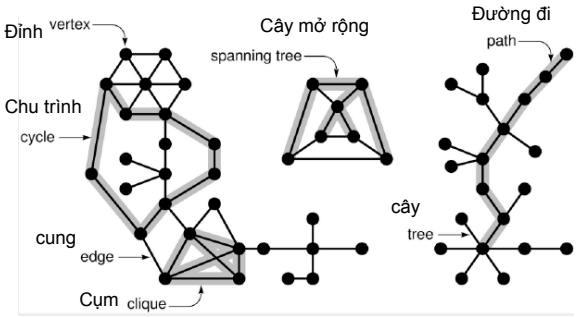
Đồ thị vô hướng

Created by anhtt-fit@mail.hut.edu.vn
Updated by huonglt-fit@mail.hut.edu.vn

Đồ thị vô hướng

- Đồ thị G=(V, E) với V (vertices) là tập các đỉnh, E là tập các cung (edges).
- Tại sao cần các thuật toán đồ thị?
  - Hay và cách trừu tượng hóa của chúng được sử dụng rộng rãi.
  - Là hướng tương đối khó trong khoa học máy tính và toán rời rạc.
  - Có hàng trăm thuật toán đồ thị.
  - Hàng nghìn ứng dụng thực tế.
    - Truyền thông, mạch, giao thông, lập lịch, các hệ thống phần mềm, mạng, trò chơi, mạng xã hội, mạng nơron,…

Các thuật ngữ

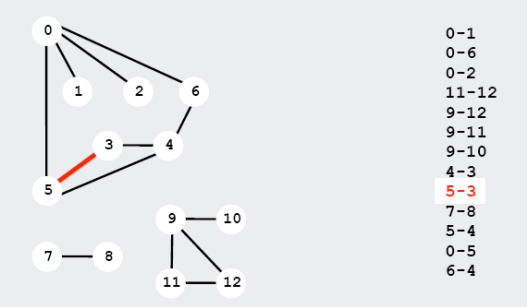


Một số vấn đề về xử lý đồ thị

- Đường đi: có 1 đường đi giữa s và t không?
- Đường đi ngắn nhất: đâu là đường đi ngắn nhất giữa s và t?
- Chu trình: có chu trình trong đồ thị không?
- Chu trình Euler: có chu trình nào đi qua mỗi cạnh đúng 1 lần không?
- Chu trình Hamilton: Có chu trình nào đi qua mỗi đỉnh ít nhất 1 lần không?
- Kết nối: có cách nào kết nối tất cả các đỉnh không?
- MST: Cách nào tốt nhất để nối tất cả các đỉnh?
- Biconnectivity: có đỉnh nào mà việc loại bỏ nó sẽ làm đồ thị không liên thông không?

Biểu diễn đồ thị

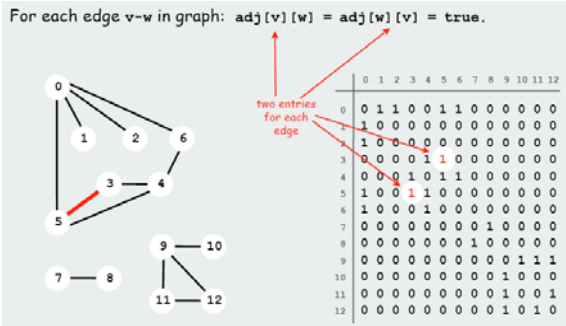
- Dùng danh sách các cạnh



- Không thích hợp cho tìm kiếm

Biểu diễn đồ thị

- Dùng ma trận liền kề



- Thích hợp cho truy cập ngẫu nhiên đến các cạnh

	A	B	C	D	E	F	G	H	I	J	K	L	M
A	1	1	1	0	0	1	1	0	0	0	0	0	0
B	1	1	0	0	0	0	0	0	0	0	0	0	0
C	1	0	1	0	0	0	0	0	0	0	0	0	0
D	0	0	0	1	1	1	0	0	0	0	0	0	0
E	0	0	0	1	1	1	1	0	0	0	0	0	0
F	1	0	0	1	1	1	0	0	0	0	0	0	0
G	1	0	0	0	1	0	1	0	0	0	0	0	0
H	0	0	0	0	0	0	0	1	1	0	0	0	0
I	0	0	0	0	0	0	0	1	1	0	0	0	0
J	0	0	0	0	0	0	0	0	0	1	1	1	1
K	0	0	0	0	0	0	0	0	0	1	1	0	0
L	0	0	0	0	0	0	0	0	0	1	0	1	1
M	0	0	0	0	0	0	0	0	0	1	0	1	1

```
#define maxV 50
int j, x, y, V, E;
int a[maxV][maxV];
adjmatrix()
  {
    scanf("%d %d\n", &V, &E);
    for (x = 1; x <= V; x++)
      for (y = 1; y <= V; y++) a[x][y] = 0;
    for (x = 1; x <= V; x++) a[x][x] = 1;
    for (j = 1; j <= E; j++)
      {
        scanf("%c %c\n", &v1, &v2);
        x = index(v1); y = index(v2);
        a[x][y] = 1; a[y][x] = 1;
      }
  }
```

V tập các chữ cái đầu tiên trong bảng mã ASCII
Ví dụ về hàm index() (cho biết ý nghĩa của hàm này?)
Int index(char c){ return(c-'A'+1); }

Cấu trúc đồ thị

- Sử dụng mảng động để biểu diễn đồ thị

```
typedef struct {
    int * matrix;
    int sizemax;
} Graph;
```

- Định nghĩa hàm API

```
Graph createGraph(int sizemax);
void setEdge(Graph* graph, int v1, int v2);
int connected(Graph* graph, int v1, int v2);
int getConnectedVertices(Graph* graph, int vertex, int[]
    output); // return số các đỉnh liên thông
```

Bài tập 1

- Để mô tả hệ thống tàu điện ngầm tại 1 thành phố, ta có thể lưu dữ liệu trong 1 file như sau:

```
[STATIONS]
S1=Name of station 1
S2=Name of station 2
…
[LINES]
M1=S1 S2 S4 S3 S7
M2=S3 S5 S6 S8 S9
…
```

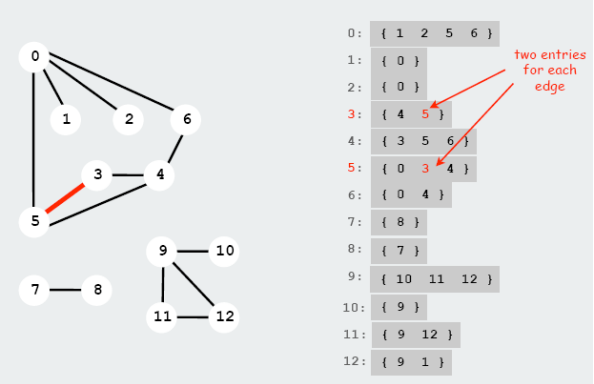
- Viết chương trình đọc file đó và thiết lập 1 mạng các ga tàu điện ngầm trong bộ nhớ máy tính sử dụng mảng 2 chiều.
- Viết 1 hàm để tìm tất cả các ga nối trực tiếp với 1 ga có tên ga nhập từ bàn phím.

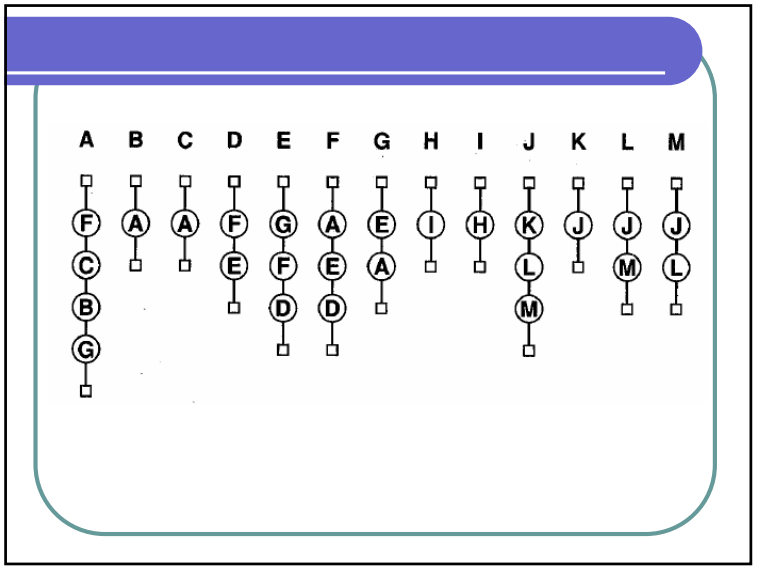
Danh sách liền kề

- Ma trận liền kề (Adjacency Matrix) có thể được ưa chuộng khi đồ thị dày, hoặc khi ta cần có kết luận nhanh chóng về việc có 1 cạnh nối 2 đỉnh cho trước hay không
- Danh sách liền kề (Adjacency List) thường được ưa chuộng hơn, vì nó cung cấp 1 cách thức cô đọng nhất để biểu diễn đồ thị thưa, với $|E| << |V|^2$

Biểu diễn đồ thị

- Dùng danh sách liền kề






```
#define maxV 1000
struct node
  { int v; struct node *next; };
int j, x, y, V, E;
struct node *t, *z;
struct node *adj[maxV];
adjlist()
  {
    scanf("%d %d\n", &V, &E);
    z = (struct node *) malloc(sizeof *z);
    z->next = z;
    for (j = 1; j <= V; j++) adj[j] = z;
    for (j = 1; j <= E; j++)
      {
        scanf("%c %c\n", &v1, &v2);
        x = index(v1); y = index(v2);
        t = (struct node *) malloc(sizeof *t);
        t->v = x; t->next = adj[y]; adj[y] = t;
        t = (struct node *) malloc(sizeof *t);
        t->v = y; t->next = adj[x]; adj[x] = t;
      }
  }
```

Cài đặt

- Cây đỏ đen có thể dùng để lưu trữ đồ thị này: mỗi nút trên cây là 1 đỉnh và giá trị của nó ứng với 1 tập các đỉnh có kết nối với nó.
- Tập các đỉnh kết nối cũng được lưu trữ trong câu đỏ đen.

Bài tập 2

- Sử dụng thư viện libfdr để cài đặt 1 API để thao tác với đồ thị, với khai báo như sau:

```
typedef JRB Graph;
Graph createGraph();
void setEdge(Graph* graph, int v1, int v2);
int connected(Graph* graph, int v1, int v2);
void forEachConnectedVertex(Graph* graph, int vertex, void
   (*func)(int, int) );
// hàm cuối cùng là hàm hoa tiêu: lặp với tất cả các đỉnh liên
   thông để làm 1 việc gì đó. func là 1 con trỏ trỏ đến hàm xử lý
   trên các đỉnh kết nối.
```

- Viết lại chương trình quản lý mạng tàu điện ngầm sử dụng API mới này.