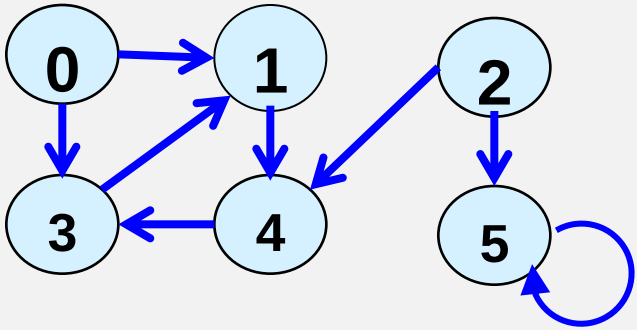
graph traversal

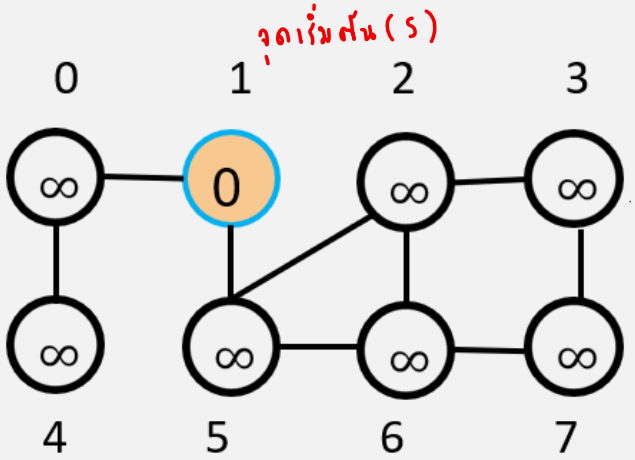


- กระบวนการเข้าไปเยือนโหนดในกราฟ
- แต่ละโหนดจะถูกเยือนเพียงครั้งเดียว

- ในกราฟระหว่างโหนดอาจจะมีหลายเส้นทาง ดังนั้นเพื่อป้องกันการท่องไปในเส้นทางที่ซ้ำเดิมจึงจำเป็นต้องทำเครื่องหมายบริเวณที่ได้เยือนเสร็จเรียบร้อยแล้วเพื่อไม่ให้เข้าไปเยือนอีก
- วิธีการท่องกราฟมี 2 แบบดังนี้
  - BFS *ได้เส้นทางสั้นที่สุด
  - DFS

4.3 Breadth-first search (BFS) : Algorithm for searching a graph

Definition : Given a graph G = (V,E) and distinguished source vertex s, bfs systematically explores the edges of G to discover every vertex that is reachable from s to all such reachable vertices.

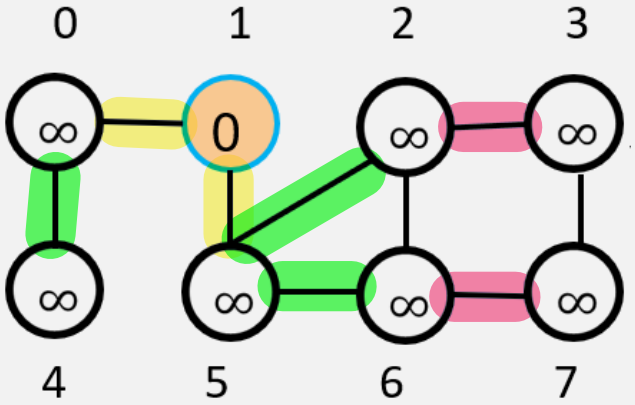


ใช้พวก list ธรรมดาในการออกสอบ, เข้าคิว

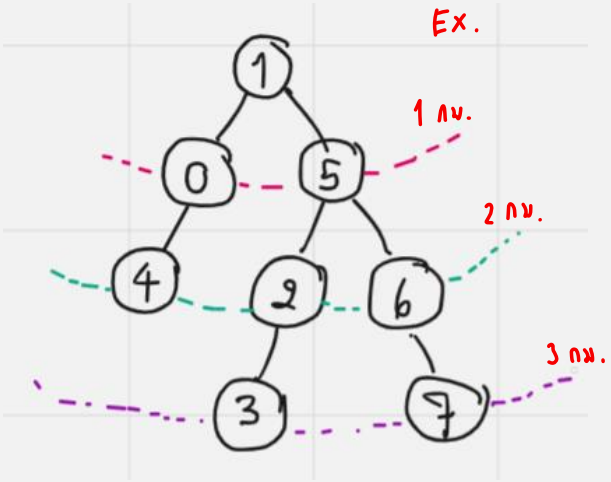
* Queue

Output - Compute distance เส้นทางที่สั้นที่สุด จาก S ไปยังทุกจุดที่เป็นไปได้

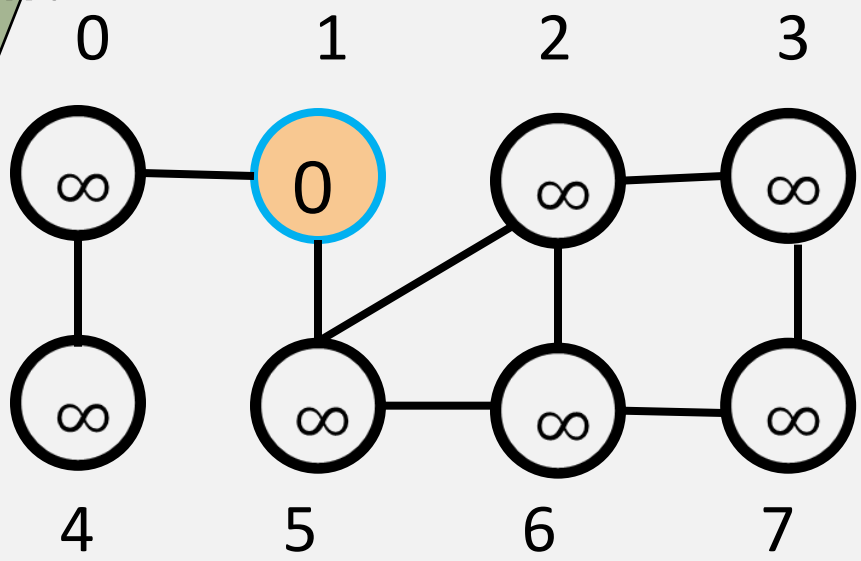
- Produces a bfs tree.



* ระยะทาง ระหว่าง node เท่ากัน



ทุกเส้นทาง
มีระยะ
เท่ากัน



linked list , Queue Size

Q | 1 | 0 | 5 *เอาคนออก deQueue ออก

Q | 0 | 5 | 4

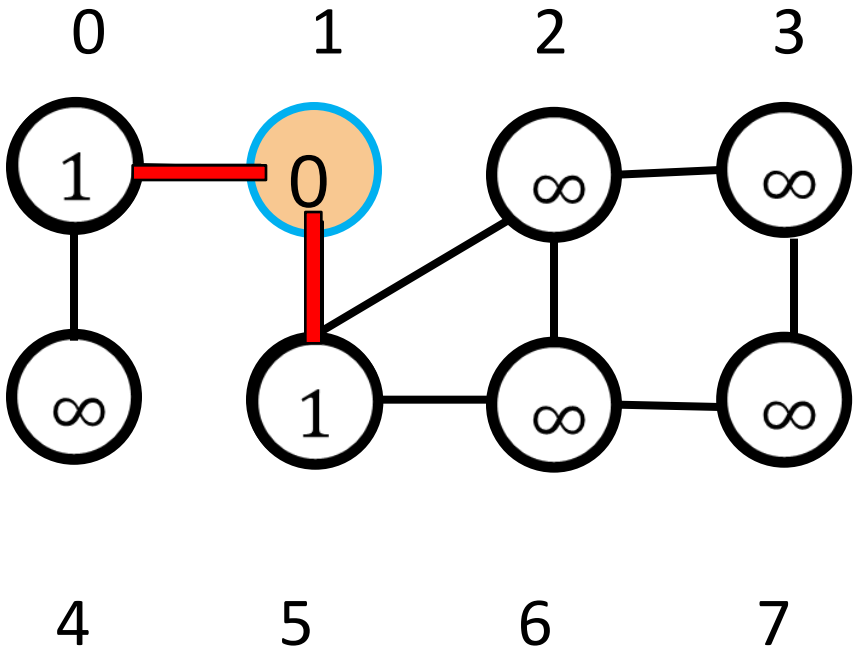
Q | 5 | 4 | 2 | 6

Q | 4 | 2 | 6

Q | 2 | 6 | 3

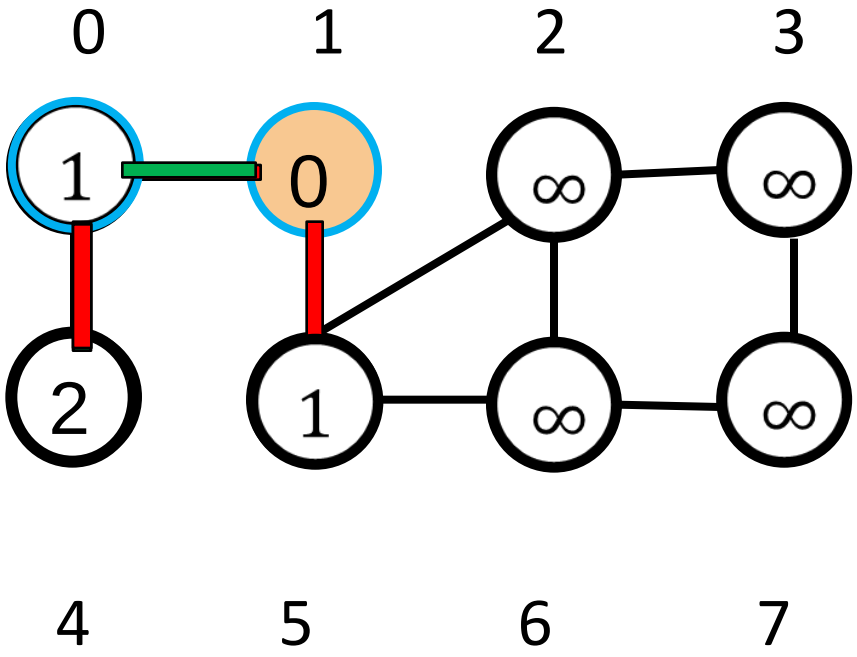
Q | 6 | 3 | 7

Q | 3 | 7 * เดินครบแล้ว



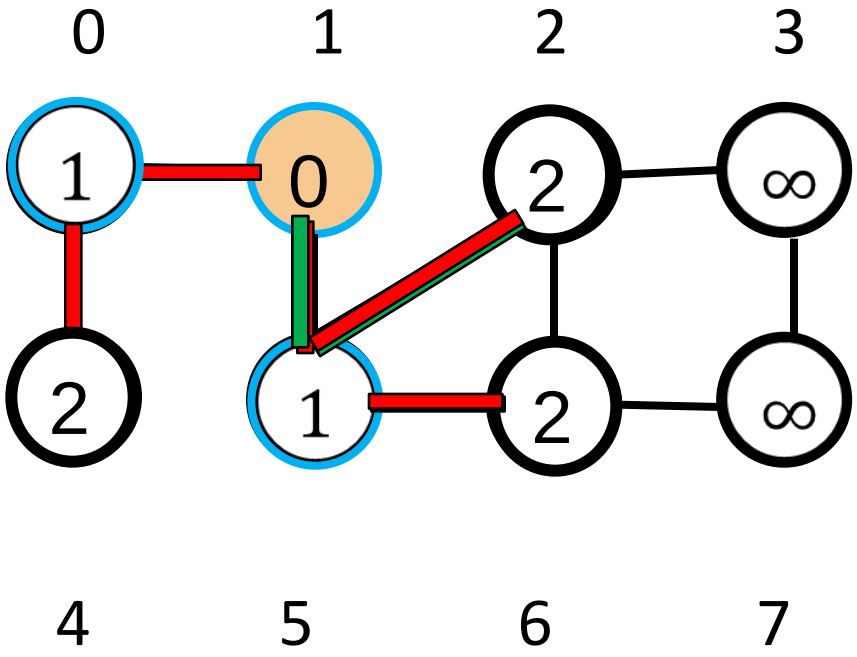
Q

1	0	5



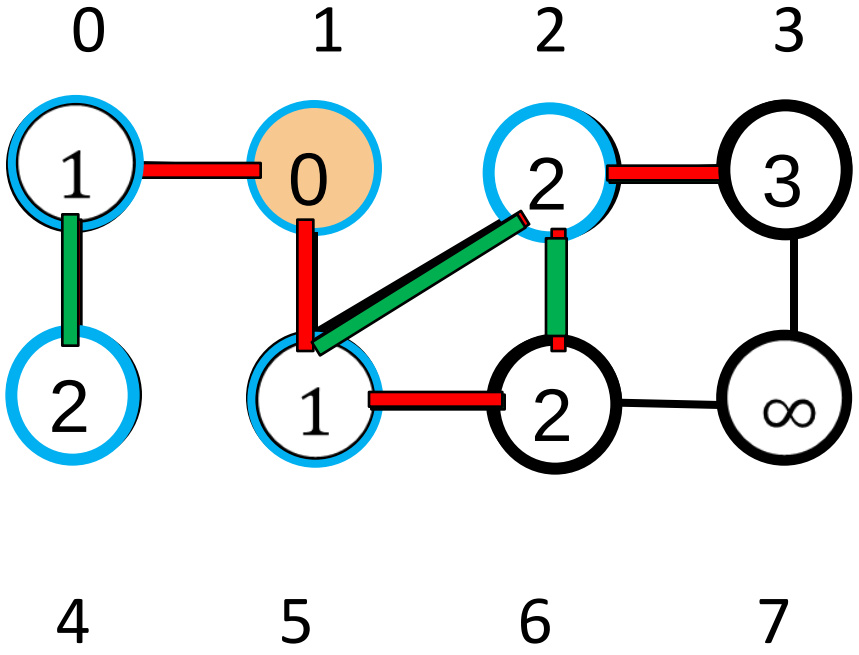
Q

0	5	4



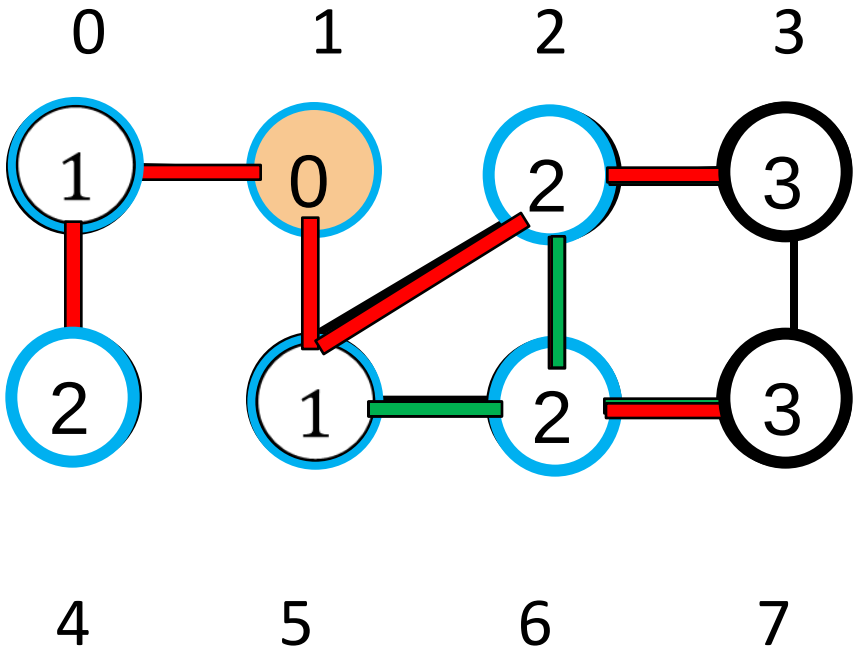
Q

5	4	2	6



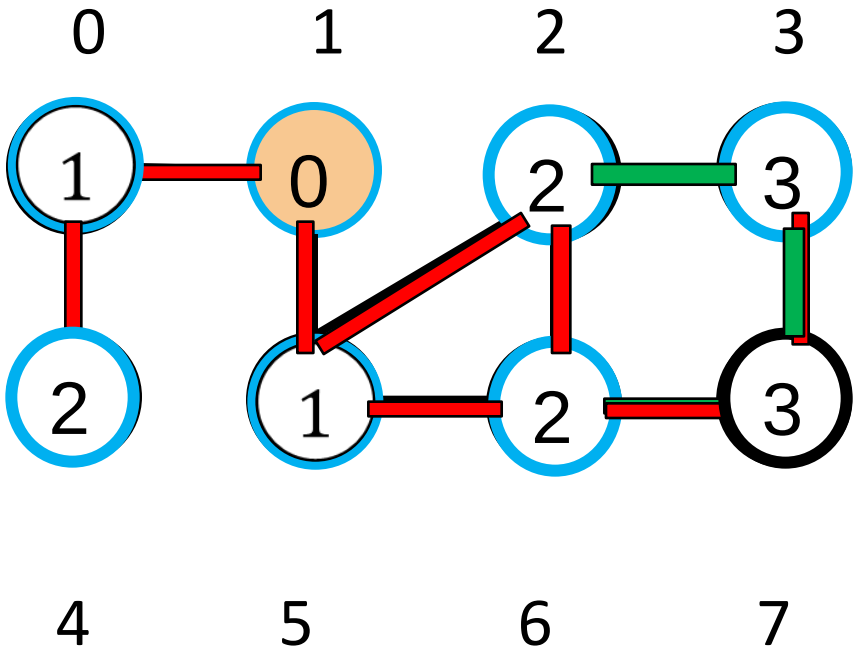
Q

4	2	6	3



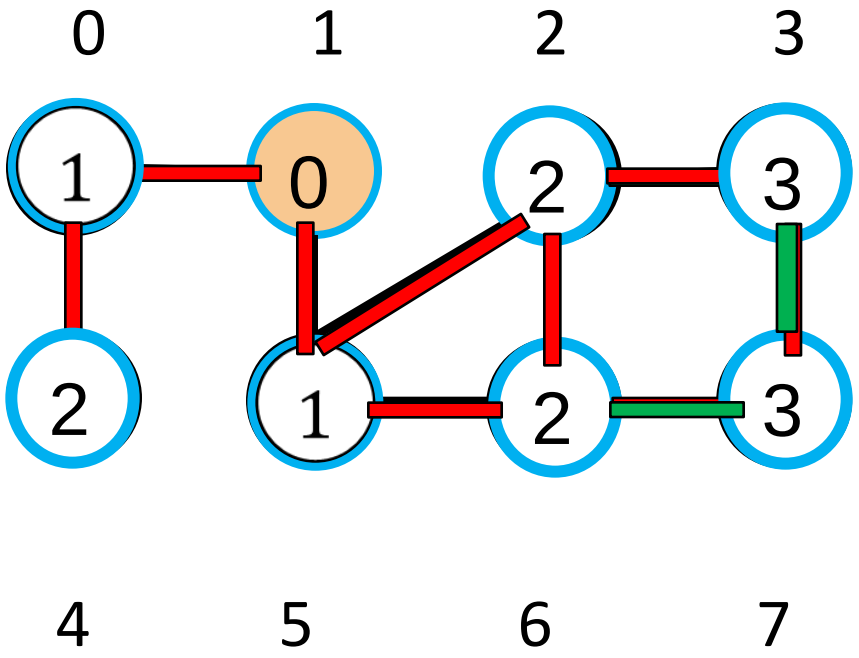
Q

6	3	7



Q

3	7



Q

7

4.3.1 BFS Algorithm

```
BFS(G,s)
for each vertex u E V[G] – {s}
do          pass[u]     = 0
            d[u]        = -1
            pred[u]     = -1
```

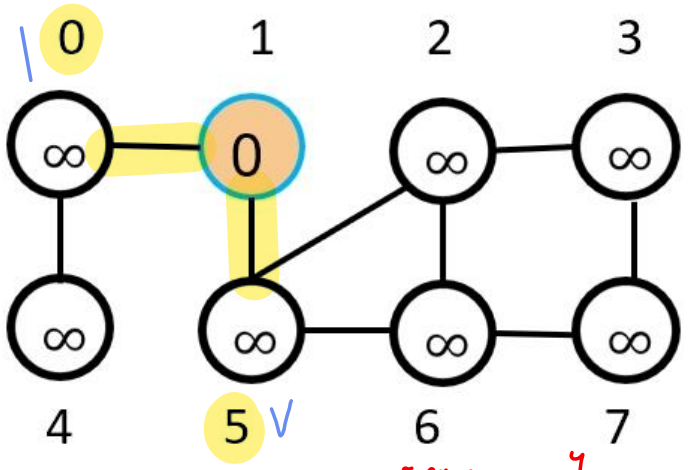
s = โหนดใด

```
pass[s]  =1
d[s]     =0
pred[s]  =s
```

```
Enqueue(s);
while (queuesize() != 0)
 {   u = head(q)
     node=adj[u];
     while( node!= NULL)
    {   v= node->value;
        if  (pass[v]=0)
        {   pass[v] = 1
            d[v] = d[u]+1
            pred[v] = u
            enqueue(v)
         }
         node=node->next;
     }
     dequeue()
}
```

Q | 1 | 0 | 5 | 4 | 2 | 6 | 3 |

ตามว่า 1 ไป 5 ที่ไกล 4 สุด
จาก 1 ไป 4 ใช้เส้นทางใด
(4,0) , (0,1)



* adj

				เดินผ่าน pass	ระยะทาง d	จากไหน pred
0	1080	1	4 (4000)	0	0 1	-1 -1
1	1024	0	5 (1050)	1	0 1	-1 -1
2		3	5	6	2	0 1	-1 1
3		2	7		3	0 1	3 -1	2 -1
4		0			4	0 1	2 -1	0 -1
5	2000	1	2 (3000)	6 (3050)	5	0 1	1 -1	1 -1
6		2	5	7	6	0 1	2 -1	5 -1
7		3	6		7	0 1	3 -1	6 -1

1. Edges are explored out of the most recently discovered vertex v that still has unexplored edge leaving it.

2. When all of v's edges have been explored, the search backtracks to explore edges leaving the vertex from which v has discovered. Until we have discovered all the vertices that are reachable from the original source vertex