BÀI 2
BIỂU DIỄN ĐỒ THỊ TRÊN MÁY TÍNH

Mục tiêu

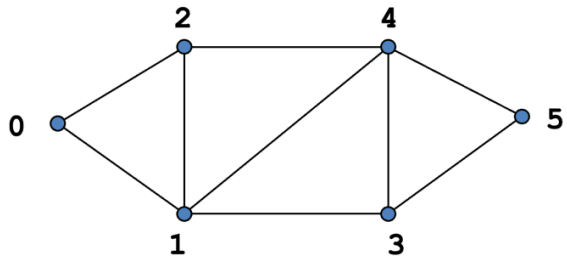
- Nắm vững các phương pháp biểu diễn đồ thị trên máy tính;
- Đánh giá ưu, nhược điểm của các phương pháp;
- Cài đặt được các cấu trúc dùng để biểu diễn đồ thị trên máy tính.

1. Ma trận kề (Adjacency Matrix)

Định nghĩa: Cho đồ thị $\boldsymbol{G} = (\boldsymbol{V}, \boldsymbol{E})$ gồm n đỉnh. Ma trận kề của đồ thị $\boldsymbol{G}$ là một ma trận vuông $a[n\ x\ n]$ có đặc điểm sau:

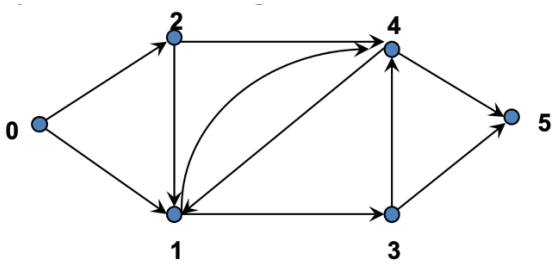
- $\boldsymbol{G}$ không có trọng số: $\boldsymbol{a[u,v] = 1}$ nếu đỉnh $\boldsymbol{u}$ **kề đỉnh** $\boldsymbol{v}$. Ngược lại, $\boldsymbol{a[u,v] = 0}$ nếu đỉnh $\boldsymbol{u}$ **không kề đỉnh** $\boldsymbol{v}$.
- $\boldsymbol{G}$ có trọng số: $\boldsymbol{a[u,v] = w}$ nếu đồ thị có cạnh $(\boldsymbol{u,v})$ có trọng số là $\boldsymbol{w}$. Ngược lại, $\boldsymbol{a[u,v] = 0}$.

Bài tập 1. Cho đồ thị $\boldsymbol{G} = (\boldsymbol{V}, \boldsymbol{E})$ gồm 6 đỉnh như hình vẽ bên dưới. Xác định ma trận kề của đồ thị.



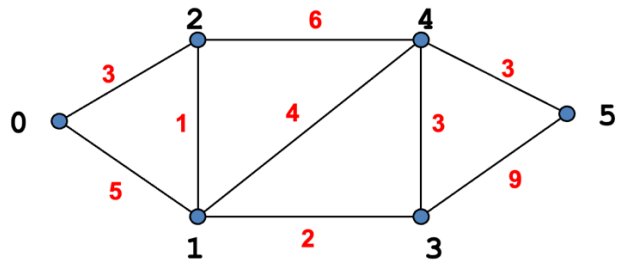
	0	1	2	3	4	5
0	0	1	1	0	0	0
1	1	0	1	1	1	0
2	1	1	0	0	1	0
3	0	1	0	0	1	1
4	0	1	1	1	0	1
5	0	0	0	1	1	0

Bài tập 2. Cho đồ thị $\boldsymbol{G} = (\boldsymbol{V}, \boldsymbol{E})$ gồm 6 đỉnh như hình vẽ bên dưới. Xác định ma trận kề của đồ thị.



	0	1	2	3	4	5
0	0	1	1	0	0	0
1	0	0	0	1	1	0
2	0	1	0	0	1	0
3	0	0	0	0	1	1
4	0	1	0	0	0	1
5	0	0	0	0	0	0

Bài tập 3. Cho đồ thị $\boldsymbol{G} = (\boldsymbol{V}, \boldsymbol{E})$ gồm 6 đỉnh như hình vẽ bên dưới. Xác định ma trận kề của đồ thị.

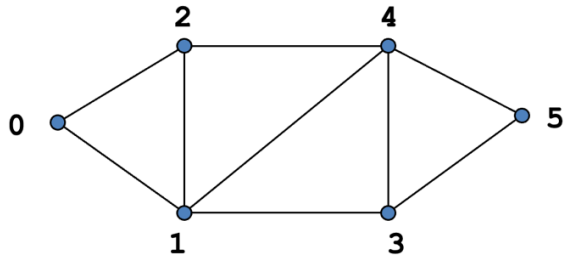


	0	1	2	3	4	5
0	0	5	3	0	0	0
1	5	0	1	2	4	0
2	3	1	0	0	6	0
3	0	2	0	0	3	9
4	0	4	6	3	0	3
5	0	0	0	9	3	0

Các tính chất của ma trận kề:

- Ma trận kề của đơn đồ thị có các phần tử trên đường chéo chính bằng 0;
- Nếu đồ thị vô hướng:
  - Ma trận kề đối xướng qua đường chéo chính ($a[u,v] = a[v,u]$);
  - Bậc của đỉnh u là số phần tử khác 0 trên dòng u (hoặc cột u).
- Nếu đồ thị có hướng:
  - Bậc ra của đỉnh u là số phần tử khác 0 trên dòng u;
  - Bậc vào của đỉnh u là số phần tử khác 0 trên cột u.

Cấu trúc dữ liệu: int[,] adjMatrix;



Input.txt

```
6
0 1 1 0 0 0
1 0 1 1 1 0
1 1 0 0 1 0
0 1 0 0 1 1
0 1 1 1 0 1
0 0 0 1 1 0
```

Nhận xét:

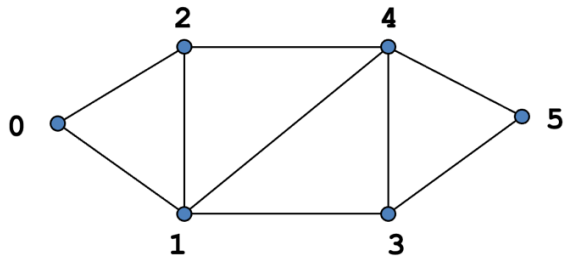
- Ưu điểm:
  - Đơn giản, dễ cài đặt;
  - Kiểm tra nhanh 2 đỉnh bất kỳ có kề nhau hay không (độ phức tạp $O(1)$).
- Nhược điểm:
  - Tốn bộ nhớ: ma trận kề luôn chiếm $O(n^2)$ ô nhớ, rất lãng phí bộ nhớ khi đồ thị thưa (*số đỉnh lớn nhưng có rất ít cạnh*);
  - Chỉ phù hợp với đồ thị có kích thước nhỏ (thường $n \leq 1000$).

2. Danh sách cạnh (Edge List)

Định nghĩa: Cho đồ thị $\boldsymbol{G} = (\boldsymbol{V}, \boldsymbol{E})$ gồm m cạnh. Danh sách cạnh của đồ thị $\boldsymbol{G}$ là một danh sách gồm m phần tử, mỗi phần tử là một cặp giá trị (u, v) tương ứng với cạnh (u, v) của đồ thị.

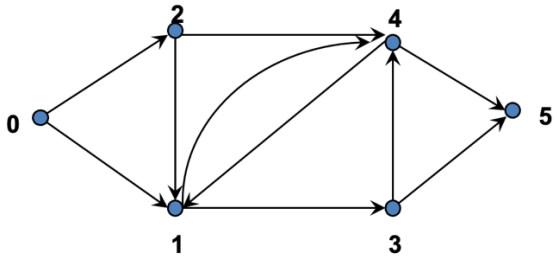
Đối với đồ thị có trọng số, mỗi phần tử là một bộ 3 giá trị (u, v, w) tương ứng với cạnh (u, v) có trọng số w.

Bài tập 1. Cho đồ thị $\boldsymbol{G} = (\boldsymbol{V}, \boldsymbol{E})$ gồm 6 đỉnh, 9 cạnh như hình vẽ bên dưới. Xác định danh sách cạnh của đồ thị.



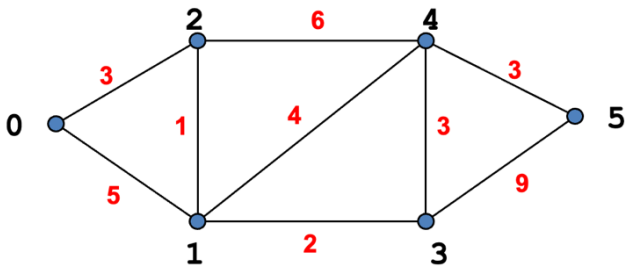

```
9
0 1
0 2
1 2
1 3
1 4
2 4
3 4
3 5
4 5
```

Bài tập 2. Cho đồ thị $\boldsymbol{G} = (\boldsymbol{V}, \boldsymbol{E})$ gồm 6 đỉnh, 10 cạnh như hình vẽ bên dưới. Xác định danh sách cạnh của đồ thị.




```
10
0 1
0 2
1 3
1 4
2 1
2 4
3 4
3 5
4 1
4 5
```

Bài tập 3. Cho đồ thị $\boldsymbol{G} = (\boldsymbol{V}, \boldsymbol{E})$ gồm 6 đỉnh, 9 cạnh như hình vẽ bên dưới. Xác định danh sách cạnh của đồ thị.




```
9
0 1 5
0 2 3
1 2 1
1 3 2
1 4 4
2 4 6
3 4 3
3 5 9
4 5 3
```

Cấu trúc dữ liệu:

- **Đồ thị không có trọng số**: mỗi cạnh được biểu diễn dưới dạng một Tuple gồm 2 thành phần (u, v)
  - Dùng mảng một chiều: **Tuple<int, int>[] edges;**
  - Dùng List: **List<Tuple<int, int>> edges**
  - Dùng LinkedList: **LinkedList<Tuple<int, int>> edges**;
- **Đồ thị có trọng số**: mỗi cạnh được biểu diễn dưới dạng một Tuple gồm 3 thành phần (u, v, w)
  - Dùng mảng một chiều: **Tuple<int, int, int>[] edges;**
  - Dùng List: **List<Tuple<int, int, int>> adjList;**
  - Dùng LinkedList: **LinkedList<Tuple<int, int, int>> edges**;
- Có thể thay thế kiểu Tuple bằng **cấu trúc cạnh (*Edge*)** do người dùng tự định nghĩa:

```
class Edge
{
    public int From { get; }    // Đỉnh bắt đầu
    public int To { get; }      // Đỉnh kết thúc
    public int Weight { get; }  // Trọng số (tùy chọn)
    // Khởi tạo cạnh. Mặc định weight=0: cạnh không trọng số
    public Edge(int from, int to, int weight = 0)
    {
        From = from;
        To = to;
        Weight = weight;
    }
    // Trả về chuỗi biểu diễn thông tin cạnh
    public override string ToString()
    {
        if (Weight == 0)
            return $"{From} -> {To}";
        else
            return $"{From} --({Weight})--> {To}";
    }
}
```

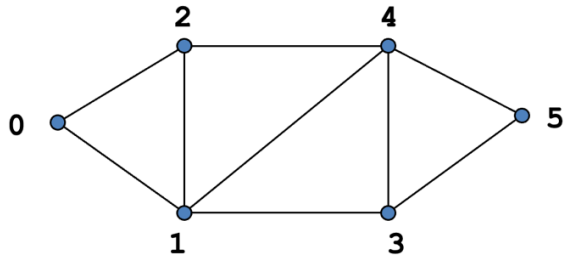
Nhận xét:

- Ưu điểm:
  - Tiết kiệm bộ nhớ, đặc biệt trong trường hợp đồ thị thưa;
  - Hiệu quả khi cần duyệt qua tất cả các cạnh (*phù hợp với các thuật toán cần duyệt tập cạnh như Kruskal*).
- Nhược điểm: Thao tác kiểm tra mối quan hệ giữa 2 đỉnh không hiệu quả vì luôn phải duyệt qua toàn bộ danh sách cạnh.

3. Danh sách kề (Adjcency List)

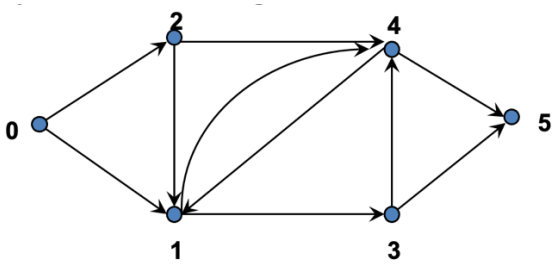
Định nghĩa: Cho đồ thị $\boldsymbol{G} = (\boldsymbol{V}, \boldsymbol{E})$ gồm n đỉnh. Danh sách kề của đồ thị $\boldsymbol{G}$ là một danh sách gồm n phần tử, mỗi phần tử đại diện cho một đỉnh và là một danh sách con chứa các đỉnh kề với đỉnh đó trong đồ thị.

Bài tập 1. Cho đồ thị $\boldsymbol{G} = (\boldsymbol{V}, \boldsymbol{E})$ gồm 6 đỉnh như hình vẽ bên dưới. Xác định danh sách kề của đồ thị.



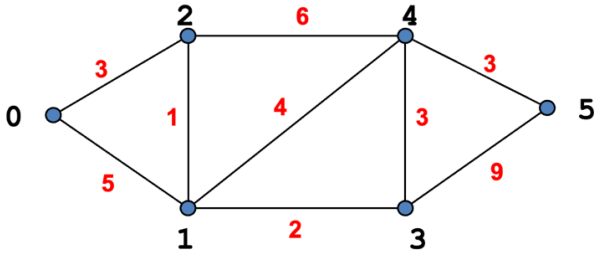
Đỉnh u	Danh sách kề của u
0	1 2
1	0 2 3 4
2	0 1 4
3	1 4 5
4	1 2 3 5
5	3 4

Bài tập 2. Cho đồ thị $\boldsymbol{G} = (\boldsymbol{V}, \boldsymbol{E})$ gồm 6 đỉnh như hình vẽ bên dưới. Xác định danh sách kề của đồ thị.



Đỉnh u	Danh sách kề của u
0	1 2
1	3 4
2	1 4
3	4 5
4	1 5
5	∅

Bài tập 3. Cho đồ thị $\boldsymbol{G} = (\boldsymbol{V}, \boldsymbol{E})$ gồm 6 đỉnh như hình vẽ bên dưới. Xác định ma trận kề của đồ thị.



Đỉnh u	Danh sách kề của u
0	1 5 2 3
1	0 5 2 1 3 2 4 4
2	0 3 1 1 4 6
3	1 2 4 3 5 9
4	1 4 2 6 3 3 5 3
5	3 9 4 3

Chú ý: Khi xử lý danh sách kề của mỗi đỉnh u cần xử lý các cặp số v, w liền kề nhau → cạnh (u, v, w)

Cấu trúc dữ liệu:

- **Đồ thị không có trọng số**:
  - Dùng mảng một chiều chứa các danh sách: **List<int>[] adjList;**
  - Dùng danh sách chứa các danh sách: **List<List<int>> adjList**;
  - Dùng kiểu từ điển: Dictionary<int, List<int>> adjList; (***key**: đỉnh u, **value**: danh sách các đỉnh kề với u*);
  - Có thể lưu danh sách kề của mỗi đỉnh dưới dạng một danh sách liên kết để tăng hiệu quả các thao tác thêm, xóa đỉnh kề so với danh sách thông thường, ví dụ: **List<LinkedList<int>> adjList**;
- **Đồ thị có trọng số**: mở rộng mỗi phần tử trong danh sách kề của đỉnh u thành một cặp giá trị (v, w) (*tham khảo cách làm ở cấu trúc Danh sách cạnh*).

Nhận xét:

- Ưu điểm:
  - Khắc phục được một phần nhược điểm của ma trận kề cũng như danh sách cạnh: *tiết kiệm bộ nhớ nhưng vẫn hiệu quả khi cần thao tác nhanh với các đỉnh kề của một đỉnh*;
  - Tính bậc của một đỉnh nhanh chóng: *bậc của một đỉnh bằng kích thước danh sách kề của đỉnh đó*.
- Nhược điểm:
  - Phức tạp và khó cài đặt hơn so với ma trận kề và danh sách cạnh;
  - Kém hiệu quả hơn ma trận kề nếu đồ thị dày đặc và có trọng số (*mỗi cạnh cần 2 ô nhớ để lưu đỉnh kề và trọng số*).

BÀI TẬP VỀ NHÀ

Định nghĩa lớp Graph mô tả một đồ thị:

- **Dữ liệu** (*Fields*): số đỉnh, số cạnh, ma trận kề, danh sách cạnh, danh sách kề;
- **Phương thức khởi tạo** (*Constructor*): cài đặt các phương thức khởi tạo phù hợp;
- **Phương thức** (*Methods*): đọc và in đồ thị dưới dạng ma trận kề, danh sách cạnh, danh sách kề.

--- HẾT ---