ thị  $G$ , ngoài khả năng thông qua của các cung  $c(u,v)$ , ở mỗi đỉnh  $v \in V$  còn có khả năng thông qua các đỉnh là  $d(v)$ , và đòi hỏi tổng luồng đi vào đỉnh  $v$  không được vượt quá  $d(v)$ , tức là.

$$f(w,v) \leq d(v) \quad v \in V$$

Cần phải tìm luồng cực đại giữa  $s$  và  $t$  trong mạng như vậy. Xây dựng một mạng  $G'$  sao cho: mỗi đỉnh  $v$  của  $G$  tương ứng với 2 đỉnh  $v^+, v^-$  trong  $G'$ , mỗi cung  $(u,v)$  trong  $G$  ứng với cung  $(u^+, v^+)$  trong  $G'$ , mỗi cung  $(v, e)$  trong  $G$  ứng với mỗi cung  $(v^-, w^+)$  trong  $G'$ . Ngoài ra, mỗi cung  $(v^+, v^-)$  trong  $G'$  có khả năng thông qua là  $d(v)$ , tức là bằng khả năng thông qua của đỉnh  $v$  trong  $G$ . Do luồng đi vào đỉnh  $v^+$  phải đi qua cung  $(v^+, v^-)$  với khả năng thông qua  $d(v)$ , nên luồng cực đại trong  $G'$  sẽ bằng luồng cực đại trong  $G$  với khả năng thông qua của các cung và các đỉnh.



Hình 5. Hình 5a cho ví dụ mạng  $G$  với khả năng thông qua ở cung và đỉnh.

Hình 5b là mạng  $G'$  tương ứng chỉ có khả năng thông qua trên các cung.

## Mạng trong đó khả năng thông qua của mỗi cung bị chặn hai phía.

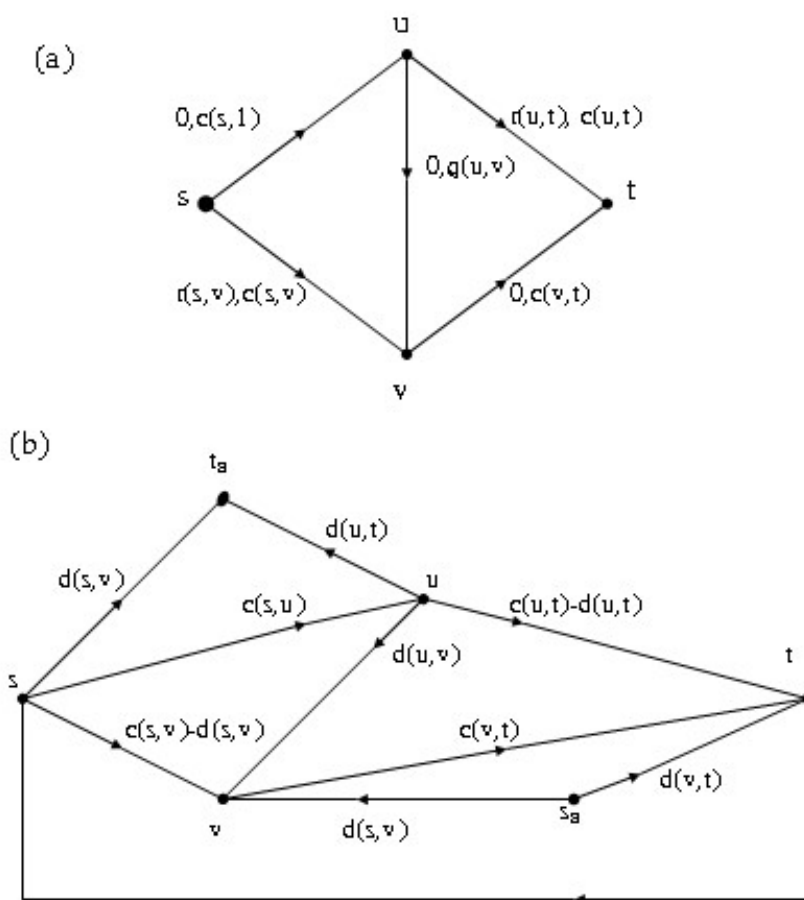
Xét mạng  $G$  mà trong đó mỗi cung  $(u, v)$  có khả năng thông qua (cận trên của luồng trên cung)  $c(u,v)$  và cận dưới của luồng là  $d(u,v)$ . Bài toán đặt ra là liệu có tồn tại luồng tương thích từ  $s$  đến  $t$ , tức là luồng  $f(u,v): u, v \in V$  thỏa mãn thêm ràng buộc

$$d(u,v) \leq f(u,v) \leq c(u,v), \forall u, v \in E,$$

hay không?

Đưa vào mạng  $G$  đỉnh phát  $s$  và đỉnh thu  $t$  và xây dựng mạng  $G_a$  theo qui tắc:

Mỗi cung  $(u,v)$  mà  $d(u,v) \neq 0$  sẽ tương ứng với 2 cung  $(s_a, v)$  và  $(u, t_a)$  với khả năng thông qua là  $d(u,v)$ . Giảm  $c(u,v)$  đi  $d(u,v)$  tức là thay khả năng thông qua của cung  $(u,v)$  bởi  $c(u,v) - d(u,v)$  còn cận dưới của nó đặt bằng 0. Ngoài ra thêm vào cung  $(t,s)$  với  $c(t,s) = \infty$ .



Hình 6. Mạng với khả năng thông qua bị chặn hai phía.



Hình 6(a) cho ví dụ mạng  $G$  với khả năng thông qua của các cung bị chặn cả hai phía. Đồ thị  $G_a$  tương ứng được cho trong hình 6(b).

Ký hiệu  $d^* = ? d(u,v)$ .

$$d(u,v) \neq 0$$

**Định lý 4.**

- 1) Nếu luồng lớn nhất trong mạng  $G_a$  từ  $s$  đến  $t$  bằng  $d^*$  thì tồn tại luồng tương thích trong  $G$ .
- 2) Nếu luồng lớn nhất trong mạng  $G_a$  từ  $s$  đến  $t$  là khác  $d^*$  thì không tồn tại luồng tương thích trong  $G$ .

# Một số ứng dụng trong tổ hợp

Bài toán luồng cực đại có rất nhiều ứng dụng trong việc giải bài toán tổ hợp. Khó khăn chính ở đây là phải xây dựng mạng tương ứng sao cho việc tìm luồng cực đại trong nó sẽ tương đương với việc giải bài toán tổ hợp đặt ra. Mục này sẽ giới thiệu một số bài toán như vậy.

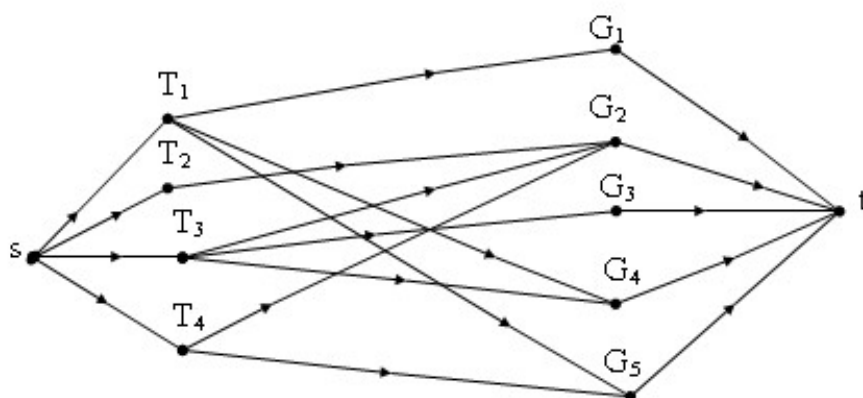
## Bài toán đám cưới vùng quê.

Có  $m$  chàng trai ở một vùng quê nọ. Đối với mỗi chàng trai ta biết các cô gái mà anh ta vừa ý. Hỏi khi nào thì có thể tổ chức các đám cưới trong đó chàng trai nào cũng sánh duyên với các cô gái mà mình vừa ý. Ta có thể xây dựng đồ thị với các đỉnh biểu thị các chàng trai và các cô gái, còn các cung biểu thị sự vừa ý của các chàng trai với các cô gái. Khi đó ta thu được một đồ thị hai phía.

Thí dụ. Có 4 chàng trai ?  $T_1, T_2, T_3, T_4$  và 5 cô gái ?  $G_1, G_2, G_3, G_4, G_5$ . Sự vừa ý cho trong bảng sau

Chàng trai	Các cô gái mà chàng trai ưng ý
$T_1$	$G_1, G_4, G_5$
$T_2$	$G_2$
$T_3$	$G_2, G_3, G_4$
$T_4$	$G_2, G_4$

Đồ thị tương ứng được cho trong hình 7.



## Hình 7. Mạng tương ứng với bài toán đám cưới vùng quê

Đưa vào điểm phát  $s$  và điểm thu  $t$ . Nối  $s$  với tất cả các đỉnh biểu thị các chàng trai, và nối  $t$  với tất cả các đỉnh biểu thị các cô gái. Tất cả các cung của đồ thị đều có khả năng thông qua bằng 1. Bắt đầu từ luồng 0, ta tìm luồng cực đại trong mạng xây dựng được theo thuật toán Ford-Fulkerson. Từ định lý về tính nguyên, luồng trên các cung là các số hoặc 1. Rõ ràng là nếu luồng cực đại trong đồ thị có giá trị  $V_{\max} = m$ , thì bài toán có lời giải, và các cung với luồng bằng 1 sẽ chỉ ra cách tổ chức đám cưới thoả mãn điều kiện đặt ra. Ngược lại, nếu bài toán có lời giải thì  $V_{\max} = m$ . Bài toán về đám cưới vùng quê là một trường hợp riêng của bài toán về cặp ghép trên đồ thị hai phía mà để giải nó có thể xây dựng thuật toán hiệu quả hơn.

### Bài toán về hệ thống đại diện chung.

Cho tập  $m$  phần tử  $X = \{z_1, z_2, \dots, z_m\}$ . Giả sử  $\langle A_1, A_2, \dots, A_n \rangle$  và  $\langle B_1, B_2, \dots, B_n \rangle$  là hai dãy các tập con của  $X$ . Dãy gồm  $n$  phần tử khác nhau của  $X$ :  $\langle a_1, a_2, \dots, a_n \rangle$  được gọi là hệ thống các đại diện chung của hai dãy đã cho nếu như tìm được một hoán vị  $\sigma$  của tập  $\{1, 2, \dots, n\}$  sao cho  $\langle a_1, a_2, \dots, a_n \rangle$  là hệ thống các đại diện phân biệt của hai dãy  $\langle A_1, A_2, \dots, A_n \rangle$  và  $\langle B_{\sigma(1)}, B_{\sigma(2)}, \dots, B_{\sigma(n)} \rangle$ , tức là điều kiện sau được thoả mãn:  $a_i \in A_i \cap B_{\sigma(i)}$ ,  $i = 1, 2, \dots, n$ . Xây dựng mạng  $G=(V,E)$  với tập đỉnh

$$V = \{s, t\} \cup \{x_1, x_2, \dots, x_n\} \cup \{u_1, u_2, \dots, u_n\} \cup \{v_1, v_2, \dots, v_n\} \cup \{y_1, y_2, \dots, y_n\}.$$

trong đó đỉnh  $x_i$  tương ứng với tập  $A_i$ , đỉnh  $y_i$  tương ứng với tập  $B_i$ , các phần tử  $u_j, y_j$  tương ứng với phần tử  $z_j$ . Tập các cung của mạng  $G$  được xác định như sau

$$E = \{(s, x_i): 1 \leq i \leq n\} \cup \{(x_i, u_j): \text{với } z_j \in A_i, 1 \leq i \leq n, 1 \leq j \leq m\} \cup \{(u_j, v_j): 1 \leq j \leq m\} \cup \{(v_j, y_i): \text{với } z_j \in B_i, 1 \leq i \leq n, 1 \leq j \leq m\} \cup \{(y_i, t): 1 \leq i \leq n\}$$

Khả năng thông qua của tất cả các cung được đặt bằng 1. Dễ dàng thấy rằng hệ thống đại diện chung của hai dãy và tồn tại khi và chỉ khi trong mạng  $G=(V,E)$  tìm được luồng với giá trị  $n$ . Để xét sự tồn tại của luồng như vậy có thể sử dụng thuật toán tìm luồng cực đại từ  $s$  đến  $t$  trong mạng  $G=(V,E)$ .

### Về một bài toán tối ưu rời rạc.

Trong mục này ta sẽ trình bày thuật toán được xây dựng dựa trên thuật toán tìm luồng cực đại để giải một bài toán tối ưu rời rạc là mô hình toán học cho một số bài toán tối ưu tổ hợp.

Xét bài toán tối ưu rời rạc:

$$f(x_1, x_2, \dots, x_n) =$$

$$\max_{1 \leq i \leq m} \sum_{j=1}^n x_{ij} \rightarrow \min$$

(1)

với điều kiện

$\sum_{j=1}^n a_{ij} x_{ij} = p_i, i=1, 2, \dots, m$	(2)
$x_{ij}=0 \text{ hoặc } 1, j=1, 2, \dots, n$	(3)

trong đó  $a_{ij} \in \{0,1\}$ ,  $i = 1, 2, \dots, m$ ;  $j=1, 2, \dots, n$ ,  $p_i$  –nguyên dương,  $i = 1, 2, \dots, m$ .

Bài toán (1)-(3) là mô hình toán học cho nhiều bài toán tối ưu tổ hợp thực tế. Dưới đây ta dẫn ra một vài ví dụ điển hình.

**Bài toán phân nhóm sinh hoạt.** Có  $m$  sinh viên và  $n$  nhóm sinh hoạt chuyên đề. Với mỗi sinh viên  $i$ , biết  $a_{ij}=1$ , nếu sinh viên  $i$  có nguyện vọng tham gia vào nhóm  $j$ ,  $a_{ij}=0$ , nếu ngược lại, và  $p_{ij}$  là số lượng nhóm chuyên đề mà sinh viên  $i$  phải tham dự,  $i = 1, 2, \dots, m$ ;  $j=1, 2, \dots, n$ . Trong số các cách phân các sinh viên vào nhóm chuyên đề mà họ có nguyện vọng tham gia và đảm bảo mỗi sinh viên  $i$  phải tham gia đúng  $p_i$  nhóm, hãy tìm cách phân phối với số người trong nhóm có nhiều sinh viên tham gia nhất là nhỏ nhất có thể được.

Đưa vào biến số

$x_{ij}=1$ , nếu sinh viên  $i$  tham gia vào nhóm  $j$ ,

$x_{ij}=0$ , nếu ngược lại,

$i=1, 2, \dots, m$ ,  $j=1, 2, \dots, n$ , khi đó dễ thấy mô hình toán học cho bài toán đặt ra chính là bài toán (1)-(3).

**Bài toán lập lịch cho hội nghị.** Một hội nghị có  $m$  tiểu ban, mỗi tiểu ban cần sinh hoạt trong một ngày tại phòng họp phù hợp với nó. Có  $n$  phòng họp dành cho việc sinh hoạt của các tiểu ban. Biết

$a_{ij} = 1$ , nếu phòng họp  $i$  là thích hợp với tiểu ban  $j$ ,  $a_{ij}=0$ , nếu ngược lại,  $i=1, 2, \dots, m$ ,  $j=1, 2, \dots, n$ . Hãy bố trí các phòng họp cho các tiểu ban sao cho hội nghị kết thúc sau ít ngày làm việc nhất.

Đưa vào biến số  $x_{ij} = 1$ , nếu bộ trí tiểu ban  $i$  làm việc ở phòng  $j$ ,  $x_{ij}=0$ , nếu ngược lại,  $i=1, 2, \dots, m$ ,  $j=1, 2, \dots, n$ , khi đó dễ thấy mô hình toán học cho bài toán đặt ra chính là bài toán (1)-(3), trong đó  $p_i=1$ ,  $i=1, 2, \dots, m$ .

## Bổ đề 2.

Bài toán (1)-(3) có phương án tối ưu khi và chỉ khi

$$\sum_{j=1}^n a_{ij} x_{ij} = p_i, \quad i=1, 2, \dots, m$$

$$x_{ij}=0 \text{ hoặc } 1, \quad j=1, 2, \dots, n$$

**Chứng minh.** Điều kiện cần của bổ đề là hiển nhiên vì từ sự tồn tại phương án của bài toán suy ra các bất đẳng thức trong (4) được thực hiện ít nhất dưới dạng dấu đẳng thức. Để chứng minh điều kiện đủ, chỉ cần chỉ ra rằng nếu điều kiện (4) được thực hiện thì bài toán luôn có phương án. Thực vậy, giả sử điều kiện (4) được thực hiện. Khi đó nếu ký hiệu  $I_+ = \{j: 1 \leq j \leq n: a_{ij}=1\}$ , thì  $I_+ \neq \emptyset$ ,  $i=1, 2, \dots, m$ . Do đó nếu gọi  $I_i \subset I_+$ ,  $I_i \neq \emptyset$ ,  $i=1, 2, \dots, m$ , thì  $X^* = (x_{ij}^*)_{m \times n}$  với các thành phần được xác định theo công thức

$x_{ij}^* = 1, j \in I_i, x_{ij}^* = 0, j \notin I_i, i=1, 2, \dots, m$ , là phương án của bài toán (1)-(3). Bổ đề được chứng minh.

Do (4) là điều kiện cần để bài toán (1)-(3) có phương án, nên trong phần tiếp theo ta sẽ luôn giả thiết rằng điều kiện này được thực hiện.

Bây giờ ta chỉ ra rằng việc giải bài toán (1)-(3) có thể dẫn về việc giải một số hữu hạn bài toán luồng cực đại trong mạng. Trước hết, với mỗi số nguyên dương  $k$ , xây dựng mạng  $G(k) = (V, E)$  với tập đỉnh

$$V = \{s\} \cup \{u_i: i=1, 2, \dots, m\} \cup \{w_j: j=1, 2, \dots, n\} \cup \{t\}$$

trong đó  $s$  là điểm phát,  $t$  là điểm thu, và tập cung

$$E = \{(s, u_i): i=1, 2, \dots, m\} \cup \{(u_i, w_j): i=1, 2, \dots, m; j=1, 2, \dots, n\} \cup \{(w_j, t): j=1, 2, \dots, n\}.$$

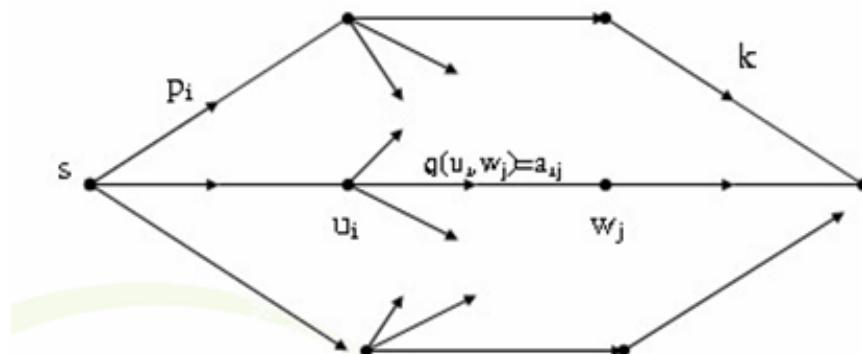
Mỗi cung  $e \in E$  được gán với khả năng thông qua  $q(e)$  theo qui tắc sau:

$$q(s, u_i) = p_i, \quad i=1, 2, \dots, m,$$

$$q(u_i, w_j) = a_{ij}, \quad i=1, 2, \dots, m; j=1, 2, \dots, n;$$

$$q(w_j, t) = k, \quad j=1, 2, \dots, n.$$

Hình 8 chỉ ra cách xây dựng mạng  $G(k)$ .



Hình 8. Mạng  $G(k)$ .

Bổ đề sau đây cho thấy mối liên hệ giữa luồng cực đại trong mạng  $G(k)$  và phương án của bài toán (1)-(3).

### Bổ đề 3.

Giả sử đối với số nguyên dương  $k$  nào đó, luồng cực đại nguyên trong mạng  $G(k)$  có giá trị là  $\sigma$ . Khi đó  $X^* = (x^*_{ij})_{m \times n}$  với các thành phần được xác định theo công thức

$$x^*_{ij} = f^*(u_i, w_j), i=1, 2, \dots, m; j=1, 2, \dots, n.$$

là phương án của bài toán (1)-(3).

**Chứng minh.** Thực vậy, do luồng cực đại trong mạng có giá trị  $\sigma$  và là luồng nguyên nên

$$f^*(s, u_i) = p_i, i=1, 2, \dots, m,$$

$$f^*(u_i, w_j) \in \{0, 1\}, i=1, 2, \dots, m, j=1, 2, \dots, n,$$

từ đó suy ra

Vậy  $X^*$  là phương án của bài toán (1)-(3). Bổ đề được chứng minh.

### Bổ đề 4.

Giả sử  $X^* = (x^*_{ij})$  là phương án tối ưu và  $k^*$  là giá trị tối ưu của bài toán (1)-(3). Khi đó luồng cực đại trong mạng  $G(k^*)$  có giá trị  $\sigma$ .

**Chứng minh.** Do giá trị của luồng cực đại trong mạng  $G(k^*)$  không vượt quá  $\sigma$ , nên để chứng minh bổ đề ta chỉ cần chỉ ra luồng với giá trị  $\sigma$  trong mạng  $G(k^*)$ .

Xây dựng luồng  $?^*$  theo công thức sau:

$$?^*(s, u_i) = p_i, ?^*(u_i, w_j) = x^*_{ij},$$

Dễ dàng kiểm tra được  $?^*$  là luồng trong mạng  $G(m)$  có giá trị  $\sigma$ . Bổ đề được chứng minh.

**Bổ đề 5.** Nếu  $k=m$  thì luồng cực đại trong mạng  $G(m)$  có giá trị  $\sigma$ .

**Chứng minh.** Lập luận tương tự như trong Bổ đề 4, ta chỉ cần chỉ ra luồng với giá trị  $\sigma$  trong mạng  $G(m)$ . Thực vậy, giả sử  $X^* = (x^*_{ij})_{m \times n}$  là phương án của bài toán (1)-(3) xây dựng theo công thức (5). Xây dựng luồng  $?^*$  theo công thức giống như trong chứng minh bổ đề 4, ta có luồng với giá trị  $\sigma$ . Bổ đề được chứng minh.

Từ bổ đề 3 và 4 suy ra việc giải bài toán (1)-(3) dẫn về việc tìm giá trị  $k^*$  nguyên dương nhỏ nhất sao cho luồng cực đại trong mạng  $G(k^*)$  có giá trị  $\sigma$ . Bổ đề 5 cho ta thấy giá trị  $k^* \in [1, m]$ . Vì vậy để giải bài toán (1)-(3) ta có thể áp dụng phương pháp tìm kiếm nhị phân trên đoạn  $[1, m]$  để tìm giá trị  $k^*$ , trong đó ở mỗi bước cần giải một bài toán luồng cực đại. Để giải bài toán tìm luồng cực đại trong mạng có thể sử dụng các thuật toán đa thức như đã nói ở trên. Từ đó suy ra kết quả sau

**Định lý 5.**

Bài toán (1)-(3) giải được nhờ thuật toán đa thức với độ phức tạp tính toán là  $\log_2 m \cdot \text{ONF}$ , trong đó ONF là độ phức tạp tính toán của bài toán tìm luồng cực đại trong mạng  $G(k)$ .

# Bài tập chương 3

## Bài 1 : Các miền trên bảng

Cho một bảng chữ nhật chia thành  $M \times N$  ô vuông ( $M$  dòng,  $N$  cột). Mỗi ô vuông ghi một số nguyên dương (trong khoảng từ 1 đến 255). Một miền của bảng là tập hợp tất cả các ô có cùng giá trị số sao cho chúng đi được sang nhau bằng cách đi qua các ô có chung cạnh và có cùng giá trị số đang xét. Địa chỉ của một miền là tọa độ [dòng, cột] của ô đầu tiên thuộc miền theo thứ tự duyệt từ trái sang phải và từ trên xuống dưới. Diện tích của một miền là số ô thuộc miền đó. Thí dụ bảng:

1	1	2	2	2
1	2	2	1	2
3	1	1	1	2

có 4 miền, miền tô màu xám (giá trị các ô là 2) có địa chỉ [1, 3] và diện tích là 7.

Cần xác định:

Số miền của mảng.

Miền có diện tích lớn nhất (chỉ rõ giá trị diện tích và địa chỉ của miền).

Dữ liệu vào cho trong file văn bản (tên file đọc từ bàn phím) có dạng:

M N

A[1, 1] A[1, 2]...A[1, N]

A[2, 1] A[2, 2]...A[2, N]

.....

A[M, 1] A[M, 2]...A[M, N]

trong đó A[i, j] là giá trị số của ô [i, j], các số trên cùng một dòng ghi cách nhau ít nhất một dấu trắng.



Yêu cầu chương trình thiết kế theo menu gồm các chức năng:

Đọc dữ liệu vào từ file

Giải bài toán bằng tìm kiếm theo chiều rộng.

Giải bài toán bằng tìm kiếm theo chiều sâu.

Kết thúc chương trình.

Kết quả tìm được đưa ra màn hình.

Giới hạn kích thước:  $M, N \leq 100$ .

## **Bài 2.**

Cho một mạng  $N$  ( $N \leq 20$ ) máy tính được đánh số từ 1 đến  $N$ . Sơ đồ mạng được cho bởi hệ gồm  $M$  kênh (đoạn) nối trực tiếp giữa một số cặp máy trong mạng,  $m$  kênh tương ứng với  $m$  cặp. Cho biết chi phí truyền 1 đơn vị thông tin theo mỗi kênh của mạng.

Người ta cần chuyển một bức thông điệp từ máy  $s$  đến máy  $t$ . Để đảm bảo an toàn, người ta chuyển bức thông điệp này theo hai đường truyền tin khác nhau (tức không có kênh nào) của mạng được sử dụng trong cả hai đường truyền tin; cho phép hai đường truyền tin cùng đi qua một số máy tính). Chi phí của một đường truyền được hiểu là tổng chi phí trên các kênh của nó. Đơn giá đường truyền từ máy  $s$  sang máy  $t$  được tính như sau:

Với hai máy  $s$  và  $t$ , cùng bức thông điệp có độ dài là 1 đơn vị thông tin, đơn giá truyền cho cặp  $(s, t)$  được tính bằng tổng chi phí chuyển thông điệp an toàn (bằng tổng chi phí của hai đường truyền tin) là nhỏ nhất.

Người ta mong muốn mạng máy tính (mạng truyền tin nói trên thỏa mãn tính chất an toàn theo nghĩa là từ một máy bất kỳ luôn truyền được (một cách an toàn) thông điệp tới một máy bất kỳ khác. Khi một mạng an toàn, người ta tính được đơn giá của mạng là tổng đơn giá mọi đường truyền từ một máy bất kỳ tới một máy bất kỳ khác.

Ma trận đơn giá của mạng là mảng hai chiều  $A$  có  $N$  dòng và  $N$  cột, mà giá trị phần tử  $A[i, j]$  chính là đơn giá từ máy  $i$  sang máy  $j$ .

## **Bài 3 :**

Một khóa học gồm  $N$  môn học, môn học  $i$  phải học trong  $t_i$  ngày. Giữa các môn học có mối quan hệ trước/sau: có môn học chỉ học được sau khi đã học một số môn học khác. Mối quan hệ đó được thể hiện bởi một mảng hai chiều  $A[i, j]$ ;  $i, j = 1, \dots, N$  trong đó

$A[i, j] = 1/0$  và  $A[i, i]$  bằng 0 với mọi  $i$ ,  $A[i, j] = 1$  khi và chỉ khi môn học  $i$  phải được dạy xong trước khi học môn  $j$  (ngày kết thúc môn  $i$  phải trước ngày bắt đầu môn  $j$ ). Môn học  $i$  phải dạy trước môn học  $j$  nếu có một dãy môn học  $i_1, i_2, \dots, i_k$  sao cho  $a[i_t, i_{t+1}] = 1, 1 \leq t \leq k-1, i_1=i$  và  $i_k=j$ . Nếu có một nhóm các môn học từng đôi một không có quan hệ trước/sau thì trong mỗi ngày, về nguyên tắc, ta có thể học đồng thời tất cả những môn học này (nếu không vi phạm quan hệ với các môn học khác). Mảng  $A[i, j]$  được gọi là bế tắc nếu có một dãy các môn học  $i_1, i_2, \dots, i_k, k > 1$ , mà môn  $i_1$  phải dạy trước môn  $i_2$ , môn  $i_2$  phải dạy trước môn  $i_3, \dots$ , môn  $i_{k-1}$  phải dạy trước môn  $i_k$ , môn  $i_k$  phải dạy trước môn  $i_1$ .

Hãy viết chương trình với tên KT3.PAS làm các việc sau:

Hãy xét xem mảng  $A$  có bế tắc hay không.

Nếu mảng  $A$  không bế tắc, hãy tính xem khóa học có thể kết thúc trong thời gian nhanh nhất là bao nhiêu ngày.

Theo các học bảo đảm thời gian hoàn thành ngắn nhất ở câu 2, hãy tính xem một học sinh trong quá trình học phải học đồng thời trong một ngày nhiều nhất bao nhiêu môn.

Dữ liệu vào được cho bởi file text có tên MH.DAT trong đó số  $N$  ghi ở dòng thứ nhất, trong nhóm  $N$  dòng tiếp theo, dòng thứ  $i$  ghi  $N$  số  $A[i, 1], \dots, A[i, N]$  dòng cuối cùng ghi  $N$  số nguyên dương  $t_i$  không lớn hơn 30,  $1 \leq i \leq N; N \leq 30$ .

Kết quả ghi ra file TKB.DAT như sau: dòng thứ nhất ghi số 1/0 tùy theo mảng  $A$  bế tắc / không bế tắc. Nếu dòng thứ nhất ghi số 0, ta mới ghi tiếp kết quả câu 2 và 3.

Kết quả câu 2 ghi tiếp vào file TKB.DAT  $N+1$  dòng như sau: dòng đầu ghi số  $T$  là số ngày tối thiểu có thể hoàn thành khóa học, tiếp theo là  $N$  dòng trong đó dòng thứ  $i$  ghi 2 số  $X, Y$  với ý nghĩa môn học thứ  $i$  học từ ngày thứ  $X$  đến ngày thứ  $Y$  (chú ý rằng  $Y-X=t_i-1$ ).

Kết quả câu 3 ghi tiếp vào file TKB.DAT như sau: dòng thứ nhất ghi 2 số  $Z, W$  với ý nghĩa trong ngày  $Z$  phải học  $W$  môn ( $W$  là số nhiều nhất các môn học phải học đồng thời trong một ngày), tiếp theo là một dòng ghi tên các môn học phải học đồng thời trong ngày  $Z$ .

Trong các câu 2 và 3, có thể có nhiều lời giải tương đương chỉ cần đưa ra một lời giải.

Ví dụ 1

MH.DAT	TKB.DAT
4	1
0 1 0 0	
0 0 1 0	
0 0 0 1	
1 0 0 0	
1 1 1 1	

Ví dụ 2

MH.DAT	TKB.DAT
7	0
0 1 0 0 0 0 0	22
0 0 0 1 0 0 0	1 2
0 0 0 1 0 0 0	3 4
0 0 0 0 1 1 0	1 8
0 0 0 0 0 0 0	9 12
0 0 0 0 0 0 1	13 22
0 0 0 0 0 0 0	13 14
2 2 8 4 10 2 3	15 17
	1 2
	1 3

#### Bài 4: Hình vuông Latinh

Hình vuông la tinh cấp 4 là ma trận vuông kích thước 4x4 mà mỗi dòng và mỗi cột của nó đều là một hoán vị của các chữ cái A, B, C, D. Hai hình vuông la tinh được gọi là tương đương nếu từ hình này ta có thể thu được hình kia nhờ sử dụng các phép biến đổi sau: 1) đổi chỗ hai dòng; 2) đổi chỗ hai cột; 3) đổi tên hai chữ cái.

**Ví dụ:** Hai hình vuông la tinh

A B C D          A B C D

$\theta_1 = B D A C$      $\theta_2 = B C D A$

C A D B          C D A B

D C B A          D A B C

là tương đương, bởi vì đổi chỗ dòng 3 và 4 của  $\theta_1$  ta thu được

A B C D

B D A C

D C B A

C A D B

tiếp đến đổi chỗ cột 3 và 4 ta thu được

A B D C

B D C A

D C A B

C A B D

cuối cùng, đổi tên hai chữ C và D cho nhau (nghĩa là thay C bởi D và thay D bởi C) ta thu được  $\theta_2$ .

**Yêu cầu:** Cho  $\theta_1$ ,  $\theta_2$  là hai hình vuông la tinh cấp 4. Hãy xác định xem hai hình vuông đã cho có tương đương hay không?

**Dữ liệu vào:** file văn bản LATIN.INP có cấu trúc như sau:

4 dòng đầu tiên chứa các dòng của hình vuông  $\theta_1$ ;

4 dòng tiếp theo chứa các dòng của hình vuông  $\theta_2$ ;

(các phần tử trong một dòng được viết liền nhau);

**Kết quả:** Ghi ra file văn bản có tên LATIN.OUT dưới dạng sau:

Dòng đầu tiên ghi số lượng phép biến đổi  $k$  ( $k=0$ , nếu hai hình vuông là không tương đương);

Các dòng tiếp theo ghi dãy các phép biến đổi cần áp dụng để từ  $\theta_1$  thu được  $\theta_2$ ; thông tin về một phép biến đổi bao gồm: chỉ số của phép biến đổi, chỉ số hai dòng (cột) cần đổi chỗ hoặc hai chữ cái cần đổi tên cho nhau.

**Ví dụ:** các file dữ liệu và kết quả có thể:

LATIN.INP	LATIN.OUT
ABCD	3
BDAC	1 3 4
DCBA	2 3 4
CADB	3 C D
ABCD	
BCDA	
CDAB	
DABC	

## Bài 5: Truyền tin trên mạng

Có một nhóm gồm  $N$  lập trình viên được đánh số từ 1 tới  $N$ , một số người trong họ có biết địa chỉ email của nhau. Khi biết một thông tin nào mới họ gửi thông tin đó cho nhau. Bạn là một người rất quan trọng và bạn biết tất cả các mối quan hệ của họ cũng như bạn có một thông tin rất đặc biệt mà muốn cho tất cả họ đều biết. Hãy lập trình chỉ ra một số

ít nhất các lập trình viên cần cho họ biết thông tin sao cho những người đó có thể thông báo cho tất cả những người còn lại thông tin của bạn.

Dữ liệu cho trong file văn bản với tên INFOR.INP trong đó dòng đầu chứa số N ( $N \leq 1000$ ), dòng thứ I trong N dòng tiếp theo chứa danh sách các lập trình viên mà người I biết địa chỉ email của họ. Nếu người thứ I không biết địa chỉ của bất cứ ai thì dòng này là dòng trống.

Kết quả ghi ra file văn bản với tên INFOR.OUT trong đó dòng đầu ghi số K là số người cần cho họ biết thông tin. Dòng thứ hai ghi ra chỉ số của những người đó.

Ví dụ:

INFOR.INP	INFOR.OUT
6	3
2 3	1
1	4
1	6
5	
4	

## Bài 6: Mọi con đường đều về 0

Một số tự nhiên dương N đều tìm được 2 số nguyên dương  $x \leq y$  sao cho  $N=xy$ . Với cách phân tích này ta sẽ hình thành ra số tự nhiên mới  $N_1=(x-1)(y+1)$ . Số này có thể dương hoặc bằng 0. Nếu nó dương ta có thể chọn một cách phân tích nào đó và lặp lại những thao tác trên.

**Bài toán:** Với số tự nhiên N ( $N \leq 10000$ ) cho trước, hãy cho biết ta có thể sinh ra được những số nào.

**Dữ liệu vào** cho trong file text với tên VE0.INP trong đó chứa một số duy nhất N.

**Kết quả** ghi ra file text VE0.OUT trong đó liệt kê ra tất cả các số tìm được theo thứ tự tăng dần.

Ví dụ:

VE0.INP	VE0.OUT
12	0 3 4 6 7 10

## Bài 7: Chuyển bi

Cậu bé vẽ  $N$  ( $N \leq 100$ ) vòng tròn đánh số từ 1 tới  $N$  và tô màu các vòng tròn đó (có thể có các vòng tròn có màu giống nhau) sau đó nối từng cặp lại bằng các cung định hướng, mỗi cung có một màu nhất định. Các màu của cung và của vòng tròn được đánh số từ 1 đến 100.

Cậu bé chọn 3 số nguyên khác nhau  $L$ ,  $K$  và  $Q$  nằm trong phạm vi từ 1 tới  $N$  đặt vào trong các vòng tròn số  $L$  và  $K$  mỗi vòng tròn một hòn bi sau đó bắt đầu di chuyển theo quy tắc sau :

Bi chỉ được chuyển theo cung có màu trùng với màu vòng tròn chứa bi thứ 2

Bi chỉ được chuyển theo chiều cung

Hai viên bi không được đồng thời ở cùng một vòng tròn

Không nhất thiết phải di chuyển lần lượt các viên bi.

Quá trình di chuyển kết thúc khi một trong hai viên bi tới vòng tròn  $Q$

Hãy lập trình xác định cách di chuyển để chấm dứt quá trình sau một số ít nhất các bước chuyển.

**Dữ liệu vào:** file BI.INP

Dòng đầu : 4 số nguyên  $N$   $L$   $K$   $Q$

Dòng thứ 2 :  $N$  số nguyên  $C_1, C_2, \dots, C_N$ ,  $C_i$  là màu vòng tròn  $i$ .

Dòng thứ 3 : Số nguyên  $M$  ( $0 \leq M \leq 10000$ )

$M$  dòng sau : mỗi dòng 3 số nguyên  $A_i, B_i, D_i$  xác định cung màu  $D_i$  đi từ vòng tròn  $A_i$  tới vòng tròn  $B_i$ .

Các số trên một dòng cách nhau một dấu cách.

**Kết quả:** đưa ra file BI.OUT

Dòng đầu : CO hoặc KHONG cho biết quá trình có kết thúc được hay không.

Nếu dòng đầu là CO thì dòng 2 chứa số nguyên xác định số bước di chuyển tối thiểu.

Ví dụ :

BI.INP

5 3 4 1

2 3 2 1 4

8

2 1 2

4 1 5

4 5 2

5 1 3

3 2 2

2 3 4

5 3 1

3 5 1

BI.OUT

CO

3

## **Bài 8: Gỡ mìn**

Cho bảng hình chữ nhật kích thước  $M \times N$  ( $M$  số dòng,  $N$  số cột) ô vuông. Mỗi ô mang giá trị 0 hoặc 1, nếu ô  $(i, j)$  có mìn  $A[i, j] = 1$ , ngược lại thì  $A[i, j] = 0$ .



(a) Một người xuất phát từ ô  $(X1, Y1)$  không có mìn, kiểm tra xem người này có thể di chuyển đến ô  $(X2, Y2)$  được hay không bằng cách di chuyển sang những ô chung cạnh không có mìn.

(b) Nếu kết quả câu a là người đó không thể di chuyển đến  $(X2, Y2)$  được thì hãy chỉ ra cách gỡ ít nhất những quả mìn để anh ta có thể di chuyển đến  $(X2, Y2)$ .

**Dữ liệu vào:** file text GOMIN.INP

Dòng đầu là 6 số  $M, N, X1, Y1, X2, Y2$  cách nhau bởi khoảng trắng.  $M$  dòng tiếp theo, mỗi dòng gồm  $N$  số 0/1 tương ứng có mìn hoặc không có mìn, mỗi số cách nhau bởi khoảng trắng.

**Dữ liệu ra:** file text GOMIN.OUT

Dòng đầu chứa số 0/1 tương ứng với đi được / không đi được.

Nếu là không đi được thì dòng thứ hai là số  $K$  tương ứng với số mìn ít nhất cần phải gỡ.

Nếu có số  $K$  ở dòng thứ hai thì  $K$  dòng tiếp theo, mỗi dòng  $i$  gồm 2 số tương ứng với chỉ số cột và chỉ số dòng của ô thứ  $i$  cần phải gỡ mìn.

## **Bài 9:**

Cho một đồ thị vô hướng  $G$  gồm  $N$  đỉnh xác định bởi ma trận kề  $A[N, N]$  trong đó  $A[i, j] = 1$  nếu cạnh  $(i, j) \in G$  và  $A[i, j] = 0$  nếu  $(i, j) \notin G$ . Hãy cài đặt thuật toán và viết chương trình xác định tính liên thông của đồ thị  $G$ . Nếu  $G$  không liên thông, hãy cho biết số thành phần liên thông trong  $G$  và liệt kê cụ thể các thành phần liên thông này.

## **Bài 10: Tìm kiếm**

Một vùng đất hình chữ nhật được chia thành các mảnh đất theo dạng ô lưới  $M \times N$ . Mỗi ô có thể trồng trọt được sẽ được đánh dấu là 1, các ô còn lại được đánh dấu là 0. Hai ô gọi là kề nhau nếu chúng có chung 1 cạnh (ngang hay dọc). Một khu đất trồng là tập hợp các ô có thể trồng trọt được nằm kề nhau lớn nhất nghĩa là các khu đất trồng phân cách nhau bởi các ô được đánh dấu là 0. Hãy xác định số lượng và chỉ rõ cụ thể các khu đất trồng nằm trong vùng đất. Giả thiết rằng số lượng các ô khá lớn.

# Bài tập chương 4

## Bài 1: Hội nghị bàn tròn

Tổng thư ký Đại hội đồng Liên hợp quốc triệu tập một cuộc họp có N nhà ngoại giao của N tổ chức tham gia. Các đại diện ngoại giao được bố trí ngồi quanh một bàn tròn. Giữa một số tổ chức có quan hệ căng thẳng, vì vậy không thể xếp họ ngồi cạnh nhau được. Thông tin về quan hệ giữa các tổ chức được cho dưới dạng cặp số nguyên  $i, j$  nếu giữa 2 tổ chức này có quan hệ căng thẳng. Hãy lập trình giúp Tổng thư ký Liên hợp quốc bố trí chỗ ngồi quanh bàn họp. Các tổ chức được đánh số từ 1 tới N,  $0 < N \leq 500$ .

**Dữ liệu vào:** từ file CONF.INP, dòng đầu tiên chứa số nguyên N, các dòng sau, mỗi dòng một cặp số  $i, j$  cho biết các đại diện  $i$  và  $j$  không ngồi cạnh nhau được. Kết thúc là một dòng chứa 2 số 0.

**Kết quả:** đưa ra file CONF.OUT. Nếu không có cách bố trí thỏa mãn yêu cầu thì đưa ra thông báo KHONG CO, trong trường hợp ngược lại – đưa ra dãy N số nguyên xác định vị trí ai ngồi cạnh ai quanh bàn tròn.

Ví dụ:

CONF.INP    CONF.OUT

11 1 9 7 4 11 5 8 2 10 3 6

1 4

1 7

5 7

10 7

10 8

10 9

3 4

0 0

## Bài 2: Domino

Cho trước N con cờ domino, hãy viết chương trình cho biết rằng có cách sắp N con cờ đó theo đúng luật domino sao cho các con cờ hình thành vòng tròn hay không, nếu có hãy chỉ ra một cách sắp.

**Dữ liệu vào:** file DOMINO.INP

Dòng đầu chứa số N.N dòng tiếp theo, mỗi dòng chứa 2 số từ 0 đến 6 mang giá trị đại diện cho 2 đầu của domino.

**Dữ liệu ra:** file DOMINO.OUT

Dòng đầu chứa số 1 hoặc 0 tương ứng với sắp được quân domino thành vòng tròn và không sắp được. Nếu dòng đầu là 1 (tương ứng với sắp được) thì dòng thứ hai liệt kê ra N chỉ số của N con cờ domino theo thứ tự để sắp thành vòng tròn. Nếu dòng đầu là 0 thì không có dòng thứ hai.

Ví dụ:

DOMINO.INP

4

5 3

5 6

2 3

6 2

DOMINO.OUT

1

1 3 4 2

## Bài 3: Kiểm tra đường

Một trạm quảng đường giao thông phải chịu trách nhiệm về tình trạng của một mạng lưới giao thông nối giữa các điểm dân cư. Hàng tháng, họ phải cử một đội đi kiểm tra một vòng qua khắp mạng lưới để xem xét tình trạng hiện thời của các đường giao thông nhằm báo sửa chữa kịp thời nếu có nhu cầu. Hãy viết chương trình nhập vào mạng lưới

giao thông và giúp trạm quyết định lộ trình của đội kiểm tra sao cho có thể thăm tất cả các con đường mà tổng chiều dài đoạn đường đi qua là nhỏ nhất.

#### **Bài 4: Mã đi tuần**

Hãy cài đặt chương trình xác định lộ trình của con mã trên bàn cờ  $8 \times 8$  ô bắt đầu từ ô  $(i, j)$  đi qua tất cả các ô của bàn cờ và mỗi ô chỉ 1 lần duy nhất. Mở rộng với trường hợp bàn cờ kích thước  $N \times N$ .

#### **Bài 5: Hội nghị bàn tròn**

Có 12 người ngồi chung 1 bàn tiệc tròn. Mỗi người có ít nhất 6 người quen. Hãy chỉ ra cách sắp xếp sao cho mỗi người đều ngồi cạnh người mình quen. Tổng quát, hãy sắp  $N$  người ngồi chung quanh bàn tròn sao cho mỗi người đều ngồi cạnh người mình quen. Biết mỗi người có ít nhất  $(N + 1)/2$  người quen.

# Bài tập chương 5

## Bài 1 : Mạng an toàn

Cho một mạng  $N$  ( $N \leq 20$ ) máy tính được đánh số từ 1 đến  $N$ . Sơ đồ mạng được cho bởi hệ gồm  $M$  kênh (đoạn) nối trực tiếp giữa một số cặp máy trong mạng,  $m$  kênh tương ứng với  $m$  cặp. Cho biết chi phí truyền 1 đơn vị thông tin theo mỗi kênh của mạng. Người ta cần chuyển một bức thông điệp từ máy  $s$  đến máy  $t$ . Để đảm bảo an toàn, người ta chuyển bức thông điệp này theo hai đường truyền tin khác nhau (tức không có kênh nào) của mạng được sử dụng trong cả hai đường truyền tin; cho phép hai đường truyền tin cùng đi qua một số máy tính). Chi phí của một đường truyền được hiểu là tổng chi phí trên các kênh của nó. Đơn giá đường truyền từ máy  $s$  sang máy  $t$  được tính như sau:

Với hai máy  $s$  và  $t$ , cùng bức thông điệp có độ dài là 1 đơn vị thông tin, đơn giá truyền cho cặp  $(s, t)$  được tính bằng tổng chi phí chuyển thông điệp an toàn (bằng tổng chi phí của hai đường truyền tin) là nhỏ nhất. Người ta mong muốn mạng máy tính (mạng truyền tin nói trên thỏa mãn tính chất an toàn theo nghĩa là từ một máy bất kỳ luôn truyền được (một cách an toàn) thông điệp tới một máy bất kỳ khác. Khi một mạng an toàn, người ta tính được đơn giá của mạng là tổng đơn giá mọi đường truyền từ một máy bất kỳ tới một máy bất kỳ khác. Ma trận đơn giá của mạng là mảng hai chiều  $A$  có  $N$  dòng và  $N$  cột, mà giá trị phần tử  $A[i, j]$  chính là đơn giá từ máy  $i$  sang máy  $j$ .

**Câu 1:** Cho trước một mạng, hãy kiểm tra tính an toàn của mạng đó.

**Câu 2:** Khi mạng không an toàn được phép bổ sung một số kênh truyền để nó trở thành an toàn. Đơn giá mỗi kênh truyền bổ sung theo được coi bằng hai lần giá trị cực đại đơn giá các kênh đã có. Mọi kênh bổ sung được coi có đơn giá như nhau. Hãy tìm cách bổ sung các kênh mới mà đơn giá mạng là nhỏ nhất.

**Câu 3:** Khi mạng an toàn hoặc sau khi bổ sung kênh để mạng an toàn, hãy in ra đơn giá mạng và ma trận đơn giá.

**Dữ liệu vào:** cho trong file INP.B2 với cấu trúc như sau:

Dòng đầu tiên ghi 2 số  $n, m$  cách nhau bởi dấu cách. Mỗi dòng thứ  $i$  trong số  $m$  dòng tiếp theo ghi thông tin về kênh nối thứ  $i$  của mạng gồm 3 số  $d[i], c[i], g[i]$  trong đó  $d[i], c[i]$  là chỉ số của hai máy tương ứng với kênh này và  $g[i]$  (nguyên dương) là chi phí để truyền một đơn vị thông tin từ máy  $d[i]$  đến máy  $c[i]$  theo kênh này. Các giá trị  $g[i]$  cho trước không vượt quá 40 (và như vậy đơn giá các kênh bổ sung không vượt quá 80).

**Kết quả:** ghi ra file OUT.B2 theo qui cách sau:

Dòng đầu tiên ghi 1 số nguyên  $p$  thể hiện mạng có an toàn hay không và  $p$  có ý nghĩa là số lượng kênh cần bổ sung.  $p=0$  có nghĩa mạng an toàn;  $p>0$  có nghĩa mạng không an toàn và cần bổ sung  $p$  kênh nữa để mạng an toàn với chi phí bổ sung ít nhất.  $p$  dòng tiếp theo ghi  $p$  kênh bổ sung. Cách ghi như trong file dữ liệu vào. Dòng tiếp theo ghi đơn giá của mạng an toàn.  $N$  dòng tiếp theo ghi ma trận đơn giá của mạng an toàn: mỗi hàng của ma trận ghi trên một dòng.

## **Bài 2: Xây dựng đường ống nước**

Có 1 trạm cấp nước và  $N$  điểm dân cư. Hãy xây dựng chương trình thiết kế tuyến đường ống nước cung cấp đến mọi nhà sao cho tổng chiều dài đường ống phải dùng là ít nhất. Giả sử rằng các đường ống chỉ được nối giữa 2 điểm dân cư hoặc giữa trạm cấp nước với điểm dân cư.

# Bài tập chương 6

## Bài 1: Di chuyển trên các hình tròn

Cho  $N$  hình tròn (đánh số từ 1 đến  $N$ ). Một người muốn đi từ hình tròn này sang hình tròn khác cần tuân theo qui ước:

Nếu khoảng cách giữa 2 điểm gần nhất của 2 hình tròn không quá 50 cm thì có thể bước sang. Nếu khoảng cách này hơn 50cm và không quá 80cm thì có thể nhảy sang. Các trường hợp khác không thể sang được. Một đường đi từ hình tròn này sang hình tròn khác được gọi là càng "tốt" nếu số lần phải nhảy là càng ít. Hai đường đi có số lần nhảy bằng nhau thì đường đi nào có số hình tròn đi qua ít hơn thì đường đi đó "tốt" hơn. Các hình tròn được cho trong một file văn bản, trong đó dòng thứ  $i$  mô tả hình tròn số hiệu  $i$  ( $i = 1, 2, \dots, N$ ) bao gồm 3 số thực: hoành độ tâm, tung độ tâm, độ lớn bán kính (đơn vị đo bằng mét). Lập trình đọc các hình tròn từ một file văn bản (tên file vào từ bàn phím), sau đó cứ mỗi lần đọc số hiệu hình tròn xuất phát  $S$  và hình tròn kết thúc  $T$  từ bàn phím, chương trình sẽ đưa ra đường đi từ  $S$  đến  $T$  là "tốt nhất" theo nghĩa đã nêu (hoặc thông báo là không có). Yêu cầu đường đi được viết dưới dạng một dãy các số hiệu hình tròn lần lượt cần được đi qua trong đó nói rõ tổng số các bước nhảy, tổng số các hình tròn đi qua và những bước nào cần phải nhảy. Giới hạn số hình tròn không quá 100.

## Bài 2: Tìm hành trình tốn ít xăng nhất

Trên một mạng lưới giao thông, một người muốn đi từ điểm  $A$  đến điểm  $B$  bằng xe máy. Xe chứa được tối đa 3 lít xăng và chạy 100km hết 2,5 lít. Các trạm xăng chỉ được đặt ở các điểm dân cư, không đặt ở giữa đường và người này không mang theo bất kỳ thùng chứa xăng nào khác. Hãy viết chương trình nhập vào mạng lưới giao thông và xác định giúp người này tuyến đường đi từ  $A$  đến  $B$  sao cho ít tốn xăng nhất.

## Bài 3: Di chuyển giữa các đảo

Trên một đảo quốc, có  $N$  hòn đảo. Giả sử tất cả các đảo đều có hình dạng là hình chữ nhật nằm ngang. Trên mỗi hòn đảo có thể có sân bay nằm ở trung tâm đảo, có thể có cảng nằm ở 4 góc đảo. Trên mỗi đảo đều có tuyến đường xe buýt nối 4 góc đảo với nhau và với trung tâm đảo. Giữa 2 đảo có thể đi lại bằng máy bay nếu cả 2 đảo đều có sân bay và có thể đi lại bằng tàu nếu cả 2 đảo đều có cảng.

Giả sử rằng:

Các tuyến đường (bộ, không, thủy) đều là đường thẳng. Chi phí cho mỗi km và tốc độ của mỗi loại phương tiện là:

Phương tiện	Tốc độ (km/h)	Chi phí (đ/km)
Máy bay	1000	1000
Xe buýt	70	100
Tàu thủy	30	50

Hãy viết chương trình xác định tuyến đường và cách di chuyển giữa 2 hòn đảo trong đảo quốc sao cho:

Thời gian di chuyển ít nhất.

Chi phí di chuyển ít nhất.

Thời gian di chuyển ít nhất nhưng với một số tiền chi phí không quá Đ đồng.

Chi phí di chuyển ít nhất nhưng với thời gian di chuyển không vượt quá T giờ.

#### Bài 4: Hành trình tới y

Các ô tô đi từ các thành phố khác nhau  $x_1, x_2, \dots, x_n$  và cùng tới một địa điểm thống nhất y. Nếu tồn tại đường đi từ  $x_i$  đến  $x_j$  thì ta ký hiệu  $t_{ij}$  là thời gian cần thiết để đi từ  $x_i$  đến  $x_j$ ,  $c_{ij}$  là lượng ô tô có thể đi trên con đường đó trong một đơn vị thời gian ( $c_{ij} = 0$  nếu không có đường đi),  $c_{ii}$  là lượng ô tô có thể nghỉ đồng thời ở thành phố  $x_i$ ,  $a_i$  là số lượng xe ban đầu có ở  $x_i$ . Hãy tổ chức hành trình sao cho trong khoảng thời gian  $\Delta t$  số ô tô tới y là nhiều nhất.

#### Bài 5: Tìm đường ngắn nhất

Giả sử X là tập các khu dân cư, U là tập các đường sá nối liền các khu đó. Ta giả sử mọi chỗ giao nhau của các con đường đều thuộc X. Với con đường u, số  $l(u)$  là độ dài của u tính bằng km. Hãy chỉ ra tuyến đường đi từ một khu i sang khu j sao cho tổng chiều dài là nhỏ nhất.

#### Bài 6: Đường đi trên lưới

Cho 1 ma trận  $A[M, N]$ , mỗi phần tử của nó chứa 1 số tự nhiên. Từ 1 ô (i, j) ta có thể đi sang ô kề nó (có chung 1 cạnh) nếu giá trị của ô kề này nhỏ hơn giá trị lưu trong (i, j). Hãy tìm 1 đường đi từ ô (i, j) tới ô (k, l) trên ma trận sao cho phải đi qua ít ô nhất. Hãy tìm 1 đường đi từ ô (i, j) tới ô (k, l) trên ma trận sao cho tổng giá trị các ô phải đi qua nhỏ nhất.



## Bài 7 : Tìm đường với chi phí phải trả cho phép

Có N thành phố được đánh số từ 1..N nối với nhau bằng các đoạn đường *một chiều*. Mỗi đoạn đường bao gồm 2 thông số : Độ dài và chi phí đi của đoạn đường. A sống tại thành phố 1 và A muốn di chuyển đến thành phố N nhanh nhất có thể. Bạn hãy giúp A tìm ra *đường đi ngắn nhất* từ thành phố 1 đến thành phố N mà A có *khả năng chi trả* tiền. Dữ liệu vào: ROADS.IN

Dòng đầu tiên chứa số nguyên K,  $0 \leq K \leq 10000$ , số tiền mà A có. Dòng thứ 2 chứa số nguyên N,  $2 \leq N \leq 100$ , số thành phố. Dòng thứ 3 chứa số nguyên R,  $1 \leq R \leq 10000$ , tổng số đoạn đường. Mỗi dòng trong số R dòng tiếp theo mô tả một đoạn đường bằng các số S, D, L và T cách nhau bởi ít nhất một khoảng trắng.

S là thành phố khởi hành,  $1 \leq S \leq N$ .

D là thành phố đến,  $1 \leq D \leq N$ .

L là độ dài của đoạn đường,  $1 \leq L \leq 100$ .

T là lộ phí,  $0 \leq T \leq 100$ .

Chú ý rằng giữa 2 thành phố có thể có nhiều đoạn đường nối 2 thành phố này.

Dữ liệu ra: ROADS.OUT

Chỉ có duy nhất 1 dòng chứa tổng độ dài của đường đi ngắn nhất từ 1->N và nhỏ hơn K.

<i>ROADS.IN</i>	<i>ROADS.IN</i>
5	0
6	4
7	4
1 2 2 3	1 4 5 2
2 4 3 3	1 2 1 0
3 4 2 4	2 3 1 1
1 3 4 1	3 4 1 0
4 6 2 1	<i>ROADS.OUT</i>
3 5 2 0	-1
5 4 3 2	
<i>ROADS.OUT</i>	
11	

# Bài tập chương 7

## Bài 1

Cho  $G=(V,E)$  đồ thị có hướng trong đó không có cung  $(s,t)$ . Chứng minh rằng số đường đi cơ bản nối hai đỉnh  $s$  và  $t$  là bằng số ít nhất các đỉnh của đồ thị cần loại bỏ để trong đồ thị không còn đường đi nối  $s$  với  $t$ .

## Bài 2

Xây dựng thuật toán tìm tập  $E_1$  tất cả các cung của đồ thị mà việc tăng khả năng thông qua của bất kỳ cung nào trong  $E$  đều dẫn đến tăng giá trị của luồng cực đại trong mạng.

## Bài 3

Cho hai dãy số nguyên dương  $\{p_i, i=1,2,\dots,m\}$  và  $\{q_j, j=1,2,\dots,n\}$ . Hãy xây dựng ma trận  $A=\{a_{ij} : i=1,2,\dots,m; j=1,2,\dots,n\}$  với các phần tử  $a_{ij} \in \{0,1\}$  có tổng các phần tử trên dòng  $i$  là  $p_i$ , tổng các phần tử trên cột  $j$  là  $q_j$ .

## Bài 4

Có  $m$  chàng trai,  $n$  cô gái và  $k$  bà mối, Mỗi bà mối  $p$  ( $p=1,2,\dots,k$ ) có một danh sách  $L_p$  một số chàng trai và cô gái trong số các chàng trai và cô gái nói trên là khách hàng của bà ta. Bà mối  $p$  có thể se duyên cho bất cứ cặp trai gái nào là khách hàng của bà ta, nhưng không đủ sức tổ chức quá  $d_p$  đám cưới. Hãy xây dựng thuật toán căn cứ vào danh sách  $L_p, d_p, p=1,2,\dots,k$ ; đưa ra cách tổ chức nhiều nhất các đám cưới giữa  $m$  chàng trai và  $n$  cô gái với sự giúp đỡ của các bà mối.

## Bài 5 : Chuyển bi

Cậu bé vẽ  $N$  ( $N \leq 100$ ) vòng tròn, đánh số từ 1 tới  $N$  và tô màu các vòng tròn đó (có thể có các vòng tròn có màu giống nhau), sau đó nối từng cặp các cung định hướng, mỗi cung có một màu nhất định. Các màu (của cung và vòng tròn) được đánh số từ 1 đến 100. Cậu bé chọn 3 số nguyên khác nhau  $L, K$  và  $Q$  nằm trong phạm vi từ 1 tới  $N$ , đặt vào trong các vòng tròn số  $L$  và  $K$  mỗi vòng một hòn bi, sau đó bắt đầu di chuyển bi theo nguyên tắc sau:

Bi chỉ được chuyển theo cung có màu trùng với màu của vòng tròn chứa viên bi thứ 2. Bi chỉ được chuyển theo chiều cung

Hai viên bi không được đồng thời ở cùng một vòng tròn;

Không nhất thiết phải di chuyển lần lượt các viên bi,

Quá trình di chuyển kết thúc, khi một trong hai viên bi tới vòng tròn Q.

Hãy lập trình xác định cách di chuyển để chấm dứt quá trình sau một số ít nhất các bước chuyển.

Dữ liệu vào từ file BL.INP:

Dòng đầu: 4 số nguyên N L K Q

Dòng thứ 2: N số nguyên  $C_1, C_2, \dots, C_n$ ;  $C_i$  là màu vòng tròn i

Dòng thứ 3: số nguyên M ( $0 \leq M \leq 10000$ )

M dòng sau: mỗi dòng 3 số nguyên  $A_i B_i D_i$ ; xác định cung màu  $D_i$  từ vòng tròn  $A_i$  tới vòng tròn  $B_i$ .

Các số trên một dòng cách nhau một dấu cách.

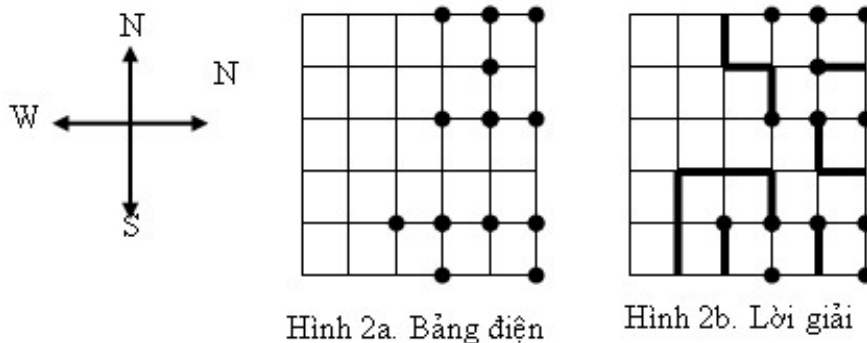
Kết quả đưa ra file BL.OUT:

Dòng đầu: CO hoặc KHONG, cho biết quá trình có kết thúc được hay không, nếu dòng đầu là CO thì dòng 2 chứa số nguyên xác định số bước chuyển tối thiểu .

BL.INP	BL.OUT
5 3 4 1	CO
2 3 2 1 4	3
8	
2 1 2	
4 1 5	
4 5 2	
4 5 2	
5 1 3	
3 2 2	
2 3 4	
5 3 1	
3 5 1	

### Bài 6 : Bảng điện

Một lưới ô vuông được phủ trên một bảng điện hình vuông. Vị trí nằm trên giao của 2 đường kẻ của lưới sẽ được gọi là nút. Tất cả có nxn nút trên lưới.



Có một số nút chứa tiếp điểm. Nhiệm vụ của bạn là cần nối các tiếp điểm với các nút ở trên biên của bảng bởi các đoạn dây dẫn (gọi là các mạch). Các mạch chỉ được chạy dọc theo các đường kẻ của lưới (nghĩa là không được chạy theo đường chéo). Hai mạch không được phép có điểm chung, vì vậy hai mạch bất kỳ không được phép cùng chạy qua cùng một đoạn đường kẻ của lưới cũng như không được chạy qua cùng một nút của lưới. Các mạch cũng không được chạy dọc theo các đoạn kẻ của lưới ở trên biên (mạch phải kết thúc khi nó gặp biên) và cũng không được chạy qua nút chứa tiếp điểm khác.

**Ví dụ:** Bảng điện và các tiếp điểm được cho trong hình 2a. Nút tô đậm trong hình vẽ thể hiện vị trí các tiếp điểm.

**Yêu cầu:** Viết chương trình cho phép nối được một số nhiều nhất các tiếp điểm với biên. Các tiếp điểm ở trên biên đã thỏa mãn đòi hỏi đặt ra, vì thế không nhất thiết phải thực hiện mạch nối chúng. Nếu như có nhiều lời giải thì chỉ cần đưa ra một trong số chúng.

**Dữ liệu vào:** file văn bản ELE.INP:

Dòng đầu tiên chứa số nguyên  $n$  ( $3 \leq n \leq 15$ ). Mỗi dòng trong số  $n$  dòng tiếp theo chứa  $n$  ký tự phân cách nhau bởi một dấu cách. Mỗi ký tự chỉ là 0 hoặc 1. Ký tự 1 thể hiện tiếp điểm, ký tự 0 thể hiện nút không có tiếp điểm trên vị trí tương ứng của lưới. Các nút được đánh số từ 1 đến  $n \times n$  theo thứ tự từ trái sang phải, từ trên xuống dưới. Chỉ số của nút chứa tiếp điểm sẽ là chỉ số của tiếp điểm.

**Kết quả:** ghi ra file ELE.OUT:

Dòng đầu tiên chứa  $k$  là số tiếp điểm lớn nhất có thể nối với biên bởi các mạch. Mỗi dòng trong số  $k$  dòng tiếp theo mô tả một mạch nối một trong số  $k$  tiếp điểm với biên theo qui cách sau: đầu tiên là chỉ số của tiếp điểm được nối, tiếp đến là dãy các ký tự mô tả hướng của mạch nối: E: đông, W: tây, N: bắc, S: nam. Giữa chỉ số và dãy ký tự phải có đúng một dấu cách, còn giữa các ký tự trong dãy ký tự không được có dấu cách. Kết quả phải được đưa ra theo thứ tự tăng dần của chỉ số tiếp điểm.

Ví dụ:

ELE.IN	ELE.OUT
6	11 E
0 0 0 1 1 1	16 NWN
0 0 0 0 1 0	17 SE
0 0 0 1 1 1	27 S
0 0 0 0 0 0	28 NWWSS
0 0 1 1 1 1	29 S
0 0 0 1 0 1	

# Các bài tập khác

## Bài 1:

Một khóa học gồm  $N$  môn học, môn học  $i$  phải học trong  $t_i$  ngày. Giữa các môn học có mối quan hệ trước/sau: có môn học chỉ học được sau khi đã học một số môn học khác. Mối quan hệ đó được thể hiện bởi một mảng hai chiều  $A[i, j]$ ;  $i, j = 1, \dots, N$  trong đó  $A[i, j] = 1/0$  và  $A[i, i]$  bằng 0 với mọi  $i$ ,  $A[i, j] = 1$  khi và chỉ khi môn học  $i$  phải được dạy xong trước khi học môn  $j$  (ngày kết thúc môn  $i$  phải trước ngày bắt đầu môn  $j$ ). Môn học  $i$  phải dạy trước môn học  $j$  nếu có một dãy môn học  $i_1, i_2, \dots, i_k$  sao cho  $a[i_t, i_{t+1}] = 1, 1 \leq t \leq k-1, i_1=i$  và  $i_k=j$ . Nếu có một nhóm các môn học từng đôi một không có quan hệ trước/sau thì trong mỗi ngày, về nguyên tắc, ta có thể học đồng thời tất cả những môn học này (nếu không vi phạm quan hệ với các môn học khác). Mảng  $A[i, j]$  được gọi là bế tắc nếu có một dãy các môn học  $i_1, i_2, \dots, i_k, k > 1$ , mà môn  $i_1$  phải dạy trước môn  $i_2$ , môn  $i_2$  phải dạy trước môn  $i_3, \dots$ , môn  $i_{k-1}$  phải dạy trước môn  $i_k$ , môn  $i_k$  phải dạy trước môn  $i_1$ . Hãy viết chương trình với tên KT3.PAS làm các việc sau:

1. Hãy xét xem mảng  $A$  có bế tắc hay không.
2. Nếu mảng  $A$  không bế tắc, hãy tính xem khóa học có thể kết thúc trong thời gian nhanh nhất là bao nhiêu ngày.
3. Theo các học bảo đảm thời gian hoàn thành ngắn nhất ở câu 2, hãy tính xem một học sinh trong quá trình học phải học đồng thời trong một ngày nhiều nhất bao nhiêu môn.

Dữ liệu vào được cho bởi file text có tên MH.DAT trong đó số  $N$  ghi ở dòng thứ nhất, trong nhóm  $N$  dòng tiếp theo, dòng thứ  $i$  ghi  $N$  số  $A[i, 1], \dots, A[i, N]$  dòng cuối cùng ghi  $N$  số nguyên dương  $t_i$  không lớn hơn 30,  $1 \leq i \leq N; N \leq 30$ .

Kết quả ghi ra file TKB.DAT như sau: dòng thứ nhất ghi số 1/0 tùy theo mảng  $A$  bế tắc / không bế tắc. Nếu dòng thứ nhất ghi số 0, ta mới ghi tiếp kết quả câu 2 và 3.

Kết quả câu 2 ghi tiếp vào file TKB.DAT  $N+1$  dòng như sau: dòng đầu ghi số  $T$  là số ngày tối thiểu có thể hoàn thành khóa học, tiếp theo là  $N$  dòng trong đó dòng thứ  $i$  ghi 2 số  $X, Y$  với ý nghĩa môn học thứ  $i$  học từ ngày thứ  $X$  đến ngày thứ  $Y$  (chú ý rằng  $Y - X = t_i - 1$ ). Kết quả câu 3 ghi tiếp vào file TKB.DAT như sau: dòng thứ nhất ghi 2 số  $Z, W$  với ý nghĩa trong ngày  $Z$  phải học  $W$  môn ( $W$  là số nhiều nhất các môn học phải học đồng thời trong một ngày), tiếp theo là một dòng ghi tên các môn học phải học đồng thời trong ngày  $Z$ . Trong các câu 2 và 3, có thể có nhiều lời giải tương đương chỉ cần đưa ra một lời giải.



Ví dụ 1

MH.DAT	TKB.DAT
4	1
0 1 0 0	
0 0 1 0	
0 0 0 1	
1 0 0 0	
1 1 1 1	

Ví dụ 2

MH.DAT	TKB.DAT
7	0
0100000	22
0001000	1 2
0001000	3 4
0000110	1 8
0000000	12
0000001	13 22
0000000	13 14
22841023	15 17
	1 2
	1 3

## Bài 2:

Giám đốc một công ty quyết định tổ chức buổi tiệc trà gặp gỡ toàn thể nhân viên trong công ty. Công ty được tổ chức theo mô hình phân cấp lãnh đạo và mỗi quan hệ *thủ trưởng – nhân viên* tạo thành cây có gốc là giám đốc. Để đảm bảo không khí tự nhiên, giám đốc quyết định không để thủ trưởng và nhân viên dưới quyền ngồi cùng một bàn. P gọi là thủ trưởng của Q, nếu P là thủ trưởng trực tiếp của Q hoặc tồn tại dãy  $P_1, P_2, \dots, P_k$  ( $1 < k$ ), sao cho  $P = P_1$ ,  $Q = P_k$  và  $P_i$  là thủ trưởng trực tiếp của  $P_{i+1}$  ( $i = 1, 2, \dots, k-1$ ). Tất cả mọi người trong công ty được đánh số từ 1 đến N (N là tổng số người trong công ty với giám đốc bắt đầu từ 1).

**Yêu cầu:** tính số lượng bàn ít nhất cần thiết để có thể bố trí cho mọi người ngồi theo yêu cầu nêu trên và cho một phương án bố trí người ở mỗi bàn.

**Dữ liệu vào:** file text COMPANY.INP, dòng đầu tiên là số nguyên m – số ghế tối đa cho một bàn, dòng thứ 2 – số nguyên N – số người trong công ty, dòng thứ ba (và các dòng sau nếu cần) là dãy số nguyên, các số cách nhau ít nhất một dấu cách hoặc nhóm ký tự xuống dòng, số nguyên thứ i trong dãy cho biết ai là thủ trưởng trực tiếp của nhân viên i. Giám đốc không có thủ trưởng nên số này bằng 0.

$2 \leq m \leq 10, 1 \leq N \leq 200$ .

**Kết quả:** đưa ra file text COMPANY.OUT, dòng đầu tiên là số bàn ít nhất cần thiết, các dòng sau mỗi dòng tương ứng với một bàn và chứa dãy số nguyên xác định ai được bố trí ngồi sau bàn đó.

Ví dụ:

COMPANY.INP

4

13

0 1 9 9 9 2 2 1 1 7 8 8 10

File kết quả COMPANY.OUT có thể có nội dung:

5

13 3 4 5

10 6 8

7 9 11 12

2

1

### Bài 3:

Trên bàn cờ  $N \times N$ , hãy tìm cách sắp xếp số lượng tối đa các con hậu sao cho không con nào có thể ăn con nào.

### Bài 4:

Cho 1 đồ thị có hướng  $G$ , hãy tìm một tập hợp  $X_0$  ít nhất các đỉnh của  $G$  sao cho mọi đỉnh  $i$  của  $G$  hoặc thuộc  $X_0$  hoặc  $i$  nối trực tiếp với đỉnh  $j$  thuộc  $X_0$ . Làm bài 14 trong trường hợp  $G$  vô hướng.

### **Bài 5:**

Trên bàn cờ  $N \times N$ , hãy tìm cách sắp xếp số lượng tối thiểu các con hậu sao cho mọi ô cờ trên bàn cờ bị chiếu bởi ít nhất 1 con.

### **Bài 6:**

Một ký túc xá nuôi 15 cô gái. Hàng ngày các cô đi chơi với nhau theo bộ 3. Hỏi có thể đưa các cô đi chơi trong tối đa bao nhiêu ngày để không có 2 cô nào đi chung trong một bộ 3 quá 1 lần. Hãy tổng quát hóa bài toán.

### **Bài 7:**

Trong 1 trại giam ở thành phố A, mỗi nhà giam có một trạm gác độc lập, nhưng người đứng gác, chẳng hạn ở nhà giam  $x_0$ , cũng có thể thấy những gì xảy ra ở các trại giam  $x_1, x_2, x_3, \dots$  khác do các nhà giam này thông với  $x_0$  bởi 1 hành lang thẳng. Giả sử biết các thông tin trên, hãy xác định số lượng tối thiểu các lính canh để có thể quan sát được mọi nhà giam.

### **Bài 8:**

Một số hải cảng  $x_1, x_2, x_3, \dots$  có các mặt hàng mà các hải cảng  $y_1, y_2, y_3, \dots$  cần đến. Lượng hàng có ở  $x_i$  là  $s_i$  và yêu cầu hàng hóa của  $y_j$  là  $d_j$ . Nếu có tàu đi từ  $x_i$  tới  $y_j$  thì ta ký hiệu  $c_{ij}$  là tổng lượng hàng mà các tàu có thể vận chuyển từ  $x_i$  tới  $y_j$ . Vậy có thể thỏa mãn mọi yêu cầu không? Tổ chức vận chuyển ra sao? Hãy viết chương trình giải quyết bài toán trên.

### **Bài 9:**

Trong một cuộc du lịch,  $m$  gia đình phân nhau đi trên  $n$  xe. Các gia đình tương ứng có  $r_1, r_2, \dots, r_m$  người và các xe tương ứng có  $s_1, s_2, \dots, s_n$  chỗ ngồi. Hãy tìm cách phân phối sao cho 2 người cùng gia đình không ngồi chung một xe hoặc cho biết không thể làm như vậy.

### **Bài 10:**

Trong một trường trung học, mỗi học sinh nữ có  $m$  bạn nam và mỗi học sinh nam có  $m$  bạn nữ. Hãy chỉ ra cách sắp xếp để mỗi cô gái có thể lần lượt khiêu vũ với các bạn trai của mình và các chàng trai có thể lần lượt khiêu vũ với các bạn gái của mình.

### **Bài 11:**

Một nhà in phải sản xuất  $n$  cuốn sách bằng 2 máy: một để in, một để đóng sách. Gọi  $a_k$  là thời gian cần cho việc in cuốn thứ  $k$  và  $b_k$  là thời gian cần cho việc đóng cuốn đó. Tất nhiên là sách phải in xong mới đóng, do đó máy đóng có thể phải chờ đợi lâu hay chóng. Vậy tiến hành theo thứ tự nào để có thể xong việc sớm nhất.

### **Bài 12: Ổn định**

Trong một lớp có  $N$  dãy bàn và mỗi dãy có  $M$  chỗ ngồi. Trong lớp có  $K$  cán sự lớp. Mỗi một cán sự cầm một đề bài tập. Các cán sự này có nhiệm vụ chuyển đề bài tập đến các học sinh khác ngồi kề mình ở phía trước, sau, trái và phải. Sau khi các cán sự làm xong công việc của mình, mỗi học sinh thông báo số lượng đề bài tập mình đã nhận được. Dựa trên thông tin này hãy xác định vị trí của các cán sự trong lớp.

### **Bài 13:**

Có một máy thu và một máy phát tín hiệu. Giả sử máy phát có thể phát đi 5 loại tín hiệu khác nhau  $a, b, c, d, e$ . Ở máy thu mỗi tín hiệu có thể được hiểu theo 2 cách khác nhau: tín hiệu  $a$  có thể hiểu  $p$  hay  $q$ , tín hiệu  $b$  có thể hiểu  $q$  hay  $r$ , ... Số cực đại các tín hiệu mà ta có thể sử dụng là bao nhiêu để cho ở máy thu không xảy ra nhầm lẫn giữa các tín hiệu được sử dụng.

### **Bài 14:**

Cho 1 đồ thị vô hướng  $G$ . Người ta muốn tô các đỉnh của  $G$  bằng 2 màu đen, trắng sao cho 2 đỉnh kề nhau không được tô bởi cùng 1 màu. Hãy cho biết có thể làm được điều này không. Nếu được hãy chỉ ra cách tô.

### **Bài 15:**

Cho một đồ thị không định hướng trong một tệp văn bản với cách mã hóa như sau

Dòng đầu là số đỉnh ( $n$ ). Các đỉnh được xem đánh số liên tiếp từ 1 đến  $n$ .

Mỗi dòng sau là mô tả một cạnh cho bằng chỉ số 2 đỉnh là đầu mút của cạnh đó.

Yêu cầu tô màu một số đỉnh thành màu đỏ sao cho số đỉnh đỏ là lớn nhất nhưng không có cạnh nào được phép có cả hai đầu mút là các đỉnh đỏ. Kết quả cho ra trên hai dòng. Dòng thứ nhất là số đỉnh màu đỏ, dòng thứ hai là số thứ tự các đỉnh có màu đỏ.

## Bài 16:

Giám đốc một công ty quyết định tổ chức buổi tiệc trà gặp gỡ toàn thể nhân viên trong công ty. Công ty được tổ chức theo mô hình phân cấp lãnh đạo và mỗi quan hệ *thủ trưởng – nhân viên* tạo thành cây có gốc là *giám đốc*. Để đảm bảo không khí tự nhiên, giám đốc quyết định không để thủ trưởng và nhân viên dưới quyền ngồi cùng một bàn. P gọi là thủ trưởng của Q, nếu là P thủ trưởng trực tiếp của Q hoặc tồn tại dãy  $P_1, P_2, \dots, P_k$  ( $1 < k$ ), sao cho  $P = P_1$ ,  $Q = P_k$  và  $P_i$  là thủ trưởng trực tiếp của  $P_{i+1}$  ( $i = 1, 2, \dots, k-1$ ). Tất cả mọi người trong công ty được đánh số từ 1 đến  $N$  – tổng số người trong công ty, bắt đầu từ giám đốc với số là 1.

Yêu cầu tính số lượng bàn *ít nhất* cần thiết để có thể bố trí cho mọi người ngồi theo yêu cầu nêu trên và cho một phương án bố trí người ở mỗi bàn.

**Dữ liệu vào :** File ASCII COMPANY.INP, dòng đầu tiên là số nguyên  $m$  – số ghế tối đa cho một bàn, dòng thứ 2 – số nguyên  $N$  – số người trong công ty, dòng thứ 3 (và các dòng sau nếu cần) là dãy số nguyên, các số cách nhau ít nhất một dấu cách hoặc nhóm ký tự xuống dòng, số nguyên thứ  $i$  trong dãy cho biết ai là thủ trưởng trực tiếp của nhân viên  $i$ . Giám đốc không có thủ trưởng nên số này bằng 0.  $2 \leq m \leq 10$ ,  $1 \leq N \leq 200$ .

**Kết quả :** đưa ra file ASCII COMPANY.OUT, dòng đầu tiên là số bàn ít nhất cần thiết, các dòng sau mỗi dòng tương ứng với một bàn và chứa dãy số nguyên xác định ai được bố trí ngồi sau bàn đó.

Ví dụ :

COMPANY.INP

4

13

0 1 9 9 9 2 2 1 1 7 8 8 10

File kết quả COMPANY.OUT có thể có nội dung :

5

13 3 4 5

10 6 8

7 9 11 12

2

1

### Bài 17:

Một hệ thống có N ( $N \leq 50$ ) thành phố (được đánh số từ 1 đến N) được nối với nhau bằng một hệ thống giao thông được cho bằng các đường nối trực tiếp các cặp hai thành phố: mỗi đoạn nối trực tiếp có độ dài là 1 và là đường đi hai chiều. Hệ thống này được gọi là "chẵn-lẻ" nếu như khi hai thành phố được nối với nhau (có thể qua các thành phố khác) bởi một đường đi với **độ dài chẵn** thì luôn tìm được **đường đi có độ dài lẻ** nối hai thành phố đó. Các bài toán được đặt ra :

**Câu 1 :** Với một hệ thống giao thông đã cho, kiểm tra xem nó có tính "chẵn-lẻ" hay không ?

**Câu 2 :** Nếu câu trả lời của câu 1 là phủ định thì hãy chỉ ra tập con X trong hệ thống các thành phố nói trên và có nhiều nhất các thành phố mà đảm bảo điều kiện sau: Nếu có đường đi nối hai thành phố bất kỳ trong X thì độ dài đường đi đó phải chẵn (giữ nguyên hệ thống giao thông).

Điều kiện kỹ thuật :

File dữ liệu vào tên CHANLE.INP:

Dòng đầu tiên chứa giá trị N.

Các dòng tiếp theo chứa lượng chẵn các số nguyên không âm mà hai số nguyên cuối cùng là 0, còn mọi số nguyên khác chỉ là chỉ số thành phố; từng cặp hai số nguyên sẽ chỉ ra số hiệu c3a hai thành phố được nối trực tiếp với nhau.

File kết quả ra có tên CHANLE.OUT:

Dòng đầu có số 0 hoặc 1 cho câu 1, trong đó 1 chỉ ra rằng hệ thống nói trên có tính chẵn lẻ, 0 phủ định.

Nếu có kết quả phủ định thì các dòng tiếp theo có dạng :

+ Dòng thứ 2 có số lượng phần tử của X.

+ Các dòng tiếp theo (từ dòng thứ 3 trở đi) chứa số hiệu các thành phố có trong X.

Ví dụ:

CHANLE.INP có nội dung :

4

1 2 1 3 0 0

CHANLE.OUT

0

3

2 3 4

Nếu file CHANLE.INP có nội dung :

5

1 2 2 3

3 4 4 5 5 1

0 0

CHANLE.OUT

1



## Tham gia đóng góp

Tài liệu: Giáo trình lý thuyết đồ thị

Biên tập bởi: Thạc sĩ Nguyễn Thanh Hùng

URL: <http://voer.edu.vn/c/9c021e14>

Giấy phép: <http://creativecommons.org/licenses/by/3.0/>

Module: Các khái niệm cơ bản của lý thuyết đồ thị

Các tác giả: Thạc sĩ Nguyễn Thanh Hùng

URL: <http://www.voer.edu.vn/m/dff2cfd5>

Giấy phép: <http://creativecommons.org/licenses/by/3.0/>

Module: Các thuật ngữ cơ bản

Các tác giả: Thạc sĩ Nguyễn Thanh Hùng

URL: <http://www.voer.edu.vn/m/8bb57be5>

Giấy phép: <http://creativecommons.org/licenses/by/3.0/>

Module: Đường đi, chu trình, độ liên thông

Các tác giả: Thạc sĩ Nguyễn Thanh Hùng

URL: <http://www.voer.edu.vn/m/89915f67>

Giấy phép: <http://creativecommons.org/licenses/by/3.0/>

Module: Một số dạng đồ thị đặc biệt

Các tác giả: Thạc sĩ Nguyễn Thanh Hùng

URL: <http://www.voer.edu.vn/m/dc7bfbe3>

Giấy phép: <http://creativecommons.org/licenses/by/3.0/>

Module: Biểu diễn đồ thị trên máy vi tính

Các tác giả: Thạc sĩ Nguyễn Thanh Hùng

URL: <http://www.voer.edu.vn/m/a2d8be8f>

Giấy phép: <http://creativecommons.org/licenses/by/3.0/>

Module: Danh sách cạnh(cung)

Các tác giả: Thạc sĩ Nguyễn Thanh Hùng

URL: <http://www.voer.edu.vn/m/59983be8>

Giấy phép: <http://creativecommons.org/licenses/by/3.0/>

Module: Danh sách kề

Các tác giả: Thạc sĩ Nguyễn Thanh Hùng

URL: <http://www.voer.edu.vn/m/68b497ec>

Giấy phép: <http://creativecommons.org/licenses/by/3.0/>

Module: Các thuật toán tìm kiếm trên đồ thị và ứng dụng

Các tác giả: Thạc sĩ Nguyễn Thanh Hùng

URL: <http://www.voer.edu.vn/m/ace8b5e6>

Giấy phép: <http://creativecommons.org/licenses/by/3.0/>

Module: Tìm kiếm theo chiều rộng trên đồ thị

Các tác giả: Thạc sĩ Nguyễn Thanh Hùng

URL: <http://www.voer.edu.vn/m/89d6341f>

Giấy phép: <http://creativecommons.org/licenses/by/3.0/>

Module: Tìm đường đi và kiểm tra tính liên thông

Các tác giả: Thạc sĩ Nguyễn Thanh Hùng

URL: <http://www.voer.edu.vn/m/d2dc68dc>

Giấy phép: <http://creativecommons.org/licenses/by/3.0/>

Module: Đồ thị Euler và đồ thị Hamiton

Các tác giả: Thạc sĩ Nguyễn Thanh Hùng

URL: <http://www.voer.edu.vn/m/86e257a7>

Giấy phép: <http://creativecommons.org/licenses/by/3.0/>

Module: Đồ thị Hamilton

Các tác giả: Thạc sĩ Nguyễn Thanh Hùng

URL: <http://www.voer.edu.vn/m/bfd2a4cd>

Giấy phép: <http://creativecommons.org/licenses/by/3.0/>

Module: Cây và cây khung của đồ thị

Các tác giả: Thạc sĩ Nguyễn Thanh Hùng

URL: <http://www.voer.edu.vn/m/0e5f8db1>

Giấy phép: <http://creativecommons.org/licenses/by/3.0/>

Module: Cây khung của đồ thị

Các tác giả: Thạc sĩ Nguyễn Thanh Hùng

URL: <http://www.voer.edu.vn/m/38e67226>

Giấy phép: <http://creativecommons.org/licenses/by/3.0/>

Module: Xây dựng tập các chu trình cơ bản của đồ thị

Các tác giả: Thạc sĩ Nguyễn Thanh Hùng

URL: <http://www.voer.edu.vn/m/c81ff79e>

Giấy phép: <http://creativecommons.org/licenses/by/3.0/>

Module: Bài toán cây khung nhỏ nhất

Các tác giả: Thạc sĩ Nguyễn Thanh Hùng

URL: <http://www.voer.edu.vn/m/a471d510>

Giấy phép: <http://creativecommons.org/licenses/by/3.0/>

Module: Bài toán đường đi ngắn nhất

Các tác giả: Thạc sĩ Nguyễn Thanh Hùng

URL: <http://www.voer.edu.vn/m/379a7e15>

Giấy phép: <http://creativecommons.org/licenses/by/3.0/>

Module: Đường đi ngắn nhất xuất phát từ một đỉnh

Các tác giả: Thạc sĩ Nguyễn Thanh Hùng

URL: <http://www.voer.edu.vn/m/8a2a4473>

Giấy phép: <http://creativecommons.org/licenses/by/3.0/>

Module: Trường hợp ma trận trọng số không âm-Thuật toán Dijkstra

Các tác giả: Thạc sĩ Nguyễn Thanh Hùng

URL: <http://www.voer.edu.vn/m/6596ee64>

Giấy phép: <http://creativecommons.org/licenses/by/3.0/>

Module: Đồ thị trong đồ thị không có chu trình

Các tác giả: Thạc sĩ Nguyễn Thanh Hùng

URL: <http://www.voer.edu.vn/m/9907bd7f>

Giấy phép: <http://creativecommons.org/licenses/by/3.0/>

Module: Đường đi ngắn nhất giữa tất cả các cặp đỉnh

Các tác giả: Thạc sĩ Nguyễn Thanh Hùng

URL: <http://www.voer.edu.vn/m/0458afc2>

Giấy phép: <http://creativecommons.org/licenses/by/3.0/>

Module: Bài toán luồng cực đại trong mạng

Các tác giả: Thạc sĩ Nguyễn Thanh Hùng

URL: <http://www.voer.edu.vn/m/6ecc9764>

Giấy phép: <http://creativecommons.org/licenses/by/3.0/>

Module: Lát cắt.Đường tăng luồng.Định lý Ford\_Fulkerson

Các tác giả: Thạc sĩ Nguyễn Thanh Hùng

URL: <http://www.voer.edu.vn/m/e53b84ad>

Giấy phép: <http://creativecommons.org/licenses/by/3.0/>

Module: Thuật toán tìm luồng cực đại

Các tác giả: Thạc sĩ Nguyễn Thanh Hùng

URL: <http://www.voer.edu.vn/m/b8002118>

Giấy phép: <http://creativecommons.org/licenses/by/3.0/>

Module: Một số bài toán luồng tổng quát

Các tác giả: Thạc sĩ Nguyễn Thanh Hùng

URL: <http://www.voer.edu.vn/m/71b4fb8b>

Giấy phép: <http://creativecommons.org/licenses/by/3.0/>

Module: Một số ứng dụng trong tổ hợp

Các tác giả: Thạc sĩ Nguyễn Thanh Hùng

URL: <http://www.voer.edu.vn/m/24536c8c>

Giấy phép: <http://creativecommons.org/licenses/by/3.0/>

Module: Bài tập chương 3

Các tác giả: Thạc sĩ Nguyễn Thanh Hùng

URL: <http://www.voer.edu.vn/m/2e1f015b>

Giấy phép: <http://creativecommons.org/licenses/by/3.0/>

Module: Bài tập chương 4

Các tác giả: Thạc sĩ Nguyễn Thanh Hùng

URL: <http://www.voer.edu.vn/m/81c4fff8>

Giấy phép: <http://creativecommons.org/licenses/by/3.0/>

Module: Bài tập chương 5

Các tác giả: Thạc sĩ Nguyễn Thanh Hùng

URL: <http://www.voer.edu.vn/m/2ebf2242>

Giấy phép: <http://creativecommons.org/licenses/by/3.0/>

Module: Bài tập chương 6

Các tác giả: Thạc sĩ Nguyễn Thanh Hùng

URL: <http://www.voer.edu.vn/m/b3ee2dc1>

Giấy phép: <http://creativecommons.org/licenses/by/3.0/>

Module: Bài tập chương 7

Các tác giả: Thạc sĩ Nguyễn Thanh Hùng

URL: <http://www.voer.edu.vn/m/1f563e19>

Giấy phép: <http://creativecommons.org/licenses/by/3.0/>

Module: Các bài tập khác

Các tác giả: Thạc sĩ Nguyễn Thanh Hùng

URL: <http://www.voer.edu.vn/m/5c9f35e1>

Giấy phép: <http://creativecommons.org/licenses/by/3.0/>

## **Chương trình Thư viện Học liệu Mở Việt Nam**

Chương trình Thư viện Học liệu Mở Việt Nam (Vietnam Open Educational Resources – VOER) được hỗ trợ bởi Quỹ Việt Nam. Mục tiêu của chương trình là xây dựng kho Tài nguyên giáo dục Mở miễn phí của người Việt và cho người Việt, có nội dung phong phú. Các nội dung đều tuân thủ Giấy phép Creative Commons Attribution (CC-by) 4.0 do đó các nội dung đều có thể được sử dụng, tái sử dụng và truy nhập miễn phí trước hết trong môi trường giảng dạy, học tập và nghiên cứu sau đó cho toàn xã hội.

Với sự hỗ trợ của Quỹ Việt Nam, Thư viện Học liệu Mở Việt Nam (VOER) đã trở thành một cổng thông tin chính cho các sinh viên và giảng viên trong và ngoài Việt Nam. Mỗi ngày có hàng chục nghìn lượt truy cập VOER ([www.voer.edu.vn](http://www.voer.edu.vn)) để nghiên cứu, học tập và tải tài liệu giảng dạy về. Với hàng chục nghìn module kiến thức từ hàng nghìn tác giả khác nhau đóng góp, Thư Viện Học liệu Mở Việt Nam là một kho tàng tài liệu khổng lồ, nội dung phong phú phục vụ cho tất cả các nhu cầu học tập, nghiên cứu của độc giả.

Nguồn tài liệu mở phong phú có trên VOER có được là do sự chia sẻ tự nguyện của các tác giả trong và ngoài nước. Quá trình chia sẻ tài liệu trên VOER trở nên dễ dàng như đếm 1, 2, 3 nhờ vào sức mạnh của nền tảng Hanoi Spring.

Hanoi Spring là một nền tảng công nghệ tiên tiến được thiết kế cho phép công chúng dễ dàng chia sẻ tài liệu giảng dạy, học tập cũng như chủ động phát triển chương trình giảng dạy dựa trên khái niệm về học liệu mở (OCW) và tài nguyên giáo dục mở (OER). Khái niệm chia sẻ tri thức có tính cách mạng đã được khởi xướng và phát triển tiên phong bởi Đại học MIT và Đại học Rice Hoa Kỳ trong vòng một thập kỷ qua. Kể từ đó, phong trào Tài nguyên Giáo dục Mở đã phát triển nhanh chóng, được UNESCO hỗ trợ và được chấp nhận như một chương trình chính thức ở nhiều nước trên thế giới.