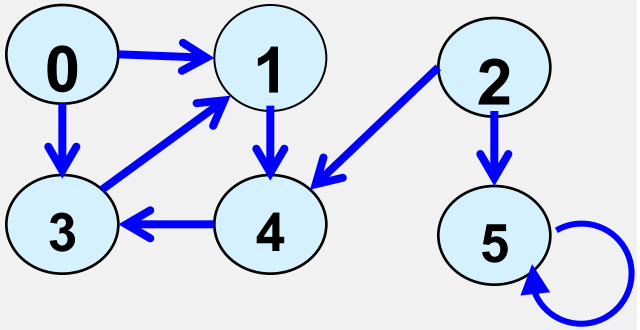
graph traversal



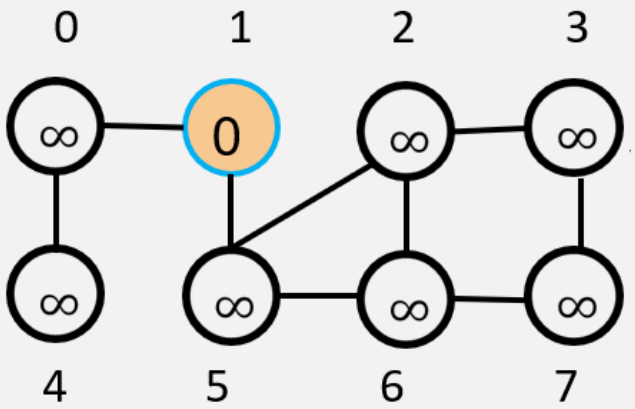
- กระบวนการเข้าไปเยือนโหนดในกราฟ
- แต่ละโหนดจะถูกเยือนเพียงครั้งเดียว

- ในกราฟระหว่างโหนดอาจจะมีหลายเส้นทาง ดังนั้นเพื่อป้องกันการท่องไปในเส้นทางที่ซ้ำเดิมจึงจำเป็นต้องทำเครื่องหมายบริเวณที่ได้เยือนเสร็จเรียบร้อยแล้วเพื่อไม่ให้เข้าไปเยือนอีก
- วิธีการท่องกราฟมี 2 แบบดังนี้
  - BFS
  - DFS

กระจาย

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

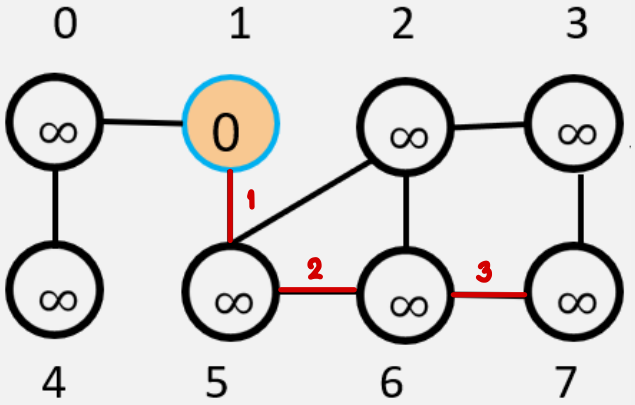
Definition : Given a graph G = (V,E) and distinguished source vertex s, bfs systematically explores the edges of G to discover every vertex that is reachable from s to all such reachable vertices.

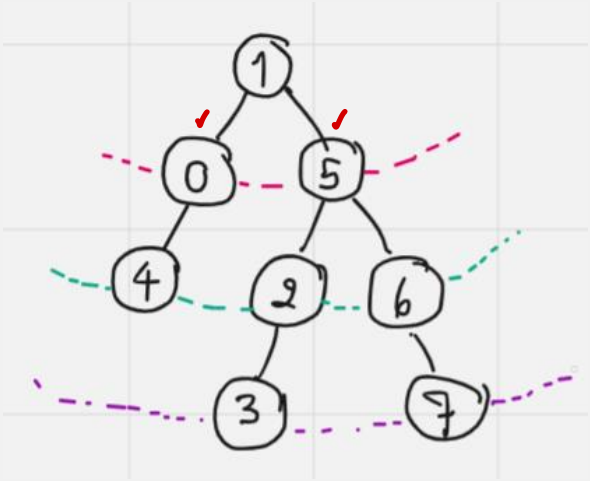


- ไปทุก vertex ที่ไปได้
- ได้ระยะทาง

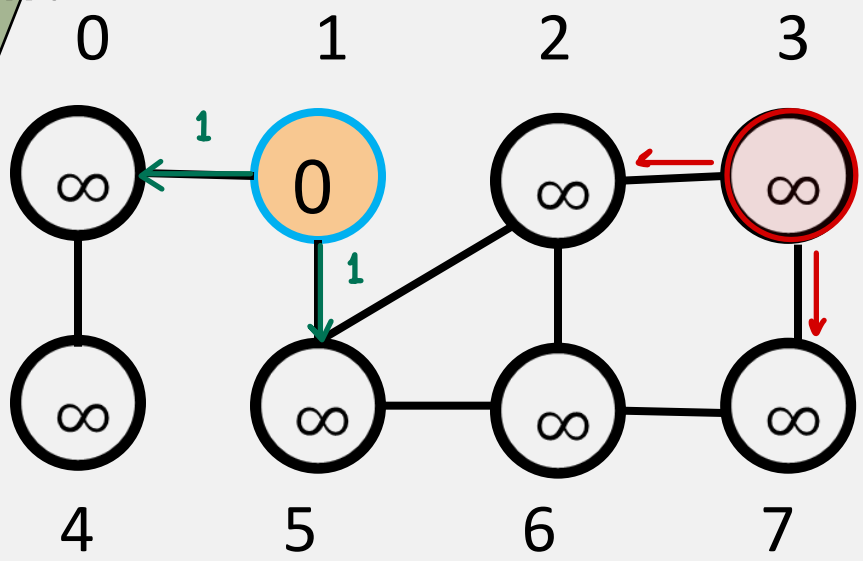
Output - Compute distance

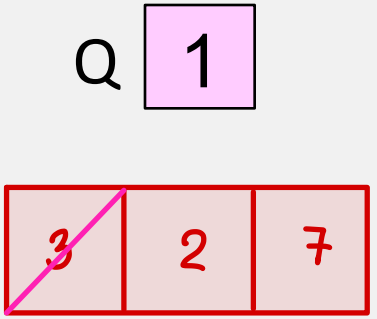
- Produces a bfs tree.

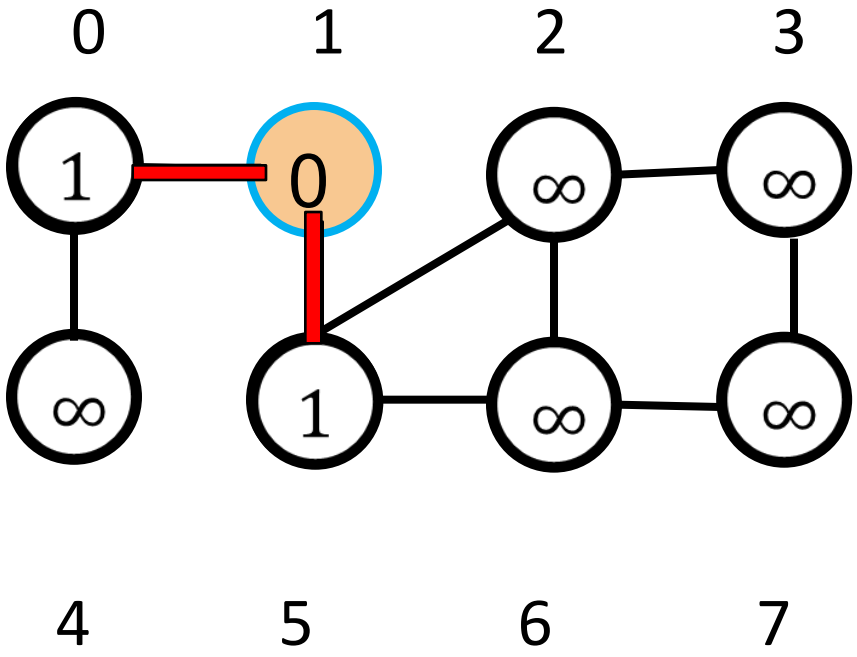




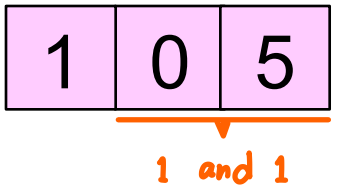
ทุกเส้นทางมีระยะเท่ากัน

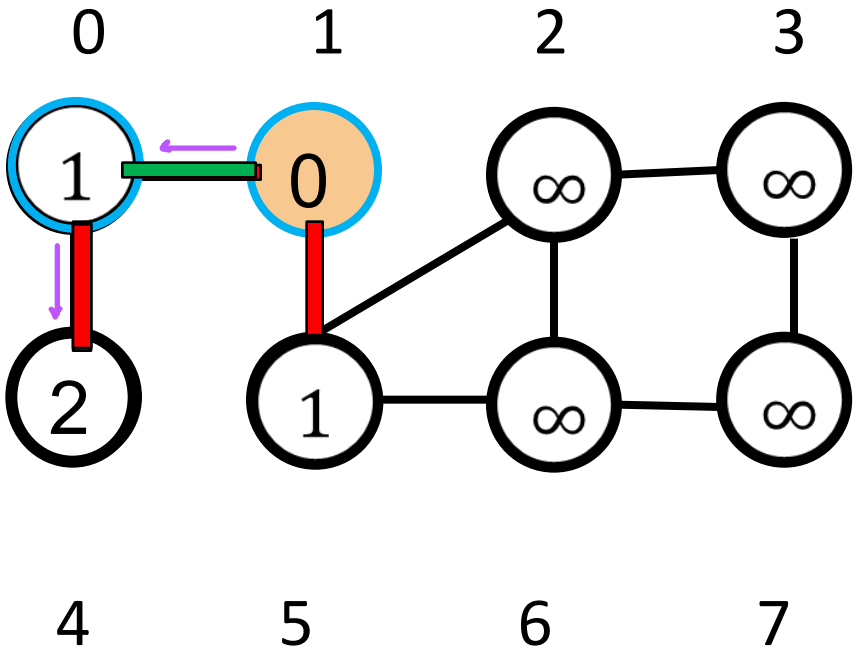






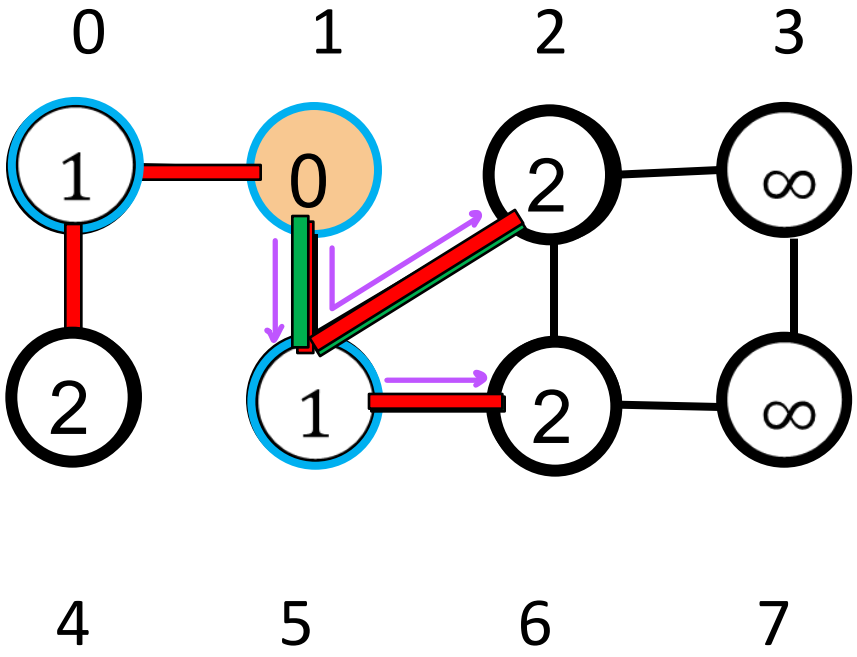
Q





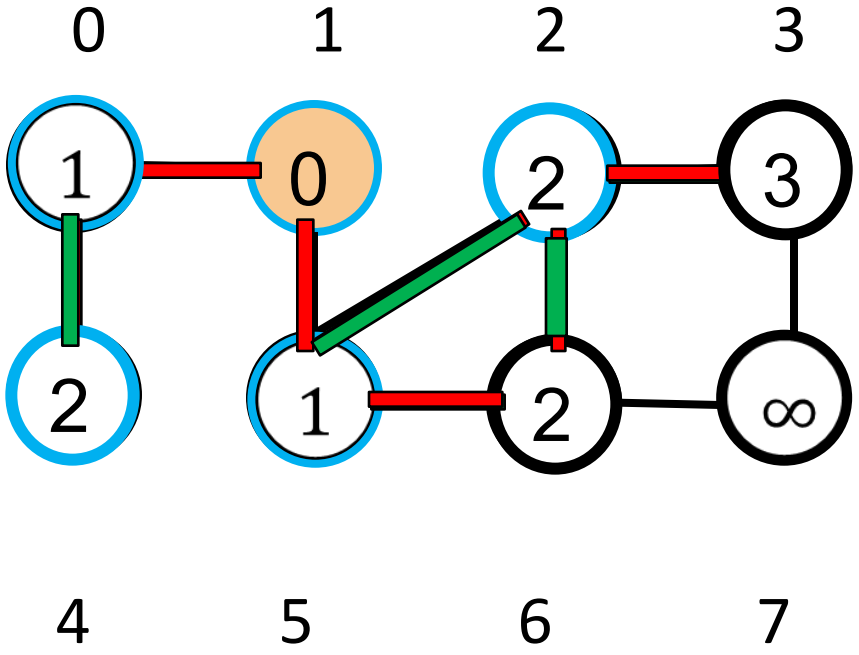
Q

0	5	4



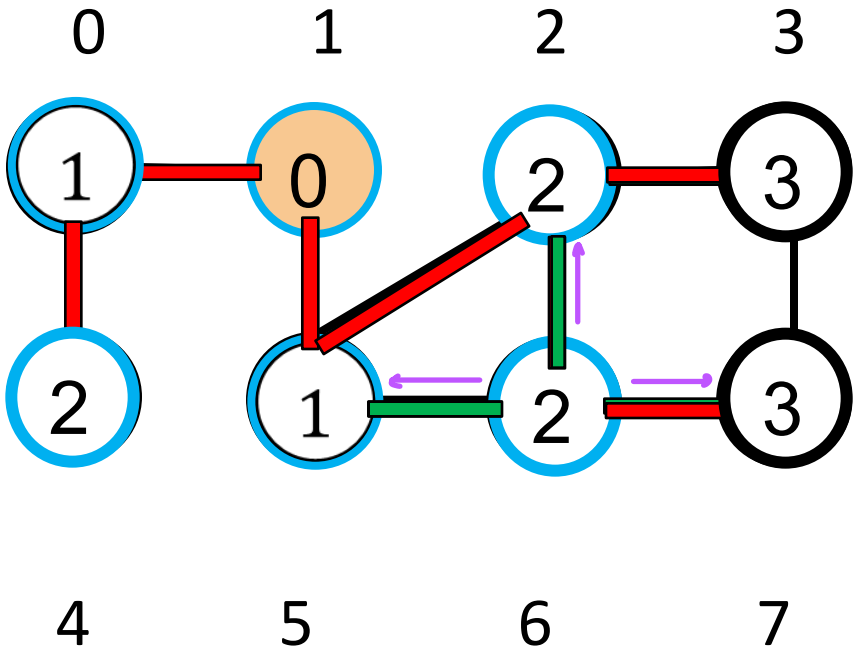
Q

5	4	2	6



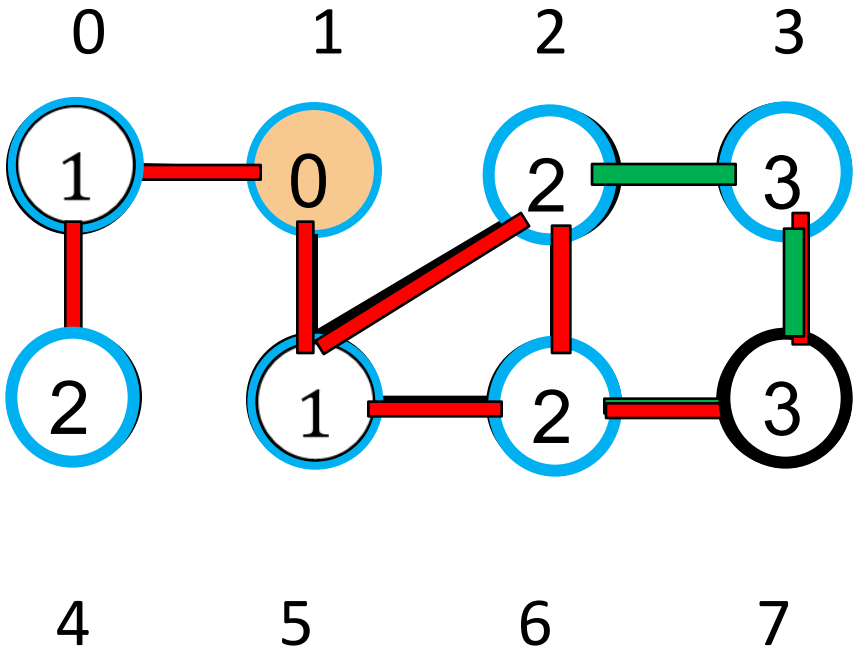
Q

4	2	6	3



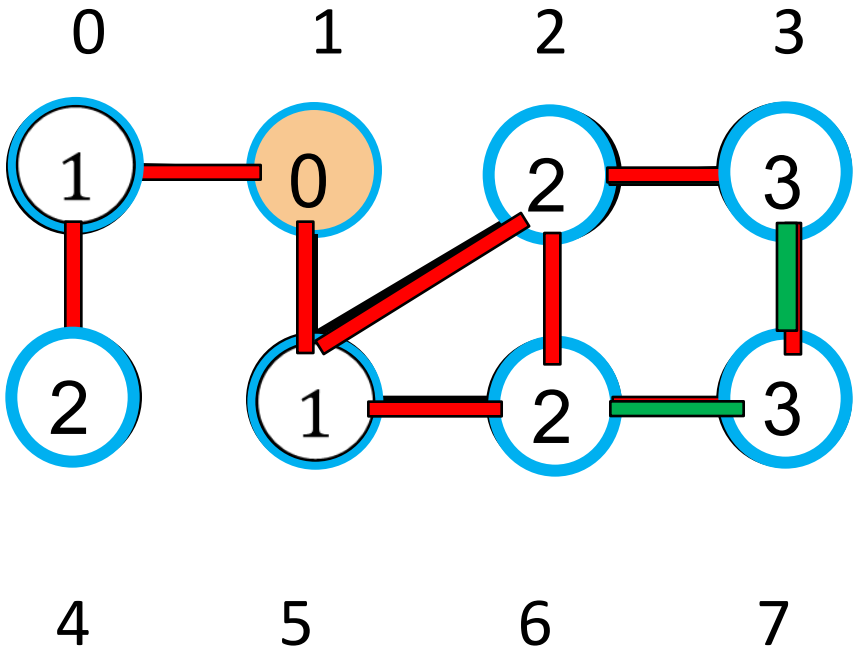
Q

6	3	7



Q

3	7



Q

7

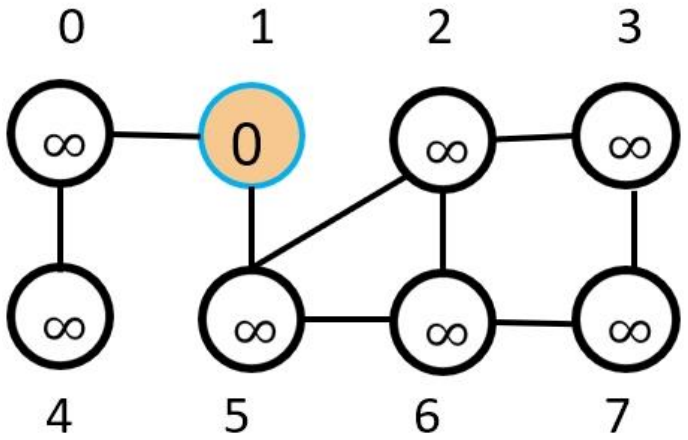
4.3.1 BFS Algorithm

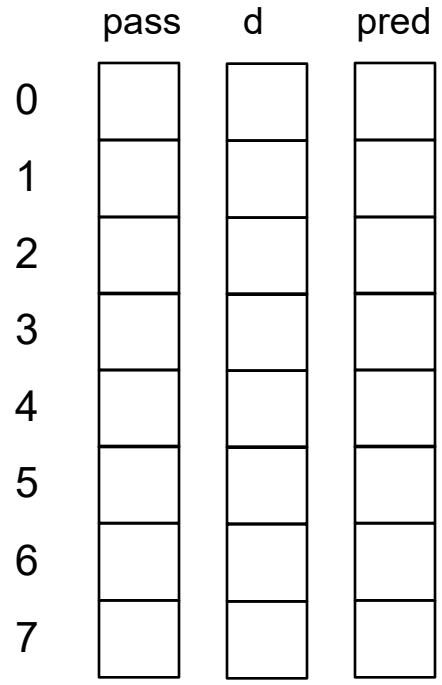
```
BFS(G,s)
for each vertex u E V[G] – {s}
do          pass[u]     = 0
            d[u]        = -1
            pred[u]     = -1
pass[s]  =1
d[s]     =0
pred[s]  =s
```

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
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()
}
```





```
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()
}
```

S = 1 Queue = 1

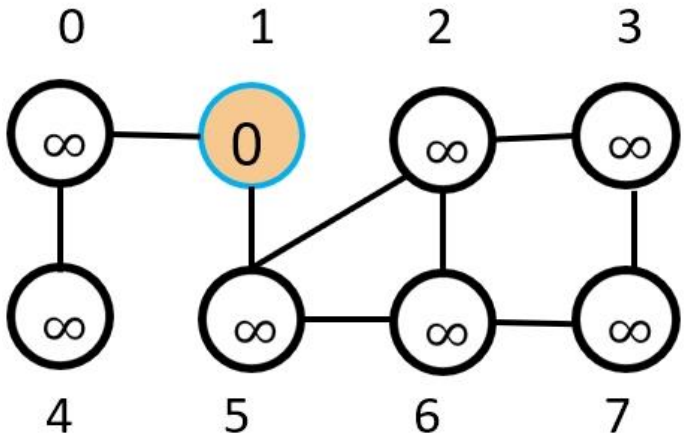
Q = [1]

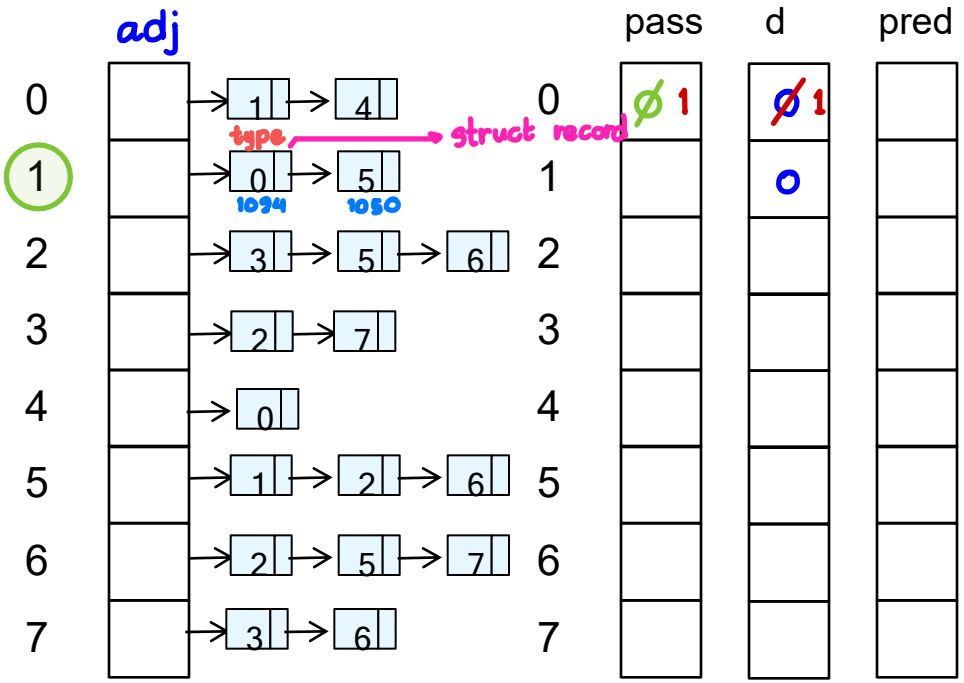
u = 1 < 0, 5

node = 1024

pred = predisessor

เก็บตัวเดินผ่าน



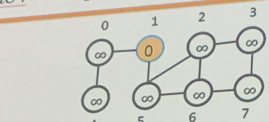


Enter source : 1
cin >> s;

03603212 : Module4 – Graph 15

s = 1;

```
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()
}
```



pointer

		pass	d	pred : parent
0	0	1	0	1
1	1	1	0	1
2	2	1	0	5
3	3	1	0	2
4	4	1	0	0
5	5	1	0	1
6	6	1	0	5
7	7	1	0	6

Q [1] [0] [5]

u = 1 < 5

u = 0 < 1, 4

u = 1
V = 0, V = 5

node 1024
1050

45%

Very humid
ENG
3:22 PM
9/11/202

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

23607

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