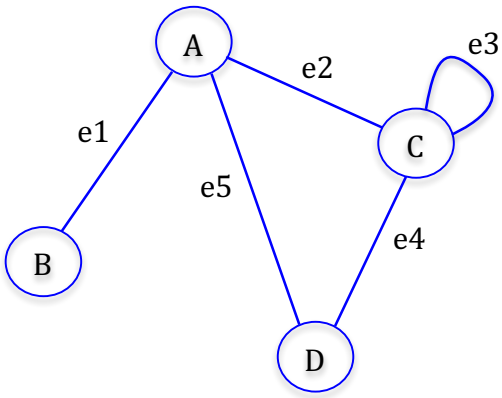
BIỂU DIỄN ĐỒ THỊ

Phạm Nguyên Khang

1.1 Giới thiệu



Theo định nghĩa, đồ thị gồm hai tập hợp: V và E. Vì thế, cách đơn giản nhất để biểu diễn (hay lưu trữ) đồ thị là biểu diễn thông tin về đỉnh và cung.

- Đỉnh: nhãn của đỉnh (thông tin về vị trí đỉnh không cần lưu trữ nếu ta không quan tâm đến việc hiển thị đồ thị chính xác như đồ thì gốc)
- Cung: mỗi cung lưu hai đầu mút của nó. Ngoài ra còn có nhiều cách biểu diễn khác nữa.

1.2 Các phương pháp biểu diễn

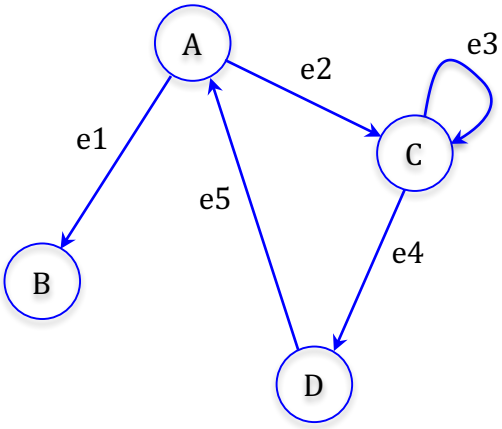
1.2.1 Phương pháp danh sách cung (edge list)

Ý tưởng:

- Mỗi cung lưu 2 đỉnh đầu mút (endpoint) của nó, hay lưu 2 số nguyên tương ứng với 2 dỉnh đầu mút này.
- Lưu tất cả các cung của đồ thị vào một danh sách (danh sách đặc hoặc danh sách liên kết)

Trong ví dụ trên, danh sách các cung bao gồm:
(A, B)
(C, A)
(C, C)
(C, D)
(A, D)

Đồ thị có hướng: chú ý thứ tự của hai đầu mút



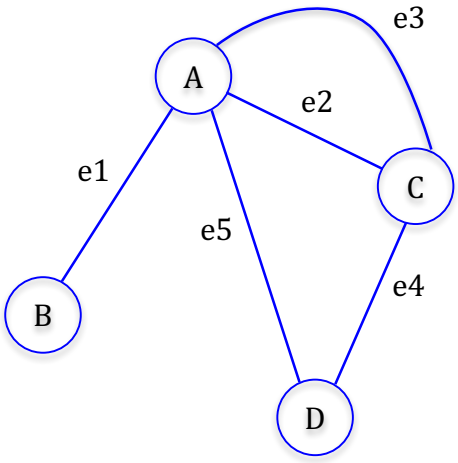
(A, B)
(A, C)
(C, C)
(C, D)
(D, A)

1.2.2 Phương pháp danh sách kề (adjacency list)

Ý tưởng:

- Với mỗi đỉnh ta lưu các đỉnh kề với nó vào trong một danh sách.
- Nếu đồ thị không chứa đa cung: danh sách đỉnh kề không chứa các phần tử trùng nhau. Ngược lại nếu đồ thị có chứa đa cung, **danh sách đỉnh kề sẽ có thể chứa nhiều đỉnh giống nhau**.
- Đồ thị sẽ bao gồm danh sách đỉnh kề của tất cả các đỉnh trong đồ thị hay một mảng các danh sách.

Ví dụ:



Các danh sách đỉnh kề:

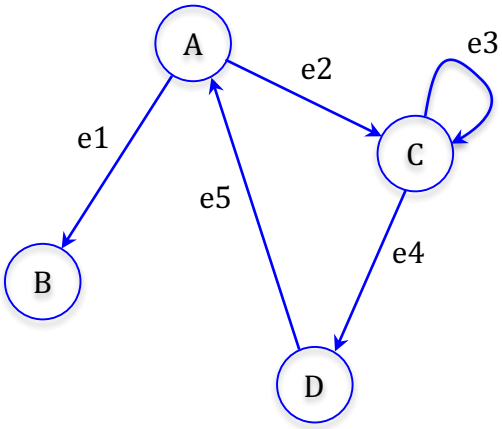
adj[A] = [B, C, C, D]

adj[B] = [A]

adj[C] = [A, A, D]

adj[D] = [A, C]

Đồ thị có hướng:



adj[A] = [B, C]

adj[B] = []

adj[C] = [C, D]

adj[D] = [A]

1.2.3 Phương pháp ma trận kề/ma trận đỉnh – đỉnh (adjacency matrix)

Ý tưởng:

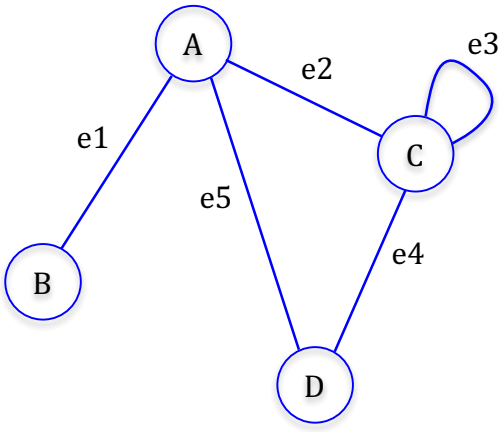
Dùng 1 ma trận vuông n hàng, n cột: A = {a_{ij}} với i = 1, 2, …, n, j = 1, 2, …, n

Phần tử hàng i, cột j có giá trị:

- a_{ij} = 1 nếu đỉnh i kề với đỉnh j.
- a_{ij} = 0 ngược lại.

Ma trận kề mô tả mối quan hệ kề nhau giữa hai đỉnh. Trường hợp đồ thị có chứa khuyên, thì phần tử a_{ii} tương ứng với đỉnh i sẽ có giá trị 1.

Ví dụ:



Ma trận kề:

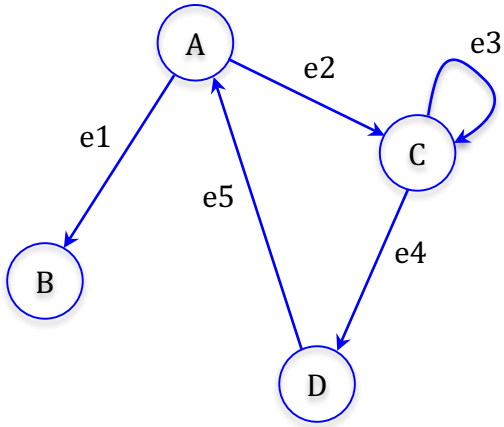
	A	**B**	**C**	**D**
A	0	1	1	1
B	1	0	0	0
C	1	0	1	1
D	1	0	1	0

Ma trận kề của đồ thị vô hướng là ma trận đối xứng.

Đồ thị có hướng:

Phần tử ở hàng i, cột j có giá trị:

- a_{ij} = 1, nếu i kề với j (có cung đi từ i đến j).
- a_{ij} = 0, nếu i không kề với j (không có cung đi từ i đến j).



Ma trận kề:

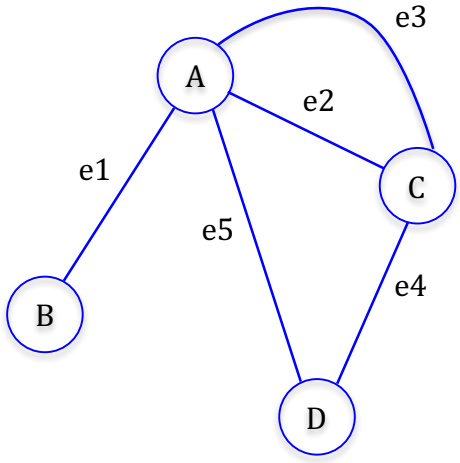
	A	**B**	**C**	**D**
A	0	1	1	0
B	0	0	0	0
C	0	0	1	1
D	1	0	0	0

Đa đồ thị (vô hướng/có hướng):

Phần tử ở hàng i, cột j có giá trị:

- a_{ij} = số cung từ i đến j.

Ví dụ:



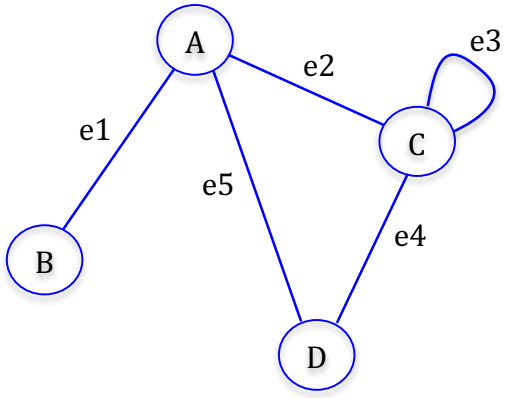
Ma trận kề:

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

1.3 Biểu diễn đồ thị trên máy tính

Để dễ dàng hơn cho việc biểu diễn các cung trên máy tính, người ta thường đánh số các đỉnh có trong đồ thị. Lưu số (nguyên) trên máy tính sẽ dễ hơn các kiểu dữ liệu khác!

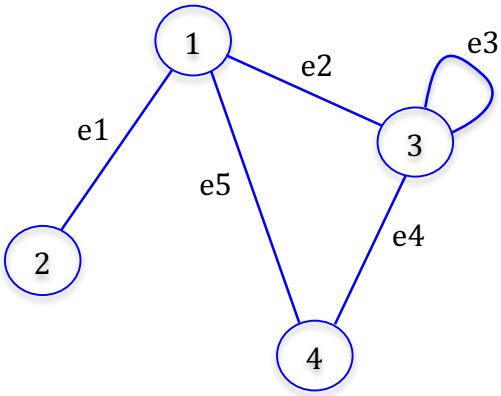
Ví dụ:



Giả sử ta đánh số các đỉnh như sau:

- A: 1
- B: 2
- C: 3
- D: 4

Đồ thị sẽ trở thành:



Sau khi đánh số xong, nếu quan tâm đến nhãn của đỉnh, ta chỉ cần thêm một mảng lables có kiểu char (nếu nhãn là ký tự) hoặc kiểu char* (nếu nhãn là chuỗi).

Việc đánh số này sẽ giúp việc biểu diễn đồ thị theo các phương pháp trên dễ dàng hơn. Trong tài liệu thực hành, đồ thị đã được giả sử là đã đánh số rồi.

1.3.1 Danh sách cung

Mỗi cung sẽ lưu hai số nguyên tương ứng với hai đầu mút của nó:

```
#define MAX_M 500

typedef struct {
  int u, v; // đỉnh đầu v, đỉnh cuối v
} Edge;

typedef struct {
  int n, m;          // n: đỉnh, m: cung
  Edge edges[MAX_M]; // lưu các cung của đồ thị
} Graph;
```

1.3.2 Danh sách kề

Mỗi đỉnh có một danh sách các đỉnh (List). Lưu 1 mảng (danh sách) các danh sách kề.

```
#define MAX_VERTICES 100

typedef struct {
  int n; /* n: so dinh */
  /* mang danh sach cac dinh ke */
  List adj[MAX_ VERTICES];
} Graph;
```

1.3.3 Ma trận kề

Sử dụng một cấu trúc gồm các trường sau:

- A[][]: mảng hai chiều lưu ma trận kề (đỉnh – đỉnh)
- n: số đỉnh

```
#define MAX_VERTICES 100

typedef struct {
  int n; /* n: so dinh */
  /* ma tran dinh – dinh */
  int A[MAX_VERTICES][MAX_VERTICES];
} Graph;
```