LECTURE 15

GRAPH

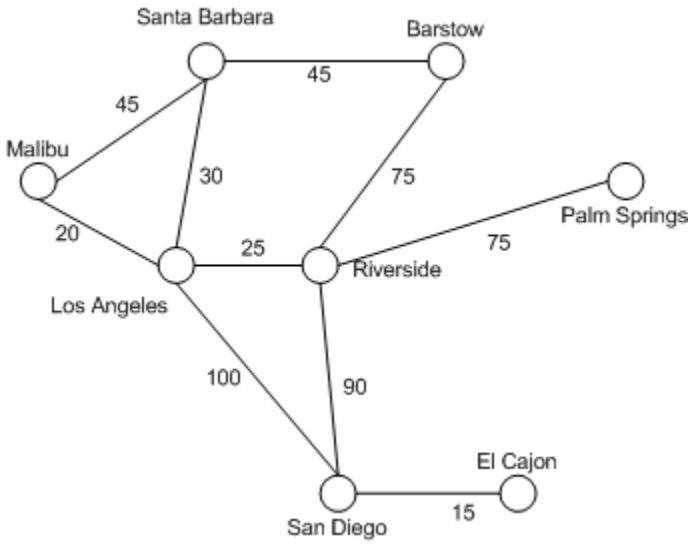
Big-O Coding

Website: www.bigocoding.com

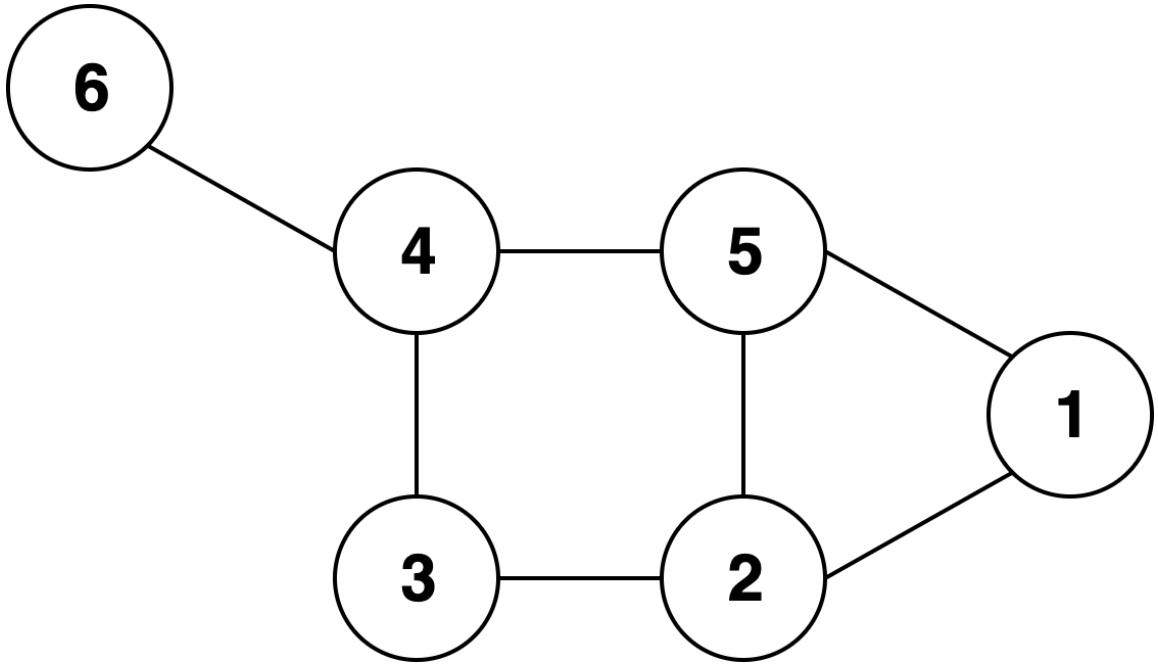
Khái niệm

- **Graph** (đồ thị) là một data structure (cấu trúc dữ liệu) bao gồm các **vertex** (đỉnh), các **edge** (cạnh) nối đến các đỉnh đó và trọng số giữa chúng: G = (V, E, W).
  - V: tập hợp các đỉnh.
  - E: tập hợp các cạnh nối giữa các đỉnh.
  - W: trọng số trên mỗi cạnh.

Graph trong cuộc sống

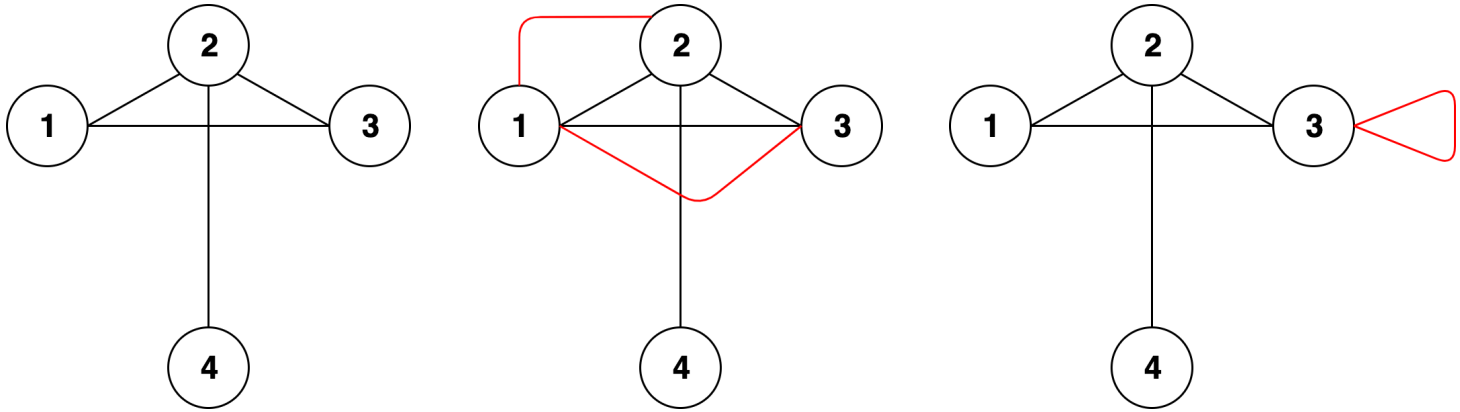


Graph trong lập trình



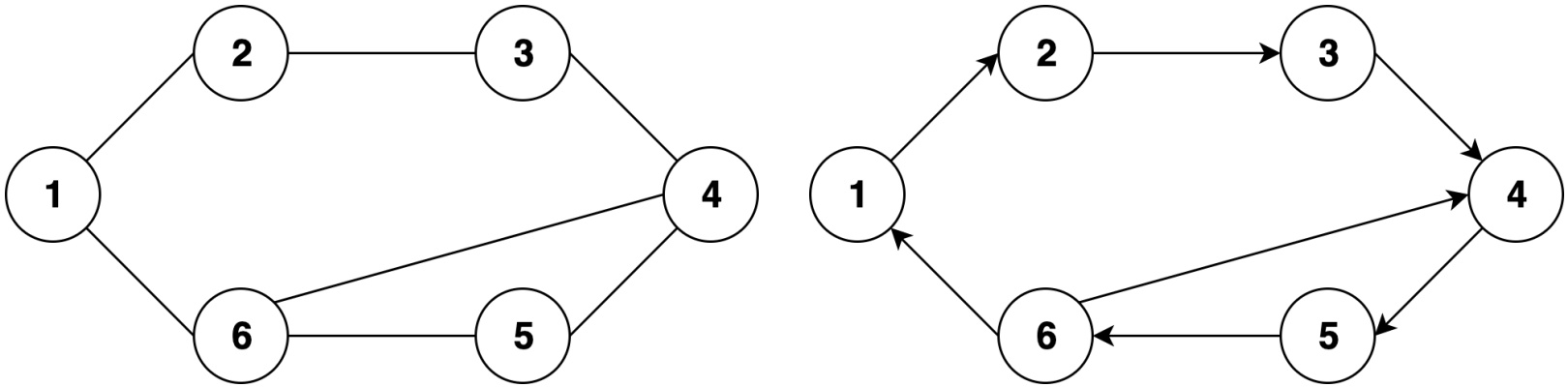
Simple graph

- **Simple graph:** Đồ thị đơn, giữa 2 đỉnh bất kì trong đồ thị: hoặc là ko có cạnh nối, hoặc là chỉ có 1 cạnh nối 2 đỉnh đó.
- **Nonsimple graph:** đồ thị đa, giữa 2 đỉnh có thể có nhiều hơn 1 cạnh nối.
- **Graph with loop:** đồ thị có khuyên, đỉnh bắt đầu và kết thúc của 1 cạnh cùng là 1 đỉnh duy nhất.



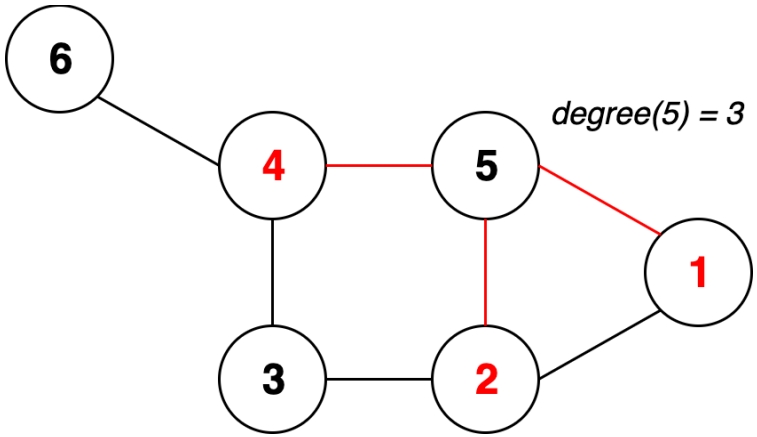
Directed graph

- **Undirected graph:** đồ thị vô hướng là đồ thị mà tất cả các cạnh không có định hướng.
- **Directed graph:** đồ thị có hướng là đồ thị mà tất cả các cạnh đều có định hướng.



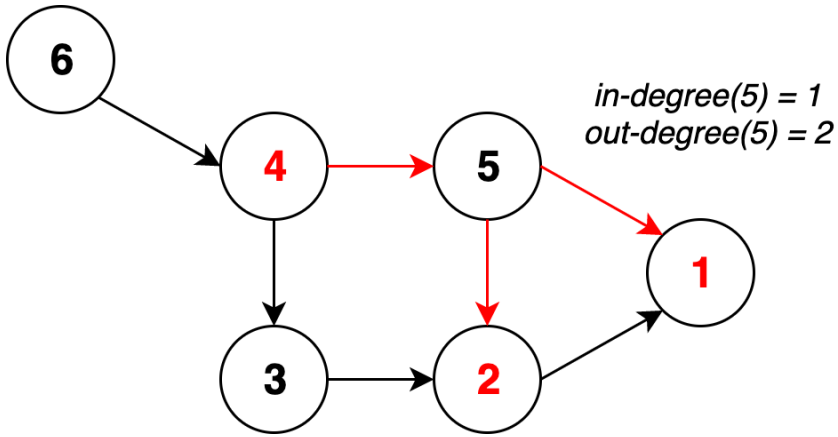
Degree

- Nếu một cạnh e, nối 2 đỉnh u và v
  - 2 đỉnh u và v được gọi là 2 đỉnh kề nhau (adjacent).
- Bậc của đỉnh: degree(v) = số cạnh nối đến đỉnh v = số đỉnh kề với đỉnh v.
  - Đồ thị G có m cạnh, thì tổng các bậc của tất cả các đỉnh trong đồ thị là 2m.



Degree

- Với directed graph:
  - out-degree(v) = số cạnh **đi ra khỏi** đỉnh v.
  - in-degree(v) = số cạnh **đi vào** đỉnh v.

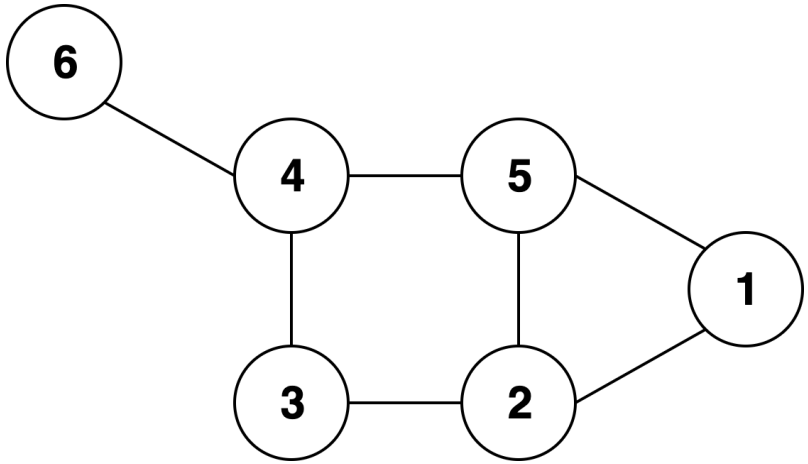


Path & circuit

- **Path:** đường đi, bao gồm các đỉnh v0, v1, …, vp đi qua các cạnh (v0, v1), (v1, v2), …, (vp-1, vp).
  - Khi đó, ta nói v0 đến được vp từ path p.
- **Circuit:** chu trình, là một đường đi mà v0 = vp.

Ví dụ

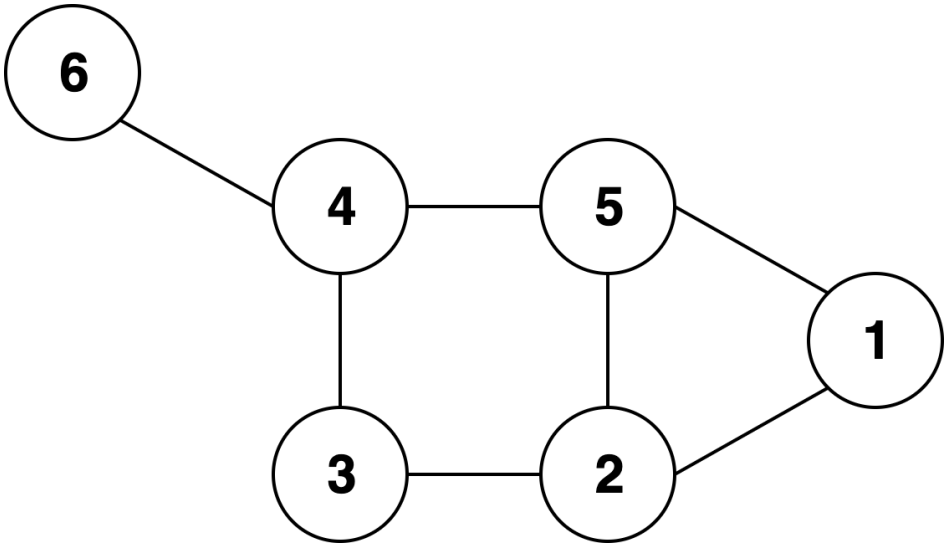
- Các path từ đỉnh 1 đến đỉnh 6:
  - (1, 5), (5, 4), (4, 6)
  - (1, 2), (2, 3), (3, 4), (4, 6)
  - (1, 2), (2, 5), (5, 4), (4, 6)



- Circuit (1, 2), (2, 3), (3, 4), (4, 5), (5, 1).

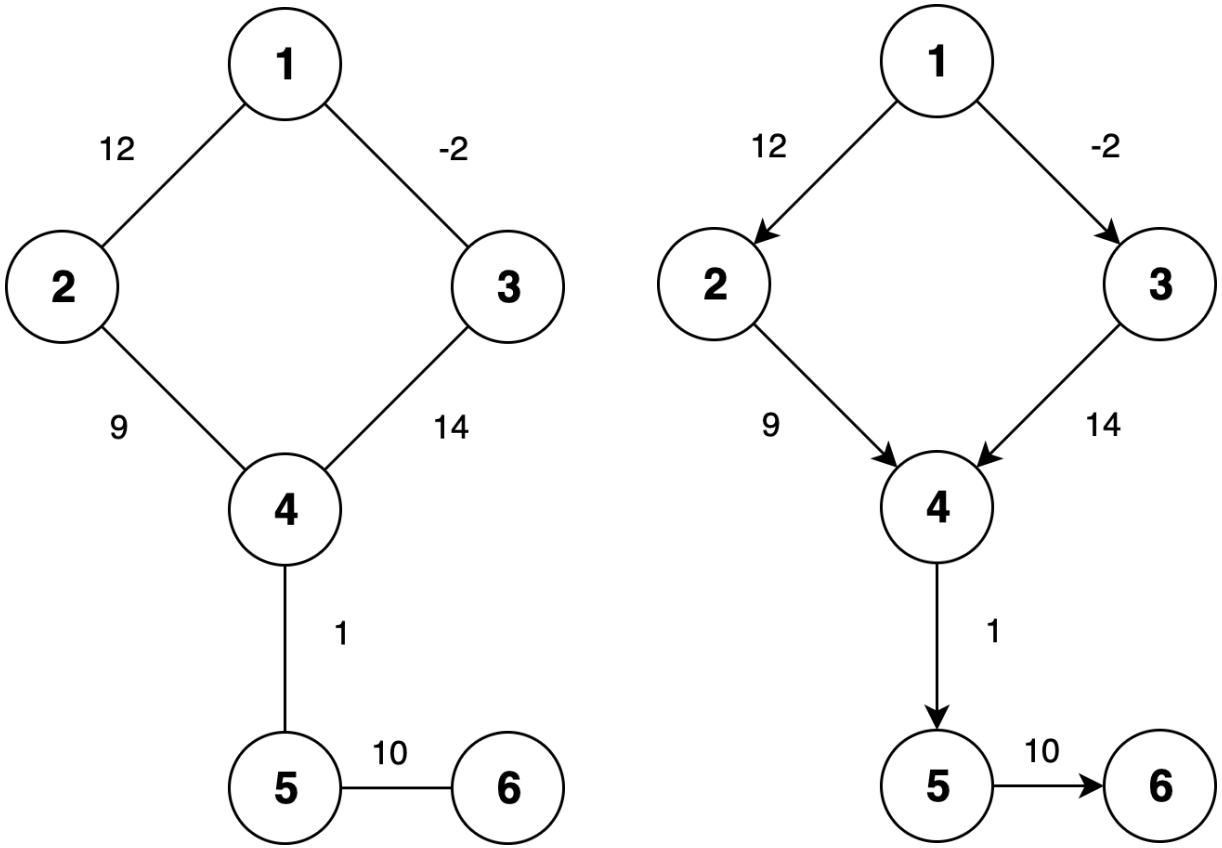
Connected graph

- Một đồ thị vô hướng gọi là liên thông (**connected**) nếu mọi cặp đỉnh (u, v) ta đều tìm được đường đi từ u đến v.



Weighted graph

- Weighted graph: đồ thị có trọng số, là đồ thị mà mỗi cạnh được gắn với một trọng số nhất định.

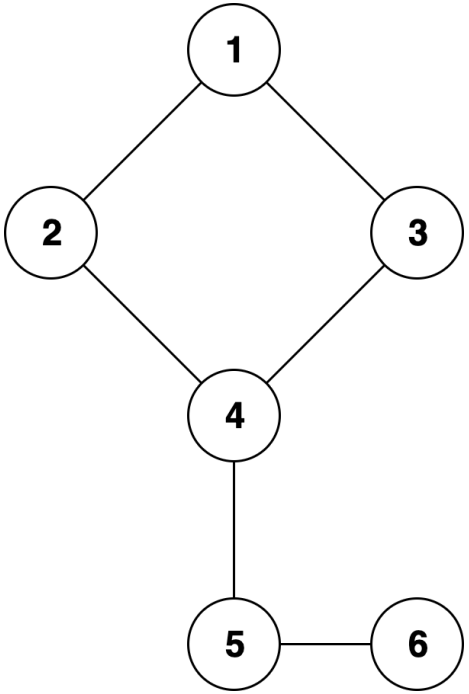


3 cách biểu diễn đồ thị

- Adjacency matrix – ma trận kề
- Edge list – danh sách cạnh
- Adjacency list – danh sách kề

Undirected graph & Adjacency matrix

- a[i][j] = 1: có cạnh nối từ đỉnh i đến đỉnh j.
- a[i][j] = 0: không có cạnh nối từ đỉnh i đến đỉnh j.
- Với đồ thị vô hướng, a[i][j] = a[j][i].



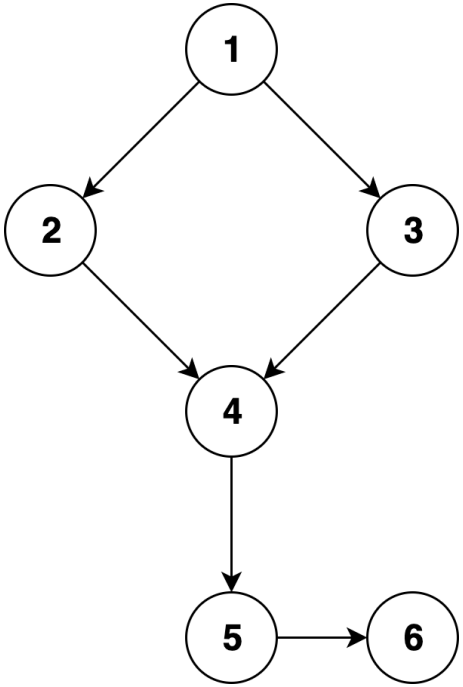
0	1	1	0	0	0
1	0	0	1	0	0
1	0	0	1	0	0
0	1	1	0	1	0
0	0	0	1	0	1
0	0	0	0	1	0

Adjacency matrix

- Ma trận kề.
- Một mảng 2 chiều.
- Kích thước N dòng x N cột.
- Giá trị mỗi ô cho biết có cạnh nào nối giữa đỉnh i và đỉnh j hay không.

Directed graph & Adjacency matrix

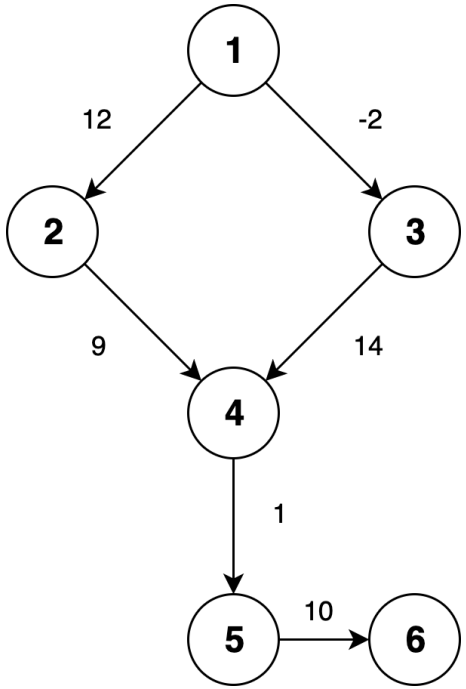
- Với đồ thị có hướng a[i][j] ≠ a[j][i].



0	1	1	0	0	0
0	0	0	1	0	0
0	0	0	1	0	0
0	0	0	0	1	0
0	0	0	0	0	1
0	0	0	0	0	0

Weighted directed graph & Adjacency matrix

- Với đồ thị có trọng số, a[i][j] = trọng số của cạnh nối đỉnh i và đỉnh j.



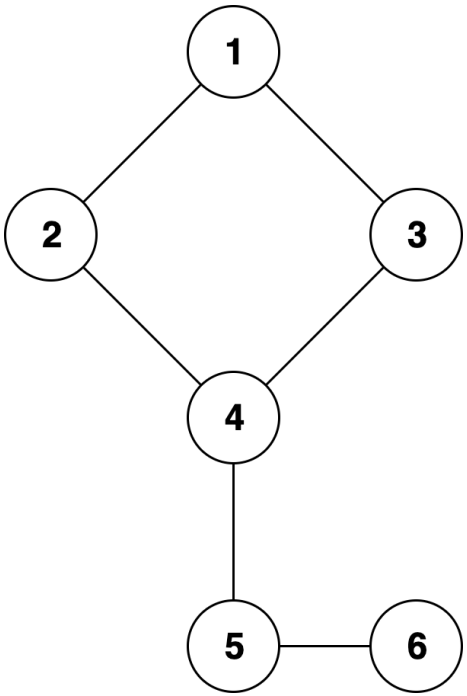
0	12	-2	0	0	0
0	0	0	9	0	0
0	0	0	14	0	0
0	0	0	0	1	0
0	0	0	0	0	10
0	0	0	0	0	0

Nhận xét

- Đối với đồ thị vô hướng,
  - Ma trận kề là ma trận đối xứng, nghĩa là a[i][j] = a[j][i].
  - Degree(i) = Tổng các số trên hàng i = tổng các số trên cột i.
- Đối với đồ thị có hướng,
  - out-degree(i) = tổng các số trên hàng i.
  - in-degree(i) = tổng các số trên cột i.

Edge list

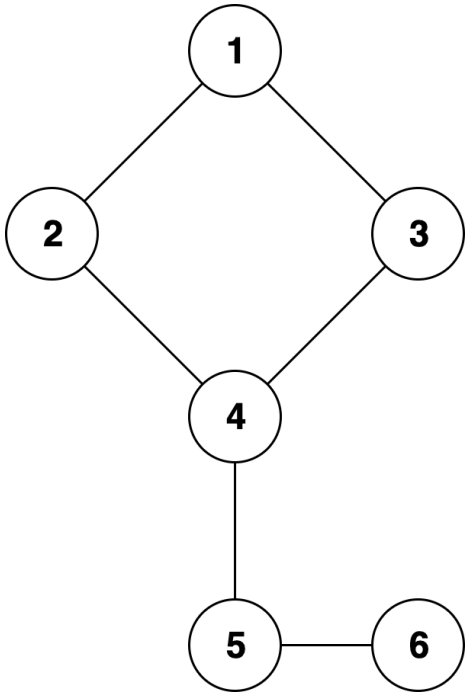
- Edge list là một danh sách lưu tất cả các cạnh trong đồ thị.



(1, 2)	(1, 3)	(2, 4)	(3, 4)	(4, 5)	(5, 6)

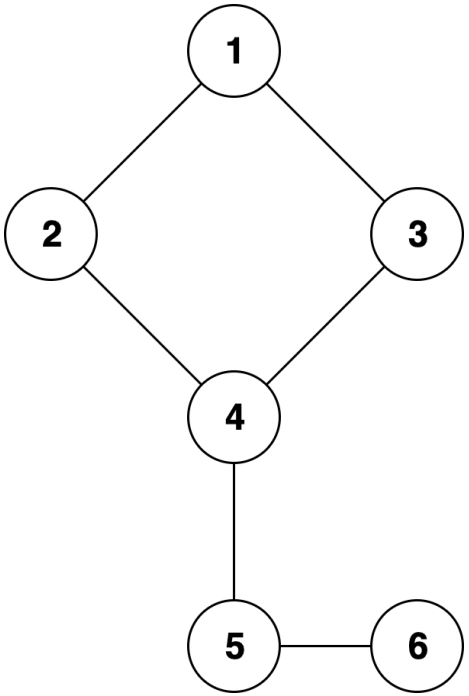
Adjacency list

- Adjacency list được tạo ra bằng cách với mỗi đỉnh trong đồ thị, lưu danh sách các đỉnh kề với nó.



2	3	
1	3	
1	4	
2	3	5
4	6	
5		

Human view vs Computer view



VS

0	1	1	0	0	0
0	0	0	1	0	0
0	0	0	1	0	0
0	0	0	0	1	0
0	0	0	0	0	1
0	0	0	0	0	0

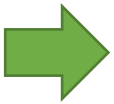
2	3	
1	3	
1	4	
2	3	5
4	6	
5		

(1, 2)	(1, 3)	(2, 4)	(3, 4)	(4, 5)	(5, 6)

BT1 – TẠO DANH SÁCH CẠNH

- Cho đồ thị vô hướng được biểu diễn bằng ma trận kề (adjacency matrix). Đọc đồ thị trên và biểu thị bằng danh sách cạnh (edge list).

0	1	1	0	0	0
1	0	0	1	0	0
1	0	0	1	0	0
0	1	1	0	1	0
0	0	0	1	0	1
0	0	0	0	1	0



(1, 2)	(1, 3)	(2, 4)	(3, 4)	(4, 5)	(5, 6)

BT1 – Gợi ý – Xử lí chính

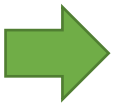
1. Khai báo struct/class Edge, có 2 thuộc tính u, v dùng để biểu diễn cạnh u-v.
2. Adjacency matrix là 1 mảng 2 chiều → sử dụng kĩ thuật đọc mảng 2 chiều ở L06 để đọc vào mảng a.

0	1	1	0	0	0
1	0	0	1	0	0
1	0	0	1	0	0
0	1	1	0	1	0
0	0	0	1	0	1
0	0	0	0	1	0

BT1 – Gợi ý

3. Chuyển từ adjacency matrix thành edge list.
   a. Tạo mảng edge_list rỗng.
   b. Dùng 2 vòng lặp, duyệt qua từng phần tử trong ma trận adj_matrix.
      - Nếu a[u][v] ==1 (có cạnh nối giữa đỉnh u và đỉnh v), thêm cạnh mới Edge(u, v) vào edge_list.

0	1	1	0	0	0
1	0	0	1	0	0
1	0	0	1	0	0
0	1	1	0	1	0
0	0	0	1	0	1
0	0	0	0	1	0



(1, 2)	(1, 3)	(2, 4)	(3, 4)	(4, 5)	(5, 6)

BT1 – Gợi ý

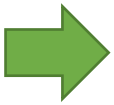
4. In kết quả,
   - Duyệt qua từng phần tử trong edge_list, in edge_list[i].u và edge_list[i].v.

(1, 2)	(1, 3)	(2, 4)	(3, 4)	(4, 5)	(5, 6)

BT2 – KIỂM TRA ĐỒ THỊ VÔ HƯỚNG

- Cho đồ thị được biểu diễn bằng ma trận kề (adjacency matrix). Kiểm tra đồ thị đó có phải là đồ thị vô hướng hay không.

0	1	1	0	0	0
1	0	0	1	0	0
1	0	0	1	0	0
0	1	1	0	1	0
0	0	0	1	0	1
0	0	0	0	1	0



flag = True (undirected)
flag = False (directed)

BT2 – Gợi ý

- Đối với đồ thị vô hướng, ma trận kề là ma trận đối xứng, nghĩa là a[u][v] = a[v][u].
- Do đó, hãy dùng **kĩ thuật đặt cờ hiệu**, tìm ra 1 vị trí a[u][v] ≠ a[v][u].

BT2 – Gợi ý – Xử lí chính

1. Sử dụng kĩ thuật đọc mảng 2 chiều ở L06, để đọc vào mảng 2 chiều a.
2. Khởi tạo cờ hiệu: flag = True (tin tưởng rằng đây là đồ thị vô hướng undirected).
3. Sử dụng 2 vòng lặp, duyệt qua các phần tử a[i][j] trong mảng.
   - Nếu a[u][v] != a[v][u], flag = False.
4. In kết quả.

BT3 – TÌM BẬC CỦA ĐỈNH

- Cho đồ thị vô hướng được biểu diễn bằng ma trận kề (adjacency matrix) và 1 đỉnh trong đồ thị. Tìm bậc của đỉnh đó.

0	1	1	0	0	0
1	0	0	1	0	0
1	0	0	1	0	0
0	1	1	0	1	0
0	0	0	1	0	1
0	0	0	0	1	0

x = 3

count = 2

BT3 – Gợi ý

- Trong đồ thị vô hướng, degree(x) = Tổng các số trên dòng x.
- Do đó, hãy sử dụng **kĩ thuật đếm**, để đếm xem dòng x có bao nhiêu giá trị khác 0.

BT3 – Gợi ý – Xử lí chính

1. Đọc vào n (số đỉnh) và x (đỉnh cần tính bậc).
2. Sử dụng kĩ thuật đọc mảng 2 chiều ở L06, để đọc vào mảng 2 chiều a.
3. Khởi tạo đếm: count = 0 (tin tưởng rằng đây là đồ thị vô hướng undirected).
4. Sử dụng 1 vòng lặp, duyệt qua các phần tử a[x][j] trong mảng.
   - Nếu a[x][j] != 0, count +=1.
5. In kết quả.

BT4 – TỔNG TRỌNG SỐ CẠNH NHỎ NHẤT

- Cho đồ thị được biểu diễn bằng danh sách cạnh (edge list), có các cạnh có thể có nhiều hơn một trọng số. Hãy tính tổng trọng số các cạnh có trọng số nhỏ nhất.

(1, 2, 1)	(3, 1, 2)	(1, 4, 6)	**(3, 1, 1)**	(2, 4, 5)

1 * 2 = 2

BT4 – Gợi ý (C1)

1. Khai báo struct / class Edge có 3 thuộc tính: đỉnh u, đỉnh v và trọng số w.
2. Tạo edge_list rỗng.
3. Đọc từng dòng 3 giá trị u, v, w, tạo đối tượng Edge(u, v, w) và thêm vào list edge_list.
4. Sử dụng kĩ thuật **đặt lính canh**, tìm giá trị trọng số nhỏ nhất minw.
5. Duyệt lại list edge_list, tính tổng trọng số các cạnh là minw.

BT5 – TÍCH CÁC CẠNH KHUYÊN

- Cho đồ thị vô hướng được biểu diễn bằng danh sách cạnh (edge list). Trong đồ thị có các cạnh khuyên hãy đếm số lượng cạnh khuyên và tính tích trọng số cạnh khuyên đó.

(1, 2, 1)	(3, 1, 2)	**(1, 1, 3)**	(3, 1, 4)	**(4, 4, 5)**

→

2 15

BT5 – Gợi ý

- Cạnh khuyên là cạnh có dạng (u, u) tức là cạnh có đỉnh đầu và đỉnh cuối là cùng 1 đỉnh.
    - Sử dụng kĩ thuật đếm.

Hỏi đáp