

Day 4 - DFS (Depth First Search)

Stack → Tree(Binary Tree) → DFS ⇒ Data Search, BFS

전, 중, 후위 순회

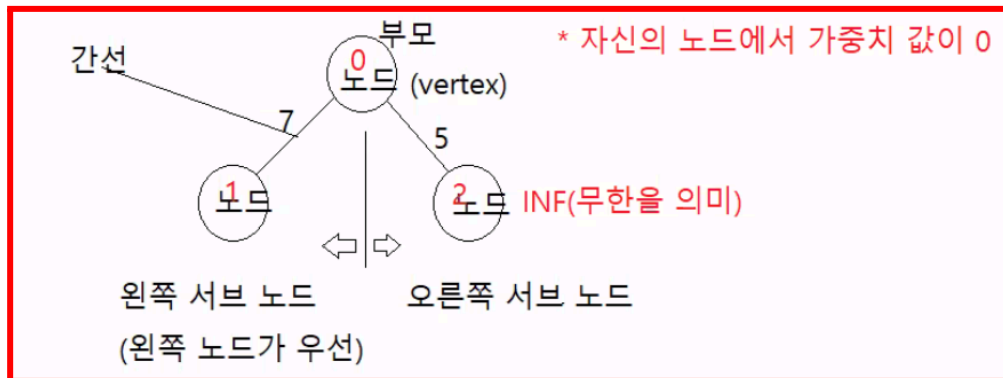
→ Graph

DFS → Stack (FILO)

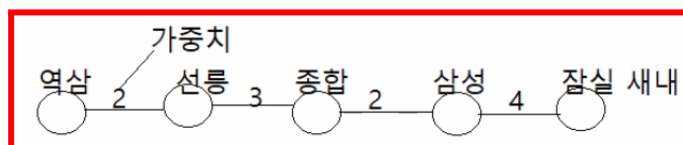
BFS → Queue

DFS (Depth First Search)

- 깊이 우선 탐색
 - 트리에서 깊은 부분을 우선적으로 탐색하는 알고리즘
- **Binary Tree:**

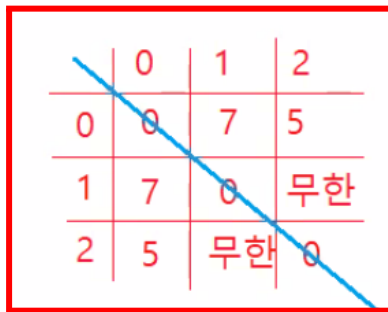


- 두가지 방식으로 표현한다
 1. 인접 행렬 → 2차원 배열로 연결된 관계를 이야기 한다.
 2. 인접 리스트 → 리스트로 연결된 관계를 표현하는 방식



- 가중치가 있고 그래프가 그려져 있으면 최단 거리 찾는 문제이다.
- 가중치가 없으면 특정 데이터를 몇번만에 찾는지, 어디있는지 알아내는 문제이다.
- 자신의 노드에서는 가중치 값이 0
- INF (무한을 의미)

1. 인접 행렬 (2차원 배열로 표시)



	0	1	2
0	0	7	5
1	7	0	무한
2	5	무한	0

- 자기 자신은 0 이다.

```
graph = [  
    [0, 7, 5],  
    [7, 0, INF],  
    [5, INF, 0]  
]
```

노드 진입

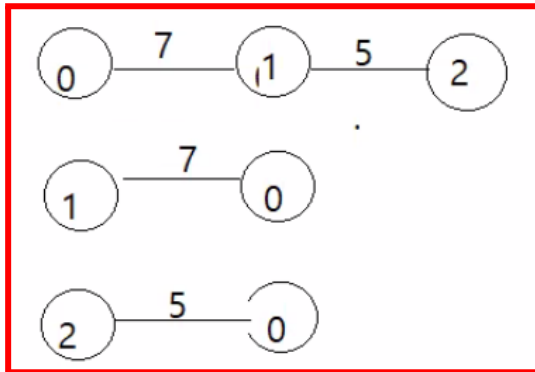
- 노드에 몇개가 들어오는지

노드 진출

- 노드에서 몇개가 나가는지

2. 인접 리스트 (Linked List)

- 직접적으로 연결된것만 따라갈 수 있다.



- $0 \rightarrow 1 \rightarrow 2$ 를 거쳐서 이동한다. (트리)
- 다른 언어에서는 Array를 파이썬에서는 리스트 자료형으로 표현한다.
- 파이썬은 인접 행렬을 리스트로 구현
 - 리스트로 구현 \rightarrow append()
- 파이썬, 자바에서 클래스로 선언 되는 자료형
 - String
 - Array

상식

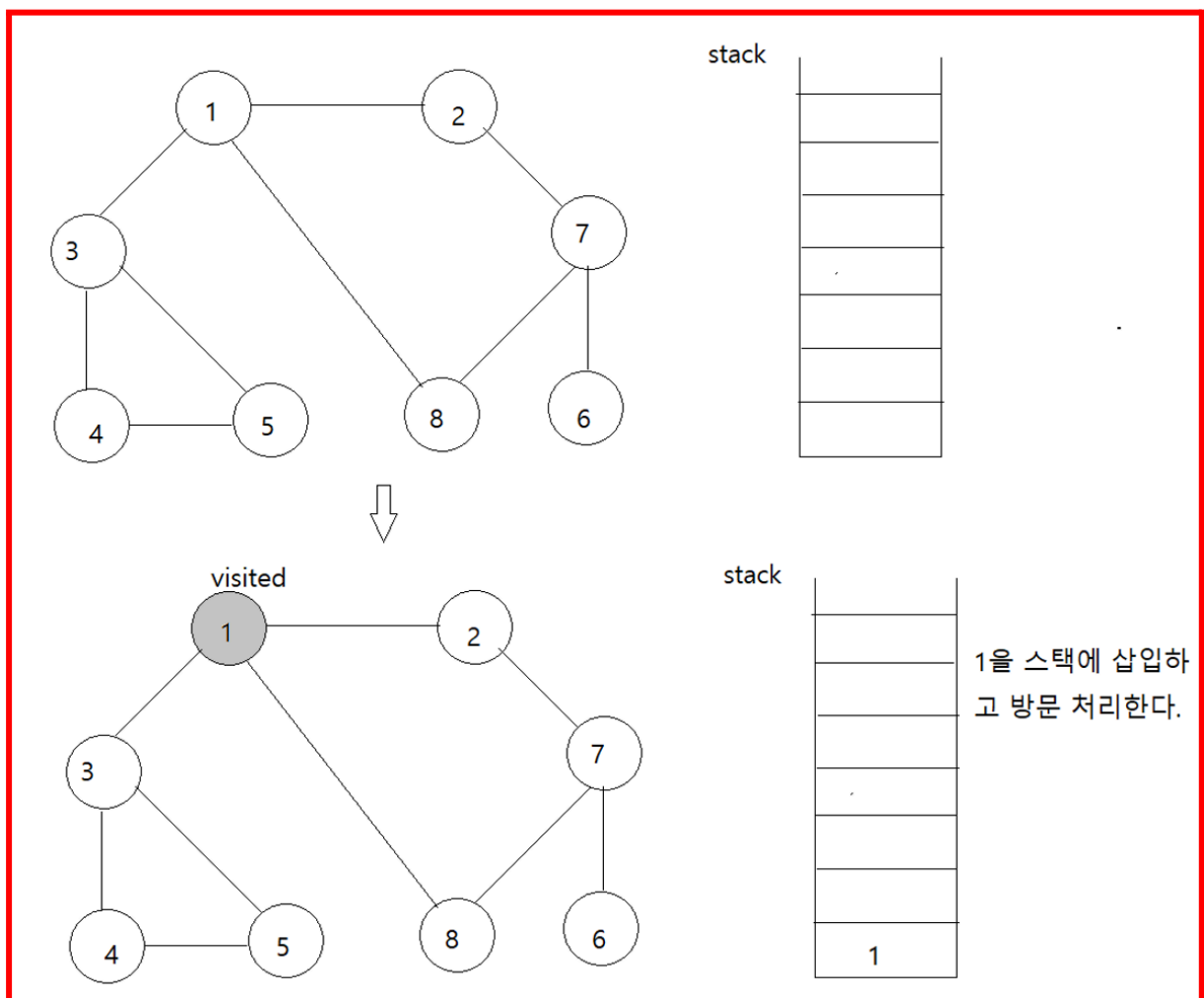
- Test t
 - Test 클래스의 변수 (선언자 없이는 객체가 될 수 없다)
- String str
 - 객체
- Array arr
 - 객체

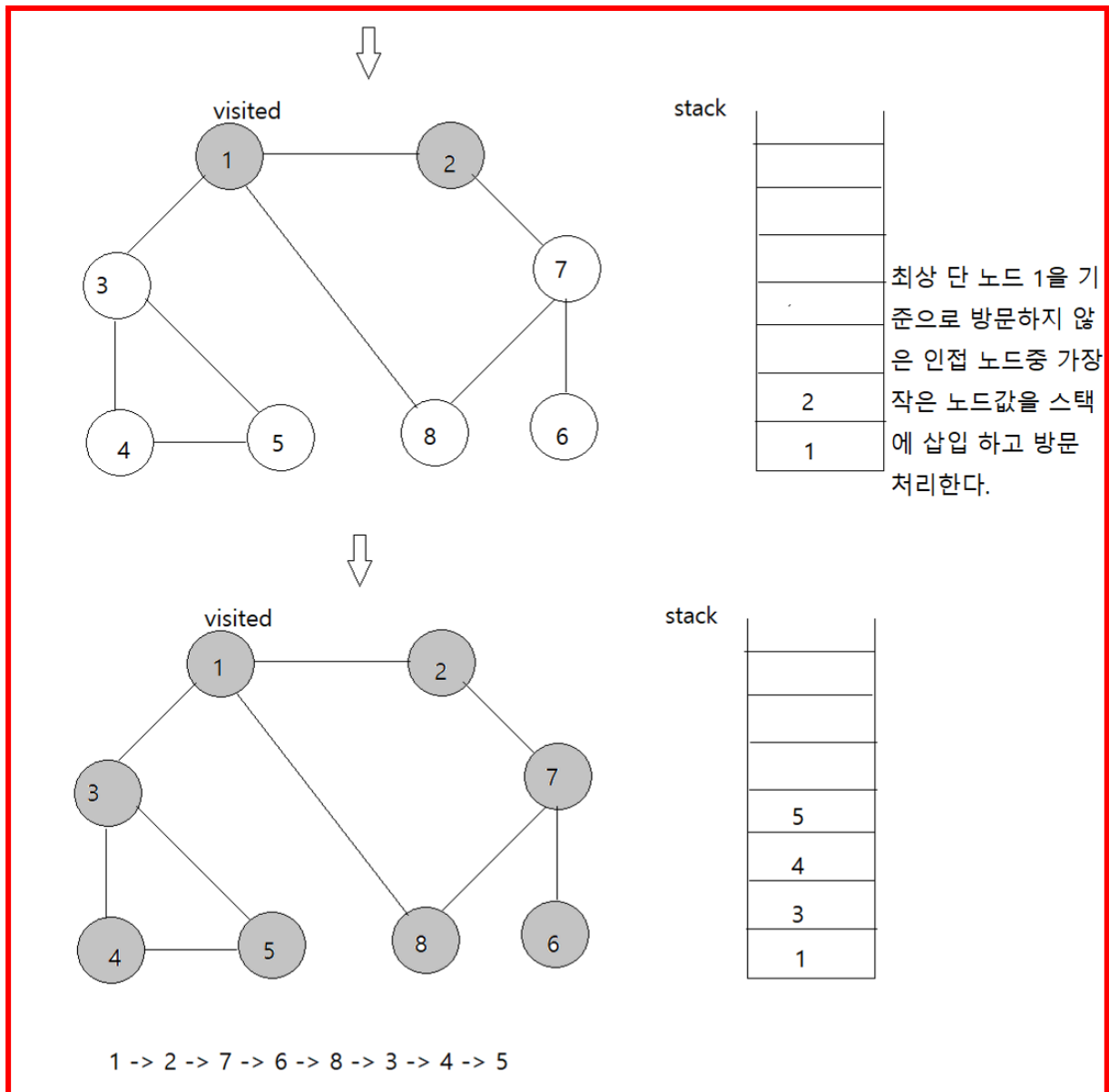
코딩 예제:

<https://colab.research.google.com/drive/1LZf2Cd10ghknsU3CIlyNR8UhCxs49IWX>

DFS(깊이 우선 탐색)

1. 빈 스택 생성
2. 탐색 시작 노드를 스택에 삽입하고 방문 처리한다.
 - a. 아무런 조건이 없으면 데이터가 가장 적은 노트에서 시작한다.
 - b. 스택에 append() 후 방문 처리한다.
 - c. 카운트가 늘어나고 줄어드는거기때문에 민감하다
3. 방문 한 노드를 기준으로 방문하지 않은 인접 노드가 있으면 그 인접 노드를 스택에 삽입하고 방문 처리한다. (방문하지 않은 노드가 없으면 스택에서 최상단 노드를 꺼낸다)
 - a. visited는 현재 노트에 서 있다는걸 의미한다.
4. 위의 2번 과정을 더 이상 수행할 수 없을 때까지 반복 한다.





DFS 함수 만들기:

https://colab.research.google.com/drive/1aOepLyDdntUSgHZkaj99q20zex1naa1s#scrollTo=_1BR5idjaWTh

DFS는 동작 방식은 스택에 의해서 동작되고, 구현은 재귀 함수를 이용하여 구현한다.