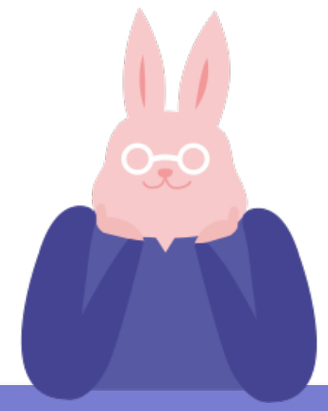
/* elice */

Graph Algorithm

2016. 12. 26.

신현규



[활동문제 0] DFS와 BFS

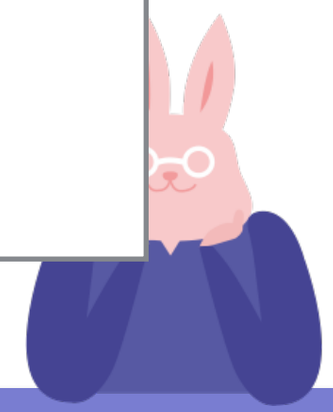
- DFS와 BFS한 결과를 각각 출력

입력의 예

```
7 8
0 1
0 3
1 2
1 3
2 4
2 5
2 6
5 6
```

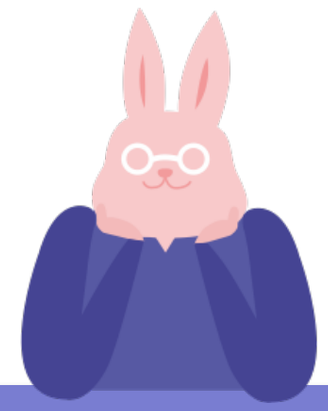
출력의 예

```
0 1 2 4 5 6 3
0 1 3 2 4 5 6
```



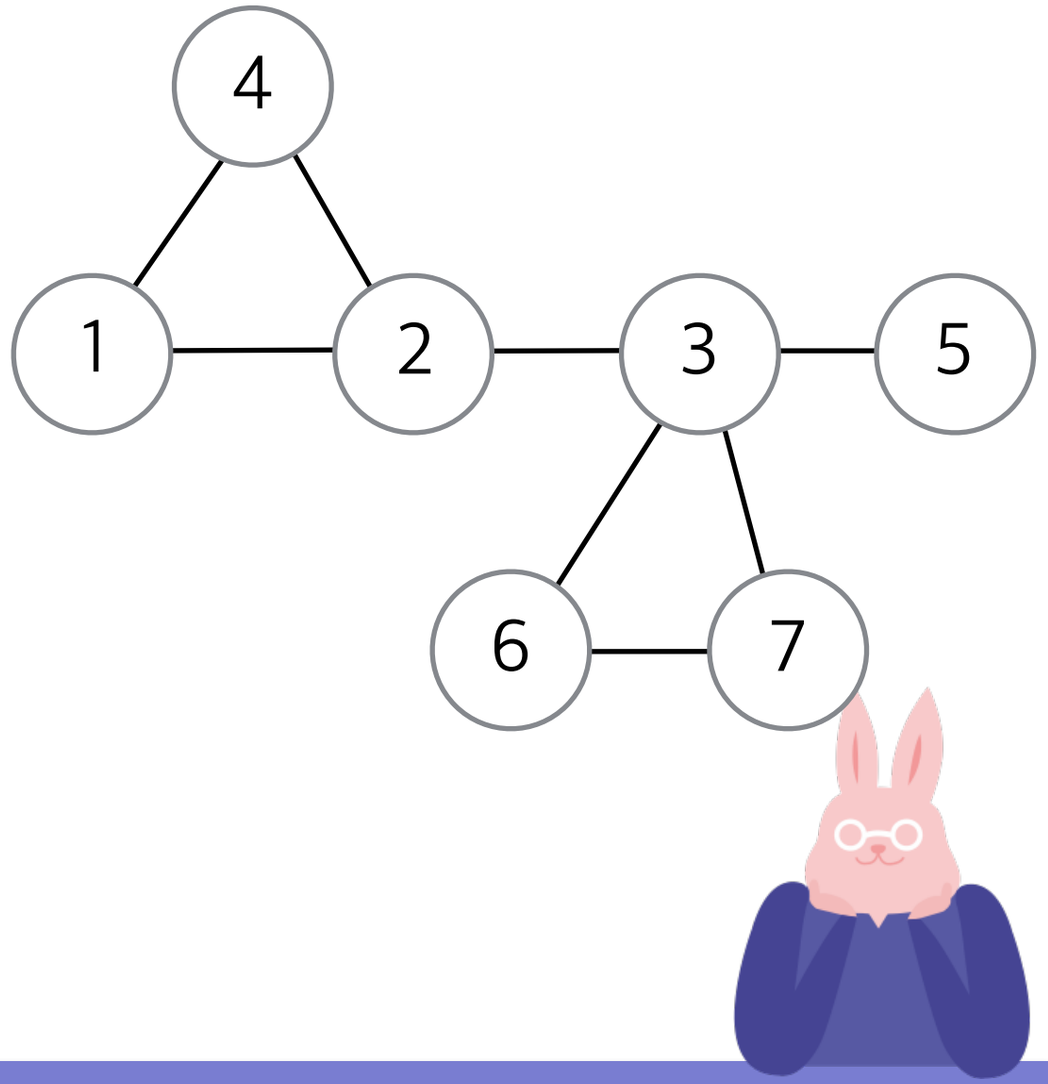
그래프 순회

- 깊이 우선 탐색 (DFS)
- 너비 우선 탐색 (BFS)



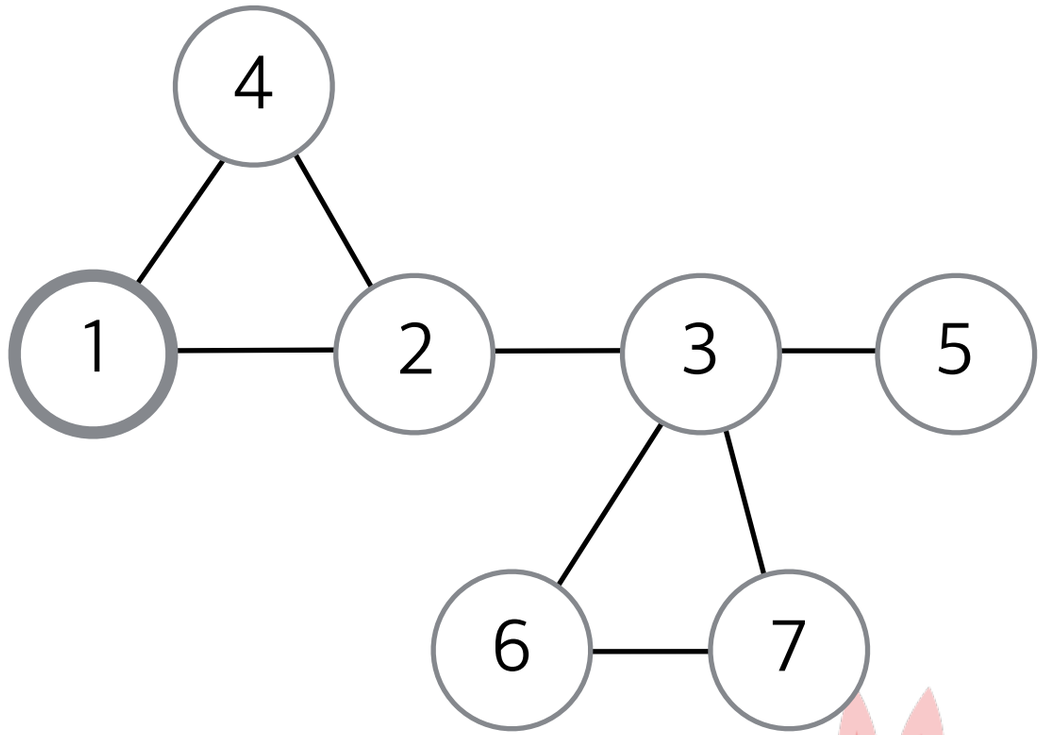
깊이 우선 탐색 (DFS)

- 인접한 Node 중에서 방문하지 않은 Node로 간다



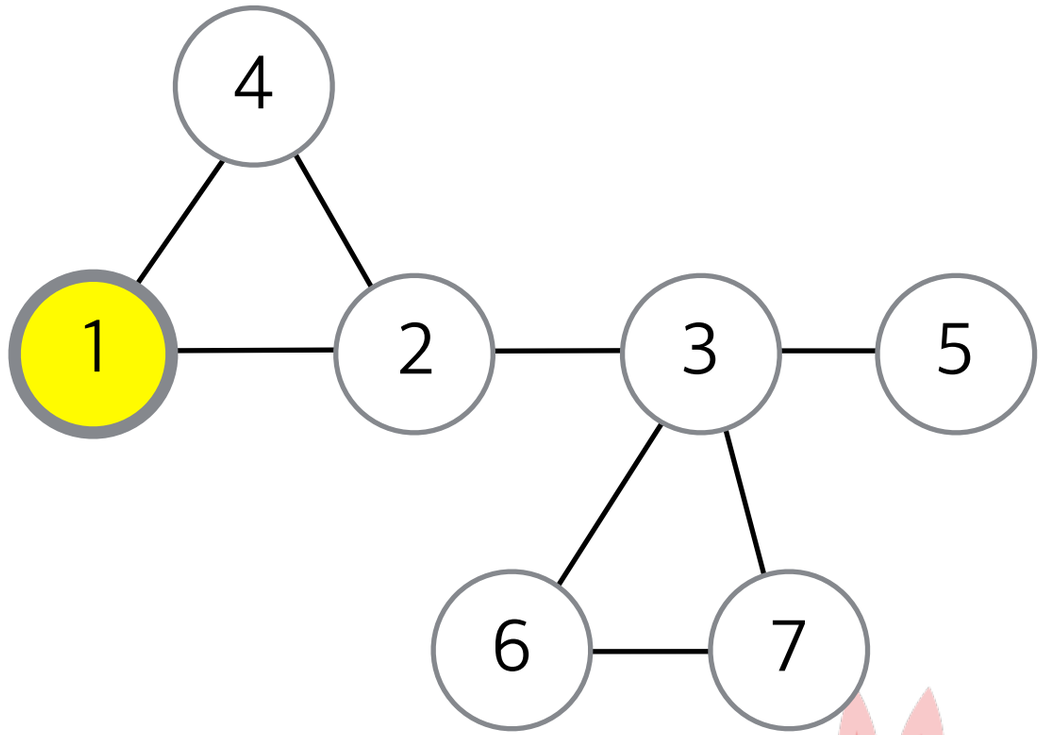
깊이 우선 탐색 (DFS)

- 인접한 Node 중에서 방문하지 않은 Node로 간다



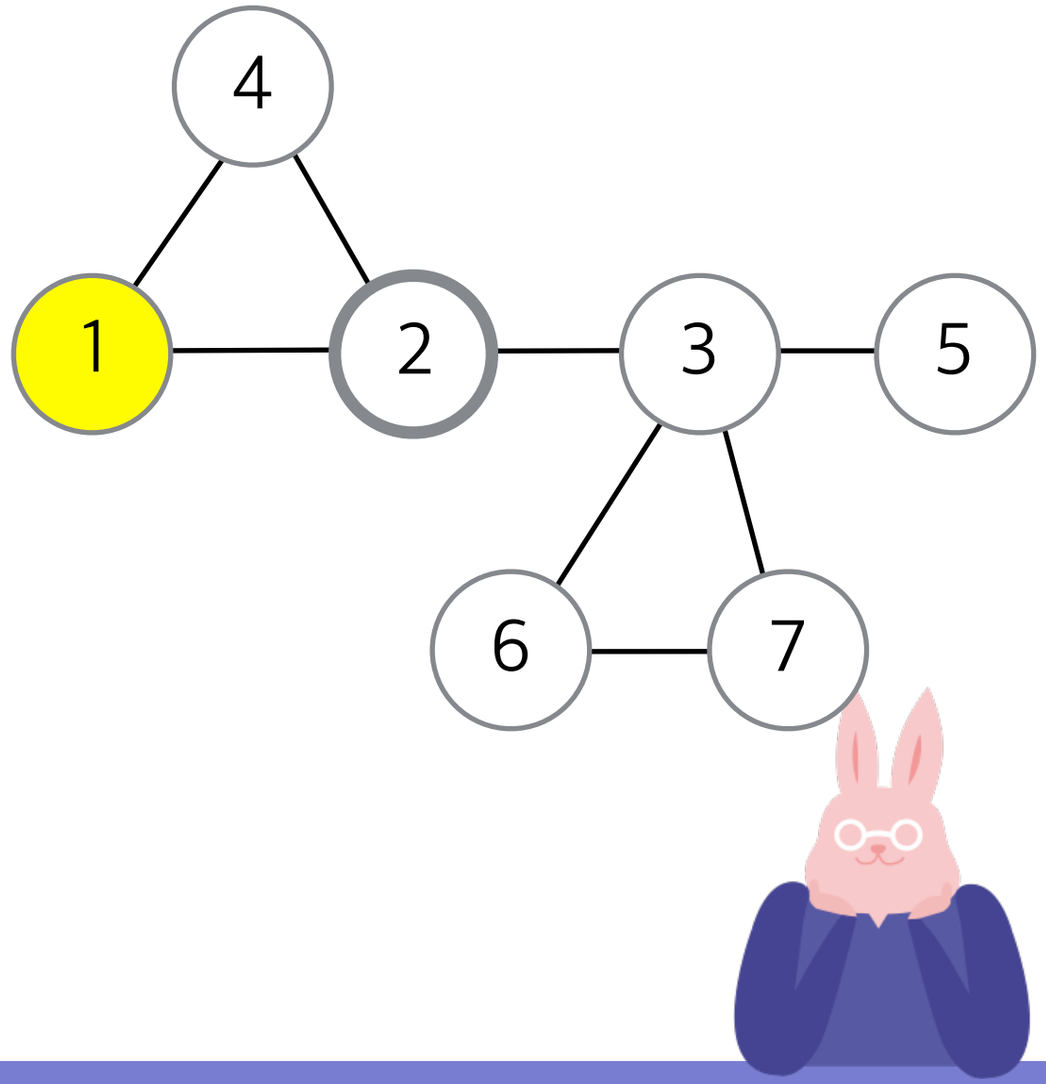
깊이 우선 탐색 (DFS)

- 인접한 Node 중에서 방문하지 않은 Node로 간다



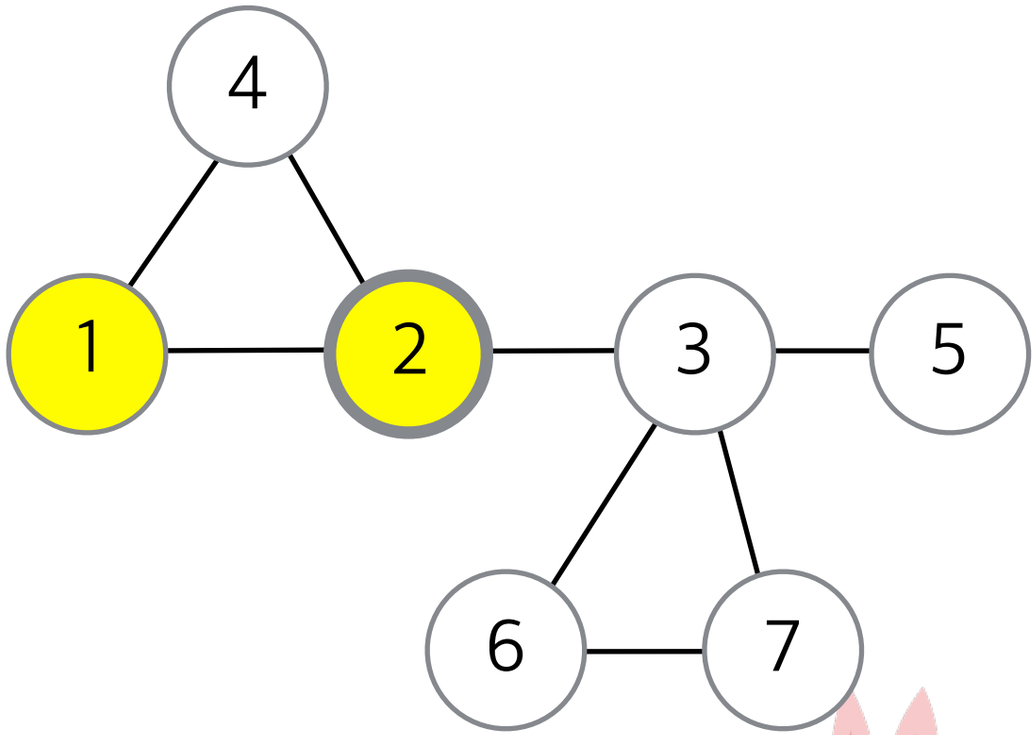
깊이 우선 탐색 (DFS)

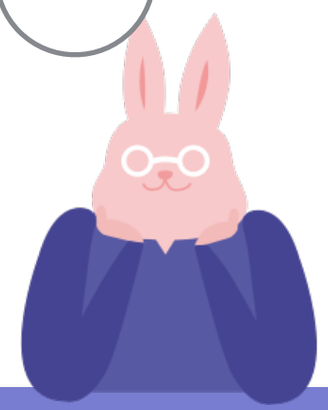
- 인접한 Node 중에서 방문하지 않은 Node로 간다



깊이 우선 탐색 (DFS)

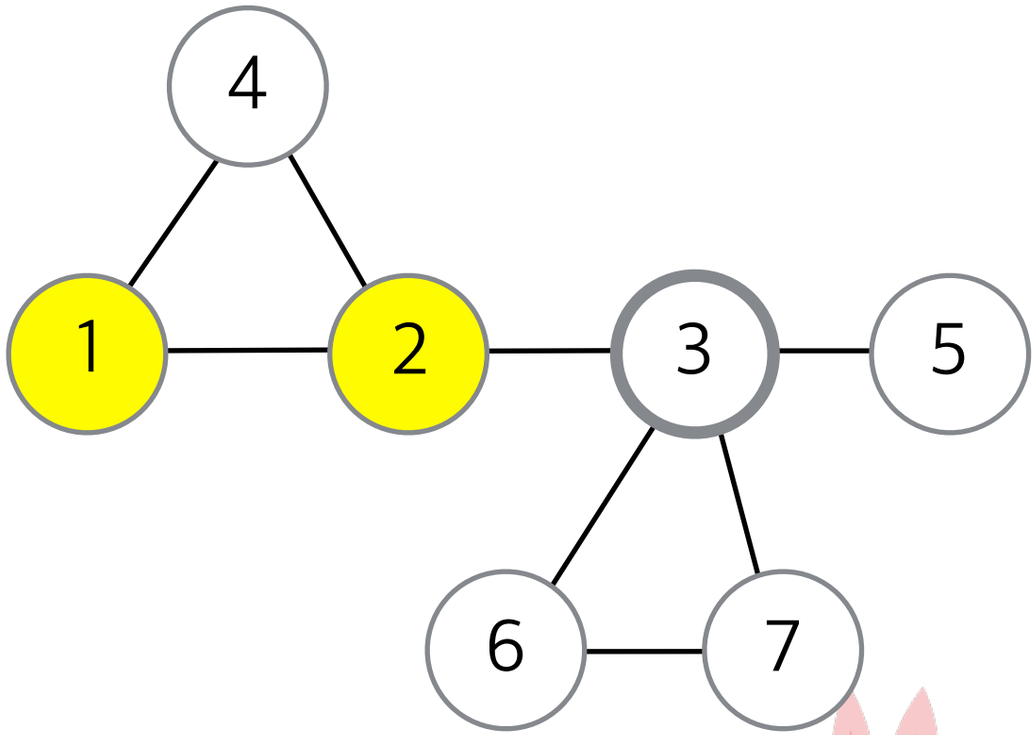
- 인접한 Node 중에서 방문하지 않은 Node로 간다

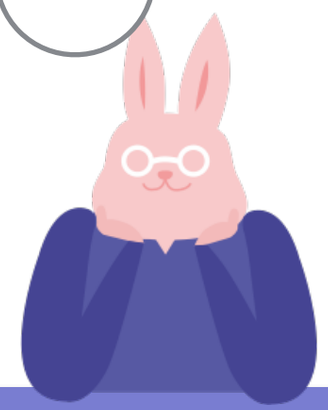




깊이 우선 탐색 (DFS)

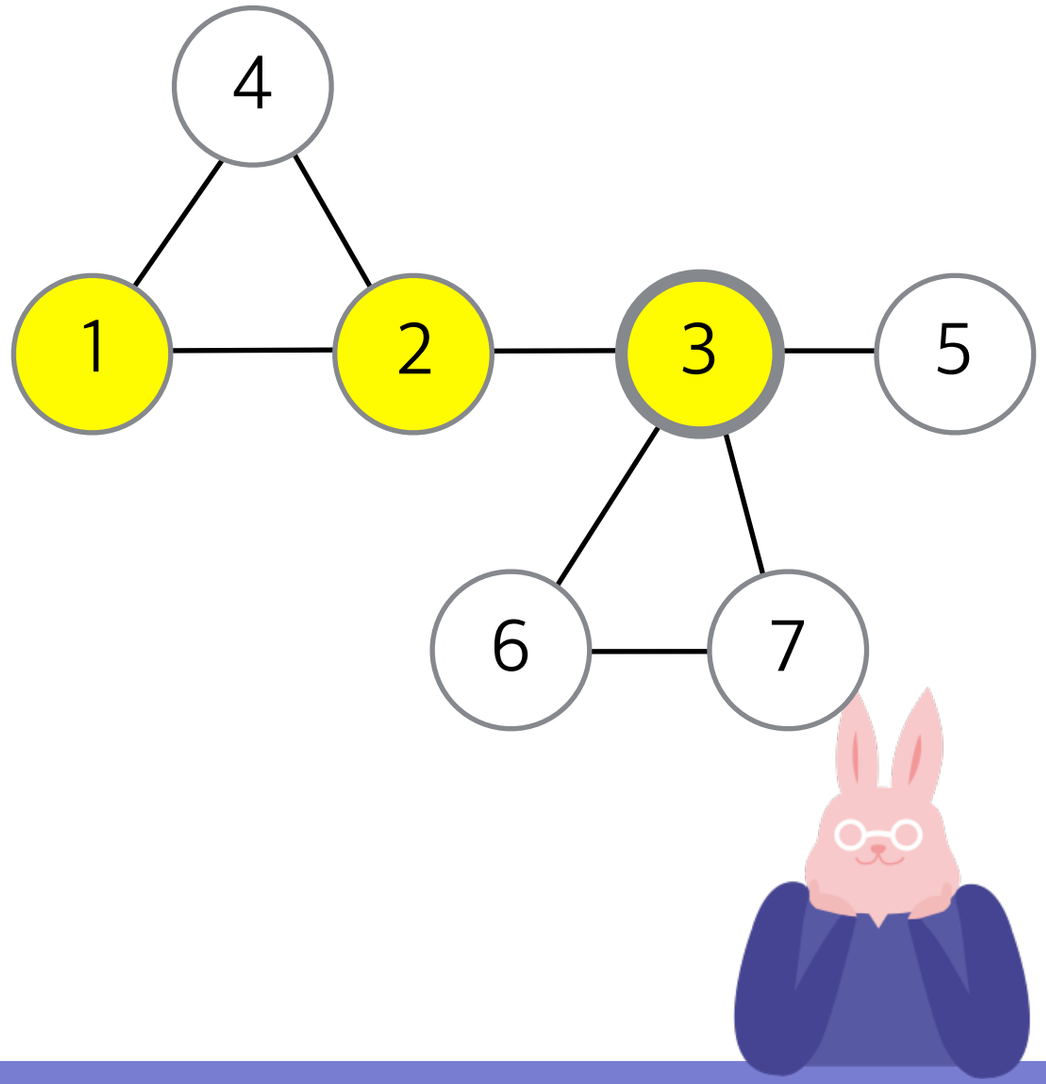
- 인접한 Node 중에서 방문하지 않은 Node로 간다





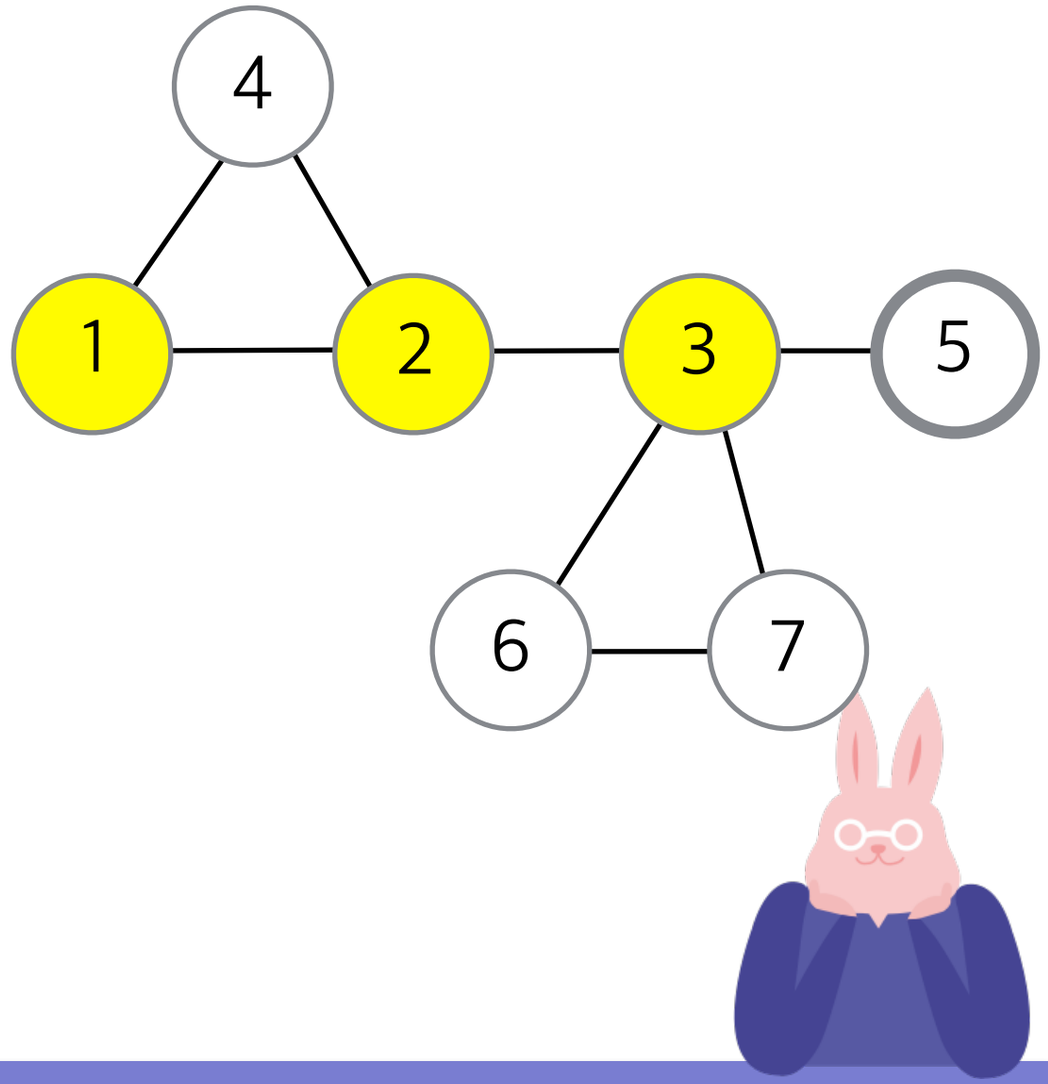
깊이 우선 탐색 (DFS)

- 인접한 Node 중에서 방문하지 않은 Node로 간다



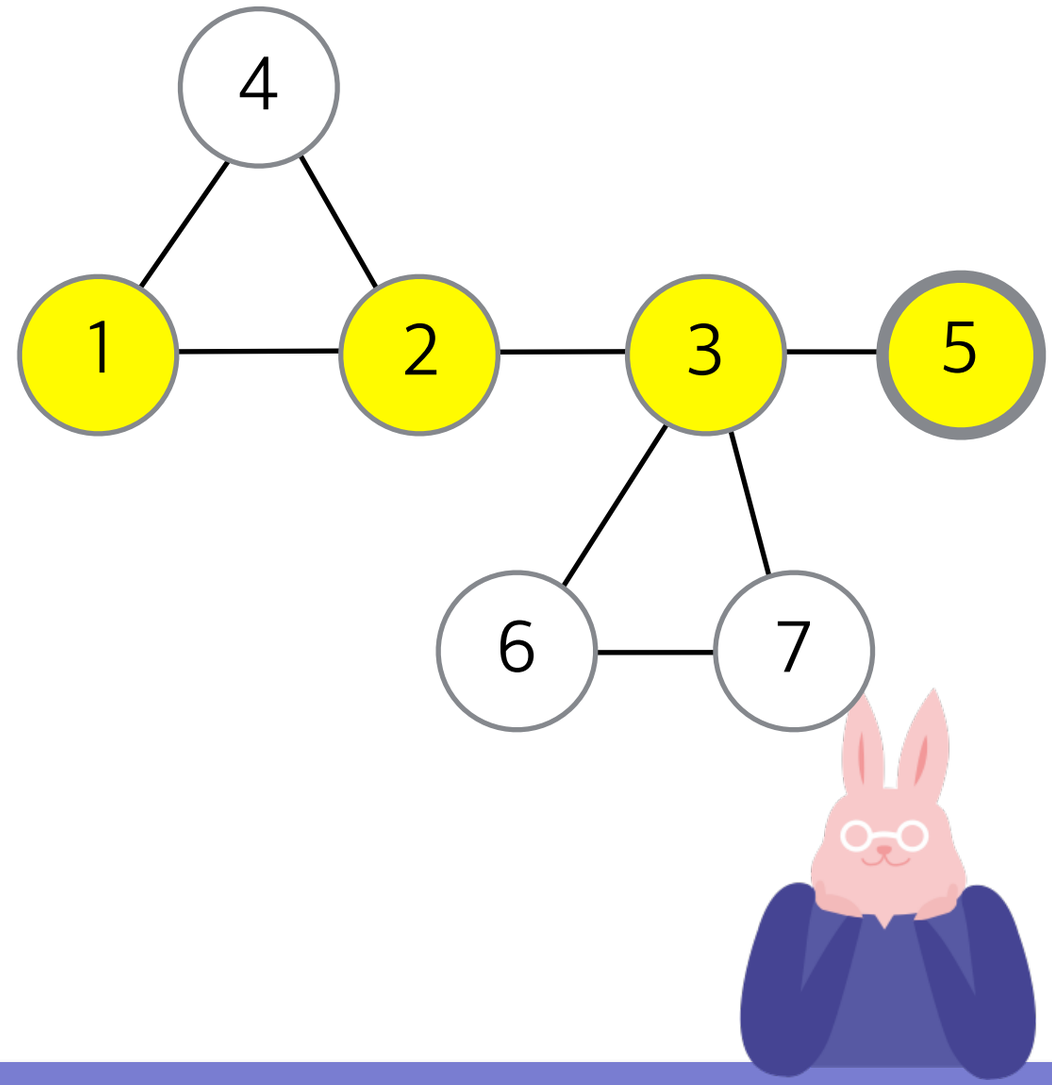
깊이 우선 탐색 (DFS)

- 인접한 Node 중에서 방문하지 않은 Node로 간다



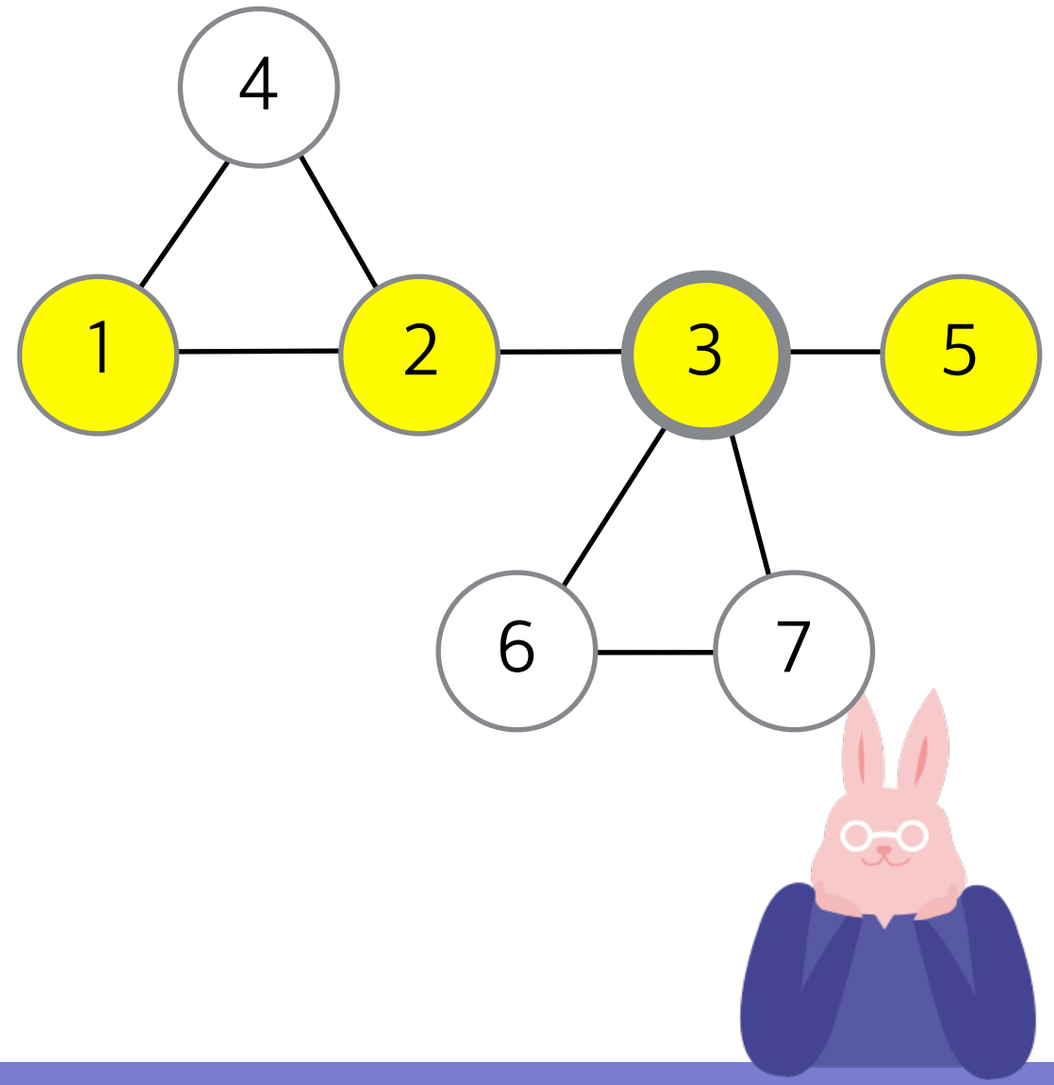
깊이 우선 탐색 (DFS)

- 인접한 Node 중에서 방문하지 않은 Node로 간다



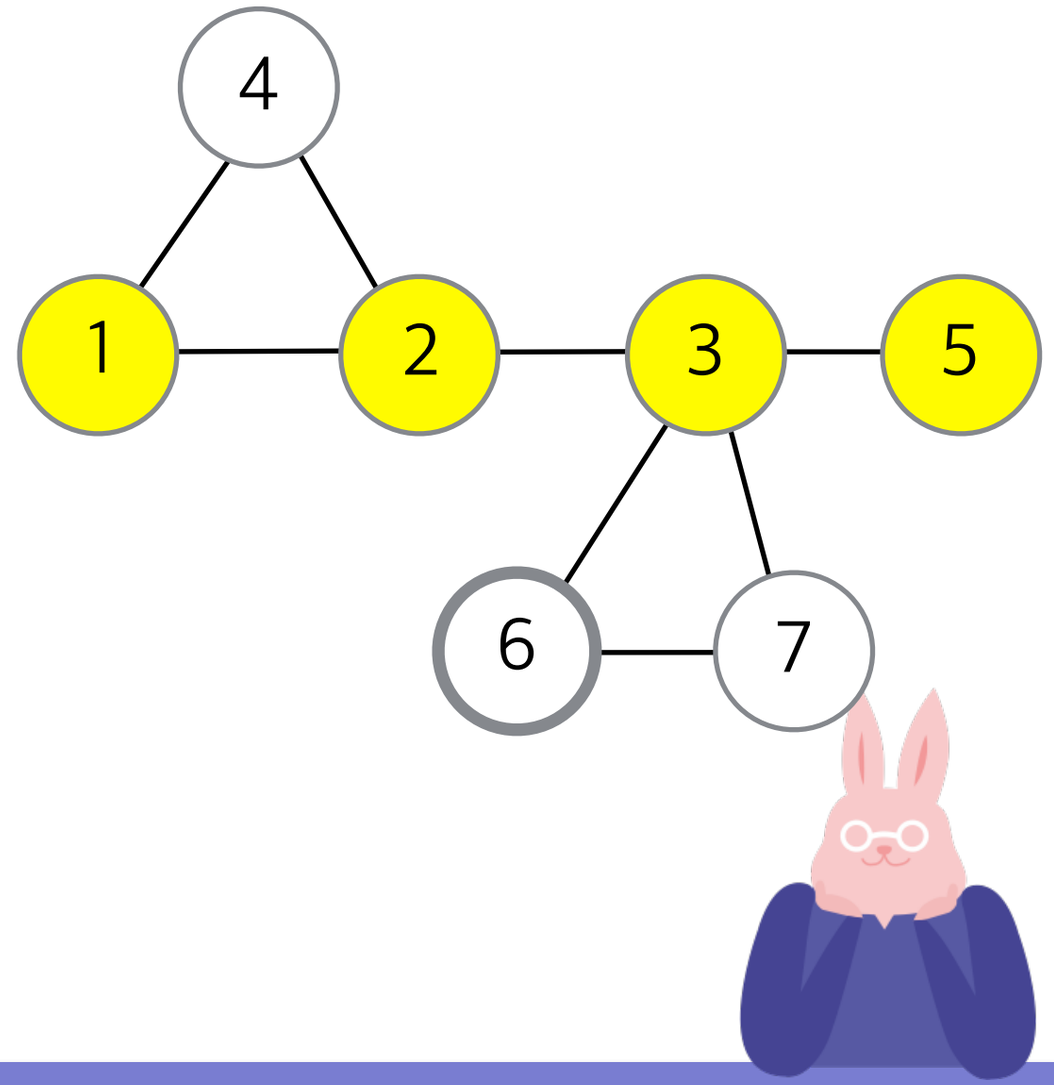
깊이 우선 탐색 (DFS)

- 인접한 Node 중에서 방문하지 않은 Node로 간다



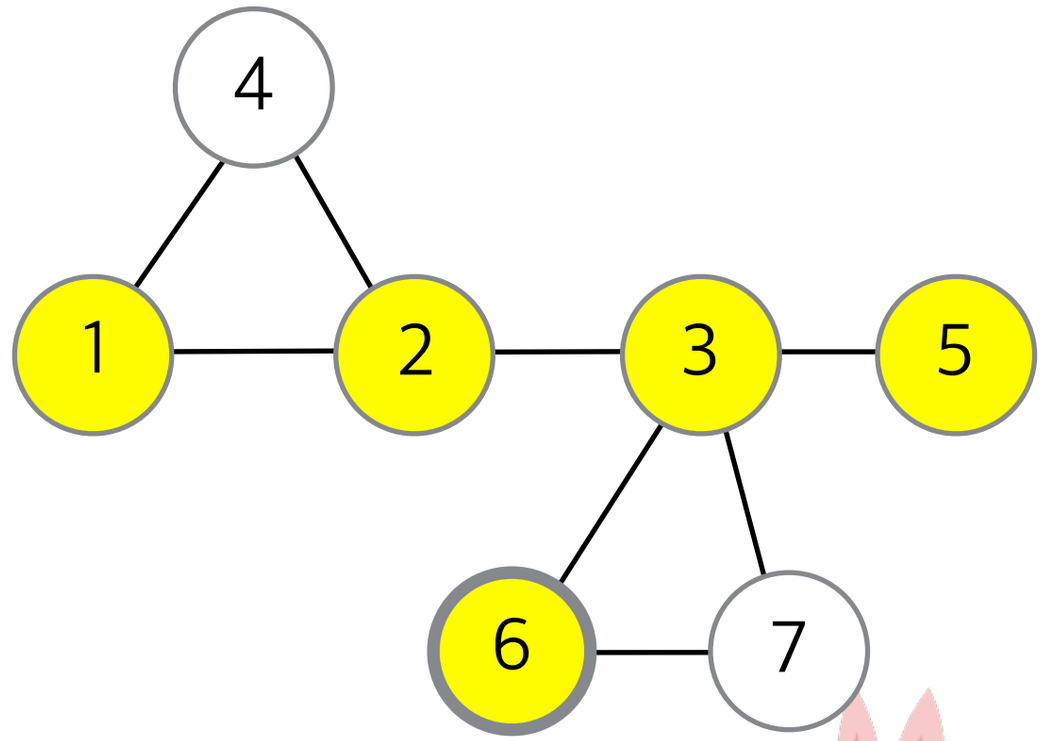
깊이 우선 탐색 (DFS)

- 인접한 Node 중에서 방문하지 않은 Node로 간다



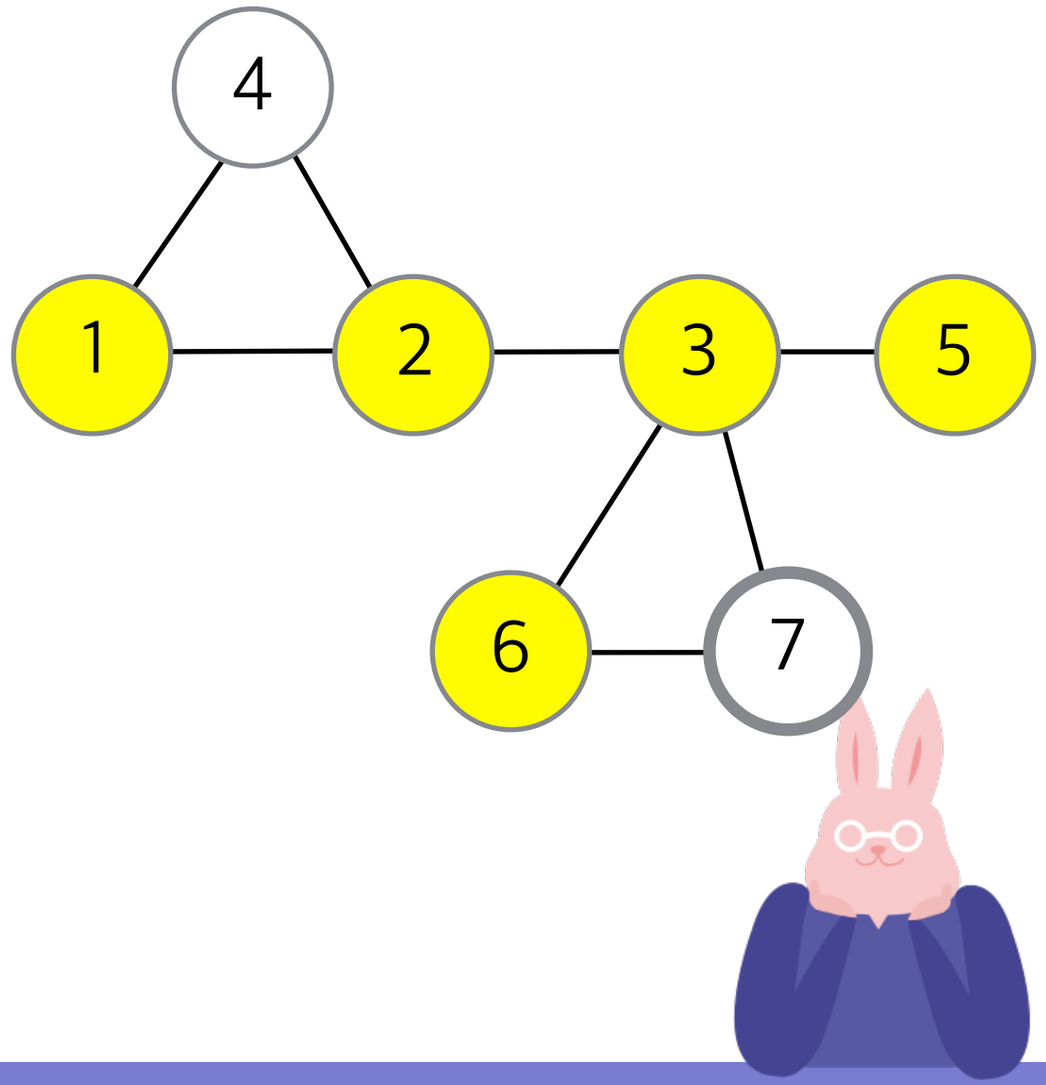
깊이 우선 탐색 (DFS)

- 인접한 Node 중에서 방문하지 않은 Node로 간다



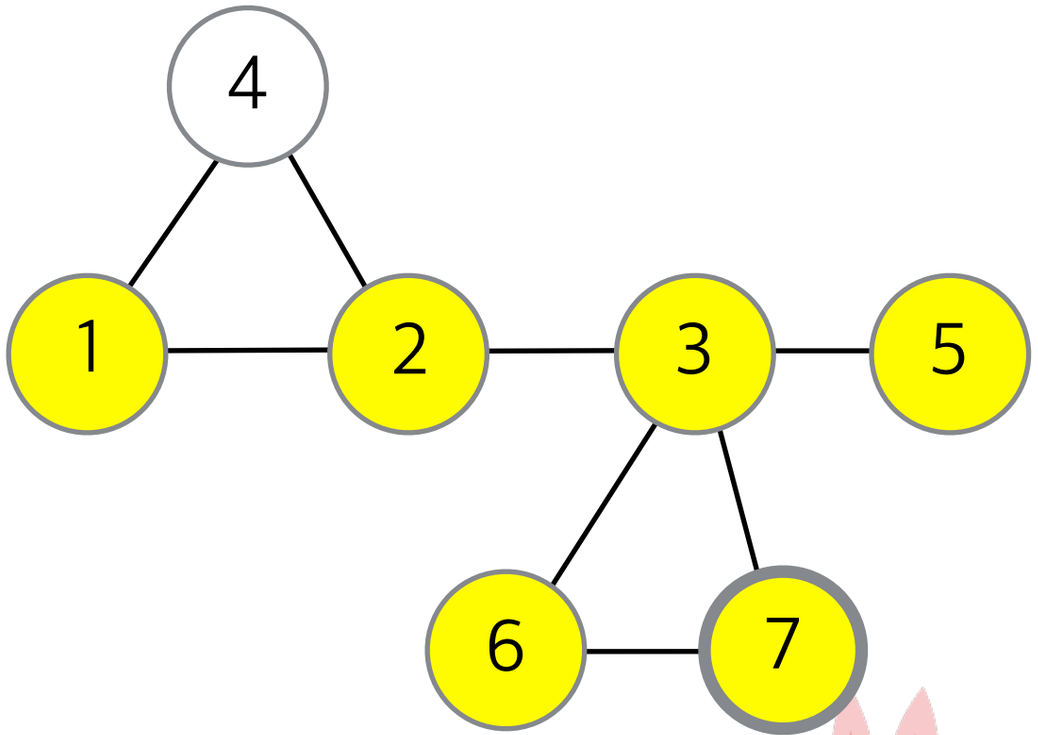
깊이 우선 탐색 (DFS)

- 인접한 Node 중에서 방문하지 않은 Node로 간다



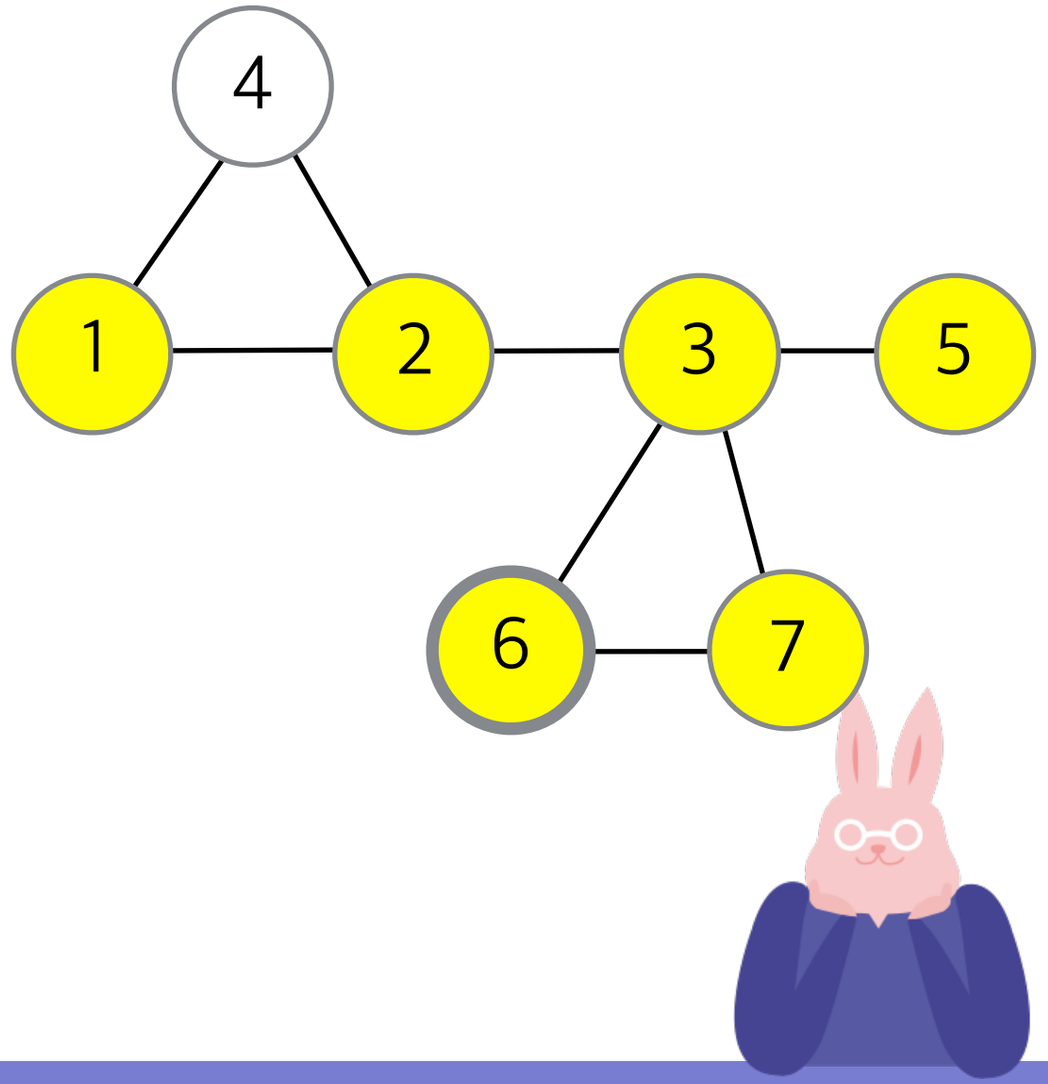
깊이 우선 탐색 (DFS)

- 인접한 Node 중에서 방문하지 않은 Node로 간다



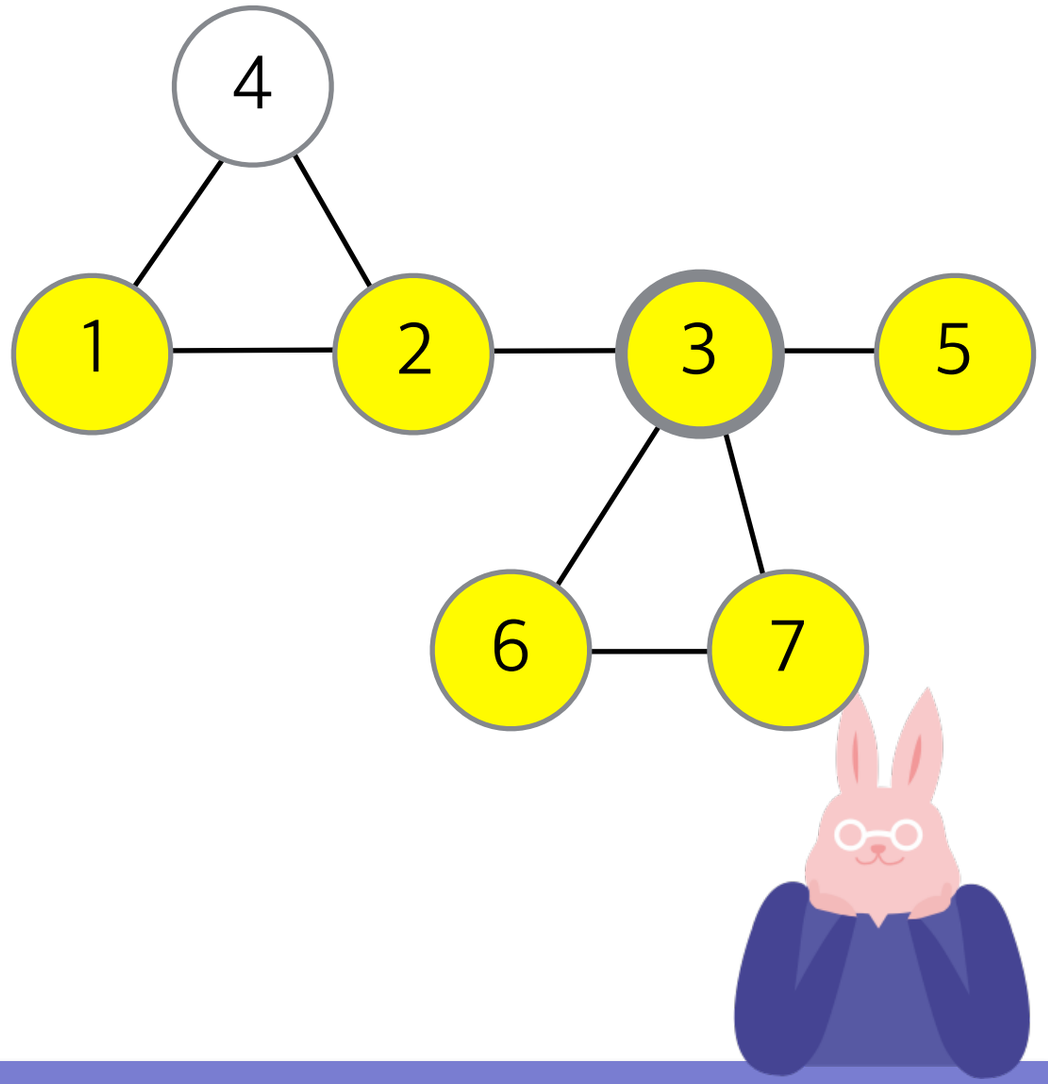
깊이 우선 탐색 (DFS)

- 인접한 Node 중에서 방문하지 않은 Node로 간다



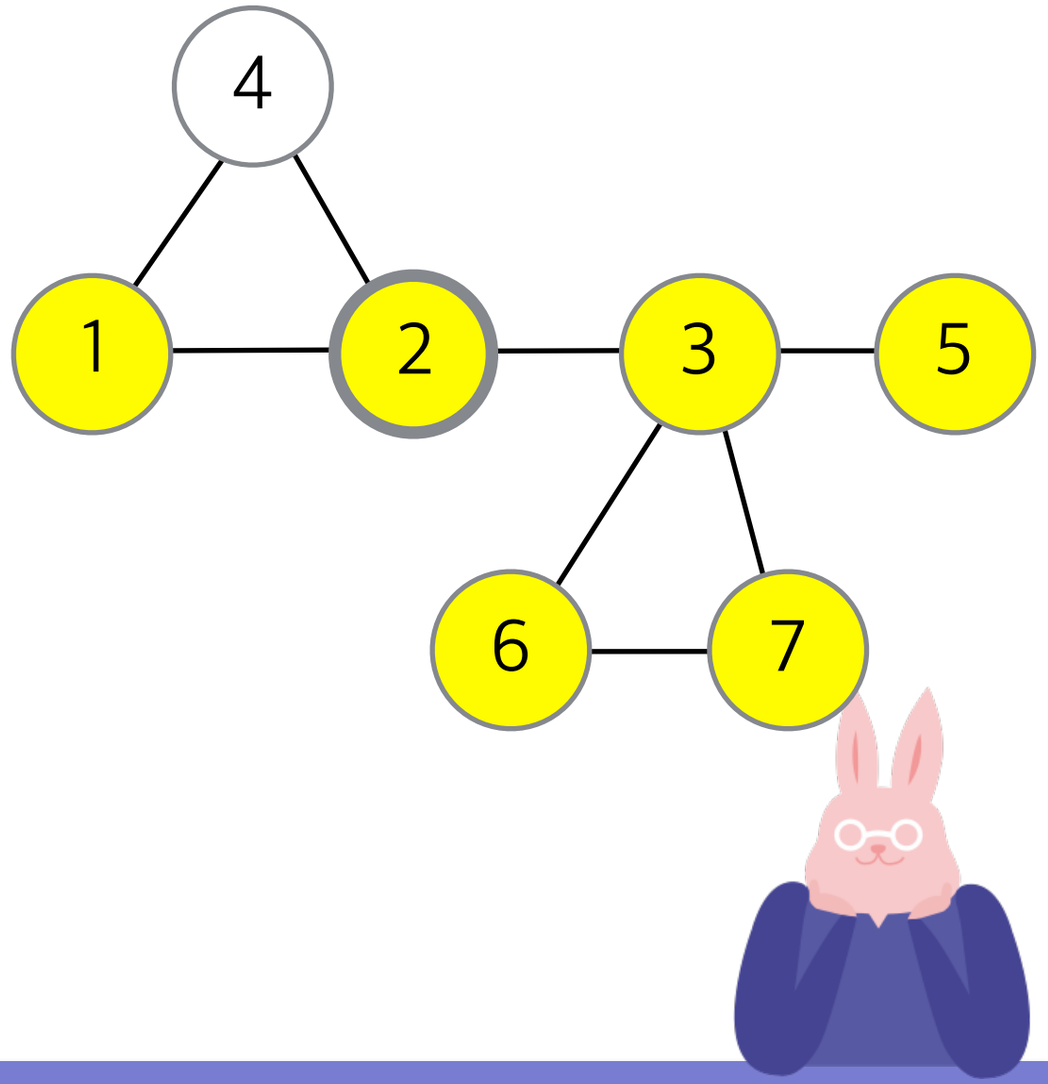
깊이 우선 탐색 (DFS)

- 인접한 Node 중에서 방문하지 않은 Node로 간다



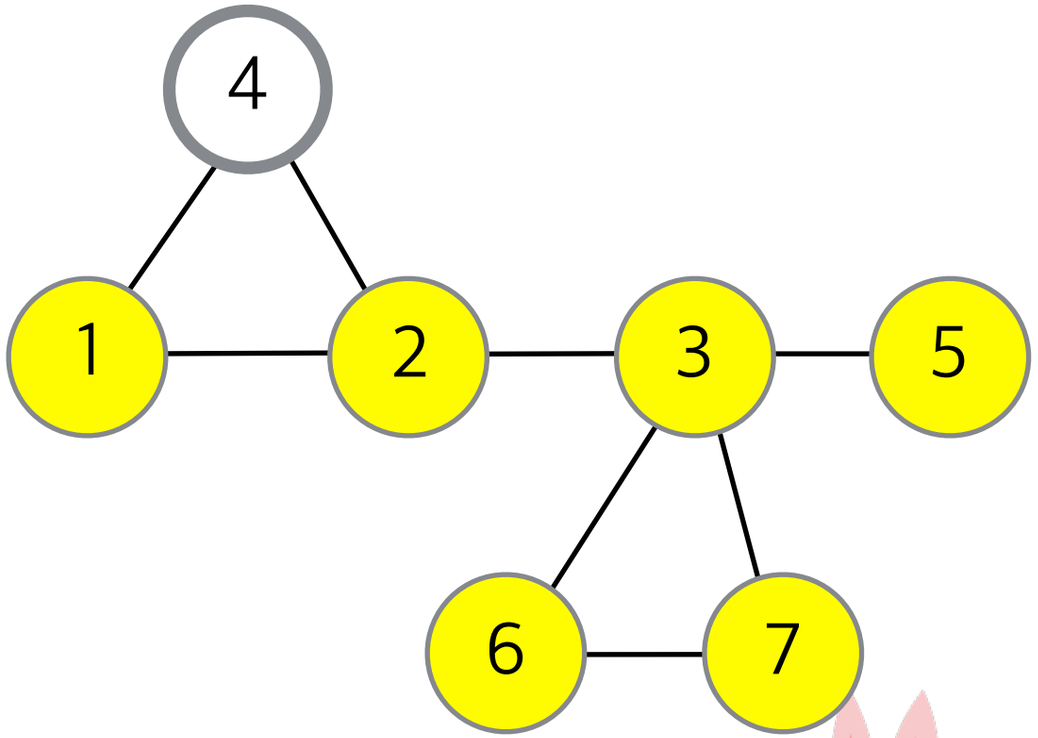
깊이 우선 탐색 (DFS)

- 인접한 Node 중에서 방문하지 않은 Node로 간다



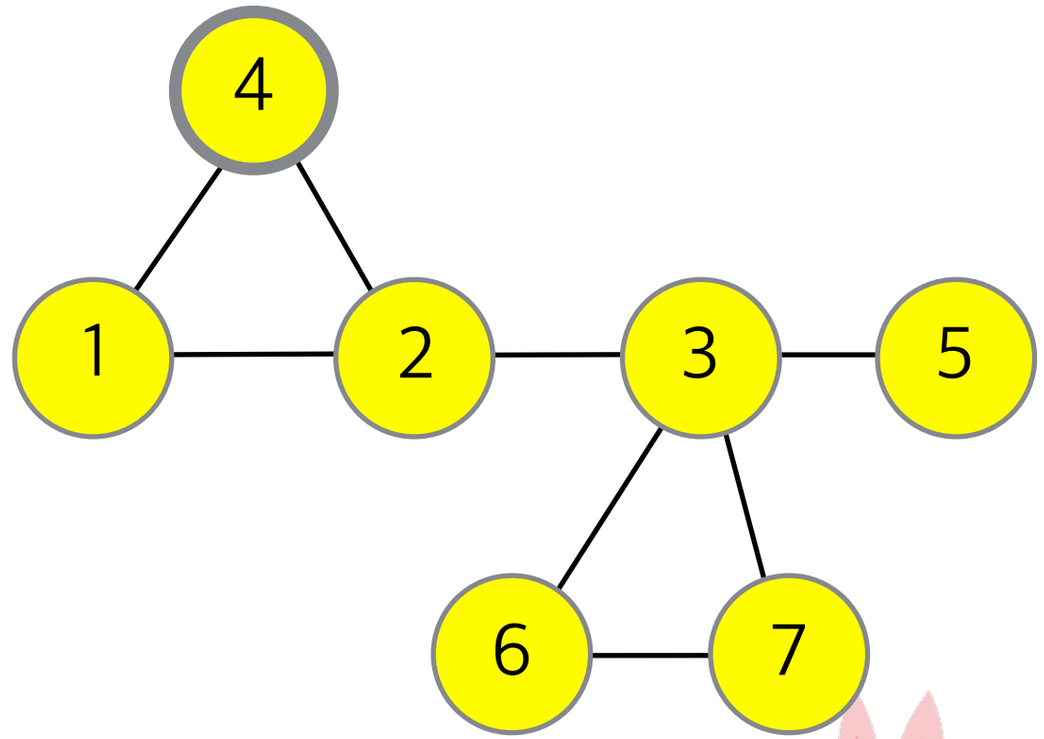
깊이 우선 탐색 (DFS)

- 인접한 Node 중에서 방문하지 않은 Node로 간다



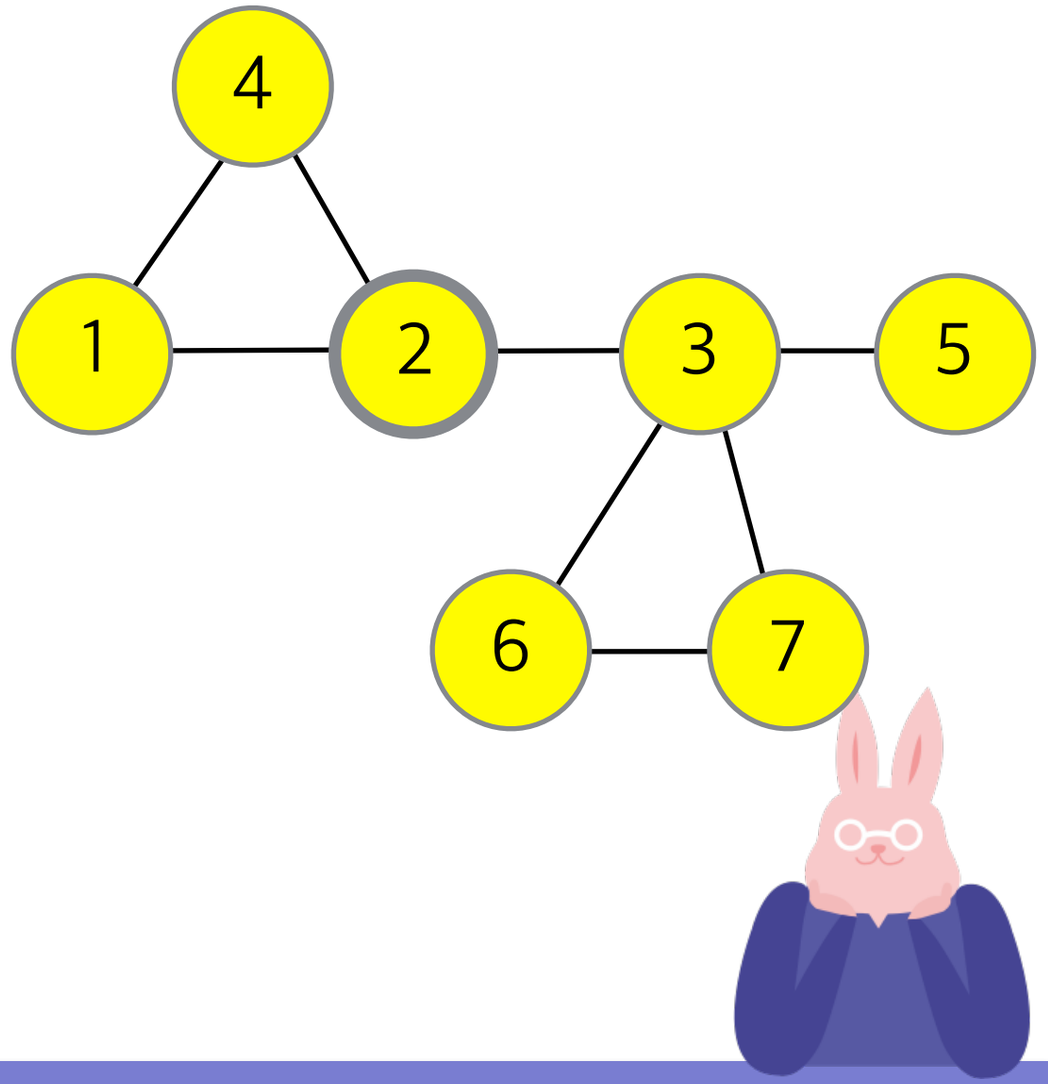
깊이 우선 탐색 (DFS)

- 인접한 Node 중에서 방문하지 않은 Node로 간다



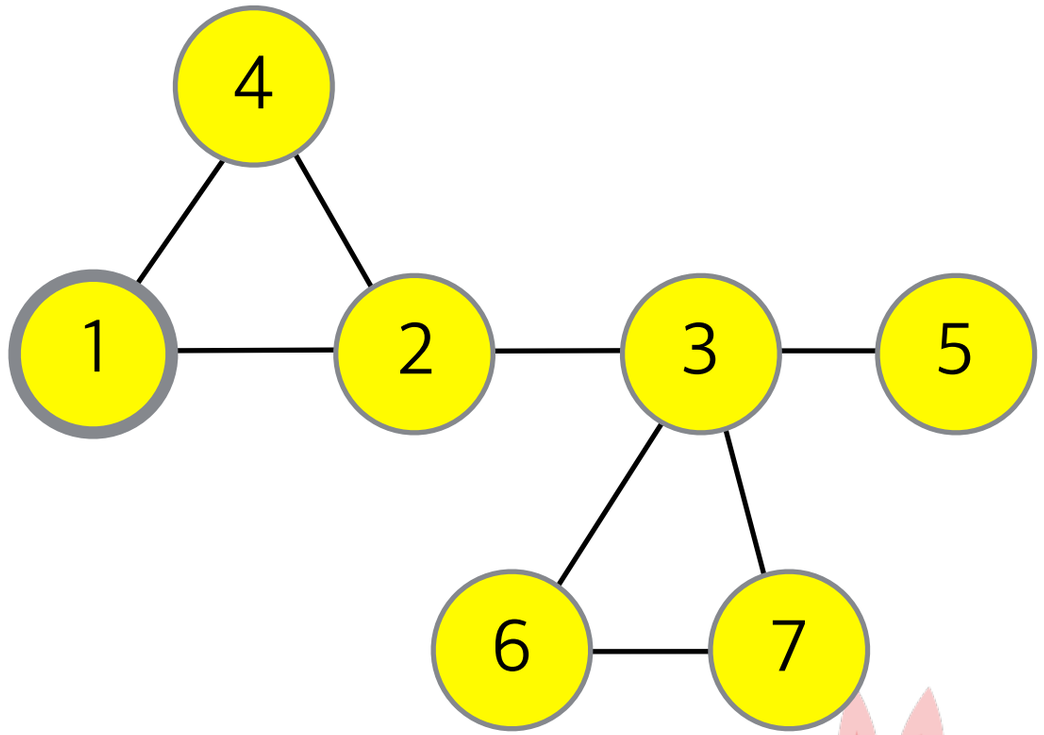
깊이 우선 탐색 (DFS)

- 인접한 Node 중에서 방문하지 않은 Node로 간다



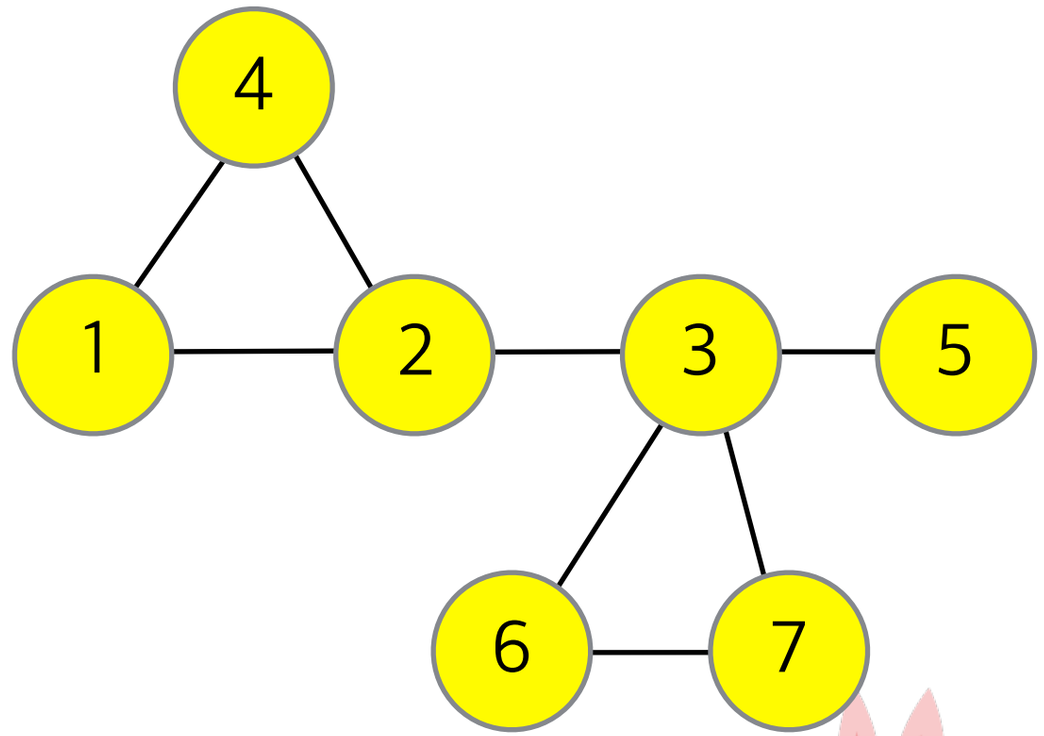
깊이 우선 탐색 (DFS)

- 인접한 Node 중에서 방문하지 않은 Node로 간다



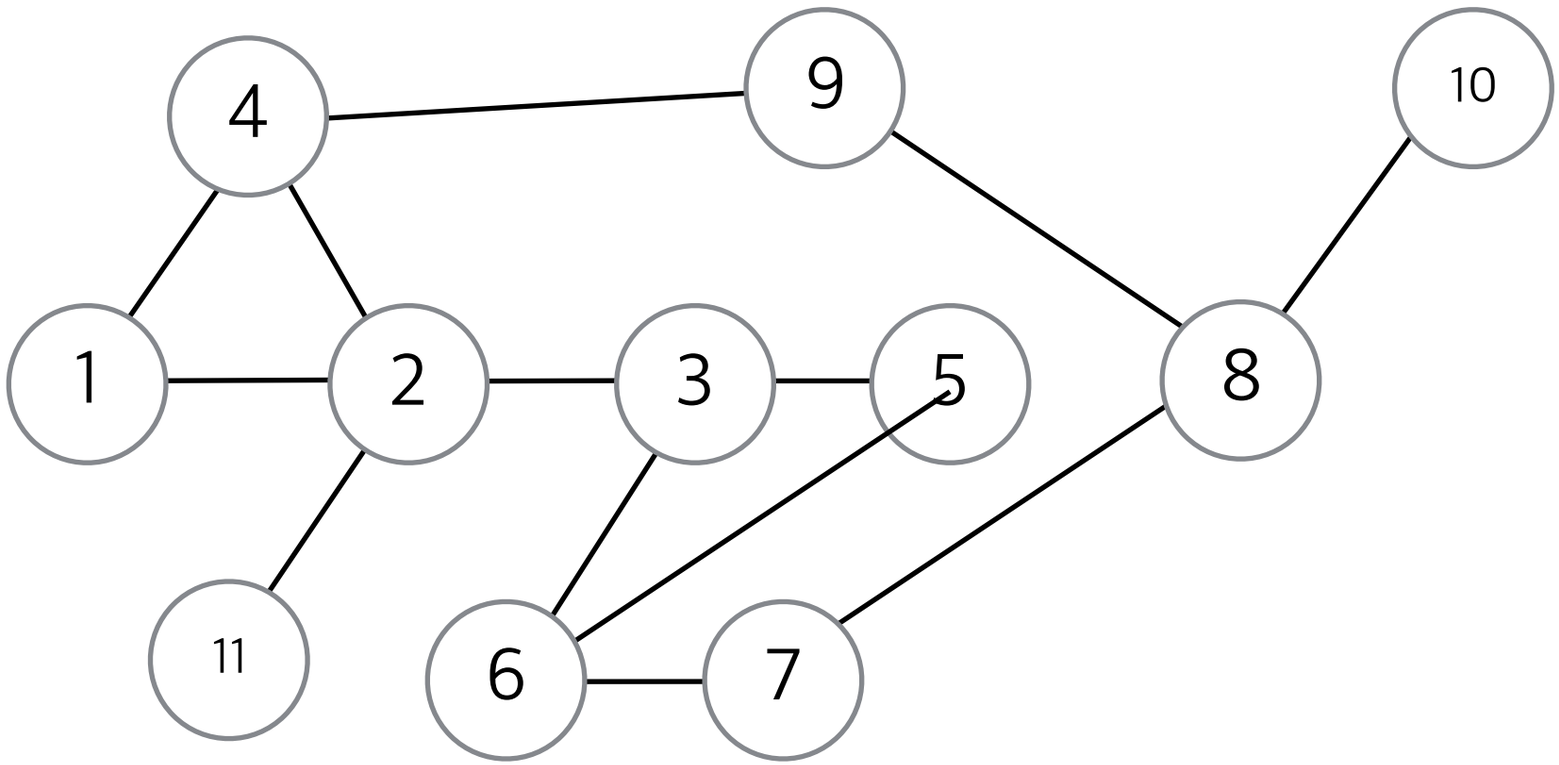
깊이 우선 탐색 (DFS)

- 인접한 Node 중에서 방문하지 않은 Node로 간다



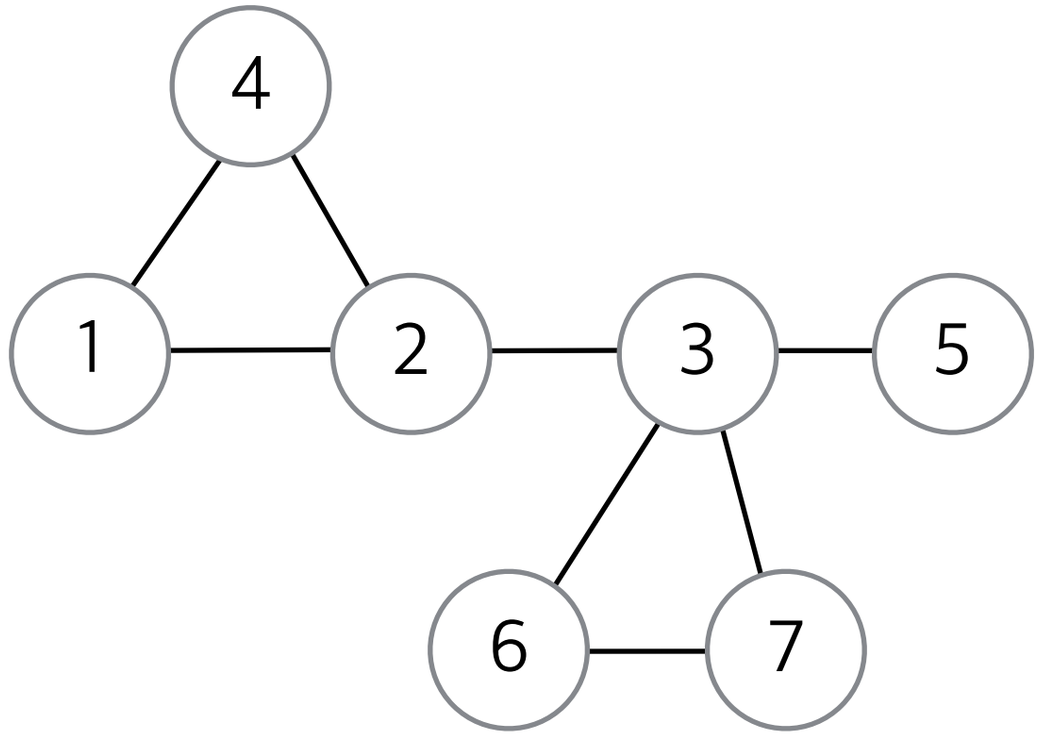
깊이 우선 탐색 (DFS)

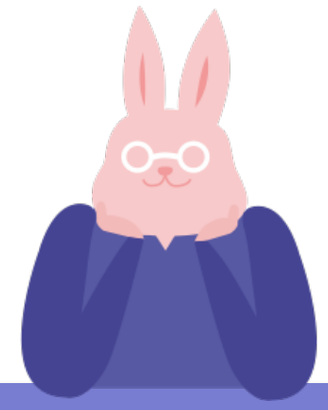
- 다음 그래프에 대하여 1을 시작으로 DFS한 결과는 ?
  - 단, 인접한 노드가 여러개일 경우 노드 번호가 작은 노드부터 방문



깊이 우선 탐색 (DFS) 구현

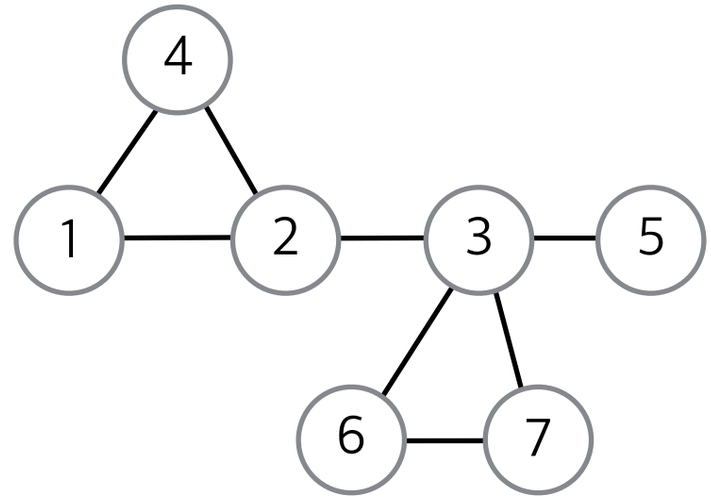
```
def DFS(graph, x, visited) :
    visited[x] = True
    result = [x]

    for v in graph[x] :
        if visited[v] == False :
            result = result +
                     DFS(graph, v, visited)

    return result
```





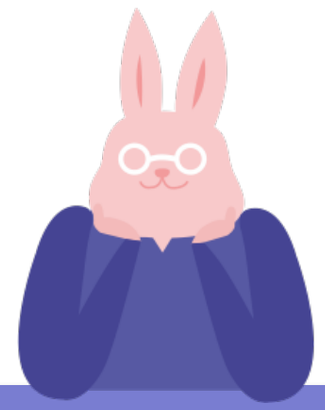
깊이 우선 탐색 (DFS) 구현

```
def DFS(graph, x, visited) :
      visited[x] = True
      result = [x]

      for v in graph[x] :
            if visited[v] == False :
                  result = result +
                           DFS(graph, v, visited)

      return result
```



visited= [F, T, F, F, F, F, F, F]

graph[1] = [2, 4]
graph[2] = [3, 4]
graph[3] =[2, 5, 6, 7]
graph[4] = [1, 2]
graph[5] = [3]
graph[6] = [3, 7]
graph[7] = [3, 6]

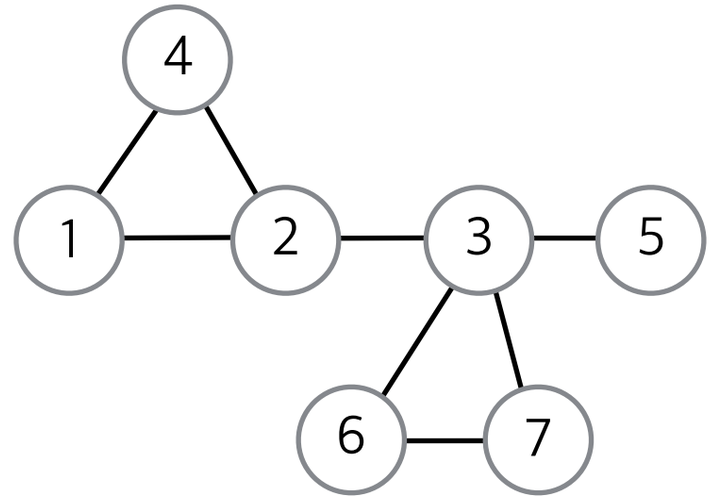


깊이 우선 탐색 (DFS) 구현

```
def DFS(graph, x, visited) :
    visited[x] = True
    result = [x]

    for v in graph[x] :
        if visited[v] == False :
            result = result +
                     DFS(graph, v, visited)

    return result
```

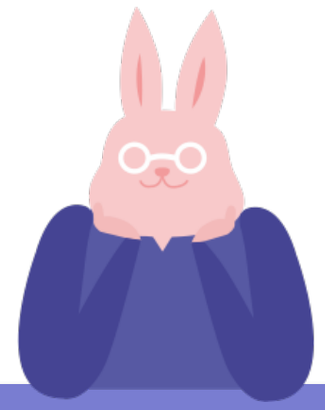
DFS(graph, 1, visited)

result = null



visited= [F, T, F, F, F, F, F, F]

graph[1] = [2, 4]
graph[2] = [3, 4]
graph[3] =[2, 5, 6, 7]
graph[4] = [1, 2]
graph[5] = [3]
graph[6] = [3, 7]
graph[7] = [3, 6]

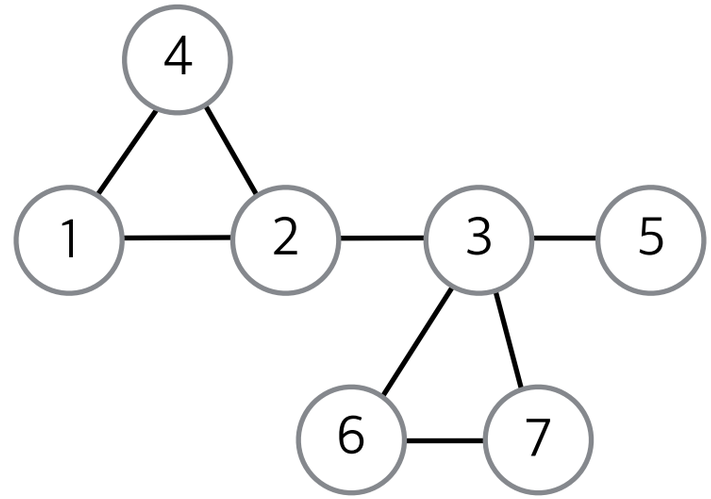


깊이 우선 탐색 (DFS) 구현

```
def DFS(graph, x, visited) :
    visited[x] = True
    result = [x]

    for v in graph[x] :
        if visited[v] == False :
            result = result +
                     DFS(graph, v, visited)

    return result
```

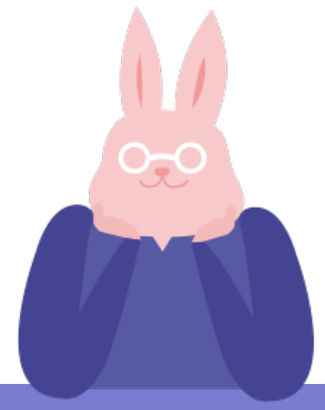
DFS(graph, 1, visited)

result = null



visited= [F, T, F, F, F, F, F, F]

graph[1] = [2, 4]
graph[2] = [3, 4]
graph[3] =[2, 5, 6, 7]
graph[4] = [1, 2]
graph[5] = [3]
graph[6] = [3, 7]
graph[7] = [3, 6]

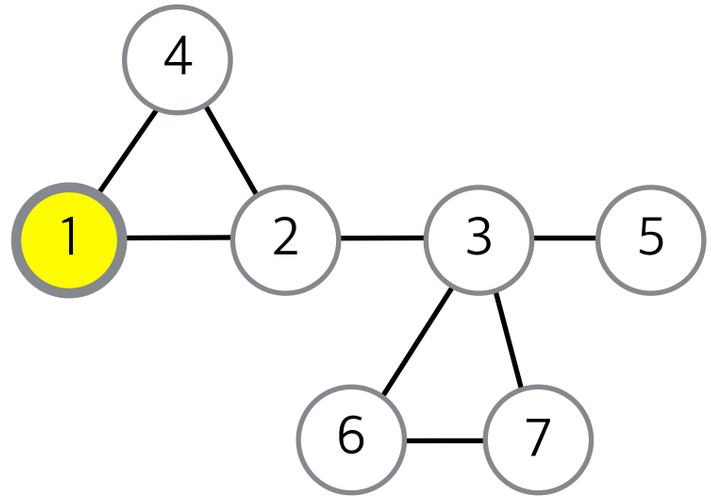


깊이 우선 탐색 (DFS) 구현

```
def DFS(graph, x, visited) :
    visited[x] = True
    result = [x]

    for v in graph[x] :
        if visited[v] == False :
            result = result +
                     DFS(graph, v, visited)

    return result
```

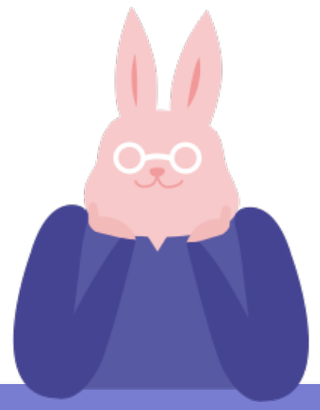
DFS(graph, 1, visited)

result = null



visited= [F, T, F, F, F, F, F, F]

graph[1] = [2, 4]
graph[2] = [3, 4]
graph[3] =[2, 5, 6, 7]
graph[4] = [1, 2]
graph[5] = [3]
graph[6] = [3, 7]
graph[7] = [3, 6]

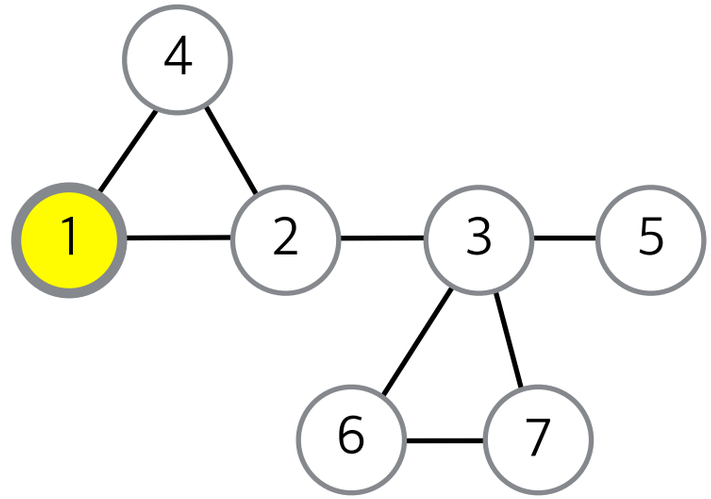


깊이 우선 탐색 (DFS) 구현

```
def DFS(graph, x, visited) :
    visited[x] = True
    result = [x]

    for v in graph[x] :
        if visited[v] == False :
            result = result +
                     DFS(graph, v, visited)

    return result
```

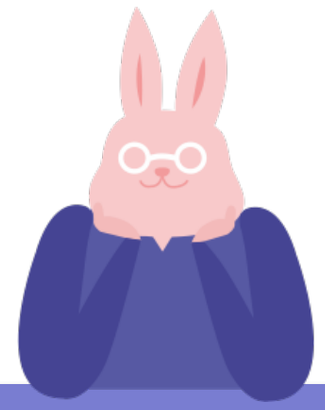
DFS(graph, 1, visited)

result = [1]



visited= [F, T, F, F, F, F, F, F]

graph[1] = [2, 4]
graph[2] = [3, 4]
graph[3] =[2, 5, 6, 7]
graph[4] = [1, 2]
graph[5] = [3]
graph[6] = [3, 7]
graph[7] = [3, 6]

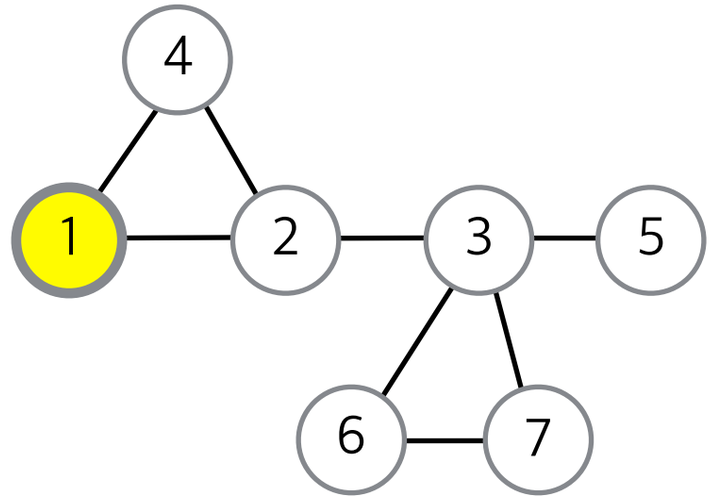


깊이 우선 탐색 (DFS) 구현

```
def DFS(graph, x, visited) :
    visited[x] = True
    result = [x]

    for v in graph[x] :
        if visited[v] == False :
            result = result +
                     DFS(graph, v, visited)

    return result
```

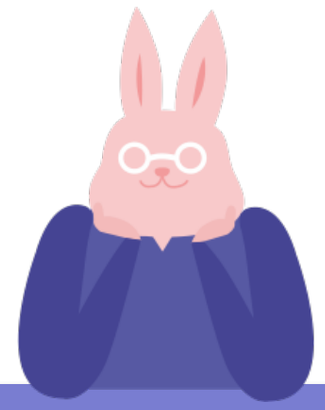
DFS(graph, 1, visited)

result = [1], v = 2



visited= [F, T, F, F, F, F, F, F]

graph[1] = [2, 4]
graph[2] = [3, 4]
graph[3] =[2, 5, 6, 7]
graph[4] = [1, 2]
graph[5] = [3]
graph[6] = [3, 7]
graph[7] = [3, 6]

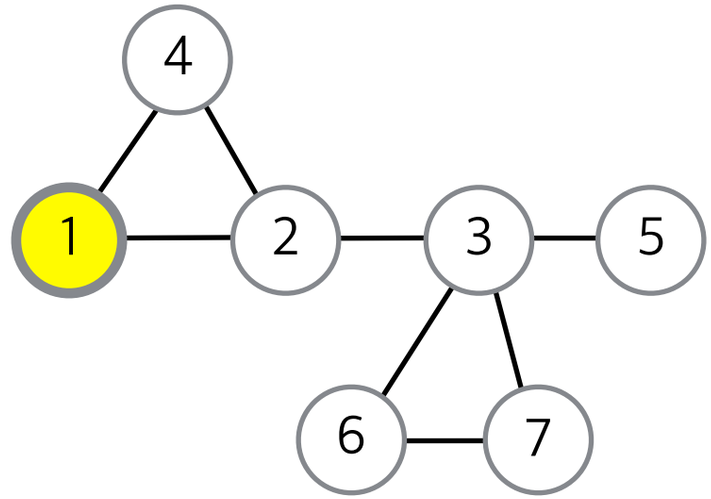


깊이 우선 탐색 (DFS) 구현

```
def DFS(graph, x, visited) :
    visited[x] = True
    result = [x]

    for v in graph[x] :
        if visited[v] == False :
            result = result +
                     DFS(graph, v, visited)

    return result
```

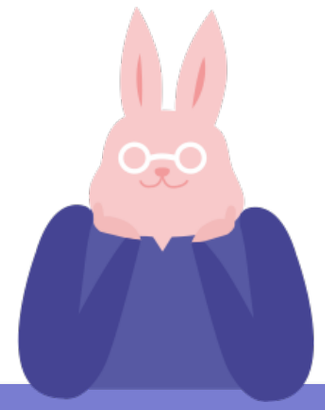
DFS(graph, 1, visited)

result = [1], v = 2



visited= [F, T, F, F, F, F, F, F]

graph[1] = [2, 4]
graph[2] = [3, 4]
graph[3] =[2, 5, 6, 7]
graph[4] = [1, 2]
graph[5] = [3]
graph[6] = [3, 7]
graph[7] = [3, 6]



깊이 우선 탐색 (DFS) 구현

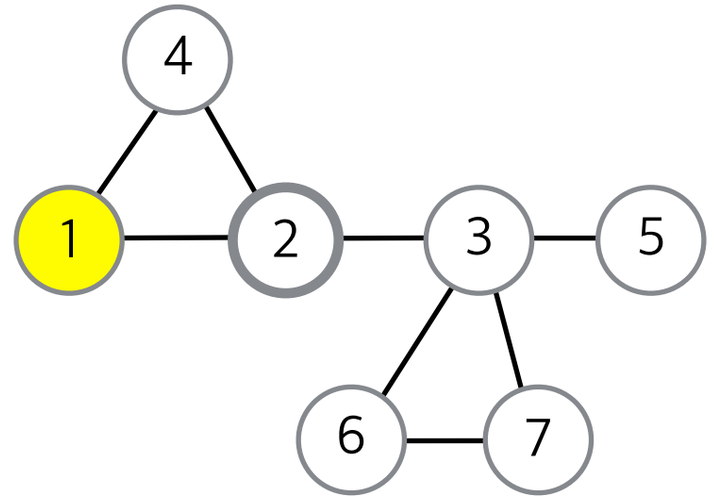
```
def DFS(graph, x, visited) :
    visited[x] = True
    result = [x]

    for v in graph[x] :
        if visited[v] == False :
            result = result +
                     DFS(graph, v, visited)

    return result
```

DFS(graph, 1, visited)

result = [1], v = 2

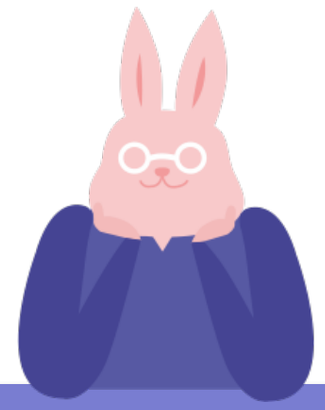
DFS(graph, 2, visited)

result = null



visited= [F, T, F, F, F, F, F, F]

graph[1] = [2, 4]
graph[2] = [3, 4]
graph[3] =[2, 5, 6, 7]
graph[4] = [1, 2]
graph[5] = [3]
graph[6] = [3, 7]
graph[7] = [3, 6]



깊이 우선 탐색 (DFS) 구현

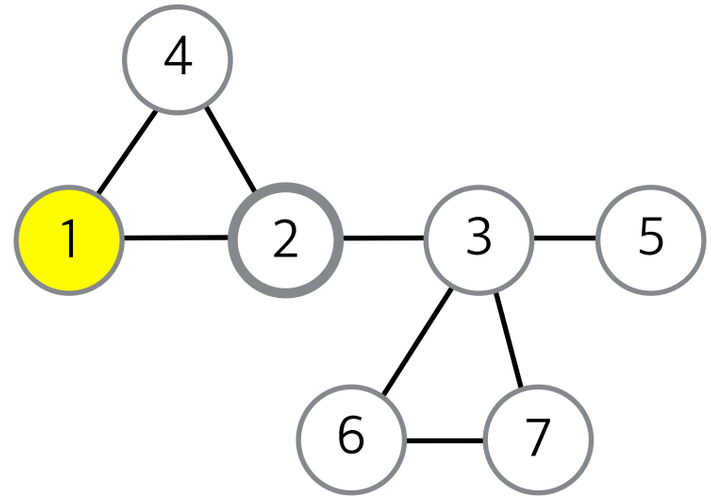
```
def DFS(graph, x, visited) :
    visited[x] = True
    result = [x]

    for v in graph[x] :
        if visited[v] == False :
            result = result +
                     DFS(graph, v, visited)

    return result
```

DFS(graph, 1, visited)

result = [1], v = 2

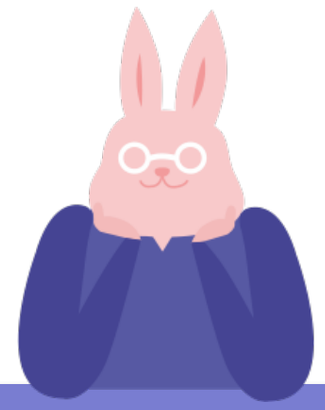
DFS(graph, 2, visited)

result = null



visited= [F, T, F, F, F, F, F, F]

graph[1] = [2, 4]
graph[2] = [3, 4]
graph[3] =[2, 5, 6, 7]
graph[4] = [1, 2]
graph[5] = [3]
graph[6] = [3, 7]
graph[7] = [3, 6]



깊이 우선 탐색 (DFS) 구현

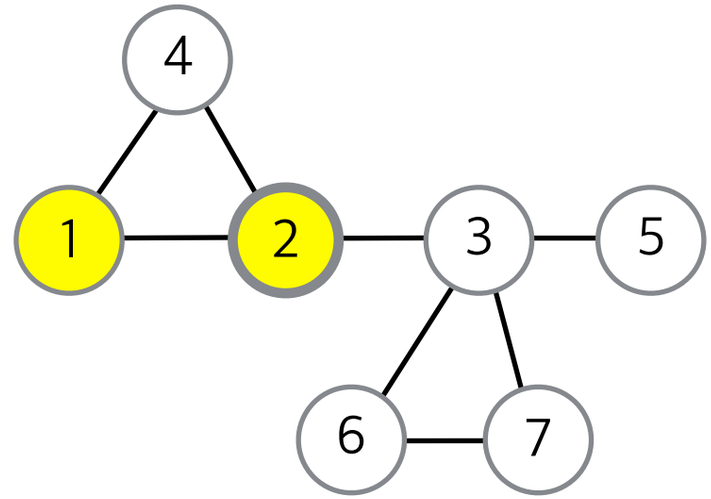
```
def DFS(graph, x, visited) :
    visited[x] = True
    result = [x]

    for v in graph[x] :
        if visited[v] == False :
            result = result +
                     DFS(graph, v, visited)

    return result
```

DFS(graph, 1, visited)

result = [1], v = 2

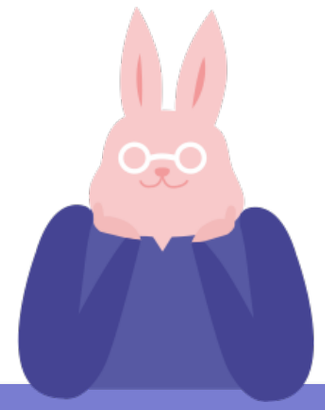
DFS(graph, 2, visited)

result = null



visited= [F, T, **T**, F, F, F, F, F]

graph[1] = [2, 4]
graph[2] = [3, 4]
graph[3] =[2, 5, 6, 7]
graph[4] = [1, 2]
graph[5] = [3]
graph[6] = [3, 7]
graph[7] = [3, 6]



깊이 우선 탐색 (DFS) 구현

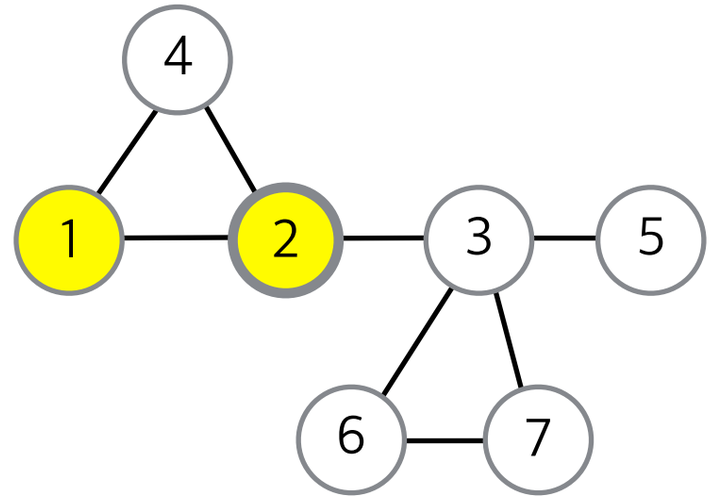
```
def DFS(graph, x, visited) :
    visited[x] = True
    result = [x]

    for v in graph[x] :
        if visited[v] == False :
            result = result +
                     DFS(graph, v, visited)

    return result
```

DFS(graph, 1, visited)

result = [1], v = 2

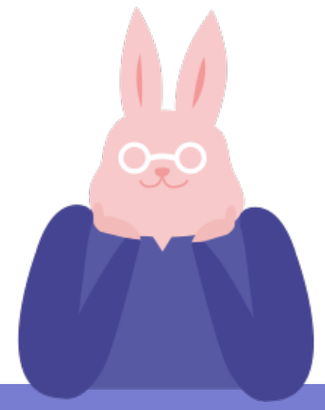
DFS(graph, 2, visited)

result = [2]



visited= [F, T, T, F, F, F, F, F]

graph[1] = [2, 4]
graph[2] = [3, 4]
graph[3] =[2, 5, 6, 7]
graph[4] = [1, 2]
graph[5] = [3]
graph[6] = [3, 7]
graph[7] = [3, 6]



깊이 우선 탐색 (DFS) 구현

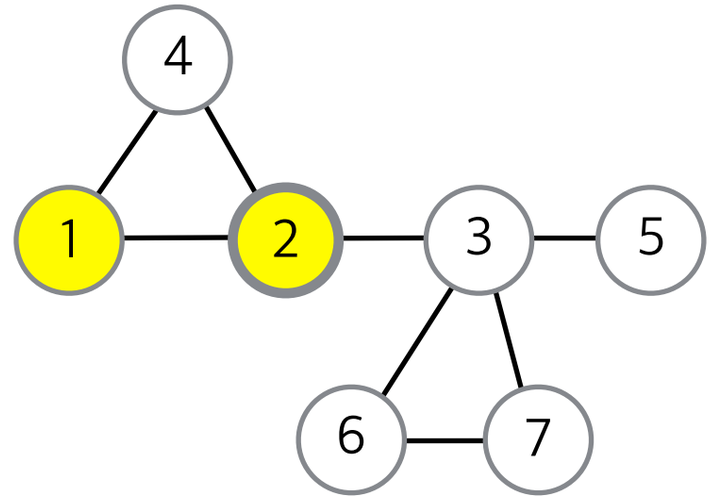
```
def DFS(graph, x, visited) :
    visited[x] = True
    result = [x]

    for v in graph[x] :
        if visited[v] == False :
            result = result +
                     DFS(graph, v, visited)

    return result
```

DFS(graph, 1, visited)

result = [1], v = 2

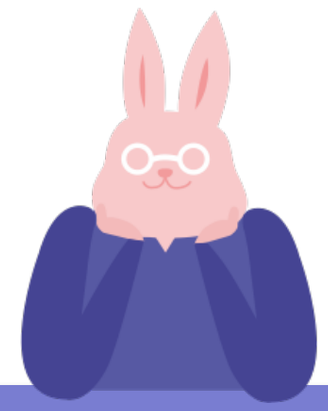
DFS(graph, 2, visited)

result = [2], v = 3



visited= [F, T, T, F, F, F, F, F]

graph[1] = [2, 4]
graph[2] = [3, 4]
graph[3] =[2, 5, 6, 7]
graph[4] = [1, 2]
graph[5] = [3]
graph[6] = [3, 7]
graph[7] = [3, 6]



깊이 우선 탐색 (DFS) 구현

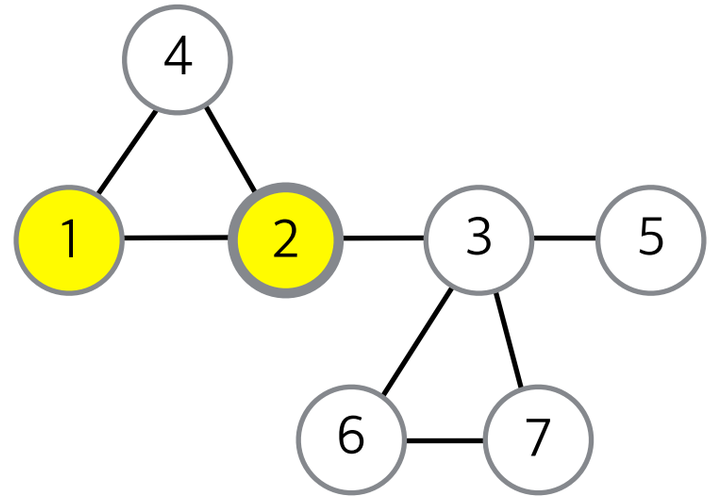
```
def DFS(graph, x, visited) :
      visited[x] = True
      result = [x]

      for v in graph[x] :
            if visited[v] == False :
                  result = result +
                           DFS(graph, v, visited)

      return result
```

DFS(graph, 1, visited)

result = [1], v = 2

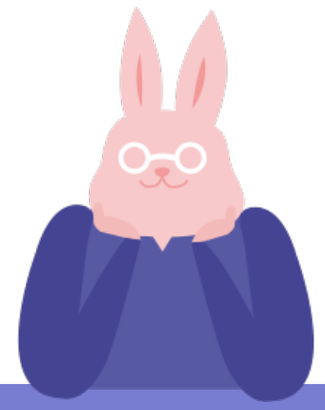
DFS(graph, 2, visited)

result = [2], v = 3



visited= [F, T, T, F, F, F, F, F]

graph[1] = [2, 4]
graph[2] = [3, 4]
graph[3] =[2, 5, 6, 7]
graph[4] = [1, 2]
graph[5] = [3]
graph[6] = [3, 7]
graph[7] = [3, 6]



깊이 우선 탐색 (DFS) 구현

```
def DFS(graph, x, visited) :
    visited[x] = True
    result = [x]

    for v in graph[x] :
        if visited[v] == False :
            result = result +
                     DFS(graph, v, visited)

    return result
```

DFS(graph, 1, visited)

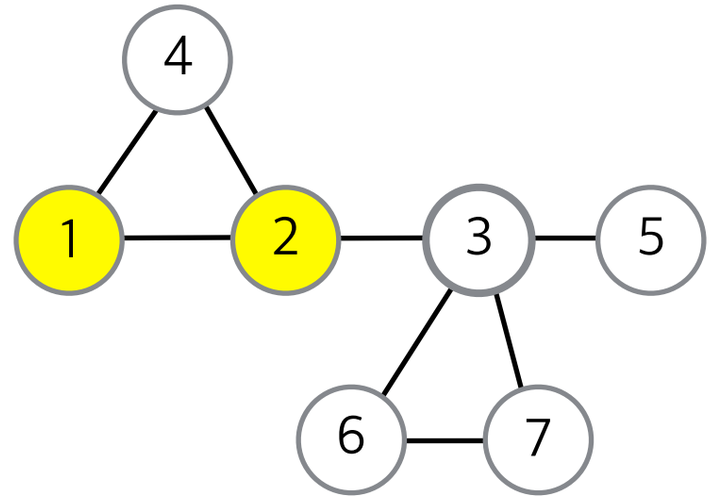
result = [1], v = 2

DFS(graph, 2, visited)

result = [2], v = 3

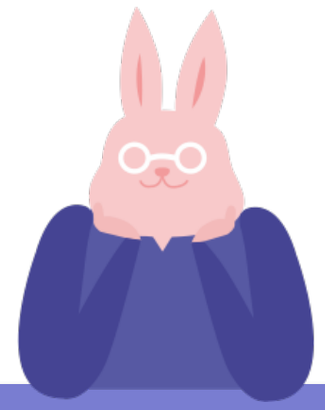
DFS(graph, 3, visited)

result = null



visited= [F, T, T, F, F, F, F, F]

graph[1] = [2, 4]
graph[2] = [3, 4]
graph[3] =[2, 5, 6, 7]
graph[4] = [1, 2]
graph[5] = [3]
graph[6] = [3, 7]
graph[7] = [3, 6]



깊이 우선 탐색 (DFS) 구현

```
def DFS(graph, x, visited) :
    visited[x] = True
    result = [x]

    for v in graph[x] :
        if visited[v] == False :
            result = result +
                     DFS(graph, v, visited)

    return result
```

DFS(graph, 1, visited)

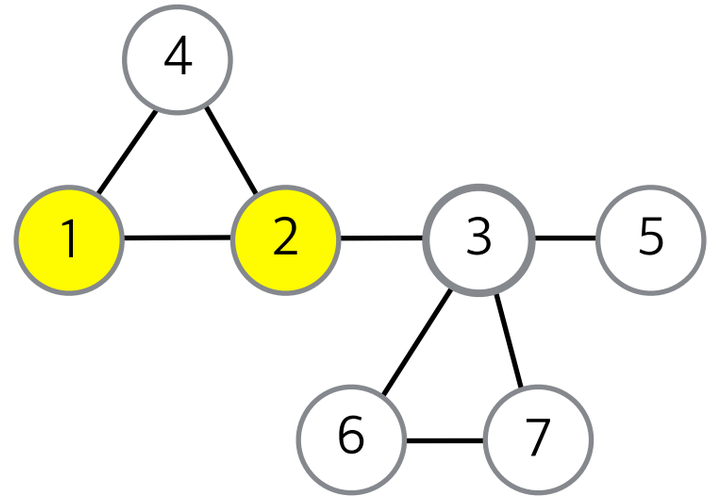
result = [1], v = 2

DFS(graph, 2, visited)

result = [2], v = 3

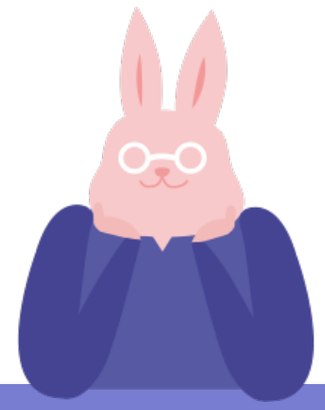
DFS(graph, 3, visited)

result = null



visited= [F, T, T, F, F, F, F, F]

graph[1] = [2, 4]
graph[2] = [3, 4]
graph[3] =[2, 5, 6, 7]
graph[4] = [1, 2]
graph[5] = [3]
graph[6] = [3, 7]
graph[7] = [3, 6]



깊이 우선 탐색 (DFS) 구현

```
def DFS(graph, x, visited) :
    visited[x] = True
    result = [x]

    for v in graph[x] :
        if visited[v] == False :
            result = result +
                     DFS(graph, v, visited)

    return result
```

DFS(graph, 1, visited)

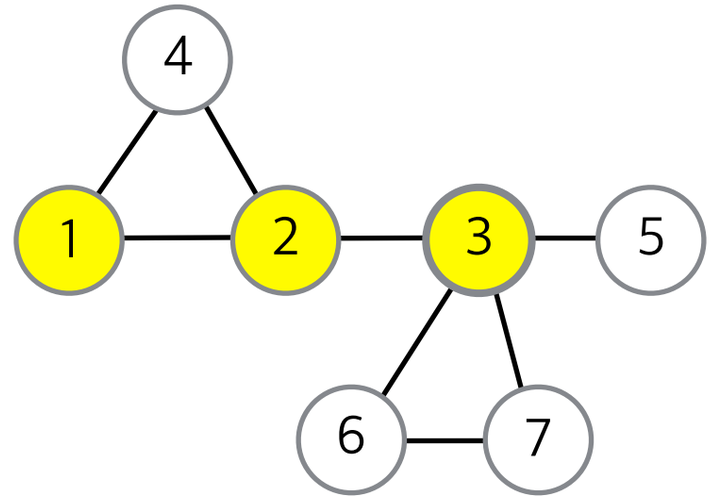
result = [1], v = 2

DFS(graph, 2, visited)

result = [2], v = 3

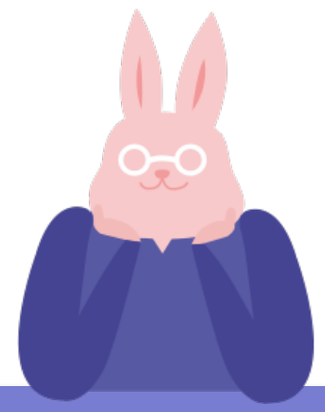
DFS(graph, 3, visited)

result = null



visited= [F, T, T, T, F, F, F, F]

graph[1] = [2, 4]
graph[2] = [3, 4]
graph[3] =[2, 5, 6, 7]
graph[4] = [1, 2]
graph[5] = [3]
graph[6] = [3, 7]
graph[7] = [3, 6]



깊이 우선 탐색 (DFS) 구현

```
def DFS(graph, x, visited) :
    visited[x] = True
    result = [x]

    for v in graph[x] :
        if visited[v] == False :
            result = result +
                     DFS(graph, v, visited)

    return result
```

DFS(graph, 1, visited)

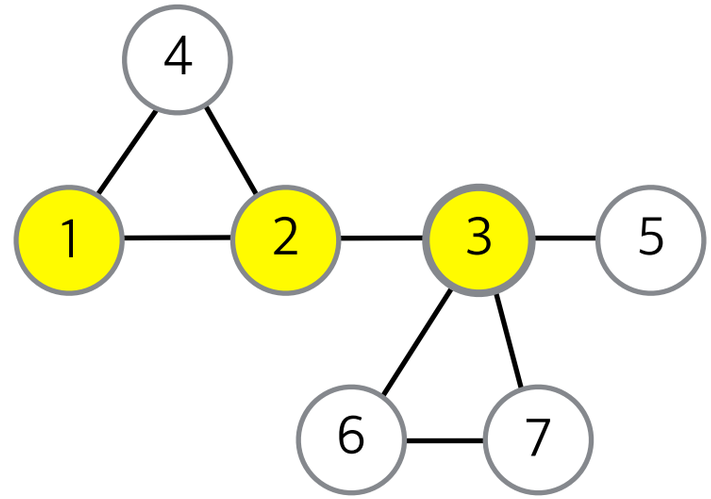
result = [1], v = 2

DFS(graph, 2, visited)

result = [2], v = 3

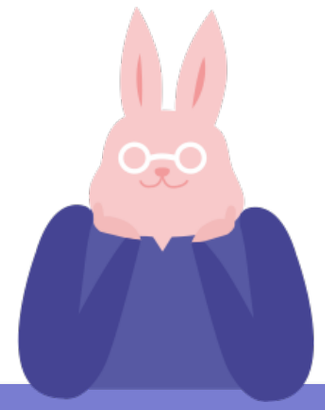
DFS(graph, 3, visited)

result = [3]



visited= [F, T, T, T, F, F, F, F]

graph[1] = [2, 4]
graph[2] = [3, 4]
graph[3] =[2, 5, 6, 7]
graph[4] = [1, 2]
graph[5] = [3]
graph[6] = [3, 7]
graph[7] = [3, 6]



깊이 우선 탐색 (DFS) 구현

```
def DFS(graph, x, visited) :
    visited[x] = True
    result = [x]

    for v in graph[x] :
        if visited[v] == False :
            result = result +
                     DFS(graph, v, visited)

    return result
```

DFS(graph, 1, visited)

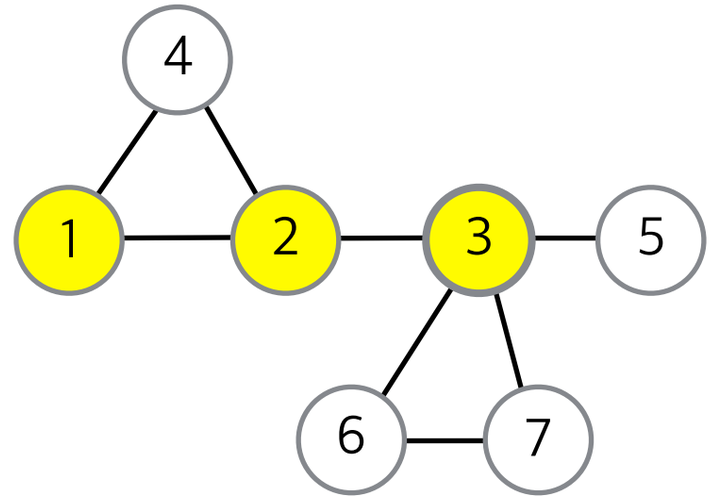
result = [1], v = 2

DFS(graph, 2, visited)

result = [2], v = 3

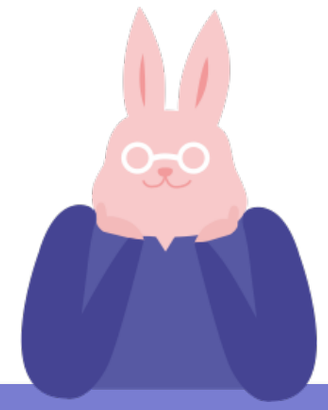
DFS(graph, 3, visited)

result = [3], v = 2



visited= [F, T, T, T, F, F, F, F]

graph[1] = [2, 4]
graph[2] = [3, 4]
graph[3] =[2, 5, 6, 7]
graph[4] = [1, 2]
graph[5] = [3]
graph[6] = [3, 7]
graph[7] = [3, 6]



깊이 우선 탐색 (DFS) 구현

```
def DFS(graph, x, visited) :
    visited[x] = True
    result = [x]

    for v in graph[x] :
        if visited[v] == False :
            result = result +
                     DFS(graph, v, visited)

    return result
```

DFS(graph, 1, visited)

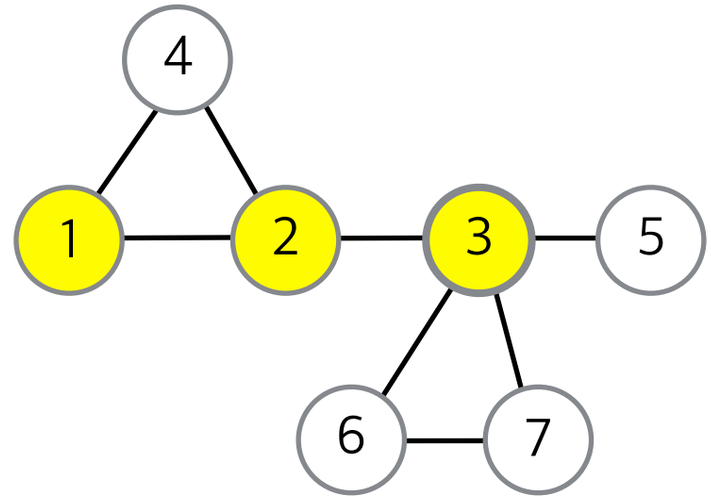
result = [1], v = 2

DFS(graph, 2, visited)

result = [2], v = 3

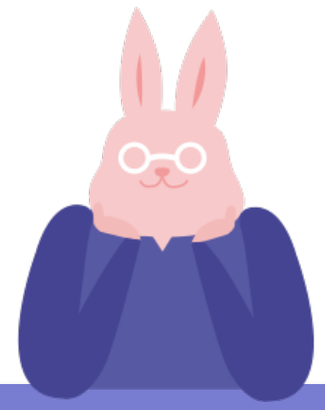
DFS(graph, 3, visited)

result = [3], v = 2



visited= [F, T, T, T, F, F, F, F]

graph[1] = [2, 4]
graph[2] = [3, 4]
graph[3] =[2, 5, 6, 7]
graph[4] = [1, 2]
graph[5] = [3]
graph[6] = [3, 7]
graph[7] = [3, 6]



깊이 우선 탐색 (DFS) 구현

```
def DFS(graph, x, visited) :
    visited[x] = True
    result = [x]

    for v in graph[x] :
        if visited[v] == False :
            result = result +
                     DFS(graph, v, visited)

    return result
```

DFS(graph, 1, visited)

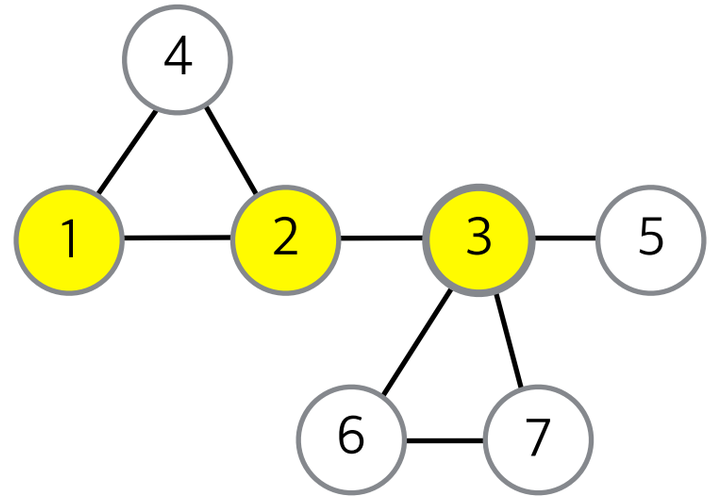
result = [1], v = 2

DFS(graph, 2, visited)

result = [2], v = 3

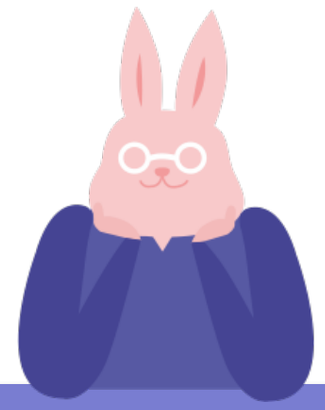
DFS(graph, 3, visited)

result = [3], v = 5



visited= [F, T, T, T, F, F, F, F]

graph[1] = [2, 4]
graph[2] = [3, 4]
graph[3] =[2, 5, 6, 7]
graph[4] = [1, 2]
graph[5] = [3]
graph[6] = [3, 7]
graph[7] = [3, 6]



깊이 우선 탐색 (DFS) 구현

```
def DFS(graph, x, visited) :
      visited[x] = True
      result = [x]

      for v in graph[x] :
            if visited[v] == False :
                  result = result +
                           DFS(graph, v, visited)

      return result
```

DFS(graph, 1, visited)

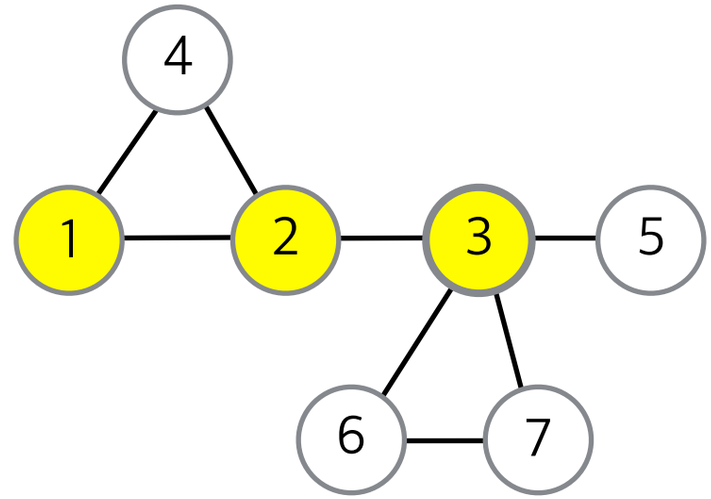
result = [1], v = 2

DFS(graph, 2, visited)

result = [2], v = 3

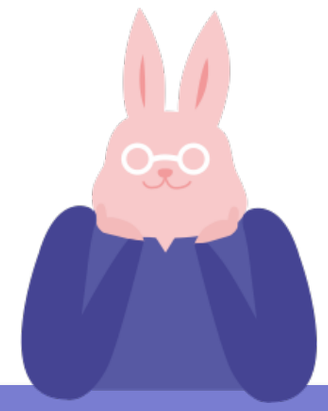
DFS(graph, 3, visited)

result = [3], v = 5



visited= [F, T, T, T, F, F, F, F]

graph[1] = [2, 4]
graph[2] = [3, 4]
graph[3] =[2, 5, 6, 7]
graph[4] = [1, 2]
graph[5] = [3]
graph[6] = [3, 7]
graph[7] = [3, 6]



깊이 우선 탐색 (DFS) 구현

```
def DFS(graph, x, visited) :
    visited[x] = True
    result = [x]

    for v in graph[x] :
        if visited[v] == False :
            result = result +
                     DFS(graph, v, visited)

    return result
```

DFS(graph, 1, visited)

result = [1], v = 2

DFS(graph, 2, visited)

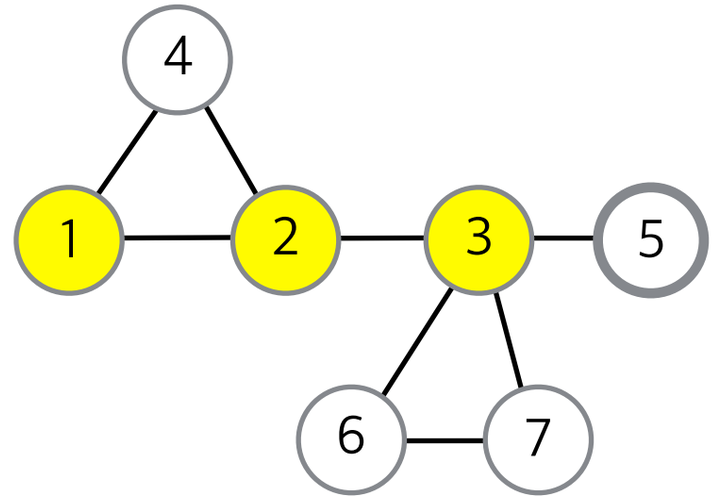
result = [2], v = 3

DFS(graph, 3, visited)

result = [3], v = 5

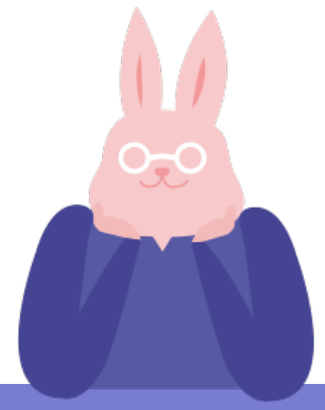
DFS(graph, 5, visited)

result = null



visited= [F, T, T, T, F, F, F, F]

graph[1] = [2, 4]
graph[2] = [3, 4]
graph[3] =[2, 5, 6, 7]
graph[4] = [1, 2]
graph[5] = [3]
graph[6] = [3, 7]
graph[7] = [3, 6]



깊이 우선 탐색 (DFS) 구현

```
def DFS(graph, x, visited) :
    visited[x] = True
    result = [x]

    for v in graph[x] :
        if visited[v] == False :
            result = result +
                     DFS(graph, v, visited)

    return result
```

DFS(graph, 1, visited)

result = [1], v = 2

DFS(graph, 2, visited)

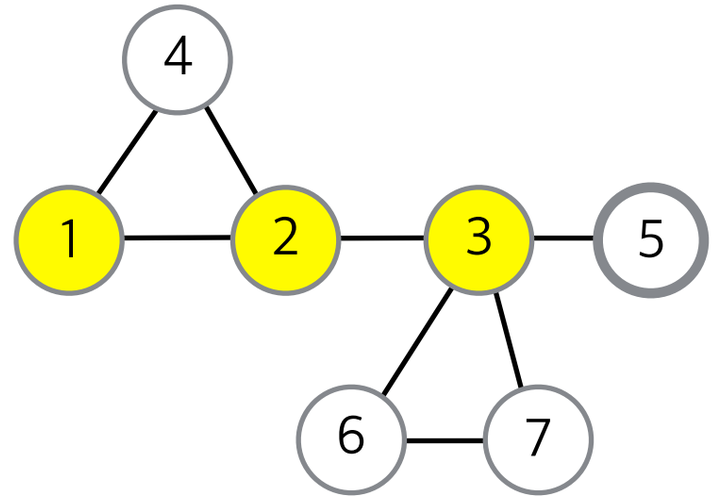
result = [2], v = 3

DFS(graph, 3, visited)

result = [3], v = 5

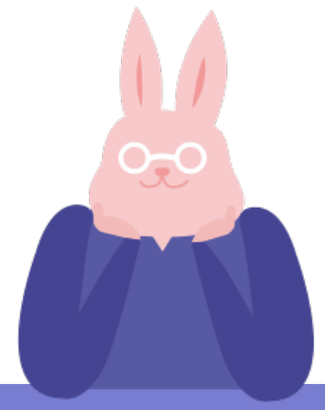
DFS(graph, 5, visited)

result = null



visited= [F, T, T, T, F, F, F, F]

graph[1] = [2, 4]
graph[2] = [3, 4]
graph[3] =[2, 5, 6, 7]
graph[4] = [1, 2]
graph[5] = [3]
graph[6] = [3, 7]
graph[7] = [3, 6]



깊이 우선 탐색 (DFS) 구현

```
def DFS(graph, x, visited) :
    visited[x] = True
    result = [x]

    for v in graph[x] :
        if visited[v] == False :
            result = result +
                     DFS(graph, v, visited)

    return result
```

DFS(graph, 1, visited)

result = [1], v = 2

DFS(graph, 2, visited)

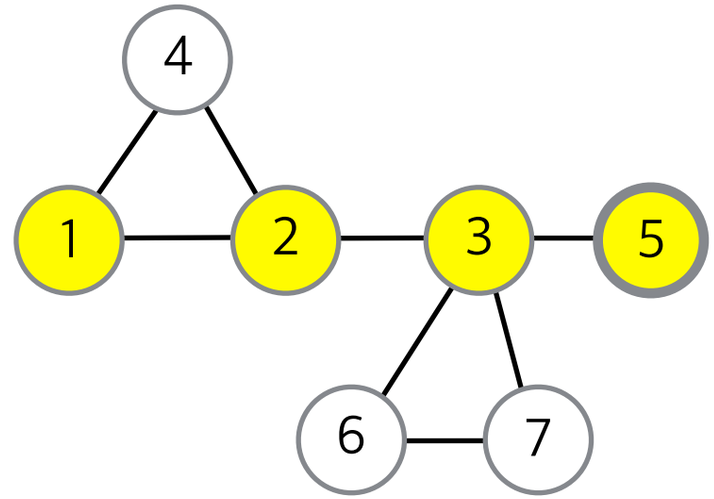
result = [2], v = 3

DFS(graph, 3, visited)

result = [3], v = 5

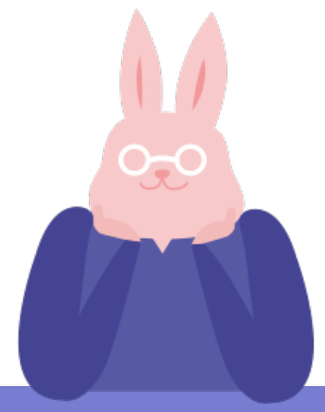
DFS(graph, 5, visited)

result = null



visited= [F, T, T, T, F, T, F, F]

graph[1] = [2, 4]
graph[2] = [3, 4]
graph[3] =[2, 5, 6, 7]
graph[4] = [1, 2]
graph[5] = [3]
graph[6] = [3, 7]
graph[7] = [3, 6]



깊이 우선 탐색 (DFS) 구현

```
def DFS(graph, x, visited) :
    visited[x] = True
    result = [x]

    for v in graph[x] :
        if visited[v] == False :
            result = result +
                     DFS(graph, v, visited)

    return result
```

DFS(graph, 1, visited)

result = [1], v = 2

DFS(graph, 2, visited)

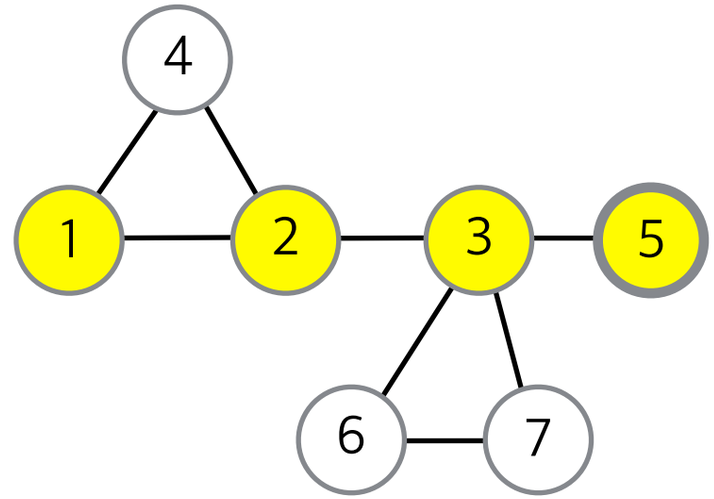
result = [2], v = 3

DFS(graph, 3, visited)

result = [3], v = 5

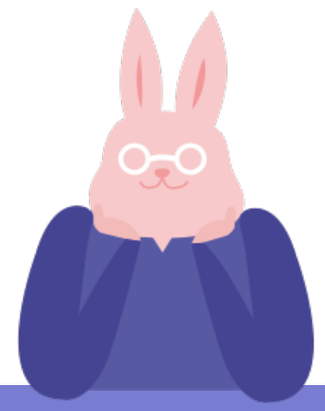
DFS(graph, 5, visited)

result = [5]



visited= [F, T, T, T, F, T, F, F]

graph[1] = [2, 4]
graph[2] = [3, 4]
graph[3] =[2, 5, 6, 7]
graph[4] = [1, 2]
graph[5] = [3]
graph[6] = [3, 7]
graph[7] = [3, 6]



깊이 우선 탐색 (DFS) 구현

```
def DFS(graph, x, visited) :
    visited[x] = True
    result = [x]

    for v in graph[x] :
        if visited[v] == False :
            result = result +
                     DFS(graph, v, visited)

    return result
```

DFS(graph, 1, visited)

result = [1], v = 2

DFS(graph, 2, visited)

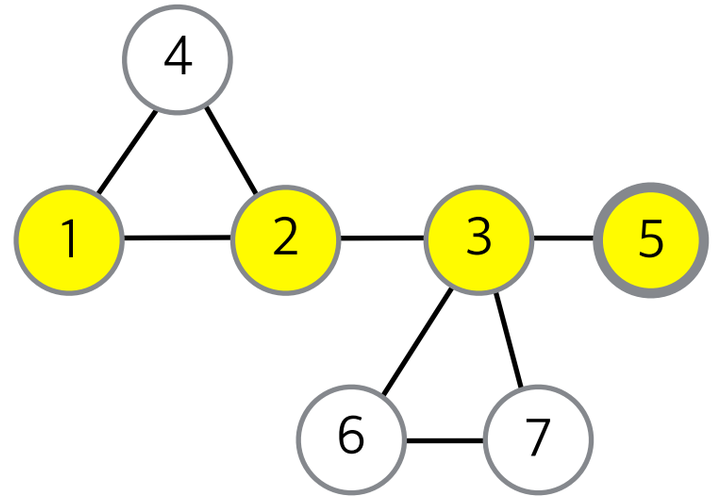
result = [2], v = 3

DFS(graph, 3, visited)

result = [3], v = 5

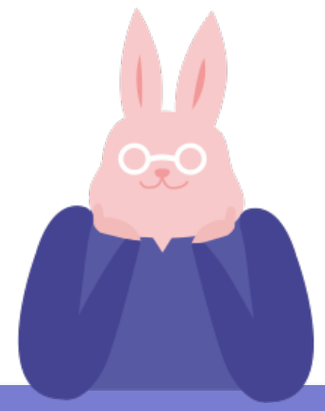
DFS(graph, 5, visited)

result = [5], v = 3



visited= [F, T, T, T, F, T, F, F]

graph[1] = [2, 4]
graph[2] = [3, 4]
graph[3] =[2, 5, 6, 7]
graph[4] = [1, 2]
graph[5] = [3]
graph[6] = [3, 7]
graph[7] = [3, 6]



깊이 우선 탐색 (DFS) 구현

```
def DFS(graph, x, visited) :
    visited[x] = True
    result = [x]

    for v in graph[x] :
        if visited[v] == False :
            result = result +
                     DFS(graph, v, visited)

    return result
```

DFS(graph, 1, visited)

result = [1], v = 2

DFS(graph, 2, visited)

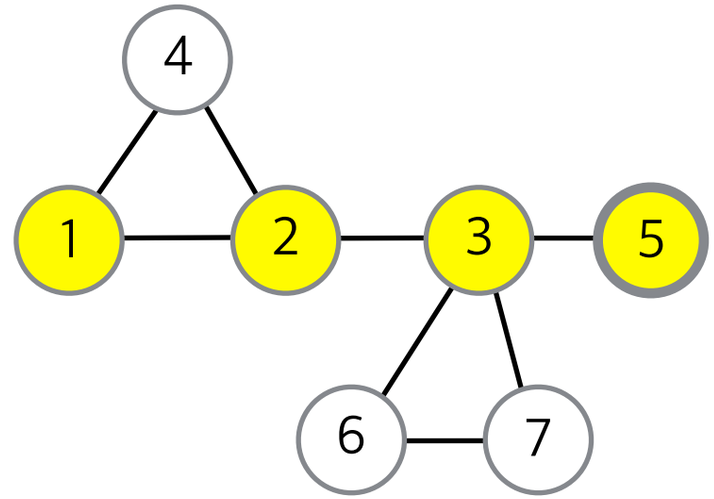
result = [2], v = 3

DFS(graph, 3, visited)

result = [3], v = 5

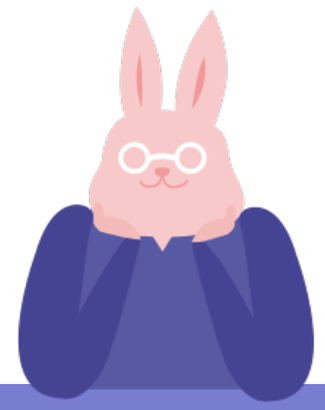
DFS(graph, 5, visited)

result = [5], v = 3



visited= [F, T, T, T, F, T, F, F]

graph[1] = [2, 4]
graph[2] = [3, 4]
graph[3] =[2, 5, 6, 7]
graph[4] = [1, 2]
graph[5] = [3]
graph[6] = [3, 7]
graph[7] = [3, 6]



깊이 우선 탐색 (DFS) 구현

```
def DFS(graph, x, visited) :
      visited[x] = True
      result = [x]

      for v in graph[x] :
            if visited[v] == False :
                  result = result +
                        DFS(graph, v, visited)

      return result
```

DFS(graph, 1, visited)

result = [1], v = 2

DFS(graph, 2, visited)

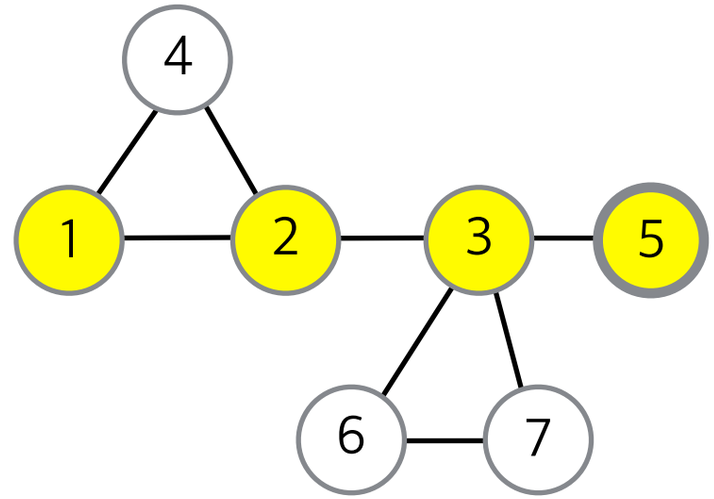
result = [2], v = 3

DFS(graph, 3, visited)

result = [3], v = 5

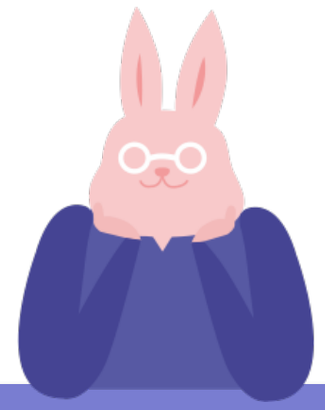
DFS(graph, 5, visited)

result = [5], v = 3



visited= [F, T, T, T, F, T, F, F]

graph[1] = [2, 4]
graph[2] = [3, 4]
graph[3] =[2, 5, 6, 7]
graph[4] = [1, 2]
graph[5] = [3]
graph[6] = [3, 7]
graph[7] = [3, 6]



깊이 우선 탐색 (DFS) 구현

```
def DFS(graph, x, visited) :
      visited[x] = True
      result = [x]

      for v in graph[x] :
            if visited[v] == False :
                  result = result +
                           DFS(graph, v, visited)

      return result
```

DFS(graph, 1, visited)

result = [1], v = 2

DFS(graph, 2, visited)

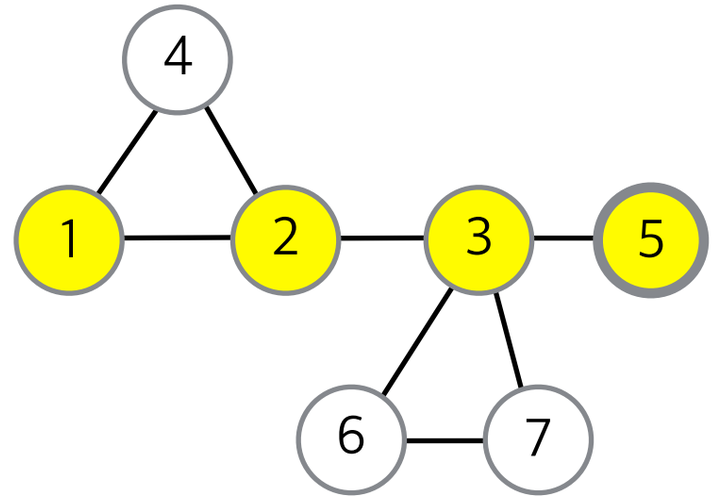
result = [2], v = 3

DFS(graph, 3, visited)

result = [3], v = 5

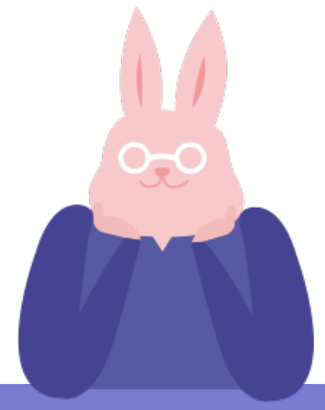
DFS(graph, 5, visited)

result = [5], v = 3



visited= [F, T, T, T, F, T, F, F]

graph[1] = [2, 4]
graph[2] = [3, 4]
graph[3] =[2, 5, 6, 7]
graph[4] = [1, 2]
graph[5] = [3]
graph[6] = [3, 7]
graph[7] = [3, 6]



깊이 우선 탐색 (DFS) 구현

```
def DFS(graph, x, visited) :
      visited[x] = True
      result = [x]

      for v in graph[x] :
            if visited[v] == False :
                  result = result +
                           DFS(graph, v, visited)

      return result
```

DFS(graph, 1, visited)

result = [1], v = 2

DFS(graph, 2, visited)

result = [2], v = 3

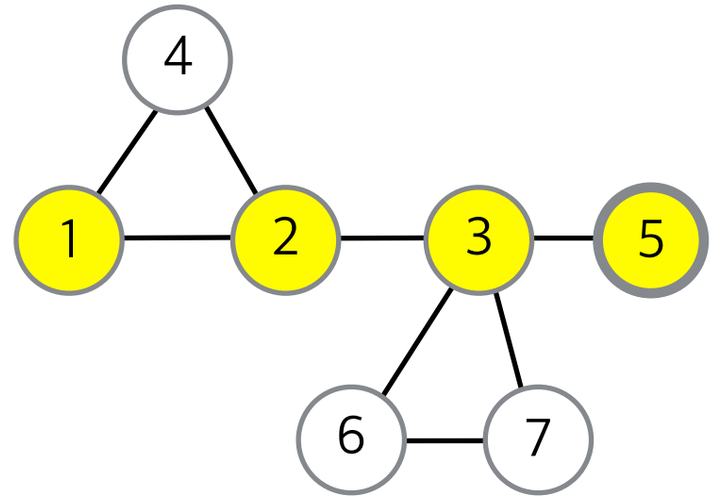
DFS(graph, 3, visited)

result = [3], v = 5

DFS(graph, 5, visited)

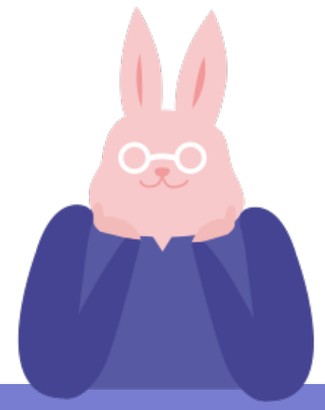
result = [5], v = 3

[5]



visited= [F, T, T, T, F, T, F, F]

graph[1] = [2, 4]
graph[2] = [3, 4]
graph[3] =[2, 5, 6, 7]
graph[4] = [1, 2]
graph[5] = [3]
graph[6] = [3, 7]
graph[7] = [3, 6]



깊이 우선 탐색 (DFS) 구현

```
def DFS(graph, x, visited) :
    visited[x] = True
    result = [x]

    for v in graph[x] :
        if visited[v] == False :
            result = result +
                     DFS(graph, v, visited)

    return result
```

DFS(graph, 1, visited)

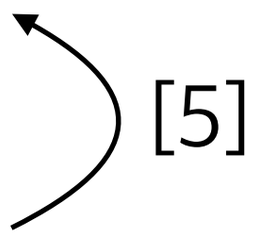
result = [1], v = 2

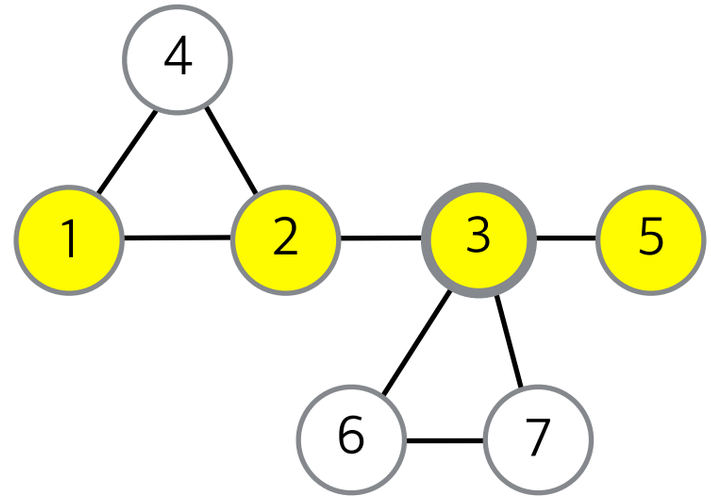
DFS(graph, 2, visited)

result = [2], v = 3

DFS(graph, 3, visited)

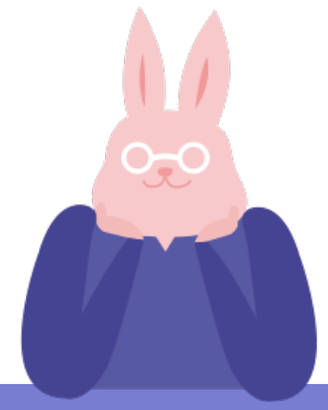
result = [3], v = 5





visited= [F, T, T, T, F, T, F, F]

graph[1] = [2, 4]
graph[2] = [3, 4]
graph[3] =[2, 5, 6, 7]
graph[4] = [1, 2]
graph[5] = [3]
graph[6] = [3, 7]
graph[7] = [3, 6]



깊이 우선 탐색 (DFS) 구현

```
def DFS(graph, x, visited) :
    visited[x] = True
    result = [x]

    for v in graph[x] :
        if visited[v] == False :
            result = result +
                     DFS(graph, v, visited)

    return result
```

DFS(graph, 1, visited)

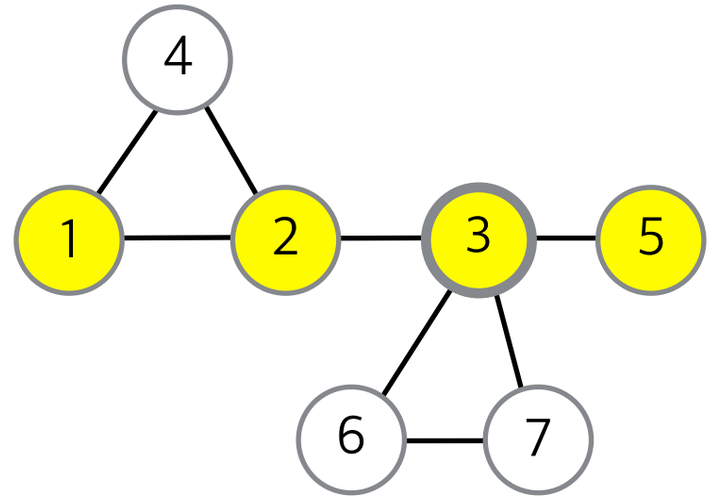
result = [1], v = 2

DFS(graph, 2, visited)

result = [2], v = 3

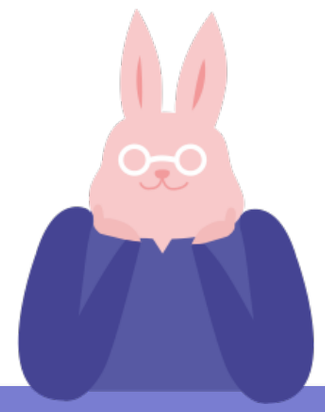
DFS(graph, 3, visited)

result = [3, 5], v = 5



visited= [F, T, T, T, F, T, F, F]

graph[1] = [2, 4]
graph[2] = [3, 4]
graph[3] =[2, 5, 6, 7]
graph[4] = [1, 2]
graph[5] = [3]
graph[6] = [3, 7]
graph[7] = [3, 6]



깊이 우선 탐색 (DFS) 구현

```
def DFS(graph, x, visited) :
    visited[x] = True
    result = [x]

    for v in graph[x] :
        if visited[v] == False :
            result = result +
                     DFS(graph, v, visited)

    return result
```

DFS(graph, 1, visited)

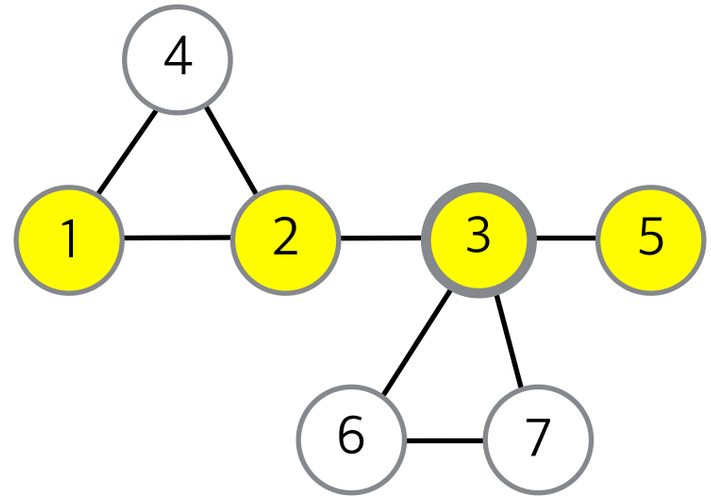
result = [1], v = 2

DFS(graph, 2, visited)

result = [2], v = 3

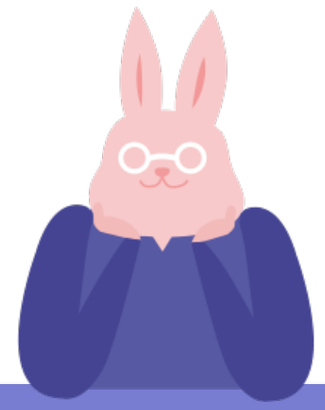
DFS(graph, 3, visited)

result = [3, 5], v = 6



visited= [F, T, T, T, F, T, F, F]

graph[1] = [2, 4]
graph[2] = [3, 4]
graph[3] =[2, 5, 6, 7]
graph[4] = [1, 2]
graph[5] = [3]
graph[6] = [3, 7]
graph[7] = [3, 6]



깊이 우선 탐색 (DFS) 구현

```
def DFS(graph, x, visited) :
    visited[x] = True
    result = [x]

    for v in graph[x] :
        if visited[v] == False :
            result = result +
                     DFS(graph, v, visited)

    return result
```

DFS(graph, 1, visited)

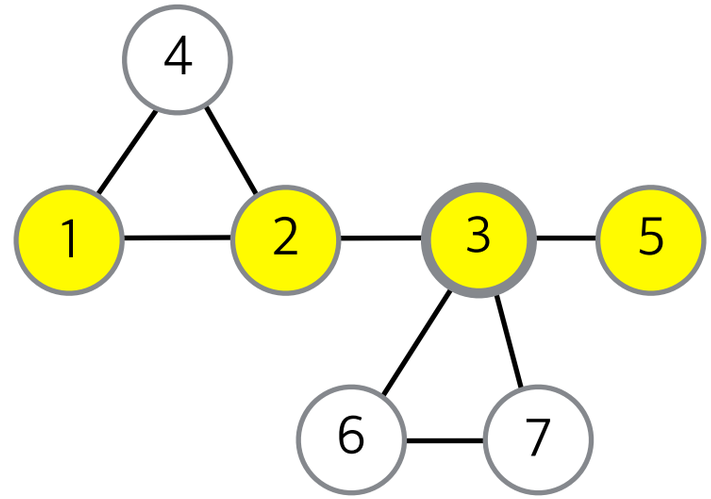
result = [1], v = 2

DFS(graph, 2, visited)

result = [2], v = 3

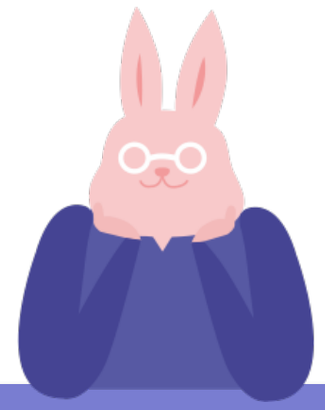
DFS(graph, 3, visited)

result = [3, 5], v = 6



visited= [F, T, T, T, F, T, F, F]

graph[1] = [2, 4]
graph[2] = [3, 4]
graph[3] =[2, 5, 6, 7]
graph[4] = [1, 2]
graph[5] = [3]
graph[6] = [3, 7]
graph[7] = [3, 6]



깊이 우선 탐색 (DFS) 구현

```
def DFS(graph, x, visited) :
    visited[x] = True
    result = [x]

    for v in graph[x] :
        if visited[v] == False :
            result = result +
                     DFS(graph, v, visited)

    return result
```

DFS(graph, 1, visited)

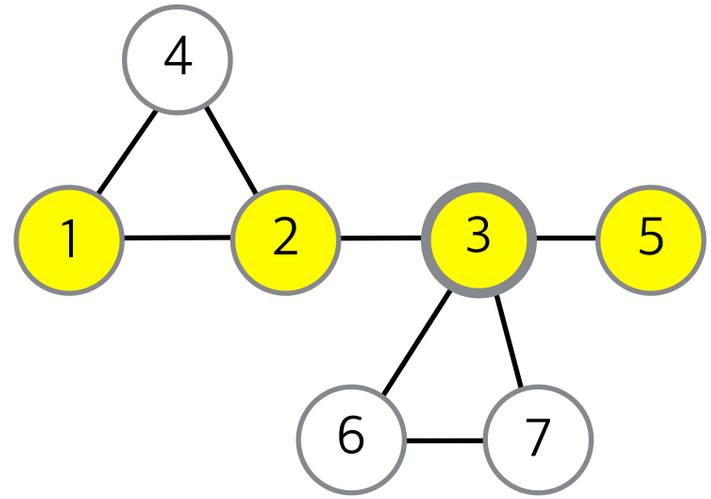
result = [1], v = 2

DFS(graph, 2, visited)

result = [2], v = 3

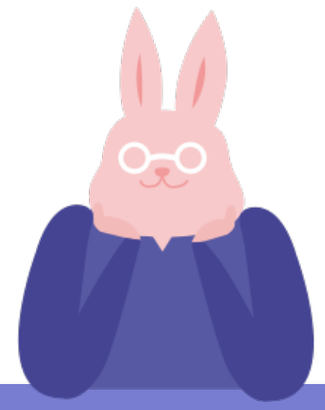
DFS(graph, 3, visited)

result = [3, 5], v = 6



visited= [F, T, T, T, F, T, F, F]

graph[1] = [2, 4]
graph[2] = [3, 4]
graph[3] =[2, 5, 6, 7]
graph[4] = [1, 2]
graph[5] = [3]
graph[6] = [3, 7]
graph[7] = [3, 6]



깊이 우선 탐색 (DFS) 구현

```
def DFS(graph, x, visited) :
    visited[x] = True
    result = [x]

    for v in graph[x] :
        if visited[v] == False :
            result = result +
                     DFS(graph, v, visited)

    return result
```

DFS(graph, 1, visited)

result = [1], v = 2

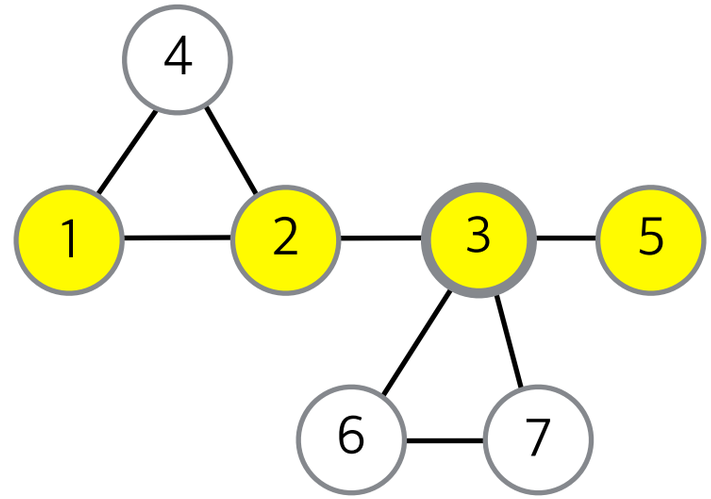
DFS(graph, 2, visited)

result = [2], v = 3

DFS(graph, 3, visited)

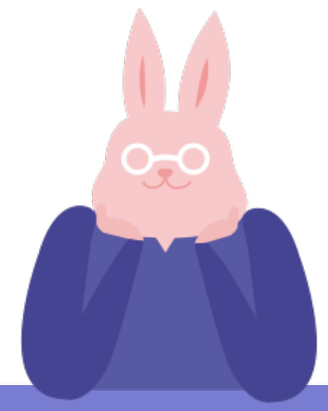
result = [3, 5], v = 6

DFS(graph, 6, visited)가 return하는 값은 ?



visited= [F, T, T, T, F, T, F, F]

graph[1] = [2, 4]
graph[2] = [3, 4]
graph[3] =[2, 5, 6, 7]
graph[4] = [1, 2]
graph[5] = [3]
graph[6] = [3, 7]
graph[7] = [3, 6]



깊이 우선 탐색 (DFS) 구현

```
def DFS(graph, x, visited) :
    visited[x] = True
    result = [x]

    for v in graph[x] :
        if visited[v] == False :
            result = result +
                     DFS(graph, v, visited)

    return result
```

DFS(graph, 1, visited)

result = [1], v = 2

DFS(graph, 2, visited)

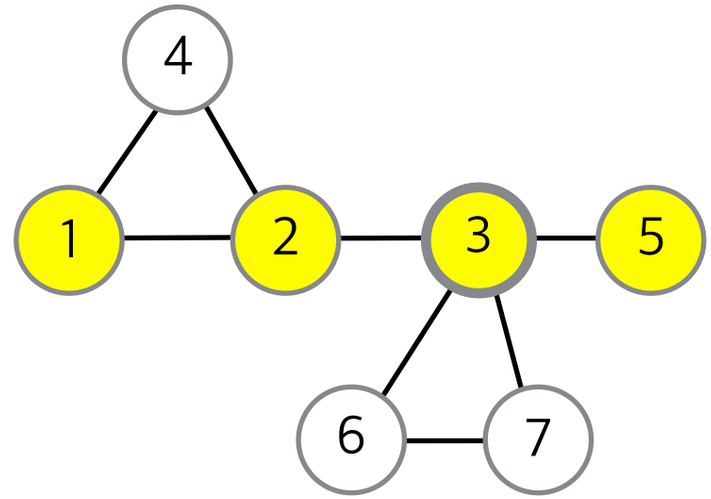
result = [2], v = 3

DFS(graph, 3, visited)

result = [3, 5], v = 6

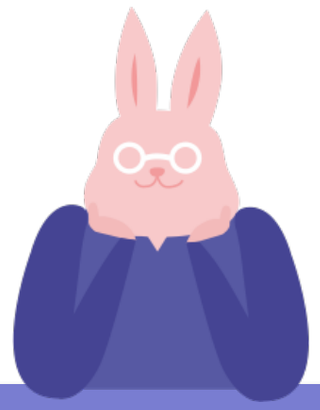
DFS(graph, 6, visited)

[6, 7]



visited= [F, T, T, T, F, T, F, F]

graph[1] = [2, 4]
graph[2] = [3, 4]
graph[3] =[2, 5, 6, 7]
graph[4] = [1, 2]
graph[5] = [3]
graph[6] = [3, 7]
graph[7] = [3, 6]



깊이 우선 탐색 (DFS) 구현

```
def DFS(graph, x, visited) :
    visited[x] = True
    result = [x]

    for v in graph[x] :
        if visited[v] == False :
            result = result +
                     DFS(graph, v, visited)

    return result
```

DFS(graph, 1, visited)

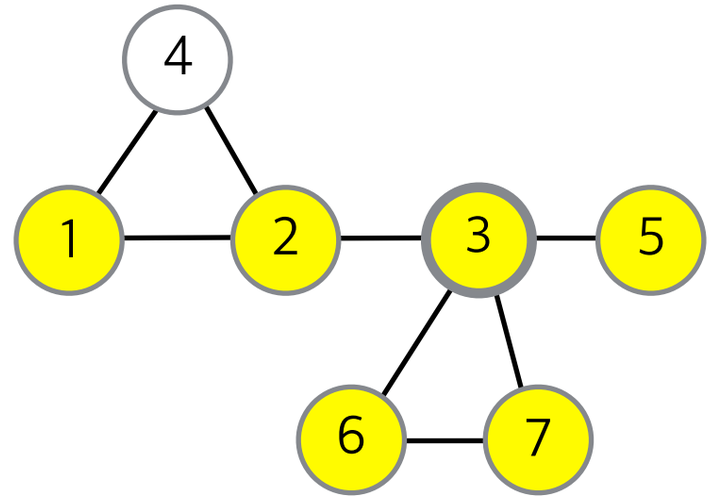
result = [1], v = 2

DFS(graph, 2, visited)

result = [2], v = 3

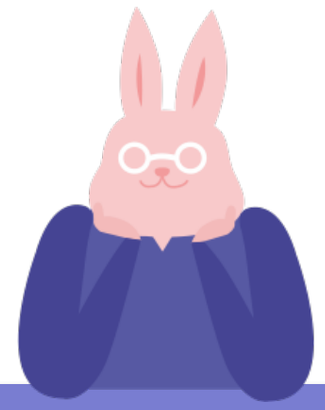
DFS(graph, 3, visited)

result = [3, 5, 6, 7], v = 6



visited= [F, T, T, T, F, T, **T**, **T**]

graph[1] = [2, 4]
graph[2] = [3, 4]
graph[3] =[2, 5, 6, 7]
graph[4] = [1, 2]
graph[5] = [3]
graph[6] = [3, 7]
graph[7] = [3, 6]



깊이 우선 탐색 (DFS) 구현

```
def DFS(graph, x, visited) :
    visited[x] = True
    result = [x]

    for v in graph[x] :
        if visited[v] == False :
            result = result +
                     DFS(graph, v, visited)

    return result
```

DFS(graph, 1, visited)

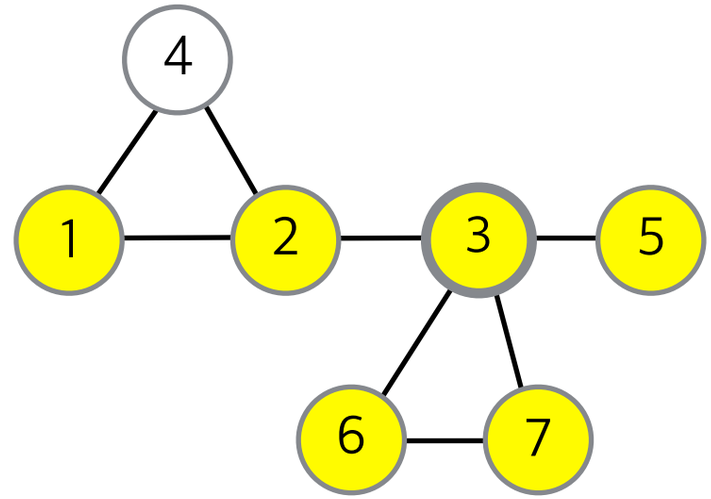
result = [1], v = 2

DFS(graph, 2, visited)

result = [2], v = 3

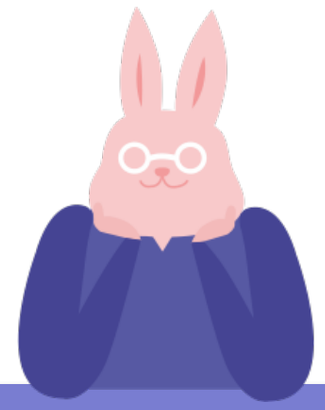
DFS(graph, 3, visited)

result = [3, 5, 6, 7], v = 7



visited= [F, T, T, T, F, T, T, T]

graph[1] = [2, 4]
graph[2] = [3, 4]
graph[3] =[2, 5, 6, 7]
graph[4] = [1, 2]
graph[5] = [3]
graph[6] = [3, 7]
graph[7] = [3, 6]



깊이 우선 탐색 (DFS) 구현

```
def DFS(graph, x, visited) :
    visited[x] = True
    result = [x]

    for v in graph[x] :
        if visited[v] == False :
            result = result +
                     DFS(graph, v, visited)

    return result
```

DFS(graph, 1, visited)

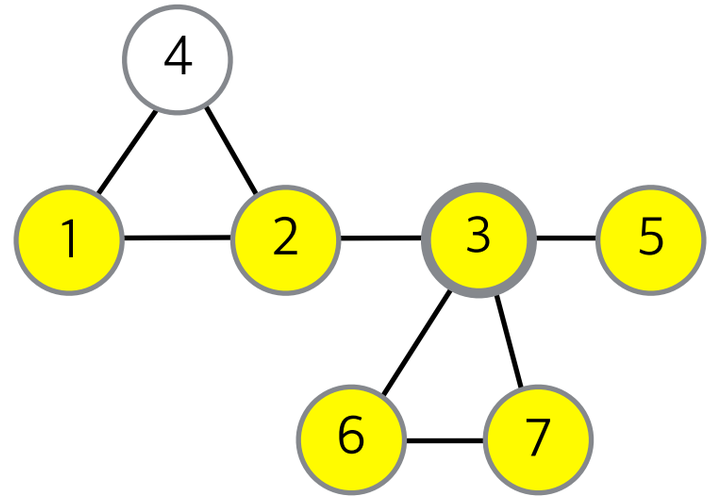
result = [1], v = 2

DFS(graph, 2, visited)

result = [2], v = 3

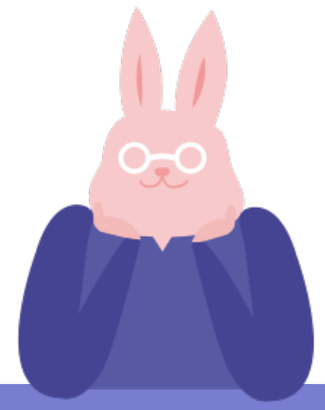
DFS(graph, 3, visited)

result = [3, 5, 6, 7], v = 7



visited= [F, T, T, T, F, T, T, T]

graph[1] = [2, 4]
graph[2] = [3, 4]
graph[3] =[2, 5, 6, 7]
graph[4] = [1, 2]
graph[5] = [3]
graph[6] = [3, 7]
graph[7] = [3, 6]



깊이 우선 탐색 (DFS) 구현

```
def DFS(graph, x, visited) :
      visited[x] = True
      result = [x]

      for v in graph[x] :
            if visited[v] == False :
                  result = result +
                        DFS(graph, v, visited)

      return result
```

DFS(graph, 1, visited)

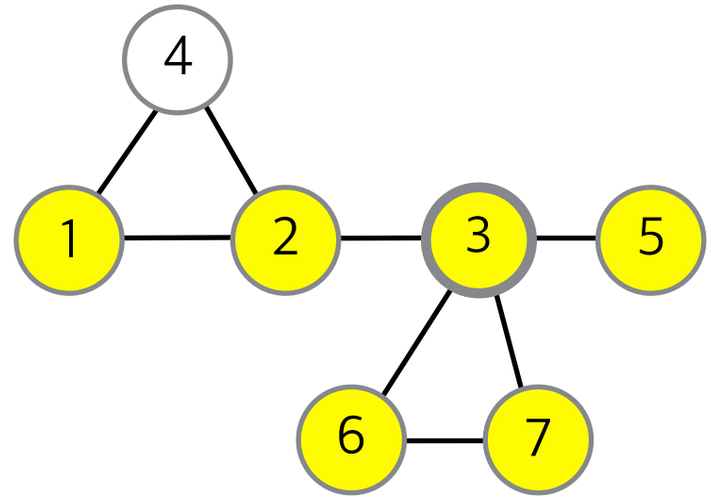
result = [1], v = 2

DFS(graph, 2, visited)

result = [2], v = 3

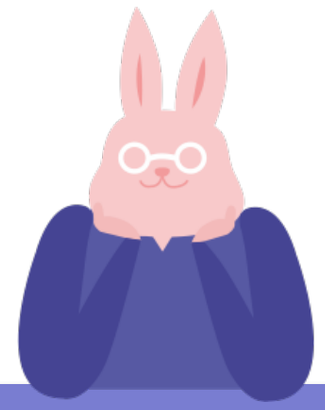
DFS(graph, 3, visited)

result = [3, 5, 6, 7], v = 7



visited= [F, T, T, T, F, T, T, T]

graph[1] = [2, 4]
graph[2] = [3, 4]
graph[3] =[2, 5, 6, 7]
graph[4] = [1, 2]
graph[5] = [3]
graph[6] = [3, 7]
graph[7] = [3, 6]



깊이 우선 탐색 (DFS) 구현

```
def DFS(graph, x, visited) :
      visited[x] = True
      result = [x]

      for v in graph[x] :
            if visited[v] == False :
                  result = result +
                           DFS(graph, v, visited)

      return result
```

DFS(graph, 1, visited)

result = [1], v = 2

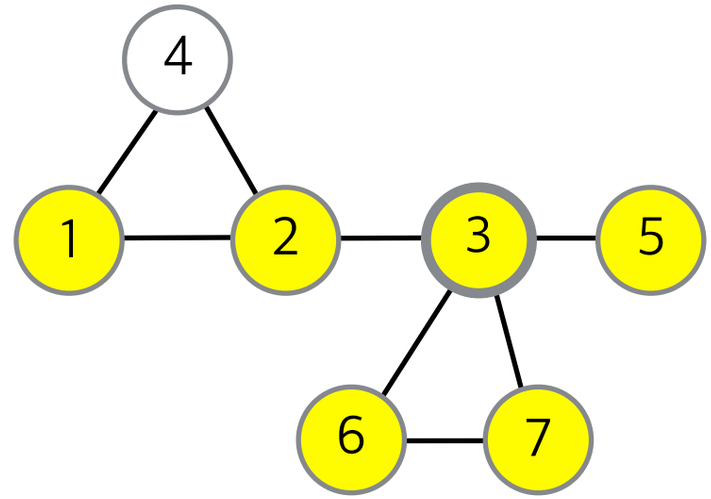
DFS(graph, 2, visited)

result = [2], v = 3

DFS(graph, 3, visited)

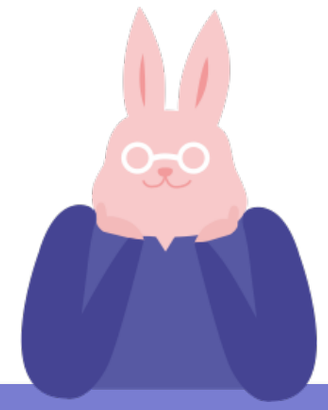
result = [3, 5, 6, 7], v = 7

[3, 5, 6, 7]



visited= [F, T, T, T, F, T, T, T]

graph[1] = [2, 4]
graph[2] = [3, 4]
graph[3] =[2, 5, 6, 7]
graph[4] = [1, 2]
graph[5] = [3]
graph[6] = [3, 7]
graph[7] = [3, 6]



깊이 우선 탐색 (DFS) 구현

```
def DFS(graph, x, visited) :
      visited[x] = True
      result = [x]

      for v in graph[x] :
            if visited[v] == False :
                  result = result +
                           DFS(graph, v, visited)

      return result
```

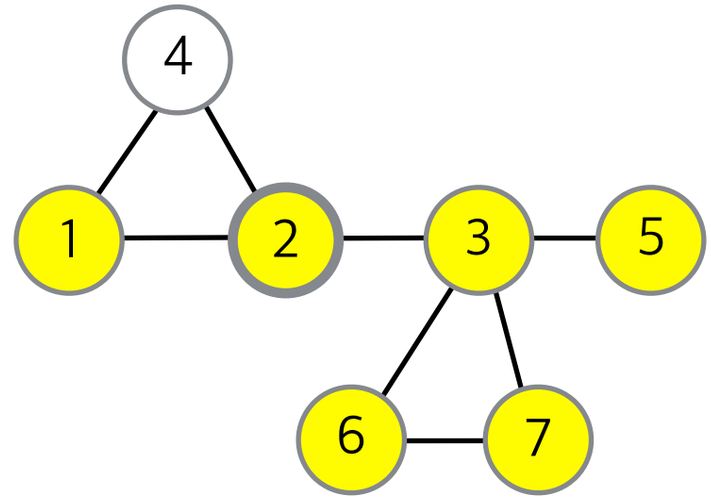
DFS(graph, 1, visited)

result = [1], v = 2

DFS(graph, 2, visited)

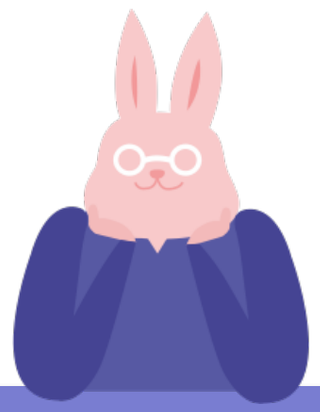
result = [2], v = 3

[3, 5, 6, 7]



visited= [F, T, T, T, F, T, T, T]

graph[1] = [2, 4]
graph[2] = [3, 4]
graph[3] =[2, 5, 6, 7]
graph[4] = [1, 2]
graph[5] = [3]
graph[6] = [3, 7]
graph[7] = [3, 6]



깊이 우선 탐색 (DFS) 구현

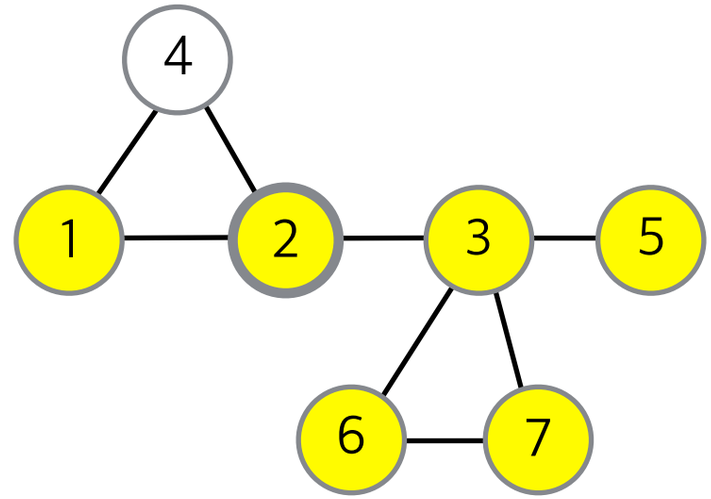
```
def DFS(graph, x, visited) :
    visited[x] = True
    result = [x]

    for v in graph[x] :
        if visited[v] == False :
            result = result +
                     DFS(graph, v, visited)

    return result
```

DFS(graph, 1, visited)

result = [1], v = 2

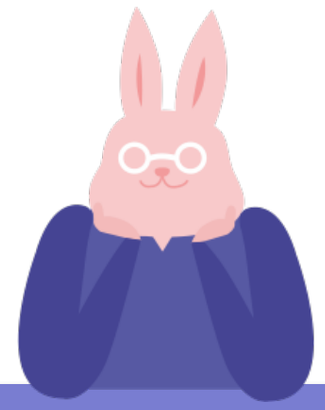
DFS(graph, 2, visited)

result = [2, 3, 5, 6, 7], v = 3



visited= [F, T, T, T, F, T, T, T]

graph[1] = [2, 4]
graph[2] = [3, 4]
graph[3] =[2, 5, 6, 7]
graph[4] = [1, 2]
graph[5] = [3]
graph[6] = [3, 7]
graph[7] = [3, 6]



깊이 우선 탐색 (DFS) 구현

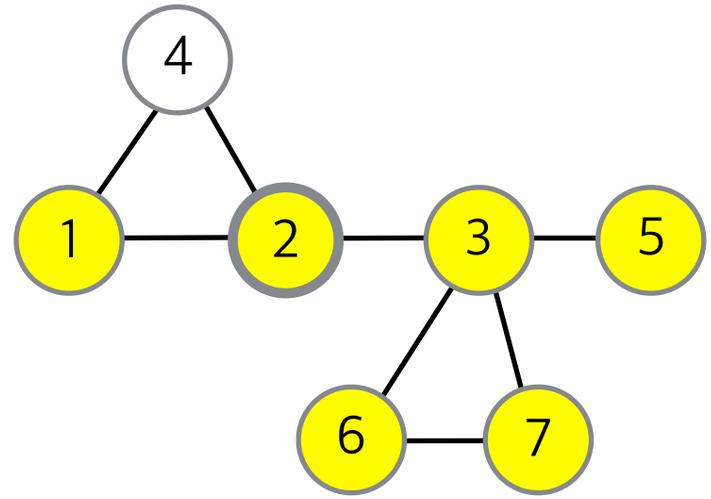
```
def DFS(graph, x, visited) :
    visited[x] = True
    result = [x]

    for v in graph[x] :
        if visited[v] == False :
            result = result +
                     DFS(graph, v, visited)

    return result
```

DFS(graph, 1, visited)

result = [1], v = 2

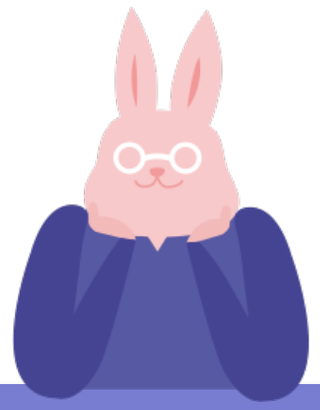
DFS(graph, 2, visited)

result = [2, 3, 5, 6, 7], v = 4



visited= [F, T, T, T, F, T, T, T]

graph[1] = [2, 4]
graph[2] = [3, 4]
graph[3] =[2, 5, 6, 7]
graph[4] = [1, 2]
graph[5] = [3]
graph[6] = [3, 7]
graph[7] = [3, 6]



깊이 우선 탐색 (DFS) 구현

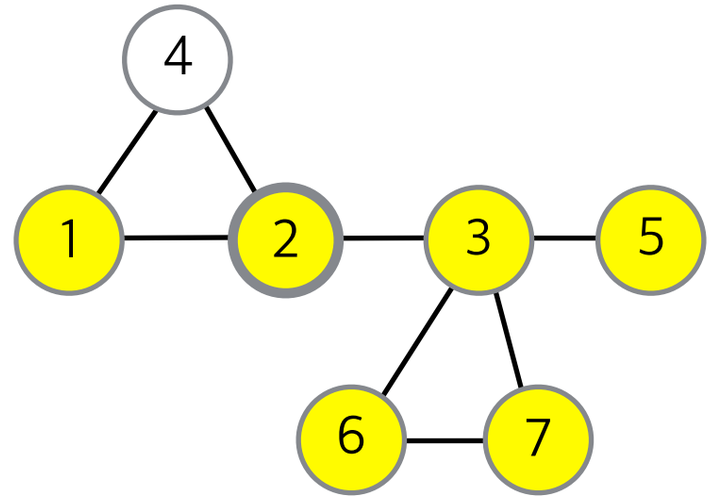
```
def DFS(graph, x, visited) :
      visited[x] = True
      result = [x]

      for v in graph[x] :
            if visited[v] == False :
                  result = result +
                           DFS(graph, v, visited)

      return result
```

DFS(graph, 1, visited)

result = [1], v = 2

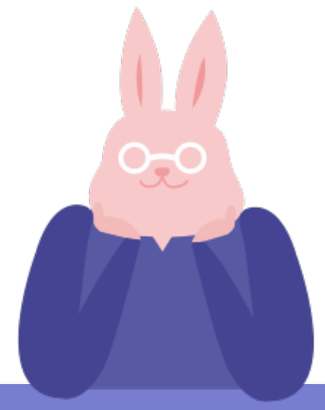
DFS(graph, 2, visited)

result = [2, 3, 5, 6, 7], v = 4



visited= [F, T, T, T, F, T, T, T]

graph[1] = [2, 4]
graph[2] = [3, 4]
graph[3] =[2, 5, 6, 7]
graph[4] = [1, 2]
graph[5] = [3]
graph[6] = [3, 7]
graph[7] = [3, 6]



깊이 우선 탐색 (DFS) 구현

```
def DFS(graph, x, visited) :
    visited[x] = True
    result = [x]

    for v in graph[x] :
        if visited[v] == False :
            result = result +
                     DFS(graph, v, visited)

    return result
```

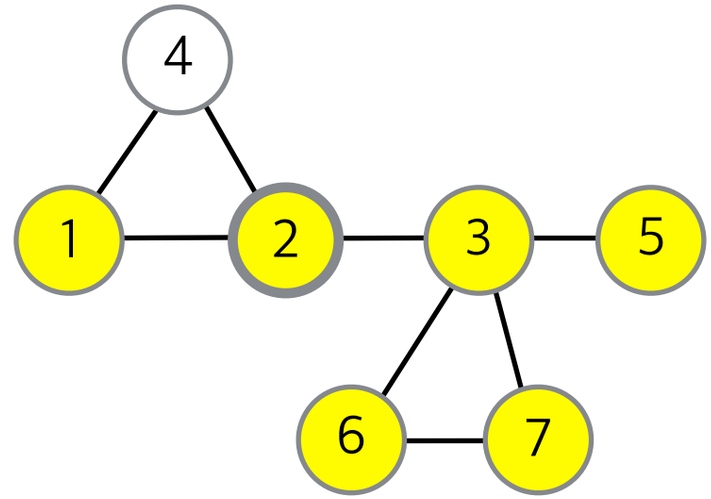
DFS(graph, 1, visited)

result = [1], v = 2

DFS(graph, 2, visited)

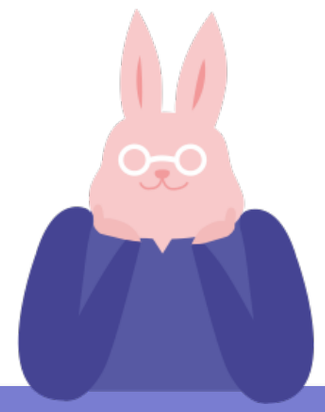
result = [2, 3, 5, 6, 7], v = 4

DFS(graph, 4, visited)가 return하는 값은 ?



visited= [F, T, T, T, F, T, T, T]

graph[1] = [2, 4]
graph[2] = [3, 4]
graph[3] =[2, 5, 6, 7]
graph[4] = [1, 2]
graph[5] = [3]
graph[6] = [3, 7]
graph[7] = [3, 6]



깊이 우선 탐색 (DFS) 구현

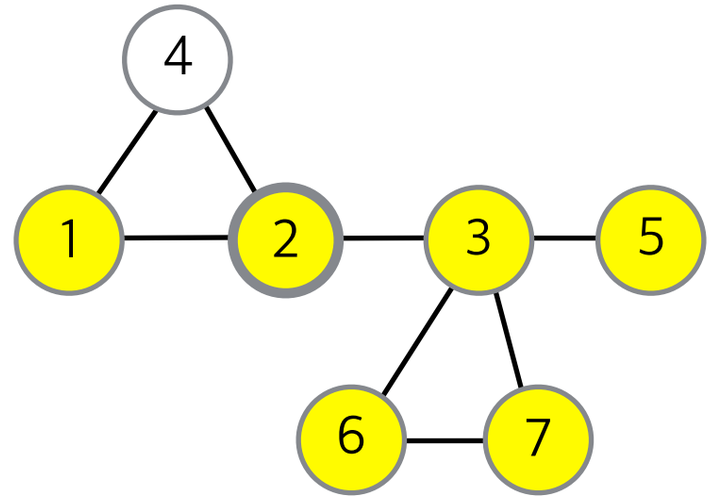
```
def DFS(graph, x, visited) :
      visited[x] = True
      result = [x]

      for v in graph[x] :
            if visited[v] == False :
                  result = result +
                           DFS(graph, v, visited)

      return result
```

DFS(graph, 1, visited)

result = [1], v = 2

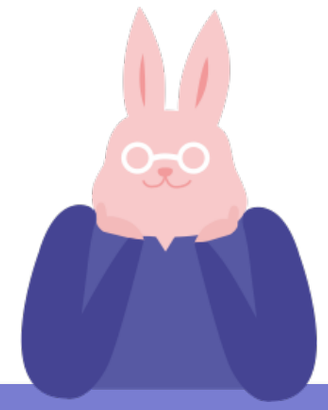
DFS(graph, 2, visited)

result = [2, 3, 5, 6, 7], v = 4 [4]



visited= [F, T, T, T, F, T, T, T]

graph[1] = [2, 4]
graph[2] = [3, 4]
graph[3] =[2, 5, 6, 7]
graph[4] = [1, 2]
graph[5] = [3]
graph[6] = [3, 7]
graph[7] = [3, 6]



깊이 우선 탐색 (DFS) 구현

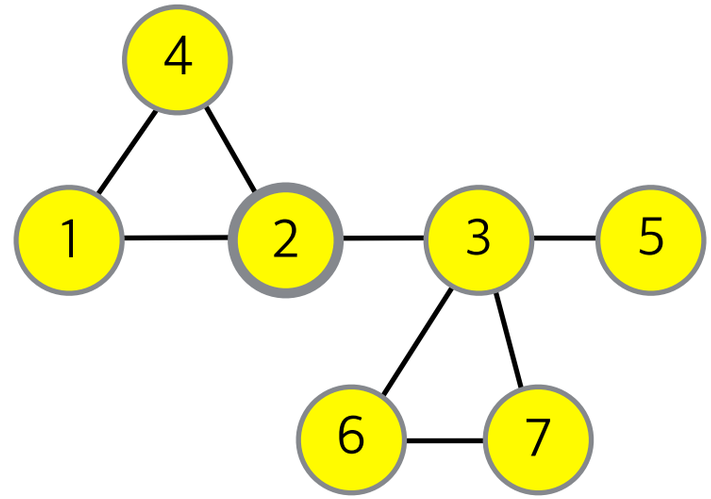
```
def DFS(graph, x, visited) :
    visited[x] = True
    result = [x]

    for v in graph[x] :
        if visited[v] == False :
            result = result +
                     DFS(graph, v, visited)

    return result
```

DFS(graph, 1, visited)

result = [1], v = 2

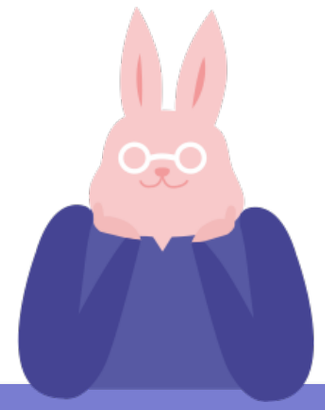
DFS(graph, 2, visited)

result = [2, 3, 5, 6, 7, 4], v = 4



visited= [F, T, T, T, **T**, T, T, T]

graph[1] = [2, 4]
graph[2] = [3, 4]
graph[3] =[2, 5, 6, 7]
graph[4] = [1, 2]
graph[5] = [3]
graph[6] = [3, 7]
graph[7] = [3, 6]



깊이 우선 탐색 (DFS) 구현

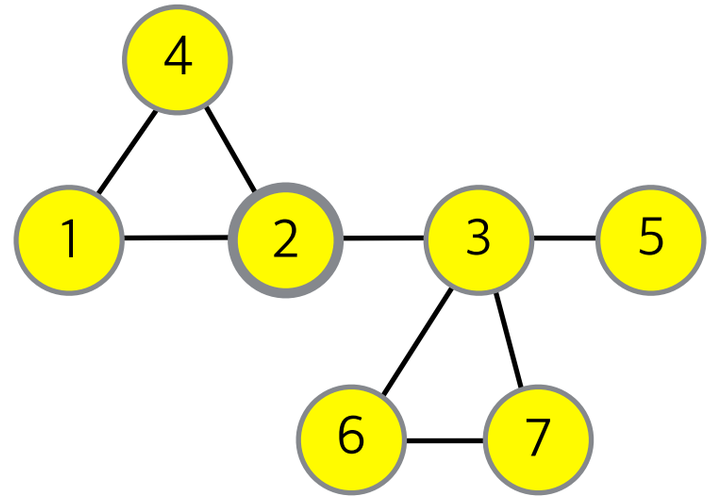
```
def DFS(graph, x, visited) :
      visited[x] = True
      result = [x]

      for v in graph[x] :
            if visited[v] == False :
                  result = result +
                           DFS(graph, v, visited)

      return result
```

DFS(graph, 1, visited)

result = [1], v = 2

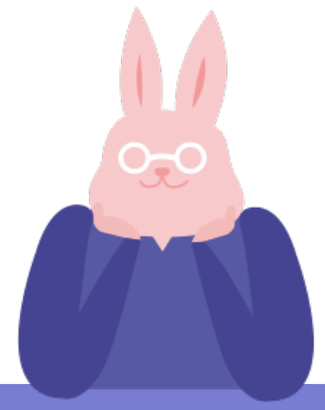
DFS(graph, 2, visited)

result = [2, 3, 5, 6, 7, 4], v = 4



visited= [F, T, T, T, T, T, T, T]

graph[1] = [2, 4]
graph[2] = [3, 4]
graph[3] =[2, 5, 6, 7]
graph[4] = [1, 2]
graph[5] = [3]
graph[6] = [3, 7]
graph[7] = [3, 6]



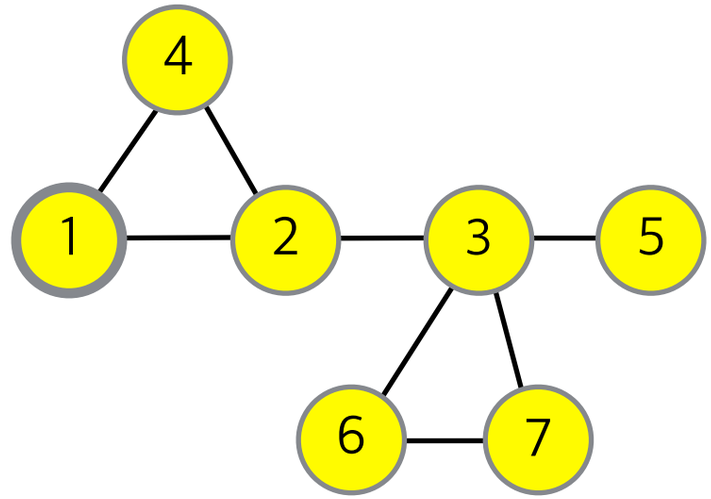
깊이 우선 탐색 (DFS) 구현

```
def DFS(graph, x, visited) :
      visited[x] = True
      result = [x]

      for v in graph[x] :
            if visited[v] == False :
                  result = result +
                           DFS(graph, v, visited)

      return result
```

DFS(graph, 1, visited)

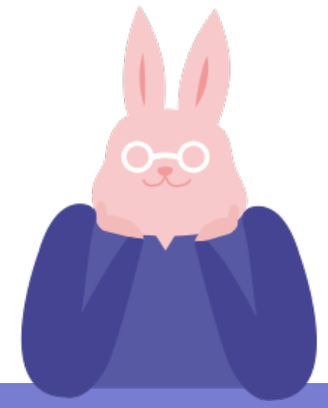
result = [1], v = 2

[2, 3, 5, 6, 7, 4]



visited= [F, T, T, T, T, T, T, T]

graph[1] = [2, 4]
graph[2] = [3, 4]
graph[3] =[2, 5, 6, 7]
graph[4] = [1, 2]
graph[5] = [3]
graph[6] = [3, 7]
graph[7] = [3, 6]

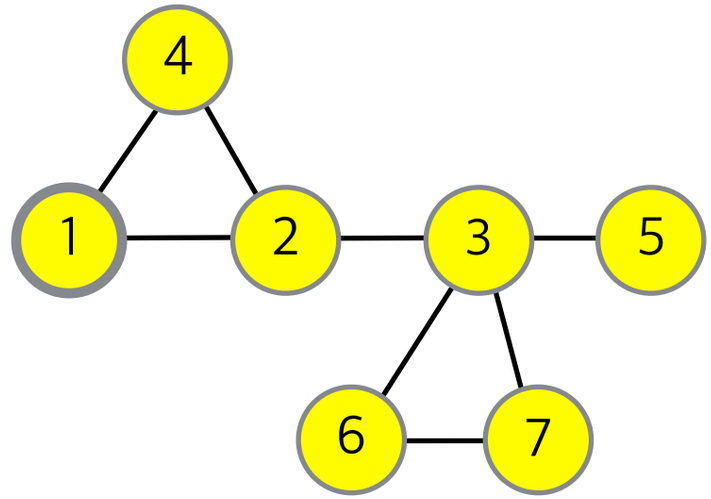


깊이 우선 탐색 (DFS) 구현

```
def DFS(graph, x, visited) :
    visited[x] = True
    result = [x]

    for v in graph[x] :
        if visited[v] == False :
            result = result +
                     DFS(graph, v, visited)

    return result
```

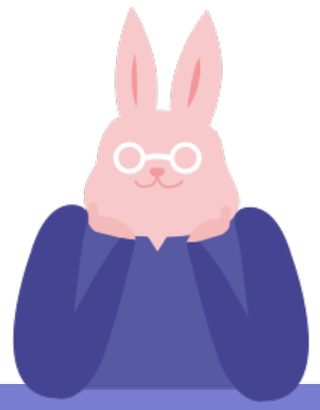
DFS(graph, 1, visited)

result = [1, 2, 3, 5, 6, 7, 4], v = 2



visited= [F, T, T, T, T, T, T, T]

graph[1] = [2, 4]
graph[2] = [3, 4]
graph[3] =[2, 5, 6, 7]
graph[4] = [1, 2]
graph[5] = [3]
graph[6] = [3, 7]
graph[7] = [3, 6]

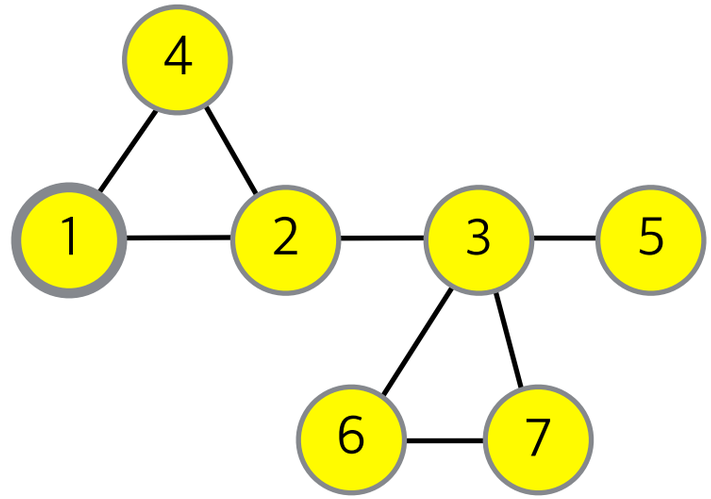


깊이 우선 탐색 (DFS) 구현

```
def DFS(graph, x, visited) :
    visited[x] = True
    result = [x]

    for v in graph[x] :
        if visited[v] == False :
            result = result +
                     DFS(graph, v, visited)

    return result
```

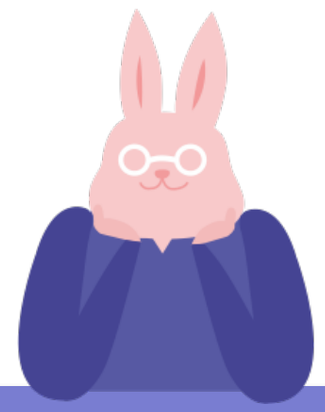
DFS(graph, 1, visited)

result = [1, 2, 3, 5, 6, 7, 4], v = 4



visited= [F, T, T, T, T, T, T, T]

graph[1] = [2, 4]
graph[2] = [3, 4]
graph[3] =[2, 5, 6, 7]
graph[4] = [1, 2]
graph[5] = [3]
graph[6] = [3, 7]
graph[7] = [3, 6]

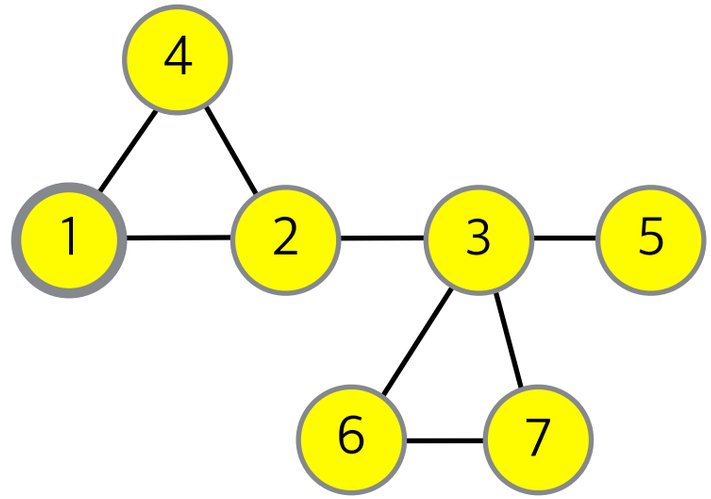


깊이 우선 탐색 (DFS) 구현

```
def DFS(graph, x, visited) :
    visited[x] = True
    result = [x]

    for v in graph[x] :
        if visited[v] == False :
            result = result +
                     DFS(graph, v, visited)

    return result
```

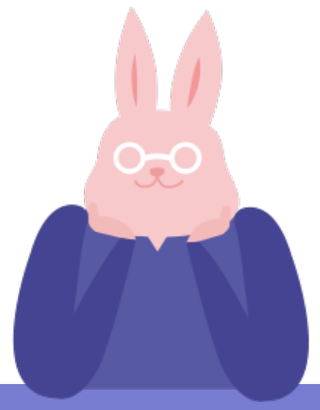
DFS(graph, 1, visited)

result = [1, 2, 3, 5, 6, 7, 4], v = 4



visited= [F, T, T, T, T, T, T, T]

graph[1] = [2, 4]
graph[2] = [3, 4]
graph[3] =[2, 5, 6, 7]
graph[4] = [1, 2]
graph[5] = [3]
graph[6] = [3, 7]
graph[7] = [3, 6]

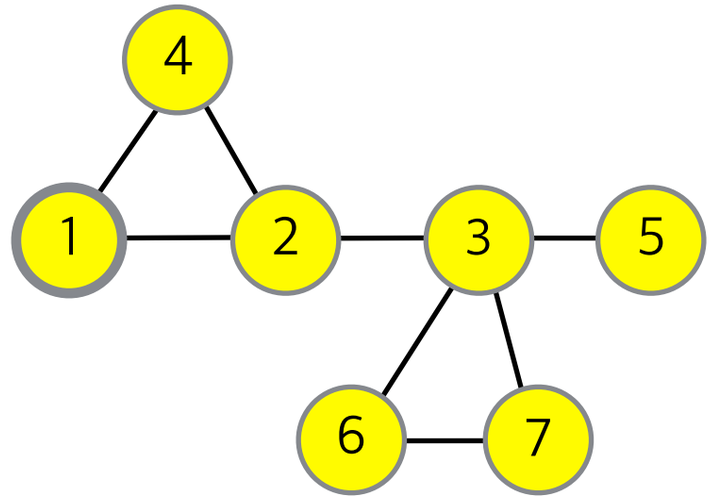


깊이 우선 탐색 (DFS) 구현

```
def DFS(graph, x, visited) :
      visited[x] = True
      result = [x]

      for v in graph[x] :
            if visited[v] == False :
                  result = result +
                           DFS(graph, v, visited)

      return result
```

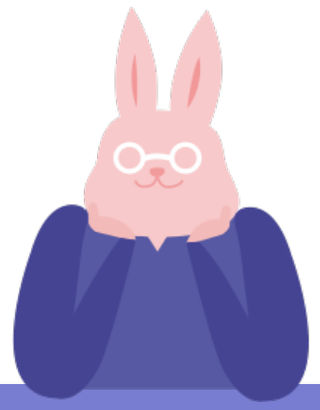
DFS(graph, 1, visited)

result = [1, 2, 3, 5, 6, 7, 4], v = 4



visited= [F, T, T, T, T, T, T, T]

graph[1] = [2, 4]
graph[2] = [3, 4]
graph[3] =[2, 5, 6, 7]
graph[4] = [1, 2]
graph[5] = [3]
graph[6] = [3, 7]
graph[7] = [3, 6]



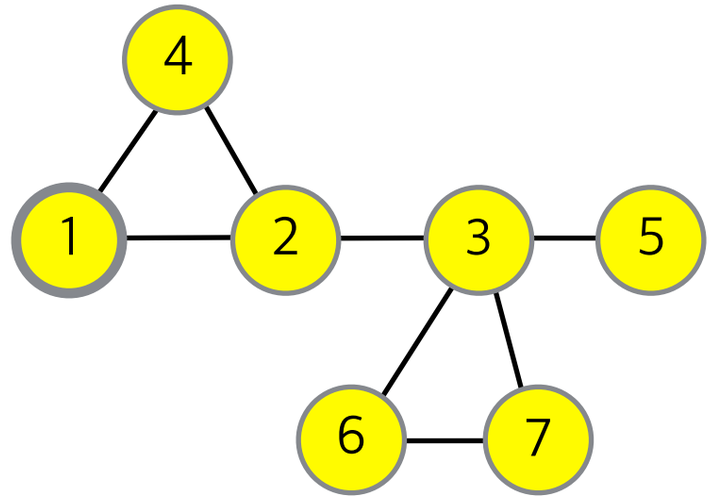
깊이 우선 탐색 (DFS) 구현

```
def DFS(graph, x, visited) :
      visited[x] = True
      result = [x]

      for v in graph[x] :
            if visited[v] == False :
                  result = result + DFS(graph, v, visited)

      return result
```

[1, 2, 3, 5, 6, 7, 4]

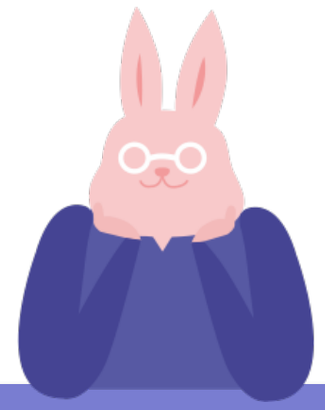
DFS(graph, 1, visited)

result = [1, 2, 3, 5, 6, 7, 4], v = 4



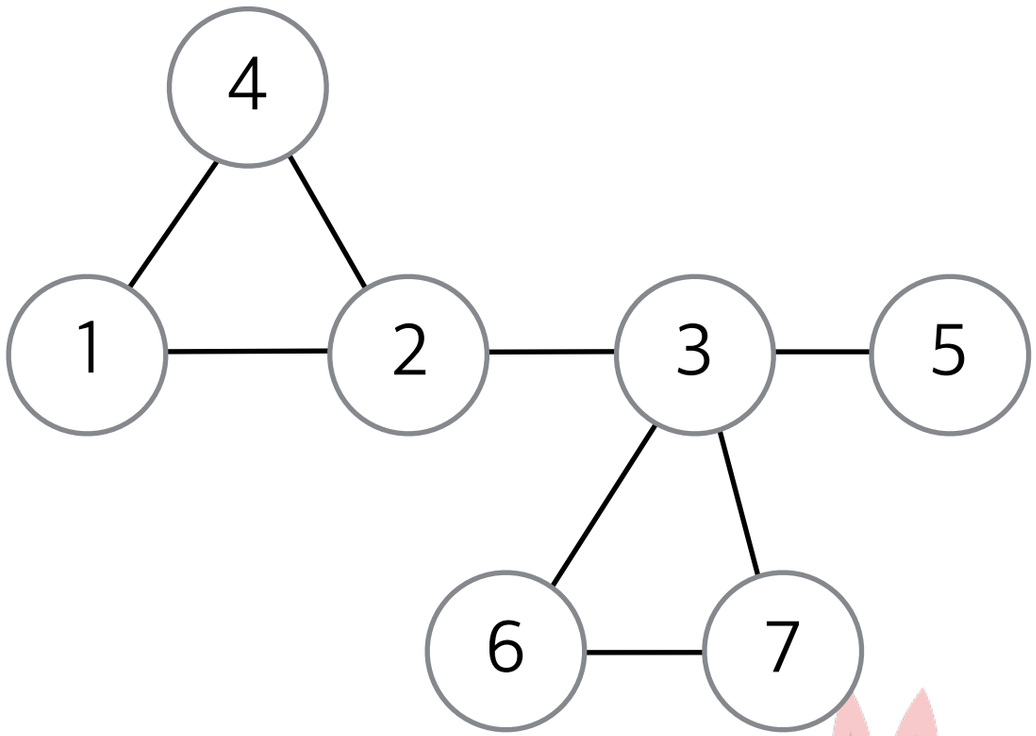
visited= [F, T, T, T, T, T, T, T]

graph[1] = [2, 4]
graph[2] = [3, 4]
graph[3] =[2, 5, 6, 7]
graph[4] = [1, 2]
graph[5] = [3]
graph[6] = [3, 7]
graph[7] = [3, 6]



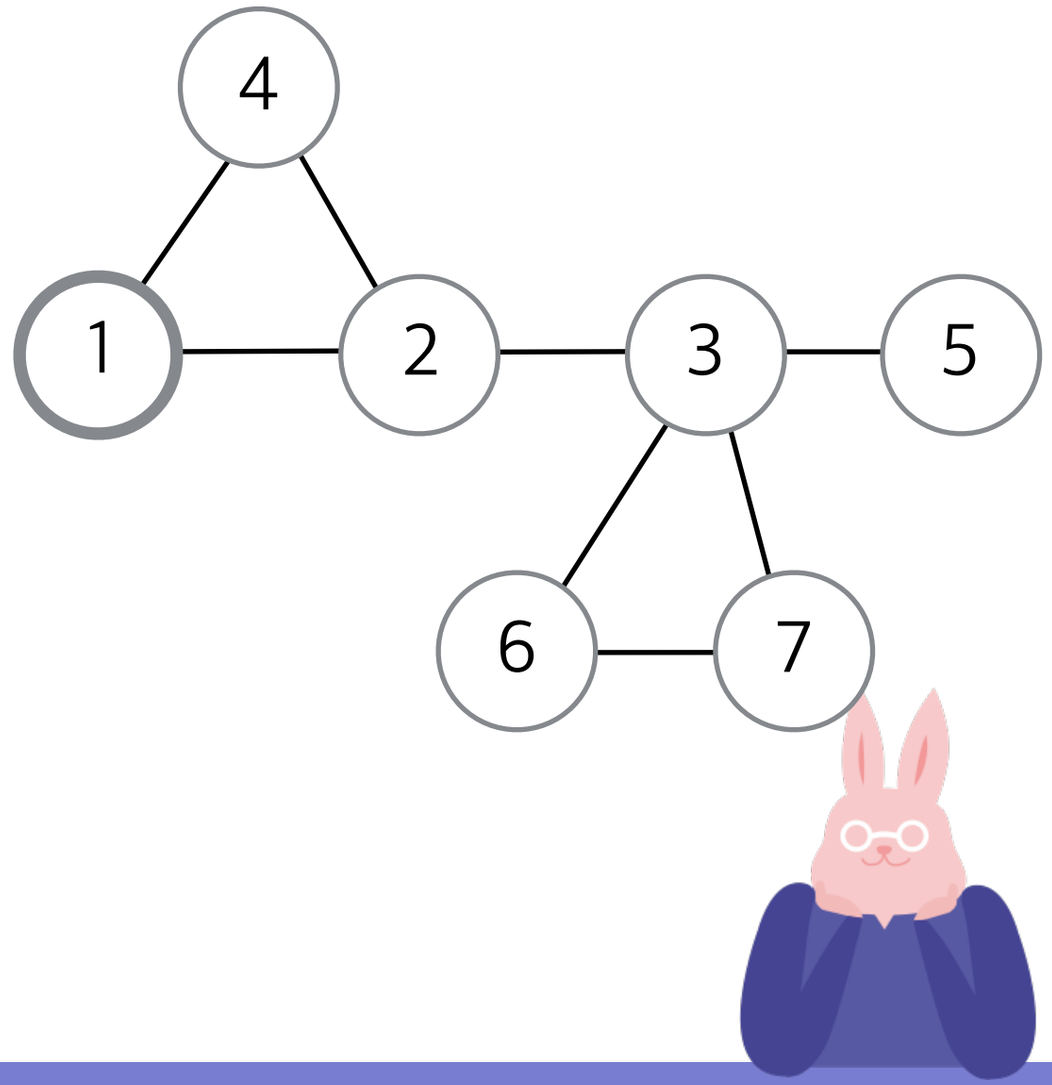
너비 우선 탐색 (BFS)

- 인접한 노드들을 우선 모두 색칠해 두고 뻗어나간다



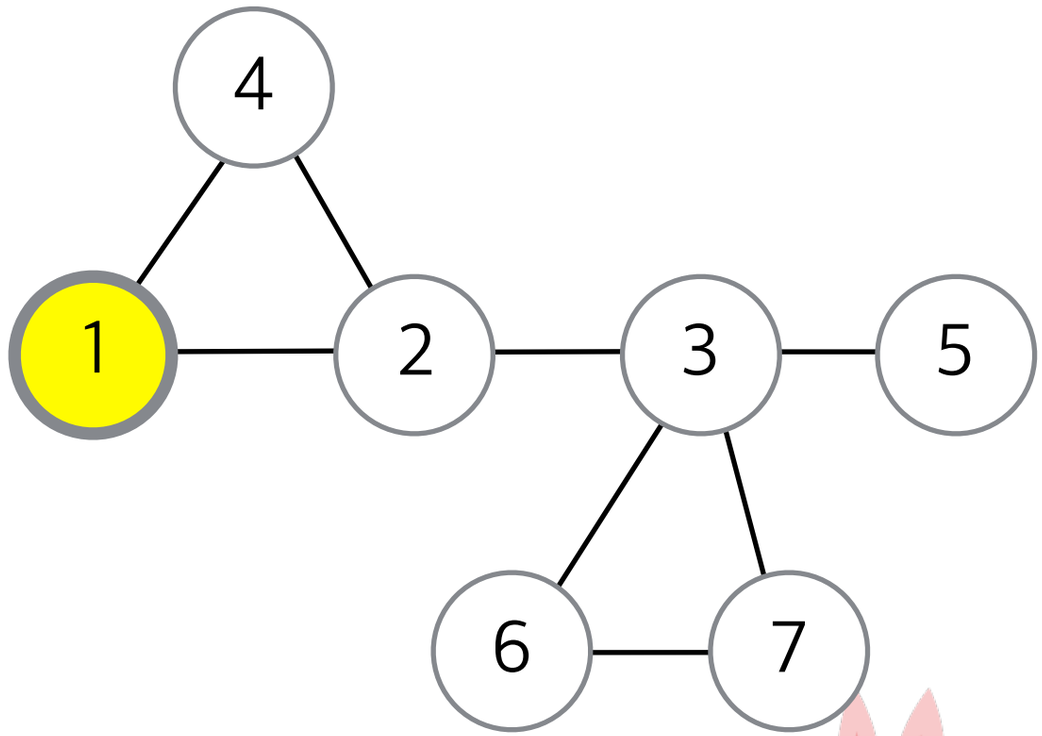
너비 우선 탐색 (BFS)

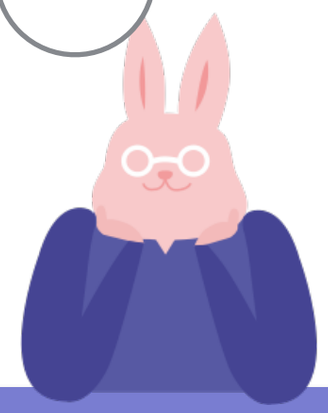
- 인접한 노드들을 우선 모두 색칠해 두고 뻗어나간다



너비 우선 탐색 (BFS)

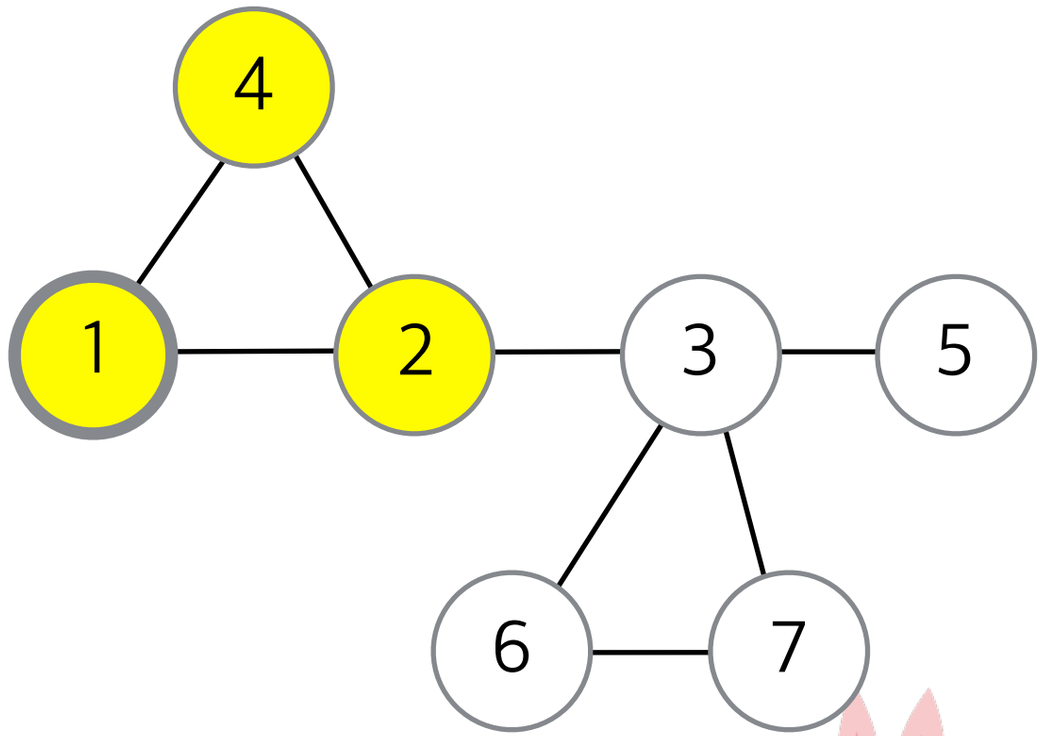
- 인접한 노드들을 우선 모두 색칠해 두고 뻗어나간다

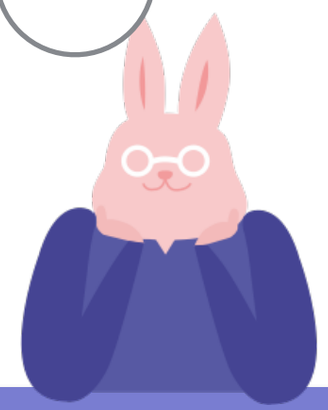




너비 우선 탐색 (BFS)

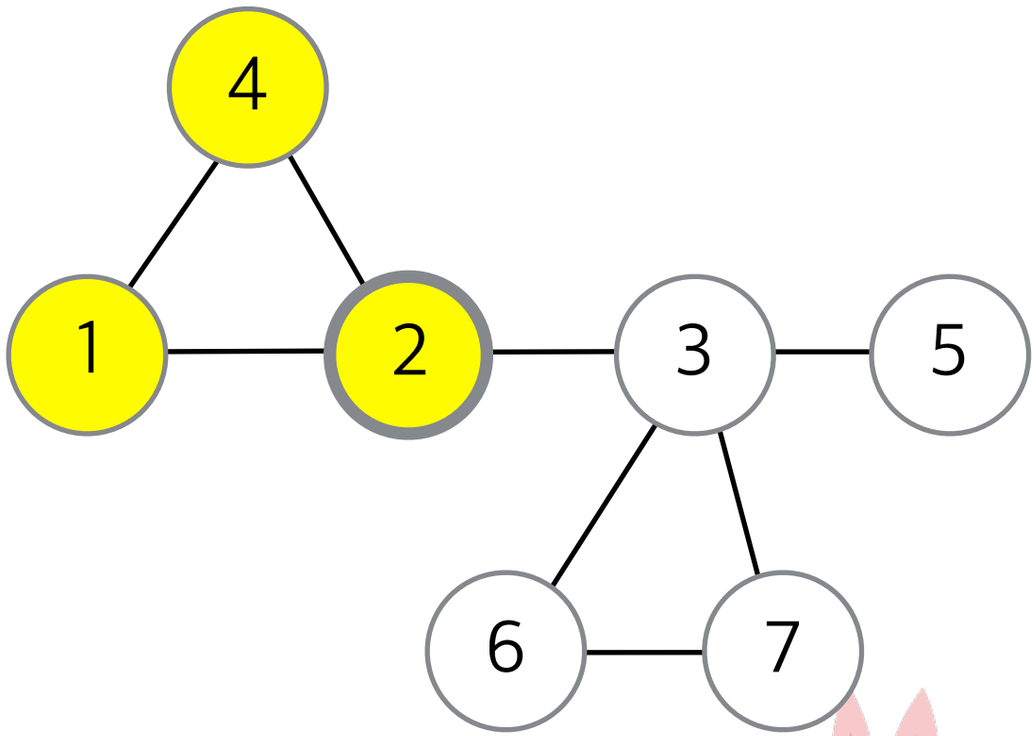
- 인접한 노드들을 우선 모두 색칠해 두고 뻗어나간다





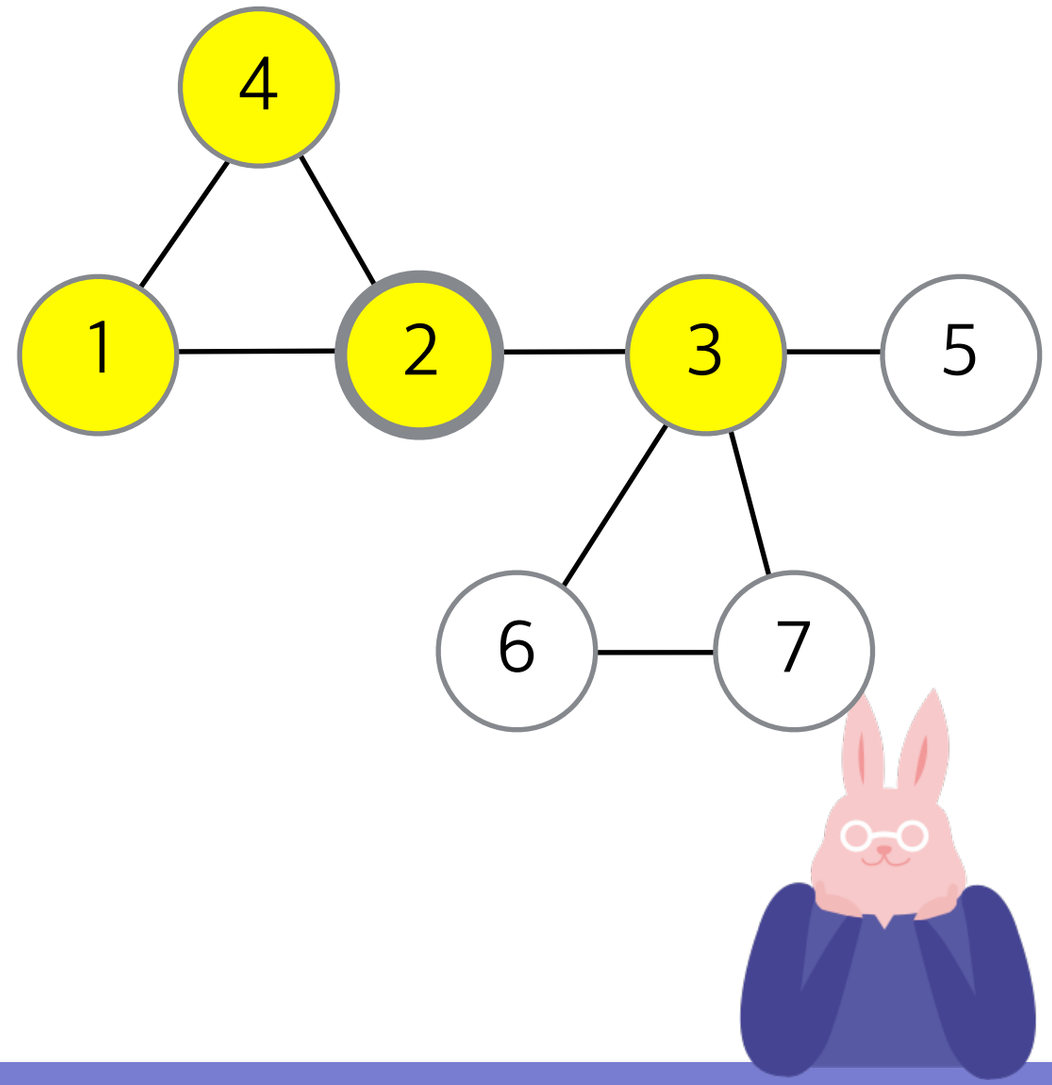
너비 우선 탐색 (BFS)

- 인접한 노드들을 우선 모두 색칠해 두고 뻗어나간다



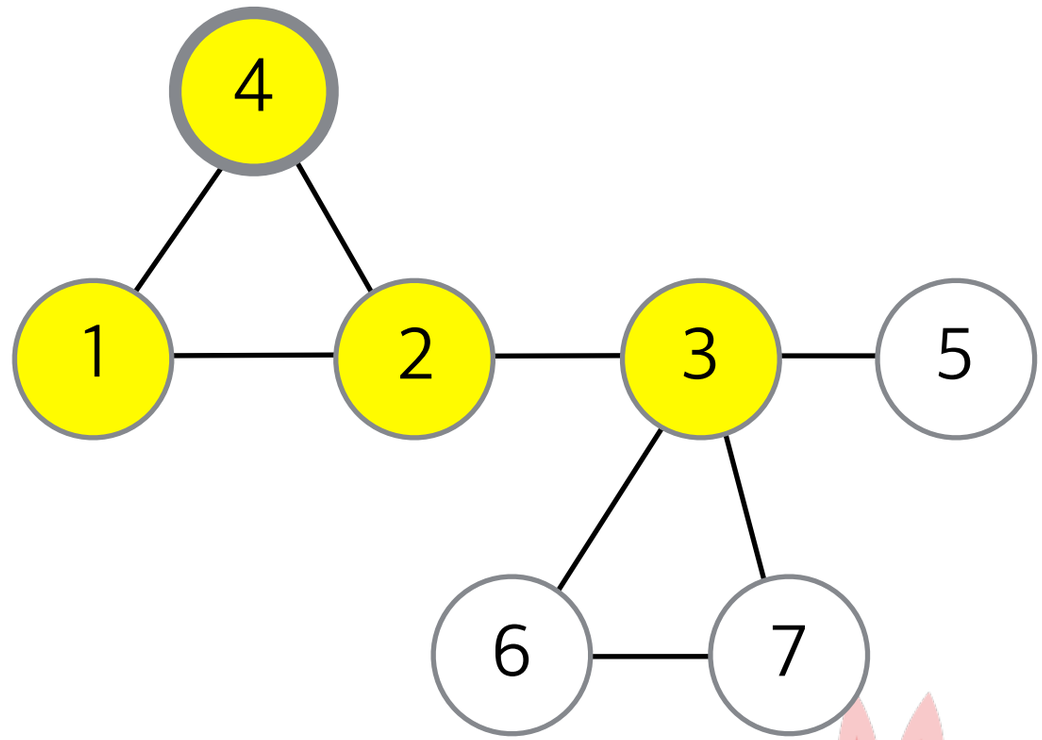
너비 우선 탐색 (BFS)

- 인접한 노드들을 우선 모두 색칠해 두고 뻗어나간다



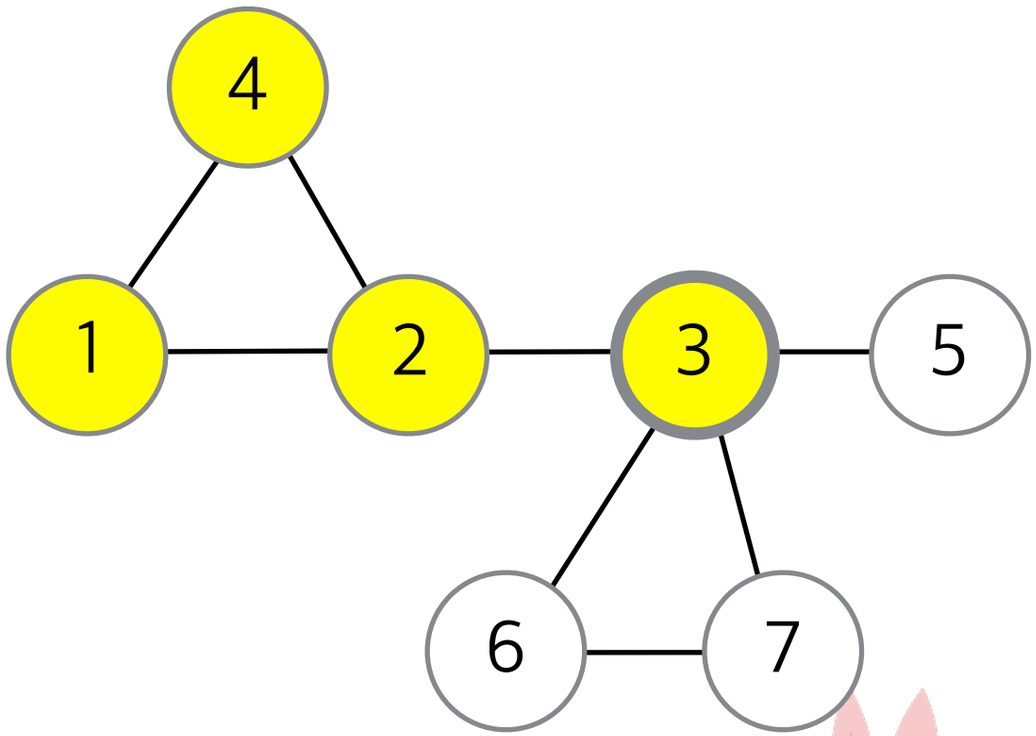
너비 우선 탐색 (BFS)

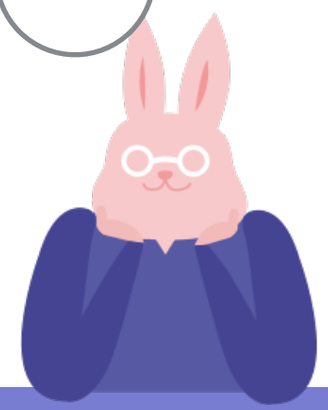
- 인접한 노드들을 우선 모두 색칠해 두고 뻗어나간다



너비 우선 탐색 (BFS)

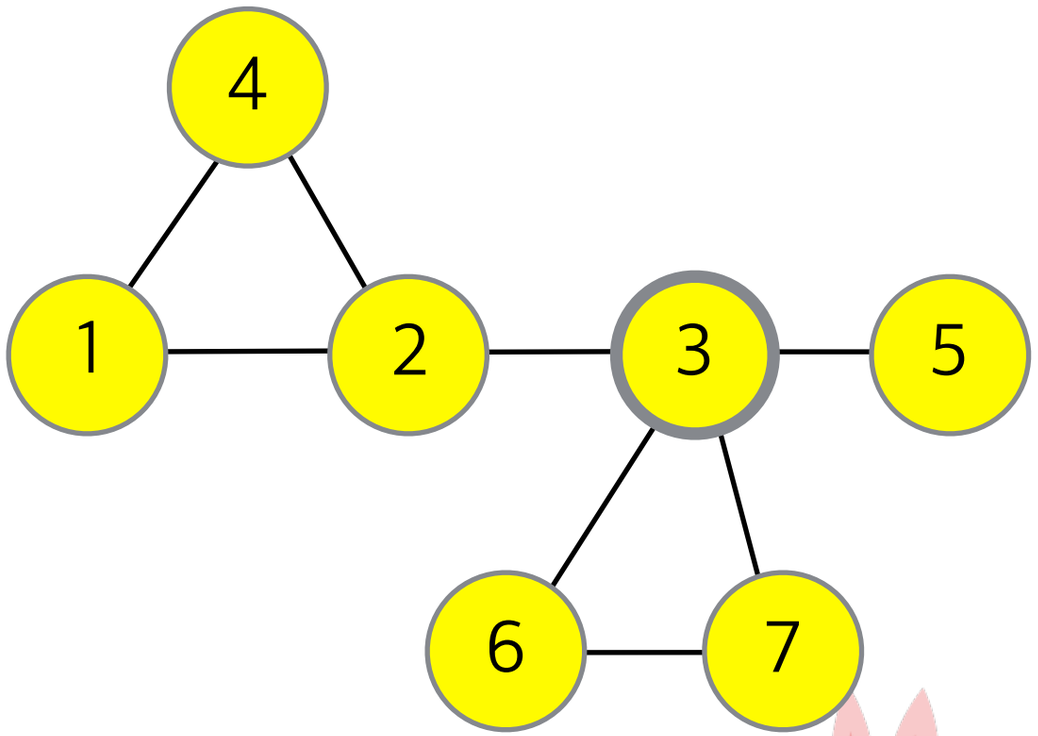
- 인접한 노드들을 우선 모두 색칠해 두고 뻗어나간다

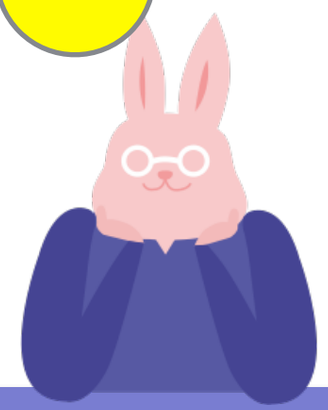




너비 우선 탐색 (BFS)

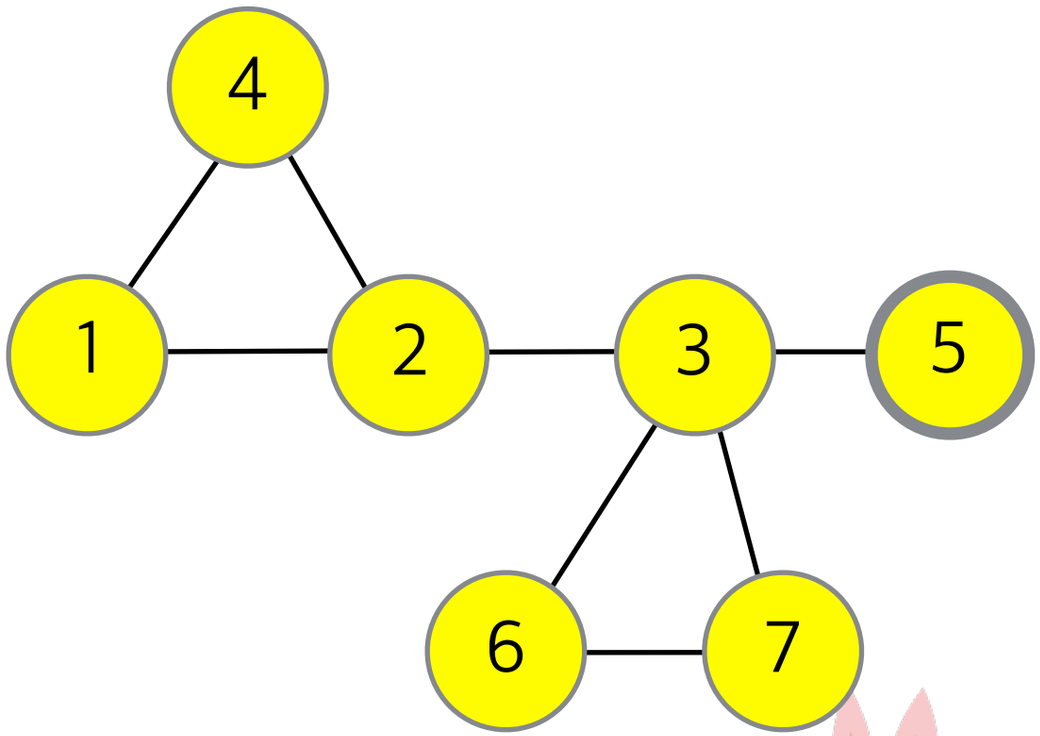
- 인접한 노드들을 우선 모두 색칠해 두고 뻗어나간다

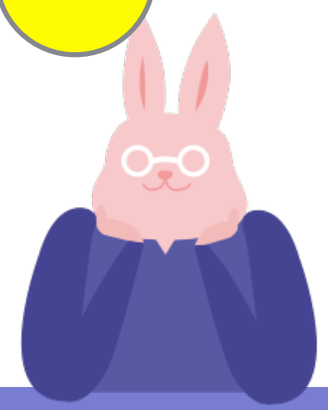




너비 우선 탐색 (BFS)

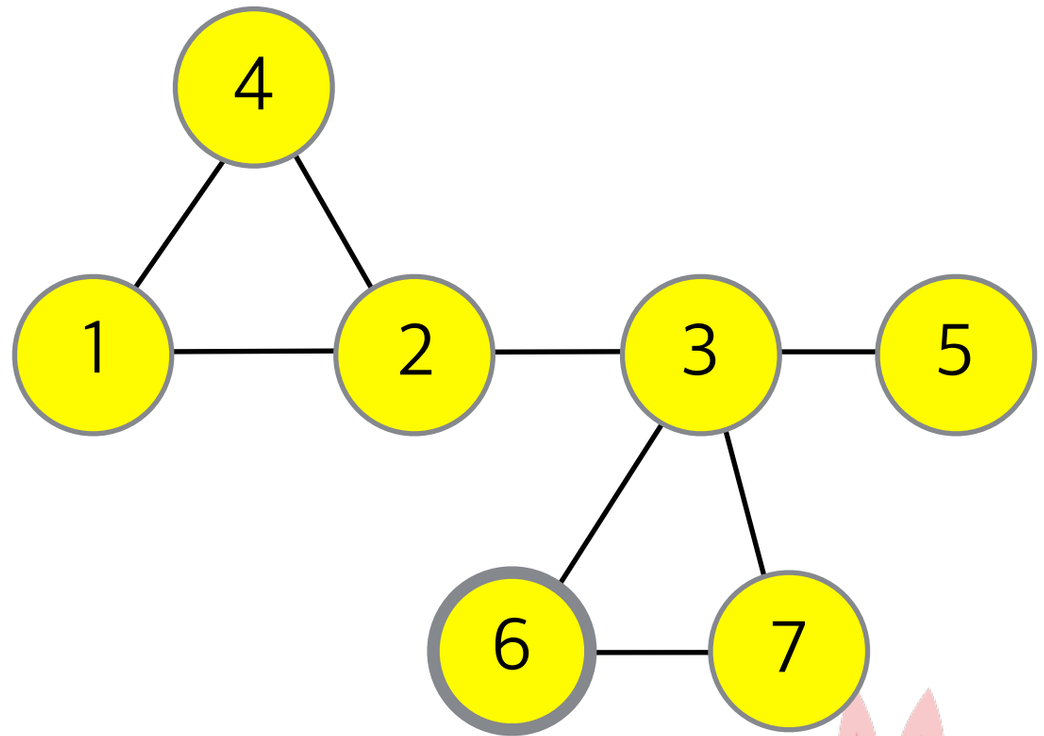
- 인접한 노드들을 우선 모두 색칠해 두고 뻗어나간다

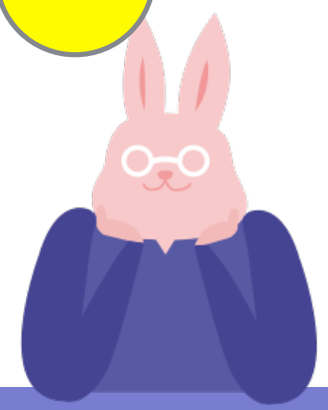




너비 우선 탐색 (BFS)

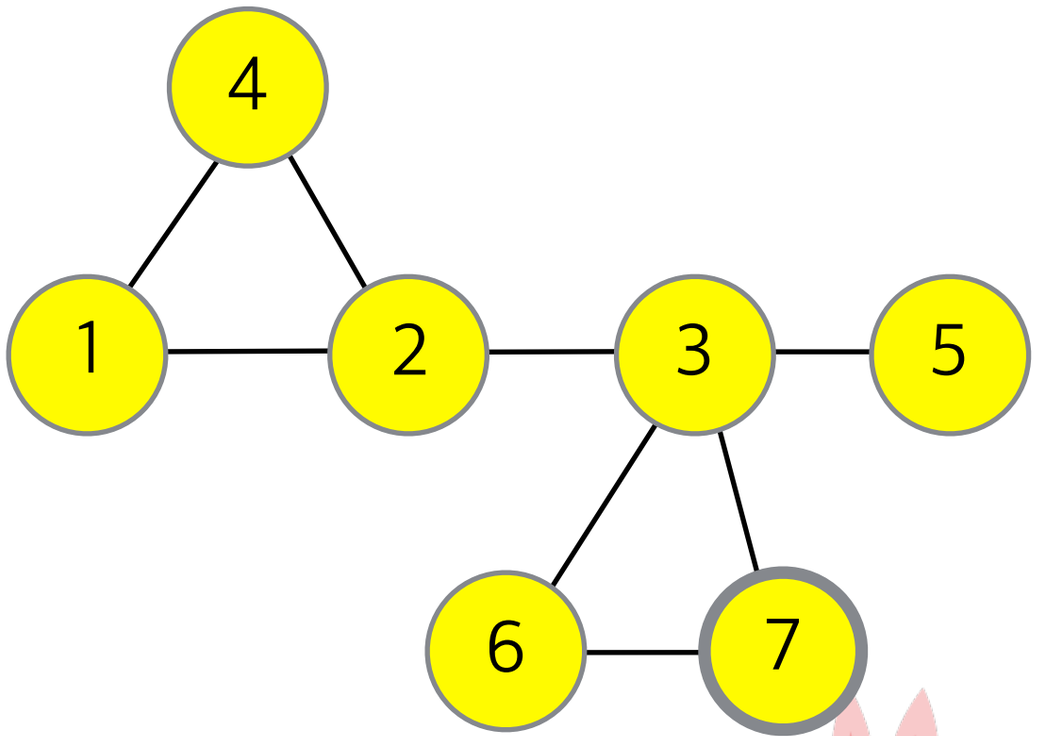
- 인접한 노드들을 우선 모두 색칠해 두고 뻗어나간다

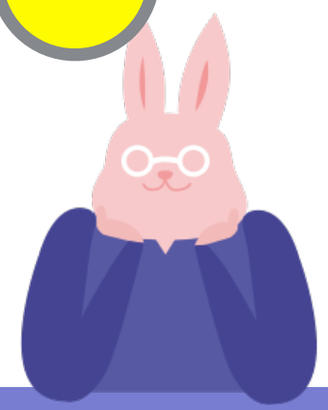




너비 우선 탐색 (BFS)

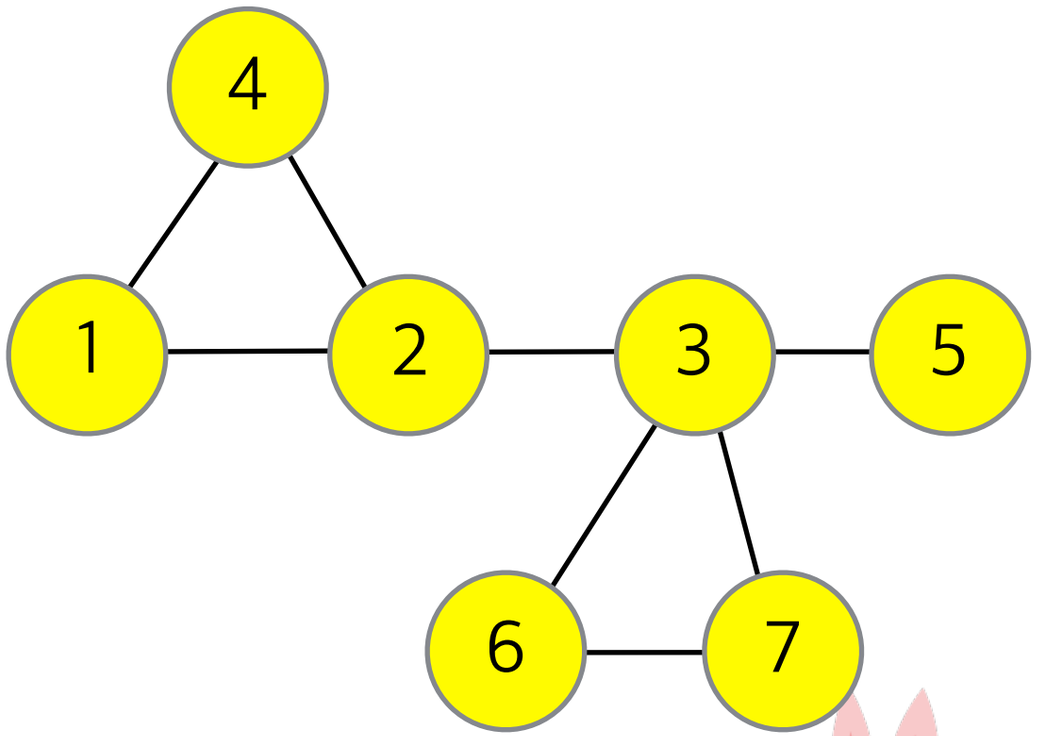
- 인접한 노드들을 우선 모두 색칠해 두고 뻗어나간다

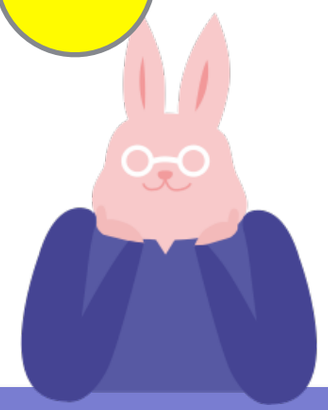




너비 우선 탐색 (BFS)

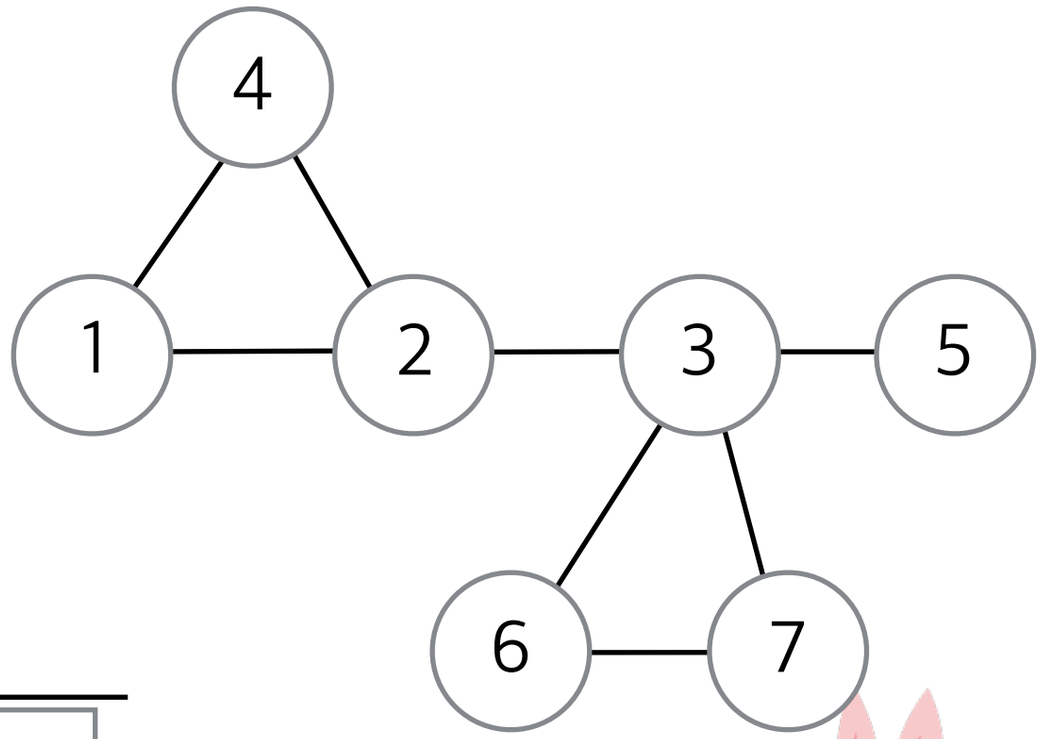
- 인접한 노드들을 우선 모두 색칠해 두고 뻗어나간다

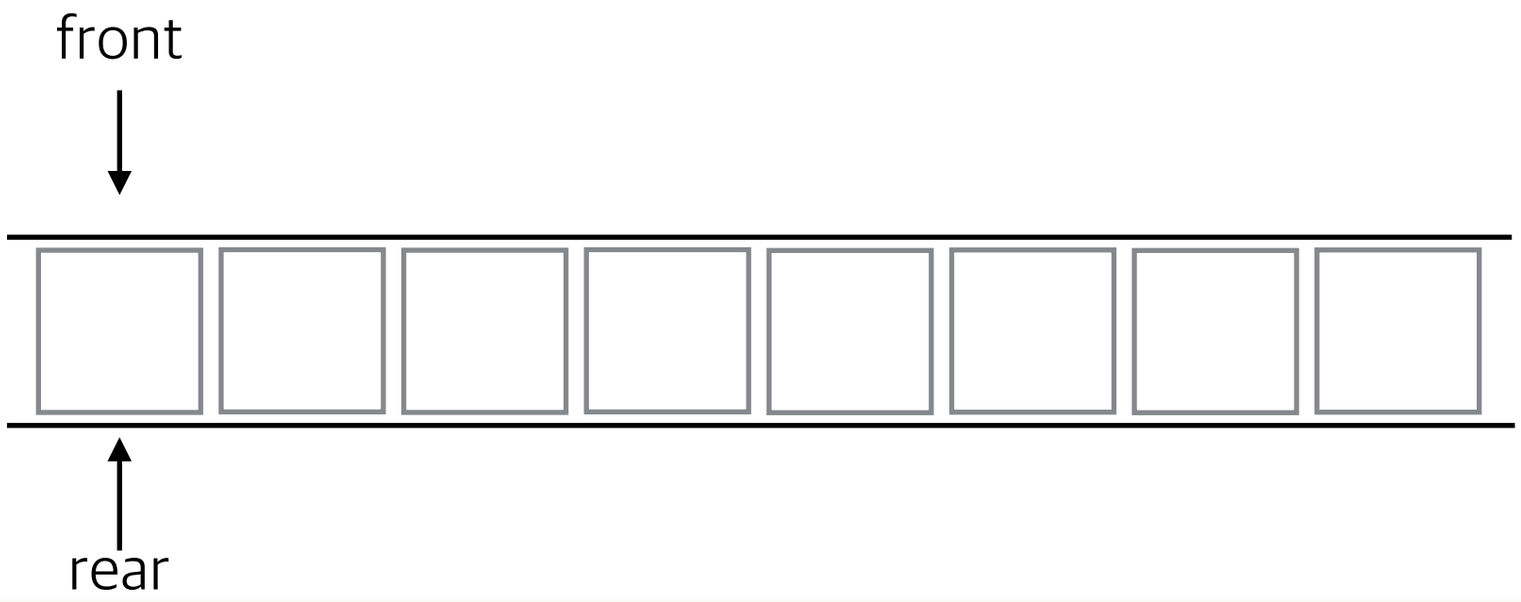


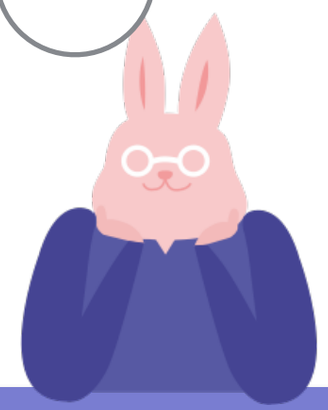


너비 우선 탐색 (BFS)

- 인접한 노드들을 우선 모두 색칠해 두고 뻗어나간다
  1. Node 선택
  2. 인접한 노드 색칠

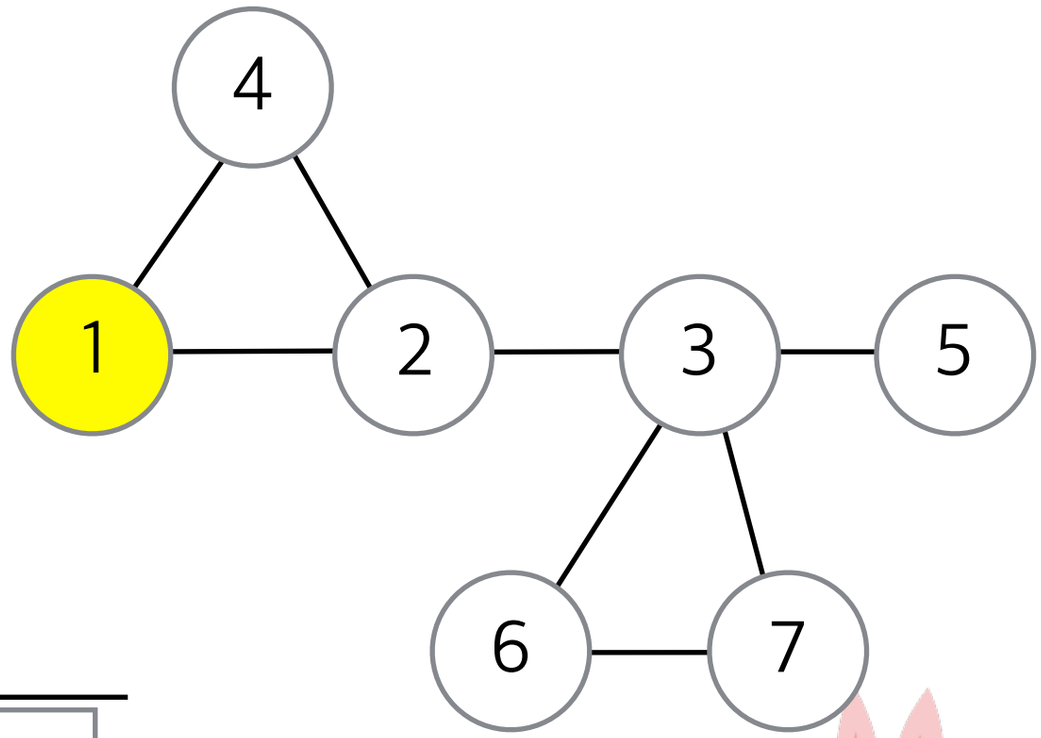




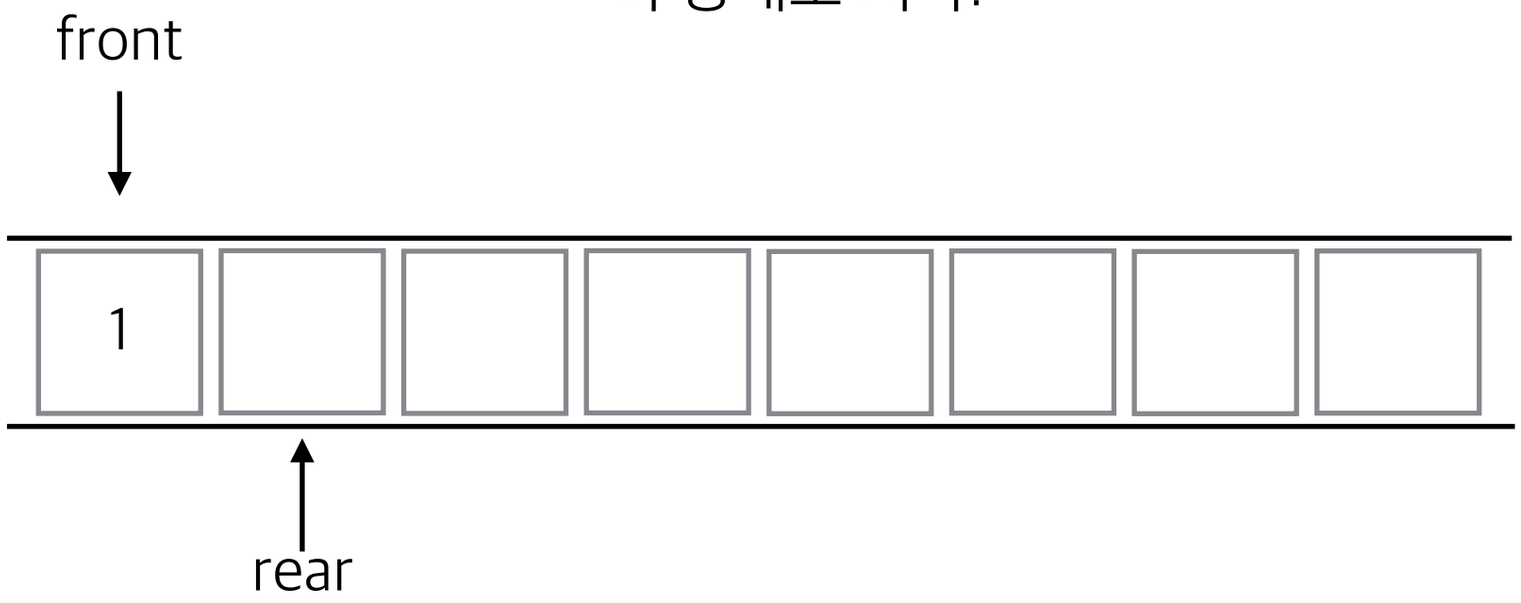


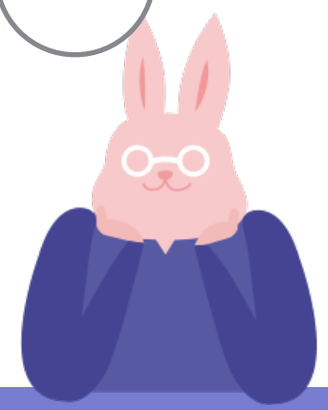
너비 우선 탐색 (BFS)

- 인접한 노드들을 우선 모두 색칠해 두고 뻗어나간다
  1. Node 선택
  2. 인접한 노드 색칠



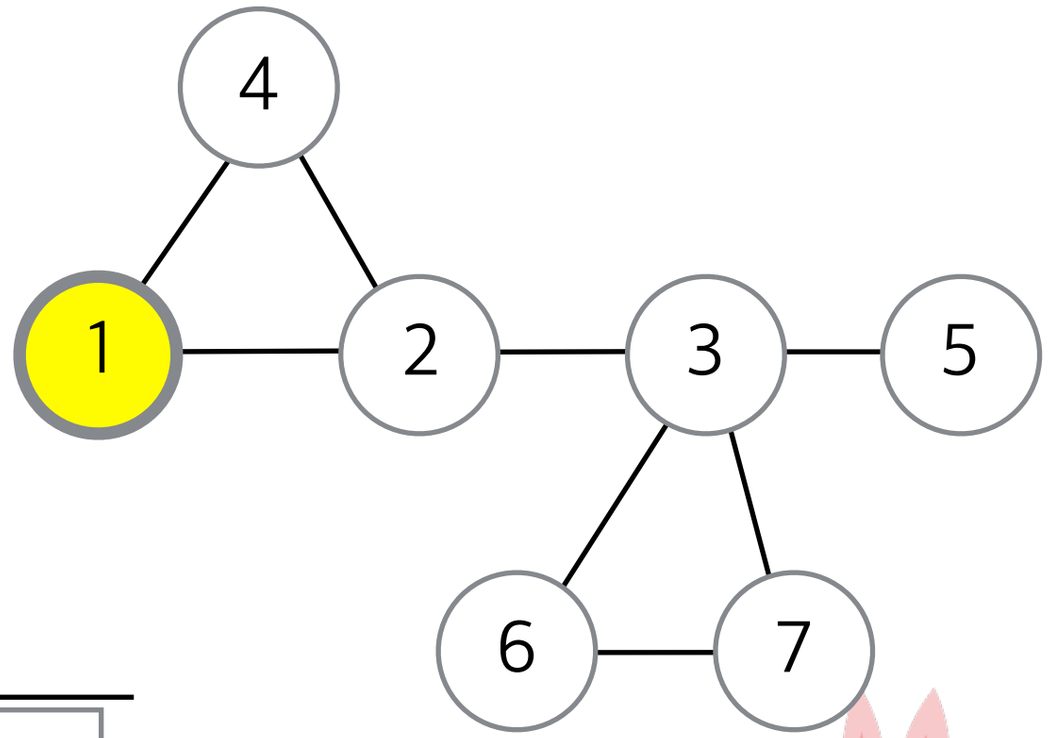
이 상태로 시작!

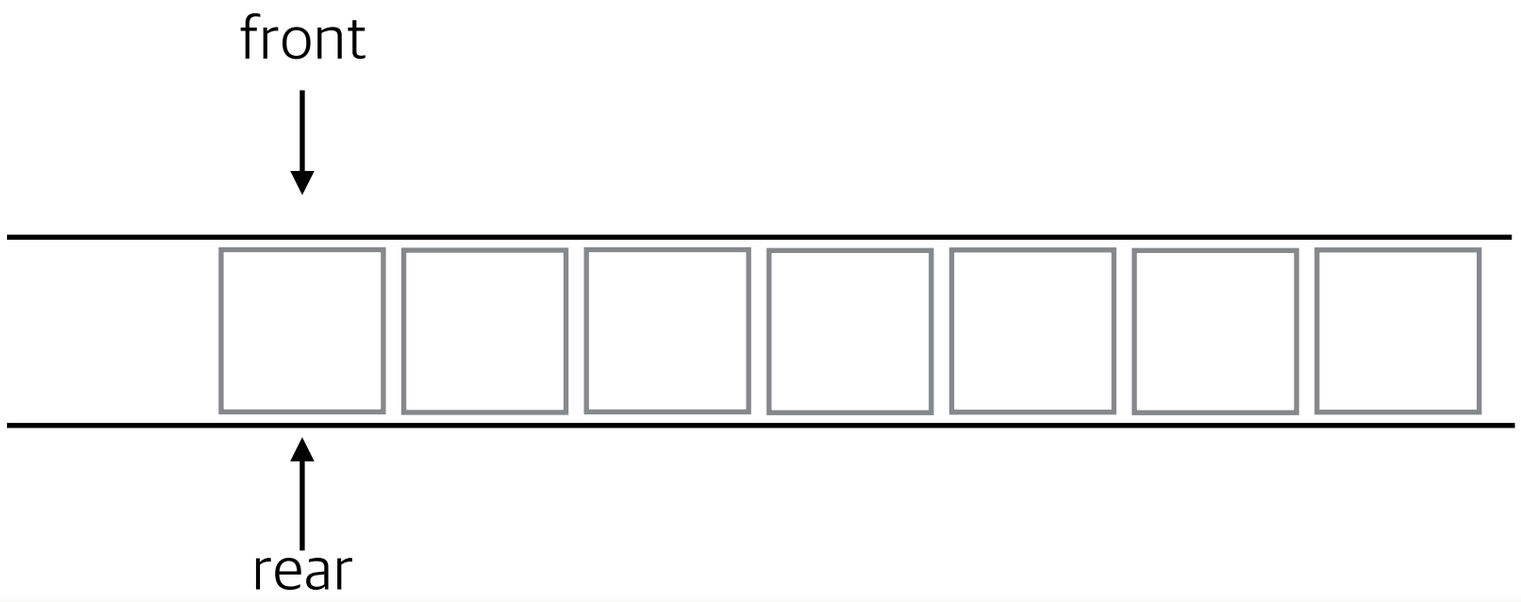




너비 우선 탐색 (BFS)

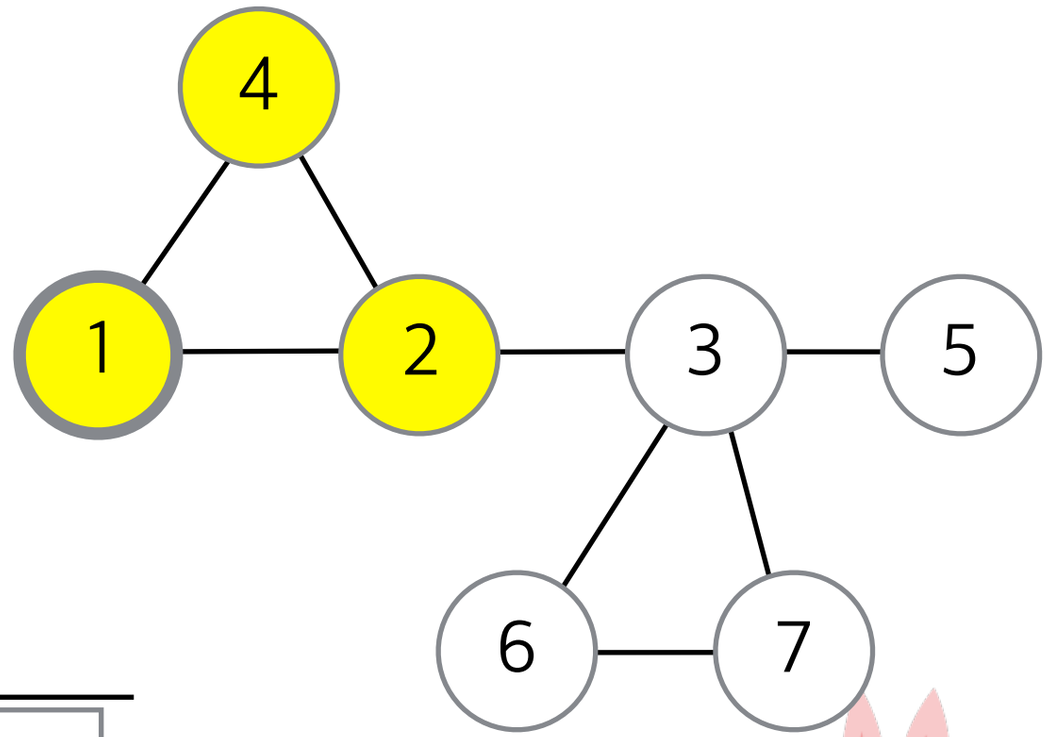
- 인접한 노드들을 우선 모두 색칠해 두고 뻗어나간다
  1. **Node 선택**
  2. 인접한 노드 색칠

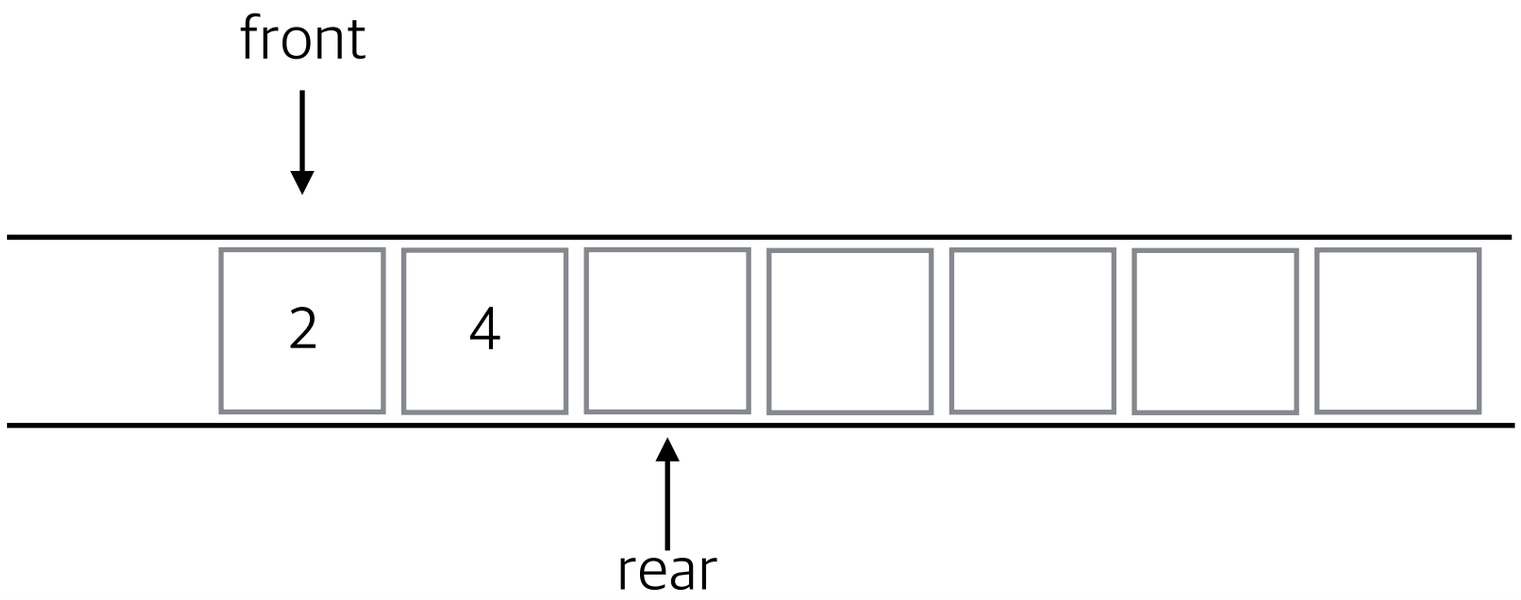


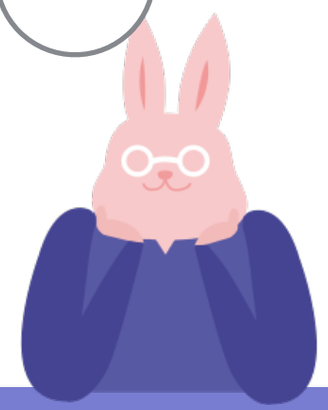


너비 우선 탐색 (BFS)

- 인접한 노드들을 우선 모두 색칠해 두고 뻗어나간다
  1. Node 선택
  2. **인접한 노드 색칠**

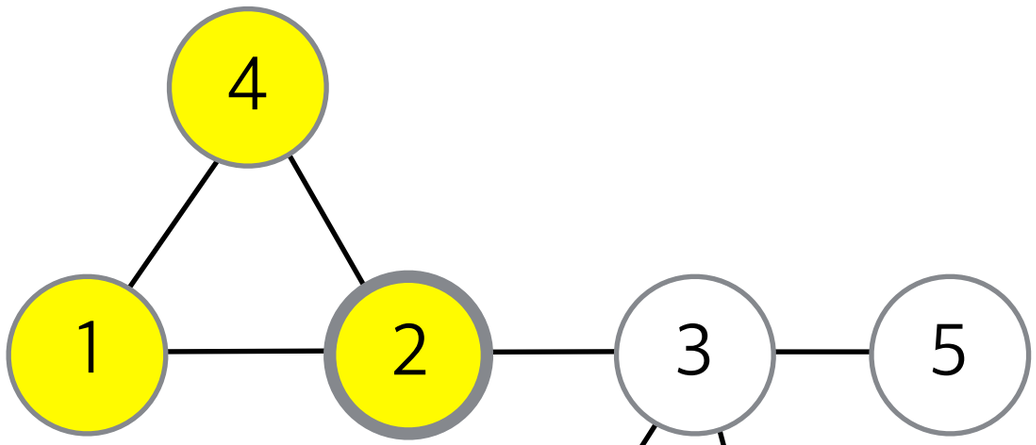


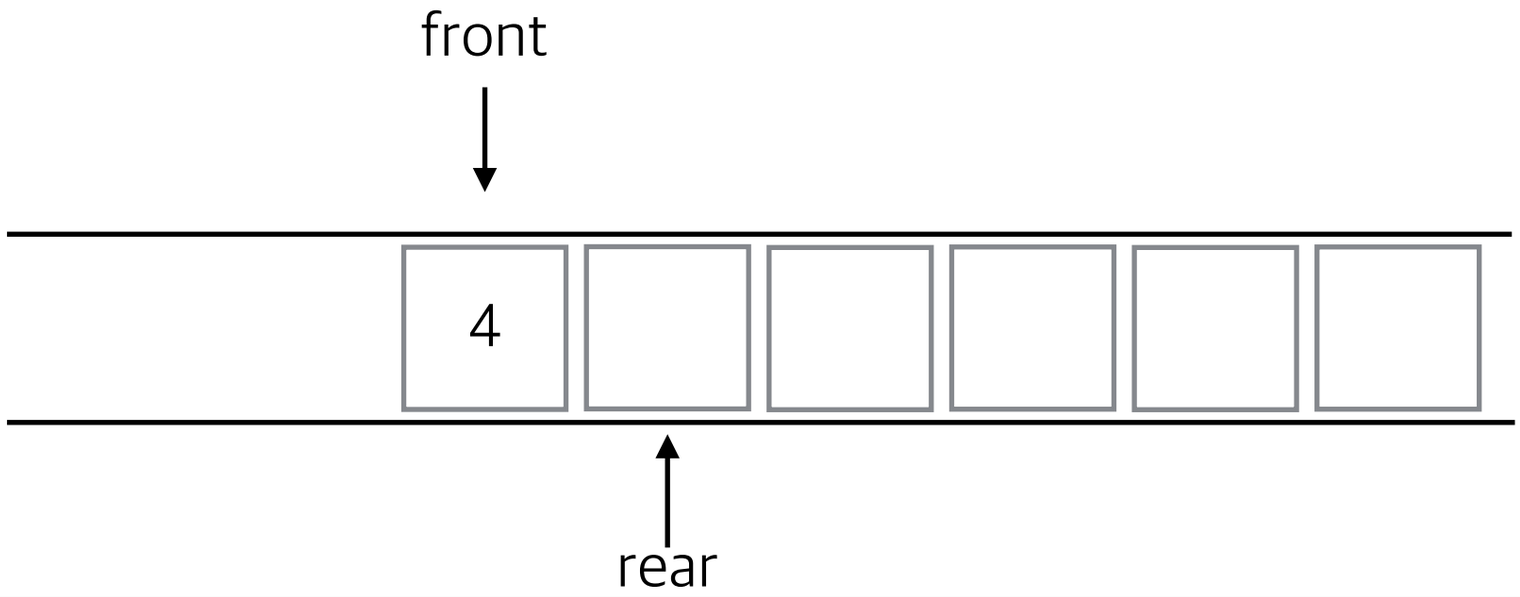


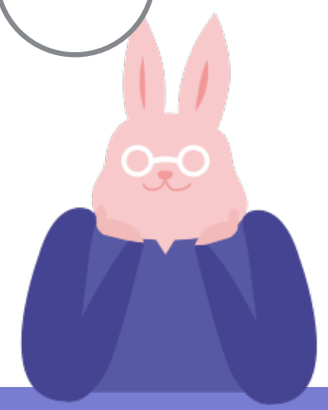


너비 우선 탐색 (BFS)

- 인접한 노드들을 우선 모두 색칠해 두고 뻗어나간다
  1. **Node 선택**
  2. 인접한 노드 색칠

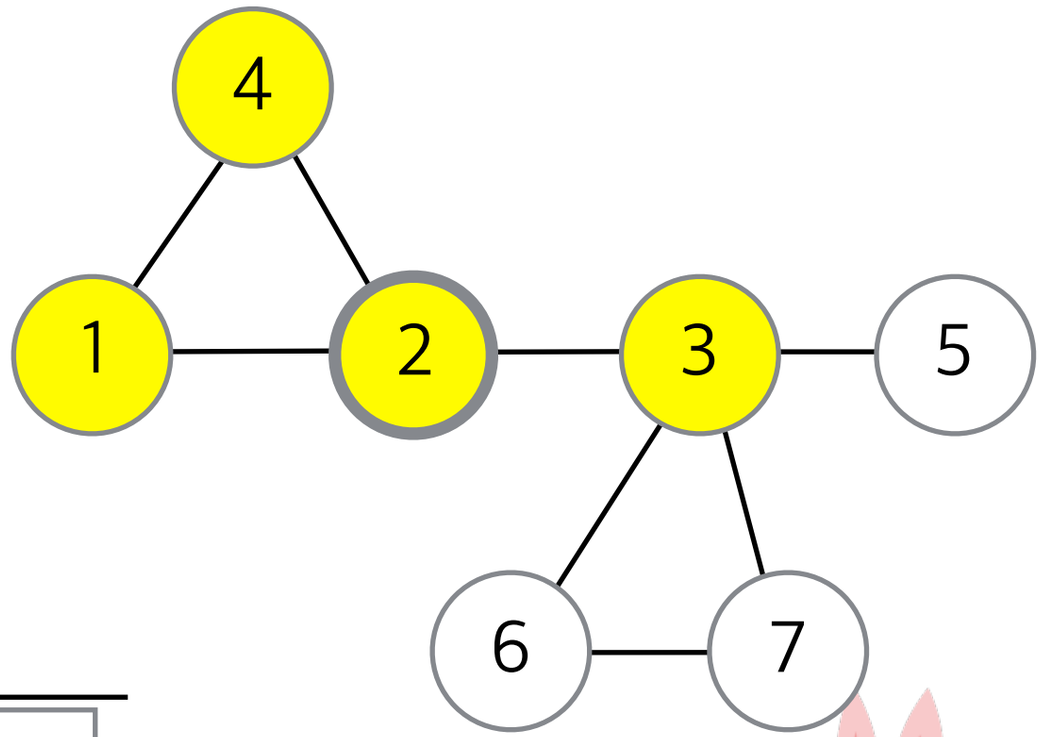


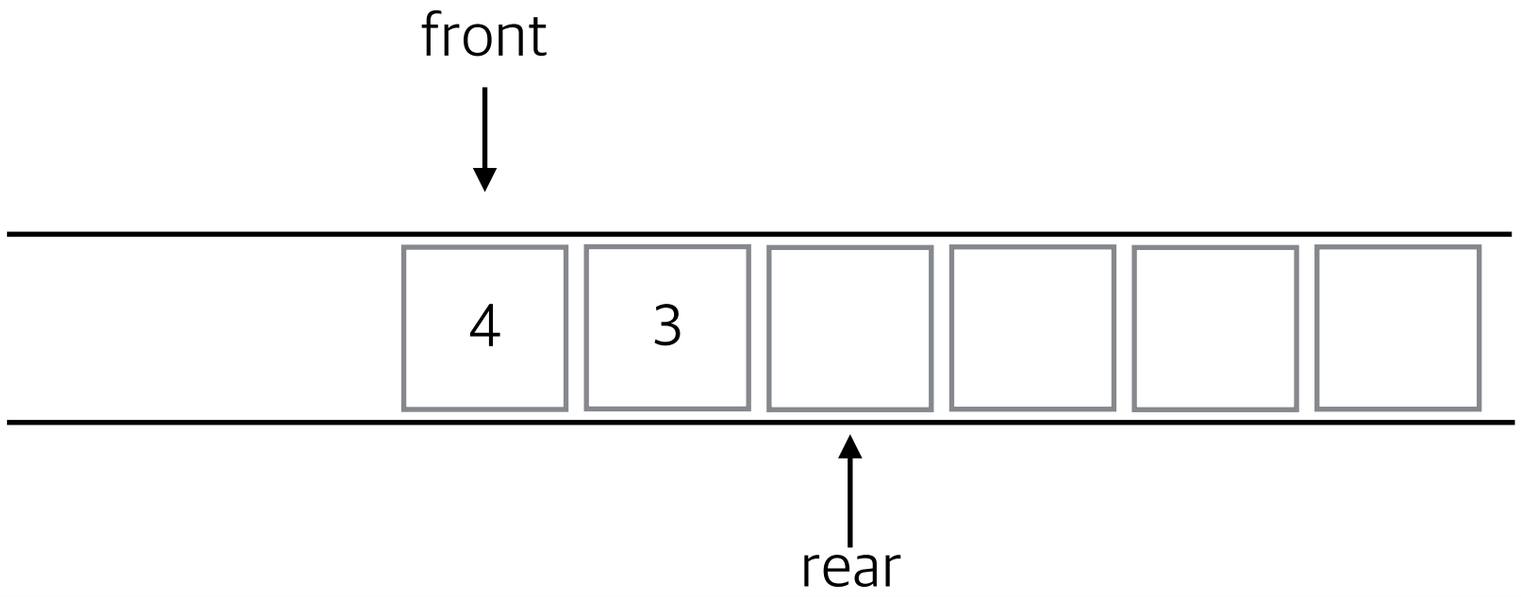




너비 우선 탐색 (BFS)

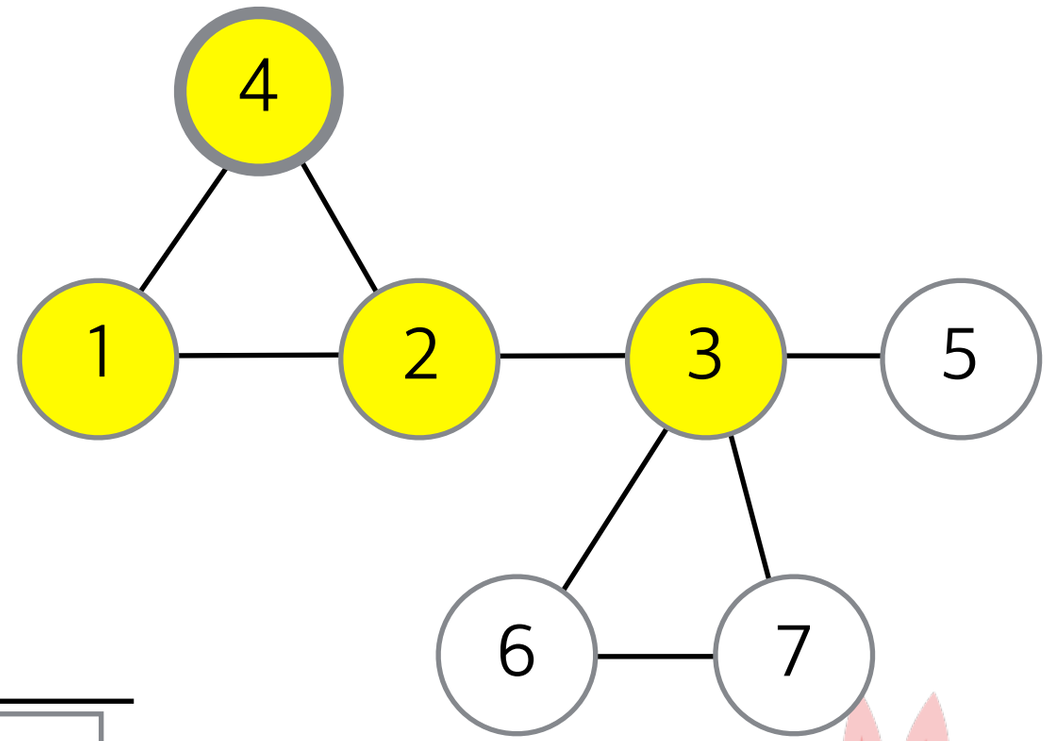
- 인접한 노드들을 우선 모두 색칠해 두고 뻗어나간다
  1. Node 선택
  2. **인접한 노드 색칠**

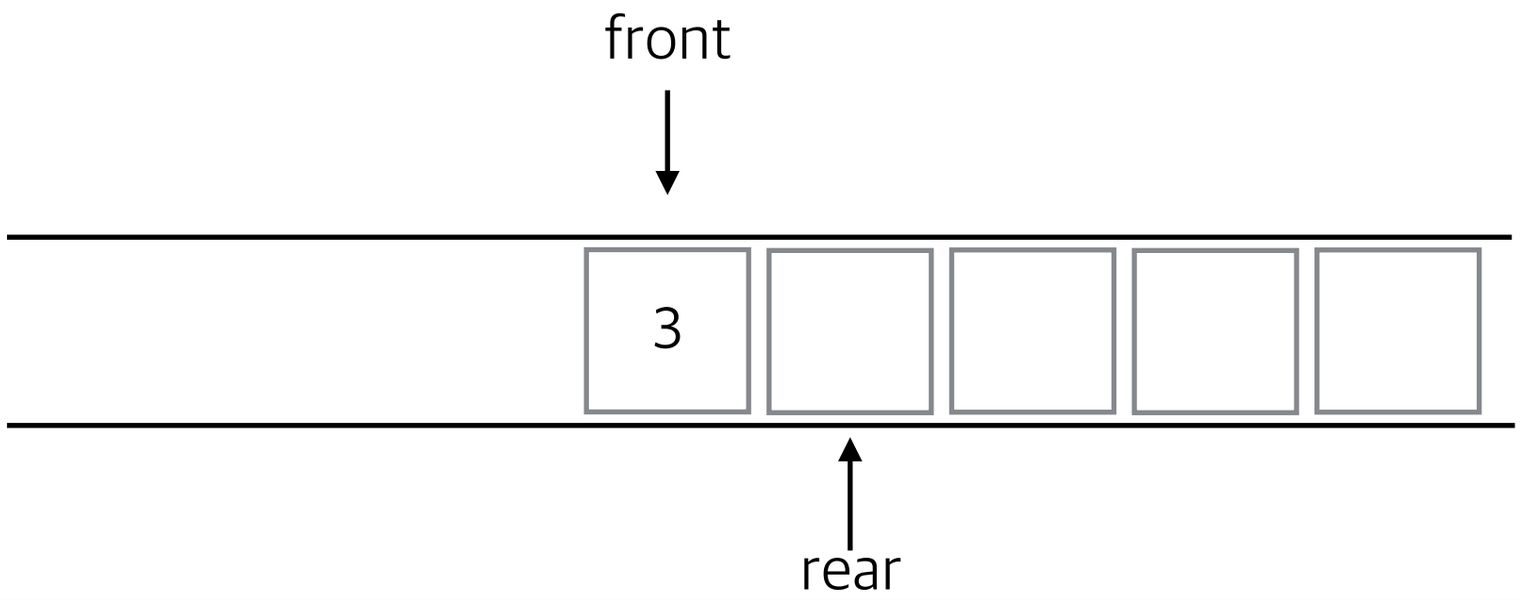




너비 우선 탐색 (BFS)

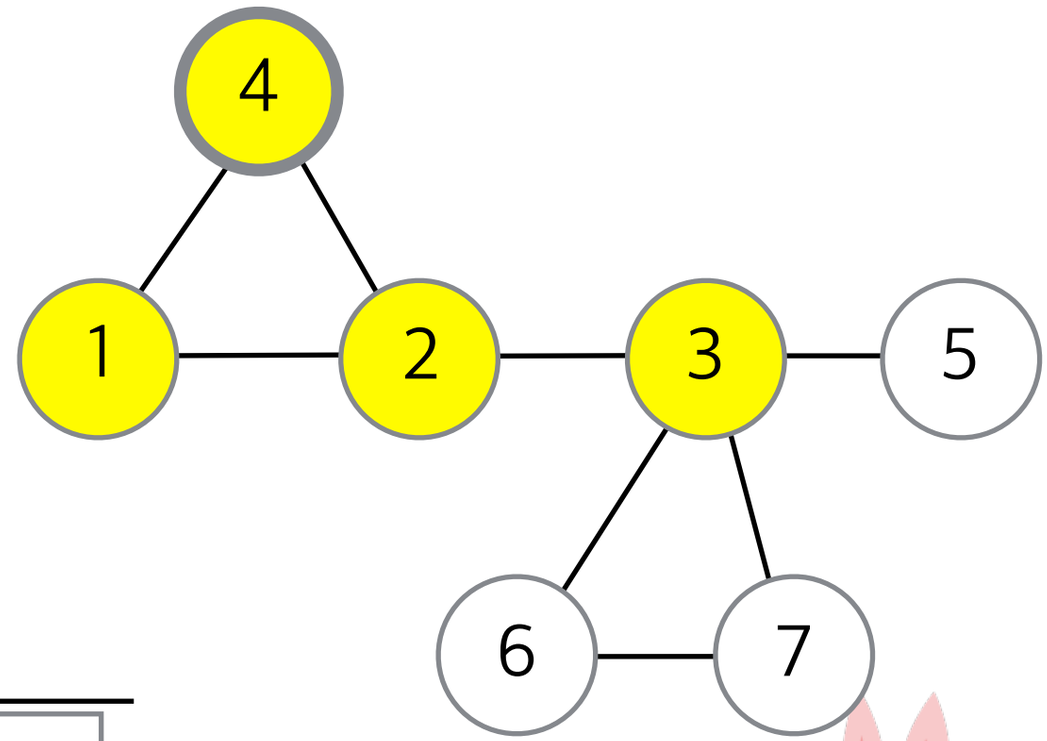
- 인접한 노드들을 우선 모두 색칠해 두고 뻗어나간다
  1. **Node 선택**
  2. 인접한 노드 색칠

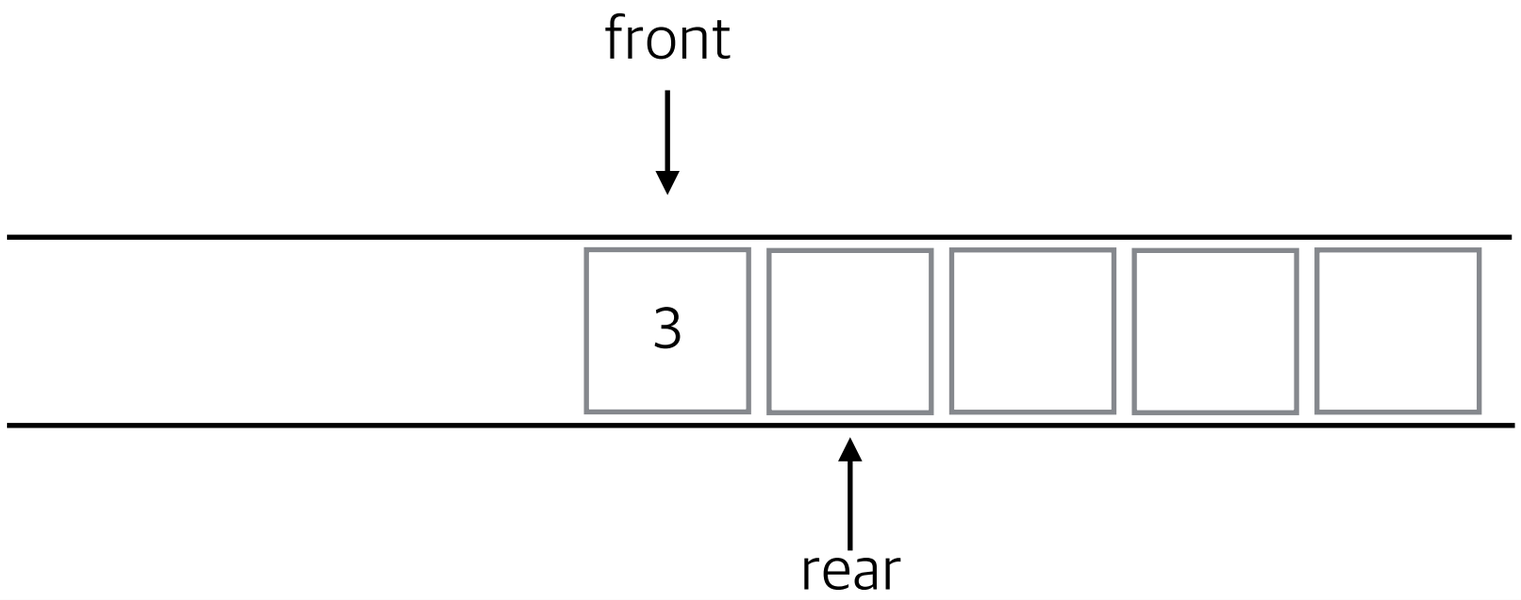




너비 우선 탐색 (BFS)

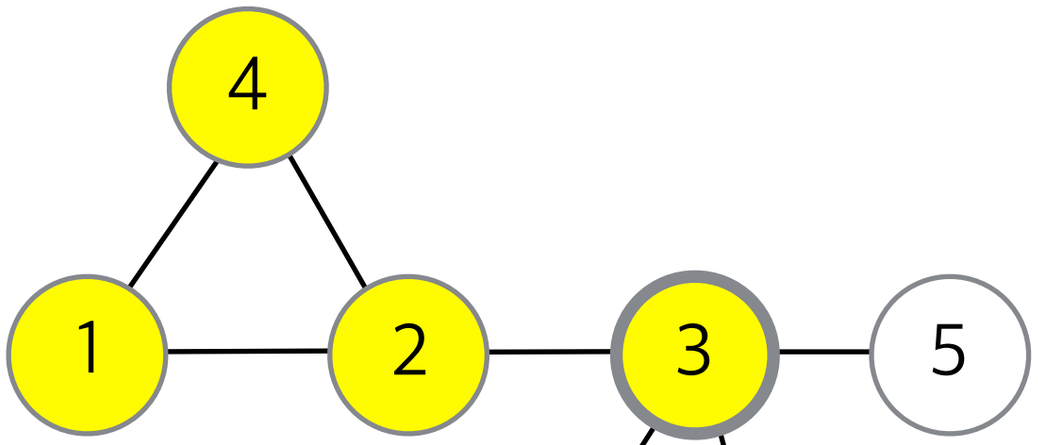
- 인접한 노드들을 우선 모두 색칠해 두고 뻗어나간다
  1. Node 선택
  2. **인접한 노드 색칠**

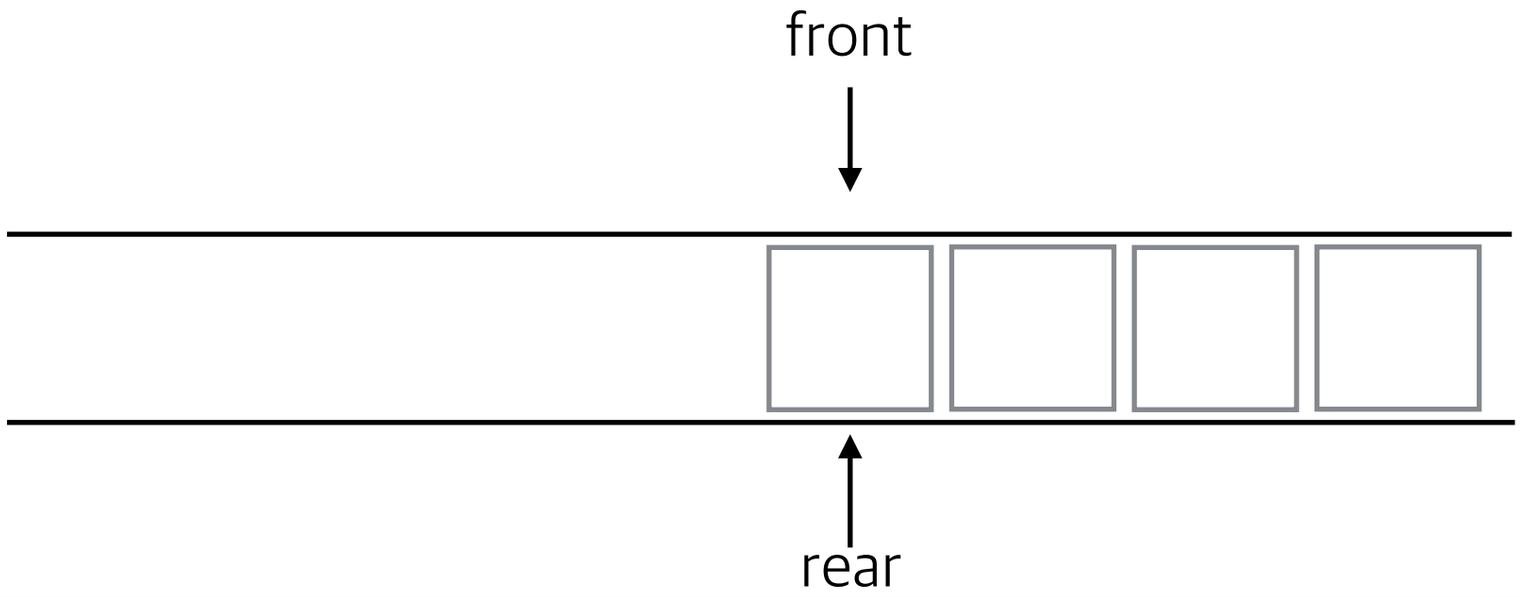


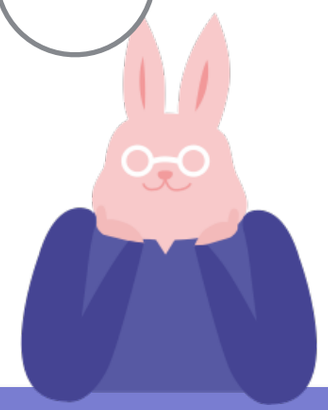


너비 우선 탐색 (BFS)

- 인접한 노드들을 우선 모두 색칠해 두고 뻗어나간다
  1. **Node 선택**
  2. 인접한 노드 색칠

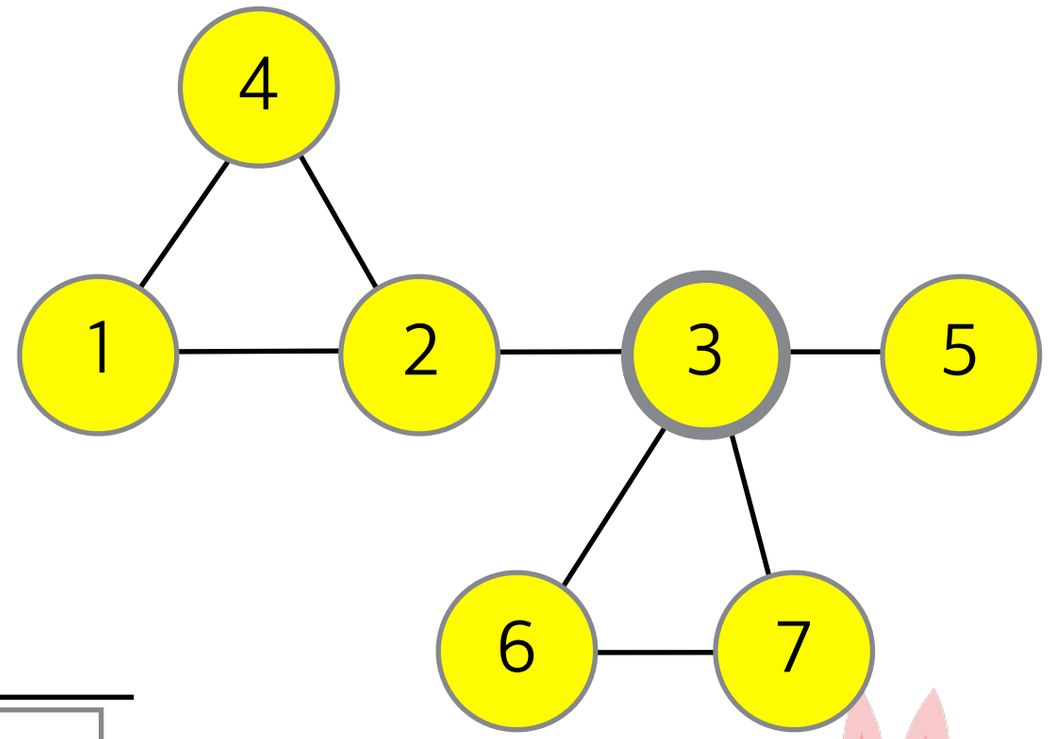


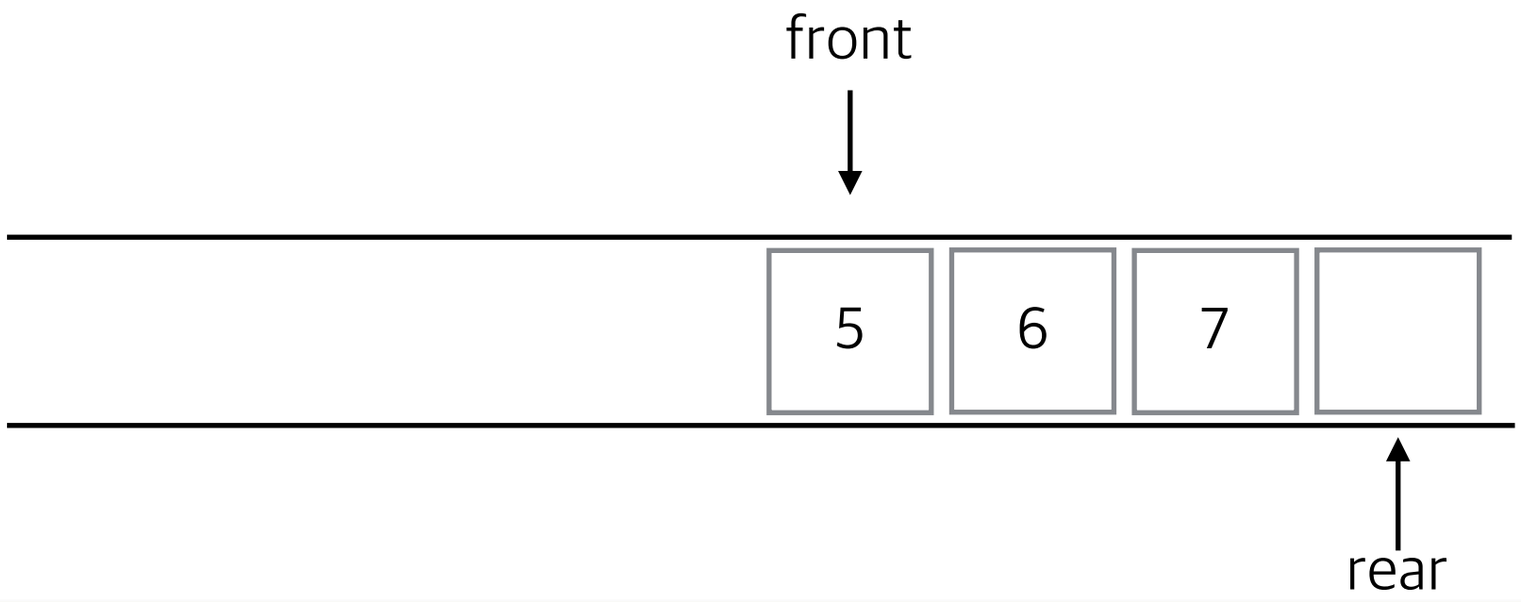


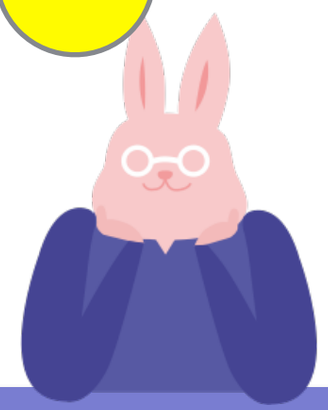


너비 우선 탐색 (BFS)

- 인접한 노드들을 우선 모두 색칠해 두고 뻗어나간다
  1. Node 선택
  2. **인접한 노드 색칠**

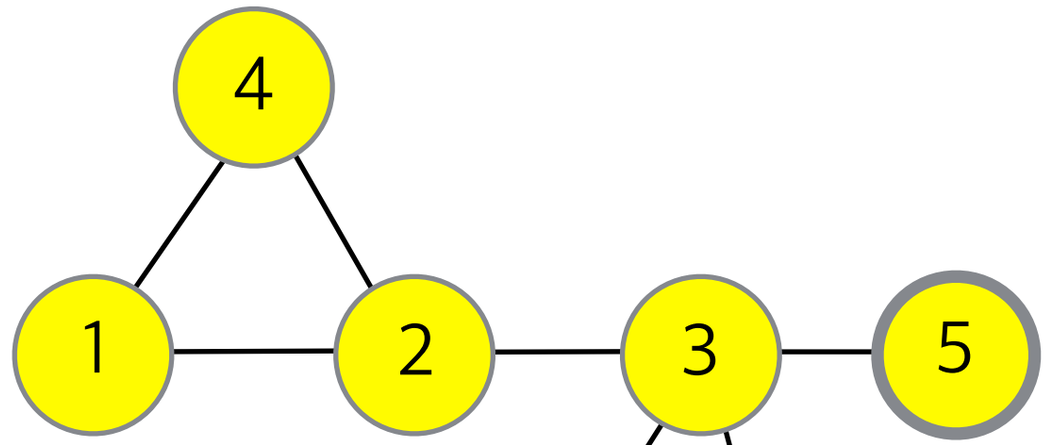


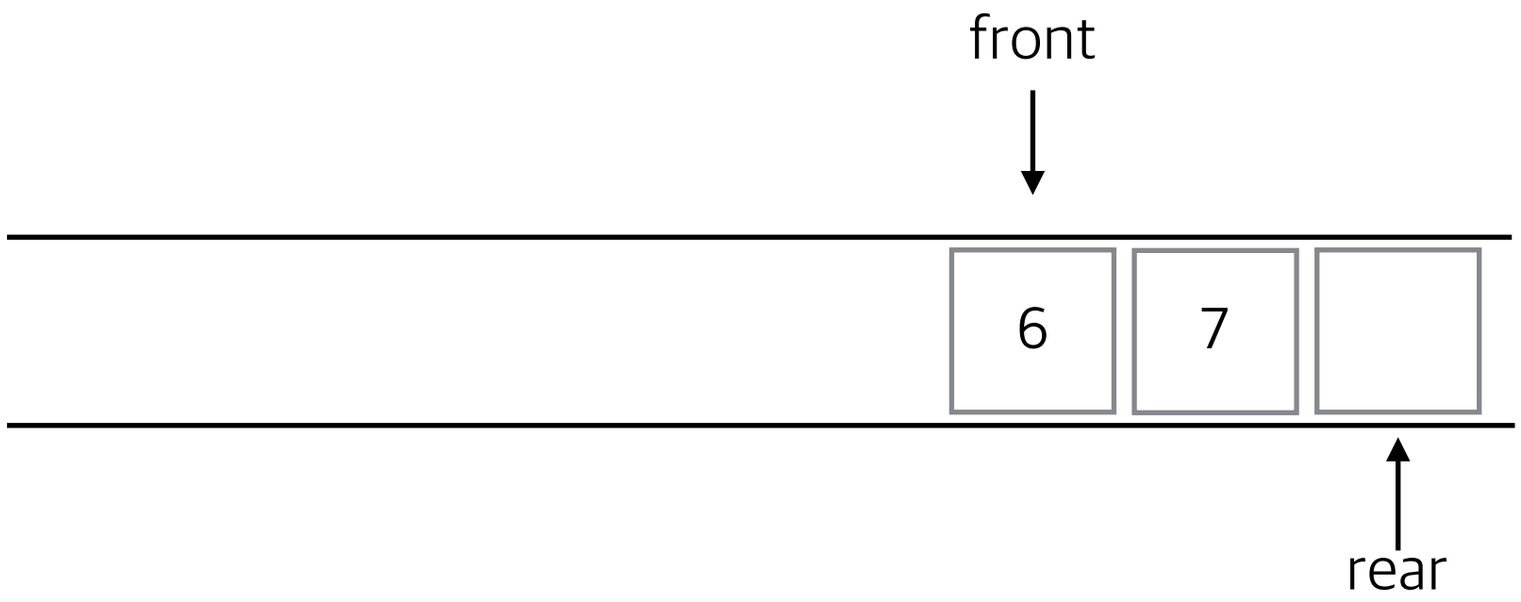


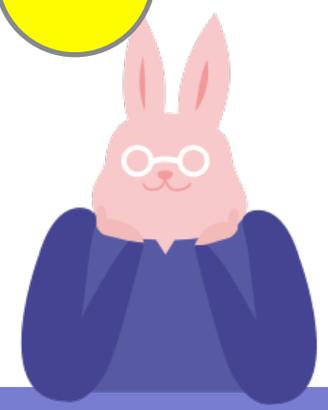


너비 우선 탐색 (BFS)

- 인접한 노드들을 우선 모두 색칠해 두고 뻗어나간다
  1. **Node 선택**
  2. 인접한 노드 색칠

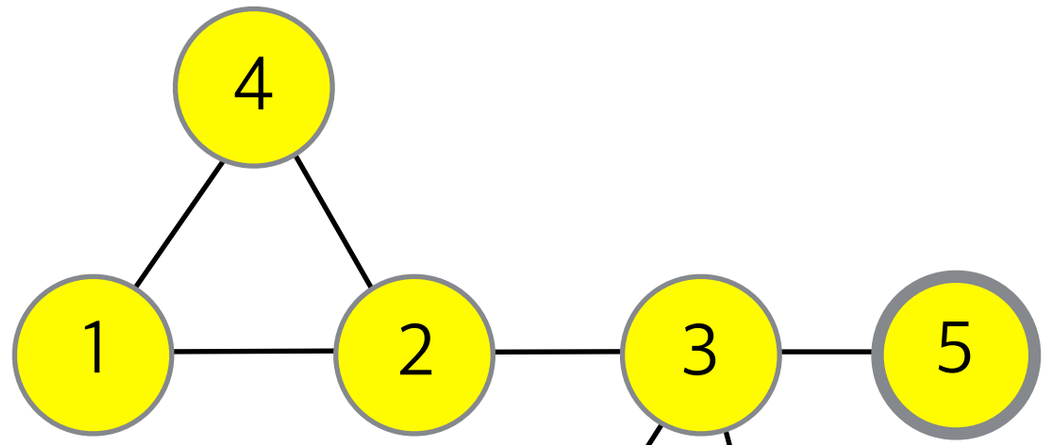


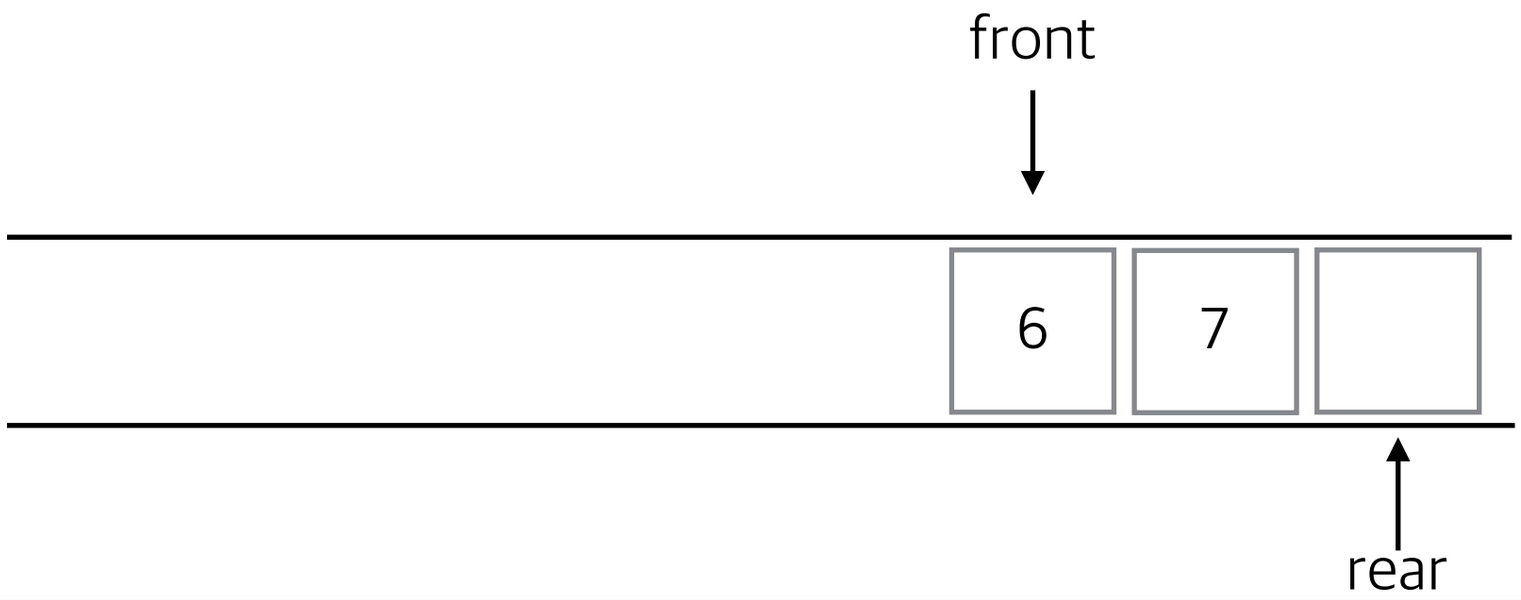


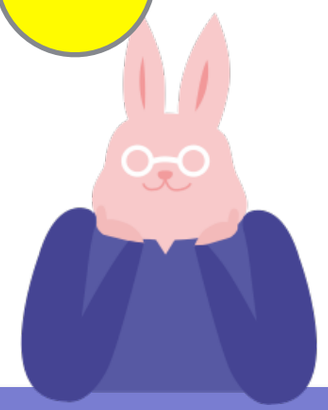


너비 우선 탐색 (BFS)

- 인접한 노드들을 우선 모두 색칠해 두고 뻗어나간다
  1. Node 선택
  2. **인접한 노드 색칠**

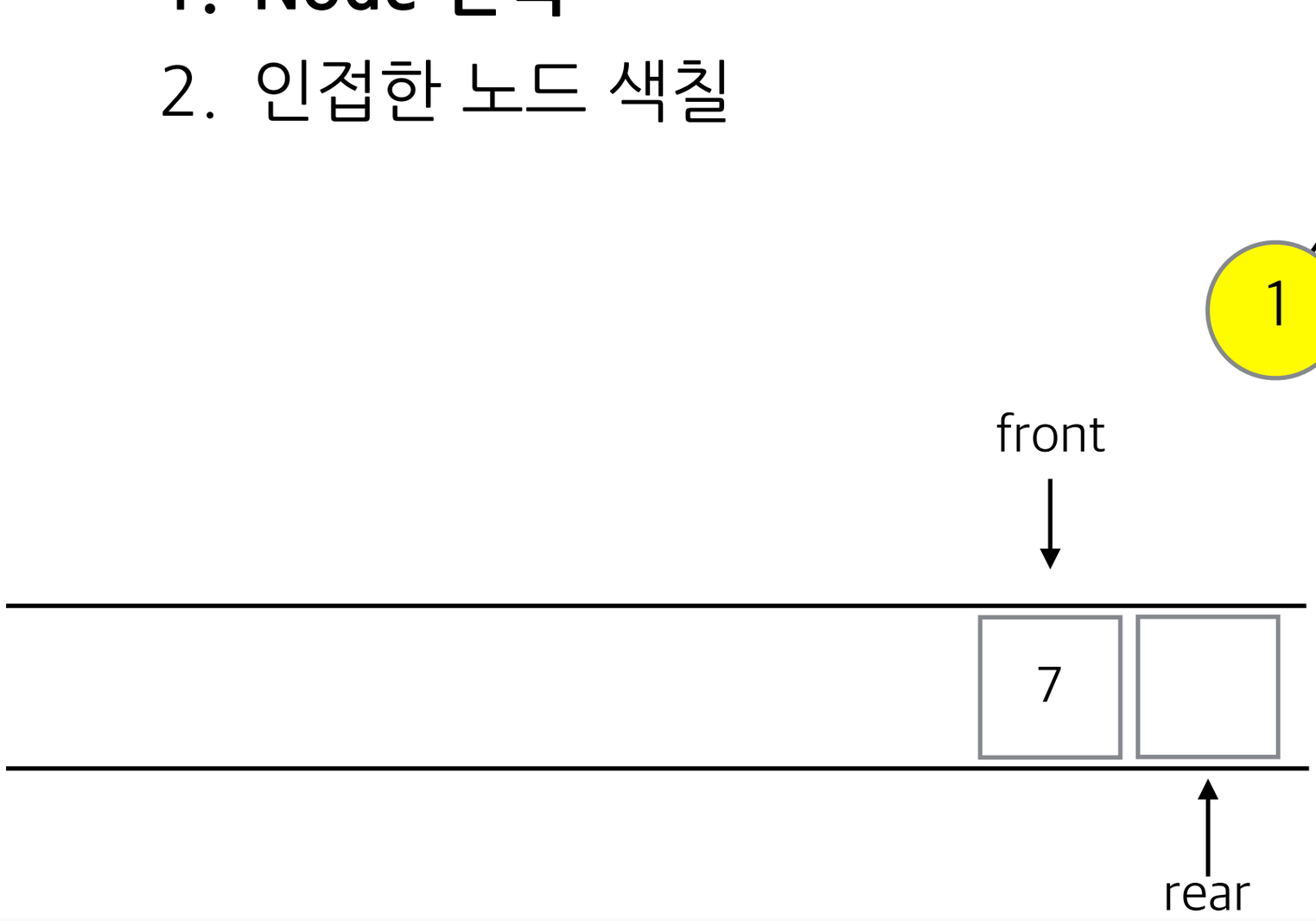






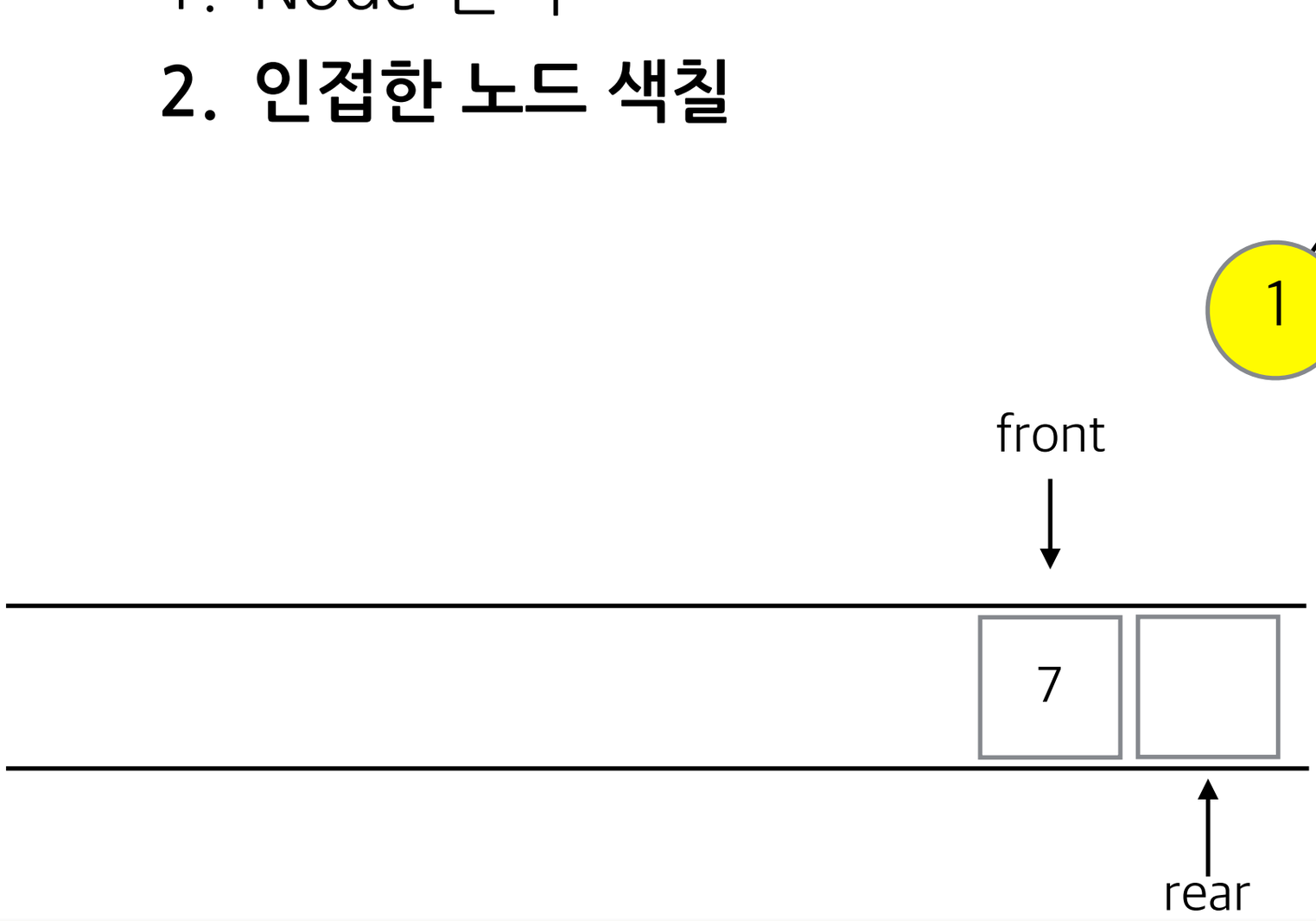
너비 우선 탐색 (BFS)

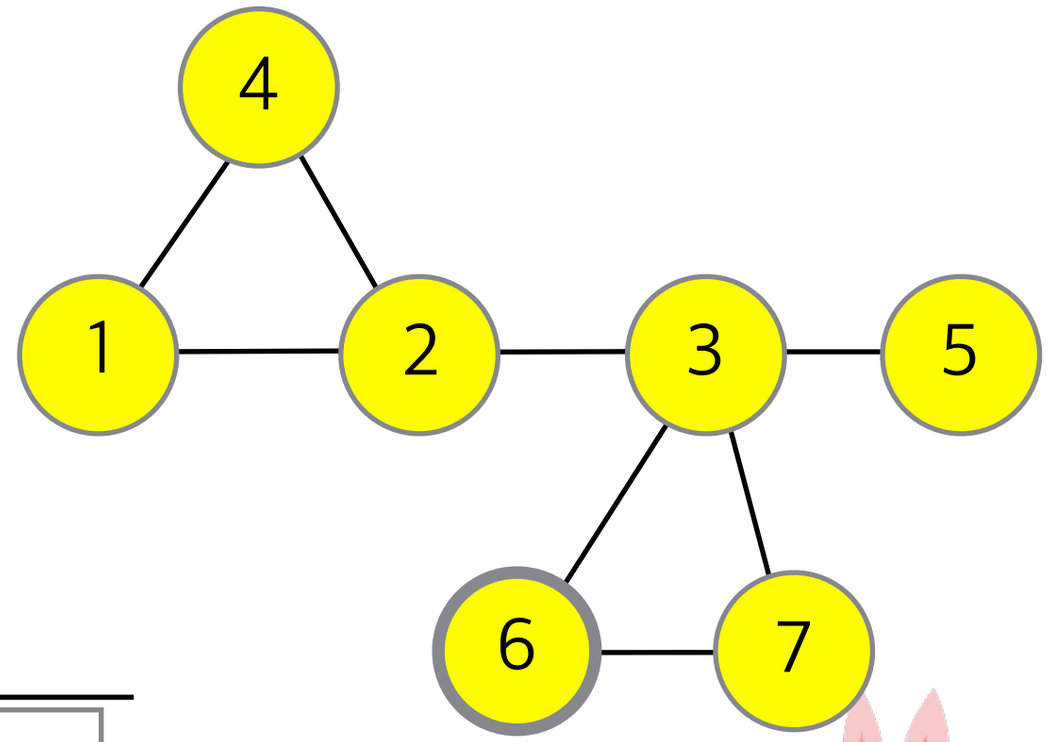
- 인접한 노드들을 우선 모두 색칠해 두고 뻗어나간다
  1. **Node 선택**
  2. 인접한 노드 색칠



너비 우선 탐색 (BFS)

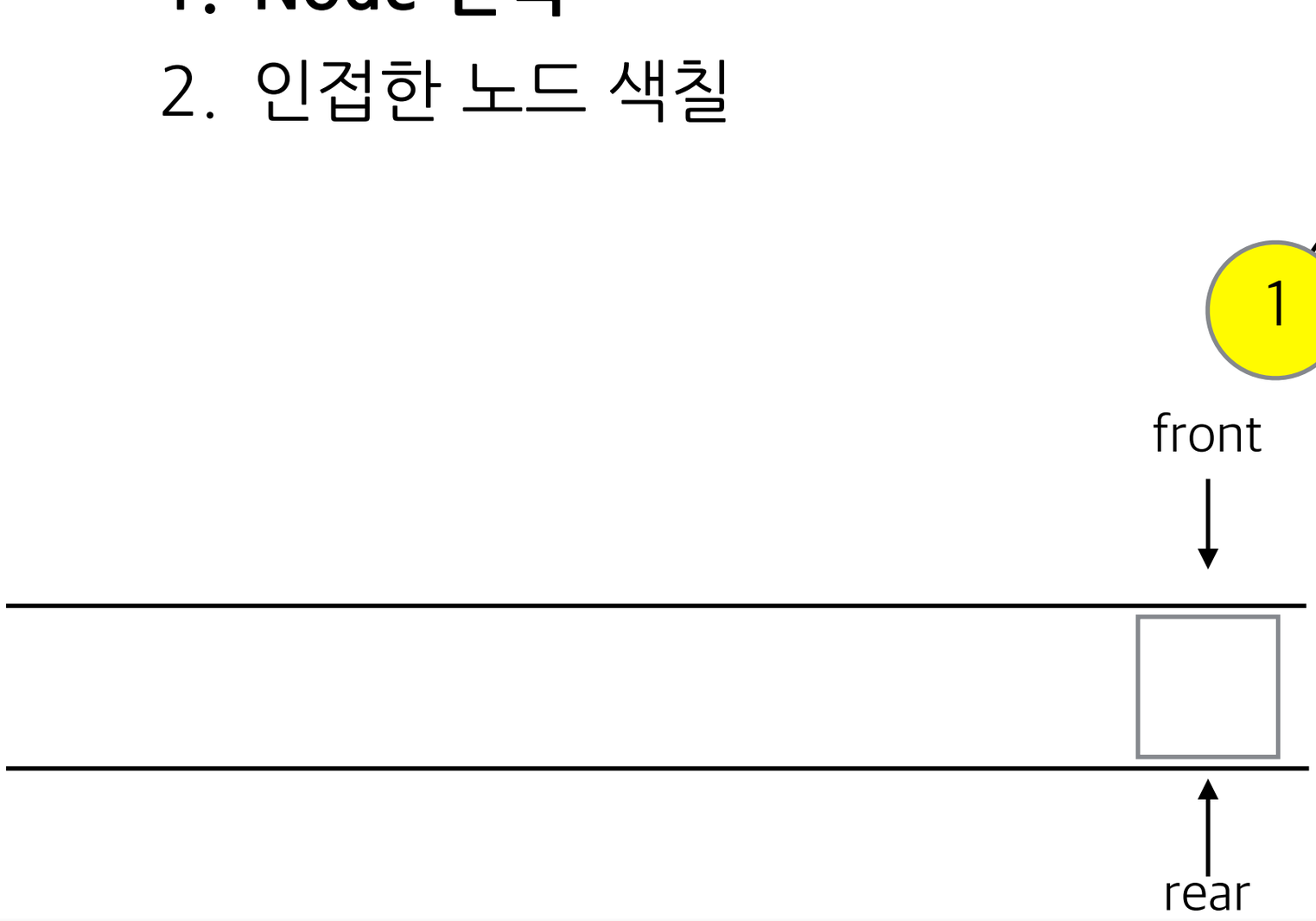
- 인접한 노드들을 우선 모두 색칠해 두고 뻗어나간다
  1. Node 선택
  2. **인접한 노드 색칠**





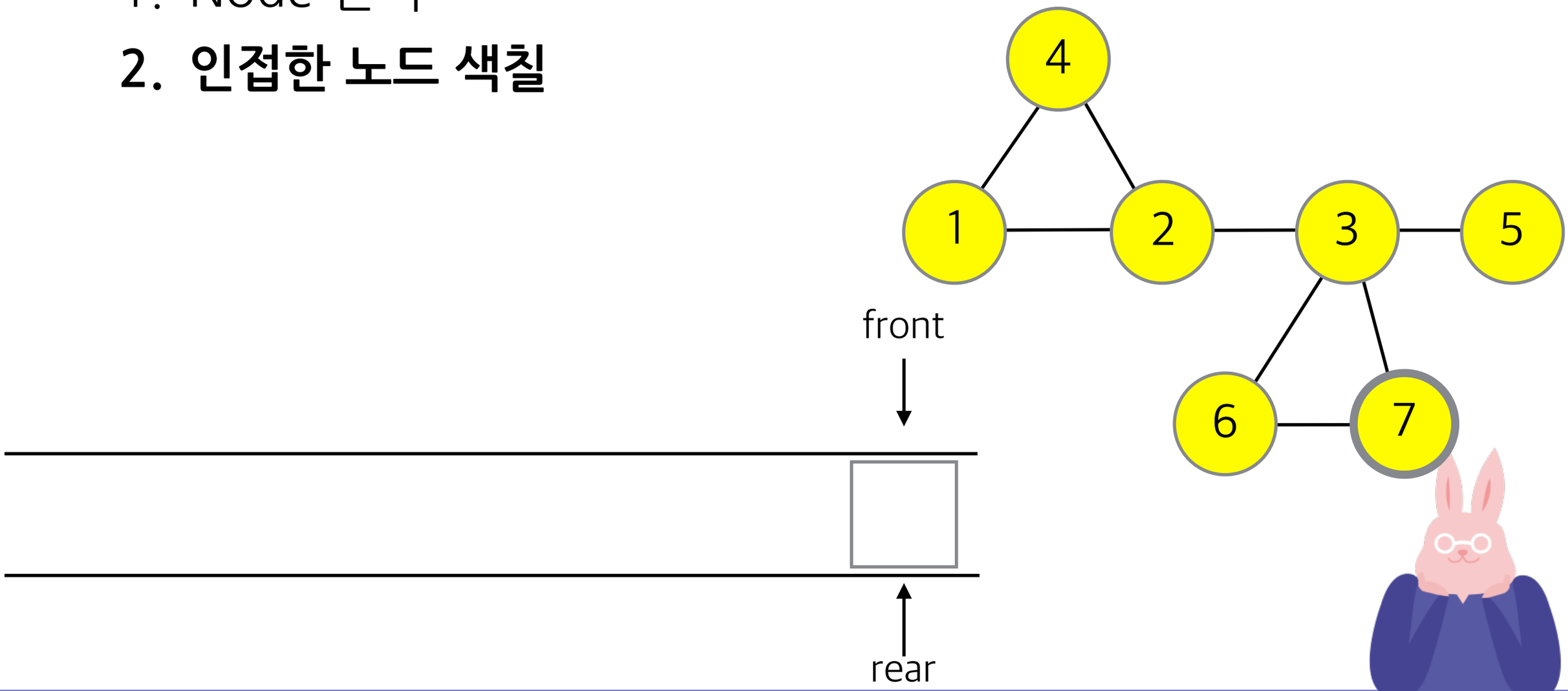
너비 우선 탐색 (BFS)

- 인접한 노드들을 우선 모두 색칠해 두고 뻗어나간다
  1. **Node 선택**
  2. 인접한 노드 색칠



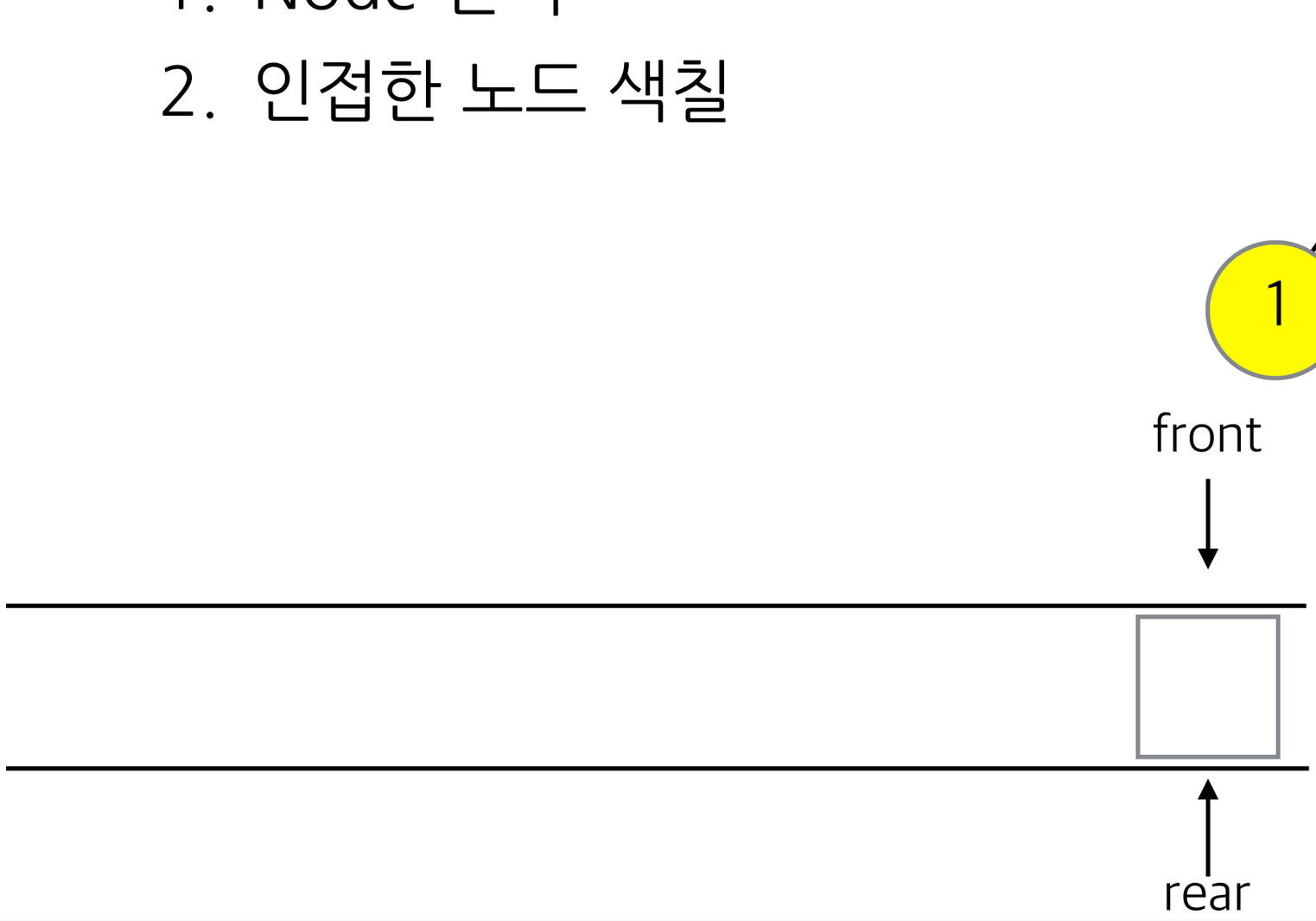
너비 우선 탐색 (BFS)

- 인접한 노드들을 우선 모두 색칠해 두고 뻗어나간다
  1. Node 선택
  2. **인접한 노드 색칠**



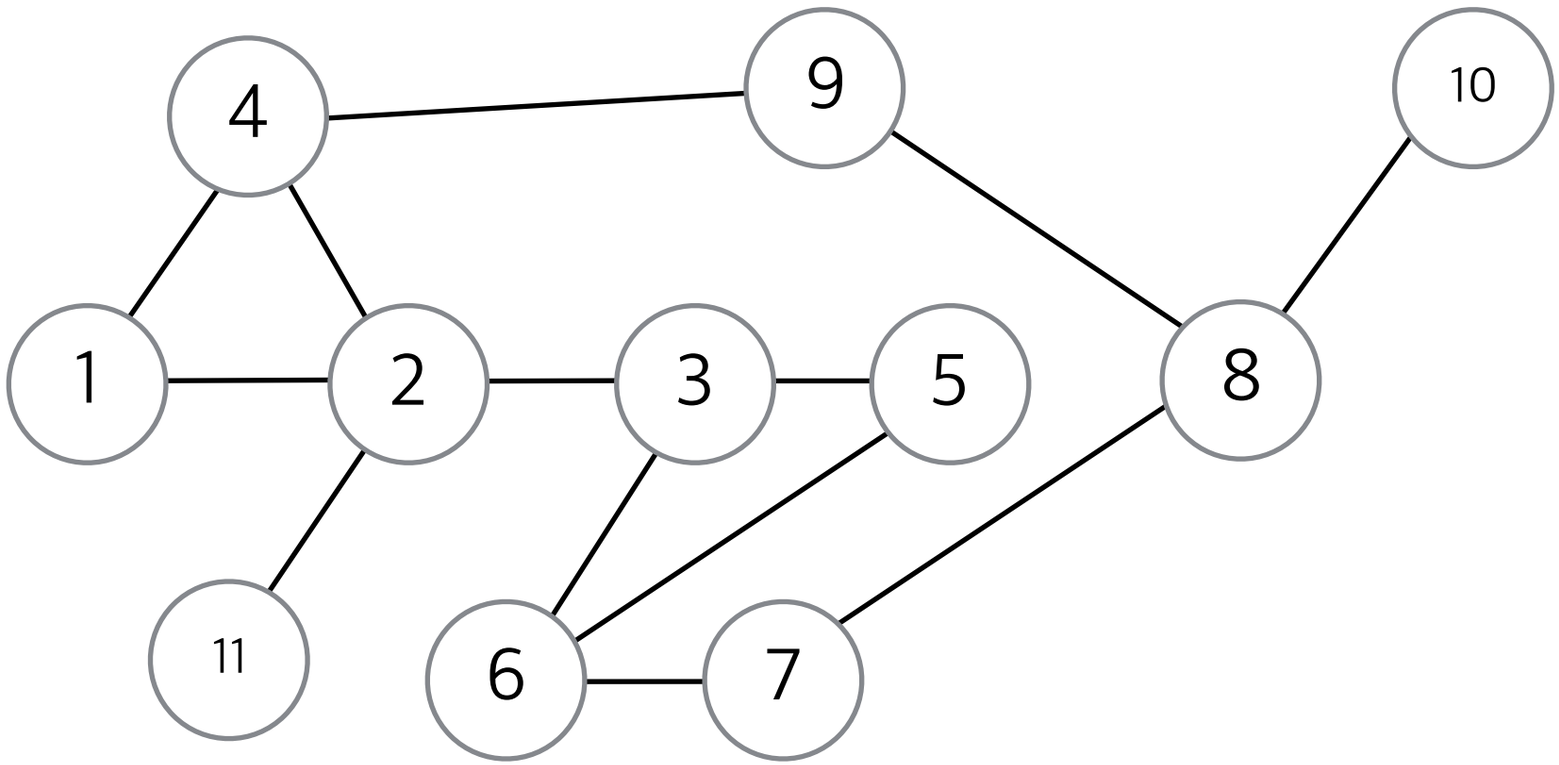
너비 우선 탐색 (BFS)

- 인접한 노드들을 우선 모두 색칠해 두고 뻗어나간다
  1. Node 선택
  2. 인접한 노드 색칠



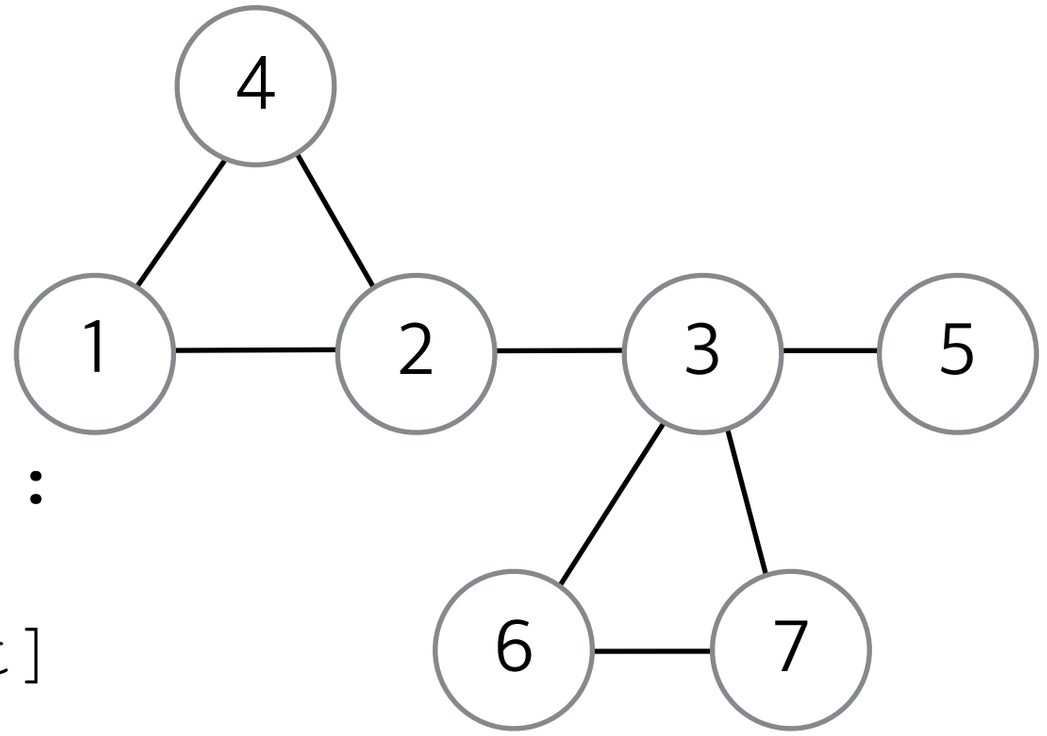
너비 우선 탐색 (BFS)

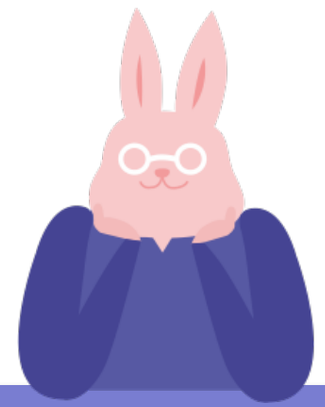
- 다음 그래프에 대하여 1을 시작으로 BFS한 결과는 ?
  - 단, 인접한 노드가 여러개일 경우 노드 번호가 작은 노드부터 방문



너비 우선 탐색 (BFS) 구현

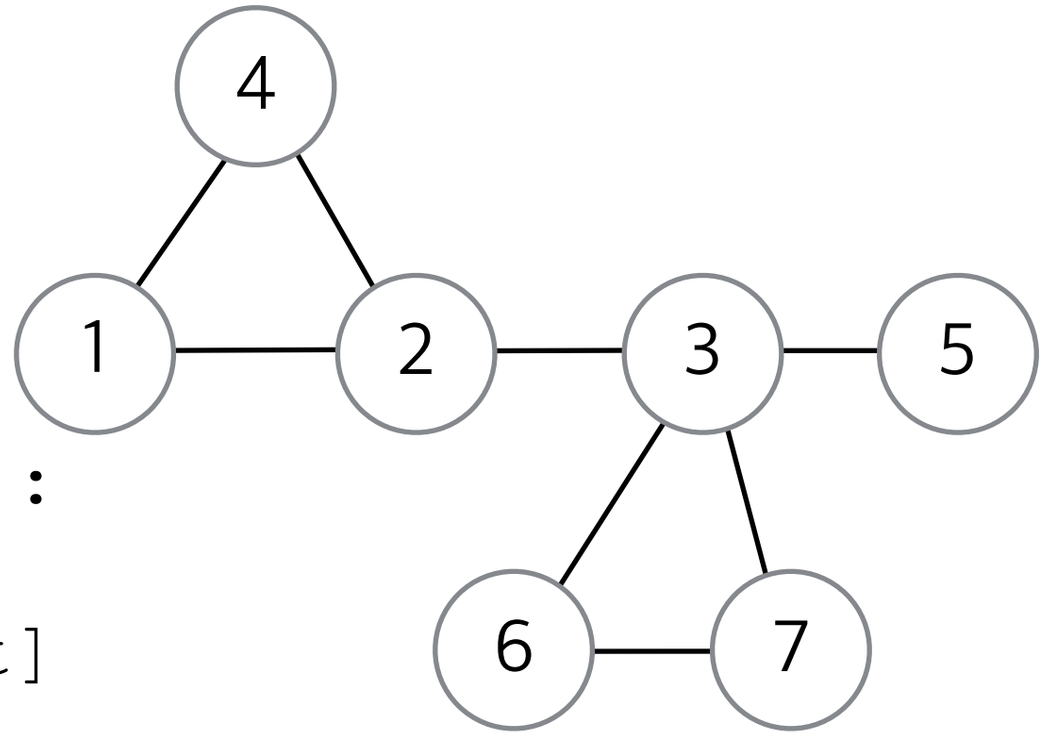
```
def BFS(graph, x, visited) :
    result = []
    visited[x] = True
    Queue.put(x)

    while Queue.empty() == False :
        current = Queue.get()
        result = result + [current]

        for v in graph[current] :
            if visited[v] == False :
                visited[v] = True
                Queue.put(v)

return result
```

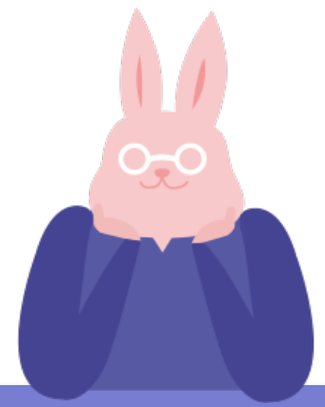




너비 우선 탐색 (BFS) 구현

```
def BFS(graph, x, visited) :
    result = []
    visited[x] = True
    Queue.put(x)

    while Queue.empty() == False :
        current = Queue.get()
        result = result + [current]

        for v in graph[current] :
            if visited[v] == False :
                visited[v] = True
                Queue.put(v)

return result
```

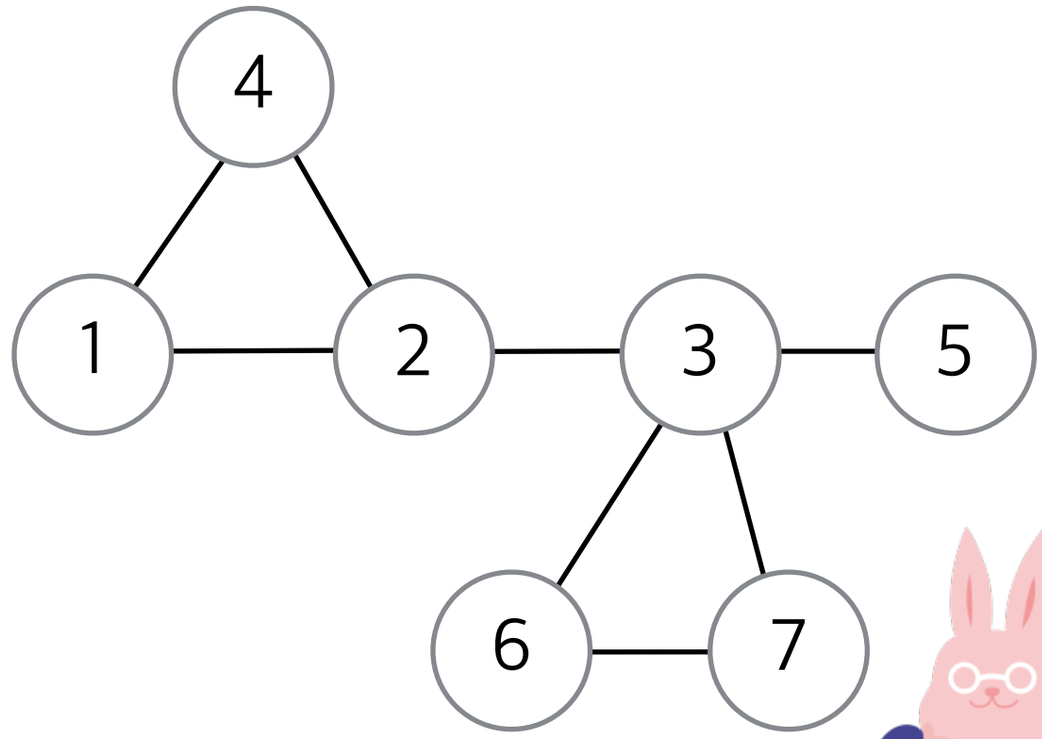




DFS와 BFS의 정확성 증명 및 시간복잡도

- 정확성 증명
  - 질문 : 모든 Node와 Edge를 방문하는가 ?

- 시간복잡도



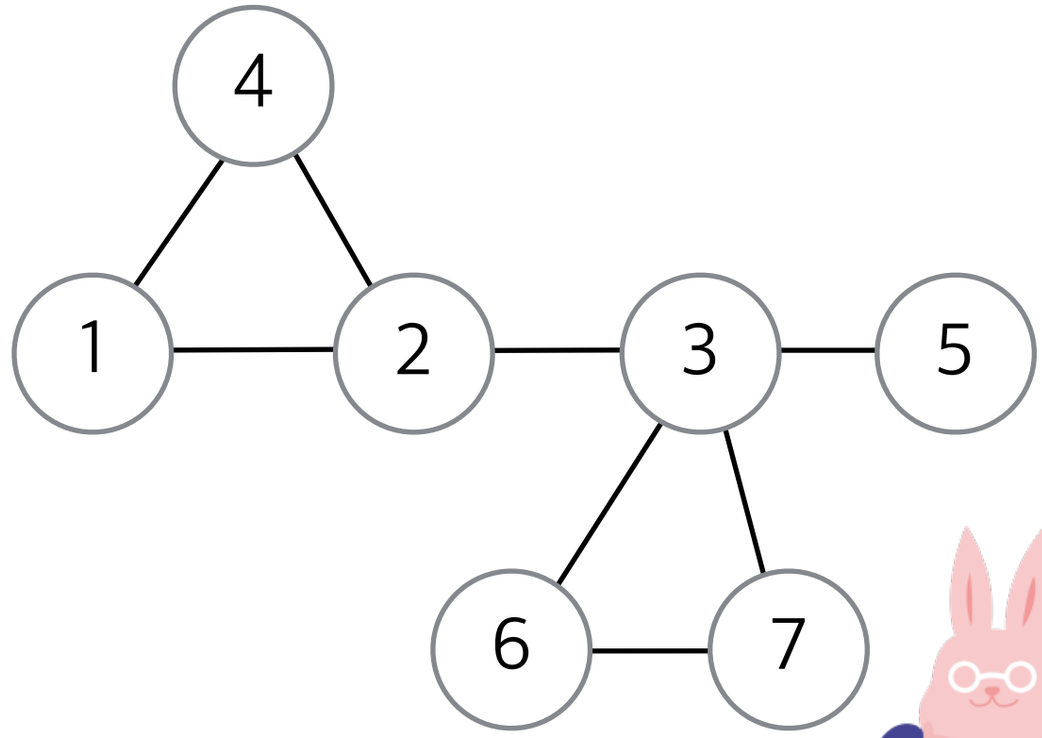
DFS와 BFS의 정확성 증명 및 시간복잡도

- 정확성 증명
  - 질문 : 모든 Node와 Edge를 방문하는가 ? **YES!**

- 시간복잡도
  - O(V+E)

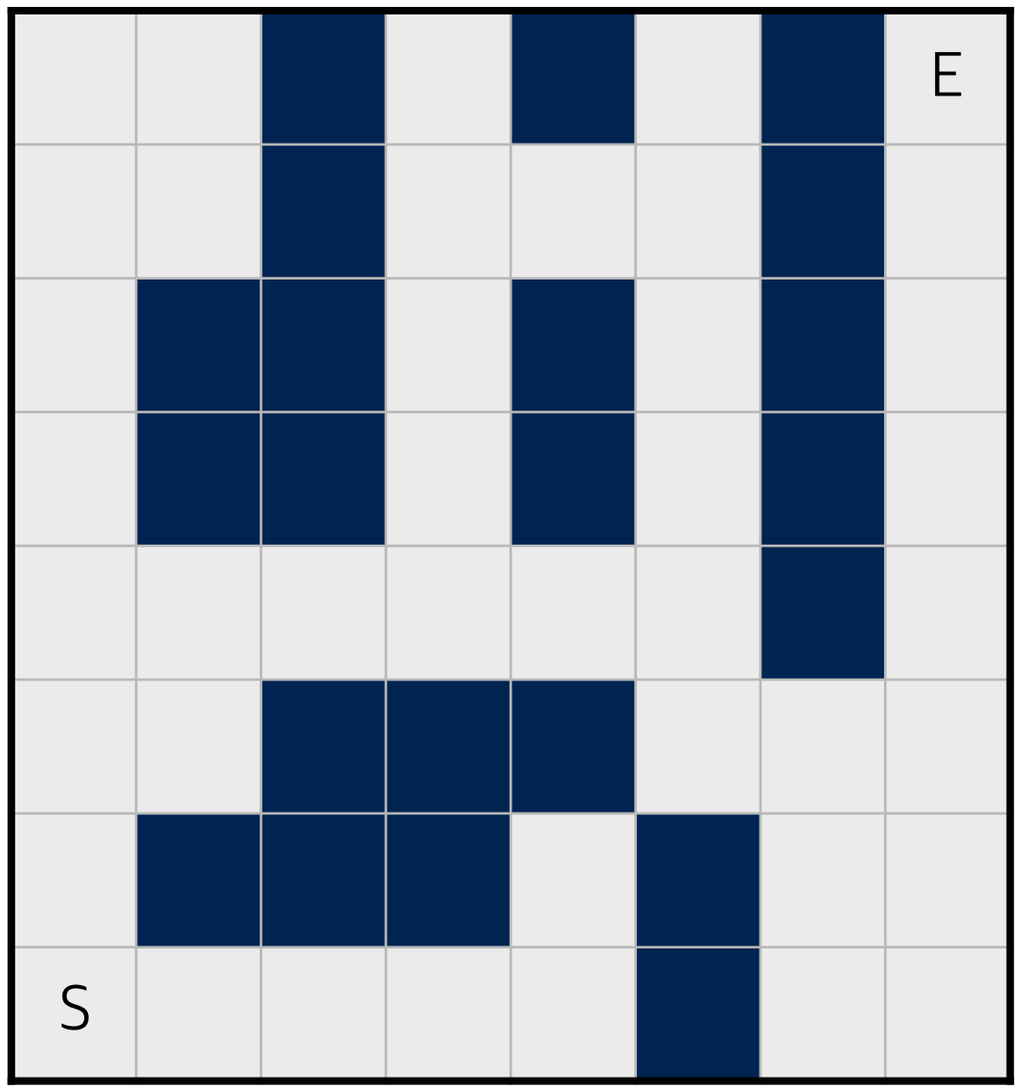
  V : Node의 갯수

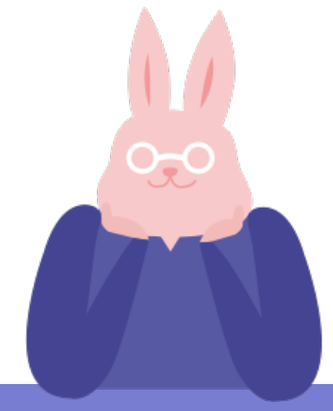
  E : Edge의 개수



DFS와 BFS의 응용 : 미로찾기

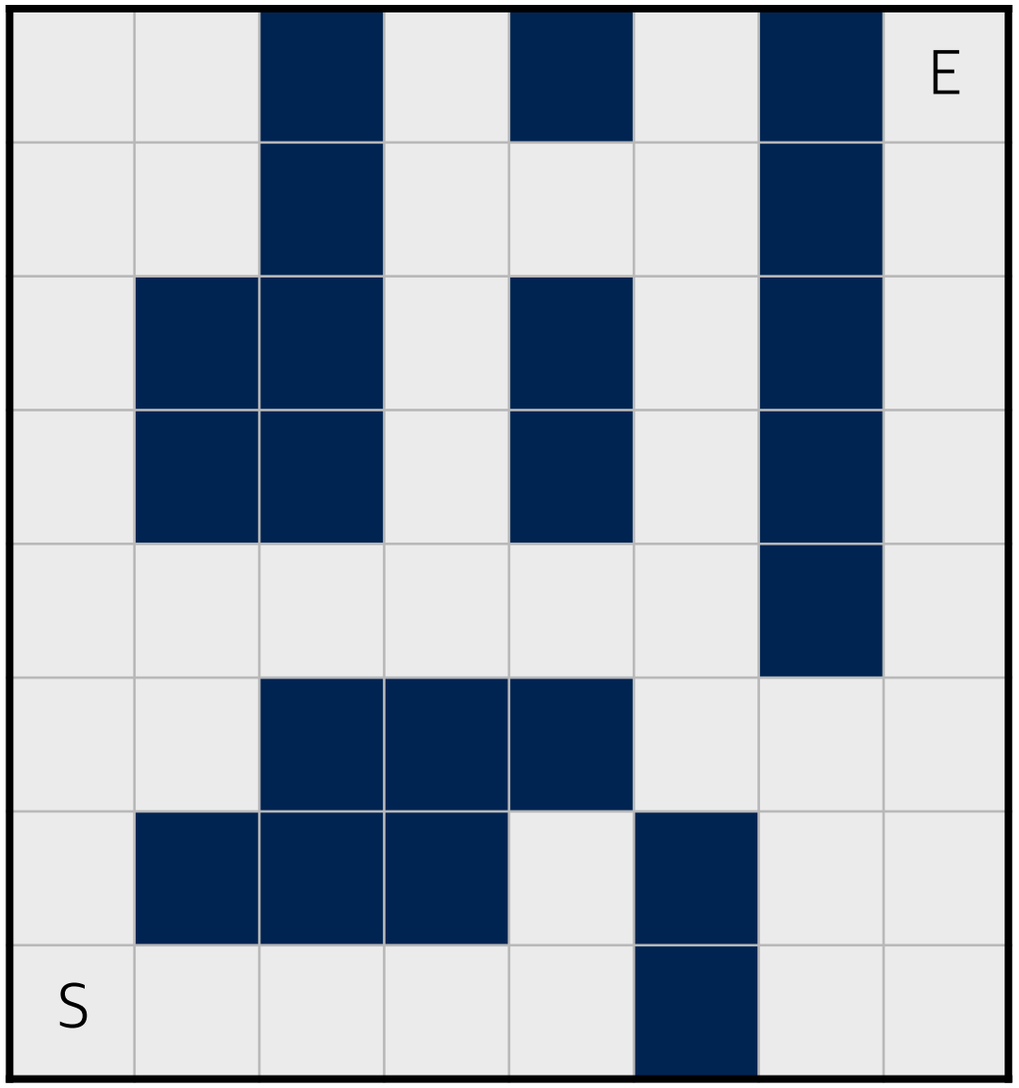
- 시작점에서 도착점까지 갈 때의 최단거리를 출력하라

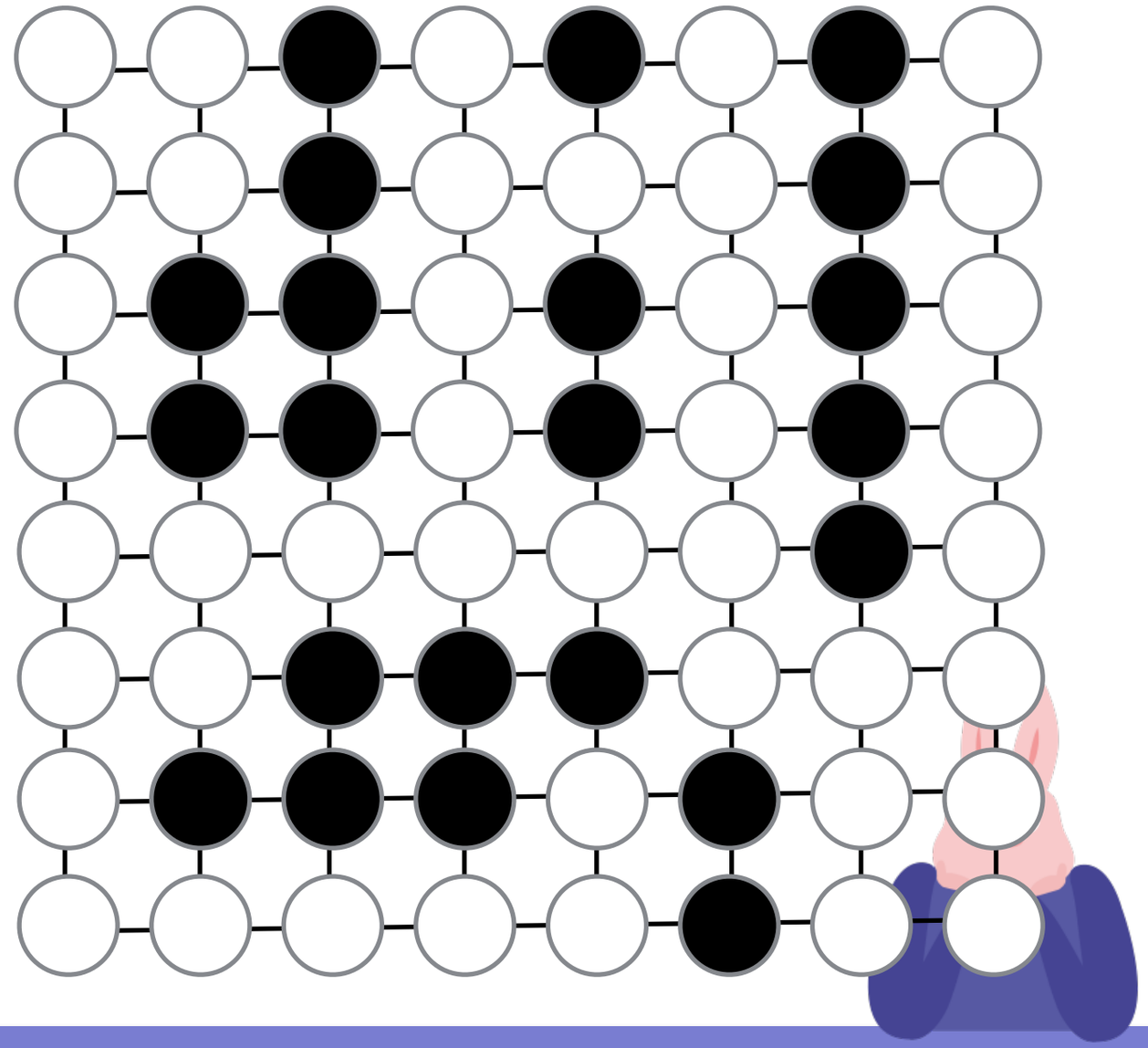




DFS와 BFS의 응용 : 미로찾기

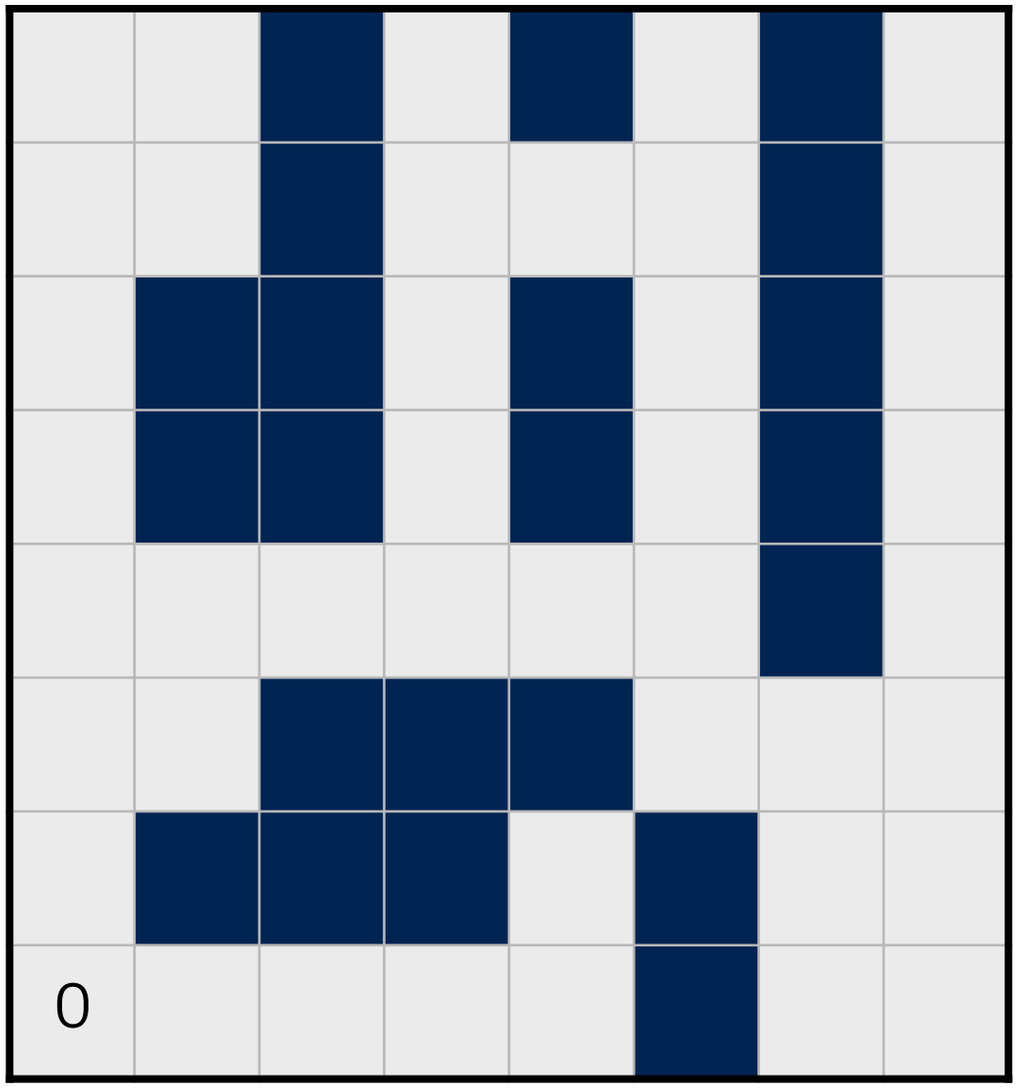
1. Graph로 변환

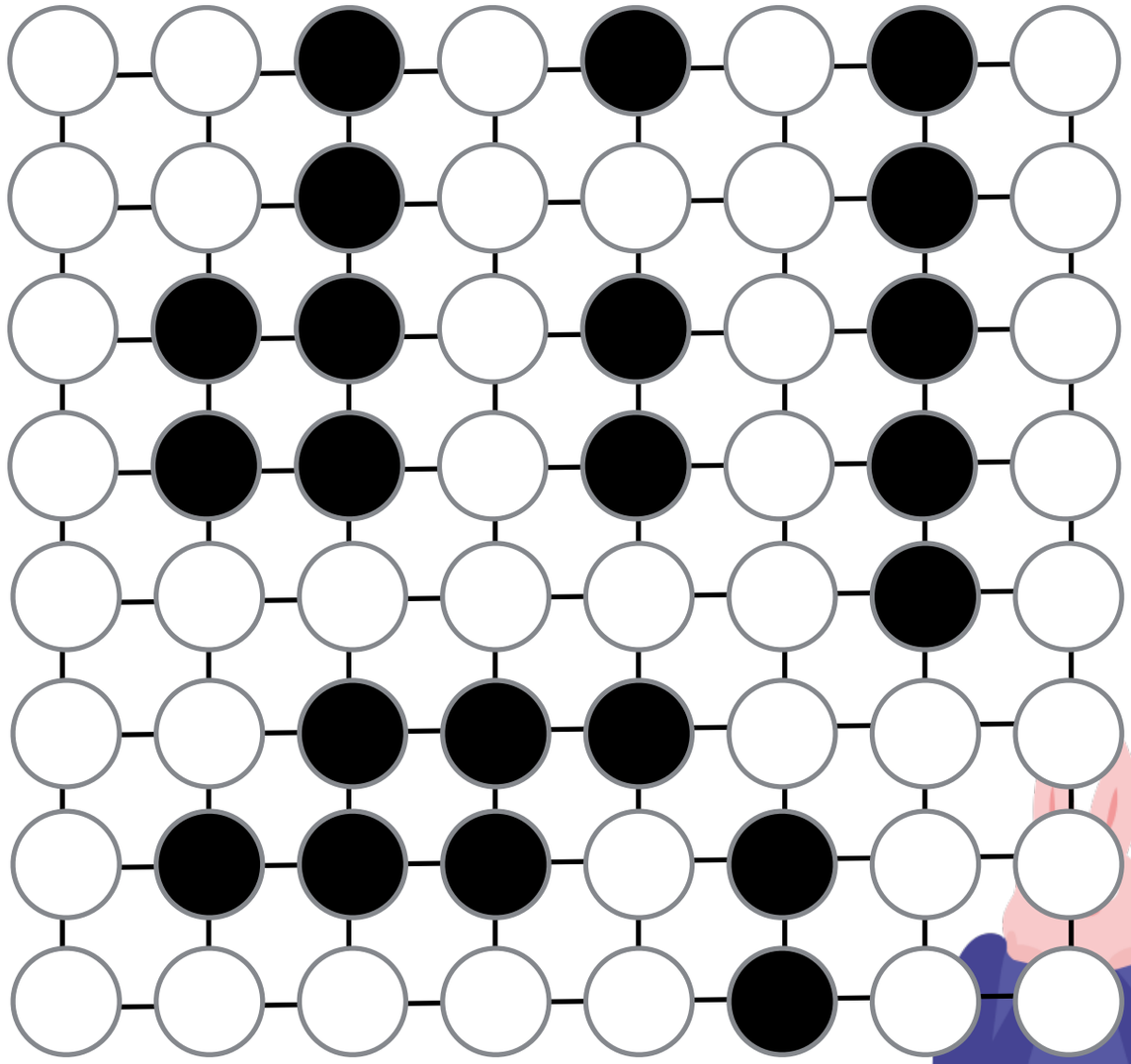




DFS와 BFS의 응용 : 미로찾기

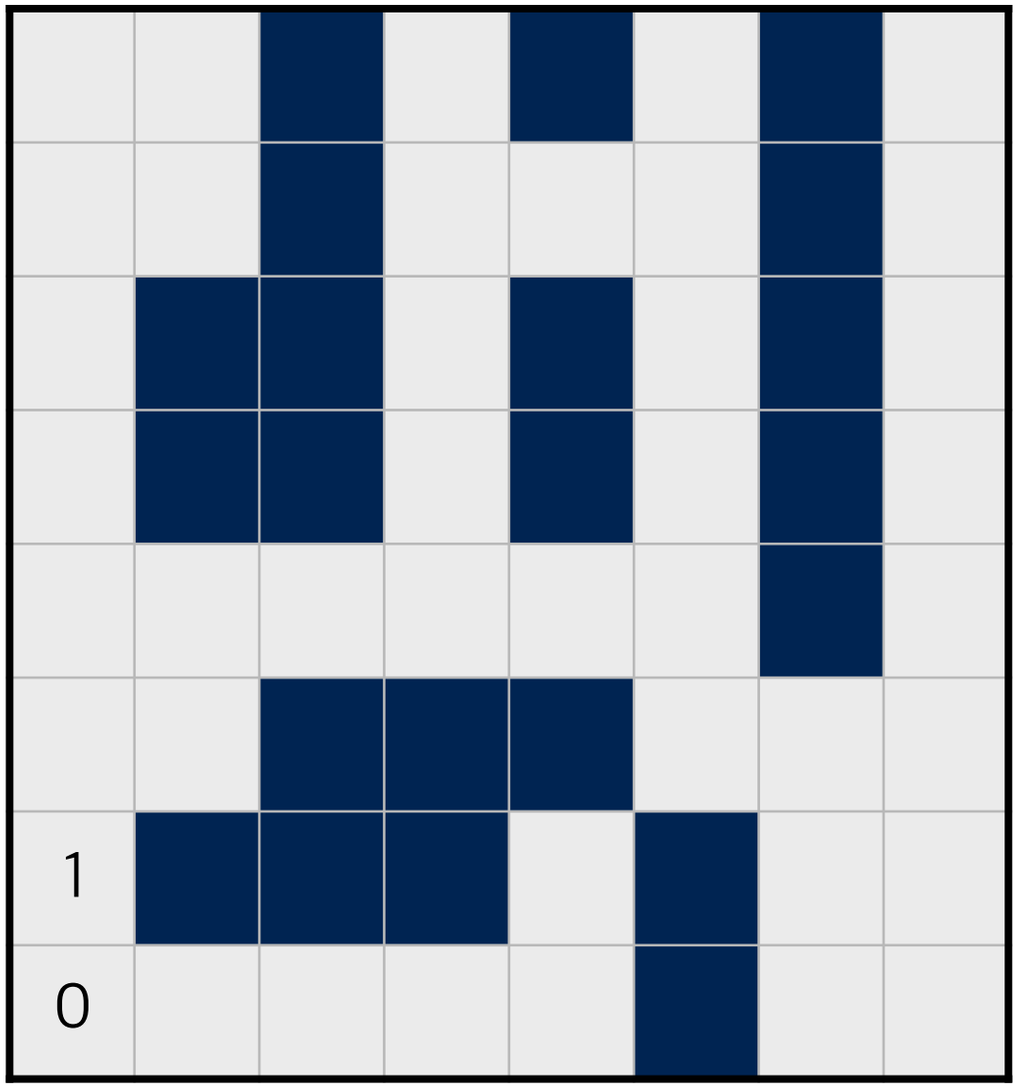
2. 각 노드에, 그 노드에 도착하기 위한 최단거리를 저장

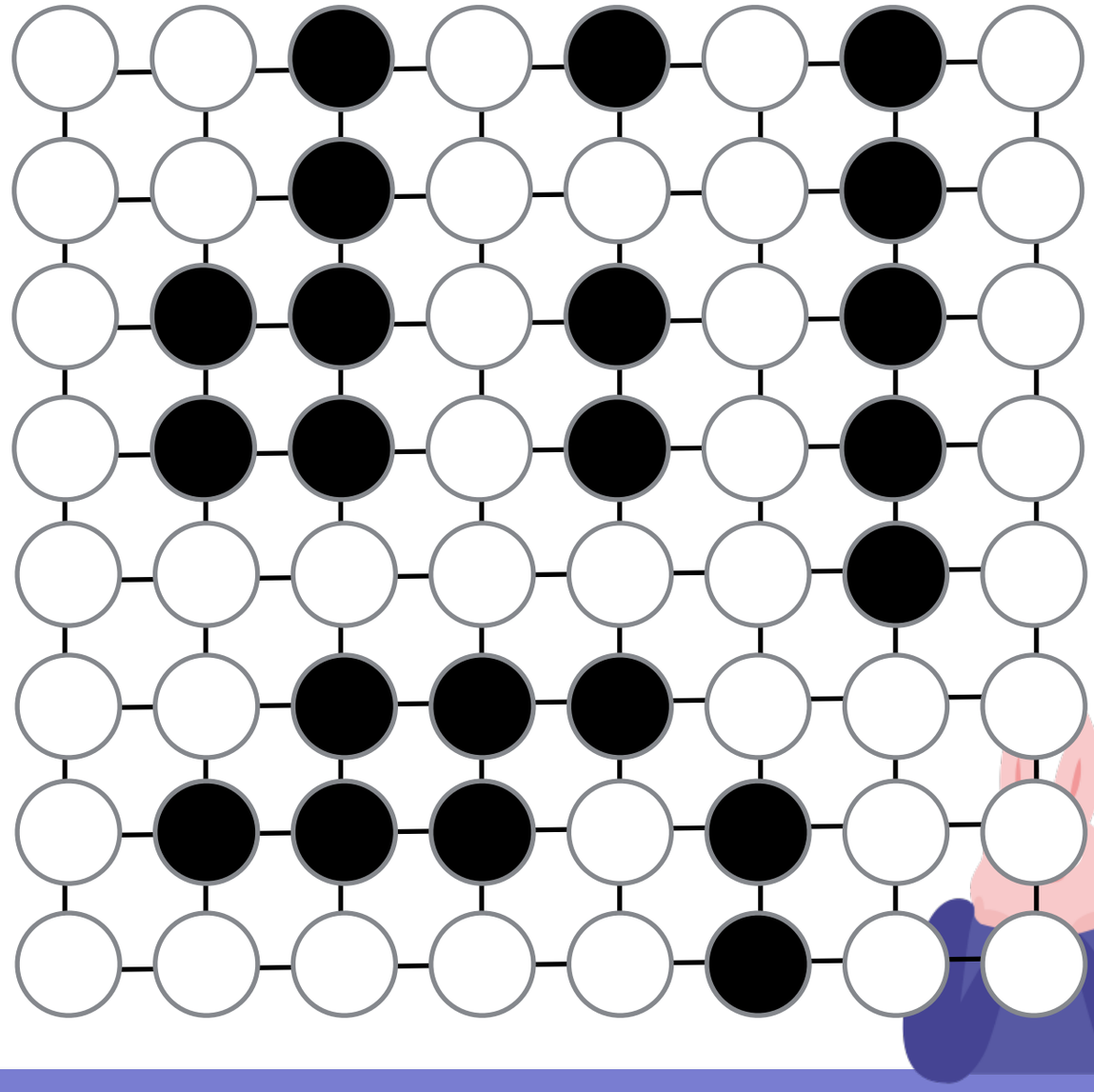




DFS와 BFS의 응용 : 미로찾기

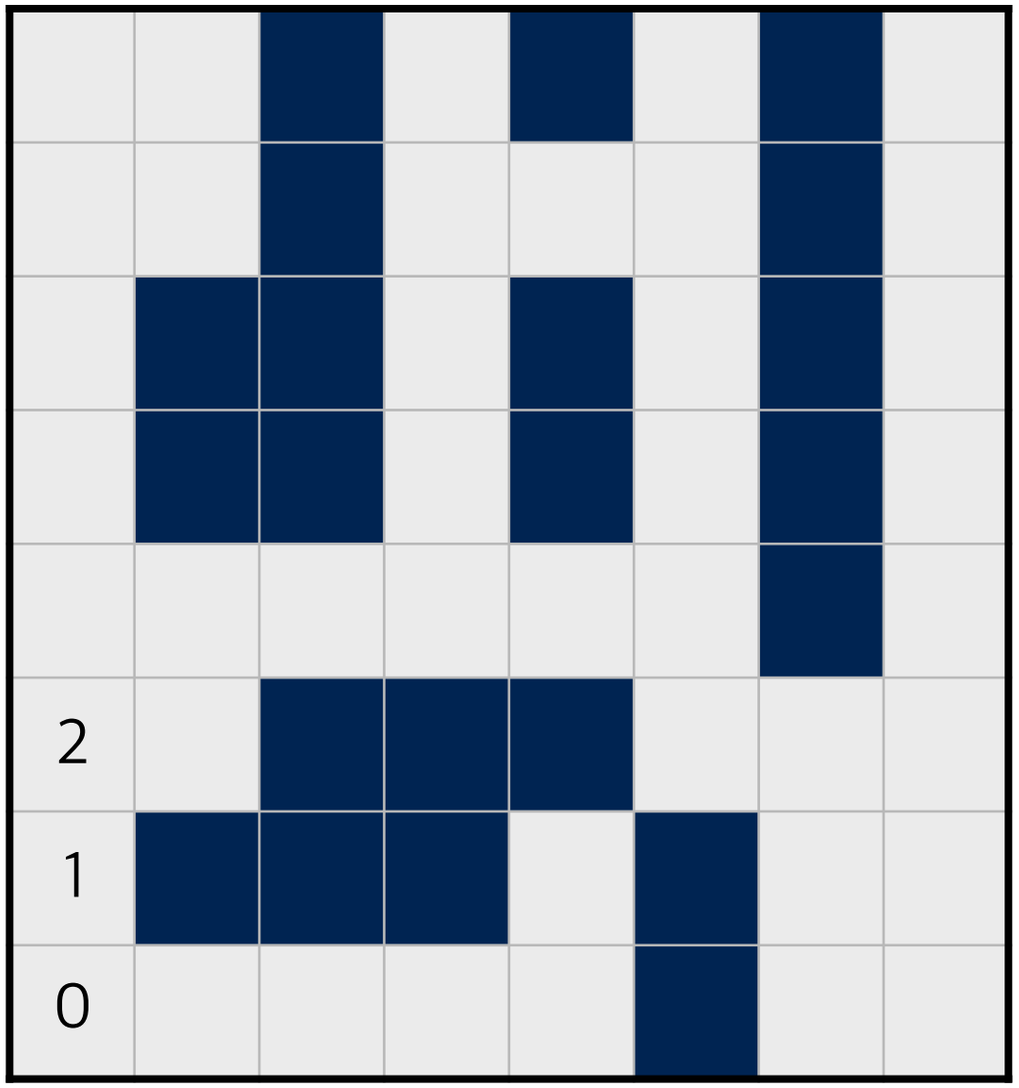
2. 각 노드에, 그 노드에 도착하기 위한 최단거리를 저장

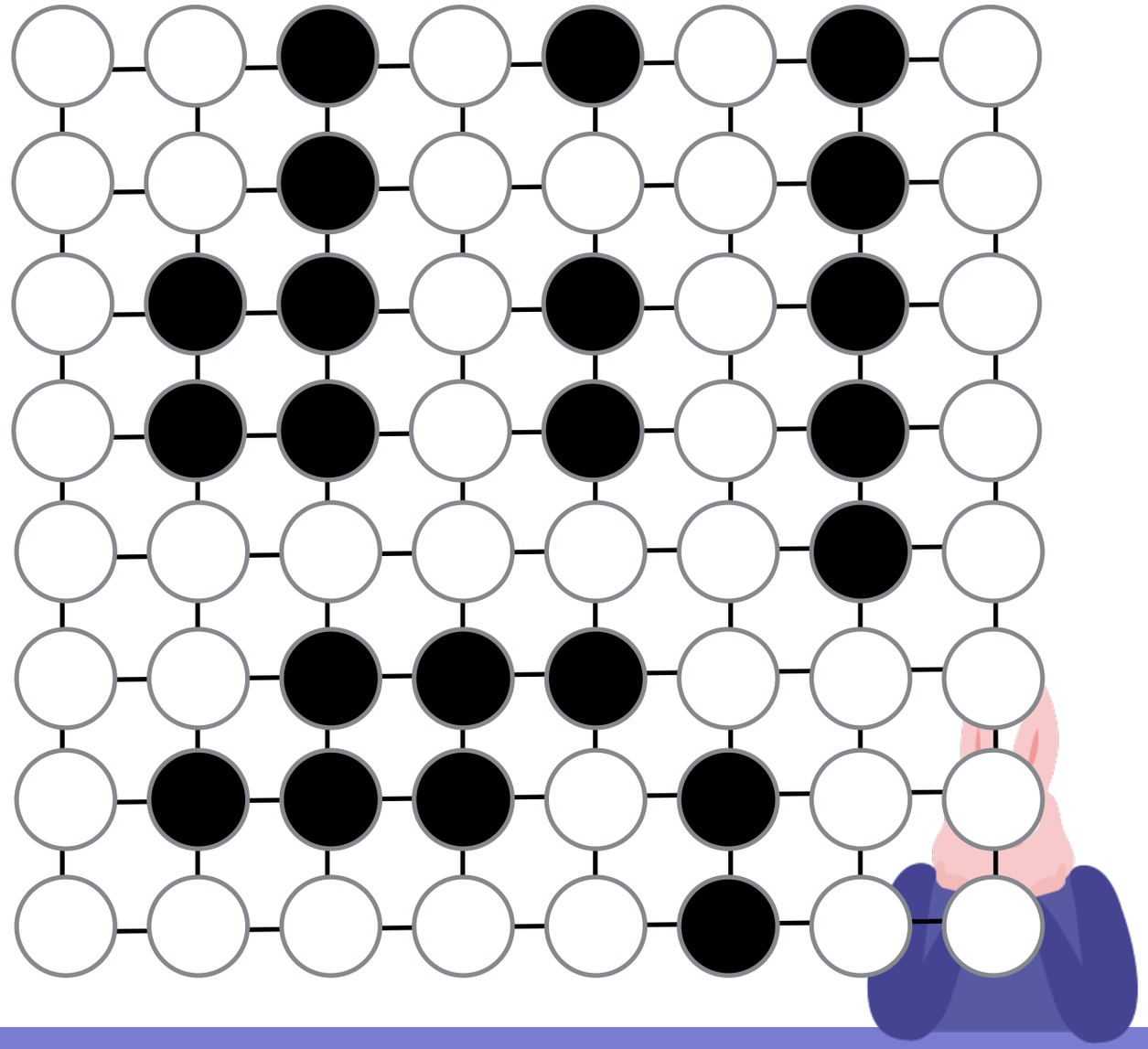




DFS와 BFS의 응용 : 미로찾기

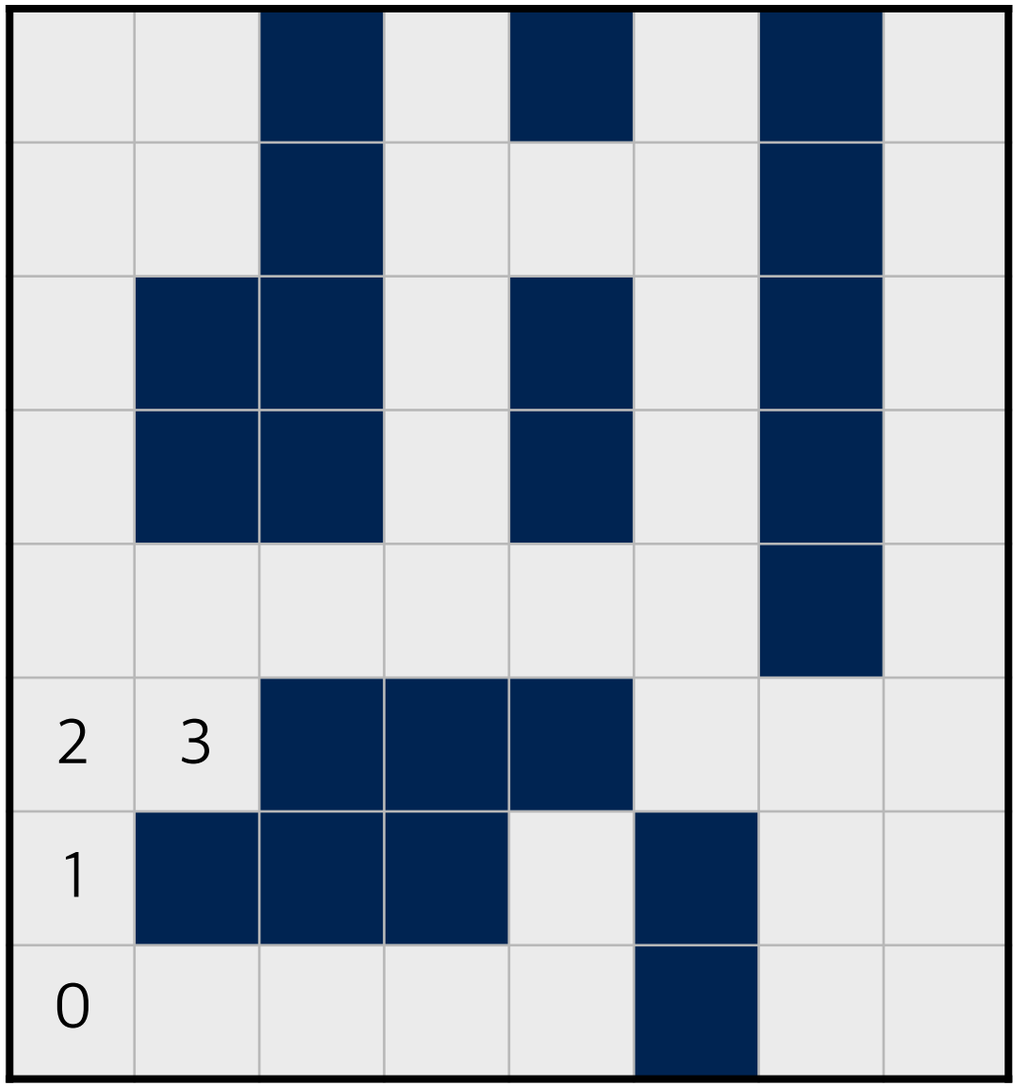
2. 각 노드에, 그 노드에 도착하기 위한 최단거리를 저장

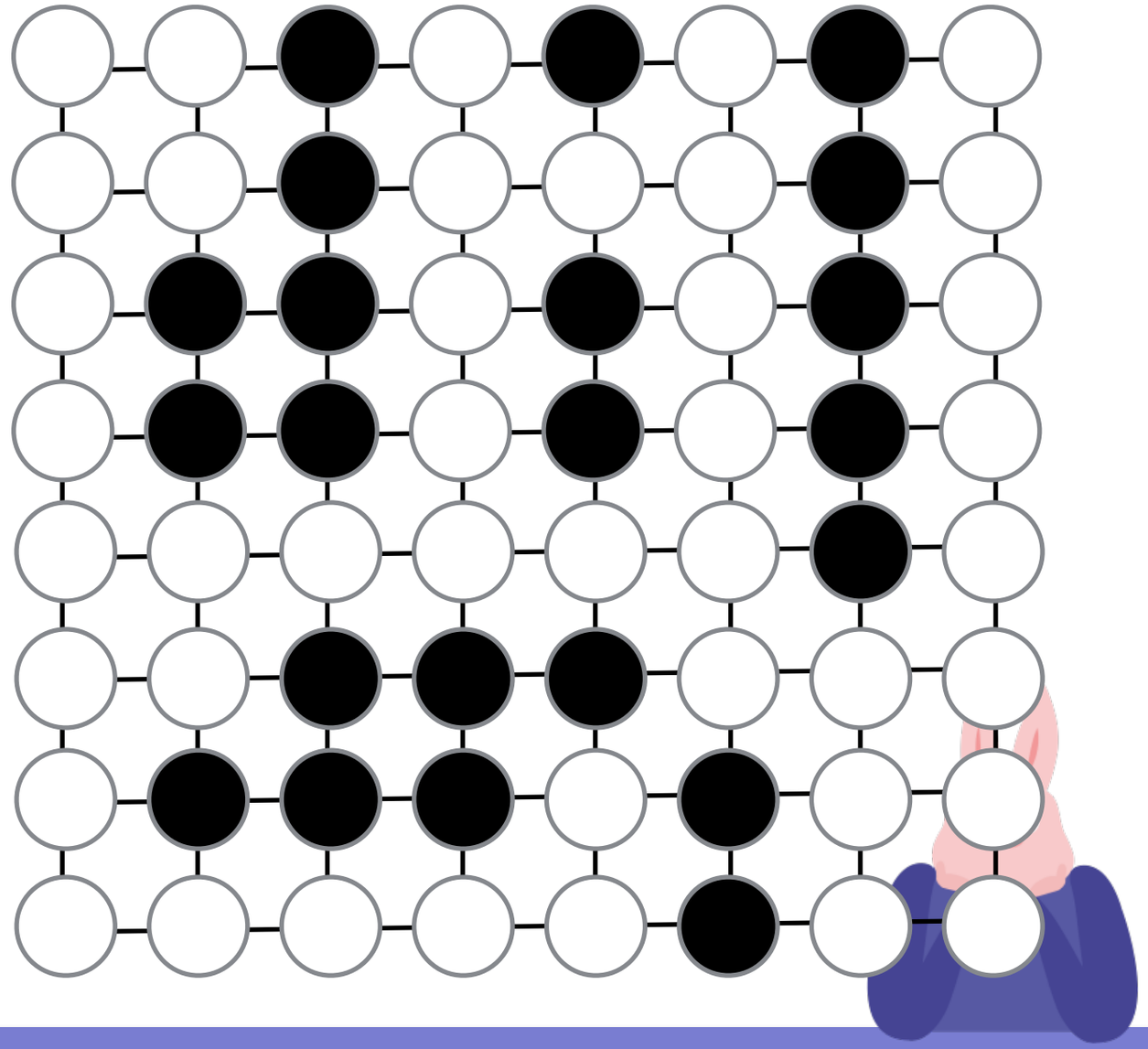




DFS와 BFS의 응용 : 미로찾기

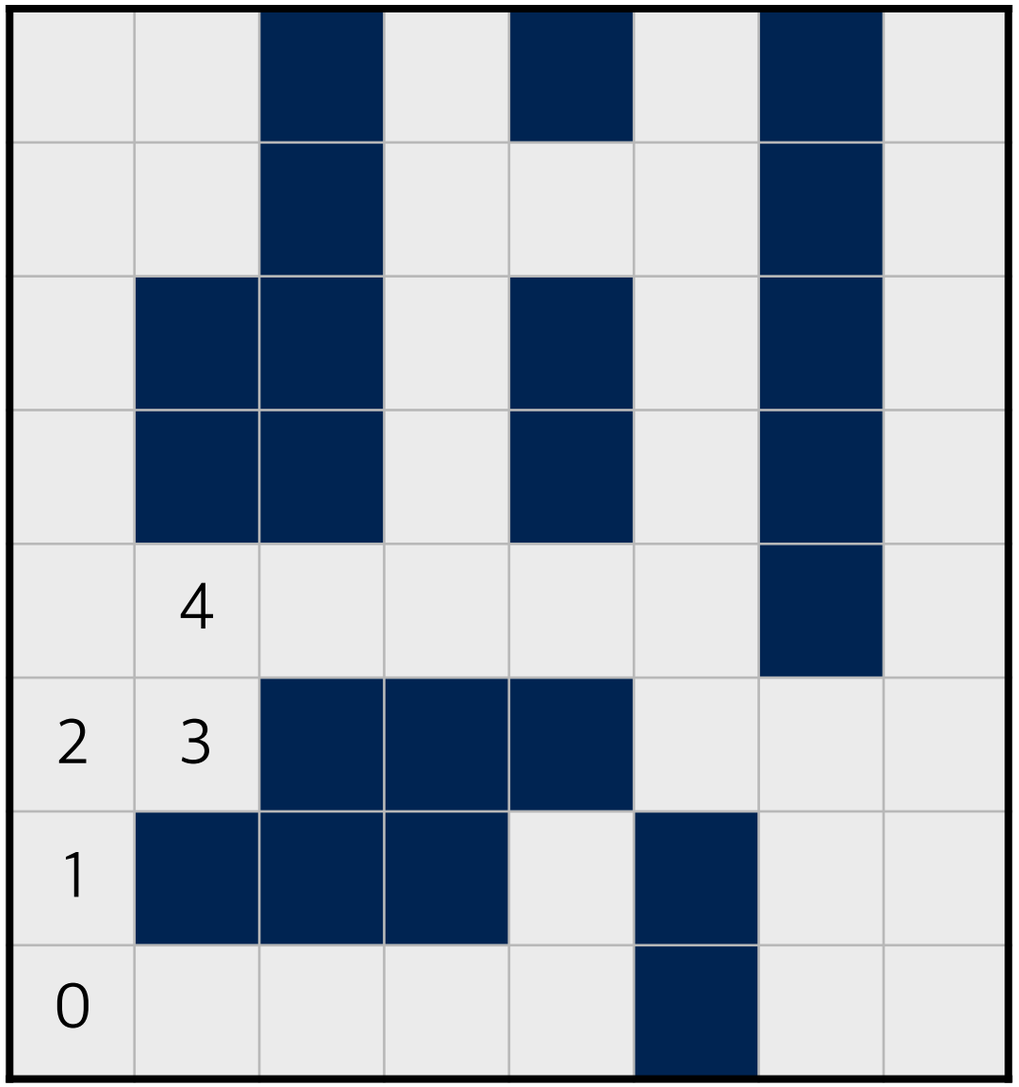
2. 각 노드에, 그 노드에 도착하기 위한 최단거리를 저장

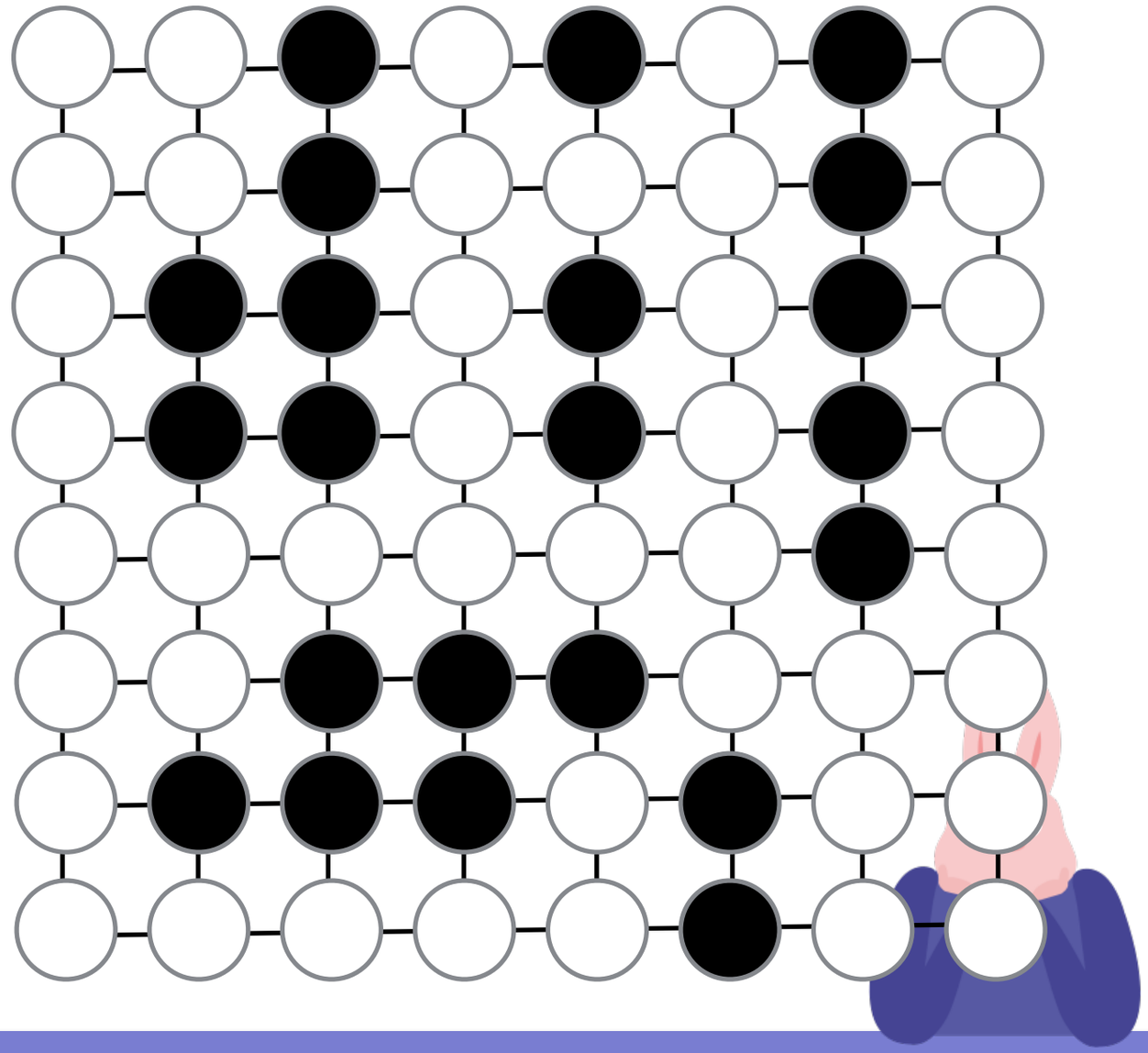




DFS와 BFS의 응용 : 미로찾기

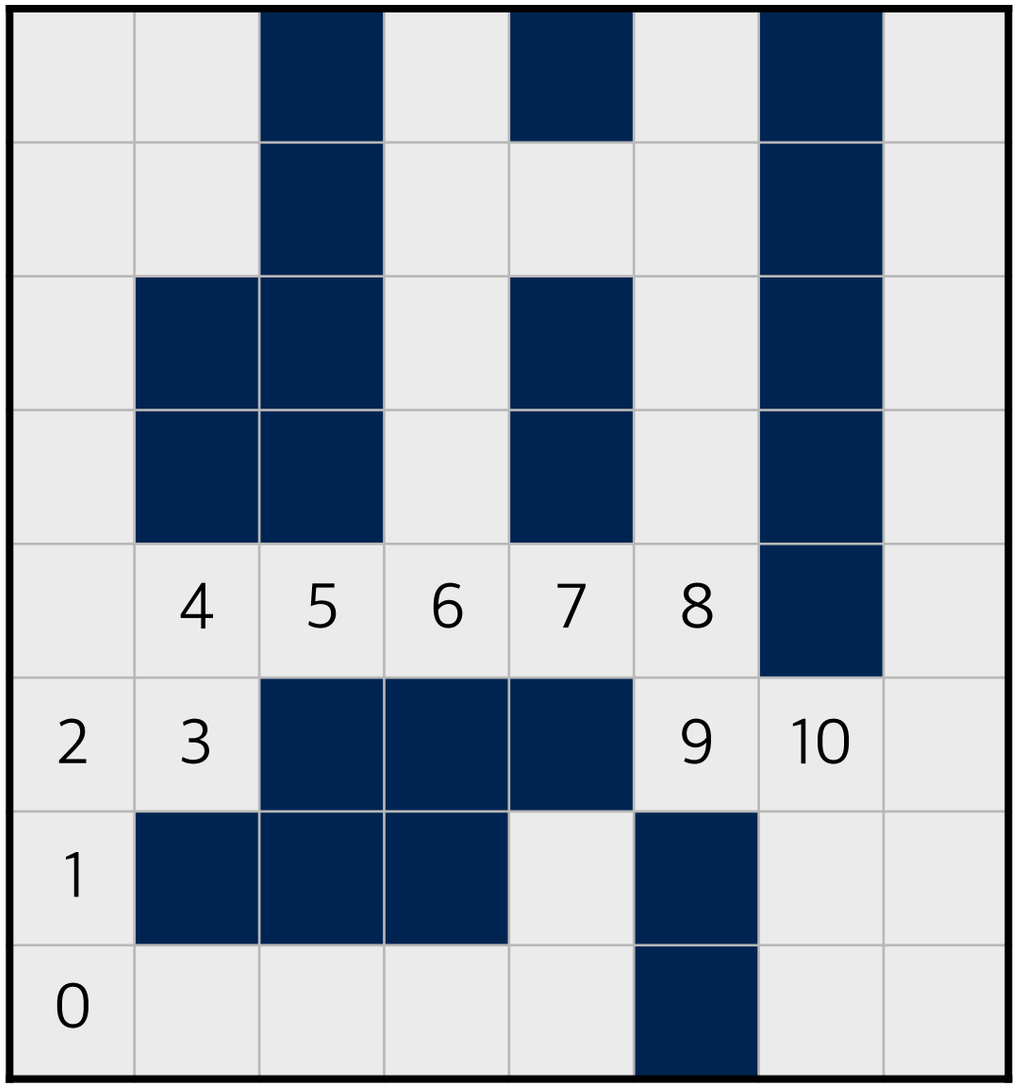
2. 각 노드에, 그 노드에 도착하기 위한 최단거리를 저장

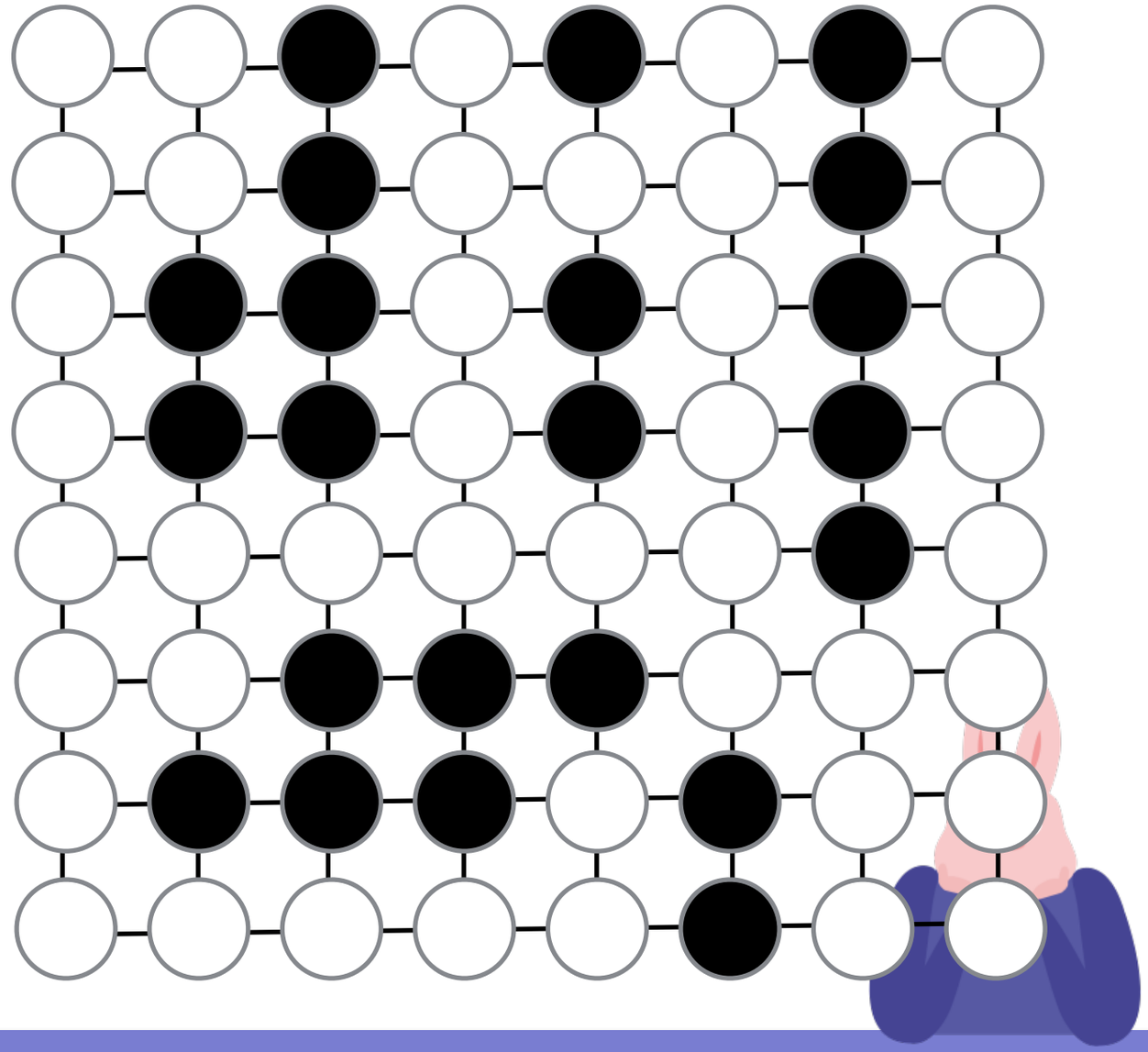




DFS와 BFS의 응용 : 미로찾기

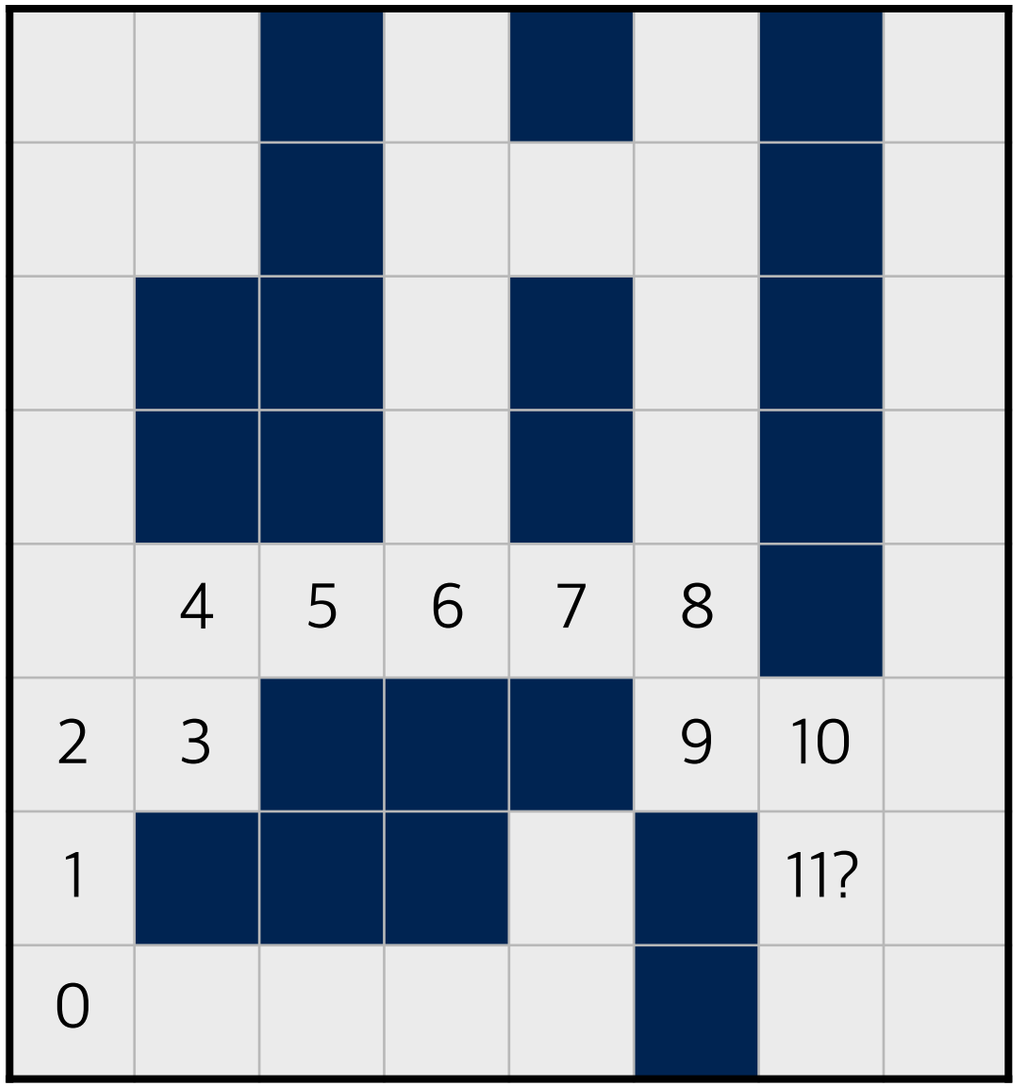
2. 각 노드에, 그 노드에 도착하기 위한 최단거리를 저장

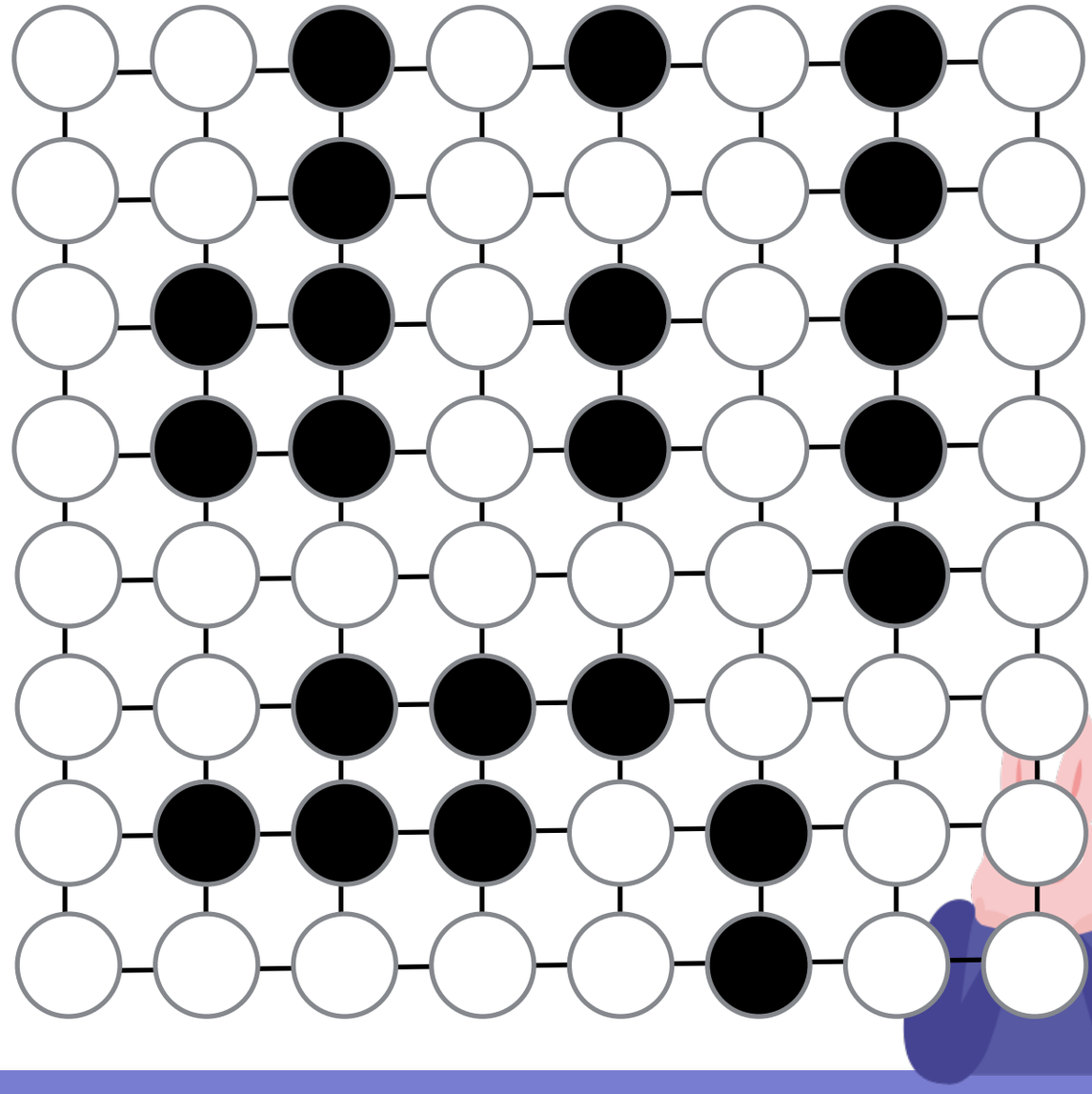




DFS와 BFS의 응용 : 미로찾기

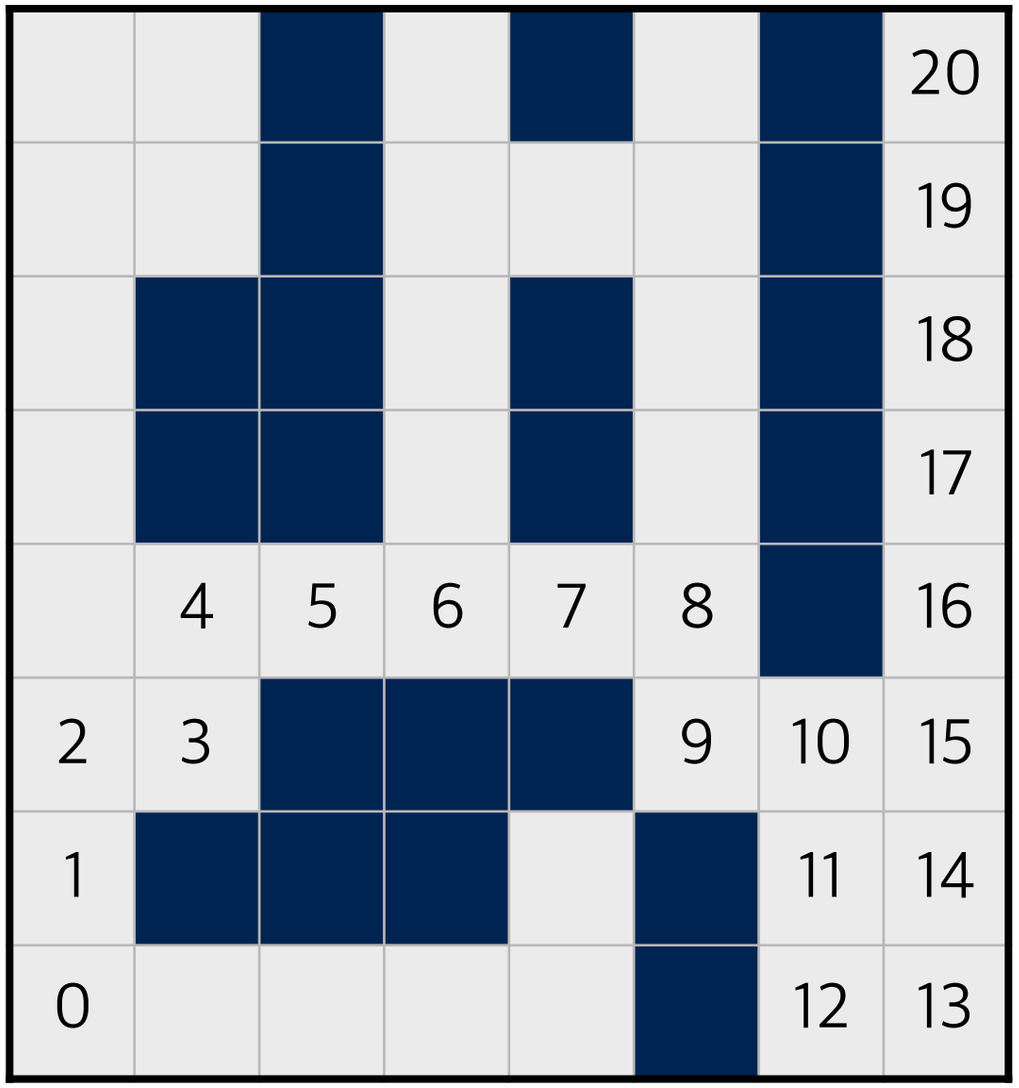
2. 각 노드에, 그 노드에 도착하기 위한 최단거리를 저장

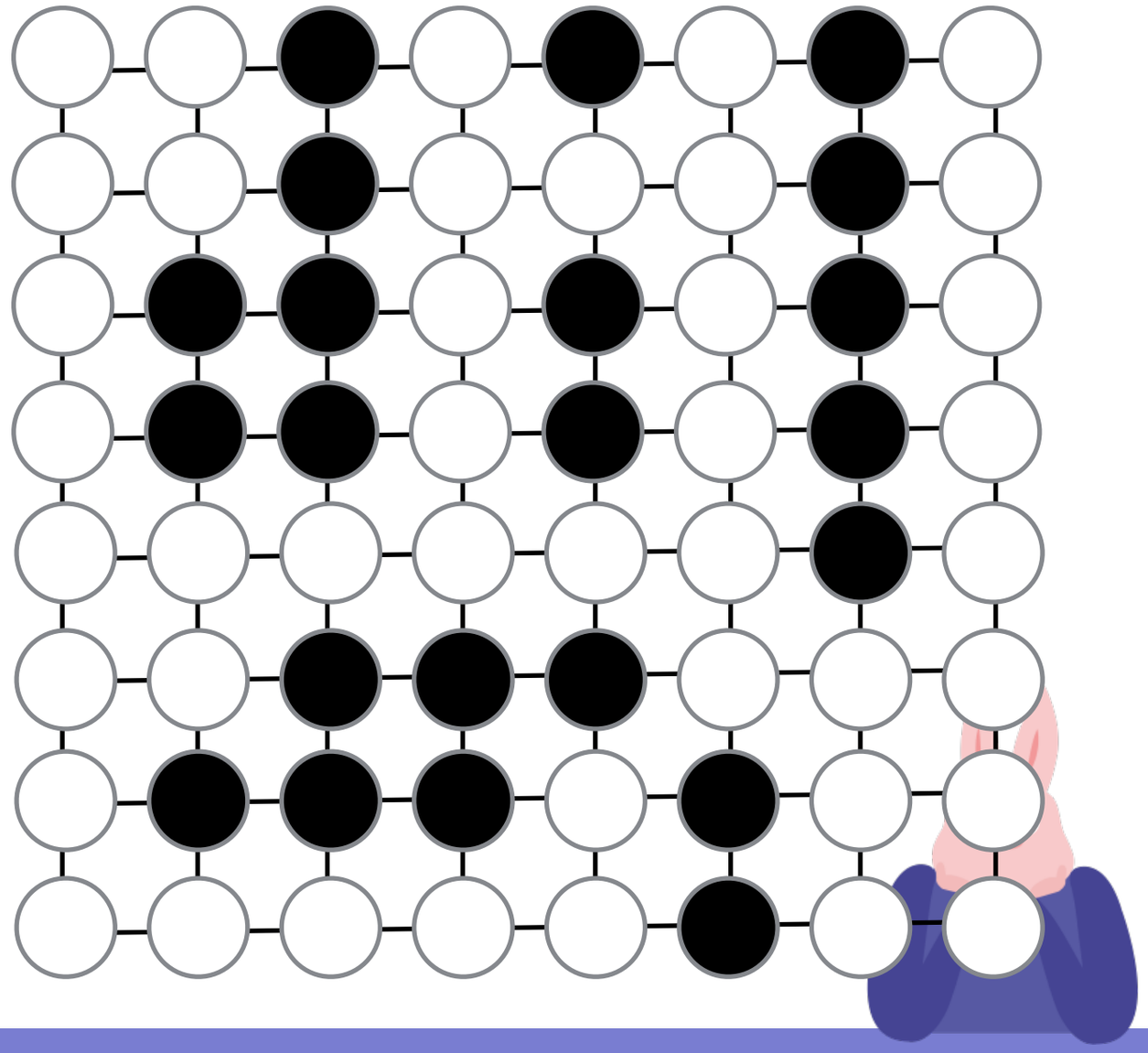




DFS와 BFS의 응용 : 미로찾기

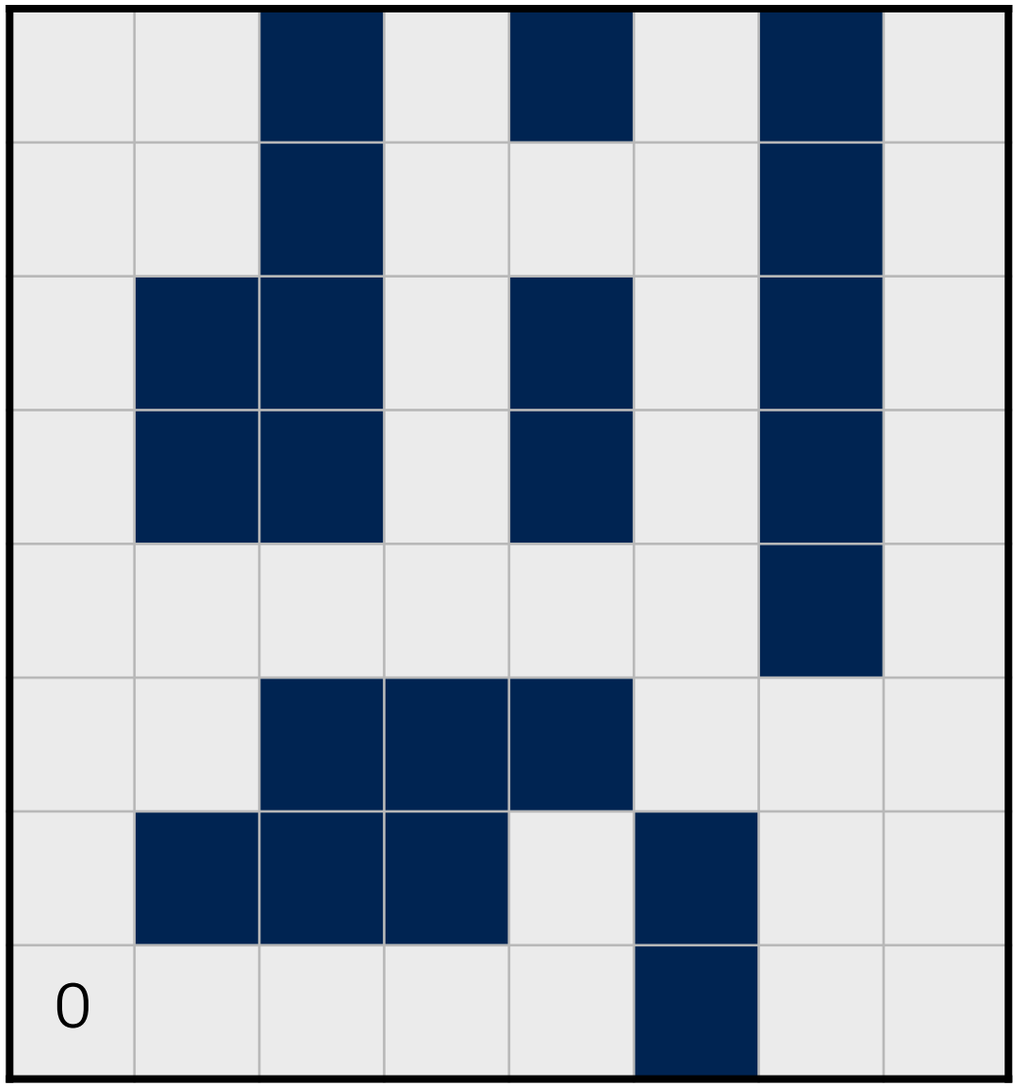
2. 각 노드에, 그 노드에 도착하기 위한 최단거리를 저장

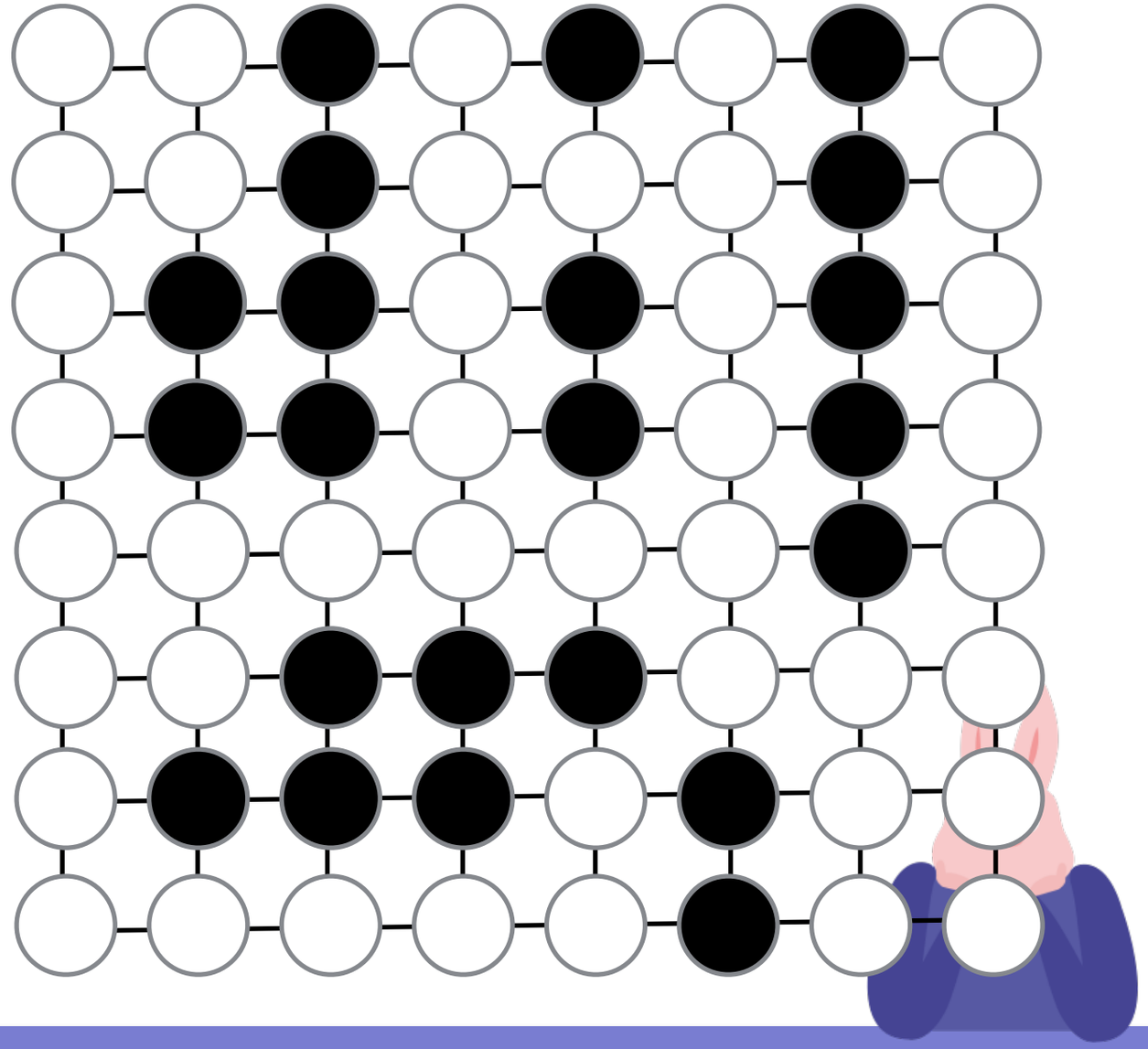




DFS와 BFS의 응용 : 미로찾기

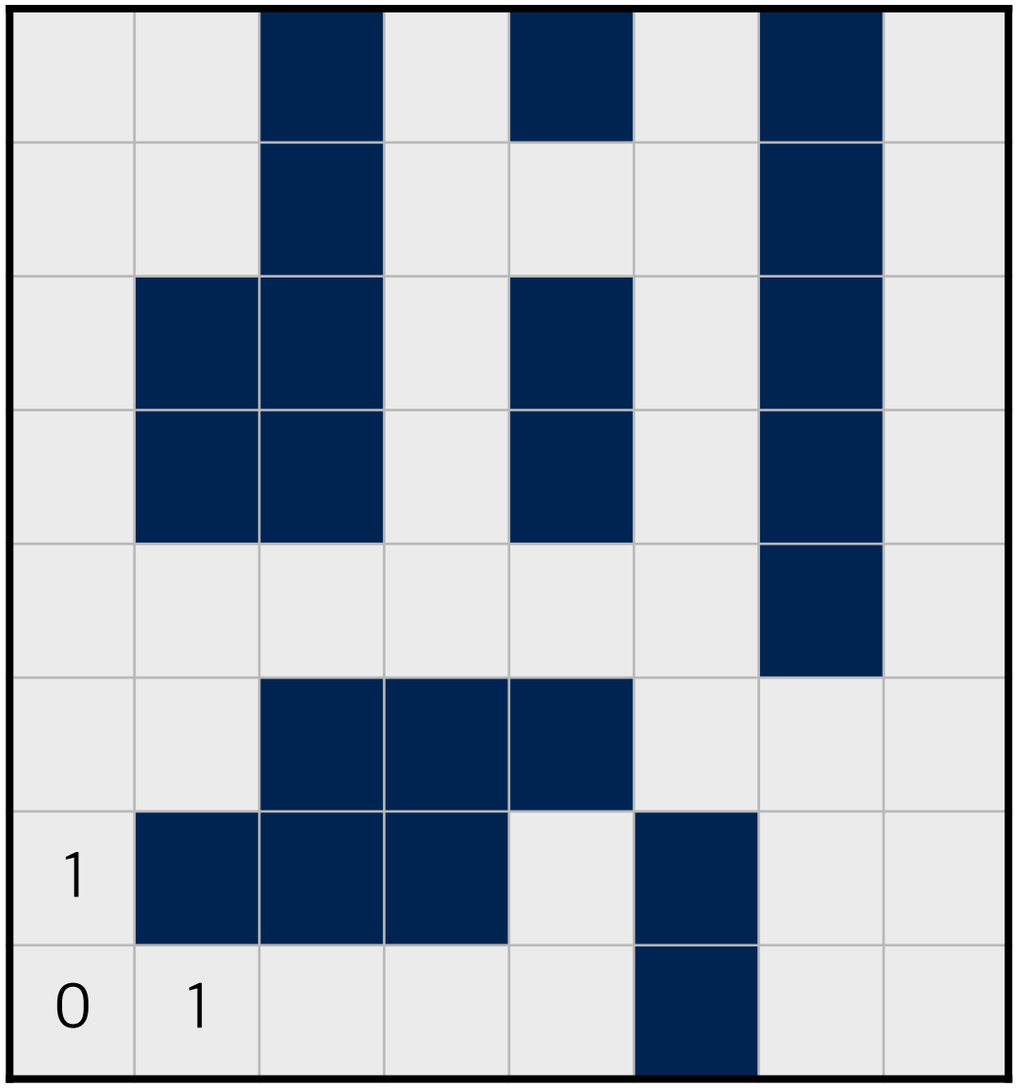
2. 각 노드에, 그 노드에 도착하기 위한 최단거리를 저장

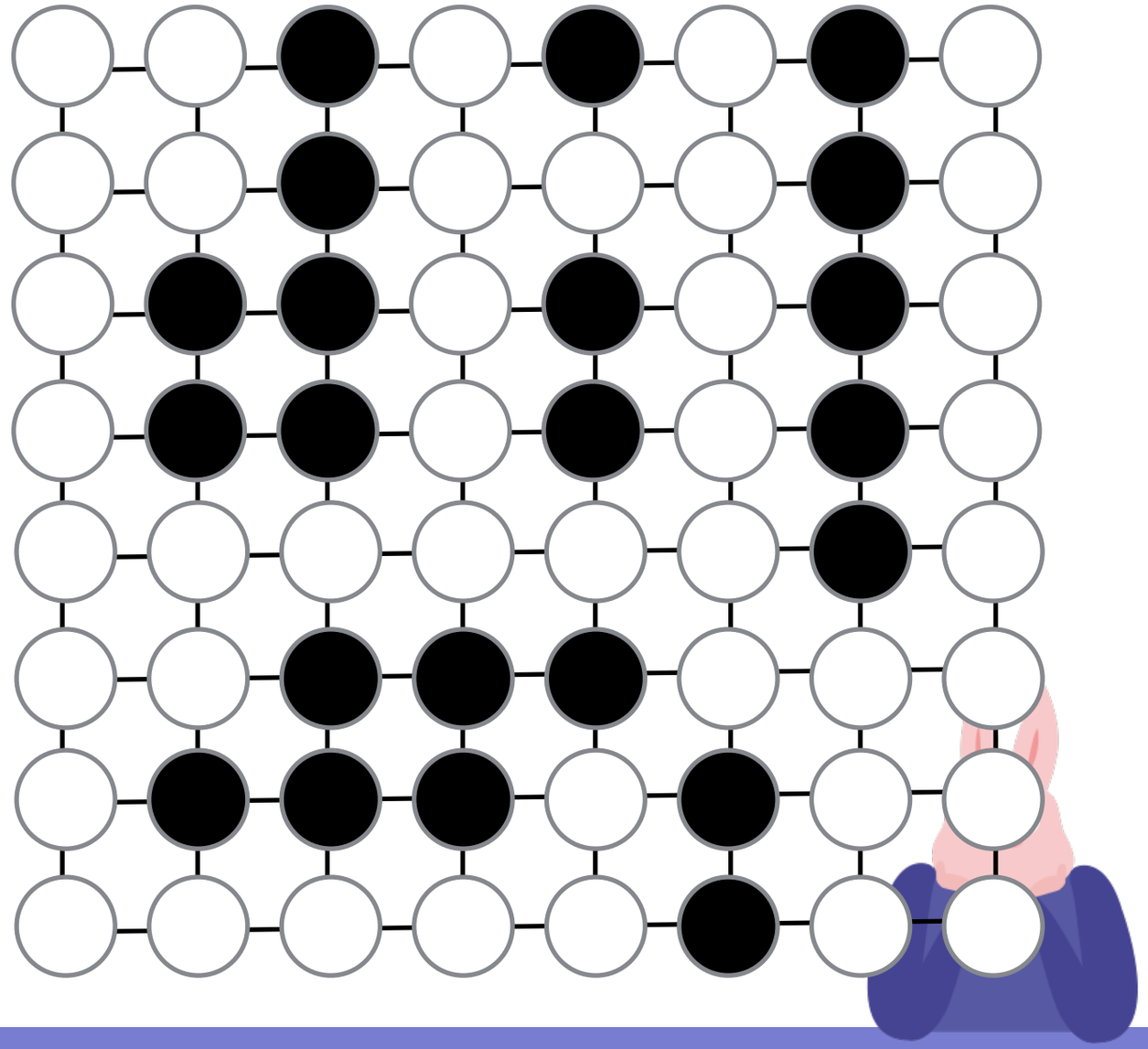




DFS와 BFS의 응용 : 미로찾기

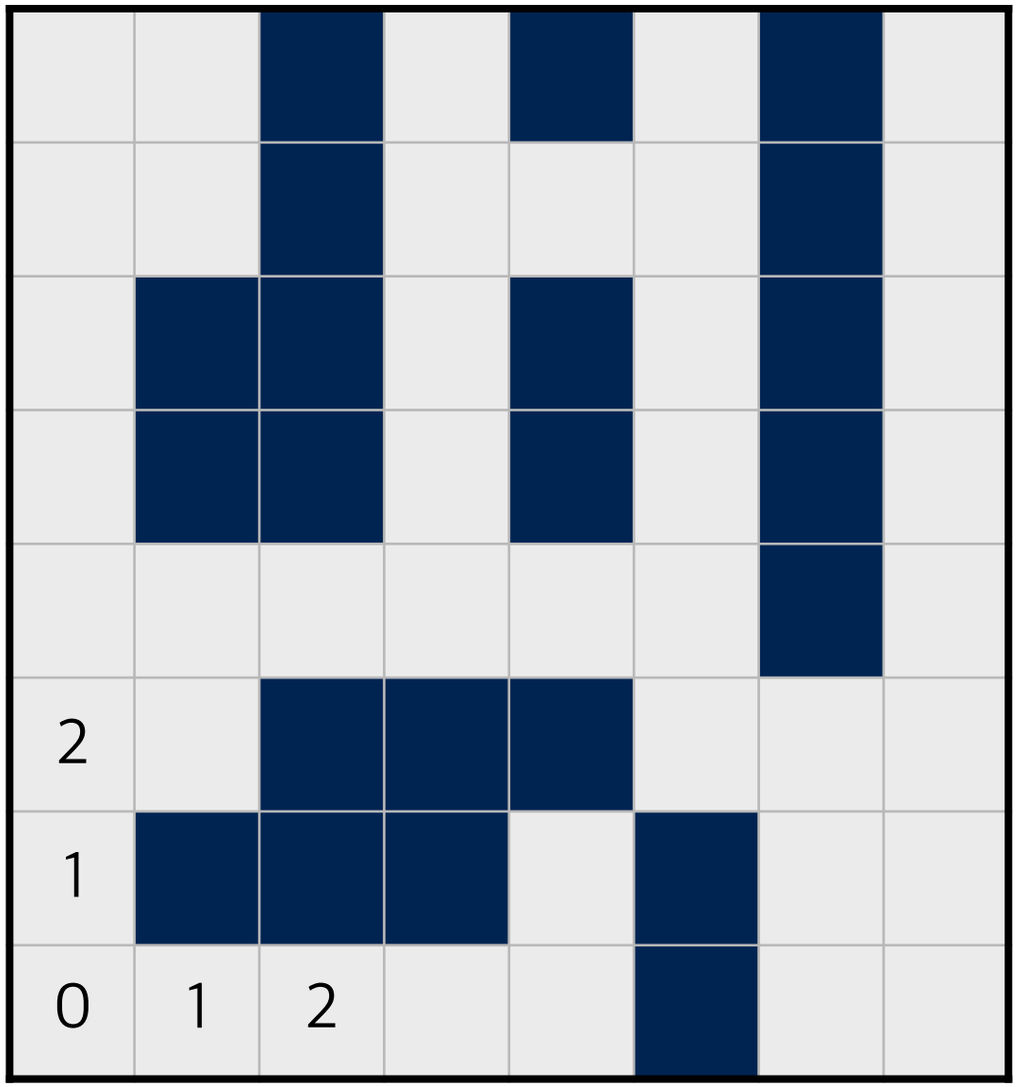
2. 각 노드에, 그 노드에 도착하기 위한 최단거리를 저장

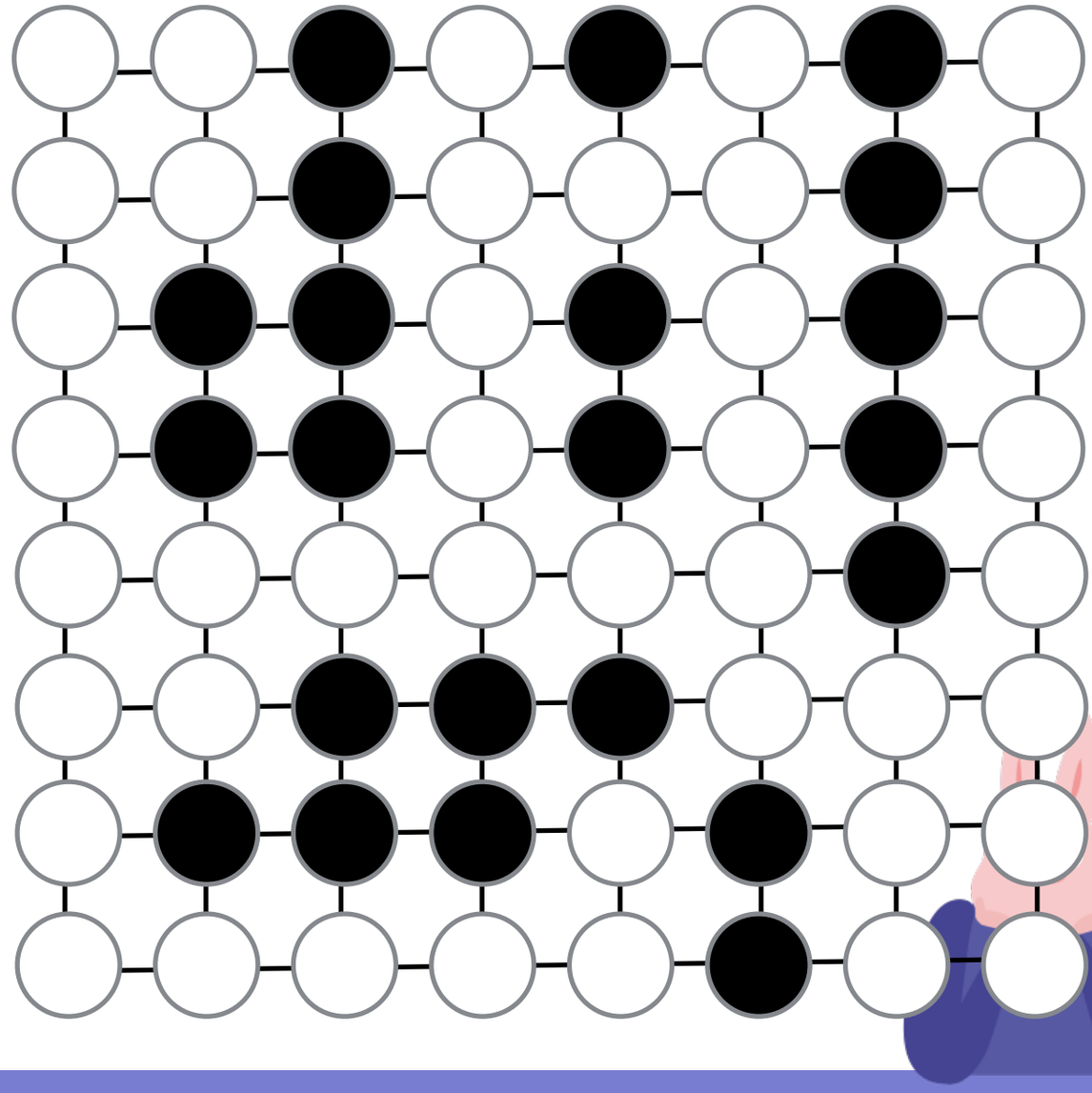




DFS와 BFS의 응용 : 미로찾기

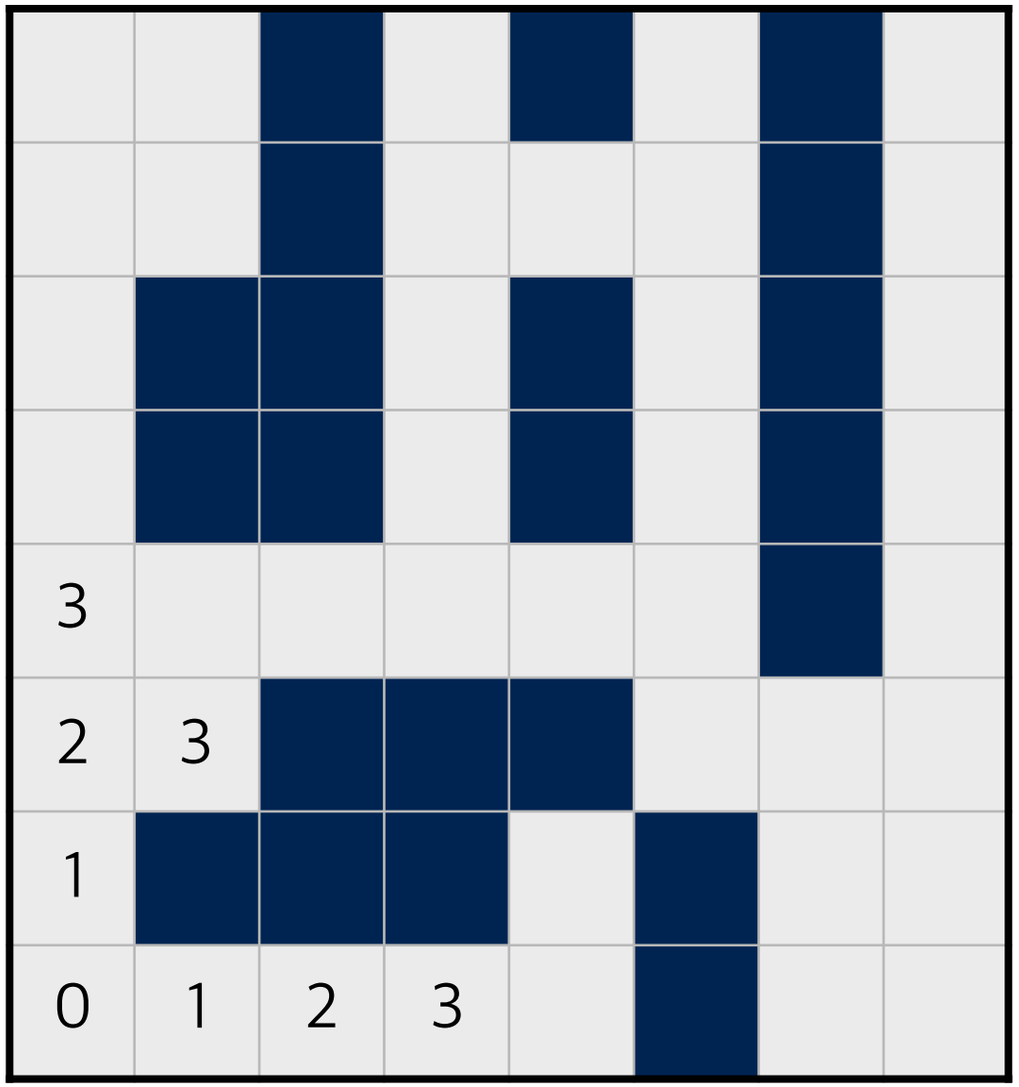
2. 각 노드에, 그 노드에 도착하기 위한 최단거리를 저장

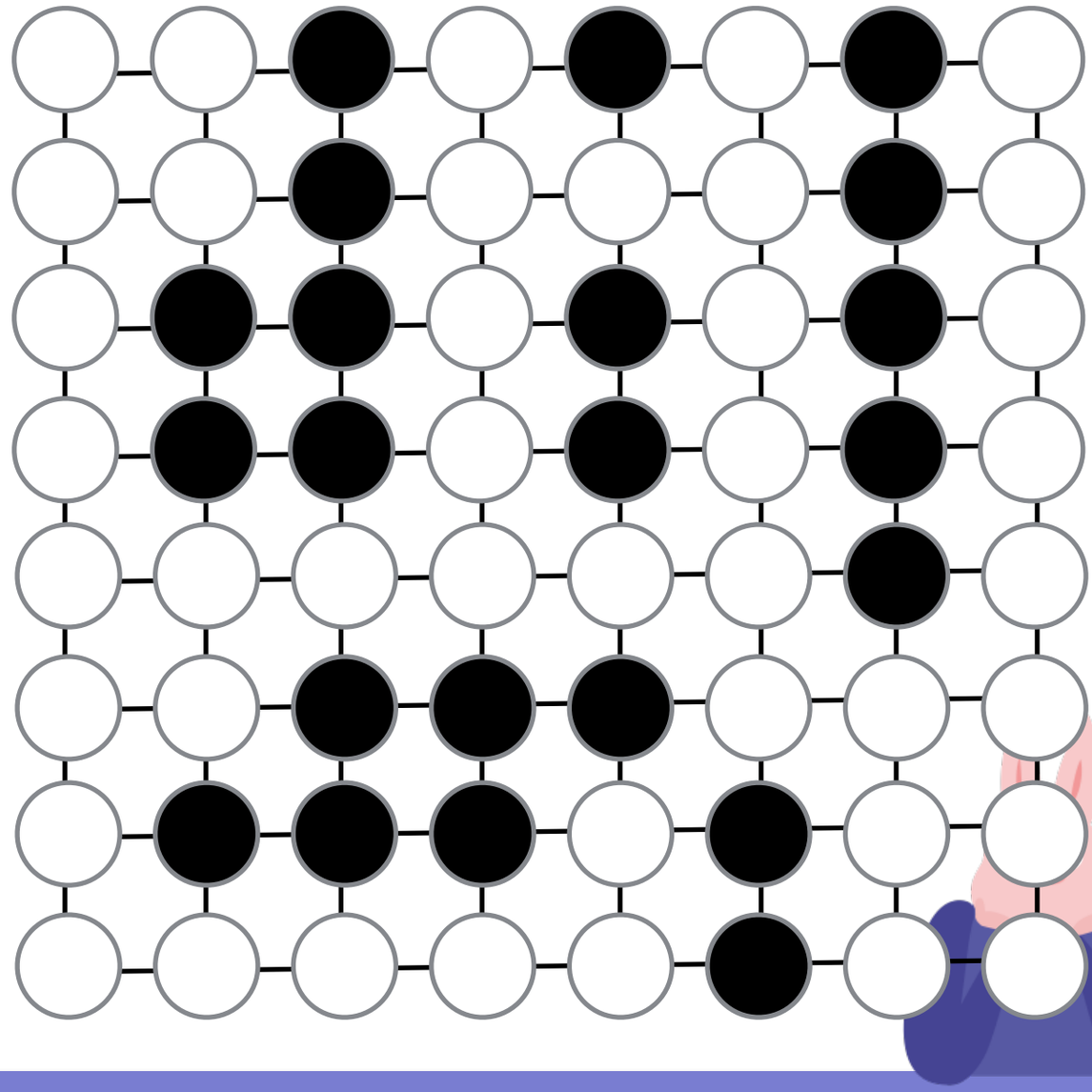




DFS와 BFS의 응용 : 미로찾기

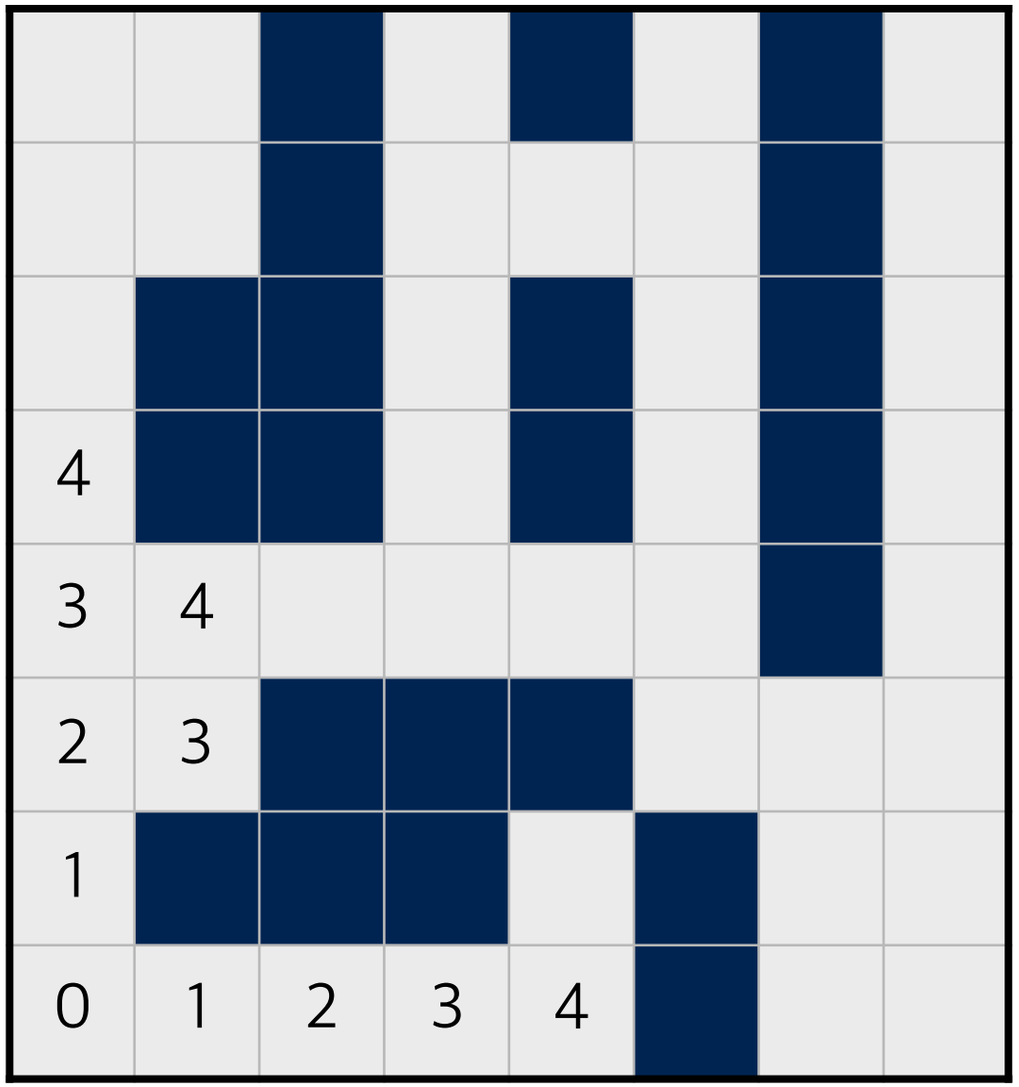
2. 각 노드에, 그 노드에 도착하기 위한 최단거리를 저장

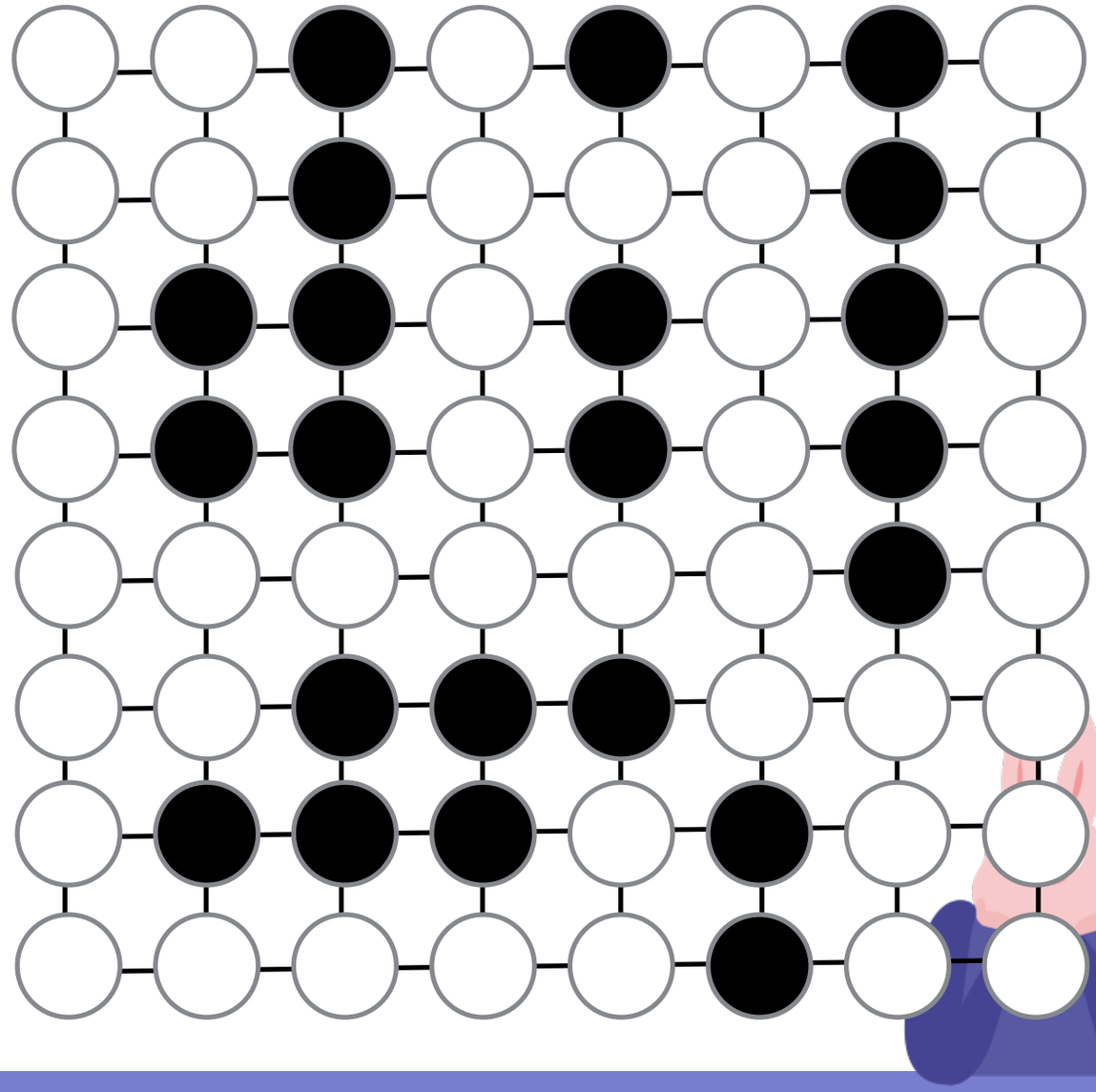




DFS와 BFS의 응용 : 미로찾기

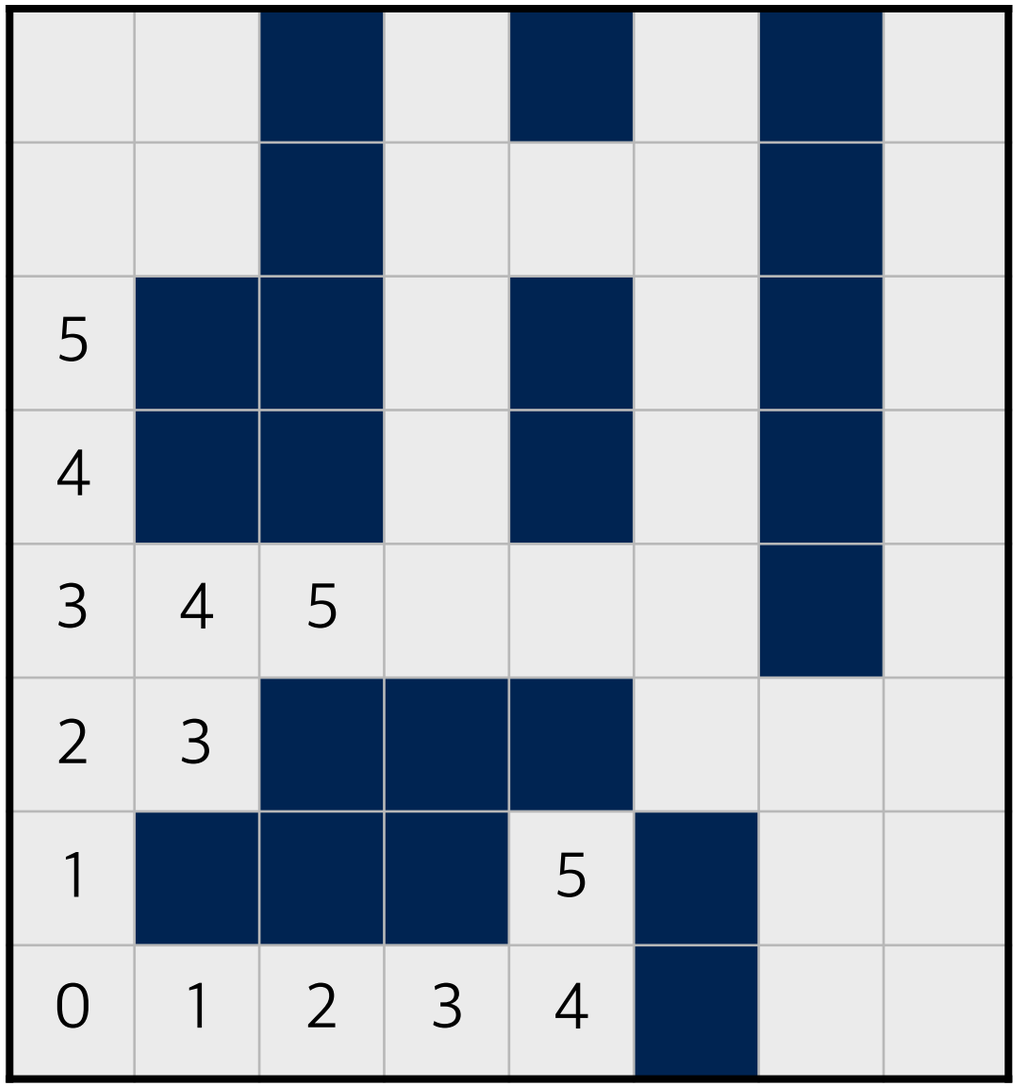
2. 각 노드에, 그 노드에 도착하기 위한 최단거리를 저장

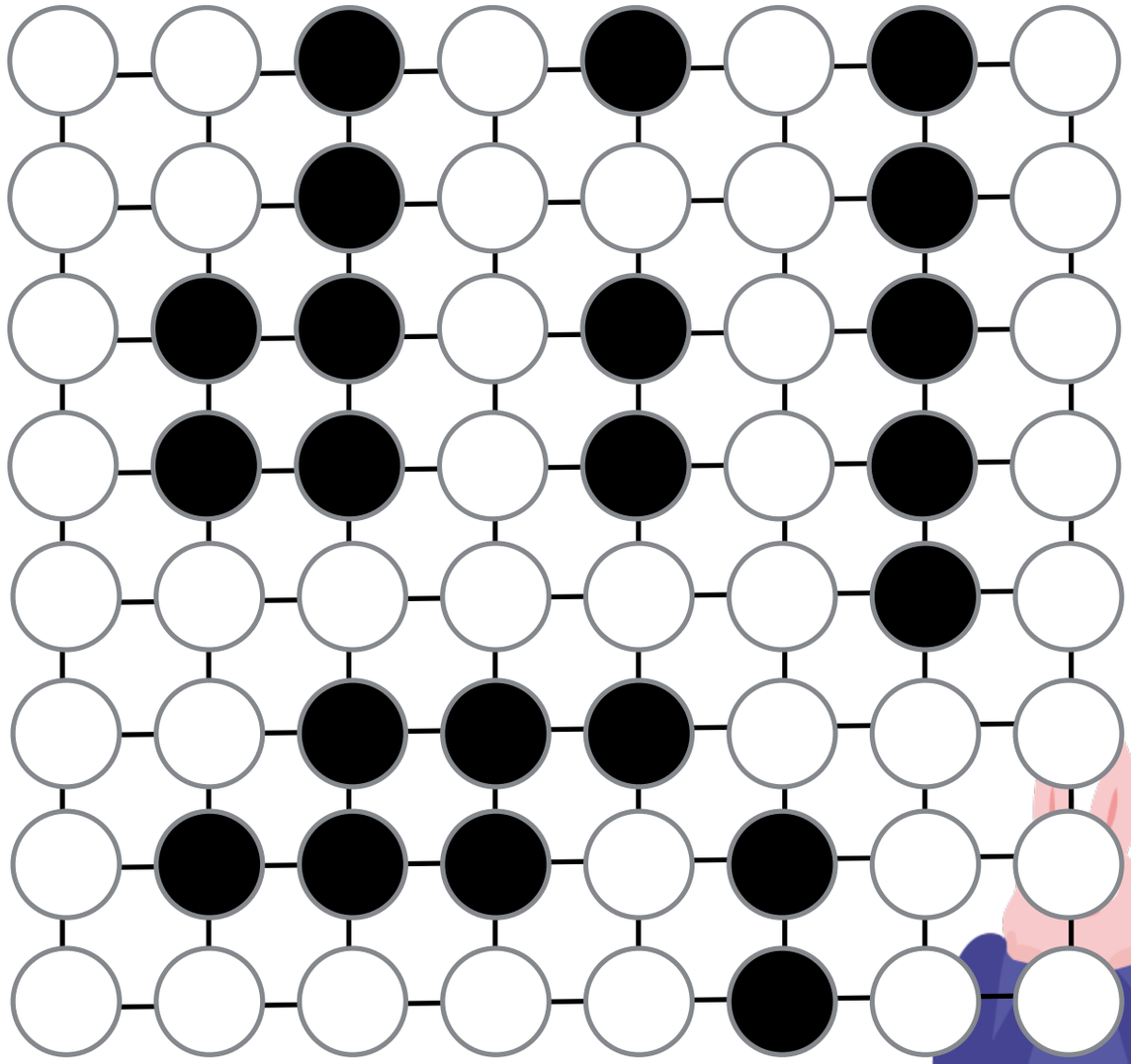




DFS와 BFS의 응용 : 미로찾기

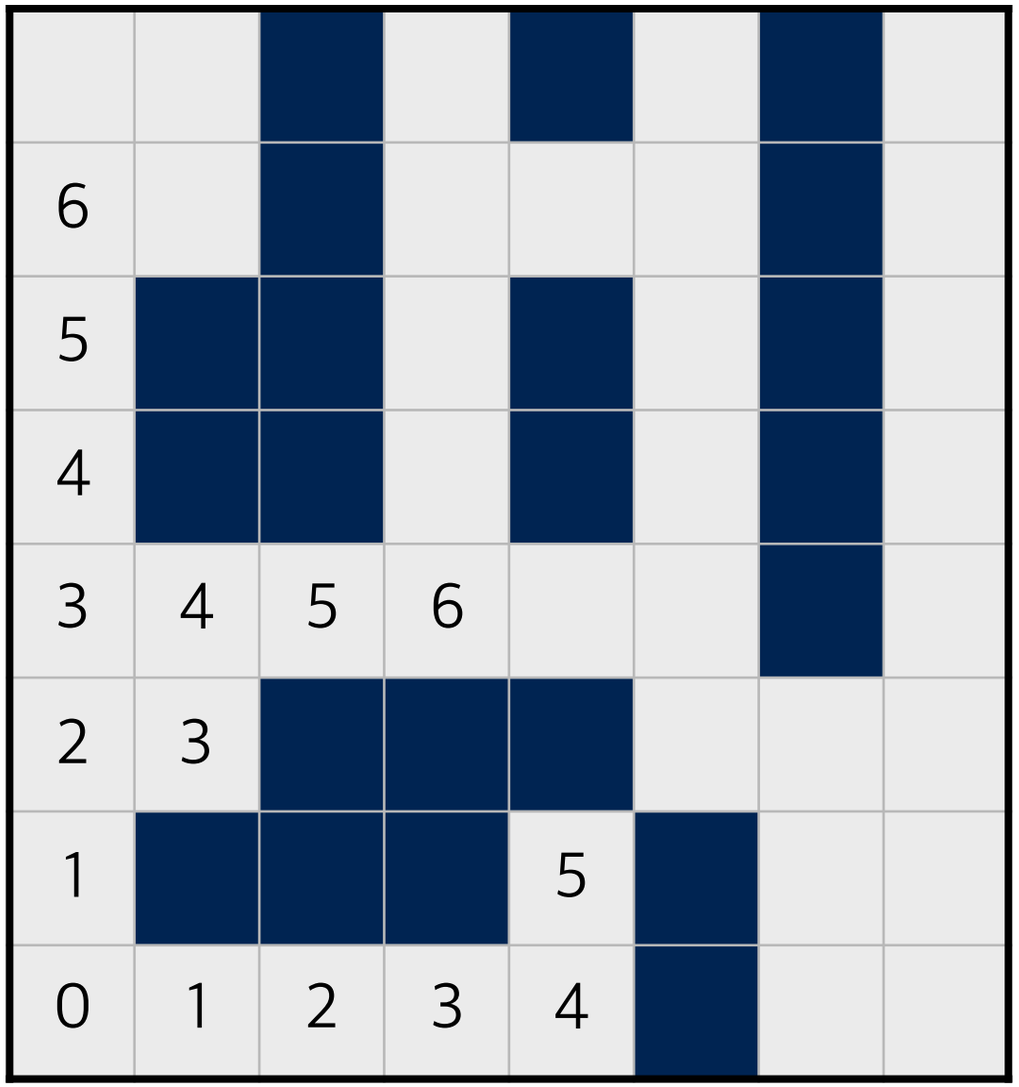
2. 각 노드에, 그 노드에 도착하기 위한 최단거리를 저장

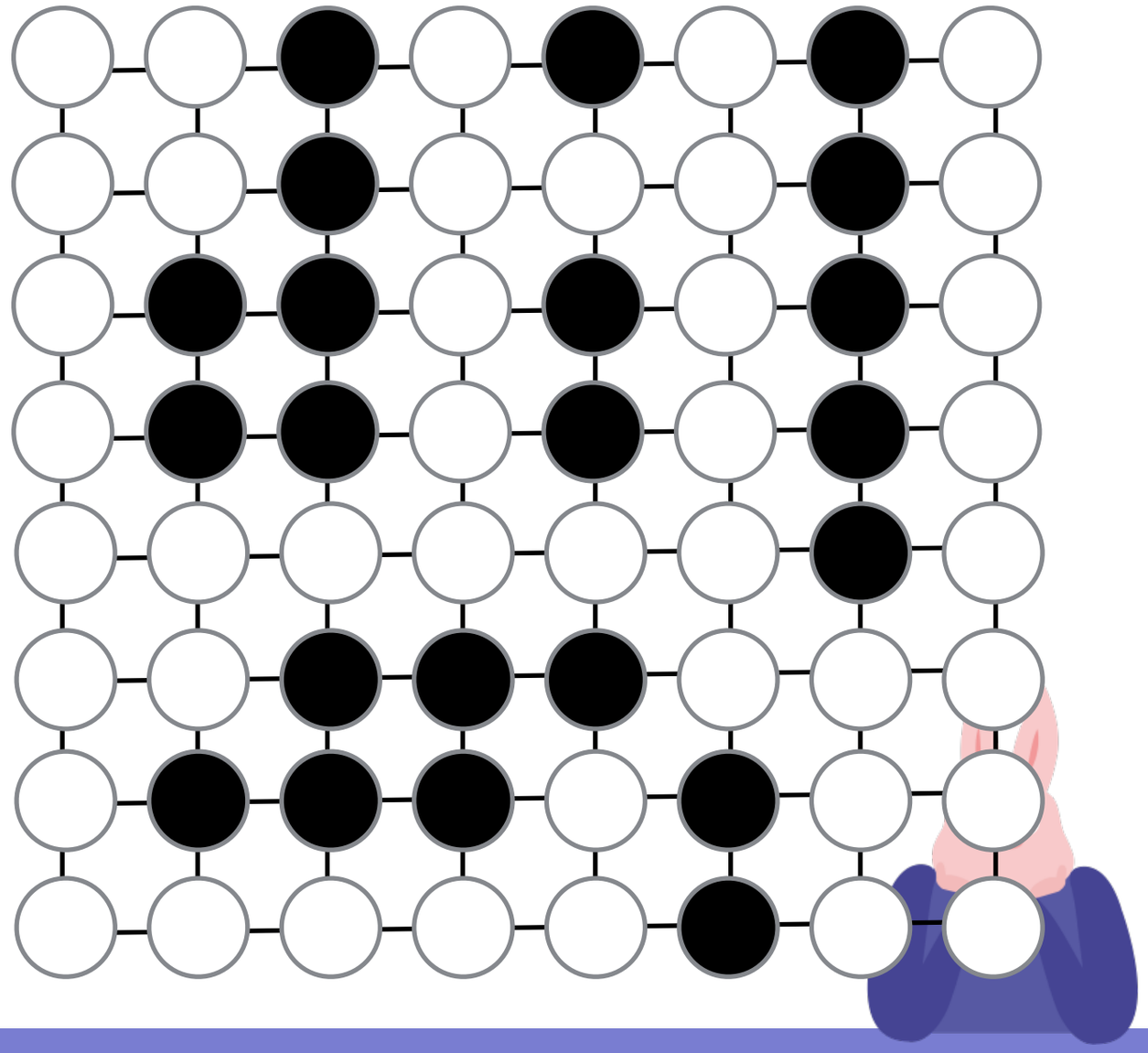




DFS와 BFS의 응용 : 미로찾기

2. 각 노드에, 그 노드에 도착하기 위한 최단거리를 저장

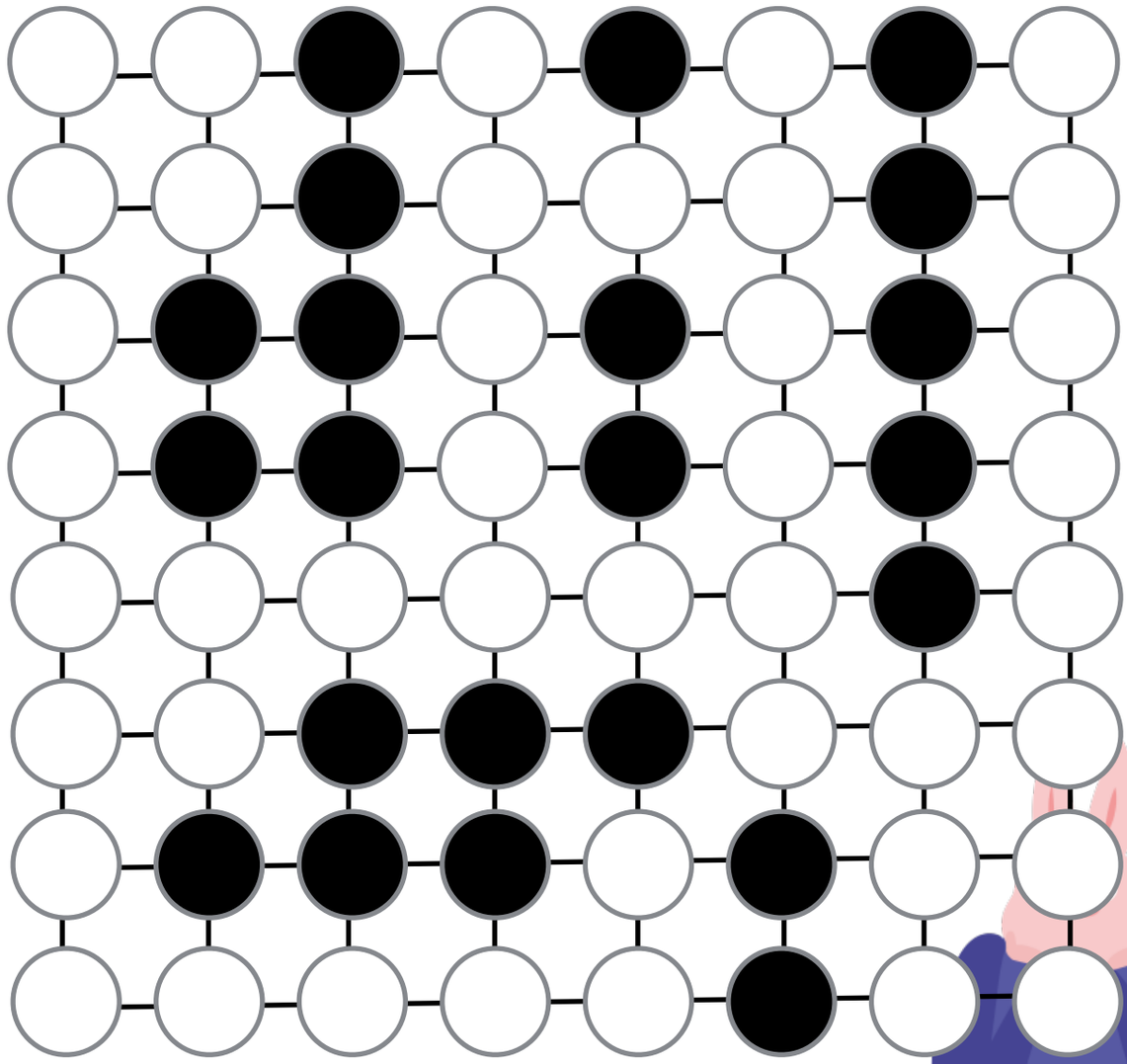




DFS와 BFS의 응용 : 미로찾기

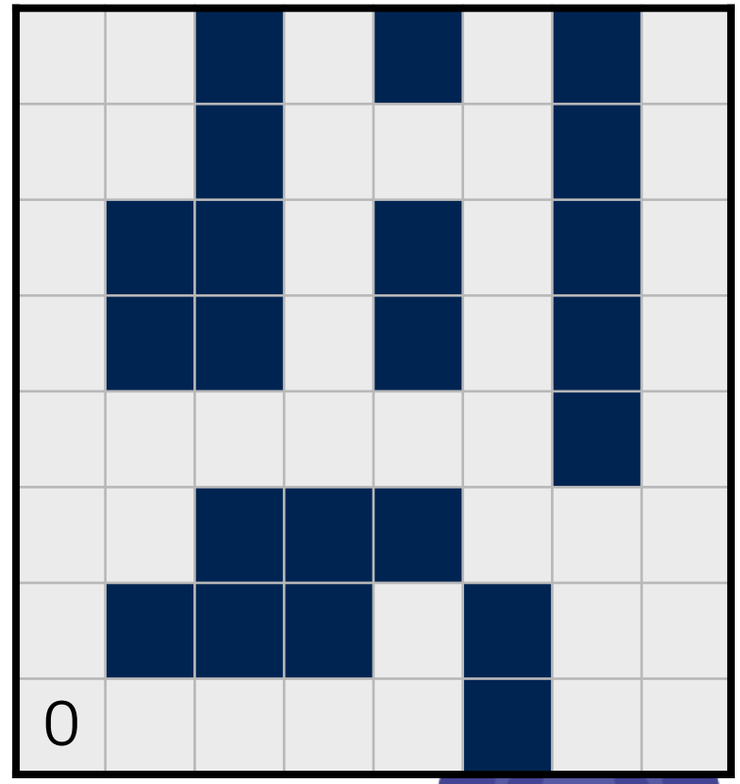
2. 각 노드에, 그 노드에 도착하기 위한 최단거리를 저장

7	8		10		12		16
6	7		9	10	11		15
5			8		10		14
4			7		9		13
3	4	5	6	7	8		12
2	3				9	10	11
1				5		11	12
0	1	2	3	4		12	13



BFS의 응용 : 미로찾기 (정확성 증명)

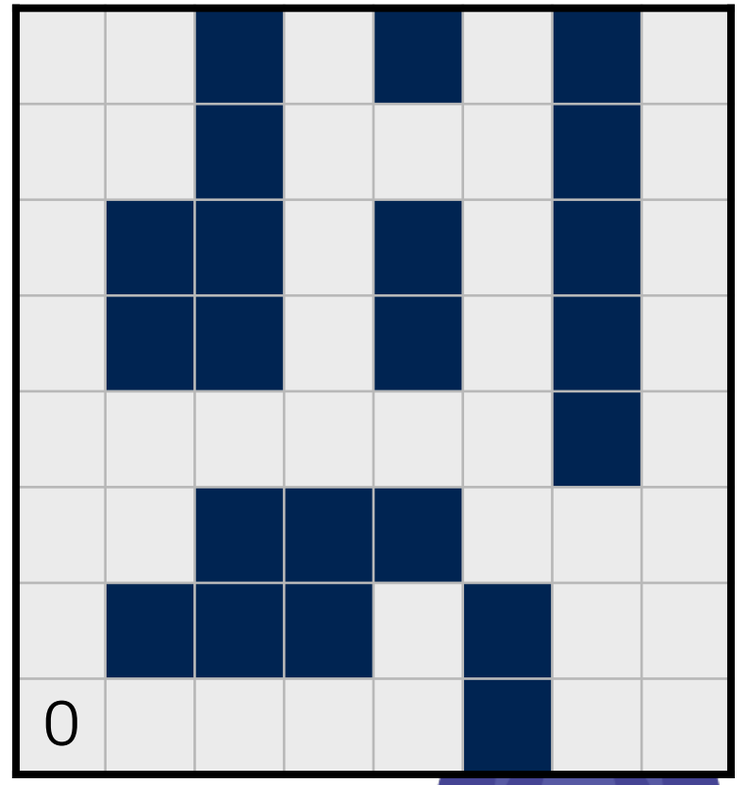
- 주장 : BFS를 이용하여 방문하면 **최단거리**를 구할 수 있다



BFS의 응용 : 미로찾기 (정확성 증명)

- 주장 : BFS를 이용하여 방문하면 **최단거리**를 구할 수 있다

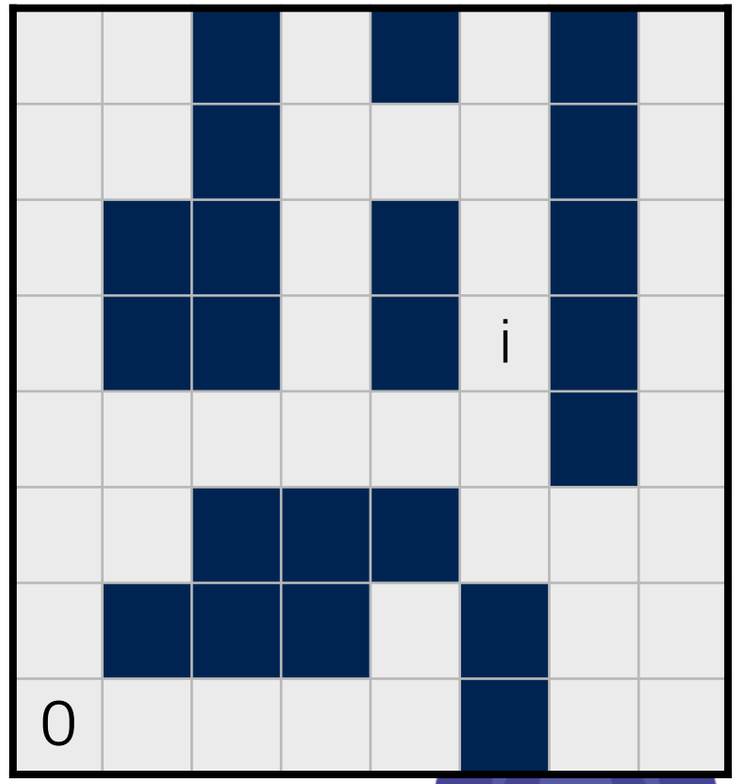
- 증명 (귀류법)



BFS의 응용 : 미로찾기 (정확성 증명)

- 주장 : BFS를 이용하여 방문하면 **최단거리**를 구할 수 있다

- 증명 (귀류법)
  - 만약 최단거리를 구하지 못했다고 가정하자
    그리고, 최단거리를 구하지 못한 칸을 i라 하자

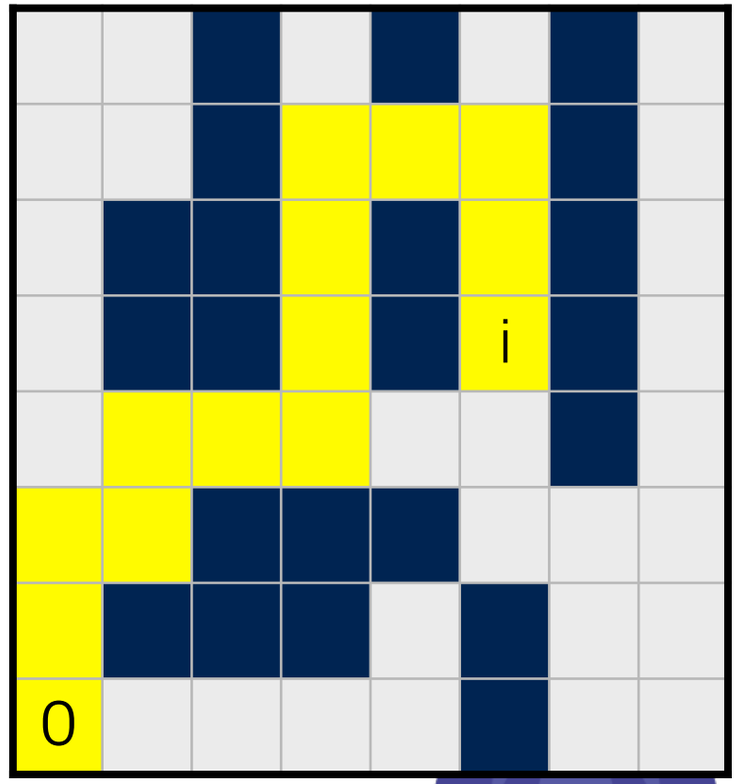


BFS의 응용 : 미로찾기 (정확성 증명)

- 주장 : BFS를 이용하여 방문하면 **최단거리**를 구할 수 있다

- 증명 (귀류법)
  - 만약 최단거리를 구하지 못했다고 가정하자
    그리고, 최단거리를 구하지 못한 칸을 i라 하자

  - 현재 i 까지 따라온 경로를 P(i)라 하자
    최단거리를 구하지 못했으므로, P(i)는 최단경로가 아니다.



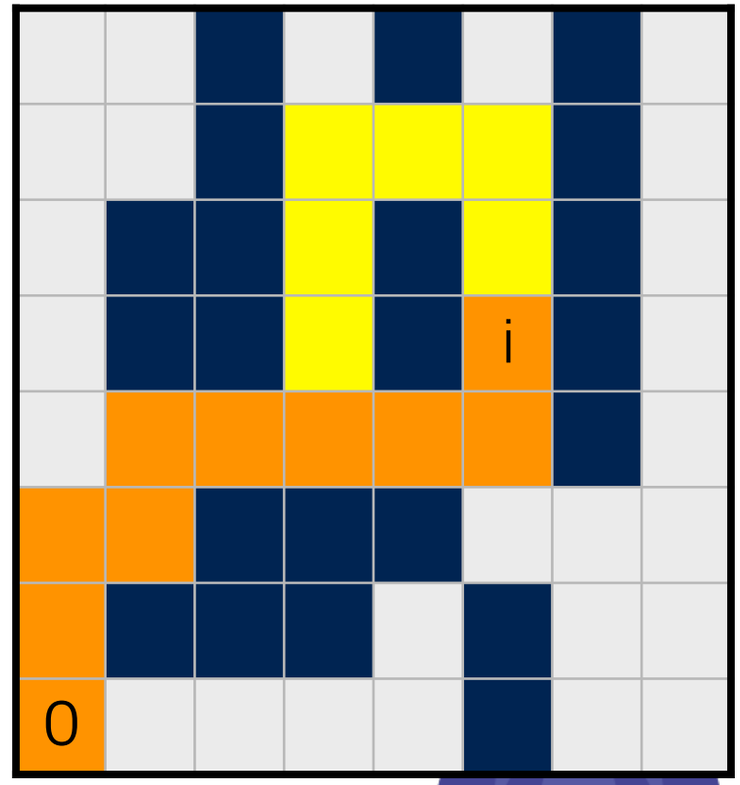
BFS의 응용 : 미로찾기 (정확성 증명)

- 주장 : BFS를 이용하여 방문하면 **최단거리**를 구할 수 있다

- 증명 (귀류법)
  - 만약 최단거리를 구하지 못했다고 가정하자
    그리고, 최단거리를 구하지 못한 칸을 i라 하자

  - 현재 i 까지 따라온 경로를 P(i)라 하자
    최단거리를 구하지 못했으므로, P(i)는 최단경로가 아니다.

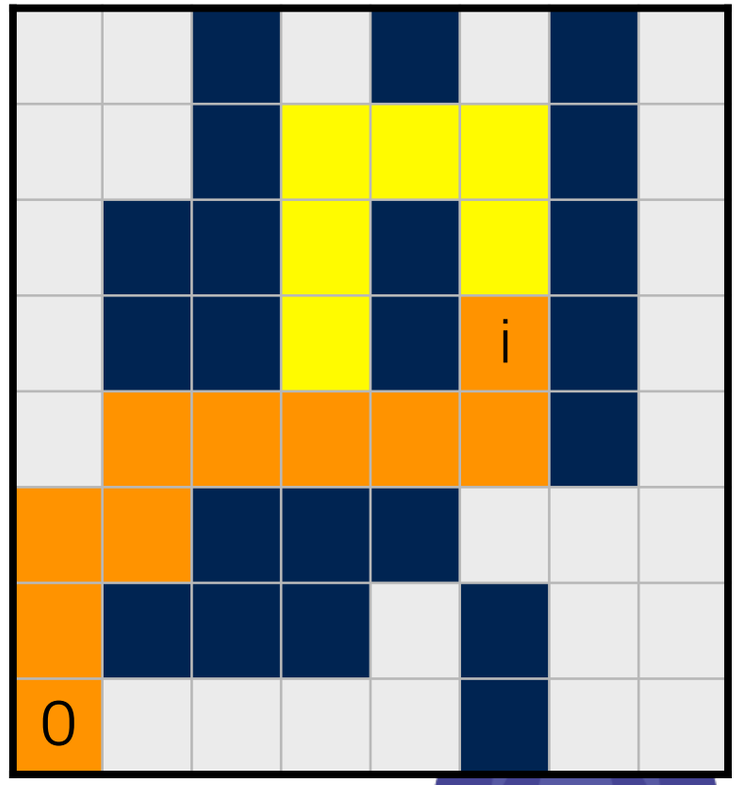
  - 구하지 못한 최단경로를 S(i)라 하자



BFS의 응용 : 미로찾기 (정확성 증명)

- 주장 : BFS를 이용하여 방문하면 **최단거리**를 구할 수 있다

- 증명 (귀류법)
  - 만약 최단거리를 구하지 못했다고 가정하자
    그리고, 최단거리를 구하지 못한 칸을 i라 하자
  - 현재 i 까지 따라온 경로를 P(i)라 하자
    최단거리를 구하지 못했으므로, P(i)는 최단경로가 아니다.
  - 구하지 못한 최단경로를 S(i)라 하자
  - BFS의 알고리즘에 따르면, P(i)를 S(i)보다 먼저 구할 수 없다
    따라서 가정은 모순이다



[활동문제 5] 미로찾기

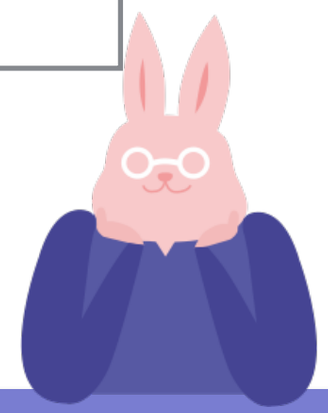
- 시작점에서 출발하여 끝점에 도달하기 위한 최단경로를 출력
  (시작점은 항상 왼쪽 아래, 끝점은 항상 오른쪽 위)

입력의 예

```
5 6
0 1 0 1 1 0
0 1 0 0 1 0
0 0 0 0 1 0
0 1 1 0 0 0
0 1 0 0 0 0
```

출력의 예

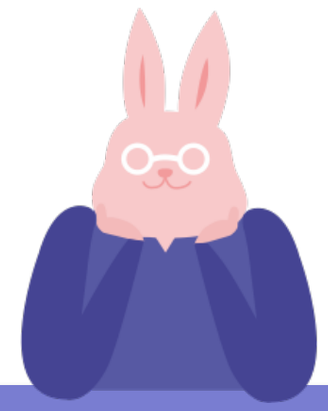
```
11
```



Graph 문제의 어려운 점

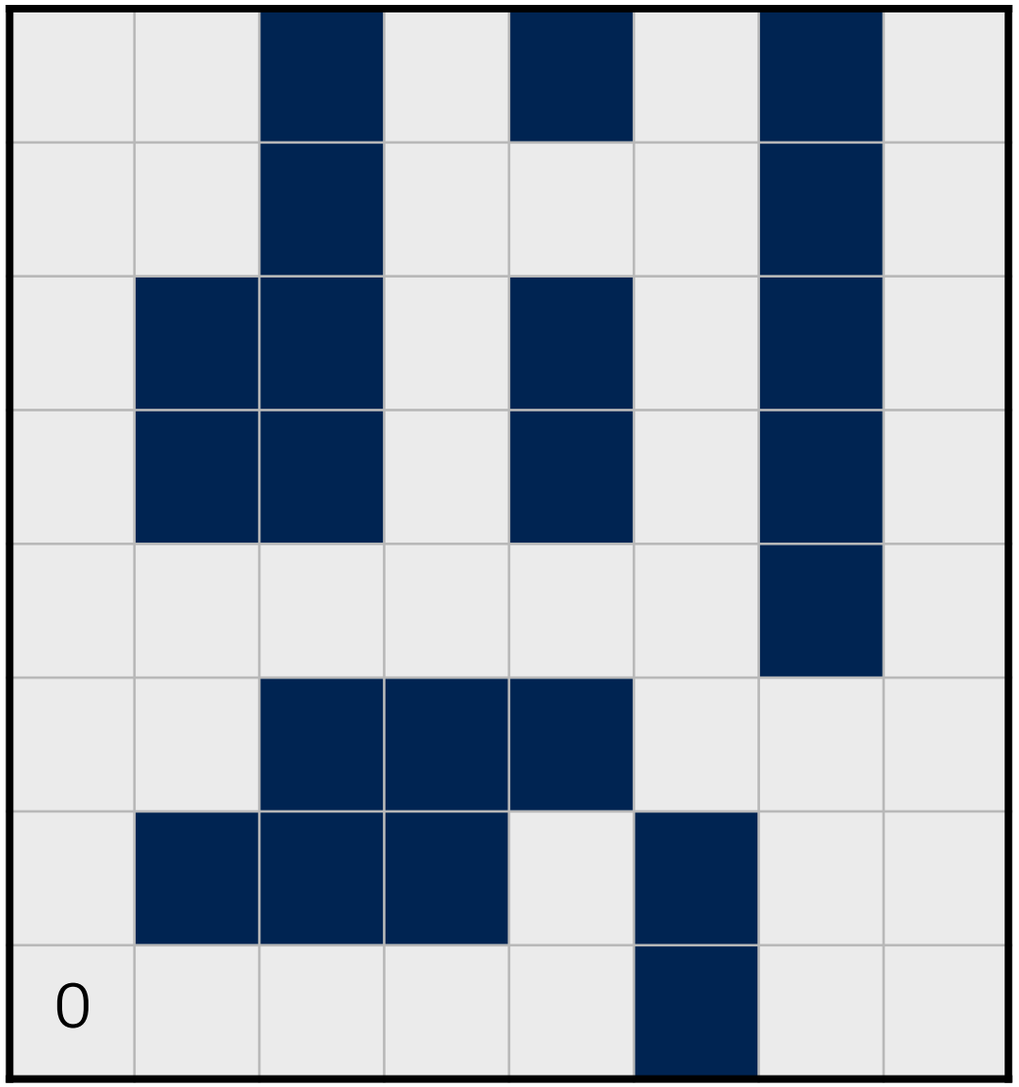
- 이 문제를 풀기 위해서 그래프를 어떻게 이용해야 하는가 ?

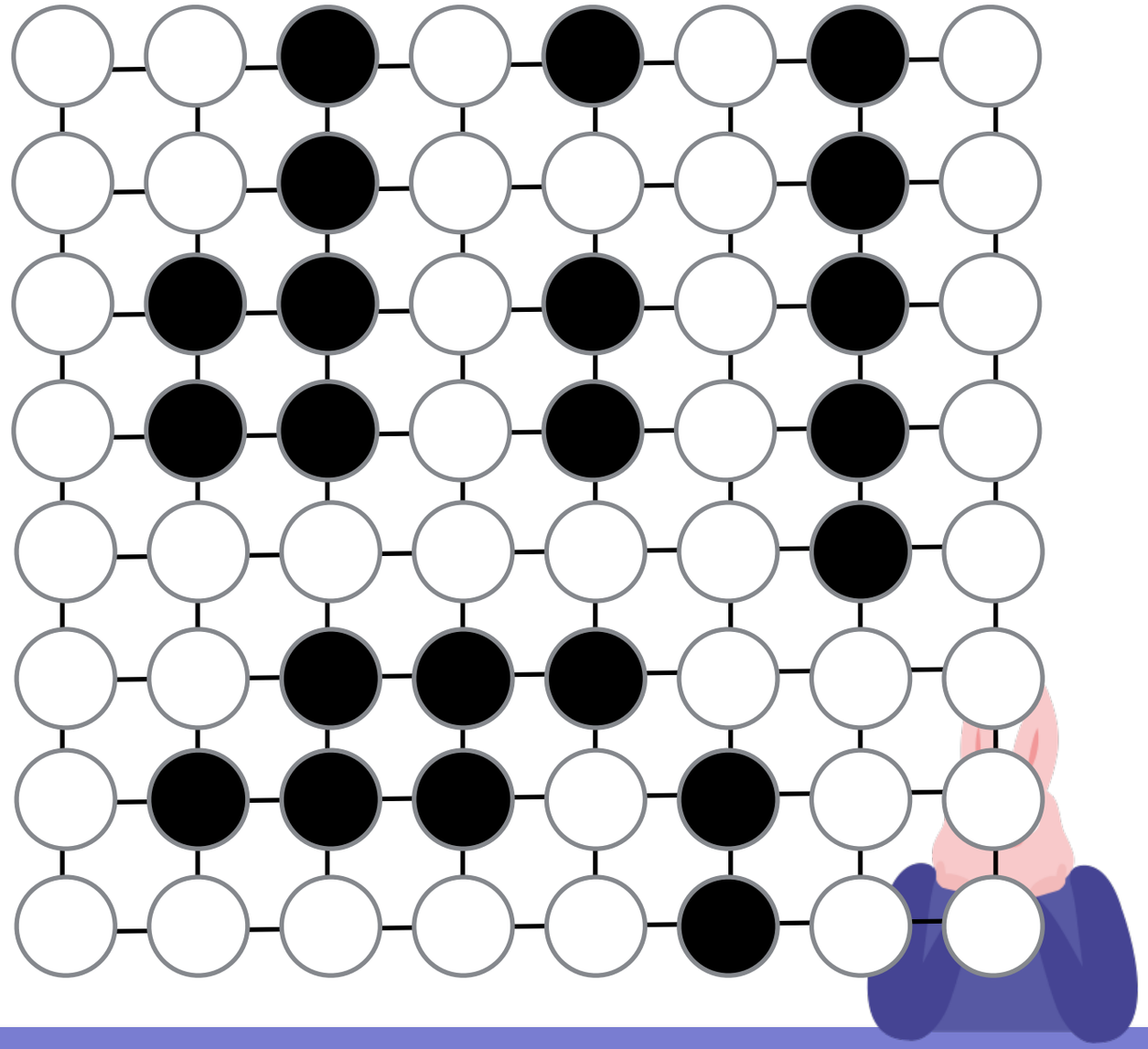
- **Modeling**
  - 정점은 무엇을 의미하는가 ?
  - 간선은 무엇을 의미하는가 ?



BFS의 응용 : Flood Fill

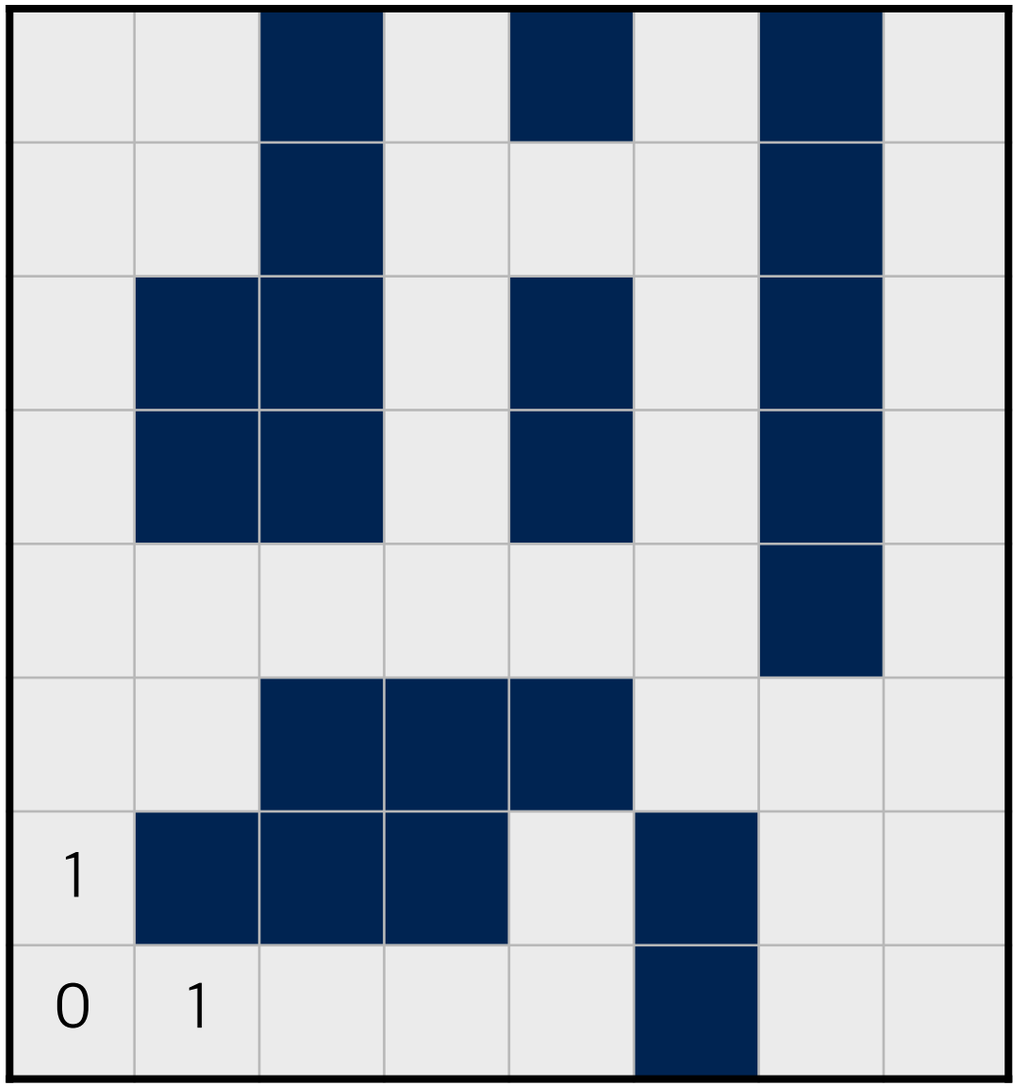
- 물이 차오르는 듯 하여 Flood fill 이라 부름

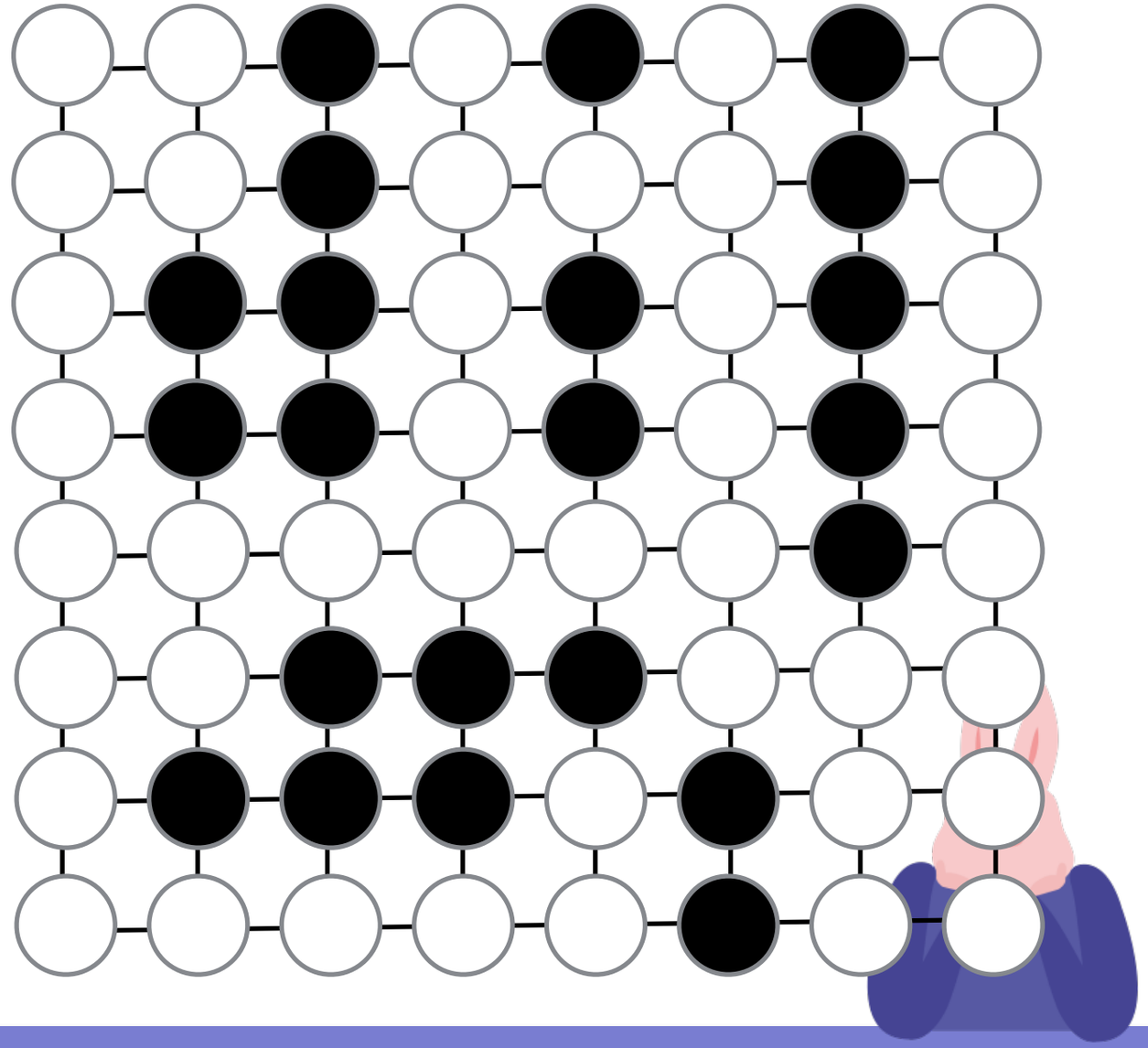




BFS의 응용 : Flood Fill

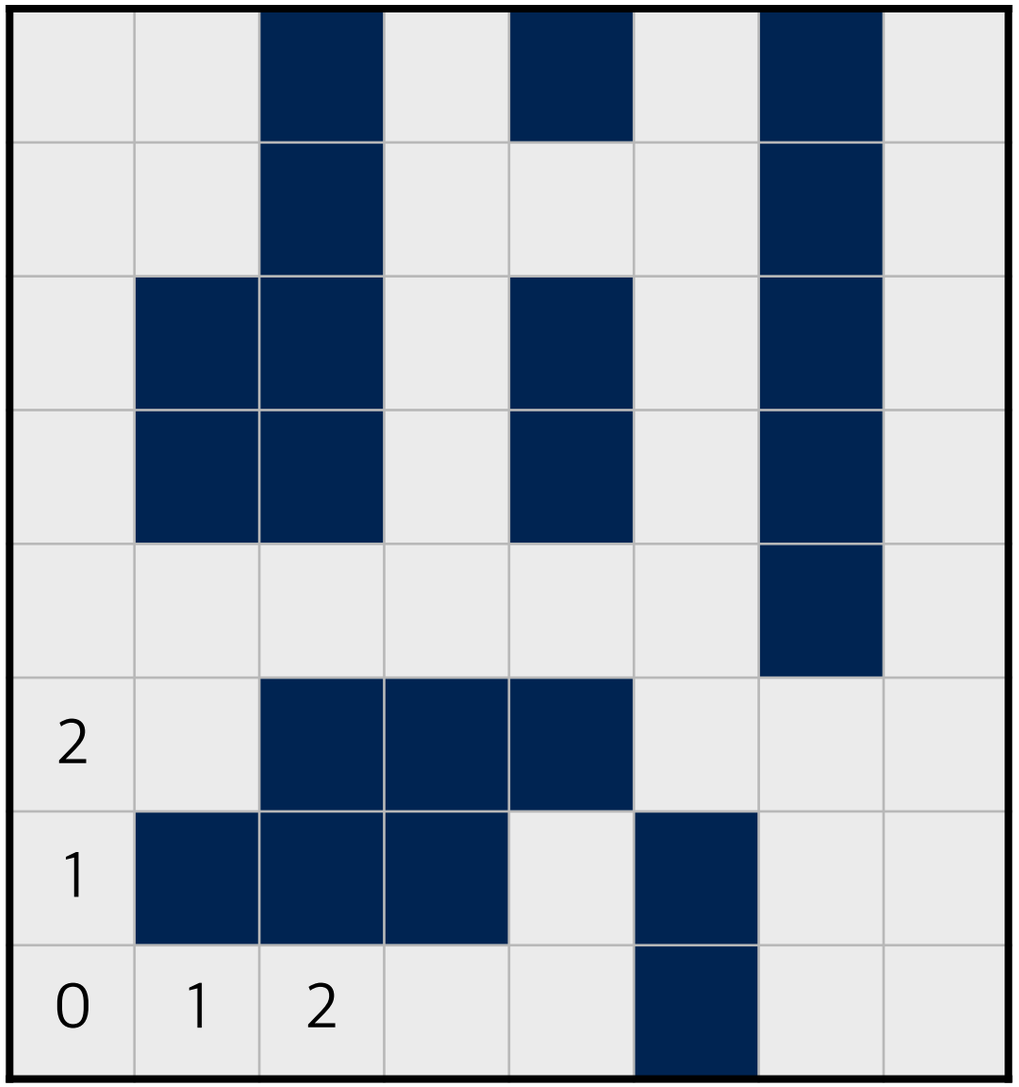
- 물이 차오르는 듯 하여 Flood fill 이라 부름

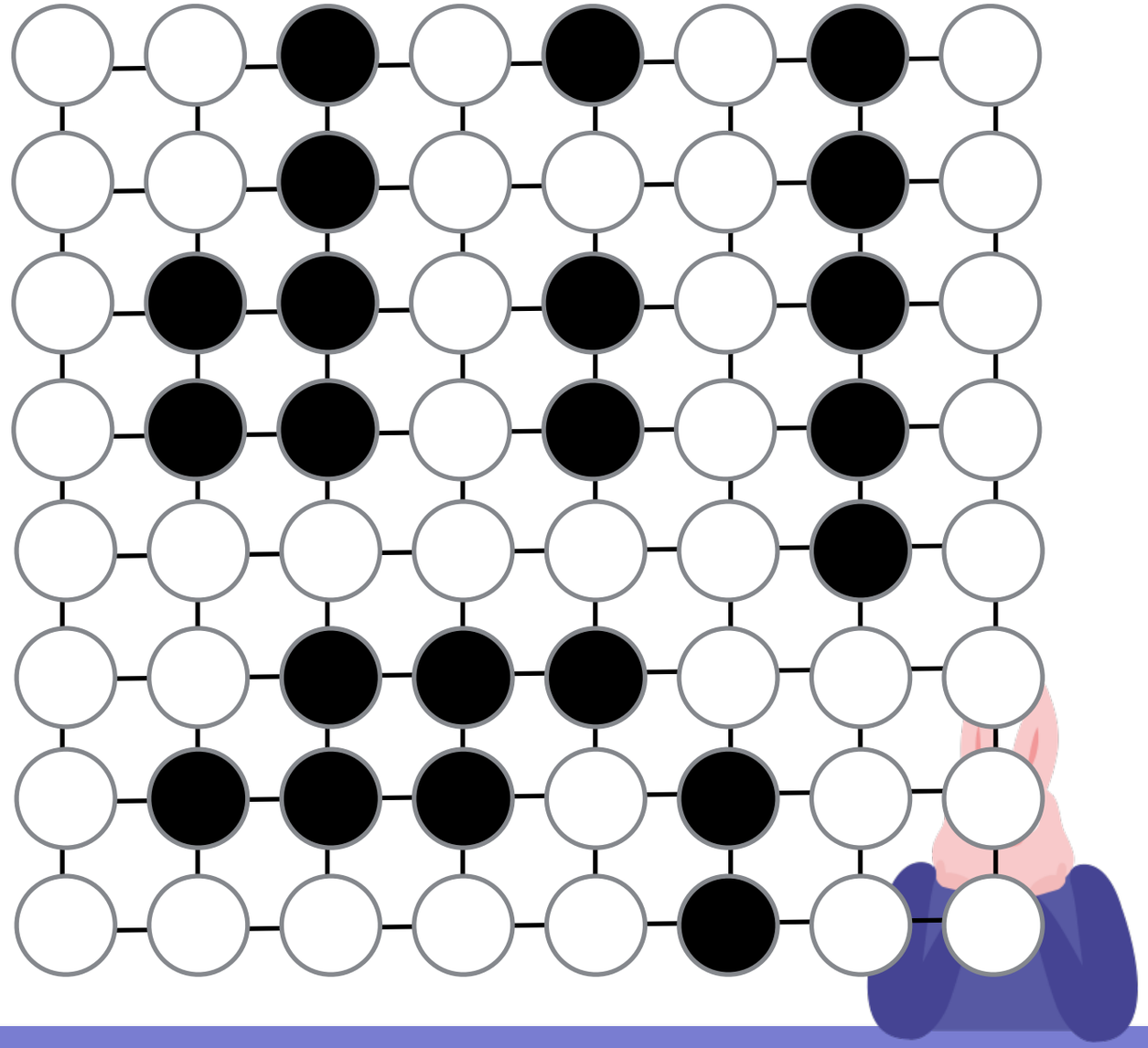




BFS의 응용 : Flood Fill

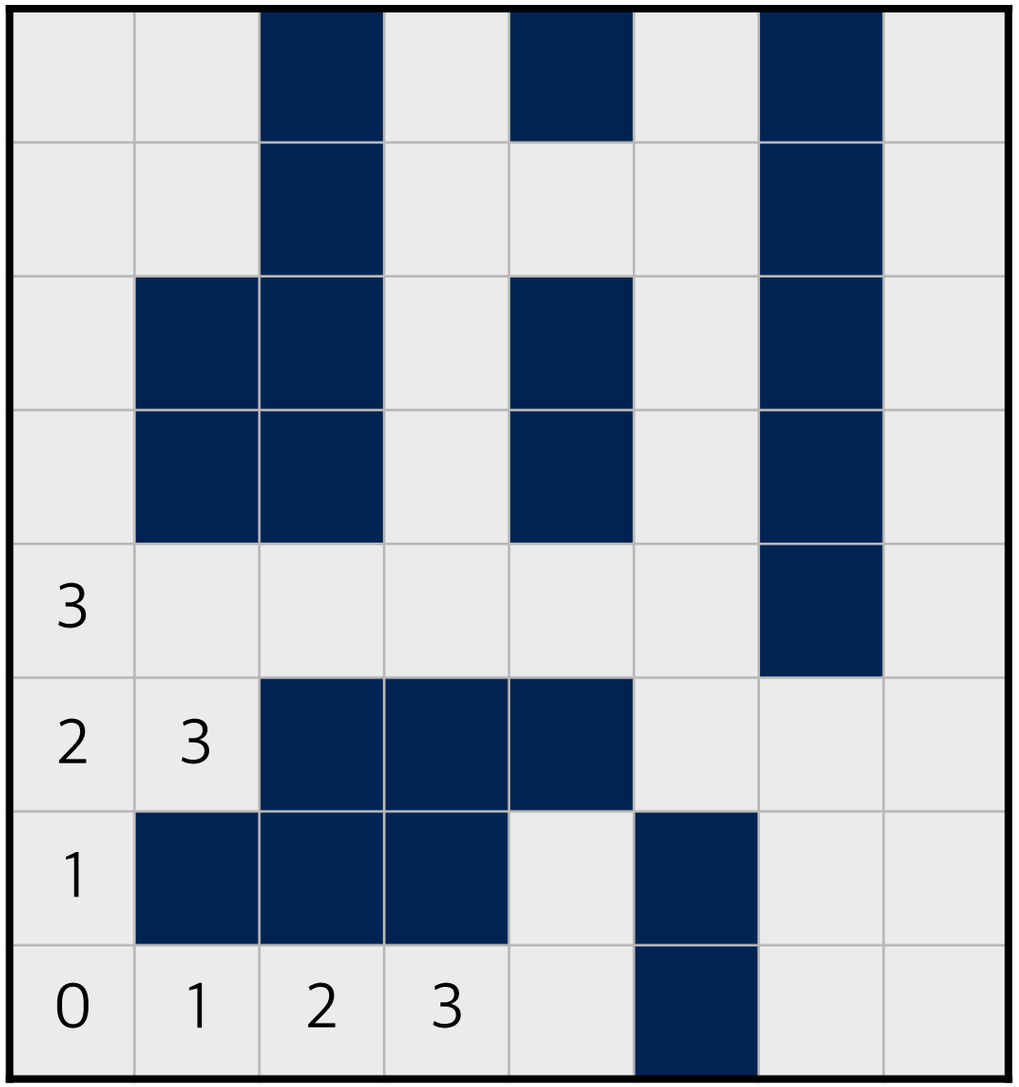
- 물이 차오르는 듯 하여 Flood fill 이라 부름

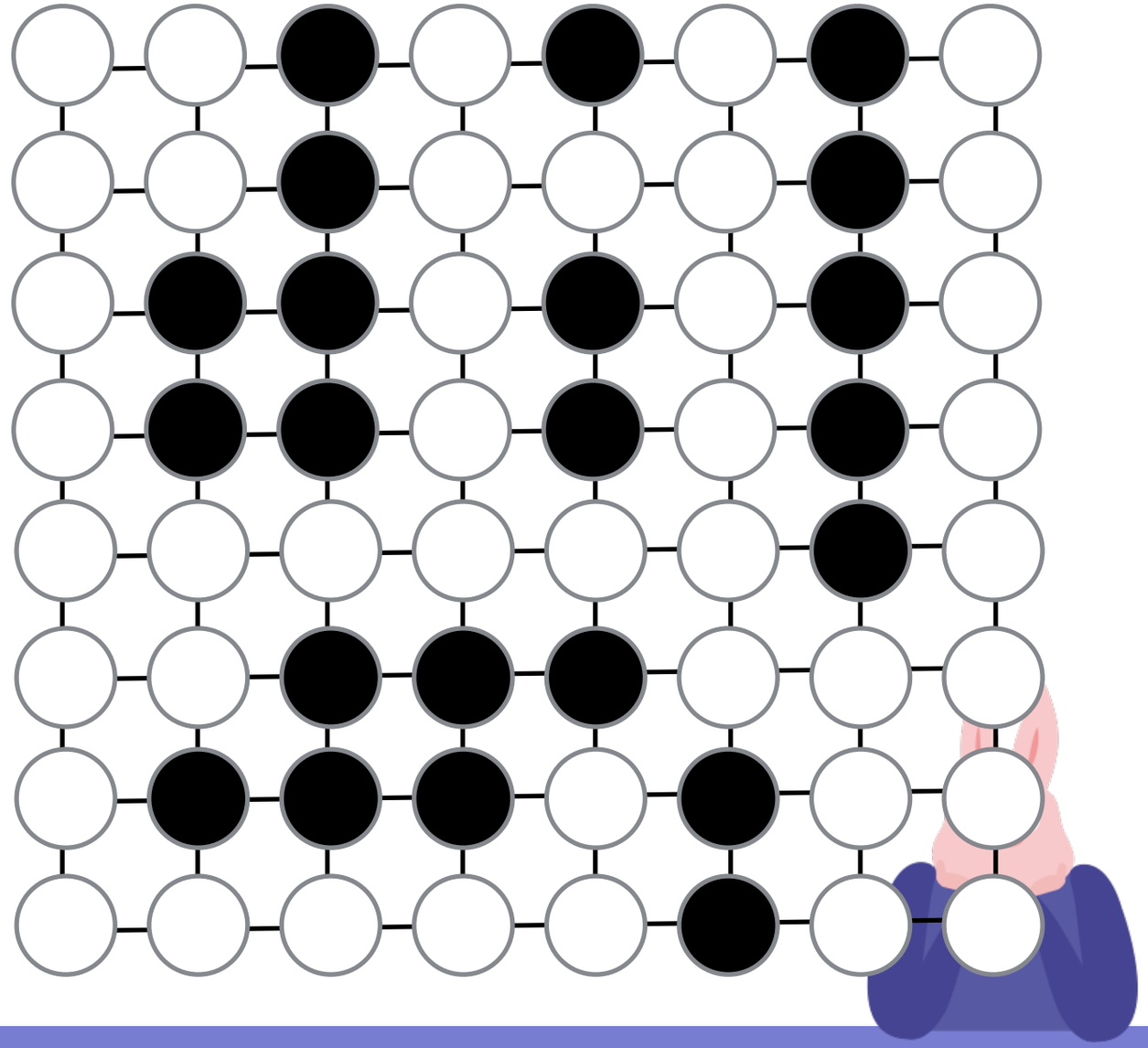




BFS의 응용 : Flood Fill

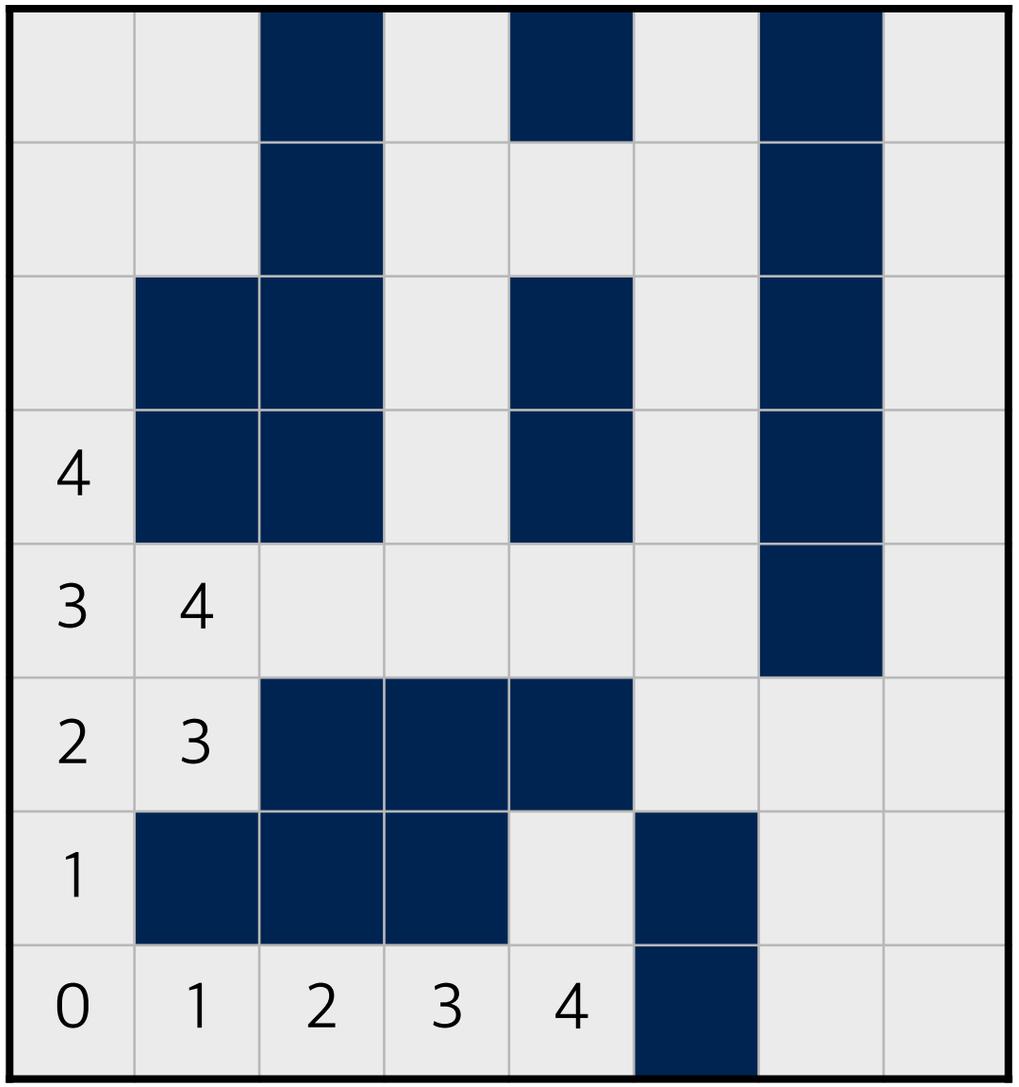
- 물이 차오르는 듯 하여 Flood fill 이라 부름

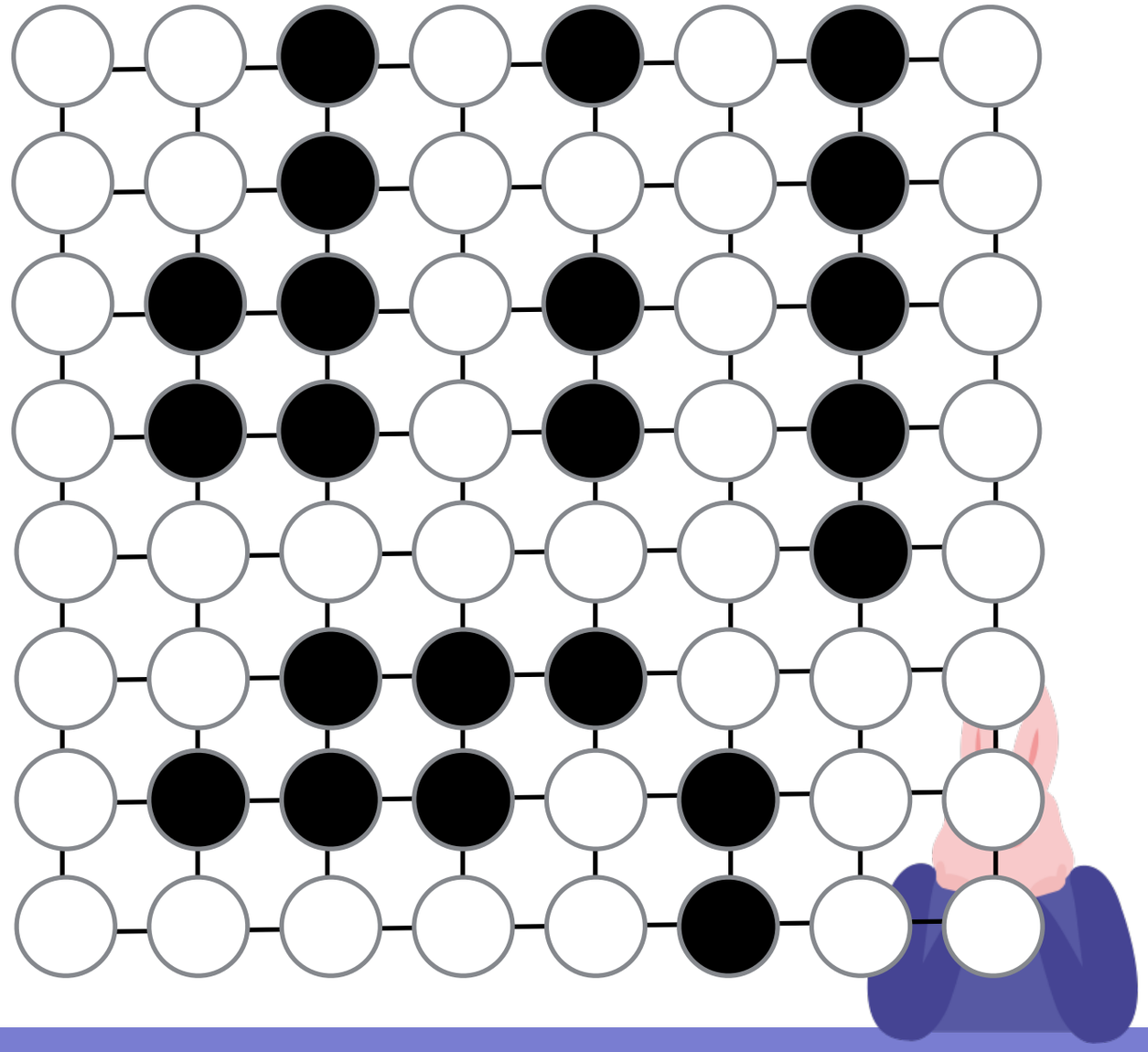




BFS의 응용 : Flood Fill

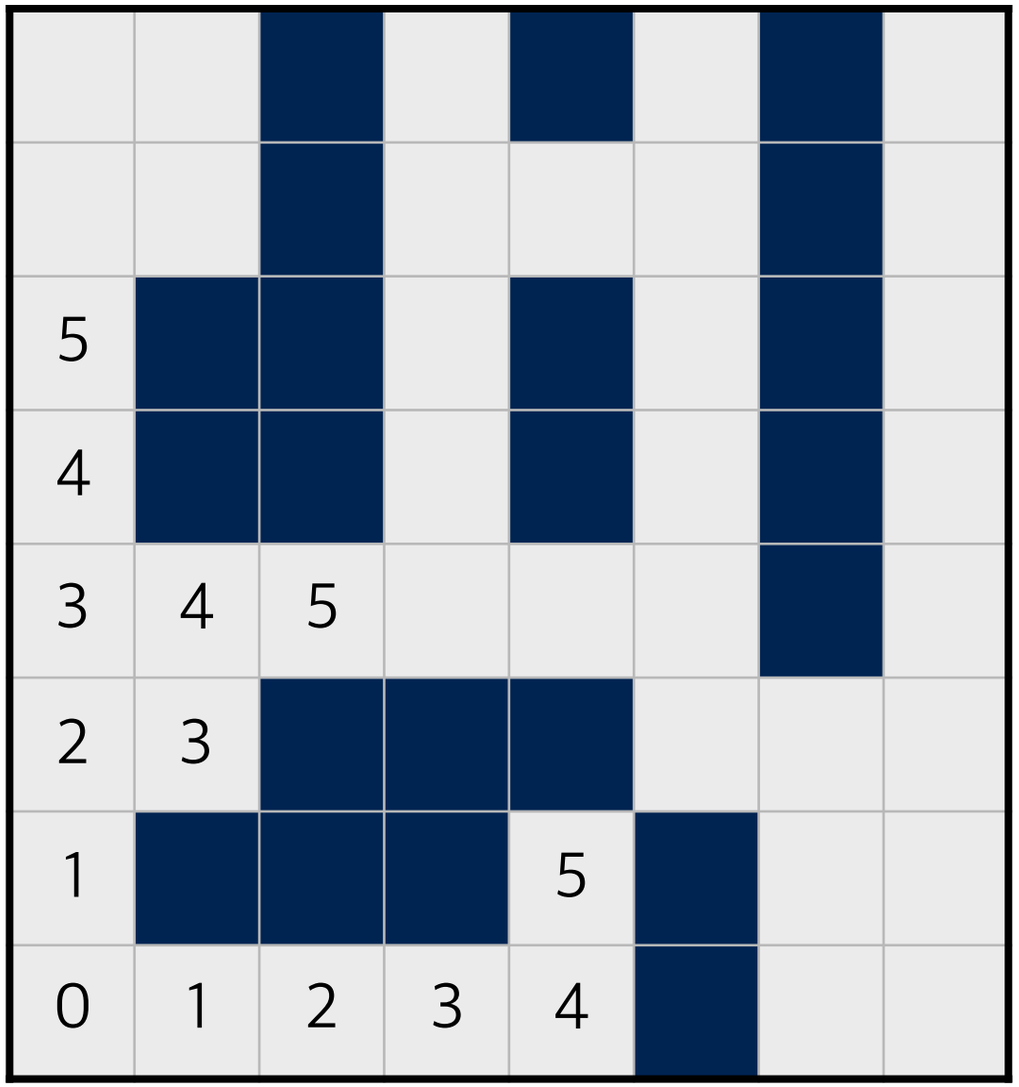
- 물이 차오르는 듯 하여 Flood fill 이라 부름

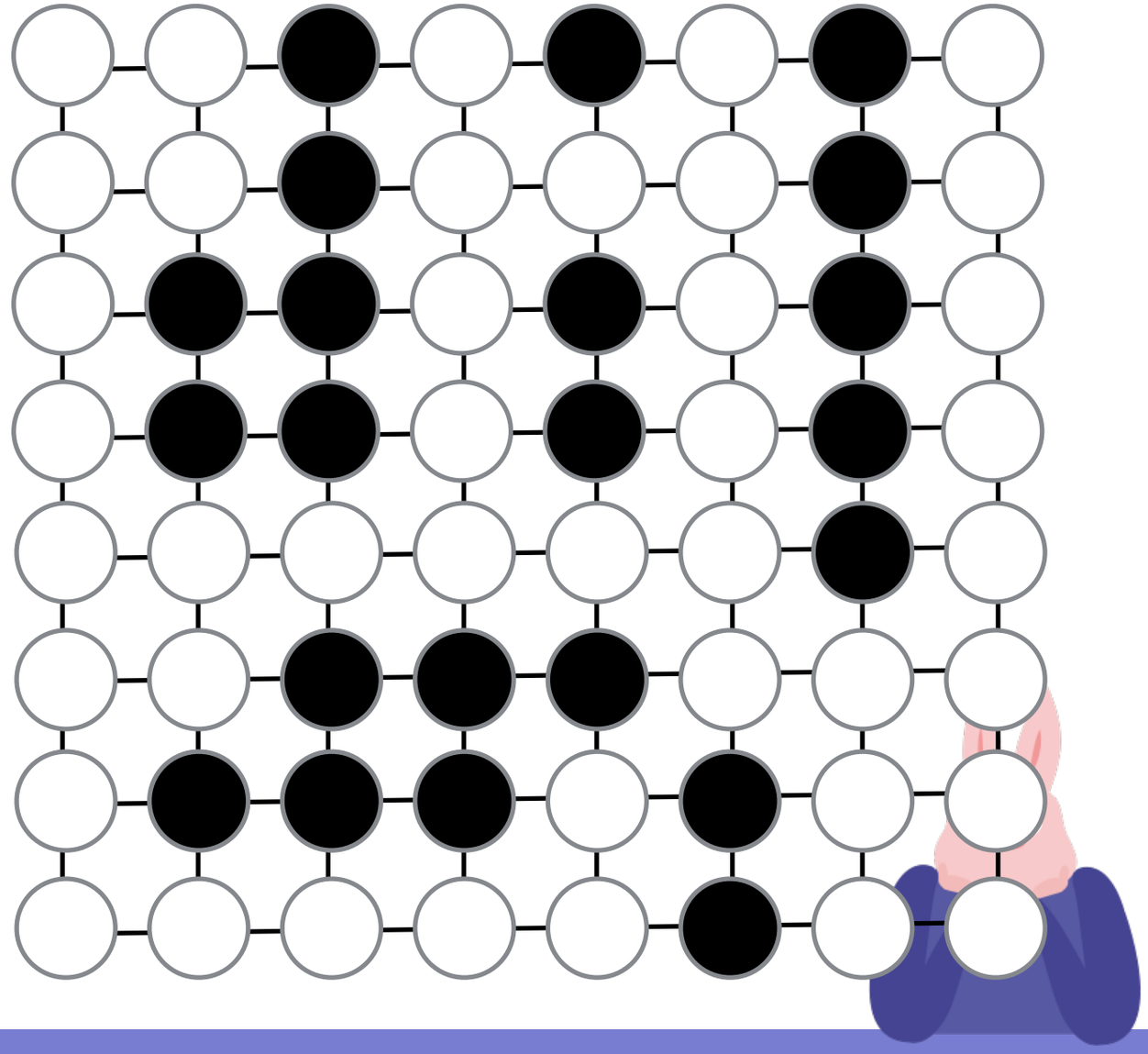




BFS의 응용 : Flood Fill

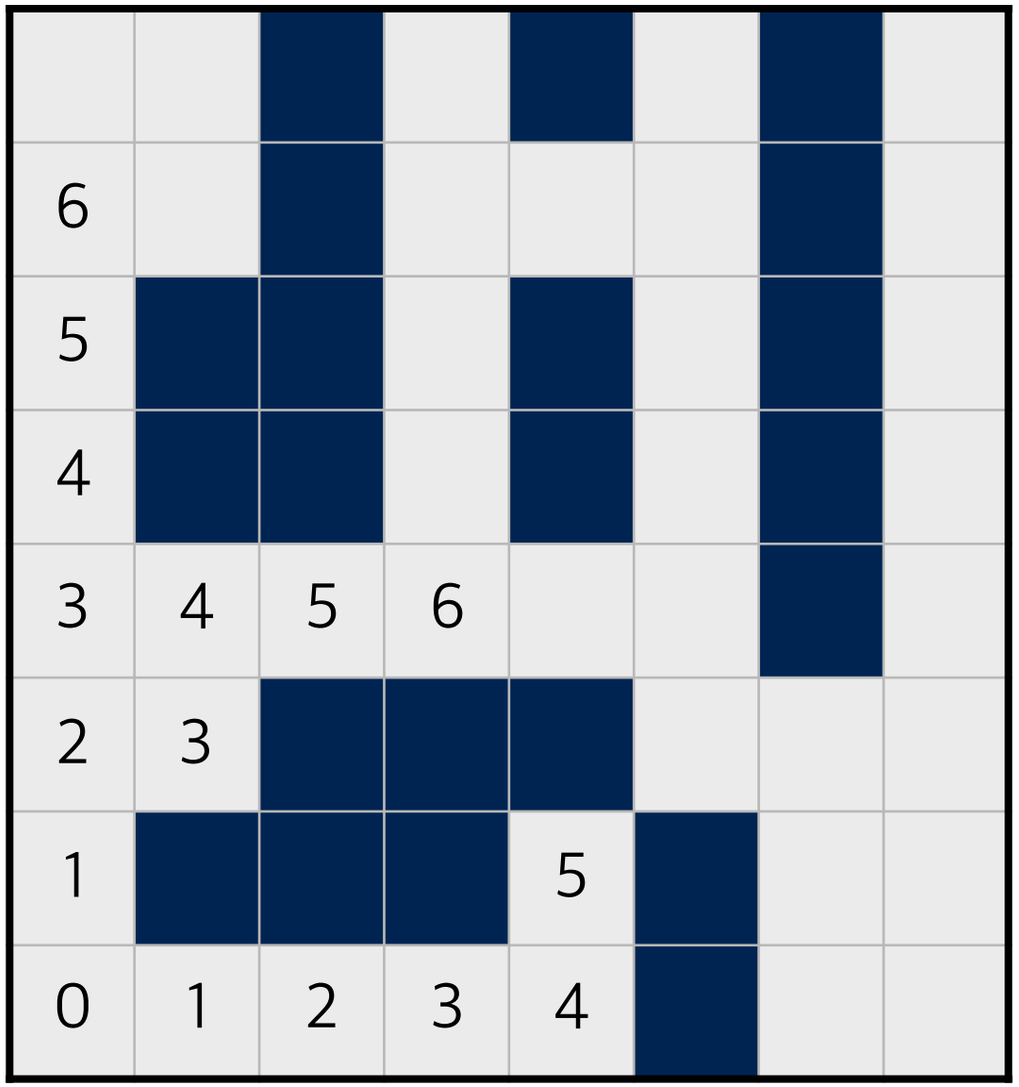
- 물이 차오르는 듯 하여 Flood fill 이라 부름

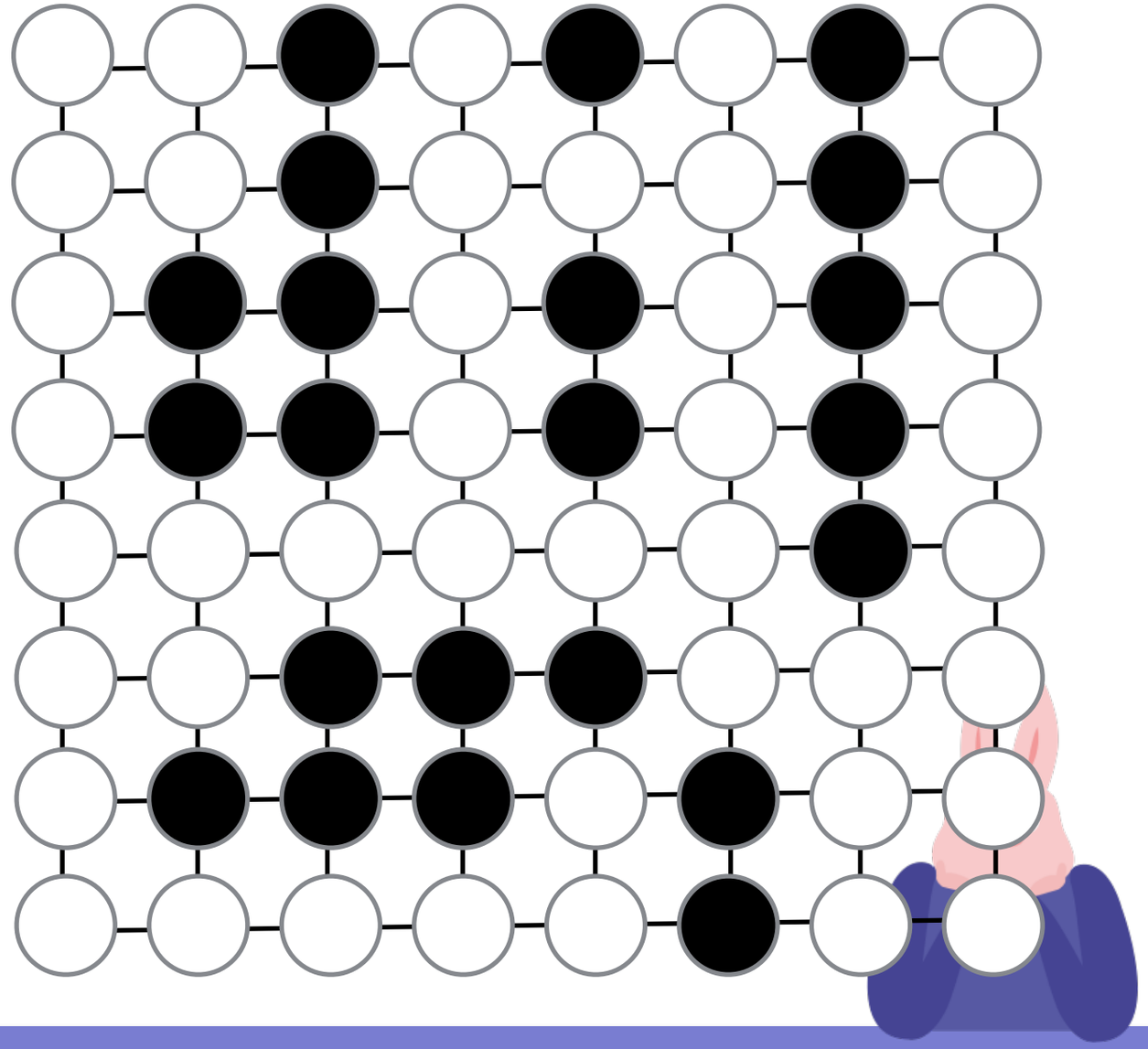




BFS의 응용 : Flood Fill

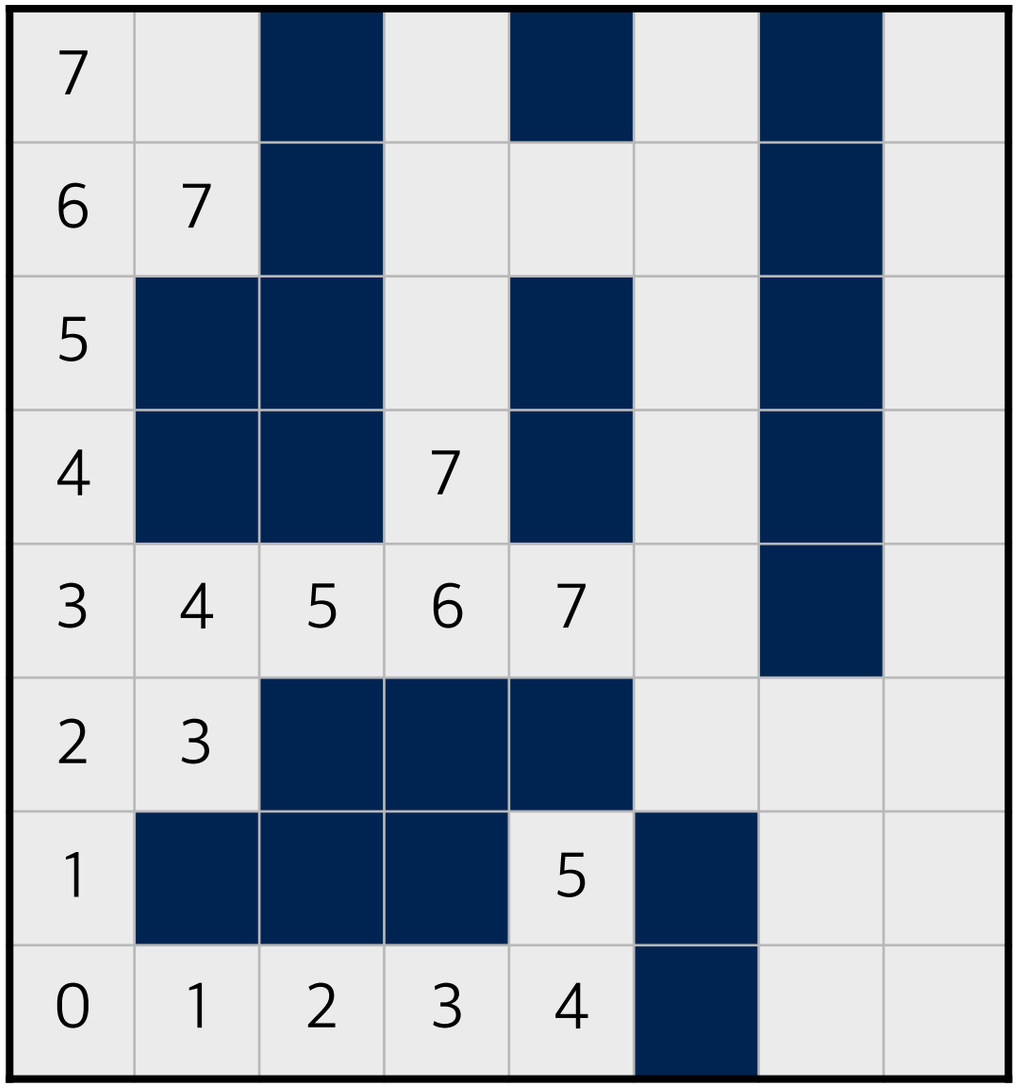
- 물이 차오르는 듯 하여 Flood fill 이라 부름

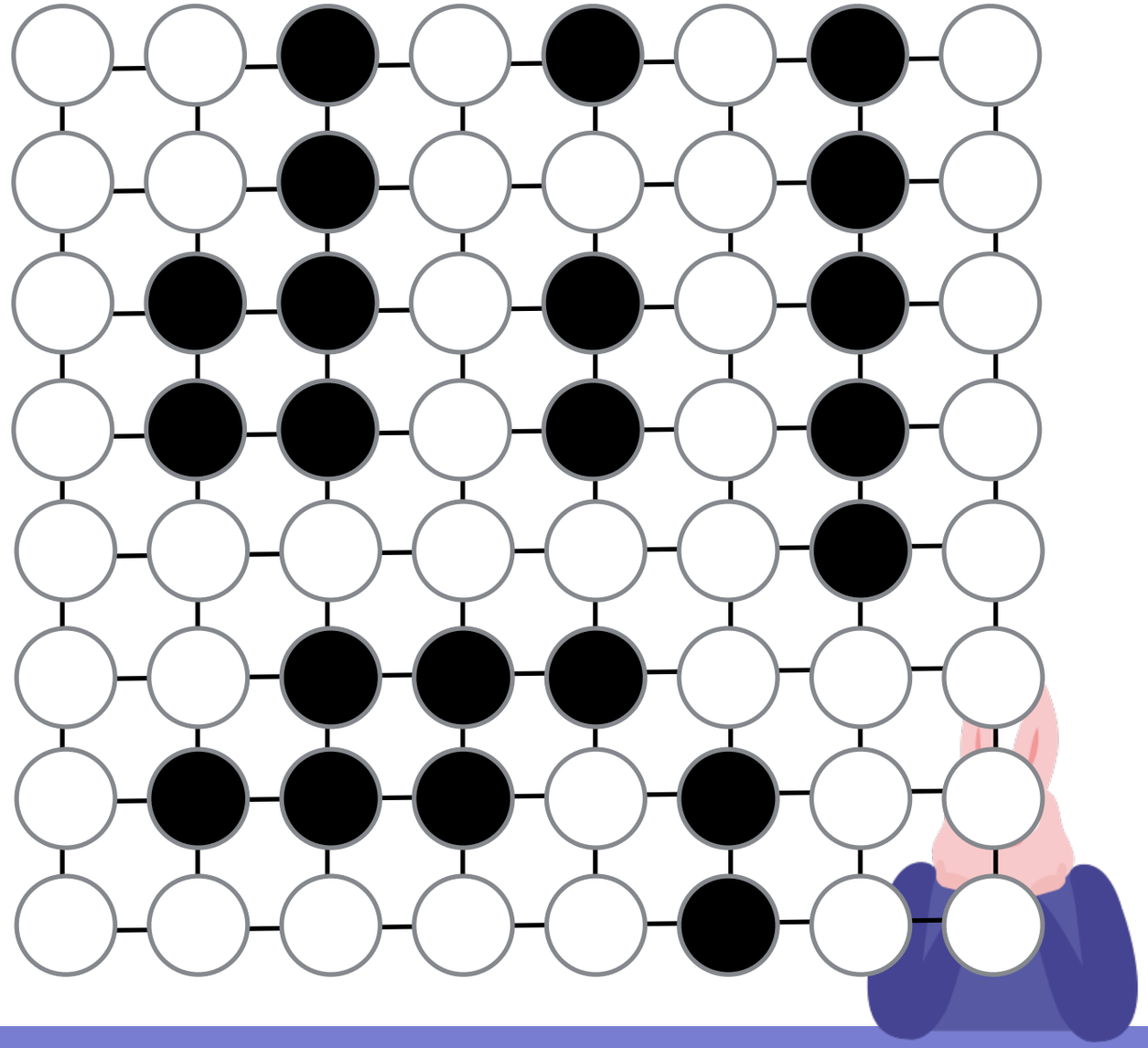




BFS의 응용 : Flood Fill

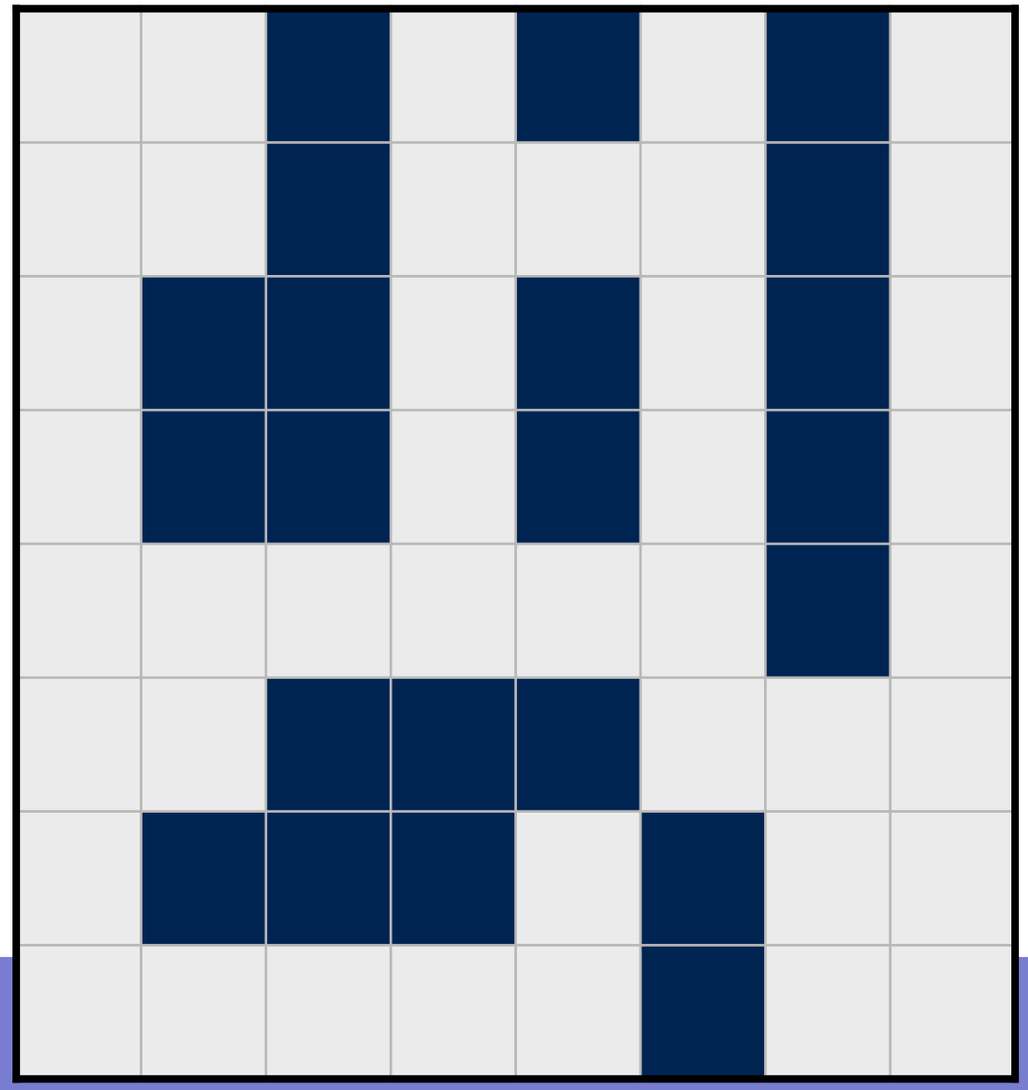
- 물이 차오르는 듯 하여 Flood fill 이라 부름

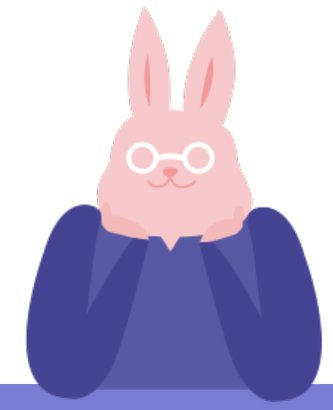




BFS의 응용 : Flood Fill

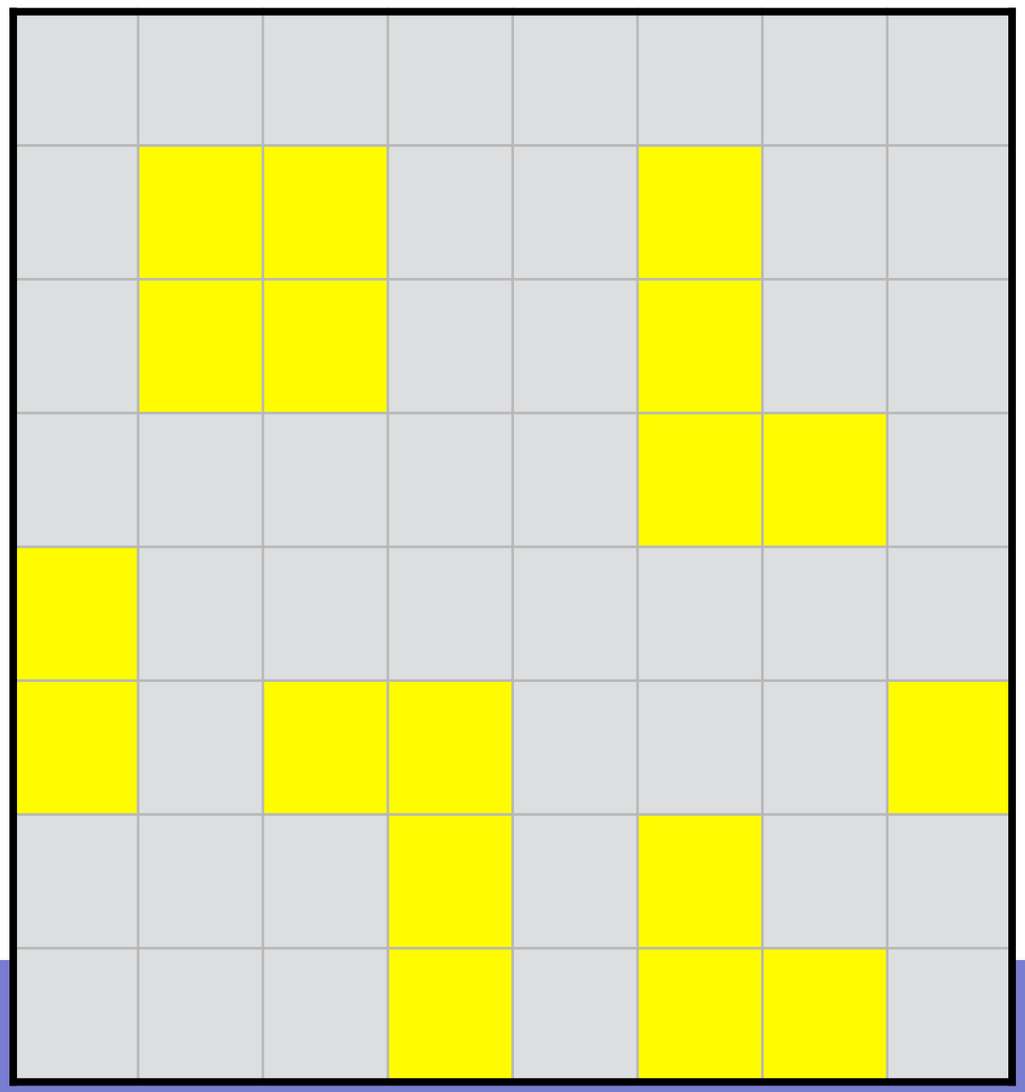
- 언제 쓰나?
  - 인접한 블럭의 집합에 색칠하기

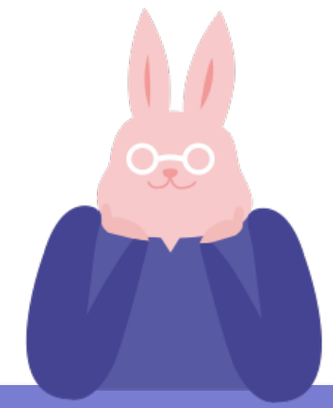




BFS의 응용 : Flood Fill

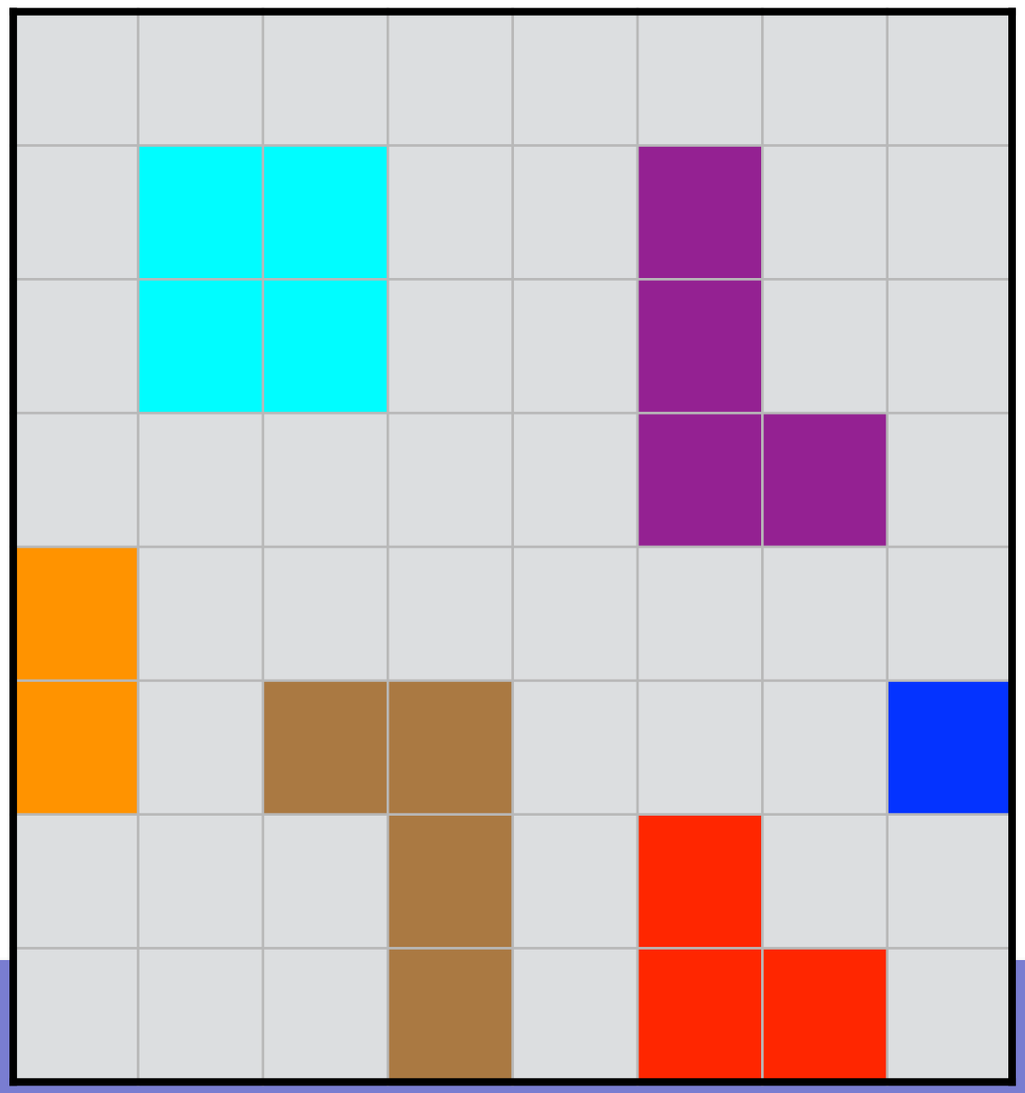
- 언제 쓰나?
  - 아래의 그림에 몇개의 서로 다른 덩어리가 있는가 ?

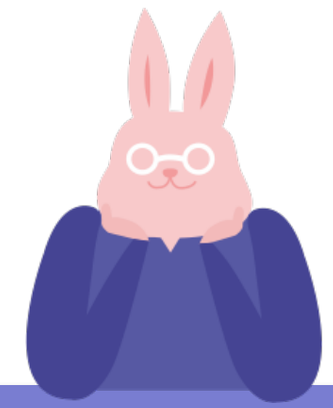




BFS의 응용 : Flood Fill

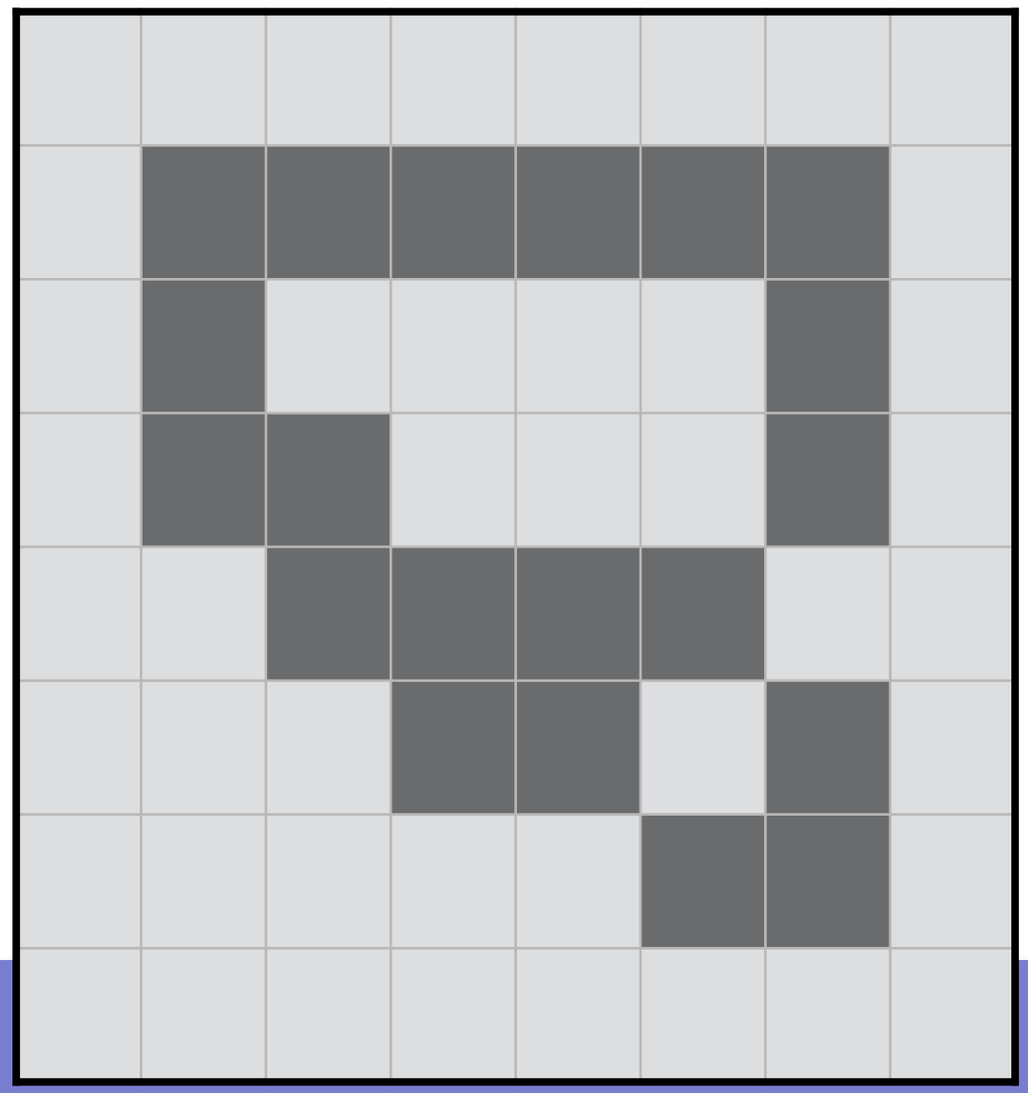
- 언제 쓰나?
  - 아래의 그림에 몇개의 서로 다른 덩어리가 있는가 ? **6개!**

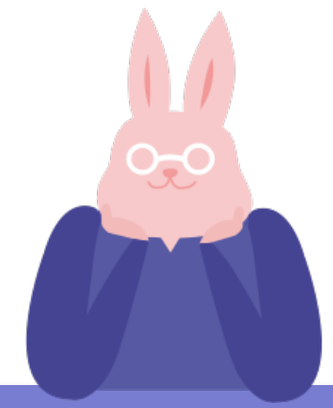




BFS의 응용 : Flood Fill

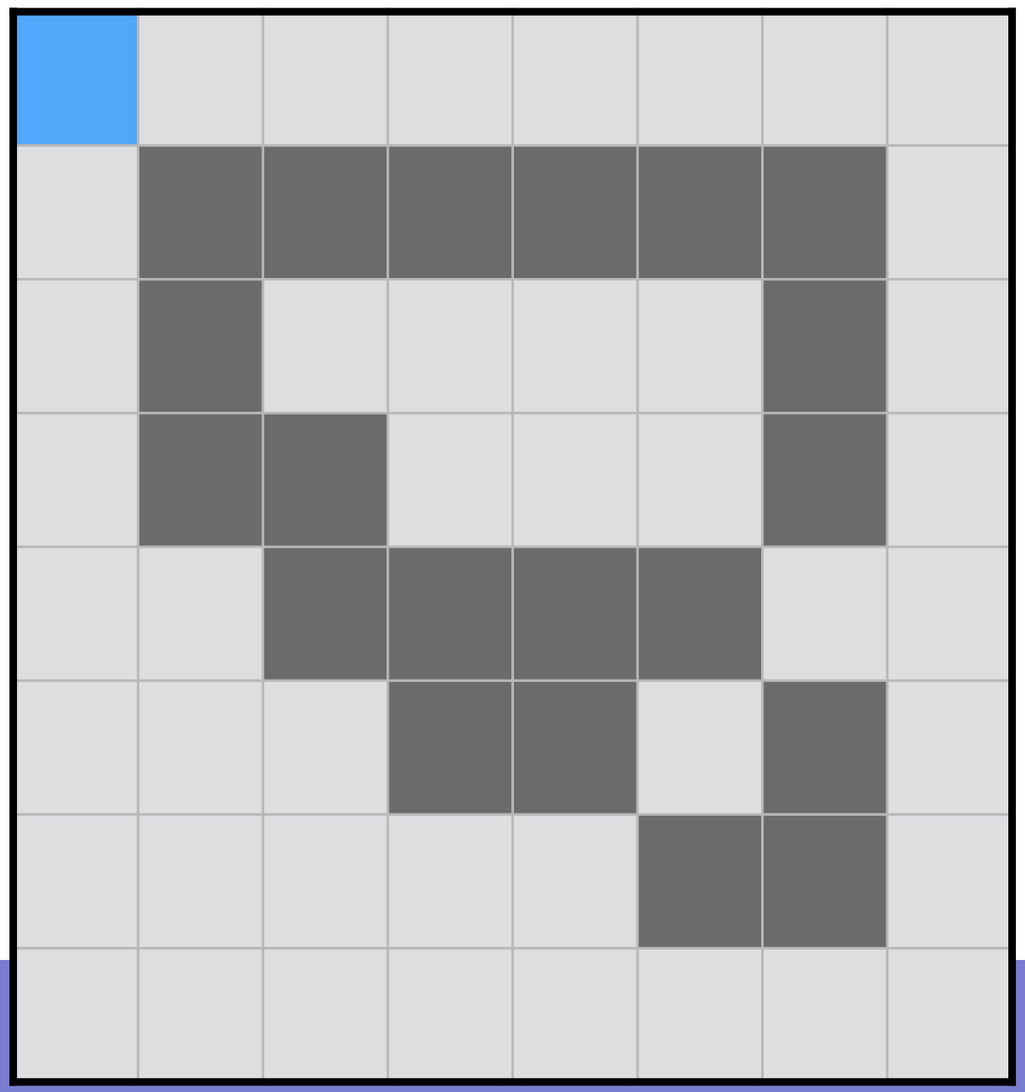
- 언제 쓰나?
  - 아래의 그림에서 외부공기와 내부공기를 판별하라

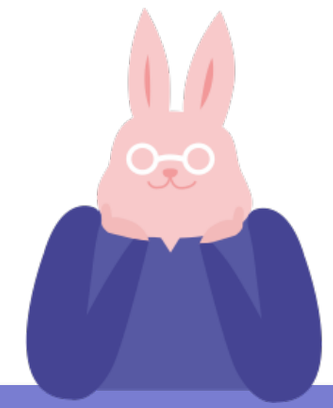




BFS의 응용 : Flood Fill

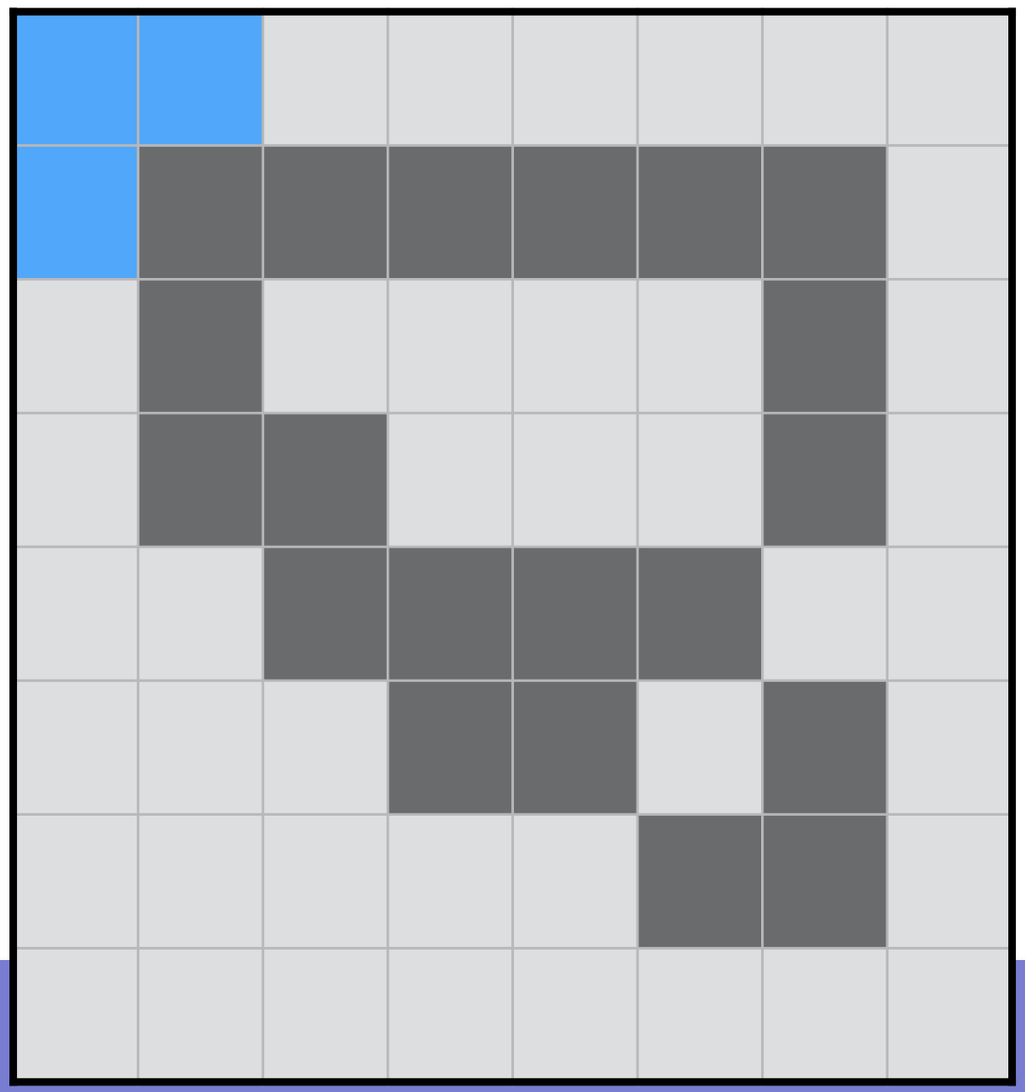
- 언제 쓰나?
  - 아래의 그림에서 외부공기와 내부공기를 판별하라

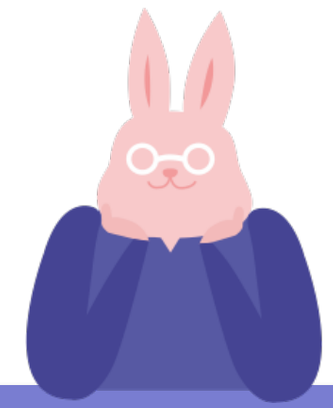




BFS의 응용 : Flood Fill

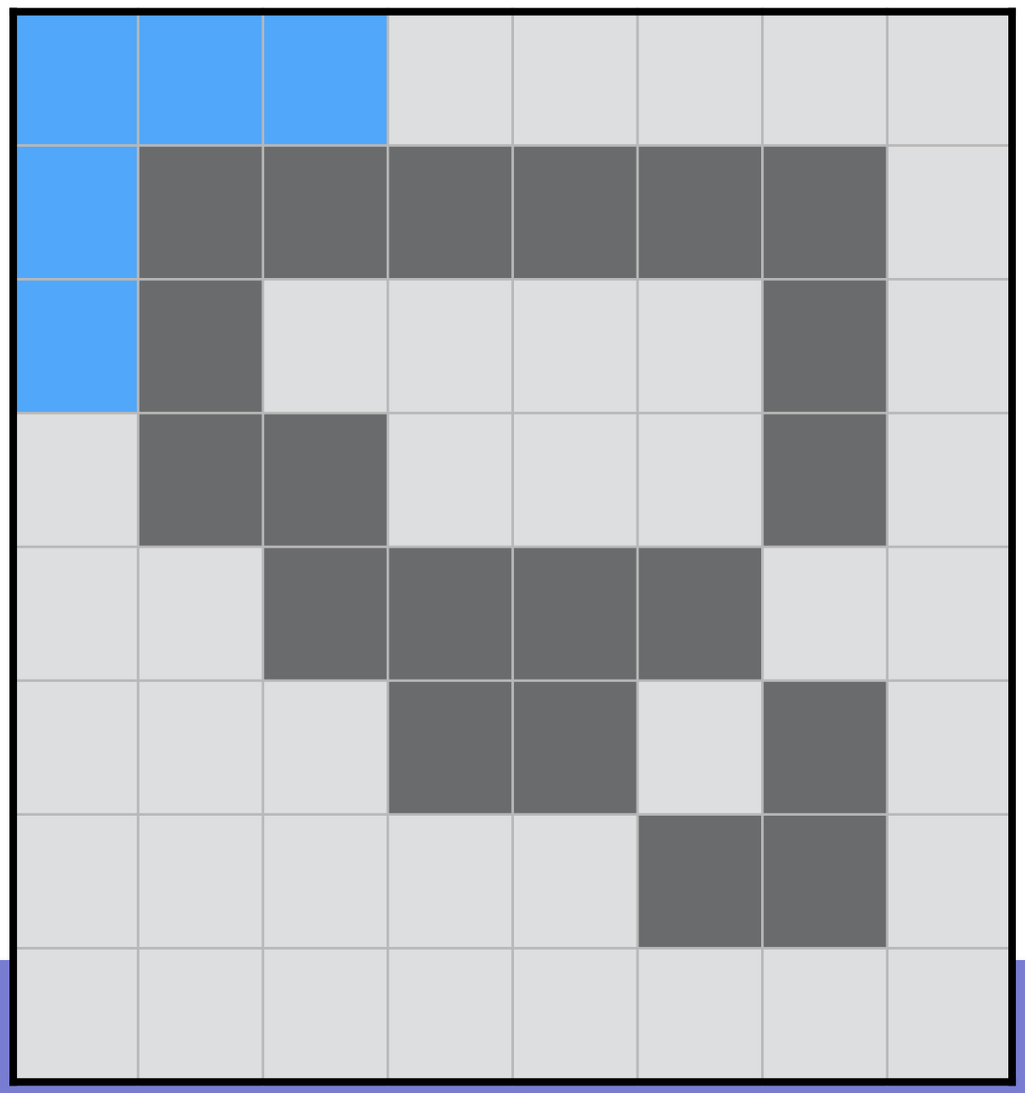
- 언제 쓰나?
  - 아래의 그림에서 외부공기와 내부공기를 판별하라

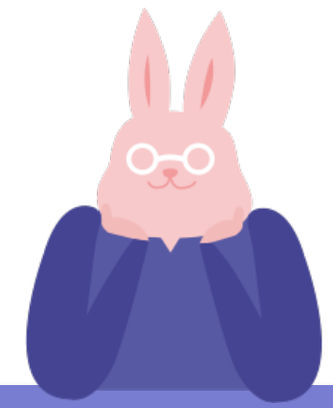




BFS의 응용 : Flood Fill

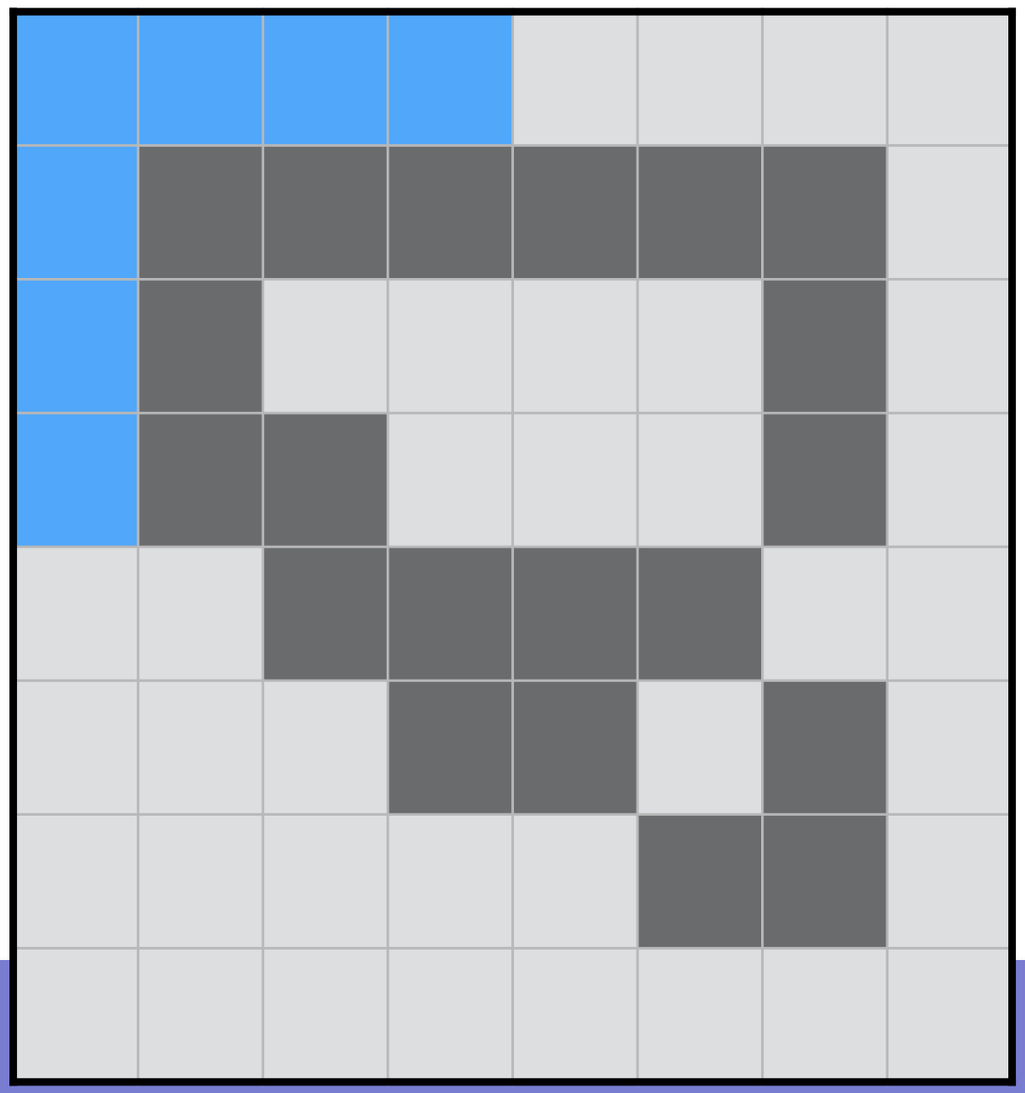
- 언제 쓰나?
  - 아래의 그림에서 외부공기와 내부공기를 판별하라

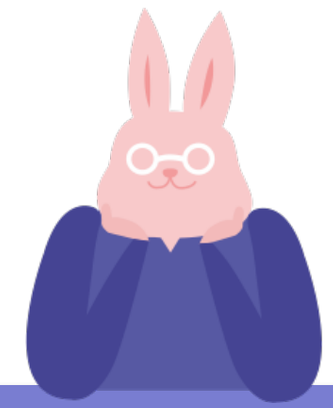




BFS의 응용 : Flood Fill

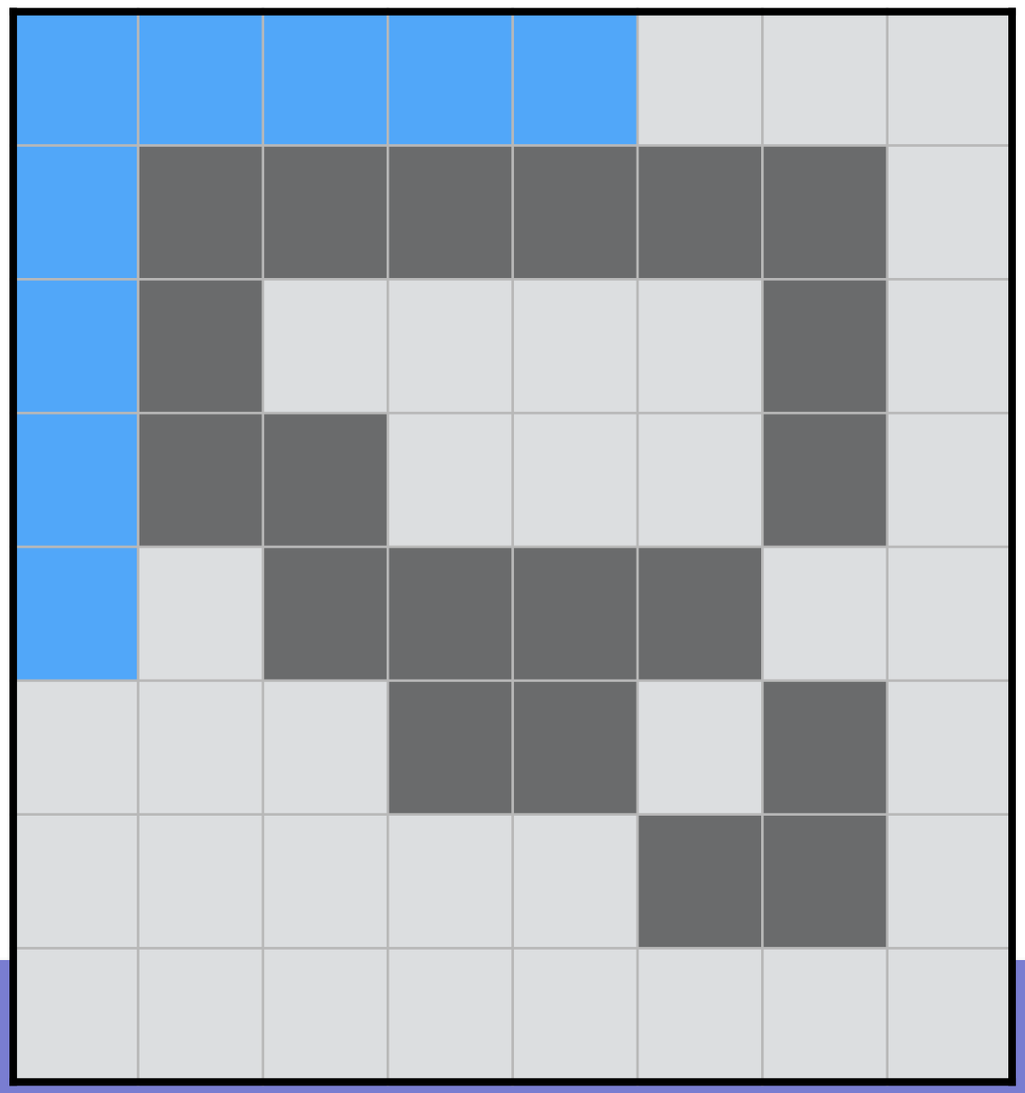
- 언제 쓰나?
  - 아래의 그림에서 외부공기와 내부공기를 판별하라

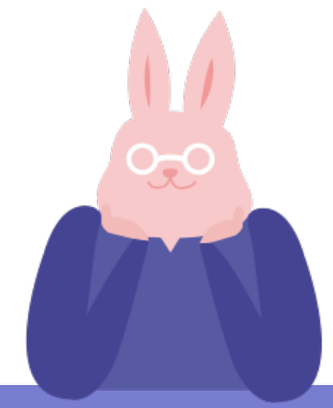




BFS의 응용 : Flood Fill

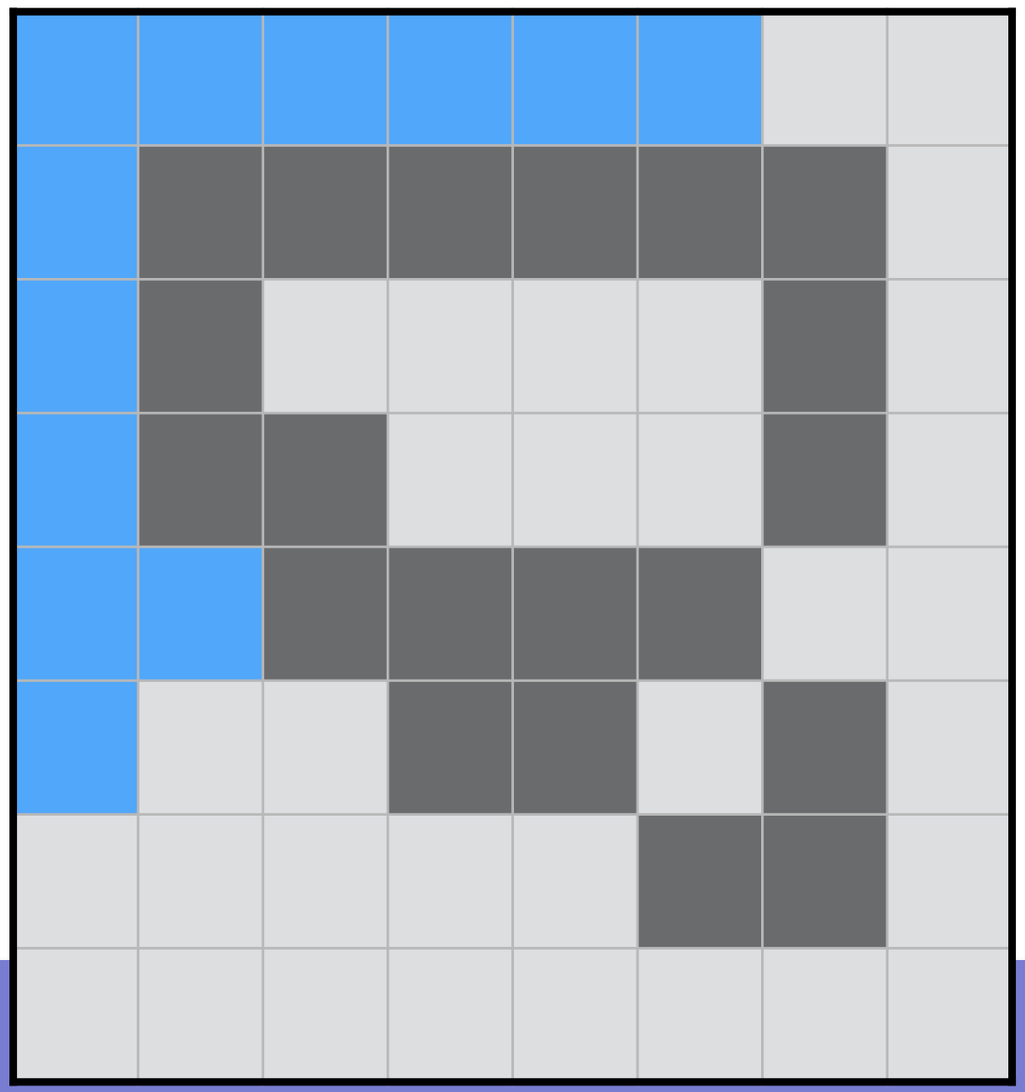
- 언제 쓰나?
  - 아래의 그림에서 외부공기와 내부공기를 판별하라

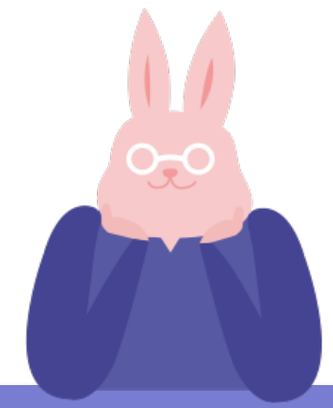




BFS의 응용 : Flood Fill

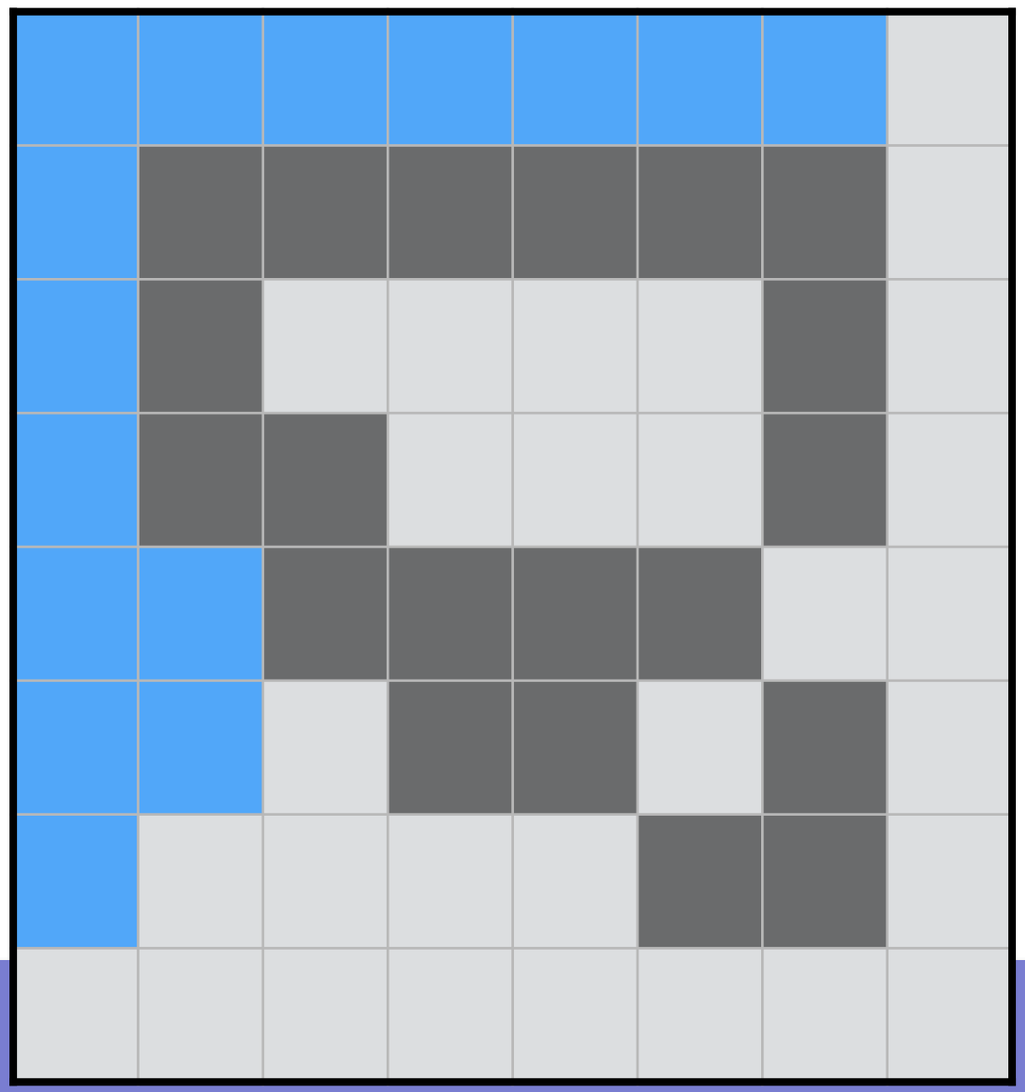
- 언제 쓰나?
  - 아래의 그림에서 외부공기와 내부공기를 판별하라

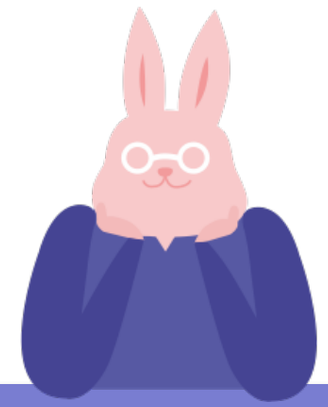




BFS의 응용 : Flood Fill

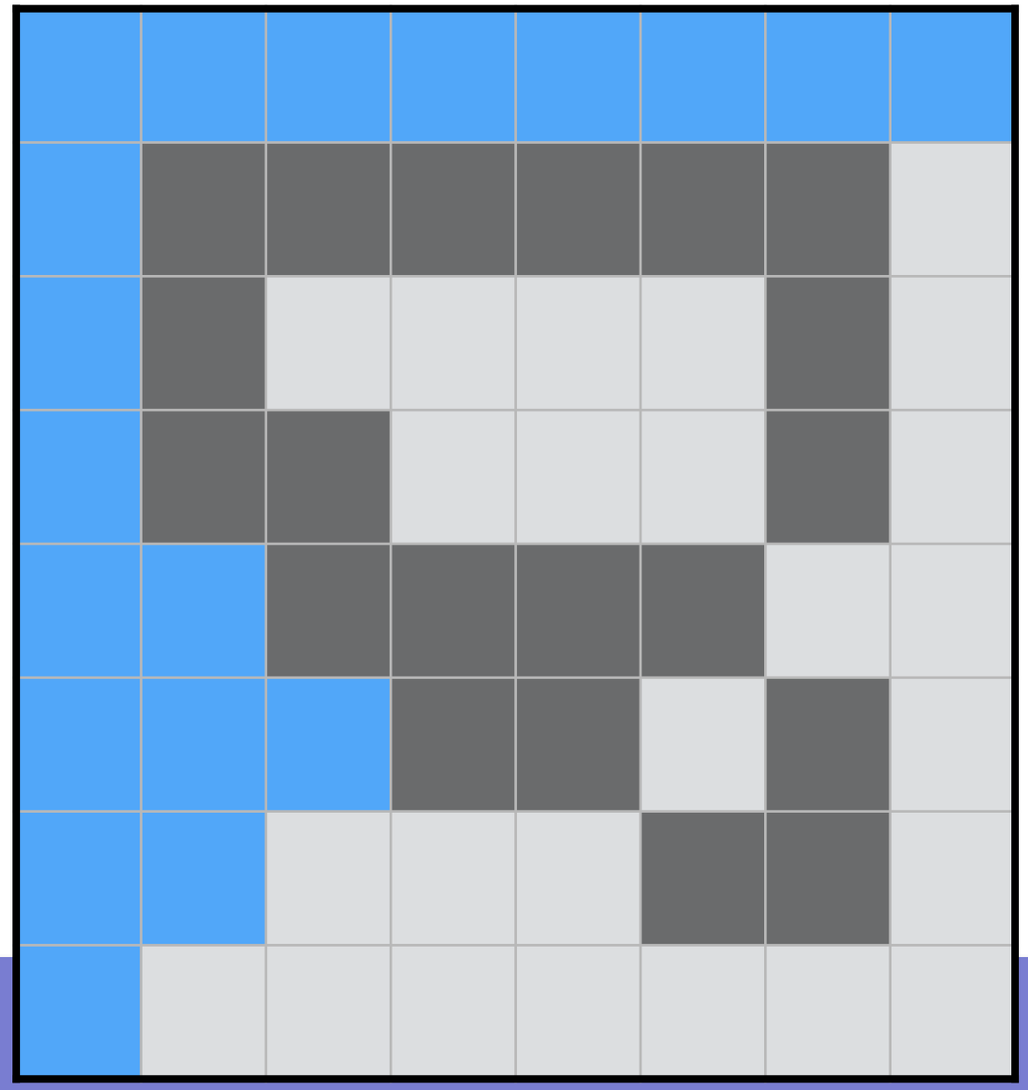
- 언제 쓰나?
  - 아래의 그림에서 외부공기와 내부공기를 판별하라

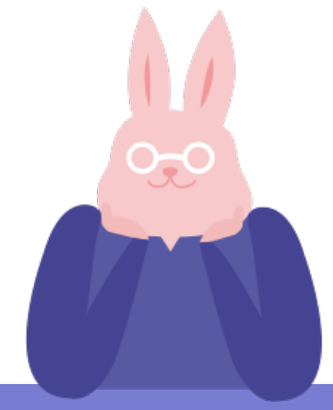




BFS의 응용 : Flood Fill

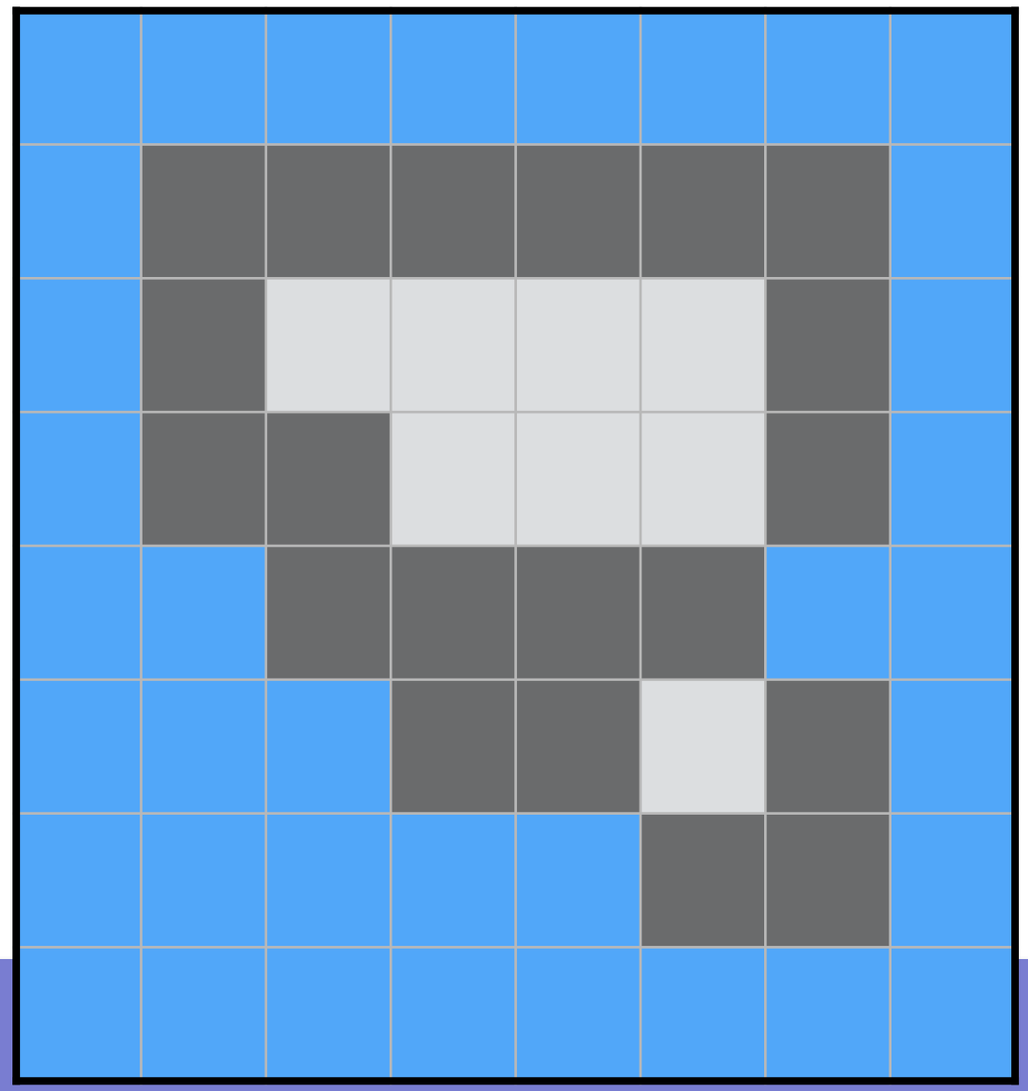
- 언제 쓰나?
  - 아래의 그림에서 외부공기와 내부공기를 판별하라

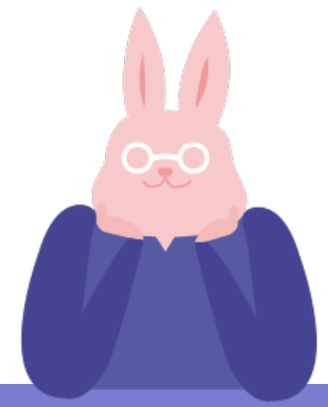




BFS의 응용 : Flood Fill

- 언제 쓰나?
  - 아래의 그림에서 외부공기와 내부공기를 판별하라





[활동문제 2] 미로찾기

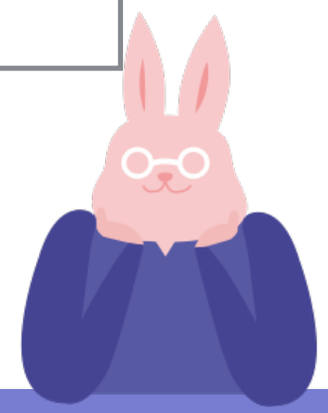
- 시작점에서 출발하여 끝점에 도달하기 위한 최단거리를 출력
  (시작점은 항상 왼쪽 아래, 끝점은 항상 오른쪽 위)

입력의 예

```
5 6
0 1 0 1 1 0
0 1 0 0 1 0
0 0 0 0 1 0
0 1 1 0 0 0
0 1 0 0 0 0
```

출력의 예

```
11
```



[예제] 바이러스

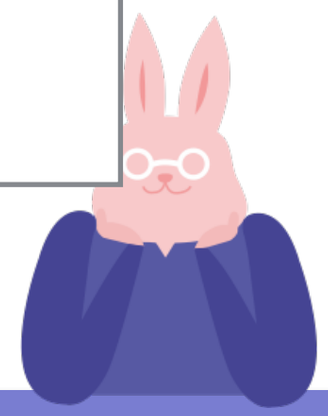
- k마리의 바이러스가 1초 후에는 k*2 마리, 혹은 k/3마리가 됨
  N마리가 되기 위해 걸리는 최소 시간과 그 경로는 ?
  (처음엔 1마리에서 시작, k는 항상 10000 이하여야 함)

입력의 예

```
10
```

출력의 예

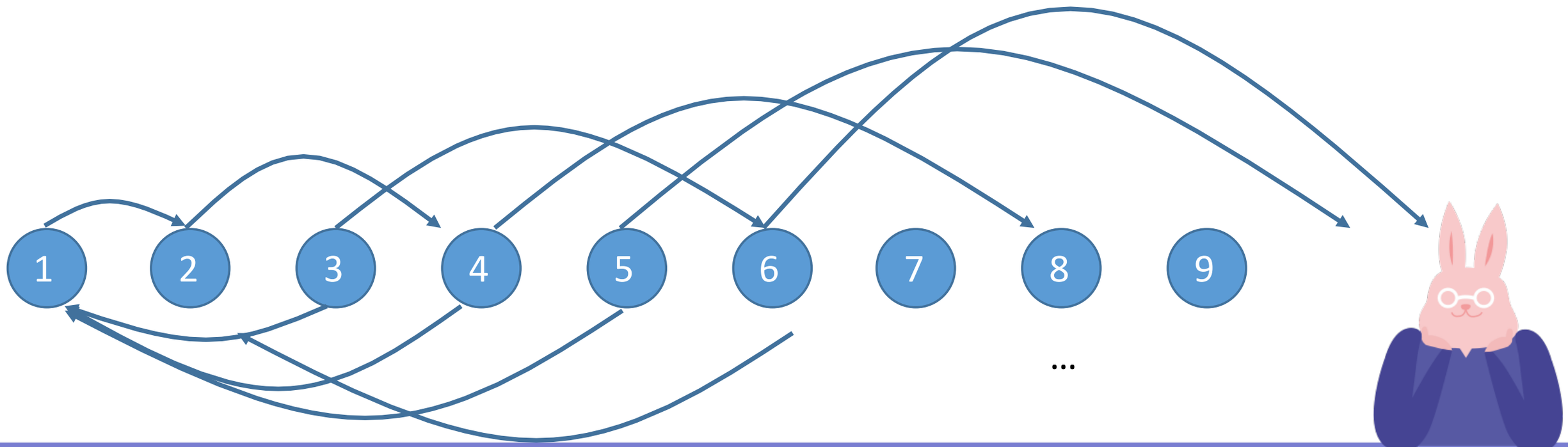
```
6
1 2 4 8 16 5 10
```



[예제] 바이러스

- 각 정점은 **바이러스가 몇 마리** 있는지에 대한 상태
  - 정점 X : 현재 바이러스가 X마리 있는 상태이다

- 각 간선은 바이러스의 마릿수의 변화를 나타냄
  - X —> Y : X 마리에서 Y마리로 변화할 수 있음

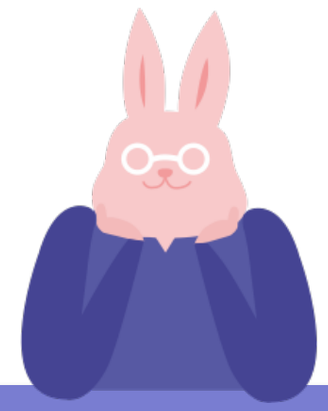


[예제] 바이러스

- 각 정점은 **바이러스가 몇 마리** 있는지에 대한 상태
  - 정점 X : 현재 바이러스가 X마리 있는 상태이다

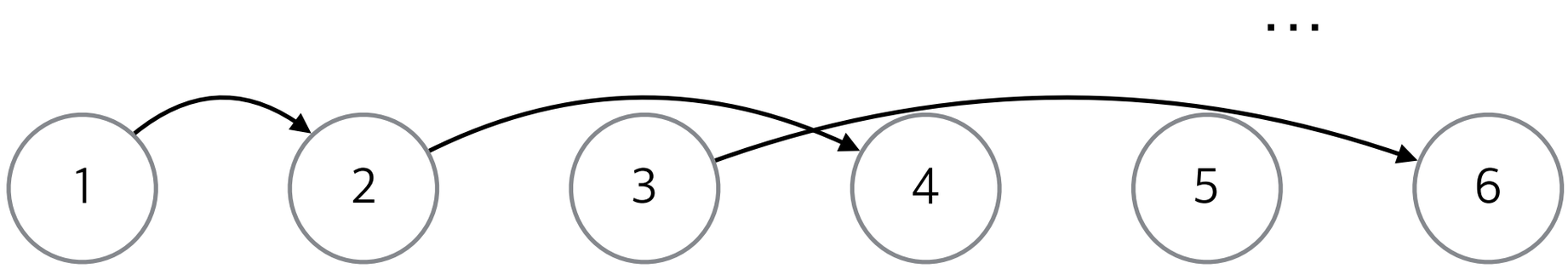
- 각 간선은 바이러스의 마릿수의 변화를 나타냄
  - X —> Y : X 마리에서 Y마리로 변화할 수 있음

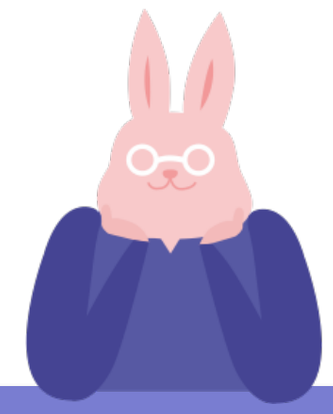
- **정점 1에서 정점 N으로 가는 최단경로는 무엇인가 ?**
  - BFS로 해결



[예제] 바이러스

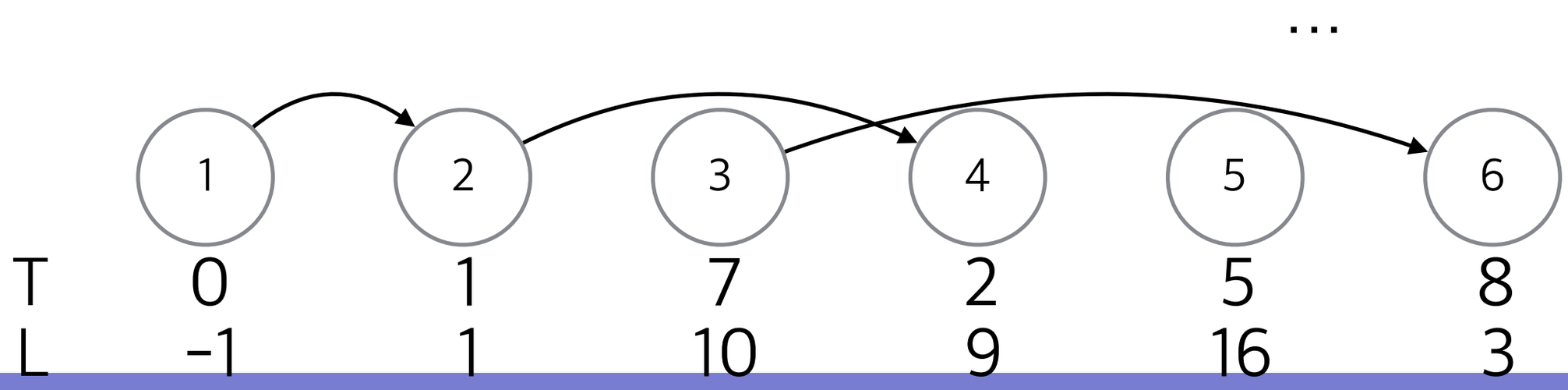
- 거리는 어떻게 구하는가 ?
  - D(x) : Node x 에 도달하기 위한 거리
- 경로는 어떻게 구하는가 ?
  - L(x) : Node x에 도달하기 직전에 있었던 노드 번호

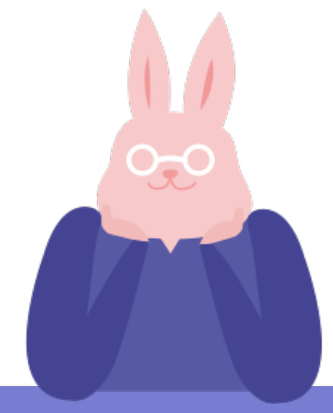




[예제] 바이러스

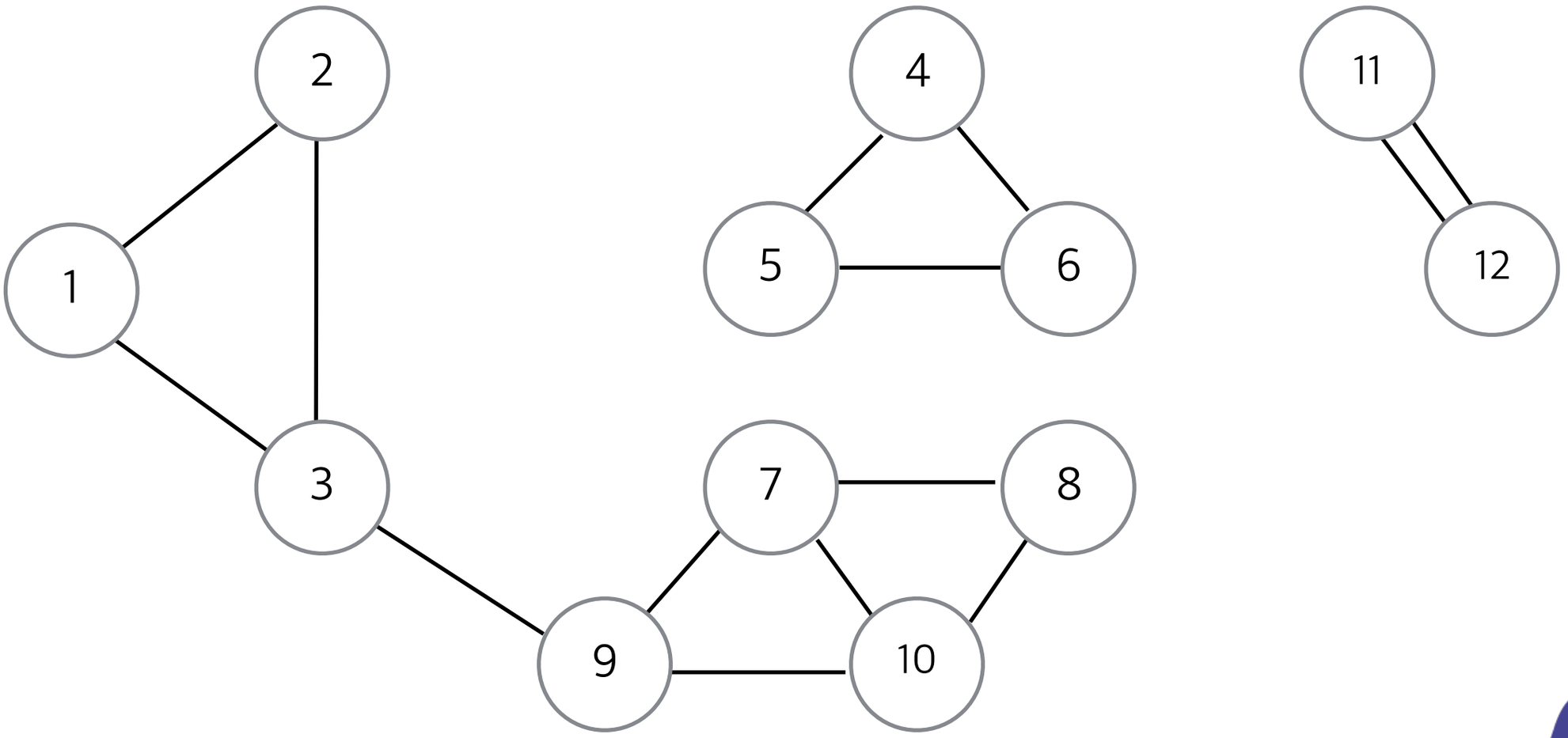
- 거리는 어떻게 구하는가 ?
  - D(x) : Node x 에 도달하기 위한 거리
- 경로는 어떻게 구하는가 ?
  - L(x) : Node x에 도달하기 직전에 있었던 노드 번호

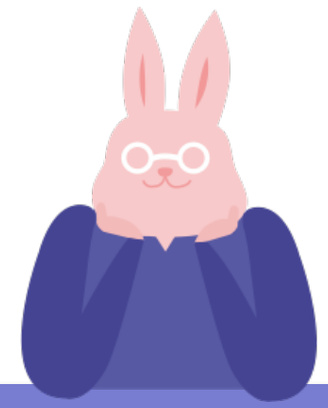




Connected Component

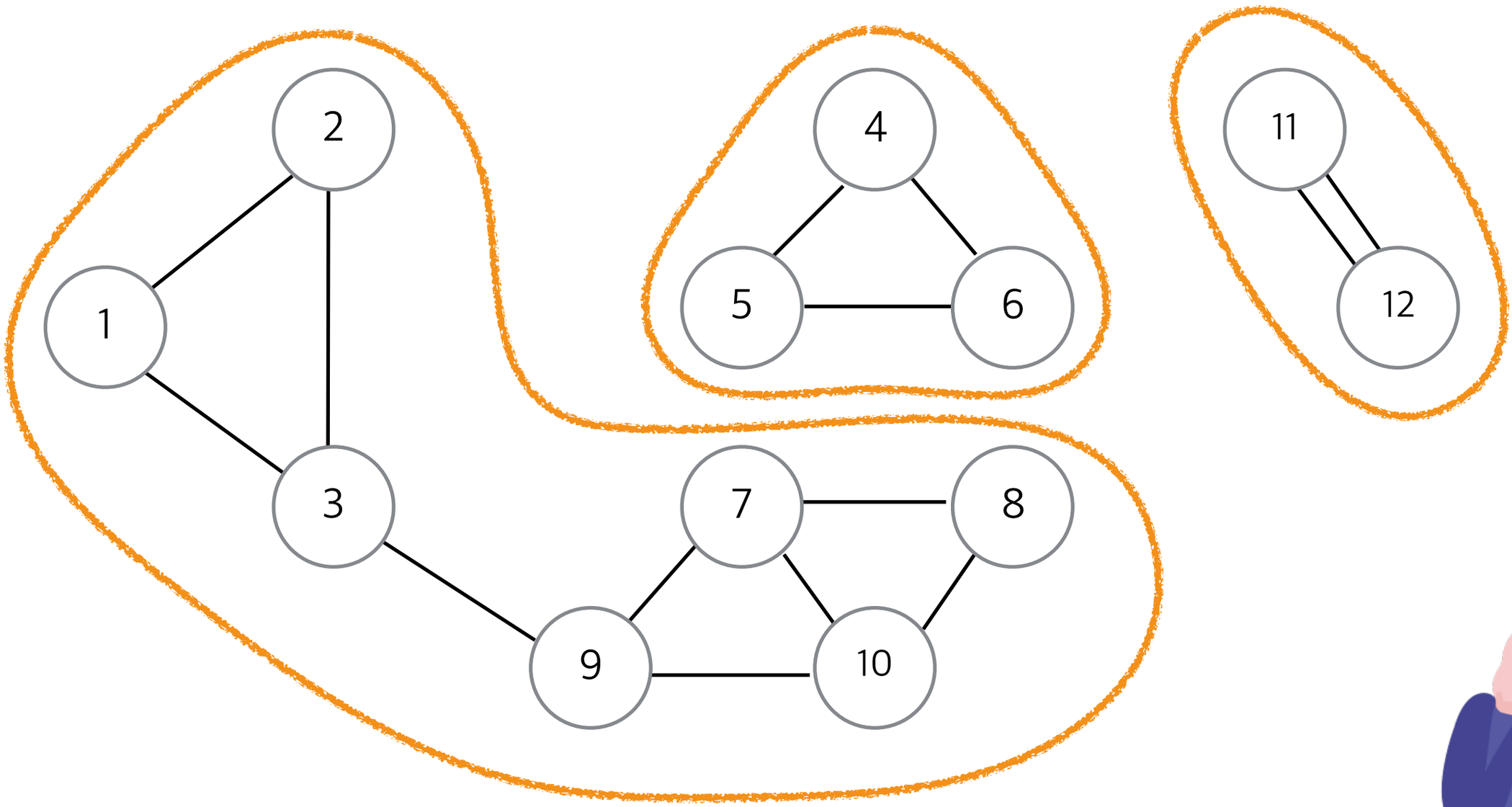
- 연결되어 있는 graph들로 나누어라 !

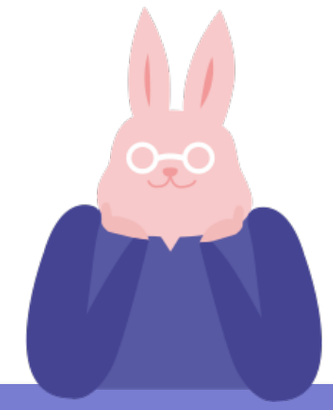




Connected Component

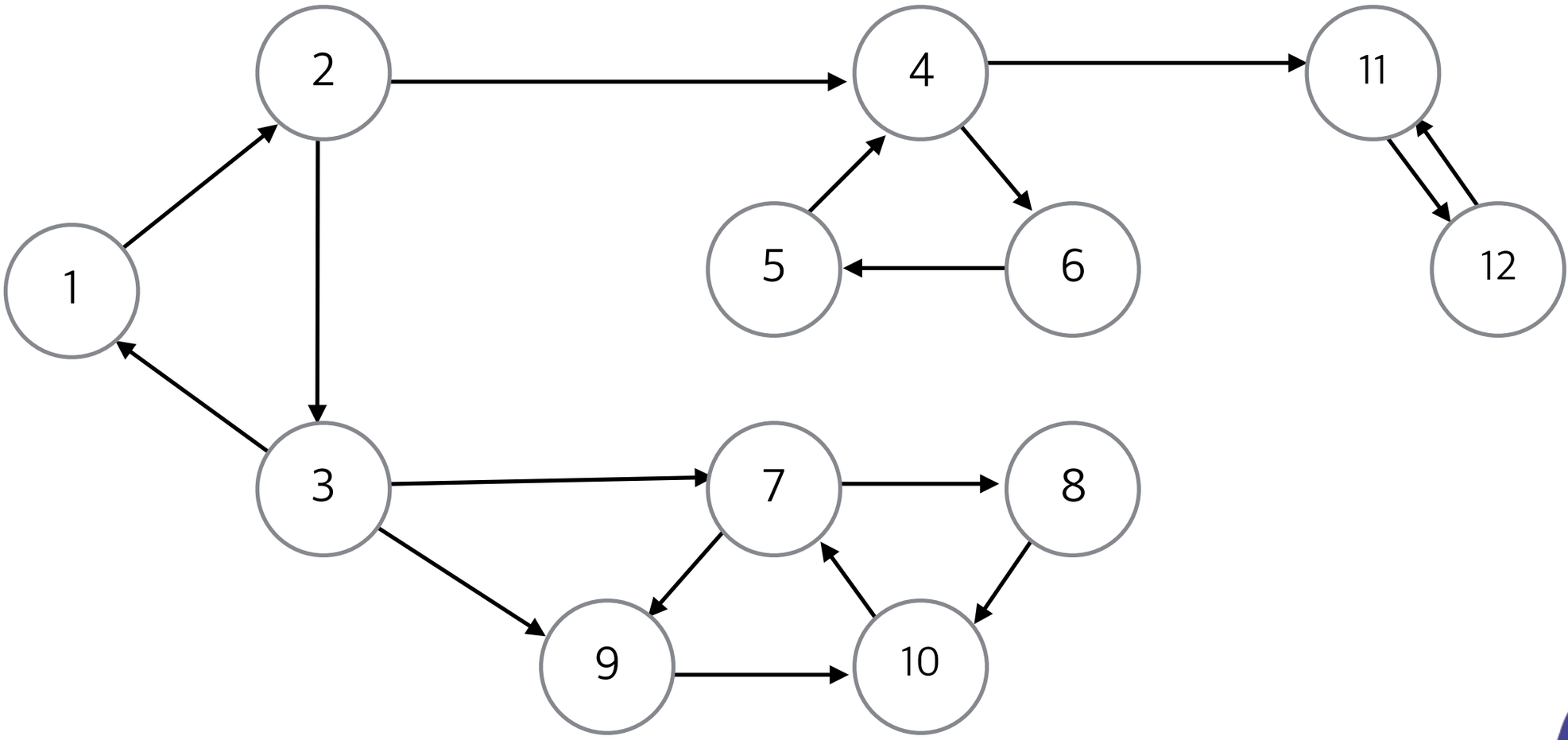
- 연결되어 있는 graph들로 나누어라 !

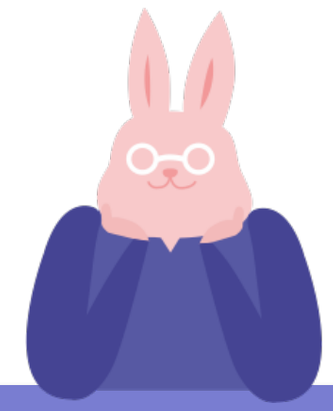




Connected Component

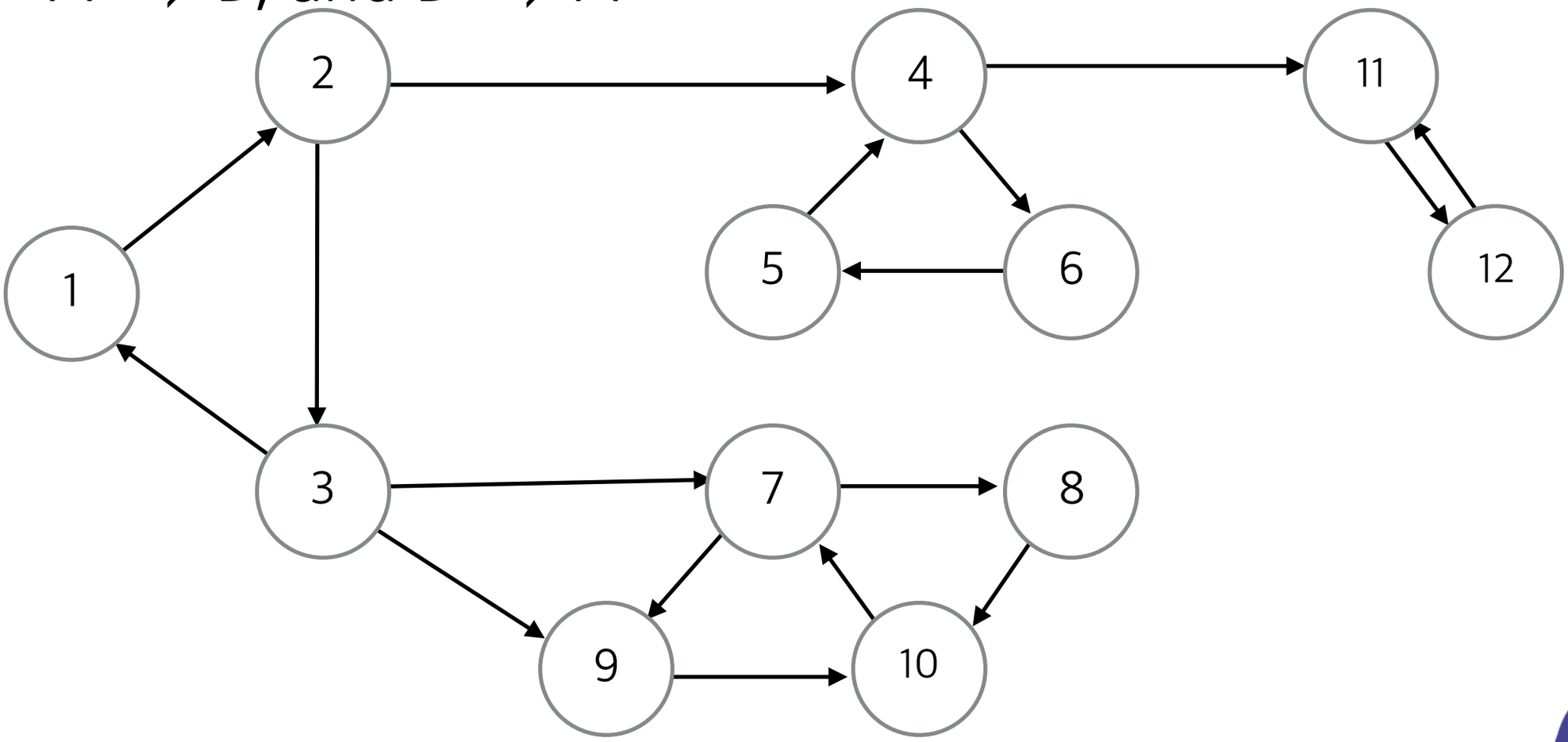
- 연결되어 있는 graph들로 나누어라 ?

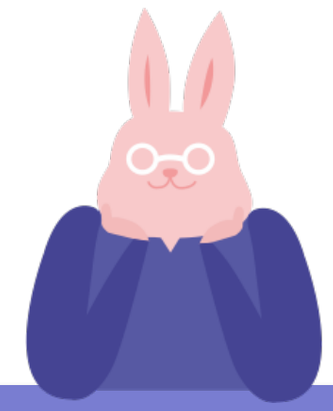




Connected Component

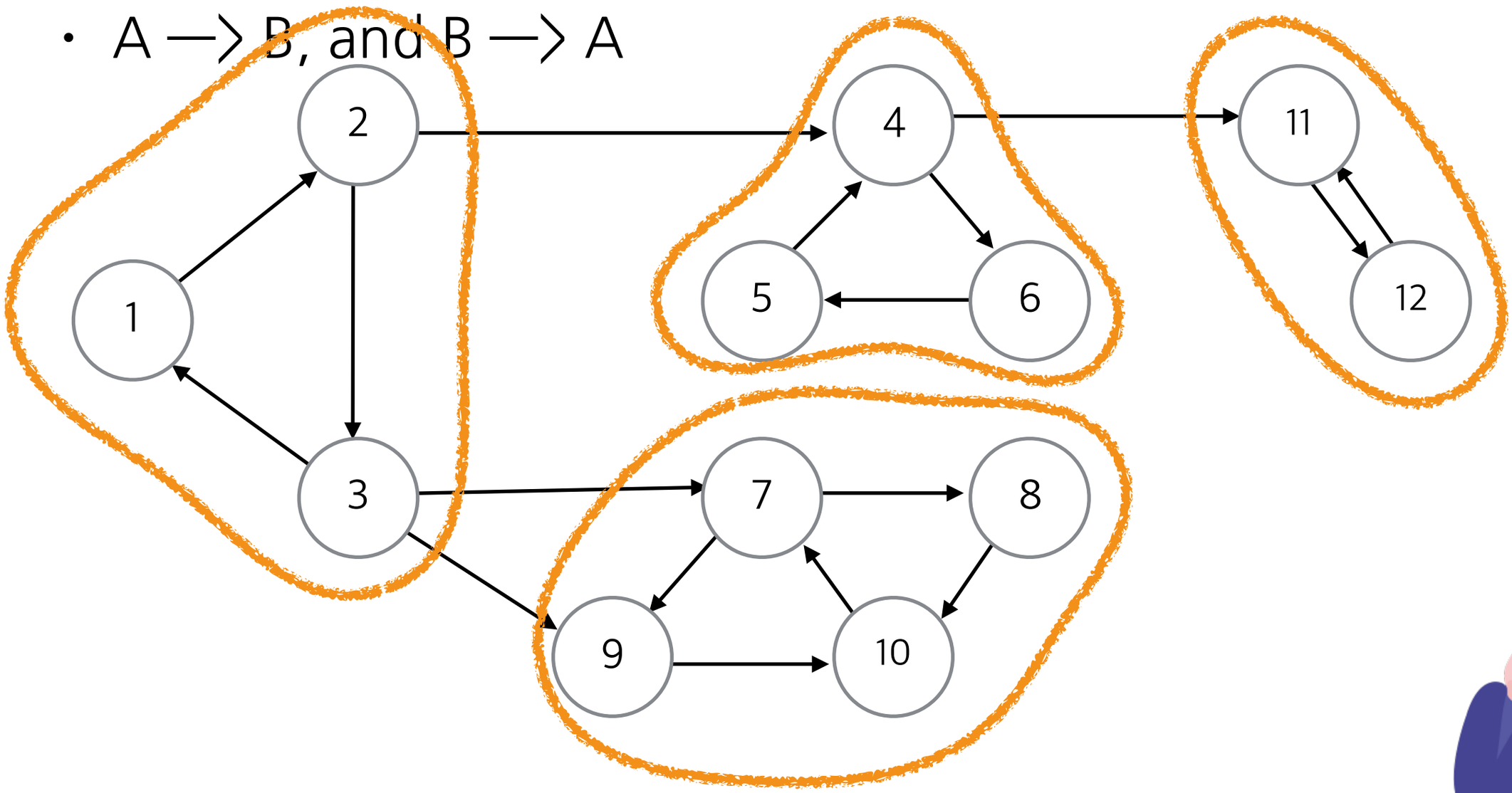
- **연결**되어 있는 graph들로 나누어라
  - A —> B, and B —> A

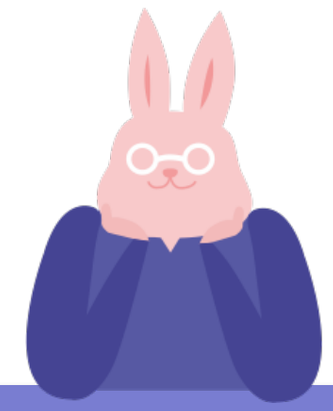




Strongly Connected Component

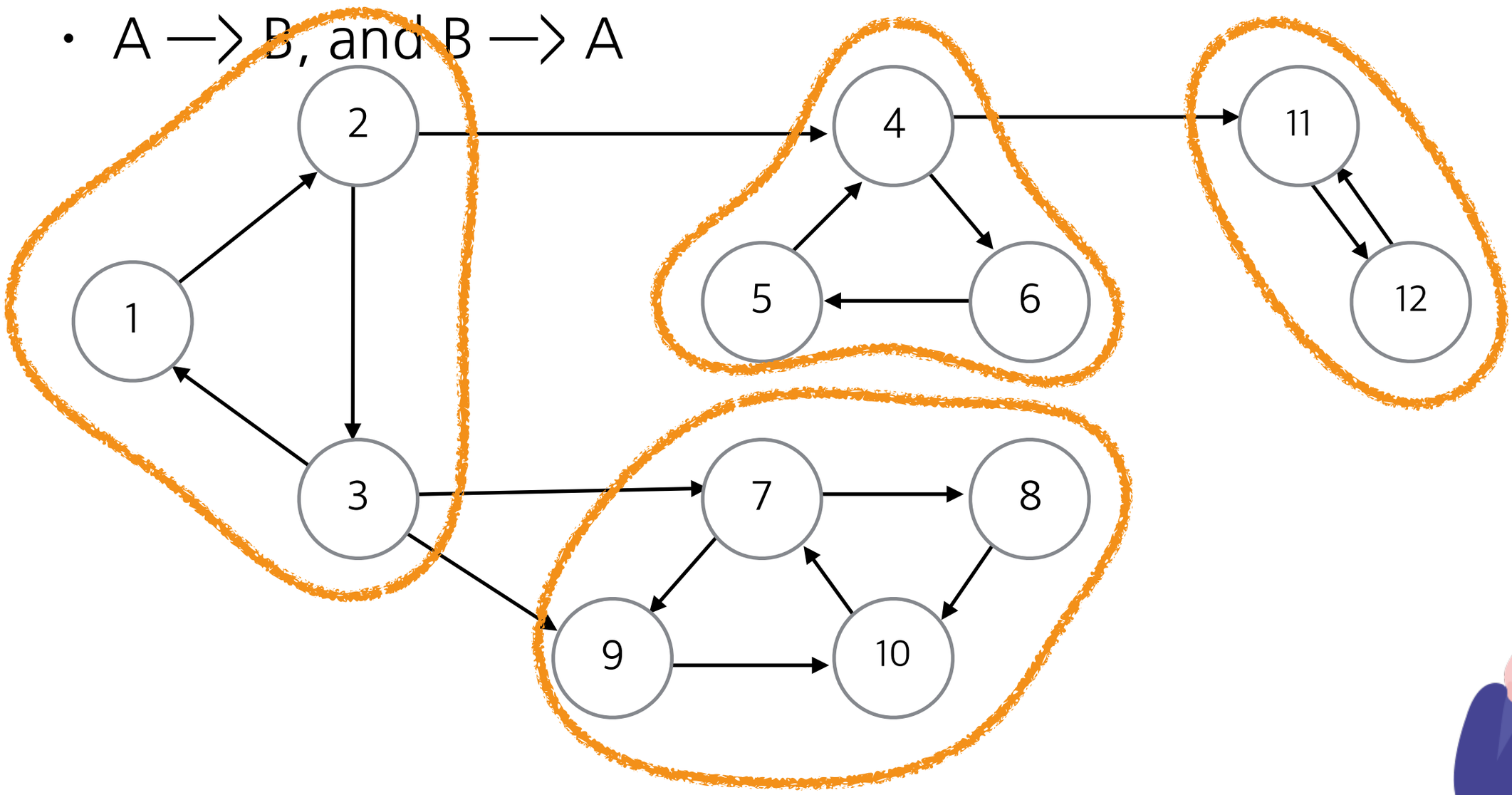
- **연결**되어 있는 graph들로 나누어라
  - A —> B, and B —> A

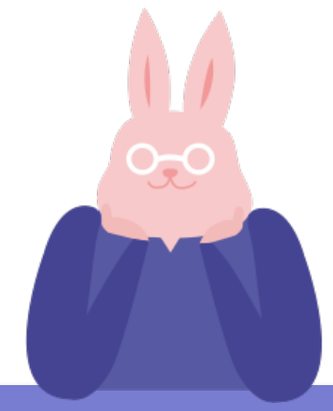




Strongly Connected Component

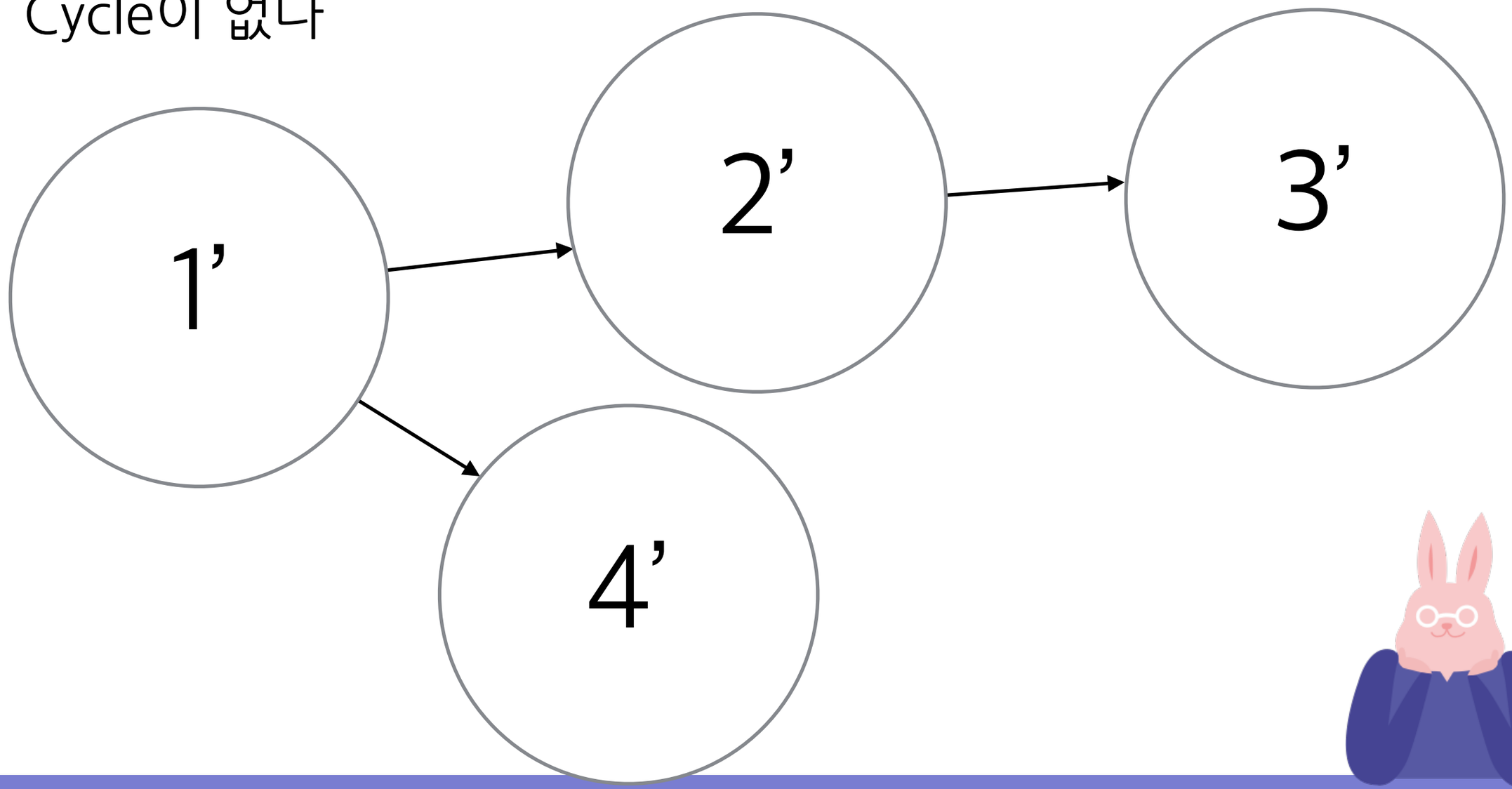
- **연결**되어 있는 graph들로 나누어라
  - A —> B, and B —> A





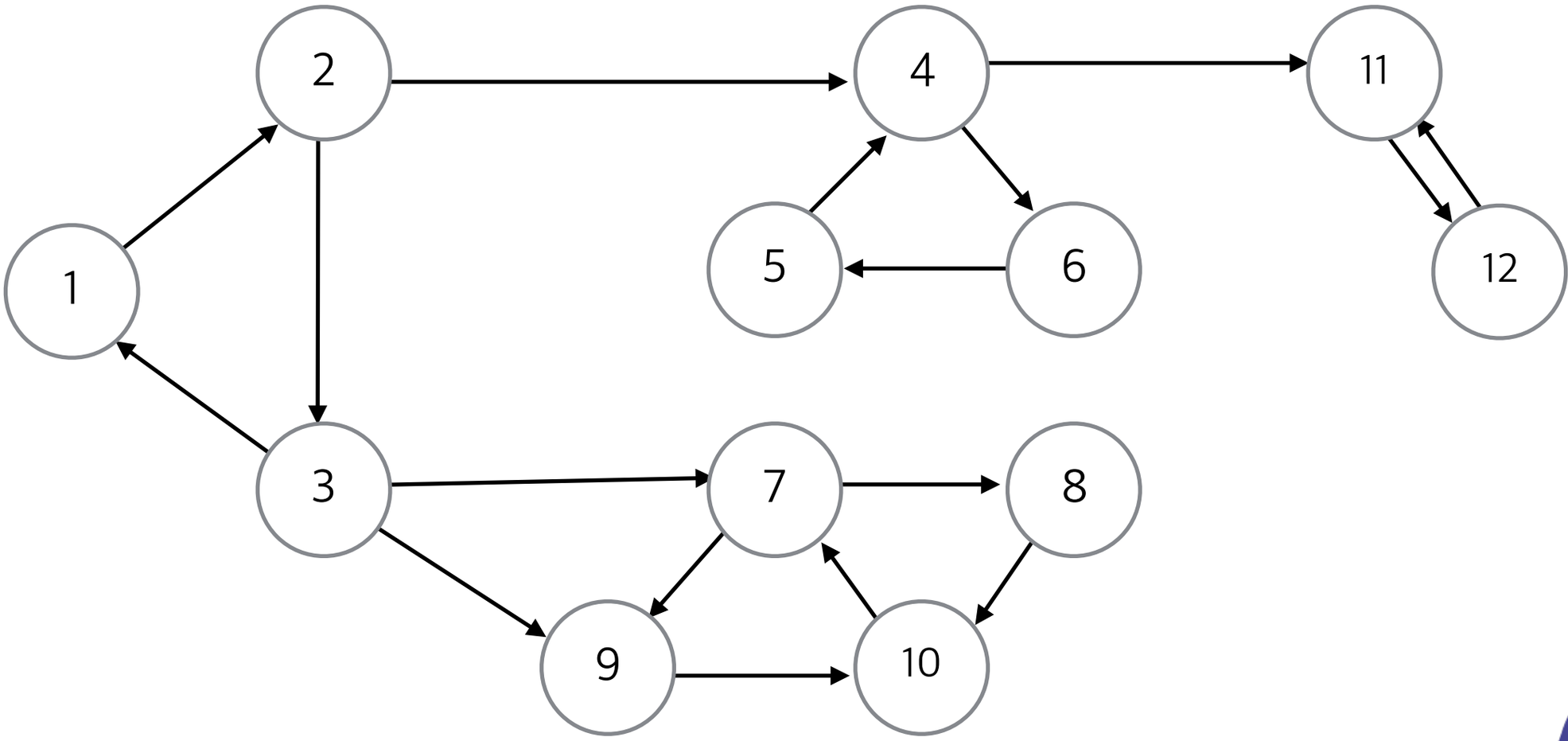
Strongly Connected Component

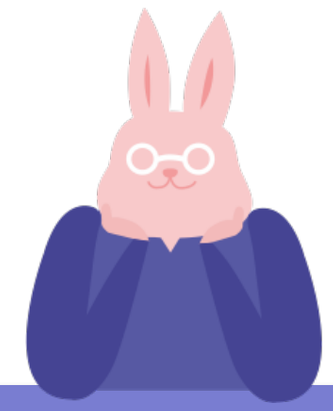
- 다음과 같이 새로운 그래프를 그릴 수 있음 (meta-graph)
  - Cycle이 없다



Strongly Connected Component

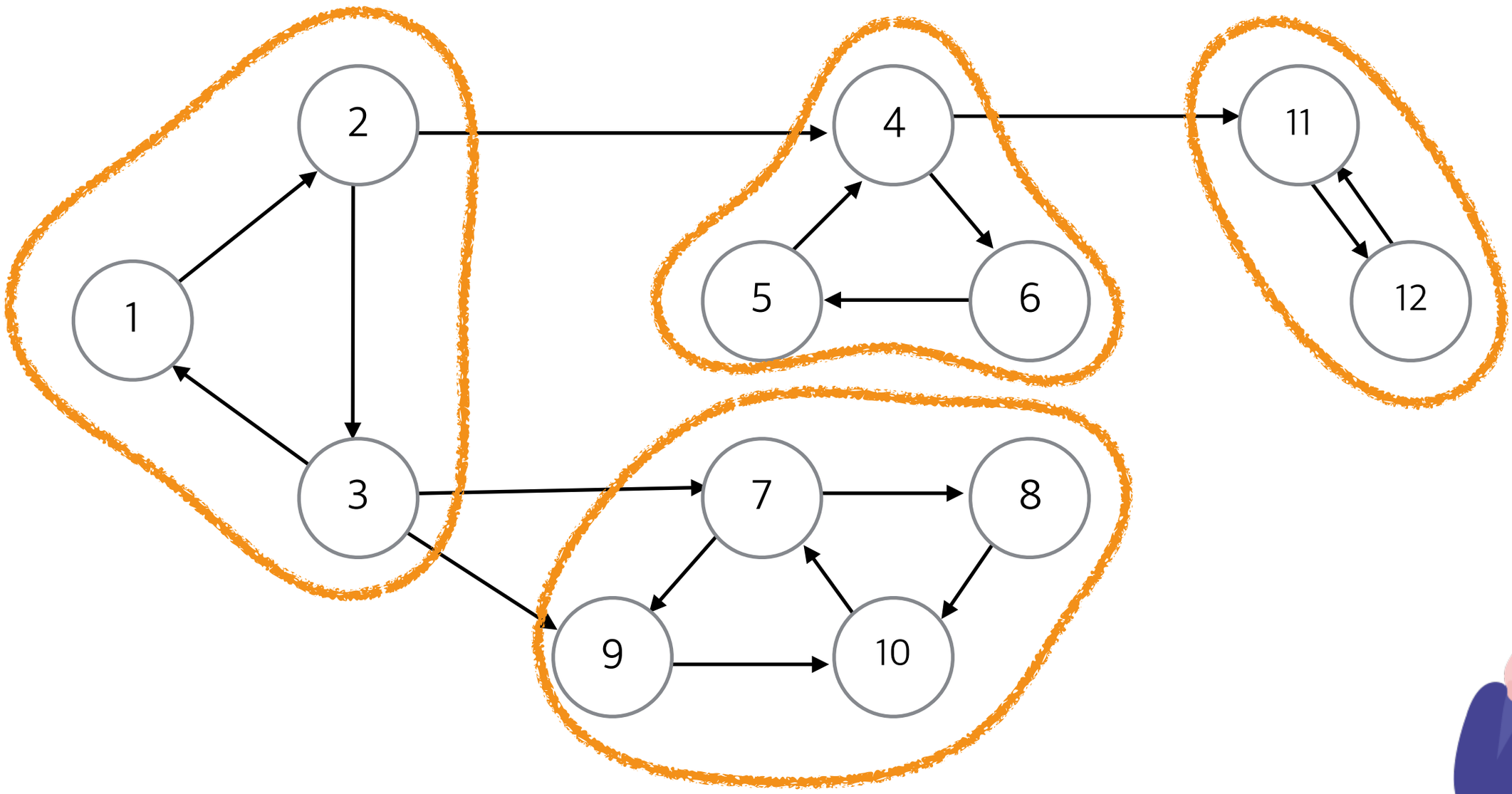
- SCC를 어떻게 구하나 ?

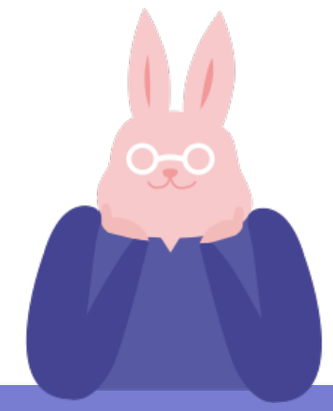




Strongly Connected Component

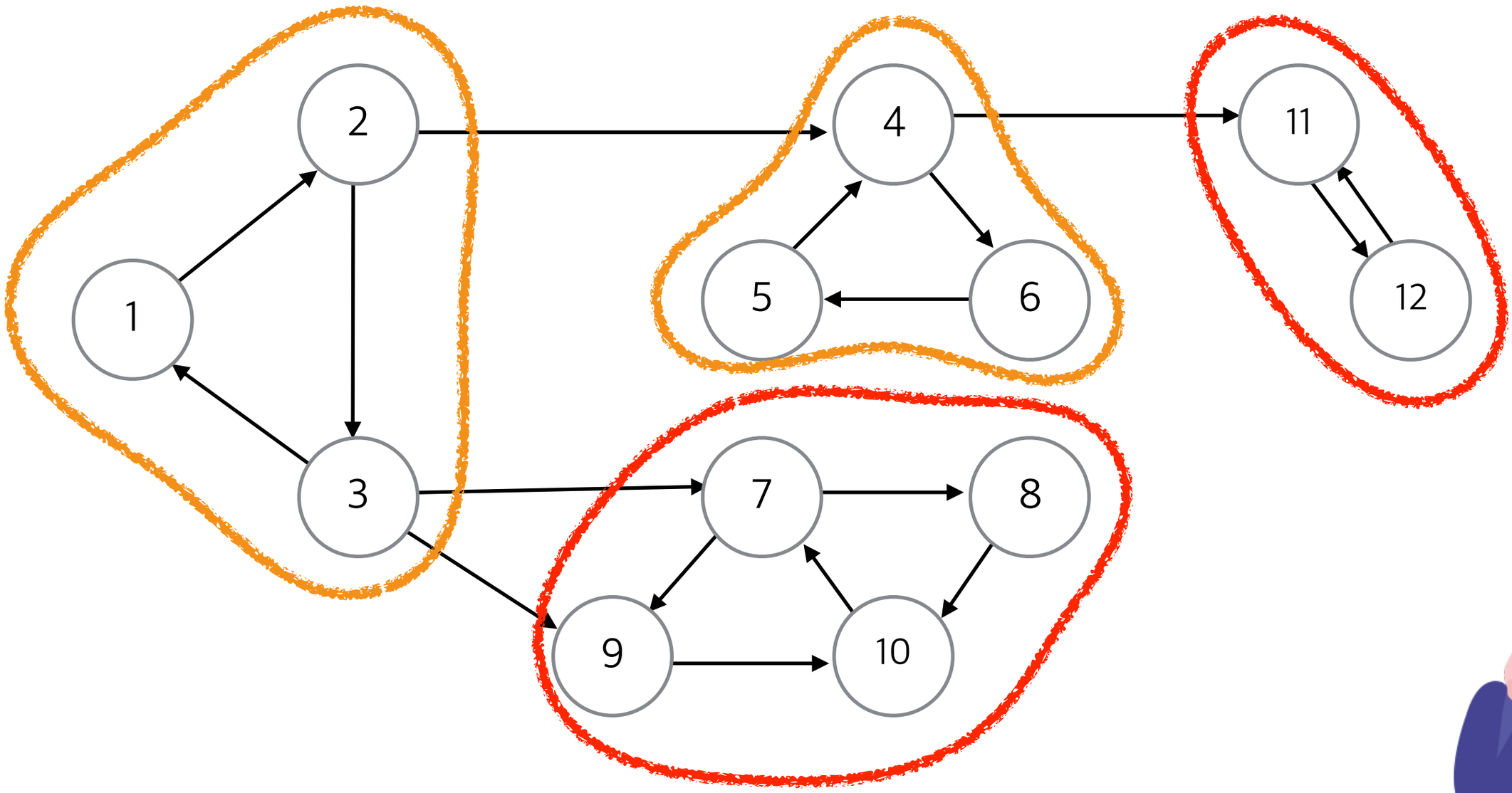
- Idea 1 : SCC가 존재한다고 가정하자.

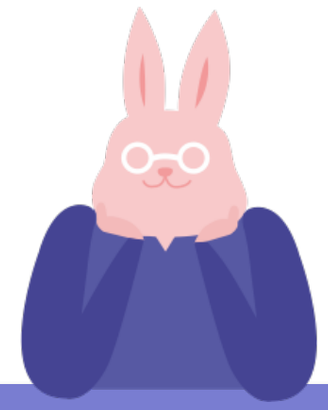




Strongly Connected Component

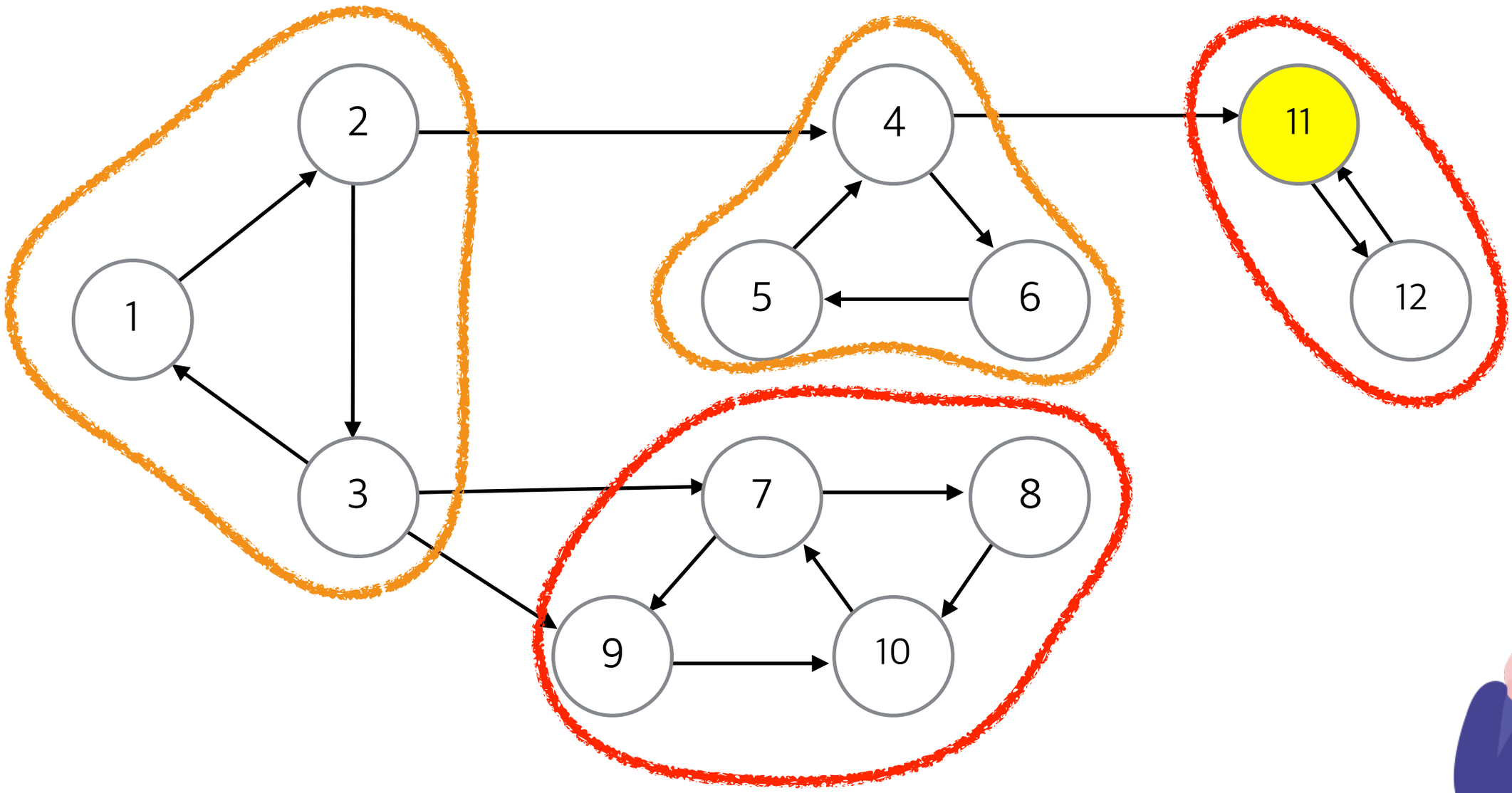
- Idea 2 : Meta-graph의 가장 끝 vertex에 주목하면

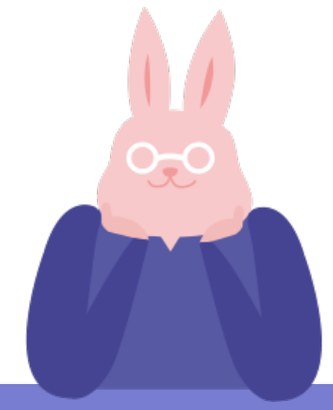




Strongly Connected Component

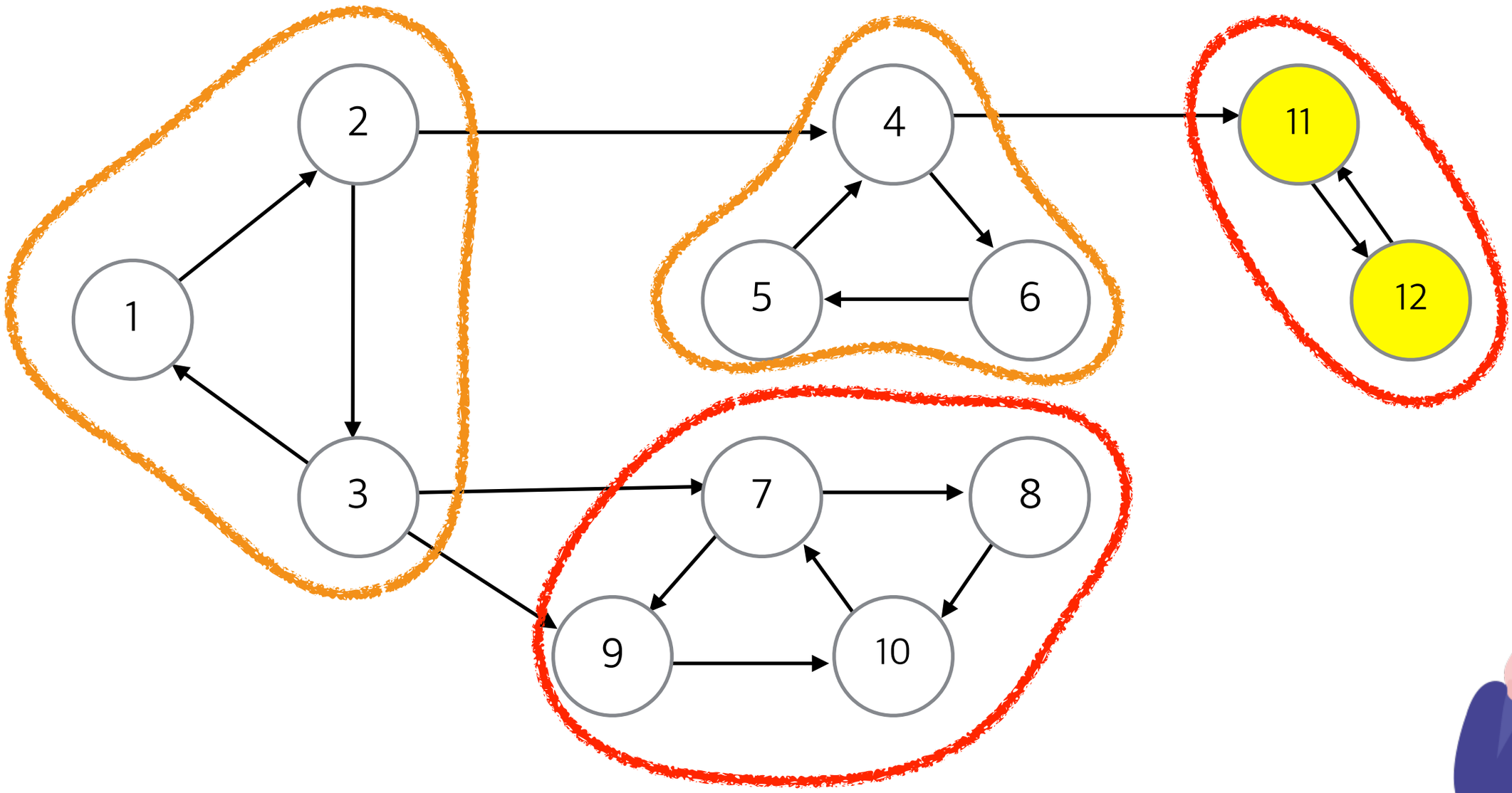
- Idea 2 : 단순 traverse가 하나의 그룹을 뽑아낸다

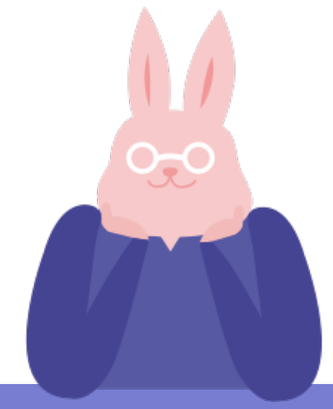




Strongly Connected Component

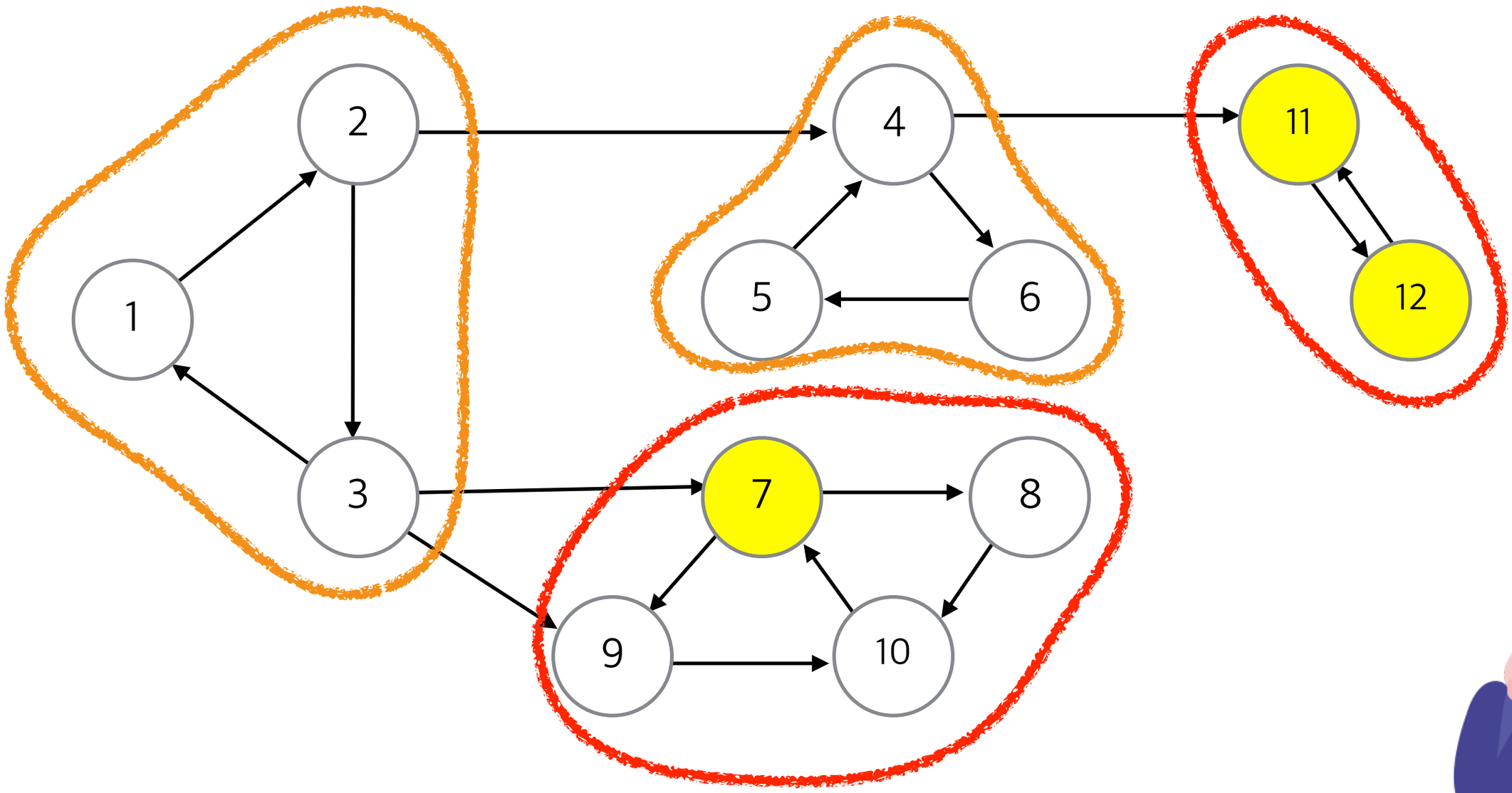
- Idea 2 : 단순 traverse가 하나의 그룹을 뽑아낸다

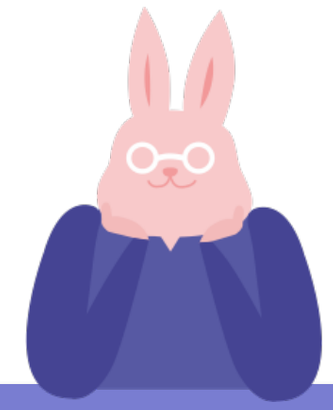




Strongly Connected Component

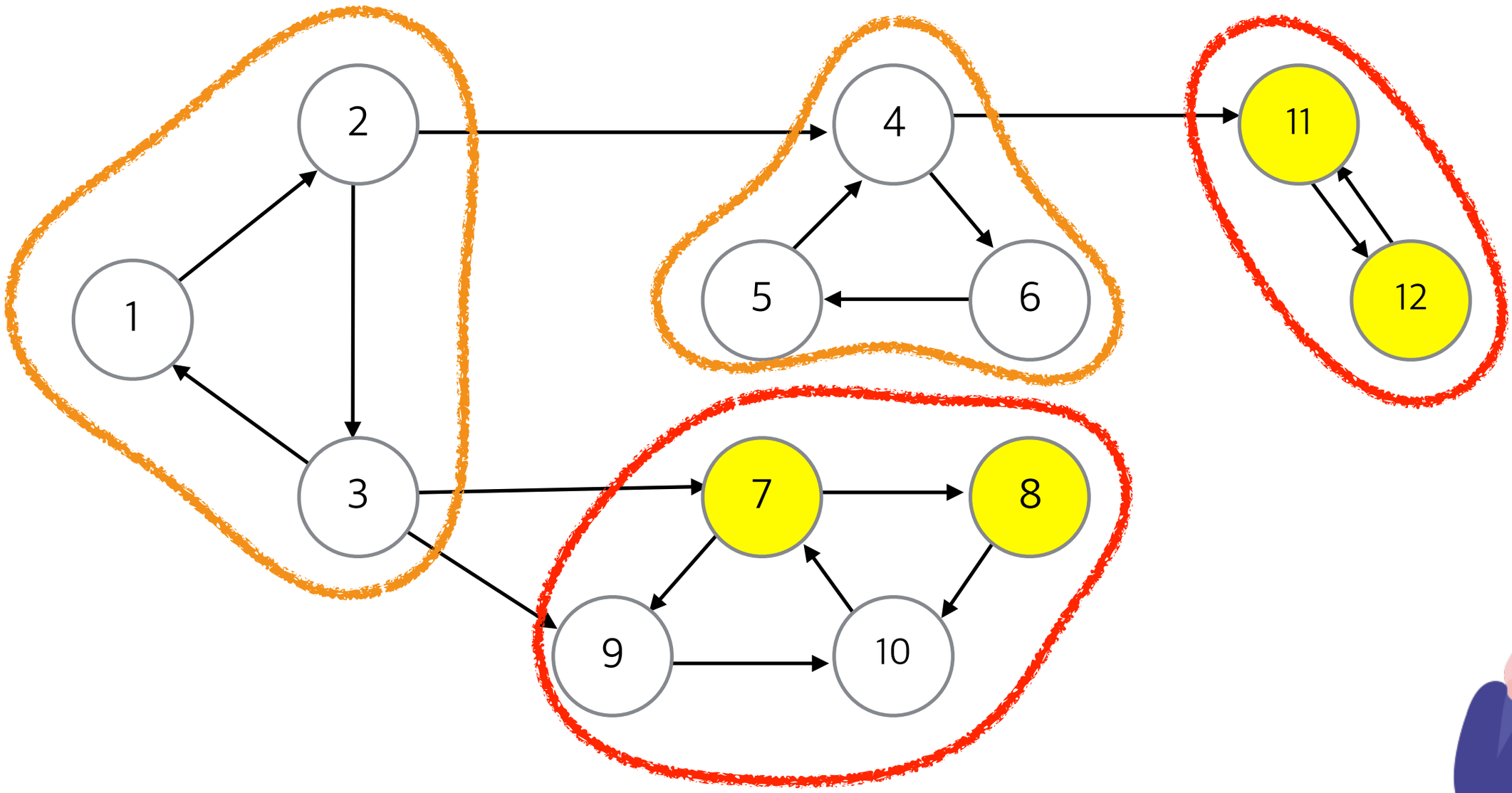
- Idea 2 : 단순 traverse가 하나의 그룹을 뽑아낸다

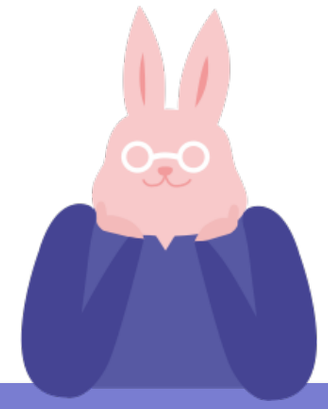




Strongly Connected Component

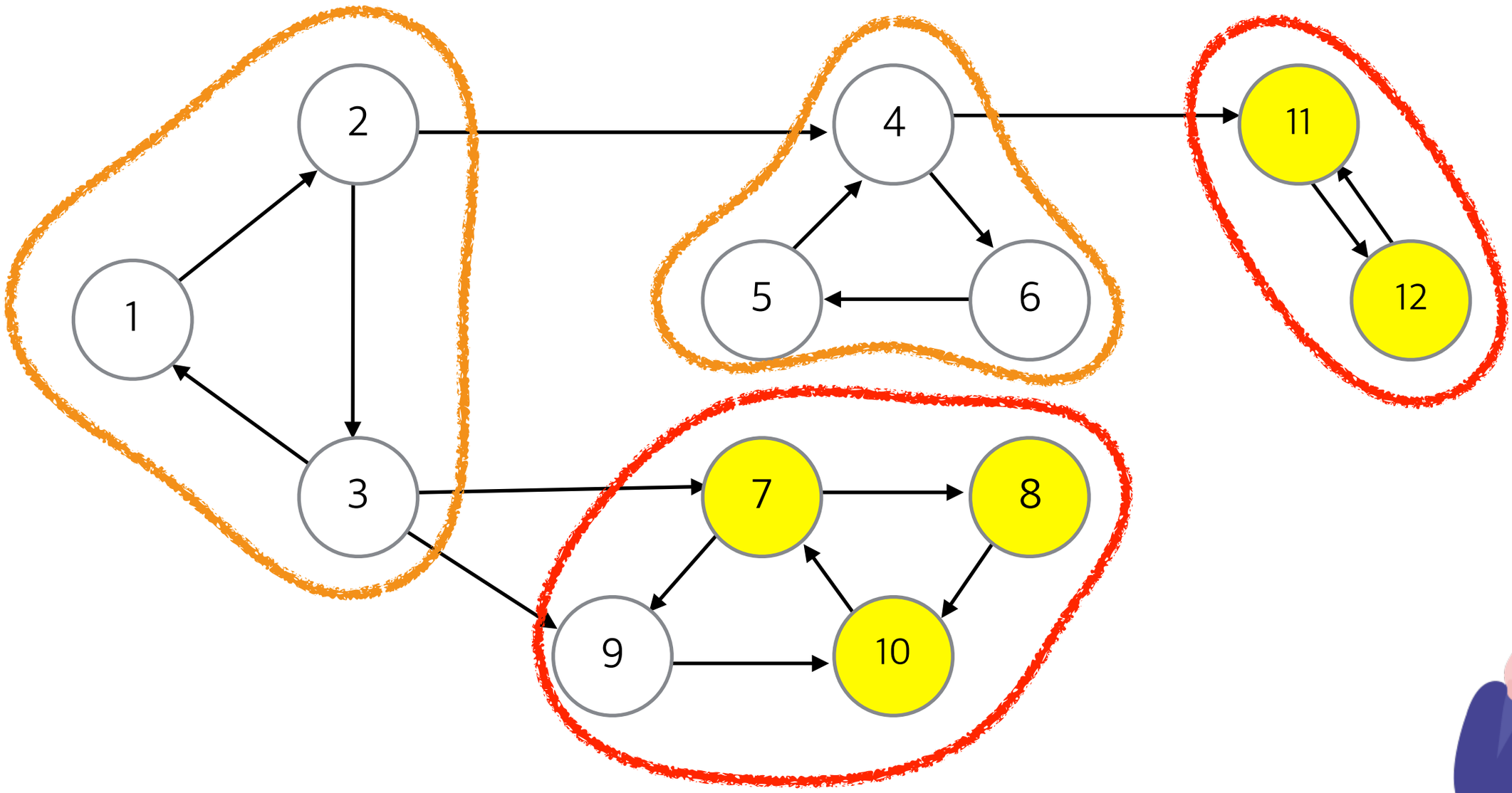
- Idea 2 : 단순 traverse가 하나의 그룹을 뽑아낸다

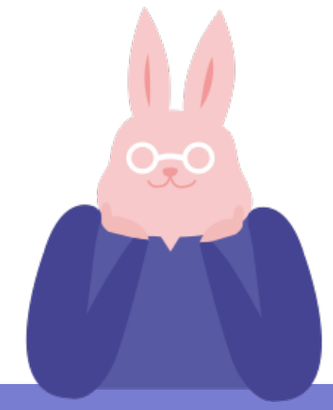




Strongly Connected Component

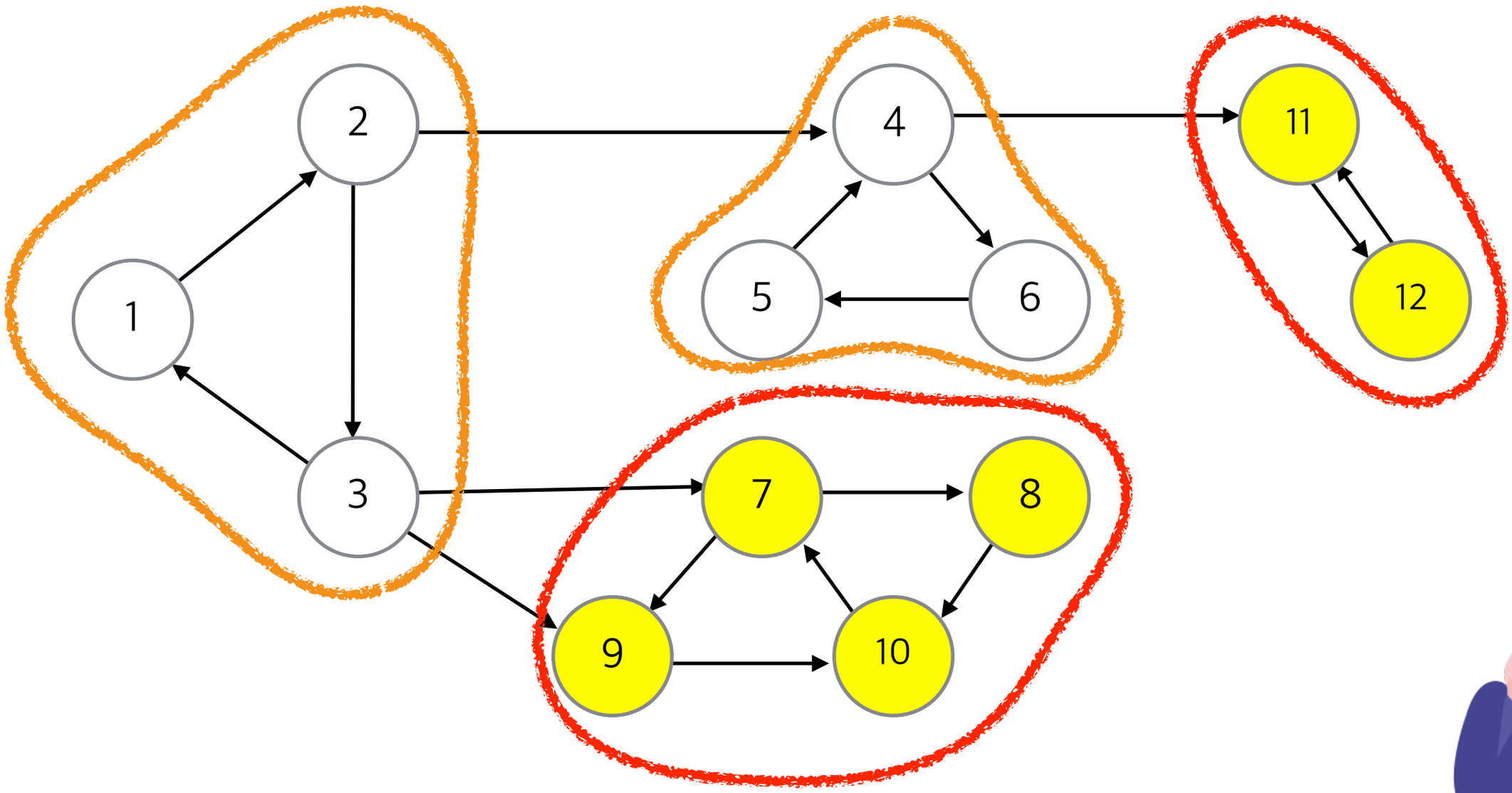
- Idea 2 : 단순 traverse가 하나의 그룹을 뽑아낸다

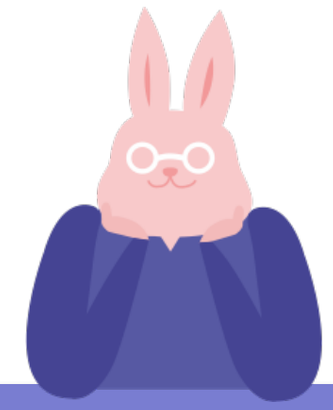




Strongly Connected Component

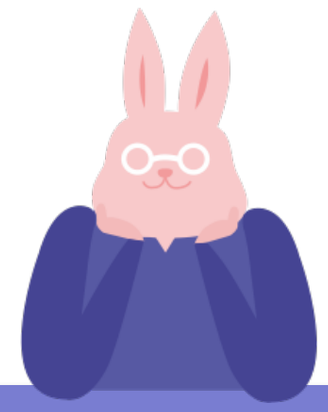
- Idea 2 : 단순 traverse가 하나의 그룹을 뽑아낸다





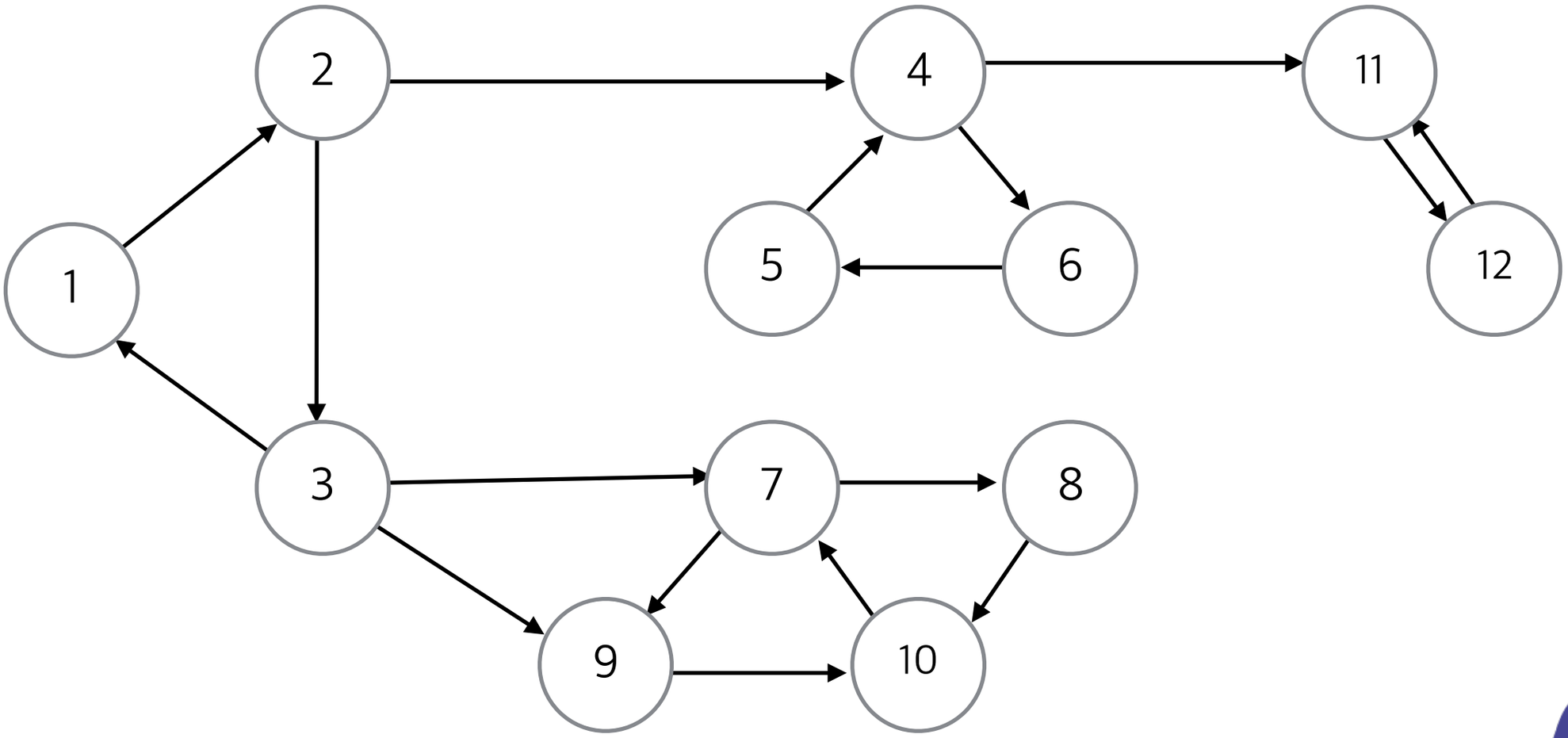
Strongly Connected Component

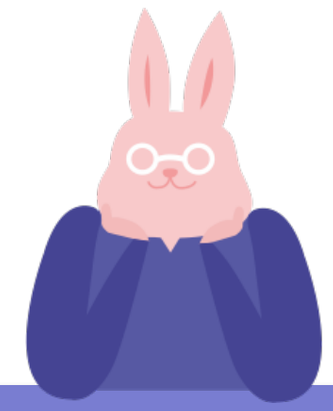
- Algorithm
  1. Meta graph의 가장 끝 vertex V 내에 있는 vertex v를 잡는다.
  2. v부터 시작하여 traverse한다. 이로써 group 하나를 찾는다.
  3. 찾은 group을 graph에서 제외한다.
  4. 다시 1.로 돌아간다.



Strongly Connected Component

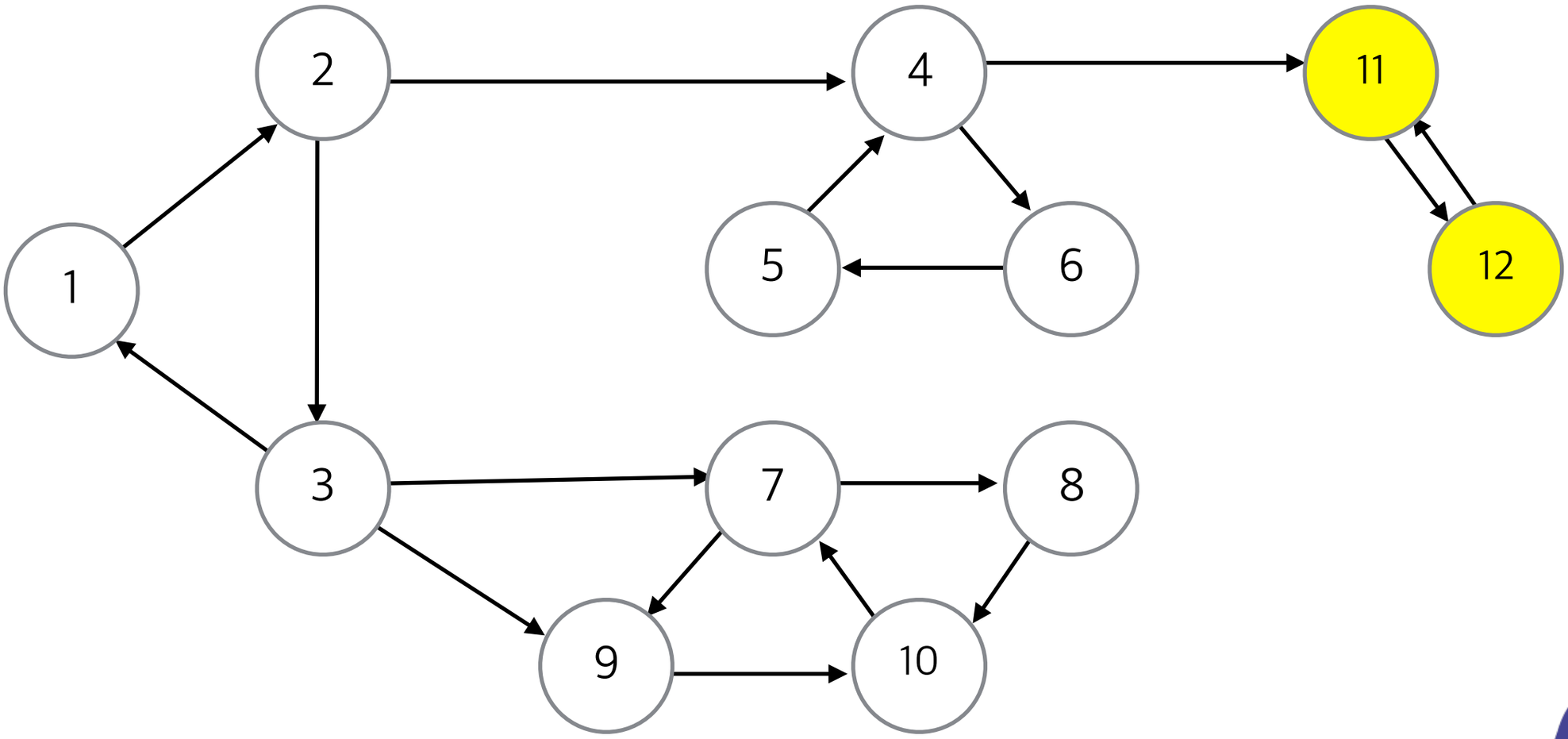
- 동작 예시

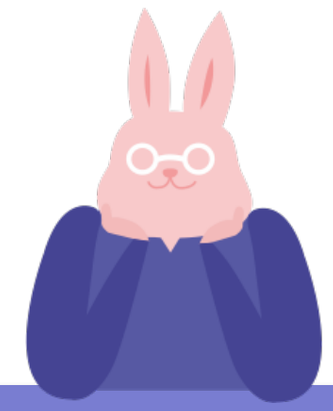




Strongly Connected Component

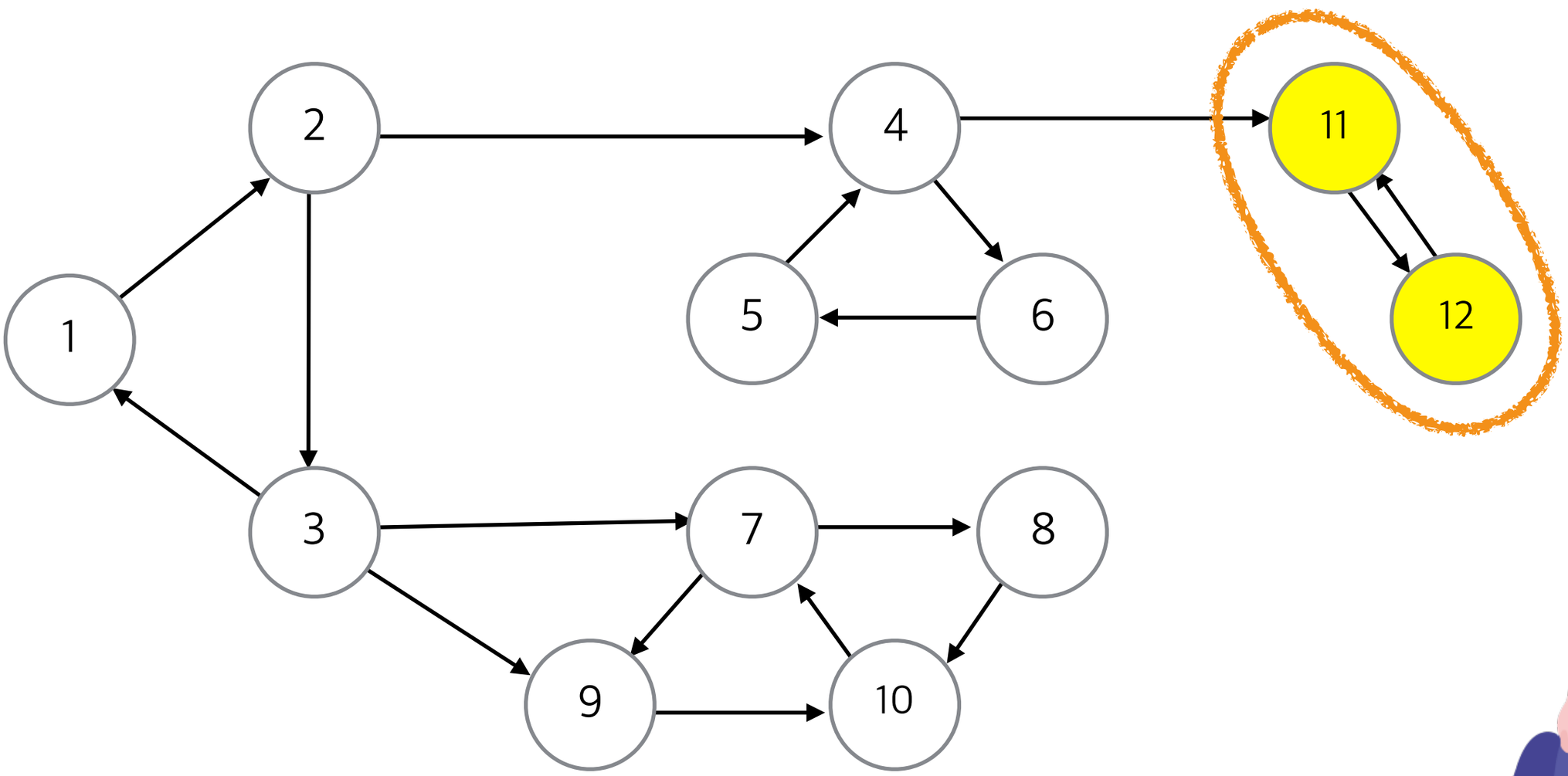
- 동작 예시

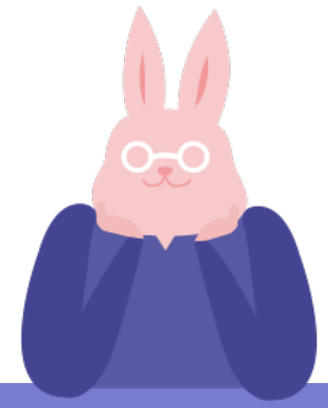




Strongly Connected Component

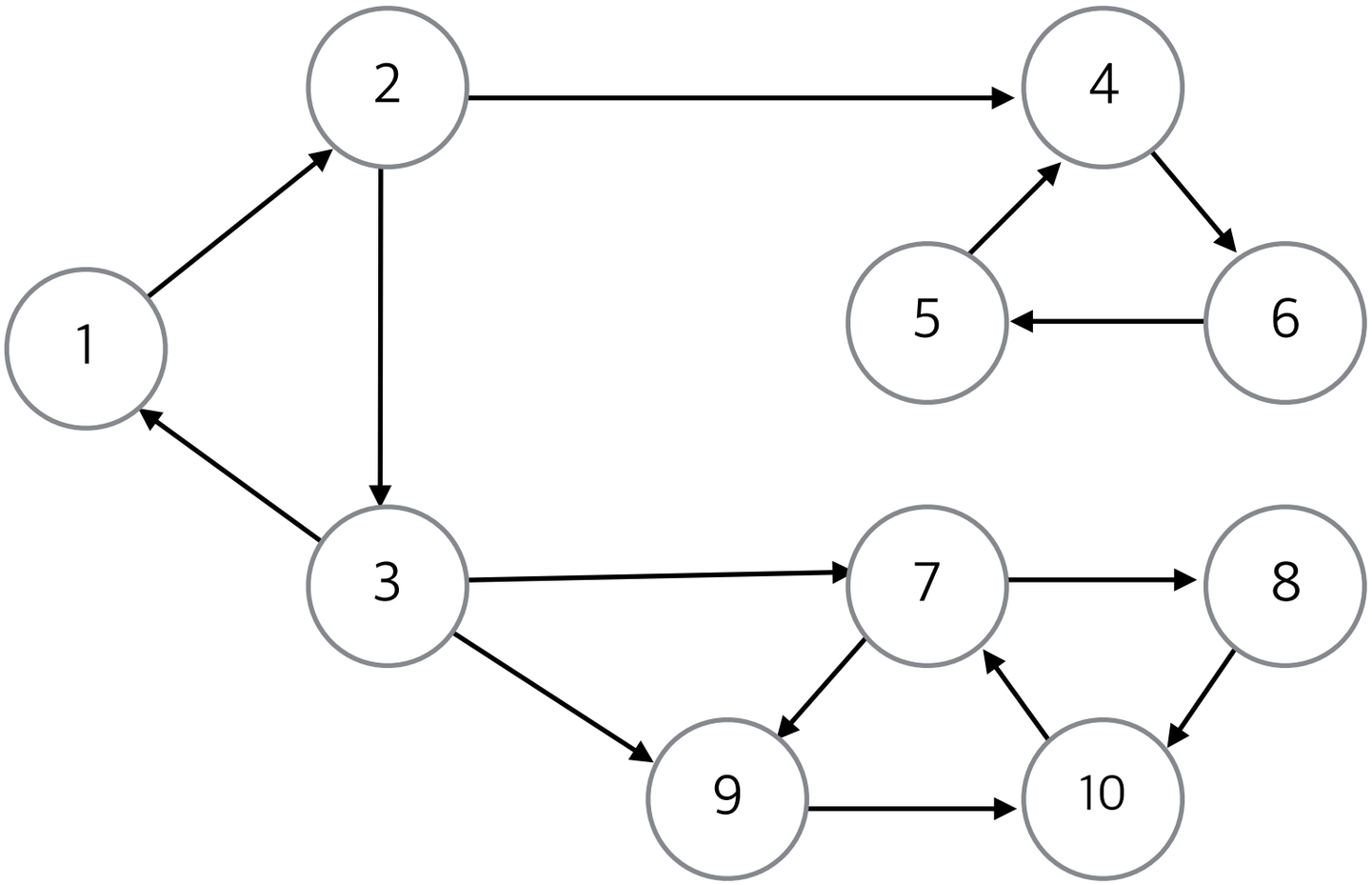
- 동작 예시





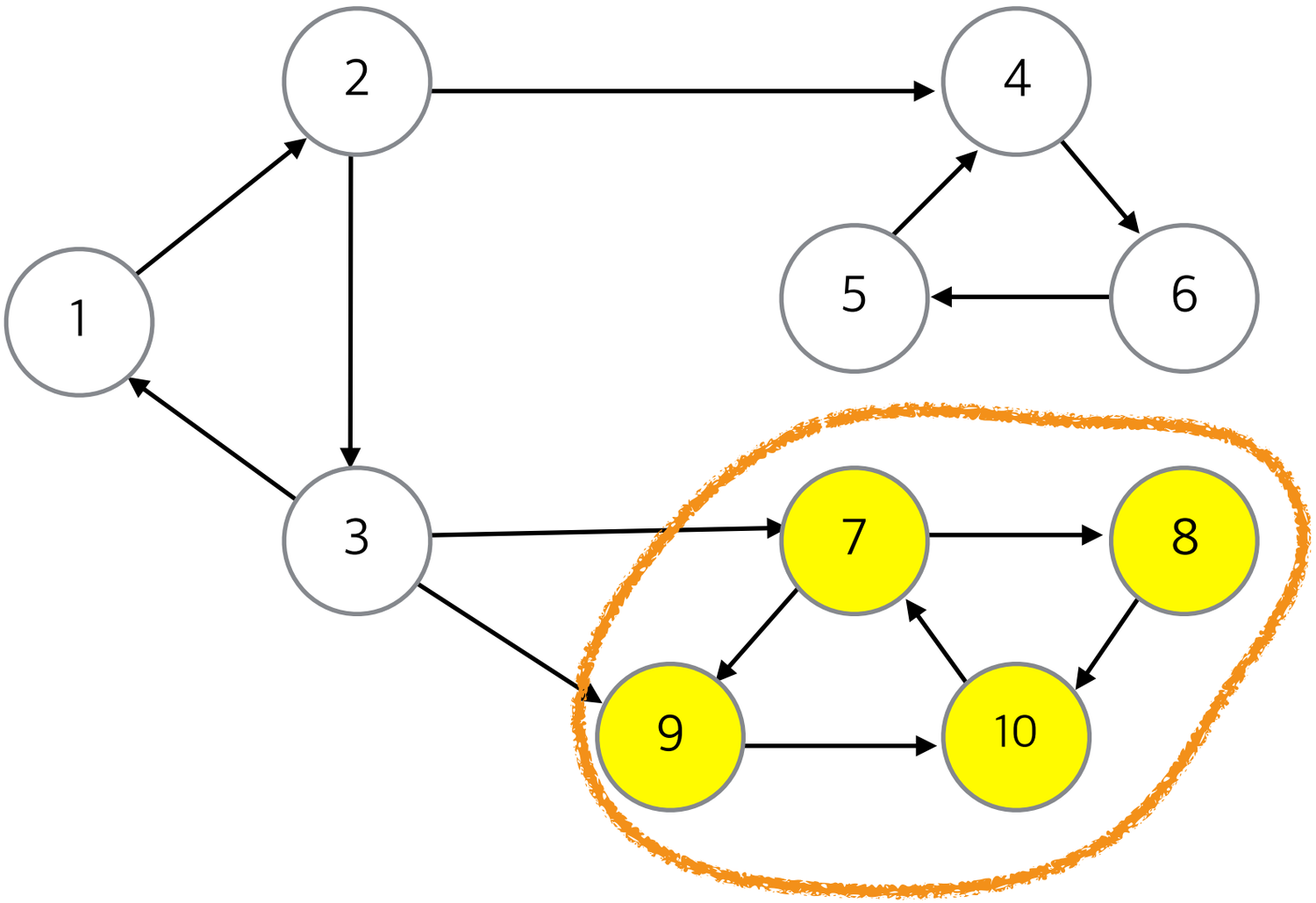
Strongly Connected Component

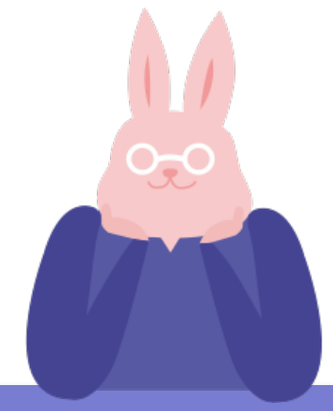
- 동작 예시



Strongly Connected Component

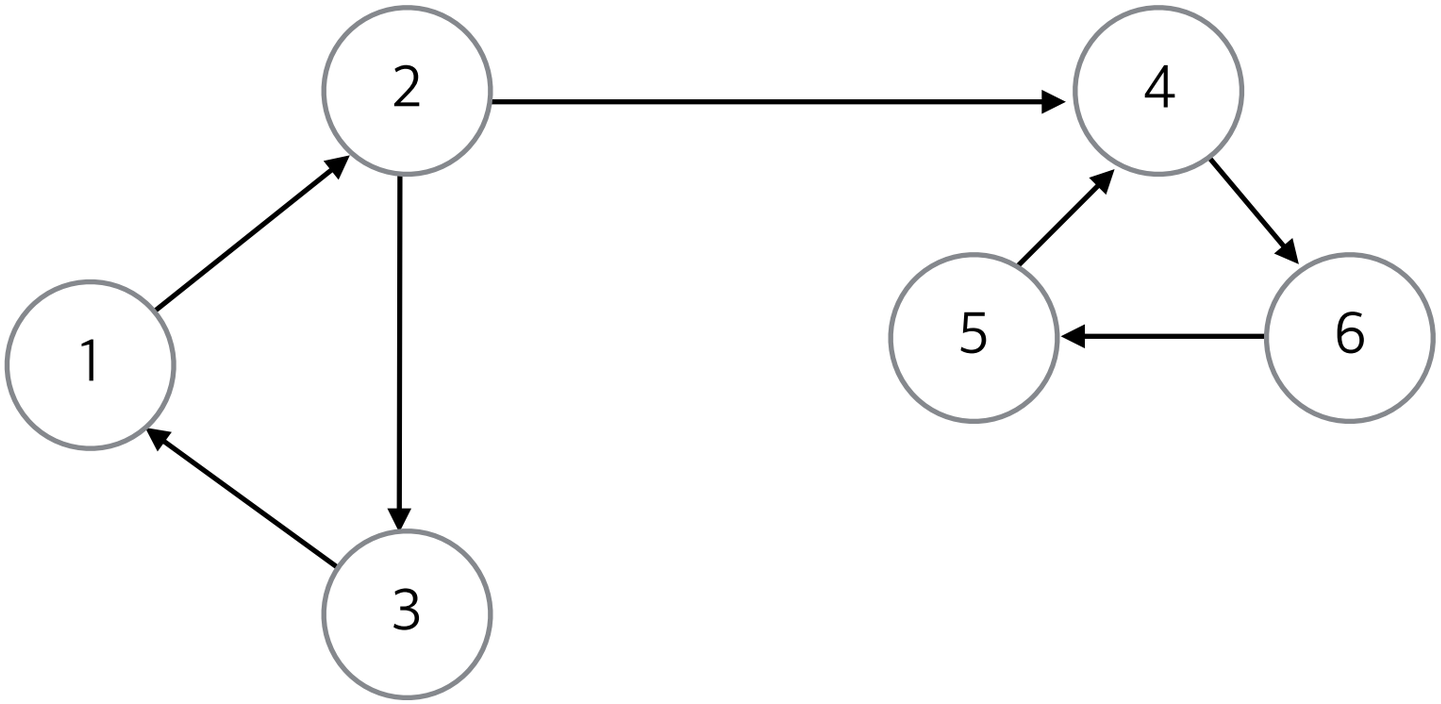
- 동작 예시

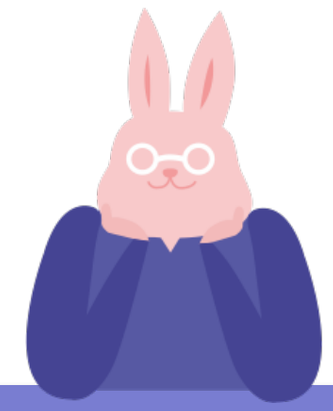




Strongly Connected Component

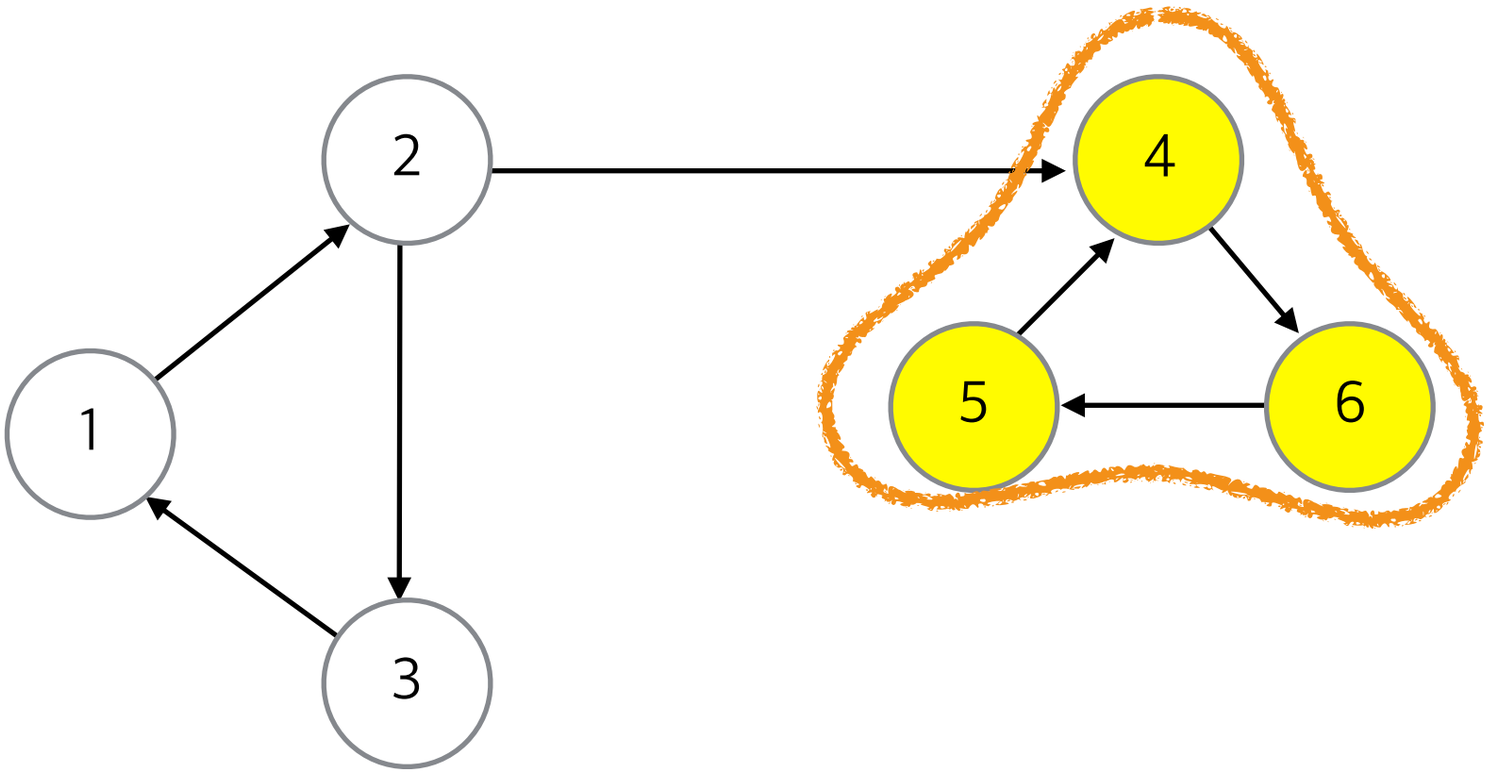
- 동작 예시

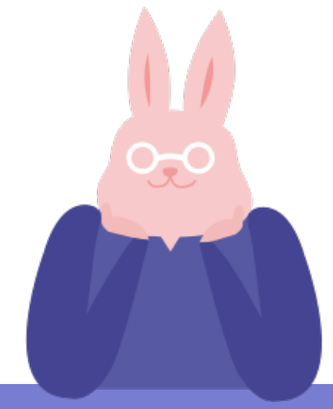




Strongly Connected Component

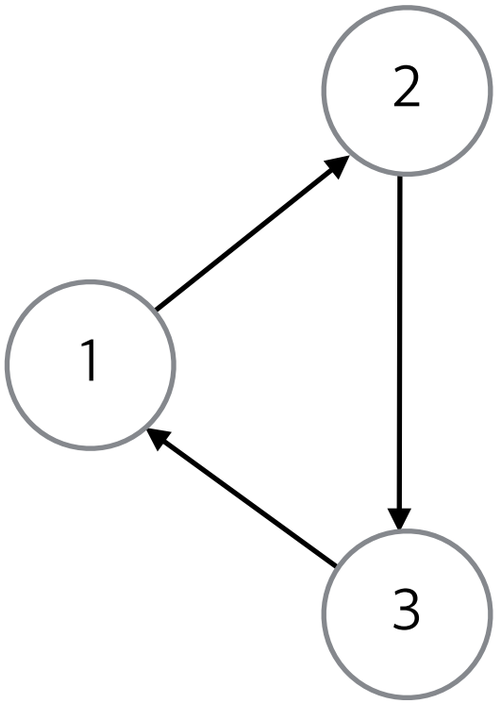
- 동작 예시

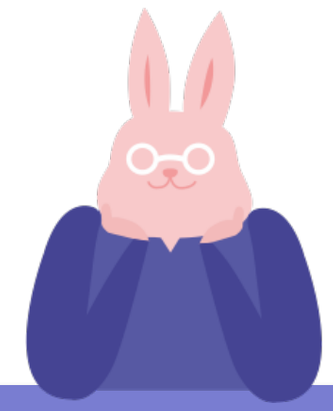




Strongly Connected Component

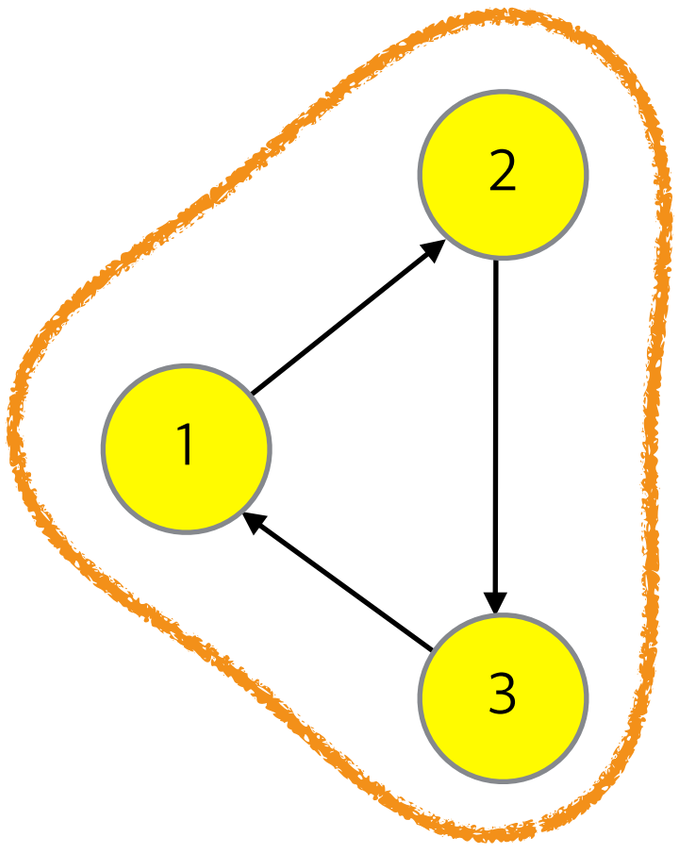
- 동작 예시

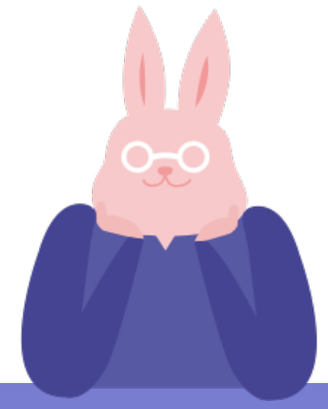




Strongly Connected Component

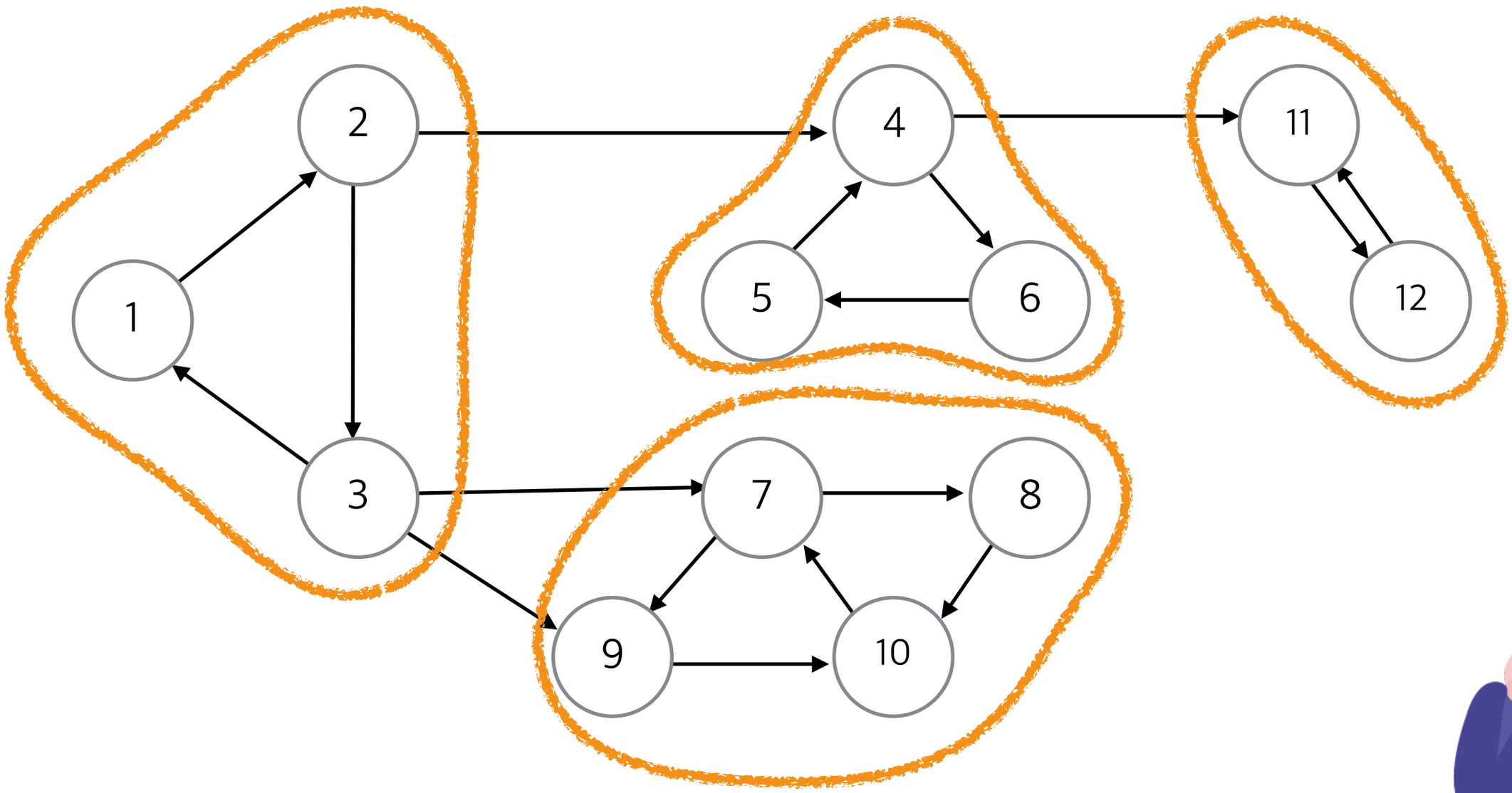
- 동작 예시

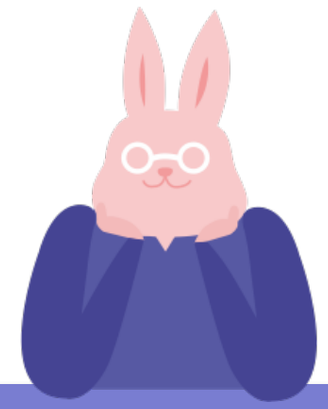




Strongly Connected Component

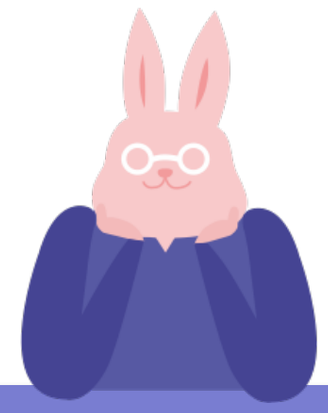
- 동작 예시





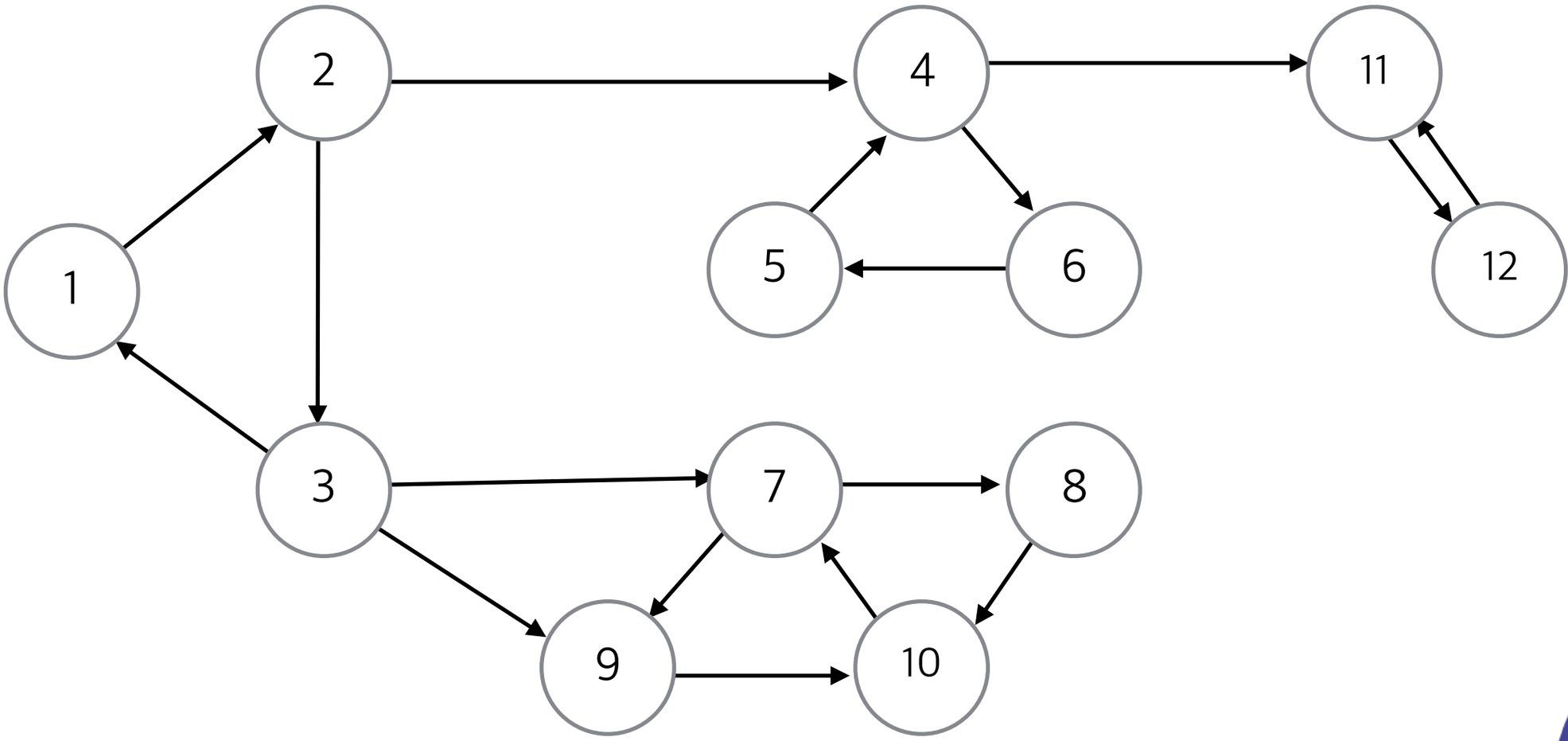
Strongly Connected Component

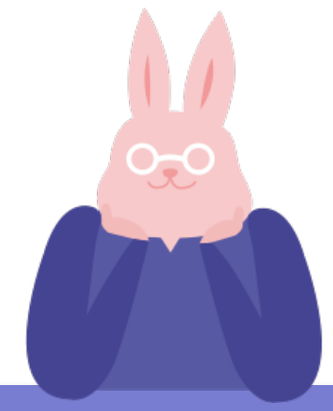
- Algorithm
  1. **Meta graph의 가장 끝 vertex V 내에 있는 vertex v를 잡는다.**
  2. v부터 시작하여 traverse한다. 이로써 group 하나를 찾는다.
  3. 찾은 group을 graph에서 제외한다.
  4. 다시 1.로 돌아간다.



Strongly Connected Component

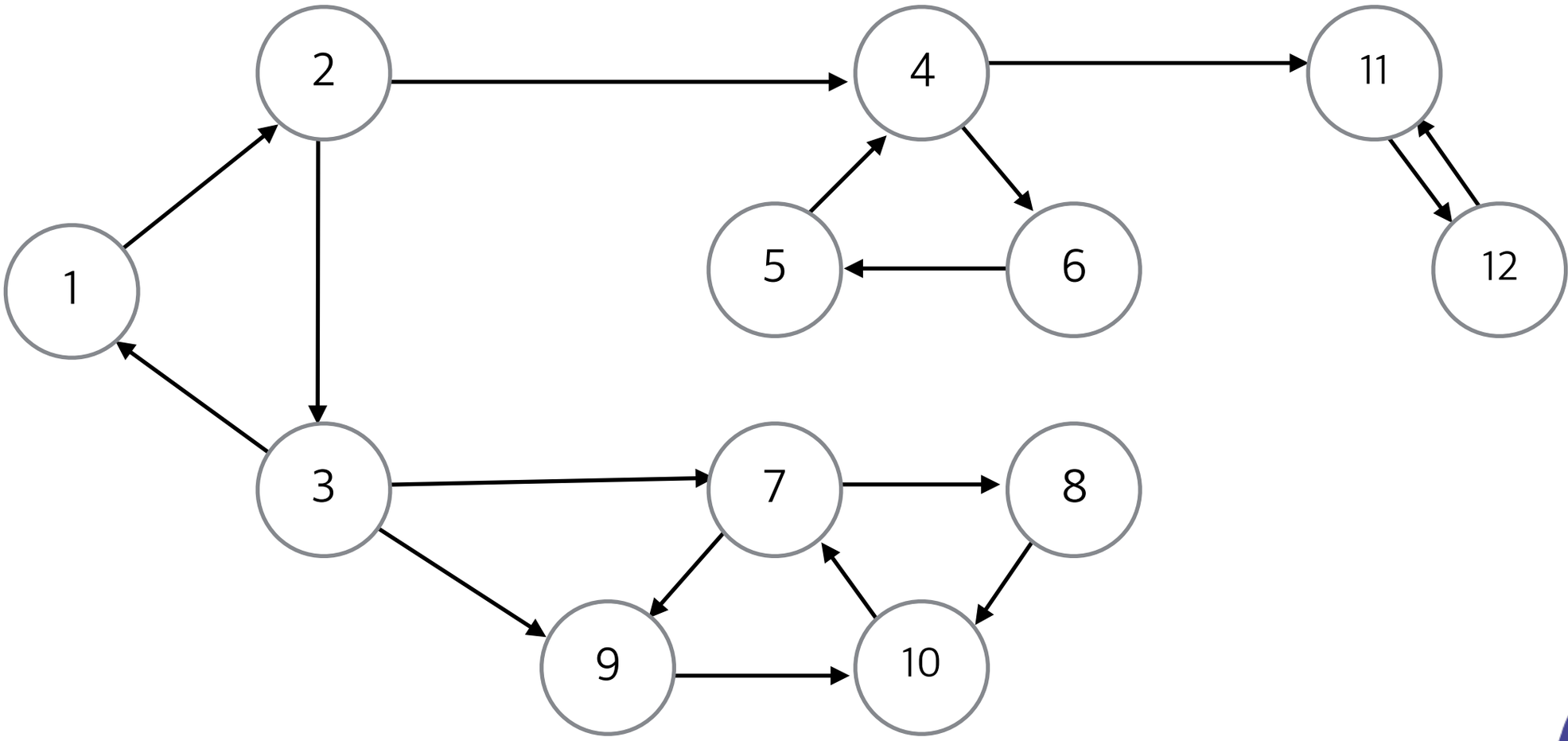
- 어떻게 vertex v를 잡을 것인가 ?

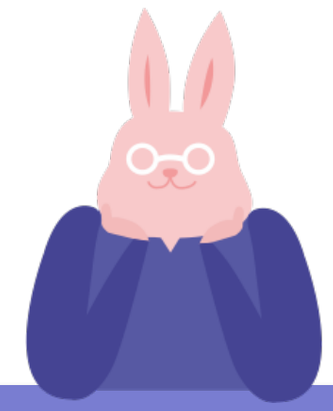




Strongly Connected Component

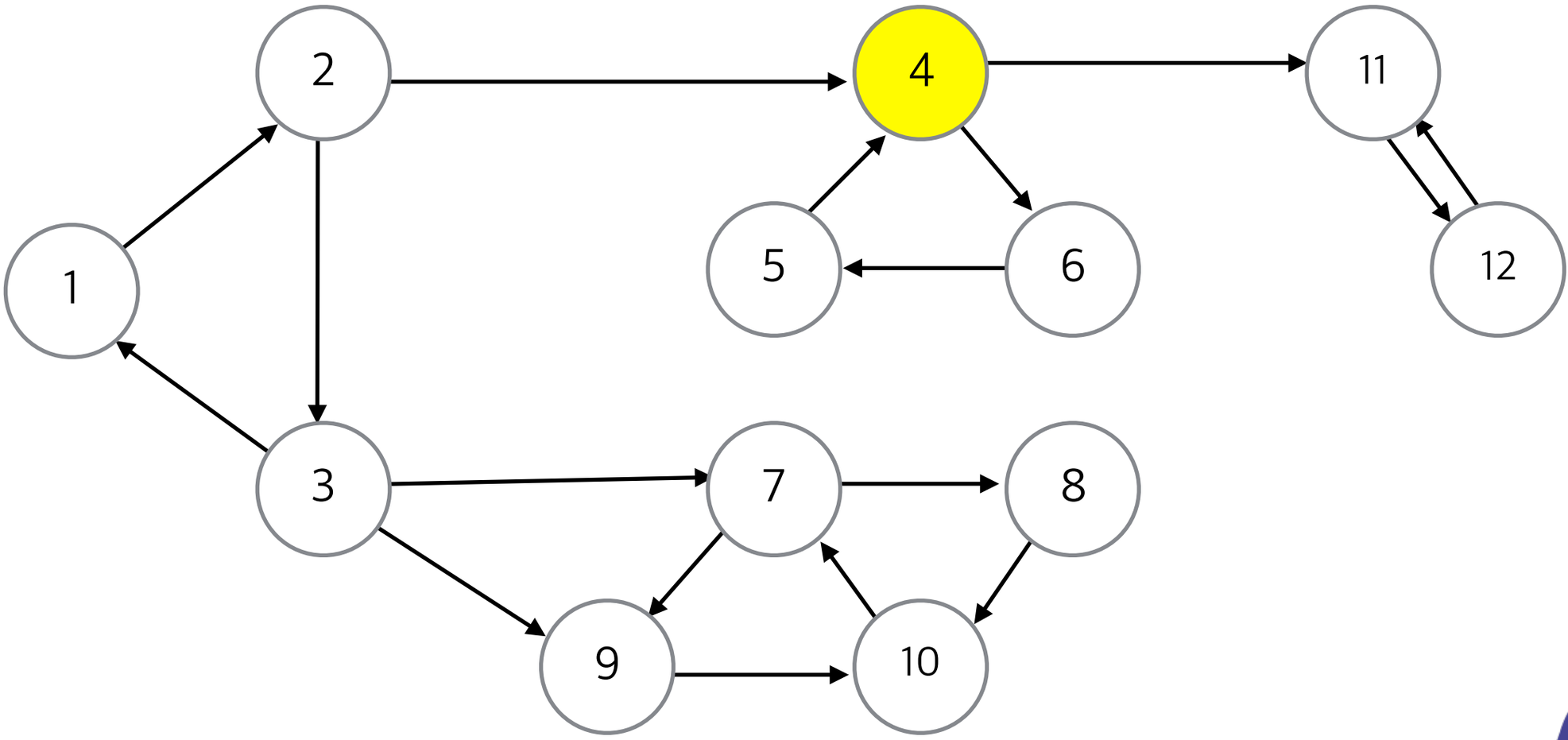
- Idea 1. DFS를 하면 부모-자식 관계(?)가 파악된다

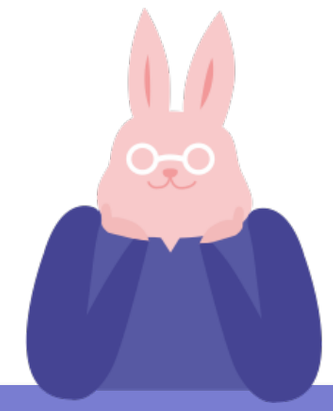




Strongly Connected Component

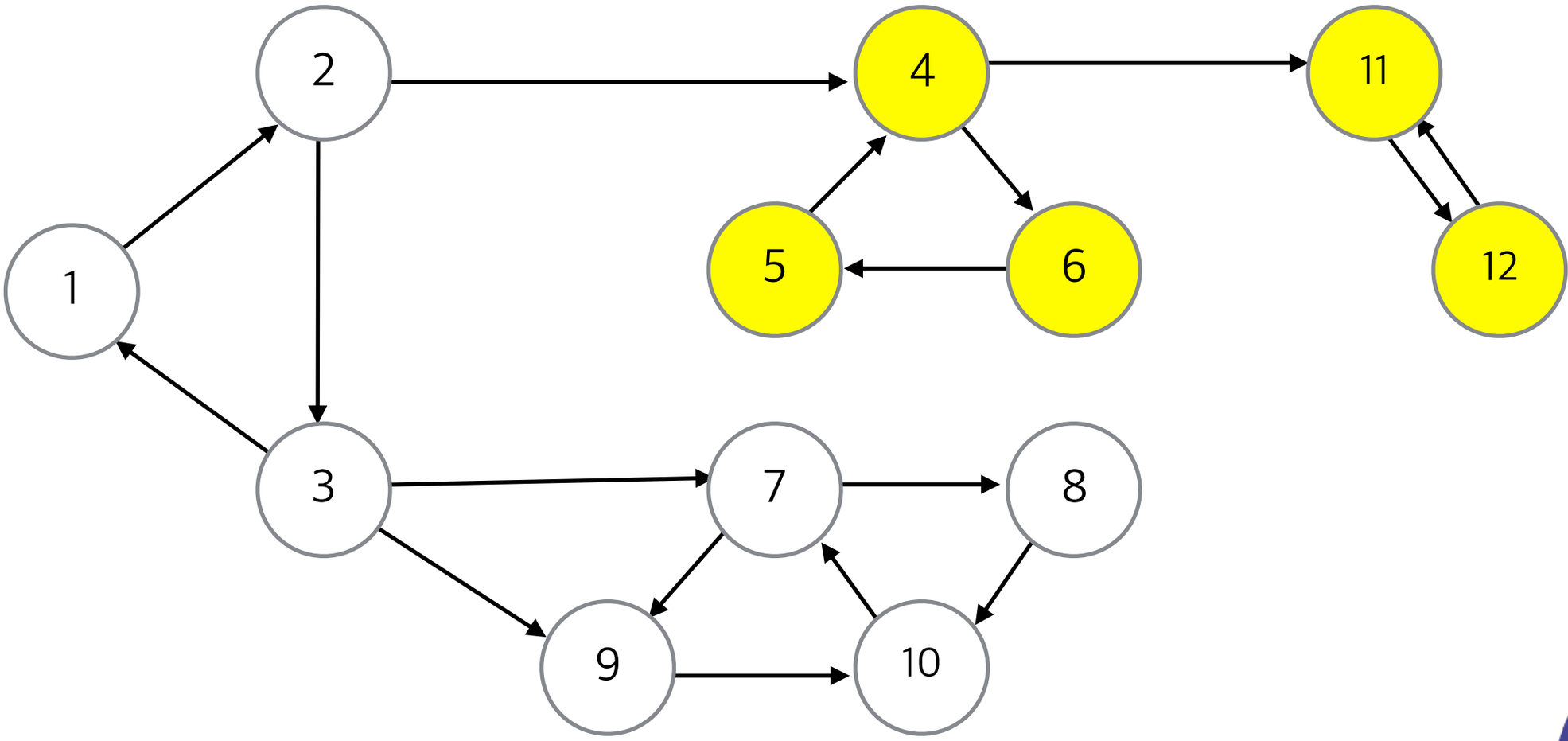
- Idea 1. DFS를 하면 부모-자식 관계(?)가 파악된다

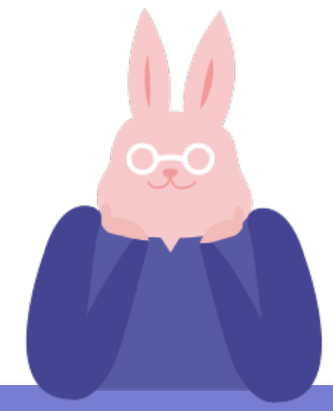




Strongly Connected Component

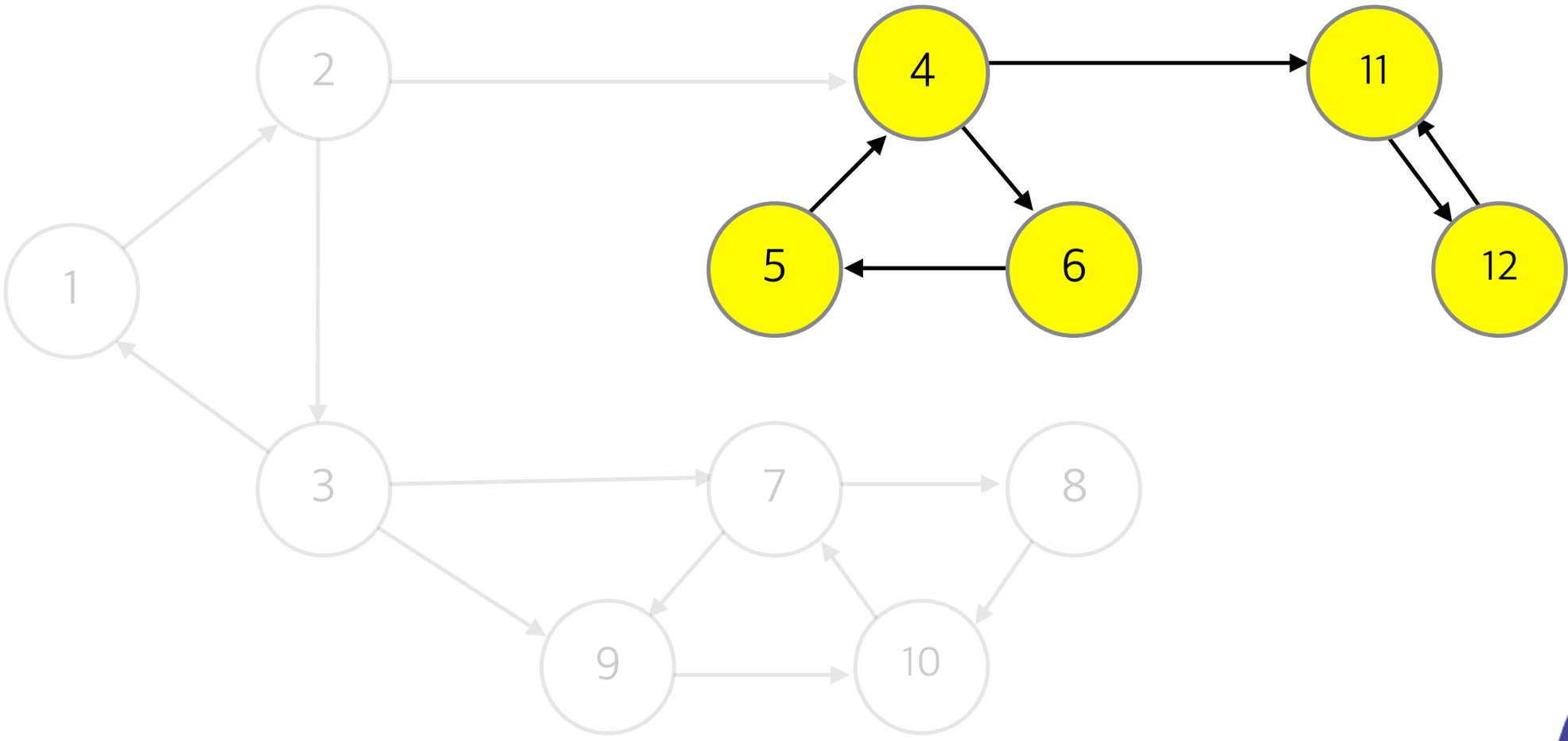
- Idea 1. DFS를 하면 부모-자식 관계(?)가 파악된다

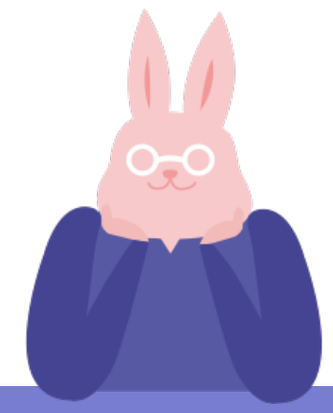




Strongly Connected Component

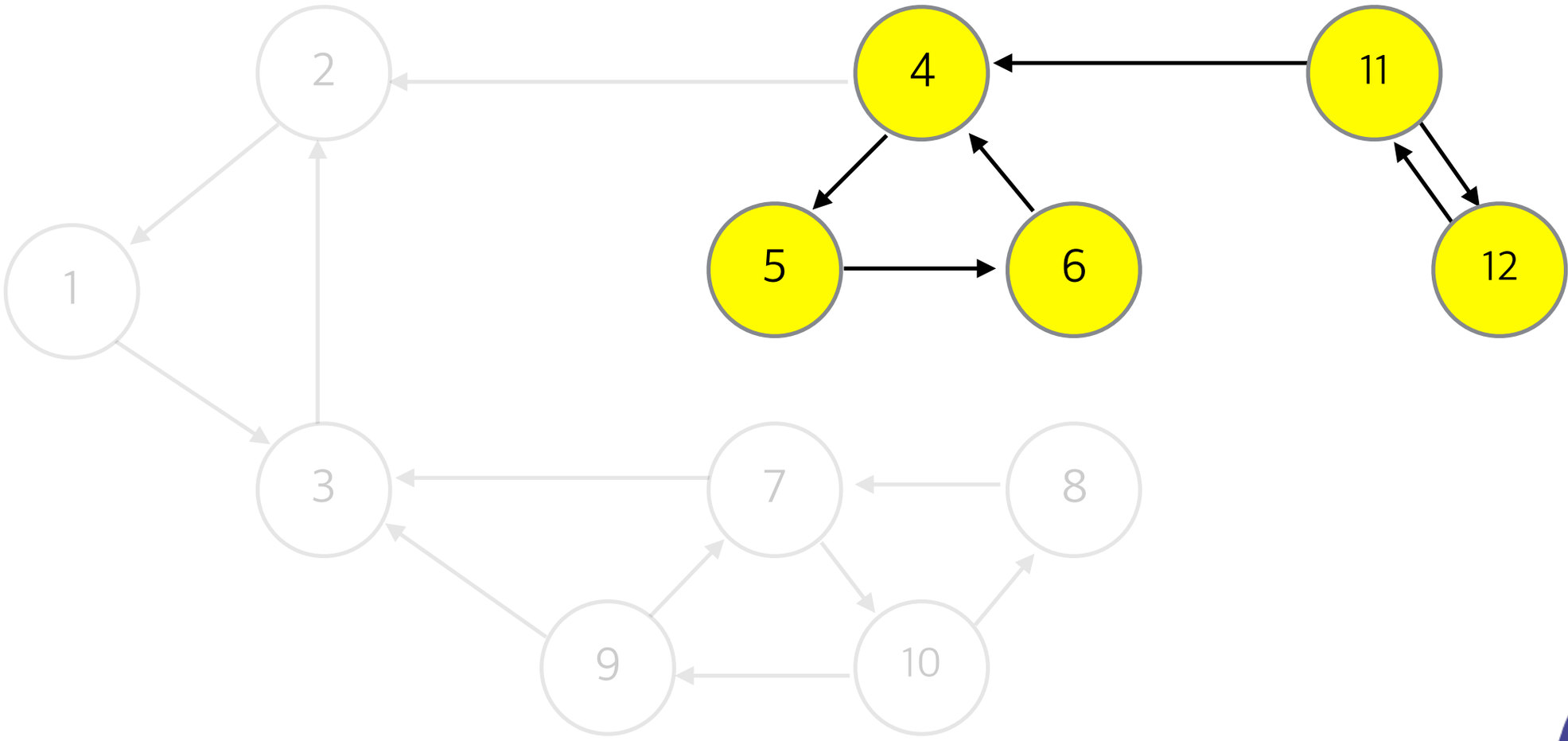
- Idea 1. DFS를 하면 부모-자식 관계(?)가 파악된다

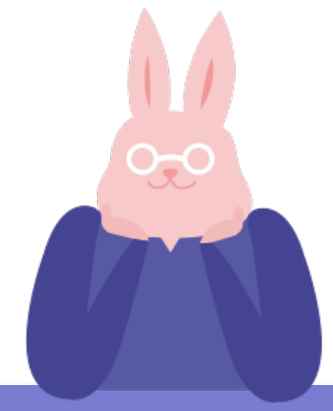




Strongly Connected Component

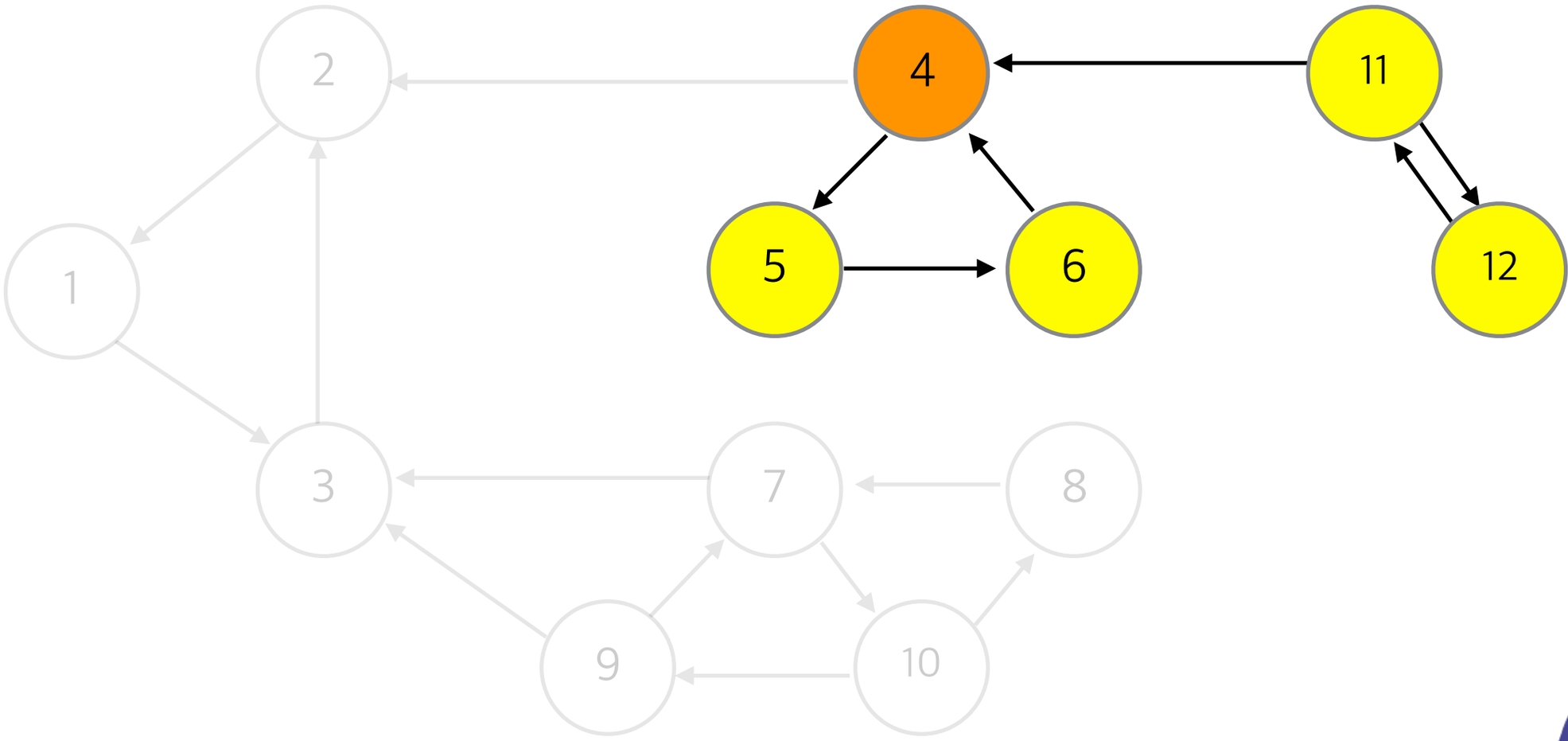
- Idea 2. 그래프의 **간선 방향을 모두 뒤집는다.**

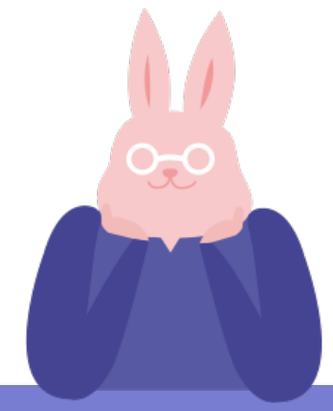




Strongly Connected Component

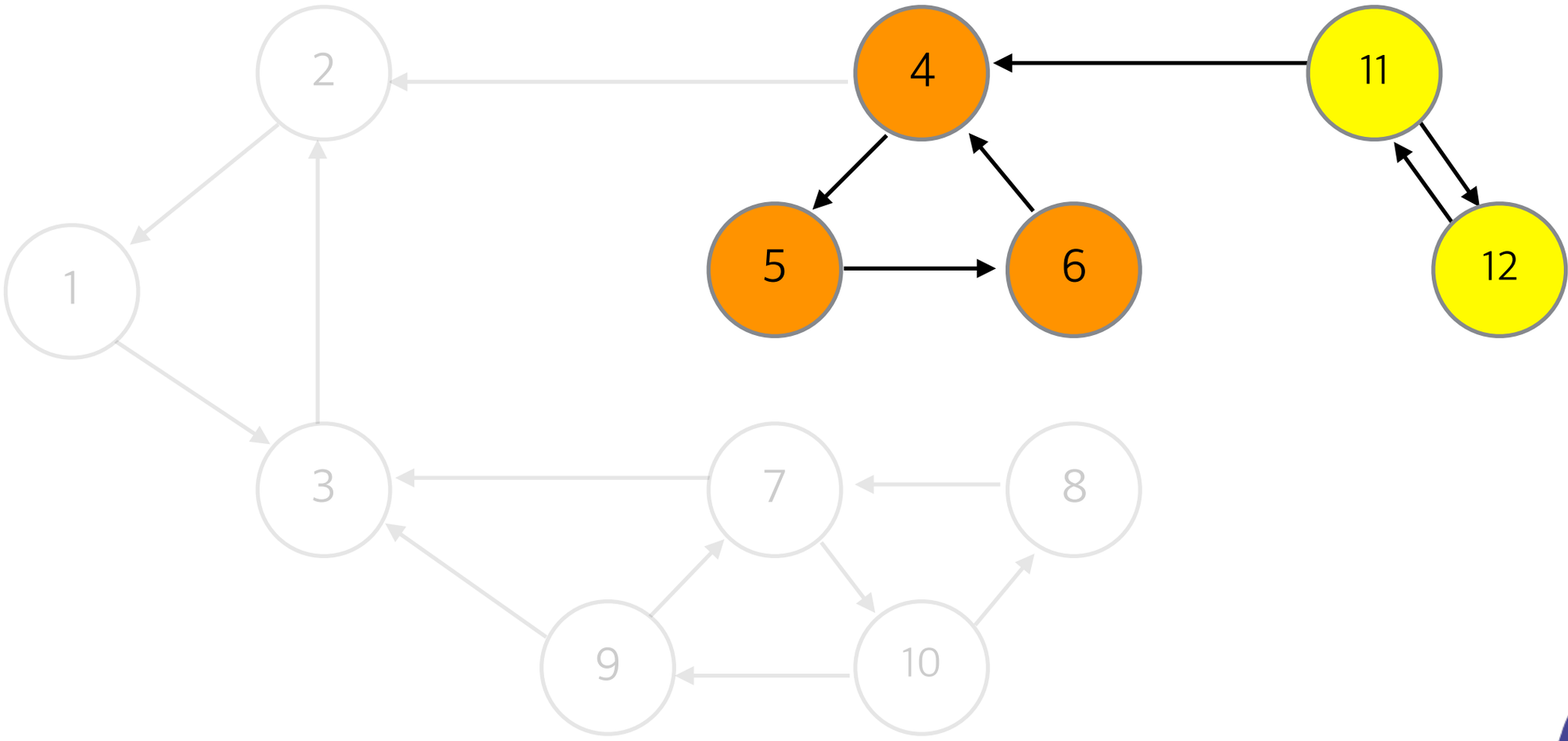
- Idea 2. 그래프의 **간선 방향을 모두 뒤집는다.**

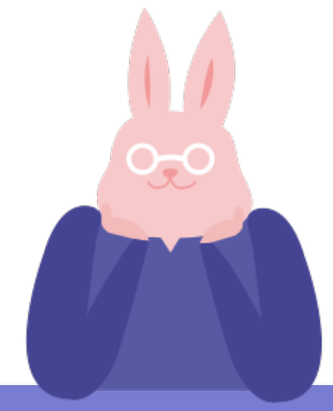




Strongly Connected Component

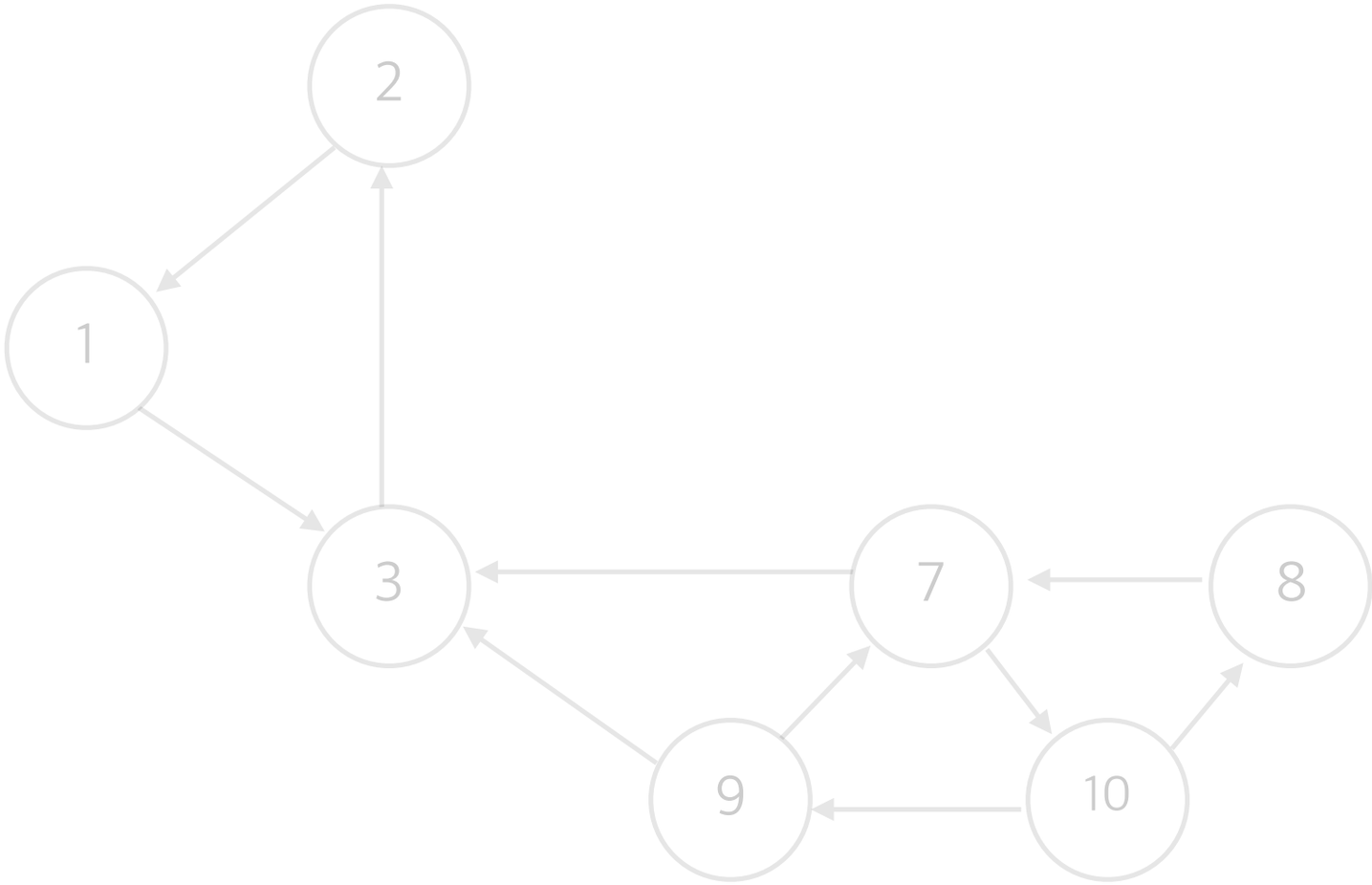
- Idea 2. 그래프의 **간선 방향을 모두 뒤집는다.**

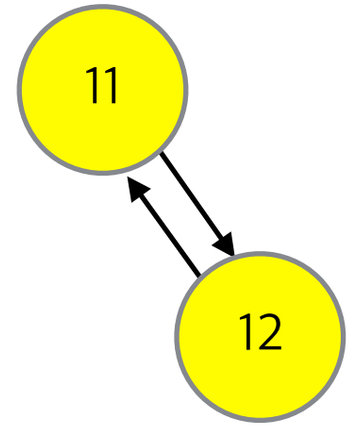


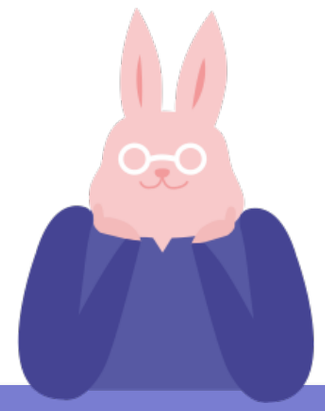


Strongly Connected Component

- Idea 2. 그래프의 **간선 방향을 모두 뒤집는다.**

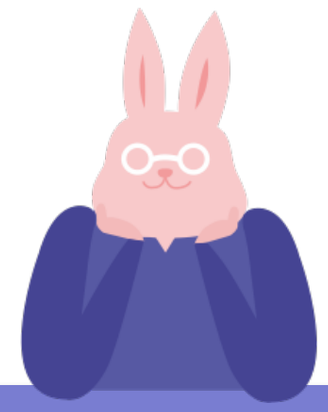






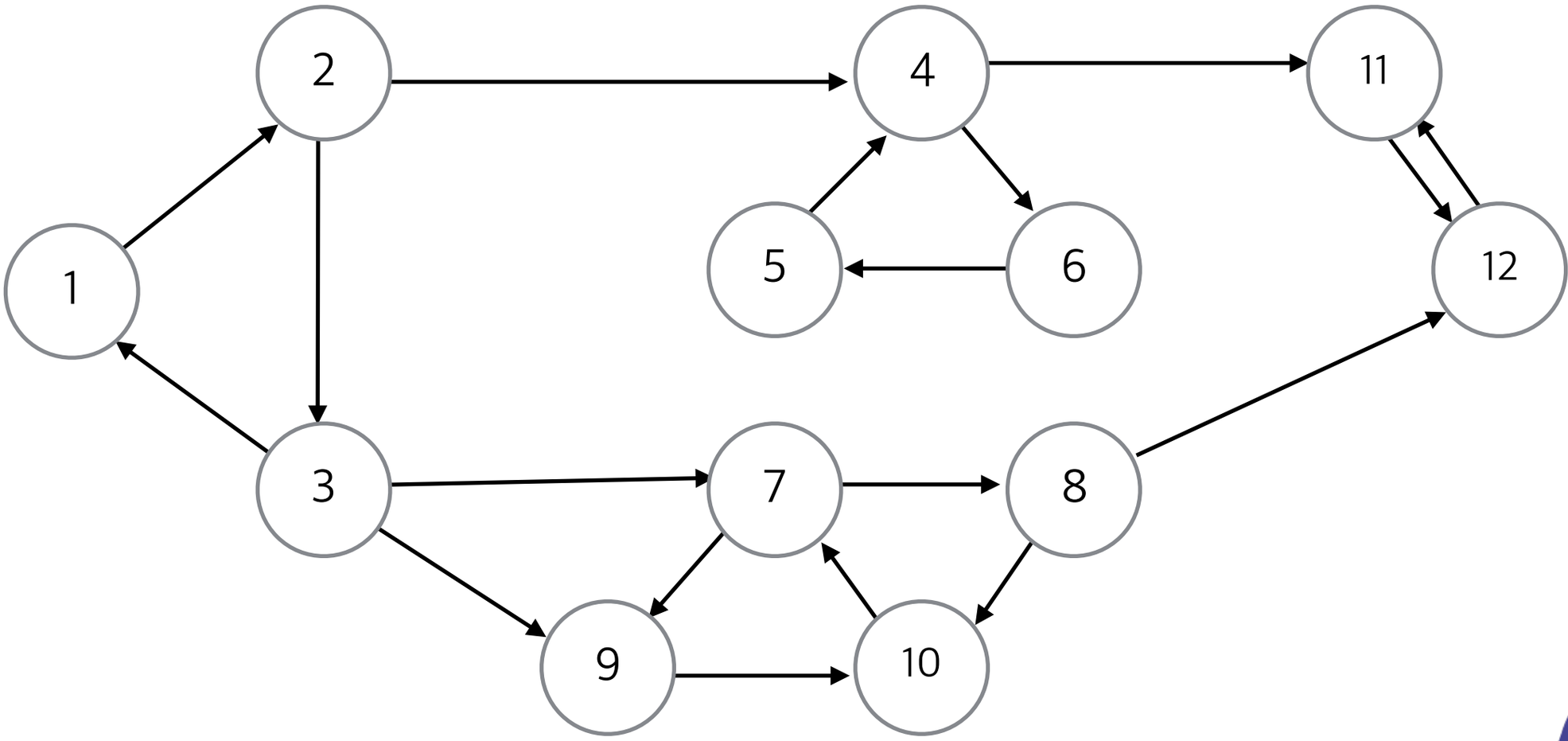
Strongly Connected Component

- 현재까지의 요약
  1. Node 하나를 잡는다. 이를 v라 하자.
  2. Traverse하여 방문하는 Node에 색칠을 한다
  3. Graph를 뒤집는다
  4. 색칠한 Node들 중에서 v가 도달할 수 있는 vertex들을 group으로 묶는다.
  5. 이 group을 graph로부터 없앤다.
  6. 위의 과정을 반복한다.



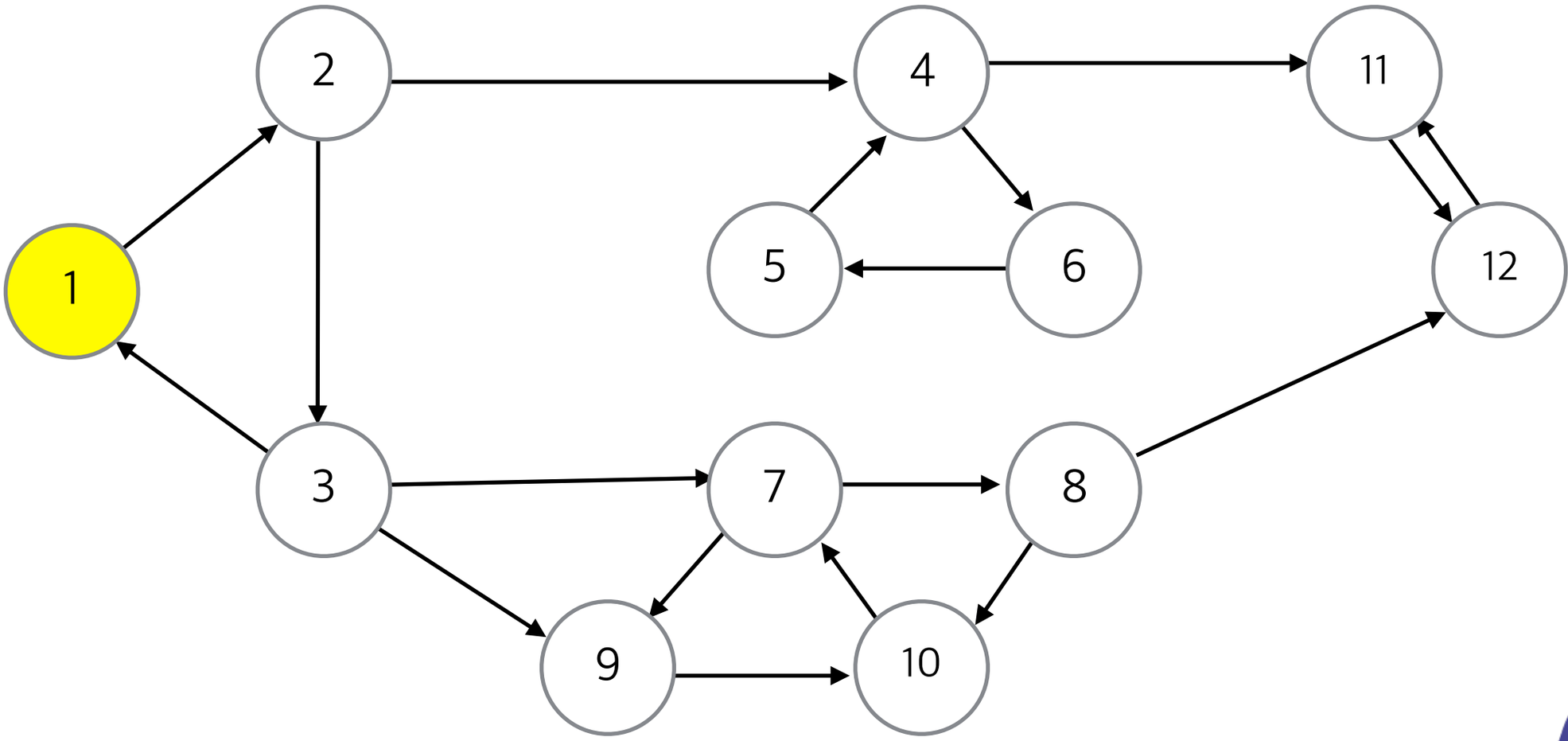
Strongly Connected Component

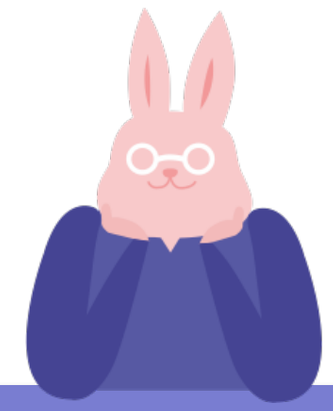
- Question. 한 번에 여러 group은 구할 수 없는가 ?



Strongly Connected Component

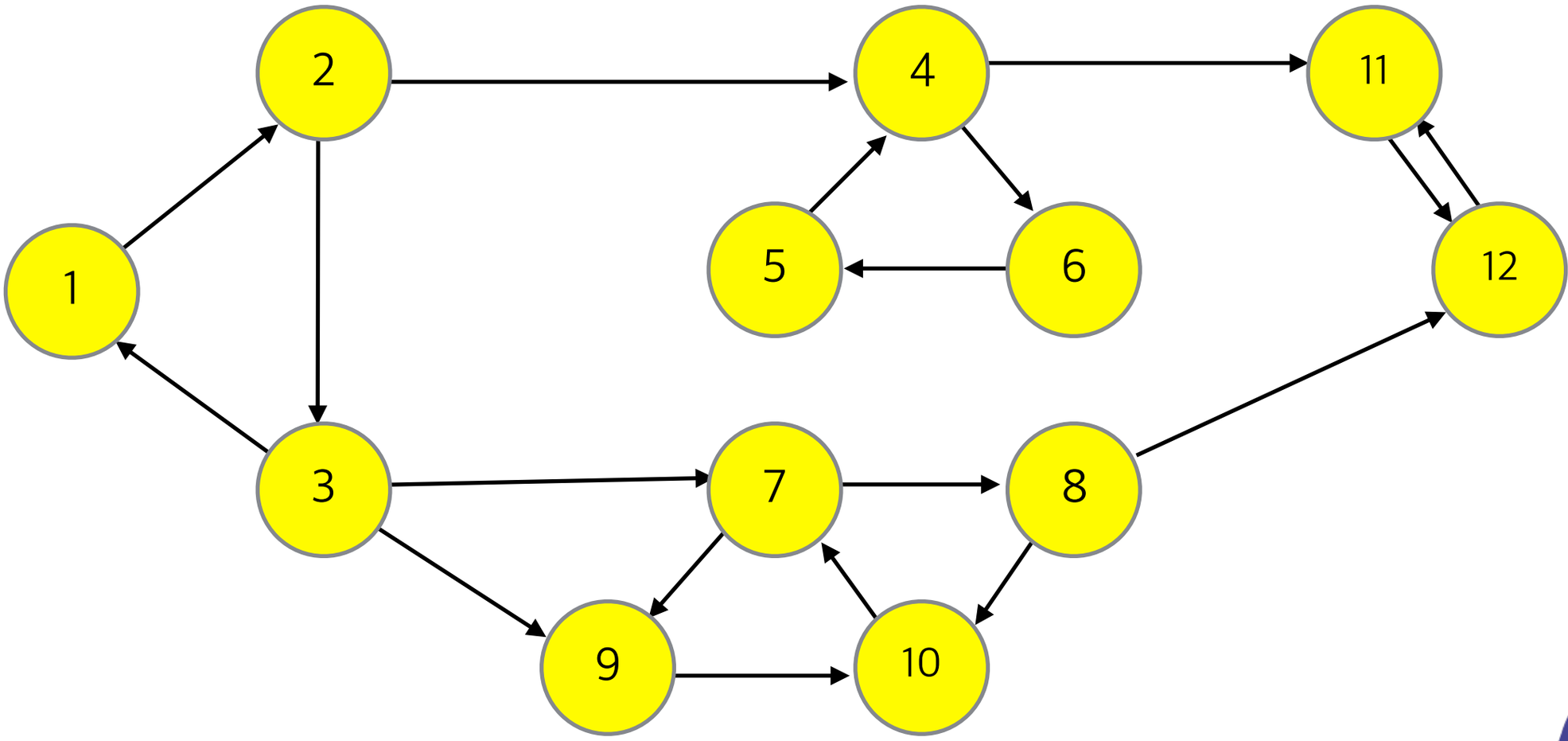
- Question. 한 번에 여러 group은 구할 수 없는가 ?

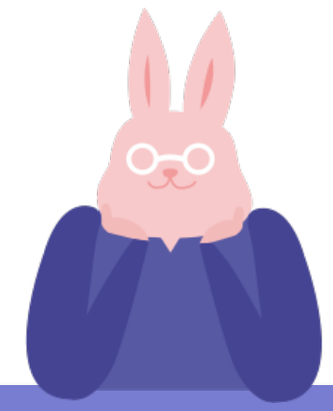




Strongly Connected Component

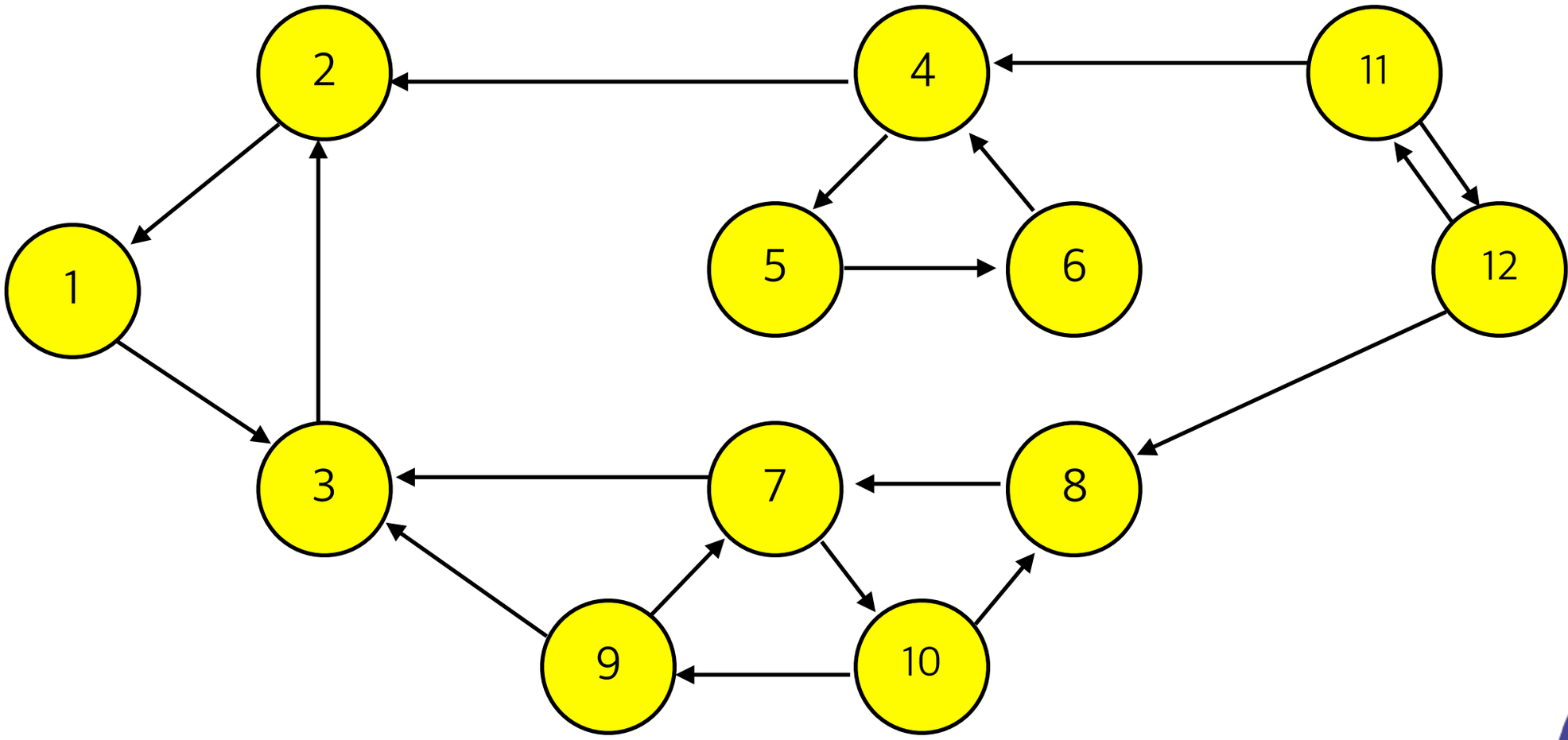
- Question. 한 번에 여러 group은 구할 수 없는가 ?

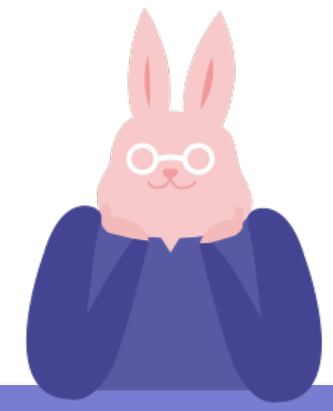




Strongly Connected Component

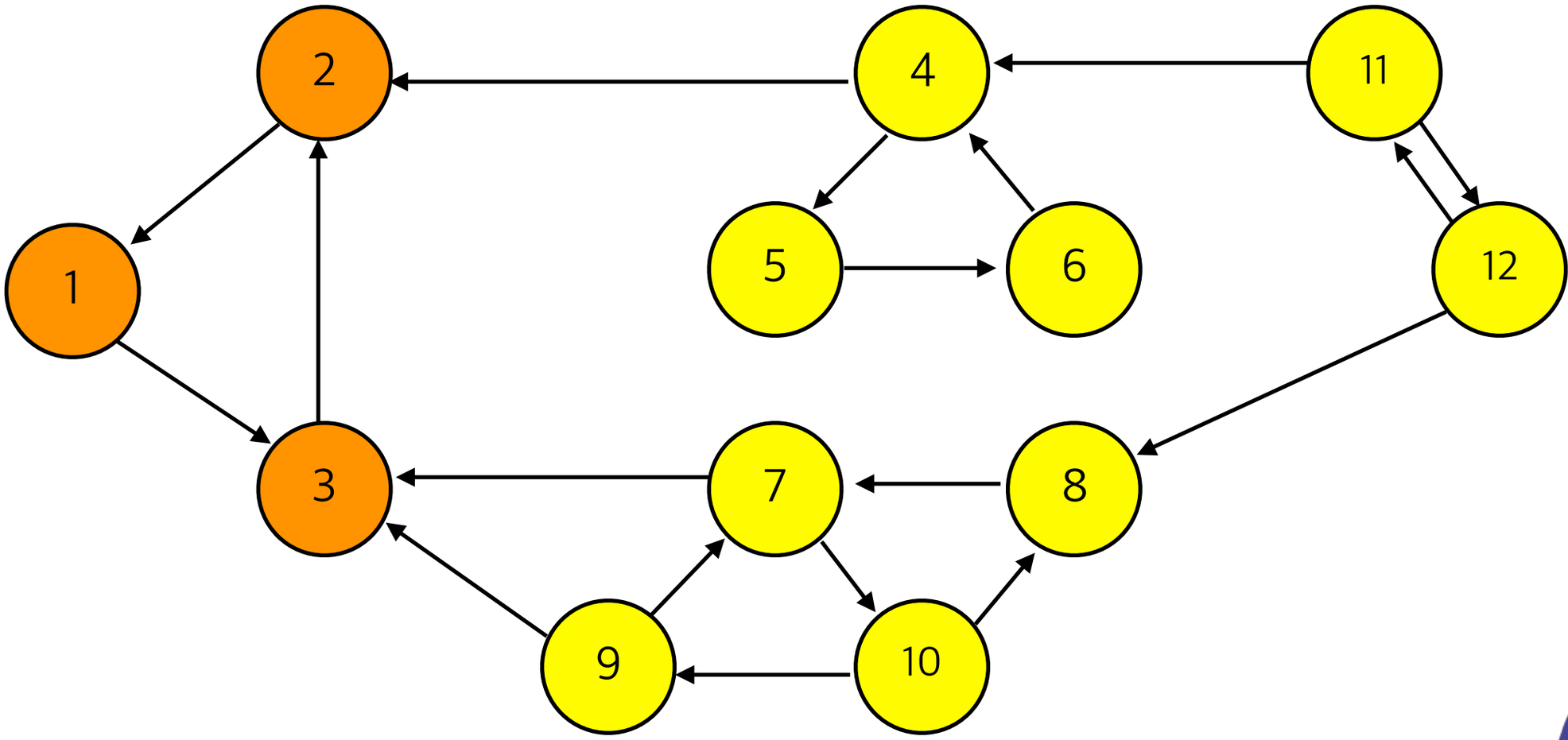
- Question. 한 번에 여러 group은 구할 수 없는가 ?

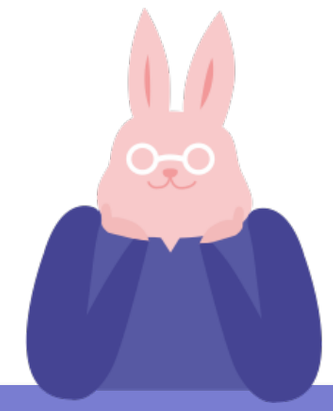




Strongly Connected Component

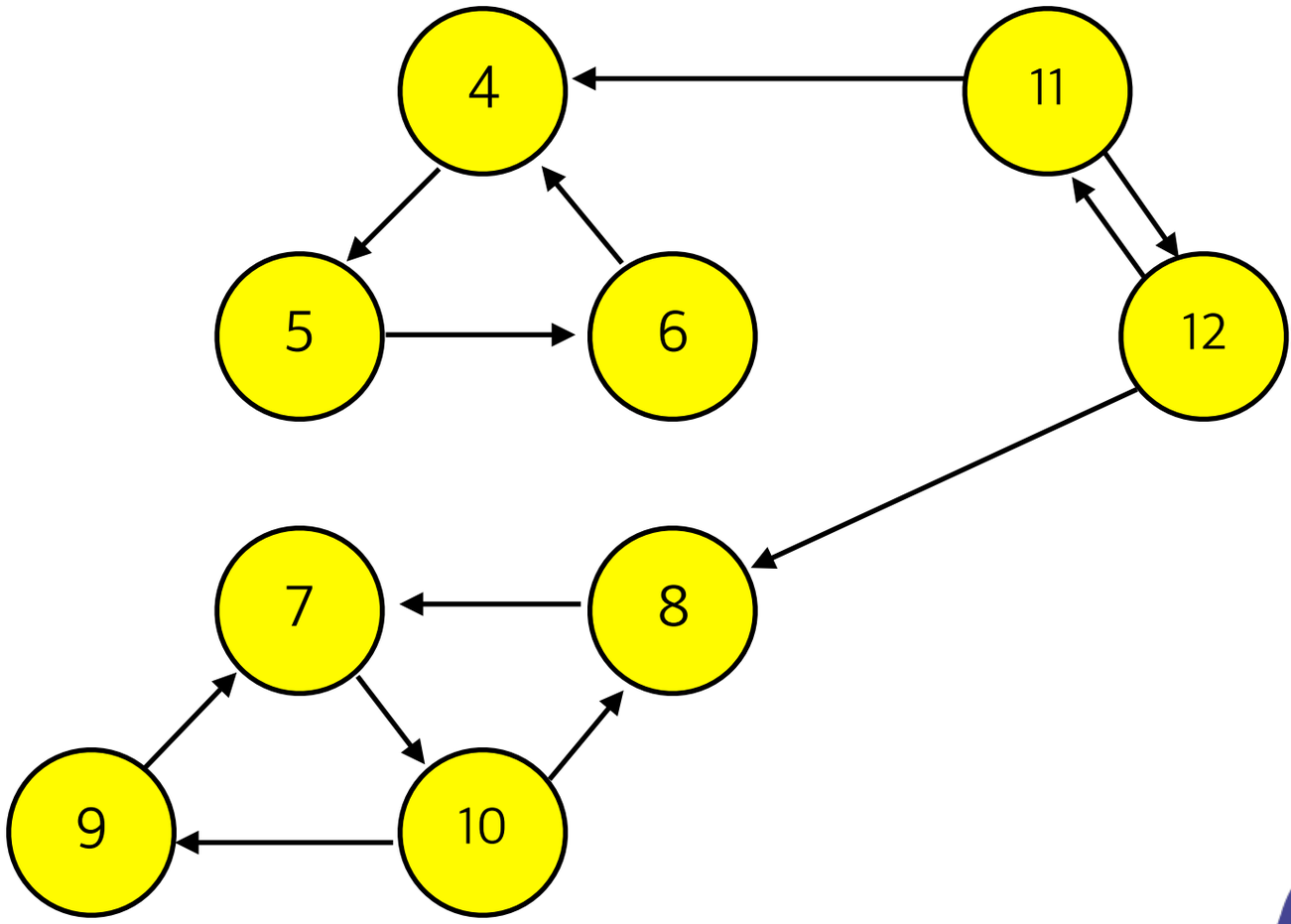
- Question. 한 번에 여러 group은 구할 수 없는가 ?

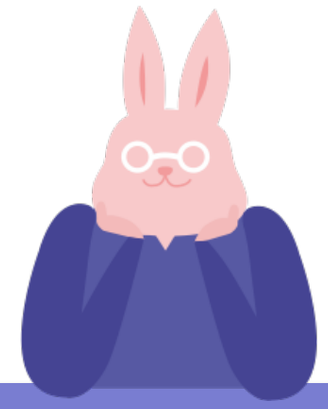




Strongly Connected Component

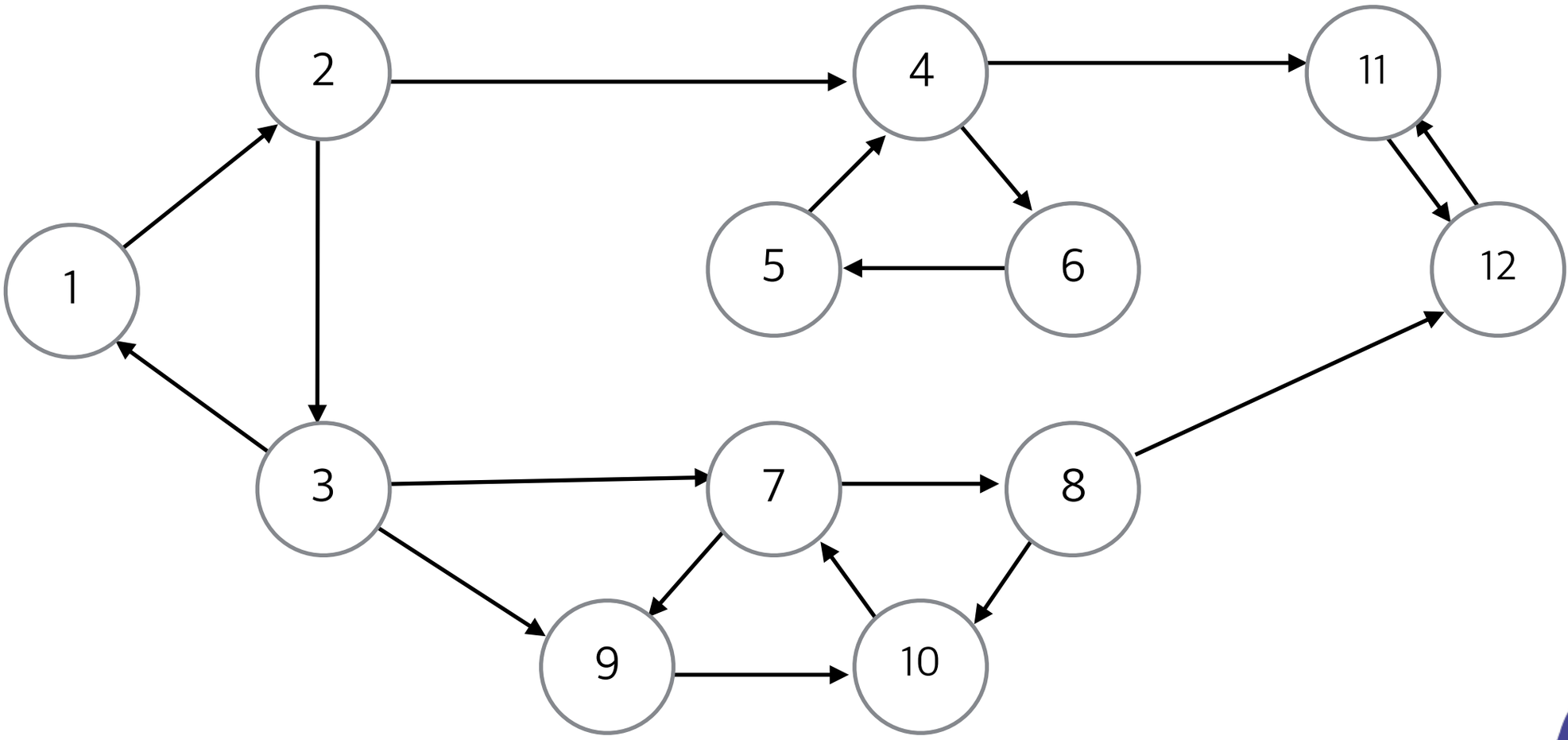
- Question. 한 번에 여러 group은 구할 수 없는가 ?

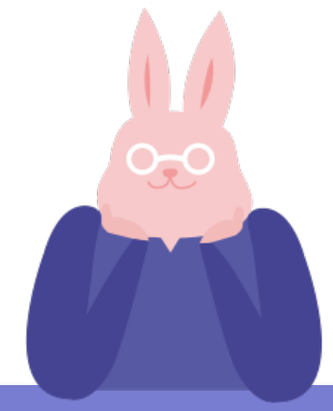




Strongly Connected Component

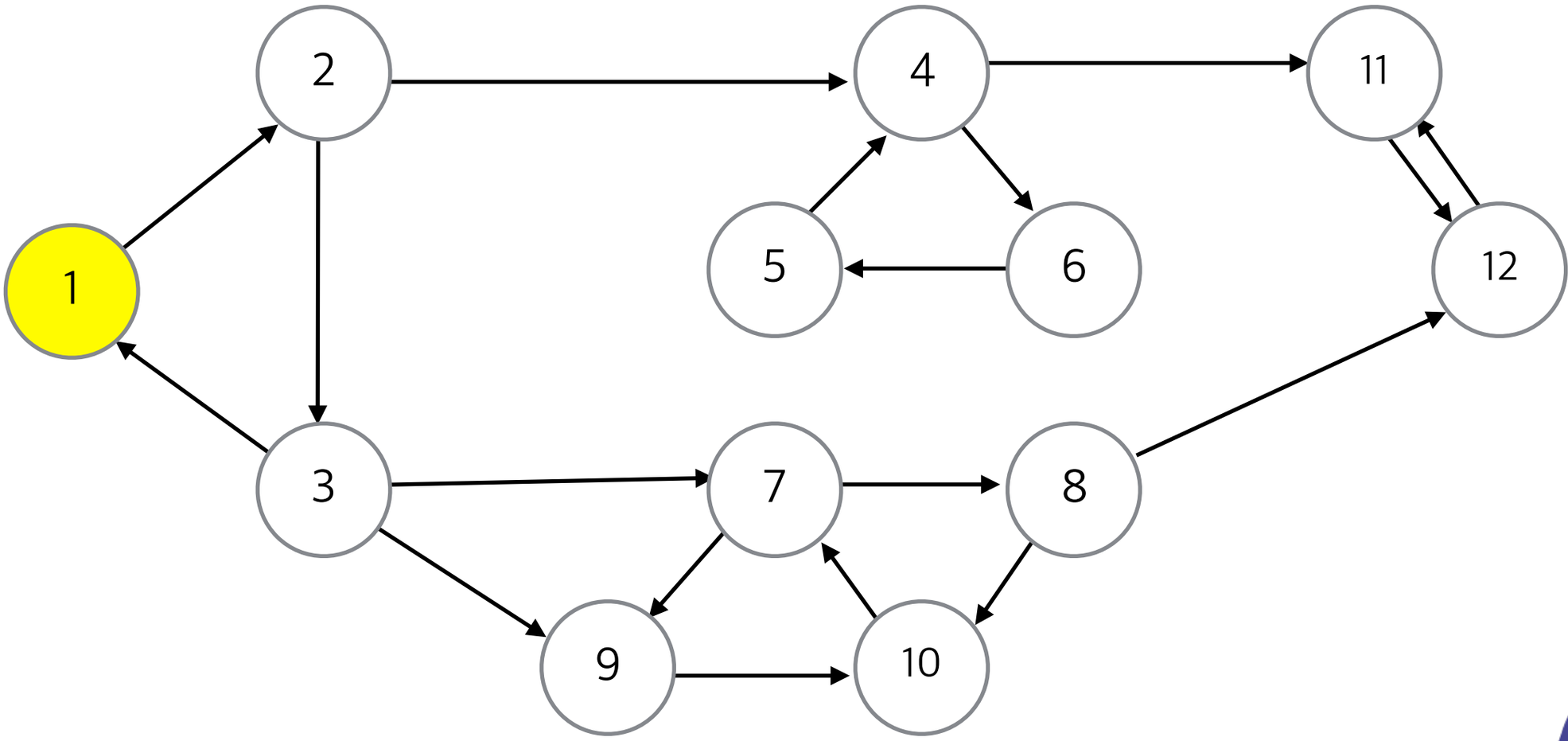
- vertex를 방문하면서 **빠져나오는 시간**을 적어놓자

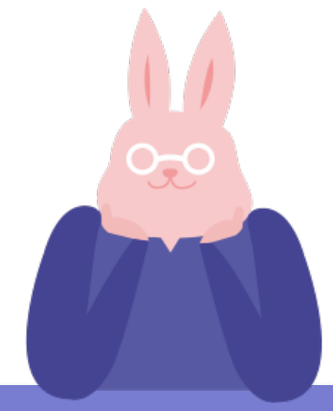




Strongly Connected Component

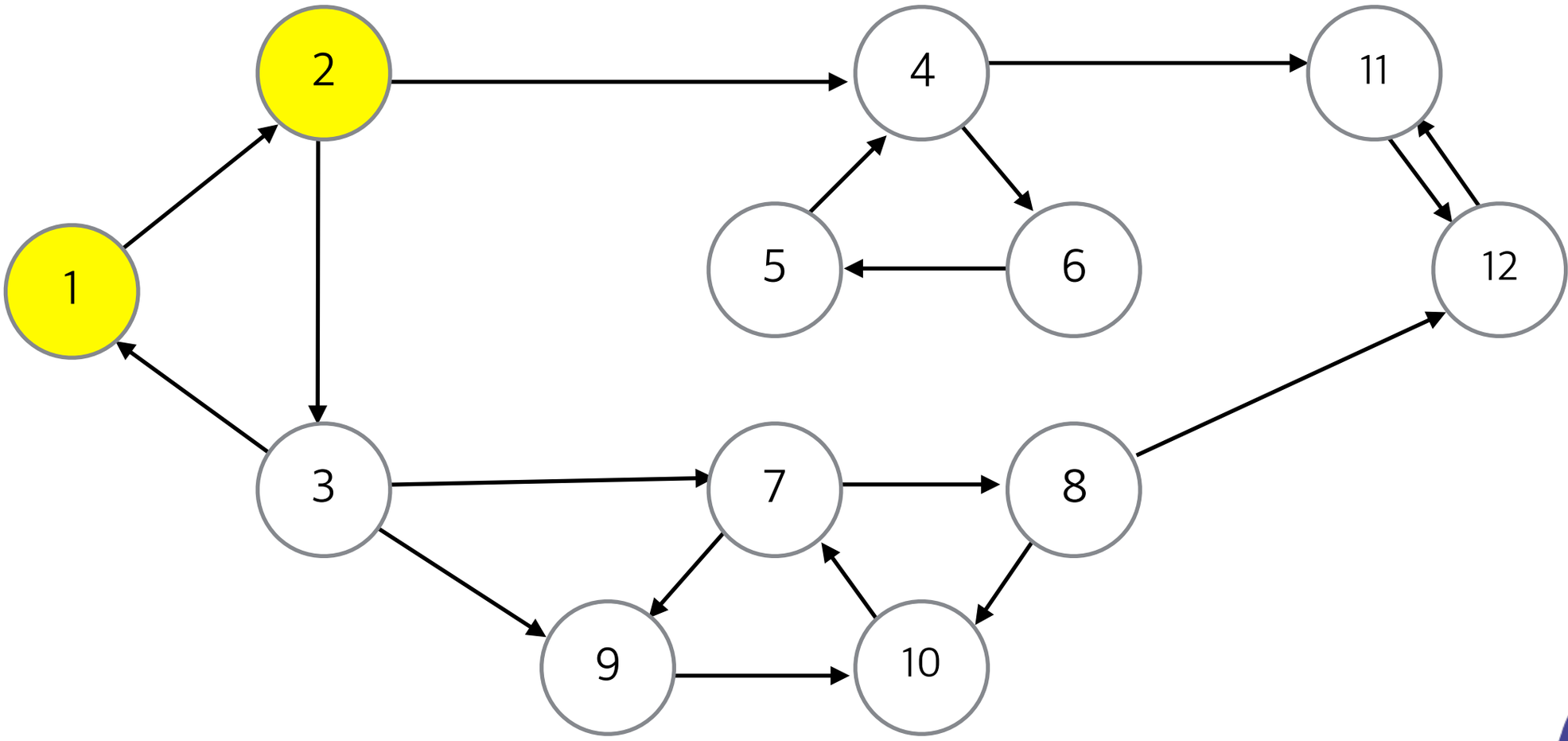
- vertex를 방문하면서 **빠져나오는 시간**을 적어놓자

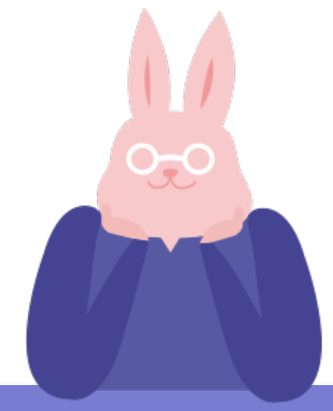




Strongly Connected Component

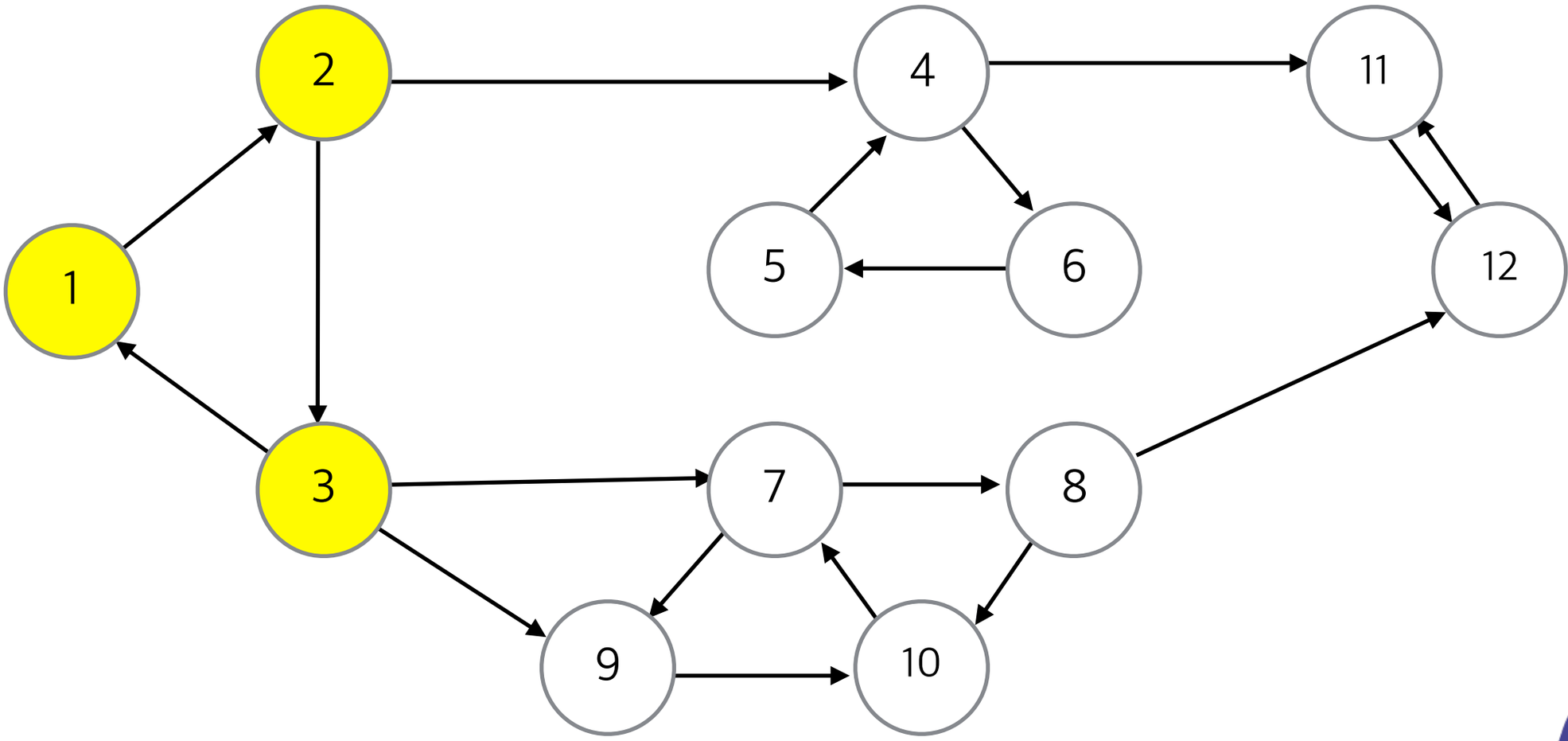
- vertex를 방문하면서 **빠져나오는 시간**을 적어놓자

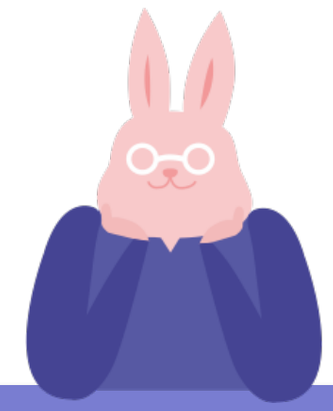




Strongly Connected Component

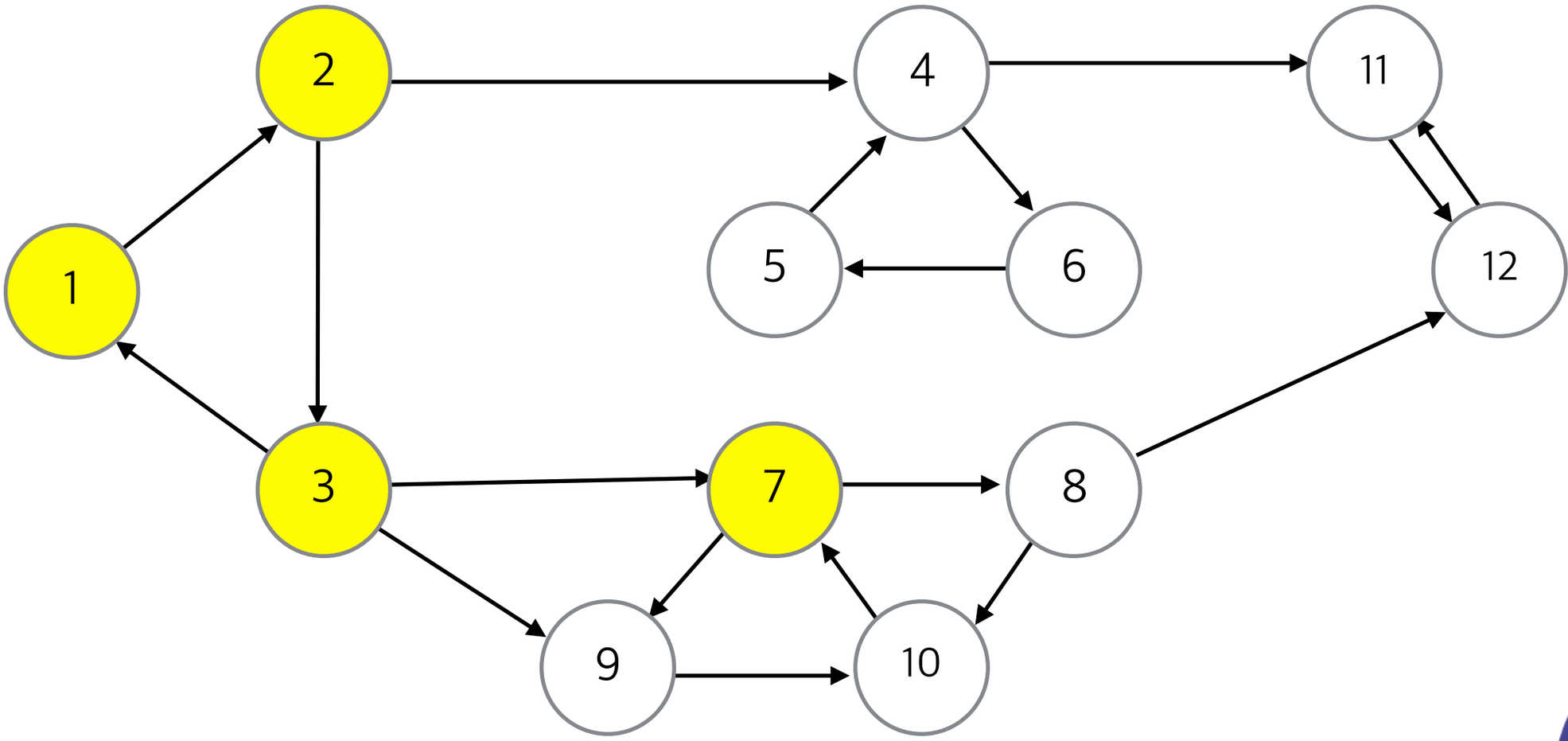
- vertex를 방문하면서 **빠져나오는 시간**을 적어놓자

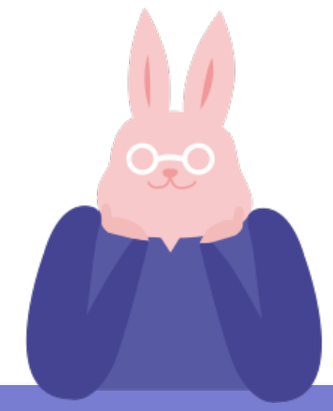




Strongly Connected Component

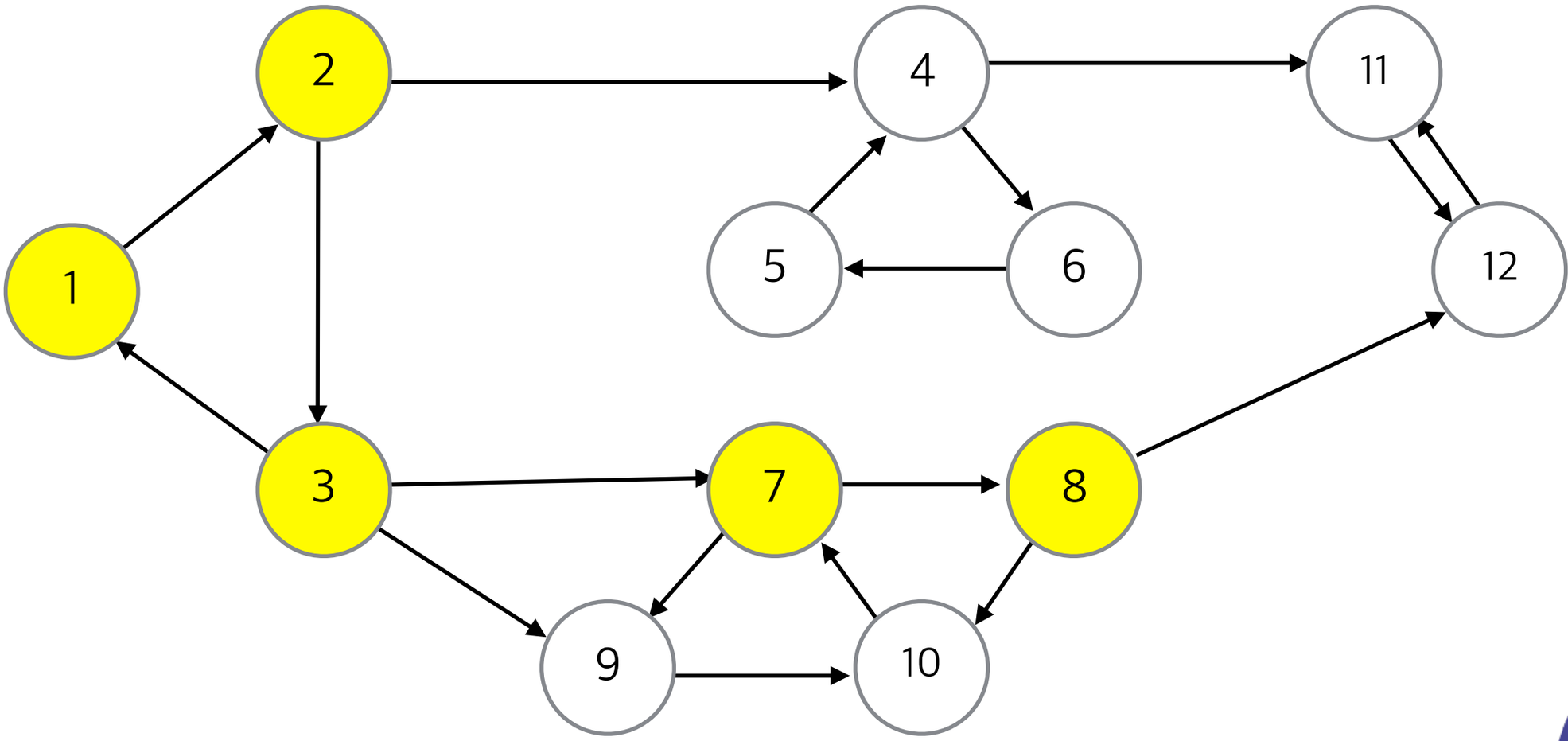
- vertex를 방문하면서 **빠져나오는 시간**을 적어놓자

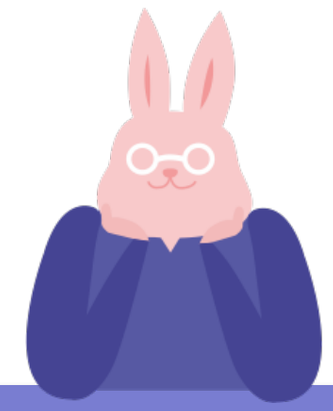




Strongly Connected Component

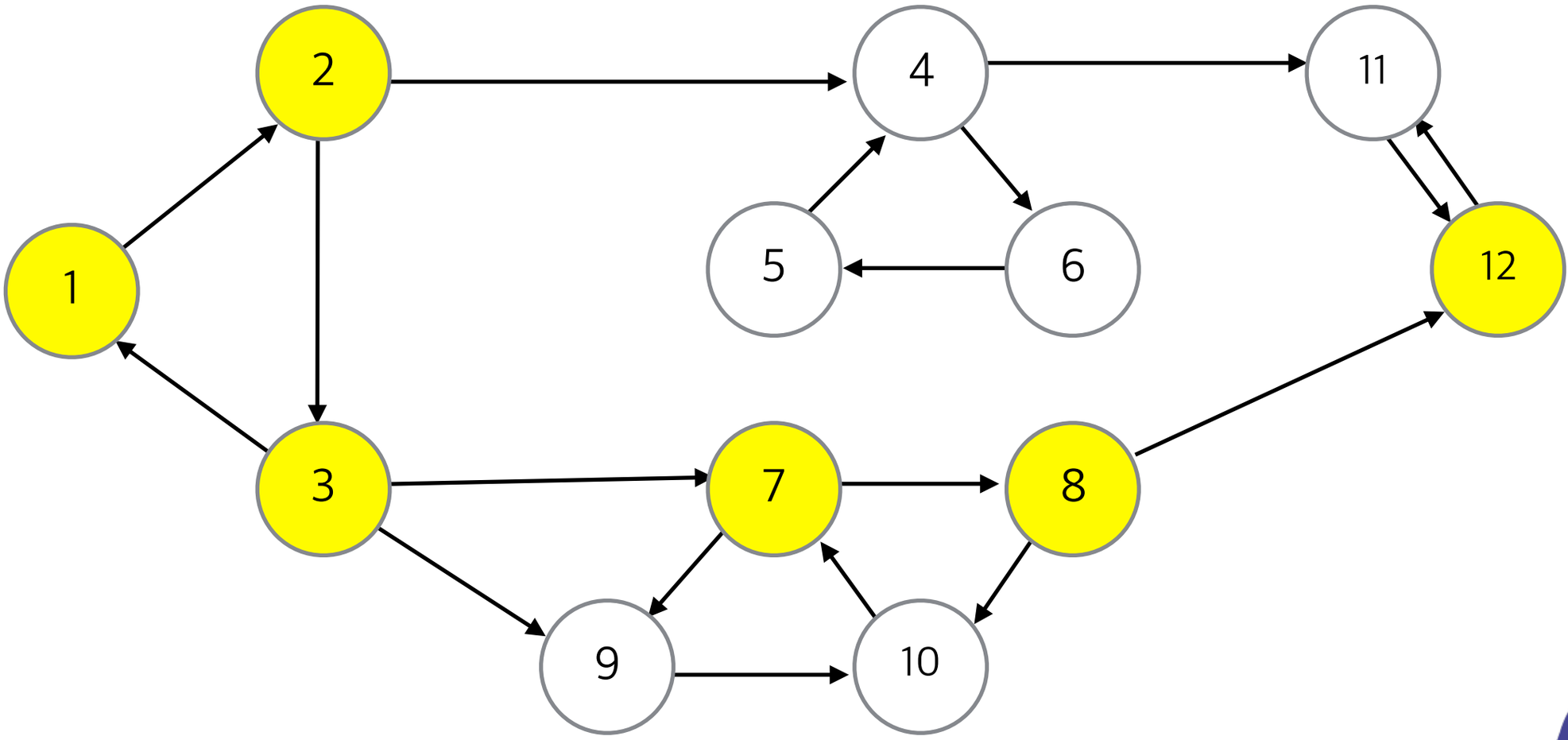
- vertex를 방문하면서 **빠져나오는 시간**을 적어놓자

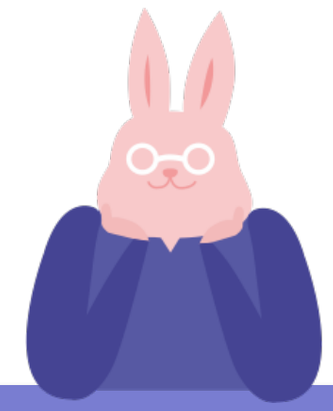




Strongly Connected Component

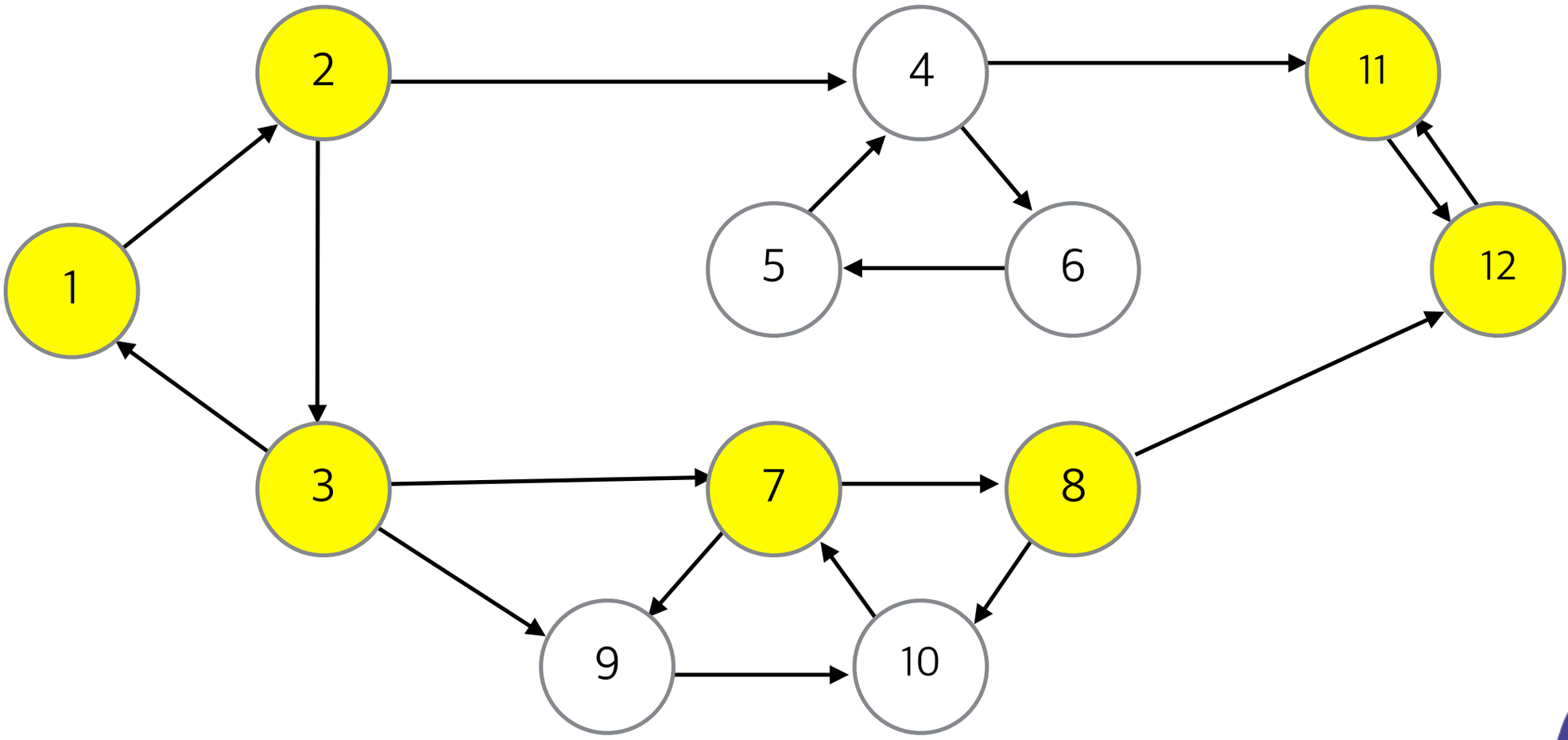
- vertex를 방문하면서 **빠져나오는 시간**을 적어놓자

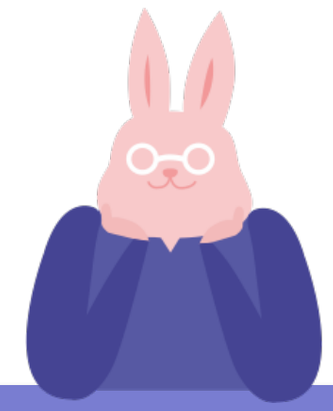




Strongly Connected Component

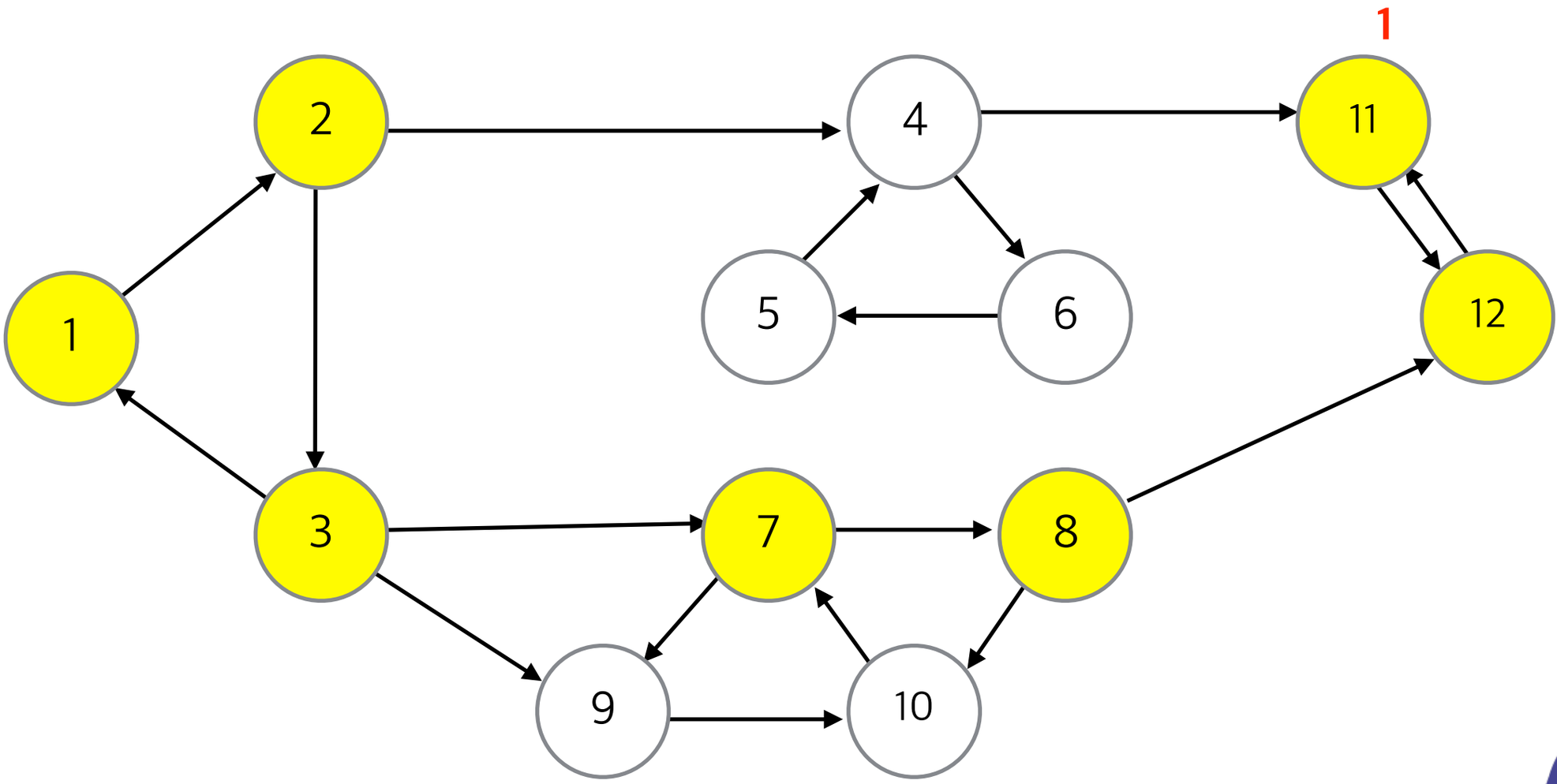
- vertex를 방문하면서 **빠져나오는 시간**을 적어놓자





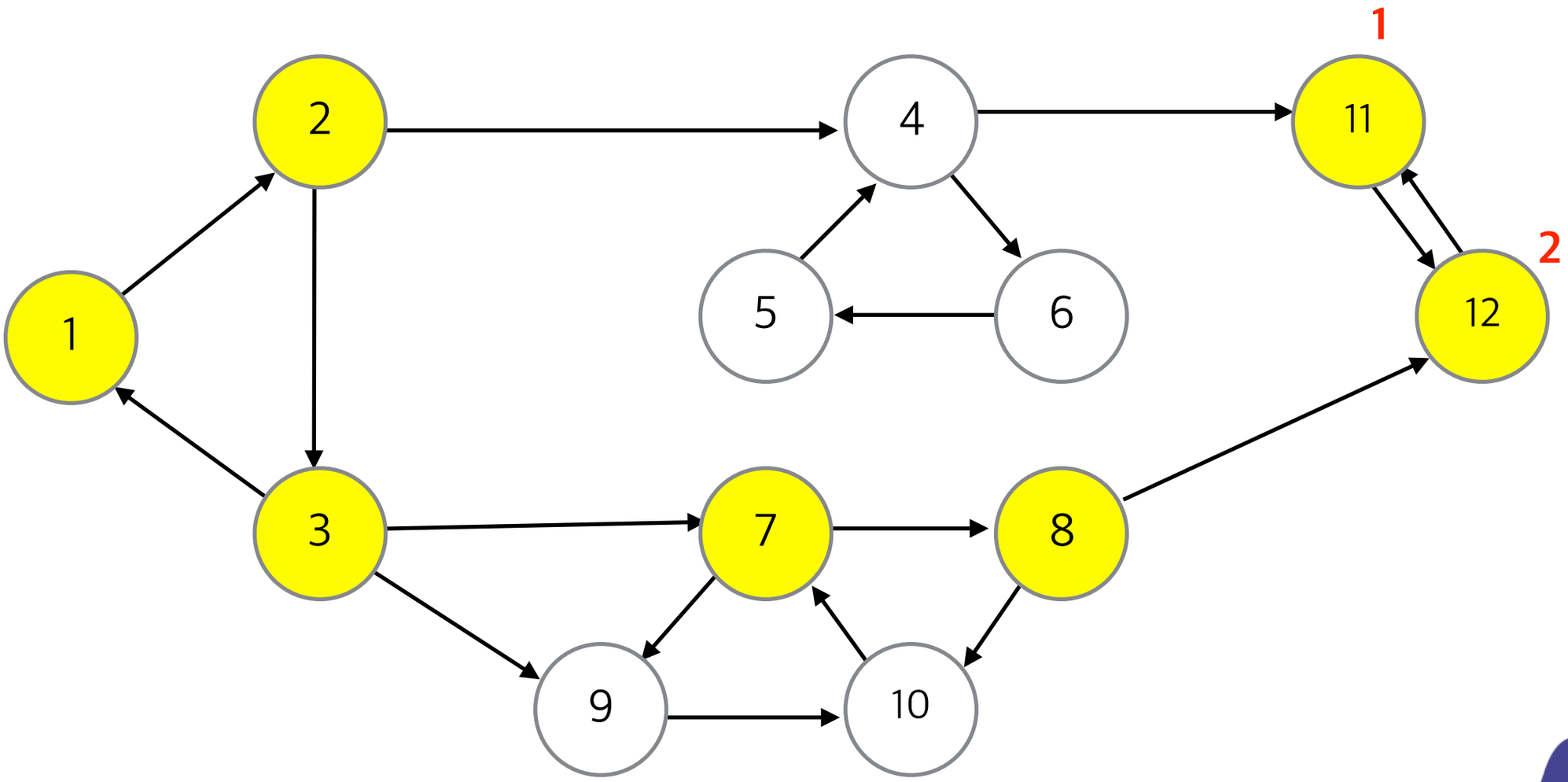
Strongly Connected Component

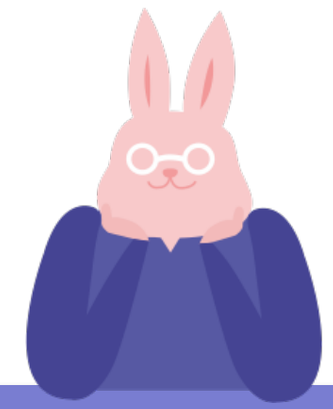
- vertex를 방문하면서 **빠져나오는 시간**을 적어놓자



Strongly Connected Component

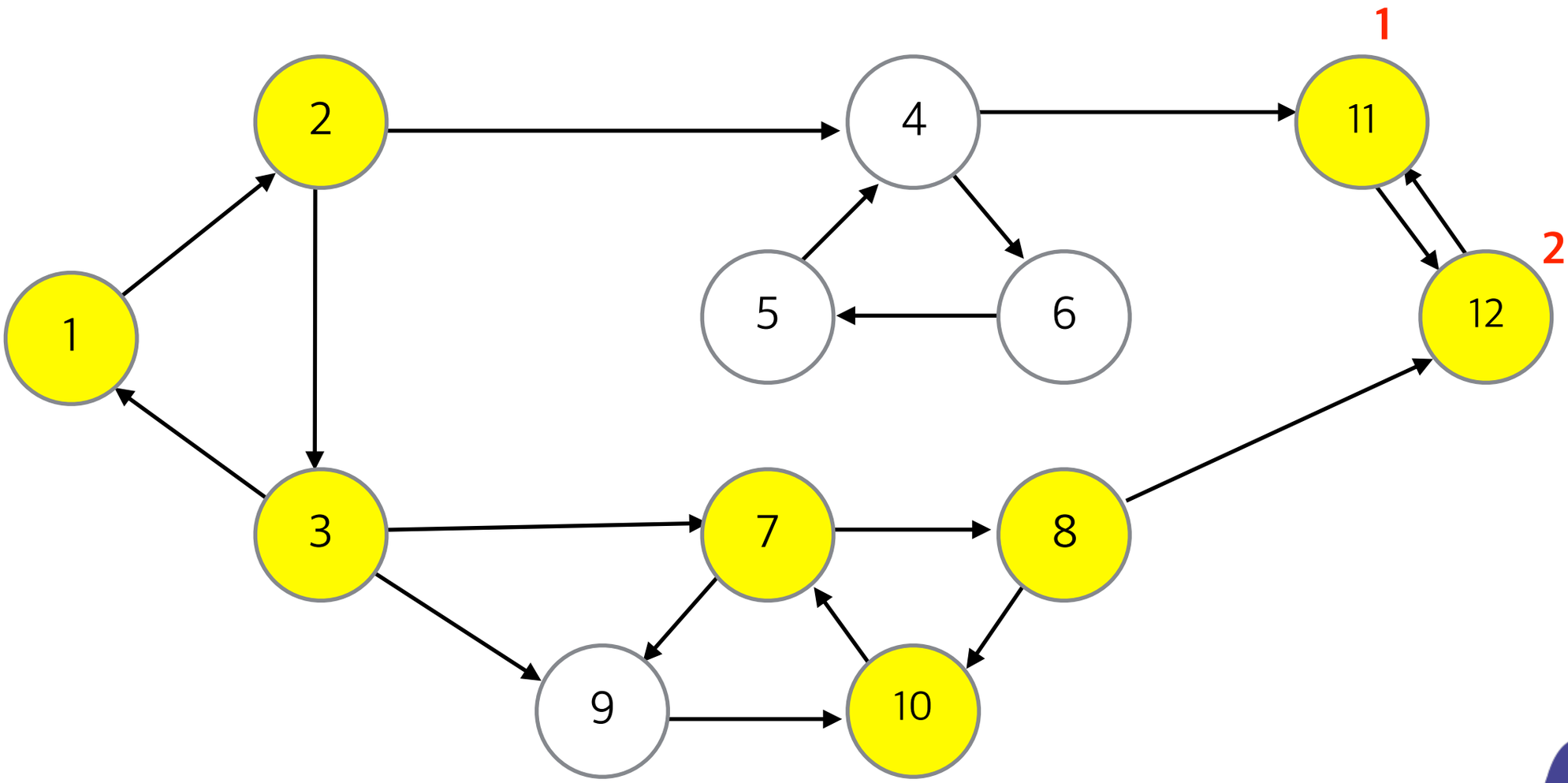
- vertex를 방문하면서 **빠져나오는 시간**을 적어놓자





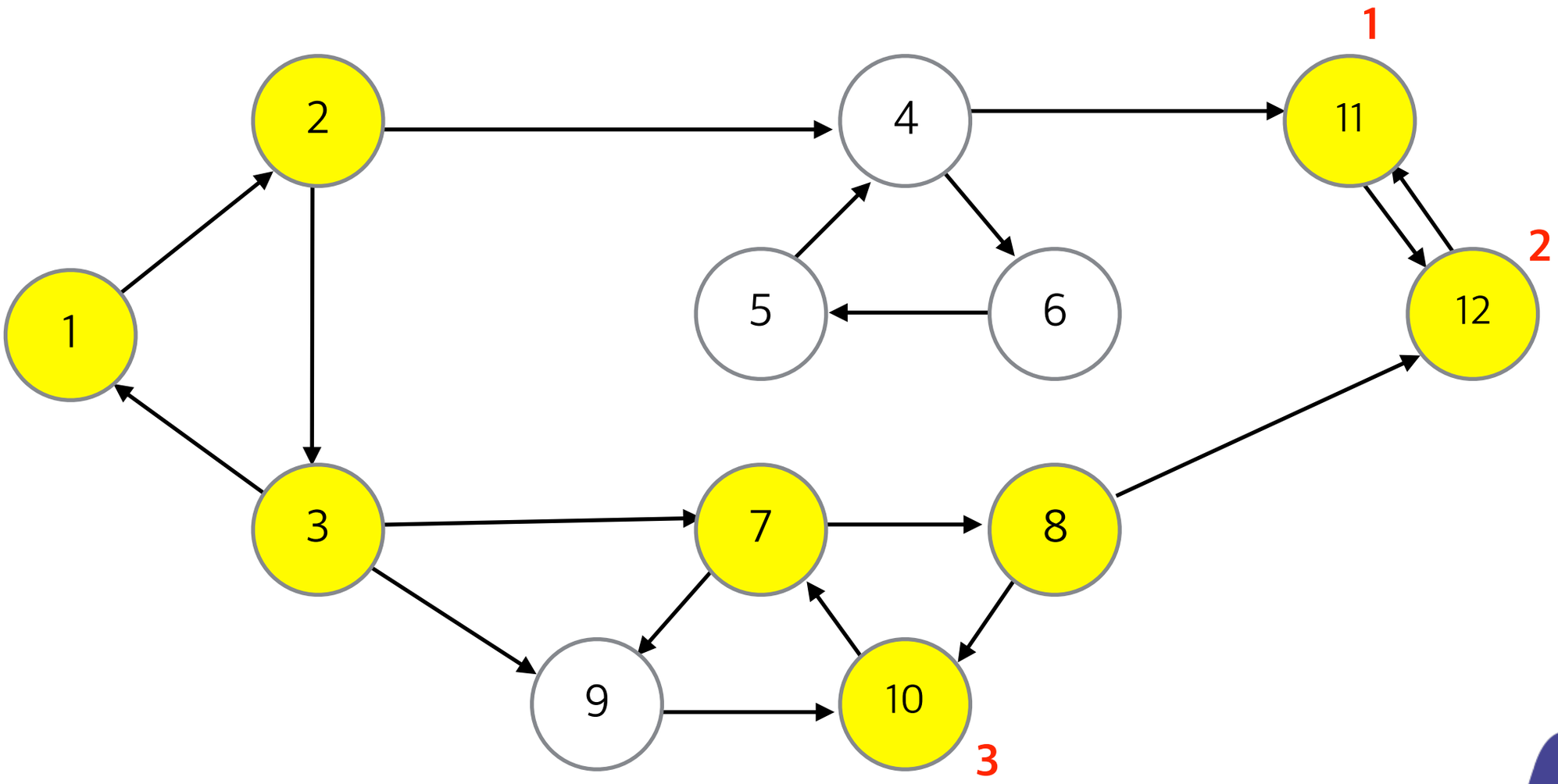
Strongly Connected Component

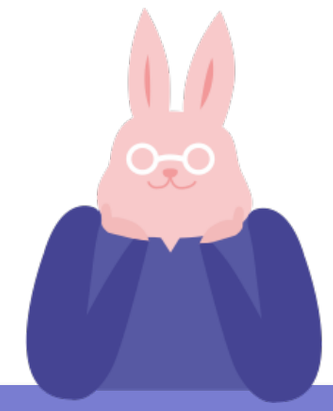
- vertex를 방문하면서 **빠져나오는 시간**을 적어놓자



Strongly Connected Component

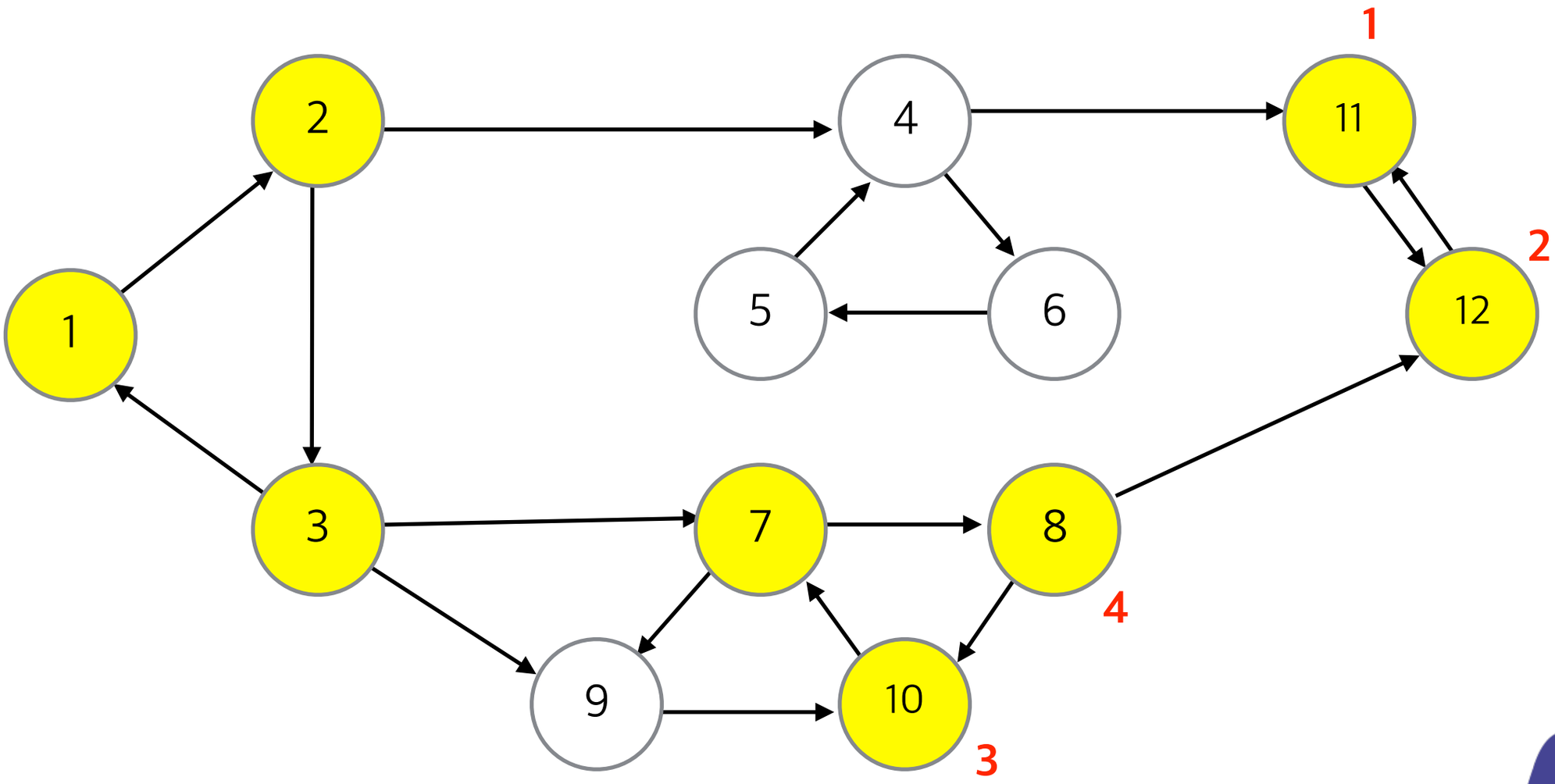
- vertex를 방문하면서 **빠져나오는 시간**을 적어놓자





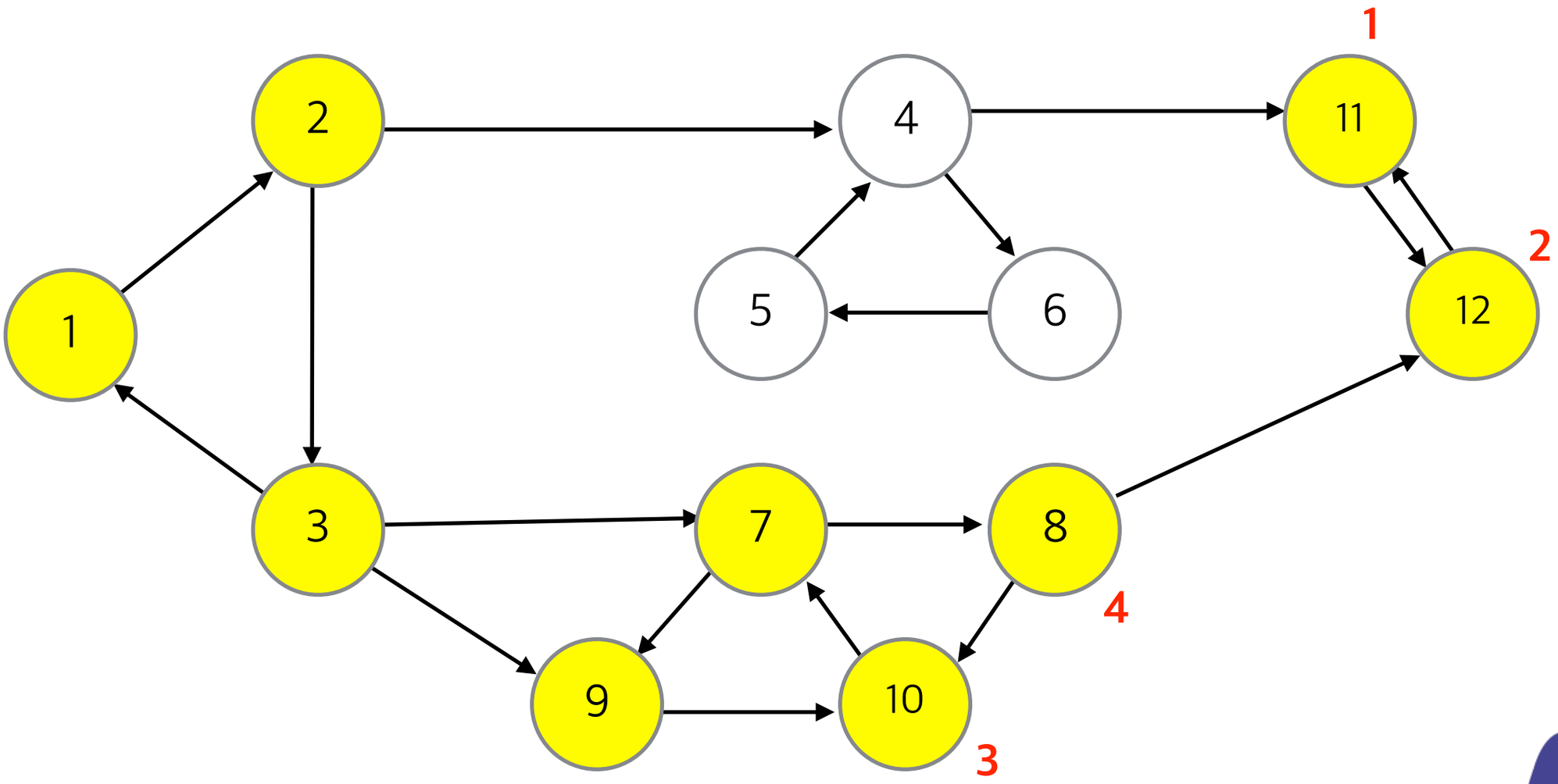
Strongly Connected Component

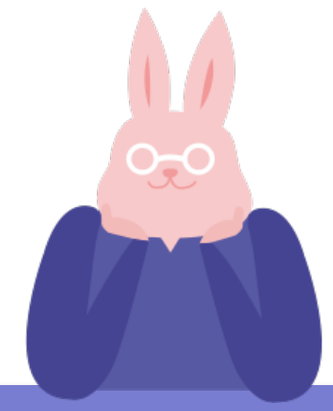
- vertex를 방문하면서 **빠져나오는 시간**을 적어놓자



Strongly Connected Component

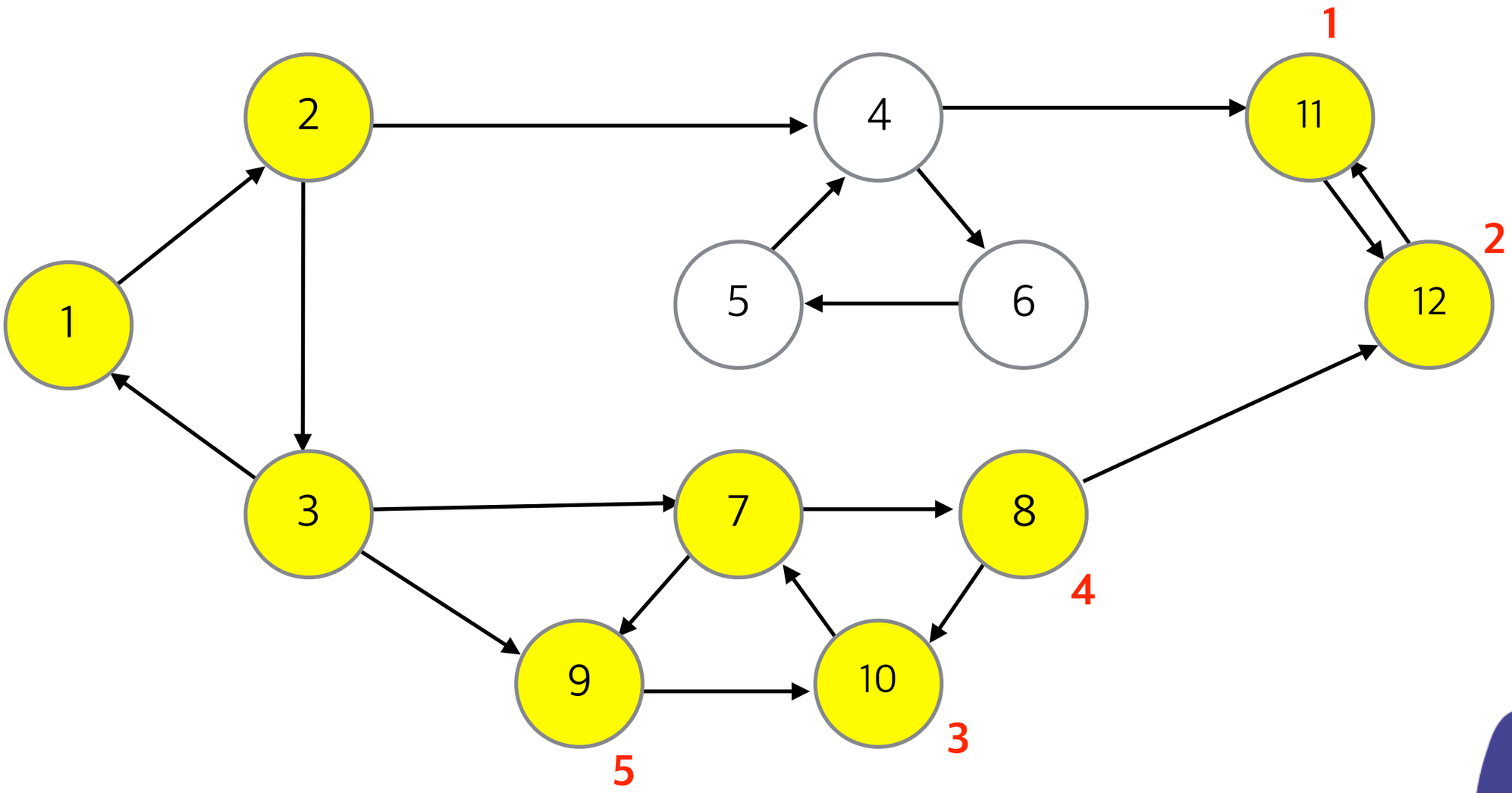
- vertex를 방문하면서 **빠져나오는 시간**을 적어놓자

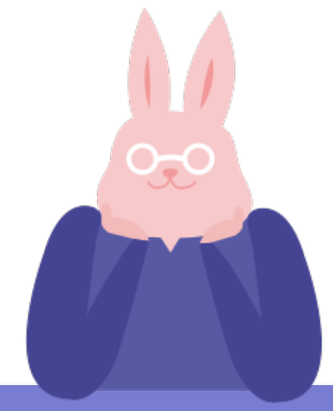




Strongly Connected Component

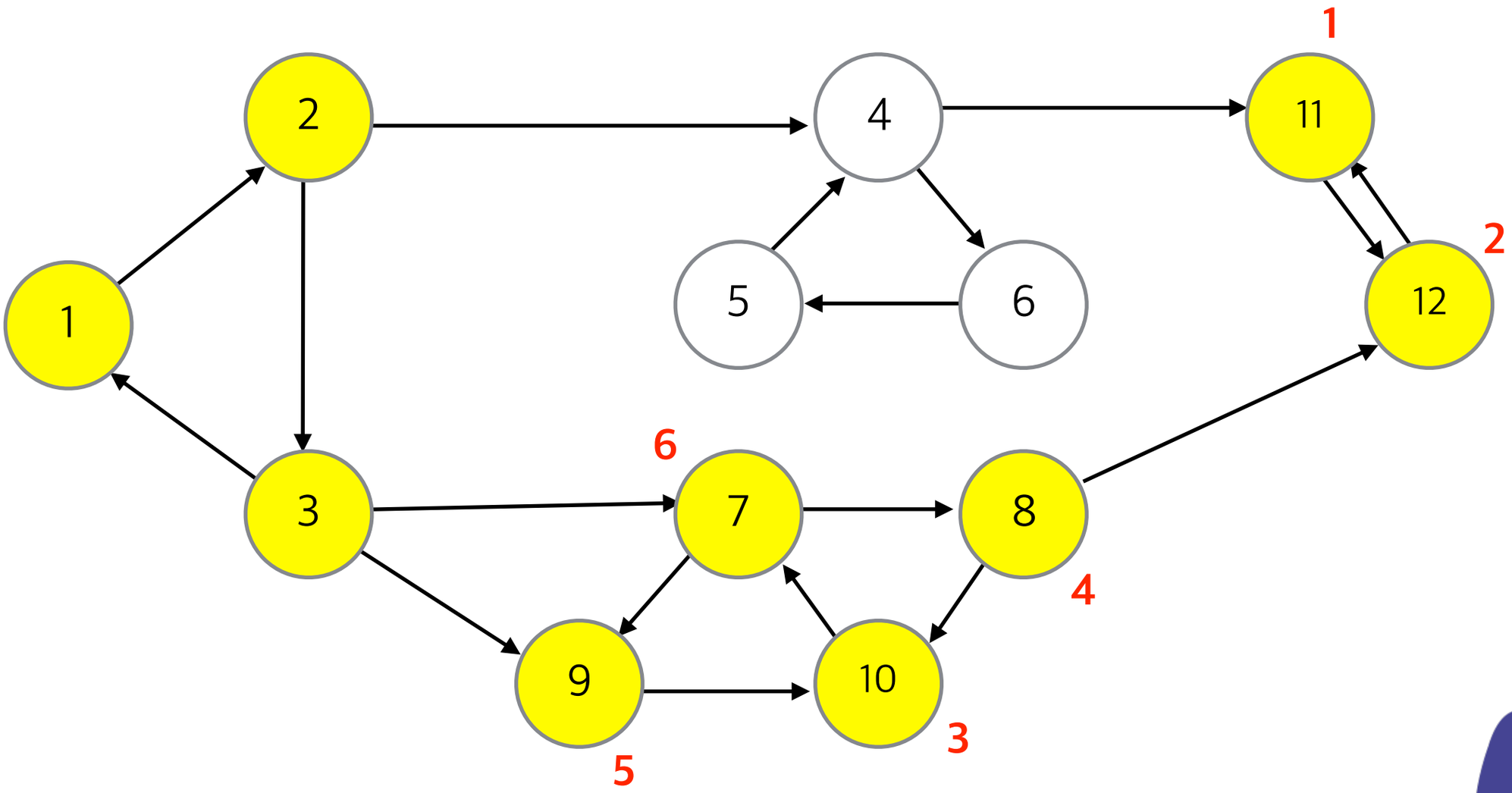
- vertex를 방문하면서 **빠져나오는 시간**을 적어놓자

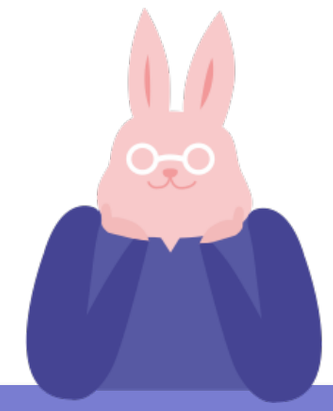




Strongly Connected Component

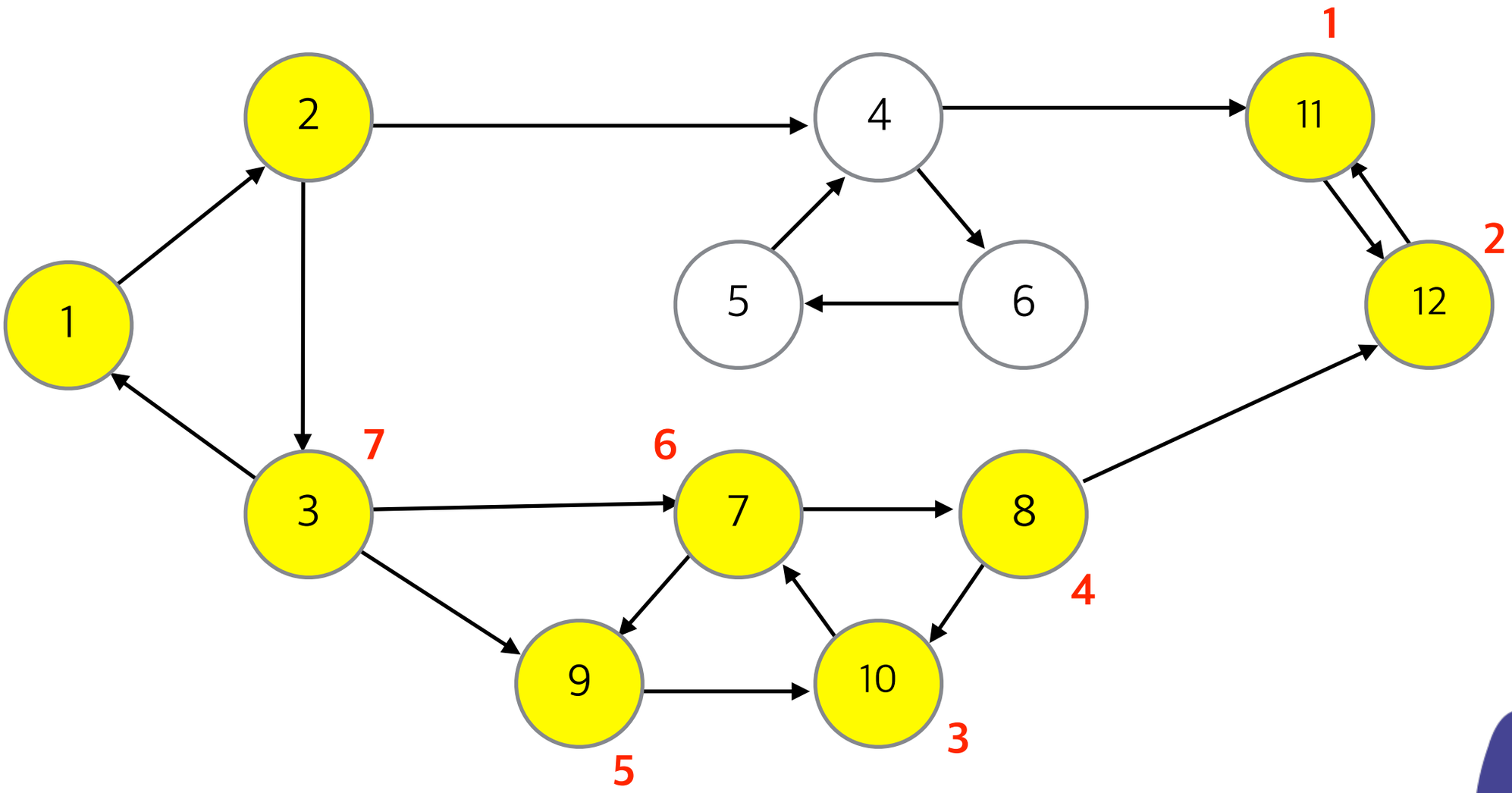
- vertex를 방문하면서 **빠져나오는 시간**을 적어놓자

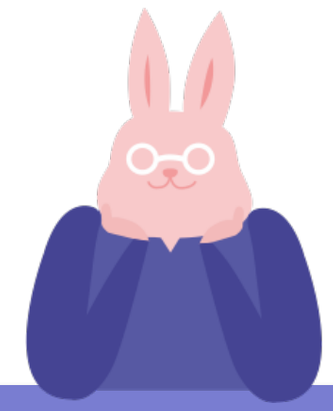




Strongly Connected Component

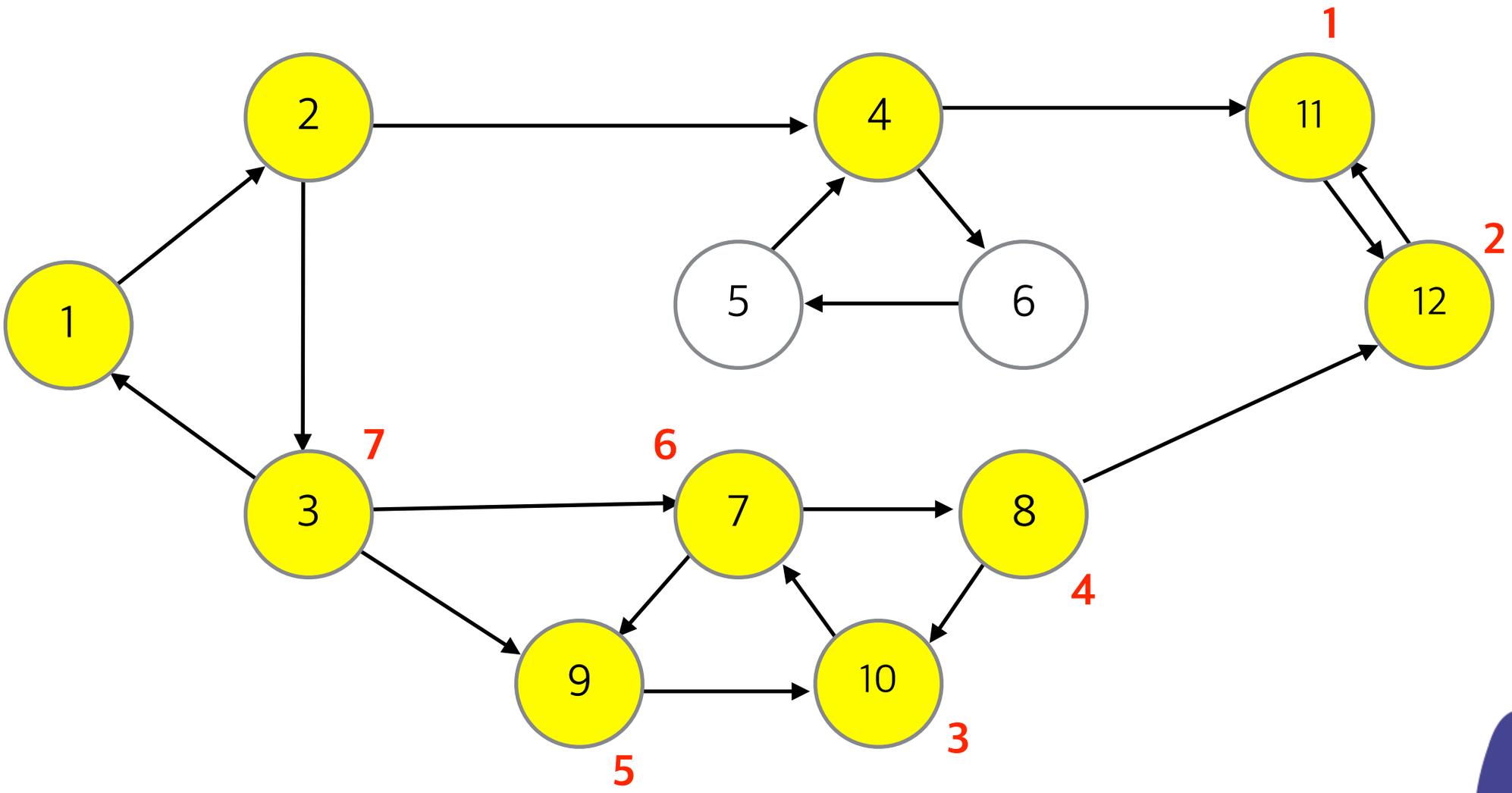
- vertex를 방문하면서 **빠져나오는 시간**을 적어놓자

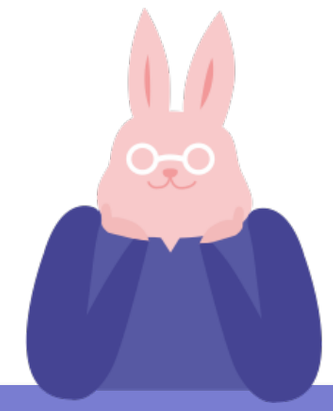




Strongly Connected Component

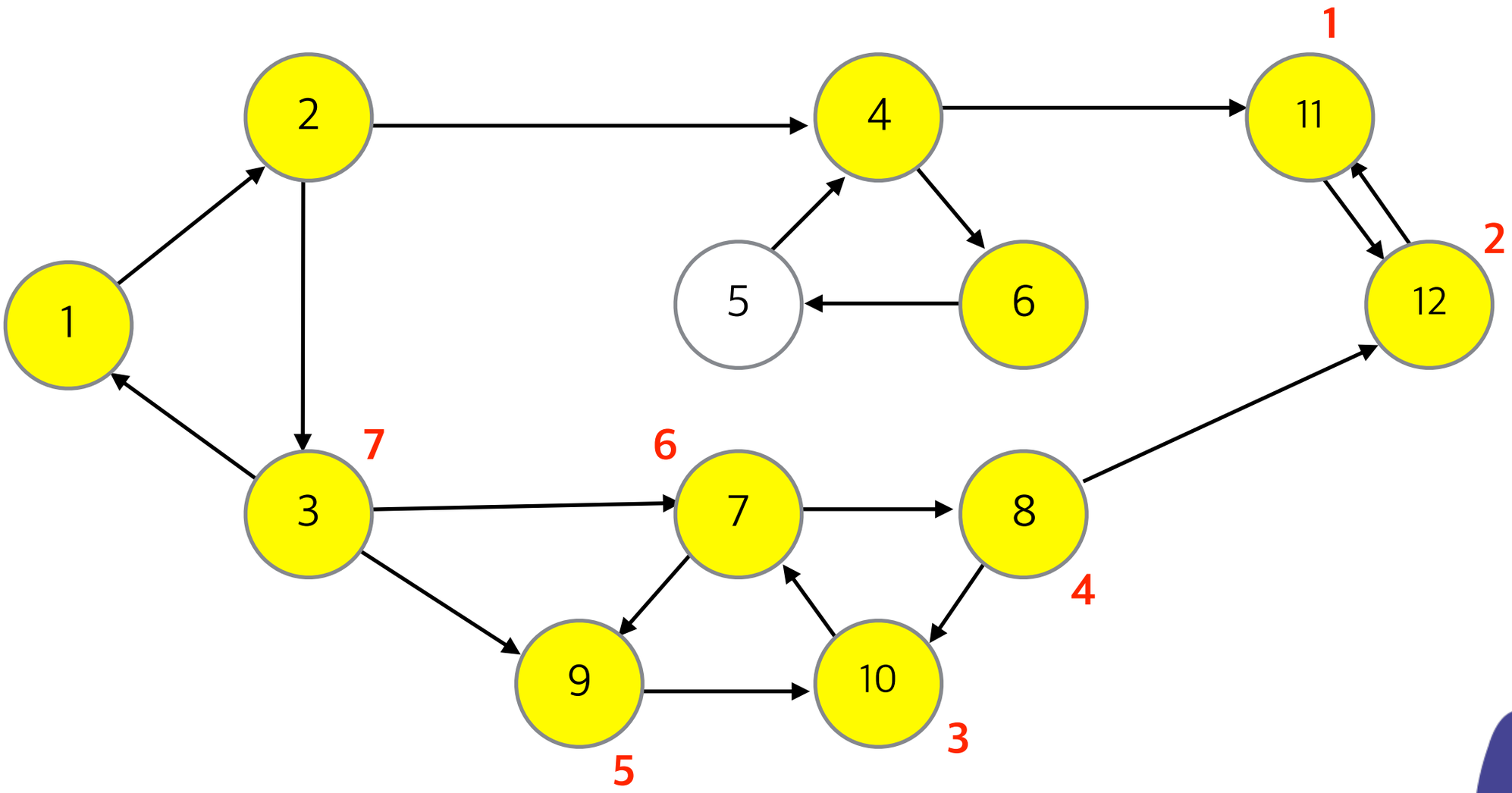
- vertex를 방문하면서 **빠져나오는 시간**을 적어놓자

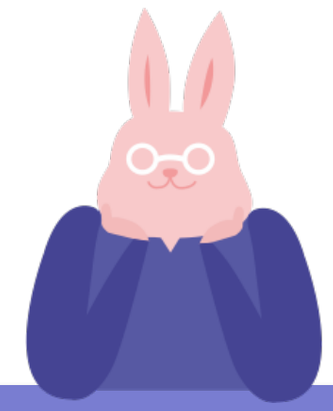




Strongly Connected Component

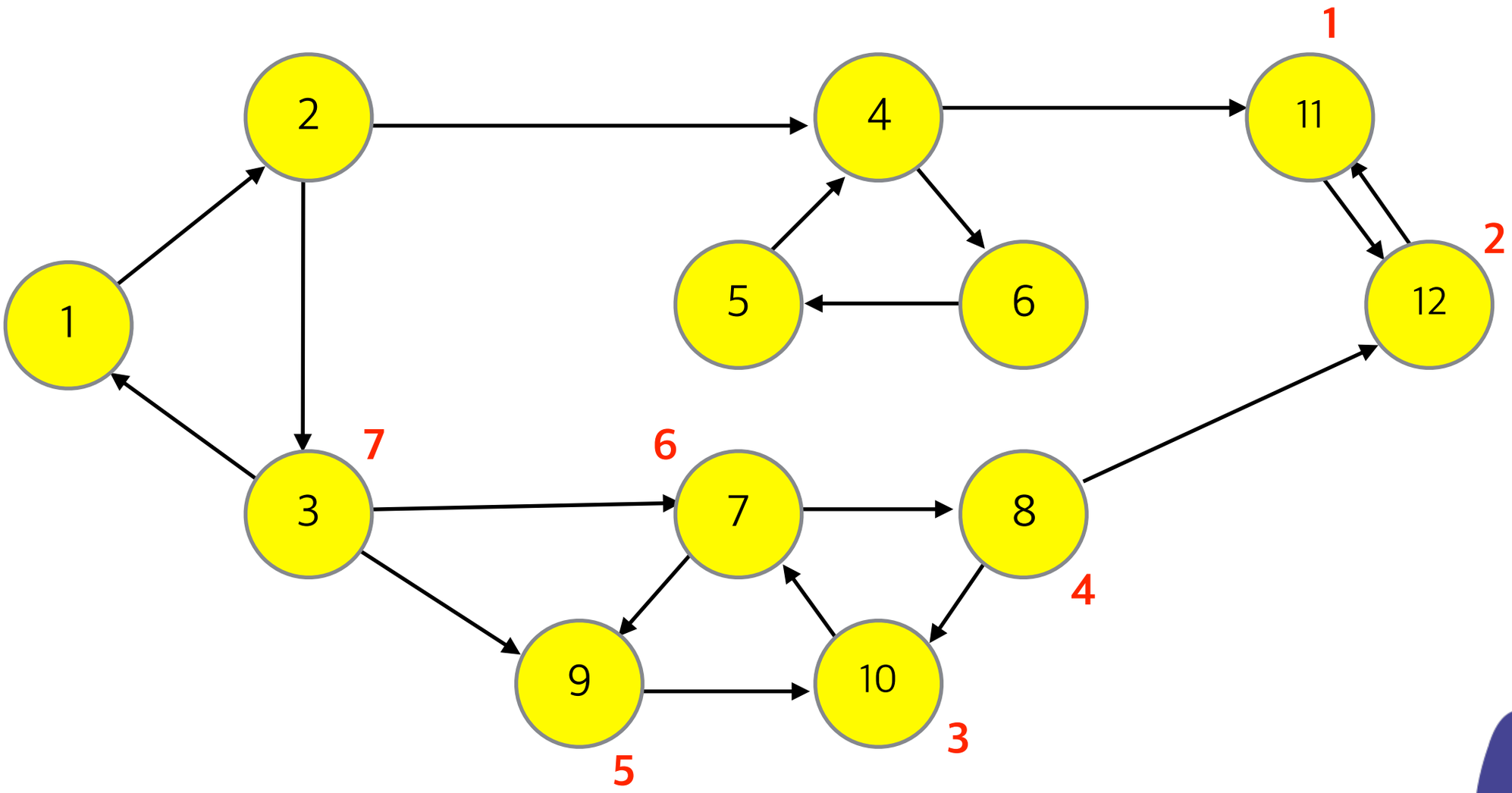
- vertex를 방문하면서 **빠져나오는 시간**을 적어놓자

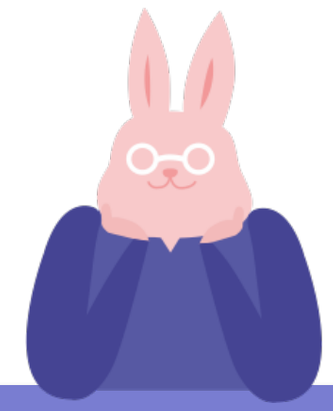




Strongly Connected Component

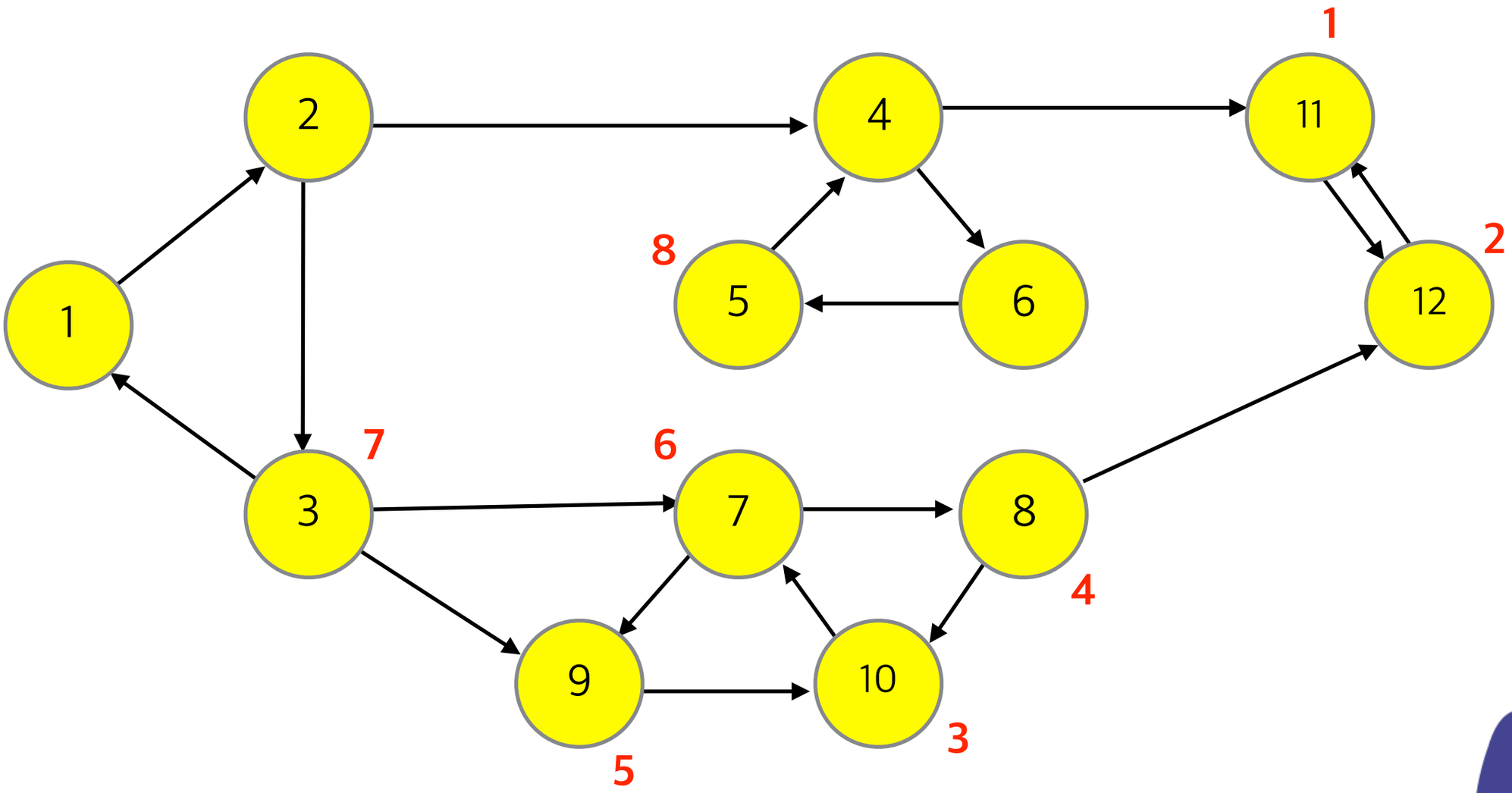
- vertex를 방문하면서 **빠져나오는 시간**을 적어놓자

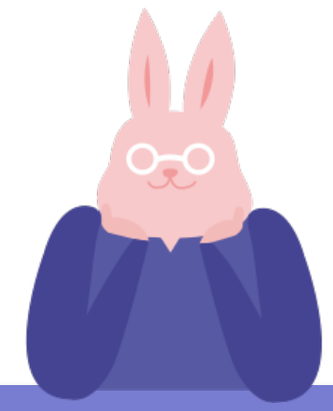




Strongly Connected Component

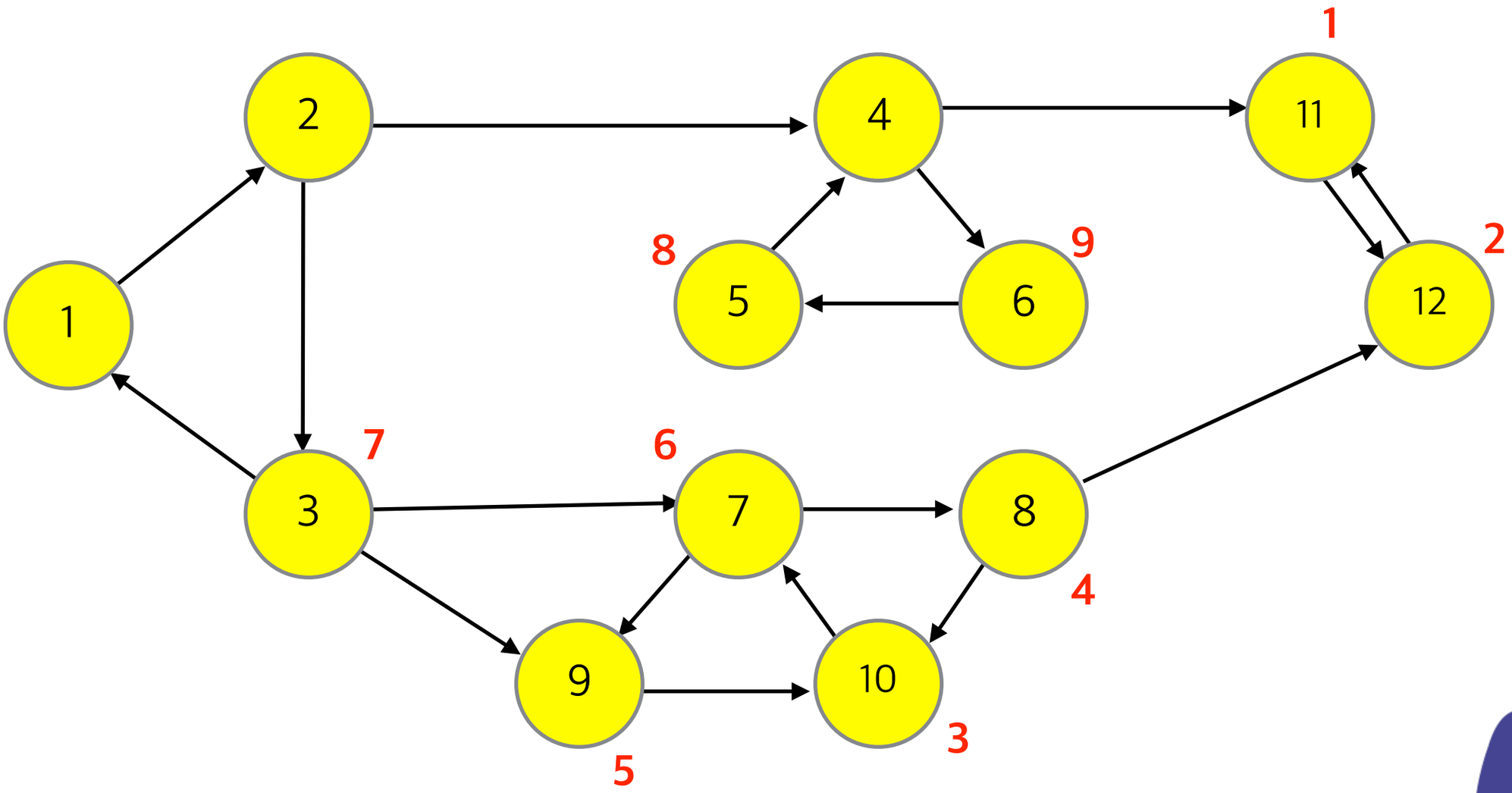
- vertex를 방문하면서 **빠져나오는 시간**을 적어놓자

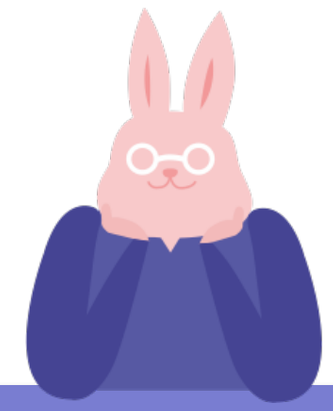




Strongly Connected Component

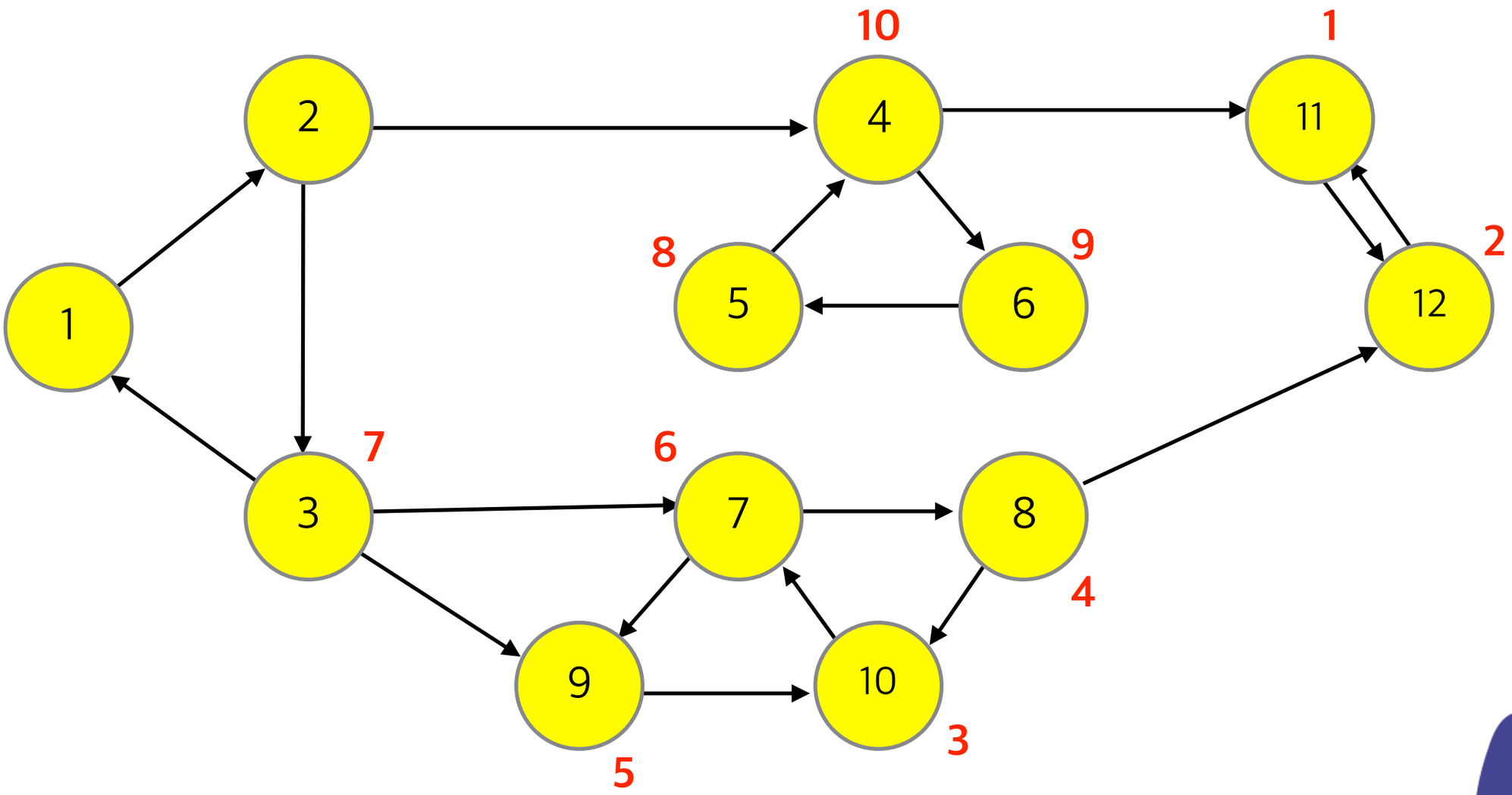
- vertex를 방문하면서 **빠져나오는 시간**을 적어놓자

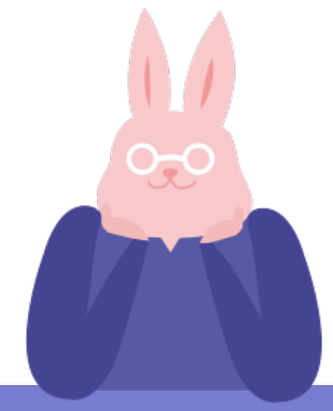




Strongly Connected Component

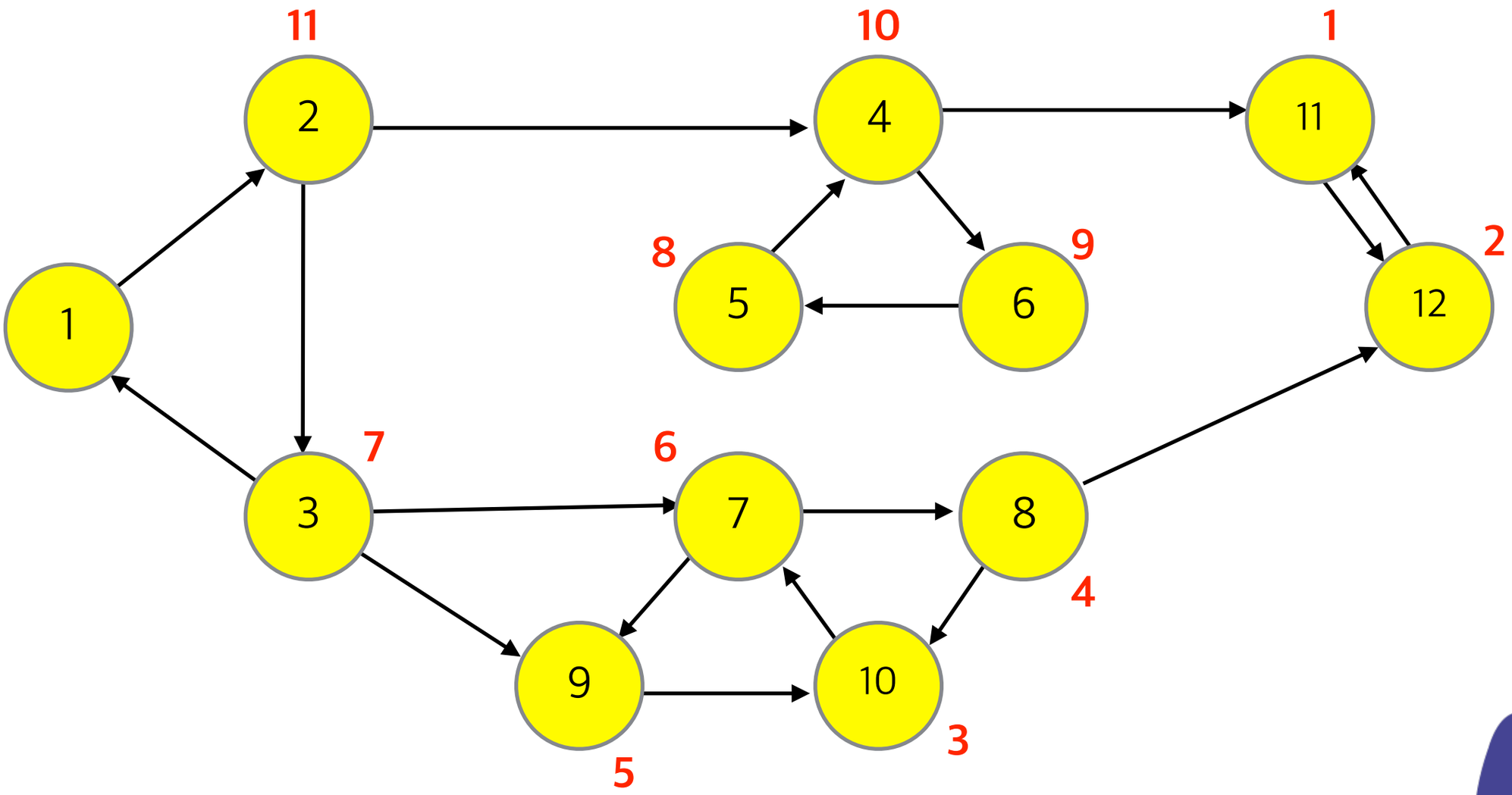
- vertex를 방문하면서 **빠져나오는 시간**을 적어놓자

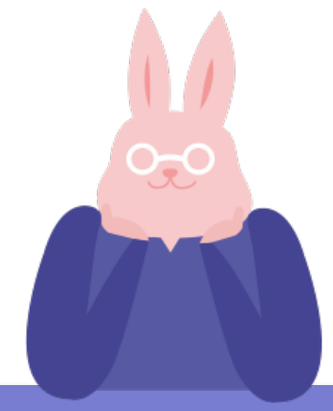




Strongly Connected Component

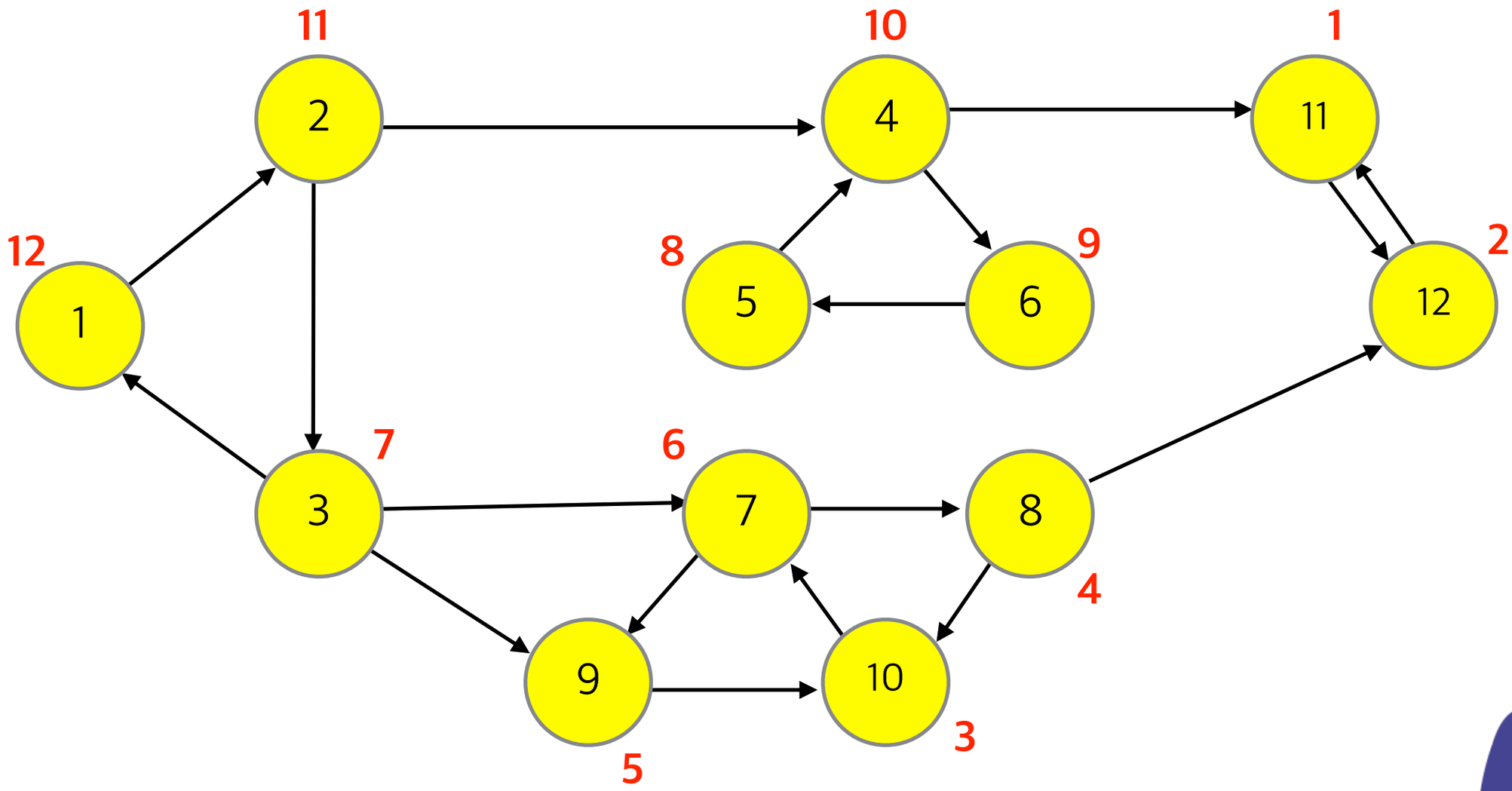
- vertex를 방문하면서 **빠져나오는 시간**을 적어놓자

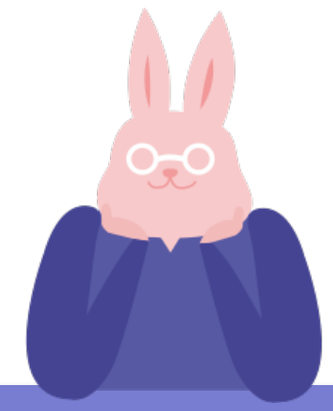




Strongly Connected Component

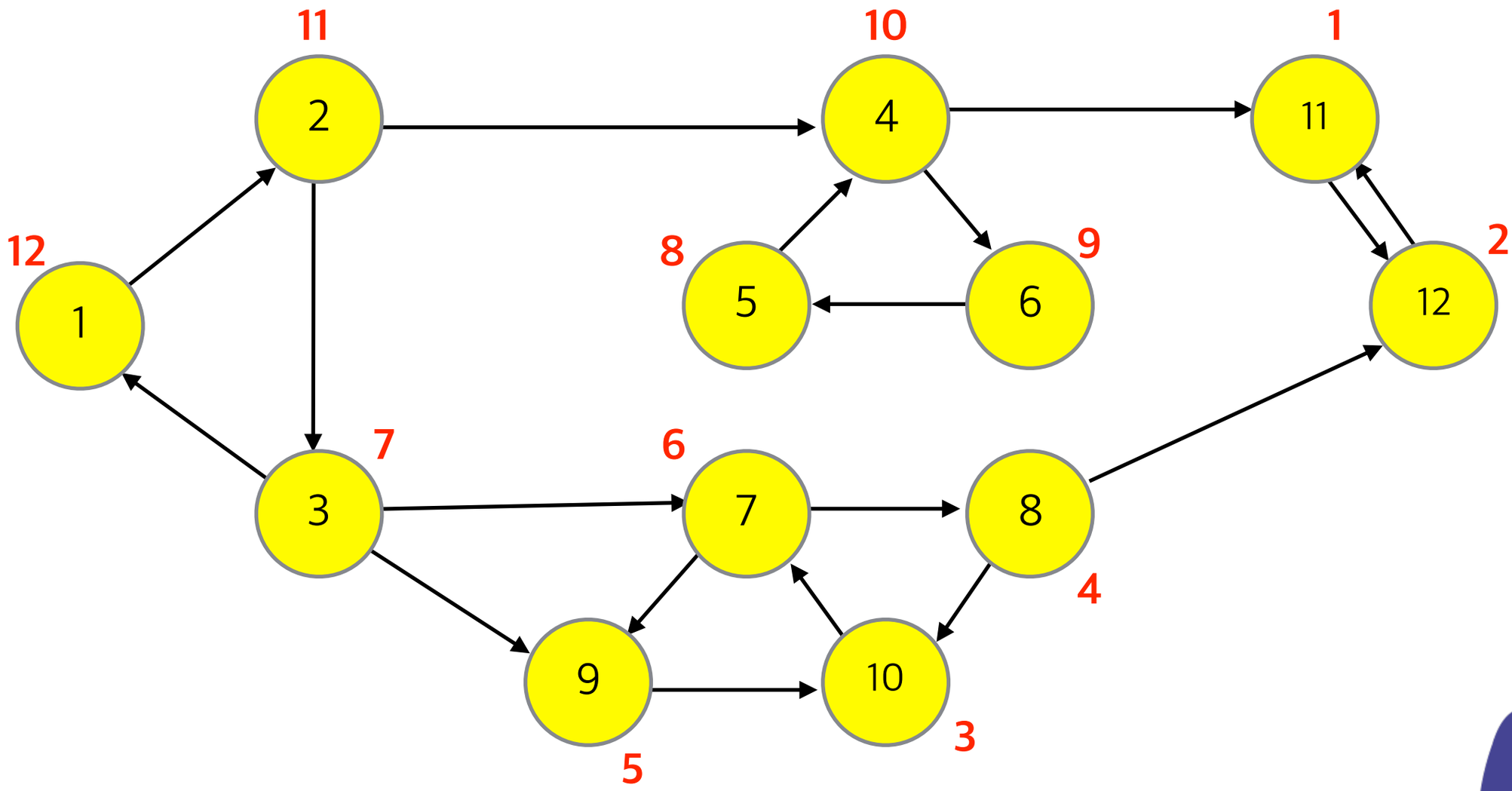
- vertex를 방문하면서 **빠져나오는 시간**을 적어놓자

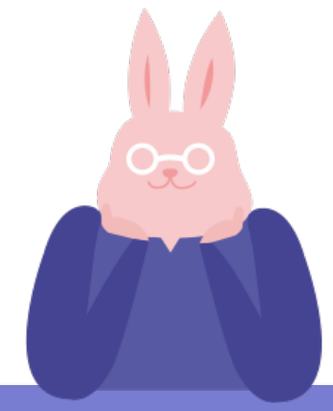




Strongly Connected Component

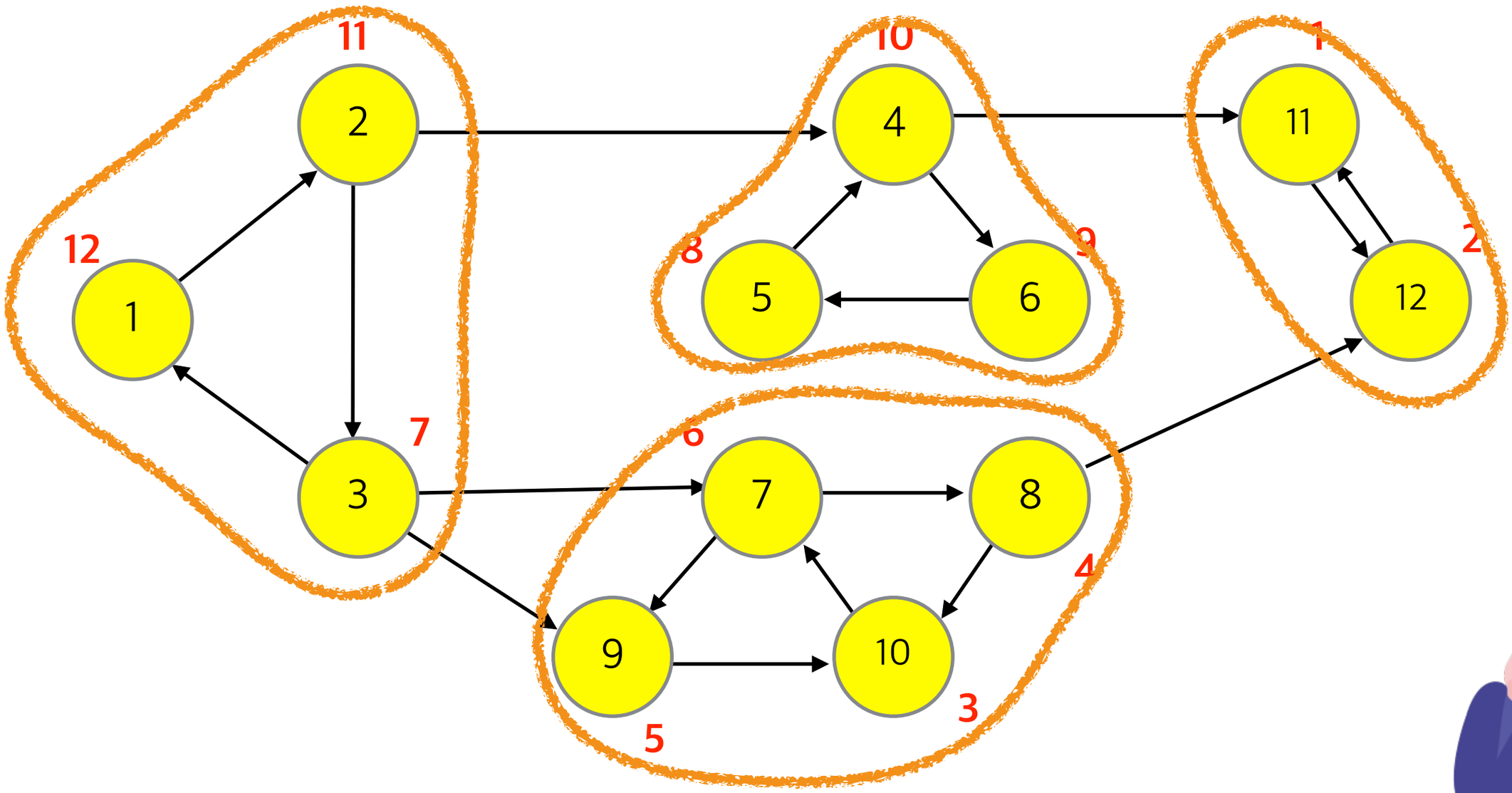
- 이 숫자를 이용해서 방문 순서를 생각하면 ?

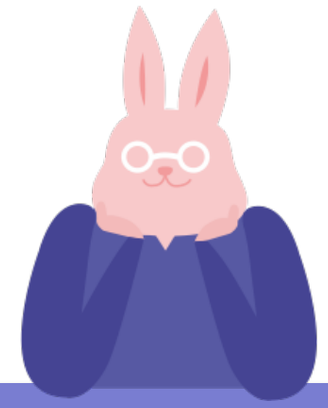




Strongly Connected Component

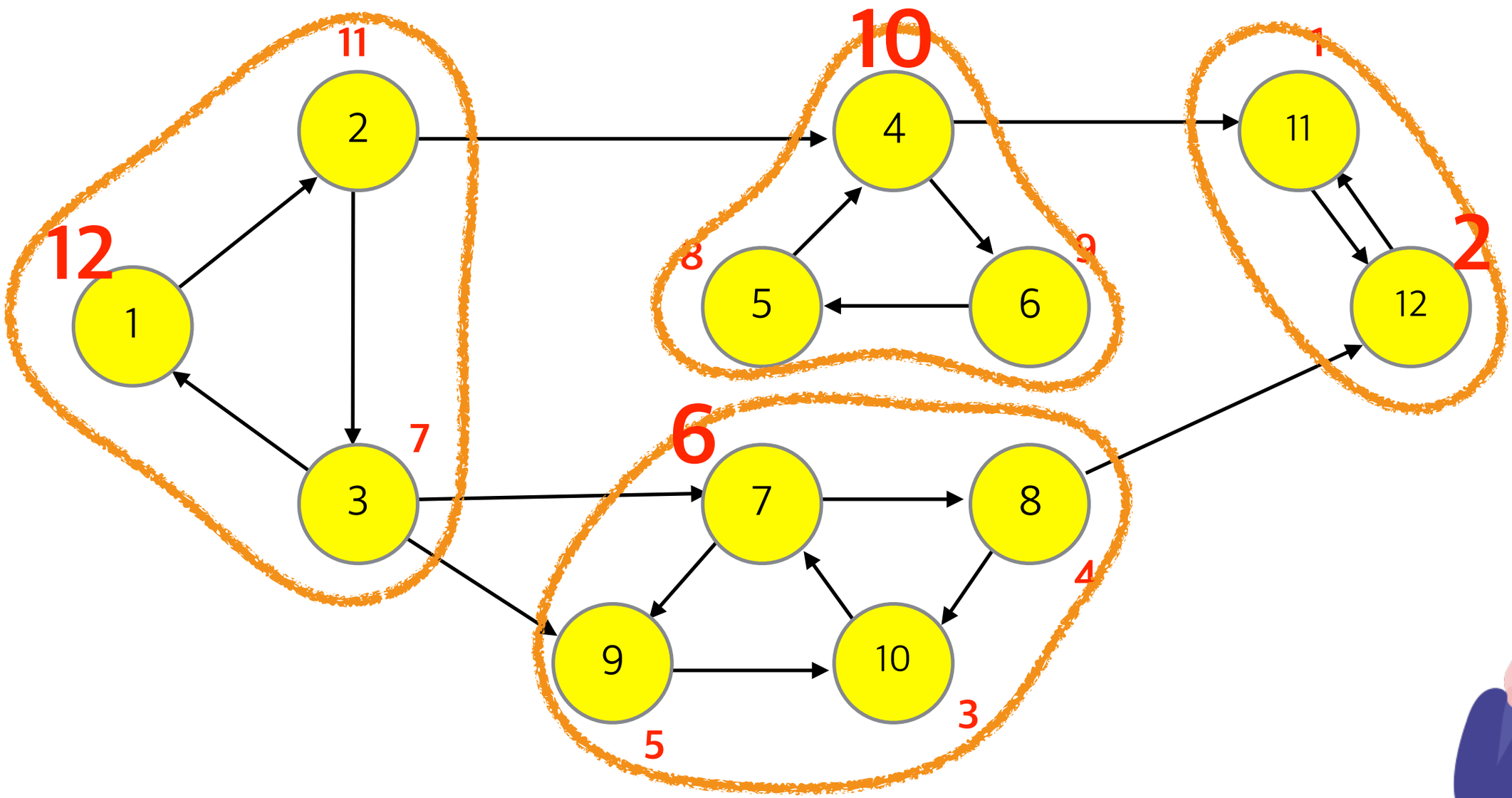
- 이 숫자를 이용해서 방문 순서를 생각하면 ?

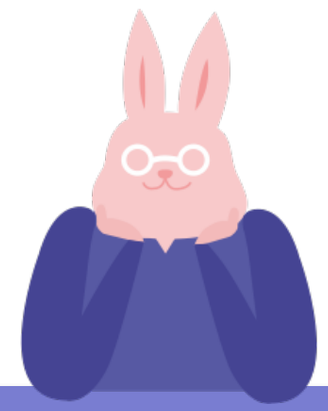




Strongly Connected Component

- 이 숫자를 이용해서 방문 순서를 생각하면 ?

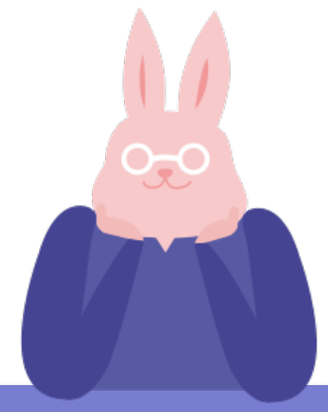




Strongly Connected Component

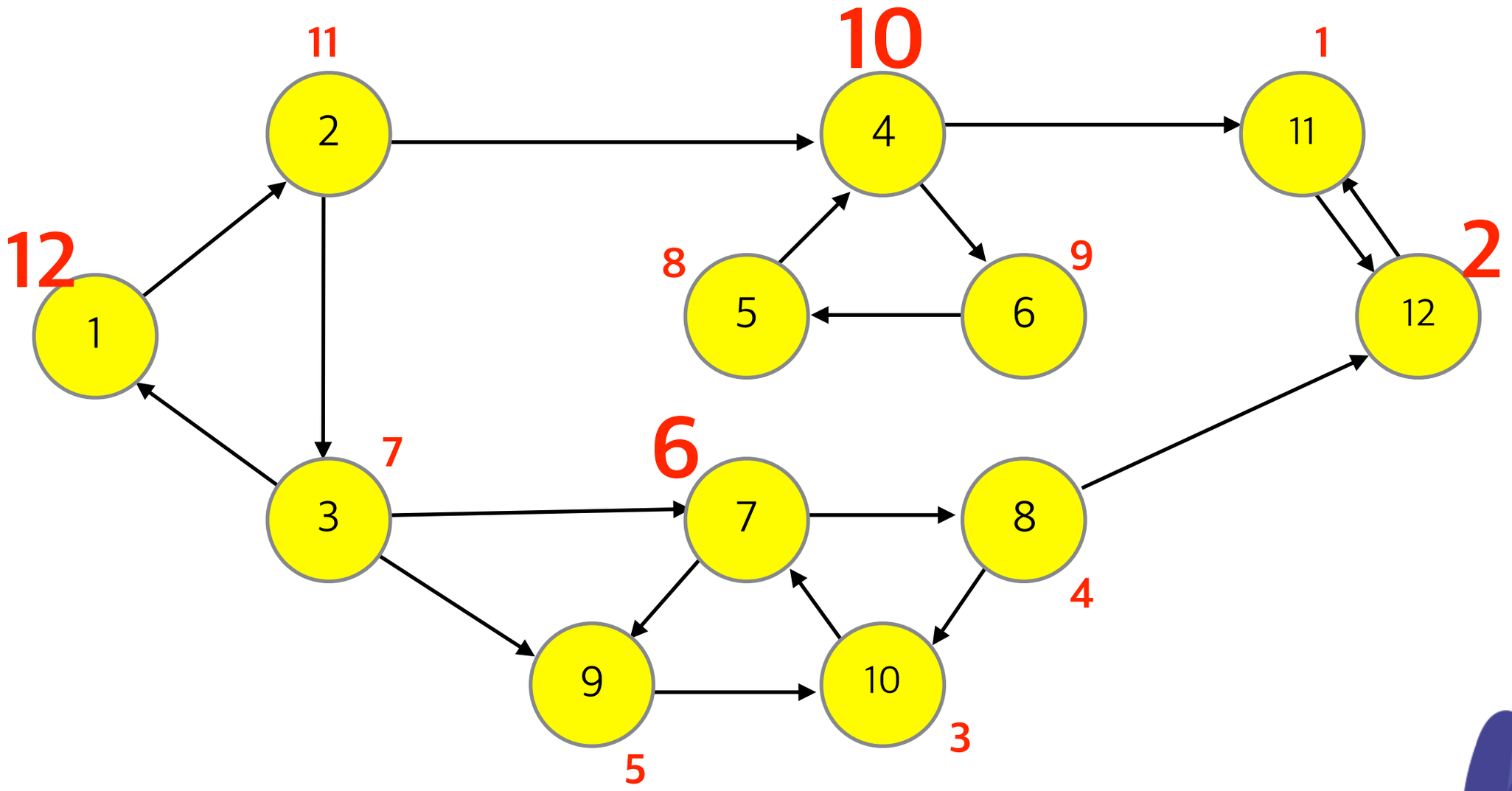
- Claim
  - 숫자가 큰 Node부터 traverse해보면 된다

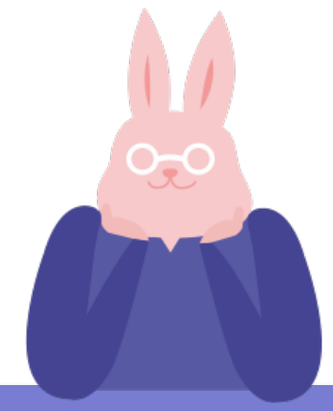
- Proof



Strongly Connected Component

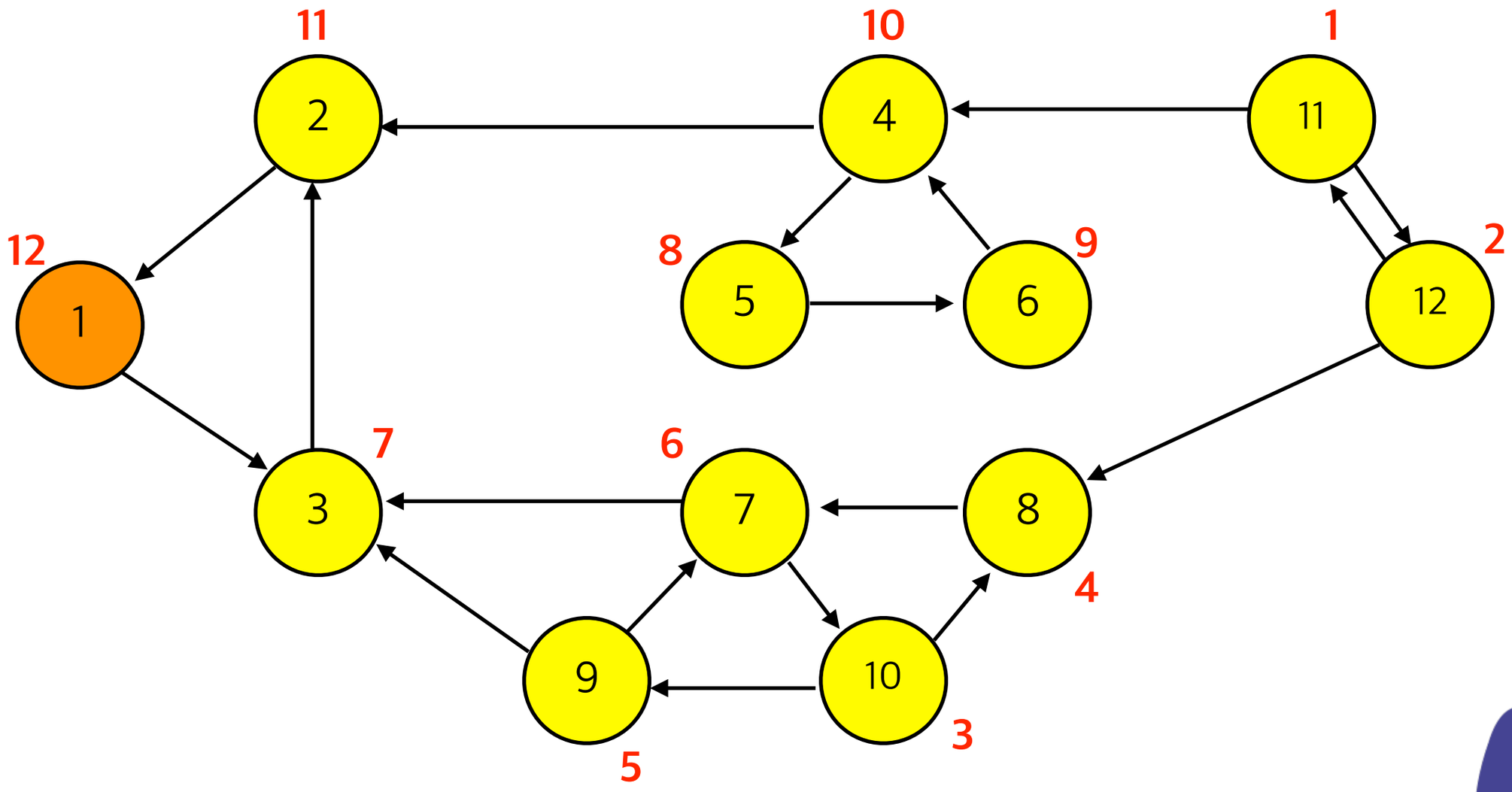
- 이 숫자를 이용해서 방문 순서를 생각하면 ?

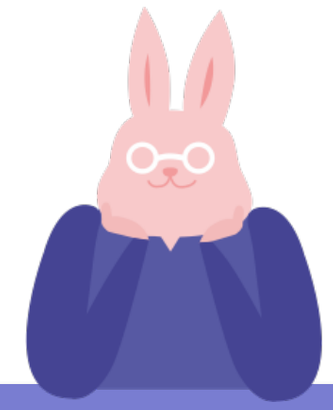




Strongly Connected Component

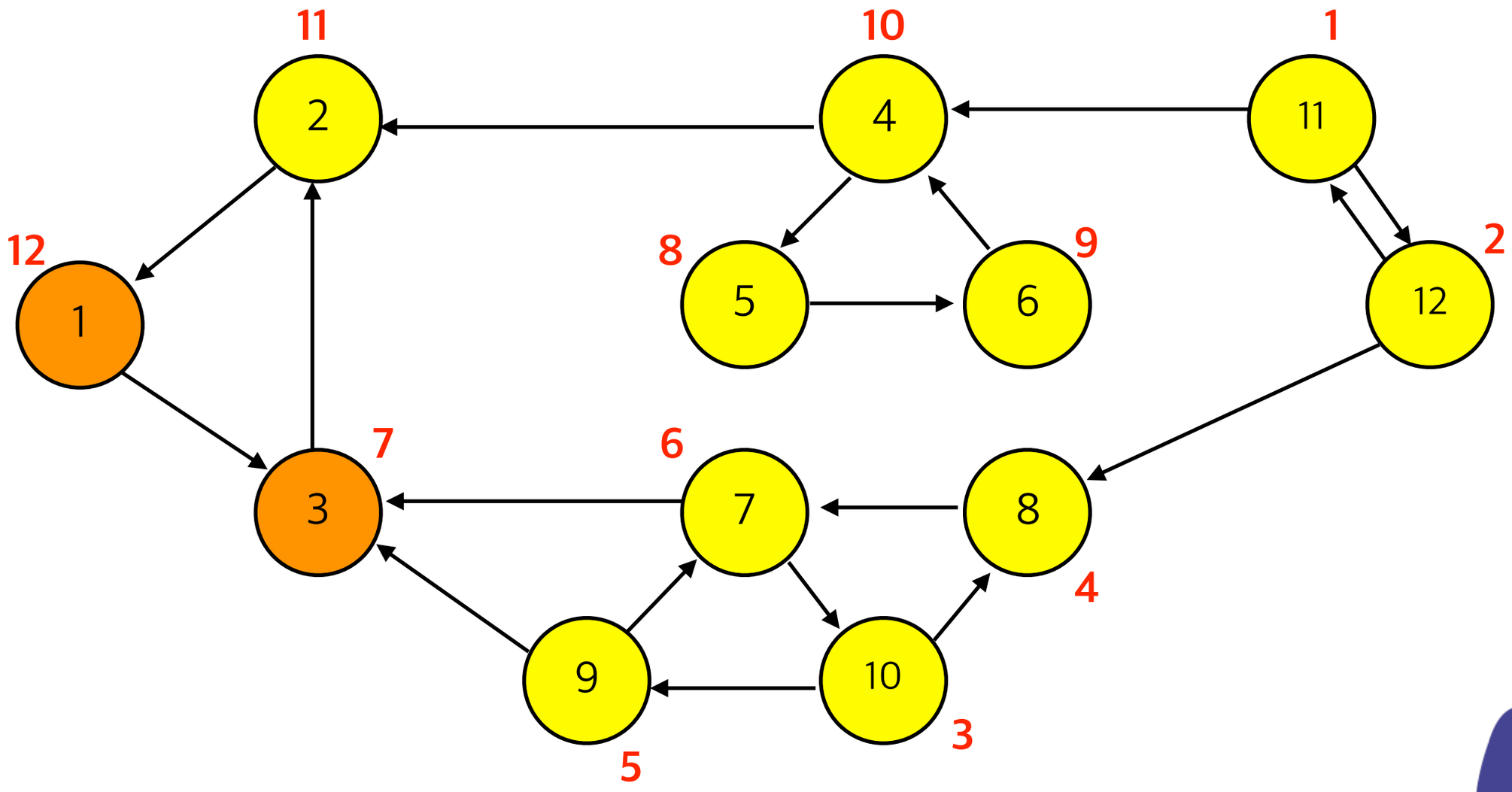
- 이 숫자를 이용해서 방문 순서를 생각하면 ?

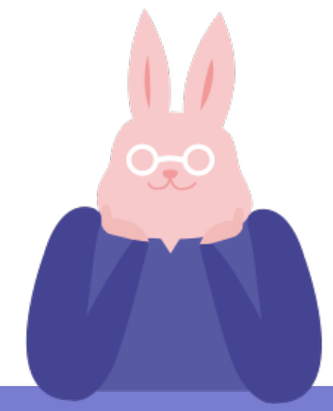




Strongly Connected Component

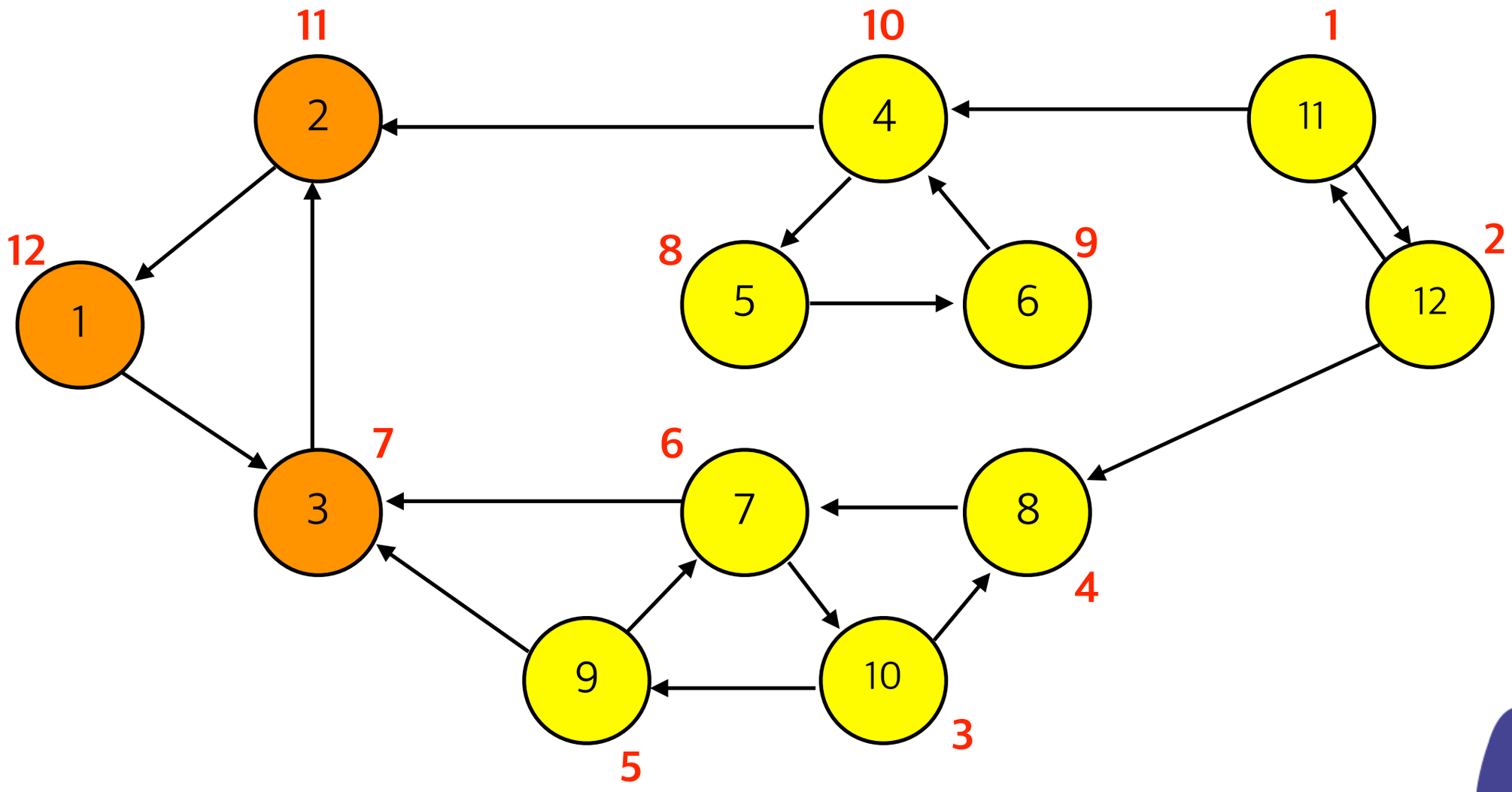
- 이 숫자를 이용해서 방문 순서를 생각하면 ?

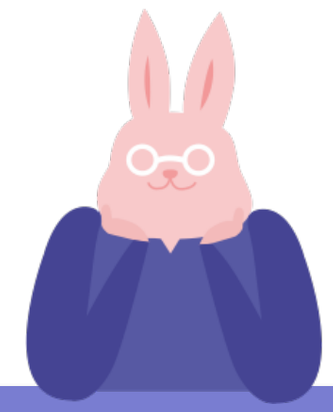




Strongly Connected Component

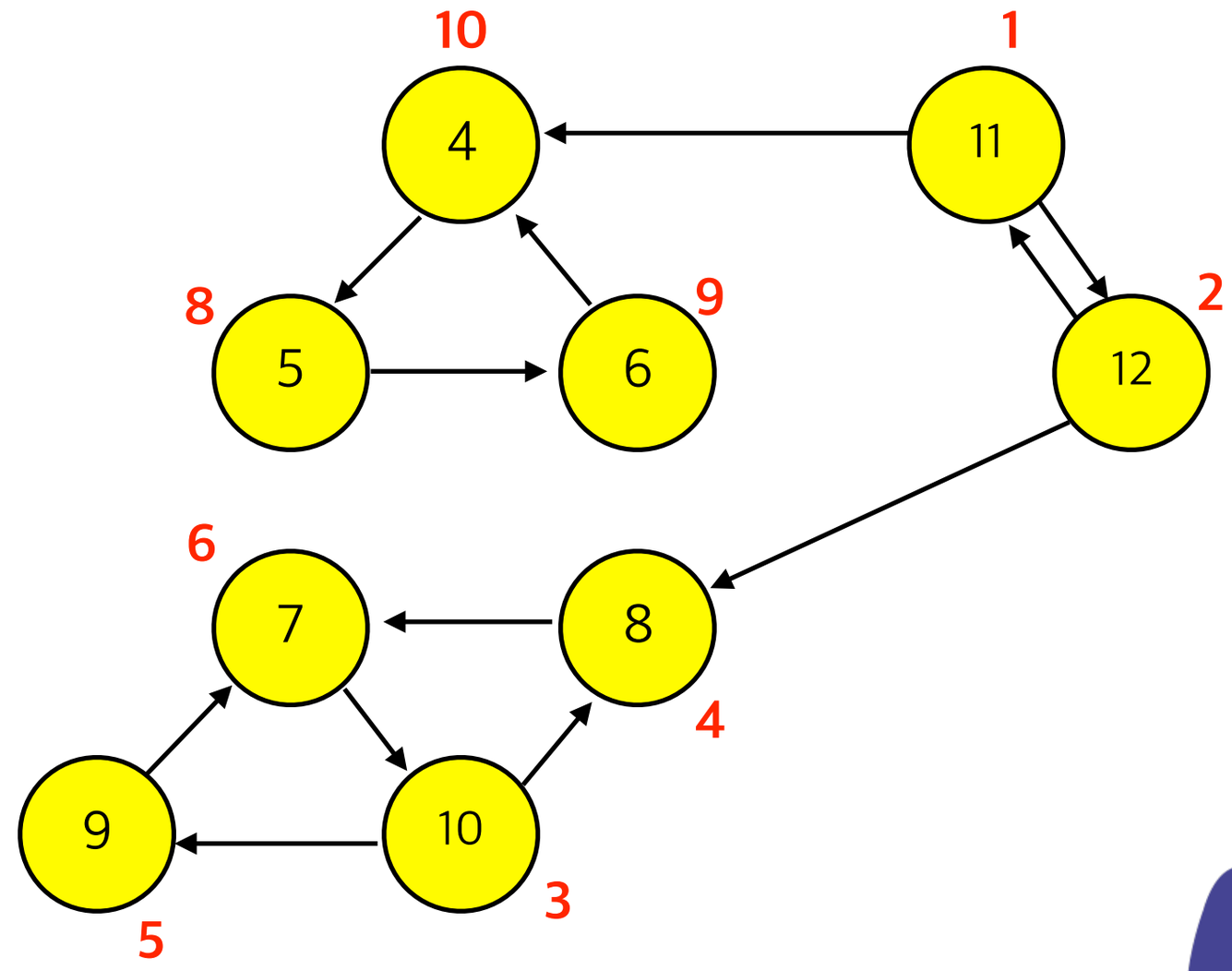
- 이 숫자를 이용해서 방문 순서를 생각하면 ?

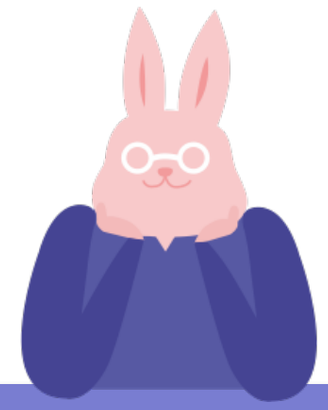




Strongly Connected Component

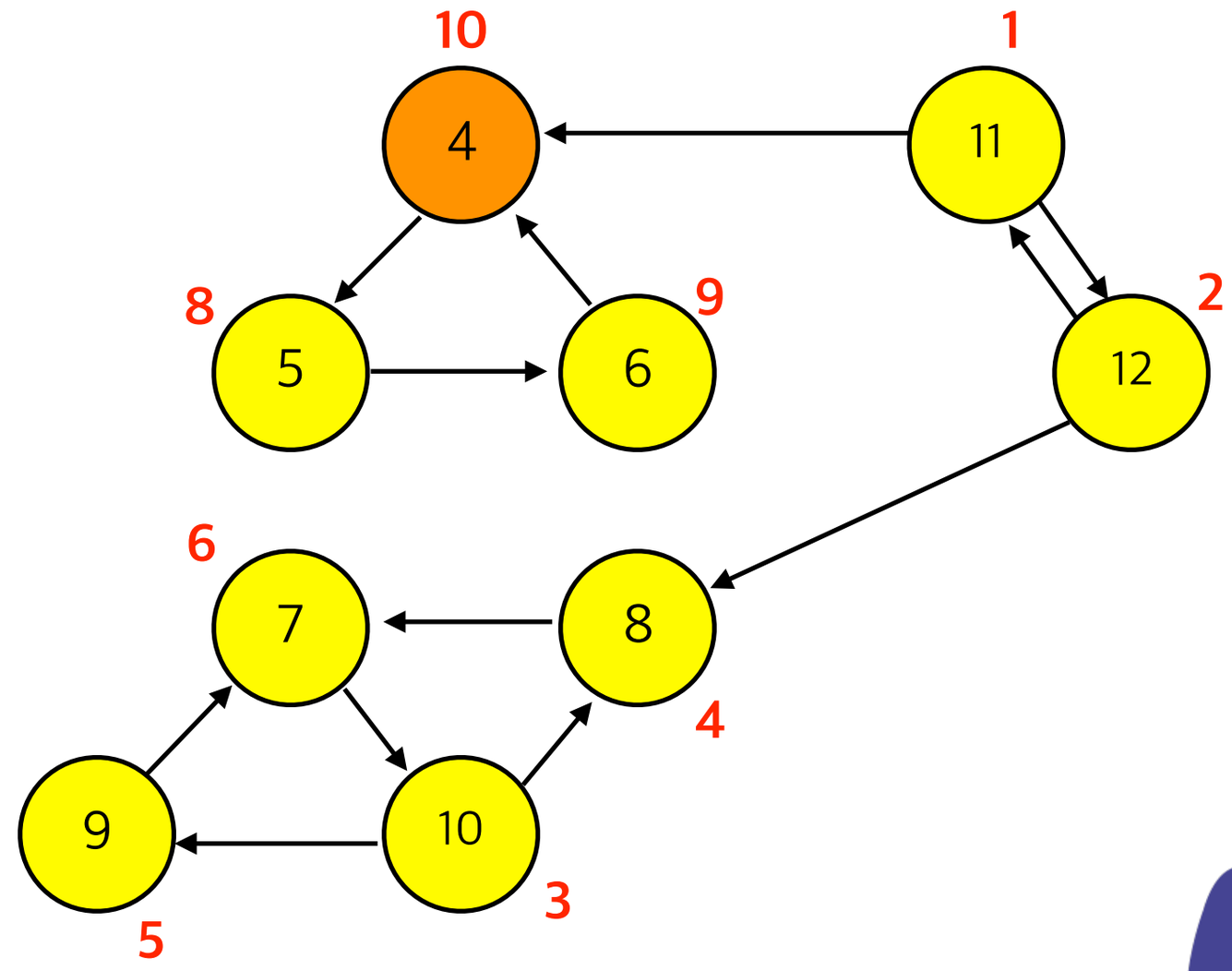
- 이 숫자를 이용해서 방문 순서를 생각하면 ?

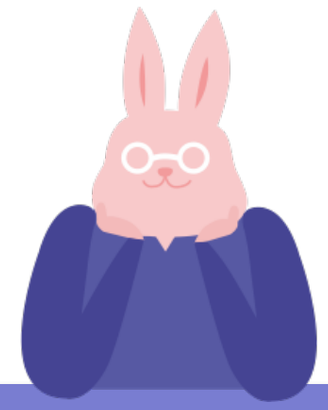




Strongly Connected Component

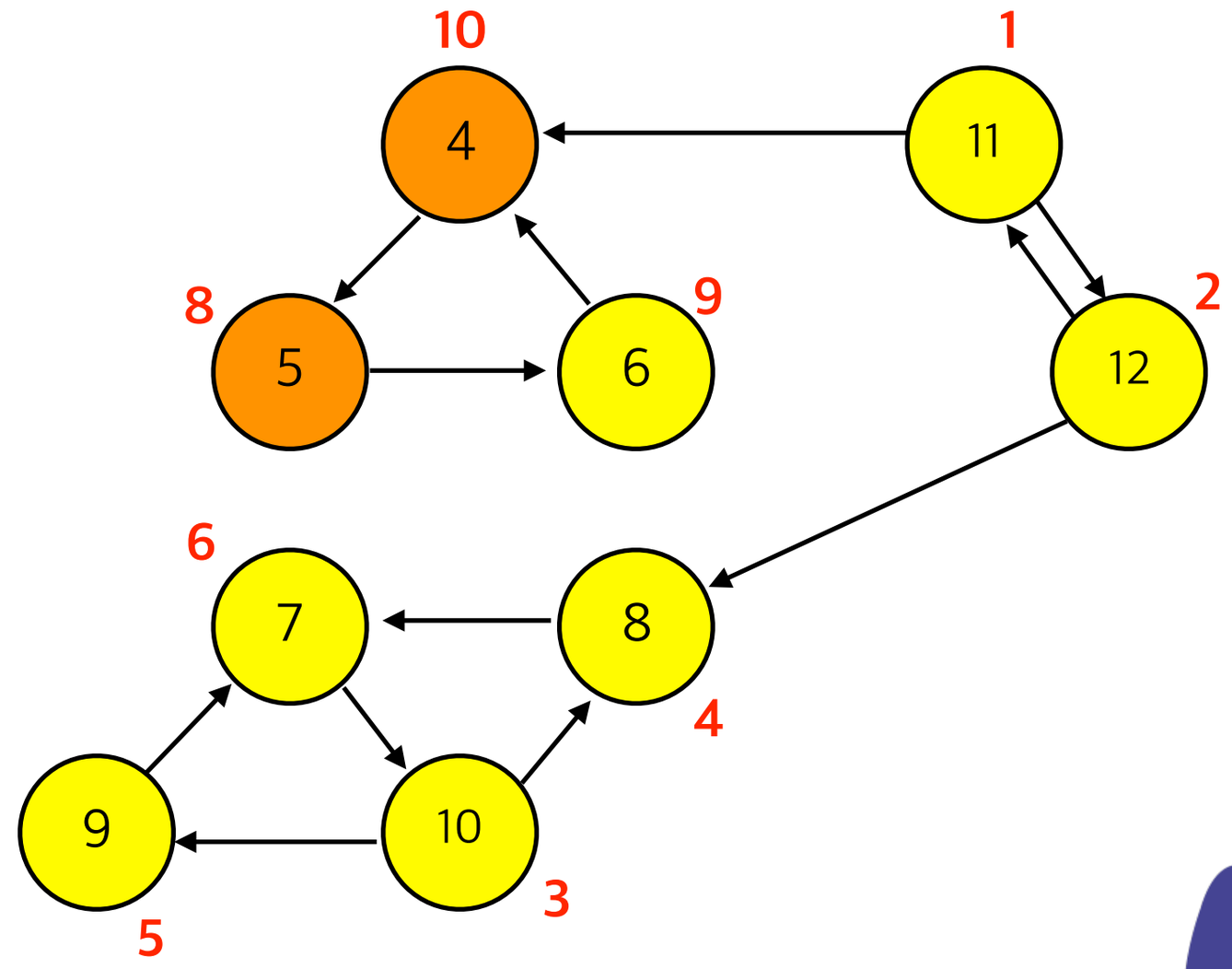
- 이 숫자를 이용해서 방문 순서를 생각하면 ?

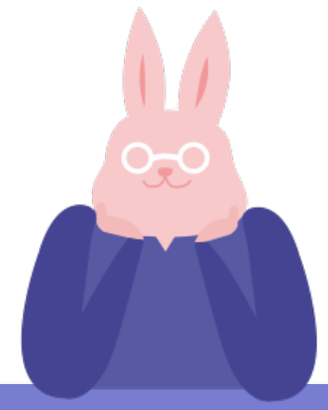




Strongly Connected Component

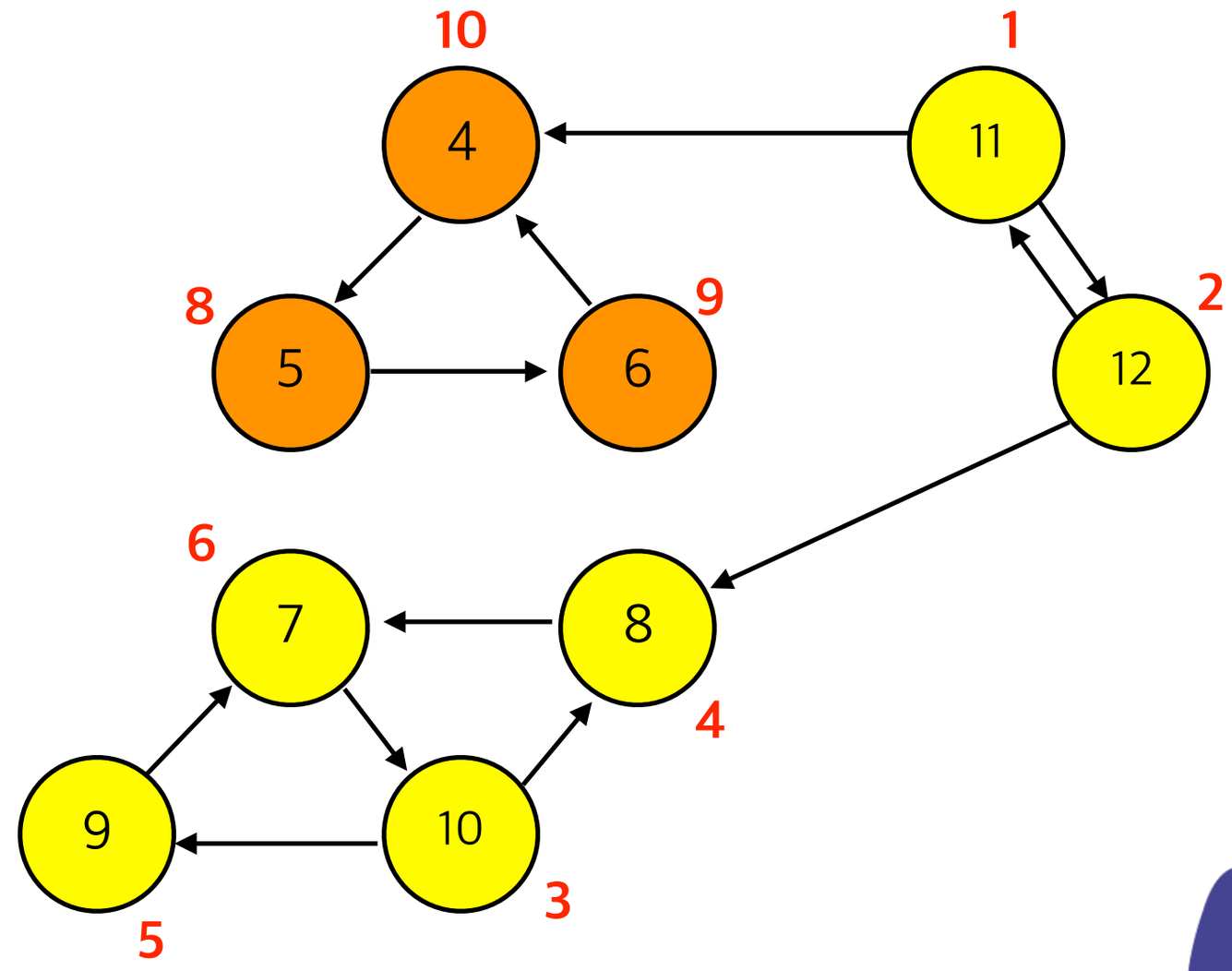
- 이 숫자를 이용해서 방문 순서를 생각하면 ?

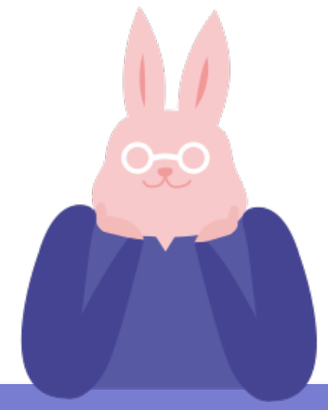




Strongly Connected Component

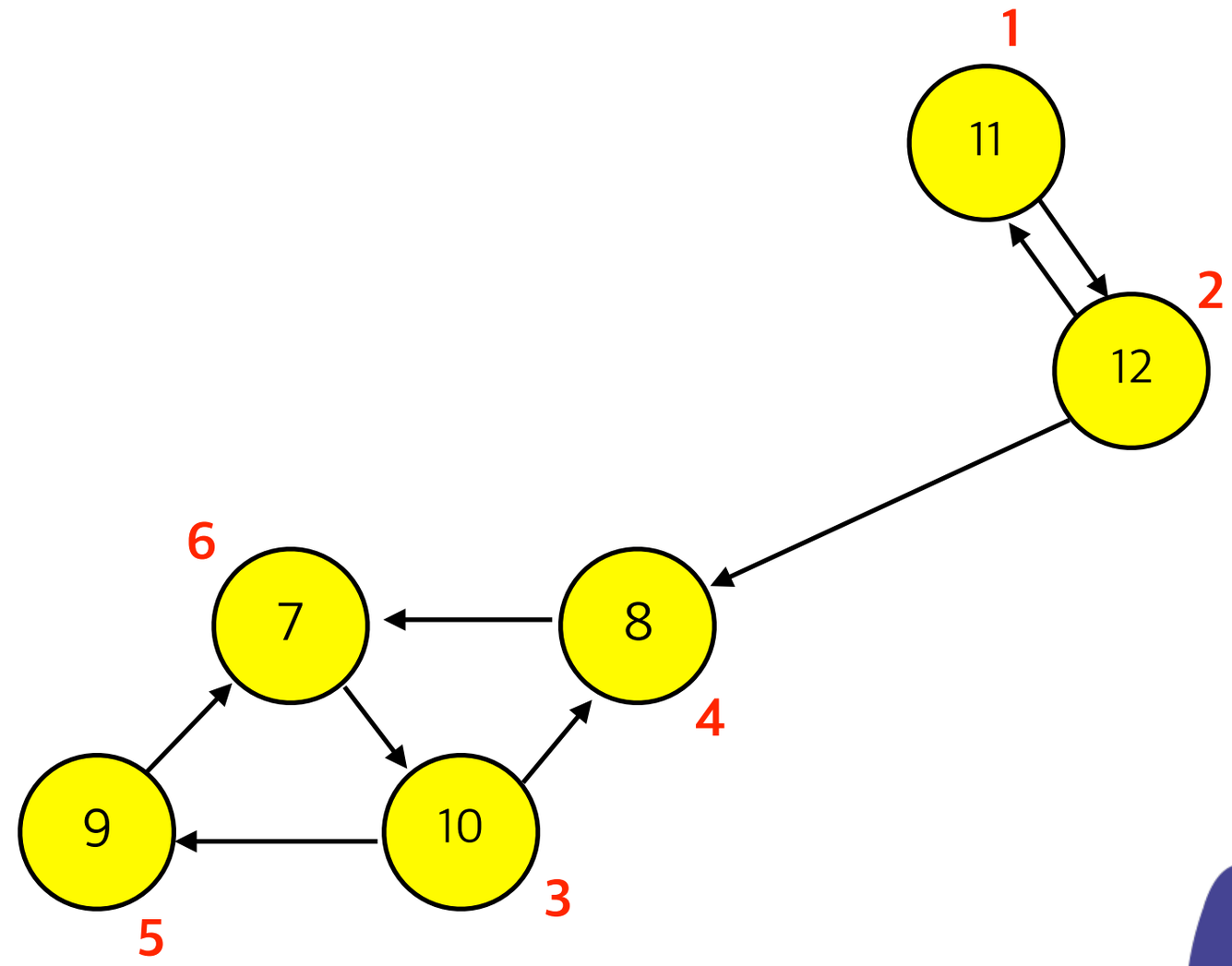
- 이 숫자를 이용해서 방문 순서를 생각하면 ?

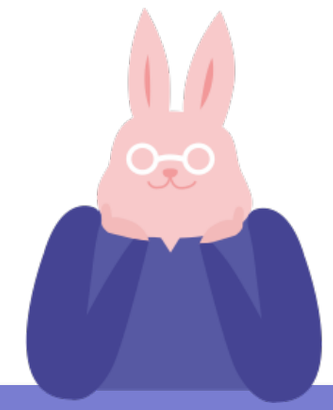




Strongly Connected Component

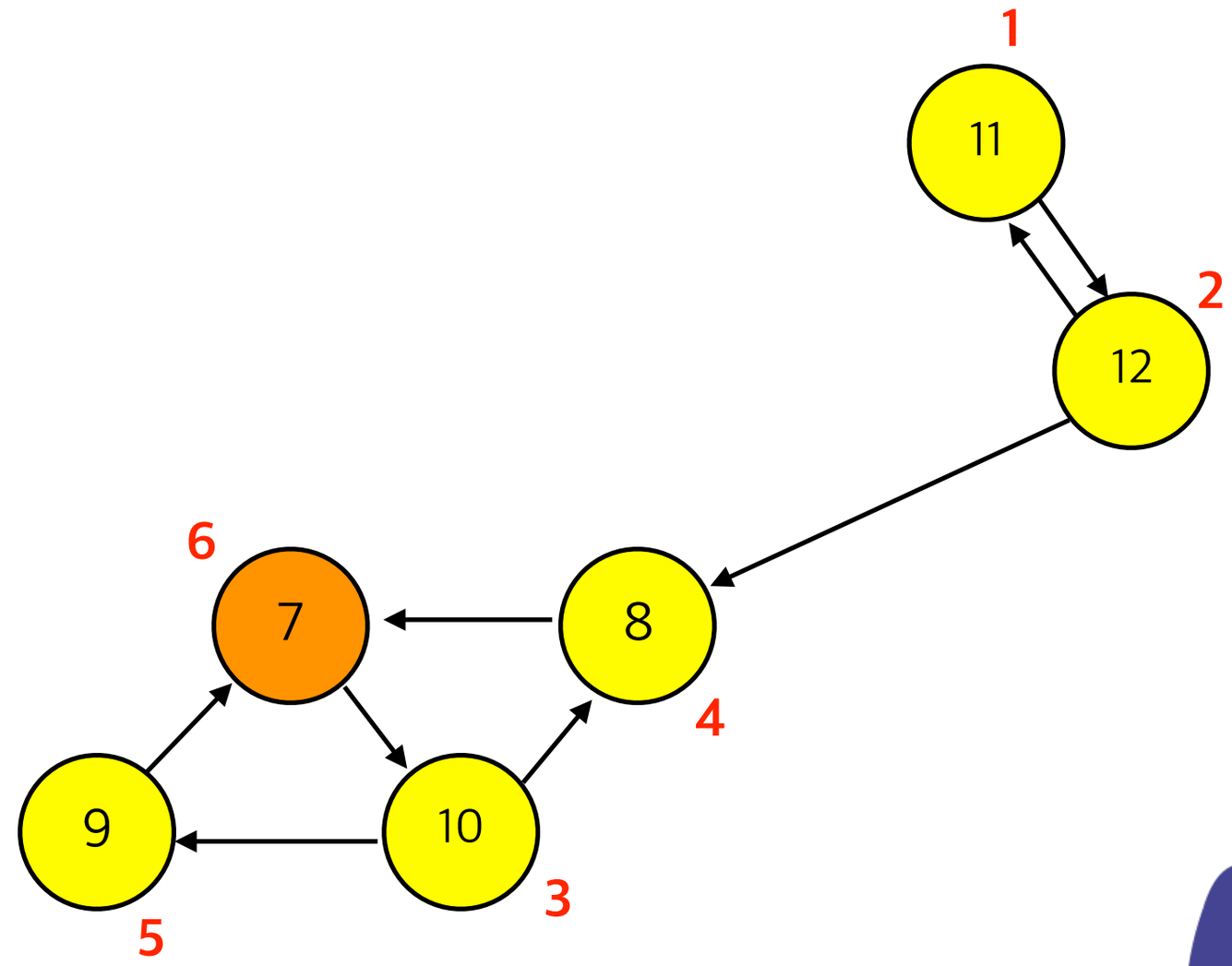
- 이 숫자를 이용해서 방문 순서를 생각하면 ?

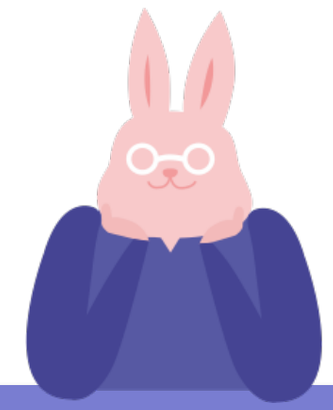




Strongly Connected Component

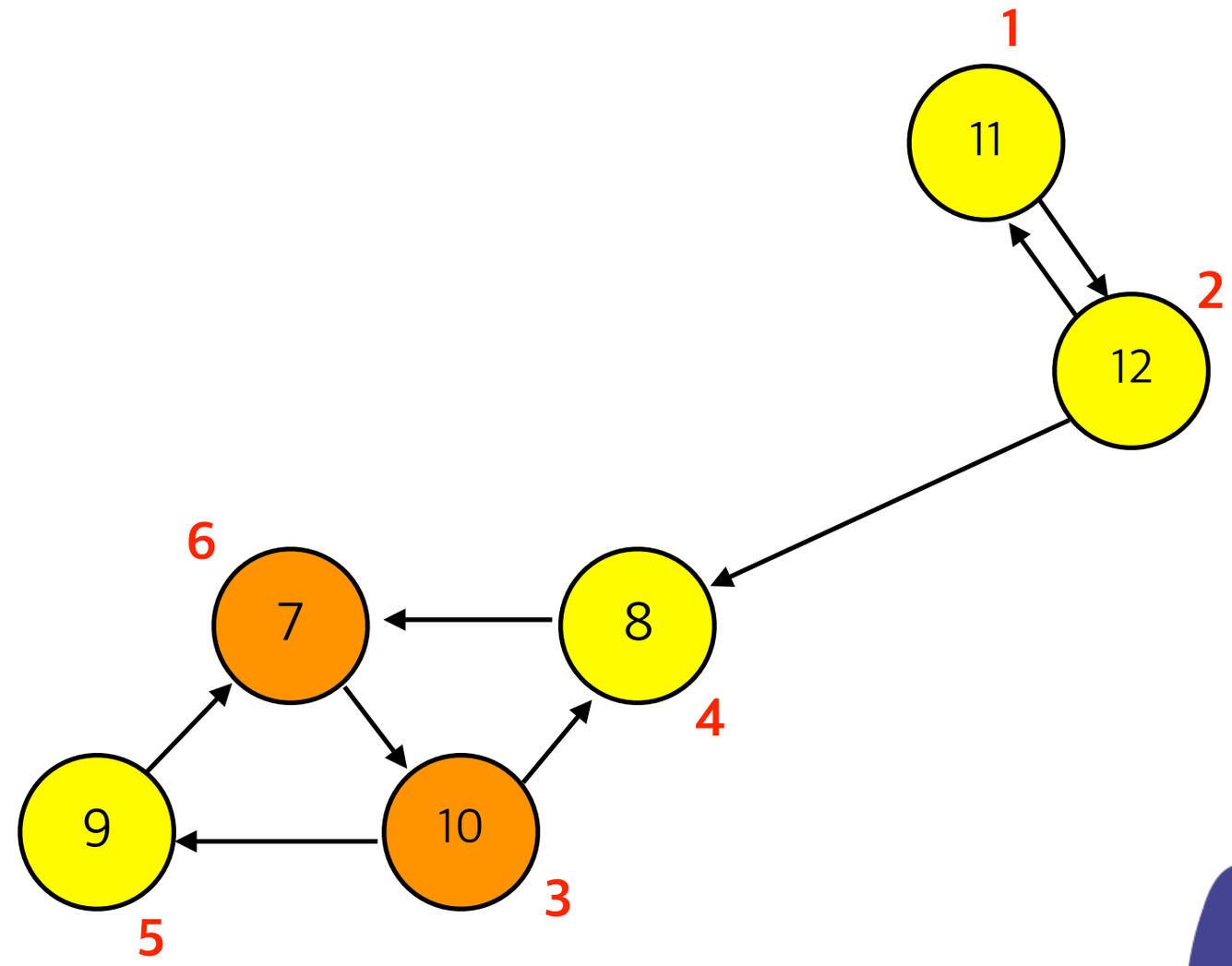
- 이 숫자를 이용해서 방문 순서를 생각하면 ?

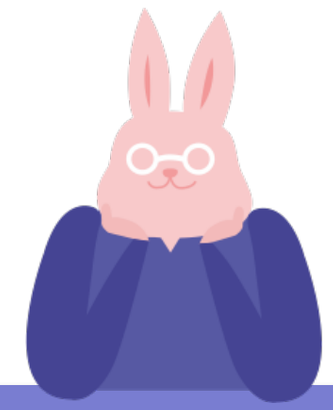




Strongly Connected Component

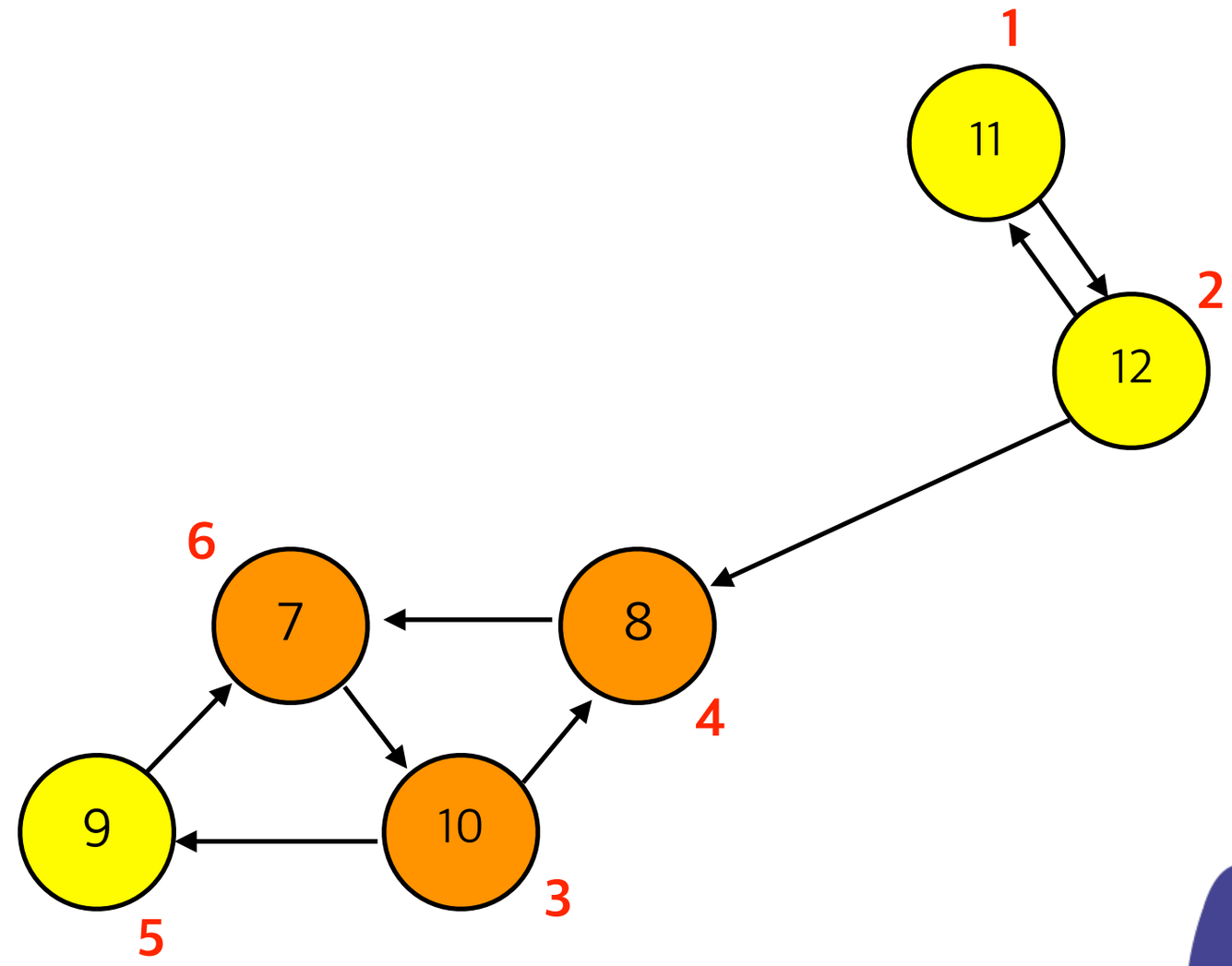
- 이 숫자를 이용해서 방문 순서를 생각하면 ?

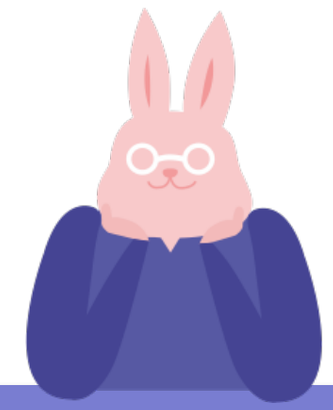




Strongly Connected Component

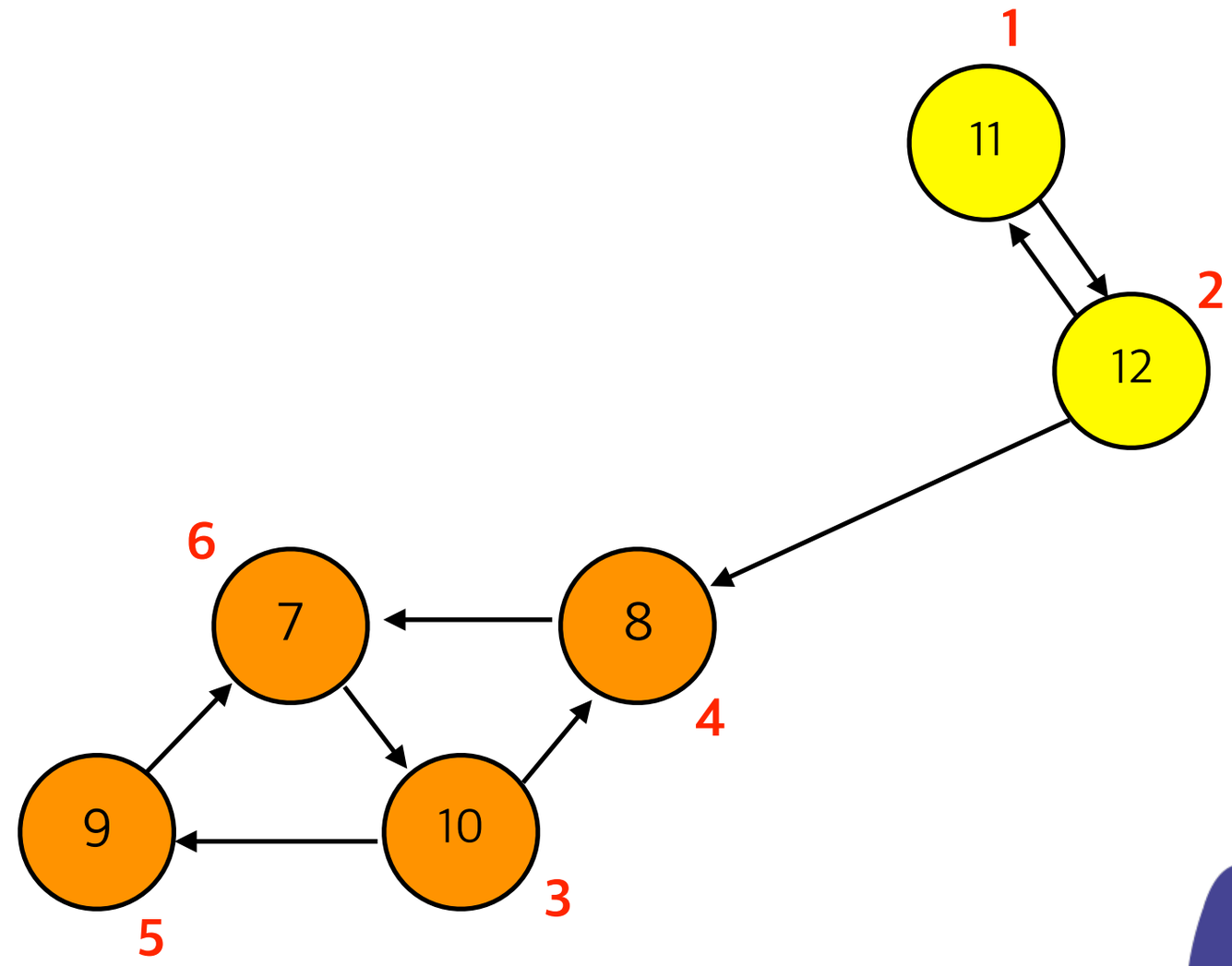
- 이 숫자를 이용해서 방문 순서를 생각하면 ?

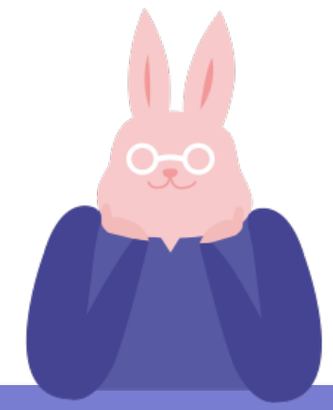




Strongly Connected Component

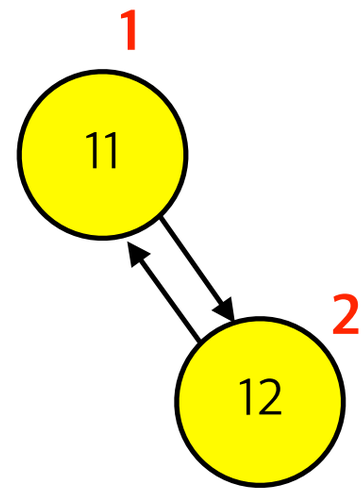
- 이 숫자를 이용해서 방문 순서를 생각하면 ?

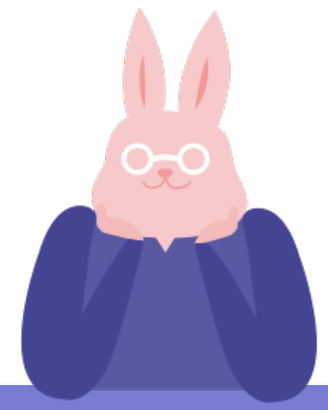




Strongly Connected Component

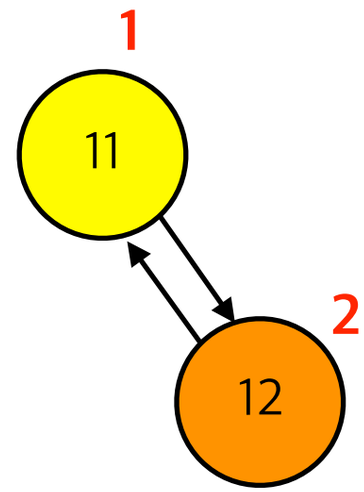
- 이 숫자를 이용해서 방문 순서를 생각하면 ?

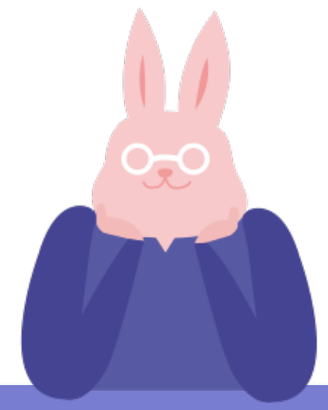




Strongly Connected Component

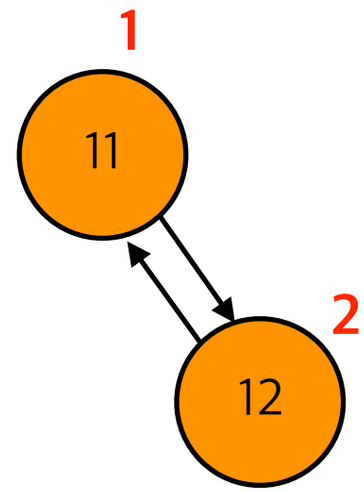
- 이 숫자를 이용해서 방문 순서를 생각하면 ?

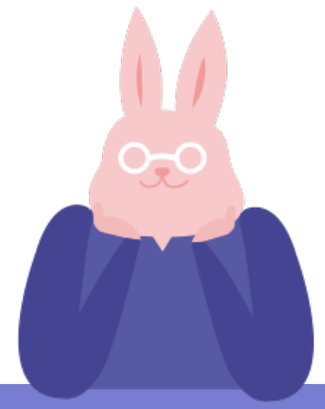




Strongly Connected Component

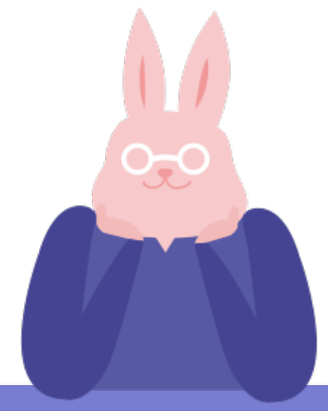
- 이 숫자를 이용해서 방문 순서를 생각하면 ?





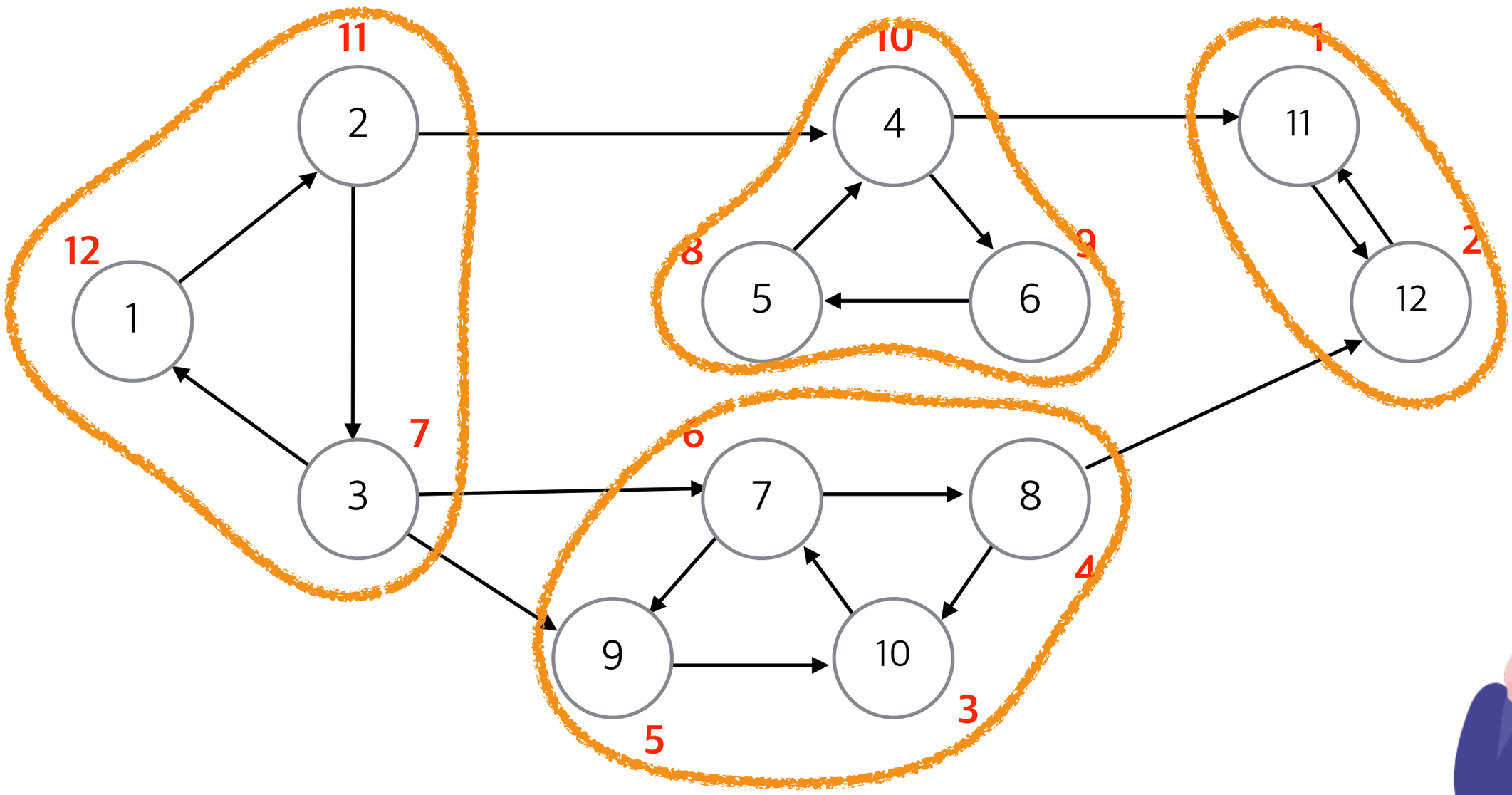
Strongly Connected Component

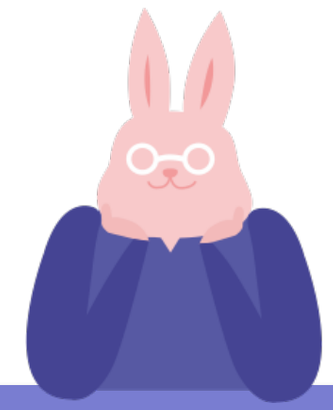
- 이 숫자를 이용해서 방문 순서를 생각하면 ?



Strongly Connected Component

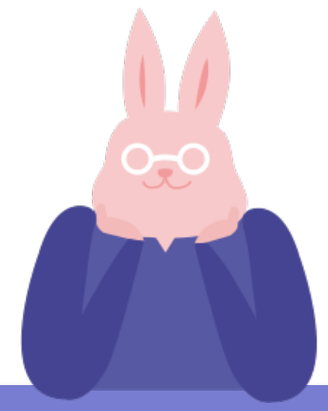
- Summary





남은 것 (할건 아니고…)

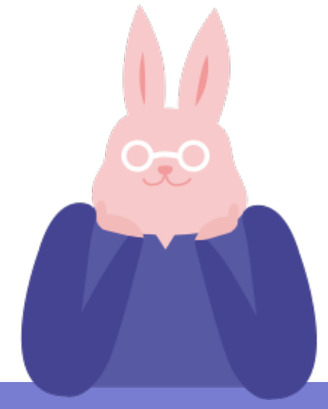
- 지금까지는 쉬운 문제를 다룬 것
  - 쉬운 문제 : N에 대하여 다항시간안에 풀리는 문제



남은 것 (할건 아니고…)

- 지금까지는 쉬운 문제를 다룬 것
  - 쉬운 문제 : N에 대하여 다항시간안에 풀리는 문제

- 어려운 문제의 예제
  - Maximum Independent Set

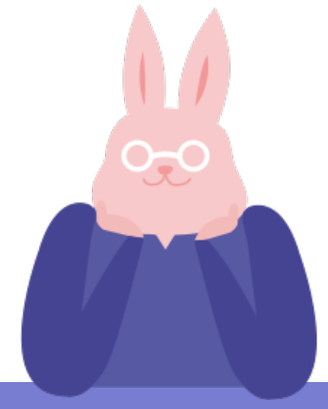


남은 것 (할건 아니고…)

- 지금까지는 쉬운 문제를 다룬 것
  - 쉬운 문제 : N에 대하여 다항시간안에 풀리는 문제

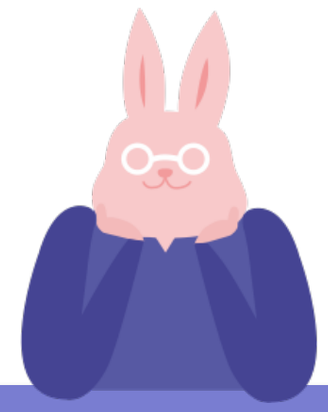
- 어려운 문제의 예제
  - Maximum Independent Set

- **문제가 주어졌을 때 어려운 문제인지 어떻게 판단하는가 ?**



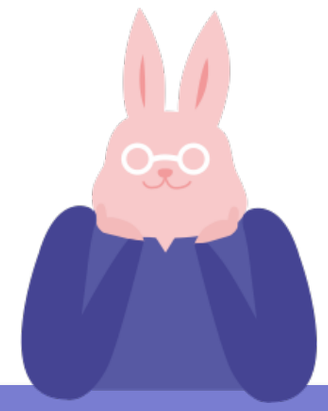
남은 것 (할건 아니고…)

- **Shortest Path**
  - Dijkstra's Algorithm
  - Bellman-Ford Algorithm
  - Floyd Algorithm
- **Minimum Spanning Tree**
  - Prim's algorithm
  - Kruskal's algorithm
- **Maximum Flow**
  - Ford-Fulkerson algorithm
  - Max-flow Min-cut theorem
- **NP-Complete**
  - 3-SAT Problem
  - Independent Set
  - Vertex Cover
  - Maximum Clique
  - …



Software Engineer Interview

- (알고리즘) 문제 풀이 능력을 시험
- 지금까지 접해본 문제들과 크게 다르지 않음



/* elice */

감사합니다!

신현규

E-mail : hyungyu.sh@kaist.ac.kr

Kakao : yougatup