

BỘ GIÁO DỤC VÀ ĐÀO TẠO  
TRƯỜNG ĐẠI HỌC TÂY ĐÔ



**BÀI GIẢNG**  
**TOÁN RỜI RẠC 2**

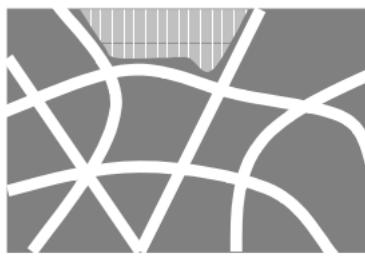
**Biên soạn: TS. Nguyễn Hữu Danh  
ThS. Nguyễn Chí Thắng**

**Năm 2023**

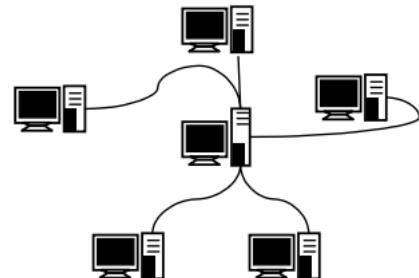
# Chương 1      **Những khái niệm cơ bản của lý thuyết đồ thị**

## 1.1 Định nghĩa đồ thị

Đồ thị là một cấu trúc rời rạc gồm các đỉnh và các cạnh (vô hướng hoặc có hướng) nối các đỉnh đó. Người ta phân loại đồ thị tùy theo đặc tính và số các cạnh nối các cặp đỉnh của đồ thị.

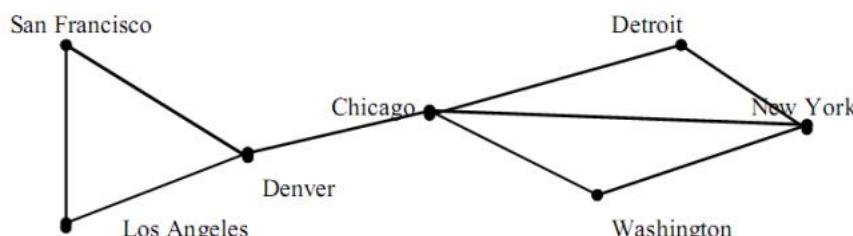


Sơ đồ giao thông

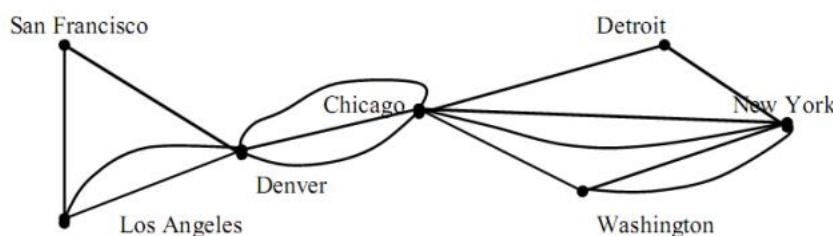


Mạng máy tính

**Định nghĩa 1.1** Một đơn đồ thị vô hướng  $G = (V, E)$  gồm một tập khác rỗng  $V$  mà các phần tử của nó gọi là các đỉnh và một tập  $E$  mà các phần tử của nó gọi là các cạnh, đó là các cặp không có thứ tự của các đỉnh phân biệt.



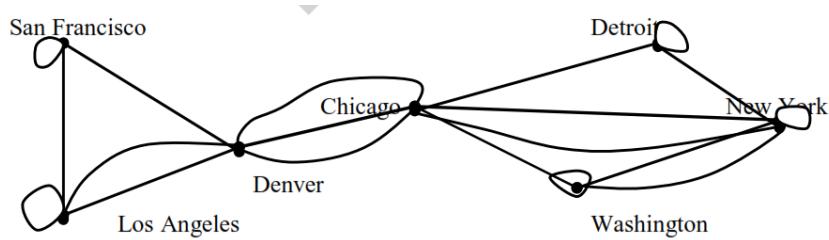
**Định nghĩa 1.2** Một đa đồ thị vô hướng  $G = (V, E)$  gồm một tập khác rỗng  $V$  mà các phần tử của nó gọi là các đỉnh và một tập  $E$  mà các phần tử của nó gọi là các cạnh, đó là các cặp không có thứ tự của các đỉnh phân biệt. Hai cạnh được gọi là cạnh bội hay song song nếu chúng cùng tương ứng với một cặp đỉnh.



Rõ ràng mỗi đơn đồ thị là đa đồ thị, nhưng không phải đa đồ thị nào cũng là đơn đồ thị.

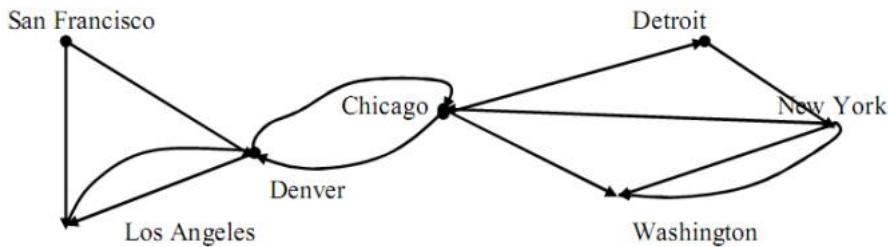
**Định nghĩa 1.3** Một giả đồ thị vô hướng  $G = (V, E)$  gồm một tập khác rỗng  $V$  mà các phần tử của nó gọi là các đỉnh và một tập  $E$  mà các phần tử của nó gọi là các cạnh, đó là các cặp không có thứ tự của các đỉnh (không nhất thiết là phân biệt).

Với  $v \in V$ , nếu  $(v, v) \in E$  thì ta nói có một khuyên tại đỉnh  $v$ .



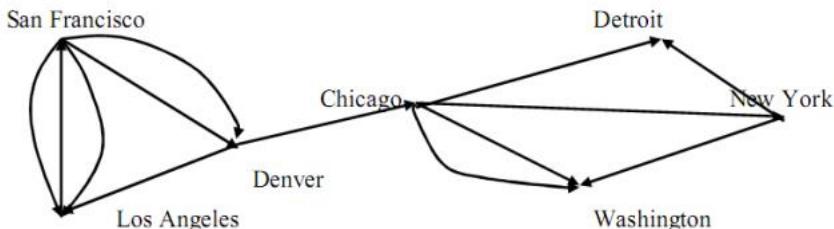
Tóm lại, giả đồ thị vô hướng là loại đồ thị vô hướng tổng quát nhất vì nó có thể chứa các khuyên và các cạnh bội. Đa đồ thị là loại đồ thị vô hướng có thể chứa cạnh bội nhưng không thể có các khuyên, còn đơn đồ thị là loại đồ thị vô hướng không chứa cạnh bội hoặc các khuyên.

**Định nghĩa 1.4** Một đơn đồ thị có hướng  $G = (V, E)$  gồm một tập khác rỗng  $V$  mà các phần tử của nó gọi là các đỉnh và một tập  $E$  mà các phần tử của nó gọi là các cung, đó là các cặp có thứ tự của các phần tử thuộc  $V$ .



**Định nghĩa 1.5** Một đa đồ thị có hướng  $G = (V, E)$  gồm một tập khác rỗng  $V$  mà các phần tử của nó gọi là các đỉnh và một tập  $E$  mà các phần tử của nó gọi là các cung, đó là các cặp có thứ tự của các phần tử thuộc  $V$ .

Đồ thị vô hướng nhận được từ đồ thị có hướng  $G$  bằng cách xoá bỏ các chiều mũi tên trên các cung được gọi là đồ thị vô hướng nền của  $G$ .



## Phân biệt

Loại đồ thị	Cạnh	Có cạnh bội	Có khuyên
Đơn đồ thị vô hướng	Vô hướng	Không	Không
Đa đồ thị vô hướng	Vô hướng	Có	Không
Giả đồ thị vô hướng	Vô hướng	Có	Có
Đơn đồ thị có hướng	Có hướng	Không	Không
Đa đồ thị có hướng	Có hướng	Có	Có

**Ví dụ:**

**1) Thi đấu vòng tròn.** Một cuộc thi đấu thể thao trong đó mỗi đội đấu với mỗi đội khác đúng một lần gọi là đấu vòng tròn. Cuộc thi đấu như thế có thể được mô hình bằng một đồ thị có hướng trong đó mỗi đội là một đỉnh. Một cung đi từ đỉnh a đến đỉnh b nếu đội a thắng đội b.

**2) Các chương trình máy tính** có thể thi hành nhanh hơn bằng cách thi hành đồng thời một số câu lệnh nào đó. Điều quan trọng là không được thực hiện một câu lệnh đòi hỏi kết quả của câu lệnh khác chưa được thực hiện. Sự phụ thuộc của các câu lệnh vào các câu lệnh trước có thể biểu diễn bằng một đồ thị có hướng. Mỗi câu lệnh được biểu diễn bằng một đỉnh và có một cung từ một đỉnh tới một đỉnh khác nếu câu lệnh được biểu diễn bằng đỉnh thứ hai không thể thực hiện được trước khi câu lệnh được biểu diễn bằng đỉnh thứ nhất được thực hiện. Đồ thị này được gọi là **đồ thị có ưu tiên trước sau**.

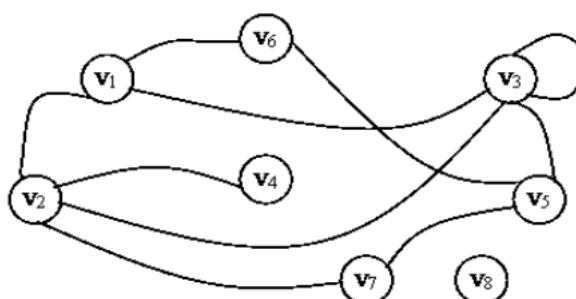
## 1.2 Các thuật ngữ cơ bản

**Định nghĩa 1.6** Hai đỉnh u và v trong đồ thị (vô hướng)  $G = (V, E)$  được gọi là liền kề nếu  $(u, v) \in E$ . Nếu  $e = (u, v)$  thì e gọi là cạnh liên thuộc với các đỉnh u và v. Cạnh e cũng được gọi là cạnh nối các đỉnh u và v. Các đỉnh u và v gọi là các điểm đầu mút của cạnh e.

**Định nghĩa 1.7** Bậc của đỉnh v trong đồ thị  $G = (V, E)$ , ký hiệu  $\deg(v)$ , là số các cạnh liên thuộc với nó, riêng **khuyên** tại một đỉnh được tính **hai lần** cho bậc của nó.

Đỉnh v gọi là đỉnh treo nếu  $\deg(v) = 1$  và gọi là đỉnh cô lập nếu  $\deg(v) = 0$ .

**Ví dụ:**



Ta có  $\deg(v_1) = 3$ ,  $\deg(v_2) = 4$ ,  $\deg(v_3) = 5$ ,  $\deg(v_4) = 1$ ,

$\deg(v_5) = 3$ ,  $\deg(v_6) = 2$ ,  $\deg(v_7) = 2$ . Đỉnh  $v_8$  là đỉnh cô lập và  $v_4$  là đỉnh treo.

**Mệnh đề 1.8** Cho đồ thị  $G = (V, E)$ . Khi đó

$$2|E| = \sum_{v \in V} \deg(v).$$

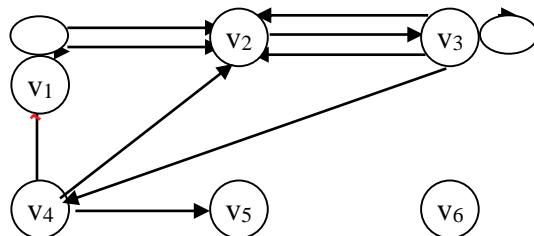
**Hệ quả 1.9** Số đỉnh bậc lẻ của một đồ thị là một số chẵn.

**Mệnh đề 1.10** Trong một đồ thị, luôn tồn tại hai đỉnh có cùng bậc.

**Định nghĩa 1.11** Đỉnh u được gọi là nối tới v hay v được gọi là được nối từ u trong đồ thị có hướng G nếu  $(u, v)$  là một cung của G. Đỉnh u gọi là đỉnh đầu và đỉnh v gọi là đỉnh cuối của cung này.

**Định nghĩa 1.12** Độ vào (t.ú. bậc ra) của đỉnh v trong đồ thị có hướng G, ký hiệu  $\deg^-(v)$  (t.ú.  $\deg^+(v)$ ), là số các cung có đỉnh cuối (t.ú. đỉnh đầu) là v.

**Ví dụ:**



$$\deg^-(v_1) = 2, \deg^+(v_1) = 3,$$

$$\deg^-(v_2) = 5, \deg^+(v_2) = 1,$$

$$\deg^-(v_3) = 2, \deg^+(v_3) = 4,$$

$$\deg^-(v_4) = 1, \deg^+(v_4) = 3,$$

$$\deg^-(v_5) = 1, \deg^+(v_5) = 0,$$

$$\deg^-(v_6) = 0, \deg^+(v_6) = 0.$$

Đỉnh có bậc vào và bậc ra cùng bằng 0 gọi là đỉnh cô lập. Đỉnh có bậc vào bằng 1 và bậc ra bằng 0 gọi là đỉnh treo, cung có đỉnh cuối là đỉnh treo gọi là cung treo.

**Mệnh đề 1.13** Cho  $G = (V, E)$  là một đồ thị có hướng. Khi đó

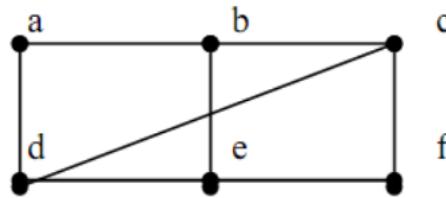
$$\sum_{v \in V} \deg^-(v) = \sum_{v \in V} \deg^+(v) = |E|.$$

### 1.3 Đường đi, chu trình, đồ thị liên thông

**Định nghĩa 1.14** Đường đi **độ dài n** từ đỉnh u đến đỉnh v trên đồ thị vô hướng  $G = (V, E)$  là dãy  $x_0, x_1, \dots, x_{n-1}, x_n$ , trong đó n là số nguyên dương,  $x_0 = u, x_n = v, (x_i, x_{i+1}) \in E, i = 0, 1, 2, \dots, n-1$ .

- Đường đi như trên còn có thể biểu diễn thành dãy các cạnh  $(x_0, x_1), (x_1, x_2), \dots, (x_{n-1}, x_n)$ .
- Đỉnh u là đỉnh đầu, đỉnh v là đỉnh cuối của đường đi.
- Đường đi có **đỉnh đầu trùng với đỉnh cuối** ( $u = v$ ) được gọi là **chu trình**.
- Đường đi hay **chu trình** được gọi là **đơn** nếu như **không có cạnh nào lặp lại**.

**Ví dụ:**

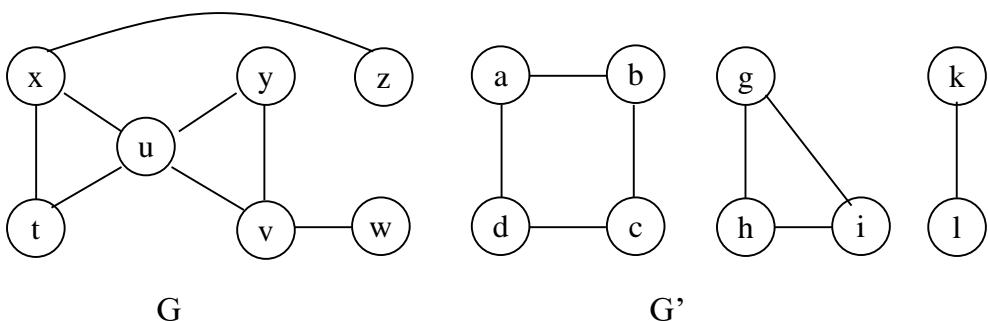


- a, d, c, f, e là đường đi đơn độ dài 4.
- d, e, c, a không là đường đi vì (e,c) không phải là cạnh của đồ thị.
- Dãy b, c, f, e, b là chu trình độ dài 4.
- Đường đi a, b, e, d, a, b có độ dài 5 không phải là đường đi đơn vì cạnh (a,b) có mặt hai lần.

**Định nghĩa 1.15** Một đồ thị (vô hướng) được gọi là **liên thông** nếu có đường đi giữa mọi cặp đỉnh phân biệt của đồ thị.

Một đồ thị không liên thông là hợp của hai hay nhiều đồ thị con liên thông, mỗi cặp các **đồ thị con này không có đỉnh chung**. Các đồ thị con liên thông rời nhau như vậy được gọi là các thành phần liên thông của đồ thị đang xét. Như vậy, một đồ thị là liên thông khi và chỉ khi nó chỉ có một thành phần liên thông.

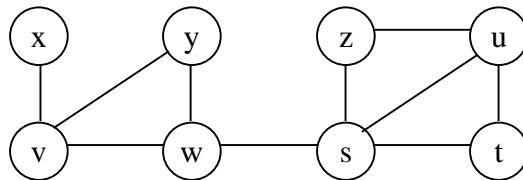
**Ví dụ:**



Đồ thị G là liên thông, nhưng đồ thị G' không liên thông và có 3 thành phần liên thông.

**Định nghĩa 1.16:** Một đỉnh trong đồ thị G mà khi xoá đi nó và tất cả các cạnh liên thuộc với nó ta nhận được đồ thị con mới có nhiều thành phần liên thông hơn đồ thị G được gọi là **đỉnh cắt hay điểm khop**. Việc xoá đỉnh cắt khỏi một đồ thị liên thông sẽ tạo ra một đồ thị con không liên thông. Hoàn toàn tương tự, một cạnh mà khi ta bỏ nó đi sẽ tạo ra một đồ thị có nhiều thành phần liên thông hơn so với đồ thị xuất phát được gọi là **cạnh cắt hay là cầu**.

**Ví dụ:**



Trong đồ thị trên, các đỉnh cắt là v, w, s và các cầu là (x,v), (w,s).

**Mệnh đề 1.17** Giữa mọi cặp đỉnh phân biệt của một đồ thị liên thông luôn có đường đi sơ cấp.

**Mệnh đề 1.18** Mọi đơn đồ thị  $n$  đỉnh ( $n \geq 2$ ) có tổng bậc của hai đỉnh tuy ý không nhỏ hơn  $n$  đều là đồ thị liên thông.

**Hệ quả 1.19** Đơn đồ thị mà bậc của mỗi đỉnh của nó không nhỏ hơn một nửa số đỉnh là đồ thị liên thông.

**Mệnh đề 1.20** Nếu một đồ thị có đúng hai đỉnh bậc lẻ thì hai đỉnh này phải liên thông, tức là có một đường đi nối chúng.

**Mệnh đề 1.21** Cho  $G = (V, E)$  là một đồ thị liên thông. Khi đó một đỉnh của  $G$  là điểm khorp khi và chỉ khi trong  $G$  tồn tại hai đỉnh  $u$  và  $v$  sao cho mỗi đường đi nối  $u$  và  $v$  đều phải đi qua đỉnh này.

**Định lý 1.22** Cho  $G$  là một đơn đồ thị có  $n$  đỉnh,  $m$  cạnh và  $k$  thành phần liên thông. Khi đó

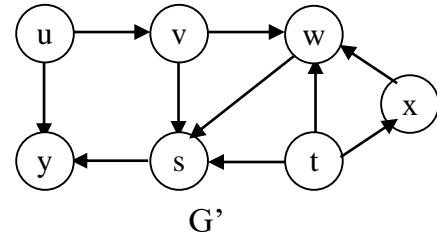
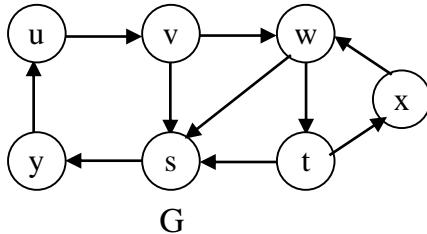
$$n - k \leq m \leq \frac{(n - k)(n - k + 1)}{2}.$$

**Định nghĩa 1.23** Đồ thị có hướng  $G$  được gọi là liên thông mạnh nếu với hai đỉnh phân biệt bất kỳ  $u$  và  $v$  của  $G$  đều có đường đi từ  $u$  tới  $v$  và đường đi từ  $v$  tới  $u$ .

Đồ thị có hướng  $G$  được gọi là liên thông yếu nếu đồ thị vô hướng nền của nó là liên thông.

Đồ thị có hướng  $G$  được gọi là liên thông một chiều nếu với hai đỉnh phân biệt bất kỳ  $u$  và  $v$  của  $G$  đều có đường đi từ  $u$  tới  $v$  hoặc đường đi từ  $v$  tới  $u$ .

**Ví dụ:**



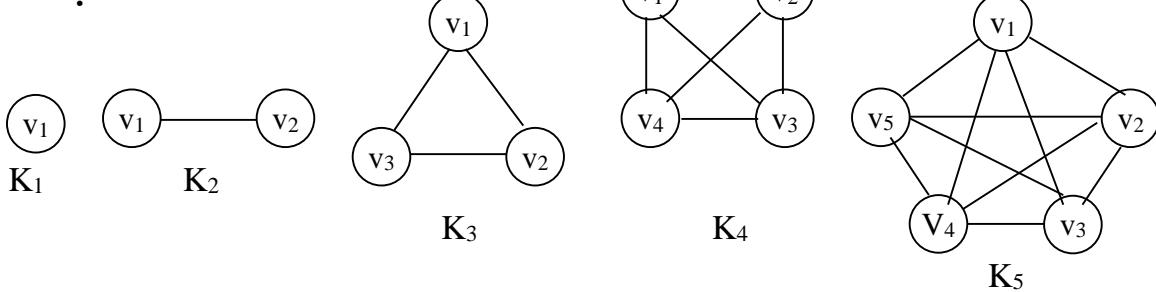
Đồ thị  $G$  là liên thông mạnh nhưng đồ thị  $G'$  là liên thông yếu (không có đường đi từ  $u$  tới  $x$  cũng như từ  $x$  tới  $u$ ).

**Mệnh đề 1.24** Cho  $G$  là một đồ thị (vô hướng hoặc có hướng) với ma trận liền kề  $A$  theo thứ tự các đỉnh  $v_1, v_2, \dots, v_n$ . Khi đó số các đường đi khác nhau độ dài  $r$  từ  $v_i$  tới  $v_j$  trong đó  $r$  là một số nguyên dương, bằng giá trị của phần tử dòng  $i$  cột  $j$  của ma trận  $A^r$ .

#### 1.4 Một số dạng đặc biệt của đồ thị

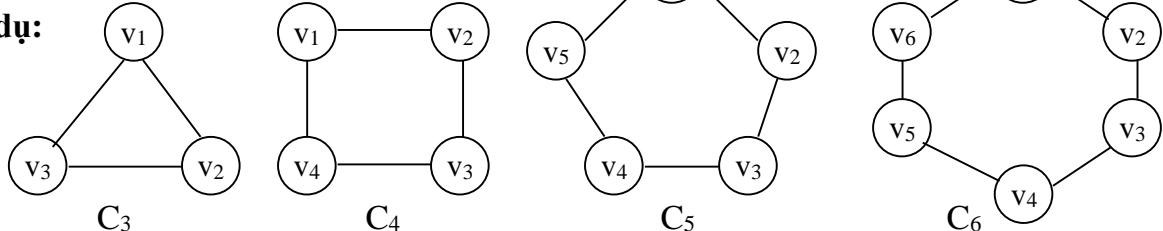
**Đồ thị đầy đủ:** Đồ thị đầy đủ  $n$  đỉnh, ký hiệu là  $K_n$ , là đơn đồ thị mà hai đỉnh phân biệt bất kỳ của nó luôn liền kề. Như vậy,  $K_n$  có  $\frac{n(n-1)}{2}$  cạnh và mỗi đỉnh của  $K_n$  có bậc là  $n - 1$ .

**Ví dụ:**



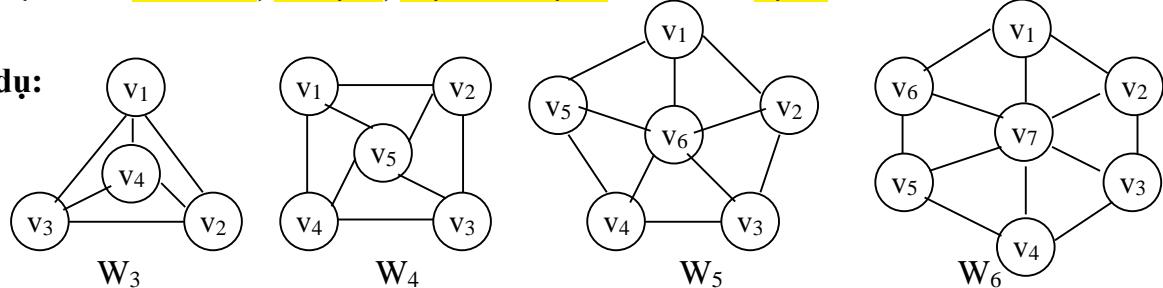
**Đồ thị vòng:** Đơn đồ thị **n** đỉnh  $v_1, v_2, \dots, v_n$  ( $n \geq 3$ ) và **n** cạnh  $(v_1, v_2), (v_2, v_3), \dots, (v_{n-1}, v_n), (v_n, v_1)$  được gọi là đồ thị vòng, ký hiệu là  $C_n$ . Như vậy, mỗi đỉnh của  $C_n$  có bậc là 2.

**Ví dụ:**



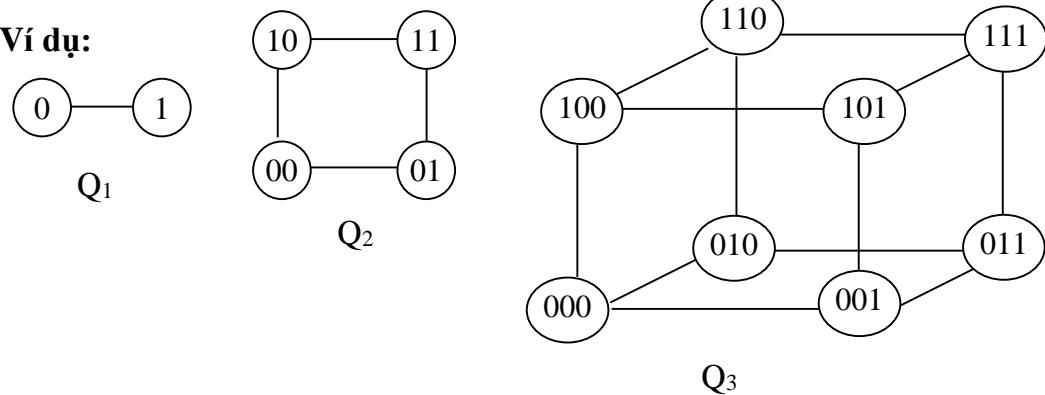
**Đồ thị bánh xe:** Từ đồ thị vòng  $C_n$ , thêm vào đỉnh  $v_{n+1}$  và các cạnh  $(v_{n+1}, v_1), (v_{n+1}, v_2), \dots, (v_{n+1}, v_n)$ , ta nhận được đơn đồ thị gọi là đồ thị bánh xe, ký hiệu là  $W_n$ . Như vậy, đồ thị  $W_n$  có  **$n+1$  đỉnh**,  **$2n$  cạnh**, **một đỉnh bậc  $n$**  và  **$n$  đỉnh bậc 3**.

**Ví dụ:**



**Đồ thị lập phương:** Đơn đồ thị  $2^n$  đỉnh, tương ứng với  $2^n$  xâu nhị phân độ dài  $n$  và hai đỉnh kề nhau khi và chỉ khi 2 xâu nhị phân tương ứng với hai đỉnh này chỉ khác nhau đúng một bit được gọi là đồ thị lập phương, ký hiệu là  $Q_n$ . Như vậy, mỗi đỉnh của  $Q_n$  có bậc là  $n$  và số cạnh của  $Q_n$  là  $n \cdot 2^{n-1}$  (từ công thức  $2|E| = \sum_{v \in V} \deg(v)$ ).

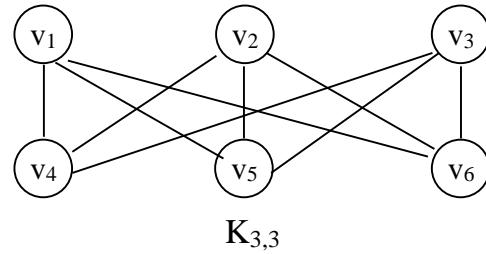
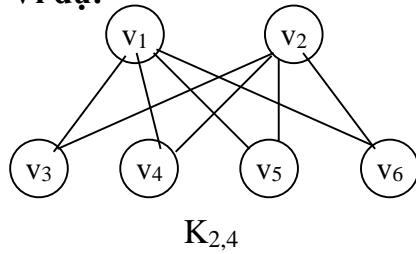
**Ví dụ:**



**Đồ thị phân đôi (đồ thị hai phe):** Đơn đồ thị  $G = (V, E)$  sao cho  $V = V_1 \cup V_2$ ,  $V_1 \cap V_2 = \emptyset$ ,  $V_1 \neq \emptyset$ ,  $V_2 \neq \emptyset$  và mỗi cạnh của  $G$  được nối một đỉnh trong  $V_1$  và một đỉnh trong  $V_2$  được gọi là đồ thị phân đôi.

Nếu đồ thị phân đôi  $G = (V_1 \cup V_2, E)$  sao cho với mọi  $v_1 \in V_1, v_2 \in V_2, (v_1, v_2) \in E$  thì  $G$  được gọi là đồ thị phân đôi đầy đủ. Nếu  $|V_1| = m, |V_2| = n$  thì đồ thị phân đôi đầy đủ  $G$  ký hiệu là  $K_{m,n}$ . Như vậy  $K_{m,n}$  có  $m \cdot n$  cạnh, các đỉnh của  $V_1$  có bậc  $n$  và các đỉnh của  $V_2$  có bậc  $m$ .

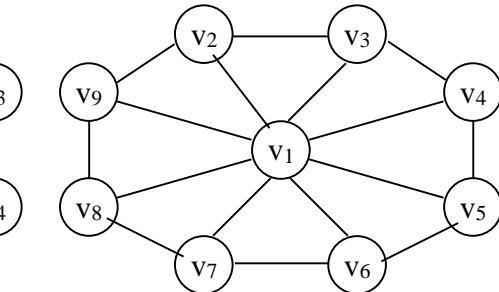
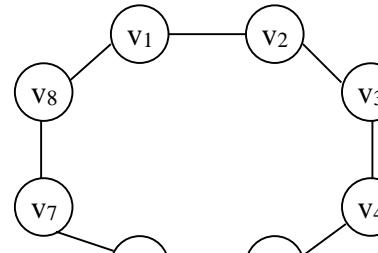
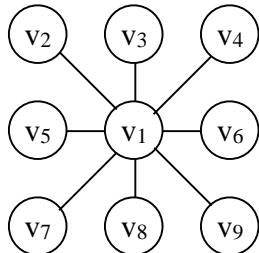
**Ví dụ:**



### 1.5 Những ứng dụng của đồ thị

**Các mạng cục bộ (LAN):** Một số mạng cục bộ dùng cấu trúc hình sao, trong đó tất cả các thiết bị được nối với thiết bị điều khiển trung tâm. Mạng cục bộ kiểu này có thể biểu diễn bằng một đồ thị phân đôi đầy đủ  $K_{1,n}$ . Các thông báo gửi từ thiết bị này tới thiết bị khác đều phải qua thiết bị điều khiển trung tâm.

Mạng cục bộ cũng có thể có cấu trúc vòng tròn, trong đó mỗi thiết bị nối với đúng hai thiết bị khác. Mạng cục bộ kiểu này có thể biểu diễn bằng một đồ thị vòng  $C_n$ . Thông báo gửi từ thiết bị này tới thiết bị khác được truyền đi theo vòng tròn cho tới khi đến nơi nhận.



Cuối cùng, một số mạng cục bộ dùng cấu trúc hỗn hợp của hai cấu trúc trên. Các thông báo được truyền vòng quanh theo vòng tròn hoặc có thể qua thiết bị trung tâm. Sự đứt thưa này có thể làm cho mạng đáng tin cậy hơn. Mạng cục bộ kiểu này có thể biểu diễn bằng một đồ thị bánh xe  $W_n$ .

**Xử lý song song:** Các thuật toán để giải các bài toán được thiết kế để thực hiện một phép toán tại mỗi thời điểm là thuật toán nối tiếp. Tuy nhiên, nhiều bài toán với số lượng tính toán rất lớn như bài toán mô phỏng thời tiết, tạo hình trong y học hay phân tích mật mã không thể giải được trong một khoảng thời gian hợp lý nếu dùng thuật toán nối tiếp ngay cả khi dùng các siêu máy tính. Ngoài ra, do những giới hạn về mặt

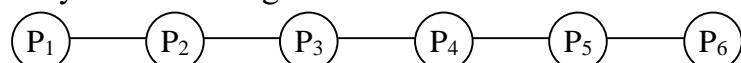
vật lý đối với tốc độ thực hiện các phép toán cơ sở, nên thường gặp các bài toán không thể giải trong khoảng thời gian hợp lý bằng các thao tác nối tiếp. Vì vậy, người ta phải nghĩ đến kiểu xử lý song song.

Khi xử lý song song, người ta dùng các máy tính có nhiều bộ xử lý riêng biệt, mỗi bộ xử lý có bộ nhớ riêng, nhờ đó có thể khắc phục được những hạn chế của các máy nối tiếp. Các thuật toán song song phân chia bài toán chính thành một số bài toán con sao cho có thể giải đồng thời được. Do vậy, bằng các thuật toán song song và nhờ việc sử dụng các máy tính có bộ đa xử lý, người ta hy vọng có thể giải nhanh các bài toán phức tạp. Trong thuật toán song song có một dãy các chỉ thị theo dõi việc thực hiện thuật toán, gửi các bài toán con tới các bộ xử lý khác nhau, chuyển các thông tin vào, thông tin ra tới các bộ xử lý thích hợp.

Khi dùng cách xử lý song song, mỗi bộ xử lý có thể cần các thông tin ra của các bộ xử lý khác. Do đó chúng cần phải được kết nối với nhau. Người ta có thể dùng loại đồ thị thích hợp để biểu diễn mạng kết nối các bộ xử lý trong một máy tính có nhiều bộ xử lý. Kiểu mạng kết nối dùng để thực hiện một thuật toán song song cụ thể phụ thuộc vào những yêu cầu với việc trao đổi dữ liệu giữa các bộ xử lý, phụ thuộc vào tốc độ mong muốn và tất nhiên vào phần cứng hiện có.

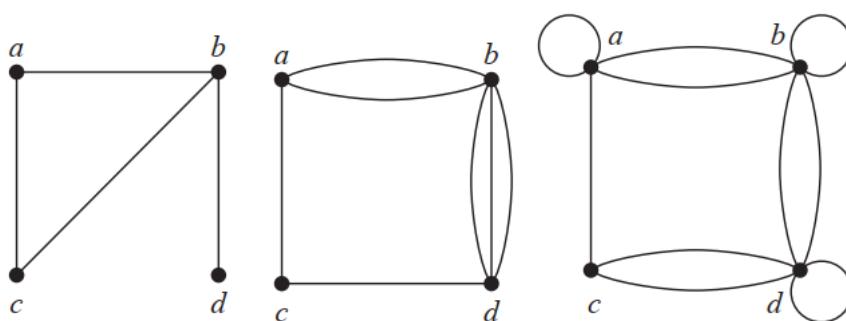
Mạng kết nối các bộ xử lý đơn giản nhất và cũng đắt nhất là có các liên kết hai chiều giữa mỗi cặp bộ xử lý. Các mạng này có thể mô hình bằng đồ thị đầy đủ  $K_n$ , trong đó  $n$  là số bộ xử lý. Tuy nhiên, các mạng liên kết kiểu này có số kết nối quá nhiều mà trong thực tế số kết nối cần phải có giới hạn.

Các bộ xử lý có thể kết nối đơn giản là sắp xếp chúng theo một mảng một chiều. Ưu điểm của mảng một chiều là mỗi bộ xử lý có nhiều nhất 2 đường nối trực tiếp với các bộ xử lý khác. Nhược điểm là nhiều khi cần có rất nhiều các kết nối trung gian để các bộ xử lý trao đổi thông tin với nhau.

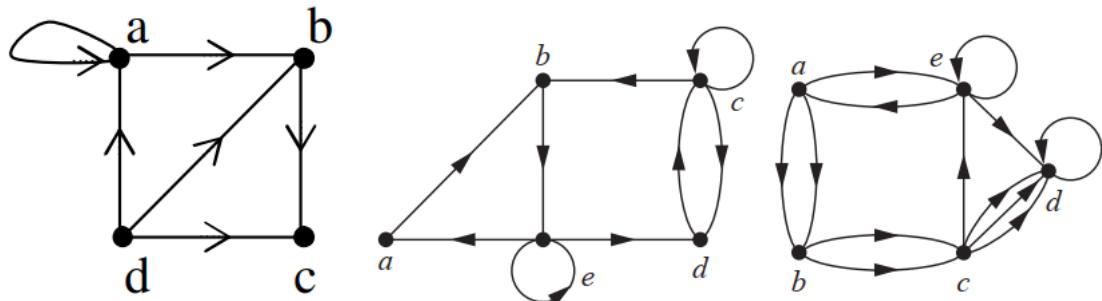


### Bài tập

1. Tìm số đỉnh, số cạnh, bậc của mỗi đỉnh trong các đồ thị vô hướng sau: (chỉ rõ đỉnh cô lập và đỉnh treo, nếu có)



2. Xác định số đỉnh, số cạnh, số bậc vào và số bậc ra của mỗi đỉnh đối với các đồ thị có hướng sau



3. Tìm số đỉnh và số cạnh của các đồ thị  $K_n$ ,  $C_n$ ,  $W_n$ .

## Chương 2 Biểu diễn đồ thị trên máy tính

### 2.1 Ma trận kè

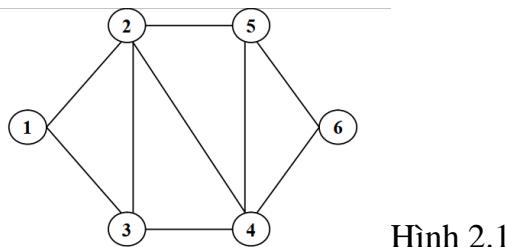
Cấu trúc dữ liệu phổ dụng nhất để biểu diễn đồ thị là biểu diễn đồ thị bằng ma trận. Về lý thuyết, người ta đã chứng minh được mỗi ma trận vuông  $(0,1)$  cấp  $n$  đều đẳng cấu với một đơn đồ thị vô hướng hoặc có hướng. Mục này, chúng ta sẽ xem xét phương pháp biểu diễn các loại đồ thị khác nhau bằng ma trận kè.

#### Ma trận kè của đồ thị vô hướng

Xét đồ thị đơn 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 kè của đồ thị  $G$  là ma trận có các phần tử hoặc bằng 0 hoặc bằng 1 theo qui định như sau:

$$A = \{ a_{ij} : a_{ij} = 1 \text{ nếu } (i, j) \in E, a_{ij} = 0 \text{ nếu } (i, j) \notin E; i, j = 1, 2, \dots, n\}.$$

**Ví dụ:** Biểu diễn đồ thị trong hình dưới đây bằng ma trận kè



$$\begin{vmatrix} 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 \end{vmatrix}$$

#### Tính chất ma trận kè đối với đồ thị vô hướng:

- Tổng các phần tử của ma trận bằng hai lần số cạnh:

$$\sum_{i=1}^n \sum_{j=1}^n a_{ij} = 2m$$

- Tổng các phần tử của hàng  $u$  là bậc của đỉnh  $u$ :

$$\deg(u) = \sum_{j=1}^n a_{uj}$$

Ví dụ, với ma trận kè biểu diễn đồ thị Hình 2.1, tổng các phần tử của hàng 1 là bậc của đỉnh 1, vì vậy  $\deg(1)=2$ ; tổng các phần tử của hàng 2 là bậc của đỉnh 2, vì vậy  $\deg(2)=4$ .

- Tổng các phần tử của cột  $u$  là bậc của đỉnh  $u$ :

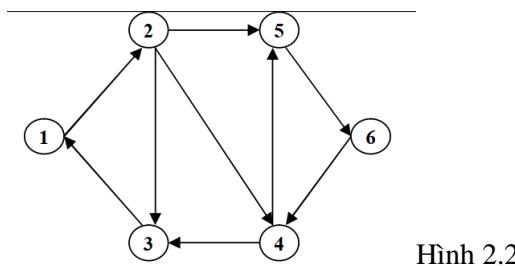
$$\deg(u) = \sum_{j=1}^n a_{ju}$$

Ví dụ, với ma trận kè biểu diễn đồ thị Hình 2.1, tổng các phần tử của cột 1 là bậc của đỉnh 1, vì vậy  $\deg(1)=2$ ; tổng các phần tử của cột 2 là bậc của đỉnh 2, vì vậy  $\deg(2)=4$

### Ma trận kè của đồ thị có hướng

Ma trận kè của đồ thị có hướng cũng được định nghĩa hoàn toàn tương tự, chúng ta chỉ cần lưu ý tới hướng của cạnh. Ma trận kè của đồ thị có hướng là không đối xứng.

**Ví dụ:** Tìm ma trận kè của đồ thị có hướng trong Hình 2.2.



$$\begin{vmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{vmatrix}$$

### Tính chất của ma trận kè của đồ thị có hướng:

- Tổng các phần tử của ma trận bằng số cạnh.

$$\sum_{i=1}^n \sum_{j=1}^n a_{ij} = m$$

- Tổng các phần tử của hàng  $u$  là ~~bán~~ ~~đỉnh~~ bậc ra của đỉnh  $u$ :

$$\deg^+(u) = \sum_{j=1}^n a_{uj}$$

Ví dụ, với ma trận kè biểu diễn đồ thị Hình 2.2, tổng các phần tử của hàng 1 là ~~bán~~ ~~đỉnh~~ bậc ~~và~~ của đỉnh 1, vì vậy  $\deg^+(1)=1$ ; tổng các phần tử của hàng 2 là ~~bán~~ ~~đỉnh~~ bậc ra của đỉnh ~~2~~, vì vậy  $\deg^+(2)=3$ .

- Tổng các phần tử của cột  $u$  là ~~bán~~ ~~đỉnh~~ bậc vào của đỉnh  $u$ :

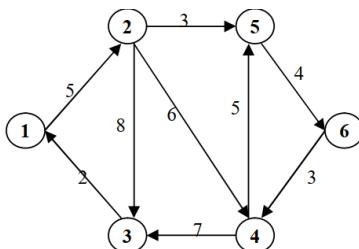
$$\deg^-(u) = \sum_{j=1}^n a_{ju}$$

Ví dụ, với ma trận kè biểu diễn đồ thị Hình 2.2, tổng các phần tử cột 1 là bán đỉnh bậc vào của đỉnh 1, vì vậy  $\deg(1)=1$ ; tổng các phần tử của cột 2 là bán đỉnh bậc vào của đỉnh 2, vì vậy  $\deg(2)=1$ .

## 2.2 Ma trận trọng số

Trong rất nhiều ứng dụng khác nhau của lý thuyết đồ thị, mỗi cạnh  $e = (u, v)$  của nó được gán bởi một số  $c(e) = c(u, v)$  gọi là trọng số của cạnh  $e$ . Đồ thị trong trường hợp như vậy gọi là đồ thị trọng số. Trong trường hợp đó, ma trận kè của đồ thị được thay bởi ma trận trọng số  $c = c[i, j]$ ,  $i, j = 1, 2, \dots, n$ .  $c[i, j] = c(i, j)$  nếu  $(i, j) \in E$ ,  $c[i, j] = \theta$  nếu  $(i, j) \notin E$ . Trong đó,  $\theta$  nhận các giá trị:  $0, \infty, -\infty$  tùy theo từng tình huống cụ thể của thuật toán.

**Ví dụ:** Ma trận kè của đồ thị có trọng số trong Hình 2.3



Hình 2.3

$\infty$	5	$\infty$	$\infty$	$\infty$	$\infty$
$\infty$	$\infty$	8	6	3	$\infty$
2	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$
$\infty$	$\infty$	7	$\infty$	5	$\infty$
$\infty$	$\infty$	$\infty$	$\infty$	$\infty$	4
$\infty$	$\infty$	$\infty$	3	$\infty$	$\infty$

## 2.3 Danh sách cạnh (cung)

Trong trường hợp đồ thị thưa (đồ thị có số cạnh  $m \leq 6n$ ), người ta thường biểu diễn đồ thị dưới dạng danh sách cạnh. Trong phép biểu diễn này, chúng ta sẽ lưu trữ danh sách tất cả các cạnh (cung) của đồ thị vô hướng (có hướng). Mỗi cạnh (cung)  $e(x, y)$  được tương ứng với hai biến  $dau[e]$ ,  $cuoi[e]$ . Như vậy, để lưu trữ đồ thị, ta cần  $2m$  đơn vị bộ nhớ. Nhược điểm lớn nhất của phương pháp này là để nhận biết những cạnh nào kè với cạnh nào chúng ta cần  $m$  phép so sánh trong khi duyệt qua tất cả  $m$  cạnh (cung) của đồ thị. Nếu là đồ thị có trọng số, ta cần thêm  $m$  đơn vị bộ nhớ để lưu trữ trọng số của các cạnh.

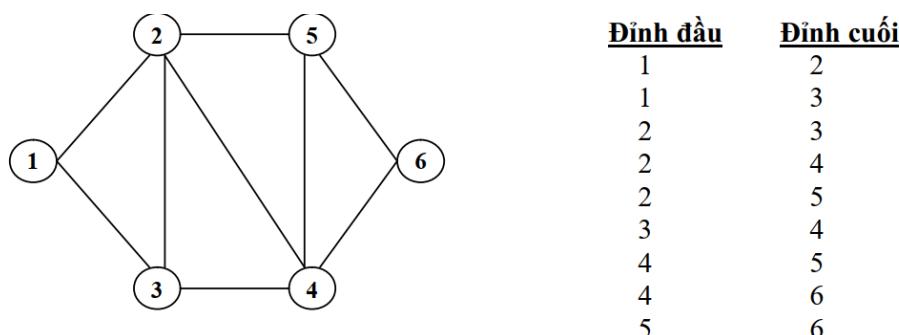
### Biểu diễn đồ thị vô hướng bằng danh sách cạnh

Đối với đồ thị vô hướng, mỗi cạnh là bộ không tính đến thứ tự các đỉnh. Ví dụ cạnh  $(u, v)$  và cạnh  $(v, u)$  được xem là một. Do vậy, trong khi biểu diễn đồ thị vô hướng bằng danh sách cạnh ta chỉ cần liệt kê các cạnh  $(u, v)$  mà không cần liệt kê cạnh  $(v, u)$ . Để tránh nhầm lẫn, ta nên liệt kê các cạnh theo thứ tự tăng dần của đỉnh đầu mỗi cạnh. Trong trường hợp biểu diễn đa đồ thị vô hướng, ta bổ sung thêm một cột là số cạnh

(socanh) nối giữa hai đỉnh của đồ thị. Hình 2.4 dưới đây mô tả chi tiết phương pháp biểu diễn đồ thị vô hướng bằng danh sách cạnh.

### Tính chất danh sách cạnh của đồ thị vô hướng:

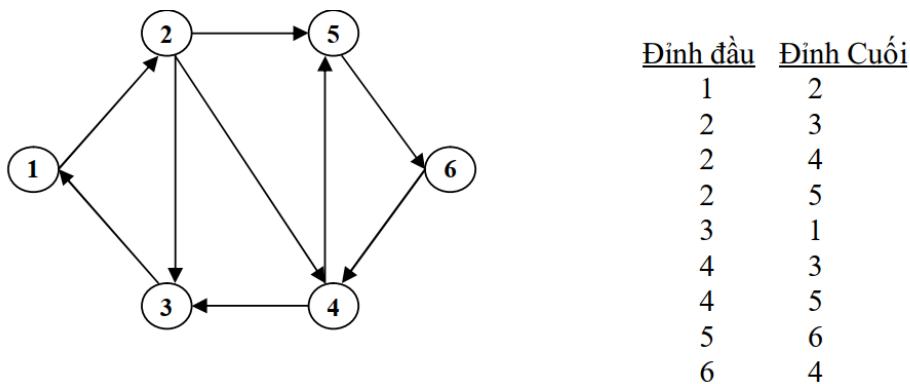
- Đỉnh đầu nhỏ hơn đỉnh cuối mỗi cạnh.
- Số đỉnh có giá trị  $u$  thuộc **cả vế phải và vế trái** của danh sách cạnh là bậc của đỉnh  $u$ . Ví dụ, giá trị  $u=1$  xuất hiện 2 lần từ đó ta suy ra  $\deg(1)=2$ , số 2 xuất hiện 4 lần vì vậy  $\deg(2)=4$



Hình 2.4

### Biểu diễn đồ thị có hướng bằng danh sách cạnh

Trong trường hợp đồ thị có hướng, mỗi cạnh là bộ có tính đến thứ tự các đỉnh. Ví dụ, cạnh  $(u, v)$  khác với cạnh  $(v, u)$ . Do vậy, trong khi biểu diễn đồ thị vô hướng bằng danh sách cạnh ta đặc biệt chú ý đến hướng của các cạnh. Hình 2.5 dưới đây mô tả chi tiết phương pháp biểu diễn đồ thị có hướng bằng danh sách cạnh.



Hình 2.5

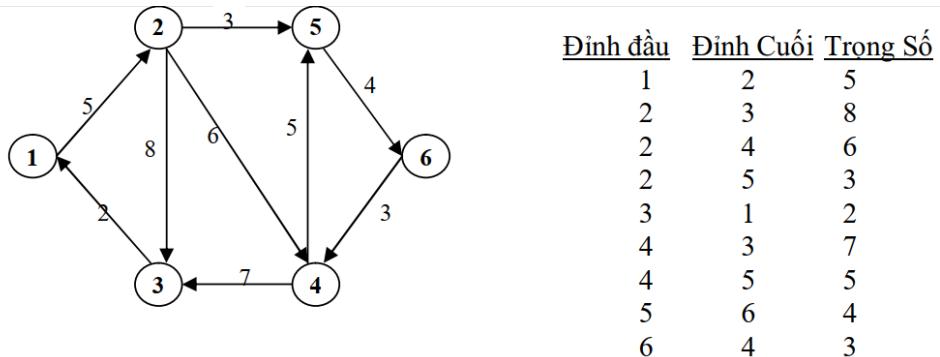
### Tính chất danh sách cạnh của đồ thị vô hướng:

- Đỉnh đầu không nhất thiết phải nhỏ hơn đỉnh cuối mỗi cạnh
- Số đỉnh có giá trị  $u$  thuộc **cả vế phải** các cạnh là  $\deg^+(u)$ . Ví dụ, giá trị  $u=1$  xuất hiện 1 lần ở vế **phải** của tất cả các cạnh nên  $\deg^+(1)=1$ , giá trị  $u=2$  xuất hiện 3 lần ở vế **phải** của tất cả các cạnh nên  $\deg^+(2)=3$ .

- Số đỉnh có giá trị  $u$  thuộc cả về **trái** các cạnh là  $\text{deg}(u)$ . Ví dụ, giá trị  $u=1$  xuất hiện 1 lần ở về **trái** của tất cả các cạnh nên  $\text{deg}(1)=1$ , giá trị  $u=2$  xuất hiện 1 lần ở về **trái** của tất cả các cạnh nên  $\text{deg}(2)=1$ .

### Biểu diễn đồ thị trọng số bằng danh sách cạnh

Trong trường hợp đồ thị có hướng (hoặc vô hướng) có trọng số, ta bổ sung thêm một cột là trọng số của mỗi cạnh. Hình 2.6 dưới đây mô tả chi tiết phương pháp biểu diễn đồ thị trọng số bằng danh sách cạnh.



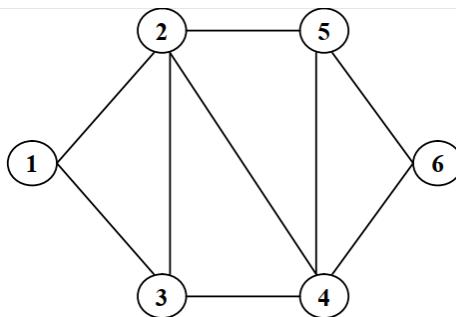
Hình 2.6

### 2.4 Danh sách kè

Trong rất nhiều ứng dụng, cách biểu diễn đồ thị dưới dạng danh sách kè thường được sử dụng. Trong biểu diễn này, với mỗi đỉnh  $u$  của đồ thị chúng ta lưu trữ danh sách các đỉnh kè với nó mà ta ký hiệu là  $Ke(u)$ , nghĩa là

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

Với cách biểu diễn này, mỗi đỉnh  $u$  của đồ thị, ta làm tương ứng với một danh sách tất cả các đỉnh kè với nó và được ký hiệu là  $List(u)$ . Để biểu diễn  $List(u)$ , ta có thể dùng các kiểu dữ liệu kiểu tập hợp, mảng hoặc danh sách liên kết. Hình 2.8 dưới đây đưa ra ví dụ chi tiết về biểu diễn đồ thị bằng danh sách kè.

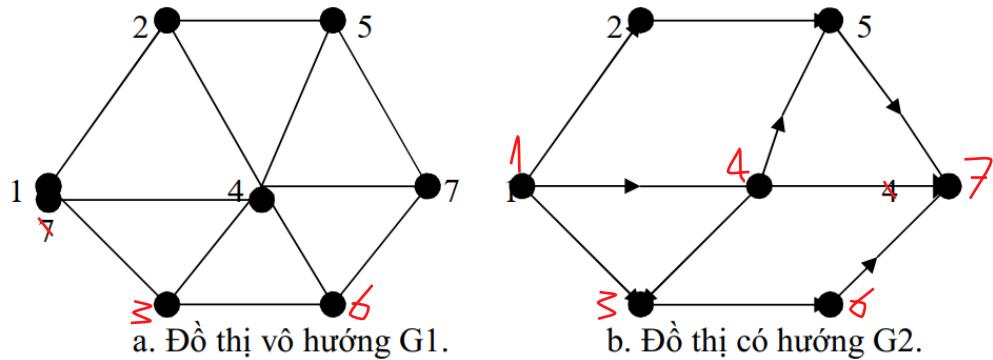


Hình 2.8

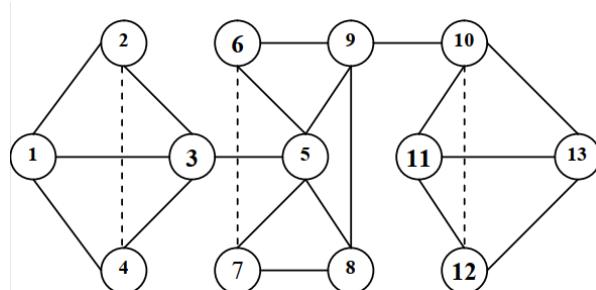
- $K_e(1) = \{2, 3\}$ .  
 $K_e(2) = \{1, 3, 4, 5\}$ .  
 $K_e(3) = \{1, 2, 4\}$ .  
 $K_e(4) = \{2, 3, 5, 6\}$ .  
 $K_e(5) = \{2, 4, 6\}$ .  
 $K_e(6) = \{4, 5\}$ .

### Bài tập

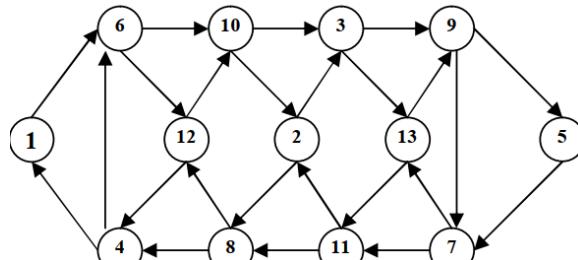
1. Hãy biểu diễn các đồ thị  $G_1, G_2$  dưới đây dưới dạng: ma trận kề, danh sách cạnh, danh sách kề.



2. Hãy biểu diễn đồ thị dưới đây dưới dạng ma trận kề, danh sách cạnh, danh sách kề



3. Hãy biểu diễn đồ thị dưới đây dưới dạng ma trận kề, danh sách cạnh, danh sách kề



4. Vẽ các đồ thị **có** hướng biểu diễn bằng các ma trận liền kề sau

a. 
$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

b. 
$$\begin{bmatrix} 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix}$$

c. 
$$\begin{bmatrix} 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 4 & 0 \\ 1 & 1 & 1 & 0 \end{bmatrix}$$

5. Vẽ các đồ thị có hướng biểu diễn bằng các ma trận liền kề sau

a. 
$$\begin{bmatrix} 1 & 2 & 3 \\ 2 & 0 & 4 \\ 3 & 4 & 0 \end{bmatrix}$$

b. 
$$\begin{bmatrix} 1 & 2 & 0 & 1 \\ 2 & 0 & 3 & 0 \\ 0 & 3 & 1 & 1 \\ 1 & 0 & 1 & 0 \end{bmatrix}$$

## Chương 3      Các thuật toán tìm kiếm trên đồ thị

### 3.1 Tìm kiếm theo chiều sâu trên đồ thị (Depth First Search)

Tư tưởng cơ bản của thuật toán tìm kiếm theo chiều sâu là bắt đầu tại một đỉnh  $v_0$  nào đó, chọn một đỉnh  $u$  bất kỳ kề với  $v_0$  và lấy nó làm đỉnh duyệt tiếp theo. Cách duyệt tiếp theo được thực hiện tương tự như đối với đỉnh  $v_0$  với đỉnh bắt đầu là  $u$ .

Để kiểm tra việc duyệt mỗi đỉnh đúng một lần, chúng ta sử dụng một mảng *chuaxet[]* gồm  $n$  phần tử (tương ứng với  $n$  đỉnh), nếu đỉnh thứ  $u$  đã được duyệt, phần tử tương ứng trong mảng *chuaxet[u]* có giá trị *FALSE*. Ngược lại, nếu đỉnh chưa được duyệt, phần tử tương ứng trong mảng có giá trị *TRUE*.

#### Biểu diễn thuật toán DFS(u)

**Thuật toán DFS(u) có thể được mô tả bằng thủ tục đệ quy như sau:**

*Thuật toán DFS (u): //u là đỉnh bắt đầu duyệt*

*Begin*

*<Thăm đỉnh u>; //Duyệt đỉnh u*

*chuaxet[u] := FALSE; //Xác nhận đỉnh u đã duyệt*

*for each v ∈ ke(u) do //Lấy mỗi đỉnh v ∈ Ke(u).*

*if (chuaxet[v] ) then //Nếu đỉnh v chưa duyệt*

*DFS(v); //Duyệt theo chiều sâu bắt từ đỉnh v*

*EndIf;*

*EndFor;*

*End*

**Thuật toán DFS(u) có thể khử đệ quy bằng cách sử dụng ngăn xếp như dưới đây:**

**Thuật toán DFS(u):**

*Begin*

**Bước 1 (Khởi tạo):**

*stack = Ø; //Khởi tạo stack là Ø Push(stack, u); //Đưa đỉnh u vào ngăn xếp*

*<Thăm đỉnh u>; //Duyệt đỉnh u*

*chuaxet[u] = False; //Xác nhận đỉnh u đã duyệt*

**Bước 2 (Lặp) :**

```

while ( stack ≠ ∅ ) do
    s = Pop(stack); //Loại đỉnh ở đầu ngăn xếp for each t ∈ Ke(s) do //Lấy mỗi
    đỉnh t ∈ Ke(s)

    if ( chuaxet[t] ) then //Nếu t đúng là chưa duyệt <Thăm đỉnh t>; //Duyệt
    đỉnh t

        chuaxet[t] = False; // Xác nhận đỉnh t đã duyệt Push(stack, s); //Đưa s vào
        stack

        Push(stack, t); //Đưa t vào stack

        break; //Chỉ lấy một đỉnh t

    EndIf; EndFor;

    EndWhile;

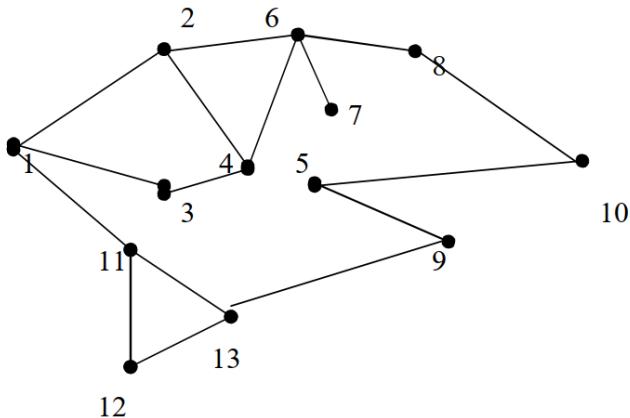
Bước 3 (Trả lại kết quả):

Return(<Tập đỉnh đã duyệt>);

End

```

**Ví dụ:** Kiểm nghiệm thuật toán DFS(1) trên đồ thị gồm 13 đỉnh trong Hình 3.1 dưới đây?



Hình 3.1

Đỉnh bắt đầu duyệt	Các đỉnh đã duyệt: chuaxet[u]=False	Các đỉnh chưa duyệt chuaxet[u]=True
DFS(1)	1	2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13
DFS(2)	1, 2	3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13
DFS(4)	1, 2, 4	3, 5, 6, 7, 8, 9, 10, 11, 12, 13
DFS(3)	1,2,4, 3	5, 6, 7, 8, 9, 10, 11, 12, 13
DFS(6)	1,2,4,3, 6	5, 7, 8, 9, 10, 11, 12, 13
DFS(7)	1,2,4,3, 6,7	5, 8, 9, 10, 11, 12, 13
DFS(8)	1,2,4,3, 6,7,8	5, 9, 10, 11, 12, 13
DFS(10)	1,2,4,3, 6,7,8,10	5, 9, 11, 12, 13
DFS(5)	1,2,4,3, 6,7,8,10,5	9, 11, 12, 13
DFS(9)	1,2,4,3, 6,7,8,10,5,9	11, 12, 13
DFS(13)	1,2,4,3, 6,7,8,10,5,9,13	11, 12
DFS(11)	1,2,4,3, 6,7,8,10,5,9,13,11	12
DFS(12)	1,2,4,3, 6,7,8,10,5,9,13,11,12	∅

Kết quả duyệt: 1, 2, 4, 3, 6, 7, 8, 10, 5, 9, 13, 11, 12

### 3.2. Thuật toán tìm kiếm theo chiều rộng (Breadth First Search)

#### Biểu diễn thuật toán

Để ý rằng, với thuật toán tìm kiếm theo chiều sâu, đỉnh thăm càng muộn sẽ trở thành đỉnh sớm được duyệt xong. Đó là kết quả tất yếu vì các đỉnh thăm được nạp vào stack trong thủ tục đệ qui. Khác với thuật toán tìm kiếm theo chiều sâu, thuật toán tìm kiếm theo chiều rộng thay thế việc sử dụng stack bằng hàng đợi (queue). Trong thủ tục này, đỉnh được nạp vào hàng đợi đầu tiên là u, các đỉnh kề với u là (v<sub>1</sub>, v<sub>2</sub>, ..., v<sub>k</sub>) được nạp vào hàng đợi nếu như nó chưa được xét đến. Quá trình duyệt tiếp theo được bắt đầu từ các đỉnh còn có mặt trong hàng đợi.

Để ghi nhận trạng thái duyệt các đỉnh của đồ thị, ta cũng vẫn sử dụng mảng chuaxet[] gồm n phần tử thiết lập giá trị ban đầu là TRUE. Nếu đỉnh u của đồ thị đã được duyệt, giá trị chuaxet[u] sẽ nhận giá trị FALSE. Thuật toán dừng khi hàng đợi rỗng.

#### Thuật toán BFS(u):

Begin

#### Bước 1(Khởi tạo):

Queue = ∅; Push(Queue,u); chuaxet[u] = False;

#### Bước 2 (Lặp):

while (Queue ≠ ∅) do

s = Pop(Queue); <Thăm đỉnh s>; for each t ∈ Ke(s) do

if ( chuaxet[t] ) then

*Push(Queue, t); chuaxet[t] = False; EndIf;*

*EndFor ;*

*EndWhile ;*

**Bước 3 (Trả lại kết quả) :**

*Return(<Tập đỉnh được duyệt>);*

End.

### 3.3. Tìm đường đi và kiểm tra tính liên thông

#### Tìm đường đi giữa các đỉnh trên đồ thị

##### a) Đặt bài toán

Cho đồ thị  $G = (V, E)$  (vô hướng hoặc có hướng), trong đó  $V$  là tập đỉnh,  $E$  là tập cạnh. Bài toán đặt ra là hãy tìm đường đi từ đỉnh  $s \in V$  đến đỉnh  $t \in V$ ?

##### b) Mô tả thuật toán

Cho đồ thị  $G = (V, E)$ ,  $s, t$  là hai đỉnh thuộc  $V$ . Khi đó, dễ dàng nhận thấy, nếu  $t \in \text{DFS}(s)$  hoặc  $t \in \text{BFS}(s)$  thì ta có thể kết luận có đường đi từ  $s$  đến  $t$  trên đồ thị. Nếu  $t \notin \text{DFS}(s)$  hoặc  $t \notin \text{BFS}(s)$  thì ta có thể kết luận không có đường đi từ  $s$  đến  $t$  trên đồ thị. Vẫn đề còn lại là ta ghi nhận thế nào đường đi từ  $s$  đến  $t$ ?

Để ghi nhận đường đi từ  $s$  đến  $t$  dựa vào hai thuật toán  $\text{DFS}(s)$  hoặc  $\text{BFS}(s)$  ta sử dụng một mảng  $\text{truoc}[]$  gồm  $n$  phần tử ( $n = |V|$ ). Khởi tạo ban đầu  $\text{truoc}[u] = 0$  với mọi  $u = 1, 2, \dots, n$ . Trong quá trình thực hiện hai thuật toán  $\text{DFS}(s)$  và  $\text{BFS}(s)$ , mỗi khi ta đưa đỉnh  $v \in \text{Ke}(s)$  vào ngăn xếp (trong trường hợp ta sử dụng thuật toán  $\text{DFS}$ ) hoặc hàng đợi (trong trường hợp ta sử dụng thuật toán  $\text{BFS}$ ) ta ghi nhận  $\text{truoc}[v] = s$ . Điều này có nghĩa, để đi được đến  $v$  ta phải qua đỉnh  $s$ . Khi hai thuật toán  $\text{DFS}$  và  $\text{BFS}$  duyệt đến đỉnh  $t$  thì  $\text{truoc}[t]$  sẽ nhận giá trị là một đỉnh nào đó thuộc  $V$  hay  $t \in \text{DFS}(s)$  hoặc  $t \in \text{BFS}(s)$ . Trong trường hợp hai thủ tục  $\text{DFS}$  và  $\text{BFS}$  không duyệt được đến đỉnh  $t$ , khi đó  $\text{truoc}[t] = 0$  và ta kết luận không có đường đi từ  $s$  đến  $t$ .

#### Thuật toán DFS tìm đường đi từ $s$ đến $t$

##### Thuật toán DFS(s):

Begin

##### Bước 1 (Khởi tạo):

*stack =  $\emptyset$ ; //Khởi tạo stack là  $\emptyset$  Push(stack, s); //Đưa đỉnh  $s$  vào ngăn xếp chuaxet[s] = False; //Xác nhận đỉnh  $s$  đã duyệt*

##### Bước 2 (Lặp) :

```

while ( stack ≠ ∅ ) do
    u = Pop(stack); //Loại đỉnh ở đầu ngăn xếp for each v ∈ Ke(u) do //Lấy
    mỗi đỉnh u ∈ Ke(v)

    if ( chuaxet[v] ) then //Nếu v đúng là chưa duyệt chuaxet[v] = False; //
    Xác nhận đỉnh v đã duyệt Push(stack, u); //Đưa u vào stack

    Push(stack, v); //Đưa v vào stack truoc[v] = u; //Ghi nhận truoc[v] là u
    break; //Chỉ lấy một đỉnh t

    EndIf;

    EndFor;

    EndWhile;

```

**Bước 3 (Trả lại kết quả):**

Return(<Tập đỉnh đã duyệt>);

End.

**Thuật toán BFS tìm đường đi từ s đến t**

**Thuật toán BFS(s):**

Begin

**Bước 1(Khởi tạo):**

Queue = ∅; Push(Queue,s); chuaxet[s] = False;

**Bước 2 (Lặp):**

while (Queue ≠ ∅) do u = Pop(Queue); for each v ∈ Ke(u) do

if ( chuaxet[v] ) then

Push(Queue, v);chuaxet[v]=False;truoc[v]=u; EndIf ;

EndFor ; EndWhile ;

**Bước 3 (Trả lại kết quả) :**

Return(<Tập đỉnh được duyệt>) ;

End

**Thủ tục ghi nhận đường đi từ s đến t**

**Thuật toán Ghi-Nhan-Duong-Di (s, t)**

{ if( truoc[t] == 0 ) {

<Không có đường đi từ s đến t>; }

else {

<Đưa ra đỉnh t>; //Đưa ra trước đỉnh t u = truoc[t]; //u là đỉnh trước khi đến được t while (u ≠ s) { //Lặp nếu u chưa phải là s

```

<Đưa ra đỉnh u>; //Đưa ra đỉnh u
u = truoc[u]; // Lần ngược lại đỉnh truoc[u]. }

<Đưa ra nốt đỉnh s>; }

}

```

Giả sử ta cần xác định đường đi từ đỉnh 1 đến đỉnh 13 trên đồ thị được biểu diễn dưới dạng ma trận kề.

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

Kiểm nghiệm thuật toán DFS(1).

STT	Trạng thái stack	Truoc[s]=?
1	1	0
2	1, 2	truoc[2] = 1
3	1, 2, 3	truoc[3] = 2
4	1, 2, 3, 4	truoc[4] = 3
5	1, 2, 3, 4, 7	truoc[7] = 4
6	1, 2, 3, 4, 7, 5	truoc[5] = 7
7	1, 2, 3, 4, 7, 5, 6	truoc[6] = 5
8	1, 2, 3, 4, 7, 5, 6, 12	truoc[12] = 6
9	1, 2, 3, 4, 7, 5, 6, 12, 8	truoc[8] = 12
10	1, 2, 3, 4, 7, 5, 6, 12, 10	truoc[10] = 12
11	1, 2, 3, 4, 7, 5, 6, 12, 10, 9	truoc[9] = 10
12	1, 2, 3, 4, 7, 5, 6, 12, 10, 9, 11	truoc[11] = 9
13	1, 2, 3, 4, 7, 5, 6, 12, 10, 9, 11, 13	truoc[13] = 11
14	Ø	

Kết quả đường đi từ đỉnh 1 đến đỉnh 13: 13->11-9-10-12-6-5-7-4-3-2-1

**Kiểm nghiệm thuật toán BFS(1).**

STT	Trạng thái Queue	Truoc[s]=?
1	1	truoc[1]=0
2	2, 3, 4	truoc[2]=1; truoc[3]=1; truoc[4]=1;
3	3, 4, 6	truoc[6]= 2
4	4, 6, 5	truoc[5]=3
5	6, 5, 7	truoc[7]= 4
6	5, 7, 12	truoc[12]=6
7	7, 12, 8	truoc[8]=12
8	12, 8	
9	8, 10	truoc[10]=12;
10	10	
11	9, 11, 13	truoc[9]=10; truoc[11]=10; truoc[13]=10;
12	11, 13	
13	13	
14	Ø	

Kết quả đường đi: 13-10-12-6-2-1.

**Chú ý:** Đường đi từ đỉnh s đến đỉnh t theo thuật toán BFS đi qua ít nhất các cạnh của

đồ thị (có độ dài nhỏ nhất).

**Xác định thành phần liên thông của đồ thị**

**a) Đặt bài toán**

Cho đồ thị đồ thị vô hướng  $G = (V, E)$ , trong đó  $V$  là tập đỉnh,  $E$  là tập cạnh. Bài toán đặt ra là xác định các thành phần liên thông của  $G = (V, E)$ ?

**b) Mô tả thuật toán**

Một đồ thị có thể liên thông hoặc không liên thông. Nếu đồ thị liên thông thì số thành phần liên thông của nó là 1. Điều này tương đương với phép duyệt theo thủ tục  $DFS(u)$  hoặc  $BFS(u)$  được gọi đến đúng một lần. Nói cách khác,  $DFS(u) = V$  và  $BFS(u) = V$ .

Nếu đồ thị không liên thông (số thành phần liên thông lớn hơn 1) chúng ta có thể tách chúng thành những đồ thị con liên thông. Điều này cũng có nghĩa là trong phép duyệt đồ thị, số thành phần liên thông của nó bằng đúng số lần gọi tới thủ tục  $DFS()$  hoặc  $BFS()$ . Để xác định số các thành phần liên thông của đồ thị, chúng ta sử dụng thêm biến  $solt$  để nhận các đỉnh cùng một thành phần liên thông. Khi đó, thuật toán xác định các thành phần liên thông của đồ thị được mô tả như sau.

**Thuật toán Duyet-TPLT:**

Begin

**Bước 1 (Khởi tạo):**

*solt = 0; //Khởi tạo số thành phần liên thông ban đầu là 0*

**Bước 2 (Lặp):**

*for ( u =1; u≤n; u++) do //lặp trên tập đỉnh if (chuaxet[u] ) then*

*solt = solt + 1; //Ghi nhận số thành phần liên thông <Ghi nhận các đỉnh thuộc TPLT>;*

*BFS (u); //DFS(u); //*

*endif; endfor;*

**Bước 3 (Trả lại kết quả):** *Return(solt);*

end.

Ví dụ, ta cần kiểm nghiệm thuật toán trên cho đồ thị được biểu diễn dưới dạng ma trận kề như dưới đây.

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

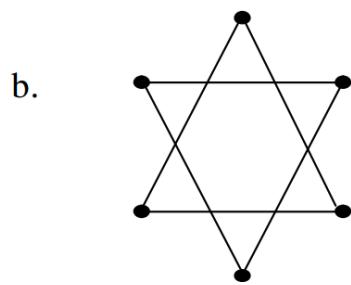
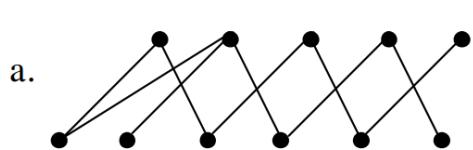
Thực hiện thuật toán DFS và BFS như được mô tả ở trên ta nhận được kết quả sau.

Thành phần liên thông 1:  $\text{BFS}(1) = \{1, 3, 5, 7, 9, 11, 13\}$ .

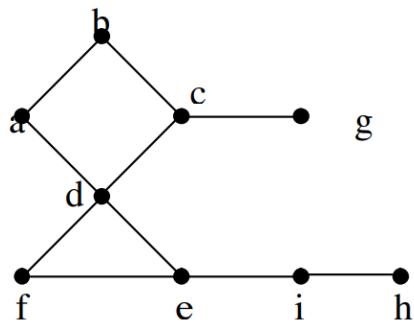
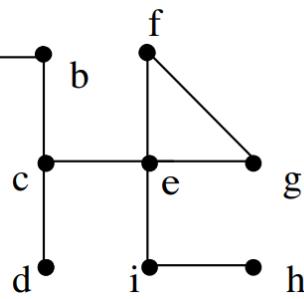
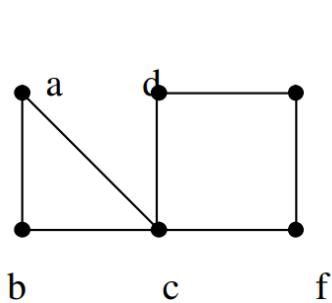
Thành phần liên thông 2:  $\text{BFS}(2) = \{2, 4, 6, 8, 10, 12\}$ .

## Bài tập

1. Các đồ thị sau đây có liên thông không?



2. Tìm tất cả các đỉnh cắt và cầu của các đồ thị sau

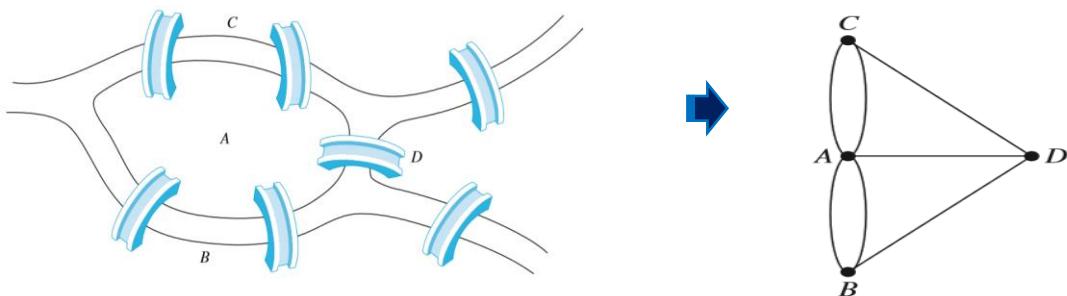


## Chương 4

## Chu trình Euler và chu trình Hamilton

### 4.1 Chu trình Euler

Có thể xem năm 1736 là năm khai sinh lý thuyết đồ thị, với việc công bố lời giải “bài toán về các cầu ở Konigsberg” của nhà toán học lão lẫm Euler (1707-1783). Thành phố Konigsberg thuộc Phổ (nay gọi là Kaliningrad thuộc Nga) được chia thành bốn vùng bằng các nhánh sông Pregel, các vùng này gồm hai vùng bên bờ sông, đảo Kneiphof và một miền nằm giữa hai nhánh của sông Pregel. Vào thế kỷ 18, người ta xây bảy chiếc cầu nối các vùng này với nhau.

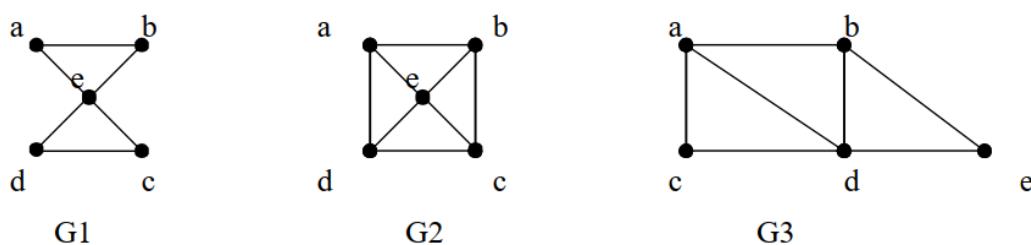


Dân thành phố từng thắc mắc: “Có thể nào đi dạo qua tất cả bảy cầu, mỗi cầu chỉ một lần thôi không?”. Nếu ta coi mỗi khu vực A, B, C, D như một đỉnh và mỗi cầu qua lại hai khu vực là một cạnh nối hai đỉnh thì ta có sơ đồ của Konigsberg là một đa đồ thị G như hình trên.

Bài toán tìm đường đi qua tất cả các cầu, mỗi cầu chỉ qua một lần có thể được phát biểu lại bằng mô hình này như sau: Có tồn tại chu trình đơn trong đa đồ thị G chứa tất cả các cạnh?

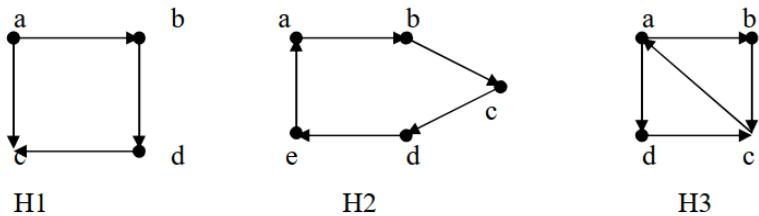
**Định nghĩa 4.1** Chu trình đơn trong đồ thị G đi qua mỗi cạnh của đồ thị đúng một lần được gọi là chu trình Euler. Đường đi đơn trong G đi qua mỗi cạnh của nó đúng một lần được gọi là đường đi Euler. Đồ thị được gọi là đồ thị Euler nếu nó có chu trình Euler. Đồ thị có đường đi Euler được gọi là nửa Euler.

**Ví dụ:** Xét các đồ thị G1, G2, G3 sau



Đồ thị G1 là đồ thị Euler vì nó có chu trình Euler a, e, c, d, e, b, a. Đồ thị G3 không có chu trình Euler nhưng chứa đường đi Euler a, c, d, e, b, d, a, b vì thế G3 là nửa Euler. G2 không có chu trình Euler cũng như đường đi Euler.

**Ví dụ:** Xét các đồ thị có hướng H1, H2, H3 sau



Đồ thị H2 là đồ thị Euler vì nó chứa chu trình Euler a, b, c, d, e, a vì vậy nó là đồ thị Euler. Đồ thị H3 không có chu trình Euler nhưng có đường đi Euler a, b, c, a, d, c nên nó là đồ thị nửa Euler. Đồ thị H1 không chứa đường đi Euler cũng như chu trình Euler.

**Định lý 4.2** Đồ thị (vô hướng) liên thông  $G$  là đồ thị Euler khi và chỉ khi mọi đỉnh của  $G$  đều có bắc chẵn.

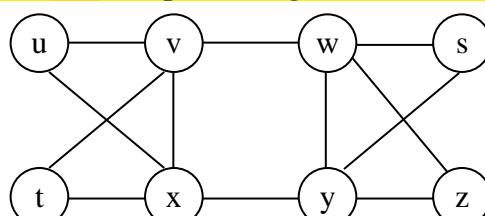
**Bố đề 4.3** Nếu bậc của mỗi đỉnh của đồ thị  $G$  không nhỏ hơn 2 thì  $G$  chứa chu trình đơn.

**Hệ quả 4.4** Đồ thị liên thông  $G$  là nửa Euler (mà không là Euler) khi và chỉ khi có đúng hai đỉnh bậc lẻ trong  $G$ .

**Chú ý:** Ta có thể vạch được một chu trình Euler trong đồ thị liên thông  $G$  có bậc của mọi đỉnh là chẵn theo [thuật toán Fleury](#) sau đây.

Xuất phát từ một định bất kỳ của G và tuân theo hai quy tắc sau:

1. *Mỗi khi đi qua một cành nào thì xoá nó đi; sau đó xoá đỉnh cô lập (nếu có);*
  2. *Không bao giờ đi qua một cành, trừ phi không còn cách đi nào khác.*



Xuất phát từ  $u$ , ta có thể đi theo cạnh  $(u,v)$  hoặc  $(u,x)$ , giả sử là  $(u,v)$  (xoá  $(u,v)$ ). Từ  $v$  có thể đi qua một trong các cạnh  $(v,w)$ ,  $(v,x)$ ,  $(v,t)$ , giả sử  $(v,w)$  (xoá  $(v,w)$ ). Tiếp tục, có thể đi theo một trong các cạnh  $(w,s)$ ,  $(w,y)$ ,  $(w,z)$ , giả sử  $(w,s)$  (xoá  $(w,s)$ ). Đi theo cạnh  $(s,y)$  (xoá  $(s,y)$  và  $s$ ). Vì  $(y,x)$  là cầu nên có thể đi theo một trong hai cạnh  $(y,w)$ ,  $(y,z)$ , giả sử  $(y,w)$  (xoá  $(y,w)$ ). Đi theo  $(w,z)$  (xoá  $(w,z)$  và  $w$ ) và theo  $(z,y)$  (xoá  $(z,y)$  và  $z$ ). Tiếp tục đi theo cạnh  $(y,x)$  (xoá  $(y,x)$  và  $y$ ). Vì  $(x,u)$  là cầu nên đi theo cạnh  $(x,v)$  hoặc  $(x,t)$ , giả sử  $(x,v)$  (xoá  $(x,v)$ ). Tiếp tục đi theo cạnh  $(v,t)$  (xoá  $(v,t)$  và  $v$ ), theo cạnh  $(t,x)$  (xoá cạnh  $(t,x)$  và  $t$ ), cuối cùng đi theo cạnh  $(x,u)$  (xoá  $(x,u)$ ,  $x$  và  $u$ ).

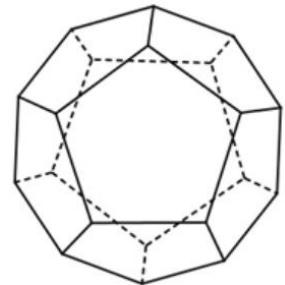
## 4.2 Chu trình Hamilton

Với đồ thị Euler, chúng ta quan tâm tới việc duyệt các cạnh của đồ thị mỗi cạnh đúng một lần, thì trong mục này, chúng ta xét đến một bài toán tương tự nhưng chỉ khác nhau là ta chỉ quan tâm tới các đỉnh của đồ thị, mỗi đỉnh đúng một lần. Sư

thay đổi này tưởng như không đáng kể, nhưng thực tế có nhiều sự khác biệt trong khi giải quyết bài toán.

Năm 1857, nhà toán học người Ailen là Hamilton(1805-1865) đưa ra trò chơi “đi vòng quanh thế giới” như sau.

Cho một hình thập nhị diện đều (đa diện đều có 12 mặt, 20 đỉnh và 30 cạnh), mỗi đỉnh của hình mang tên một thành phố nổi tiếng, mỗi cạnh của hình (nối hai đỉnh) là đường đi lại giữa hai thành phố tương ứng. Xuất phát từ một thành phố, hãy tìm đường đi thăm tất cả các thành phố khác, mỗi thành phố chỉ một lần, rồi trở về chỗ cũ.



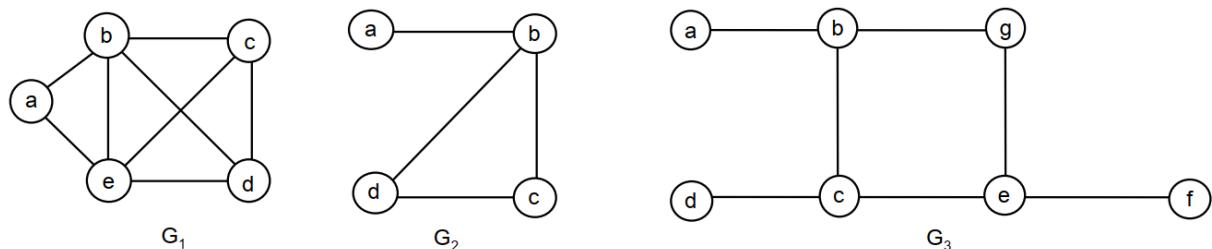
Trước Hamilton, có thể là từ thời Euler, người ta đã biết đến một câu đố hóc búa về “đường đi của con mã trên bàn cờ”. Trên bàn cờ, con mã chỉ có thể đi theo đường chéo của hình chữ nhật  $2 \times 3$  hoặc  $3 \times 2$  ô vuông. Giả sử bàn cờ có  $8 \times 8$  ô vuông. Hãy tìm đường đi của con mã qua được tất cả các ô của bàn cờ, mỗi ô chỉ một lần rồi trở lại ô xuất phát. Bài toán này được nhiều nhà toán học chú ý, đặc biệt là Euler, De Moivre, Vandermonde, ...

Hiện nay đã có nhiều lời giải và phương pháp giải cũng có rất nhiều, trong đó có quy tắc: mỗi lần bố trí con mã ta chọn vị trí mà tại vị trí này số ô chưa dùng tới do nó không chép là ít nhất. Một phương pháp khác dựa trên tính đối xứng của hai nửa bàn cờ. Ta tìm hành trình của con mã trên một nửa bàn cờ, rồi lấy đối xứng cho nửa bàn cờ còn lại, sau đó nối hành trình của hai nửa đã tìm lại với nhau.

Trò chơi và câu đố trên dẫn tới việc khảo sát một lớp đồ thị đặc biệt, đó là đồ thị Hamilton.

**Định nghĩa 4.5** Chu trình (t.ư. đường đi) sơ cấp chứa tất cả các đỉnh của đồ thị (vô hướng hoặc có hướng)  $G$  được gọi là chu trình (t.ư. đường đi) Hamilton. Một đồ thị có chứa một chu trình (t.ư. đường đi) Hamilton được gọi là đồ thị Hamilton (t.ư. nửa Hamilton).

**Ví dụ:** Xét 3 đơn đồ thị  $G_1$ ,  $G_2$ ,  $G_3$  sau



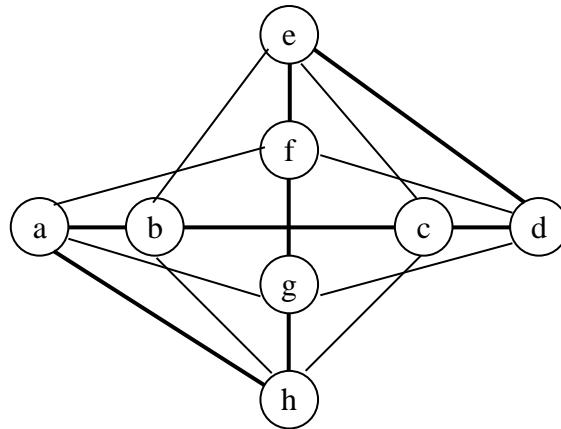
Đồ thị  $G_1$  có chu trình Hamilton ( $a, b, c, d, e, a$ ).  $G_2$  không có chu trình Hamilton vì  $\deg(a) = 1$  nhưng có đường đi Hamilton ( $a, b, c, d$ ).  $G_3$  không có cả chu trình Hamilton lẫn đường đi Hamilton, bởi vì mọi đường đi chứa tất cả các đỉnh đều phải chứa một trong các cạnh  $\{a, b\}$ ,  $\{e, f\}$  và  $\{c, d\}$  nhiều lần.

Người ta chỉ mới tìm được một vài điều kiện đủ để nhận biết một lớp rất nhỏ các đồ thị Hamilton và đồ thị nửa Hamilton. Sau đây là một vài kết quả.

**Định lý 4.6 (Rédei)** Nếu  $G$  là một đồ thị có hướng đầy đủ thì  $G$  là đồ thị nửa Hamilton.

**Định lý 4.7 (Dirac)** Nếu  $G$  là một đơn đồ thị có  $n$  đỉnh và mọi đỉnh của  $G$  đều có bậc không nhỏ hơn  $\frac{n}{2}$  thì  $G$  là một đồ thị Hamilton.

**Ví dụ:**



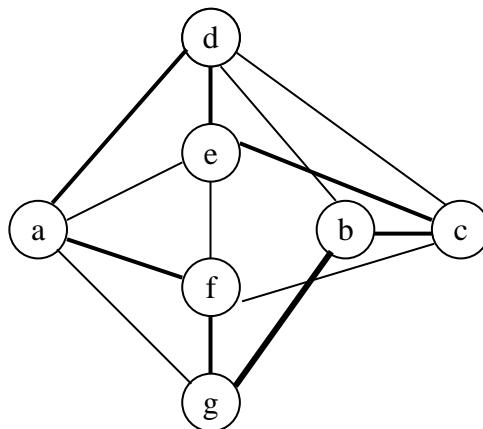
Đồ thị  $G$  này có 8 đỉnh, đỉnh nào cũng có bậc 4, nên theo Định lý Dirac,  $G$  là đồ thị Hamilton.

**Hệ quả 4.8** Nếu  $G$  là đơn đồ thị có  $n$  đỉnh và mọi đỉnh của  $G$  đều có bậc không nhỏ hơn  $\frac{n-1}{2}$  thì  $G$  là đồ thị nửa Hamilton.

**Định lý 4.9(Ore)** Nếu  $G$  là một đơn đồ thị có  $n$  đỉnh và bất kỳ hai đỉnh nào không kề nhau cũng có tổng số bậc không nhỏ hơn  $n$  thì  $G$  là một đồ thị Hamilton.

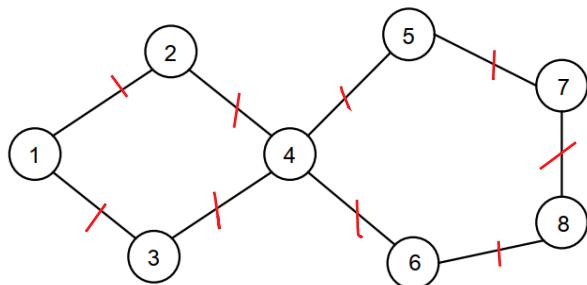
**Ví dụ:**

Đồ thị  $G'$  này có 5 đỉnh bậc 4 và 2 đỉnh bậc 2 kề nhau nên tổng số bậc của hai đỉnh không kề nhau bất kỳ bằng 7 hoặc 8, nên theo Định lý Ore,  $G'$  là đồ thị Hamilton.

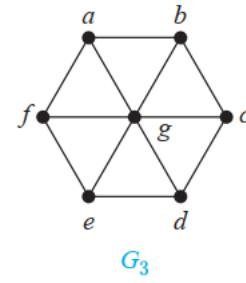
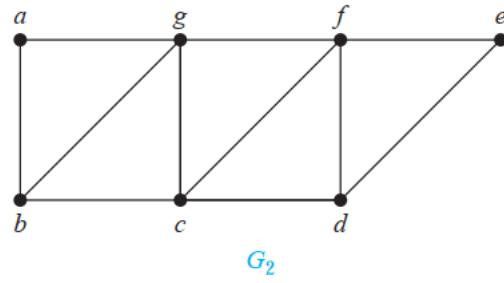
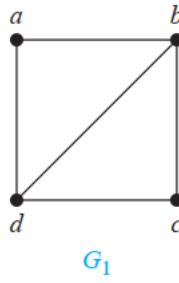


## Bài tập

1. Tìm chu trình Euler của đồ thị sau

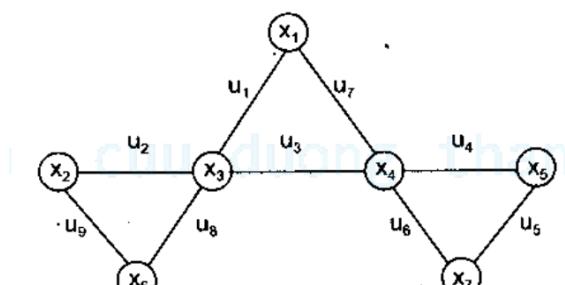
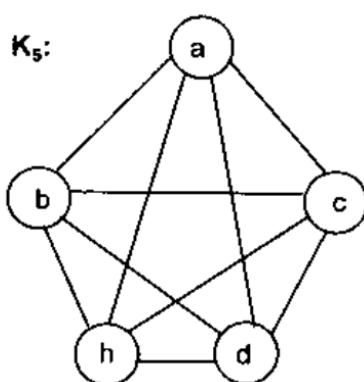
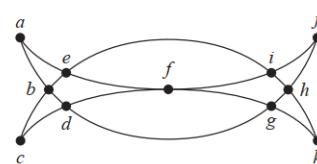
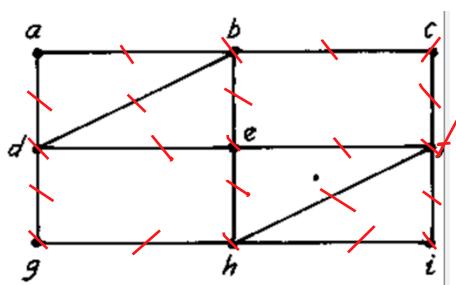
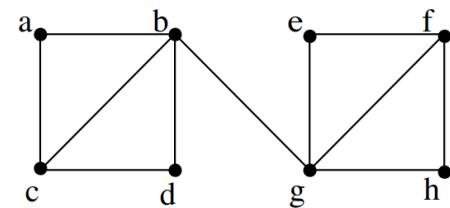
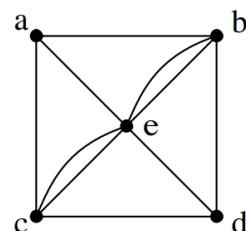
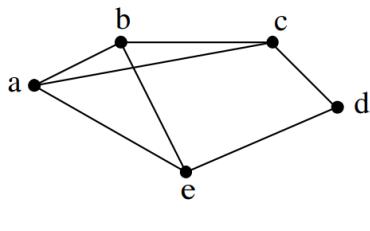


2. Cho các đồ thị sau:



Đồ thị nào trong hình trên có đường đi Euler?

3. Các đồ thị sau có chu trình Euler, đường đi Euler hay không? Nếu có, hãy xây dựng chu trình, đường đi đó.



## Chương 5 Cây và cây khung của đồ thị

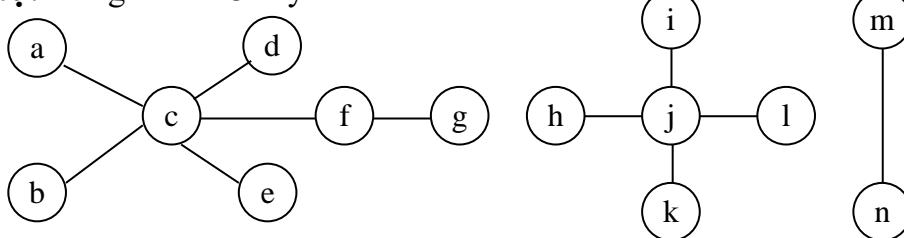
Một đồ thị liên thông và không có chu trình được gọi là cây. Cây đã được dùng từ năm 1857, khi nhà toán học Anh tên là Arthur Cayley dùng cây để xác định những dạng khác nhau của hợp chất hoá học. Từ đó cây đã được dùng để giải nhiều bài toán trong nhiều lĩnh vực khác nhau. Cây rất hay được sử dụng trong tin học. Chẳng hạn, người ta dùng cây để xây dựng các thuật toán rất có hiệu quả để định vị các phần tử trong một danh sách. Cây cũng dùng để xây dựng các mạng máy tính với chi phí rẻ nhất cho các đường điện thoại nối các máy phân tán. Cây cũng được dùng để tạo ra các mã có hiệu quả để lưu trữ và truyền dữ liệu. Dùng cây có thể mô hình các thủ tục mà để thi hành nó cần dùng một dãy các quyết định. Vì vậy cây đặc biệt có giá trị khi nghiên cứu các thuật toán sắp xếp.

### 5.1. Cây và các tính chất của cây

**Định nghĩa 5.1** Cây là một đồ thị vô hướng liên thông, không chứa chu trình và có ít nhất hai đỉnh.

Một đồ thị vô hướng không chứa chu trình và có ít nhất hai đỉnh gọi là một rừng. Trong một rừng, mỗi thành phần liên thông là một cây.

**Ví dụ:** Rừng sau có 3 cây:



**Mệnh đề 5.2** Nếu  $T$  là một cây có  $n$  đỉnh thì  $T$  có ít nhất hai đỉnh treo.

**Định lý 5.3** Cho  $T$  là một đồ thị có  $n \geq 2$  đỉnh. Các điều sau là tương đương:

- 1)  $T$  là một cây.
- 2)  $T$  liên thông và có  $n-1$  cạnh.
- 3)  $T$  không chứa chu trình và có  $n-1$  cạnh.
- 4)  $T$  liên thông và mỗi cạnh là cầu.
- 5) Giữa hai đỉnh phân biệt bất kỳ của  $T$  luôn có duy nhất một đường đi sơ cấp.
- 6)  $T$  không chứa chu trình nhưng khi thêm một cạnh mới thì có được một chu trình duy nhất.

### 5.2. Cây khung của đồ thị

**Định nghĩa 5.4** Trong đồ thị liên thông  $G$ , nếu ta loại bỏ cạnh nằm trên chu trình nào đó thì ta sẽ được đồ thị vẫn là liên thông. Nếu cứ loại bỏ các cạnh ở các chu trình khác cho đến khi nào đồ thị không còn chu trình (vẫn liên thông) thì ta thu

được một cây nối các đỉnh của G. Cây đó gọi là cây khung hay cây bao trùm của đồ thị G.

Tổng quát, nếu G là đồ thị có n đỉnh, m cạnh và k thành phần liên thông thì áp dụng thủ tục vừa mô tả đối với mỗi thành phần liên thông của G, ta thu được đồ thị gọi là rừng khung của G. Số cạnh bị loại bỏ trong thủ tục này bằng  $m - n + k$ , số này ký hiệu là  $v(G)$  và gọi là chu số của đồ thị G.

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

Xét một đồ thị vô hướng liên thông  $G = (V, E)$ ; gọi  $T = (V, F)$  là một cây khung của nó. Các cạnh của cây khung được gọi là các cạnh trong, còn các cạnh khác là các cạnh ngoài.

Nếu thêm một cạnh ngoài  $e \in E \setminus F$  vào cây khung T, thì ta được đúng một chu trình đơn trong T, ký hiệu chu trình này là  $C_e$ . Tập các chu trình:

$$\Omega = \{C_e \mid e \in E \setminus F\}$$

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

#### Các tính chất quan trọng của tập các chu trình cơ sở:

- Tập các chu trình cơ sở là phụ thuộc vào cây khung, hai cây khung khác nhau có thể cho hai tập chu trình cơ sở khác nhau.
- Nếu đồ thị liên thông có n đỉnh và m cạnh, thì trong cây khung có  $n - 1$  cạnh, còn lại  $m - n + 1$  cạnh ngoài. Tương ứng với mỗi cạnh ngoài có một chu trình cơ sở, vậy số chu trình cơ sở của đồ thị liên thông là  $m - n + 1$ .
- Tập các chu trình cơ sở là tập nhiều nhất các chu trình thỏa mãn: Mỗi chu trình có đúng một cạnh riêng, cạnh đó không nằm trong bất cứ một chu trình nào khác. Bởi nếu có một tập gồm t chu trình thỏa mãn điều đó thì việc loại bỏ cạnh riêng của một chu trình sẽ không làm mất tính liên thông của đồ thị, đồng thời không ảnh hưởng tới sự tồn tại của các chu trình khác. Như vậy nếu loại bỏ tất cả các cạnh riêng thì đồ thị vẫn liên thông và còn  $m - t$  cạnh. Đồ thị liên thông thì không thể có ít hơn  $n - 1$  cạnh nên ta có  $m - t \geq n - 1$  hay  $t \leq m - n + 1$ .
- Mọi cạnh trong một chu trình đơn bất kỳ đều phải thuộc một chu trình cơ sở. Bởi nếu có một cạnh  $(u, v)$  không thuộc một chu trình cơ sở nào, thì khi ta bỏ cạnh đó đi đồ thị vẫn liên thông và không ảnh hưởng tới sự tồn tại của các chu trình cơ sở. Lại bỏ tiếp những cạnh ngoài của các chu trình cơ sở thì đồ thị vẫn liên thông và còn lại  $m - (m - n + 1) - 1 = n - 2$  cạnh. Điều này vô lý.
- Đối với đồ thị  $G = (V, E)$  có n đỉnh và m cạnh, có k thành phần liên thông, ta có thể xét các thành phần liên thông và xét rừng các cây khung của các thành phần đó. Khi đó có thể mở rộng khái niệm tập các chu trình cơ sở cho đồ thị vô hướng tổng quát: Mỗi khi thêm một cạnh không nằm trong các cây khung vào rừng, ta được đúng một chu trình đơn, tập các chu trình đơn tạo thành bằng cách ghép các cạnh ngoài như vậy gọi là tập các chu trình cơ sở của đồ thị G. Số các chu trình cơ sở là  $m - n + k$ .

### 5.4 Bài toán tìm cây khung nhỏ nhất:

Bài toán tìm 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

phần này ta sẽ có hai thuật toán cơ bản để giải bài toán này. Trước hết, nội dung của bài toán được phát biểu như sau..

Cho  $G = (V, E)$  là đồ thị vô hướng liên thông có trọng số, mỗi cạnh  $e \in E$  có trọng số  $m(e) \geq 0$ . Giả sử  $T = (V_T, E_T)$  là cây khung của đồ thị  $G$  ( $V_T = V$ ). Ta gọi độ dài  $m(T)$  của cây khung  $T$  là tổng trọng số của các cạnh của nó:

$$m(T) = \sum_{e \in E_T} m(e).$$

Bài toán đặt ra là trong số tất cả các cây khung của đồ thị  $G$ , hãy tìm cây khung có 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 tìm cây khung nhỏ nhất.

Để minh họa cho những ứng dụng của bài toán cây khung nhỏ nhất, dưới đây là hai mô hình thực tế tiêu biểu cho 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 cứ một thành phố nào đến bất kỳ một trong số các thành phố còn lại. Một khác, trên quan điểm kinh tế đòi hỏi là chi phí về xây dựng hệ thống đường phải là nhỏ nhất. Rõ ràng là 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 hệ thống đường sắt nối hai 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  $m(i,j)$  (thông thường chi phí này phụ thuộc vào độ dài cáp cần sử dụng). Hãy tìm cách nối mạng sao cho tổng chi phí là nhỏ nhất. Bài toán này cũng dẫn về bài toán tìm cây khung nhỏ nhất.

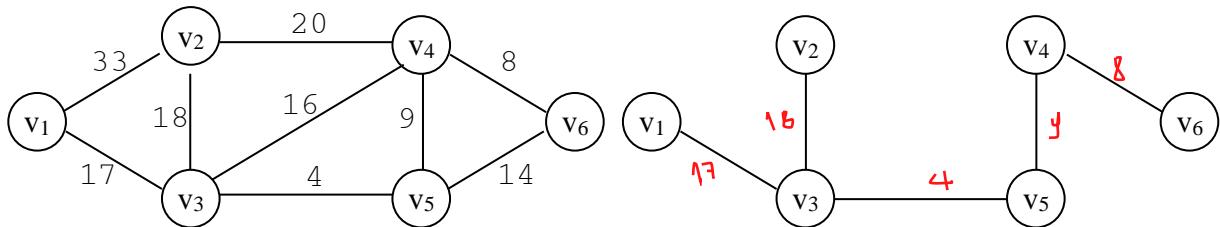
Bài toán tìm cây khung nhỏ nhất đã có những thuật toán rất hiệu quả để giải chúng. Ta sẽ xét hai trong số những thuật toán như vậy: thuật toán Kruskal và thuật toán Prim.

**Thuật toán Kruskal:** Thuật toán sẽ xây dựng tập cạnh  $E_T$  của cây khung nhỏ nhất  $T = (V_T, E_T)$  theo từng bước. Trước hết sắp xếp các cạnh của đồ thị  $G$  theo thứ tự không giảm của trọng số. Bắt đầu từ  $E_T = \emptyset$ , 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  $E_T$  không tạo thành chu trình trong tập này. **Thuật toán sẽ kết thúc khi ta thu được tập  $E_T$  gồm  $n - 1$  cạnh.** Cụ thể có thể mô tả như sau:

1. Bắt đầu từ đồ thị rỗng  $T$  có  $n$  đỉnh.
2. Sắp xếp các cạnh của  $G$  theo thứ tự **không giảm** của trọng số.
3. Bắt đầu từ cạnh đầu tiên của dãy này, ta cứ **thêm dần các cạnh của dãy** đã được xếp vào  $T$  theo nguyên tắc **cạnh thêm vào không được tạo thành chu trình** trong  $T$ .

4. Lặp lại Bước 3 cho đến khi nào số cạnh trong  $T$  bằng  $n-1$ , ta thu được cây khung nhỏ nhất cần tìm.

**Ví dụ:** Tìm cây khung nhỏ nhất của đồ thị cho trong hình dưới đây.



Bắt đầu từ đồ thị rỗng  $T$  có 6 đỉnh.

Sắp xếp các cạnh của đồ thị theo thứ tự không giảm của trọng số:

$\{(v_3, v_5), (v_4, v_6), (v_4, v_5), (v_5, v_6), (v_3, v_4), (v_1, v_3), (v_2, v_3), (v_2, v_4), (v_1, v_2)\}$ .

Thêm vào đồ thị  $T$  cạnh  $(v_3, v_5)$ .

Do số cạnh của  $T$  là  $1 < 6 - 1$  nên tiếp tục thêm cạnh  $(v_4, v_6)$  vào  $T$ . Bây giờ số cạnh của  $T$  đã là 2 vẫn còn nhỏ hơn 6, ta tiếp tục thêm cạnh tiếp theo trong dãy đã sắp xếp vào  $T$ . Sau khi thêm cạnh  $(v_4, v_5)$  vào  $T$ , nếu thêm cạnh  $(v_5, v_6)$  thì nó sẽ tạo thành với 2 cạnh  $(v_4, v_5)$ ,  $(v_4, v_6)$  đã có trong  $T$  một chu trình. Tình huống tương tự cũng xảy ra đối với cạnh  $(v_3, v_4)$  là cạnh tiếp theo trong dãy. Tiếp theo ta bổ sung cạnh  $(v_1, v_3)$ ,  $(v_2, v_3)$  vào  $T$  và thu được tập  $E_T$  gồm 5 cạnh:

$\{(v_3, v_5), (v_4, v_6), (v_4, v_5), (v_1, v_3), (v_2, v_3)\}$ .

**Thuật toán Prim:** Thuật toán Kruskal làm việc kém hiệu quả đối với những đồ thị dày (đồ thị có 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.

1.  $V_T := \{v^*\}$ , trong đó  $v^*$  là đỉnh tuỳ ý của đồ thị  $G$ .

$E_T := \emptyset$ .

2. Với mỗi đỉnh  $v_j \notin V_T$ , tìm đỉnh  $w_j \in V_T$  sao cho

$$m(w_j, v_j) = \min_{x_i \in V_T} m(x_i, v_j) =: \beta_j$$

và gán cho đỉnh  $v_j$  nhãn  $[w_j, \beta_j]$ . Nếu không tìm được  $w_j$  như vậy (tức là khi  $v_j$  không kề với bất cứ đỉnh nào trong  $V_T$ ) thì gán cho  $v_j$  nhãn  $[0, \infty]$ .

3. Chọn đỉnh  $v_{j^*}$  sao cho

$$\beta_{j^*} = \min_{v_j \notin V_T} \beta_j$$

$$V_T := V_T \cup \{v_{j^*}\},$$

$$E_T := E_T \cup \{(w_{j^*}, v_{j^*})\}.$$

Nếu  $|V_T| = n$  thì thuật toán dừng và  $(V_T, E_T)$  là cây khung nhỏ nhất.

Nếu  $|V_T| < n$  thì chuyển sang Bước 4.

4. Đối với tất cả các đỉnh  $v_j \notin V_T$  mà kề với  $v_{j^*}$ , ta thay đổi nhãn của chúng như sau:

Nếu  $\beta_j > m(v_j^*, v_j)$  thì đặt  $\beta_j := m(v_j^*, v_j)$  và nhãn của  $v_j$  là  $[v_j^*, \beta_j]$ . Ngược lại, ta giữ nguyên nhãn của  $v_j$ . Sau đó quay lại Bước 3.

**Ví dụ:** Tìm cây khung nhỏ nhất bằng thuật toán Prim của đồ thị gồm các đỉnh A, B, C, D, E, F, H, I được cho bởi ma trận trọng số sau.

$$\begin{array}{cccccccccc}
 & \mathbf{A} & \mathbf{B} & \mathbf{C} & \mathbf{D} & \mathbf{E} & \mathbf{F} & \mathbf{H} & \mathbf{I} \\
 \mathbf{A} & \infty & 15 & 16 & 19 & 23 & 20 & 32 & 18 \\
 \mathbf{B} & 15 & \infty & 33 & 13 & 34 & 19 & 20 & 12 \\
 \mathbf{C} & 16 & 33 & \infty & 13 & 29 & 21 & 20 & 19 \\
 \mathbf{D} & 19 & 13 & 13 & \infty & 22 & 30 & 21 & 11 \\
 \mathbf{E} & 23 & 34 & 29 & 22 & \infty & 34 & 23 & 21 \\
 \mathbf{F} & 20 & 19 & 21 & 30 & 34 & \infty & 17 & 18 \\
 \mathbf{H} & 32 & 20 & 20 & 21 & 23 & 17 & \infty & 14 \\
 \mathbf{I} & 18 & 12 & 19 & 11 & 21 & 18 & 14 & \infty
 \end{array} .$$

Yêu cầu viết các kết quả trung gian trong từng bước lặp, kết quả cuối cùng cần đưa ra tập cạnh và độ dài của cây khung nhỏ nhất.

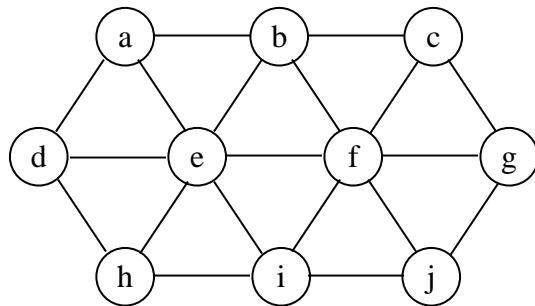
V.lặp	<b>A</b>	<b>B</b>	<b>C</b>	<b>D</b>	<b>E</b>	<b>F</b>	<b>H</b>	<b>I</b>	<b>V<sub>T</sub></b>	<b>E<sub>T</sub></b>
K.tạo	–	[A,15]	[A,16]	[A,19]	[A,23]	[A,20]	[A,32]	[A,18]	A	∅
1	–	–	[A,16]	[B,13]	[A,23]	[B,19]	[B,20]	[B,12]	A, B	(A,B)
2	–	–	[A,16]	[I,11]	[I,21]	[I,18]	[I,14]	–	A, B, I	(A,B), (B,I)
3	–	–	[D,13]	–	[I,21]	[I,18]	[I,14]	–	A, B, I, D	(A,B), (B,I), (I,D)
4	–	–	–	–	[I,21]	[I,18]	[I,14]	–	A, B, I, D, C	(A,B), (B,I), (I,D), (D,C)
5	–	–	–	–	[I,21]	[H,17]	–	–	A, B, I, D, C, H	(A,B), (B,I), (I,D), (D,C), (I,H)
6	–	–	–	–	[I,21]	–	–	–	A, B, I, D, C, H, F	(A,B), (B,I), (I,D), (D,C), (I,H), (H,F)
7	–	–	–	–	–	–	–	–	A, B, I, D, C, H, F, E	(A,B), (B,I), (I,D), (D,C), (I,H), (H,F), (I,E)

Vậy độ dài cây khung nhỏ nhất là:

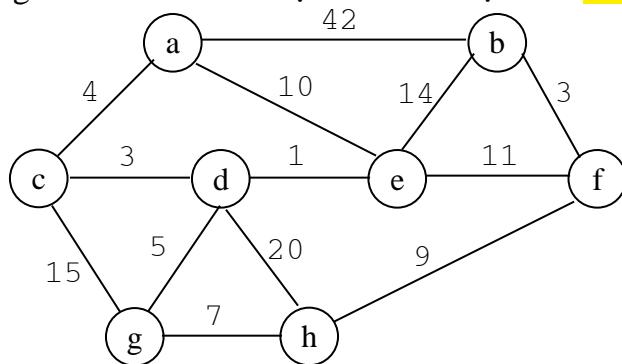
$$15 + 12 + 11 + 13 + 14 + 17 + 21 = 103.$$

## Bài tập

1. Hãy tìm cây khung của đồ thị sau bằng cách xoá đi các cạnh trong các chu trình đơn.



2. Tìm cây khung nhỏ nhất của đồ thị sau theo thuật toán Kruskal và Prim.



## Chương 6      Bài toán tìm đường đi ngắn nhất

### 6.1. Các khái niệm mở đầu

Trong đời sống, chúng ta thường gặp những tình huống như sau: để đi từ địa điểm A đến địa điểm B trong thành phố, có nhiều đường đi, nhiều cách đi; có lúc ta chọn đường đi ngắn nhất (theo nghĩa cự ly), có lúc lại cần chọn đường đi nhanh nhất (theo nghĩa thời gian) và có lúc phải cân nhắc để chọn đường đi rẻ tiền nhất (theo nghĩa chi phí), v.v...

Có thể coi sơ đồ của đường đi từ A đến B trong thành phố là một đồ thị, với đỉnh là các giao lộ (A và B coi như giao lộ), cạnh là đoạn đường nối hai giao lộ. Trên mỗi cạnh của đồ thị này, ta gán một số dương, ứng với chiều dài của đoạn đường, thời gian đi đoạn đường hoặc cước phí vận chuyển trên đoạn đường đó, ...

Đồ thị có trọng số là đồ thị  $G = (V, E)$  mà mỗi cạnh (hoặc cung)  $e \in E$  được gán bởi một số thực  $m(e)$ , gọi là trọng số của cạnh (hoặc cung)  $e$ .

Trong phần này, trọng số của mỗi cạnh được xem là một số dương và còn gọi là chiều dài của cạnh đó. Mỗi đường đi từ đỉnh  $u$  đến đỉnh  $v$ , có chiều dài là  $m(u,v)$ , bằng tổng chiều dài các cạnh mà nó đi qua. Khoảng cách  $d(u,v)$  giữa hai đỉnh  $u$  và  $v$  là chiều dài đường đi ngắn nhất (theo nghĩa  $m(u,v)$  nhỏ nhất) trong các đường đi từ  $u$  đến  $v$ .

Có thể xem một đồ thị  $G$  bất kỳ là một đồ thị có trọng số mà mọi cạnh đều có chiều dài 1. Khi đó, khoảng cách  $d(u,v)$  giữa hai đỉnh  $u$  và  $v$  là chiều dài của đường đi từ  $u$  đến  $v$  ngắn nhất, tức là đường đi qua ít cạnh nhất.

### 6.2. Đường đi ngắn nhất xuất phát từ một đỉnh

Cho đơn đồ thị liên thông, có trọng số  $G = (V, E)$ . Tìm khoảng cách  $d(u_0, v)$  từ một đỉnh  $u_0$  cho trước đến một đỉnh  $v$  bất kỳ của  $G$  và tìm đường đi ngắn nhất từ  $u_0$  đến  $v$ .

Có một số thuật toán tìm đường đi ngắn nhất; ở đây, ta có thuật toán do E. Dijkstra, nhà toán học người Hà Lan, đề xuất năm 1959. Trong phiên bản mà ta sẽ trình bày, người ta giả sử đồ thị là vô hướng, các **trọng số là dương**. Chỉ cần thay đổi đôi chút là có thể giải được bài toán tìm đường đi ngắn nhất trong đồ thị có hướng.

Phương pháp của **thuật toán Dijkstra** là: xác định tuần tự đỉnh có khoảng cách đến  $u_0$  từ nhỏ đến lớn.

Trước tiên, đỉnh có khoảng cách đến  $a$  nhỏ nhất chính là  $a$ , với  $d(u_0, u_0) = 0$ . Trong các đỉnh  $v \neq u_0$ , tìm đỉnh có khoảng cách  $k_1$  đến  $u_0$  là nhỏ nhất. Đỉnh này phải là một trong các đỉnh kề với  $u_0$ . Giả sử đó là  $u_1$ . Ta có:

$$d(u_0, u_1) = k_1.$$

Trong các đỉnh  $v \neq u_0$  và  $v \neq u_1$ , tìm đỉnh có khoảng cách  $k_2$  đến  $u_0$  là nhỏ nhất. Đỉnh này phải là một trong các đỉnh kề với  $u_0$  hoặc với  $u_1$ . Giả sử đó là  $u_2$ . Ta có:

$$d(u_0, u_2) = k_2.$$

Tiếp tục như trên, cho đến bao giờ tìm được khoảng cách từ  $u_0$  đến mọi đỉnh  $v$  của  $G$ . Nếu  $V = \{u_0, u_1, \dots, u_n\}$  thì:

$$0 = d(u_0, u_0) < d(u_0, u_1) < d(u_0, u_2) < \dots < d(u_0, u_n).$$

### 6.3 Thuật toán Dijkstra

**procedure Dijkstra** ( $G = (V, E)$  là đơn đồ thị liên thông, có trọng số với trọng số dương)

{ $G$  có các đỉnh  $a = u_0, u_1, \dots, u_n = z$  và trọng số  $m(u_i, u_j)$ , với  $m(u_i, u_j) = \infty$  nếu  $(u_i, u_j)$  không là một cạnh trong  $G$ }

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

$L(u_i) := \infty$

$L(a) := 0$

$S := V \setminus \{a\}$

$u := a$

**while**  $S \neq \emptyset$

**begin**

**for** tất cả các đỉnh  $v$  thuộc  $S$

**if**  $L(u) + m(u, v) < L(v)$  **then**  $L(v) := L(u) + m(u, v)$

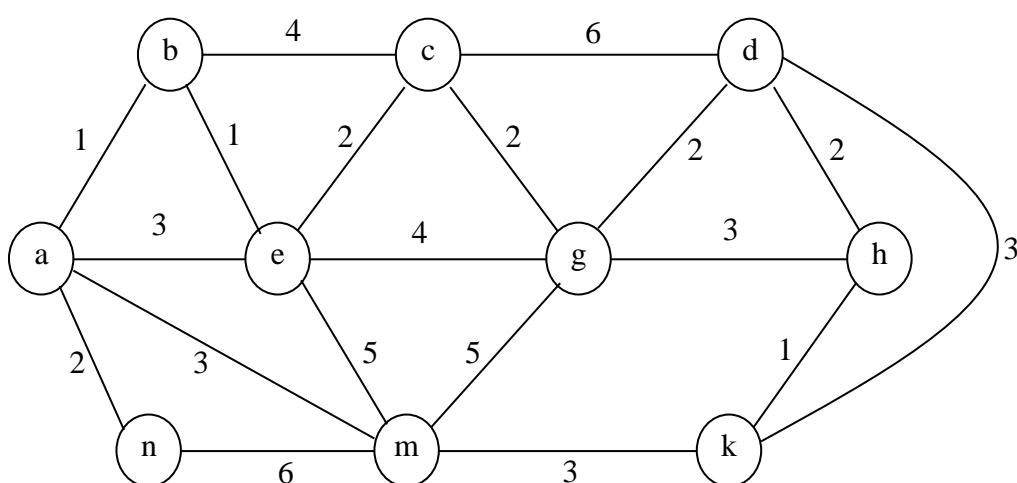
$u :=$  đỉnh thuộc  $S$  có nhãn  $L(u)$  nhỏ nhất

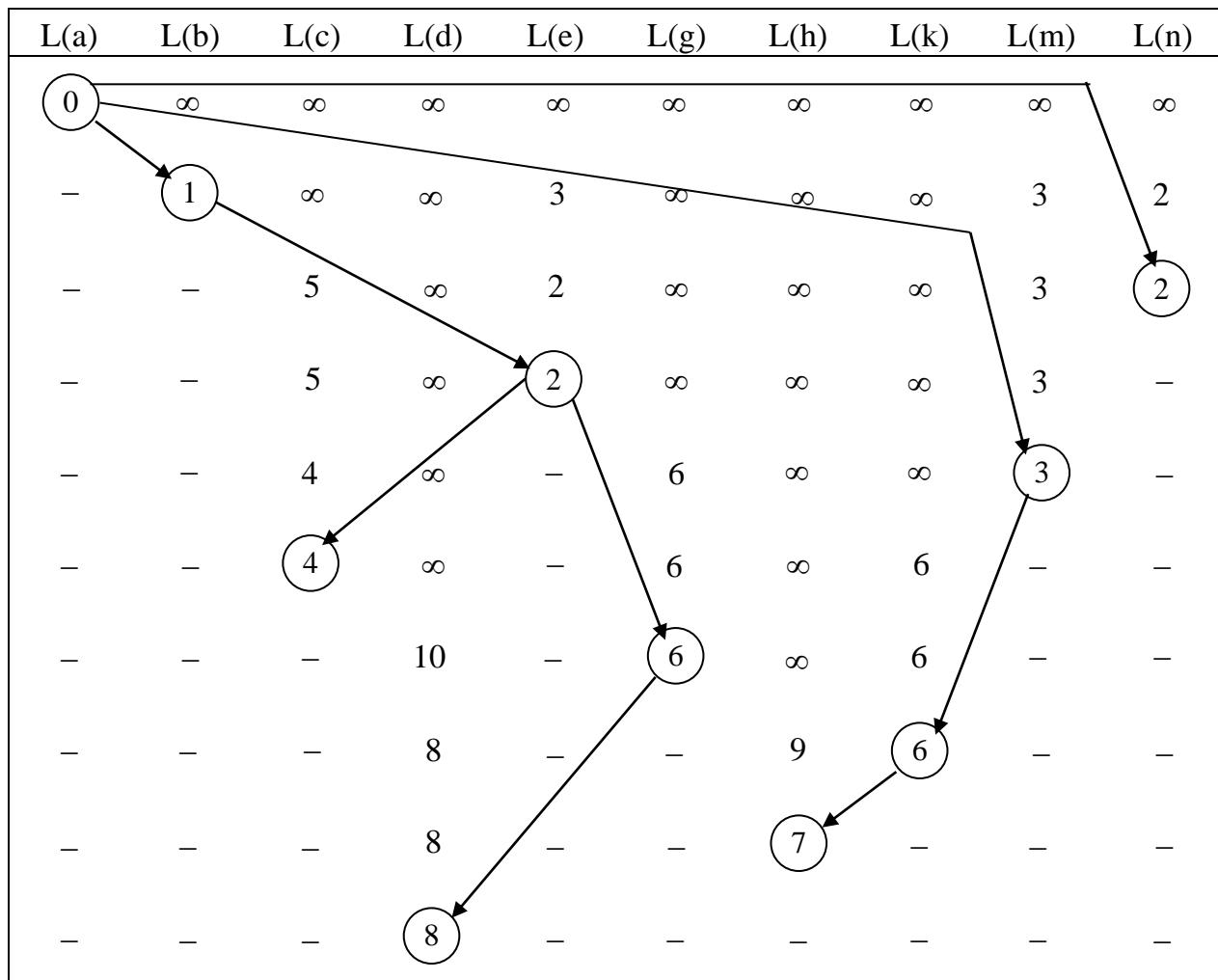
                { $L(u)$ : độ dài đường đi ngắn nhất từ  $a$  đến  $u$ }

$S := S \setminus \{u\}$

**end**

**Ví dụ:** Tìm khoảng cách  $d(a, v)$  từ  $a$  đến mọi đỉnh  $v$  và tìm đường đi ngắn nhất từ  $a$  đến  $v$  cho trong đồ thị  $G$  sau.





**Định lý 6.1** Thuật toán Dijkstra tìm được đường đi ngắn nhất từ một đỉnh cho trước đến một đỉnh tùy ý trong đơn đồ thị vô hướng liên thông có trọng số.

#### 6.4. Đường đi trong đồ thị không có chu trình

Ta có định lý sau: Giả sử  $G = (V, E)$  là đồ thị không có chu trình (có hướng - tất nhiên). Khi đó các đỉnh của nó có thể đánh số sao cho mỗi cung của nó chỉ nối từ đỉnh có chỉ số nhỏ hơn đến đỉnh có chỉ số lớn hơn. Thuật toán đánh số lại các đỉnh của đồ thị có thể mô tả như sau.

Trước hết ta chọn một đỉnh không có cung đi vào và đánh chỉ số 1 cho đỉnh đó. Sau đó xoá bỏ đỉnh này cùng với tất cả những cung từ u đi ra, ta được một đồ thị mới cũng không có chu trình, và lại đánh chỉ số 2 cho một đỉnh v nào đó không có cung đi vào, rồi lại xoá đỉnh v cùng với các cung từ v đi ra ... Thuật toán sẽ kết thúc nếu như hoặc ta đã đánh chỉ số được hết các đỉnh, hoặc tất cả các đỉnh còn lại đều có cung đi vào. Trong trường hợp tất cả các đỉnh còn lại đều có cung đi vào thì sẽ tồn tại chu trình trong đồ thị và khẳng định thuật toán tìm đường đi ngắn nhất trong mục này không áp dụng được. (Thuật toán đánh số này có thể cải tiến bằng cách dùng một hàng đợi và cho những đỉnh không có cung đi vào đứng chờ lần lượt trong hàng đợi đó, lần lượt rút các đỉnh khỏi hàng đợi và đánh số cho nó, đồng thời huỷ những cung đi ra khỏi đỉnh vừa đánh số, lưu ý sau mỗi lần loại bỏ cung (u, v),

nếu thấy bán bậc vào của  $v = 0$  thì đẩy  $v$  vào chờ trong hàng đợi, như vậy đỡ mất công duyệt để tìm những đỉnh có bán bậc vào = 0)

Nếu các đỉnh được đánh số sao cho mỗi cung phải nối từ một đỉnh tới một đỉnh khác mang chỉ số lớn hơn thì thuật toán tìm đường đi ngắn nhất có thể mô tả rất đơn giản:

Gọi  $d[v]$  là độ dài đường đi ngắn nhất từ  $S$  tới  $v$ .

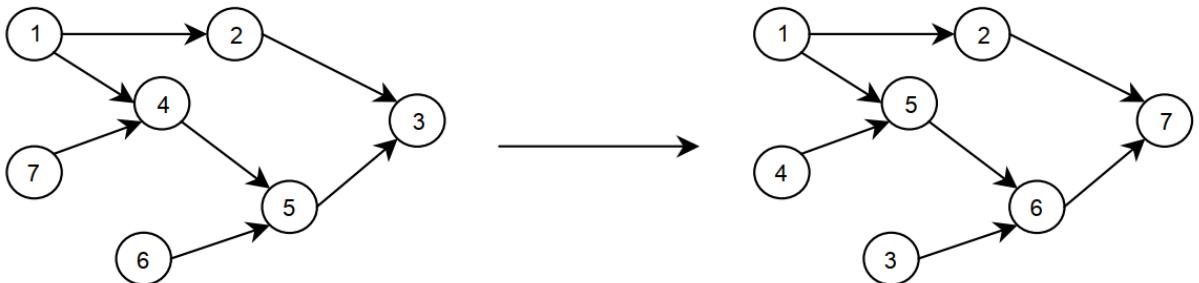
**Khởi tạo**  $d[v] = c[S, v]$ . Ta sẽ tính các  $d[v]$  như sau:

**for**  $u := 1$  to  $n - 1$  **do**  
**for**  $v := u + 1$  to  $n$  **do**

$d[v] := \min(d[v], d[u] + c[u, v]);$

(Giả thiết rằng  $c[u, v] = +\infty$  nếu như  $(u, v)$  không là cung).

Tức là dùng đỉnh  $u$ , tối ưu nhän  $d[v]$  của những đỉnh  $v$  nối từ  $u$ , với  $u$  được xét lần lượt từ 1 tới  $n - 1$ . Có thể làm tốt hơn nữa bằng cách chỉ cần cho  $u$  chạy từ đỉnh xuất phát  $S$  tới đỉnh kết thúc  $F$ . Bởi nếu  $u$  chạy tới đâu thì nhän  $d[u]$  là không thể cực tiểu hoá thêm nữa.



## 6.5. Đường đi ngắn nhất giữa tất cả các cặp đỉnh

Cho đơn đồ thị có hướng, có trọng số  $G = (V, E)$  với  $n$  đỉnh và  $m$  cạnh. Bài toán đặt ra là hãy tính tất cả các  $d(u, v)$  là khoảng cách từ  $u$  tới  $v$ . Rõ ràng là ta có thể áp dụng thuật toán tìm đường đi ngắn nhất xuất phát từ một đỉnh với  $n$  khả năng chọn đỉnh xuất phát. Nhưng ta có cách làm gọn hơn nhiều, cách làm này rất giống với thuật toán Warshall mà ta đã biết: Từ ma trận trọng số  $c$ , thuật toán Floyd tính lại các  $c[u, v]$  thành độ dài đường đi ngắn nhất từ  $u$  tới  $v$ :

Với mọi đỉnh  $k$  của đồ thị được xét theo thứ tự từ 1 tới  $n$ , xét mọi cặp đỉnh  $u, v$ . Cực tiểu hoá  $c[u, v]$  theo công thức:

$$c[u, v] := \min(c[u, v], c[u, k] + c[k, v])$$

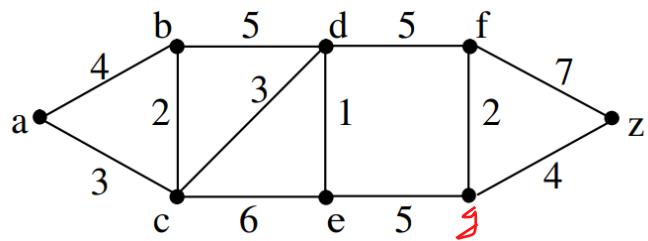
Tức là nếu như đường đi từ  $u$  tới  $v$  đang có lại dài hơn đường đi từ  $u$  tới  $k$  cộng với đường đi từ  $k$  tới  $v$  thì ta huỷ bỏ đường đi từ  $u$  tới  $v$  hiện thời và coi đường đi từ  $u$  tới  $v$  sẽ là nối của hai đường đi từ  $u$  tới  $k$  rồi từ  $k$  tới  $v$  (Chú ý rằng ta còn có việc lưu lại vết):

```
for k := 1 to n do
    for u := 1 to n do
        for v := 1 to n do
```

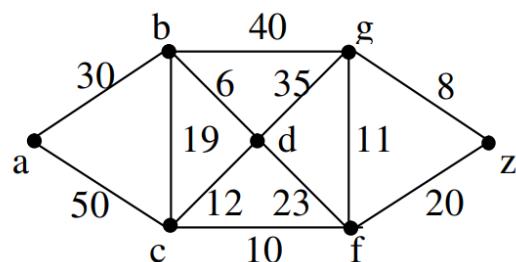
$$c[u, v] := \min(c[u, v], c[u, k] + c[k, v]).$$

**Bài tập**

1. Tìm đường đi ngắn nhất giữa a và z trong đồ thị có trọng số sau



2. Tìm đường đi ngắn nhất giữa a và z trong đồ thị có trọng số sau



**Chương 7****Bài toán luồng cực đại trong mạng****7.1 Mạng, luồng trong mạng và bài toán luồng cực đại**

Ta gọi **mạng** là một **đồ thị có hướng**  $G = (V, E)$ , trong đó có **điểm phát** duy nhất một **điểm A** **không có cung đi vào** gọi là **điểm phát**, duy nhất một **điểm B** **không có cung đi ra** gọi là **điểm thu** và **mỗi cung**  $e = (u, v) \in E$  được gán với **một số không âm**  $c(e) = c[u, v]$  gọi là **khả năng thông qua** của cung đó. Để thuận tiện cho việc trình bày, ta **quy ước** rằng nếu không có cung  $(u, v)$  thì khả năng thông qua  $c[u, v]$  của nó được gán bằng 0.

Nếu có mạng  $G = (V, E)$ . Ta gọi **luồng**  $f$  trong mạng  $G$  là một phép gán cho mỗi cung  $e = (u, v) \in E$  một số thực không âm  $f(e) = f[u, v]$  gọi là **luồng** trên cung  $e$ , thỏa mãn các điều kiện sau:

- **Luồng** trên mỗi cung không vượt quá khả năng thông qua của nó:

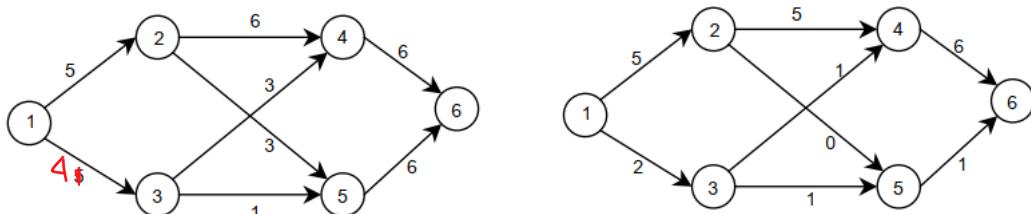
$$0 \leq f[u, v] \leq c[u, v] \quad (\forall (u, v) \in E).$$

- Với mọi **điểm**  $v$  không trùng với **điểm phát**  $A$  và **điểm thu**  $B$ , **tổng luồng** trên các cung **đi vào**  $v$  bằng **tổng luồng** trên các cung **đi ra**  $v$ :

$$\sum_{u \in \Gamma^-(v)} f(u, v) = \sum_{w \in \Gamma^+(v)} f(v, w).$$

Trong đó:  $\Gamma^-(v) = \{u \in V \mid (u, v) \in E\}$ ,  $\Gamma^+(v) = \{w \in V \mid (v, w) \in E\}$

Giá trị của một luồng là **tổng luồng** trên các cung **đi ra** **điểm phát** = **tổng luồng** trên các cung **đi vào** **điểm thu**

**Bài toán luồng cực đại:**

Cho mạng vận tải  $G = (V, E)$ . Hãy tìm luồng  $f^*$  trong mạng với **giá trị luồng** lớn nhất. Luồng như vậy gọi là **luồng cực đại** trong mạng và bài toán này gọi là **bài toán tìm luồng cực đại** trên mạng.

**7.2. Lát cắt. Đường tăng luồng. Định lý Ford - Fullkerson****Đường tăng luồng:**

**Định nghĩa** Ta gọi **lát cắt**  $(X, Y)$  là một cách phân hoạch tập **điểm**  $V$  của mạng thành hai tập rời nhau  $X$  và  $Y$ , trong đó  $X$  chứa **điểm phát** và  $Y$  chứa **điểm thu**. **Khả năng thông qua** của lát cắt  $(X, Y)$  là **tổng** tất cả các **khả năng thông qua** của các cung  $(u, v)$  có  $u \in X$  và  $v \in Y$ . Lát cắt với **khả năng thông qua** nhỏ nhất gọi là **lát cắt hẹp nhất**.

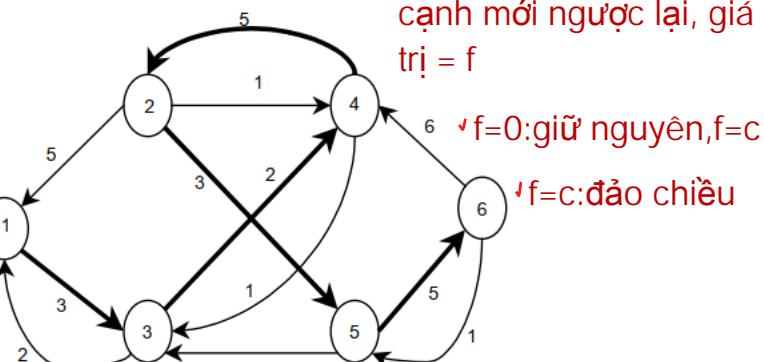
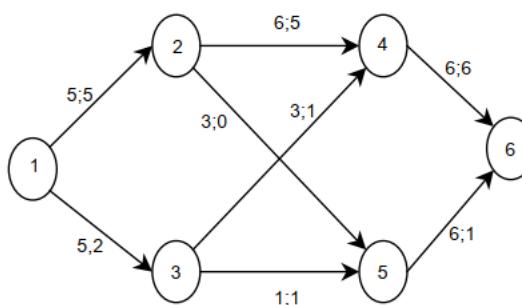
**Định lý Ford - Fullkerson**

~~Giá trị luồng cực đại trên mạng đúng bằng khả năng thông qua của lát cắt hẹp nhất.~~  
 Việc chứng minh định lý Ford- Fulkerson đã xây dựng được một thuật toán tìm luồng cực đại trên mạng:

Giả sử  $f$  là một luồng trong mạng  $G = (V, E)$ . Từ mạng  $G = (V, E)$  ta xây dựng đồ thị có trọng số  $G_f = (V, E_f)$  như sau:

Xét những cạnh  $e = (u, v) \in E$  ( $c[u, v] > 0$ ):

- Nếu  $f[u, v] < c[u, v]$  thì ta thêm cung  $(u, v)$  vào  $E_f$  với trọng số  $c[u, v] - f[u, v]$ , cung đó gọi là cung thuận. Về ý nghĩa, trọng số cung này cho biết còn có thể tăng luồng  $f$  trên cung  $(u, v)$  một lượng không quá trọng số đó.
- Xét tiếp nếu  $f[u, v] > 0$  thì ta thêm cung  $(v, u)$  vào  $E_f$  với trọng số  $f[u, v]$ , cung đó gọi là cung nghịch. Về ý nghĩa, trọng số cung này cho biết còn có thể giảm luồng  $f$  trên cung  $(u, v)$  một lượng không quá trọng số đó. Đồ thị  $G_f$  được gọi là **đồ thị tăng luồng**.

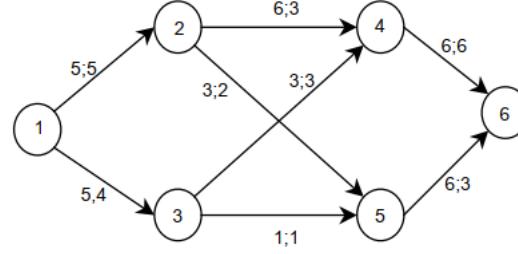
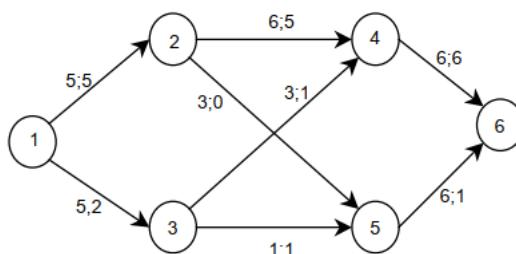


Giả sử  $P$  là một đường đi cơ bản từ đỉnh phát  $A$  tới đỉnh thu  $B$ . Gọi **Đelta là giá trị nhỏ nhất của các trọng số của các cung trên đường đi  $P$** . Ta sẽ tăng giá trị của luồng  $f$  bằng cách đặt:

- $f[u, v] := f[u, v] + \Delta$ , nếu  $(u, v)$  là cung trong đường  $P$  và là cung thuận
- $f[v, u] := f[v, u] - \Delta$ , nếu  $(u, v)$  là cung trong đường  $P$  và là cung nghịch
- Còn luồng trên những cung khác giữ nguyên

Có thể kiểm tra luồng  $f$  mới xây dựng vẫn là luồng trong mạng và giá trị của luồng  $f$  mới được tăng thêm  $\Delta$  so với giá trị luồng  $f$  cũ. Ta gọi thao tác biến đổi luồng như vậy là **tăng luồng dọc đường  $P$** , **đường đi cơ bản  $P$  từ  $A$  tới  $B$**  được gọi là **đường tăng luồng**.

Ví dụ: với đồ thị tăng luồng  $G_f$  như trên, giả sử chọn đường đi  $(1, 3, 4, 2, 5, 6)$ . Giá trị nhỏ nhất của trọng số trên các cung là 2, vậy thì ta sẽ tăng các giá trị  $f[1, 3]$ ,  $f[3, 4]$ ,  $f[2, 5]$ ,  $f[5, 6]$  lên 2, (do các cung đó là cung thuận) và giảm giá trị  $f[2, 4]$  đi 2 (do cung  $(4, 2)$  là cung nghịch). Được luồng mới mang giá trị 9.



### 7.3. Thuật toán tìm luồng cực đại trong mạng

Bước 1: Khởi tạo:

Một luồng bất kỳ trên mạng, chẳng hạn như luồng 0 (luồng trên các cung đều bằng 0), sau đó:

Tim  $f := c_{\min}$  trên 1 đường đi từ A đến B

Bước 2: Lặp hai bước sau:

- Tim đường tăng luồng P đối với luồng hiện có  $\equiv$  Tìm đường đi cơ bản từ A tới B trên đồ thị tăng luồng, nếu không tìm được đường tăng luồng thì bước lặp kết thúc.
- Tăng luồng dọc theo đường P

Bước 3: Thông báo giá trị luồng cực đại tìm được

### Bài tập

1. ~~Tìm  $W^*$  bằng cách áp dụng thuật toán Floyd vào đồ thị sau:~~

