

Bài 14: Đồ thị (1/2)

Giảng viên: Hoàng Thị Điệp

Khoa Công nghệ Thông tin – Đại học Công Nghệ

Nội dung chính



1. Đồ thị và các khái niệm liên quan
2. Cài đặt đồ thị
3. Một số bài toán tiêu biểu
 - Đi qua/duyệt đồ thị
 - BFS, DFS
 - Sắp xếp topo trên đồ thị định hướng không có chu trình
4. Đồ thị và C++
 - Tìm đường đi ngắn nhất
 - Từ một đỉnh nguồn
 - Giữa mọi cặp đỉnh
 - Tìm cây bao trùm ngắn nhất
 - Prim
 - Kruskal

1. Đồ thị và các khái niệm liên quan

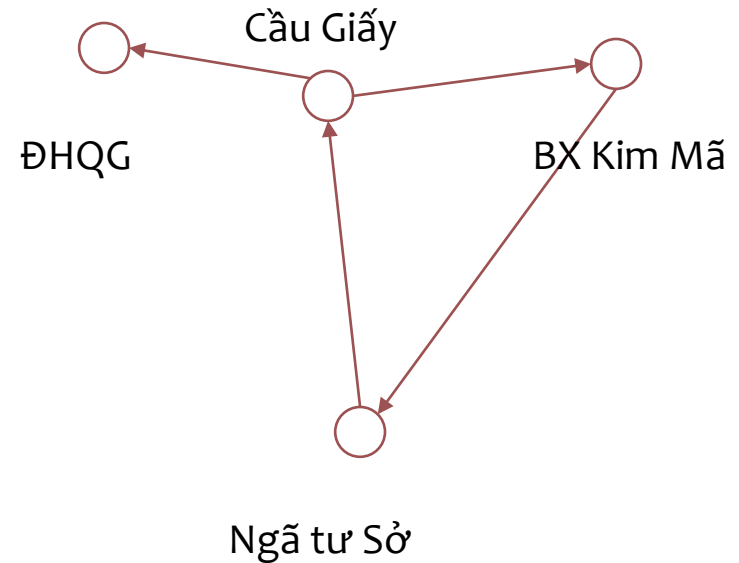
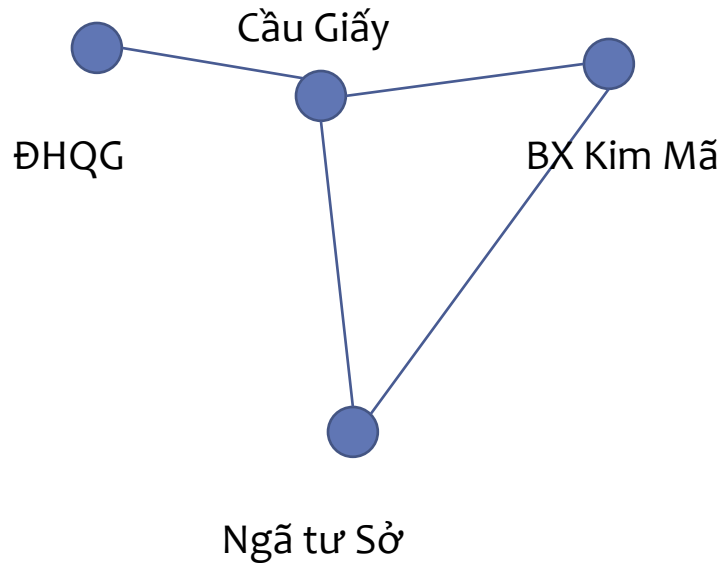
Định nghĩa: Đồ thị

Không phải là
đồ thị hàm số!

- Đồ thị là một mô hình toán học
 - được sử dụng để biểu diễn một tập đối tượng có quan hệ với nhau theo một cách nào đó.
- Định nghĩa hình thức
 - Đồ thị G được xác định bởi một cặp (V, E) , trong đó
 - V là tập đỉnh
 - E là tập các cạnh nối cặp đỉnh $E \subseteq \{(u, v) \mid u, v \in V\}$
- Đồ thị vô hướng
 - quan hệ định nghĩa bởi mỗi cạnh là quan hệ đối xứng
 - $E \subseteq \{\{u, v\} \mid u, v \in V\}$
- Đồ thị định hướng
 - $(u, v) \neq (v, u)$



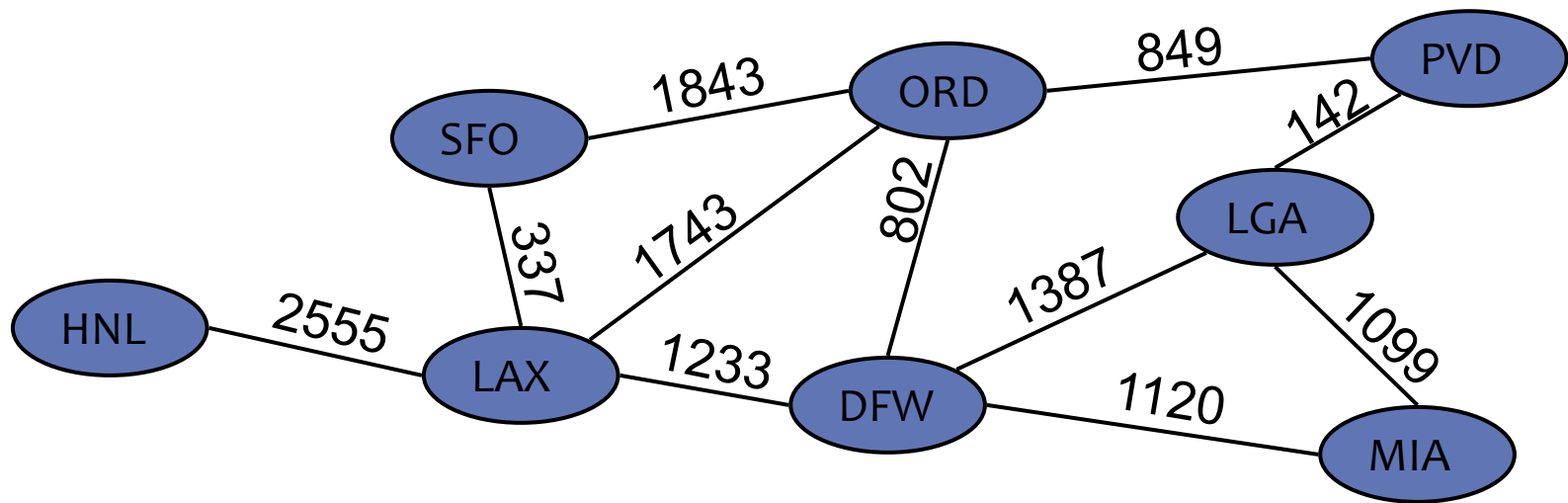
Ví dụ: đồ thị vô hướng – định hướng

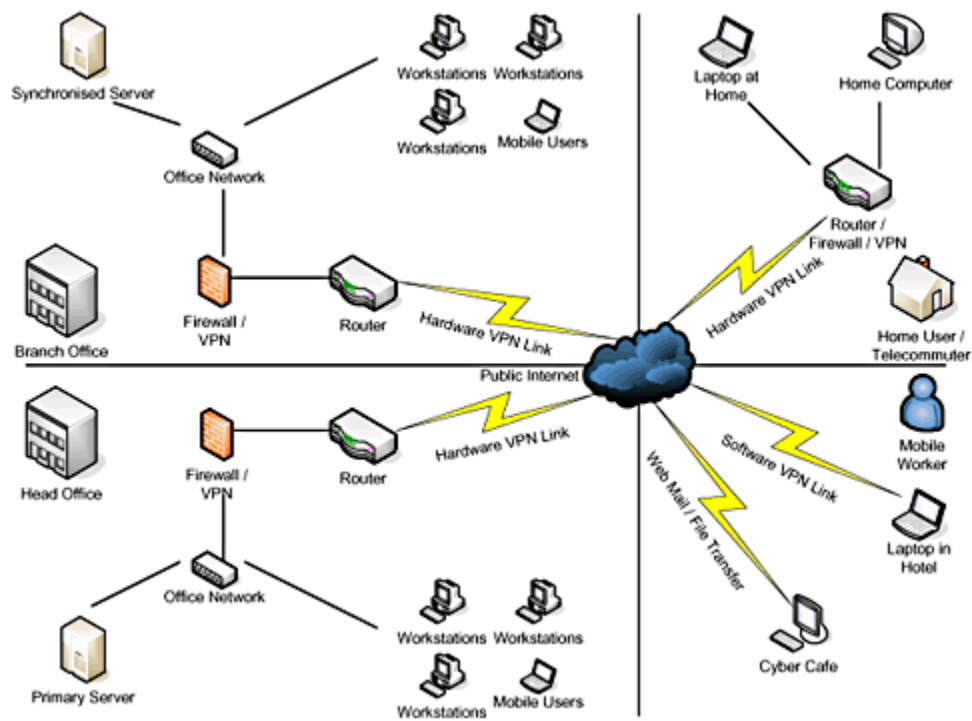


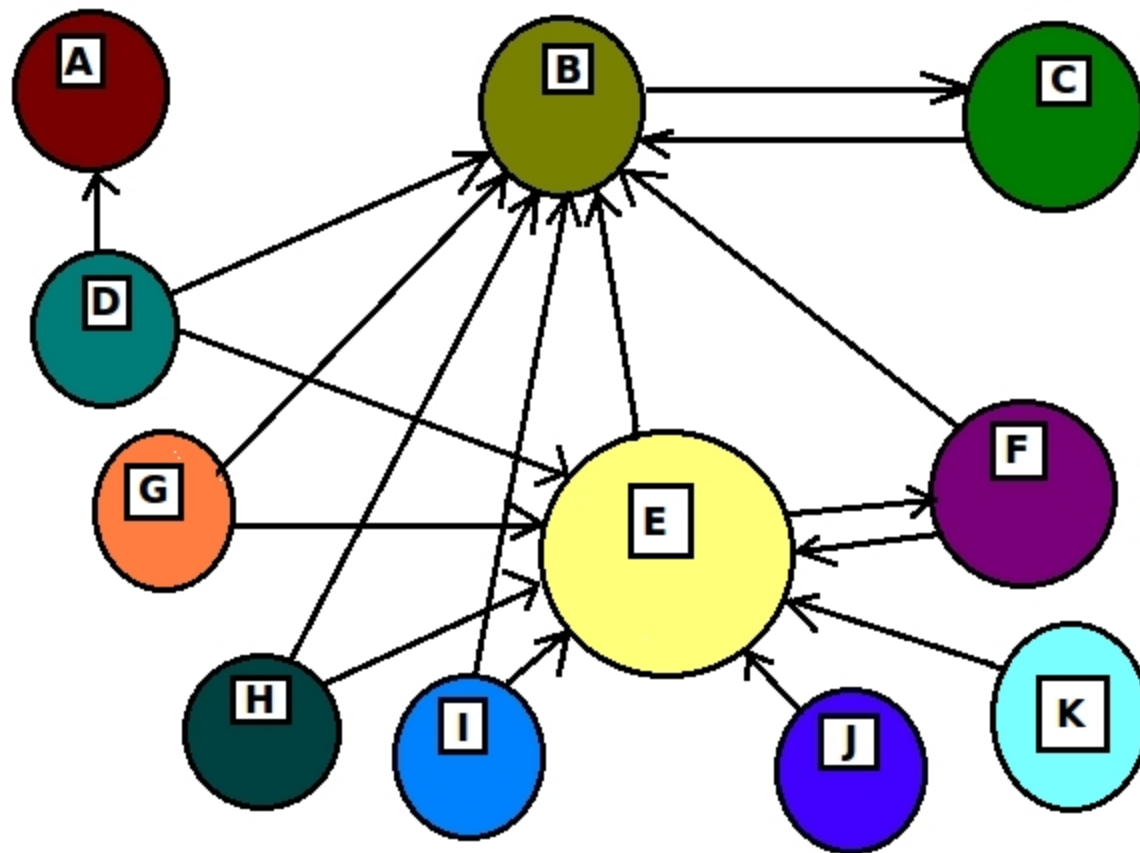
Ví dụ

- Mạng vận tải (transportation networks)
- Mạng liên lạc (communication networks)
- Mạng thông tin (information networks)
- Mạng xã hội (social networks)
- Mạng phụ thuộc (dependency networks)

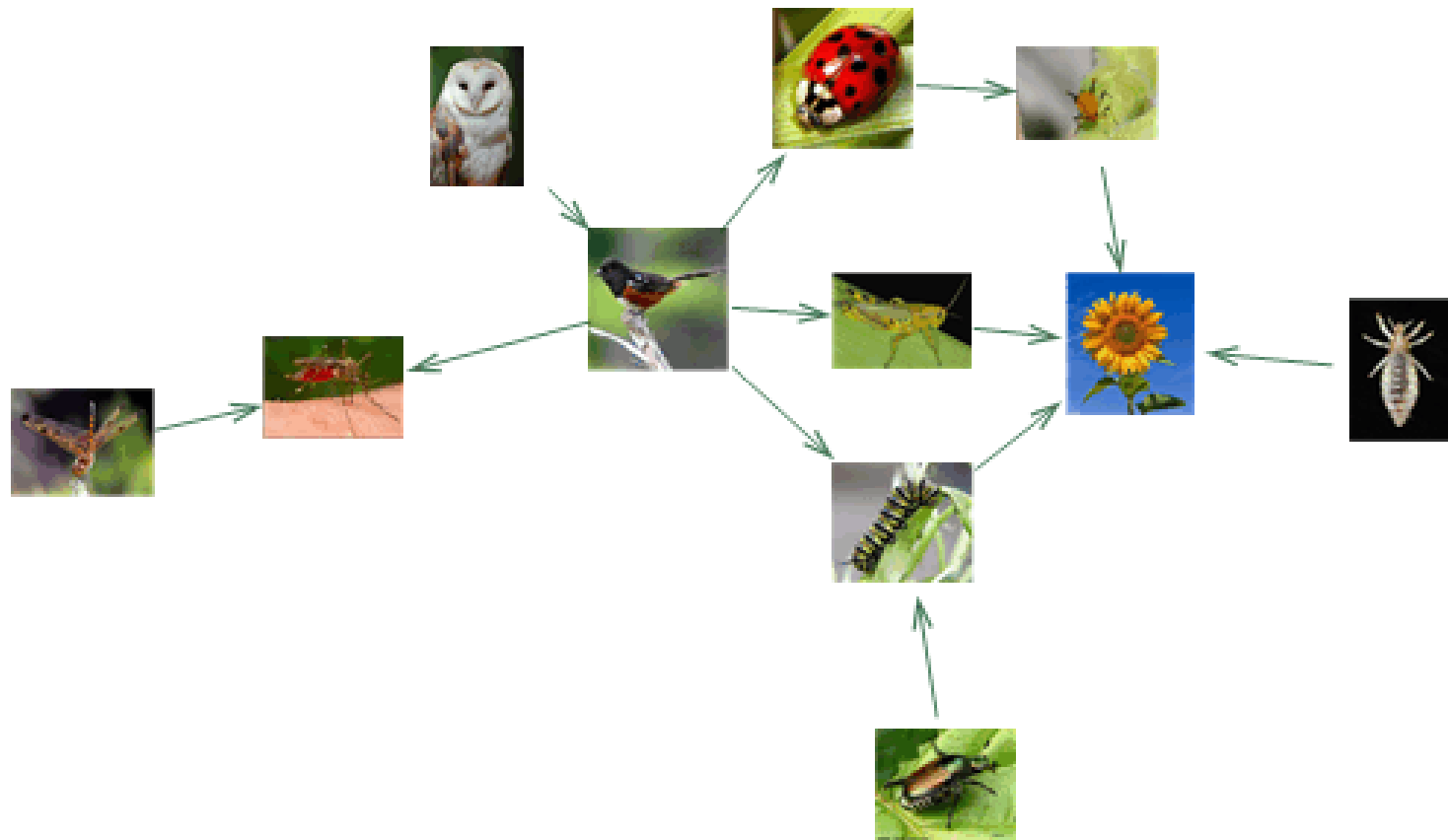
Định hướng hay vô hướng?









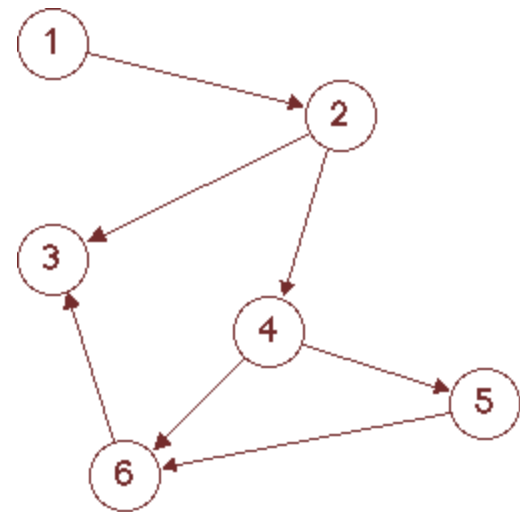
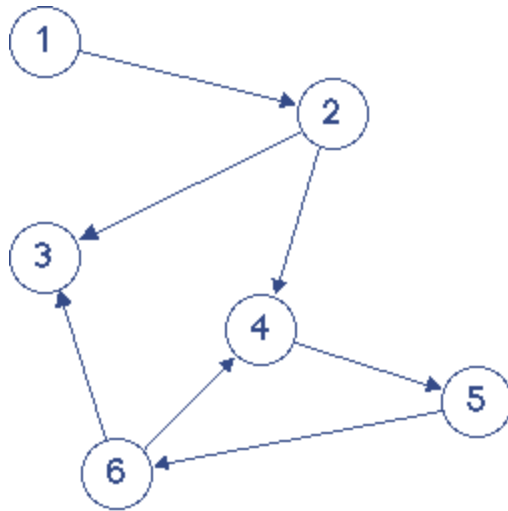


Nguồn: <http://reference.wolfram.com>

Định nghĩa: Đường đi

- Trong đồ thị vô hướng $G=(V,E)$
 - Đường đi
 - là dãy P các đỉnh v_1, v_2, \dots, v_k
 - có tính chất 2 đỉnh liên tiếp v_i, v_{i+1} được nối bởi 1 cạnh trong G .
 - P được gọi là đường đi từ v_1 đến v_k
 - Chu trình là đường đi v_1, v_2, \dots, v_k với $k > 2$ trong đó $k-1$ đỉnh đầu tiên phân biệt và $v_1 = v_k$
- Với đồ thị có hướng, trong một đường đi hay chu trình, 2 đỉnh liên tiếp (v_i, v_{i+1}) phải là một cung thuộc E

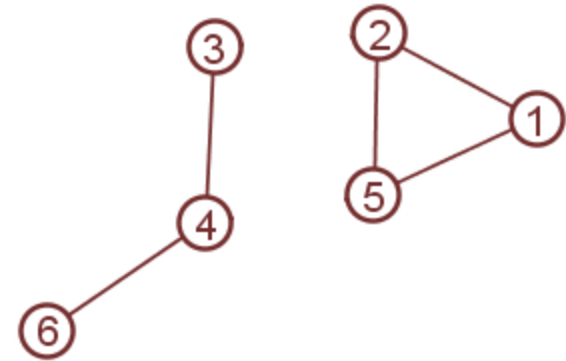
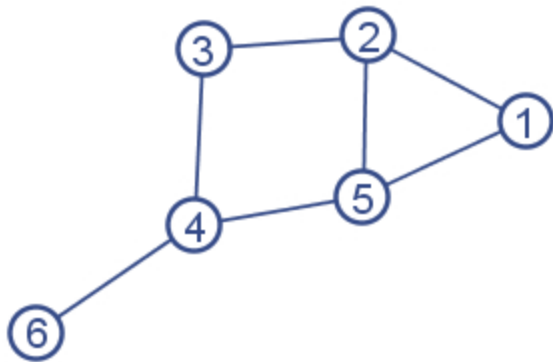
Ví dụ: đồ thị có chu trình – không có chu trình



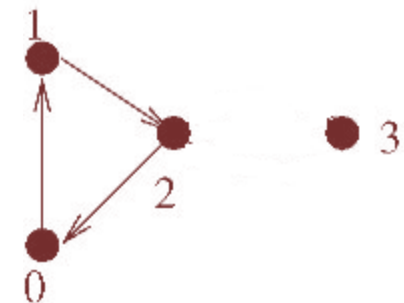
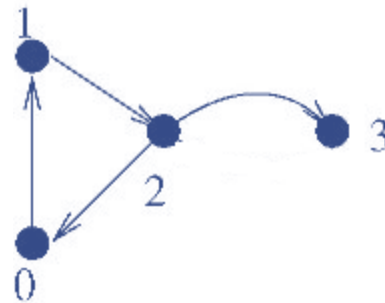
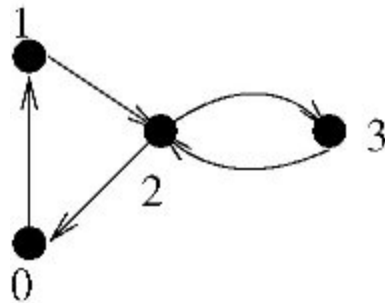
Định nghĩa: Tính liên thông

- **Đồ thị vô hướng** liên thông nếu tồn tại đường đi từ u đến v với mọi cặp đỉnh (u, v)
- **Đồ thị có hướng**
 - **liên thông yếu** nếu đồ thị vô hướng nền tảng của nó là đồ thị liên thông
 - **liên thông mạnh** nếu tồn tại một đường đi từ u đến v và một đường đi từ v đến u với mọi cặp đỉnh (u, v)

Ví dụ: đồ thị vô hướng liên thông – không liên thông

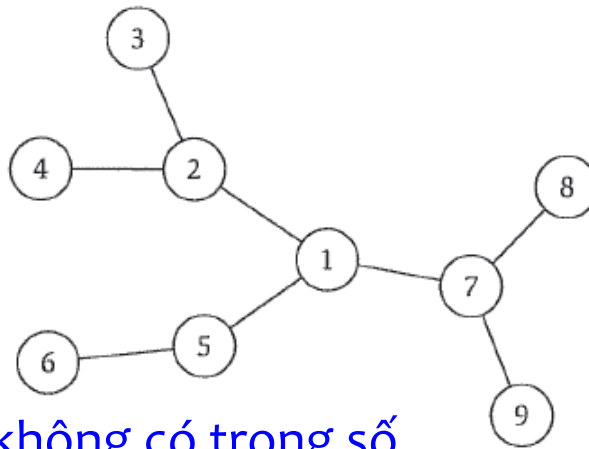


Ví dụ: đồ thị có hướng liên thông mạnh - yếu - không liên thông

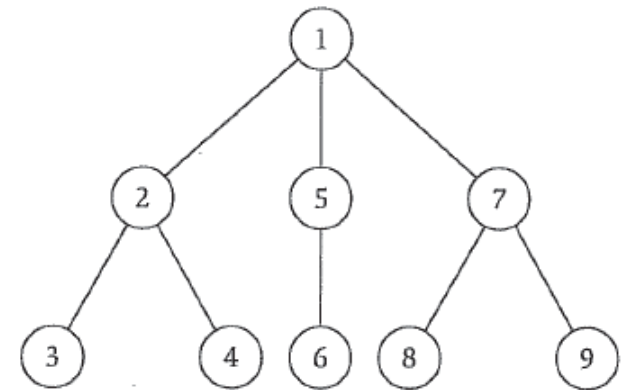


Các khái niệm khác

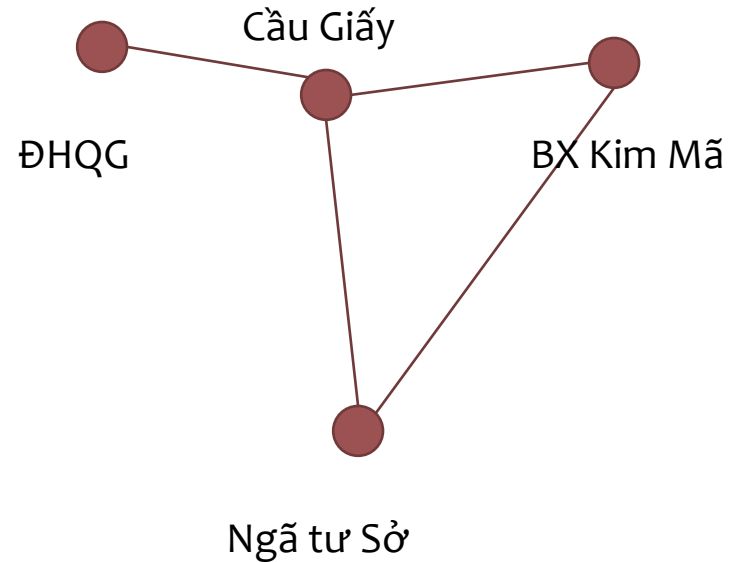
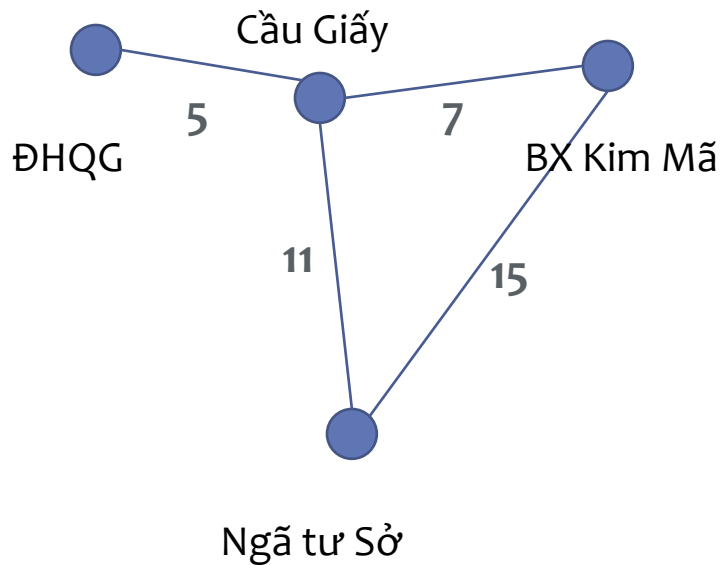
- **Khoảng cách** giữa 2 đỉnh u, v là số cạnh trên đường đi ngắn nhất từ u đến v
- **Cây trong lý thuyết đồ thị**: là đồ thị vô hướng liên thông không chứa chu trình



- Đồ thị có/không có trọng số
- Đồ thị có/không có nhãn

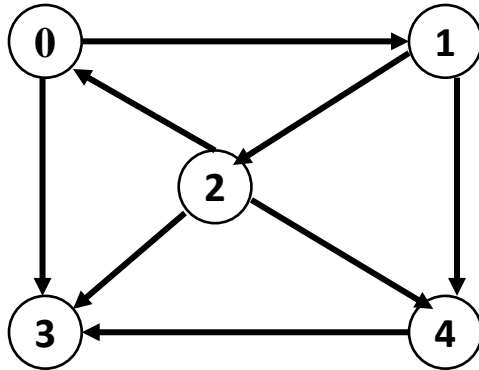


Ví dụ: đồ thị có trọng số - không trọng số

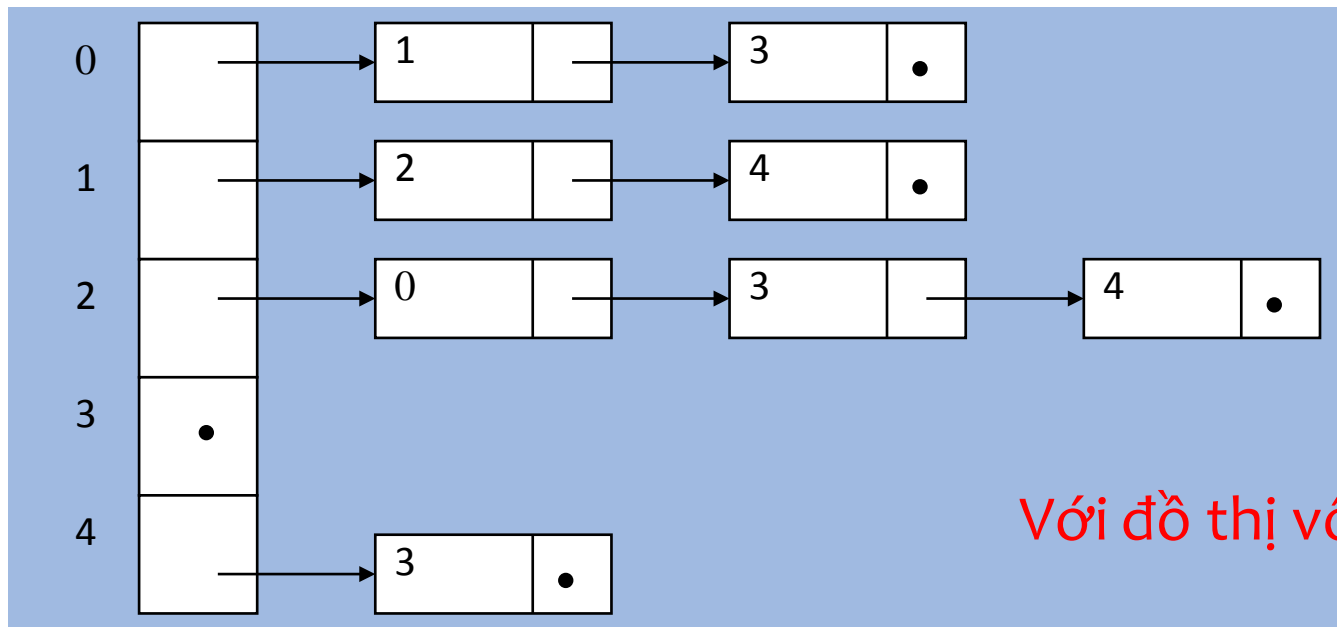


2. Cài đặt đồ thị

Hai cách cơ bản biểu diễn đồ thị

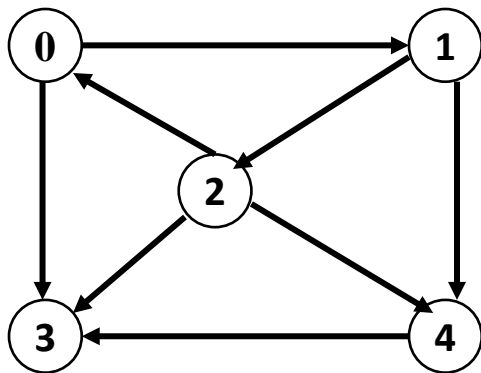


	0	1	2	3	4
0	0	1	0	1	0
1	0	0	1	0	1
2	1	0	0	1	1
3	0	0	0	0	0
4	0	0	0	1	0



Với đồ thị vô hướng?

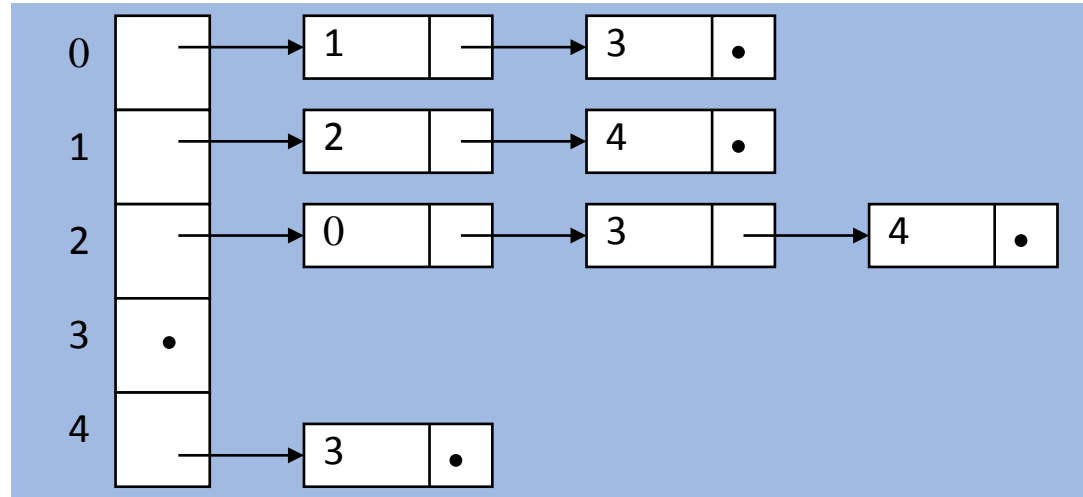
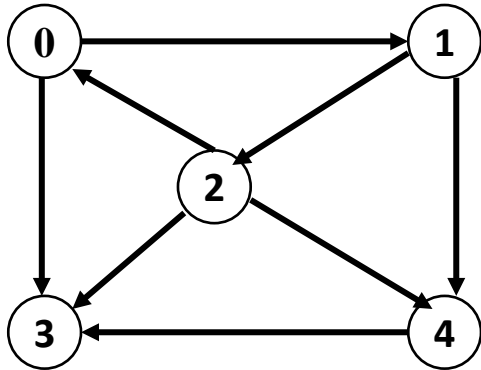
Cài đặt: Biểu diễn bằng ma trận kề



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

```
const int N = 5;  
typedef bool Graph[N][N];  
...  
Graph g1;  
g1[0][0] = 0;  
g1[0][1] = 1;  
...
```

Cài đặt: Biểu diễn bằng danh sách kề



```
struct Cell{  
    int vertex;  
    Cell * next;  
};  
const int N = 5;  
typedef Cell * Graph[N];  
...  
Graph g2;  
addFirst(g2[0], 3);  
addFirst(g2[0], 1);
```

So sánh 2 phương pháp biểu diễn

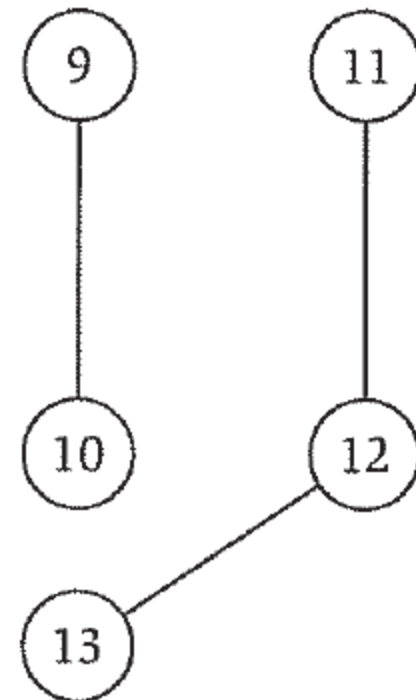
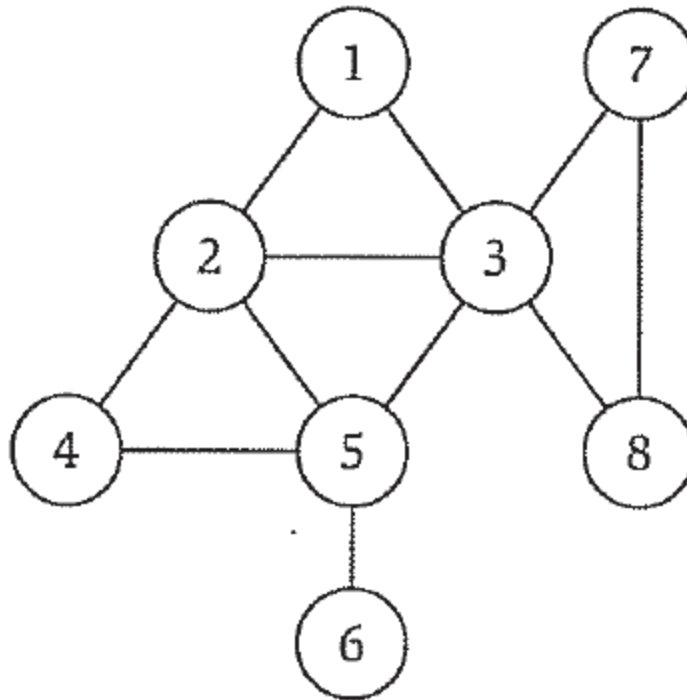
- Các yếu tố cần xét
 - Độ phức tạp thời gian của phép truy cập tới thông tin 1 cặp đỉnh u, v
 - Độ phức tạp không gian biểu diễn đồ thị
 - Độ phức tạp thời gian của phép khảo sát tập đỉnh kề với đỉnh u cho trước

3. Một số bài toán tiêu biểu

Đi qua đồ thị theo bề rộng

- Sử dụng kĩ thuật tìm kiếm theo bề rộng
 - Breadth-First Search
- Ý tưởng của tìm kiếm theo bề rộng xuất phát từ đỉnh v
 - Từ đỉnh v ta lần lượt đi thăm tất cả các đỉnh u kề đỉnh v mà u chưa được thăm.
 - Sau đó, đỉnh nào được thăm trước thì các đỉnh kề nó cũng sẽ được thăm trước.
 - Quá trình trên sẽ được tiếp tục cho tới khi ta không thể thăm đỉnh nào nữa.

Ví dụ BFS(1)



BFS(v)

Algorithm *BFS*(v)

// Tìm kiếm theo bề rộng xuất phát từ v.

Input: Đỉnh v chưa được thăm

Khởi tạo hàng đợi Q rỗng;

Đánh dấu đỉnh v đã được thăm;

Q.enqueue(v)

while Q.empty() \neq TRUE

 w \leftarrow Q.dequeue()

 for (mỗi đỉnh u kề w)

 if (u chưa được thăm)

 Đánh dấu u đã được thăm;

 Q.enqueue(u)

Thuật toán đi qua đồ thị G theo bề rộng

Algorithm *BFSTraversal*(G)

// Đi qua đồ thị $G=(V, E)$ theo bề rộng

for (mỗi $v \in V$)

 Đánh dấu v chưa được thăm;

for (mỗi $v \in V$)

 if (v chưa được thăm)

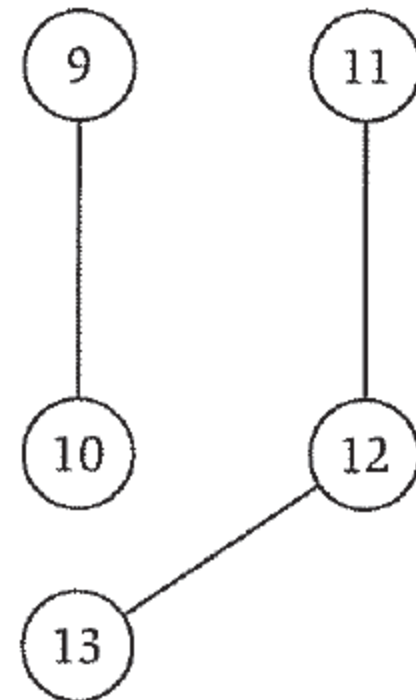
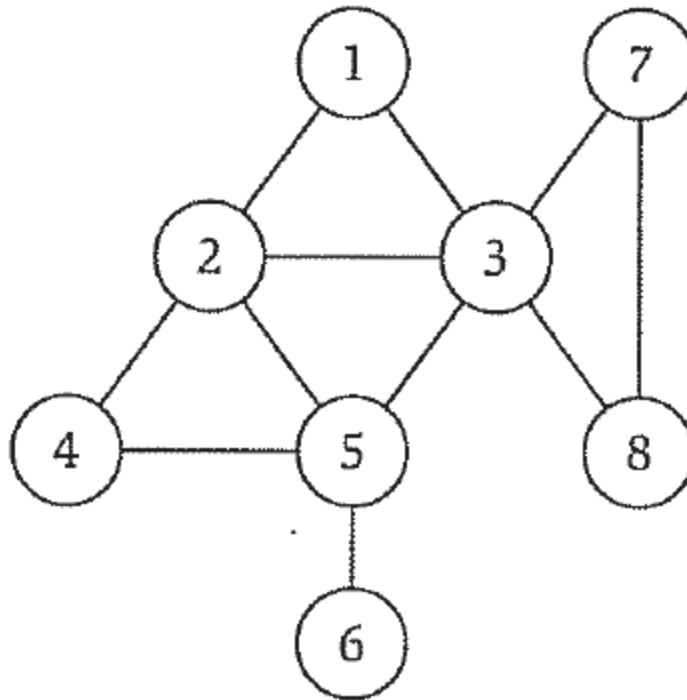
 BFS(v);

- Phân tích
- Ứng dụng
 - Vấn đề đạt tới: Giả sử v và w là hai đỉnh bất kỳ, ta muốn biết từ đỉnh v có đường đi tới đỉnh w hay không?
 - Tính liên thông và thành phần liên thông của đồ thị vô hướng

Đi qua đồ thị theo độ sâu

- Sử dụng kĩ thuật tìm kiếm theo độ sâu
 - Depth-First Search
- Ý tưởng của tìm kiếm theo độ sâu xuất phát từ đỉnh u
 - Từ đỉnh u ta đến thăm một đỉnh v kề đỉnh u . Rồi lại từ đỉnh v ta đến thăm đỉnh w kề v . Cứ thế tiếp tục chừng nào có thể được.
 - Khi đạt tới đỉnh v mà tại v ta không đi thăm tiếp được thì
 - quay lại đỉnh u và từ đỉnh u ta đi thăm đỉnh v' khác kề u (nếu có), rồi từ v' lại đi thăm tiếp đỉnh kề v' ,...
 - Quá trình trên sẽ tiếp diễn cho tới khi ta không thể tới thăm đỉnh nào nữa.

Ví dụ DFS(1)



DFS(v)

Algorithm *DFS(v)*

// Tìm kiếm theo độ sâu xuất phát từ v.

Input: Đỉnh v chưa được thăm

for (mỗi đỉnh u kề v)

 if (u chưa được thăm)

 Đánh dấu u đã được thăm;

 DFS(u)

Thuật toán đi qua đồ thị G theo độ sâu

Algorithm *DFSTraversal*(G)

// Đi qua đồ thị $G=(V, E)$ theo độ sâu

for (mỗi $v \in V$) Đánh dấu v chưa được thăm;

for (mỗi $v \in V$)

 if (v chưa được thăm)

 Thăm v và đánh dấu v đã được thăm;

 DFS(v);

- Phân tích
- Ứng dụng
 - Phân lớp các cung
 - Phát hiện chu trình trong đồ thị

Lịch trình

Tuần 14, 15

Thi cuối kỳ vào 10/01

1. Đồ thị và các khái niệm liên quan
2. Cài đặt đồ thị
3. Một số bài toán tiêu biểu
 - Đi qua/duyệt đồ thị
 - Sắp xếp topo trên đồ thị định hướng không có chu trình
 - Tìm đường đi ngắn nhất
 - Tìm cây bao trùm ngắn nhất
4. Đồ thị và C++

Chuẩn bị tuần tới

- Lý thuyết: Đọc tiếp chương 18 giáo trình
- Thực hành: Đề thi