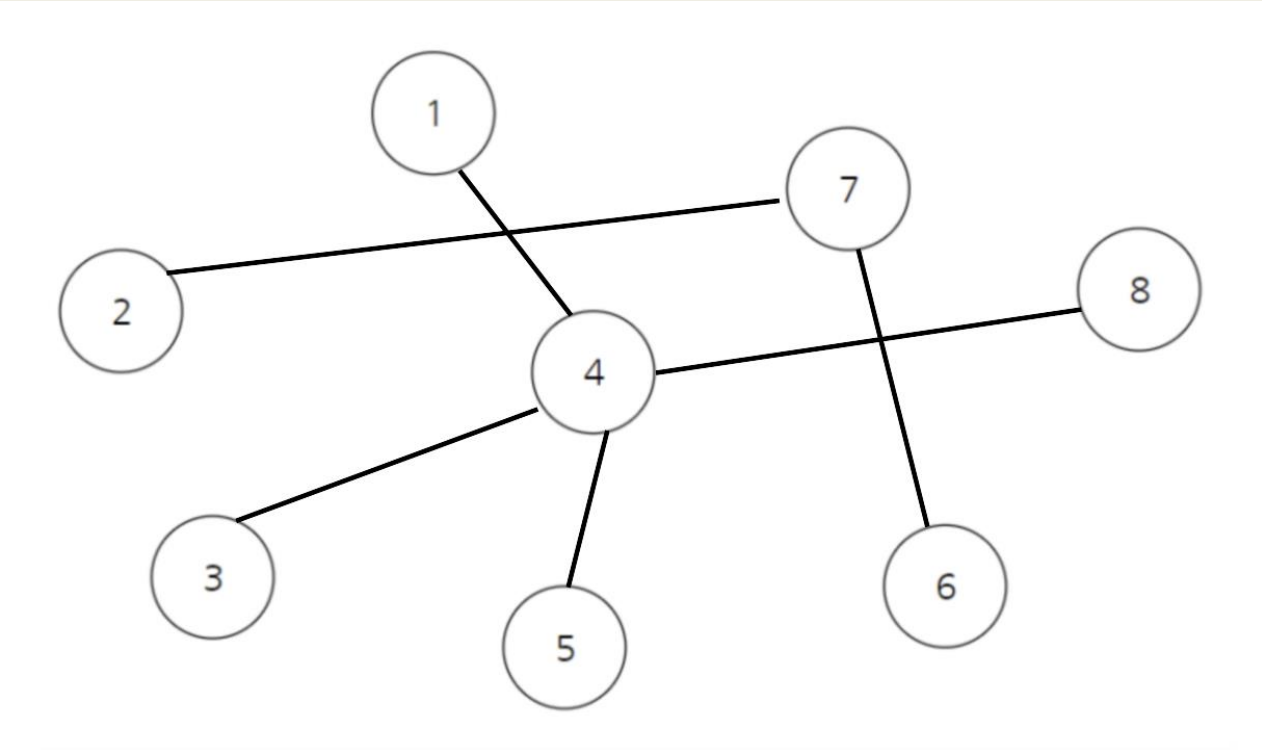
알고리즘 멘토링

- BFS, DFS –

- 주민찬 -

Graph

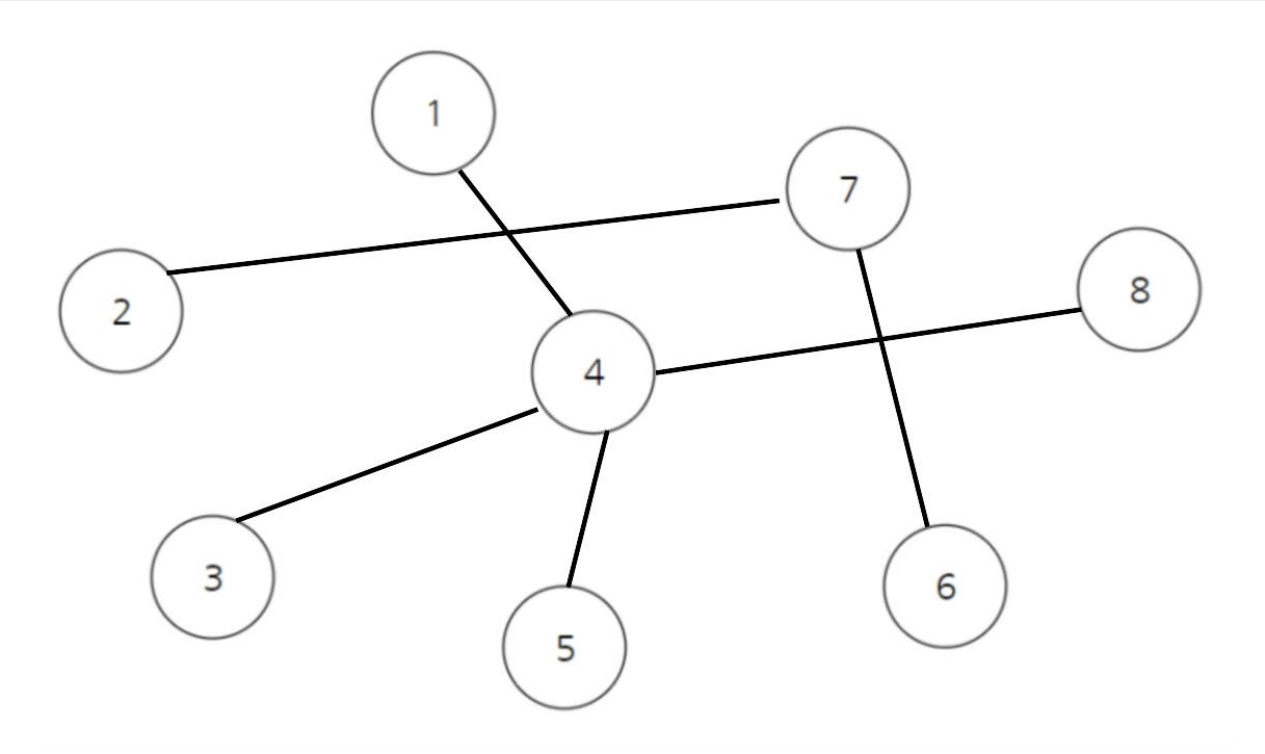


Graph는 node와 edge로 구분

Node : 1,2,3,4,5,6,7,8

Edge : (1,4), (3,4), (5,4),
(8,4), (2,7), (6,7)

Graph



파이썬에서 Graph는
어떻게 만들까?

Dictionary를 이용

{4 : [1,3,5,8]}

graph = {1 : [4], 2 : [7], 3 : [4], 4 : [1,3,5,8], 5 : [4], 6 : [7], 7 : [2,6], 8 : [4]}

Graph

- 백준에서 입력은 보통 node 수(N)와 edge 수(M)를 공백으로 구분하여 먼저 입력
- M개의 줄에 걸쳐 a,b를 공백으로 구분하여 입력해준다.
- 이때 node a와 node b가 edge로 이어져 있다는 뜻

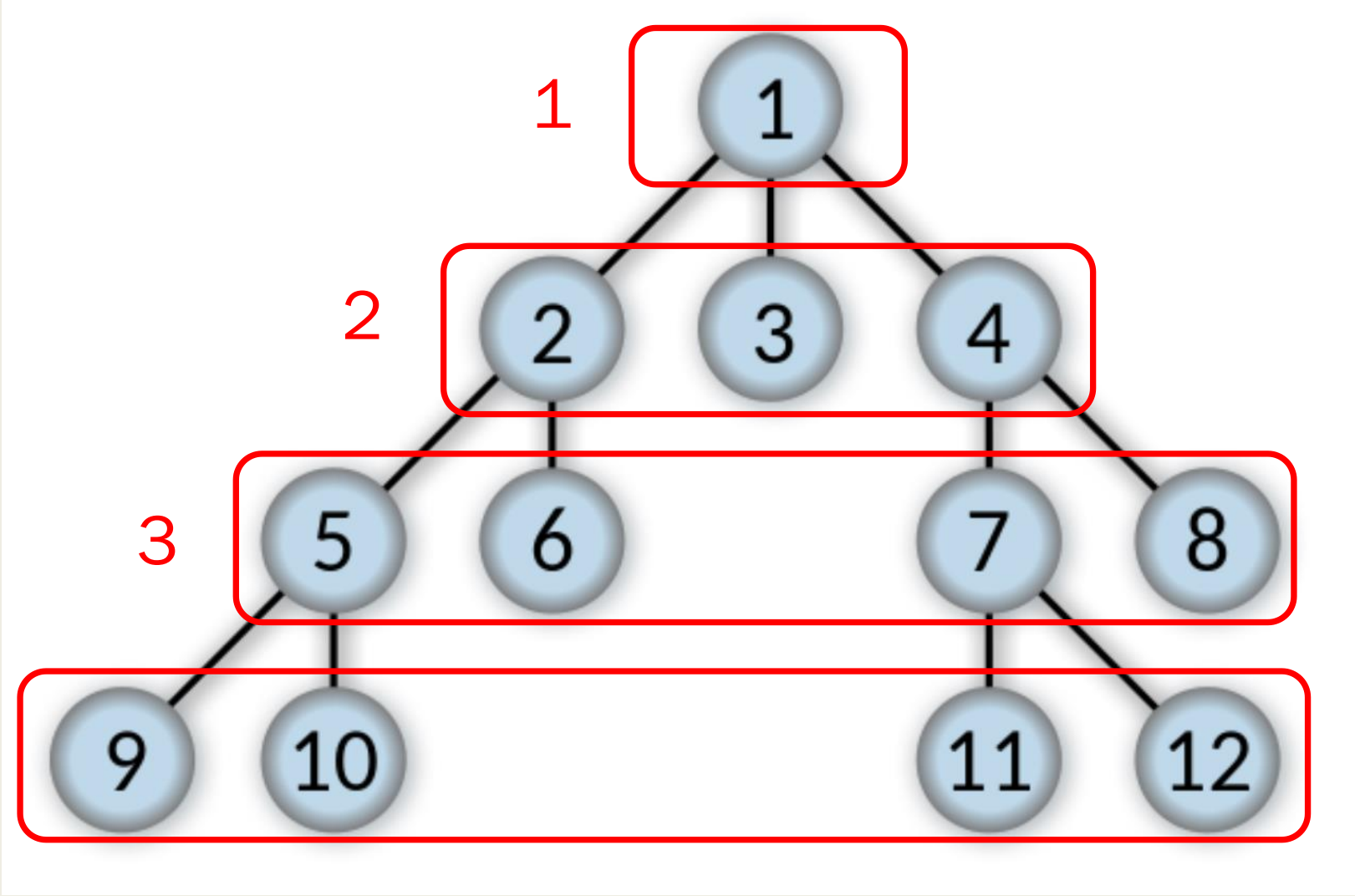
- Ex)

```
12 11
1 2
1 3
1 4
2 5
2 6
4 7
4 8
5 9
5 10
7 11
7 12
```

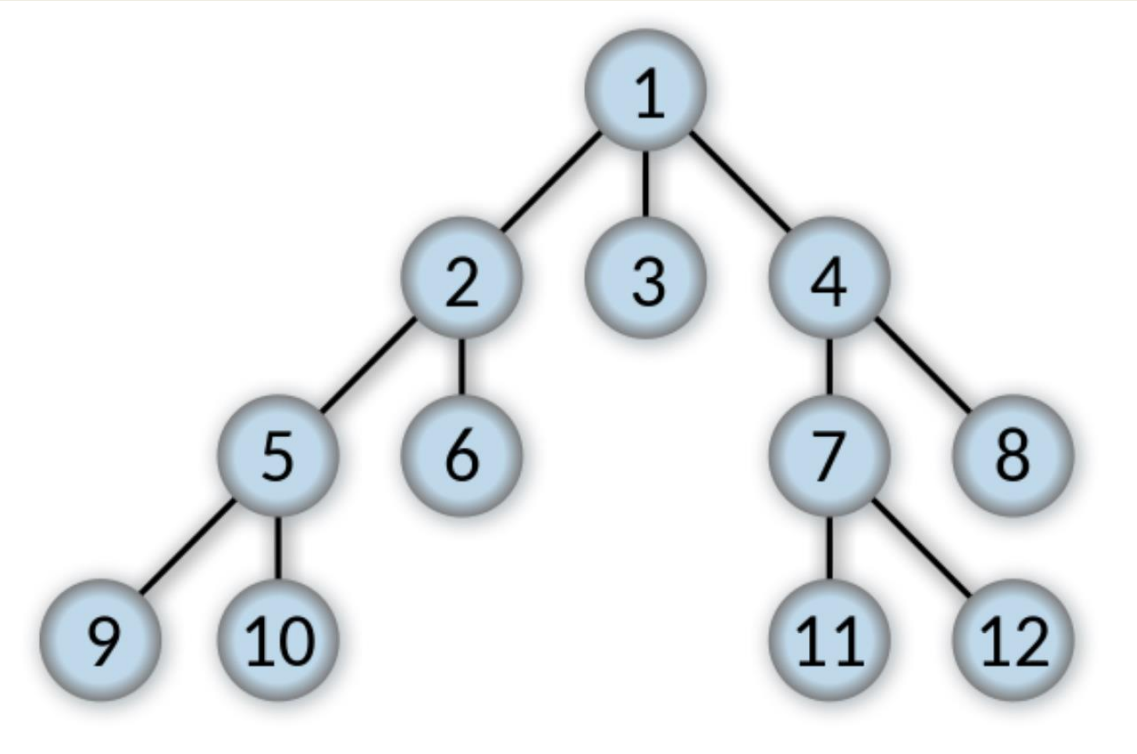
```python
N, M = map(int,input().split())
graph = {i:[] for i in range(1,N+1)}

for _ in range(M):
    a, b = map(int,input().split())
    graph[a].append(b)
    graph[b].append(a)

print(graph)
```

```
{1: [2, 3, 4], 2: [1, 5, 6], 3: [1], 4: [1, 7, 8], 5: [2, 9, 10], 6: [2], 7: [4, 11, 12], 8: [4], 9: [5], 10: [5], 11: [7], 12: [7]}
```

BFS (Breadth First Search)



BFS (Breadth First Search)

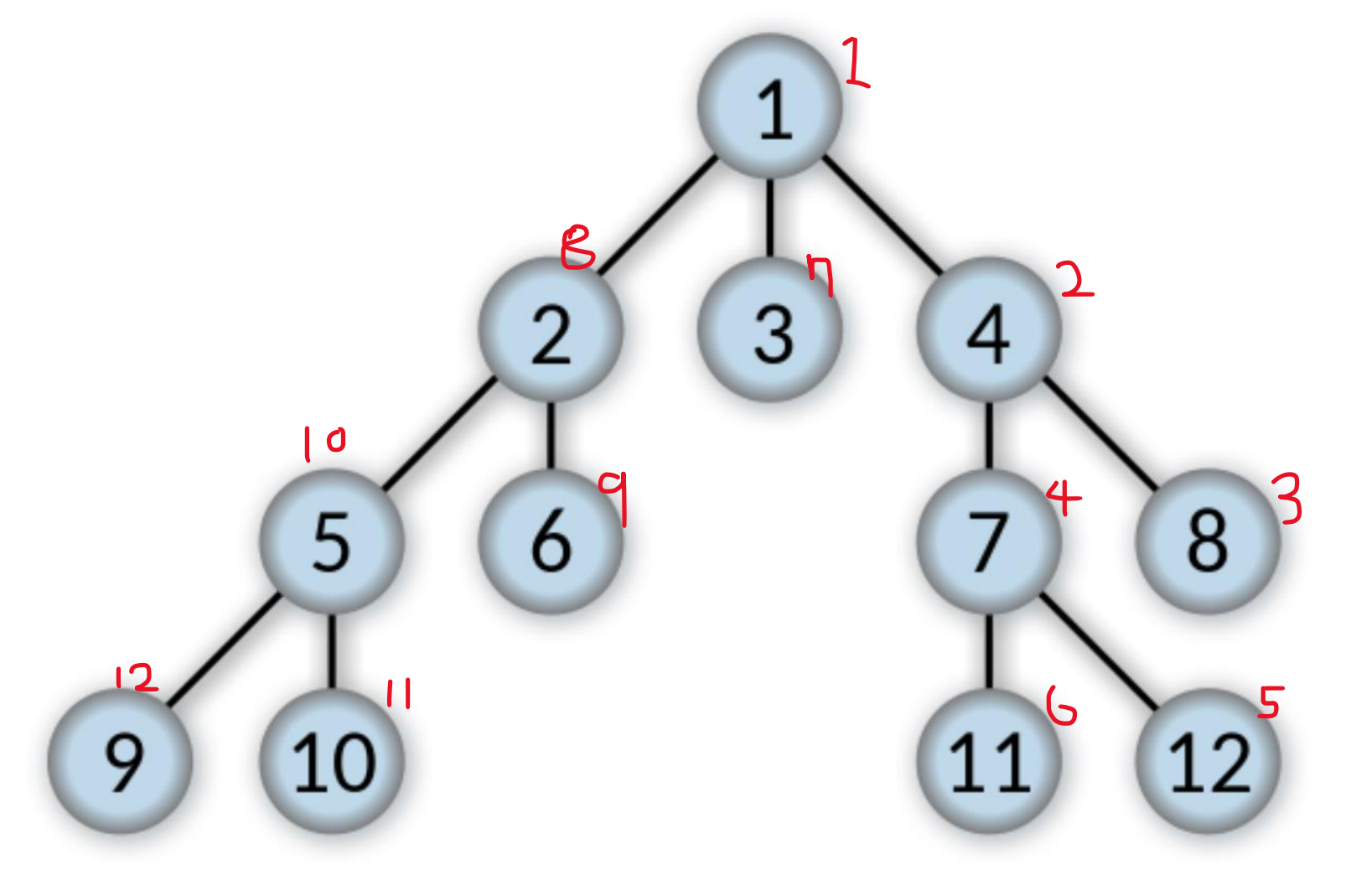


탐색 : queue 이용
순서 1 -> 2 -> 3 -> 4 -> 5 -> 6 -> 7
-> 8 -> 9 -> 10 -> 11 -> 12

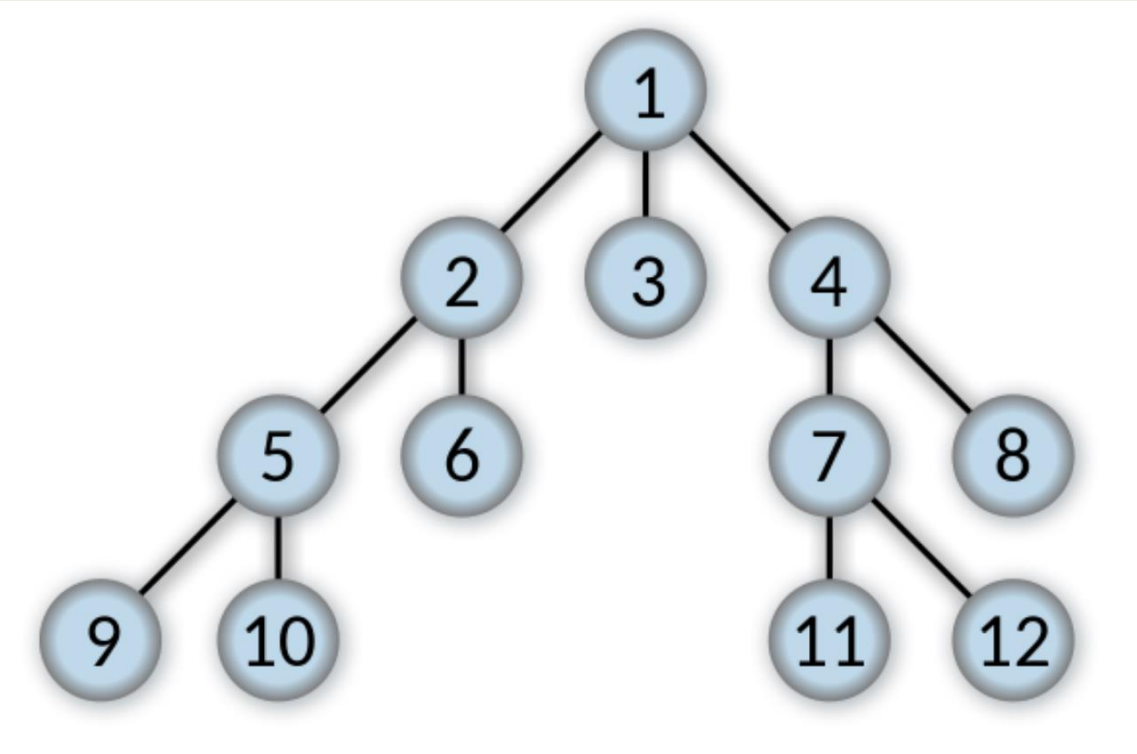
```
1  bfs(graph)
✓  0.0s

([0, 1, 2, 2, 2, 3, 3, 3, 3, 4, 4, 4, 4],
 [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12])
```

```python
from collections import deque

def bfs(graph,start=1,target=0):
    visit = [0 for _ in range(len(graph)+1)]
    queue = deque()
    queue.append(start)
    visit[start] = 1
    order = []
    while queue:
        node = queue.popleft()
        order.append(node)
        if node == target:
            return visit[node]
        for n_node in graph[node]:
            if visit[n_node]==0:
                queue.append(n_node)
                visit[n_node] = visit[node] + 1
    return visit, order
```

DFS (Depth First Search)



DFS (Depth First Search)



탐색 : stack 이용
순서 1 -> 4 -> 8 -> 7 -> 12 -> 11
-> 3 -> 2 -> 6 -> 5 -> 10 -> 9

```
1  dfs(graph)
✓ 0.0s
([0, 1, 2, 2, 2, 3, 3, 3, 3, 4, 4, 4, 4],
 [1, 4, 8, 7, 12, 11, 3, 2, 6, 5, 10, 9])
```

```python
def dfs(graph,start=1,target=0):
    visit = [0 for _ in range(len(graph)+1)]
    stack=[start]
    visit[start] = 1
    order = []
    while stack:
        node = stack.pop()
        order.append(node)
        if node == target:
            return visit[node]
        for n_node in graph[node]:
            if visit[n_node]==0:
                stack.append(n_node)
                visit[n_node] = visit[node] + 1
    return visit, order
```


실습!!