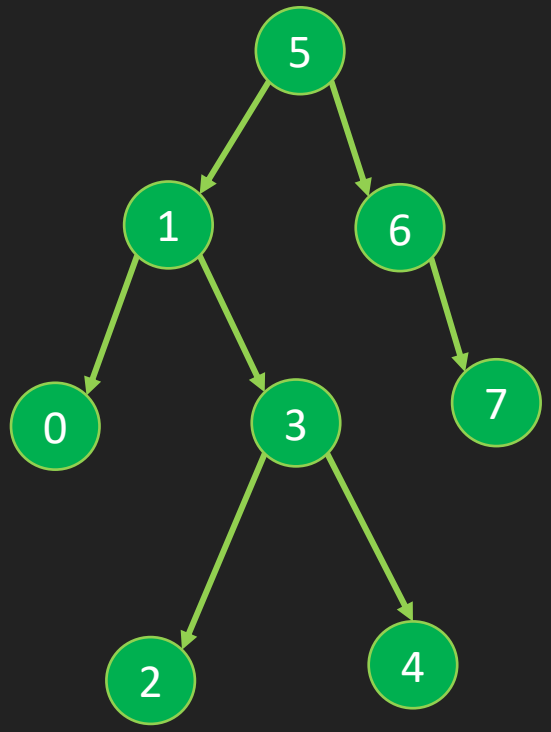
Data Structure & Algorithm

Data Structure

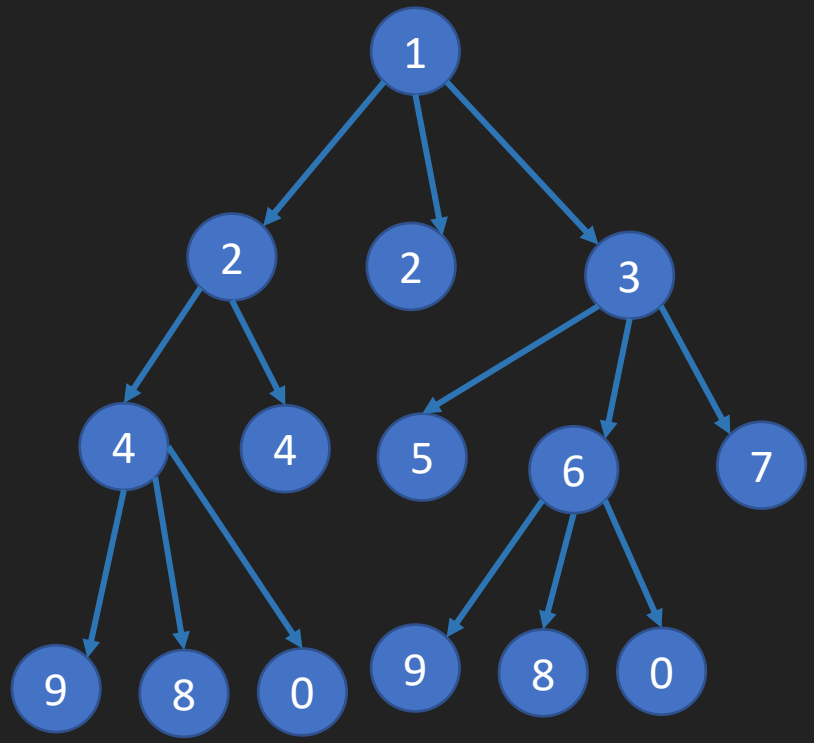
Đồ thị- Graph

1. Định nghĩa

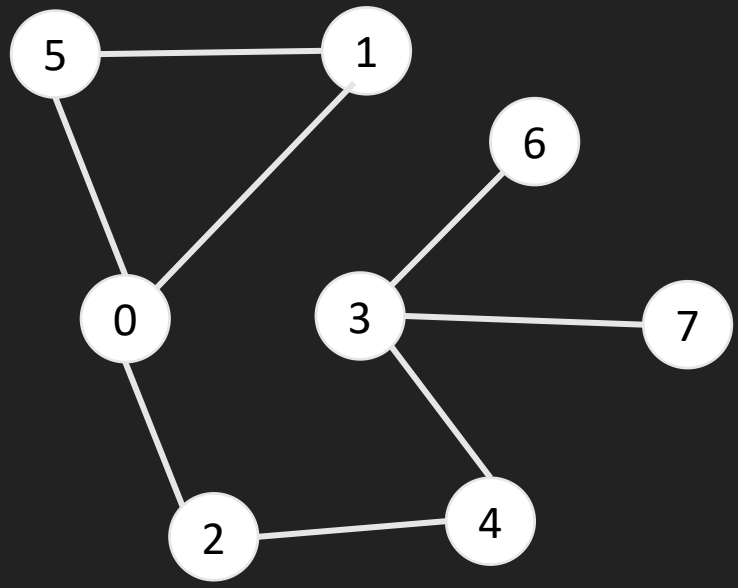
Linked List



Binary Tree

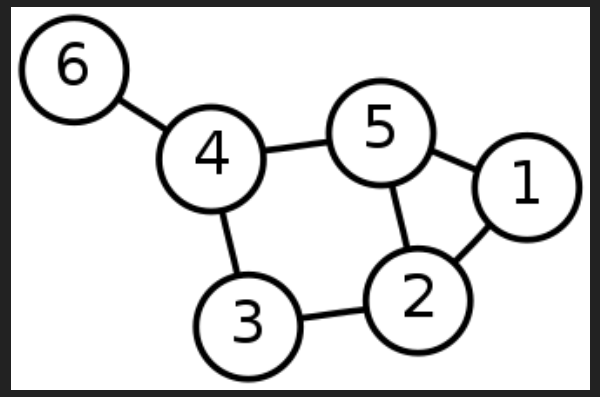


N-ary Tree



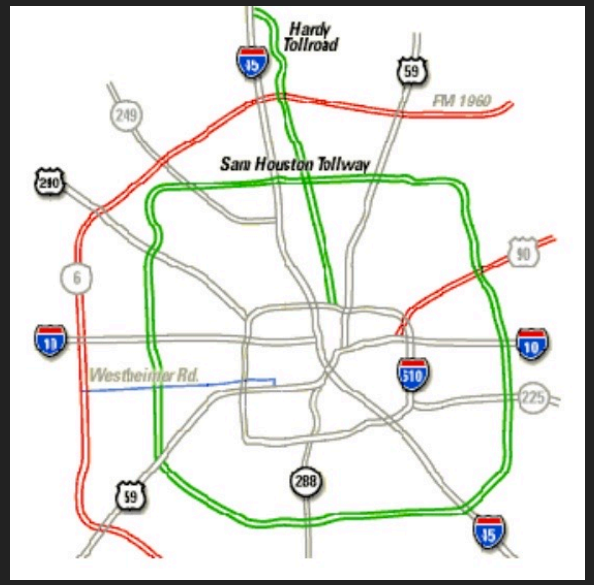
Graph

1. Định nghĩa

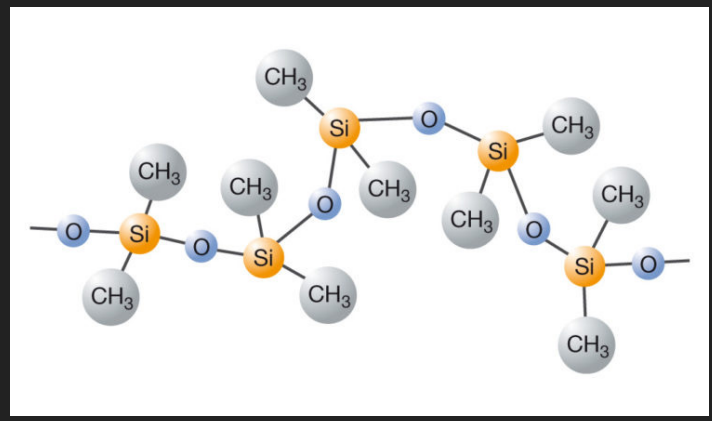


Graph

Computer Network



Map



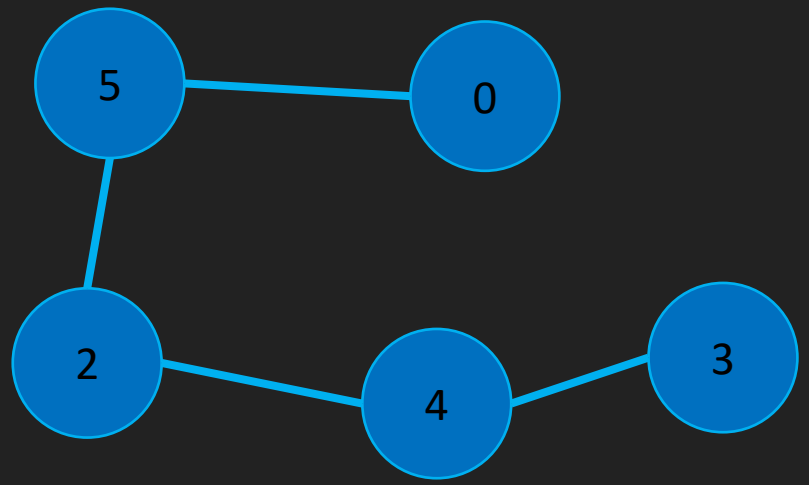
Molecular Strcuture

Data Structure & Algorithm

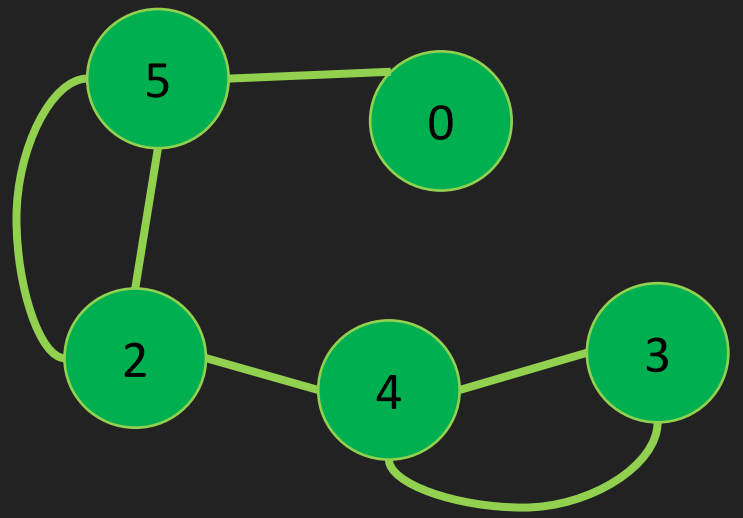
- The Brown Box -

2. Phân loại

❖ Theo số cạnh giữa 2 đỉnh



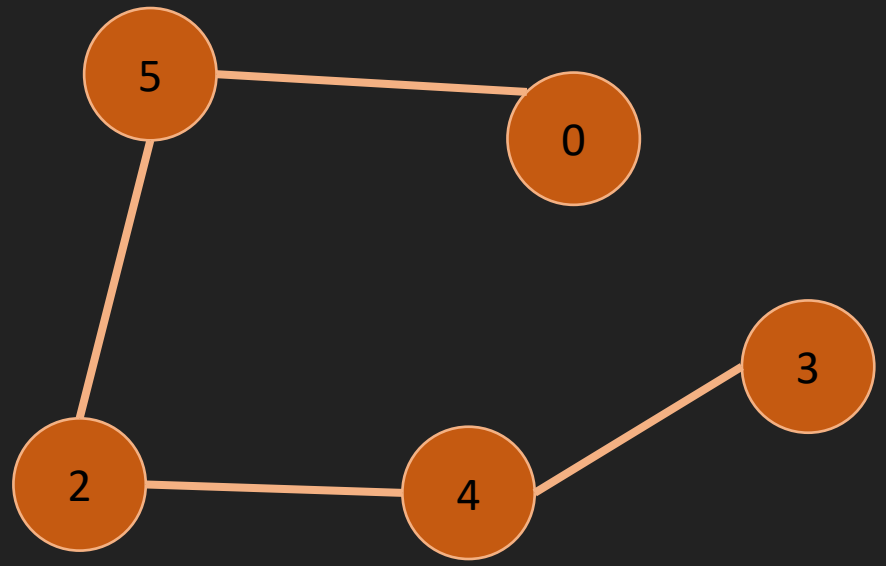
Đơn Đồ Thị



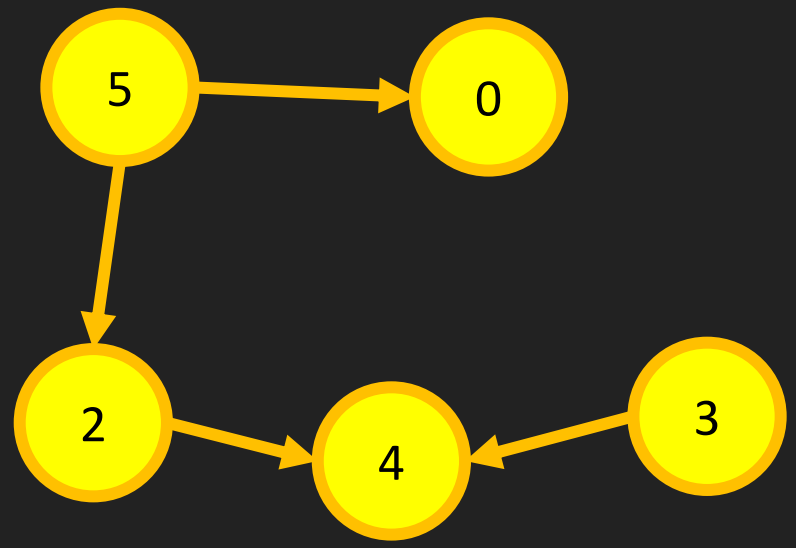
Đa Đồ Thị

2. Phân loại

❖ Tính có hướng của cạnh



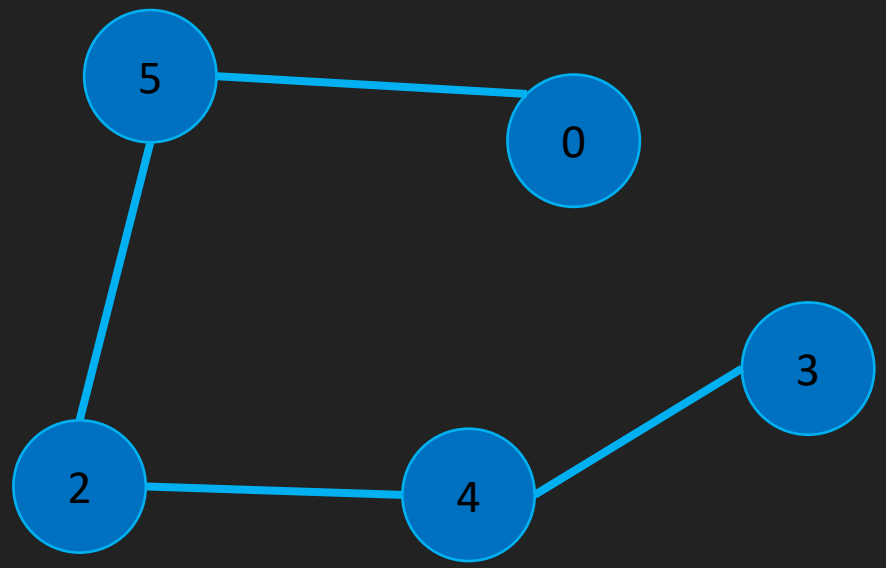
Đồ thị vô hướng



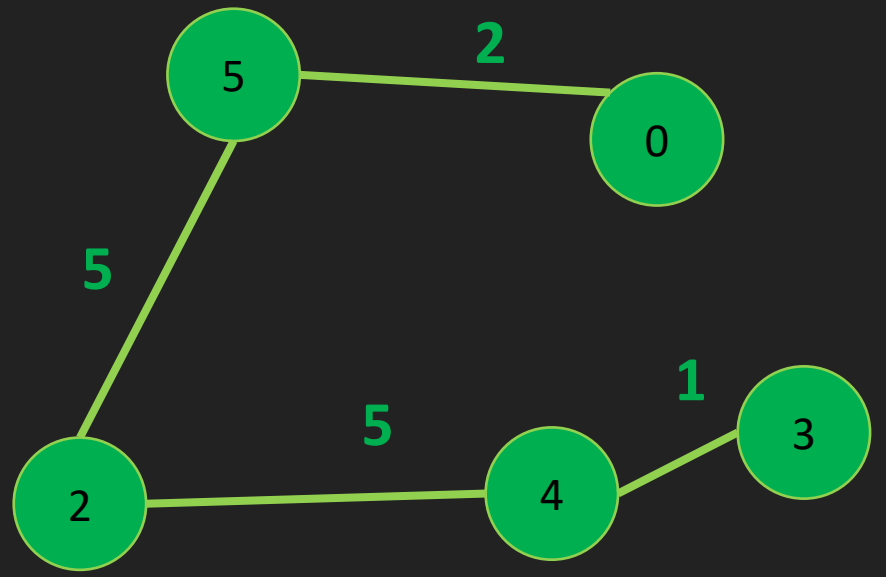
Đồ thị có hướng

2. Phân loại

❖ Trọng số



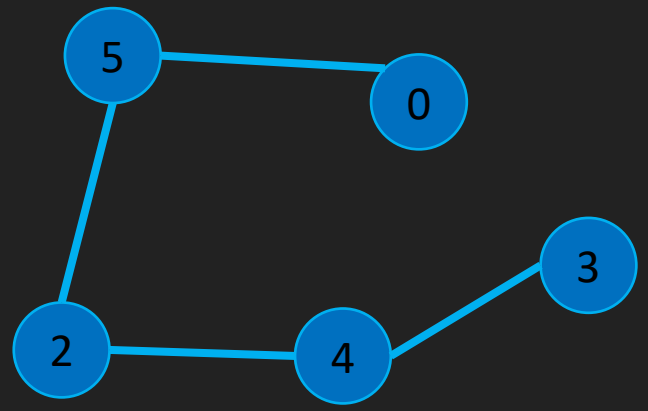
Đồ thị KHÔNG có trọng số



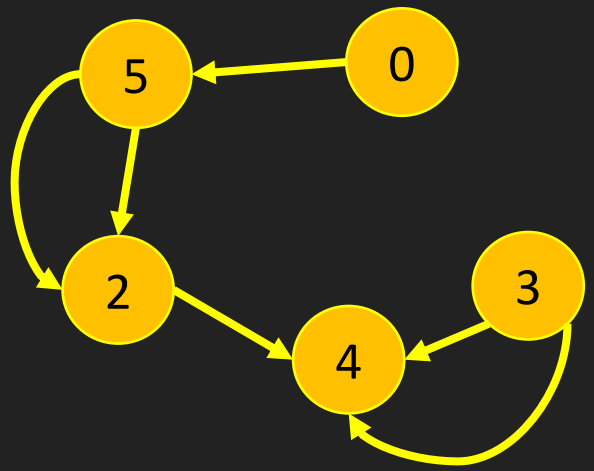
Đồ thị CÓ trọng số

2. Phân loại

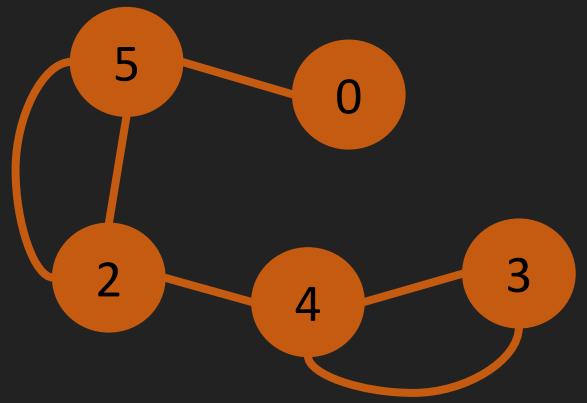
❖ Quiz:



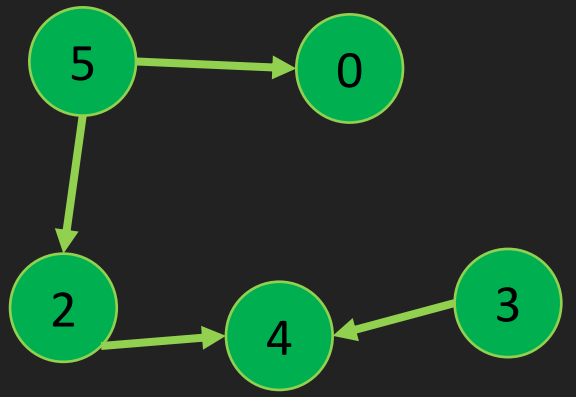
ĐƠN đồ thị VÔ HƯỚNG



ĐA đồ thị CÓ HƯỚNG



ĐA đồ thị VÔ HƯỚNG



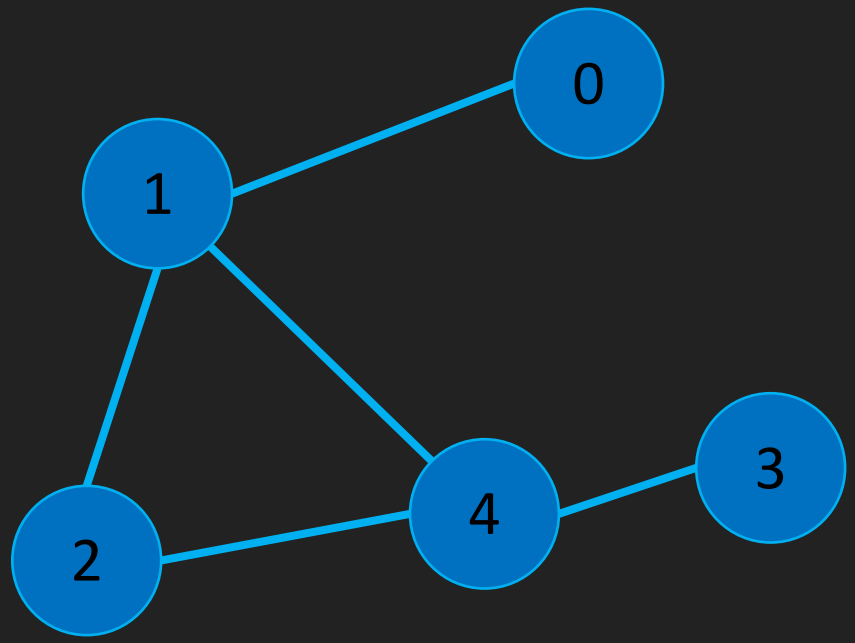
ĐƠN đồ thị CÓ HƯỚNG

3. Biểu diễn đồ thị

- Ma trận kề
- Danh sách cạnh
- Danh sách kề

3. Biểu diễn đồ thị

➢ Ma trận kề



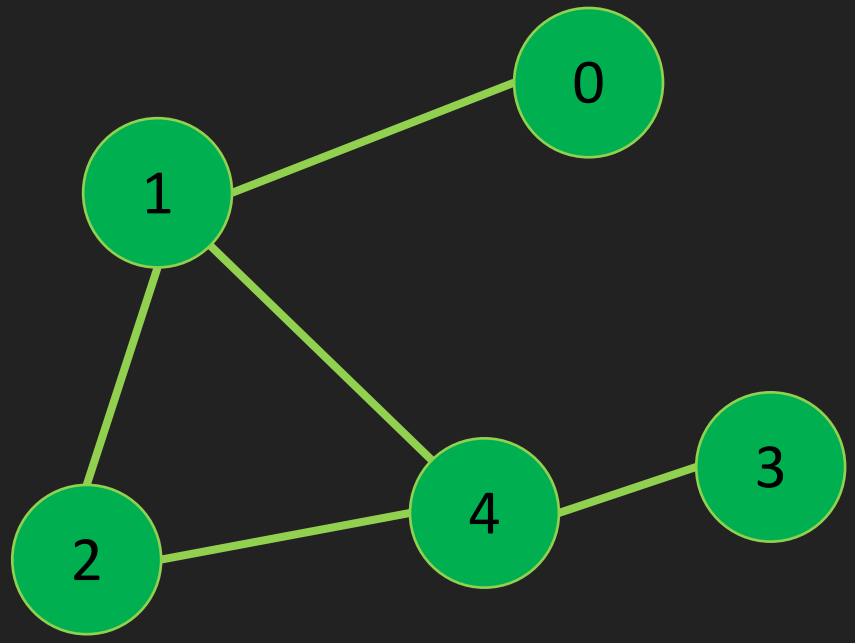
	0	1	2	3	4
0	_	1	_	_	_
1	1	_	1	_	1
2	_	1	_	_	1
3	_	_	_	_	1
4	_	1	1	1	_

❖ Áp dụng:

- ✓ Đơn đồ thị
- ✓ Vô hướng / Có hướng
- ✓ Có trọng số

3. Biểu diễn đồ thị

➢ Danh sách đỉnh kề



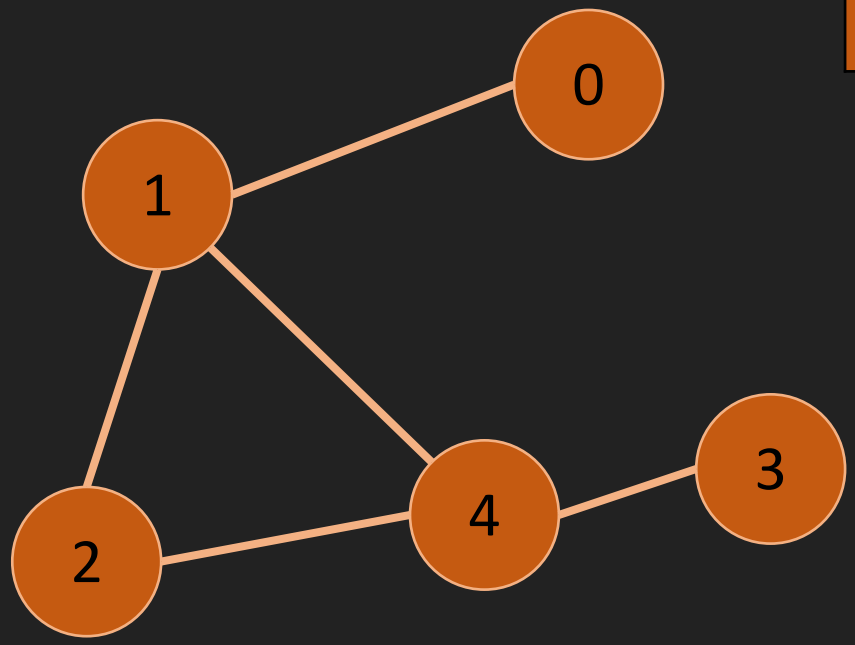
0	1		
1	0	2	4
2	1	4	
3	4		
4	1	2	3

❖ Áp dụng:

- ✓ Đơn đồ thị
- ✓ Vô hướng / Có hướng
- ✓ **Không có** trọng số

3. Biểu diễn đồ thị

➢ Danh sách cạnh



{ 0; 1 }	{ 1; 2 }	{ 1; 4 }	{ 2; 4 }	{ 3; 4 }

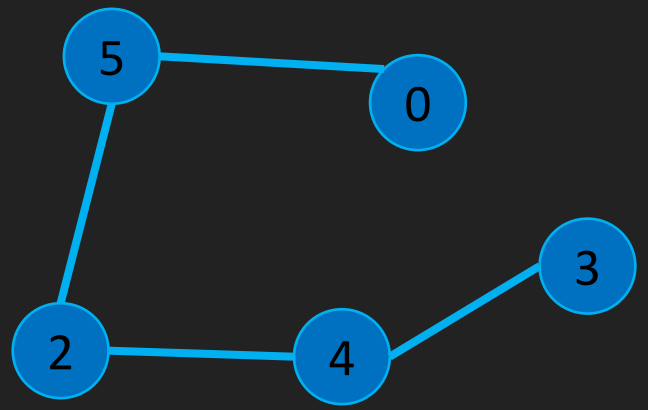
❖ Áp dụng:

- ✓ Đơn đồ thị
- ✓ Vô hướng / Có hướng
- ✓ **Không có** trọng số

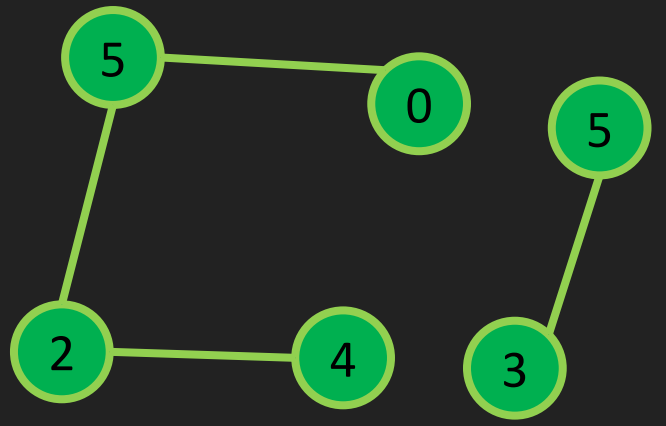
3. Biểu diễn đồ thị

	Ma trận kề **	Danh sách đỉnh kề *	Danh sách cạnh
Ưu điểm	- Đơn giản. - Trực quan. - Dễ cài đặt. - Kiểm tra cạnh u,v: a[u,v] : O(1).	- Dễ dàng duyệt qua các đỉnh kề của 1 đỉnh. - Tối ưu bộ nhớ hơn Ma trận kề.	- Tiết kiệm không gian bộ nhớ. - Dễ dàng hơn trong trường hợp muốn duyệt cạnh (Kruscal).
Nhược điểm	- Trong trường hợp đồ thị thưa (ít cạnh) : Tốn bộ nhớ O(n^2). - Trong trường hợp muốn xét các đỉnh kề: O(n).	- Việc xét qua hệ giữa 2 đỉnh u và v cần phải duyệt hết danh sách đỉnh kề của u hoặc của v. - Việc cài đặt có phần phức tạp hơn.	- Khi muốn duyệt tất cả đỉnh kề với u, phải duyệt hết các cạnh.

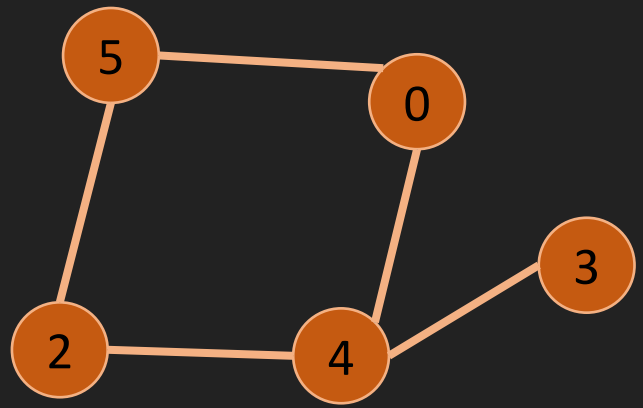
4. Một số tính chất của đồ thị



➢ Đồ thị liên thông



➢ Đồ thị KHÔNG liên thông



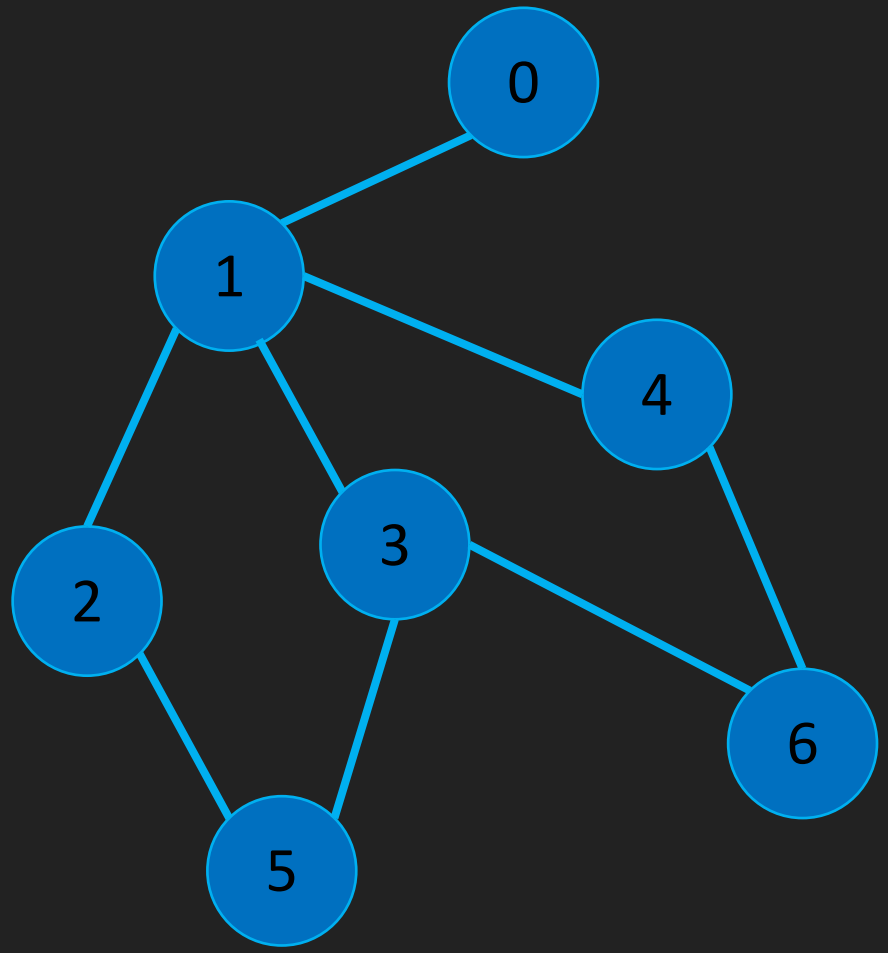
➢ Đồ thị có tồn tại chu trình

5. Tìm kiếm trên đồ thị

- Depth First Search (Stack | Recursion)
- Breadth Frist Search (Queue)

5. Tìm kiếm trên đồ thị

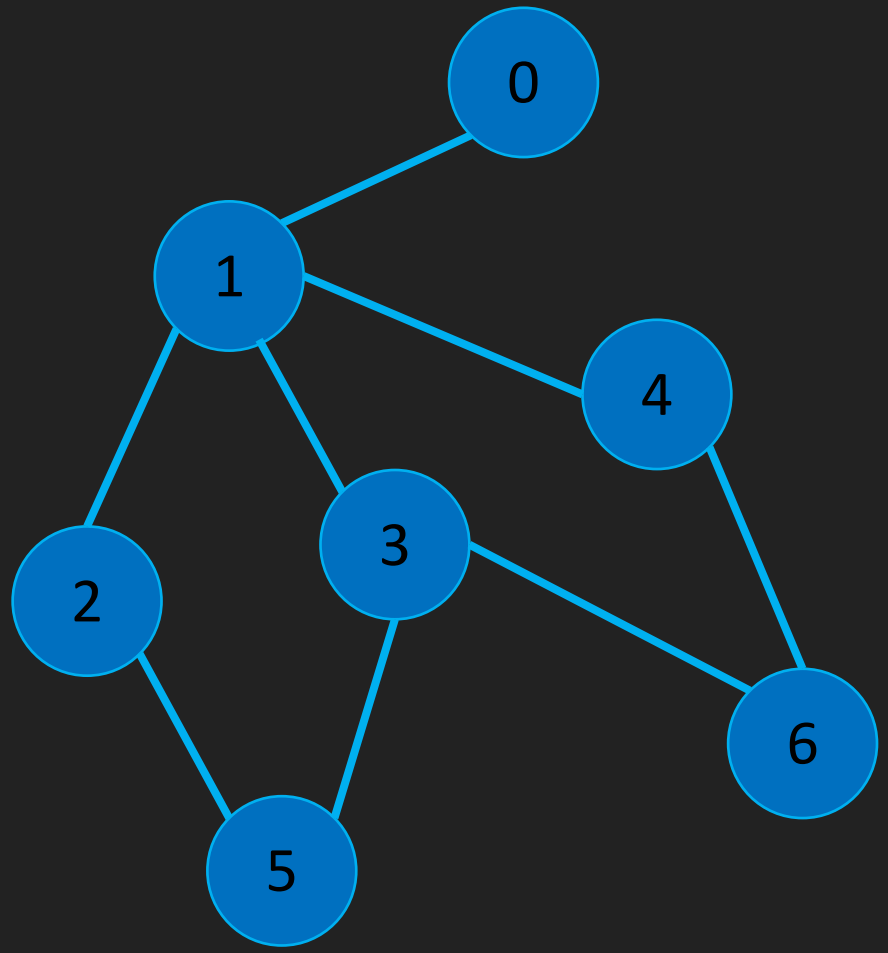
➢ Depth First Search (Stack | Recursion)



	0	1	2	3	4	5	6
0	-	1	-	-	-	-	-
1	1	-	1	1	1	-	-
2	-	1	-	-	-	1	
3	-	1	-	-	-	1	1
4	-	1	-	-	-	-	1
5	-	-	1	1	-	-	-
6	-	-	-	1	1	-	-

5. Tìm kiếm trên đồ thị

➢ Depth First Search (Stack | Recursion)



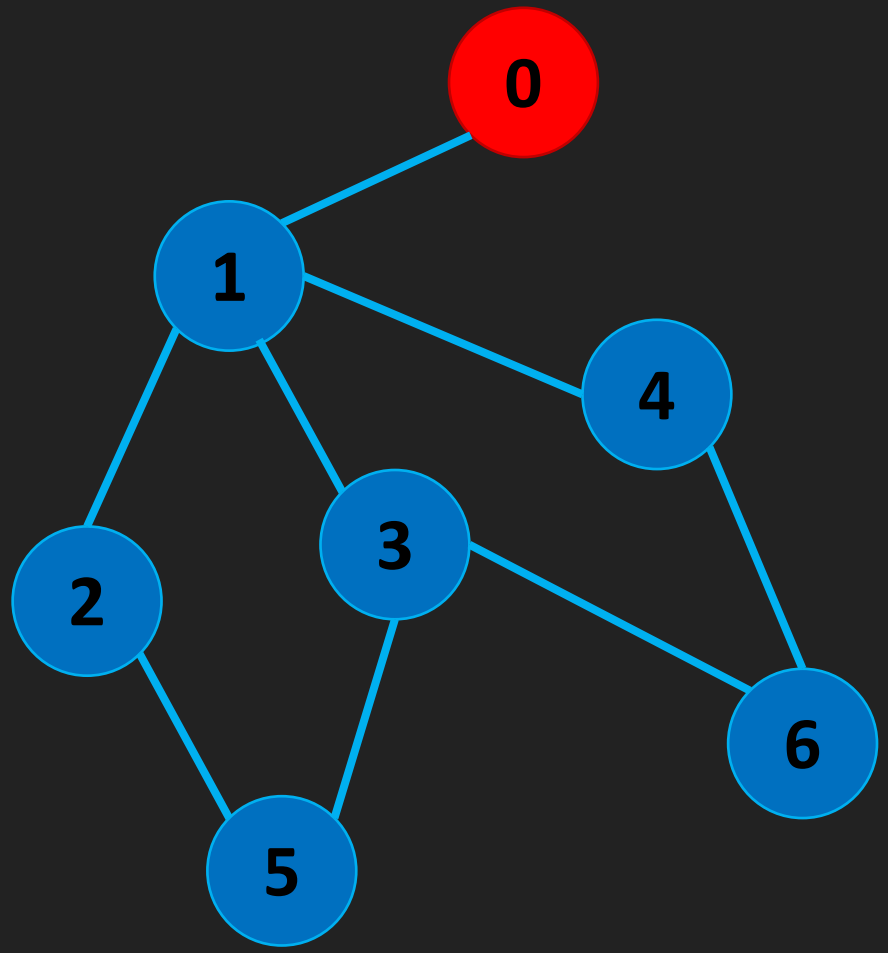
~ Thuật toán ~

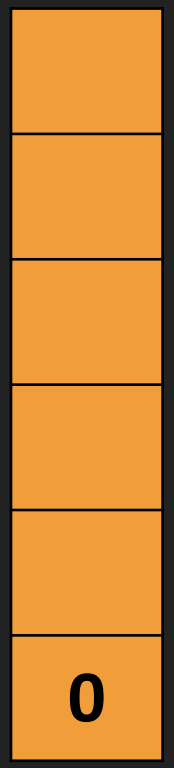
```
- Add đỉnh đầu vào stack
- Đánh dấu đã duyệt đỉnh đầu
while(!stack.isEmpty())
{
    u = stack.pop();
    process(u);
    - Add tất cả các đỉnh kề v với u
    mà chưa được duyệt vào stack.
    - Đánh dấu đã duyệt v.
}
```

5. Tìm kiếm trên đồ thị

➢ Depth First Search (Stack | Recursion)

Step 0





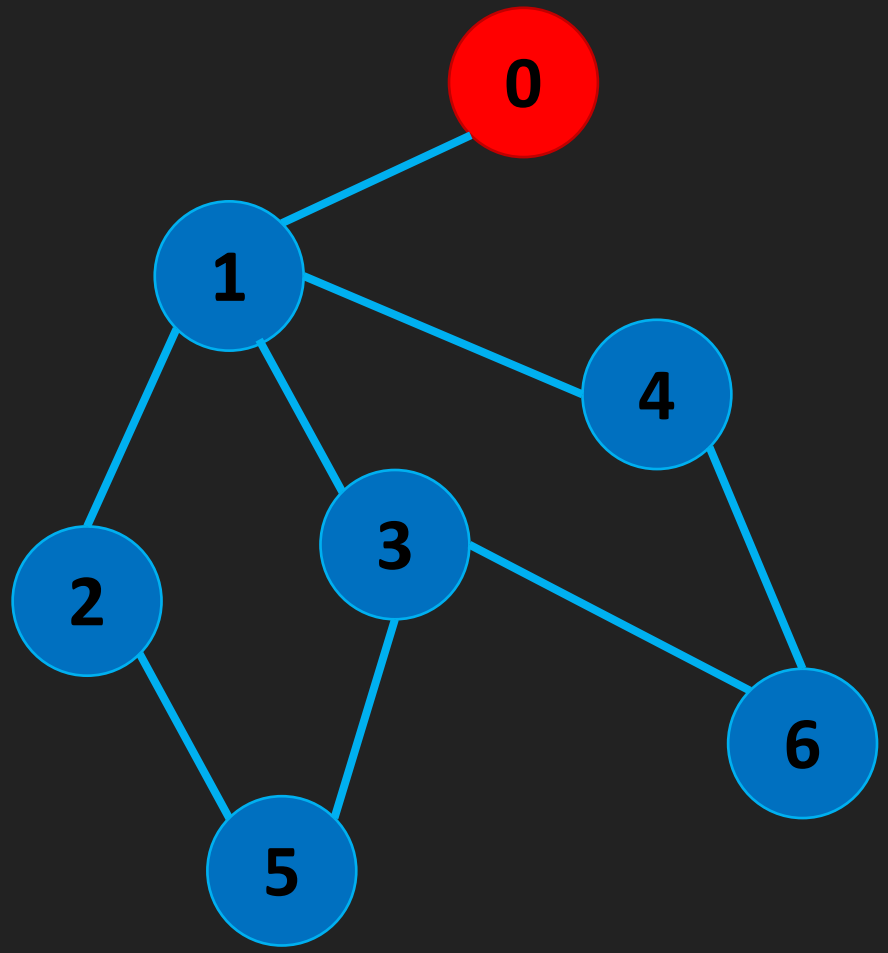
stack

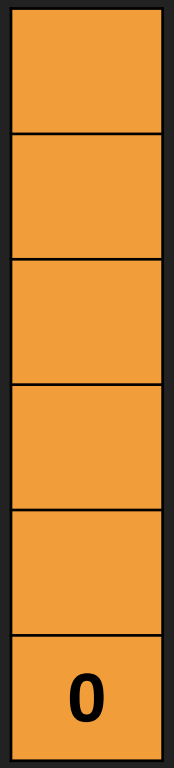
Thứ tự duyệt

5. Tìm kiếm trên đồ thị

➢ Depth First Search (Stack | Recursion)

Step 1





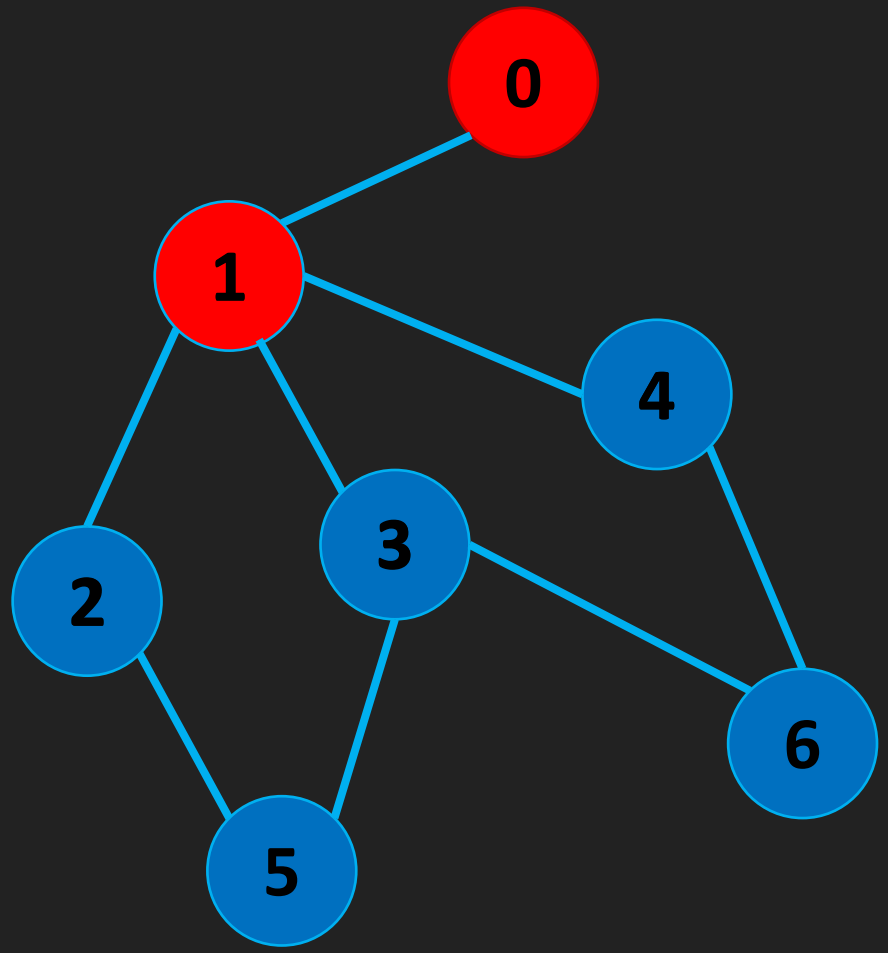
stack

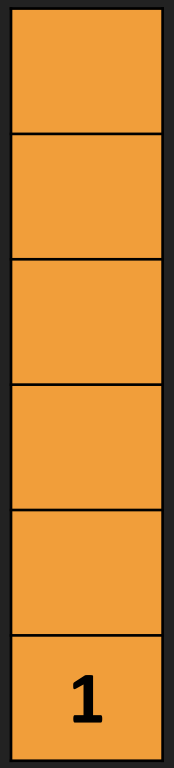
Thứ tự duyệt

5. Tìm kiếm trên đồ thị

➢ Depth First Search (Stack | Recursion)

Step 2





stack

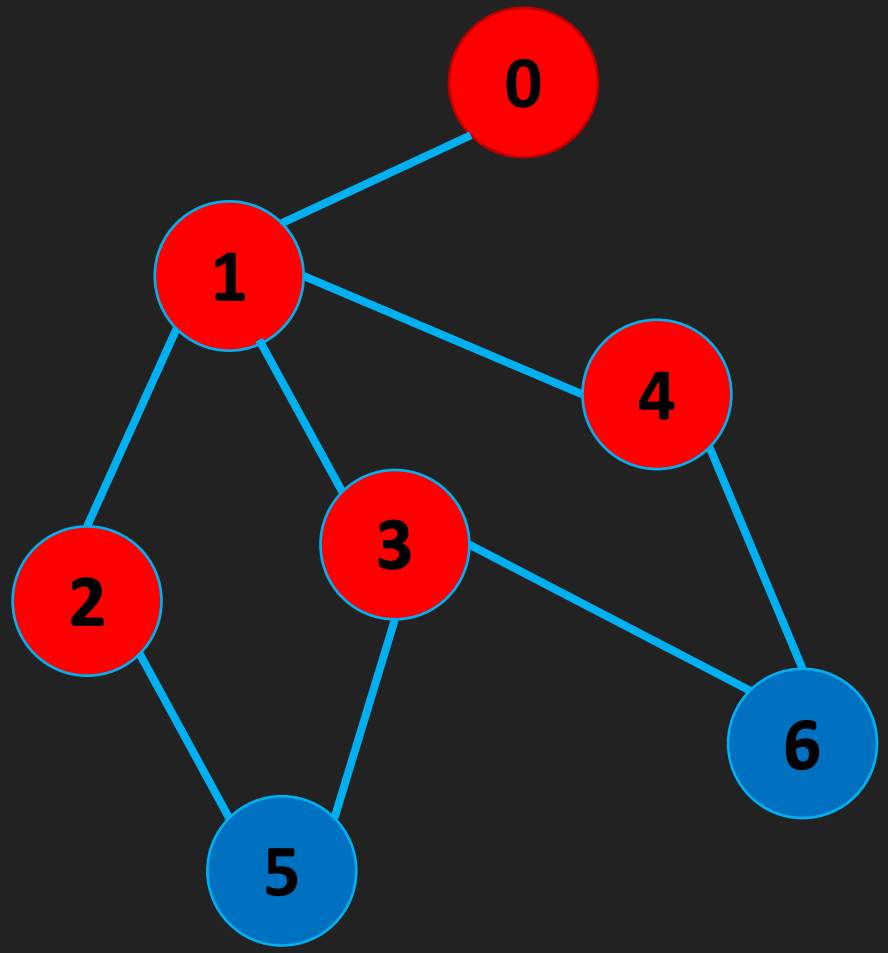
Thứ tự duyệt

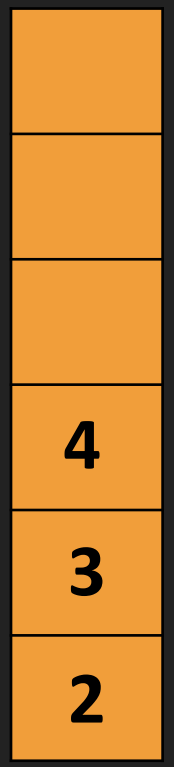
0						

5. Tìm kiếm trên đồ thị

➢ Depth First Search (Stack | Recursion)

Step 3





stack

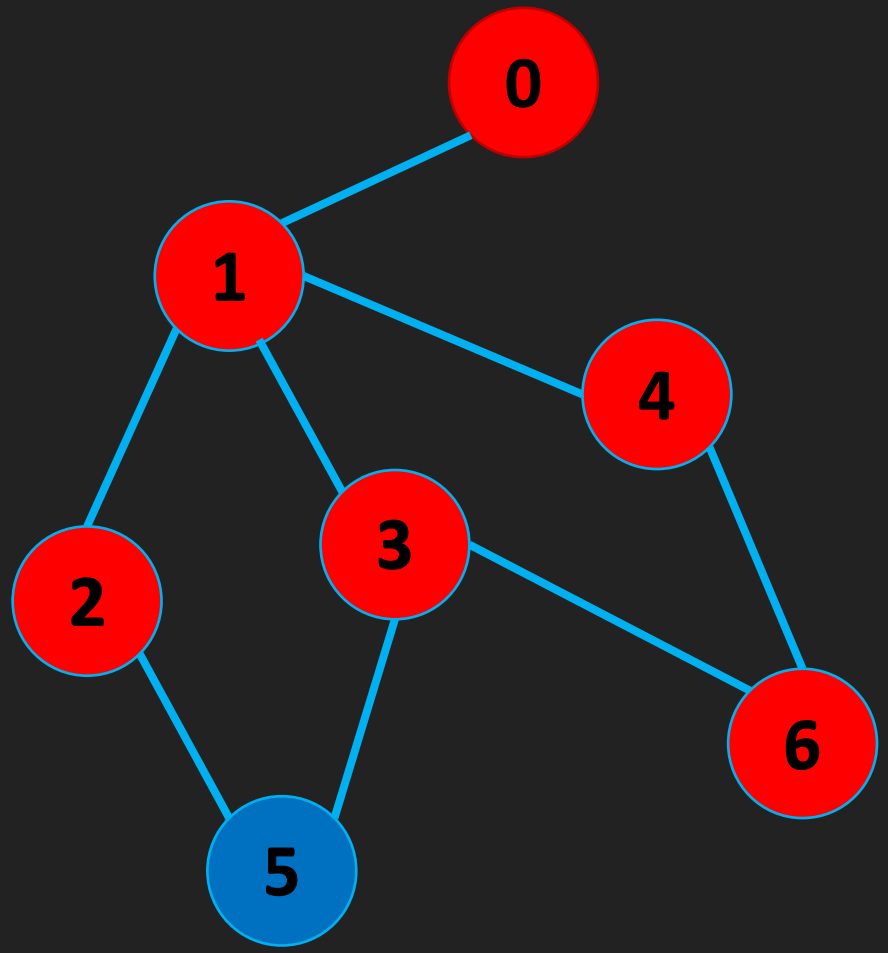
Thứ tự duyệt

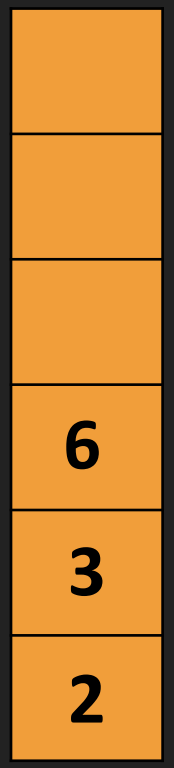
0	1					

5. Tìm kiếm trên đồ thị

➢ Depth First Search (Stack | Recursion)

Step 4





stack

Thứ tự duyệt

0	1	4				

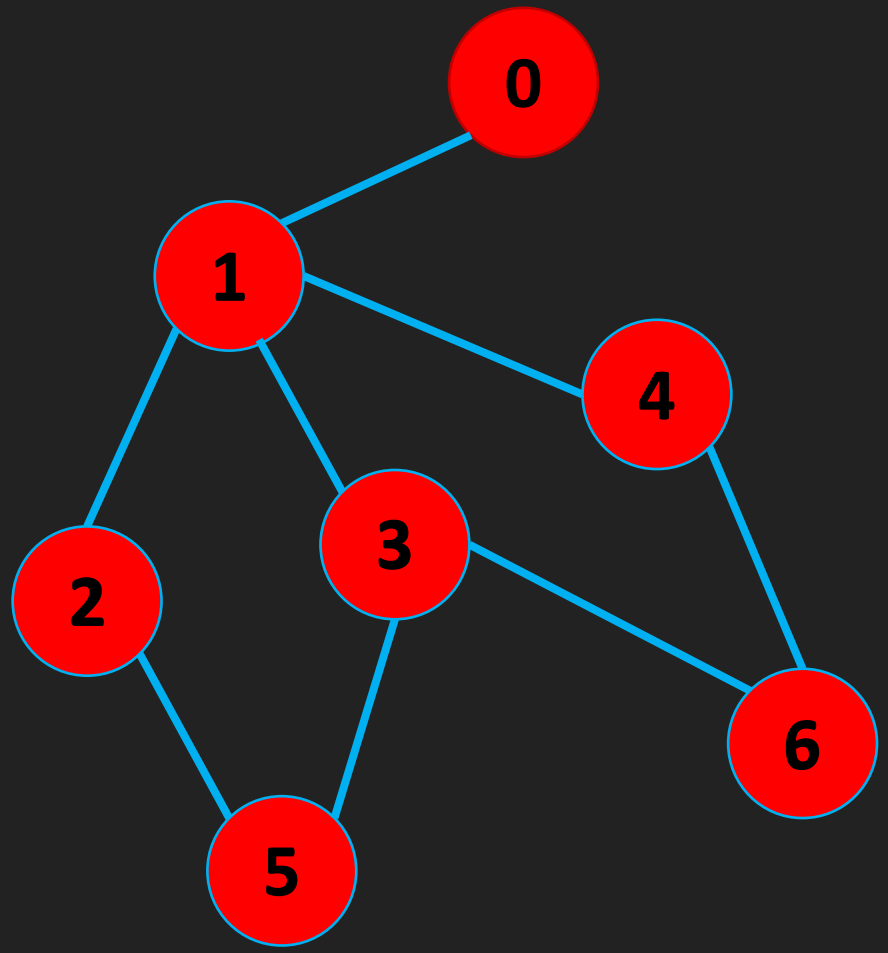
Data Structure & Algorithm

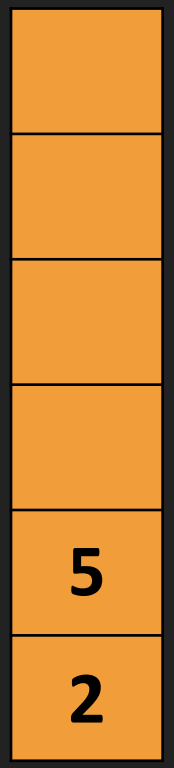
- The Brown Box -

5. Tìm kiếm trên đồ thị

➢ Depth First Search (Stack | Recursion)

Step 5





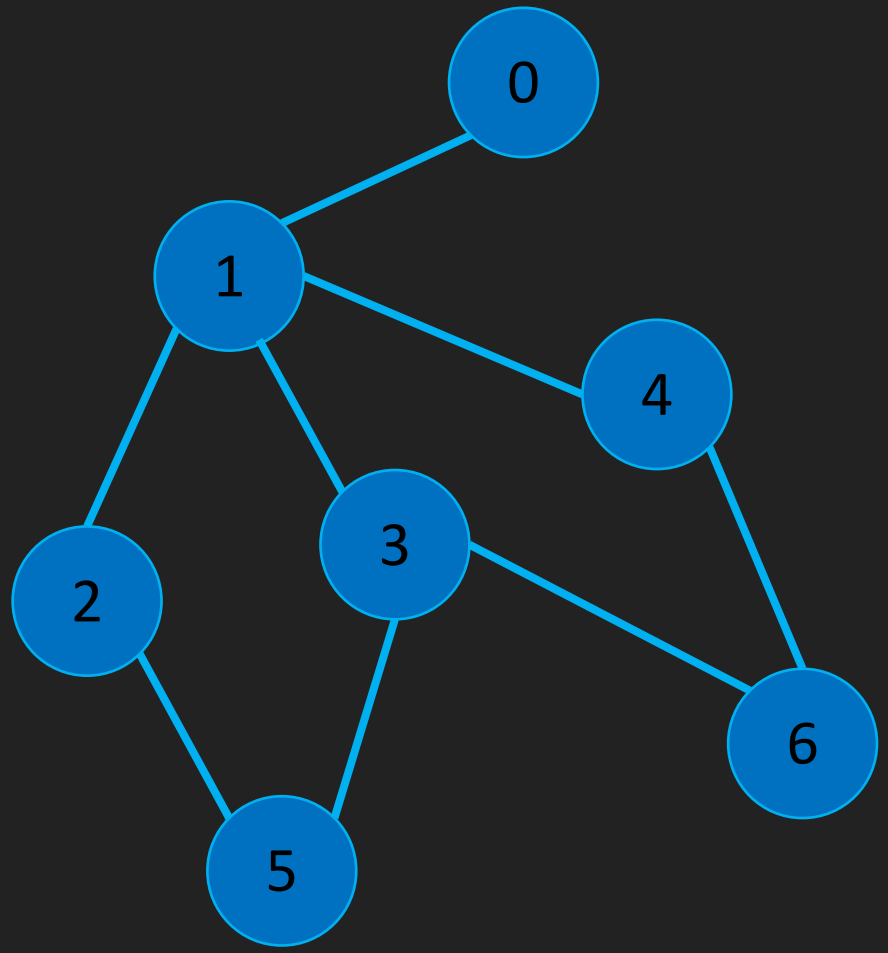
stack

Thứ tự duyệt

0	1	4	6	3		

5. Tìm kiếm trên đồ thị

➢ Depth First Search (Stack | Recursion)



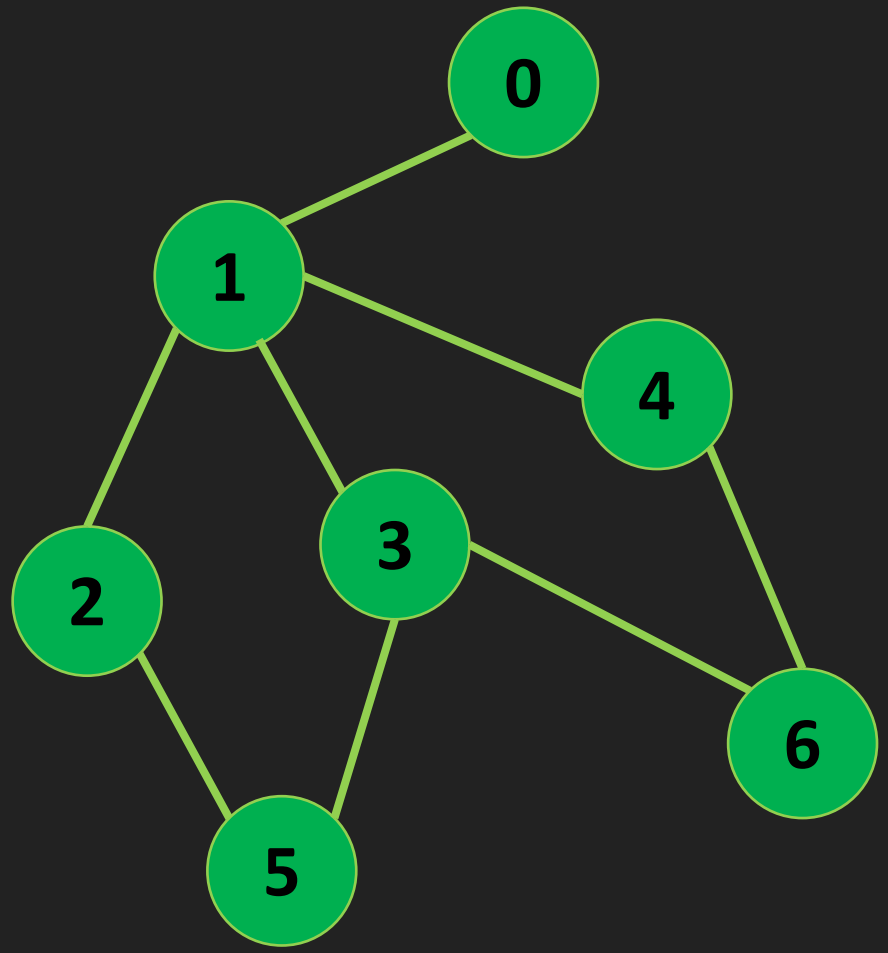
Thứ tự duyệt

0	1	4	6	3	5	2

- ✓ Implement using Stack
- ✓ Implement using Recursion

5. Tìm kiếm trên đồ thị

➢ Breadth Frist Search (Queue)



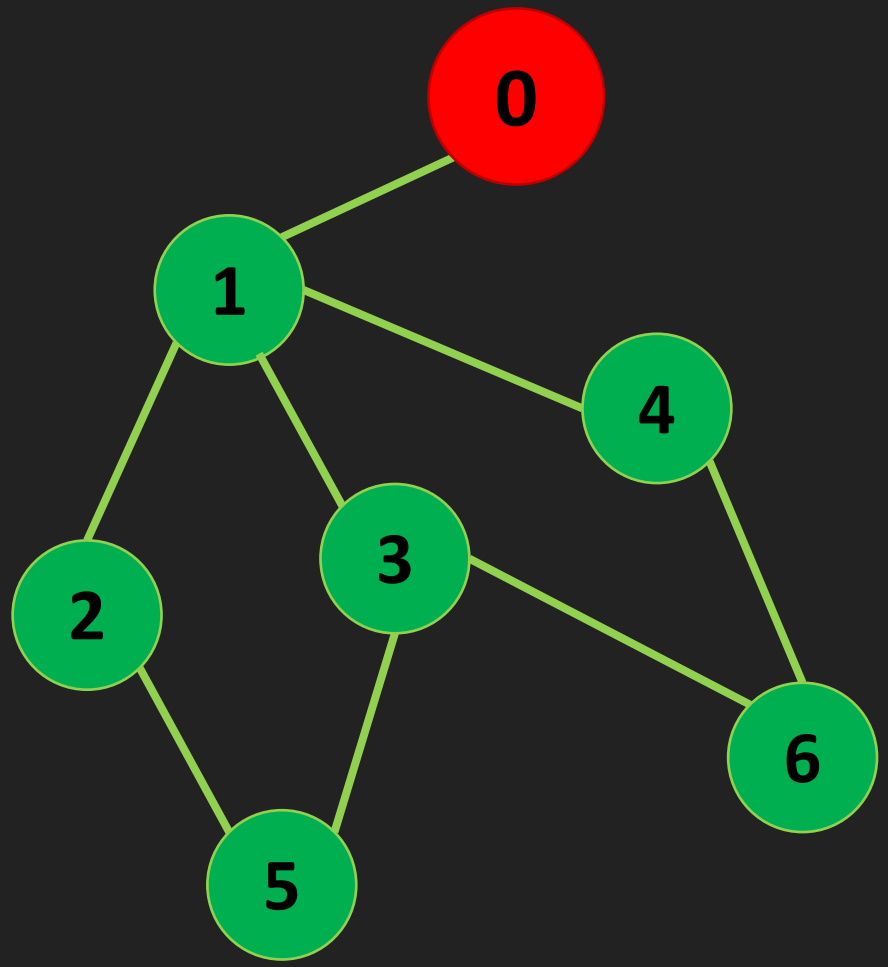
~ Thuật toán ~

```
- Add đỉnh đầu vào queue
- Đánh dấu đã duyệt đỉnh đầu
while(!queue.isEmpty())
{
    u = queue.pop();
    process(u);
    - Add tất cả các đỉnh kề v với u
    mà chưa được duyệt vào queue.
    - Đánh dấu đã duyệt v.
}
```

5. Tìm kiếm trên đồ thị

➢ Breadth Frist Search (Queue)

Step 0



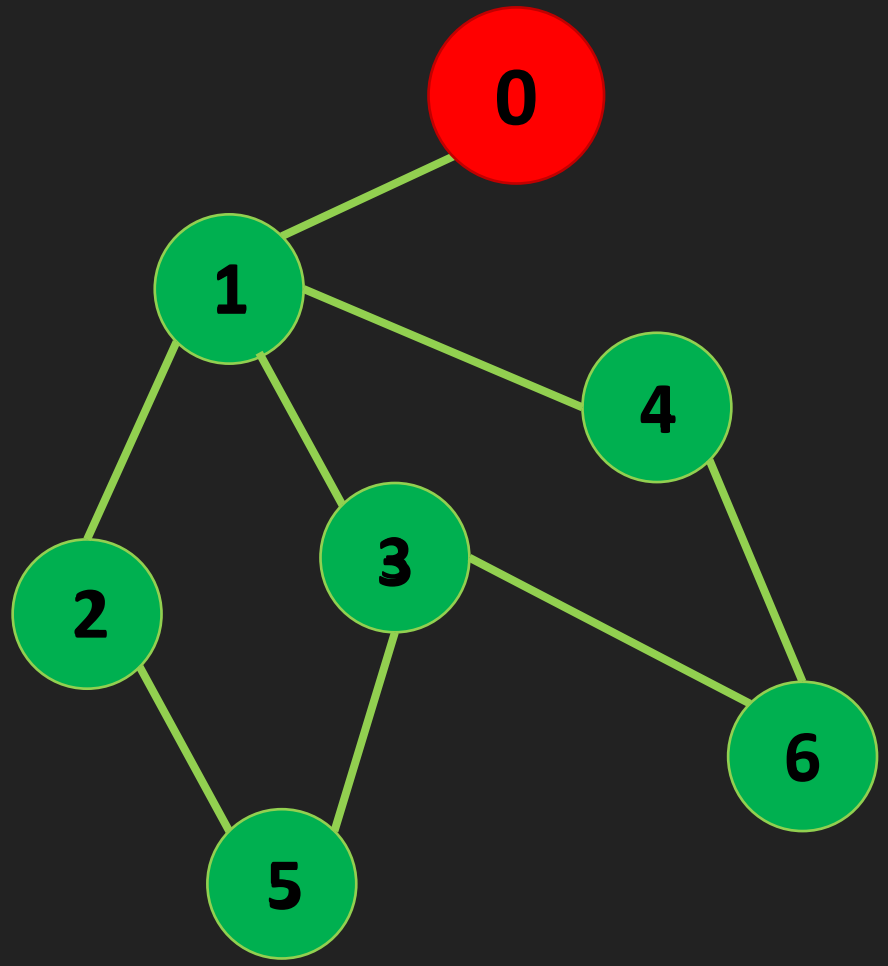
Thứ tự duyệt

queue

5. Tìm kiếm trên đồ thị

➢ Breadth Frist Search (Queue)

Step 1



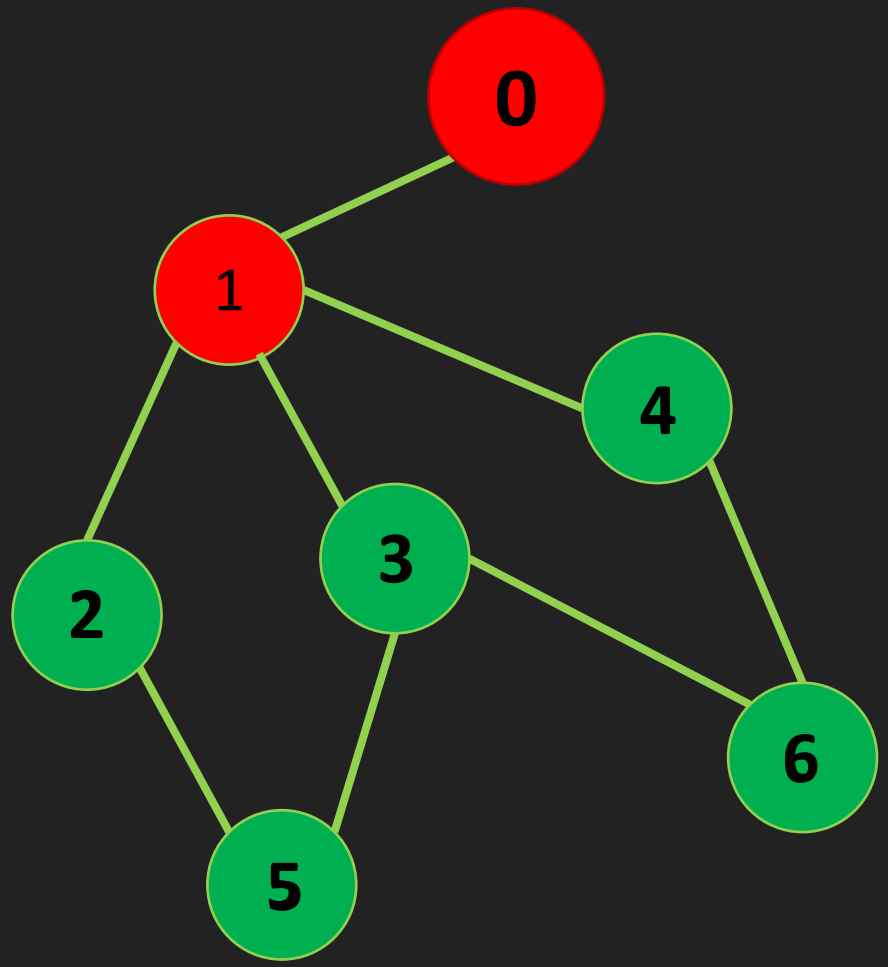
Thứ tự duyệt

queue

5. Tìm kiếm trên đồ thị

➢ Breadth Frist Search (Queue)

Step 2



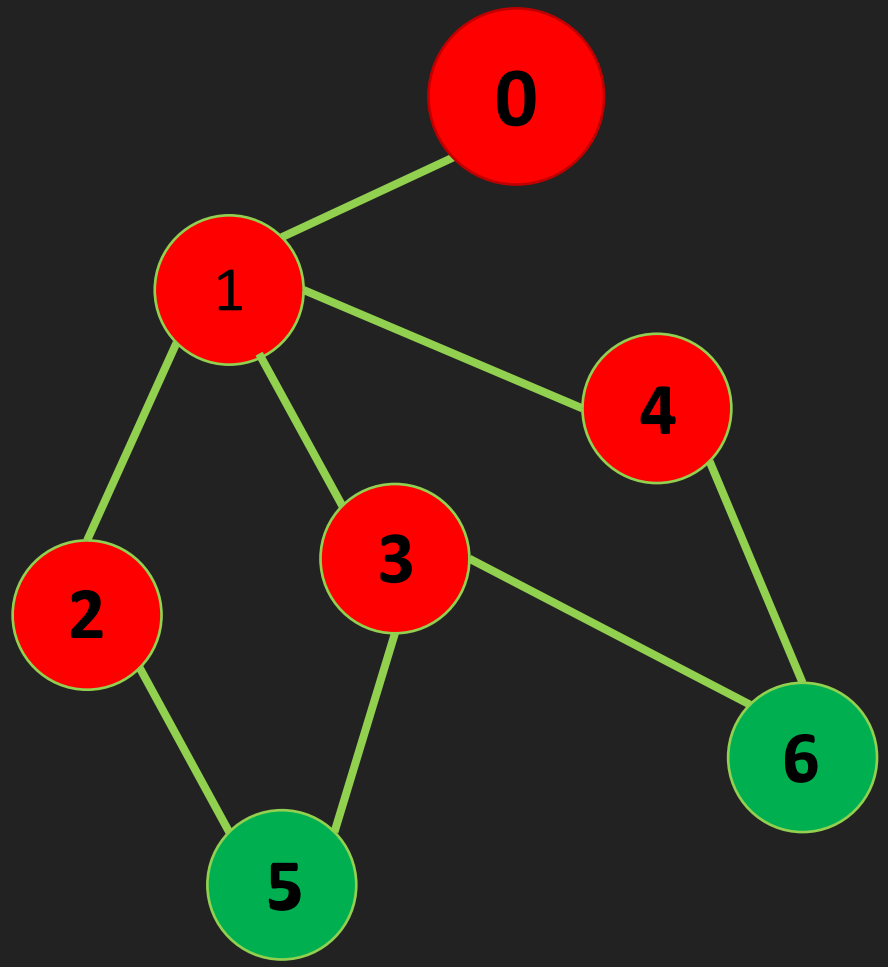
Thứ tự duyệt

queue

5. Tìm kiếm trên đồ thị

➢ Breadth Frist Search (Queue)

Step 3



Thứ tự duyệt

0	1					

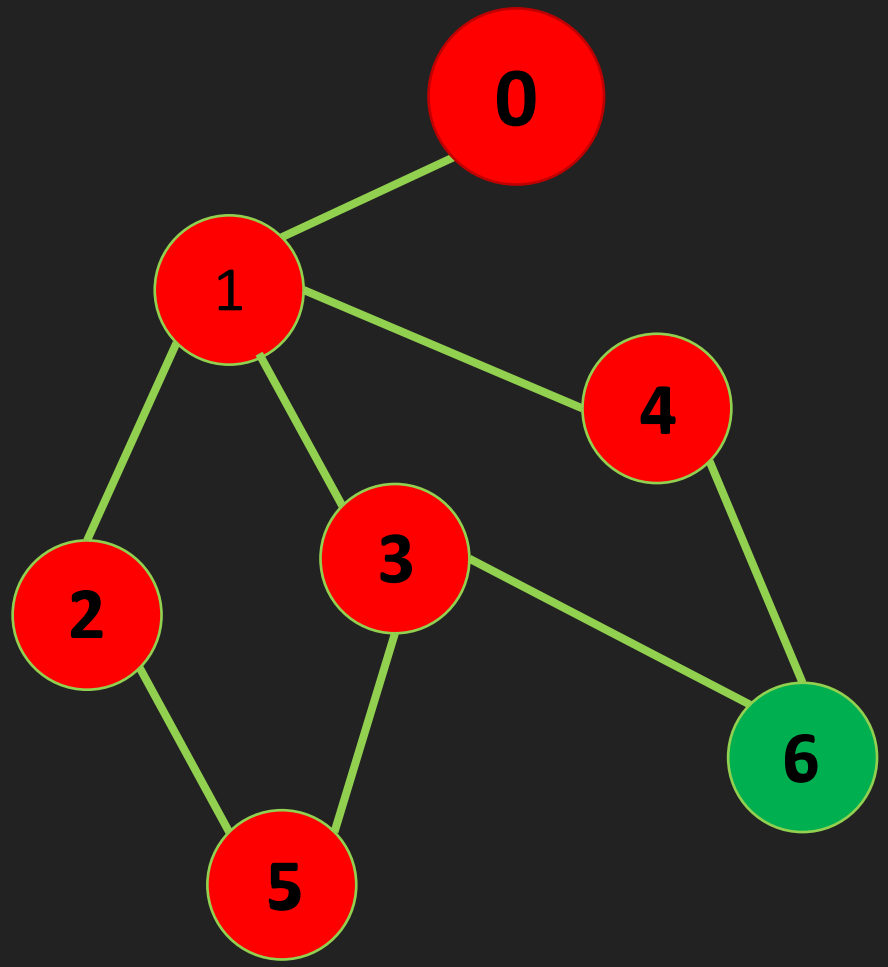
			4	3	2

queue

5. Tìm kiếm trên đồ thị

Step 4

Breadth Frist Search (Queue)



Thứ tự duyệt

0	1	2				

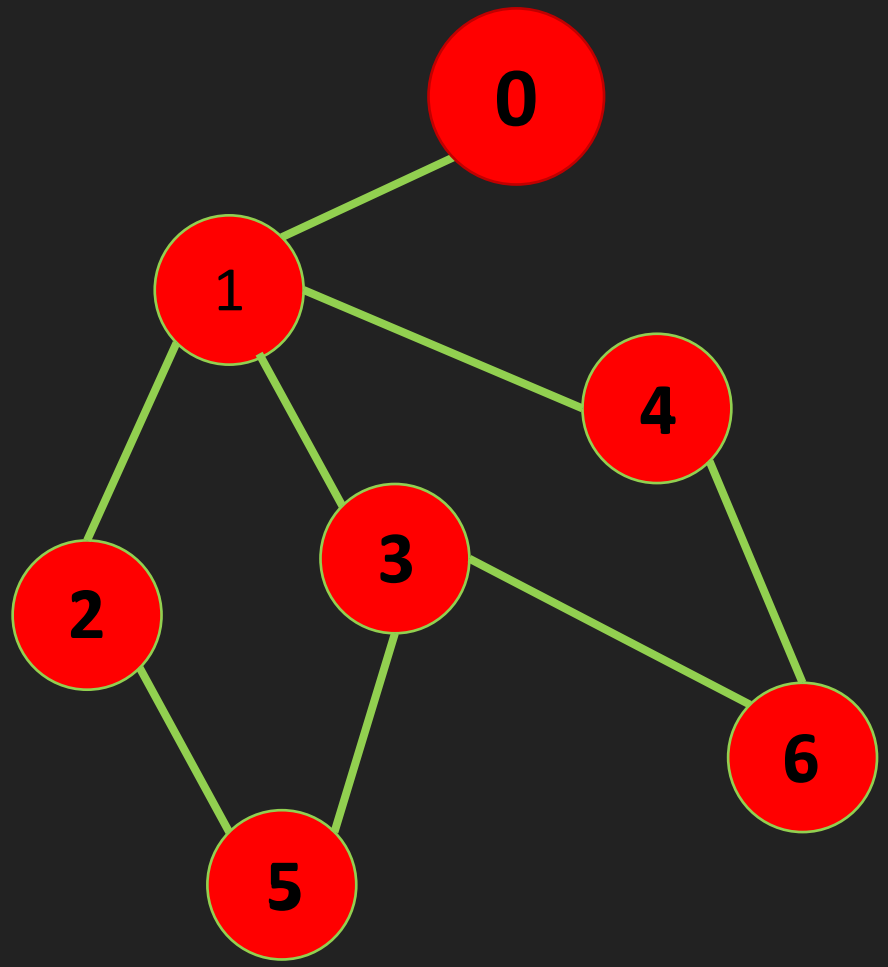
		5	4	3	

queue

5. Tìm kiếm trên đồ thị

Step 5

➢ **Breadth Frist Search (Queue)**



Thứ tự duyệt

0	1	2	3			

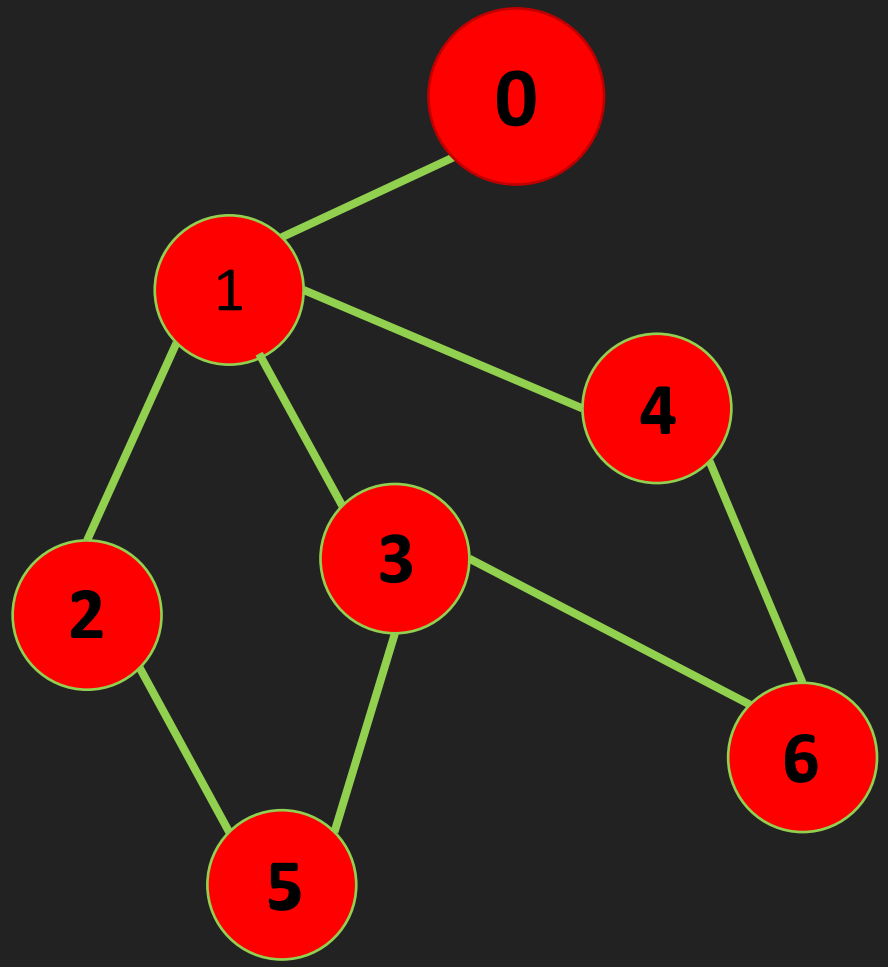
	6	5	4		

queue

5. Tìm kiếm trên đồ thị

➢ **Breadth Frist Search (Queue)**

Step 5



Thứ tự duyệt

0	1	2	3	4	5	6

✓ **Implement using Queue**

Data Structure & Algorithm

Please Like and Subcribe