CHƯƠNG 6: ĐỒ THỊ (GRAPH)

DATA STRUCTURES AND ALGORITHMS

L/O/G/O

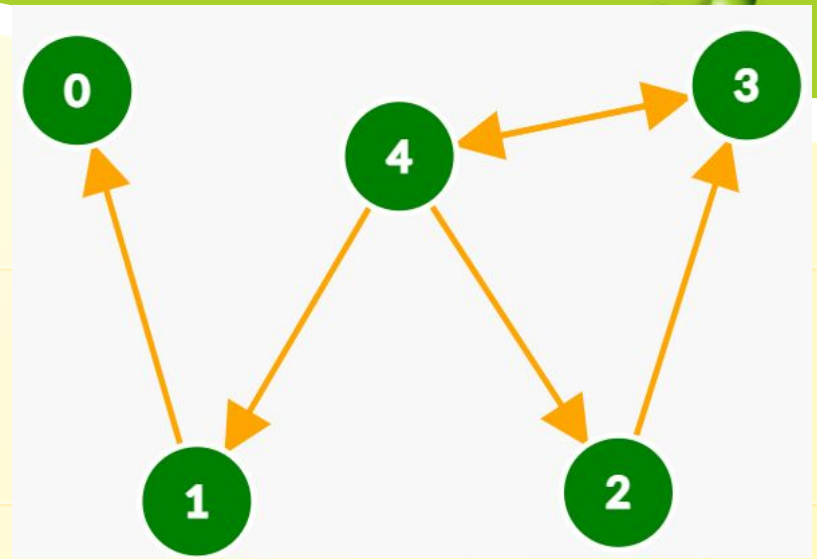
Nội dung buổi 1 (tiếp theo)

6.1 Các khái niệm trên đồ thị

- Định nghĩa
- Các loại đồ thị
- Khái niệm đường đi, chu trình, liên thông

6.2 Biểu diễn đồ thị trên máy tính

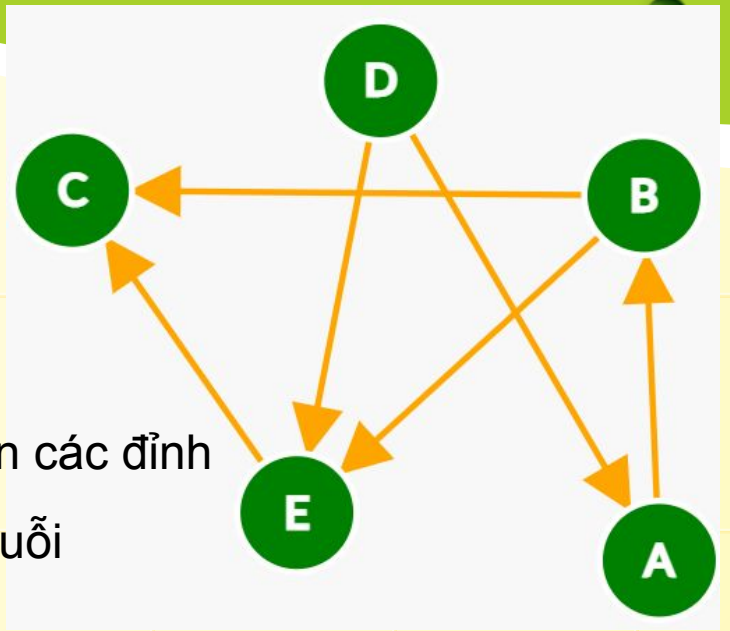
Nhập đơn đồ thị



- Trường hợp 1:
  - Cho biết số đỉnh, số cạnh
  - Quy ước đánh số các đỉnh là số nguyên (0,1,…) hoặc (1, 2, …)
- Input:
  - Dòng đầu tiên chứa 02 số v, e lần lượt là số đỉnh, số cạnh
  - e dòng tiếp theo, mỗi dòng chứa 02 số nguyên u và i thể hiện có cạnh nối từ đỉnh u sang đỉnh i

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

Nhập đơn đồ thị



- Trường hợp 2:
  - Cho biết số đỉnh, số cạnh và danh sách tên các đỉnh
  - Mỗi đỉnh của đồ thị được đặt tên là một chuỗi
- Input:
  - Dòng đầu tiên chứa 02 số v, e lần lượt là số đỉnh, số cạnh
  - Dòng tiếp theo chứa v chuỗi (chuỗi không khoảng trắng) là danh sách tên các đỉnh
  - e dòng tiếp theo, mỗi dòng chứa 02 chuỗi u và i thể hiện có cạnh nối từ đỉnh u sang đỉnh i

```
5 6
A B C D E
A B
B C
D E
D A
E C
B E
```

BIỂU DIỄN ĐỒ THỊ
(Representations of a graph)

2 cách phổ biến nhất

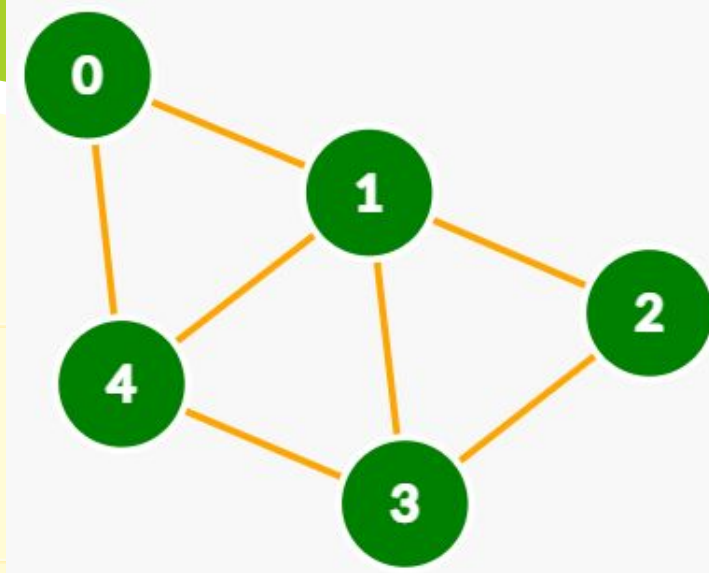
1. Ma trận kề (Adjacency Matrix)
2. Danh sách kề (Adjacency List)

Khác:

3. Ma trận liên thuộc (Incidence Matrix)
4. Danh sách liên thuộc (Incidence List)
5. Danh sách cạnh, ….

***Việc chọn cách biểu diễn nào là tùy vào tình huống sử dụng*

MA TRẬN KỀ



- ✔ **Đơn đồ thị vô hướng** G=(V,E),
  - Tập đỉnh V = {0, 1, …, n-1}, có n đỉnh
  - Tập cạnh E = $\{e_0, e_1, \ldots, e_{m-1}\}$, có m cạnh
- ✔ Ma trận 2 chiều, kích thước n x n

a = $\{a_{ij} \mid i,j=0, 1, \ldots, n-1\}$ □a[i][j]

Trong đó:

- a_{ij} = 1, nếu (i,j)∈E
- a_{ij} = 0, nếu (i,j) ∉E

- ✔ Ưu, nhược điểm?

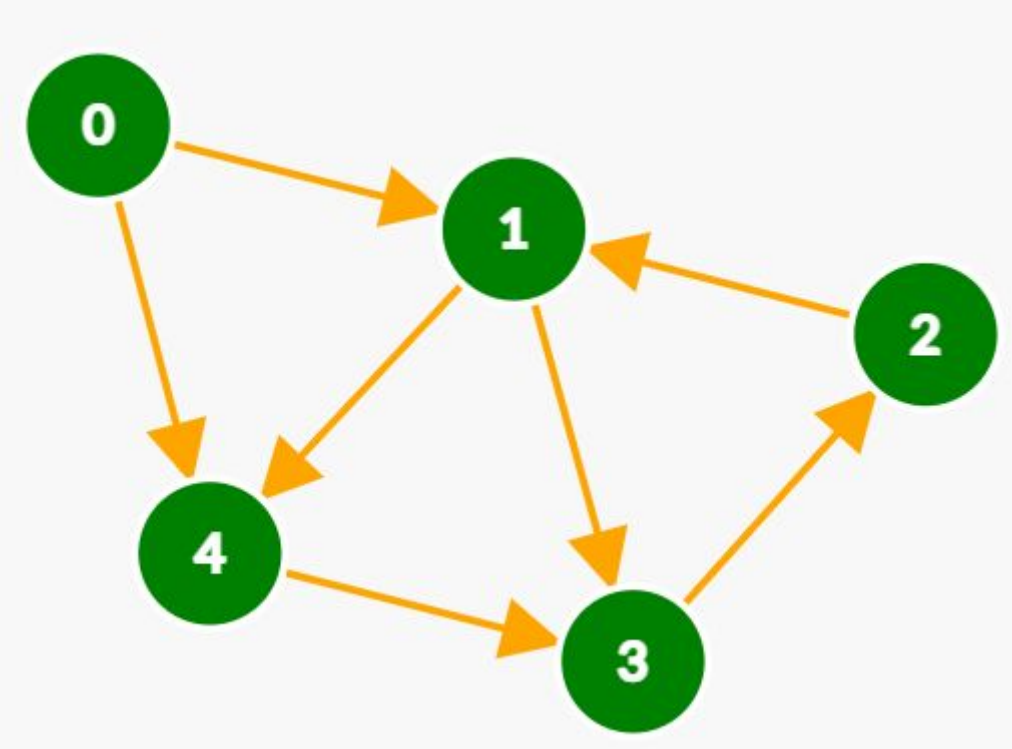
	0	1	2	3	4
0	0	1	0	0	1
1	1	0	1	1	1
2	0	1	0	1	0
3	0	1	1	0	1
4	1	1	0	1	0

1. Đối xứng: $a_{ij}=a_{ji}$
2. Tổng dòng (cột) i = bậc của đỉnh i

MA TRẬN KỀ

✔ **Đơn đồ thị có hướng:**

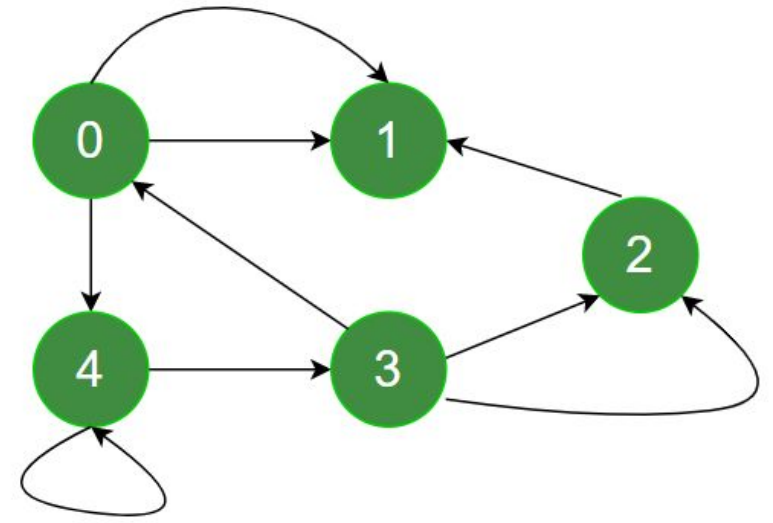
– ma trận không đối xứng ($a_{ij} \neq a_{ji}$)

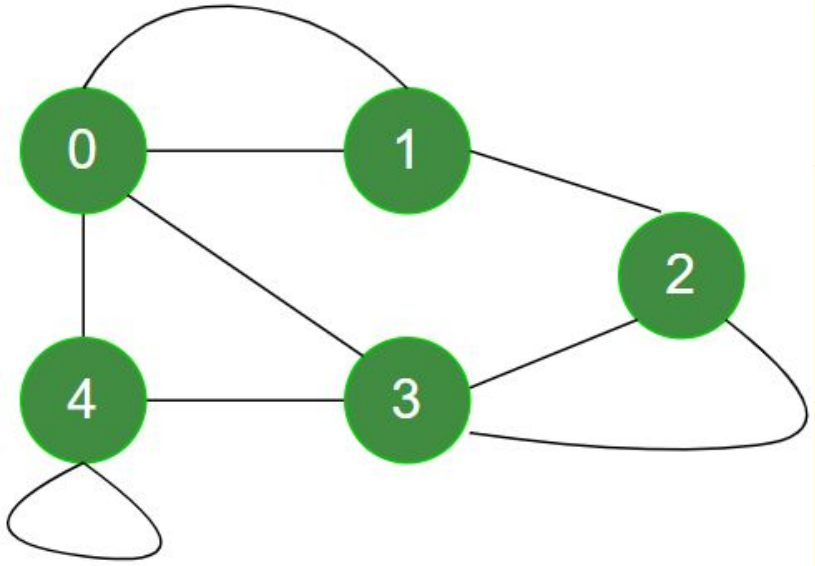


	0	1	2	3	4
0	0	1	0	0	1
1	0	0	0	1	1
2	0	1	0	0	0
3	0	0	1	0	0
4	0	0	0	1	0

MA TRẬN KỀ: dạng đặc biệt

- **Đa đồ thị**:
  - a_{ij} = k, nếu (i,j)∈E, k là số cạnh nối 2 đỉnh i và j



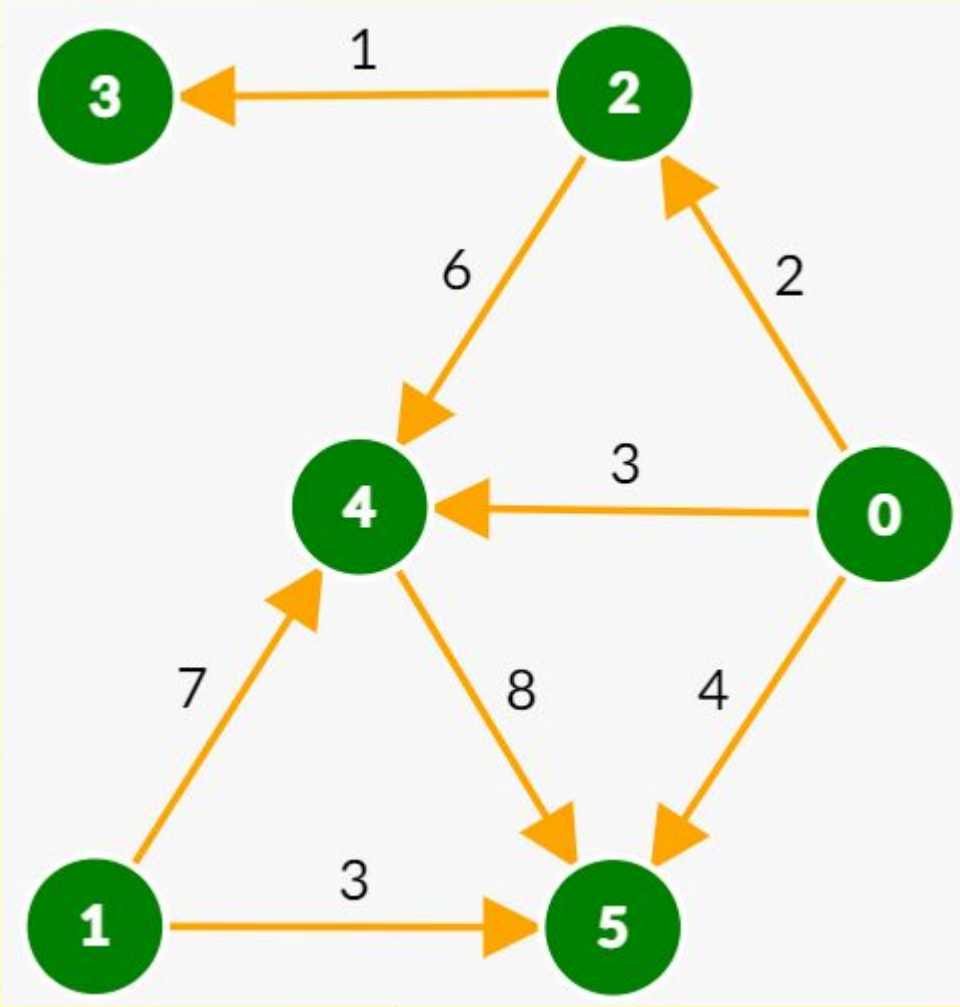


  - Câu hỏi: Ma trận sẽ như thế nào?

MA TRẬN TRỌNG SỐ

- **Đồ thị có trọng số**:
  - a_{ij} = w(i,j) là trọng số của cạnh nối 2 đỉnh i và j

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



CÀI ĐẶT: Ma trận kề/trọng số

Cách 1 (thủ công):

Bước 1: Khai báo và cấp phát động

- int **a = new int* [n]; //cấp phát động mảng con trỏ
- a[i] = new int [n]

// mỗi con trỏ a[i] quản lý một mảng động ứng với dòng thứ i của ma trận

Bước 2: Khởi tạo ma trận chứa toàn 0

Bước 3: Nhập các cạnh và lưu trữ thông tin cạnh vào ma trận kề

Cách 2: dùng STL

- vector

CÀI ĐẶT: Ma trận kề/trọng số

- ✔ Khởi tạo vector (matrix): vừa khai báo vừa khởi tạo

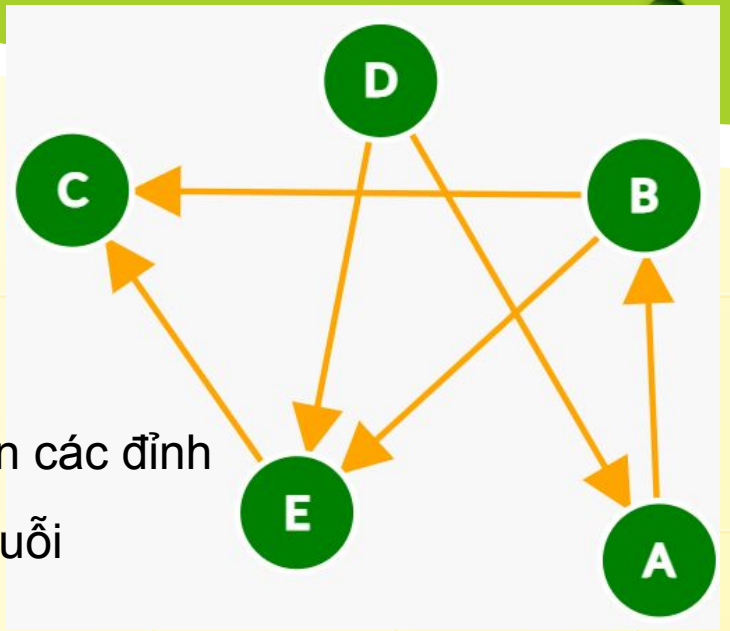
```
vector<vector<int> > G (v, vector<int> (v,0));
```

- ✔ Khai báo rồi khởi tạo sau

```
vector< vector<int> >  a;
```

```
a = vector< vector<int> >(v, vector<int>(v, 0) );
```

Nhập đơn đồ thị



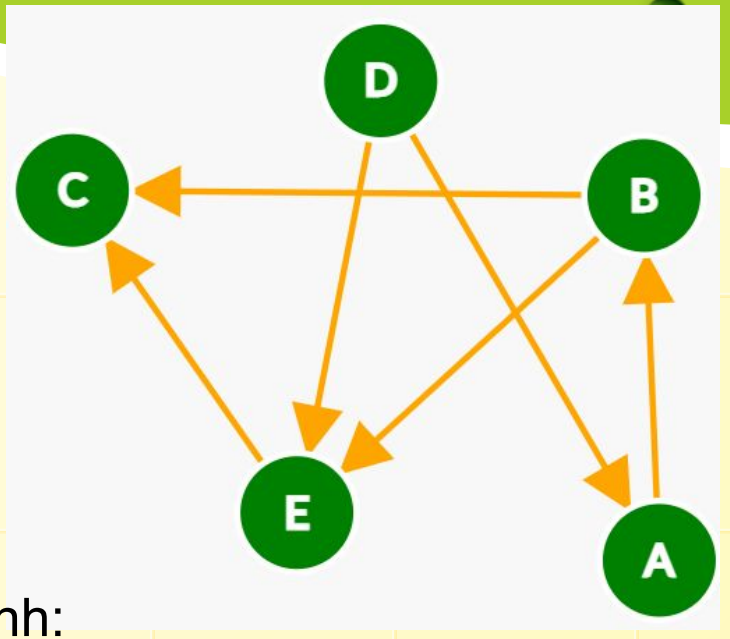
- Trường hợp 2:
  - Cho biết số đỉnh, số cạnh và danh sách tên các đỉnh
  - Mỗi đỉnh của đồ thị được đặt tên là một chuỗi

✔ Cần lưu trữ được các thông tin sau:

1. Ma trận kề của đồ thị
2. Lưu danh sách các tên (chuỗi) của đỉnh
3. Ánh xạ từ tên đỉnh (chuỗi) sang index (số nguyên) trong ma trận kề

```
5 6
A B C D E
A B
B C
D E
D A
E C
B E
```

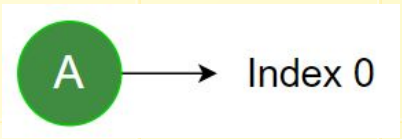
Nhập đơn đồ thị



- ✔ Xây dựng CTDL cho đồ thị:

1. Ma trận kề của đồ thị
2. Cấu trúc lưu danh sách các tên (chuỗi) đỉnh: vector/list/tree….
3. Ánh xạ từ tên đỉnh (chuỗi) sang index (số nguyên) cho biết chỉ số dòng/cột tương ứng trong ma trận

Ví dụ:



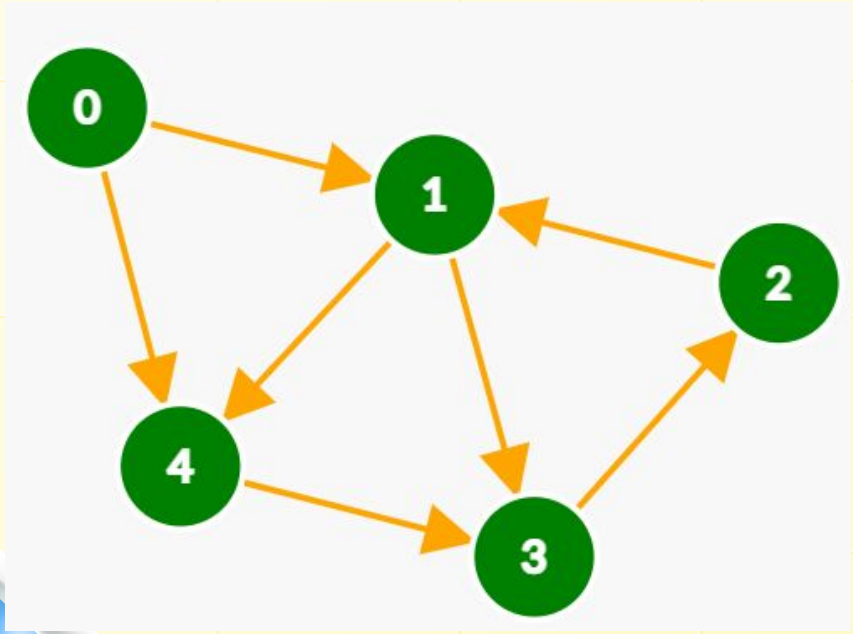

```
5 6
A B C D E
A B
B C
D E
D A
E C
B E
```

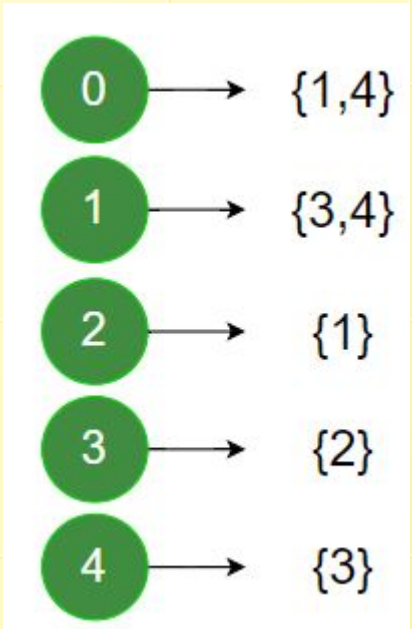
Gợi ý:

1. map<string, int> name_to_index; hoặc
2. dùng mảng lưu các đỉnh (khi muốn biết đỉnh tương ứng với index nào thì phải search)

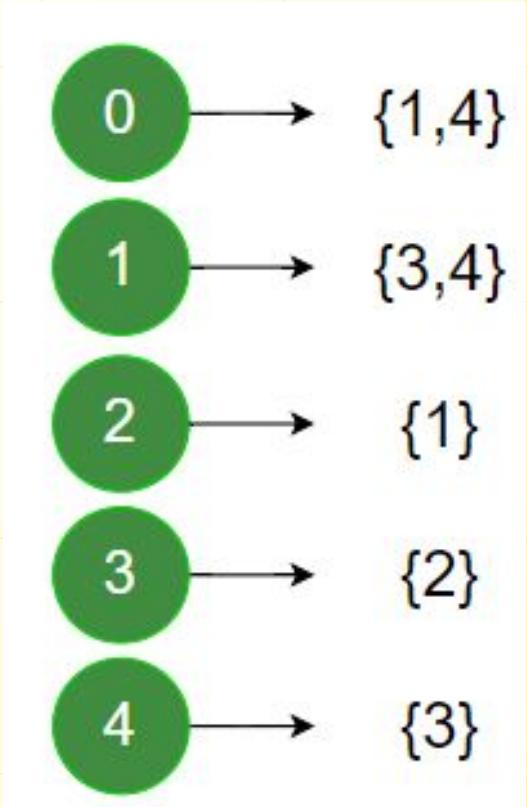
DANH SÁCH KỀ

- ✔ Với mỗi đỉnh i, cần lưu trữ được 1 tập hợp gồm các đỉnh kề với i
- ✔ Tập các đỉnh kề của i : Adj(i) = { u ∈ V | (i,u) ∈ E}
- ✔ Có nhiều cách cài đặt
- ✔ Ưu, nhược điểm?





CÀI ĐẶT: Danh sách kề



- Cách 1: Dùng map trong STL

```
map<int,set<int>> adj_list;
```

```
map<string,set<string>> adj_list;
```

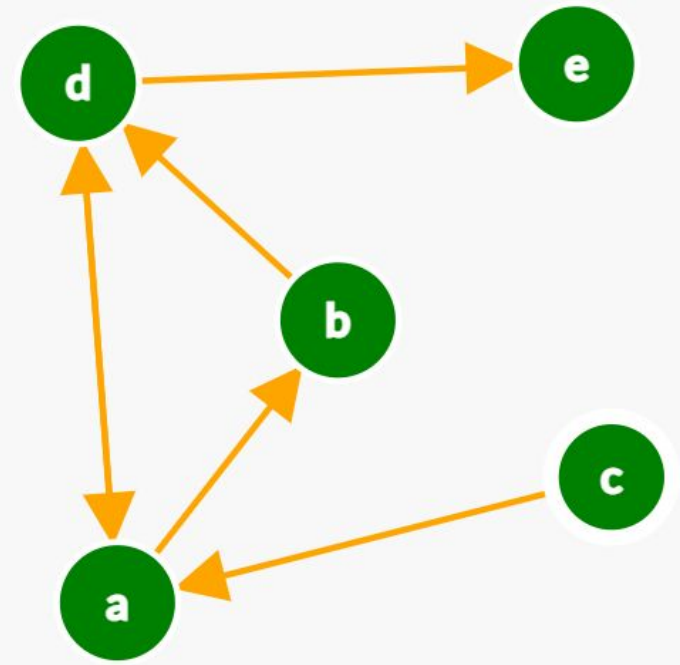
- Có thể dùng vector, list, tree thay cho set

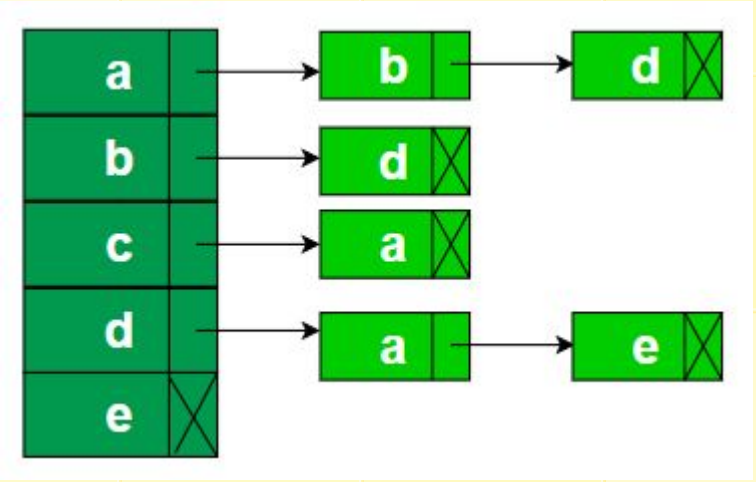
- Bỏ y vào tập kề của x (là 1 set) theo cách sau:

```
cin>>x>>y;
adj_list[x].insert(y);
```

CÀI ĐẶT: Danh sách kề

- Cách 2: Dùng các danh sách liên kết
- Giống cách cài bằng bảng băm theo kiểu nối kết trực tiếp
- Các đỉnh kề với i được nối kết trực tiếp với nhau qua một DSLK





CÀI ĐẶT: Danh sách kề

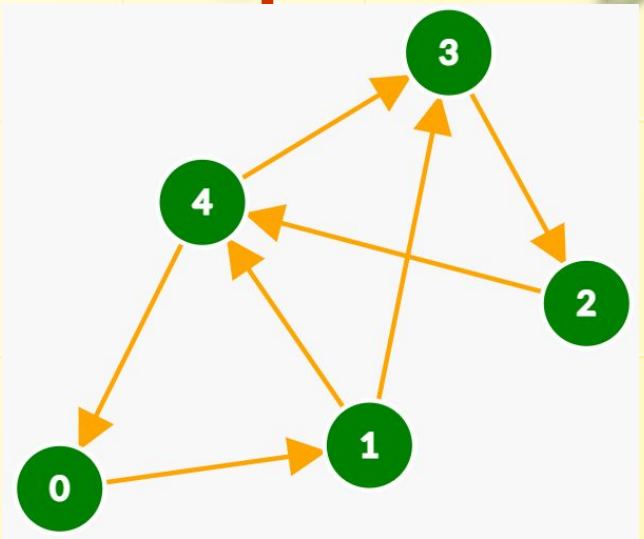
- Cách 2: Dùng các danh sách liên kết

Một số cấu trúc tham khảo	

struct node
{ string vertex;
node* next;
};
node** adjLists;
	list<string> * adj_list; //mảng của các list
	list<list<node>> adj_list; //danh sách của các danh sách
	
	dùng list trong STL

struct node
{ string vertex;
node* next;
};
struct List
{ node *head; …};
struct Graph
{ int v; // số đỉnh của đồ thị (nếu biết)
List* arr; //mảng của các danh sách
};
	vector<node> *adj_list; //mảng của các vector, vector thứ i trong mảng chứa các đỉnh kề của đỉnh thứ i trong đồ thị
	
	dùng vector trong STL

MA TRẬN LIÊN THUỘC ĐỈNH – CẠNH

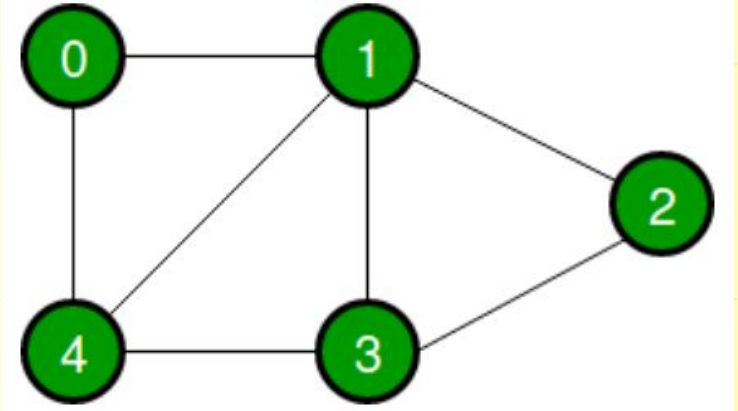


- ✓ **Đơn đồ thị có hướng** G=(V,E),
  - V = {0, 1, …, n-1}, n đỉnh
  - E = $\{e_0, e_1, \ldots, e_{m-1}\}$, m cạnh
- ✓ Ma trận 2 chiều, kích thước **n x m**
  - n dòng ↔ n đỉnh, m cột ↔ m cạnh
  - $a_{ij} = \begin{cases} 1 & \text{nếu i là đỉnh đầu của } e_j \\ -1 & \text{nếu i là đỉnh cuối của } e_j \\ 0 & \text{nếu i không là đầu mút của } e_j \end{cases}$

	(0,1)	(1,3)	(1,4)	(2,4)	(3,2)	(4,0)	(4,3)
0	1	0	0	0	0	-1	0
1	-1	1	1	0	0	0	0
2	0	0	0	1	-1	0	0
3	0	-1	0	0	1	0	-1
4	0	0	-1	-1	0	1	1

DANH SÁCH CẠNH (CUNG)

- ✔ Dùng khi đồ thị thưa (m<6n)
- ✔ Mỗi cạnh (cung) e = (x,y)
- ✔ Lưu các cặp (x,y) tương ứng điểm đầu và điểm cuối của các cạnh
- ✔ Nếu có trọng số: lưu bộ 3 (x,y,w)
- ✔ Ưu, nhược điểm?



Đầu	Cuối
0	1
1	2
1	3
1	4
2	3
3	4
4	0

Câu hỏi thảo luận

- 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. Hãy so sánh ưu nhược điểm của các phương pháp cơ bản được sử dụng để biểu diễn đồ thị trên máy tính?

Bài tập tại lớp

Cho bài toán "Tô màu bản đồ" được đặt ra như sau: Có một bản đồ các quốc gia trên thế giới, ta muốn tô màu các quốc gia này sao cho hai nước có cùng ranh giới được tô khác màu nhau. Yêu cầu tìm cách tô sao cho số màu sử dụng là ít nhất. Bài toán có thể được mô hình hóa thành một bài toán trên đồ thị, khi đó mỗi nước trên bản đồ là một đỉnh của đồ thị, hai nước láng giềng tương ứng với hai đỉnh kề nhau được nối với nhau bằng một cạnh, bài toán trở thành: tô màu các đỉnh của đồ thị sao cho mỗi đỉnh chỉ được tô một màu, hai đỉnh kề nhau có màu khác nhau và số màu sử dụng là ít nhất.

Giả sử cho thông tin đầu vào của bài toán được nhập vào chương trình như sau:

Ví dụ Input	Giải thích
15 Viet_Nam Lao Viet_Nam Trung_Quoc Thai_Lan Lao … Campuchia Thai_Lan	- Dòng đầu tiên chứa một số ***e*** là số cạnh của đồ thị - ***e*** dòng tiếp theo, mỗi dòng chứa 02 chuỗi ***u*** và ***i***, thể hiện thông tin có một cạnh nối từ đỉnh ***u*** sang đỉnh ***i*** trong đồ thị Lưu ý: không biết trước số đỉnh và danh sách các đỉnh

Hãy thực hiện các yêu cầu sau:

a. Xây dựng cấu trúc dữ liệu thích hợp để biểu diễn đồ thị nhằm lưu trữ các thông tin cần thiết trên bản đồ.

b. Viết hàm nhập đồ thị (bằng cách nhập số cạnh và danh sách các cạnh như ví dụ ở trên) và lưu trữ thông tin của đồ thị vào cấu trúc dữ liệu đã đề xuất ở câu a.

Bài tập tại lớp

ĐỀ THI CUỐI KỲ
HỌC KỲ 2 – NĂM HỌC 2021 – 2022

Trong các ứng dụng thực tế, chẳng hạn trong mạng lưới giao thông đường bộ, đường thủy hoặc đường hàng không, người ta không chỉ quan tâm đến việc tìm đường đi giữa hai địa điểm mà còn phải lựa chọn một hành trình tiết kiệm nhất (theo tiêu chuẩn không gian, thời gian hay chi phí). Vấn đề này có thể được mô hình hóa thành một bài toán trên đồ thị, trong đó mỗi địa điểm được biểu diễn bởi một đỉnh, cạnh nối hai đỉnh biểu diễn cho "đường đi trực tiếp" giữa hai địa điểm (tức không đi qua địa điểm trung gian) và trọng số của cạnh là khoảng cách giữa hai địa điểm.

Bài toán có thể phát biểu dưới dạng tổng quát như sau: Cho một đơn đồ thị có hướng và có trọng số dương G=(V,E), trong đó V là tập đỉnh, E là tập cạnh (cung) và các cạnh đều có trọng số, hãy tìm một đường đi (không có đỉnh lặp lại) ngắn nhất từ đỉnh xuất phát S thuộc V đến đỉnh đích F thuộc V.

Giả sử thông tin đầu vào của bài toán (Input) được nhập vào chương trình như sau:

Input	Giải thích
7 A B 1 B E 3 E D 3	- Dòng đầu tiên chứa một số nguyên dương e cho biết số cạnh của đồ thị - Với ***e*** dòng tiếp theo, mỗi dòng chứa hai chuỗi ***u, i*** và một số nguyên dương **x,** thể hiện thông tin có một cạnh nối từ đỉnh ***u*** sang đỉnh ***i*** trong đồ thị với độ dài (trọng số) là x

C B 4 A D 7 E C 2 C D 1 A E	- Dòng cuối cùng chứa hai chuỗi s và f, đây là đỉnh bắt đầu và đỉnh kết thúc của đường đi cần tìm **Lưu ý: không biết trước số đỉnh và danh sách các đỉnh.**

Hãy thực hiện các yêu cầu sau:

a. Xây dựng các cấu trúc dữ liệu phù hợp nhất có thể để biểu diễn đồ thị trên máy tính theo input đã cho.
   Cấu trúc được xem là tốt nếu đạt được các tiêu chuẩn sau: Tiết kiệm tài nguyên; Hỗ trợ một số thao tác cơ bản như "Kiểm tra hai đỉnh có kề nhau không", "Tìm danh sách các đỉnh kề với một đỉnh cho trước" với ràng buộc là không phải duyệt qua danh sách tất cả các cạnh của đồ thị.
b. Viết hàm nhập theo Input ở đầu bài và lưu trữ thông tin của đồ thị vào cấu trúc dữ liệu đã đề xuất ở câu a.

***** KHÔNG YÊU CẦU tìm cách giải cho bài toán này. Sinh viên ĐƯỢC PHÉP sử dụng Standard Template Library-STL với những cấu trúc dữ liệu (vector, stack, queue, list, map, set, pair, …) cũng như giải thuật được xây dựng sẵn.**