ĐỒ THỊ - GRAPH

DATA STRUCTURES & ALGORITHMS

ThS Nguyễn Thị Ngọc Diễm
diemntn@uit.edu.vn

NỘI DUNG

- Đồ thị và các khái niệm trên đồ thị
- **Biểu diễn đồ thị trên máy tính**
  - **Sử dụng ma trận kề**
  - **Sử dụng danh sách kề**
  - **Sử dụng danh sách cạnh**
  - **Độ phức tạp**
- Duyệt đồ thị theo chiều sâu (DFS)
- Duyệt đồ thị theo chiều rộng (BFS)
- Một số ứng dụng

BIỂU DIỄN ĐỒ THỊ TRÊN MÁY TÍNH

Graph Representation

Two popular computer representations of a graph. Both represent the vertex set and the edge set, but in different ways.

1. **Adjacency Matrix (Ma trận kề)**
   Use a 2D matrix to represent the graph
2. **Adjacency List (Danh sách kề)**
   Use a 1D array of linked lists
3. **Edge List (Danh sách cạnh)**
   Use a 1D array of linked lists or array

MA TRẬN KỀ

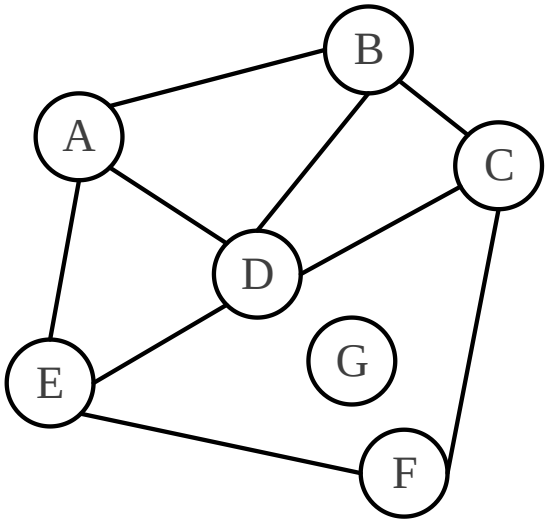
- Ma trận KỀ:
  - Xét đồ thị $G = (X, U)$, giả sử tập X gồm N đỉnh và được sắp thứ tự $X = \{x_1, x_2, \dots, x_N\}$, tập U gồm M cạnh và được sắp thứ tự $U = \{u_1, u_2, \dots, u_M\}$.
  - Ma trận kề của đồ thị G, ký hiệu $B(G)$, là một ma trận nhị phân cấp $N \times N$: $B = (B_{ij})$ với B_{ij} được định nghĩa:
    - $B_{ij} = 1$ nếu có cạnh nối x_i tới x_j,
    - $B_{ij} = 0$ trong trường hợp ngược lại.

MA TRẬN KỀ

- Ma trận của đồ thị vô hướng:
  - Xét đồ thị $G = (X, U)$ vô hướng, giả sử tập X gồm N đỉnh và được sắp thứ tự $X = \{x_1, x_2, \dots, x_N\}$, tập U gồm M cạnh và được sắp thứ tự $U = \{u_1, u_2, \dots, u_M\}$.
  - Ma trận của G, ký hiệu $A(G)$, là ma trận nhị phân NxM: $A = (A_{ij})$ với A_{ij} được định nghĩa:
    - $A_{ij} = 1$ nếu đỉnh x_i kề với cạnh u_j,
    - $A_{ij} = 0$ nếu ngược lại.

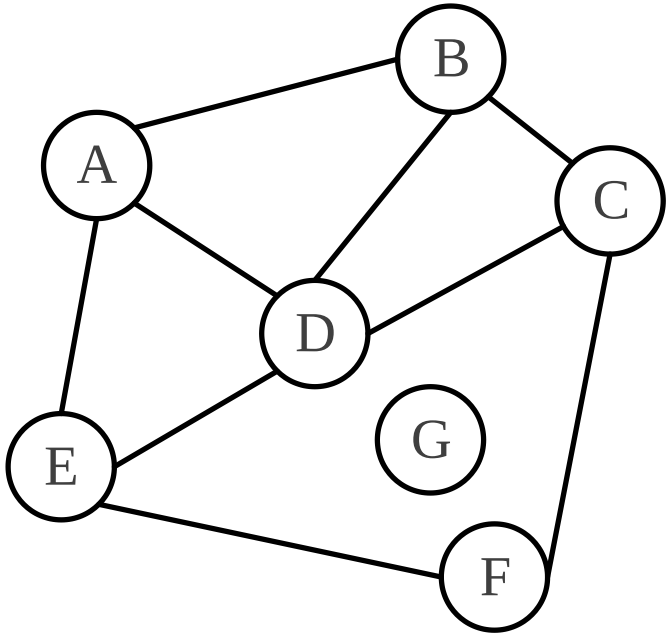
MA TRẬN KỀ

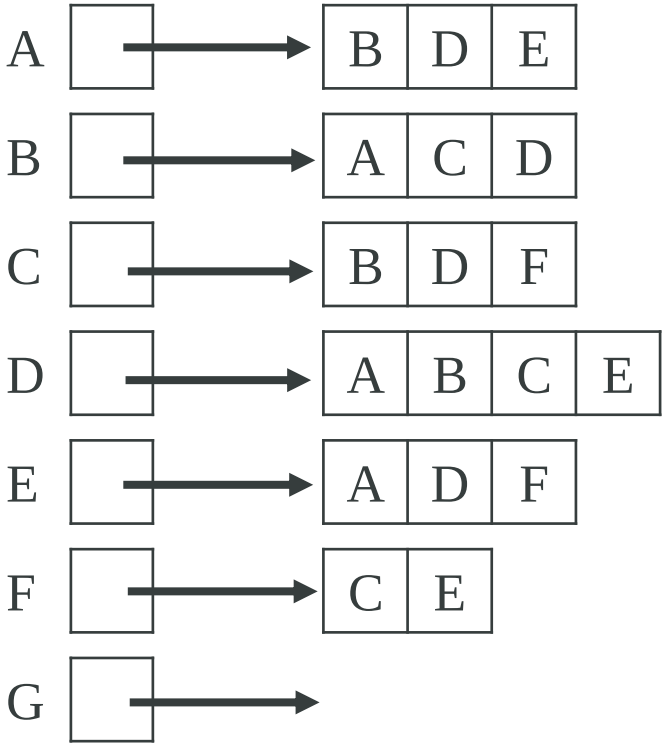
- 2D array A[0..n-1, 0..n-1], where n is the number of vertices in the graph
- Each row and column is indexed by the vertex id. e,g a=0, b=1, c=2, d=3, e=4
- An array entry A [i] [j] is equal to 1 if there is an edge connecting vertices i and j. Otherwise, A [i] [j] is 0.
- The storage requirement is Θ(n2). Not efficient if the graph has few edges.
- We can detect in O(1) time whether two vertices are connected



	A	B	C	D	E	F	G
A	0	1	0	1	1	0	0
B	1	0	1	1	0	0	0
C	0	1	0	1	0	1	0
D	1	1	1	0	1	0	0
E	1	0	0	1	0	1	0
F	0	0	1	0	1	0	0
G	0	0	0	0	0	0	0

DANH SÁCH KỀ- ADJACENCY LIST





- The adjacency list is an array L[0..n-1] of lists, where n is the number of vertices in the graph.
- Each array entry is indexed by the vertex id (as with adjacency matrix)
- The list L[i] stores the ids of the vertices adjacent to i

Storage of adjacency list

- The array takes up $\Theta(n)$ space
- Define degree of v, $\deg(v)$, to be the number of edges incident to v. Then, the total space to store the graph is proportional to:

$$\sum_{\text{vertex } v} \deg(v)$$

- An edge $e=\{u,v\}$ of the graph contributes a count of 1 to $\deg(u)$ and contributes a count 1 to $\deg(v)$
- Therefore, $\Sigma_{\text{vertex } v}\deg(v) = 2m$, where m is the total number of edges
- In all, the adjacency list takes up $\Theta(n+m)$ space.
  - If $m = O(n^2)$, both adjacent matrix and adjacent lists use $\Theta(n^2)$ space.
  - If $m = O(n)$, adjacent list outperform adjacent matrix
- However, one cannot tell in $O(1)$ time whether two vertices are connected.

Adjacency Lists vs. Matrix

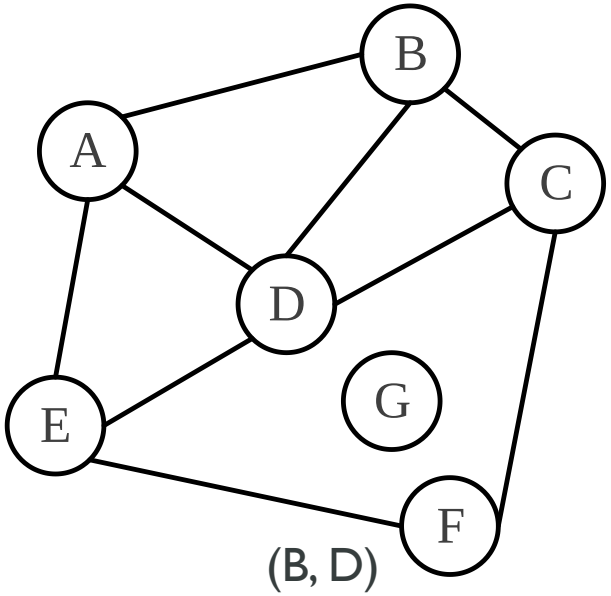
- **Adjacency Lists**
  - More compact than adjacency matrices if graph has few edges
  - Requires more time to find if an edge exists

- **Adjacency Matrix**
  - Always require n^2 space
    - This can waste a lot of space if the number of edges are sparse
  - Can quickly find if an edge exists

DANH SÁCH CẠNH- EDGE LIST

- Trong trường hợp đồ thị có n đỉnh, m cạnh, ta có thể biểu diễn đồ thị dưới dạng danh sách cạnh, trong cách biểu diễn này, người ta liệt kê tất cả các cạnh của đồ thị trong một danh sách, mỗi phần tử của danh sách là một cặp (u, v) tương ứng với một cạnh của đồ thị. (Trong trường hợp đồ thị có hướng thì mỗi cặp (u, v) tương ứng với một cung, u là đỉnh đầu và v là đỉnh cuối của cung). Danh sách được lưu trong bộ nhớ dưới dạng mảng hoặc danh sách móc nối. Ví dụ với đồ thị dưới đây:

DANH SÁCH CẠNH- EDGE LIST

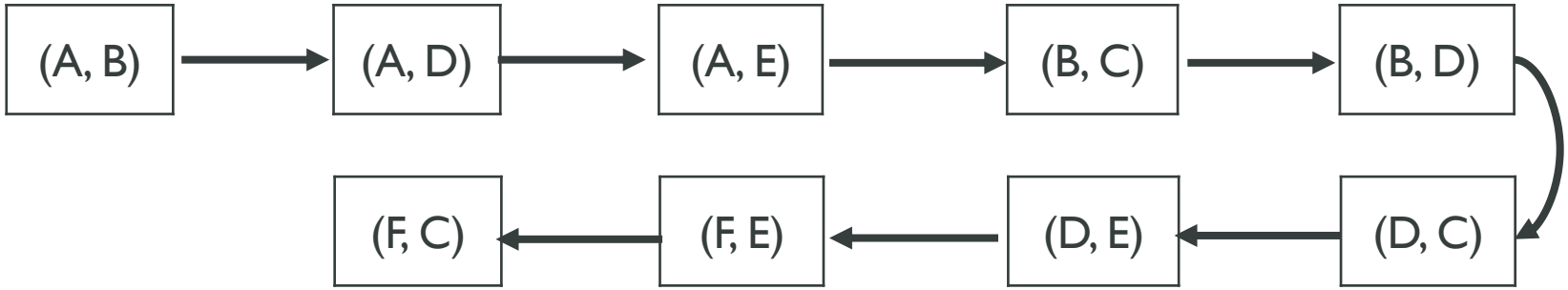


(B, D)

Using Array

0	1	2	3	4	5	6	7	8
(A, B)	(A, D)	(A, E)	(B, C)	(B, D)	(D, C)	(D, E)	(F, E)	(F, C)

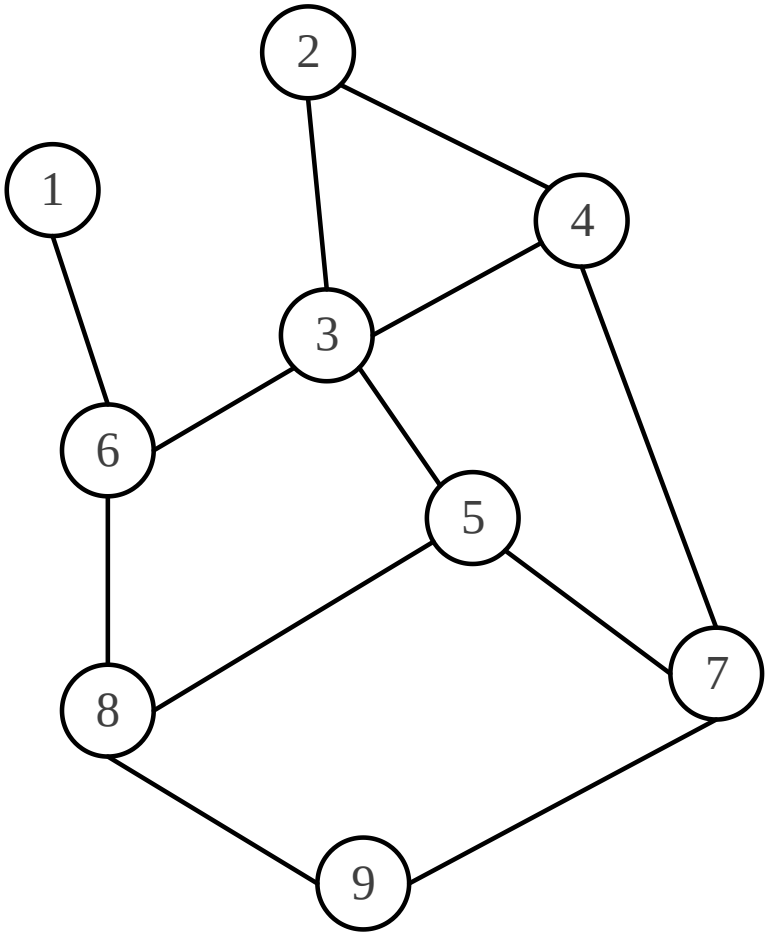
Or using Linked list



DANH SÁCH CẠNH- EDGE LIST

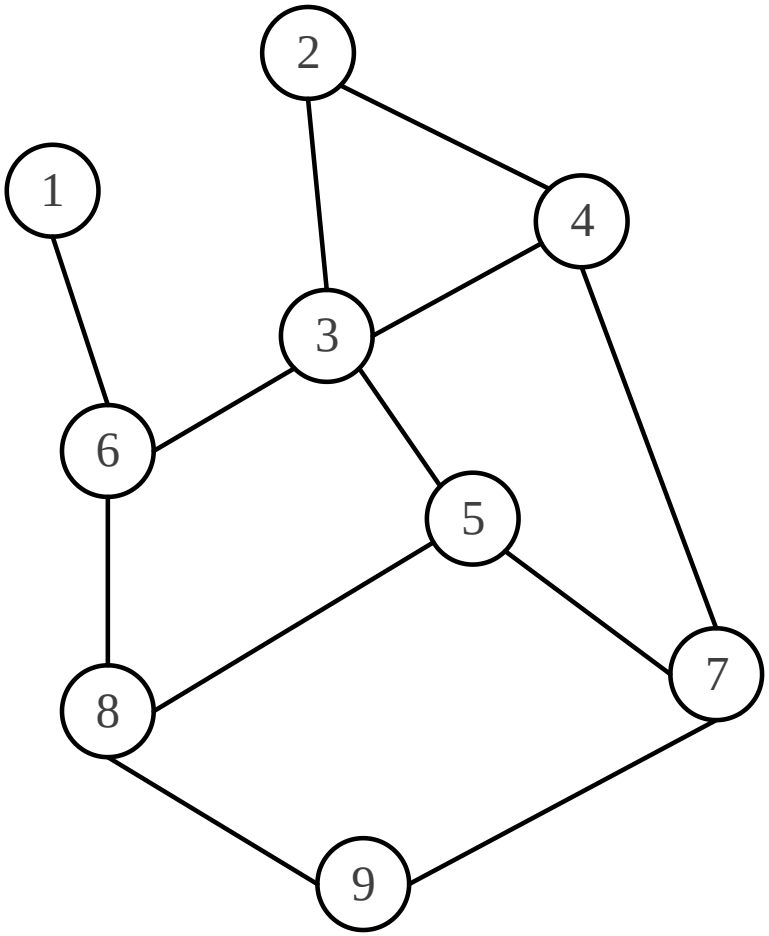
- Ưu điểm của danh sách cạnh:
  - Trong trường hợp đồ thị thưa (có số cạnh tương đối nhỏ: chẳng hạn m < 6n), cách biểu diễn bằng danh sách cạnh sẽ tiết kiệm được không gian lưu trữ, bởi nó chỉ cần 2m ô nhớ để lưu danh sách cạnh.
  - Trong một số trường hợp, ta phải xét tất cả các cạnh của đồ thị thì cài đặt trên danh sách cạnh làm cho việc duyệt các cạnh dễ dàng hơn. (Thuật toán Kruskal chẳng hạn)
- Nhược điểm của danh sách cạnh:
  - Nhược điểm cơ bản của danh sách cạnh là khi ta cần duyệt tất cả các đỉnh kề với đỉnh v nào đó của đồ thị, thì chẳng có cách nào khác là phải duyệt tất cả các cạnh, lọc ra những cạnh có chứa đỉnh v và xét đỉnh còn lại. Điều đó khá tốn thời gian trong trường hợp đồ thị dày (nhiều cạnh)

Example



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

Example



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

Storage of adjacency list

- The array takes up Θ(n) space
- Define degree of v, deg(v), to be the number of edges incident to v. Then, the total space to store the graph is proportional to:

$$\sum_{vertex\ v} \deg(v)$$

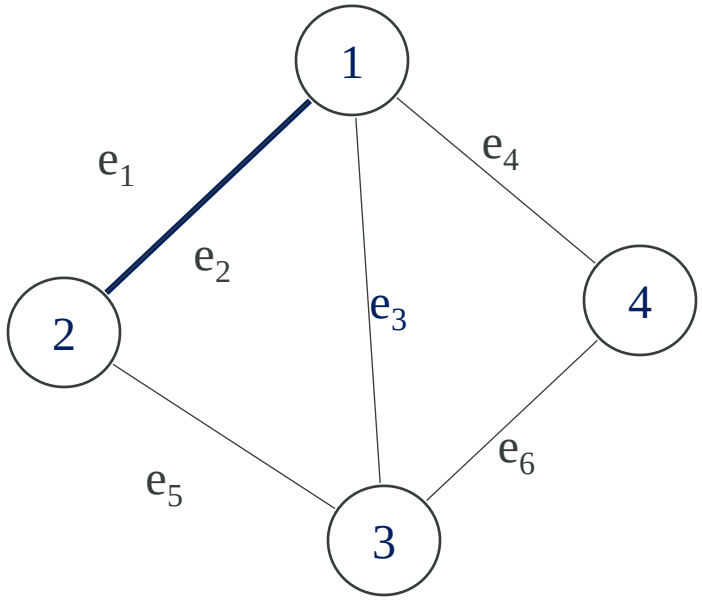
- An edge e={u, v} of the graph contributes a count of 1 to deg(u) and contributes a count 1 to deg(v)
- Therefore, $\sum_{vertex\ v} \deg(v)$=2m, where m is the total number of edges
- In all, the adjacency list takes up Θ(n+m) space.
  - If m = O(n^2), both adjacent matrix and adjacent lists use Θ(n^2) space.
  - If m = O(n), adjacent list outperform adjacent matrix
- However, one cannot tell in O(1) time whether two vertices are connected.

Adjacency Lists vs. Matrix

- Adjacency Lists
  - More compact than adjacency matrices if graph has few edges
  - Requires more time to find if an edge exists

- Adjacency Matrix
  - Always require n^2 space
    - This can waste a lot of space if the number of edges are sparse
  - Can quickly find if an edge exists

BIỂU DIỄN ĐỒ THỊ BẰNG MA TRẬN TRỌNG SỐ



G

BIỂU DIỄN ĐỒ THỊ BẰNG MA TRẬN TRỌNG SỐ

- .....

CÀI ĐẶT BẰNG NGÔN NGỮ C++

```
#define MaxV 20 // số đỉnh cực đại của đồ thị

int A[MaxV][MaxV]; //Ma trận kề

int ChuaXet[MaxV]; // sử dụng xét thành phần liên thông
```

Chúc các em học tốt!