

TRƯỜNG ĐẠI HỌC QUỐC TẾ HỒNG BÀNG
KHOA CÔNG NGHỆ THÔNG TIN
oOo

LÝ THUYẾT ĐỒ THỊ

(*Graph Theory*)

(TÀI LIỆU THAM KHẢO)

LÊ VĂN HẠNH

MỤC LỤC

1	ĐẠI CƯƠNG VỀ ĐỒ THỊ (GRAPH)	1
1.1.	KHÁI NIỆM	1
1.1.1.	Định nghĩa	1
1.1.2.	Biểu diễn đồ thị	2
1.1.3.	Đồ thị có hướng và đồ thị vô hướng	2
1.1.4.	Đơn đồ thị (simple graph), đa đồ thị (Multigraph)	2
1.1.5.	Một số dạng đơn đồ thị đặc biệt	2
1.1.6.	Bậc của một đỉnh (degree)	5
1.2.	MỘT SỐ ỨNG DỤNG CỦA CÁC ĐỒ THỊ ĐẶC BIỆT	6
1.2.1.	Các mạng cục bộ (LAN)	6
1.2.2.	Xử lý song song	6
1.3.	ĐƯỜNG ĐI, CHU TRÌNH VÀ LIÊN THÔNG	8
1.3.1.	Đường đi	8
1.3.2.	Chu trình	8
1.3.3.	Liên thông	8
1.4.	ĐỒ THỊ CON VÀ ĐỒ THỊ RIÊNG	9
1.4.1.	Định nghĩa đồ thị con	9
1.4.2.	Định nghĩa đồ thị riêng	10
1.5.	SỰ ĐẲNG HÌNH	10
1.6.	BÀI TẬP	11
1.7.	CÂU HỎI ÔN TẬP	14
2	BIỂU DIỄN ĐỒ THỊ TRÊN MÁY TÍNH & CÁC THUẬT TOÁN TÌM KIẾM TRÊN ĐỒ THỊ	15
2.1.	BIỂU DIỄN ĐỒ THỊ TRÊN MÁY TÍNH	15
2.1.1.	Ma trận kề - Ma trận trọng số	15
2.1.2.	Danh sách cạnh (cung)	16
2.1.3.	Danh sách kề	17
2.2.	TÌM KIẾM TRÊN ĐỒ THỊ	18
2.2.1.	Tìm kiếm theo chiều sâu trên đồ thị (Depth First Search)	18
2.2.2.	Tìm kiếm theo chiều rộng trên đồ thị (Breadth First Search)	23
2.2.3.	Ứng dụng DFS/BFS trong kiểm tra tính liên thông và tìm đường đi	28
2.3.	BÀI TẬP	30
2.4.	CÂU HỎI ÔN TẬP	30
3	ĐỒ THỊ EULER & ĐỒ THỊ HAMILTON	32
3.1.	ĐƯỜNG ĐI EULER VÀ ĐỒ THỊ EULER	32
3.1.1.	Giới thiệu	32
3.1.2.	Định nghĩa	32
3.1.3.	Ví dụ	32
3.1.4.	Định lý Euler	33
3.1.5.	Xác định chu trình Euler bằng thuật toán Flor	33
3.1.6.	Thuật toán tìm chu trình Euler bằng cách sử dụng cấu trúc Stack	34
3.2.	ĐƯỜNG ĐI HAMILTON VÀ ĐỒ THỊ HAMILTON	37
3.2.1.	Giới thiệu	37
3.2.2.	Đồ thị Hamilton	37
3.2.3.	Xác định đường đi và đồ thị Hamilton	38
3.2.4.	Một số ứng dụng của đồ thị Hamilton	40
3.3.	BÀI TẬP	42
4	BÀI TOÁN ĐƯỜNG ĐI NGẮN NHẤT	46

4.1.	CÁC KHÁI NIỆM	46
4.2.	THUẬT TOÁN DIJKSTRA	46
4.2.1.	Công dụng	46
4.2.2.	Cách thực hiện	46
4.2.3.	Ví dụ	47
4.3.	THUẬT TOÁN FLOYD	50
4.3.1.	Công dụng	50
4.3.2.	Thuật toán Floyd	50
4.3.3.	Ví dụ	50
4.4.	BÀI TẬP	54
5	ĐỒ THỊ PHẪNG	59
5.1.	ĐỒ THỊ PHẪNG	59
5.1.1.	Định nghĩa	59
5.1.2.	Quy ước	59
5.1.3.	Ví dụ	59
5.1.4.	Đồng phôi	59
5.1.5.	Đồ thị phẳng	60
5.1.6.	Đồ thị không phẳng	60
5.2.	TÔ MÀU ĐỒ THỊ	61
5.2.1.	Tô màu bản đồ	61
5.2.2.	Tô màu đồ thị	61
5.2.3.	Những ứng dụng của bài toán tô màu đồ thị	62
5.3.	BÀI TẬP	63
6	CÂY (TREE)	67
6.1.	ĐỊNH NGHĨA VÀ CÁC TÍNH CHẤT CƠ BẢN	67
6.1.1.	Định nghĩa	67
6.1.2.	Mệnh đề	67
6.1.3.	Định lý	67
6.2.	CÂY KHUNG VÀ BÀI TOÁN TÌM CÂY KHUNG NHỎ NHẤT	67
6.2.1.	Định nghĩa	67
6.2.2.	Bài toán tìm cây khung nhỏ nhất	68
6.2.3.	Thuật toán Kruskal	68
6.2.4.	Thuật toán Prim	69
6.3.	SO SÁNH HAI THUẬT TOÁN Prim & Kruskal	70
6.4.	CÂY CÓ GỐC	71
6.4.1.	Định nghĩa về cây có gốc	71
6.4.2.	Đỉnh ngoài – đỉnh trong	72
6.4.3.	Cây nhị phân	72
6.4.4.	Mệnh đề về cây m phân	72
6.5.	DUYỆT CÂY NHỊ PHÂN	72
6.5.1.	Định nghĩa	72
6.5.2.	Các thuật toán duyệt cây nhị phân	73
6.5.3.	Ký pháp Ba Lan	73
6.6.	BÀI TẬP	76
	TÀI LIỆU THAM KHẢO	79

ĐẠI CƯƠNG VỀ ĐỒ THỊ (GRAPH)

Lý thuyết đồ thị là một ngành khoa học được phát triển từ lâu nhưng lại có nhiều ứng dụng hiện đại. Những ý tưởng cơ bản của nó được đưa ra từ thế kỷ 18 bởi nhà toán học Thụy Sĩ tên là Leonhard Euler. Ông đã dùng đồ thị để giải quyết bài toán 7 chiếc cầu Königsberg nổi tiếng.

Đồ thị được dùng để giải các bài toán trong nhiều lĩnh vực khác nhau. Ví dụ, ta dùng đồ thị để:

- Xác định xem có thực hiện một mạch điện trên một bảng điện phẳng được không.
- Phân biệt hai hợp chất hóa học có cùng công thức phân tử nhưng có cấu trúc khác nhau nhờ đồ thị.
- Xác định xem hai máy tính có được nối với nhau bằng một đường truyền thông hay không thông qua mô hình đồ thị mạng máy tính.
- Giải các bài toán như bài toán tìm đường đi ngắn nhất giữa hai thành phố trong một mạng giao thông (sau khi đã gán các trọng số cho các cạnh của nó).
- Lập lịch thi và phân chia kênh cho các đài truyền hình.
- Lập sơ đồ khối tính toán của một thuật toán,
- Biểu diễn sự cạnh tranh các loài trong một môi trường sinh thái.
- Biểu diễn ai có ảnh hưởng lên ai trong một tổ chức nào đó.
- Biểu diễn các kết cục của cuộc thi đấu thể thao.
- Giải các bài toán như bài toán tính số các tổ hợp khác nhau của các chuyến bay giữa hai thành phố trong một mạng hàng không.
- Giải bài toán đi tham quan tất cả các đường phố của một thành phố sao cho mỗi đường phố đi qua đúng một lần.
- Tìm số các màu cần thiết để tô các vùng khác nhau của một bản đồ.
- ...

1.1. KHÁI NIỆM

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.

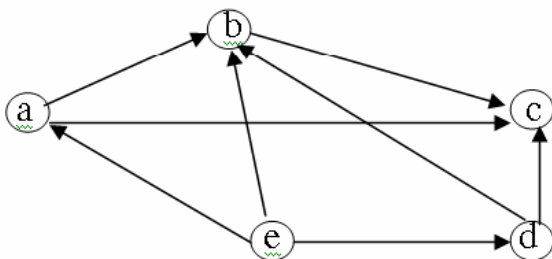
1.1.1. Định nghĩa

Đồ thị là một cặp $G = (V, E)$, trong đó:

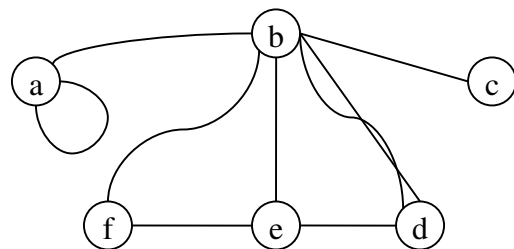
- V là tập hợp các đỉnh (*Vertex/Vertices*),
- $E \subseteq V \times V$ là tập hợp các cạnh (*Edges*).

Trong tài liệu này chúng ta chỉ xét các đồ thị hữu hạn (*finite graph*) và khác rỗng, nghĩa là các đồ thị có tập đỉnh là hữu hạn.

Ví dụ:



Hình 1.1: Đồ thị có hướng (G1)



Hình 1.2: Đồ thị vô hướng (G2)

Đồ thị G_1 ở trên có tập các đỉnh $V = \{a, b, c, d, e\}$ và tập các cạnh $E = \{(a, b), (a, c), (b, c), (d, b), (d, c), (e, a), (e, b), (e, d)\}$.

Nếu (a, b) là một cạnh của đồ thị thì ta nói rằng đỉnh b kề với đỉnh a và cả hai đỉnh a và b kề với cạnh (a, b) .

- Cạnh khuyên: một cạnh aa tương ứng với 2 đỉnh trùng nhau (a) gọi là cạnh khuyên (hình 1.2, có cạnh khuyên tại đỉnh a).
- Hai cạnh song song (*parallel edges*): là 2 cạnh phân biệt cùng tương ứng với 1 cặp đỉnh (hình 1.2, có 2 cạnh song song vì cùng tương ứng với 2 đỉnh b và d).

1.1.2. Biểu diễn đồ thị

Ta có thể biểu diễn hình học cho đồ thị trên mặt phẳng như sau:

- **Đỉnh**: biểu diễn bằng các vòng tròn nhỏ, chứa tên của đỉnh.
- **Cạnh**:
 - Cạnh vô hướng: biểu diễn bằng đoạn thẳng.
 - Cạnh có hướng: biểu diễn bằng mũi tên nối hai đỉnh của đồ thị.

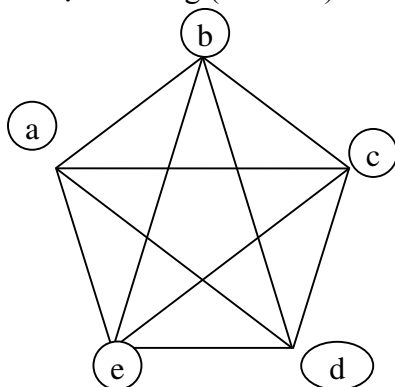
1.1.3. Đồ thị có hướng và đồ thị vô hướng

Cho đồ thị $G = (V, E)$, G được gọi là đồ thị:

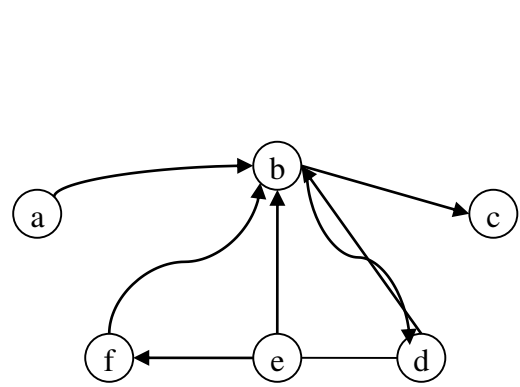
- **Vô hướng**: khi đồ thị chỉ chứa các cạnh vô hướng (hình 1.2).
- **Có hướng**: khi đồ thị chỉ chứa các cạnh có hướng (hình 1.1).

1.1.4. Đơn đồ thị (simple graph), đa đồ thị (Multigraph)

- Cho đồ thị $G = (V, E)$, G được gọi là:
 - **Đơn đồ thị**: mà mỗi cặp đỉnh được nối với nhau bởi không quá một cạnh (thường được gọi tắt là đồ thị - hình 1.1).
 - **Đa đồ thị**: khi đồ thị có những cặp đỉnh được nối với nhau nhiều hơn một cạnh thì được gọi là đa đồ thị (hình 1.2).
- Từ 1.3 và 1.4, ta có thể có các dạng đồ thị sau:
 - Đơn đồ thị vô hướng (hình 1.3)
 - Đơn đồ thị có hướng (hình 1.1)
 - Đa đồ thị vô hướng (hình 1.2)
 - Đa đồ thị có hướng (hình 1.4)



Hình 1.3. Đơn đồ thị vô hướng



Hình 1.4 Đa đồ thị có hướng

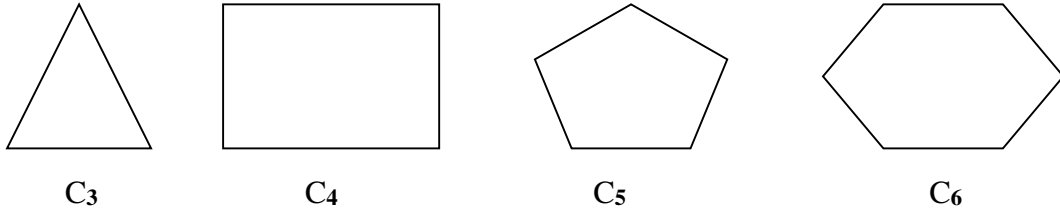
1.1.5. Một số dạng đơn đồ thị đặc biệt

1.1.5.1. Đồ thị đầy đủ (complete graph)

Đồ thị $G = (V, E)$ được gọi là đồ thị đầy đủ khi mọi cặp đỉnh đều kề nhau, hay nói cách khác là đồ thị mà mọi cặp đỉnh đều có cạnh nối với nhau (hình 1.3).

1.1.5.2. Đồ thị vòng

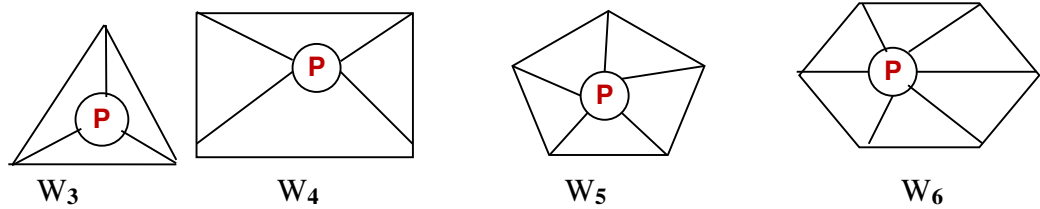
Đồ thị $G = (V, E)$ được gọi là đồ thị vòng khi số lượng đỉnh của đồ thị ≥ 3 , bậc của tất cả các đỉnh đều bằng 2 và các cạnh nối với nhau thành 1 vòng khép kín (hình 1.5). Ký hiệu: C_n



Hình 1.5. Đồ thị vòng

1.1.5.3. Đồ thị bánh xe

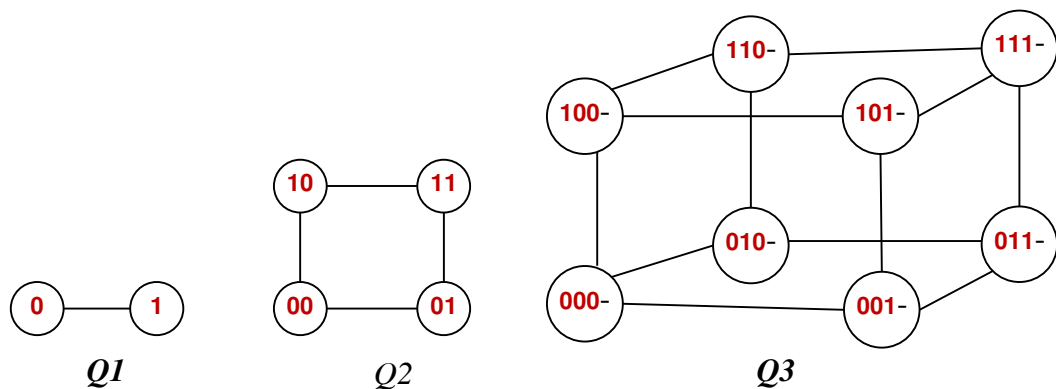
Đồ thị $G = (V, E)$ được gọi là đồ thị bánh xe khi đó là đồ thị vòng và có bổ sung thêm một đỉnh mới P , đỉnh này được nối với tất cả các đỉnh còn lại (hình 1.6). Ký hiệu W_n .



Hình 1.6. Đồ thị bánh xe

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

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



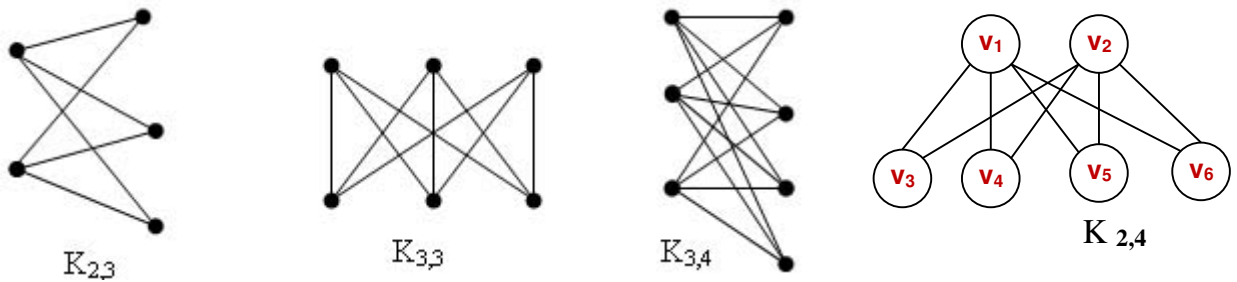
Hình 1.7. Đồ thị lập phương

1.1.5.5. 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.

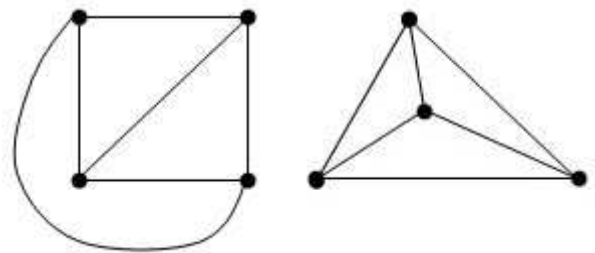
Nếu đồ thị phân đôi $G=(V_1 \cup V_2, E)$ sao cho với mọi $v_1 \in V_1, v_2 \in V_2, (v_1, v_2) \in E$ thì G được gọi là đồ thị phân đôi đầy đủ. Nếu $|V_1|=m, |V_2|=n$ thì đồ thị phân đôi đầy đủ G ký hiệu là $K_{m,n}$. Như vậy $K_{m,n}$ có $m.n$ cạnh, các đỉnh của V_1 có bậc n và các đỉnh của V_2 có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

1.1.5.6. Đồ thị phẳng



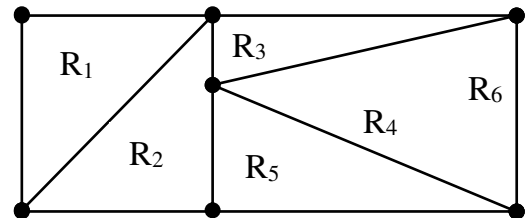
Hình 1.8. Đồ thị phân đôi

- 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.
- Ví 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 (hình 1.9).



Hình 1.9. Đồ thị K_4 là đồ thị phẳng

- Công thức Euler:
 - Biểu diễn phẳng của đồ thị sẽ chia mặt phẳng ra thành các miền. Ví dụ, biểu diễn phẳng của đồ thị cho trong hình 1.10 chia mặt phẳng ra thành 6 miền R_1, R_2, \dots, R_6 .



Hình 1.10. Các miền tương ứng với biểu diễn phẳng của đồ thị

Euler đã chứng minh được mối liên hệ giữa số miền, số đỉnh của đồ thị và số cạnh của đồ thị phẳng qua định lý sau:

- **Định lý về 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

$$r = m - n + 2$$

- **Ví 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 ?

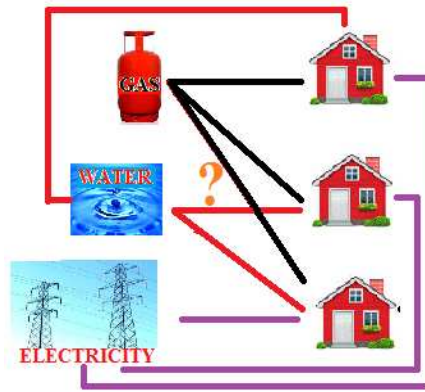
Giải: Do mỗi đỉnh của đồ thị đều có bậc là 3, nên tổng bậc của các đỉnh là $3 \times 20 = 60$. Từ đó suy ra số cạnh của đồ thị $m = 60/2 = 30$. Vì vậy, theo công thức Euler, số miền cần tìm là

$$r = 30 - 20 + 2 = 12.$$

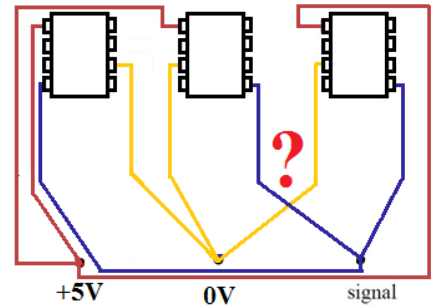
- Ứng dụng của đồ thị phẳng:

- 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 (điện, nước, gas) 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.



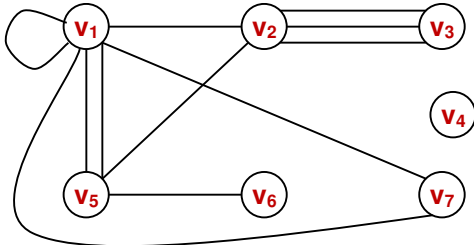
Hình 1.11. Minh họa đồ thị $K_{3,3}$ qua bài toán cung cấp 3 hệ thống năng lượng cho 3 căn hộ



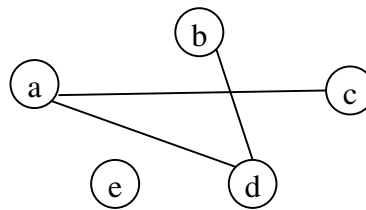
Hình 1.12. Minh họa đồ thị $K_{3,3}$ qua bài toán chế tạo mạch in

1.1.6. Bậc của một đỉnh (degree)

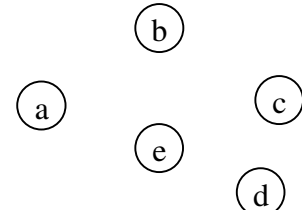
- Xét một đỉnh v bất kỳ của đồ thị $G = (V, E)$, số lượng cạnh nối tới đỉnh v gọi là bậc của đỉnh v . Cứ mỗi cạnh khuyên tại đỉnh v được tính là 2.
- Ký hiệu $d(v)$.
- Đỉnh cô lập (isolated vertex): là đỉnh có bậc = 0 (đỉnh d trong hình 1.13; đỉnh e trong hình 1.14; đỉnh a, b, c, d, e trong hình 1.15).
- Đỉnh treo (pendant vertex): là đỉnh có bậc = 1 (đỉnh f trong hình 1.13; đỉnh b, c trong hình 1.14).
- Đồ thị rỗng: là đồ thị có tất cả các đỉnh đều là đỉnh cô lập (hình 1.15).



Hình 1.13



Hình 1.14



Hình 1.15 Đồ thị rỗng

1.1.6.1. Đồ thị vô hướng

- Định lý 1: với mọi đồ thị $G = (V, E)$, số cạnh E của G được xác định bởi công thức:

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

- Hệ luận 1: Trong đồ thị vô hướng, số lượng đỉnh bậc lẻ là một số chẵn.

1.1.6.2. Đồ thị có hướng

- Định lý 2: Cho đồ thị có hướng $G = (V, E)$, ta gọi bậc trong của một đỉnh là số cung có hướng đi ra khỏi đỉnh (ký hiệu $d^+(v)$); tương tự, ta gọi bậc ngoài của một đỉnh là số cung có hướng đi vào đỉnh (ký hiệu $d^-(v)$). Ta có: tổng bậc ngoài của tất cả các đỉnh bằng với tổng bậc trong của tất cả các đỉnh

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

1.2. MỘT SỐ ỨNG DỤNG CỦA CÁC ĐỒ THỊ ĐẶC BIỆT

1.2.1. Các mạng cục bộ (LAN)

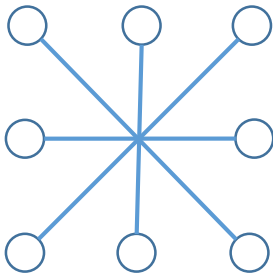
1.2.1.1. Mạng cục bộ dùng cấu trúc hình sao

Hình 1.16: 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.

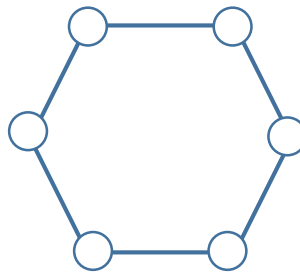
1.2.1.2. Mạng cục bộ dùng cấu trúc hình tròn

Hình 1.17: 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.

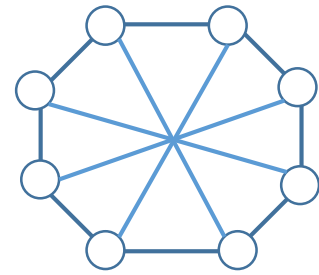
1.2.1.3. Mạng cục bộ dùng cấu trúc hỗn hợp (hay đồ thị bánh xe)



Hình 1.16. Cấu trúc hình sao



Hình 1.17. Cấu trúc vòng tròn



Hình 1.18. Cấu trúc hỗn hợp

Hình 1.18: một số mạng cục bộ dùng cấu trúc hỗn hợp của hai cấu trúc trên. Các thông báo được truyền vòng quanh theo vòng tròn hoặc có thể qua thiết bị trung tâm. Sự dư thừa này có thể làm cho mạng đáng tin cậy hơn. Mạng cục bộ kiểu này có thể biểu diễn bằng một đồ thị bánh xe W_n .

1.2.2. Xử lý song song

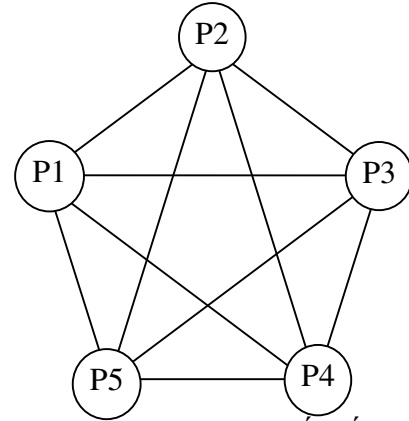
Các thuật toán để giải các bài toán được thiết kế để thực hiện một phép toán tại mỗi thời điểm là thuật toán nối tiếp. Tuy nhiên, nhiều bài toán với số lượng tính toán rất lớn như bài toán mô phỏng thời tiết, tạo hình trong y học hay phân tích mật mã không thể giải được trong một khoảng thời gian hợp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

1.2.2.1. Mạng kết nối kiểu đồ thị đầy đủ

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.

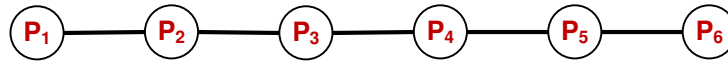


Hình 1.19. Mạng kết nối kiểu đồ thị đầy đủ

1.2.2.2. Mạng kết nối kiểu mảng 1 chiều

Các bộ xử lý có thể kết nối đơn giản là sắp xếp chúng theo một mảng một chiều.

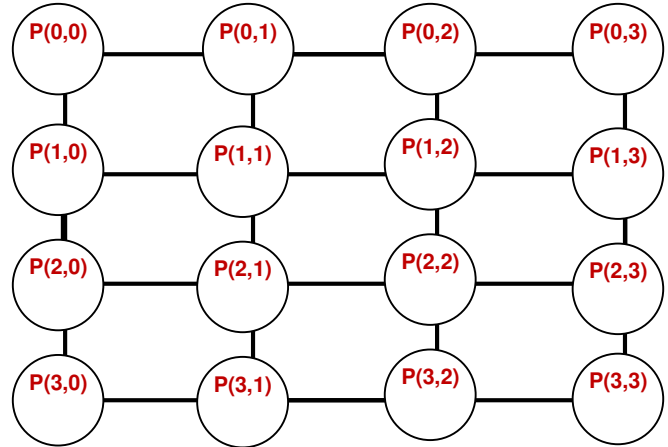
- Ưu điểm: mỗi bộ xử lý có nhiều nhất 2 đường nối trực tiếp với các bộ xử lý khác.
- Nhược điểm: nhiều khi cần có rất nhiều các kết nối trung gian để các bộ xử lý trao đổi thông tin với nhau.



Hình 1.20. Mạng kết nối kiểu mảng 1 chiều

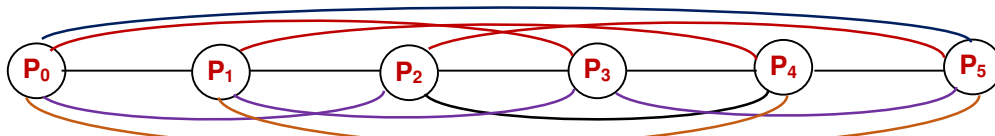
1.2.2.3. Mạng kiểu lưới (hoặc mảng hai chiều)

Rất hay được dùng cho các mạng liên kết. Trong một mạng như thế, số các bộ xử lý là một số chính phương, $n=m^2$. Các bộ xử lý được gán nhãn $P(i,j)$, $0 \leq i, j \leq m-1$. Các kết nối hai chiều sẽ nối bộ xử lý $P(i,j)$ với bốn bộ xử lý bên cạnh, tức là với $P(i,j+1)$ và $P(i-1,j)$ chừng nào các bộ xử lý còn ở trong lưới.



Hình 1.21. Mạng kết nối kiểu lưới

1.2.2.4. Mạng kết nối kiểu siêu khối



Hình 1.22. Mạng kết nối kiểu siêu khối

Với các mạng loại này số các bộ xử lý là lũy thừa của 2, $n=2^m$. Các bộ xử lý được gán nhãn là P_0, P_1, \dots, P_{n-1} . Mỗi bộ xử lý có liên kết hai chiều với m bộ xử lý khác. Bộ xử lý P_i nối với bộ xử lý có chỉ số biểu diễn bằng dãy nhị phân khác với dãy nhị phân biểu diễn i tại đúng một bit. Mạng kiểu siêu khối cân bằng số các kết nối trực tiếp của mỗi bộ xử lý và số các kết nối gián tiếp sao cho các bộ xử lý có thể truyền thông được. Nhiều máy tính đã chế tạo theo mạng kiểu siêu khối và

nhiều thuật toán đã được thiết kế để sử dụng mạng kiểu siêu khối. Đồ thị lập phương Q_m biểu diễn mạng kiểu siêu khối có 2^m bộ xử lý.

1.3. ĐƯỜNG ĐI, CHU TRÌNH VÀ LIÊN THÔNG

Giả sử $G = (V, E)$ là một đồ thị.

1.3.1. Đường đi

Đường đi trong đồ thị là một dãy các đỉnh:

$$\langle x_1, x_2, \dots, x_i, x_{i+1}, \dots, x_{k-1}, x_k \rangle$$

sao cho, mỗi đỉnh trong dãy (không kể đỉnh đầu tiên) kề với đỉnh trước nó bằng một cạnh nào đó, nghĩa là: $\forall i = 2, 3, \dots, k-1, k : (x_{i-1}, x_i) \in E$.

Vậy:

- Đường đi này đi từ đỉnh đầu x_1 đến đỉnh cuối x_k .
- Số cạnh của đường đi được gọi là *độ dài* của đường đi đó.
- Đường đi đơn là đường đi mà các đỉnh trên nó khác nhau từng đôi.

1.3.2. Chu trình

Chu trình là một đường đi khép kín (tức là đỉnh cuối của đường trùng với đỉnh đầu của đường). Ta thường ký hiệu chu trình là:

$$[x_1, x_2, \dots, x_i, x_{i+1}, \dots, x_{k-1}, x_k], \text{ trong đó } x_1 = x_k$$

Để cho gọn, trong ký hiệu của chu trình thường không viết đỉnh cuối:

$$[x_1, x_2, \dots, x_i, x_{i+1}, \dots, x_{k-1}]$$

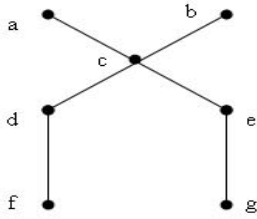
- Khi nói đến một chu trình, ta cũng không cần xác định đỉnh đầu và đỉnh cuối của chu trình đó.
- Chu trình được gọi là *chu trình đơn* nếu các đỉnh trên nó khác nhau từng đôi.
- Một số tính chất về chu trình trên đồ thị vô hướng:
 - Đồ thị vô hướng với n đỉnh ($n \geq 3$), không có đỉnh nút và bậc của mỗi đỉnh đều không nhỏ hơn 2, luôn có chu trình đơn.
 - Đồ thị vô hướng với n đỉnh ($n \geq 4$) và bậc của mỗi đỉnh đều không nhỏ hơn 3, luôn có chu trình đơn độ dài chẵn.

1.3.3. Liên thông

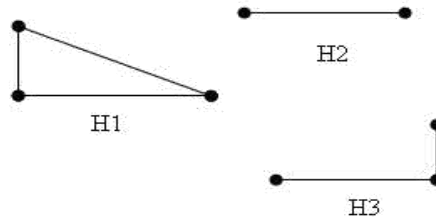
1.3.3.1. Đồ thị vô hướng

- Đồ thị liên thông: Đồ thị vô hướng $G = (V, E)$ được gọi là liên thông nếu luôn tìm được đường đi giữa hai đỉnh bất kỳ trong đồ thị (hình 1.23).
- Thành phần liên thông: Trong trường hợp đồ thị là không liên thông, nó sẽ rã ra thành một số đồ thị con liên thông không có đỉnh chung. Những đồ thị con liên thông như vậy ta sẽ gọi là các *thành phần liên thông* của đồ thị (hình 1.24).
- Ứng dụng: xác định 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.

Ví dụ:



Hình 1.23. Đồ thị liên thông



Hình 1.24. Đồ thị không liên thông gồm 3 thành phần liên thông (H_1, H_2, H_3)

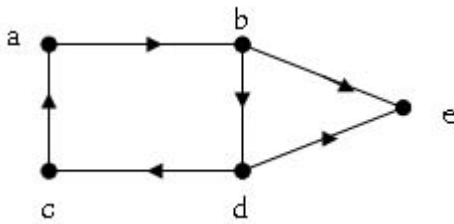
- **Mệnh đề:** Mọi đơn đồ thị n đỉnh ($n \geq 2$) có tổng bậc của hai đỉnh tùy ý không nhỏ hơn n đều là đồ thị liên thông.

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

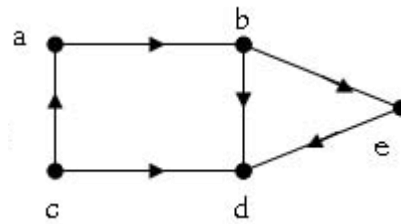
- **Mệnh đề:** Nếu một đồ thị có đúng hai đỉnh bậc lẻ thì hai đỉnh này phải liên thông, tức là có một đường đi nối chúng.

1.3.3.2. Đồ thị có hướng

- **Liên thông mạnh:** 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.
- **Liên thông yếu:** 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.



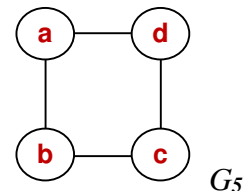
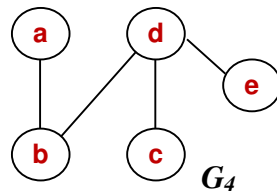
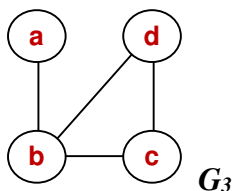
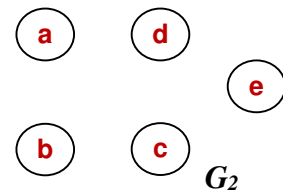
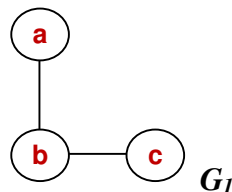
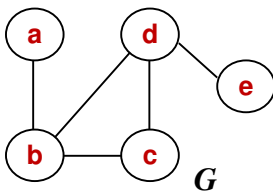
Hình 1.25. Đồ thị liên thông mạnh



Hình 1.26. Đồ thị liên thông yếu

1.4. ĐỒ THỊ CON VÀ ĐỒ THỊ RIÊNG

Giả sử $G = (V, E)$ là một đồ thị.



Hình 1.27. Đồ thị con

1.4.1. Định nghĩa đồ thị con

- Đồ thị $G' = (V', E')$ được gọi là **đồ thị con** của đồ thị G nếu:

$$V' \subseteq V \text{ và } E' = E \cap (V' \times V').$$

Mỗi tập con các đỉnh V' của đồ thị tương ứng duy nhất với một đồ thị con, do vậy để xác định một đồ thị con ta chỉ cần nêu tập đỉnh của nó.

- **Ví dụ:** xét đồ thị G có trong hình 1.27:

- G_1, G_2, G_3 và G_4 là các đồ thị con của G , trong đó G_2 và G_4 là đồ thị con bao trùm của G (có đầy đủ các đỉnh).
- G_5 không phải là đồ thị con của G (do trong G không có cạnh ad).

1.4.2. Định nghĩa đồ thị riêng

- Đồ thị $G'' = (V, E'')$ với $E'' \subseteq E$, được gọi là *đồ thị riêng* của đồ thị G .

Đồ thị riêng là đồ thị giữ nguyên tập đỉnh và bỏ bớt một số cạnh.

- **Ví dụ:** xét đồ thị G có trong hình 1.27:

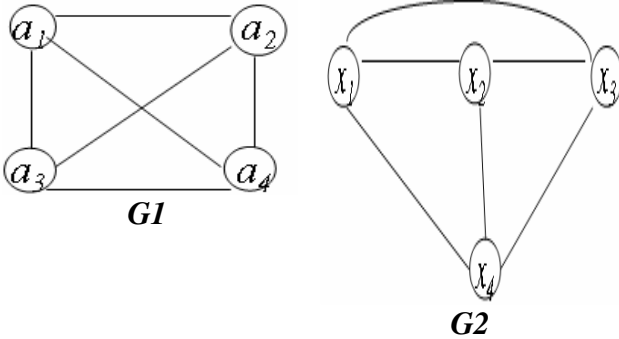
- G_2 , và G_4 là các đồ thị riêng của G .
- G_5 không phải là đồ thị riêng của G (do trong G không có cạnh ad và thiếu đỉnh e).
- G_1 và G_3 không phải là đồ thị riêng của G (do thiếu đỉnh).

1.5. SỰ ĐẲNG HÌNH

Hai đồ thị $G_1(V_1, E_1)$ và $G_2(V_2, E_2)$ được gọi là đẳng hình (isomorphic) với nhau nếu có:

- Cùng số lượng đỉnh
- Cùng số lượng đỉnh bậc k , $\forall k$ nguyên và ≥ 0 .
- Cùng số cạnh.
- Cùng số thành phần.

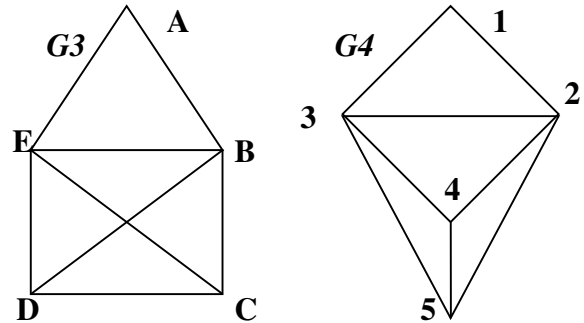
Ví dụ: Một số trường hợp hai đồ thị đẳng hình (hình 1.28 và 1.29)



Hình 1.28. G_1 và G_2 đẳng hình

hay

$a_1 \Leftrightarrow x_1$	$a_1 \Leftrightarrow x_2$
$a_2 \Leftrightarrow x_2$	$a_2 \Leftrightarrow x_3$
$a_3 \Leftrightarrow x_3$	$a_3 \Leftrightarrow x_1$
$a_4 \Leftrightarrow x_4$	$a_4 \Leftrightarrow x_4$



Hình 1.29. G_3 và G_4 đẳng hình

$A \Leftrightarrow 1$
$B \Leftrightarrow 2$
$C \Leftrightarrow 4$
$D \Leftrightarrow 5$
$E \Leftrightarrow 3$

1.6. BÀI TẬP

1.6.1. Vẽ đơn đồ thị khi biết bậc các đỉnh lần lượt là:

i)- 1, 2, 2, 3.

ii)- 4, 3, 3, 2, 2

1.6.2. Lần lượt vẽ các đồ thị mà mọi đỉnh của nó đều có bậc là k ($1 \leq k \leq 5$).

1.6.3. Có bao nhiêu đồ thị đơn có 5 đỉnh và có 4 cạnh. Thực hiện tương tự khi có 6 cạnh

1.6.4. Đồ thị có 10 đỉnh, mỗi đỉnh đều có bậc 5. Hỏi đồ thị có bao nhiêu cạnh.

1.6.5. Cho biết số lượng cạnh của đồ thị đầy đủ gồm n đỉnh.

1.6.6. Có bao nhiêu đồ thị cùng nhận $V = \{1, 2, \dots, n\}$ là tập hợp đỉnh và thỏa điều kiện:

(i) Là đơn đồ thị.

(ii) Không có vòng và có nhiều nhất 2 cạnh song song giữa mỗi cặp đỉnh.

(iii) Có nhiều nhất 1 vòng tại mỗi đỉnh và không có cạnh song song.

1.6.7. Vẽ đơn đồ thị có 6 đỉnh, trong đó có:

(i) 3 đỉnh bậc 3 và 3 đỉnh bậc 1.

(ii) bậc các đỉnh lần lượt là 1, 2, 3, 3, 4, 5.

(iii) bậc các đỉnh lần lượt là 2, 2, 4, 4, 4, 4.

1.6.8. Tìm đơn đồ thị mà mọi đỉnh của nó đều có bậc là 3 và có:

(i) 4 đỉnh

(ii) 5 đỉnh

(iii) 6 đỉnh

(iv) 8 đỉnh

1.6.9. Tìm số đỉnh của đồ thị G , biết rằng G có:

(i) 12 cạnh và bậc của tất cả các đỉnh = 2.

(ii) 15 cạnh, trong đó có 3 đỉnh bậc 4 và các đỉnh còn lại bậc 3.

(iii) 6 cạnh và mọi đỉnh đều có bậc bằng nhau.

1.6.10. Có tồn tại đồ thị đơn với 5 đỉnh với số bậc được cho sau đây hay không? Nếu có hãy vẽ đồ thị:

(i) 3, 3, 3, 3, 2

(ii) 1, 2, 3, 4, 4

(iii) 0, 1, 2, 2, 3

(iv) 1, 2, 3, 4, 5

(v) 3, 4, 3, 4, 3

1.6.11. Một đồ thị có 19 cạnh và mỗi đỉnh đều có bậc ≥ 3 . Đồ thị này có tối đa bao nhiêu đỉnh?

1.6.12. Có thể tồn tại đồ thị đơn có 15 đỉnh, mỗi đỉnh có bậc bằng 5 hay không?

1.6.13. Có thể có 1 nhóm gồm 9 người, trong đó mỗi người đều chỉ quen biết đúng 5 người?

1.6.14. Trong một giải thi đấu có n đội tham dự và $n+1$ trận đấu đã được tiến hành. Chứng minh rằng có 1 đội đã thi đấu ít nhất 3 trận.

1.6.15. Một cuộc họp có ít nhất ba đại biểu đến dự. Mỗi người quen ít nhất hai đại biểu khác. Chứng minh rằng có thể xếp được một số đại biểu ngồi xung quanh một bàn tròn, để mỗi người ngồi giữa hai người mà đại biểu đó quen.

1.6.16. Một lớp học có ít nhất 4 sinh viên. Mỗi sinh viên thân với ít nhất 3 sinh viên khác. Chứng minh rằng có thể xếp một số chỗ sinh viên ngồi quanh một cái bàn tròn để mỗi sinh viên ngồi giữa hai sinh viên mà họ thân.

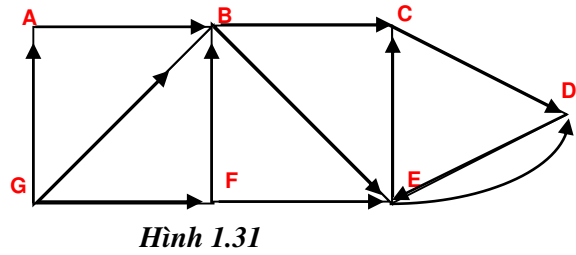
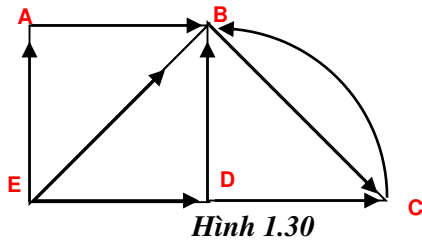
- 1.6.17.** Trong một cuộc họp có đúng hai đại biểu không quen nhau và mỗi đại biểu này có một số lẻ người quen đến dự. Chứng minh rằng luôn luôn có thể xếp một số đại biểu ngồi chen giữa hai đại biểu nói trên, để mỗi người ngồi giữa hai người mà anh ta quen.
- 1.6.18.** Một thành phố có n ($n \geq 2$) nút giao thông và hai nút giao thông bất kỳ đều có số đầu mỗi đường ngầm tới một trong các nút giao thông này đều không nhỏ hơn n . Chứng minh rằng từ một nút giao thông tùy ý ta có thể đi đến một nút giao thông bất kỳ khác bằng đường ngầm.
- 1.6.19.** Tìm các đồ thị con và đồ thị riêng trong các hình từ 1.29 đến 1.36.
- 1.6.20.** Cho $V = \{2, 3, 4, 5, 6, 7, 8\}$ và E là tập hợp các cặp phần tử (u, v) của V sao cho $u < v$ và u, v nguyên tố cùng nhau. Hãy vẽ đồ thị có hướng $G = (V, E)$. Tìm số các đường đi phân biệt độ dài 3 từ đỉnh 2 tới đỉnh 8.
- 1.6.21.** Hãy tìm số đường đi độ dài n giữa hai đỉnh liên kề (tương ứng không liên kề) tùy ý trong $K_{3,3}$ với mỗi giá trị của n sau:
- a) $n=2$ b) $n=3$ c) $n=4$ d) $n=5$.
- 1.6.22.** Vẽ các đồ thị:
- (i) K_7 (ii) $K_{4,4}$ (iii) W_7
(iv) $K_{1,8}$ (v) C_7 (vi) Q_4
- 1.6.23.** Tìm số đỉnh của một đồ thị vô hướng G trong các trường hợp sau:
- Nếu biết G có 12 cạnh và mọi đỉnh đều có bậc 2.
 - Nếu biết G có 15 cạnh, 3 đỉnh bậc 4 và các đỉnh còn lại đều có bậc 3.
 - Nếu biết G có 6 cạnh và mọi đỉnh có bậc bằng nhau.
- 1.6.24.** Biết rằng mọi đỉnh của G đều có bậc bằng số lẻ p . Chứng minh số cạnh của G là một bội số của p .
- 1.6.25.** Cho một đơn đồ thị G có số đỉnh $n \geq 3$. Chứng minh G có chứa ít nhất 2 đỉnh cùng bậc.
- 1.6.26.** Cho đồ thị vô hướng có đúng 2 đỉnh bậc lẻ. Chứng minh có một đường đi giữa 2 đỉnh này.
- 1.6.27.** Chứng minh rằng đơn đồ thị n đỉnh là liên thông nếu có nhiều hơn $(n-1)(n-2)/2$ cạnh.
- 1.6.28.** Giả sử G là đơn đồ thị vô hướng có n đỉnh ($n \geq 3$) và $\deg(x) \geq n/2$ với mọi đỉnh x của G . Chứng minh rằng G liên thông.
- 1.6.29.** Giả sử G là đơn đồ thị vô hướng n đỉnh, $n \geq 3$. Chứng minh rằng nếu $\deg(x) \geq (n-1)/2$ với mọi đỉnh x của G thì đồ thị liên thông.
- 1.6.30.** Chỉ ra rằng nếu đơn đồ thị vô hướng có k thành phần liên thông với số đỉnh tương ứng là n_1, \dots, n_k , thì tổng số cạnh của G không vượt quá $\sum_{i=1}^k C^2 n_i$.
- 1.6.31.** Gọi G là đồ thị có n đỉnh và m cạnh, gọi D và d tương ứng là bậc lớn nhất và nhỏ nhất của các đỉnh của G . Chứng minh rằng: $d \leq 2m/n \leq D$.
- 1.6.32.** Chứng minh rằng nếu G là đơn đồ thị hai phía có n đỉnh và m cạnh thì $m \leq n^2/4$.
- 1.6.33.** Cho G là đồ thị vô hướng liền thưng, m cạnh, n đỉnh. Chứng minh $m \geq n-1$.

1.6.34. Cho một đồ thị có 19 cạnh và mỗi đỉnh có bậc lớn hơn hoặc bằng 3. Đồ thị này có tối đa bao nhiêu đỉnh.

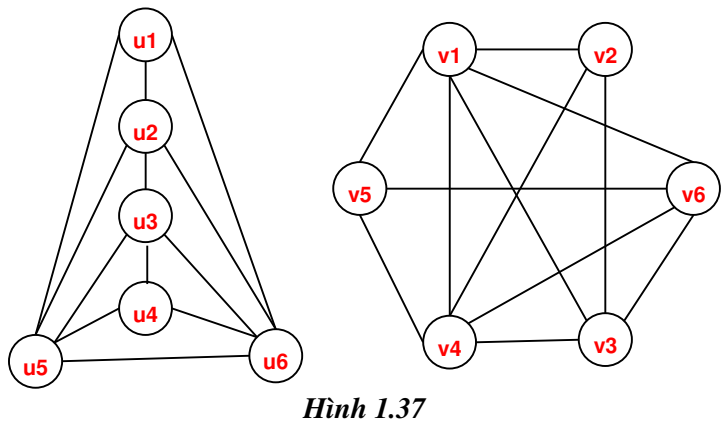
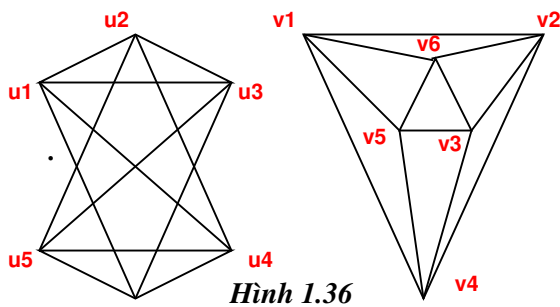
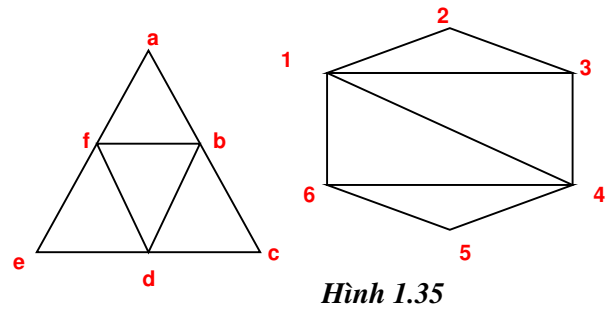
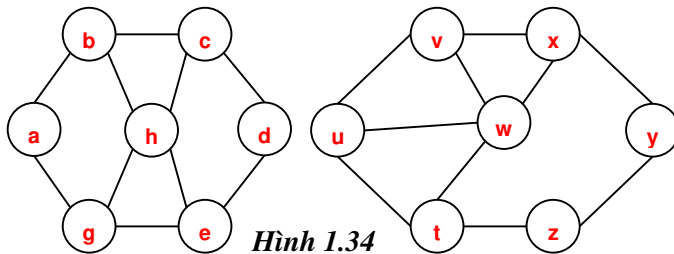
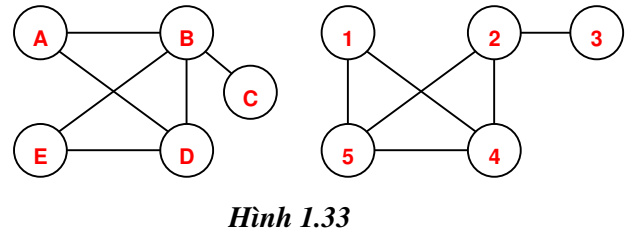
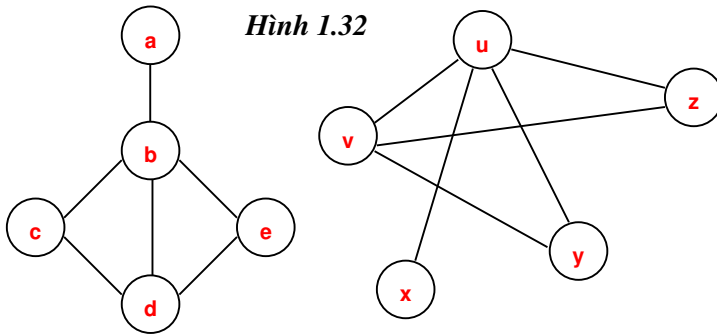
1.6.35. Chứng minh rằng trong một đơn đồ thị luôn luôn tồn tại đường đi từ một đỉnh bậc lẻ đến một đỉnh bậc lẻ khác.

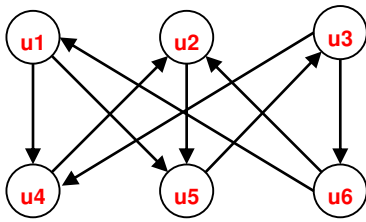
1.6.36. Trong các đồ thị sau, xác định:

- Bậc vào/bậc ra (đồ thị có hướng).
- Đồ thị liên thông mạnh hay yếu?

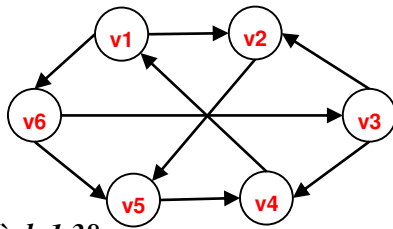


1.6.37. Tìm các cặp đồ thị đẳng hình với nhau trong từng hình sau (giải thích):

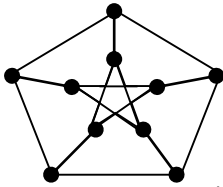




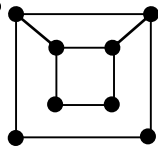
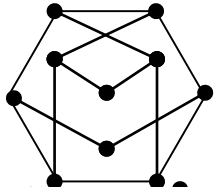
Hình 1.38



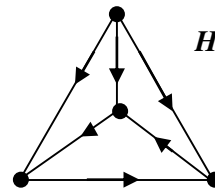
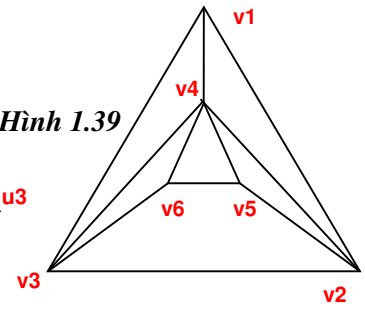
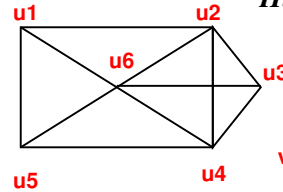
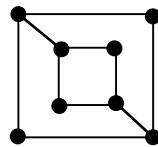
Hình 1.39



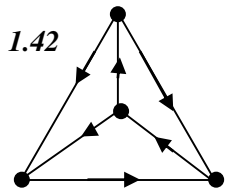
Hình 1.40



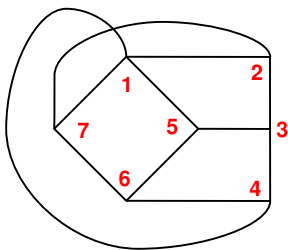
Hình 1.41



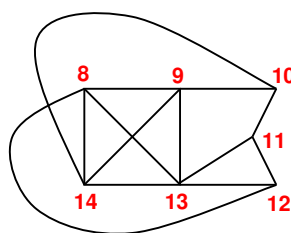
Hình 1.42



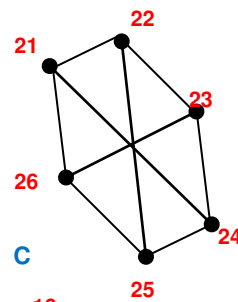
1.6.38. Tìm các cặp đồ thị đẳng hình với nhau trong hình 1.43 (giải thích):



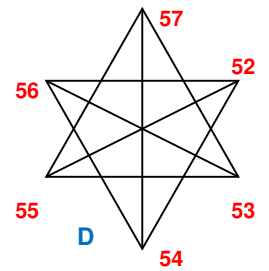
A



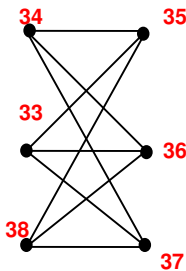
B



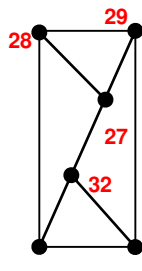
C



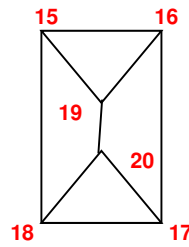
D



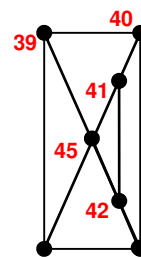
E



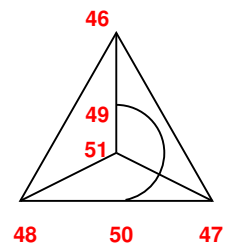
F



G



H



I

Hình 1.43

1.7. CÂU HỎI ÔN TẬP

1. Đồ thị là gì? Đồ thị bao gồm những thành phần nào? Có những loại đồ thị nào?

BIỂU DIỄN ĐỒ THỊ TRÊN MÁY TÍNH & CÁC THUẬT TOÁN TÌM KIẾM TRÊN ĐỒ THỊ

2.1. BIỂU DIỄN ĐỒ THỊ TRÊN MÁY 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 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.

2.1.1. Ma trận kề - Ma trận trọng số

2.1.1.1. Ma trận kề của đơn đồ thị

- 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:

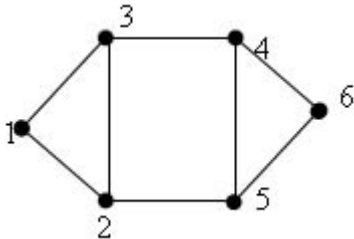
$$A = \{a_{ij} : i, j = 1, 2, \dots, n\}$$

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

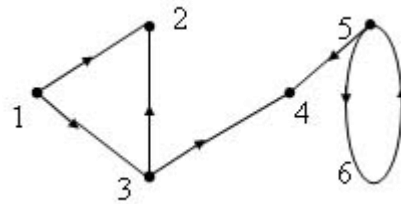
$$a_{ij} = 0, \text{ nếu } (i, j) \notin E \text{ và}$$

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

- Ví dụ:** Cho đồ thị vô hướng G (hình 2.1) và đồ thị có hướng H (hình 2.2).



Hình 2.1: đồ thị vô hướng G



Hình 2.2: đồ thị có hướng H

- Ma trận kề của 2 đồ thị trên là:

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

Ma trận kề của đồ thị G

	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

Ma trận kề của đồ thị H

- Các tính chất của ma trận kề:

[i]. Ma trận kề của đồ thị vô hướng là ma trận đối xứng, tức là

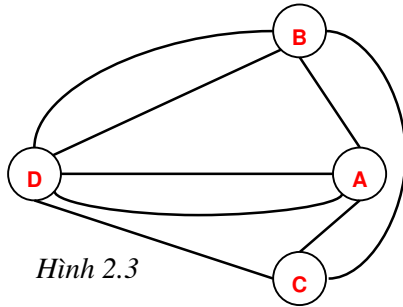
$$a_{ij} = a_{ji}; i, j = 1, 2, \dots, n.$$

[ii]. Tổng các phần tử trên dòng i của ma trận kề chính bằng bậc của đỉnh i . Tương tự, tổng các phần tử trên cột j của ma trận kề chính bằng bậc của đỉnh j .

2.1.1.2. Ma trận kề của đa đồ thị

Ma trận kề của đa đồ thị có thể xây dựng hoàn toàn tương tự đơn đồ thị, 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 .

Ví dụ: với đồ thị trong hình 2.3 được biểu diễn như sau:



Hình 2.3

	A	B	C	D
A	0	1	1	2
B	1	0	1	2
C	1	1	0	0
D	2	2	0	0

2.1.1.3. Ma trận kề của đồ thị có trọng số

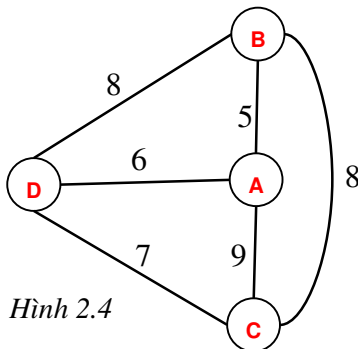
Trong rất nhiều vấn đề ứng dụng của lý thuyết đồ thị, mỗi cạnh $e=(u,v)$ của đồ thị được gán với một con số $c(e)$ (còn viết là $c(u,v)$) gọi là trọng số của cạnh e . Đồ thị trong trường hợp như vậy được gọi là đồ thị có trọng số. Trong trường hợp đồ thị có trọng số, để biểu diễn đồ thị ta sử dụng ma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

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

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

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

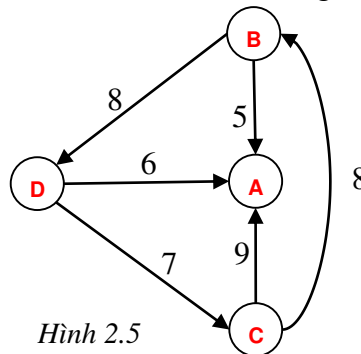
trong đó tùy từng trường hợp cụ thể, số θ có thể được đặt là một trong các giá trị sau: $0, +\infty, -\infty$.



Hình 2.4

	A	B	C	D
A	0	5	9	6
B	5	0	8	8
C	9	8	0	0
D	6	8	0	0

Ma trận biểu diễn đồ thị hình 2.4



Hình 2.5

	A	B	C	D
A	0	0	0	0
B	5	0	0	8
C	9	0	8	0
D	6	0	7	0

Ma trận biểu diễn đồ thị hình 2.5

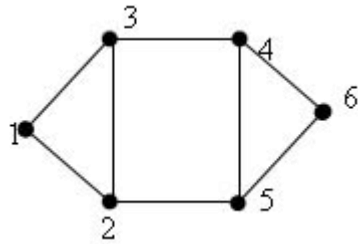
2.1.1.4. Ưu, nhược điểm của phương pháp dùng ma trận để biểu diễn đồ thị

- **Ưu điểm** (lớn nhất) giúp xác định nhanh xem hai đỉnh u,v có kề nhau trên đồ thị hay không, bằng cách thực hiện một phép so sánh giá trị tại ô (i, j) với giá trị quy định trước (như $0, +\infty$ hay $-\infty$).
- **Nhược điểm**: ta luôn phải sử dụng n^2 đơn vị bộ nhớ (n là số lượng đỉnh của đồ thị) để lưu trữ ma trận kề của nó.

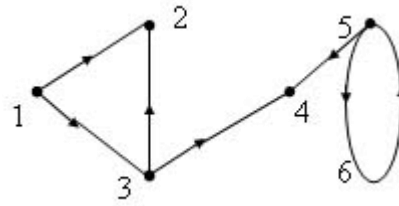
2.1.2. Danh sách cạnh (cung)

Trong trường hợp đồ thị thưa (đồ thị có số cạnh m thỏa mãn bất đẳng thức: $m < 6 \cdot n$) 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 $2 \cdot m$ 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).



Hình 2.6: đồ thị vô hướng G



Hình 2.7: đồ thị có hướng H

Ví dụ 1: Danh sách cạnh (cung) của đồ thị G (G_1) cho trong hình 1 là:

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

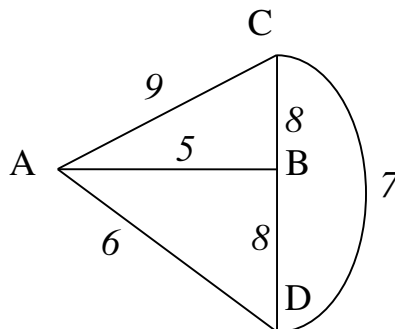
Danh sách cạnh của đồ thị vô hướng G

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

Danh sách cạnh của đồ thị có hướng H

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.

Ví dụ 2: biểu diễn đồ thị trong hình 2.8 bằng danh sách cạnh



Hình 2.8 Đồ thị có trọng số

Dau	Cuoi	Trọng số
A	B	5
A	C	9
A	D	6
B	C	8
B	D	8
C	D	7

Danh sách cạnh của đồ thị vô hướng G

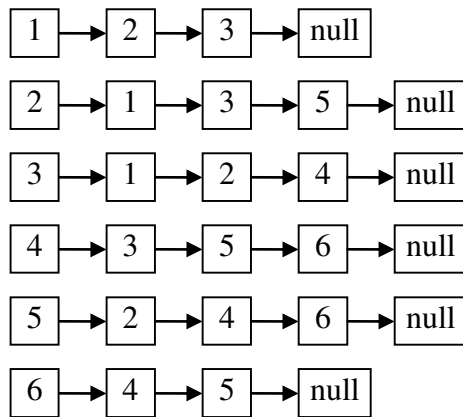
2.1.3. Danh sách kề

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.

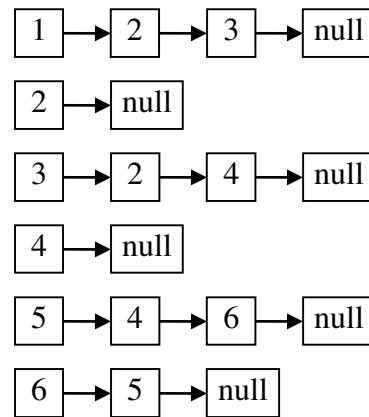
Trong cách biểu diễn này, với mỗi đỉnh v của đồ thị chúng ta lưu trữ danh sách các đỉnh kề với nó, mà ta sẽ ký hiệu là

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

Ví dụ: Danh sách kề của các đồ thị trong hình 1 được mô tả trong hình sau:



Hình 2.9: biểu diễn đồ thị G trong hình 2.1 bằng danh sách kề



Hình 2.10: biểu diễn đồ thị H trong hình 2.2 bằng danh sách kề

Trong cách biểu diễn này chúng ta cần phải sử dụng cỡ $m+n$ đơn vị bộ nhớ.

Trong các thuật toán mô tả ở các phần tiếp theo hai cấu trúc danh sách kề và ma trận trọng số được sử dụng thường xuyên.

2.2. TÌM KIẾM TRÊN ĐỒ THỊ

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. Trong phần này chúng ta sẽ tìm hiểu hai thuật toán tìm kiếm cơ bản trên đồ thị: *Thuật toán tìm kiếm theo chiều sâu (Depth First Search - DFS)* và *Thuật toán tìm kiếm theo chiều rộng (Breadth First Search - BFS)* và ứng dụng của chúng vào việc giải một số bài toán trên đồ thị.

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

2.2.1. Tìm kiếm theo chiều sâu trên đồ thị (Depth First Search)

2.2.1.1. Ý tưởng chính của thuật toán

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 . 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).

void DFS(Dinh v)

```

{ /*tìm kiếm theo chiều sâu bắt đầu từ đỉnh v;
                                     mảng Chuaxet là biến toàn cục*/

    Tham_dinh(v);
    Chuaxet[v] = false;
    for ( u ∈ Ke(v) )
        if (ChuaXet[u] )
            DFS(u);
} // đỉnh v đã duyệt xong
    
```

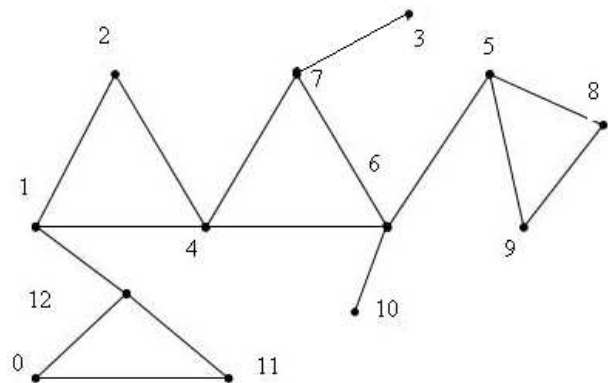
Khi đó, tìm kiếm theo chiều sâu trên đồ thị được thực hiện nhờ thuật toán sau:

```
void main()
{
    // Khởi tạo các đỉnh, cạnh của đồ thị
    for (v ∈ V) Chuaxet[v]=true;
    for (v ∈ V)
        if (Chuaxet[v])
            DFS(v);
}
```

Rõ ràng lệnh gọi DFS(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).

2.2.1.2. Ví dụ

Xét đồ thị cho trong hình 2.8 gồm 13 đỉnh, các đỉnh được đánh số từ 0 đến 12 như sau:



Hình 2.11

- Sử dụng 2 mảng 1 chiều:
 - Mảng *bool* *ChuaXet*[14] cho biết tên đỉnh tương ứng với chỉ số mảng đã xét (=false - F) hay chưa xét (=true - T)
 - Mảng *int* *LuuVet*[14] lưu giá trị chính là tên của đỉnh sẽ đi đến đỉnh tương ứng với chỉ số trong mảng. Thực ra trong DFS không cần dùng mảng này, nhưng ở đây đưa kèm mảng này vào để sử dụng cho phần sau của tài liệu (phần 2.2.3).

– Khởi tạo:

	0	1	2	3	4	5	6	7	8	9	10	11	12
<i>ChuaXet</i>	T	T	T	T	T	T	T	T	T	T	T	T	T
<i>LuuVet</i>	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1

– Quy ước:

- Chỉ duyệt qua mỗi đỉnh duy nhất 1 lần.
- Khi từ đỉnh đang xét có nhiều đường đi đến các đỉnh khác, ta chọn đỉnh có giá trị nhỏ hơn.

– Duyệt theo DFS, với đỉnh xuất phát là đỉnh **1**

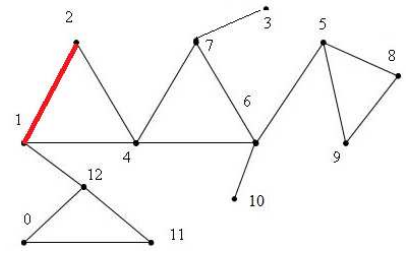
• Thực hiện lần 1:

- Gán *LuuVet*[1]=1 (đỉnh 1 đi đến đỉnh 1)
- Gán *ChuaXet*[1]=F (đã xét)

	0	1	2	3	4	5	6	7	8	9	10	11	12
<i>ChuaXet</i>	T	F	T	T	T	T	T	T	T	T	T	T	T
<i>LuuVet</i>	-1	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1

- Thực hiện lần 2: từ đỉnh 1, có 3 đường nối đến các đỉnh 2, 4, 12. Chọn đỉnh **2**.

- Gán $LuuVet[2]=1$ (đỉnh 1 đi đến đỉnh 2)
- Gán $ChuaXet[2]=F$ (đã xét)

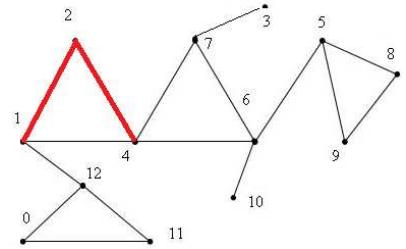


Hình 2.12

	0	1	2	3	4	5	6	7	8	9	10	11	12
<i>ChuaXet</i>	T	F	F	T	T	T	T	T	T	T	T	T	T
<i>LuuVet</i>	-1	1	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1

- Thực hiện lần 3: từ đỉnh 2, chỉ có duy nhất đường đi đến đỉnh 4. Chọn đỉnh **4**.

- Gán $LuuVet[4]=2$ (đỉnh 2 đi đến đỉnh 4)
- Gán $ChuaXet[4]=F$ (đã xét)

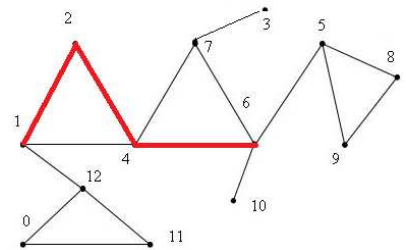


Hình 2.13

	0	1	2	3	4	5	6	7	8	9	10	11	12
<i>ChuaXet</i>	T	F	F	T	F	T	T	T	T	T	T	T	T
<i>LuuVet</i>	-1	1	1	-1	2	-1	-1	-1	-1	-1	-1	-1	-1

- Thực hiện lần 4: từ đỉnh 4, có 2 đường nối đến các đỉnh 6, 7. Chọn đỉnh **6**.

- Gán $LuuVet[6]=4$ (đỉnh 4 đi đến đỉnh 6)
- Gán $ChuaXet[6]=F$ (đã xét)

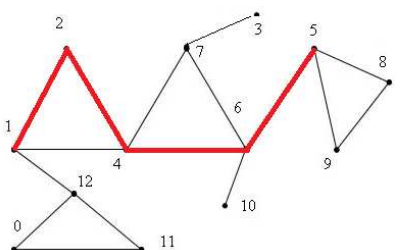


Hình 2.14

	0	1	2	3	4	5	6	7	8	9	10	11	12
<i>ChuaXet</i>	T	F	F	T	F	T	F	T	T	T	T	T	T
<i>LuuVet</i>	-1	1	1	-1	2	-1	4	-1	-1	-1	-1	-1	-1

- Thực hiện lần 5: từ đỉnh 6, có 3 đường nối đến các đỉnh 5, 7, 13. Chọn đỉnh **5**.

- Gán $LuuVet[5]=6$ (đỉnh 6 đi đến đỉnh 5)
- Gán $ChuaXet[5]=F$ (đã xét)

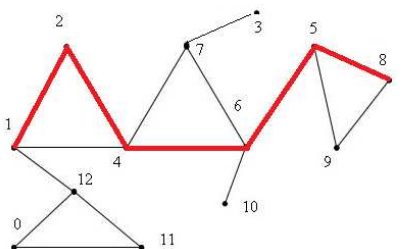


Hình 2.15

	0	1	2	3	4	5	6	7	8	9	10	11	12
<i>ChuaXet</i>	T	F	F	T	F	F	F	T	T	T	T	T	T
<i>LuuVet</i>	-1	1	1	-1	2	6	4	-1	-1	-1	-1	-1	-1

- Thực hiện lần 6: từ đỉnh 5, có 2 đường nối đến các đỉnh 8, 9. Chọn đỉnh **8**.

- Gán $LuuVet[8]=5$ (đỉnh 5 đi đến đỉnh 8)
- Gán $ChuaXet[8]=F$ (đã xét)



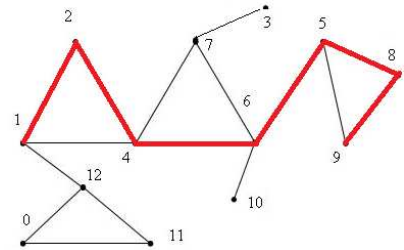
Hình 2.16

	0	1	2	3	4	5	6	7	8	9	10	11	12
ChuaXet	T	F	F	T	F	F	F	T	F	T	T	T	T
LuuVet	-1	1	1	-1	2	6	4	-1	5	-1	-1	-1	-1

- Thực hiện lần 7: từ đỉnh 8, chỉ có đường nối đến đỉnh 9.

Chọn đỉnh 9.

- Gán $LuuVet[9]=8$ (đỉnh 8 đi đến đỉnh 9)
- Gán $ChuaXet[9]=F$ (đã xét)

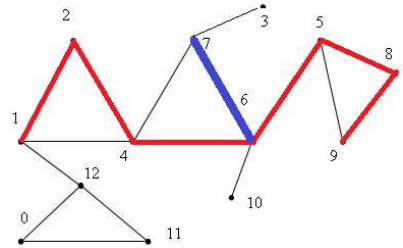


Hình 2.17

	0	1	2	3	4	5	6	7	8	9	10	11	12
ChuaXet	T	F	F	T	F	F	F	T	F	F	T	T	T
LuuVet	-1	1	1	-1	2	6	4	-1	5	8	-1	-1	-1

- Thực hiện lần 8: từ đỉnh 9, không còn đường đi đến các đỉnh chưa xét. Thực hiện quay lui lần lượt về các đỉnh liền trước đó và sẽ dừng lại khi đến đỉnh nào có đường đi tiếp đến các đỉnh chưa được xét. Dựa vào đó, ta lần lượt về các đỉnh 8, 5, 6. Khi quay lui về đến đỉnh 6, có 2 đường đi đến các đỉnh 7 và 10. Chọn đỉnh 7.

- Gán $LuuVet[7]=6$ (đỉnh 6 đi đến đỉnh 7)
- Gán $ChuaXet[7]=F$ (đã xét)



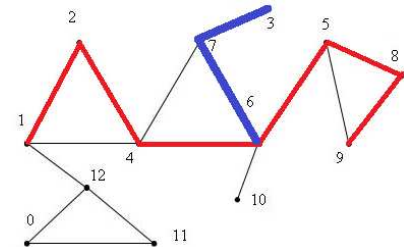
Hình 2.18

	0	1	2	3	4	5	6	7	8	9	10	11	12
ChuaXet	T	F	F	T	F	F	F	F	F	F	T	T	T
LuuVet	-1	1	1	-1	2	6	4	6	5	8	-1	-1	-1

- Thực hiện lần 8: từ đỉnh 7, chỉ có đường nối đến đỉnh 3.

Chọn đỉnh 3.

- Gán $LuuVet[3]=7$ (đỉnh 7 đi đến đỉnh 3)
- Gán $ChuaXet[3]=F$ (đã xét)

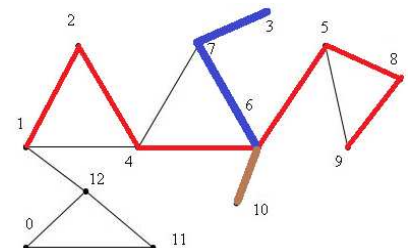


Hình 2.19

	0	1	2	3	4	5	6	7	8	9	10	11	12
ChuaXet	T	F	F	F	F	F	F	F	F	F	T	T	T
LuuVet	-1	1	1	7	2	6	4	6	5	8	-1	-1	-1

- Thực hiện lần 9: từ đỉnh 3, không còn đường đi đến các đỉnh chưa xét. Thực hiện quay lui lần lượt về các đỉnh liền trước đó (tương tự như khi thực hiện lần 7). Khi quay lui về đến đỉnh 6, có duy nhất một đường đi đến đỉnh 10. Chọn đỉnh 10.

- Gán $LuuVet[10]=6$ (đỉnh 6 đi đến đỉnh 10)
- Gán $ChuaXet[10]=F$ (đã xét)

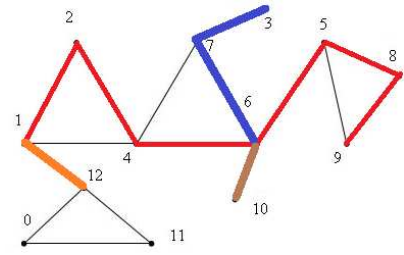


Hình 2.20

	0	1	2	3	4	5	6	7	8	9	10	11	12
ChuaXet	T	F	F	F	F	F	F	F	F	F	F	T	T
LuuVet	-1	1	1	7	2	6	4	6	5	8	6	-1	-1

- **Thực hiện lần 10:** từ đỉnh 10, không còn đường đi đến các đỉnh chưa xét. Thực hiện quay lui lần lượt về các đỉnh liền trước đó (tương tự như khi thực hiện lần 7 và lần 9). Khi quay lui về đến đỉnh 1, có duy nhất một đường đi đến đỉnh 12. Chọn đỉnh 12.

- Gán $LuuVet[12]=1$ (đỉnh 1 đi đến đỉnh 12)
- Gán $ChuaXet[12]=F$ (đã xét)

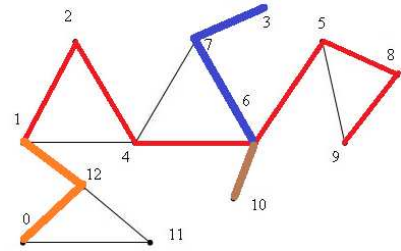


Hình 2.21

	0	1	2	3	4	5	6	7	8	9	10	11	12
<i>ChuaXet</i>	T	F	F	F	F	F	F	F	F	F	F	T	F
<i>LuuVet</i>	-1	1	1	7	2	6	4	6	5	8	6	-1	1

- **Thực hiện lần 11:** từ đỉnh 12, có hai đường nối đến các đỉnh 0 và 11. Chọn đỉnh 0.

- Gán $LuuVet[0]=12$ (đỉnh 12 đi đến đỉnh 0)
- Gán $ChuaXet[0]=F$ (đã xét)

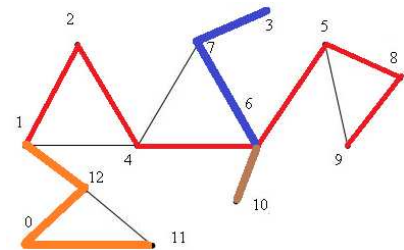


Hình 2.22

	0	1	2	3	4	5	6	7	8	9	10	11	12
<i>ChuaXet</i>	F	F	F	F	F	F	F	F	F	F	F	T	F
<i>LuuVet</i>	12	1	1	7	2	6	4	6	5	8	12	-1	1

- **Thực hiện lần 12:** từ đỉnh 10, có đường nối đến đỉnh 11. Chọn đỉnh 11.

- Gán $LuuVet[11]=10$ (đỉnh 10 đi đến đỉnh 11)
- Gán $ChuaXet[11]=F$ (đã xét)



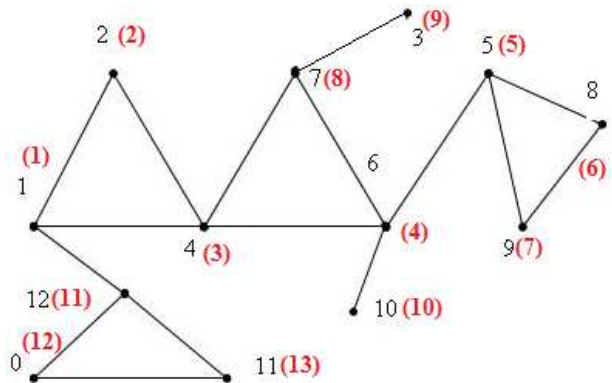
Hình 2.23

	0	1	2	3	4	5	6	7	8	9	10	11	12
<i>ChuaXet</i>	F	F	F	F	F	F	F	F	F	F	F	F	F
<i>LuuVet</i>	12	1	1	7	2	6	4	6	5	8	12	10	1

Thuật toán kết thúc vì tất cả các đỉnh đều đã được xét.

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 2.24. 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.

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.



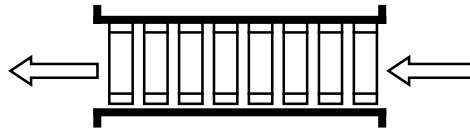
Hình 2.24. Chỉ số mới (trong ngoặc) của các đỉnh được đánh lại theo thứ tự chúng được thăm trong thuật toán tìm kiếm theo chiều sâu.

2.2.2. Tìm kiếm theo chiều rộng trên đồ thị (Breadth First Search)

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.

2.2.2.1. Cấu trúc dữ liệu Queue (hàng đợi)

- Hàng đợi là một cấu trúc gồm nhiều phần tử có thứ tự
- Hoạt động theo cơ chế “Vào trước, ra trước” (FIFO - First In First Out)



- Các thao tác cơ bản (hàm cần xây dựng) trên Queue:
 - *InitQueue* : khởi tạo Queue rỗng
 - *IsQueueEmpty*: kiểm tra Queue rỗng?
 - *IsQueueFull* : kiểm tra Queue đầy?
 - *EnQueue* : thêm 1 phần tử vào cuối Queue (\Rightarrow có thể làm Queue đầy)
 - *DeQueue* : lấy ra 1 phần tử từ đầu Queue (\Rightarrow có thể làm Queue rỗng). Sau khi lấy ra 1 phần tử sẽ thực hiện dồn mảng lại về đầu.
- Cấu trúc đề nghị dùng cho Queue:
 - *int array[]* : là mảng 1 chiều được sử dụng làm Queue.
 - *int Count* : đếm số lượng phần tử đang có trong Queue.

2.2.2.2. Xây dựng cấu trúc dữ liệu

- Cấu trúc lưu trữ đồ thị: bổ sung thêm 1 mảng vào cấu trúc đã giới thiệu ban đầu để ghi nhận thứ tự đỉnh được duyệt (tính từ 1) trong thuật toán BFS.

```
typedef struct GRAPH
{
    int n; // số đỉnh thực tế của đồ thị. Với MAX>=n
    int A[MAX][MAX]; // ma trận kề của đồ thị
    int ThuTuDuyet[MAX]; /* mảng ghi nhận thứ tự đỉnh sẽ được duyệt*/
}DOTHI;
```

- Sử dụng mảng 1 chiều (*int queue[]*) để quản lý Queue.
- Vẫn sử dụng mảng 1 chiều (*bool ChuaXet[]*) để ký hiệu đỉnh của đồ thị đã được đánh dấu hay chưa. Chỉ số của mảng chính là chỉ số đỉnh của đồ thị. Đầu chương trình cần khởi tạo tất cả các phần tử trong mảng *ChuaXet* là true.
- Tổ chức một mảng 1 chiều (*bool LuuVet[]*):
 - Chỉ số của mảng chính là chỉ số đỉnh của đồ thị.
 - Đầu chương trình cần khởi tạo tất cả các phần tử trong mảng *LuuVet* là -1 (chưa tìm được đường đi) \Rightarrow Sau khi thuật toán chạy hoàn tất nếu giá trị *LuuVet[DinhDangXet]=-1* tức là không có đường đi đến đỉnh *DinhDangXet*, hay nói cách khác, đồ thị có nhiều hơn 1 thành phần liên thông.
 - Giá trị lưu trong mảng cho biết tên của đỉnh sẽ đi tới đỉnh có tên theo chỉ số.

Ví dụ: mảng `LuuVet` sau khi duyệt trên đồ thị DFS1, giả sử đỉnh xuất phát là D, đỉnh cần đến là H:

Quy ước đỉnh A tương ứng với đỉnh 0, đỉnh B tương ứng với đỉnh 1, ...

Tên đỉnh tương ứng	A	B	C	D	E	F	G	H
Chỉ số mảng	0	1	2	3	4	5	6	7
Tên đỉnh liên trước	2	0	3	-1	1	3	5	6

Cách đọc: để đi đến 7 sẽ phải qua đỉnh 6 (=LuuVet[7])

để đi đến 6 sẽ phải qua đỉnh 5 (=LuuVet[6])

để đi đến 5 sẽ phải qua đỉnh 3 (=LuuVet[5]= đỉnh xuất phát)

2.2.2.3. Mô tả hàm BFS

void BFS (Đỉnh v)

```
{
    /*Tim kiem theo chieu rong bat dau tu dinh v,
                                   cac bien Chuaxet, Ke la bien cuc bo*/

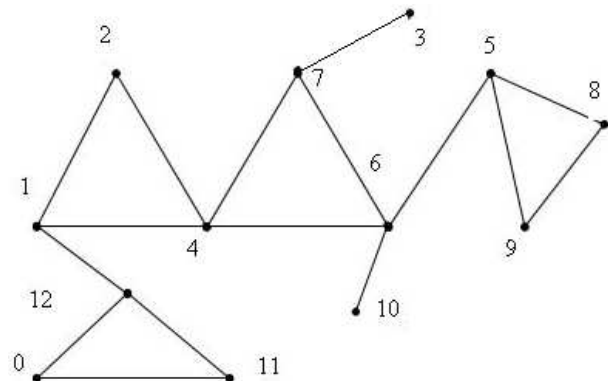
    QUEUE=Ø ;
    QUEUE ← v; //ket qua nap vao QUEUE
    Chuaxet[v]=false;
    while (QUEUE<>Ø )
    {
        p← QUEUE; //lay p tu QUEUE
        Tham_dinh(p);
        for (u∈ Ke(v) )
            if (Chuaxet[u])
            {
                QUEUE← u;
                Chuaxet[u]=false;
            }
    }
}
```

Khi đó, hàm tìm kiếm theo chiều rộng trên đồ thị (BFS) được thực hiện nhờ lời gọi hàm như sau:

```
void main()
{
    // Khởi tạo các đỉnh, cạnh của đồ thị
    for (v ∈ V)
        Chuaxet[v] = true;
    for ( v∈ V)
        if (Chuaxet[v] )
            BFS(v);
}
```

2.2.2.4. Ví dụ

Xét đồ thị cho trong hình 2.8 gồm 13 đỉnh, các đỉnh được đánh số từ 0 đến 12 như sau:



Hình 2.25

- Mảng **int HangDoi[14]** đóng vai trò 1 hàng đợi (Queue), chứa danh sách các đỉnh đang được xét. Cấu trúc đề nghị dùng cho Queue:
 - int array[]**: là mảng 1 chiều được sử dụng làm Queue.
 - int count**: đếm số lượng phần tử đang có trong Queue.
- Sử dụng 2 mảng 1 chiều:
 - Mảng **bool ChuaXet[14]** cho biết tên đỉnh tương ứng với chỉ số mảng đã xét (=false - F) hay chưa xét (=true - T)
 - Mảng **int LuuVet[14]** lưu giá trị chính là tên của đỉnh sẽ đi đến đỉnh tương ứng với chỉ số trong mảng. Thực ra trong BFS không cần dùng mảng này, nhưng ở đây đưa kèm mảng này vào để tiện sử dụng cho phần sau của tài liệu (phần 2.2.3).
- Quy ước:**
 - Chỉ duyệt qua mỗi đỉnh duy nhất 1 lần.
 - Khi từ đỉnh đang xét có nhiều đường đi đến các đỉnh khác, danh sách các đỉnh này khi đưa vào Queue sẽ được sắp xếp theo thứ tự từ nhỏ đến lớn.
- Duyệt theo BFS, với đỉnh xuất phát là đỉnh **1**

• **Khởi tạo:**

	0	1	2	3	4	5	6	7	8	9	10	11	12
ChuaXet	T	T	T	T	T	T	T	T	T	T	T	T	T
LuuVet	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1

queue.array													
-------------	--	--	--	--	--	--	--	--	--	--	--	--	--

count=0

• **Thực hiện lần 1:** bỏ tên đỉnh xuất phát vào Queue

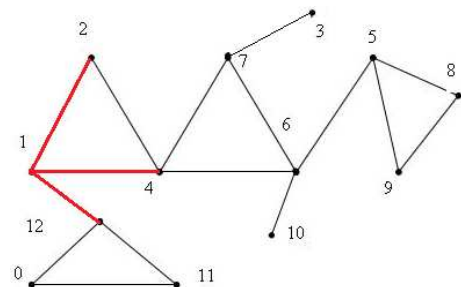
- Gán LuuVet[1]=1 (đỉnh 1 đi đến đỉnh 1)
- Gán ChuaXet[1]=F (đã xét)

	0	1	2	3	4	5	6	7	8	9	10	11	12
ChuaXet	T	F	T	T	T	T	T	T	T	T	T	T	T
LuuVet	-1	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1

queue.array	1												
-------------	---	--	--	--	--	--	--	--	--	--	--	--	--

count=1

- Thực hiện lần 2:** lấy ra đỉnh đầu tiên đang có trong Queue (đỉnh 1). Tìm các đỉnh kề (có đường nối trực tiếp) với đỉnh 1, và các đỉnh này chưa được xét: ta được các đỉnh 2, 4, 12.
 - Đưa 3 đỉnh 2, 4, 12 vào Queue.
 - Gán giá trị 1 cho các phần tử LuuVet[2], LuuVet[4], LuuVet[12]., (đỉnh 1 đi đến đỉnh 2, 4, 12)
 - Gán F cho các phần tử ChuaXet[2], ChuaXet[4], ChuaXet[12] (đã xét các đỉnh 2, 4, 12).



Hình 2.26

	0	1	2	3	4	5	6	7	8	9	10	11	12
ChuaXet	T	F	F	T	F	T	T	T	T	T	T	T	F
LuuVet	-1	1	1	-1	1	-1	-1	-1	-1	-1	-1	-1	1

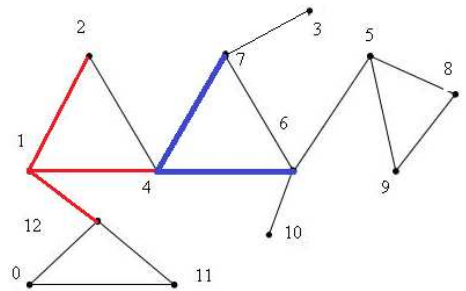
	0	1	2	3	4	5	6	7	8	9	10	11	12
queue.array	2	4	12										
count	=3												

- **Thực hiện lần 3:** lấy ra đỉnh đầu tiên đang có trong Queue (đỉnh 2). Tìm các đỉnh chưa được xét và kề với đỉnh đang xét (đỉnh 2): không tìm thấy đỉnh nào thỏa điều kiện.

	0	1	2	3	4	5	6	7	8	9	10	11	12
ChuaXet	T	F	F	T	F	T	T	T	T	T	T	T	F
LuuVet	-1	1	1	-1	1	-1	-1	-1	-1	-1	-1	-1	1

	0	1	2	3	4	5	6	7	8	9	10	11	12
queue.array	4	12											
count	=2												

- **Thực hiện lần 4:** lấy ra đỉnh đầu tiên đang có trong Queue (đỉnh 4). Tìm các đỉnh chưa được xét và kề với đỉnh 4, ta được các đỉnh 6, 7.
 - Đưa 2 đỉnh 6, 7 vào Queue.
 - Gán giá trị **4** cho các phần tử LuuVet[6], LuuVet[7] (đỉnh 4 đi đến đỉnh 6, 7)
 - Gán F cho các phần tử ChuaXet[6], ChuaXet[7] (đã xét các đỉnh 6, 7).

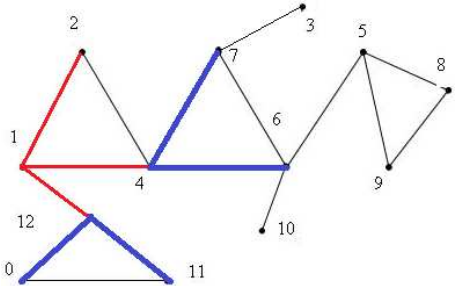


Hình 2.27

	0	1	2	3	4	5	6	7	8	9	10	11	12
ChuaXet	T	F	F	T	F	T	F	F	T	T	T	T	F
LuuVet	-1	1	1	-1	1	-1	4	4	-1	-1	-1	-1	1

	0	1	2	3	4	5	6	7	8	9	10	11	12
queue.array	12	6	7										
count	=3												

- **Thực hiện lần 5:** lấy ra đỉnh đầu tiên đang có trong Queue (đỉnh 12). Tìm các đỉnh chưa được xét và kề với đỉnh 12, ta được các đỉnh 0, 11.
 - Đưa 2 đỉnh 0, 11 vào Queue.
 - Gán giá trị **12** cho các phần tử LuuVet[0], LuuVet[11] (đỉnh 12 đi đến đỉnh 0, 11)
 - Gán F cho các phần tử ChuaXet[0], ChuaXet[11] (đã xét các đỉnh 0, 11).

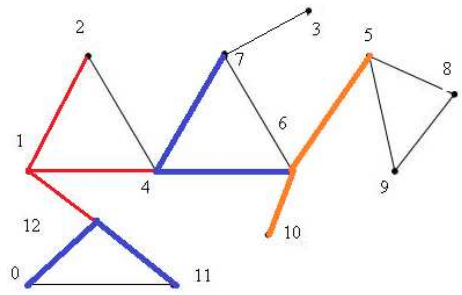


Hình 2.28

	0	1	2	3	4	5	6	7	8	9	10	11	12
ChuaXet	F	F	F	T	F	T	F	F	T	T	T	F	F
LuuVet	12	1	1	-1	1	-1	4	4	-1	-1	-1	12	1

	0	1	2	3	4	5	6	7	8	9	10	11	12
queue.array	6	7	0	11									
count	=4												

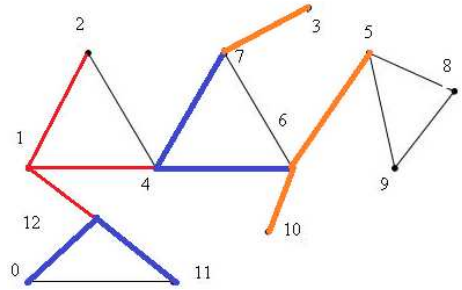
- **Thực hiện lần 6:** lấy ra đỉnh đầu tiên đang có trong Queue (đỉnh 6). Tìm các đỉnh chưa được xét và kề với đỉnh 6, ta được các đỉnh 5, 10.
 - Đưa 2 đỉnh 5, 10 vào Queue.
 - Gán giá trị **6** cho các phần tử $LuuVet[5]$, $LuuVet[10]$ (đỉnh 6 đi đến đỉnh 5, 10)
 - Gán F cho các phần tử $ChuaXet[5]$, $ChuaXet[10]$ (đã xét các đỉnh 5, 10).



Hình 2.29

	0	1	2	3	4	5	6	7	8	9	10	11	12
<i>ChuaXet</i>	F	F	F	T	F	F	F	F	T	T	F	F	F
<i>LuuVet</i>	12	1	1	-1	1	6	4	4	-1	-1	6	12	1
<i>queue.array</i>	7	0	11	5	10								
<i>count=5</i>													

- **Thực hiện lần 7:** lấy ra đỉnh đầu tiên đang có trong Queue (đỉnh 7). Tìm các đỉnh chưa được xét và kề với đỉnh 7, ta được đỉnh 3.
 - Đưa đỉnh 3 vào Queue.
 - Gán $LuuVet[3] = 7$ (đỉnh 7 đi đến đỉnh 3).
 - Gán $ChuaXet[3] = F$ (đã xét các đỉnh 5, 10).



Hình 2.30

	0	1	2	3	4	5	6	7	8	9	10	11	12
<i>ChuaXet</i>	F	F	F	F	F	F	F	F	T	T	F	F	F
<i>LuuVet</i>	12	1	1	7	1	6	4	4	-1	-1	6	12	1
<i>queue.array</i>	0	11	5	10	3								
<i>count=5</i>													

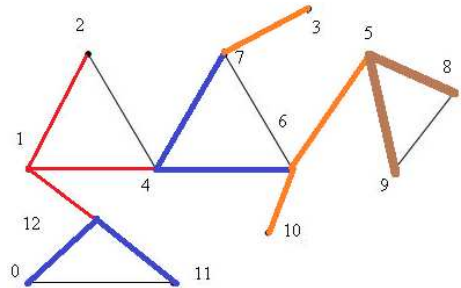
- **Thực hiện lần 8:** lấy ra đỉnh đầu tiên đang có trong Queue (đỉnh 0). Tìm các đỉnh chưa được xét và kề với đỉnh: không tìm thấy đỉnh nào thỏa điều kiện.

	0	1	2	3	4	5	6	7	8	9	10	11	12
<i>ChuaXet</i>	F	F	F	F	F	F	F	F	T	T	F	F	F
<i>LuuVet</i>	12	1	1	7	1	6	4	4	-1	-1	6	12	1
<i>queue.array</i>	11	5	10	3									
<i>count=4</i>													

- **Thực hiện lần 9:** Tương tự, lấy ra đỉnh đầu tiên đang có trong Queue (đỉnh 11). Tìm các đỉnh chưa được xét và kề với đỉnh: không tìm thấy đỉnh nào thỏa điều kiện.

	0	1	2	3	4	5	6	7	8	9	10	11	12
<i>ChuaXet</i>	F	F	F	F	F	F	F	F	T	T	F	F	F
<i>LuuVet</i>	12	1	1	7	1	6	4	4	-1	-1	6	12	1
<i>queue.array</i>	5	10	3										
<i>count=3</i>													

- **Thực hiện lần 10:** lấy ra đỉnh đầu tiên đang có trong Queue (đỉnh 5). Tìm các đỉnh chưa được xét và kề với đỉnh 5, ta được đỉnh 8 và 9.
 - Đưa đỉnh 8 và 9 vào Queue.
 - Gán $LuuVet[8] = 5$ và $LuuVet[9] = 5$ (đỉnh 5 đi đến đỉnh 8 và 9).
 - Gán $ChuaXet[8] = F$ và $ChuaXet[9] = F$ (đã xét các đỉnh 8, 9).



Hình 2.31

	0	1	2	3	4	5	6	7	8	9	10	11	12
<i>ChuaXet</i>	F	F	F	F	F	F	F	F	F	F	F	F	F
<i>LuuVet</i>	12	1	1	7	1	6	4	4	5	5	6	12	1

<i>queue.array</i>	0	1	2	3	4	5	6	7	8	9	10	11	12
	10	3	8	9									

count=4

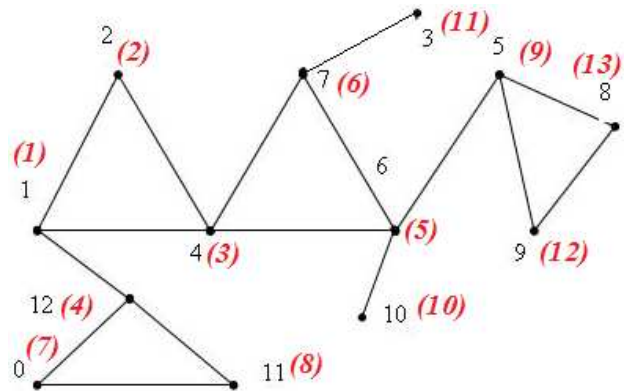
- **Thực hiện lần 11-14:** lần lượt lấy ra đỉnh đầu tiên đang có trong Queue. Dựa vào các đỉnh này ta không tìm thấy thêm được các đỉnh chưa được xét. Sau bước 14, giải thuật BFS kết thúc vì Queue đã trống (không còn đỉnh cần xét).

	0	1	2	3	4	5	6	7	8	9	10	11	12
<i>ChuaXet</i>	F	F	F	F	F	F	F	F	F	F	F	F	F
<i>LuuVet</i>	12	1	1	7	1	6	4	4	5	5	6	12	1

<i>queue.array</i>	0	1	2	3	4	5	6	7	8	9	10	11	12

count=0

Qua ví dụ trên, ta có thể trình bày lại thứ tự thăm đỉnh của đồ thị theo thuật toán tìm kiếm theo chiều rộng (được ghi trong ngoặc).



Hình 2.32: Chỉ số trong ngoặc của các đỉnh được đánh lại theo thứ tự chúng được thăm trong thuật toán tìm kiếm theo chiều sâu.

2.2.3. Ứng dụng DFS/BFS trong kiểm tra tính liên thông và tìm đường đi

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.

2.2.3.1. 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 mảng GanNhan[v] để ghi nhận đỉnh (dựa trên chỉ số của mảng) sẽ thuộc thành phần liên thông thứ mấy của đồ thị (dựa vào giá trị của phần tử mảng, và dùng thêm biến SoTPLT 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]= SoTPLT, còn câu lệnh if trong các chương trình chính gọi đến các thủ tục này cần được sửa lại như sau:

```
SoTPLT:=0;
If ( Chuaxet[v] )
{
    SoTPLT:= SoTPLT+1;
    DFS(v); //BFS(v)
}
```

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

2.2.3.2. 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) và BFS(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 các hàm như sau:

```
if (Chuaxet[u] )
{
    Truoc[u]=v;
    DFS(u);
}
```

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

```
if (Chuaxet [u])
{
    QUEUE<- u;
    Chuaxet[u]=false;
    Truoc[u]=v;
}
```

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.

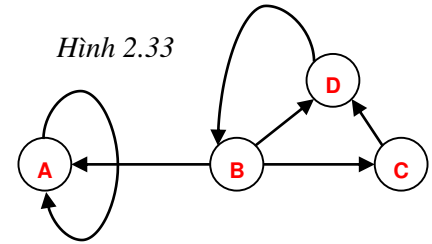
2.3. BÀI TẬP

2.3.1. Vẽ các đồ thị vô hướng được biểu diễn bởi ma trận kề sau:

$\begin{vmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{vmatrix}$	$\begin{vmatrix} 2 & 2 & 3 \\ 2 & 0 & 4 \\ 3 & 4 & 0 \end{vmatrix}$	$\begin{vmatrix} 2 & 2 & 0 & 1 \\ 2 & 0 & 3 & 0 \\ 0 & 3 & 4 & 1 \\ 1 & 0 & 1 & 0 \end{vmatrix}$	$\begin{vmatrix} 0 & 1 & 3 & 0 & 4 \\ 1 & 2 & 1 & 3 & 0 \\ 3 & 1 & 4 & 0 & 1 \\ 0 & 3 & 0 & 0 & 2 \\ 4 & 0 & 1 & 2 & 2 \end{vmatrix}$
a)	b)	c)	d)

2.3.2. Hãy lập ma trận kề của các đồ thị trong chương 1 (từ hình 1.30 đến 1.42).

2.3.3. Lập ma trận kề của đồ thị trong hình 2.11:



2.3.4. Nêu ý nghĩa của tổng các phần tử trên một hàng (tương ứng cột) của một ma trận liên kề đối với một đồ thị vô hướng? Đối với đồ thị có hướng?

2.3.5. Tìm ma trận liên kề cho các đồ thị sau:

- a) K_n b) C_n c) W_n d) $K_{m,n}$ e) Q_n

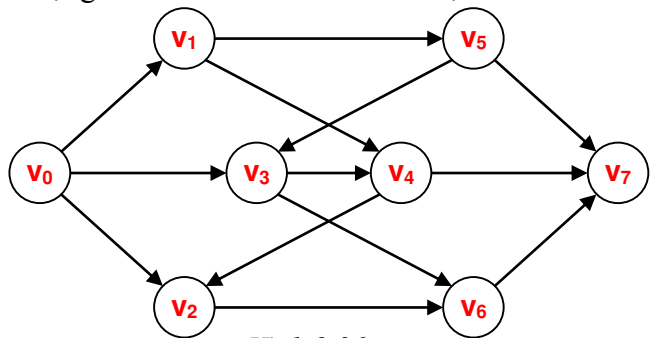
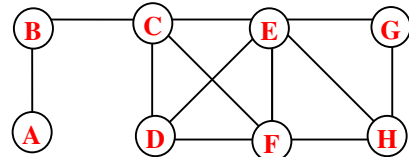
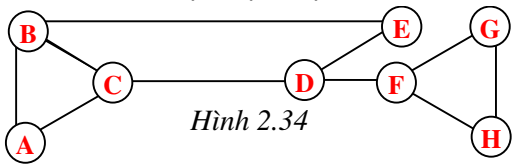
2.3.6. Có bao nhiêu đơn đồ thị không đẳng cấu với n đỉnh khi:

- a) $n=2$, b) $n=3$, c) $n=4$.

2.3.7. Hai đơn đồ thị với ma trận liên kề sau đây có là đẳng hình hay không? Giải thích

$\begin{vmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{vmatrix}$	và	$\begin{vmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{vmatrix}$		$\begin{vmatrix} 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 \end{vmatrix}$	và	$\begin{vmatrix} 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 \end{vmatrix}$
a)				b)		

2.3.8. Lần lượt thực hiện tìm kiếm theo chiều rộng và theo chiều sâu các đồ thị sau



2.4. CÂU HỎI ÔN TẬP

2.4.1. Đồ thị liên thông là gì?

2.4.2. Khi đồ thị không liên thông thì khi đó chắc chắn sẽ tồn tại thành phần liên thông con. Thế thành phần liên thông con là sao?

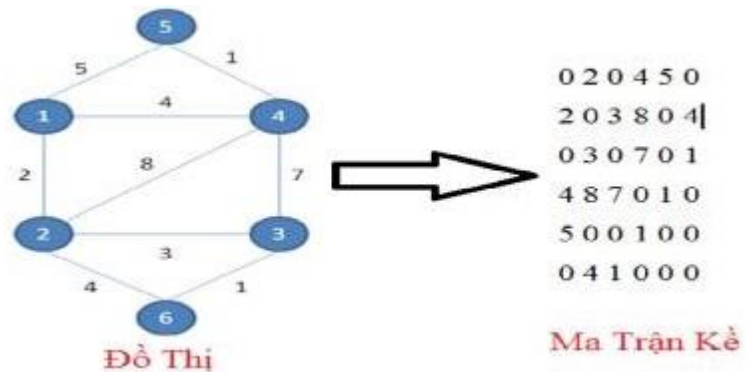
2.4.3. Cho biết các đồ thị sau có gọi là đồ thị liên thông không? Giải thích

2.4.4. Đồ thị chỉ gồm duy nhất một đỉnh và đỉnh này là đỉnh cô lập.

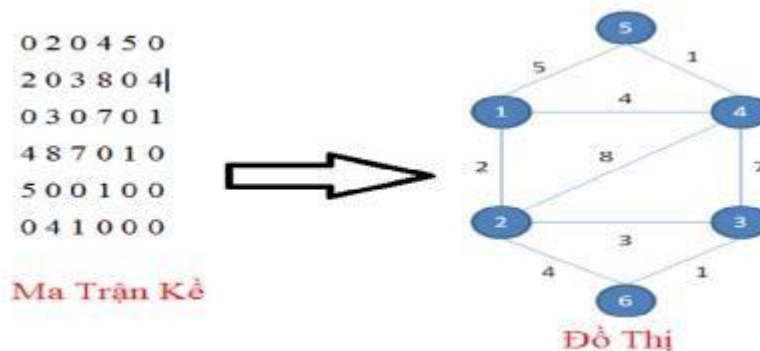
2.4.5. Đồ thị gồm 2 đỉnh và đều là đỉnh cô lập.

2.4.6. Thế nào được gọi là ma trận kề? đường chéo chính của ma trận kề là sao?

2.4.7. Cách biểu diễn/chuyển một đồ thị sang ma trận kề. Vẽ 1 đồ thị bất kỳ rồi thực hiện việc chuyển đổi này. Ví dụ



2.4.8. Cách biểu diễn (chuyển) một ma trận kề sang đồ thị. Cho 1 vài ma trận kề, thực hiện việc chuyển đổi này. Ví dụ:



2.4.9. Nêu đặc điểm của ma trận kề khi biểu diễn đồ thị dạng:

- Có khuyên/vòng?
- Đa đồ thị
- Đơn đồ thị vô hướng?
- Đơn đồ thị có hướng?
- Có đỉnh cô lập?

ĐỒ THỊ EULER & ĐỒ THỊ HAMILTON

3.1. ĐƯỜNG ĐI EULER VÀ ĐỒ THỊ EULER

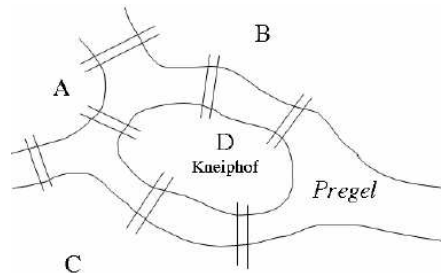
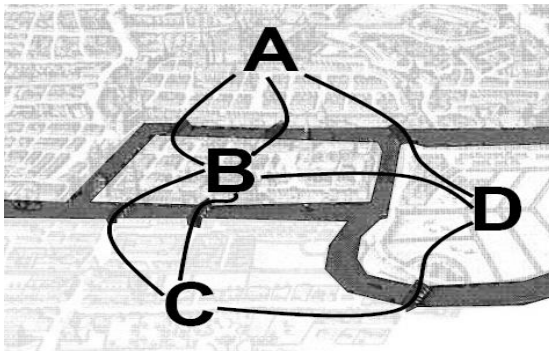
3.1.1. Giới thiệu

Có thể coi năm 1736 là năm khai sinh lý thuyết đồ thị, với việc công bố lời giải “bài toán về các cầu ở Königsberg” của nhà toán học lỗi lạc Euler (1707-1783). Thành phố Königsberg 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.

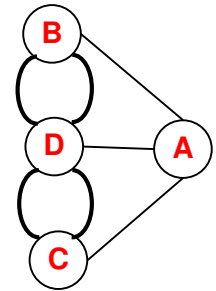
Dân thành phố từng thắc mắc: “Có thể nào đi dạo qua tất cả bảy cầu, mỗi cầu chỉ một lần thôi không?”. Nếu ta coi mỗi khu vực A, B, C, D như một đỉnh và mỗi cầu qua lại hai khu vực là một cạnh nối hai đỉnh thì ta có sơ đồ của Königsberg là một đa đồ thị G như hình trên.

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?

Năm 1736, Euler tìm ra cách giải quyết bài toán bảy cây cầu ở Königsberg trên bằng định lý của mình và đây là định lý đầu tiên của lý thuyết đồ thị.



Hình 3.1. Bảy cây cầu ở Königsberg



Hình 3.2. Đồ thị hóa bảy cây cầu ở Königsberg

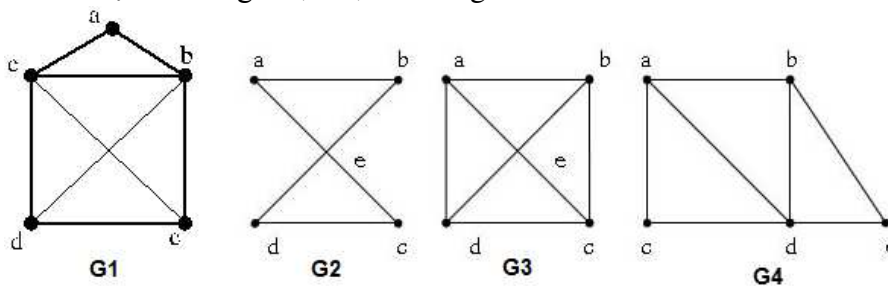
3.1.2. Định nghĩa

- Chu trình Euler: là đường đi qua mỗi cạnh của đồ thị G đúng một lần sao cho đỉnh bắt đầu và đỉnh kết thúc **phải** trùng nhau.
- Đường đi Euler: là đường đi qua mỗi cạnh của đồ thị G đúng một lần sao cho đỉnh bắt đầu và đỉnh kết thúc **không** trùng nhau.
- Đồ thị Euler: là đồ thị có chu trình Euler
- Đồ thị nửa Euler: là đồ thị có đường đi Euler.

3.1.3. Ví dụ

3.1.3.1. Ví dụ 1

Xét các đồ thị vô hướng G_1, G_2, G_3 trong hình 3.3



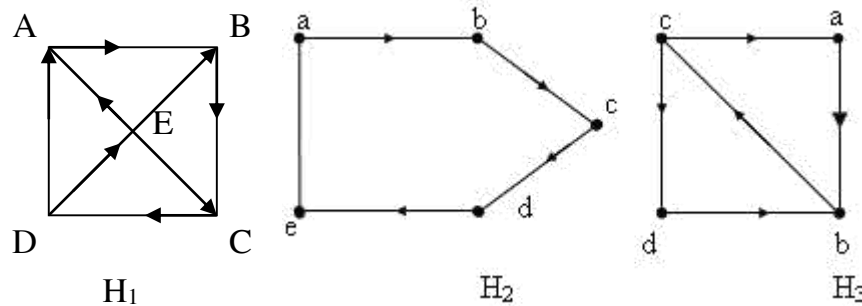
Hình 3.3. Đồ thị G_1, G_2, G_3, G_4

- Đồ thị G_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.
- Tương tự, đồ thị G_4 không có chu trình Euler nhưng nó có đường đi Euler c, b, a, e, c, d, e, b, d.
- Đồ thị G_2 không có chu trình cũng như đường đi Euler.

3.1.3.2. Ví dụ 2

Xét các đồ thị có hướng H_1, H_2, H_3 trong hình 3.4

- Đồ thị H_2 trong hình 2 là đồ thị Euler vì H_2 có chu trình Euler a, b, c, d, e, a.
- Đồ thị H_3 không có chu trình Euler nhưng H_3 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 3.4. Đồ thị H_1, H_2, H_3

3.1.4. Định lý Euler

3.1.4.1. 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.

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

3.1.4.2. Định lý 2

Đồ thị có hướng liên thông mạnh là đồ thị Euler khi và chỉ khi

$$d^+(v) = d^-(v), \forall v \in V$$

3.1.5. Xác định chu trình Euler bằng thuật toán Flor

3.1.5.1. Thuật toán Flor

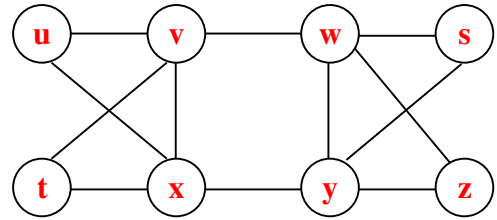
- Xuất phát từ một đỉnh u nào đó của G ta đi theo các cạnh của nó một cách tùy ý chỉ cần tuân thủ 2 quy tắc sau:
 - Xoá bỏ cạnh đã đi qua đồng thời xoá bỏ cả những đỉnh cô lập vừa được tạo thành.
 - Ở mỗi bước ta chỉ đi qua cầu khi không còn cách lựa chọn nào khác.
- **Cầu:** là cạnh khi bỏ đi sẽ làm cho đồ thị mất liên thông.

3.1.5.2. Ví dụ

Cho đồ thị G (hình 3.5). Tìm chu trình Euler.

- 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) (xóa (w,s)).
- Đi theo cạnh (s,y) (xóa (s,y) và đỉnh s).
- Vì (y,x) là cầu nên có thể đi theo một trong hai cạnh (y,w) , (y,z) , giả sử (y,w) (xóa (y,w)).
- Đi theo (w,z) (xóa (w,z) và đỉnh w) và theo (z,y) (xóa (z,y) và đỉnh z).
- Tiếp tục đi theo cạnh (y,x) (xóa (y,x) và đỉnh y).
- Vì (x,u) là cầu nên đi theo cạnh (x,v) hoặc (x,t) , giả sử (x,v) (xóa (x,v)).
- Tiếp tục đi theo cạnh (v,t) (xóa (v,t) và đỉnh v), theo cạnh (t,x) (xóa cạnh (t,x) và đỉnh t).
- Cuối cùng đi theo cạnh (x,u) (xóa (x,u) , đỉnh x và đỉnh u).



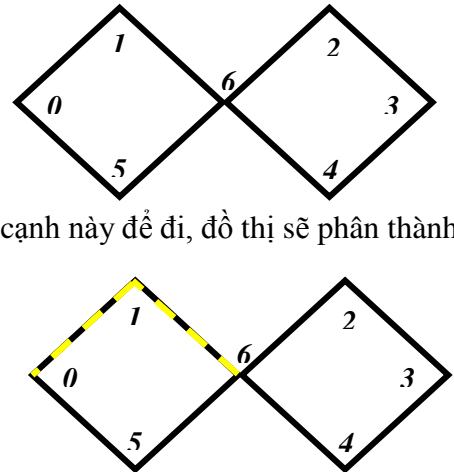
Hình 3.5 Đồ thị G

3.1.5.3. Lưu ý

Khi xóa bỏ cạnh đi qua và đỉnh cô lập, sẽ hình thành một đồ thị mới. Việc xét “cầu” tiếp theo sẽ trên đồ thị mới này.

3.1.5.4. Nhận xét

- Thuật toán này tuy đơn giản nhưng thường không được sử dụng bởi việc xác định “cầu” trong đồ thị không phải đơn giản.
- Ví dụ: Cho đồ thị sau:
 - Xuất phát từ đỉnh 0, có 2 cách đi: đến đỉnh 1 hoặc đến đỉnh 5. Giả sử chọn đỉnh 1 \Rightarrow xóa cạnh $(0,1)$
 - Tại đỉnh 1, chỉ có duy nhất 1 đường đi đến đỉnh 6, nên chọn đỉnh 6 \Rightarrow xóa cạnh $(1,6)$
 - Lúc này, cạnh $(5,6)$ trở thành “cầu”, do đó nếu chọn cạnh này để đi, đồ thị sẽ phân thành 2 đồ thị con do đó không tìm được chu trình (hoặc đường đi) Euler, mà phải chọn một trong hai đỉnh 2 hoặc 4 để hoàn tất chu trình (hoặc đường đi) Euler. Cụ thể là:
 - Nếu chọn đỉnh 2 thì chu trình Euler sẽ là: 0, 1, 6, 2, 3, 4, 6, 5, 0.
 - Nếu chọn đỉnh 4 thì chu trình Euler sẽ là: 0, 1, 6, 4, 3, 2, 6, 5, 0



3.1.6. Thuật toán tìm chu trình Euler bằng cách sử dụng cấu trúc Stack

3.1.6.1. Thuật toán tìm chu trình Euler

Cho $G=(X, E)$ là một đồ thị gồm n đỉnh. Thuật toán tìm chu trình Euler xuất phát từ u với u là đỉnh có bậc khác 0 như sau:

- Sử dụng stack tên “KetQua” là 1 stack để lưu kết quả tìm chu trình (hay đường đi).
- Nội dung hàm TimEuler:

TimEuler (u)

Với mỗi cạnh $e=(u,v)$ trong E

Loại cạnh e khỏi tập E ;

ChuTrinhEuler (v);

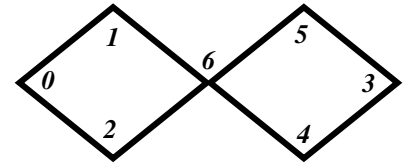
Cuối với mỗi

Thêm u vào stack ‘KetQua’;

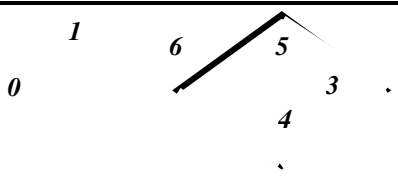
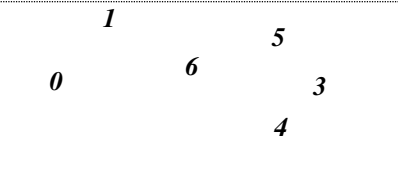
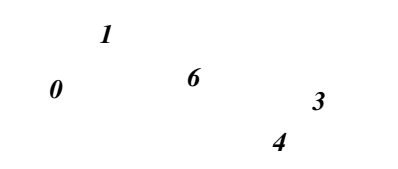
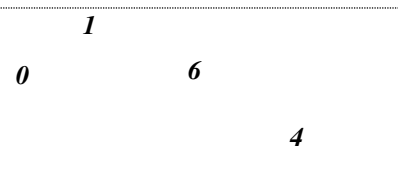



Cuối hàm

3.1.6.2. Ví dụ

Cho đồ thị sau:



Bước	Đồ thị	Stack KetQua	Diễn giải
0			Đỉnh 0 (0 đóng vai trò của đỉnh u) được chọn để xuất phát \Rightarrow có thể chọn một trong hai đỉnh 1 và 2. Chọn đỉnh 1.
1			Chọn đỉnh 1 \Rightarrow bỏ cạnh (0,1) Gọi đệ quy hàm Euler khi u=0 và v=1
2			Hàm Euler được gọi đệ quy với u=1 (= v của bước trước) Chọn đỉnh 6 \Rightarrow bỏ cạnh (1,6) Gọi đệ quy hàm Euler khi u=1 và v=6
3			Hàm Euler được gọi đệ quy với u=6 (= v của bước trước) Có thể chọn một trong các đỉnh 2, 4 và 5. Chọn đỉnh 2 \Rightarrow bỏ cạnh (6,2) Gọi đệ quy hàm Euler khi u=6 và v=2
4		0	Hàm Euler được gọi đệ quy với u=2 (= v của bước trước) Chọn đỉnh 0 \Rightarrow bỏ cạnh (2,0) Các cạnh nối với đỉnh 0 đã được xét hết. Đưa đỉnh 0 vào stack. Kết thúc việc gọi đệ quy của bước 3.
5		2 0	Kết thúc việc gọi đệ quy của bước 3. Lúc này u=2 Các cạnh nối với đỉnh 2 đã được xét hết \Rightarrow Đưa đỉnh 2 vào stack.
6		4 2 0	Kết thúc việc gọi đệ quy của bước 2 (khi u=6 và v=2). Do đỉnh 4 có cạnh nối với 6, gọi tiếp đệ quy của bước 2 (khi u=6 và v=4) Chọn đỉnh 4 \Rightarrow bỏ cạnh (6,4) Gọi đệ quy hàm Euler khi u=4
7		3 4 2 0	Chọn đỉnh 3 \Rightarrow bỏ cạnh (4,3) Gọi đệ quy hàm Euler khi u=3

Bước	Đồ thị	Stack KetQua	Diễn giải
8		<div>2</div> <div>0</div>	Chọn đỉnh 5 \Rightarrow bỏ cạnh (3,5) Gọi đệ quy hàm Euler khi $u=5$
9		<div>6</div> <div>2</div> <div>0</div>	Chọn đỉnh 6 \Rightarrow bỏ cạnh (5, 6) Tất cả các cạnh nối đến đỉnh 6 đều đã được xét Đưa đỉnh 6 vào stack
10		<div>5</div> <div>6</div> <div>2</div> <div>0</div>	Trở lại bước 8 ($u=5$). Tất cả các cạnh nối đến đỉnh 5 đều đã được xét Đưa đỉnh 5 vào stack
11		<div>3</div> <div>5</div> <div>6</div> <div>2</div> <div>0</div>	Trở lại bước 7 ($u=3$). Tất cả các cạnh nối đến đỉnh 3 đều đã được xét Đưa đỉnh 3 vào stack
12		<div>4</div> <div>3</div> <div>5</div> <div>6</div> <div>2</div> <div>0</div>	Trở lại bước 6 ($u=4$). Tất cả các cạnh nối đến đỉnh 4 đều đã được xét Đưa đỉnh 4 vào stack
13		<div>6</div> <div>4</div> <div>3</div> <div>5</div> <div>6</div> <div>2</div> <div>0</div>	Trở lại bước 3 ($u=6$). Tất cả các cạnh nối đến đỉnh 6 đều đã được xét Đưa đỉnh 6 vào stack
14		<div>0</div> <div>1</div> <div>6</div> <div>4</div> <div>3</div> <div>5</div> <div>6</div> <div>2</div> <div>0</div>	Trở lại bước 1 ($u=0$). Tất cả các cạnh nối đến đỉnh 0 đều đã được xét Đưa đỉnh 0 vào stack. Hoàn tất việc xét tất cả các cạnh có trong đồ thị. Lấy tên các đỉnh có trong stack ra (theo nguyên tắc LIFO), ta được chu trình Euler là: 0\rightarrow1\rightarrow6\rightarrow4\rightarrow3\rightarrow5\rightarrow6\rightarrow2\rightarrow0

3.1.6.3. Tìm đường đi Euler

Quá trình tìm đường đi Euler cũng như quá trình tìm chu trình Euler. Chỉ khác nhau ở chỗ đường đi Euler xuất phát từ đỉnh có bậc lẻ và kết thúc ở đỉnh bậc lẻ còn lại của đồ thị. Do đó kết quả trong stack sẽ có điểm đầu và điểm cuối khác nhau.

3.2. ĐƯỜNG ĐI HAMILTON VÀ ĐỒ THỊ HAMILTON

3.2.1. Giới thiệu

3.2.1.1. Bài toán của Hamilton

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 có 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.

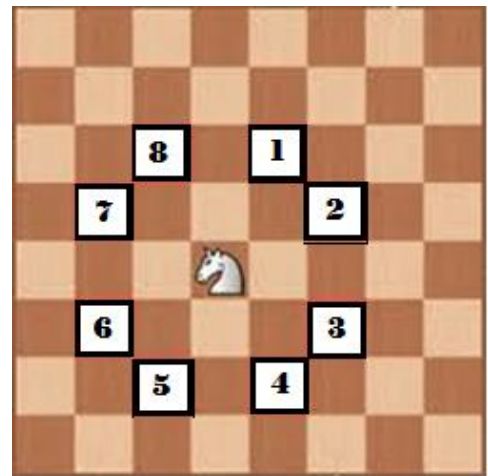
Trò chơi và câu đố trên dẫn tới việc khảo sát một lớp đồ thị đặc biệt, đó là đồ thị Hamilton.

3.2.1.2. Bài toán mã đi tuần

Trên bàn cờ, con mã chỉ có thể đi theo đường chéo của hình chữ nhật 2×3 hoặc 3×2 ô vuông. Giả sử bàn cờ có 8×8 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.

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 khống chế 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.



Hình 3.6

3.2.2. Đồ thị Hamilton

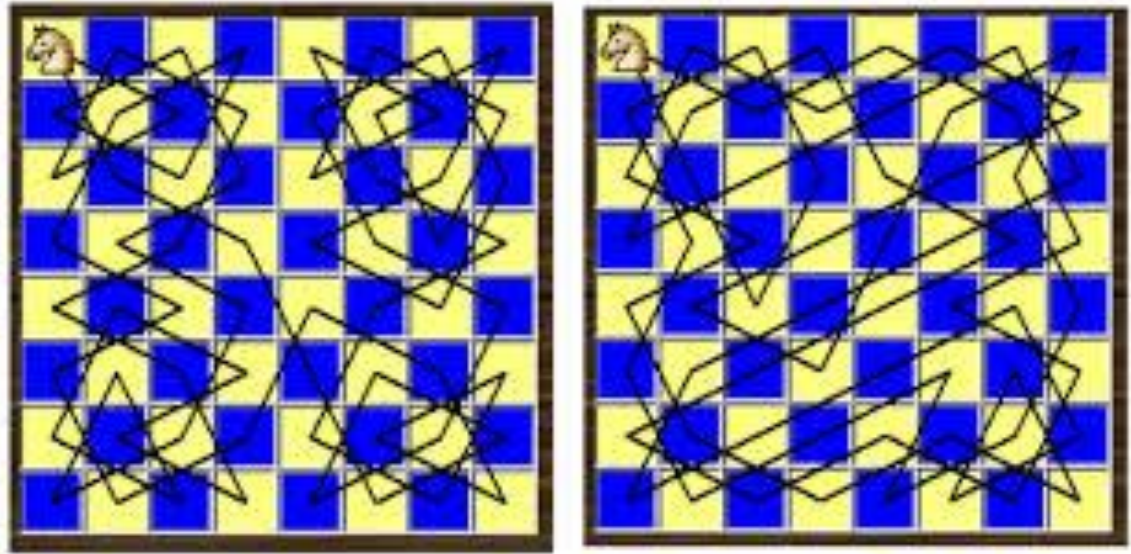
3.2.2.1. Định nghĩa

Chu trình chứa tất cả các đỉnh của đồ thị (vô hướng hoặc có hướng) G được gọi là chu trình Hamilton. Một đồ thị có chứa một chu trình Hamilton được gọi là đồ thị Hamilton.

- Đường đi Hamilton: là đường đi qua tất cả các đỉnh của đồ thị G đúng một lần.
- Chu trình Hamilton: là đường đi qua tất cả các đỉnh của đồ thị G đúng một lần sao cho đỉnh xuất phát và đỉnh kết thúc phải trùng nhau.
- Đồ thị Hamilton: là đồ thị có chu trình Hamilton
- Đồ thị nửa Hamilton: là đồ thị có đường đi Hamilton.

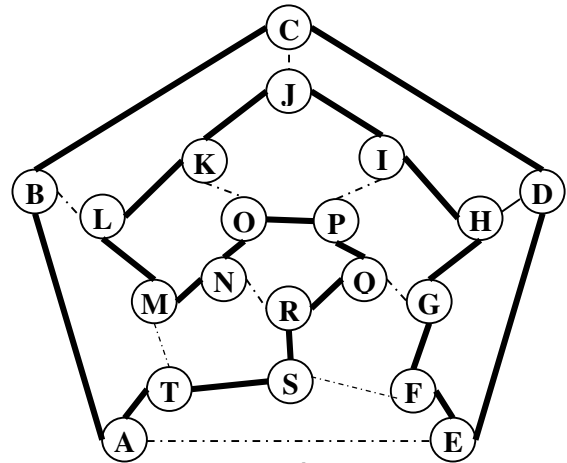
3.2.2.2. Ví dụ

- Ví dụ 1:



Hình 3.7 Hai lời giải về hành trình của con mã trên bàn cờ 8 x 8

- Ví dụ 2: Đồ thị Hamilton (hình 3.6) với chu trình Hamilton A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, S, T, A (đường tô đậm).



Hình 3.8: đồ thị Hamilton

3.2.3. Xác định đường đi và đồ thị Hamilton

Đường đi Hamilton và đồ thị Hamilton có nhiều ý nghĩa thực tiễn và đã được nghiên cứu nhiều, nhưng vẫn còn những khó khăn lớn chưa ai vượt qua được. 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.

3.2.3.1. Đồ thị Hamilton

- **Định lý 1(Dirak, 1952):** 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.
- **Định lý 2(Ore, 1960):** 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.
- **Định lý 3:** Nếu G là đồ thị phân đôi với hai tập đỉnh là V_1, V_2 có số đỉnh cùng bằng n ($n \geq 2$) và bậc của mỗi đỉnh lớn hơn $\frac{n}{2}$ thì G là một đồ thị Hamilton.

- **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.

3.2.3.2. Đồ thị nửa Hamilton

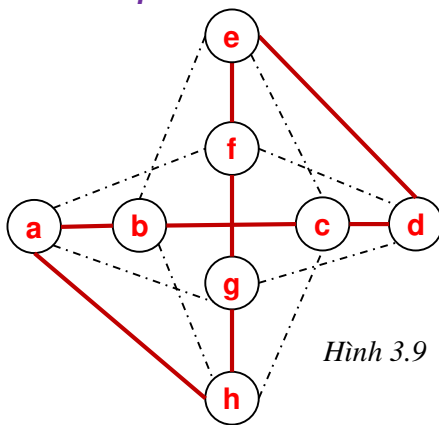
- **Định lý 5(Rédei):** Nếu G là một đồ thị có hướng đầy đủ thì G là đồ thị nửa Hamilton.

- **Hệ quả:** 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.

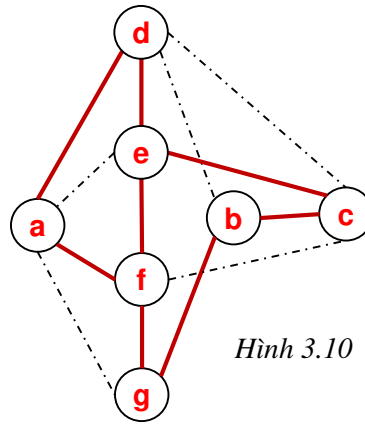
3.2.3.3. Một số nhận xét

- (i) Đồ thị có đỉnh có bậc ≤ 1 thì không có chu trình Hamilton.
- (ii) Đồ thị có các đỉnh đều có bậc ≥ 2 . Nếu có đỉnh nào có bậc bằng 2 thì mọi chu trình Hamilton (nếu có) phải đi qua hai cạnh kề với đỉnh này.
- (iii) Nếu trong đồ thị có đỉnh có ba đỉnh bậc 2 kề với nó thì đồ thị không có chu trình Hamilton.
- (iv) Nếu đỉnh a có hai đỉnh kề bậc 2 là b và c thì mọi cạnh $(a, x), x \notin \{b, c\}$ sẽ không thuộc bất kỳ chu trình Hamilton nào.
- (v) Đồ thị có đường đi vô hướng $\langle a_1, a_2, \dots, a_k \rangle$. Với chỉ số $k < n$ và các đỉnh trên đường đi (trừ a_1 và a_k) đều có bậc 2 thì cạnh (a_1, a_k) sẽ không thuộc bất kỳ chu trình Hamilton nào.
- (vi) Đồ thị hai phần $G = (V_1, V_2, F)$ với $|V_1| \neq |V_2|$ sẽ không có một chu trình Hamilton nào.

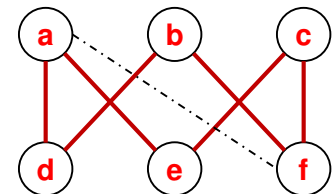
3.2.3.4. Ví dụ



Hình 3.9



Hình 3.10



Hình 3.11

- **Hình 3.9:** có 8 đỉnh, đỉnh nào cũng có bậc 7, nên theo Định lý Dirac, G_1 là đồ thị Hamilton.
- **Hình 3.10:** có 7 đỉnh bậc 6 và 2 đỉnh bậc 3 kề nhau nên tổng số bậc của hai đỉnh không kề nhau bất kỳ bằng 7 hoặc 8, nên theo Định lý Ore, G_2 là đồ thị Hamilton.
- **Hình 3.11:** Đồ thị phân đôi có bậc của mỗi đỉnh bằng 2 hoặc 3 ($> 3/2$), nên theo Định lý 3, G_3 là đồ thị Hamilton.

3.2.3.5. Thuật toán liệt kê tất cả các chu trình Hamilton

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.

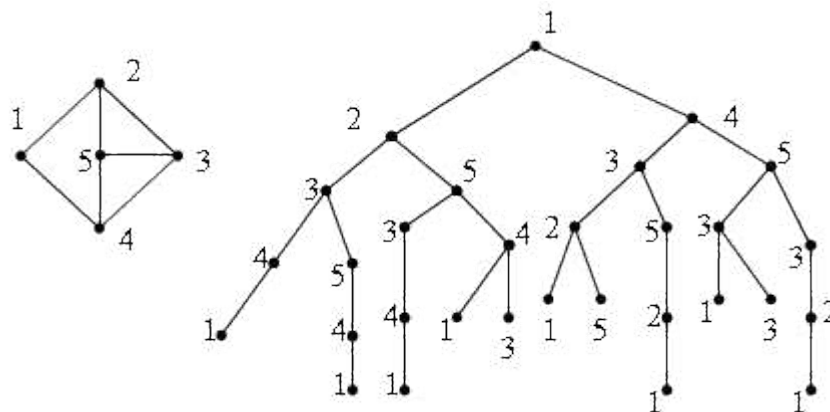
```

void main()
{
    for (v ∈ V)
        Chuaxet[v] =true;
    X[1] =0; // v0 la mot dinh nao do cua do thi
    Chuaxet[v0] =false;
    Hamilton(2);
}

void Hamilton(int k)
/* liet ke cac chu trinh Hamilton thu duoc bang viec phat trien
day dinh (X[1],... , X[k-1]) cua do thi G=(V,E) cho boi danh sach
ke: Ke(v), v∈ V */
{
    for (y ∈ Ke(X[k-1]))
        if ( (k ==N+1) && (y==v0))
            Ghinhan(X[1],... , X[n], v0)
        else
            if (Chuaxet[y])
            {
                X[k] =y;
                Chuaxet[y] =false;
                Hamilton(k+1);
                Chuaxet[y]=true;
            }
}

```

Ví dụ: Hình 3.12 mô tả cây tìm kiếm theo thuật toán vừa mô tả.



Hình 3.12. Đồ thị và cây liệt kê chu trình Hamilton của nó theo thuật toán quay lui

Trong trường hợp đồ thị có không quá nhiều cạnh thuật toán trên có thể sử dụng để kiểm tra đồ thị có phải là Hamilton hay không.

3.2.4. Một số ứng dụng của đồ thị Hamilton

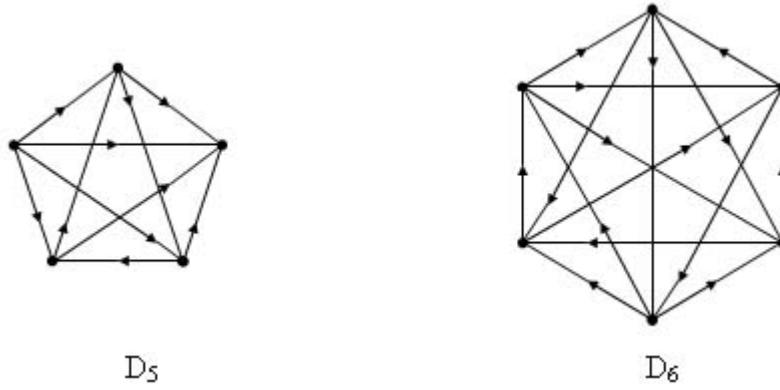
3.2.4.1. Đồ thị đấu loại

- 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**
 - Mọi đồ thị đấu loại là nửa Hamilton.
 - Mọi đồ thị đấu loại liên thông mạnh là Hamilton.

- **Ví dụ 1:** Đồ thị đầy loại D_5 , D_6 được cho trong hình 3.13.



- **Ví dụ 2:** Trong một đợt thi đấu bóng bàn có n ($n \geq 2$) đấu thủ tham gia. Mỗi đấu thủ gặp từng đấu thủ khác đúng một lần. Trong thi đấu bóng bàn chỉ có khả năng thắng hoặc thua. Chứng minh rằng sau đợt thi đấu có thể xếp tất cả các đấu thủ đứng thành một hàng dọc, để người đứng sau thắng người đứng ngay trước anh (chị) ta.

Xét đồ thị có hướng G gồm n đỉnh sao cho mỗi đỉnh ứng với một đấu thủ và có một cung nối từ đỉnh u đến đỉnh v nếu đấu thủ ứng với u thắng đấu thủ ứng với v . Như vậy, đồ thị G có tính chất là với hai đỉnh phân biệt bất kỳ u và v , có một và chỉ một trong hai cung (u,v) hoặc (v,u) , đồ thị như thế được gọi là đồ thị có hướng đầy đủ. Khi đó đường đi Hamilton trong G cho ta sự sắp xếp cần tìm.

3.2.4.2. Bài toán sắp xếp chỗ ngồi

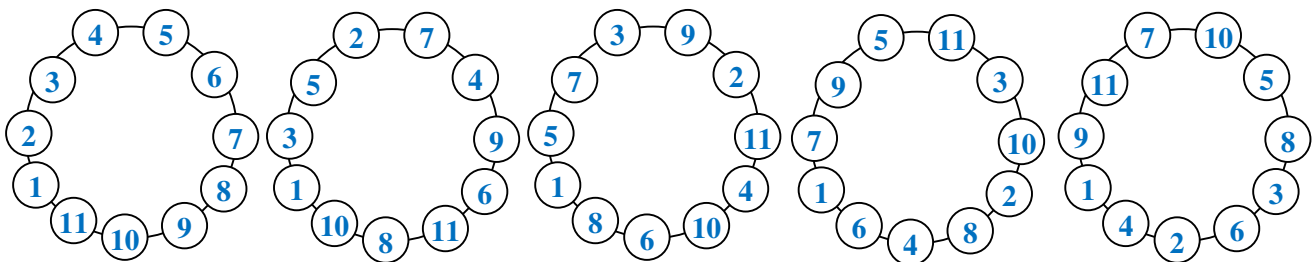
- **Định lý:** Đồ thị đầy đủ K_n với n lẻ và $n \geq 3$ có đúng $\frac{n-1}{2}$ chu trình Hamilton phân biệt.
- **Bài toán:**

Có n đại biểu từ n nước đến dự hội nghị quốc tế. Mỗi ngày họp một lần ngồi quanh một bàn tròn. Hỏi phải bố trí bao nhiêu ngày và bố trí như thế nào sao cho trong mỗi ngày, mỗi người có hai người kế bên là bạn mới. Lưu ý rằng n người đều muốn làm quen với nhau.

Xét đồ thị gồm n đỉnh, mỗi đỉnh ứng với mỗi người dự hội nghị, hai đỉnh kề nhau khi hai đại biểu tương ứng muốn làm quen với nhau. Như vậy, ta có đồ thị đầy đủ K_n . Đồ thị này là Hamilton và rõ ràng mỗi chu trình Hamilton là một cách sắp xếp như yêu cầu của bài toán. Bài toán trở thành tìm các chu trình Hamilton phân biệt của đồ thị đầy đủ K_n (hai chu trình Hamilton gọi là phân biệt nếu chúng không có cạnh chung).

- **Ví dụ:** Giải bài toán sắp xếp chỗ ngồi với $n=11$.

Có $(11-1)/2 = 5$ cách sắp xếp chỗ ngồi phân biệt như sau:



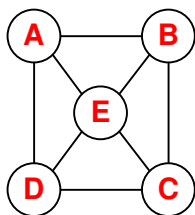
Hình 3.14. Minh họa bài toán sắp xếp chỗ ngồi với $n=11$

1 2 3 4 5 6 7 8 9 10 11 1

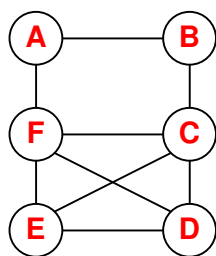
1 3 5 2 7 4 9 6 11 8 10 1
 1 5 7 3 9 2 11 4 10 6 8 1
 1 7 9 5 11 3 10 2 8 4 6 1
 1 9 11 7 10 5 8 3 6 2 4 1

3.3. BÀI TẬP

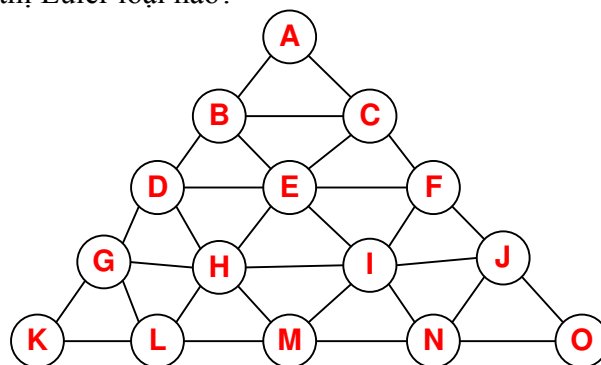
3.3.1. Xác định xem các đồ thị sau thuộc dạng đồ thị Euler loại nào?



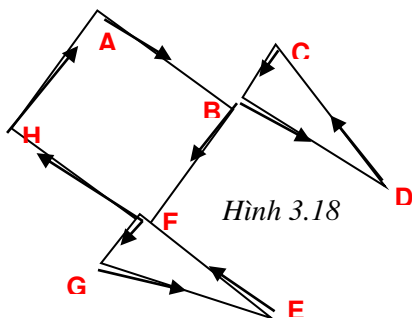
Hình 3.15



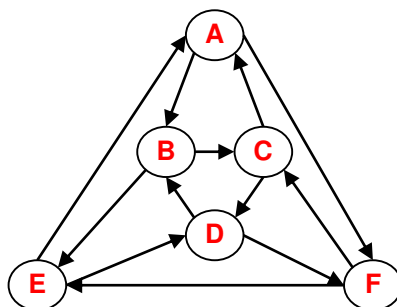
Hình 3.16



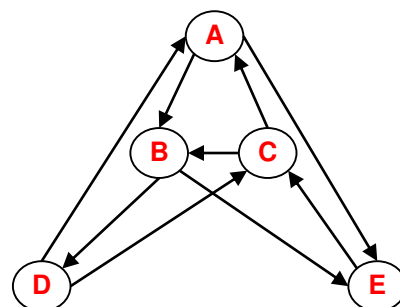
Hình 3.17



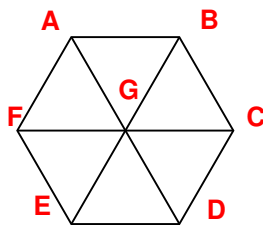
Hình 3.18



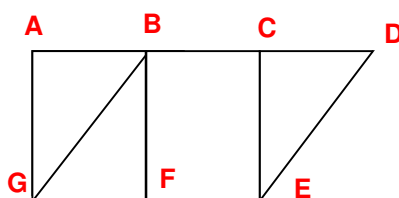
Hình 3.19



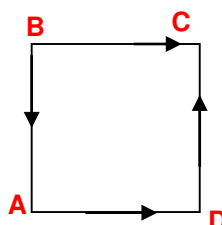
Hình 3.20



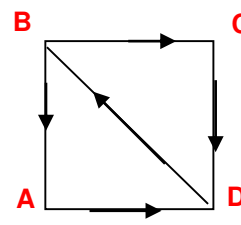
Hình 3.21



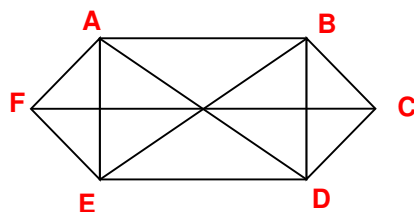
Hình 3.22



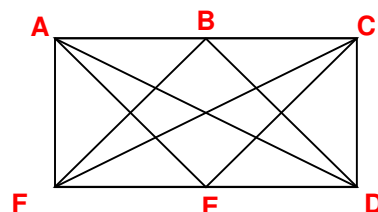
Hình 3.23



Hình 3.24

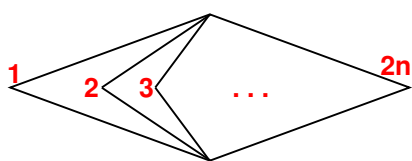


Hình 3.25

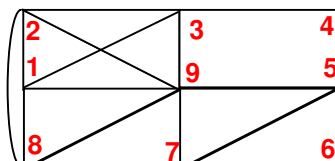


Hình 3.26

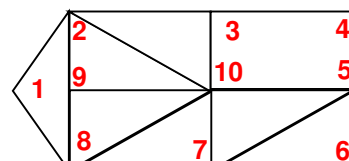
3.3.2. Xác định số lượng chu trình Euler của đồ thị trong các hình 3.27, 3.28, 3.29:



Hình 3.27



Hình 3.28



Hình 3.29

3.3.3. Tìm chu trình hoặc đường Euler (nếu có) của các đồ thị có ma trận kề như sau:

3.3.3.1.

	0	1	2	3	4	5	6	7	8	9
0		1				1				
1	1		1			1		1		
2		1		1				1	1	
3			1		1				1	1
4				1		1				
5	1	1			1		1			
6						1		1		
7		1	1				1		1	
8			1	1				1		1
9				1					1	

3.3.3.2.

	1	2	3	4	5	6	7
1		1		1			
2	1		1	1	1		
3		1		1		1	1
4	1	1	1		1	1	1
5		1		1		1	1
6			1	1	1		1
7			1	1	1	1	

3.3.3.3.

	0	1	2	3	4	5	6	7	8	9
0		1							1	1
1			1					1		
2				1			1			
3								1		1
4				1				1		
5					1		1			
6					1	1				
7	1						1		1	
8			1							1
9	1	1								

3.3.3.4.

	1	2	3	4	5	6	7
1		1	1		1	1	
2	1		1			1	
3	1	1		1		1	
4			1		2		1
5	1			2		1	
6	1	1	1		1		
7				1			

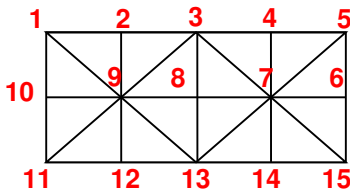
3.3.3.5.

	1	2	3	4	5	6	7	8
1	2	1			1			
2	1		1	1	1			
3		1			1	2		
4		1				1	1	1
5	1	1	1				1	
6			2	1			1	
7				1	1	1		1
8				1			1	2

3.3.3.6.

	1	2	3	4	5	6	7	8
1	1				1			
2	1				1			
3		1				1		
4		1				1		
5			1				1	
6			1				1	
7				1				1
8				1				1

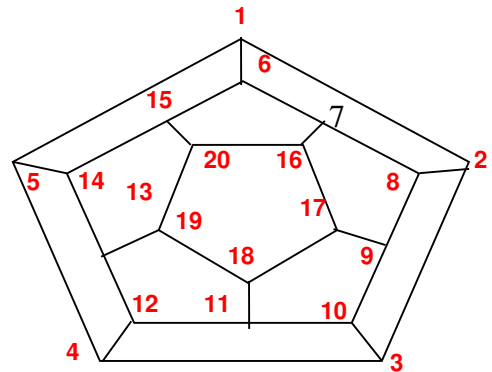
3.3.4. Tìm một chu trình Hamilton của đồ thị trong hình 3.30, 3.31, 3.32:



Hình 3.30



Hình 3.31



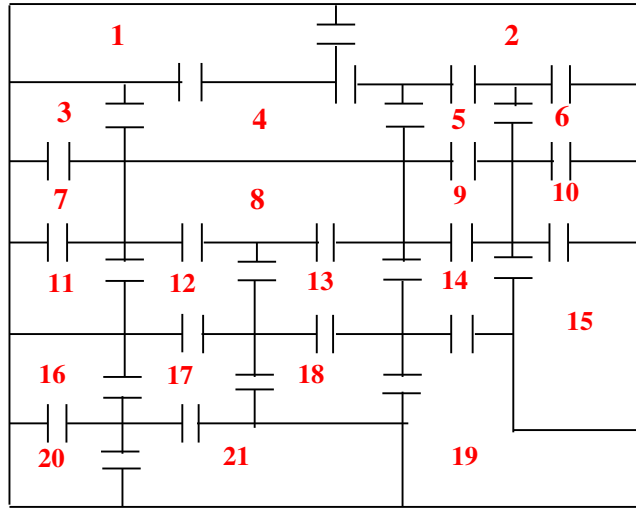
Hình 3.32

3.3.5. Tìm

3.3.5.1. Một đồ thị có chu trình Hamilton nhưng không có chu trình Euler.

3.3.5.2. Một đồ thị có chu trình Euler nhưng không có chu trình Hamilton.

- 3.3.6.** Trong một cuộc họp có 15 người mỗi ngày ngồi với nhau quanh một bàn tròn một lần. Hỏi có bao nhiêu cách sắp xếp sao cho mỗi lần ngồi họp, mỗi người có hai người bên cạnh là bạn mới, và sắp xếp như thế nào ?
- 3.3.7.** Hiệu trưởng mời $2n$ ($n \geq 2$) sinh viên giỏi đến dự tiệc. Mỗi sinh viên giỏi quen ít nhất n sinh viên giỏi khác đến dự tiệc. Chứng minh rằng luôn luôn có thể xếp tất cả các sinh viên giỏi ngồi xung quanh một bàn tròn, để mỗi người ngồi giữa hai người mà sinh viên đó quen.
- 3.3.8.** Một ông vua đã xây dựng một lâu đài để cất báu vật. Người ta tìm thấy sơ đồ của lâu đài (hình 3.33) với lời dẫn: muốn tìm báu vật, chỉ cần từ một trong các phòng bên ngoài cùng (số 1, 2, 6, 10, ...), đi qua tất cả các cửa phòng, mỗi cửa chỉ một lần; báu vật được giấu sau cửa cuối cùng. Hãy tìm nơi giấu báu vật

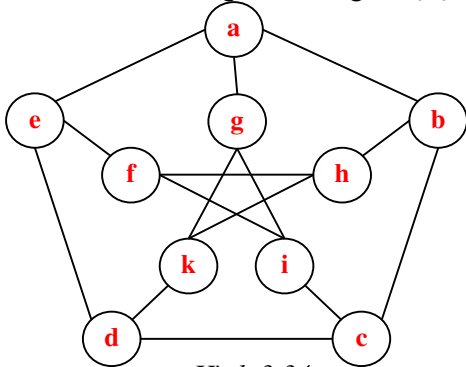


Hình 3.33

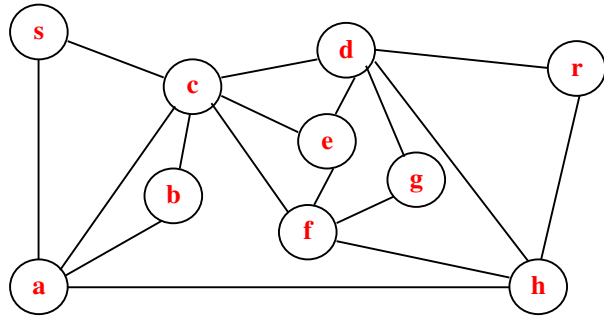
3.3.9. Đồ thị cho trong hình 3.34 gọi là đồ thị Peterson P.

3.3.9.1. Tìm một đường đi Hamilton trong P.

3.3.9.2. Chứng minh rằng $P \setminus \{v\}$, với v là một đỉnh bất kỳ của P, là một đồ thị Hamilton



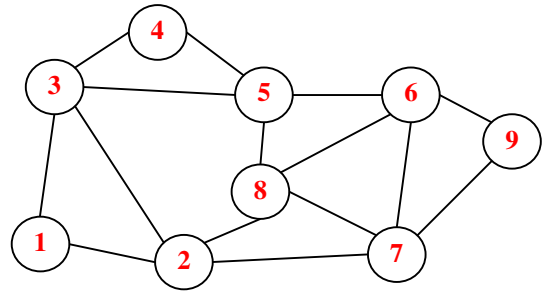
Hình 3.34



Hình 3.35

- 3.3.10.** Chứng minh rằng đồ thị cho trong hình 3.35 có đường đi Hamilton nhưng không có chu trình Hamilton.
- 3.3.11.** Cho ví dụ về:
- 3.3.11.1.** Đồ thị có một chu trình vừa là chu trình Euler vừa là chu trình Hamilton;
 - 3.3.11.2.** Đồ thị có một chu trình Euler và một chu trình Hamilton, nhưng hai chu trình đó không trùng nhau;
 - 3.3.11.3.** Đồ thị có 6 đỉnh, là đồ thị Hamilton, nhưng không phải là đồ thị Euler.
 - 3.3.11.4.** Đồ thị có 6 đỉnh, là đồ thị Euler, nhưng không phải là đồ thị Hamilton.

- 3.3.12.** Xác định xem đồ thị trong hình 3.36 đây thuộc loại nào trong những loại sau đây: Đồ thị Euler, đường đi Euler, đồ thị Hamilton, chu trình Hamilton? (vẽ hình minh họa để giải thích)



Hình 3.36



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. 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
- Chọn một phương pháp tiết kiệm nhất để đưa ra một hệ thống động lực từ trạng thái xuất phát đến trạng thái đích.
- Lập lịch thi công các công các công đoạn trong một công trình thi công lớn.
- 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.

4.1. CÁC KHÁI NIỆM

Trong chương này chúng ta xét đồ thị $G=(V,E)$, $|V|=n$, $|E|=m$ với các cung được gán trọng số là số dương (>0), nghĩa là:

- Mỗi cung $(u,v) \in E$ được đặt tương ứng với một số thực $a(u,v)$ gọi là trọng số của nó.
- Nếu $(u,v) \notin E$, Chúng ta sẽ đặt $a(u,v) = \infty$.

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

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

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) $t \in V$. Nếu như không tồn tại đường đi từ s đến t thì ta sẽ đặt $d(s,t)=\infty$.

4.2. THUẬT TOÁN DIJKSTRA

4.2.1. Công dụng

Thuật toán được xây dựng để tìm đường đi ngắn nhất từ một đỉnh cho trước đến tất cả các đỉnh còn lại với điều kiện trọng số trên các cung là không âm.

4.2.2. Cách thực hiện

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.

```
void 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  $d[v]$ ,
         $v \in V$ .
```

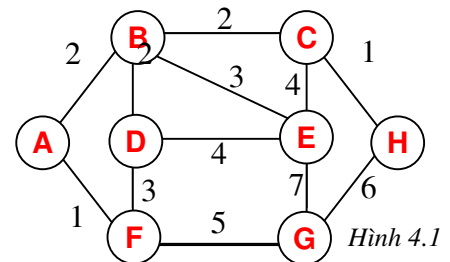
```

    Truoc[v],  $v \in V$ , ghi nhận đỉnh đi trước v trong đường đi ngắn nhất từ
    s đến v */
{
    (* Khởi tạo *)
    for (v  $\in$  V)
    {
        d[v]=a[s,v];
        Truoc[v]=s;
    }
    d[s]=0;
    T=V\{s}; /* T là tập các đỉnh cá nhân tạm thời */
    while (T  $\neq$   $\emptyset$ )
    {
        tìm đỉnh  $u \in T$  thoả mãn  $d[u]=\min \{d[z]:z \in T\}$ ;
        T=T\{u}; /* Cố định nhãn của đỉnh u */
        for (v  $\in$  T)
            if ( $d[v] > d[u] + a[u,v]$ )
            {
                d[v]=d[u]+a[u,v];
                Truoc[v]=u;
            }
    }
}

```

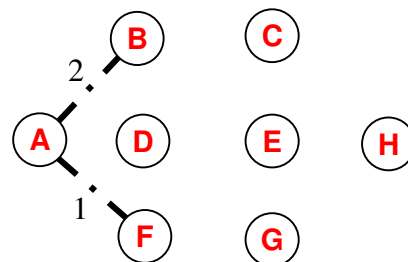
4.2.3. Ví dụ

- Tìm đường đi ngắn nhất từ A đến các đỉnh còn lại của đồ thị ở hình 1.
- **Quy ước:** khi có 2 đỉnh có cùng chiều dài nhỏ nhất, ta chọn đỉnh có số thứ tự nhỏ hơn.



- **Bước 1:** lập được bảng sau:

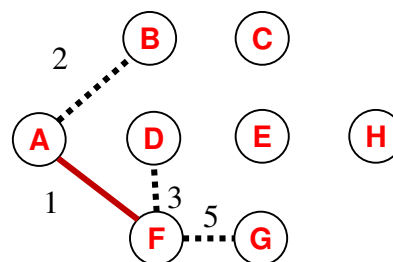
	A	B	C	D	E	F	G	H
Đã xét	L							
Chiều dài min	0	2	∞	∞	∞	1	∞	∞
Đỉnh trước		A	A	A	A	A	A	A



Hình 4.2

- **Bước 2:** Đỉnh F có chiều dài nhỏ nhất (=1) nên được chọn, lập bảng 2 như sau:

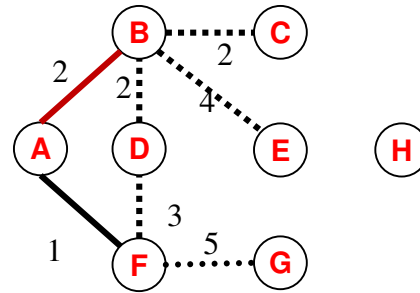
	A	B	C	D	E	F	G	H
Đã xét	X					X		
Chiều dài min	0	2	∞	4	∞	-	6	∞
Đỉnh trước		A	A	F	A	A	F	A



Hình 4.3

- **Bước 3:** Đỉnh B có chiều dài nhỏ nhất (=2) nên được chọn, lập bảng 3 như sau:

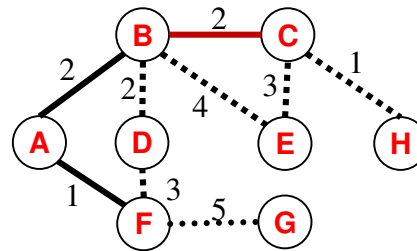
	A	B	C	D	E	F	G	H
Đã xét	X	X				X		
Chiều dài min	0	-	4	4	6	-	6	∞
Đỉnh trước		A	B	F	B	A	F	A



Hình 4.4

- **Bước 4:** Đỉnh C có chiều dài nhỏ nhất tính và là đỉnh được xét trước (=4) nên được chọn, lập được bảng 4 như sau:

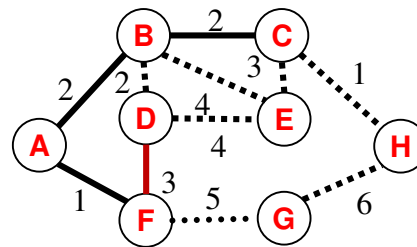
	A	B	C	D	E	F	G	H
Đã xét	X	X	X	X		X		
Chiều dài min	0	-	-	4	6	-	6	5
Đỉnh trước		A	B	F	B	A	F	C



Hình 4.5

- **Bước 5:** Đỉnh D có chiều dài nhỏ nhất (=4) nên được chọn, lập bảng 5 như sau:

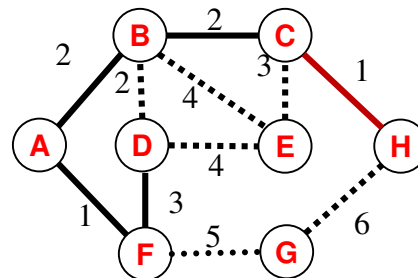
	A	B	C	D	E	F	G	H
Đã xét	X	X	X	X		X		
Chiều dài min	0	-	-	-	6	-	6	5
Đỉnh trước		A	B	F	B	A	F	C



Hình 4.6

- **Bước 6:** Đỉnh H có chiều dài nhỏ nhất (=5) nên được chọn, lập bảng 6 như sau:

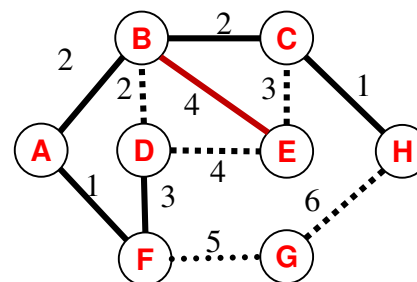
	A	B	C	D	E	F	G	H
Đã xét	X	X	X	X		X		X
Chiều dài min	0	-	-	-	6	-	6	-
Đỉnh trước		A	B	F	B	A	F	C



Hình 4.7

- **Bước 7:** Đỉnh E có chiều dài nhỏ nhất (=6) nên được chọn, lập được bảng 7 như sau:

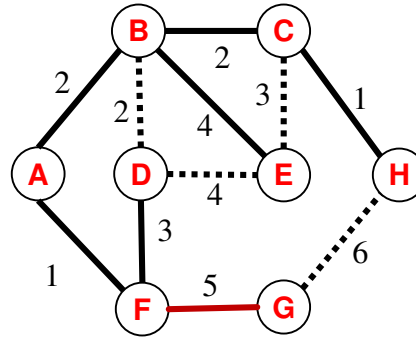
	A	B	C	D	E	F	G	H
Đã xét	X	X	X	X		X	X	X
Chiều dài min	0	-	-	-	6	-	-	-
Đỉnh trước		A	B	F	B	A	F	C



Hình 4.8

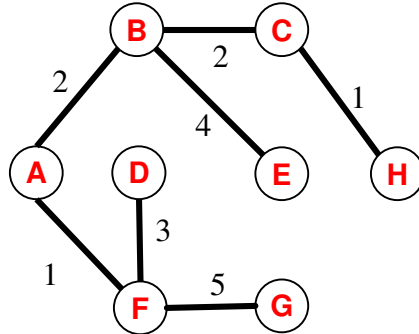
- **Bước 8:** Đỉnh G là đỉnh cuối cùng nên được chọn, lập được bảng 8 như sau:

	A	B	C	D	E	F	G	H
Đã xét	X	X	X	X	X	X	X	X
Chiều dài min	0	-	-	-	6	-	-	-
Đỉnh trước		A	B	F	B	A	F	C



Hình 4.9

Vậy đường đi từ đỉnh A đến tất cả các đỉnh với chiều dài nhỏ nhất là:



Hình 4.10

Để tránh lập bảng nhiều lần, ta có thể gộp các bước vào chung 1 bảng như sau. Trong đó ký hiệu

- ∞, a : số bên trái dấu phẩy là chiều dài min, số bên phải là đỉnh trước của đỉnh đang xét.
- Đỉnh có dấu hoa thị (\star) là đỉnh được chọn trong bước đang xét.

Bước lập	Đỉnh							
	A	B	C	D	E	F	G	H
1	0,A	2,A	∞ ,A	∞ ,A	∞ ,A	1,A \star	∞ ,A	∞ ,A
2	-	2,A \star	4,B	4,F	6,B	-	6,F	∞ ,A
3	-	-	4,B \star	4,F	6,B	-	6,F	∞ ,A
4	-	-	-	4,F \star	6,B	-	6,F	5,C
5	-	-	-	-	6,B	-	6,F	5,C \star
6	-	-	-	-	6,B \star	-	6,F	-
7	-	-	-	-	-	-	6,F \star	-

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.

4.3. THUẬT TOÁN FLOYD

4.3.1. Công dụng

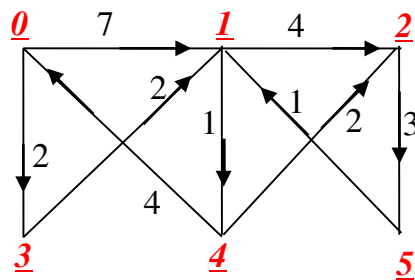
Thuật toán được xây dựng để tìm đường đi ngắn nhất giữa mọi cặp đỉnh.

4.3.2. Thuật toán Floyd

```
void Floyd(DO THI G)
{
    /* 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
             $w[i,j] =$ ,  $i, j = 1, 2, \dots, n$ .
            Trong đó  $w[i,j]$  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  $j$ . */
    /* Khởi tạo 2 ma trận P và W */
    for (i = 0 to n-1)
        for (j = 0 to n - 1)
        {
            W[i, j] = A[i,j];
            P[i, j] = i;
        }
    // Thi hành thuật toán Floyd
    for (k = 0 to n - 1)
        for (i = 0 to n - 1)
            for (j = 0 to n - 1)
                if (W[i, j] > W[i, k] + W[k, j])
                {
                    W[i, j] = W[i, k] + W[k, j];
                    P[i, j] = P[k, j];
                }
}
```

4.3.3. Ví dụ

4.3.3.1. Ví dụ đồ thị có hướng



Hình 4.11

Minh họa đồ thị hình 4.13 bằng ma trận kề

w

	0	1	2	3	4	5
0	$+\infty$	7	$+\infty$	2	$+\infty$	$+\infty$
1	$+\infty$	$+\infty$	4	$+\infty$	1	$+\infty$
2	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$	3
3	$+\infty$	4	$+\infty$	$+\infty$	$+\infty$	$+\infty$
4	2	$+\infty$	2	$+\infty$	$+\infty$	$+\infty$
5	$+\infty$	1	$+\infty$	$+\infty$	$+\infty$	$+\infty$

p

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

▪ Khi $k=0$:

- Với: $i=4; j=1 \Rightarrow$ xét $w[4,1] > w[4,0] + w[0,1]$, ta có: $+\infty > 2 + 7$

\Rightarrow gán $w[4,1] = w[4,0] + w[0,1] \Rightarrow w[4,1] = 2 + 7 = 9$

và gán $p[i,j] = p[k,j] \Rightarrow p[4,1] = p[0,1] = 0$

- Với: $i=4; j=3 \Rightarrow$ xét $w[4,3] > w[4,0] + w[0,3]$, ta có: $+\infty > 2 + 2$

\Rightarrow gán $w[4,3] = w[4,0] + w[0,3] \Rightarrow w[4,3] = 2 + 2 = 4$

và gán $p[i,j] = p[k,j] \Rightarrow p[4,3] = p[0,3] = 0$

w
($k=0$)

	0	1	2	3	4	5
0	$+\infty$	7	$+\infty$	2	$+\infty$	$+\infty$
1	$+\infty$	$+\infty$	4	$+\infty$	1	$+\infty$
2	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$	3
3	$+\infty$	4	$+\infty$	$+\infty$	$+\infty$	$+\infty$
4	2	9	2	4	$+\infty$	$+\infty$
5	$+\infty$	1	$+\infty$	$+\infty$	$+\infty$	$+\infty$

p

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

▪ Khi $k=1$:

- Với: $i=0; j=2 \Rightarrow$ xét $w[0,2] > w[0,1] + w[1,2]$, ta có: $+\infty > 7 + 4$

\Rightarrow gán $w[0,2] = w[0,1] + w[1,2] \Rightarrow w[0,2] = 7 + 4 = 11$

và gán $p[i,j] = p[k,j] \Rightarrow p[0,2] = p[1,2] = 1$

- Với: $i=0; j=4 \Rightarrow$ xét $w[0,4] > w[0,1] + w[1,4]$, ta có: $+\infty > 7 + 1$

\Rightarrow gán $w[0,4] = w[0,1] + w[1,4] \Rightarrow w[0,4] = 7 + 1 = 8$

và gán $p[i,j] = p[k,j] \Rightarrow p[0,4] = p[1,4] = 0$

- Với: $i=3; j=2 \Rightarrow$ xét $w[3,2] > w[3,1] + w[1,2]$, ta có: $+\infty > 4 + 4$

\Rightarrow gán $w[3,2] = w[3,1] + w[1,2] \Rightarrow w[3,2] = 4 + 4 = 8$

và gán $p[i,j] = p[k,j] \Rightarrow p[3,2] = p[1,2] = 1$

- Với: $i=3; j=4 \Rightarrow$ xét $w[3,4] > w[3,1] + w[1,4]$, ta có: $+\infty > 4 + 1$

\Rightarrow gán $w[3,4] = w[3,1] + w[1,4] \Rightarrow w[3,4] = 4 + 1 = 5$

và gán $p[i,j] = p[k,j] \Rightarrow p[3,4] = p[1,4] = 1$

- Với: $i=4; j=4 \Rightarrow$ xét $w[4,4] > w[4,1] + w[1,4]$, ta có: $+\infty > 9 + 1$

\Rightarrow gán $w[4,4] = w[4,1] + w[1,4] \Rightarrow w[4,4] = 9 + 1 = 10$

và gán $p[i,j] = p[k,j] \Rightarrow p[4,4] = p[1,4] = 1$

- Với: $i=5; j=2 \Rightarrow$ xét $w[5,2] > w[5,1] + w[1,2]$, ta có: $+\infty > 1 + 4$

\Rightarrow gán $w[5,2] = w[5,1] + w[1,2] \Rightarrow w[5,2] = 1 + 4 = 5$

và gán $p[i,j] = p[k,j] \Rightarrow p[5,2] = p[1,2] = 1$

- Với: $i=5; j=4 \Rightarrow$ xét $w[5,4] > w[5,1] + w[1,4]$, ta có: $+\infty > 1 + 1$
 \Rightarrow gán $w[5,4] = w[5,1] + w[1,4] \Rightarrow w[5,4] = 1 + 1 = 2$
 và gán $p[i,j] = p[k,j] \Rightarrow p[3,4] = p[5,4] = 1$

W
(k=1)

	0	1	2	3	4	5
0	$+\infty$	7	11	2	8	$+\infty$
1	$+\infty$	$+\infty$	4	$+\infty$	1	$+\infty$
2	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$	3
3	$+\infty$	4	8	$+\infty$	5	$+\infty$
4	2	9	2	4	10	$+\infty$
5	$+\infty$	1	5		2	$+\infty$

p

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

- Thực hiện tương tự, sau khi k=2, ta có:

W
(k=2)

	0	1	2	3	4	5
0	$+\infty$	7	11	2	8	14
1	$+\infty$	$+\infty$	4	$+\infty$	1	7
2	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$	3
3	$+\infty$	4	8	$+\infty$	5	11
4	2	9	2	4	10	5
5	$+\infty$	1	5	$+\infty$	2	8

p

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

- Thực hiện tương tự, sau khi k=3, ta có:

W
(k=3)

	0	1	2	3	4	5
0	$+\infty$	6	10	2	7	13
1	$+\infty$	$+\infty$	4	$+\infty$	1	7
2	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$	3
3	$+\infty$	4	8	$+\infty$	5	11
4	2	8	2	4	9	5
5	$+\infty$	1	5	$+\infty$	2	8

p

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

- Thực hiện tương tự, sau khi k=4, ta có:

W
(k=4)

	0	1	2	3	4	5
0	9	6	9	2	7	12
1	3	9	3	5	1	6
2	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$	3
3	7	4	7	9	5	10
4	2	8	2	4	9	5
5	4	1	4	6	2	7

p

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

- Cuối cùng, sau khi k=5, ta có:

W
(k=5)

	0	1	2	3	4	5
0	9	6	9	2	7	12
1	3	7	3	5	1	6
2	7	4	7	9	5	3
3	7	4	7	9	5	10
4	2	6	2	4	7	5
5	4	1	4	6	2	7

p

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

KẾT QUẢ:

- Dựa vào ma trận p để tìm đường đi ngắn nhất từ 1 đỉnh x bất kỳ đến đỉnh y bất kỳ. Ví dụ, cần tìm đường đi từ đỉnh 0 đến đỉnh 5. Tra trên dòng 0 (dòng của đỉnh xuất phát) trong ma trận p:

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

Do $p[0, 5]=2$, cho ta biết để đi từ đỉnh **0** đến đỉnh **5** phải đi qua đỉnh **2**

Tra ô $p[0, 2]=4$, cho ta biết để đi từ đỉnh **0** đến đỉnh **2** phải đi qua đỉnh **4**

Tra ô $p[0, 4]=1$, cho ta biết để đi từ đỉnh **0** đến đỉnh **4** phải đi qua đỉnh **1**

Tra ô $p[0, 1]=3$, cho ta biết để đi từ đỉnh **0** đến đỉnh **1** phải đi qua đỉnh **3**

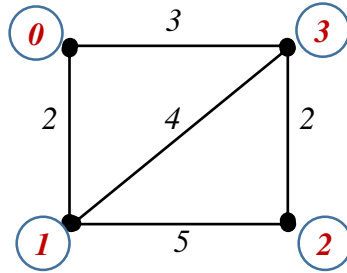
Tra ô $p[0, 3]=0$. Kết thúc quá trình tra vì 0 vừa thu được chính là tên của đỉnh xuất phát.

Vậy, lần ngược lại kết quả vừa tra ta có đường đi ngắn nhất từ đỉnh 0 đến đỉnh 5 là:

$$0 \rightarrow 3 \rightarrow 1 \rightarrow 4 \rightarrow 2 \rightarrow 5$$

- Dựa vào ma trận W để biết chiều dài ngắn nhất để đi từ đỉnh x đến đỉnh y. Ví dụ, đường đi ngắn nhất từ đỉnh 0 đến đỉnh 5 là 12 ($=w[0,5]$).

4.3.3.2. Ví dụ đồ thị vô hướng



Hình 4.12

Mảng W

Khi khởi tạo

	0	1	2	3
0	100	2	100	3
1	2	100	5	4
2	100	5	100	2
3	3	4	2	100

Sau khi xét k=0

	0	1	2	3
0	100	2	100	3
1	2	4	5	4
2	100	5	100	2
3	3	4	2	6

Mảng P

Khi khởi tạo

	0	1	2	3
0	0	0	0	0
1	1	1	1	1
2	2	2	2	2
3	3	3	3	3

Sau khi xét k=0

	0	1	2	3
0	0	0	0	0
1	1	0	1	1
2	2	2	2	2
3	3	3	3	0

Mảng W

Sau khi xét k=0

	0	1	2	3
0	100	2	100	3
1	2	4	5	4
2	100	5	100	2
3	3	4	2	6

Sau khi xét k=1

	0	1	2	3
0	4	2	7	3
1	2	4	5	4
2	7	5	10	2
3	3	4	2	6

Mảng P

Sau khi xét k=0

	0	1	2	3
0	0	0	0	0
1	1	0	1	1
2	2	2	2	2
3	3	3	3	0

Sau khi xét k=1

	0	1	2	3
0	1	0	1	0
1	1	0	1	1
2	1	2	1	2
3	3	3	3	0

Mảng W

Sau khi xét k=1

	0	1	2	3
0	4	2	7	3
1	2	4	5	4
2	7	5	10	2
3	3	4	2	6

Sau khi xét k=2

	0	1	2	3
0	4	2	7	3
1	2	4	5	4
2	7	5	10	2
3	3	4	2	4

Mảng P

Sau khi xét k=1

	0	1	2	3
0	1	0	1	0
1	1	0	1	1
2	1	2	1	2
3	3	3	3	0

Sau khi xét k=2

	0	1	2	3
0	1	0	1	0
1	1	0	1	1
2	1	2	1	2
3	3	3	3	1

Mảng W

Sau khi xét k=2

	0	1	2	3
0	4	2	7	3
1	2	4	5	4
2	7	5	10	2
3	3	4	2	4

Sau khi xét k=3

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

Mảng P

Sau khi xét k=2

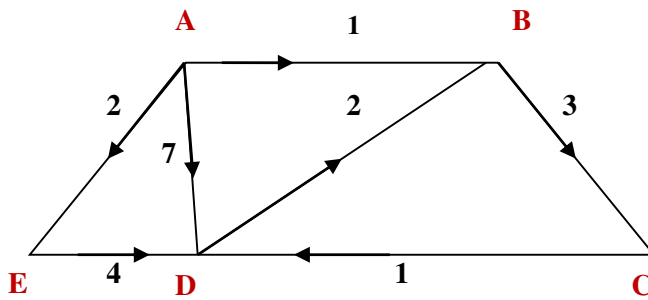
	0	1	2	3
0	1	0	1	0
1	1	0	1	1
2	1	2	1	2
3	3	3	3	3

Sau khi xét k=3

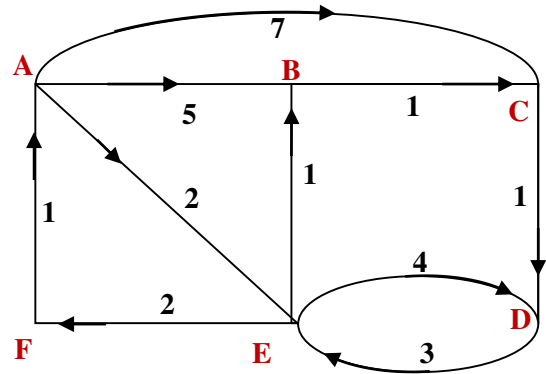
	0	1	2	3
0	1	0	3	0
1	1	0	1	1
2	3	2	3	2
3	3	3	3	3

4.4. BÀI TẬP

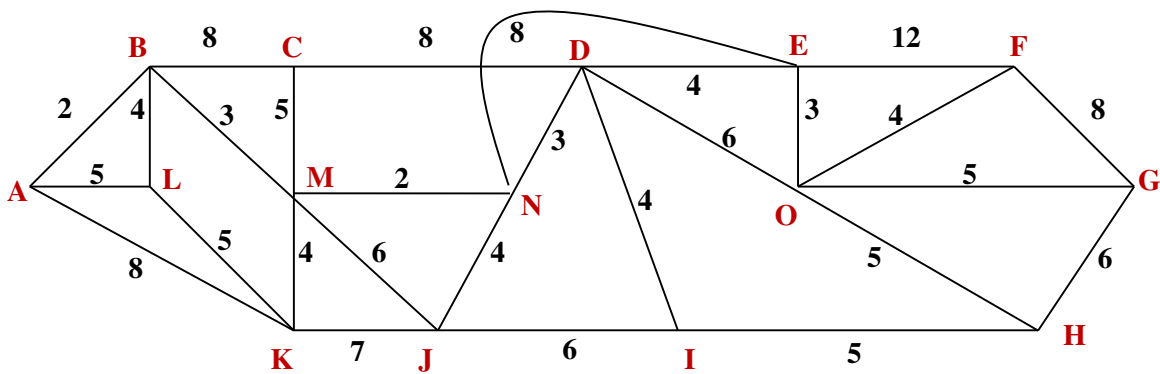
4.4.1. Tìm đường đi ngắn nhất từ đỉnh A đến tất cả các đỉnh còn lại bằng thuật toán Dijkstra cho các đồ thị sau:



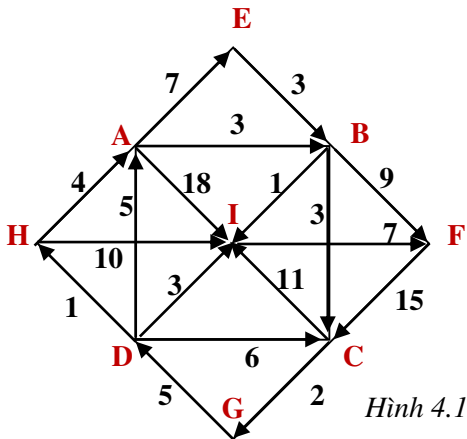
Hình 4.12



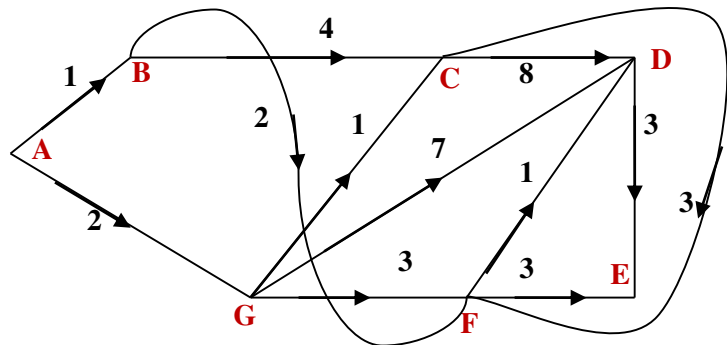
Hình 4.13



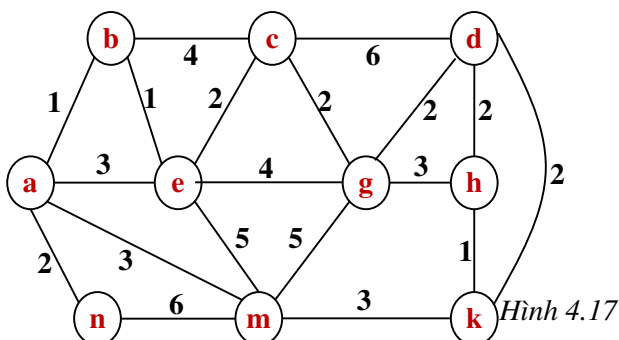
Hình 4.14



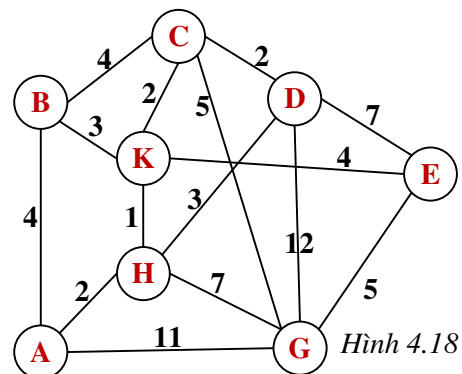
Hình 4.15



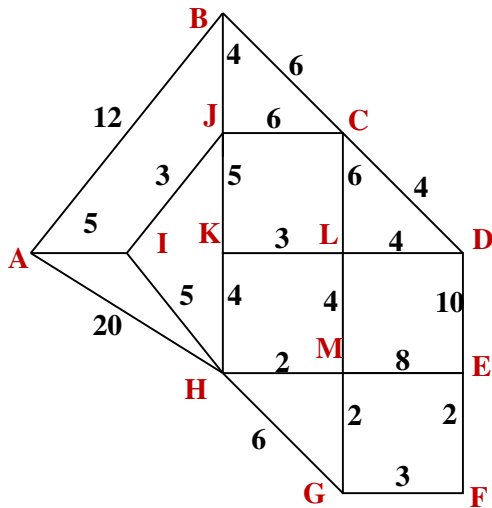
Hình 4.16



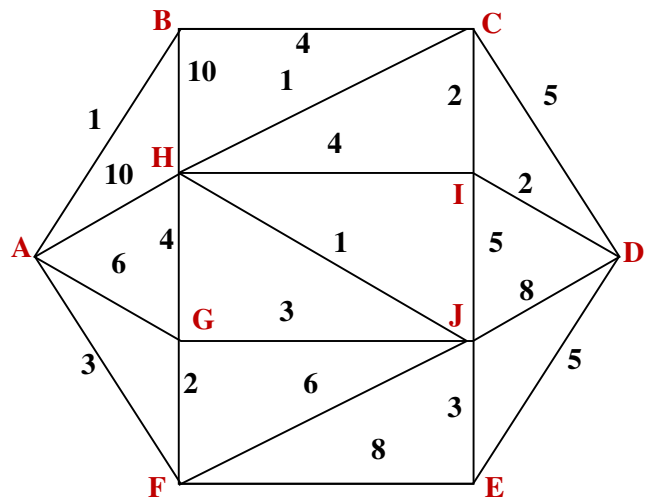
Hình 4.17



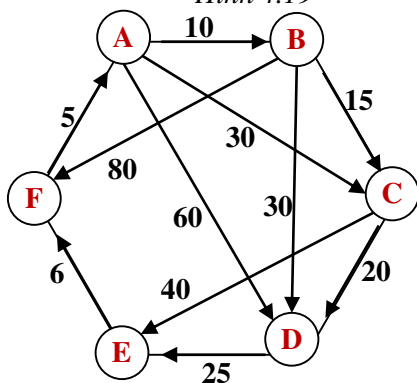
Hình 4.18



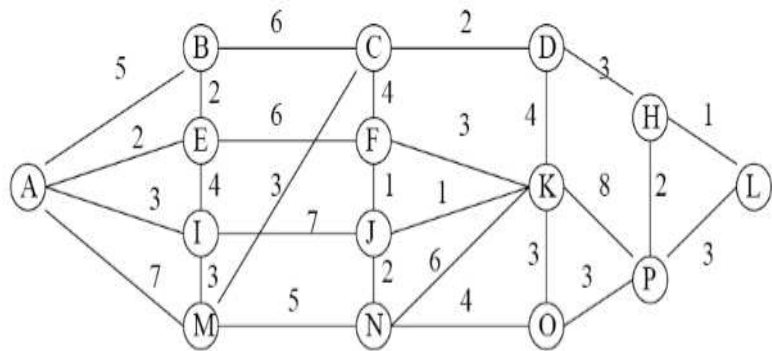
Hình 4.19



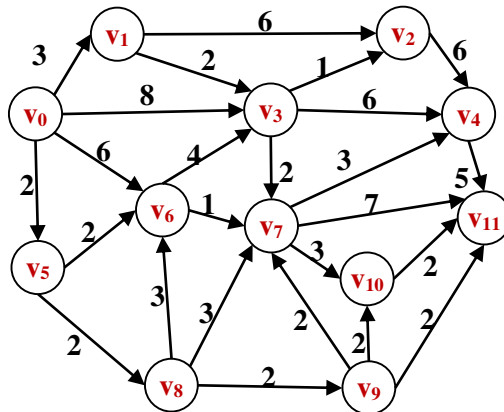
Hình 4.20



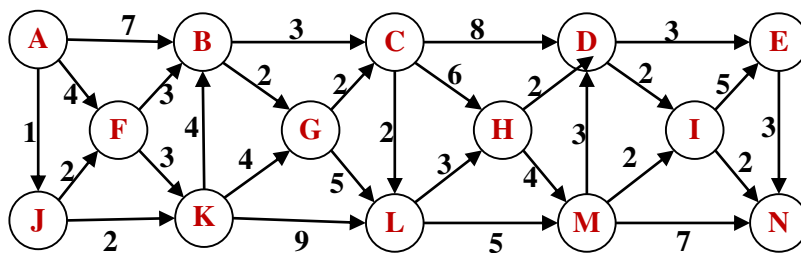
Hình 4.21



Hình 4.22



Hình 4.23



Hình 4.24

4.4.2. Cho các ma trận biểu diễn đồ thị sau. Tìm đường đi ngắn nhất từ đỉnh B đến tất cả các đỉnh còn lại bằng thuật toán Dijkstra:

4.4.2.1.

	A	B	C	D	E	F	G
A		3	6				
B	3		2	4			
C	6	2		1	4	2	
D		4	1		2		4
E			4	2		2	1
F			2		2		4
G				4	1	4	

4.4.2.2.

	A	B	C	D	E	F	G	H
A		2	2	2	4			
B	2					1		
C	2				3			1
D	2				4	3		
E	4		3	4			7	
F		1		3			5	
G					7	5		6
H			1				6	

4.4.2.3.

	A	B	C	D	E	F	G
A		4			2		
B			1	5			9
C	6	2			7	2	3
D			3	2	4	1	
E	2	3	1		4		1
F	2	3		1			2
G		1		1		1	

4.4.2.4.

	A	B	C	D	E	F	G	H
A							12	13
B	31				4			
C							4	1
D			3				8	
E						3		
F	2			8				
G								6
H	2							

4.4.2.5.

	A	B	C	D	E	F	G
A		3	6				
B	3		2	4			
C	6	2		1	4	2	
D		4	1		2		4
E			4	2		2	1
F			2		2		4
G				4	1	4	

4.4.2.6.

	A	B	C	D	E	F	G	H
A		2	2	2	4			
B	2					1		
C	2				3			1
D	2				4	3		
E	4		3	4			7	
F		1		3			5	
G					7	5		6
H			1				6	

4.4.2.7.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
A		5	8	7											
B			5	4											
C					4				7						
D					3	8									
E						5	7		6						
F								8							
G								3	4	8					
H										4	4	6			
I							4	6			6				
J												3	12		
K														5	
L													4	5	5
M															8
N															9
O															

4.4.2.8.

	A	B	C	D	E	F	G
A		3	6				
B	3		2	4			
C	6	2		1	4	2	
D		4	1		2		4
E			4	2		2	1
F			2		2		4
G				4	1	4	

4.4.2.9.

	A	B	C	D	E	F
A		25	45	14	32	24
B	9		16	2	34	23
C	22	11		33	7	
D	23	14	27		20	21
E	14	44	29	46		3
F	25	3	4	7	8	

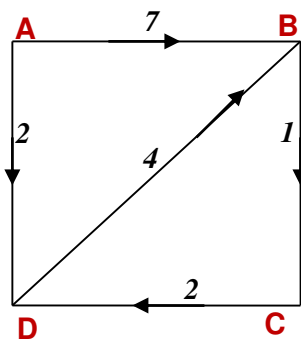
4.4.2.10.

	A	B	C	D	E	F	G
A		5	1			9	
B					3	1	
C		1		9	1		
D	6						7
E			8				2
F	5				6		3
G	4				7	14	

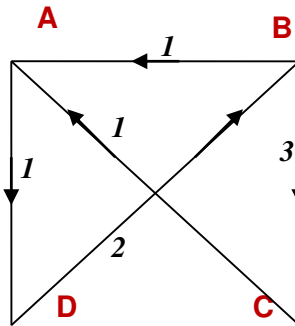
4.4.2.11.

$$D = \begin{bmatrix} 0 & 1 & \infty & \infty & 10 & \infty & \infty & 6 & 3 & \infty \\ 1 & 0 & 4 & \infty & 10 & \infty & \infty & \infty & \infty & \infty \\ \infty & 4 & 0 & 5 & 1 & 2 & \infty & \infty & \infty & \infty \\ \infty & \infty & 5 & 0 & \infty & 2 & 8 & \infty & \infty & 5 \\ 10 & 10 & 1 & \infty & 0 & 4 & 1 & 4 & \infty & \infty \\ \infty & \infty & 2 & 2 & 4 & 0 & 5 & \infty & \infty & \infty \\ \infty & \infty & \infty & 8 & 1 & 5 & 0 & 3 & 6 & 3 \\ 6 & \infty & \infty & \infty & 4 & \infty & 3 & 0 & 2 & \infty \\ 3 & \infty & \infty & \infty & \infty & \infty & 6 & 2 & 0 & 8 \\ \infty & \infty & \infty & 5 & \infty & \infty & 3 & \infty & 8 & 0 \end{bmatrix}$$

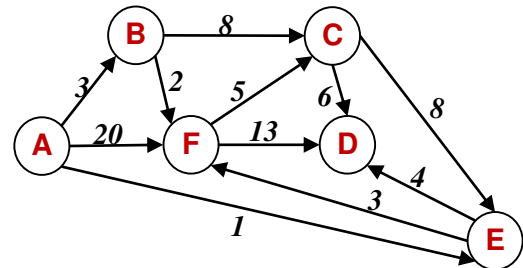
4.4.3. Dùng giải thuật Floyd để tìm đường đi giữa tất cả các đỉnh của những đồ thị sau:



Hình 4.25

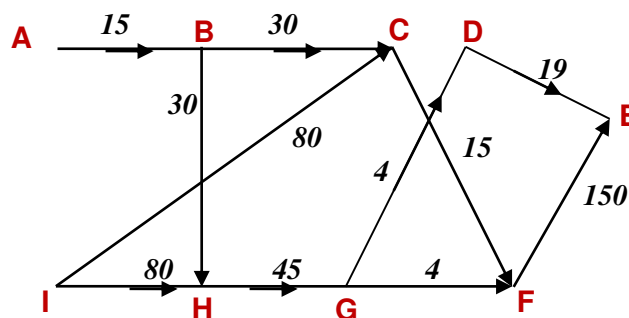


Hình 4.26



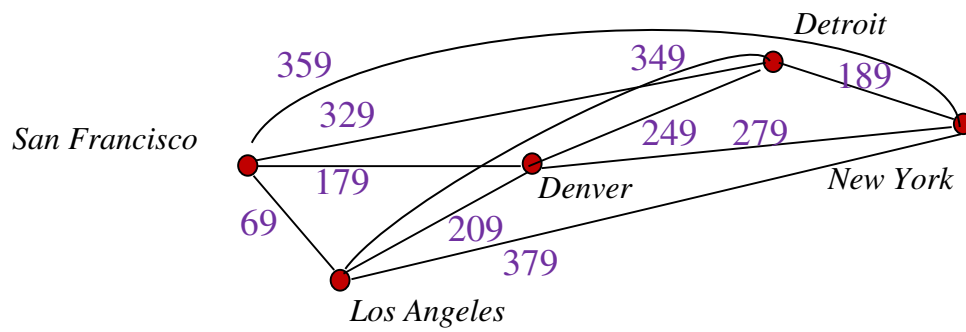
Hình 4.27

4.4.4. Xác định thời gian ngắn nhất để hoàn thành dự án. Biết rằng dự án được mô tả trong đồ thị sau:



Hình 4.28

4.4.5. Cho sơ đồ các thành phố và chi phí (tính bằng USD) để di chuyển (bằng máy bay) giữa các thành phố như sau:



Giả sử một công ty du lịch đặt tại New York muốn tổ chức các tour du lịch từ New York tới các thành phố khác. Hãy chỉ đường cho công ty này sao cho các tour là ít tốn kém về tiền vé nhất.

ĐỒ THỊ PHẪNG

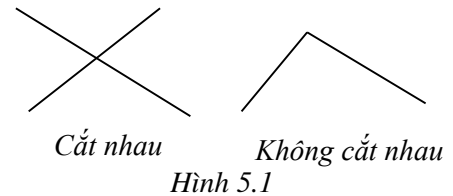
5.1. ĐỒ THỊ PHẪNG

5.1.1. Định nghĩa

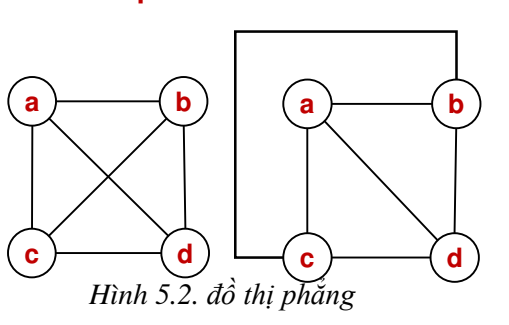
Một đồ thị được gọi là phẳng nếu nó có thể vẽ được trên một mặt phẳng mà không có hai cạnh (bất kỳ) nào cắt nhau (ở một điểm không phải là điểm mút của các cạnh). Hình vẽ như thế gọi là một biểu diễn phẳng của đồ thị.

5.1.2. Quy ước

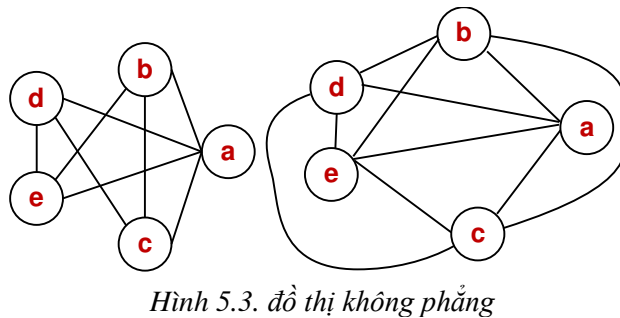
Hai cạnh có chung một đỉnh được xem là không cắt nhau.



5.1.3. Ví dụ



Hình 5.2. đồ thị phẳng



Hình 5.3. đồ thị không phẳng

5.1.4. Đồng phôi

5.1.4.1. Phép biến đổi đồng phôi

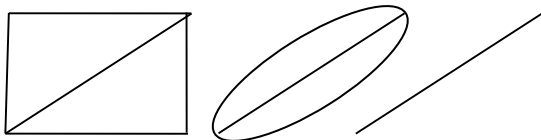
Tính phẳng của đồ thị không thay đổi khi thực hiện một trong các biến đổi sau:

- Thêm vào 1 đỉnh nằm trên 1 cạnh.
- Gộp 2 cạnh có chung đỉnh bậc 2 thành 1 cạnh
- Bỏ đi cạnh khuyên
- Bỏ bớt đi các cạnh song song (chỉ giữ lại 1 cạnh duy nhất)

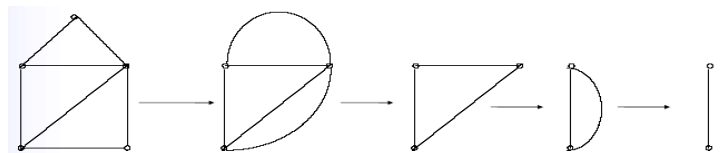
5.1.4.2. Đồ thị đồng phôi

Hai đồ thị được gọi là đồng phôi nếu mỗi đồ thị có được từ đồ thị kia bằng cách thực hiện một dãy các biến đổi đồng phôi

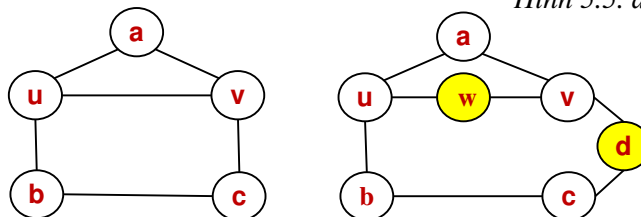
Ví dụ:



Hình 5.4. đồ thị đồng phôi



Hình 5.5. đồ thị đồng phôi



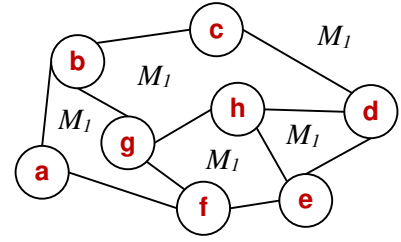
Hình 5.6. đồ thị đồng phôi

5.1.5. Đồ thị phẳng

5.1.5.1. Định lý (Euler, 1752)

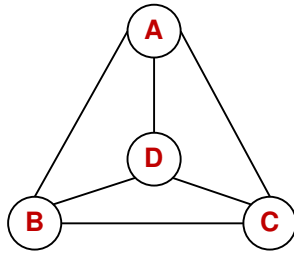
Một đồ thị phẳng liên thông có n đỉnh, p cạnh và d miền thì ta có hệ thức:

$$n - p + d = 2$$

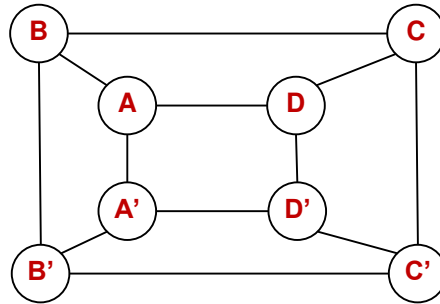


Hình 5.7. Các miền của đồ thị

Hệ thức $n - p + d = 2$ thường gọi là “*hệ thức Euler cho hình đa diện*”, vì được Euler chứng minh đầu tiên cho hình đa diện có n đỉnh, p cạnh và d mặt. Mỗi hình đa diện có thể coi là một đồ thị phẳng. Chẳng hạn hình tứ diện ABCD và hình hộp ABCDA'B'C'D' có thể biểu diễn bằng các đồ thị dưới đây.



Hình 5.8. hình tứ diện ABCD



Hình 5.9. hình hộp ABCDA'B'C'D'

5.1.5.2. Hệ quả

Trong một đồ thị phẳng liên thông tùy ý, luôn tồn tại ít nhất một đỉnh có bậc không vượt quá 5.

5.1.6. Đồ thị không phẳng

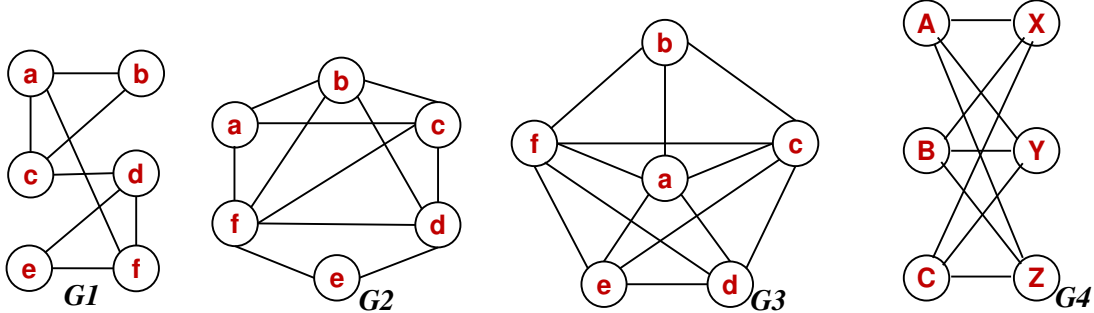
5.1.6.1. Định lý

Đồ thị phân đôi (hai phe) đầy đủ $K_{3,3}$ và đồ thị đầy đủ K_5 là đồ thị không phẳng.

5.1.6.2. Định lý (Kuratowski)

Đồ thị là không phẳng khi và chỉ khi nó chứa một đồ thị con đồng phôi với $K_{3,3}$ hoặc K_5 .

Ví dụ:



Hình 5.10. Đồ thị phẳng G_1, G_2 và đồ thị G_3, G_4 không phẳng

Giải thích:

- G_1 và G_2 là đồ thị phẳng vì:
 - Không chứa đồ thị con $K_{3,3}$ vì có đỉnh bậc 2
 - Không thể chứa đồ thị con K_5 vì có những đỉnh bậc nhỏ hơn 4

- G_3 không phẳng vì nếu xoá đỉnh b cùng các cạnh (b,a) , (b,c) , (b,f) ta được đồ thị con là K_5 .

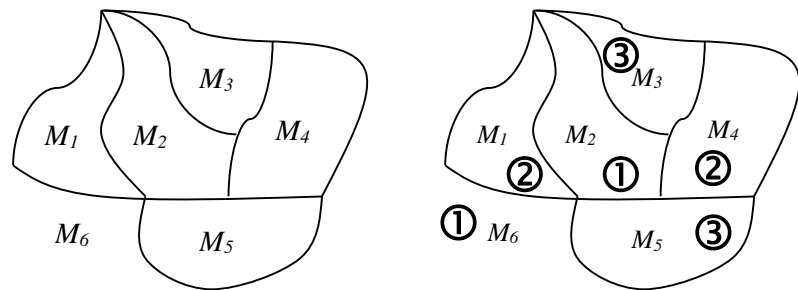
5.2. TÔ MÀU ĐỒ THỊ

5.2.1. Tô màu bản đồ

Mỗi bản đồ có thể coi là một đồ thị phẳng. Trong một bản đồ, ta coi hai miền có chung nhau một đường biên là hai miền kề nhau (hai miền chỉ có chung nhau một điểm biên không được coi là kề nhau). Một bản đồ thường được tô màu, sao cho hai miền kề nhau được tô hai màu khác nhau. Ta gọi một cách tô màu bản đồ như vậy là một cách tô màu đúng.

Để đảm bảo chắc chắn hai miền kề nhau không bao giờ có màu trùng nhau, chúng ta tô mỗi miền bằng một màu khác nhau. Tuy nhiên việc làm đó nói chung là không hợp lý. Nếu bản đồ có nhiều miền thì sẽ rất khó phân biệt những màu gần giống nhau. Do vậy người ta chỉ dùng một số màu cần thiết để tô bản đồ. Một bài toán được đặt ra là: xác định số màu tối thiểu cần có để tô màu đúng một bản đồ.

Ví dụ: Bản đồ trong hình bên có 6 miền, nhưng chỉ cần có 3 màu (vàng - ①, đỏ - ②, xanh - ③) để tô đúng bản đồ này. Chẳng hạn, bản đồ được tô như hình minh họa sau:



Hình 5.11. Sử dụng 3 màu tô

5.2.2. Tô màu đồ thị

5.2.2.1. Đồ thị đối ngẫu

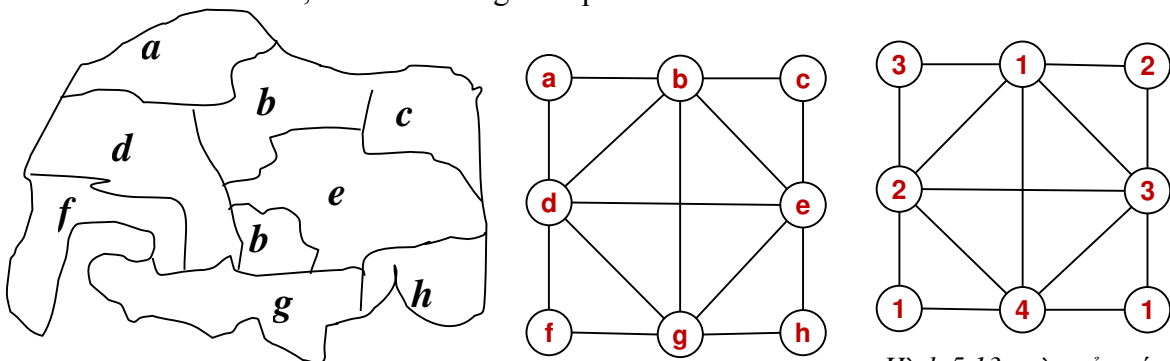
Mỗi bản đồ trên mặt phẳng có thể biểu diễn bằng một đồ thị, trong đó mỗi miền của bản đồ được biểu diễn bằng một đỉnh; các cạnh nối hai đỉnh, nếu các miền được biểu diễn bằng hai đỉnh này là kề nhau. Đồ thị nhận được bằng cách này gọi là đồ thị đối ngẫu của bản đồ đang xét. Rõ ràng mọi bản đồ trên mặt phẳng đều có đồ thị đối ngẫu phẳng. Bài toán tô màu các miền của bản đồ là tương đương với bài toán tô màu các đỉnh của đồ thị đối ngẫu sao cho không có hai đỉnh liền kề nhau có cùng một màu, mà ta gọi là tô màu đúng các đỉnh của đồ thị.

5.2.2.2. Sắc số của đồ thị

Là số màu ít nhất cần dùng để tô màu. Ký hiệu là $\chi(G)$.

5.2.2.3. Ví dụ

Cho bản đồ sau, với nước “b” gồm 2 phần rời nhau.



Hình 5.12. Đồ thị đối ngẫu

Hình 5.13. màu của các đỉnh sau khi thực hiện tô màu

Sử dụng các số nguyên đại diện cho số thứ tự màu cần tô, kết quả tô màu như hình 5.13:

Như vậy $\chi(G) = 4$.

5.2.2.4. Mệnh đề

- Nếu đồ thị G chứa một đồ thị con đồng phôi với đồ thị đầy đủ K_n thì $\chi(G) \geq n$.
- Nếu đơn đồ thị G không chứa chu trình độ dài lẻ thì $\chi(G) = 2$.
- Với mỗi số nguyên dương n , tồn tại một đồ thị không chứa K_3 và có sắc số bằng n .

5.2.2.5. Định lý

- **Định lý 5 màu của Kempe-Heawood:** Mọi đồ thị phẳng đều có thể tô đúng bằng 5 màu.
- **Định lý 4 màu của Appel-Haken:** Mọi đồ thị phẳng đều có thể tô đúng bằng 4 màu.

Định lý Bốn màu đầu tiên được đưa ra như một phỏng đoán vào năm 1850 bởi một sinh viên người Anh tên là F. Guthrie.

Có lẽ một trong những chứng minh sai nổi tiếng nhất trong toán học là chứng minh sai “bài toán bốn màu” được công bố năm 1879 bởi luật sư, nhà toán học nghiệp dư Luân Đôn tên là Alfred Kempe. Nhờ công bố lời giải của “bài toán bốn màu”, Kempe được công nhận là hội viên Hội Khoa học Hoàng gia Anh. Các nhà toán học chấp nhận cách chứng minh của ông ta cho tới 1890, khi Percy Heawood phát hiện ra sai lầm trong chứng minh của Kempe. Mặt khác, dùng phương pháp của Kempe, Heawood đã chứng minh được “bài toán năm màu” (tức là mọi bản đồ có thể tô đúng bằng 5 màu).

Như vậy, Heawood mới giải được “bài toán năm màu”, còn “bài toán bốn màu” vẫn còn đó và là một thách đố đối với các nhà toán học trong suốt gần một thế kỷ.

Trước năm 1976 cũng đã có nhiều chứng minh sai, mà thông thường rất khó tìm thấy chỗ sai, đã được công bố. Hơn thế nữa đã có nhiều cố gắng một cách vô ích để tìm phản ví dụ bằng cách cố vẽ bản đồ cần hơn bốn màu để tô nó. Việc tìm lời giải của “bài toán bốn màu” đã ảnh hưởng đến sự phát triển theo chiều hướng khác nhau của lý thuyết đồ thị.

Năm 1976, khai thác phương pháp của Kempe và nhờ công cụ máy tính điện tử, Kenneth Appel và Wolfgang Haken đã tìm ra lời giải của “bài toán bốn màu”. Chứng minh của họ dựa trên sự phân tích từng trường hợp một cách cẩn thận nhờ máy tính. Họ đã chỉ ra rằng nếu “bài toán bốn màu” là sai thì sẽ có một phản ví dụ huộc một trong gần 2000 loại khác nhau và đã chỉ ra không có loại nào dẫn tới phản ví dụ cả. Trong chứng minh của mình họ đã dùng hơn 1000 giờ máy. Cách chứng minh này đã gây ra nhiều cuộc tranh cãi vì máy tính đã đóng vai trò quan trọng biết bao. Chẳng hạn, liệu có thể có sai lầm trong chương trình và điều đó dẫn tới kết quả sai không? Lý luận của họ có thực sự là một chứng minh hay không, nếu nó phụ thuộc vào thông tin ra từ một máy tính không đáng tin cậy?

5.2.3. Những ứng dụng của bài toán tô màu đồ thị

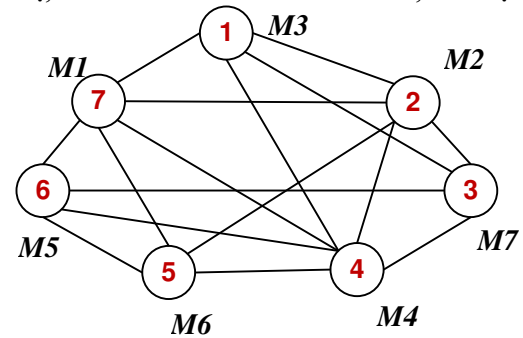
5.2.3.1. Lập lịch thi

Hãy lập lịch thi trong trường đại học sao cho không có sinh viên nào có hai môn thi cùng một lúc.

Chẳng hạn, có 7 môn thi cần xếp lịch. Giả sử các môn học được đánh số từ 1 tới 7 và các cặp môn thi sau có chung sinh viên: 1 và 2, 1 và 3, 1 và 4, 1 và 7, 2 và 3, 2 và 4, 2 và 5, 2 và 7, 3 và 4, 3 và 6, 3 và 7, 4 và 5, 4 và 6, 5 và 6, 5 và 7, 6 và 7.

Có thể giải bài toán lập lịch thi bằng mô hình đồ thị, với các đỉnh là các môn thi, có một cạnh nối hai đỉnh nếu có sinh viên phải thi cả hai môn được biểu diễn bằng hai đỉnh này. Thời gian thi của mỗi môn được biểu thị bằng các màu khác nhau. Như vậy việc lập lịch thi sẽ tương ứng với việc tô màu đồ thị này.

Hình 5.14 biểu diễn đồ thị tương ứng. Việc lập lịch thi chính là việc tô màu đồ thị này. Vì số màu của đồ thị này là 4 nên cần có 4 đợt thi.



Hình 5.14. Lập lịch thi bằng cách tô màu đồ thị

5.2.3.2. Phân chia tần số

Các kênh truyền hình từ số 1 tới số 12 được phân chia cho các đài truyền hình sao cho không có đài phát nào cách nhau không quá 240 km lại dùng cùng một kênh. Có thể chia kênh truyền hình như thế nào bằng mô hình tô màu đồ thị.

Ta xây dựng đồ thị bằng cách coi mỗi đài phát là một đỉnh. Hai đỉnh được nối với nhau bằng một cạnh nếu chúng ở cách nhau không quá 240 km. Việc phân chia kênh tương ứng với việc tô màu đồ thị, trong đó mỗi màu biểu thị một kênh.

5.2.3.3. Các thanh ghi chỉ số

Trong các bộ dịch hiệu quả cao việc thực hiện các vòng lặp được tăng tốc khi các biến dùng thường xuyên được lưu tạm thời trong các thanh ghi chỉ số của bộ xử lý trung tâm (CPU) mà không phải ở trong bộ nhớ thông thường. Với một vòng lặp cho trước cần bao nhiêu thanh ghi chỉ số?

Bài toán này có thể giải bằng mô hình tô màu đồ thị. Để xây dựng mô hình ta coi mỗi đỉnh của đồ thị là một biến trong vòng lặp. Giữa hai đỉnh có một cạnh nếu các biến biểu thị bằng các đỉnh này phải được lưu trong các thanh ghi chỉ số tại cùng thời điểm khi thực hiện vòng lặp. Như vậy số màu của đồ thị chính là số thanh ghi cần có vì những thanh ghi khác nhau được phân cho các biến khi các đỉnh biểu thị các biến này là liên kề trong đồ thị.

5.3. BÀI TẬP

5.3.1. Tìm số đỉnh của đồ thị G khi biết:

- G là một đơn đồ thị phẳng liên thông có 10 mặt, tất cả các đỉnh đều có bậc 4.
- G là đồ thị phẳng liên thông có 30 cạnh và biểu diễn phẳng của đồ thị này chia mặt phẳng thành 20 miền.

5.3.2. Tìm số miền của G khi biết:

- G là đơn đồ thị phẳng liên thông có 8 đỉnh bậc 3.
- G là đơn đồ thị phẳng liên thông có 6 đỉnh, mỗi đỉnh đều có bậc 4.
- G là đơn đồ thị phẳng liên thông có 9 đỉnh, bậc các đỉnh là 2, 2, 2, 3, 3, 3, 4, 4, 5.

5.3.3. Tìm số đỉnh, số cạnh và đai của:

5.3.3.1. K_n ;

5.3.3.2. $K_{m,n}$.

5.3.4. Chứng minh rằng:

5.3.4.1. K_n là phẳng khi và chỉ khi $n \leq 4$.

5.3.4.2. $K_{m,n}$ là phẳng khi và chỉ khi $m \leq 2$ hay $n \leq 2$.

5.3.5. Đồ thị nào trong các đồ thị không phẳng sau đây có tính chất: Bỏ một đỉnh bất kỳ và các cạnh liên thuộc của nó tạo ra một đồ thị phẳng.

5.3.5.1. K_5 ;

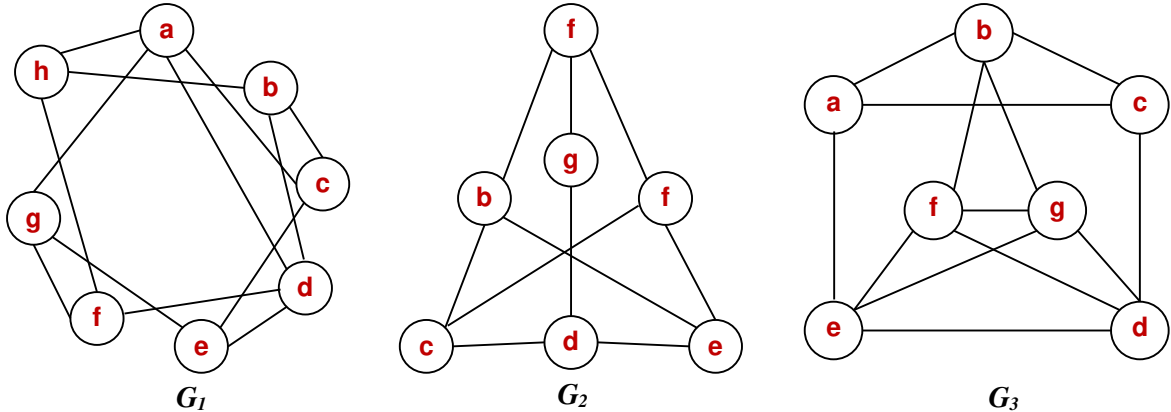
5.3.5.2. K_6 ;

5.3.5.3. $K_{3,3}$.

5.3.6. Cho G là một đơn đồ thị phẳng liên thông có n đỉnh và m cạnh, trong đó $n \geq 3$. Chứng minh rằng: $m \leq 3n - 6$.

5.3.7. Cho $G=(V, E)$ là một đơn đồ thị vô hướng, R là một quan hệ trên V sao cho $\forall u, v \in V, u R v$ khi và chỉ khi có đường đi từ u đến v hoặc $u=v$. Chứng minh R là quan hệ tương đương.

5.3.8. Trong các đồ thị ở hình dưới đây, đồ thị nào là phẳng, đồ thị nào không phẳng? Nếu đồ thị là phẳng thì có thể kẻ thêm ít nhất là bao nhiêu cạnh để được đồ thị không phẳng?



Hình 5.15

5.3.9. Chứng minh rằng:

5.3.9.1. Một đồ thị phẳng có thể tô đúng các đỉnh bằng hai màu khi và chỉ khi đó là đồ thị phân đôi.

5.3.9.2. Một đồ thị phẳng có thể tô đúng các miền bằng hai màu khi và chỉ khi đó là đồ thị Euler.

5.3.10. Tìm sắc số của các đồ thị cho trong hình 5.14.

5.3.11. Tìm sắc số của các đồ thị K_n , $K_{m,n}$, C_n , và W_n .

5.3.12. Khoa Toán có 6 hội đồng hợp mỗi tháng một lần. Cần có bao nhiêu thời điểm họp khác nhau để đảm bảo rằng không ai bị xếp lịch họp hai hội đồng cùng một lúc, nếu các hội đồng là:

$$H_1 = \{K, L, P\}, H_2 = \{L, M, T\}, H_3 = \{H, K, P\}.$$

$$H_4 = \{H, L, P\}, H_5 = \{L, M, K\}, H_6 = \{H, T, P\}.$$

5.3.13. Chứng minh rằng một đơn đồ thị phẳng có 8 đỉnh và 13 cạnh không thể được tô đúng bằng hai màu.

5.3.14. Chứng minh rằng nếu G là một đơn đồ thị phẳng có ít hơn 12 đỉnh thì tồn tại trong G một đỉnh có bậc ≤ 4 . Từ đó hãy suy ra rằng đồ thị G có thể tô đúng bằng 4 màu.

5.3.15. Cần lập lịch thi kết thúc cho các học phần có mã số 1007, 3137, 3203, 3261, 4115, 4118. Do điều kiện tiên quyết, các cặp học phần sau đây không có sinh viên nào đồng thời cùng thi:

- 1007-3137	- 1007-3261	- 3203-3261	- 1007-4115	- 3137-4115
- 1007-3203	- 3203-4115	- 3261-4115	- 1007-4118	- 3137-4118

Cho biết lịch thi gồm ít nhất bao nhiêu ngày, sao cho lịch thi phải bảo đảm mỗi sinh viên trong một ngày không được thi quá 1 môn.

5.3.16. Cần lập lịch thi lại cho những sinh viên với mã số sinh viên thi lại và mã của những môn thi lại cho trong bảng sau, sao cho số buổi tổ chức thi là ít nhất:

Mã MH MãSV	1007	2010	1091	4200	1723	4318	1102	1091	4118	2031
12013579	X			X						X
13012235	X	X					X			
12013229			X	X				X		X
12003135				X						X
11013449					X	X				
13013521	X					X			X	
10013123			X				X			
12013654								X		X
11013246		X		X	X				X	

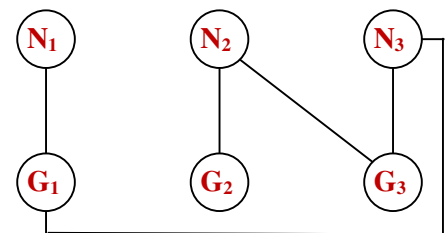
5.3.17. Từ xa xưa đã lưu truyền một bài toán cổ “Ba nhà, ba giếng”: Có ba nhà (N), mỗi nhà có 1 giếng nước riêng (G), với các điều kiện hiện tại như sau:

- Chưa có đường nối thẳng các nhà với nhau.
- Chưa có đường nối các giếng với nhau.
- Chưa có đường nối giữa các nhà với giếng nước riêng của mỗi nhà.

Có lần bất hoà với nhau, họ tìm cách làm các đường khác đến giếng của mình sao cho các đường này không giao với các đường đã có trước.

Họ có thực hiện được ý định đó không? Nếu được, hãy trình bày đường đi?

Ngược lại (nếu không được), sau khi đã làm được các đường nối trực tiếp từ nhà đến giếng riêng của mỗi nhà (không cắt các đường khác), hãy vẽ thêm duy nhất 1 đường nối để giải quyết nhu cầu đường đi.



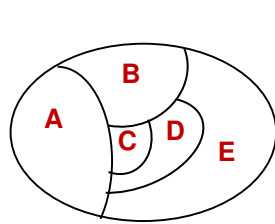
Hình 5.16

5.3.18. Khoảng cách giữa sáu đài truyền hình được cho trong bảng dưới đây. Mỗi đài sẽ được cấp 1 kênh để phát sóng. Hãy tìm số kênh ít nhất cần phát, biết rằng hai đài phát cách nhau không quá 150 dặm sẽ không được cấp phát chung một kênh.

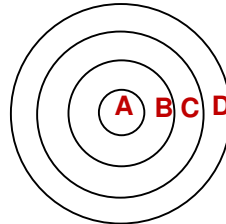
	1	2	3	4	5	6
1		85	175	200	20	100
2	85		125	175	100	160
3	175	125		100	200	250
4	200	175	100		210	220
5	10	100	200	210		100
6	100	160	250	220	100	

5.3.19. Xây dựng đồ thị mô tả cho cả 6 loài chim trong đó chim *hét* cạnh tranh với chim *cỏ đỏ* và chim *giẻ cùi xanh*, chim *cỏ đỏ* cũng cạnh tranh với chim *nhại*, chim *nhại* cạnh tranh với chim *giẻ cùi* và chim *bỏ hạt* cạnh tranh với chim *gỗ kiến*.

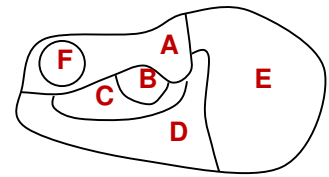
5.3.20. Xây dựng đồ thị cho các bản đồ dưới đây. Từ đó cho biết các bản đồ trên có thể được tô bằng tối thiểu bao nhiêu màu sao cho hai vùng giáp nhau phải có màu khác nhau.



Hình 5.16



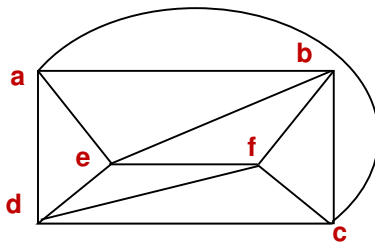
Hình 5.17



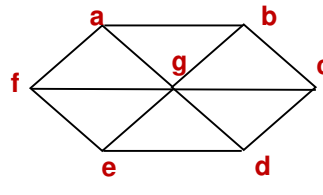
Hình 5.18

5.3.21. Lần lượt tìm số màu của các đồ thị dưới đây trong 2 trường hợp:

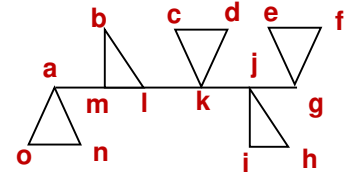
- Tô màu cho đỉnh.
- Tô màu cho cạnh.



Hình 5.19



Hình 5.20



Hình 5.21



CÂY (TREE)

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.

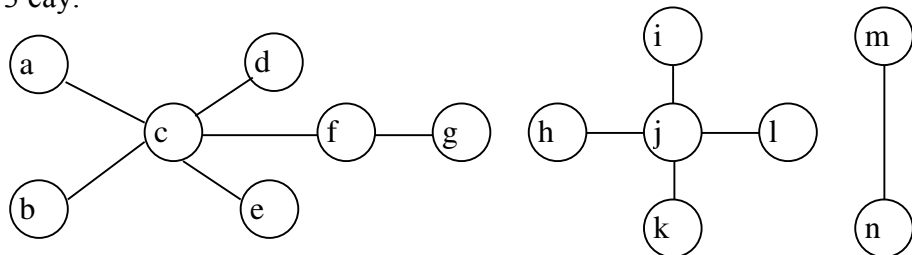
6.1. ĐỊNH NGHĨA VÀ CÁC TÍNH CHẤT CƠ BẢN

6.1.1. Định nghĩa

- 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 1: Rừng sau có 3 cây:

Hình 6.1.



6.1.2. Mệnh đề

Nếu T là một cây có n đỉnh thì T có ít nhất hai đỉnh treo.

6.1.3. Định lý

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

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

6.2. CÂY KHUNG VÀ BÀI TOÁN TÌM CÂY KHUNG NHỎ NHẤT

6.2.1. Định nghĩa

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 .

6.2.2. 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.

6.2.2.1. 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. Mặt 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.

6.2.2.2. 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 nối 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.

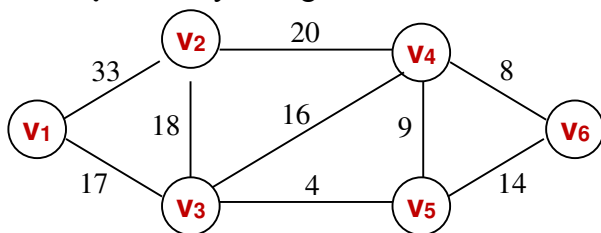
Bài toán tìm cây khung nhỏ nhất đã có những thuật toán rất hiệu quả để giải chúng. Trong tài liệu này sẽ xét hai trong số những thuật toán như vậy: thuật toán Kruskal và thuật toán Prim.

6.2.3. Thuật toán Kruskal

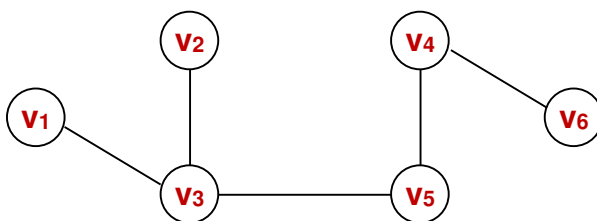
Thuật toán sẽ xây dựng tập cạnh E_T của cây khung nhỏ nhất $T=(V_T, E_T)$ theo từng bước. Trước hết sắp xếp các cạnh của đồ thị G theo thứ tự không giảm của trọng số. Bắt đầu từ $E_T=\emptyset$, ở mỗi bước ta sẽ lần lượt duyệt trong danh sách cạnh đã sắp xếp, từ cạnh có độ dài nhỏ đến cạnh có độ dài 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:

- (i). Bắt đầu từ đồ thị rỗng T có n đỉnh.
- (ii). Sắp xếp các cạnh của G theo thứ tự không giảm của trọng số.
- (iii). 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 .
- (iv). 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:



Hình 5.2. Cây ban đầu



Hình 5.3. Cây khung nhỏ nhất

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)
4	8	9	14	16	17	18	20	33

Đầu tiên, 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:

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

6.2.4. 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.

B1: $V_T := \{v^*\}$, trong đó v^* là đỉnh tùy ý của đồ thị G .

$$E_T := \emptyset.$$

B2: 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 $[0, \infty]$.

B3: 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.

B4: 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 2: 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, với gốc là A:

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.

∞	15	16	19	23	20	32	18
15	∞	33	13	34	19	20	12
16	33	∞	13	29	21	20	19
19	13	13	∞	22	30	21	11
23	34	29	22	∞	34	23	21
20	19	21	30	34	∞	17	18
32	20	20	21	23	17	∞	14
18	12	19	11	21	18	14	∞

V.lặp	A	B	C	D	E	F	H	I	V _T	E _T
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:

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

$$(A,B) + (B,I) + (I,D) + (D,C) + (I,H) + (H,F) + (I,E)$$

6.3. SO SÁNH HAI THUẬT TOÁN Prim & Kruskal

(i). Xét các cạnh đưa vào cây

- **Prime**: thực hiện việc mở rộng tập đã xét (ban đầu chỉ gồm một đỉnh, 0 cạnh thành n đỉnh, n-1 cạnh) dựa trên các cạnh ngắn nhất nối giữa tập đã xét và tập chưa xét. Tư tưởng chính là thêm vào các cạnh ngắn nhất sao cho không tạo ra chu trình. Như vậy, có trường hợp một cạnh sẽ phải xét đi xét nhiều lần rồi mới được chọn, thậm chí không hề được chọn.
- **Kruskal**: cũng thêm lần lượt các cạnh vào đồ thị, theo thứ tự từ trọng nhỏ nhất đến trọng lớn nhất (như vậy mỗi cạnh sẽ chỉ được duyệt một lần duy nhất). Ta chỉ bỏ sung cạnh vào cây khung nếu việc thêm cạnh này không làm phát sinh ra chu trình.

(ii). Kiểm tra tính liên thông của đồ thị

- **Prim**: cần kiểm tra đồ thị liên thông trước khi thi hành thuật toán.
- **Kruskal**: không cần kiểm tra đồ thị liên thông trước khi thi hành thuật toán. Nếu quá trình thi hành thuật toán tìm được cây khung/bao trùm thì đồ thị liên thông và ngược lại là đồ thị không liên thông.

(iii). Chi phí cho việc kiểm tra chu trình

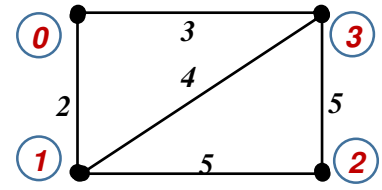
- **Prime**: mở rộng tập đỉnh đã xét (mỗi lần 1 đỉnh nên không cần kiểm tra chu trình).
- **Kruskal**: kiểm tra nếu khi thêm cạnh đang xét vào cây khung mà không làm phát sinh chu trình thì sẽ chọn cạnh này. Việc kiểm tra chu trình có khả năng gây tốn chi phí

Gợi ý cải tiến giảm chi phí kiểm tra chu trình:

- Dùng một mảng Nhãn có kích thước bằng số đỉnh của đồ thị. Giá trị Nhãn[i] tại đỉnh i được khởi tạo bằng với chính chỉ số i của đỉnh.

Ví dụ: Với đồ thị G ở trên ta có

Đỉnh	0	1	2	3
Nhãn	0	1	2	3



- Khi chọn một cạnh (u,v) được thêm vào cây khung/cây bao trùm, ta sửa **nhãn của tất cả các đỉnh** có **cùng giá trị** với **nhãn của đỉnh v** thành **nhãn của đỉnh u**.

Ví dụ:

- Sau khi thêm cạnh (0,1) vào cây khung/cây bao trùm hay tập T, ta sửa nhãn của tất cả các đỉnh có cùng giá trị với nhãn của đỉnh 1 (là 1) thành nhãn của đỉnh 0 (là 0).

Đỉnh	0	1	2	3
Nhãn	0	0	2	3

- Sau đó, khi chọn tiếp cạnh (0, 3), ta sửa nhãn của đỉnh 3 (đang có giá trị là 3) để có cùng giá trị với nhãn của đỉnh 0 (là 0).

Đỉnh	0	1	2	3
Nhãn	0	0	2	0

- Sau đó, khi chọn cạnh để thêm vào cây khung/cây bao trùm thì chỉ chọn cạnh có nhãn hai đỉnh là khác nhau sẽ không tạo thành chu trình.

Ví dụ:

- Khi ta xét tới cạnh (1,3): ta không chọn cạnh này vì đỉnh 1 và đỉnh 3 có cùng nhãn là 0
- Chọn tiếp cạnh (1,2): 2 đỉnh 1 và 2 có nhãn khác nhau nên cạnh (1,2) sẽ được chọn đưa vào cây khung/cây bao trùm.

Đỉnh	0	1	2	3
Nhãn	0	0	0	0

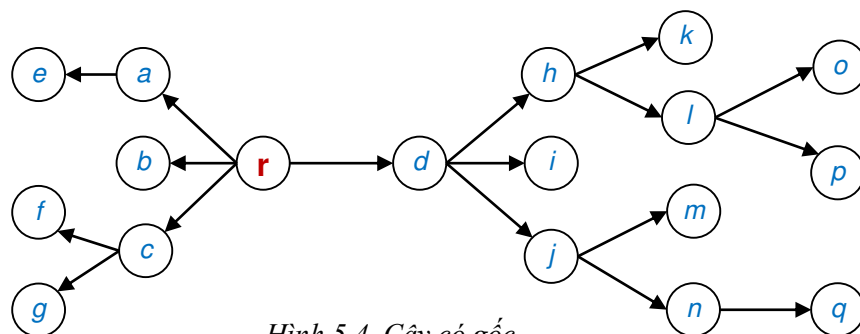
Nhận xét: Sau khi ta đã chọn đủ n-1 cạnh, mảng nhãn chỉ chứa một giá trị duy nhất.

6.4. CÂY CÓ GỐC

6.4.1. Định nghĩa về cây có gốc

- Cây có hướng là đồ thị có hướng mà đồ thị vô hướng nền của nó là một cây.
- Cây có gốc là một cây có hướng, trong đó có một đỉnh đặc biệt, gọi là gốc, từ gốc có đường đi đến mọi đỉnh khác của cây.

Ví dụ:



Hình 5.4. Cây có gốc

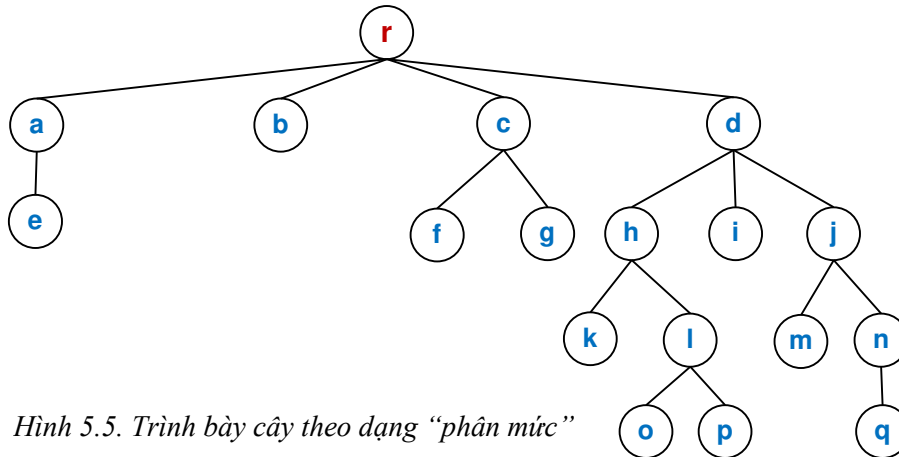
Trong cây có gốc thì gốc r có bậc vào bằng 0, còn tất cả các đỉnh khác đều có bậc vào bằng 1.

Một cây có gốc thường được vẽ với gốc r ở trên cùng và cây phát triển từ trên xuống, gốc r gọi là đỉnh mức 0. Các đỉnh kề với r được xếp ở phía dưới và gọi là đỉnh mức 1. Đỉnh ngay dưới đỉnh mức 1 là đỉnh mức 2, ...

Tổng quát, trong một cây có gốc thì v là đỉnh mức k khi và chỉ khi đường đi từ r đến v có độ dài bằng k .

Mức lớn nhất của một đỉnh bất kỳ trong cây gọi là chiều cao của cây.

Cây có gốc ở hình trên thường được vẽ như trong hình dưới đây để làm rõ mức của các đỉnh.



Hình 5.5. Trình bày cây theo dạng “phân mức”

Trong cây có gốc, mọi cung đều có hướng từ trên xuống, vì vậy vẽ mũi tên để chỉ hướng đi là không cần thiết; do đó, người ta thường vẽ các cây có gốc như là cây nền của nó.

6.4.2. Đỉnh ngoài – đỉnh trong

- Đỉnh treo v_n là đỉnh không có con; đỉnh treo cũng gọi là lá hay đỉnh ngoài;
- Một đỉnh không phải lá là một đỉnh trong.

6.4.3. Cây nhị phân

- Một cây có gốc T được gọi là cây m -phân nếu mỗi đỉnh của T có nhiều nhất là m con. Với $m=2$, ta có một cây nhị phân.
- Trong một cây nhị phân, mỗi con được chỉ rõ là con bên trái hay con bên phải; con bên trái (hoặc bên phải) được vẽ phía dưới và bên trái (hoặc bên phải) của cha.
- Cây có gốc T được gọi là một cây m -phân đầy đủ nếu mỗi đỉnh trong của T đều có m con.

6.4.4. Mệnh đề về cây m phân

- Một cây m -phân đầy đủ có i đỉnh trong thì có $mi+1$ đỉnh và có $(m-1)i+1$ lá.
- Một cây m -phân có chiều cao h thì có nhiều nhất là m^h lá.
- Một cây m -phân có l lá thì có chiều cao $h \geq \lceil \log_m l \rceil$.

6.5. DUYỆT CÂY NHỊ PHÂN

6.5.1. Định nghĩa

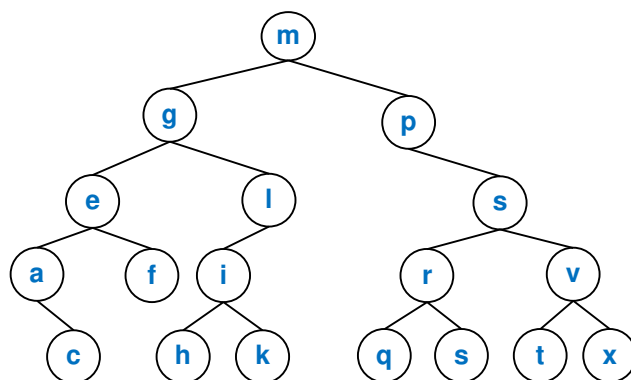
Trong nhiều trường hợp, ta cần “thăm” mọi đỉnh của một cây nhị phân, mỗi đỉnh chỉ một lần. Ta gọi đó là việc duyệt cây nhị phân hay đọc cây nhị phân.

Có nhiều thuật toán duyệt cây nhị phân, các thuật toán đó khác nhau chủ yếu ở thứ tự thăm các đỉnh.

Cây nhị phân T có gốc r được ký hiệu là $T(r)$. Giả sử r có con bên trái là u , con bên phải là v . Cây có gốc u và các đỉnh khác là mọi dòng dõi của u trong T gọi là cây con bên trái của T , ký hiệu $T(u)$. Tương tự, ta có cây con bên phải $T(v)$ của T . Một cây $T(r)$ có thể không có cây con bên trái hay bên phải.

Sau đây là ba trong các thuật toán duyệt cây nhị phân $T(r)$. Các thuật toán đều được trình bày đệ quy. Chú ý rằng khi cây $T(x)$ chỉ là một đỉnh x thì “duyệt $T(x)$ ” có nghĩa là “thăm đỉnh x ”.

Ví dụ :



Hình 5.6.

6.5.2. Các thuật toán duyệt cây nhị phân

6.5.2.1. Thuật toán Node- Left-Right

- (i). Thăm gốc r .
- (ii). Duyệt cây con bên trái của $T(r)$.
- (iii). Duyệt cây con bên phải của $T(r)$.

Kết quả duyệt cây $T(a)$ theo thuật toán Node- Left-Right là:

$a, b, d, g, l, h, e, i, m, n, c, f, j, o, p, k, q, s.$

6.5.2.2. Thuật toán Left- Node- Right

- (i). Duyệt cây con bên trái của $T(r)$.
- (ii). Thăm gốc r .
- (iii). Duyệt cây con bên phải của $T(r)$.

Kết quả duyệt cây $T(a)$ theo thuật toán Left-Node- Right là:

$g, l, d, h, b, m, i, n, e, a, c, o, j, p, f, q, k, s.$

6.5.2.3. Thuật toán Left-Right-Node

- (i). Duyệt cây con bên trái của $T(r)$.
- (ii). Duyệt cây con bên phải của $T(r)$.
- (iii). Thăm gốc r .

Kết quả duyệt cây $T(a)$ theo Left-Node- Right là:

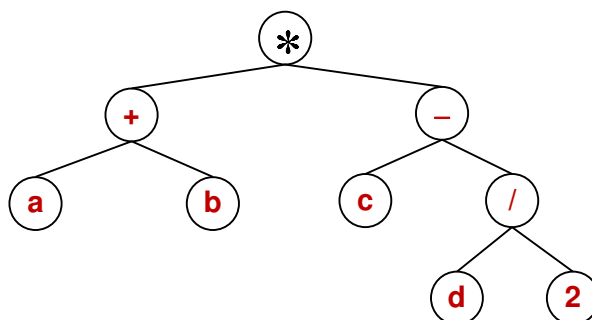
$l, g, h, d, m, n, i, e, b, o, p, j, q, s, k, f, c, a.$

6.5.3. Ký pháp Ba Lan

Xét biểu thức đại số sau đây:

$$(a+b)(c-\frac{d}{2}) \quad (1)$$

Ta vẽ một cây nhị phân như hình dưới đây, trong đó mỗi đỉnh trong mang dấu của một phép tính trong (1), gốc của cây mang phép tính sau cùng trong (1), ở đây là dấu nhân, ký hiệu là $*$, mỗi lá mang một số hoặc một chữ đại diện cho số.



Hình 5.7. Biểu diễn biểu thức $(a+b)(c-d/2)$ dưới dạng cây nhị phân

Duyệt cây nhị phân trong hình trên theo thứ tự *Left-Node-Right* là:

$$a + b * c - d / 2 \quad (2)$$

và đây là biểu thức (1) đã bỏ đi các dấu ngoặc.

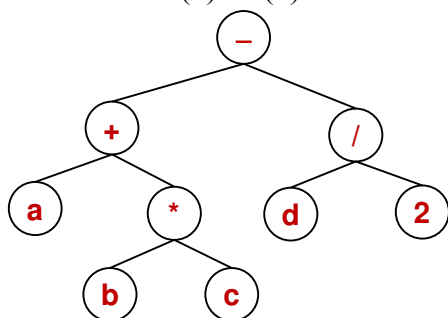
Ta nói rằng biểu thức (1) được biểu diễn bằng cây nhị phân $T(*)$ trong hình trên, hay cây nhị phân $T(*)$ này tương ứng với biểu thức (1).

Ta biết rằng các dấu ngoặc trong (1) là rất cần thiết, vì (2) có thể hiểu theo nhiều cách khác (1), chẳng hạn như

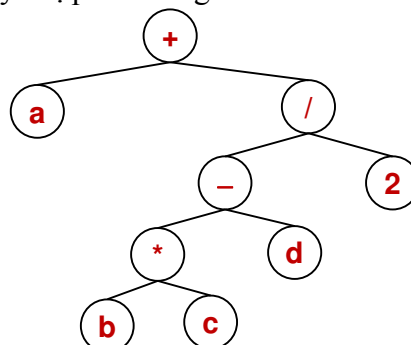
$$(a + b * c) - d / 2 \quad (3)$$

$$\text{hoặc là } a + (b * c - d) / 2 \quad (4)$$

Các biểu thức (3) và (4) có thể biểu diễn bằng cây nhị phân trong các hình sau.



Hình 5.8. Biểu thức (3) $= (a + b * c) - d / 2$ sau khi đưa vào cây nhị phân



Hình 5.9. Biểu thức (4) $= a + (b * c - d) / 2$ sau khi đưa vào cây nhị phân

Đối với cây trong hình 5.7, nếu duyệt theo *Node-Left-Right*, ta có

$$* + a b - c / d 2 \quad (5)$$

và nếu duyệt theo *Left-Right-Node*, ta có:

$$a b + c d 2 / - * \quad (6)$$

Có thể chứng minh được rằng (5) hoặc (6) xác định duy nhất cây nhị phân trong hình thứ nhất, do đó xác định duy nhất biểu thức (1) mà không cần dấu ngoặc. Chẳng hạn cây nhị phân trên hình 5.8 được duyệt theo *Node-Left-Right* là

$$- + a \star b c / d 2 \text{ khác với (5).}$$

và được duyệt theo *Left-Right-Node* là

$$a b c * + d 2 / - \text{ khác với (6).}$$

Vì vậy, nếu ta viết các biểu thức trong đại số, trong logic bằng cách duyệt cây tương ứng theo *Node-Left-Right* hoặc *Left-Right-Node* thì ta không cần dùng các dấu ngoặc mà không sợ hiểu nhầm.

Người ta gọi cách viết biểu thức theo *Node-Left-Right* là **ký pháp Ba Lan**, còn cách viết theo *Left-Right-Node* là **ký pháp Ba Lan đảo**, để ghi nhớ đóng góp của nhà toán học và logic học Ba Lan Lukasiewicz (1878-1956) trong vấn đề này.

Việc chuyển một biểu thức viết theo ký pháp quen thuộc (có dấu ngoặc) sang dạng ký pháp Ba Lan hay ký pháp Ba Lan đảo hoặc ngược lại, có thể thực hiện bằng cách vẽ cây nhị phân tương ứng, như đã làm đối với biểu thức (1). Nhưng thay vì vẽ cây nhị phân, ta có thể xem xét để xác định dần các công thức bộ phận của công thức đã cho. Chẳng hạn cho biểu thức viết theo ký pháp Ba Lan là

$$- \star \uparrow / - - a b \star 5 c 2 3 \uparrow - c d 2 \star - - a c d / \uparrow - b \star 3 d 3 5$$

Trước hết, chú ý rằng các phép toán $+$, $-$, $*$, $/$, \uparrow đều là các phép toán hai ngôi, vì vậy trong cây nhị phân tương ứng, các đỉnh mang dấu các phép toán đều là đỉnh trong và có hai con. Các chữ và số đều đặt ở lá. Theo ký pháp Ba Lan (tương ứng Ba Lan đảo) thì $T a b$ (tương ứng $a b T$) có nghĩa là $a T b$, với T là một trong các phép toán $+$, $-$, $*$, $/$, \uparrow .

$$\begin{aligned} & - \star \uparrow / - - \underbrace{a b}_{a-b} \star \underbrace{5 c}_{5c} 2 3 \uparrow - \underbrace{c d}_{c-d} 2 \star - - \underbrace{a c}_{a-c} d / \uparrow - b \star \underbrace{3 d}_{3d} 3 5 \\ & - \star \uparrow / - \underbrace{(a-b) 5 c}_{a-b-5c} 2 3 \uparrow \underbrace{(c-d) 2}_{(c-d)^2} \star - \underbrace{(a-c) d}_{a-c-d} / \uparrow - \underbrace{b (3d)}_{b-3d} 3 5 \\ & - \star \uparrow / \underbrace{(a-b-5c) 2}_{\frac{a-b-5c}{2}} 3 \underbrace{(c-d)^2}_{(c-d)^2} \star \underbrace{(a-c-d)}_{(a-c-d)} / \uparrow \underbrace{(b-3d) 3}_{(b-3d)^3} 5 \\ & - \star \uparrow \underbrace{\frac{a-b-5c}{2} 3}_{\left(\frac{a-b-5c}{2}\right)^3} \underbrace{(c-d)^2}_{(c-d)^2} \star \underbrace{(a-c-d)}_{(a-c-d)} / \underbrace{(b-3d)^3}_{\frac{(b-3d)^3}{5}} 5 \\ & - \star \underbrace{\left(\frac{a-b-5c}{2}\right)^3}_{\left(\frac{a-b-5c}{2}\right)^3} \underbrace{(c-d)^2}_{(c-d)^2} \star \underbrace{(a-c-d)}_{(a-c-d)} \underbrace{\frac{(b-3d)^3}{5}}_{\frac{(b-3d)^3}{5}} \\ & \left(\frac{a-b-5c}{2}\right)^3 (c-d)^2 - (a-c-d) \frac{(b-3d)^3}{5} \end{aligned}$$

6.6. BÀI TẬP

6.6.1. Về tất cả các cây (không đẳng cấu) có:

6.5.1.1. 4 đỉnh

6.5.1.2. 5 đỉnh

6.5.1.3. 6 đỉnh

6.6.2. Một cây có n_2 đỉnh bậc 2, n_3 đỉnh bậc 3, ..., n_k đỉnh bậc k. Hỏi có bao nhiêu đỉnh bậc 1?

6.6.3. Tìm số tối đa các đỉnh của một cây m-phân có chiều cao h.

6.6.4. Có thể tìm được một cây có 8 đỉnh và thỏa điều kiện dưới đây hay không? Nếu có, vẽ cây đó ra, nếu không, giải thích tại sao:

6.6.4.1. Mọi đỉnh đều có bậc 1.

6.6.4.2. Mọi đỉnh đều có bậc 2.

6.5.4.3. Có 6 đỉnh bậc 2 và 2 đỉnh bậc 1.

6.6.4.4. Có đỉnh bậc 7 và 7 đỉnh bậc 1.

6.6.5. Chứng minh hoặc bác bỏ các mệnh đề sau đây.

6.6.5.1. Trong một cây, đỉnh nào cũng là đỉnh cắt.

6.6.5.2. Một cây có số đỉnh không nhỏ hơn 3 thì có nhiều đỉnh cắt hơn là cầu.

6.6.6. Có bốn đội bóng đá A, B, C, D lọt vào vòng bán kết trong giải các đội mạnh khu vực. Có mấy dự đoán xếp hạng như sau:

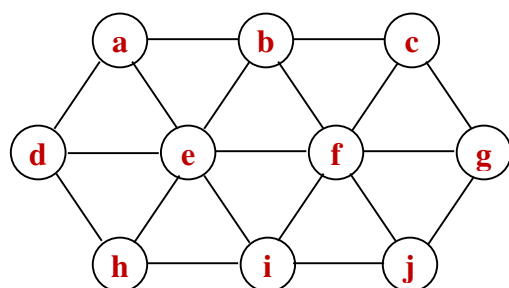
6.6.6.1. Đội B vô địch, đội D nhì.

6.6.6.2. Đội B nhì, đội C ba.

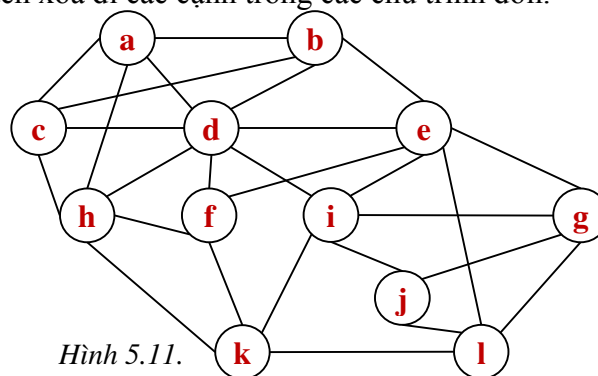
6.6.6.3. Đội A nhì, đội C tư.

Biết rằng mỗi dự đoán trên đúng về một đội. Hãy cho biết kết quả xếp hạng của các đội.

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



Hình 5.10.



Hình 5.11.

6.6.8. Cây Fibonacci có gốc T_n được định nghĩa bằng hồi quy như sau. T_1 và T_2 đều là cây có gốc chỉ gồm một đỉnh và với $n=3,4, \dots$ cây có gốc T_n được xây dựng từ gốc với T_{n-1} như là cây con bên trái và T_{n-2} như là cây con bên phải.

6.5.7.1. Hãy vẽ 7 cây Fibonacci có gốc đầu tiên.

6.5.7.2. Cây Fibonacci T_n có bao nhiêu đỉnh, lá và bao nhiêu đỉnh trong. Chiều cao của nó bằng bao nhiêu?

6.6.9. Hãy tìm cây khung cho mỗi đồ thị sau.

6.6.9.1 K_5

6.6.9.2 $K_{4,4}$

6.6.9.3 $K_{1,6}$

6.6.9.4 Q_3

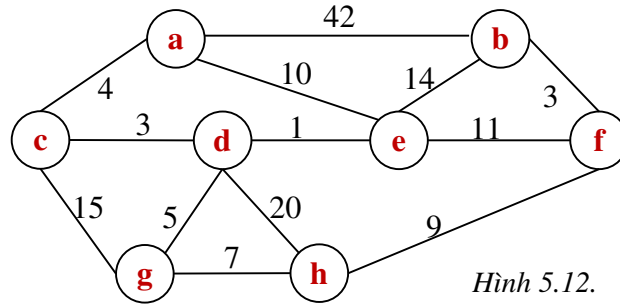
6.6.9.5 C_5

6.6.9.6 W_5 .

6.6.10. Đồ thị K_n với $n=3, 4, 5$ có bao nhiêu cây khung không đẳng cấu?

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

6.6.11.1.

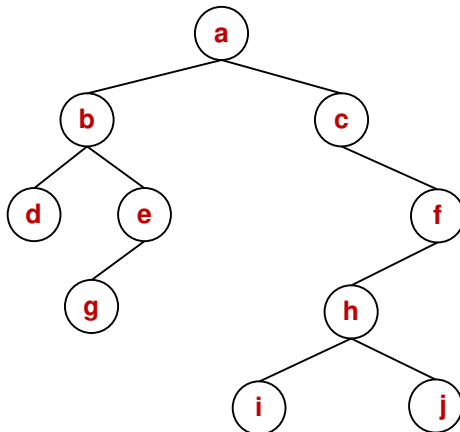


6.6.11.2.

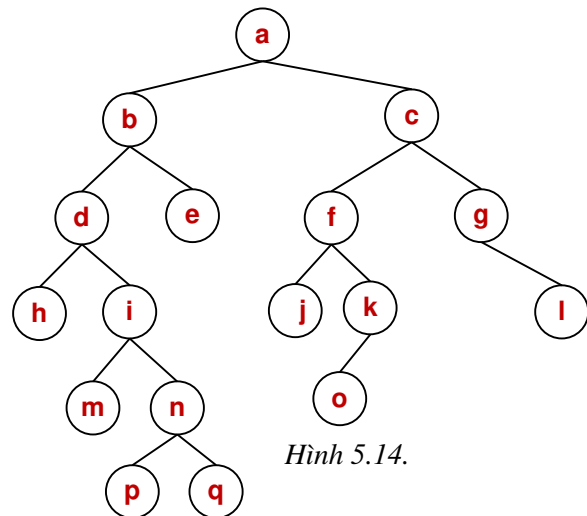
Hình 5.12.

	A	B	C	D	E	F	H	I	J
A	∞	18	13	20	28	21	26	19	25
B	18	∞	19	23	19	22	27	20	20
C	13	19	∞	16	22	23	21	26	15
D	20	23	16	∞	30	13	29	17	25
E	28	19	22	30	∞	12	23	16	21
F	21	22	23	13	12	∞	25	18	28
H	26	27	21	29	23	25	∞	24	20
I	19	20	26	17	16	18	24	∞	18
J	25	20	15	25	21	28	20	18	∞

6.6.12. Duyệt các cây sau đây lần lượt bằng 3 thuật toán đã học.



Hình 5.13.



Hình 5.14.

6.6.13. 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.

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.

	A	B	C	D	E	F	H	I
A	∞	16	15	23	19	18	32	20
B	16	∞	13	33	24	20	19	11
C	15	13	∞	13	29	21	20	19
D	23	33	13	∞	22	30	21	12
E	19	24	29	22	∞	34	23	21
F	18	20	21	30	34	∞	17	14
H	32	19	20	21	23	17	∞	18
I	20	11	19	12	21	14	18	∞

6.6.14. Viết các biểu thức sau đây theo ký pháp Ba Lan và ký pháp Ba Lan đảo.

6.6.14.1. $\frac{(A+B)(C+D)}{(A-B)C+D} + \frac{A^2+BD}{C^2-BD}$.

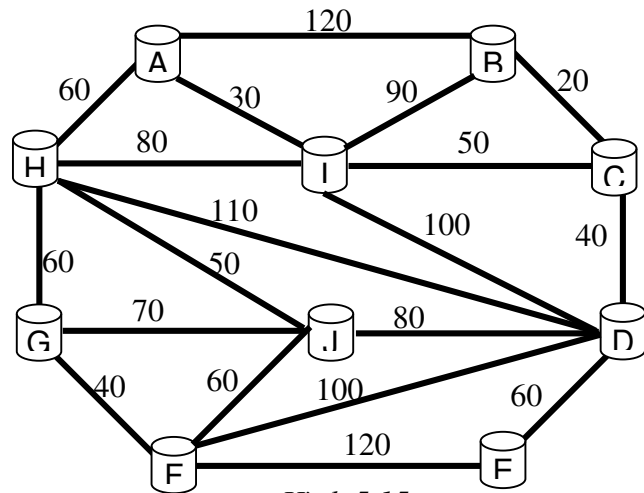
6.6.14.2. $\left[(a-b)^4 - \frac{c}{3} - 5d \right]^2 + \left(\frac{a-d}{3} \right)^4 \frac{(3a+4b-2d)^3}{5}$.

6.6.15. Viết các biểu thức sau đây theo ký pháp quen thuộc.

6.5.15.1. $xy + 2 \uparrow xy - 2 \uparrow - xy * /$.

6.5.15.2. $- * \uparrow / - - ab * 3 c 2 4 \uparrow - cd 5 * - - acd / \uparrow - b * 2 d 4 3$.

6.6.16. Một địa đạo gồm các căn hầm với vị trí căn hầm và độ dài từng địa đạo như hình vẽ dưới. Nếu chỉ yêu cầu giữa các hầm có thể có đi tới nhau được bằng đường địa đạo. Hãy đưa ra phương án phải đào những địa đạo nào trong những địa đạo đã cho để tổng chiều dài địa đạo phải đào là ít nhất.



Hình 5.15

6.6.17. Xác định thời gian ngắn nhất để hoàn thành dự án. Biết rằng dự án gồm 8 công đoạn, mỗi công đoạn có thời gian ($t[i]$) và điều kiện về thời điểm bắt đầu cho trong bảng sau:

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

TÀI LIỆU THAM KHẢO

- [1]. PGS. Hoàng Chí Thành và PGS Lê Trọng Vĩnh - bài giảng Lý Thuyết Đồ Thị, Khoa Toán - Cơ - Tin học, Trường Đại học Khoa học Tự nhiên, ĐHQG HN
- [2]. Giáo trình toán ứng dụng trong tin học, Vụ giáo dục chuyên nghiệp, NXB Giáo dục
- [3]. PTS.Nguyễn Cam - PTS.Chu Đức Khánh), Lý thuyết đồ thị (ngành Tin học), NXB Trẻ
- [4]. TS. Võ Văn Tuấn Dũng – Giáo trình toán rời rạc –NXB Lao động – Xã hội -2009
- [5]. Seymour Lipschutz PhD, MarcLars Lipson PhD – Biên dịch Thanh Tâm, Quang Huy, Xuân Toại – Tuyển chọn 1800 bài tập toán rời rạc – NXB Thống kê -2002
- [6]. Trần Ngọc Danh - Toán rời rạc nâng cao –NXB ĐHQG TP.HCM - 2004
- [7]. – Kenneth h. Rosen – Biên tập Đặng Văn Sử, Trần Hiến - Toán học rời rạc ứng dụng trong tin học – 1997
- [8]. Nguyễn Đức Nghĩa, Nguyễn Tô Thành, “Toán rời rạc”. NXB Giáo dục, 1999
- [9]. Kenneth H. Rosen, “Toán rời rạc và Ứng dụng trong tin học”, NXB Khoa học và Kỹ thuật, 2000
- [10]. Jean-Claude Fournier, “Graph Theory and Applications with Exercises and Problems”, Wiley, 2009
- [11]. Reinhard Diestel, “Graph Theory”, Springer-Verlag New York, 2005