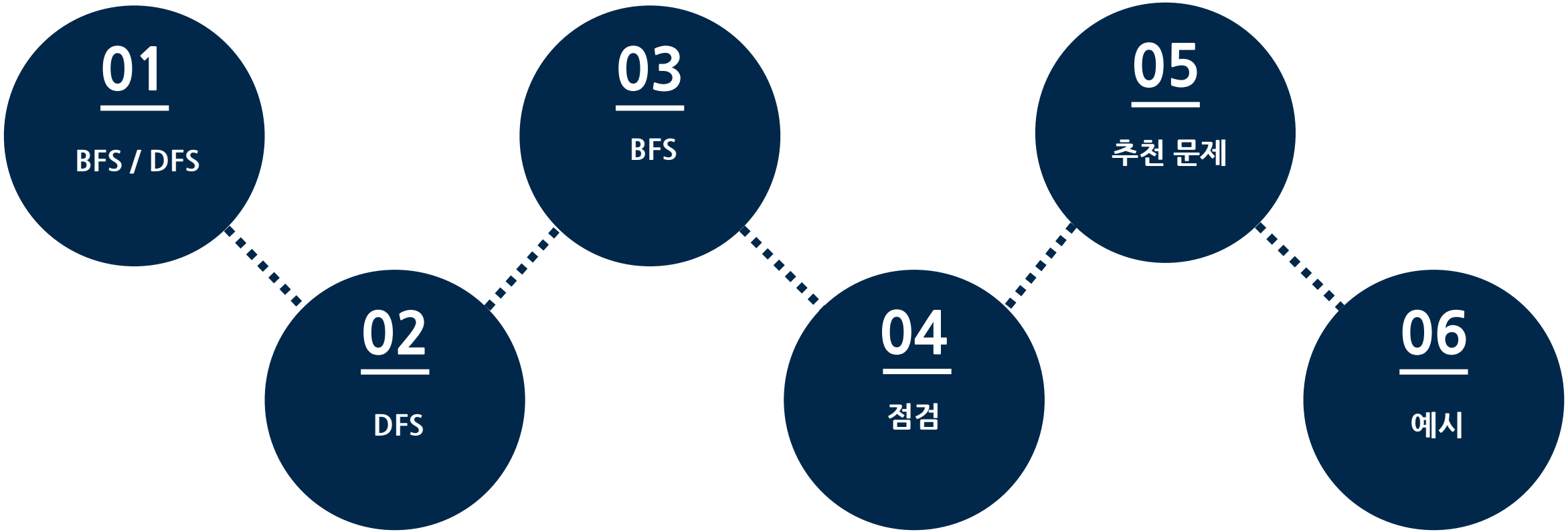
알고리즘 스터디 1주차 발표

BFS / DFS

윤민수

INDEX



BFS / DFS 뜻

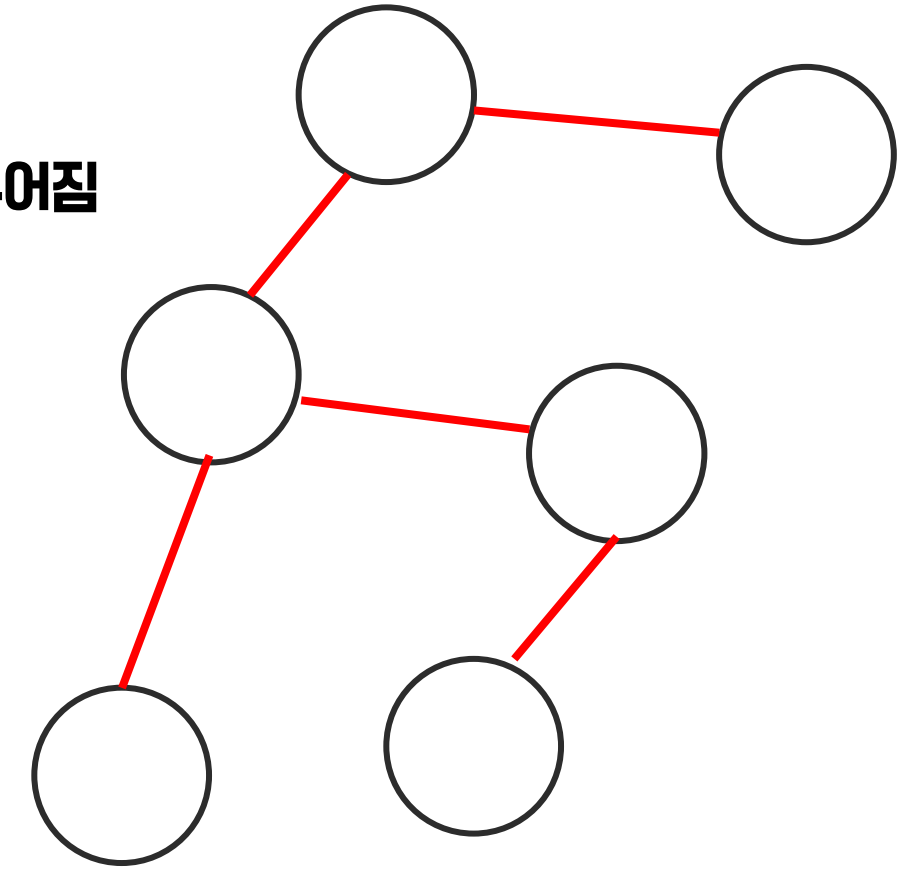
BFS: Breadth-First Search (너비우선탐색)

DFS: Depth-First Search (깊이우선탐색)

※ 그래프를 탐색하는 방법

그래프

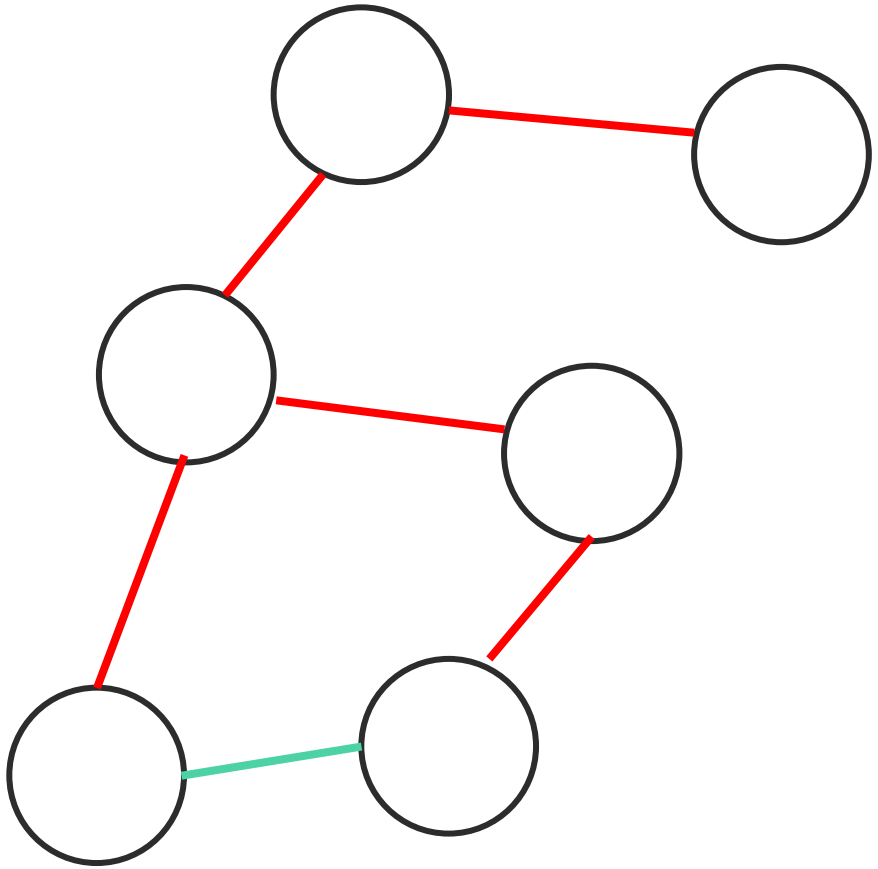
그래프: 정점과 간선으로 이루어짐



: Vertex (정점)

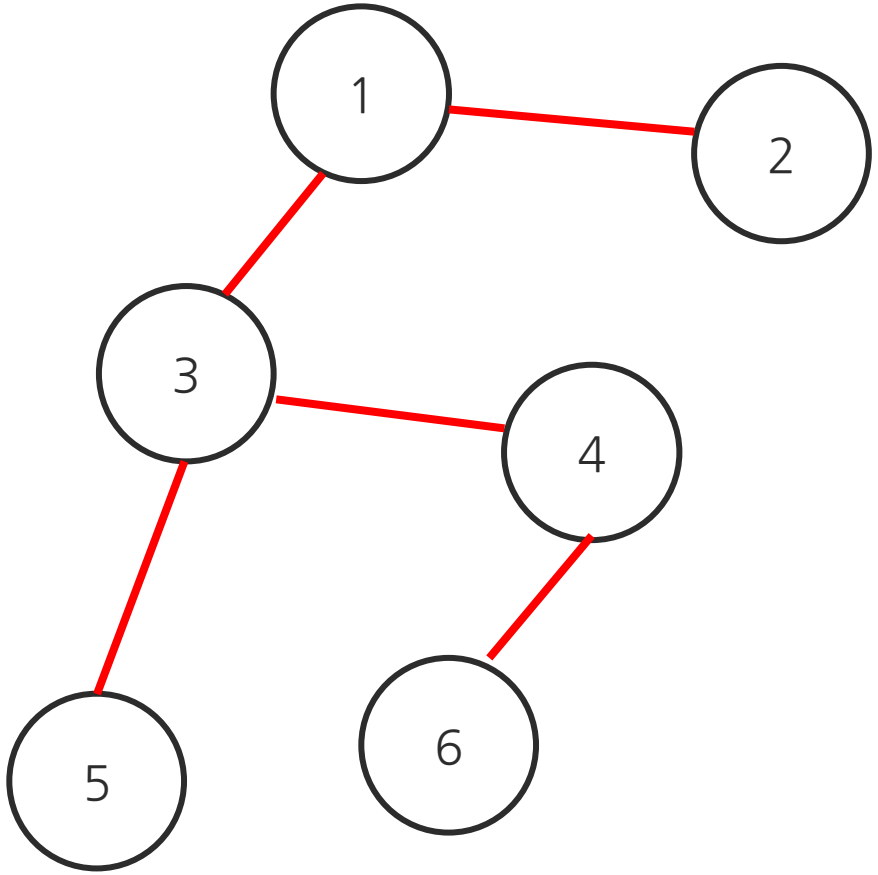
: Edge (간선)

그래프



Cycle 형성 가능

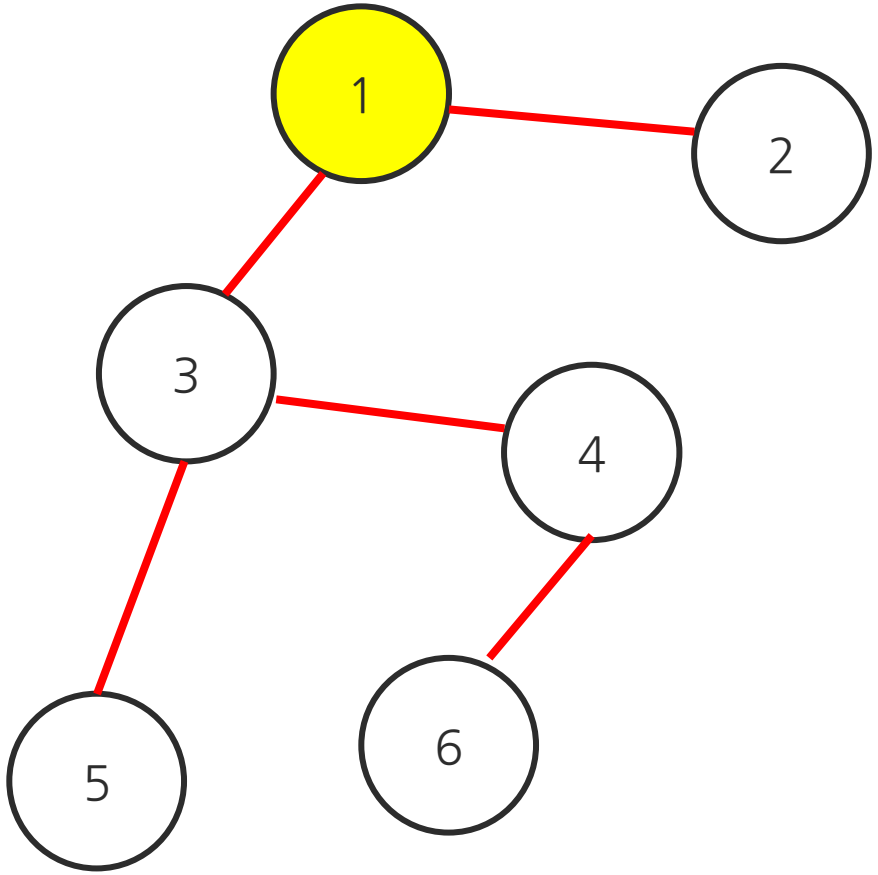
DFS



DFS: 한번 진행한 방향으로 쭉 감

1번부터 시작한다고 가정

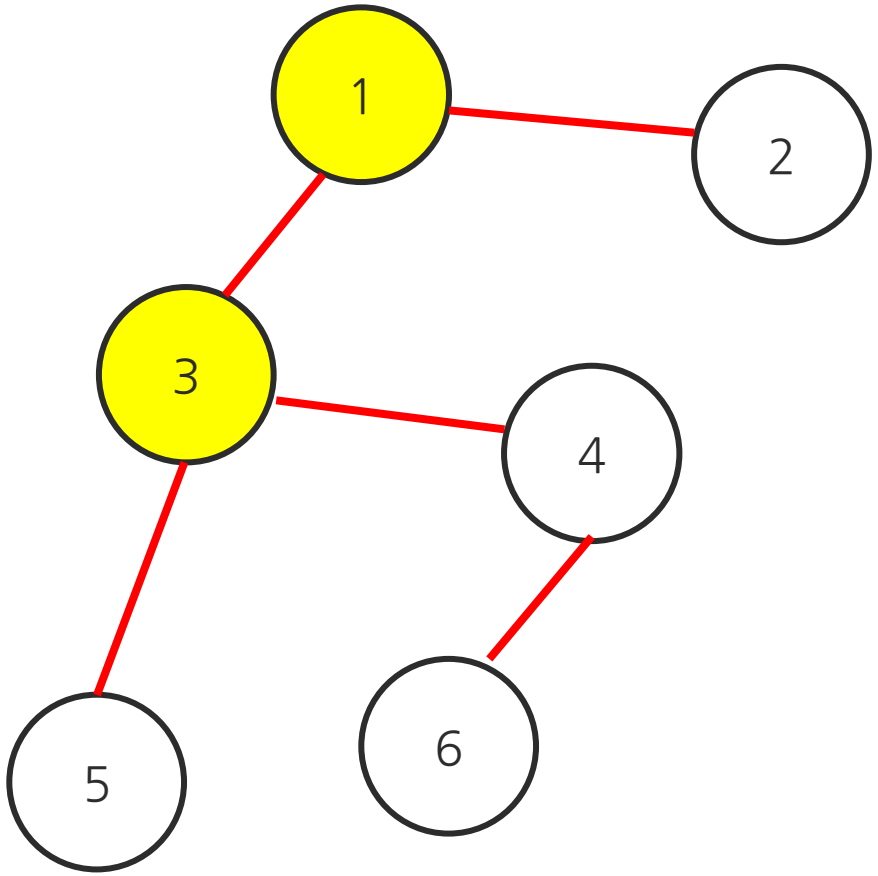
DFS



DFS: 한번 진행한 방향으로 쭉 감

순서: 1 ->

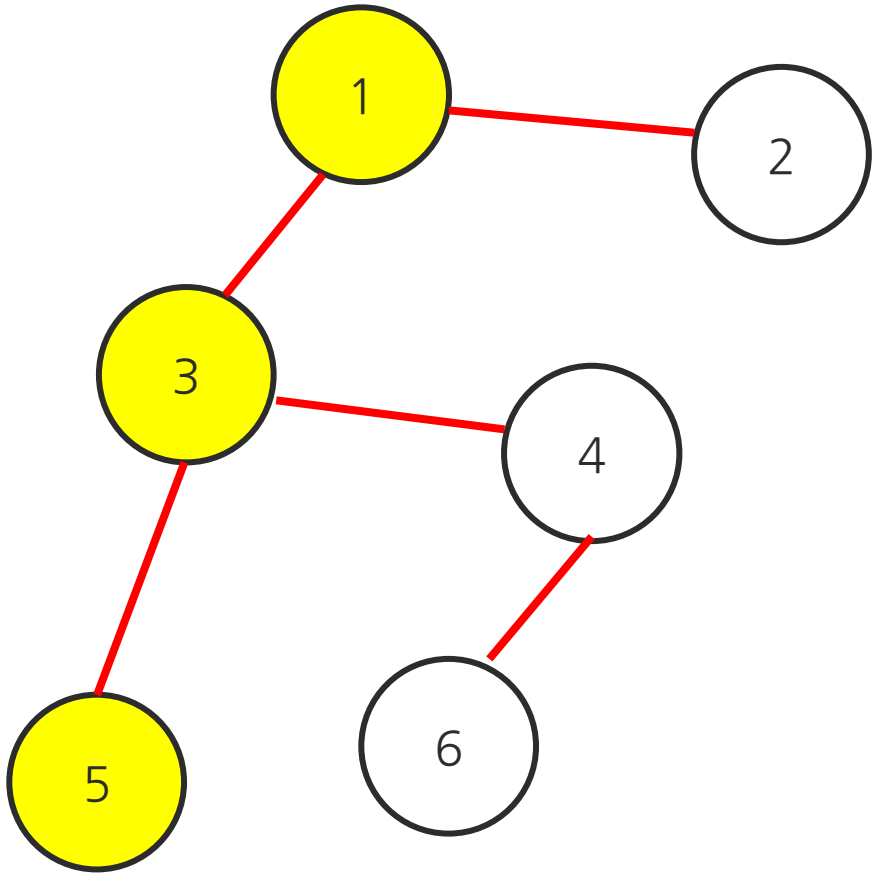
DFS



DFS: 한번 진행한 방향으로 쭉 감

순서: 1 -> 3 ->

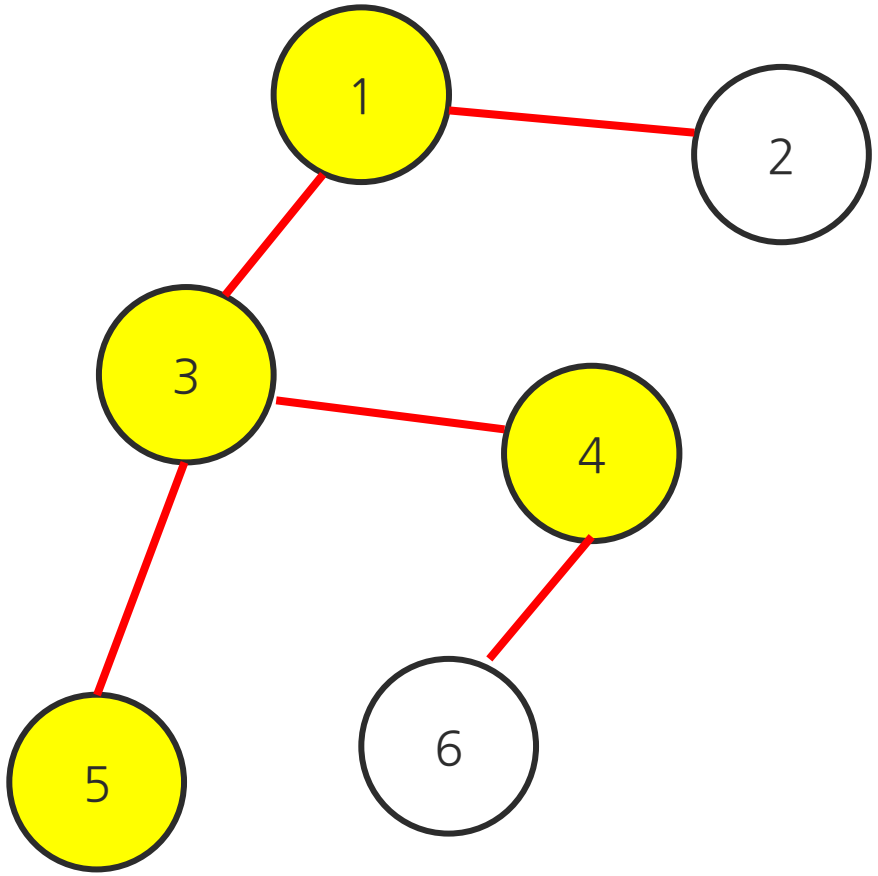
DFS



DFS: 한번 진행한 방향으로 쭉 감

순서: 1 -> 3 -> 5

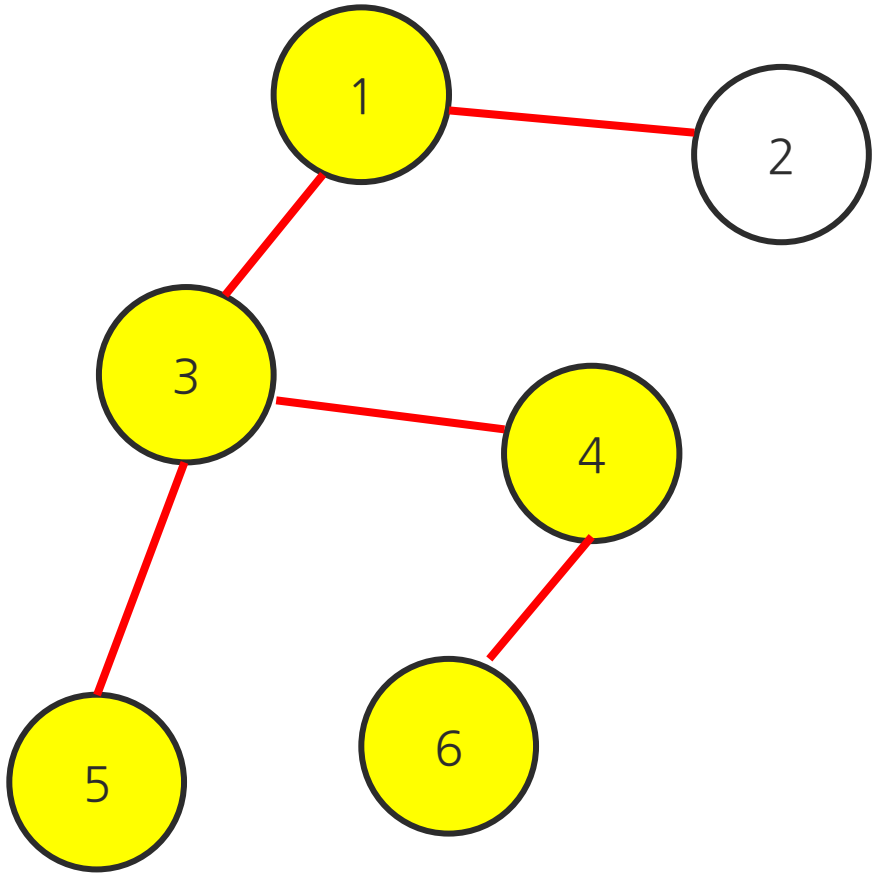
DFS



DFS: 한번 진행한 방향으로 쭉 감

순서: 1 -> 3 -> 5 -> 4

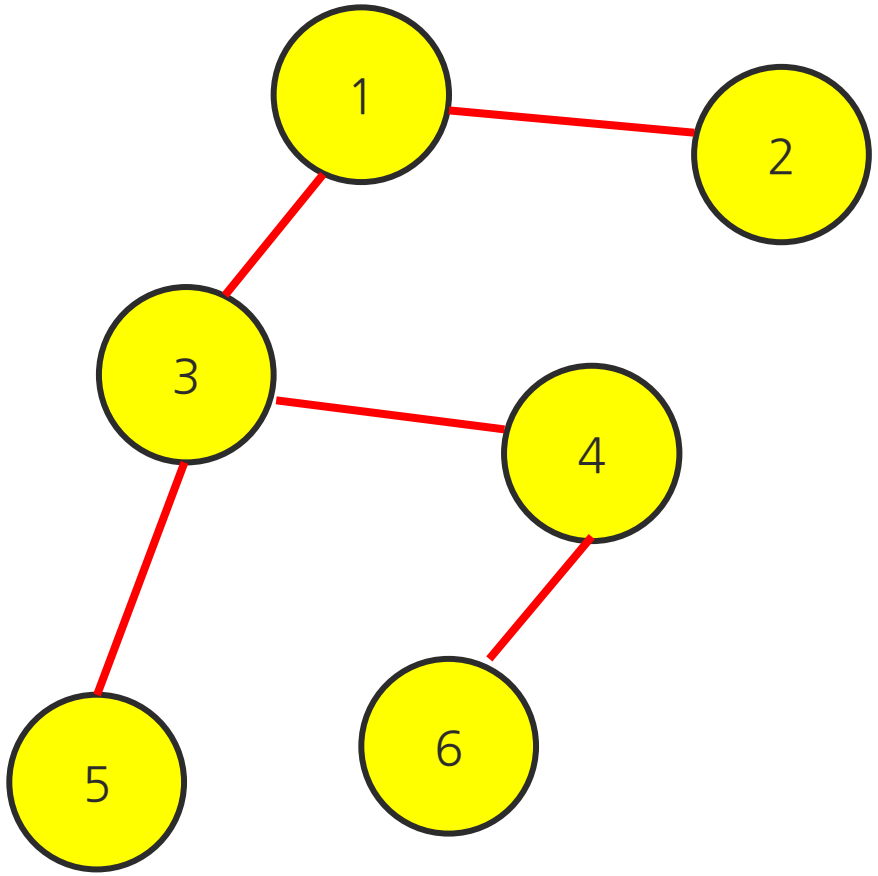
DFS



DFS: 한번 진행한 방향으로 쭉 감

순서: 1 -> 3 -> 5 -> 4 -> 6

DFS



DFS: 한번 진행한 방향으로 쭉 감

순서: 1 -> 3 -> 5 -> 4 -> 6 -> 2

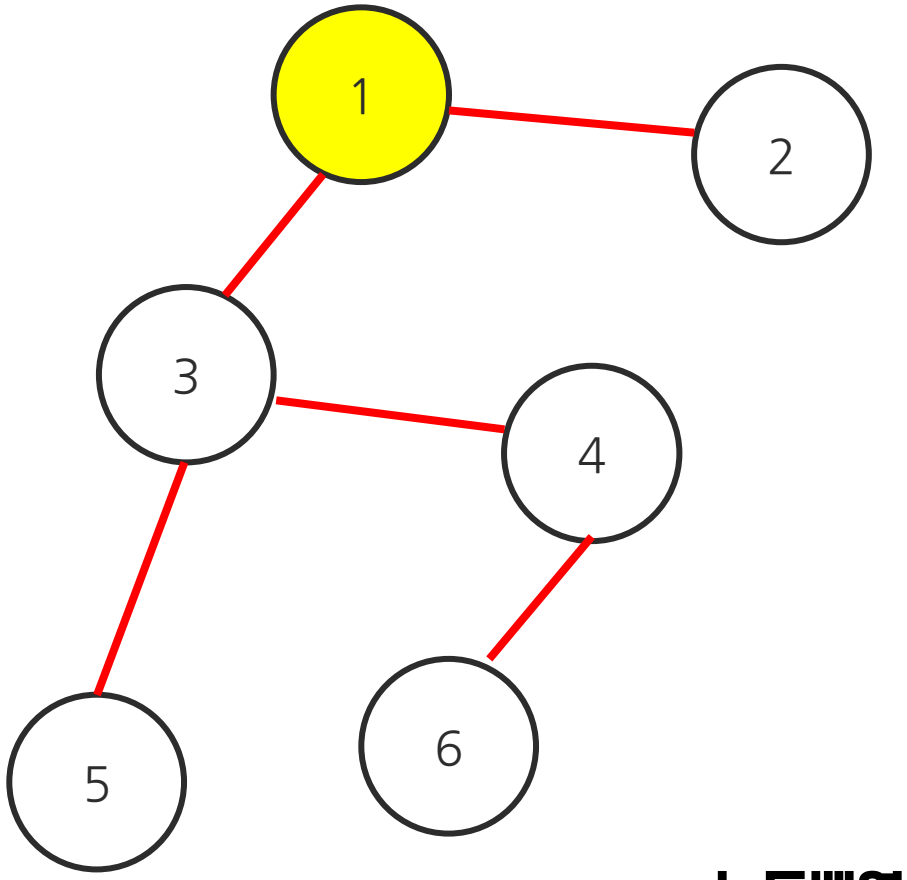
02. DFS

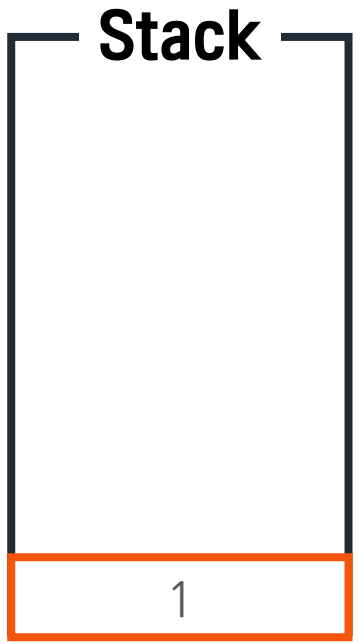
어떻게 구현???

재귀를 활용해서

```
def Vertex방문(노드배열, 현재위치, 간선배열, 비용):
    if 노드배열[현재위치] >= 0: #현재 위치에 방문 했었나???
        return

    노드배열[현재위치] = 비용
    for 다음방문노드 in 간선배열[현재위치]:
        Vertex방문(노드배열, 다음방문노드, 간선정보배열, 비용 + 1)

#main
노드배열 = [-1] * 노드갯수
현재위치 = 1
간선배열 #입력으로 받음

Vertex방문(노드배열, 현재위치, 간선배열, 0)
```

DFS





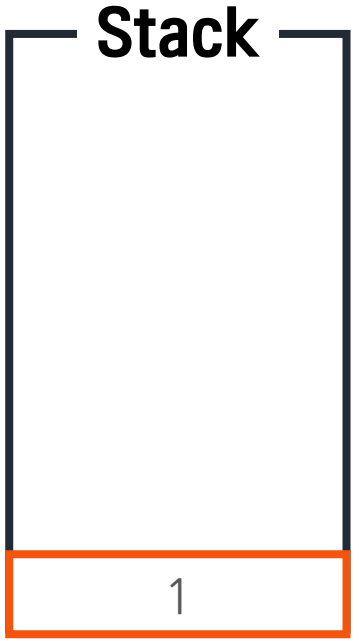
노드배열: [0, -1, -1, -1, -1, -1]

간선배열: [[3,2], [1], [1,5,4], [3,6], [3], [4]]

순서: 1 ->

DFS


```
def Vertex방문(노드배열, 현재위치, 간선배열, 비용):
    if 노드배열[현재위치] >= 0: #현재 위치에 방문 했었나???
        return

    노드배열[현재위치] = 비용
    for 다음방문노드 in 간선배열[현재위치]:
        Vertex방문(노드배열, 다음방문노드, 간선정보배열, 비용 + 1)

#main
노드배열 = [-1] * 노드갯수
현재위치 = 1
간선배열 #입력으로 받음

Vertex방문(노드배열, 현재위치, 간선배열, 0)
```

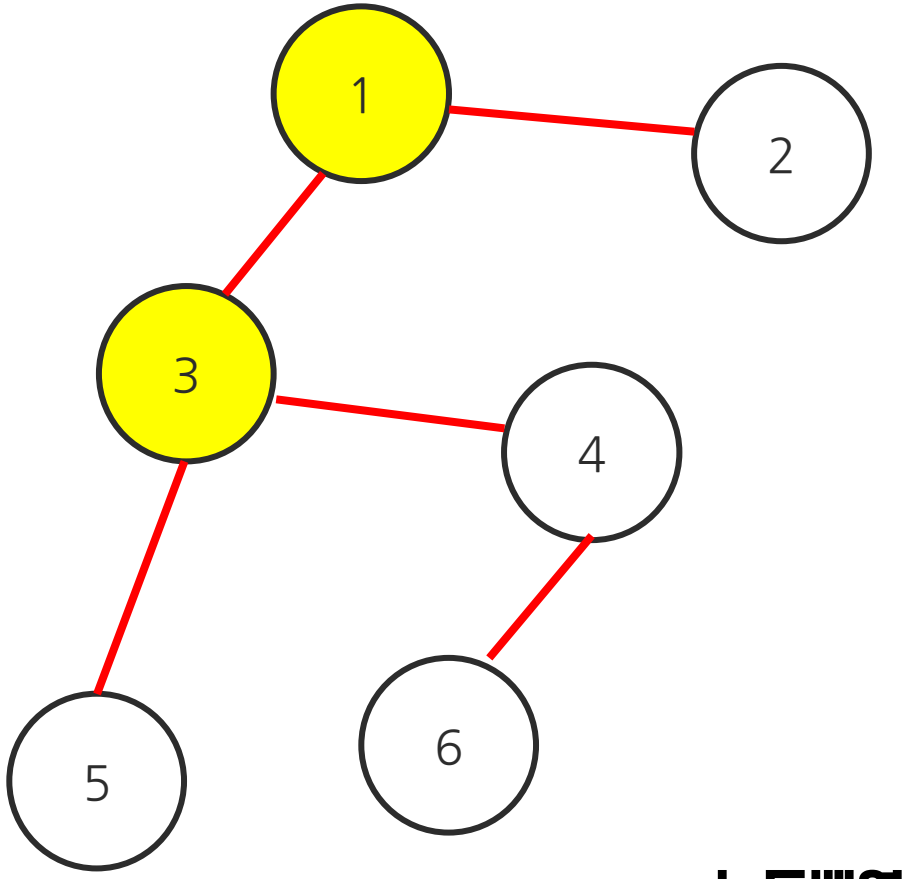


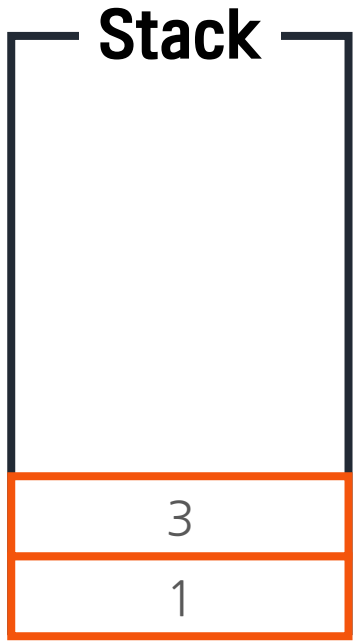
순서: 1 ->

노드배열: [0, -1, -1, -1, -1, -1]

간선배열: [[3,2], [1], [1,5,4], [3,6], [3], [4]]

DFS



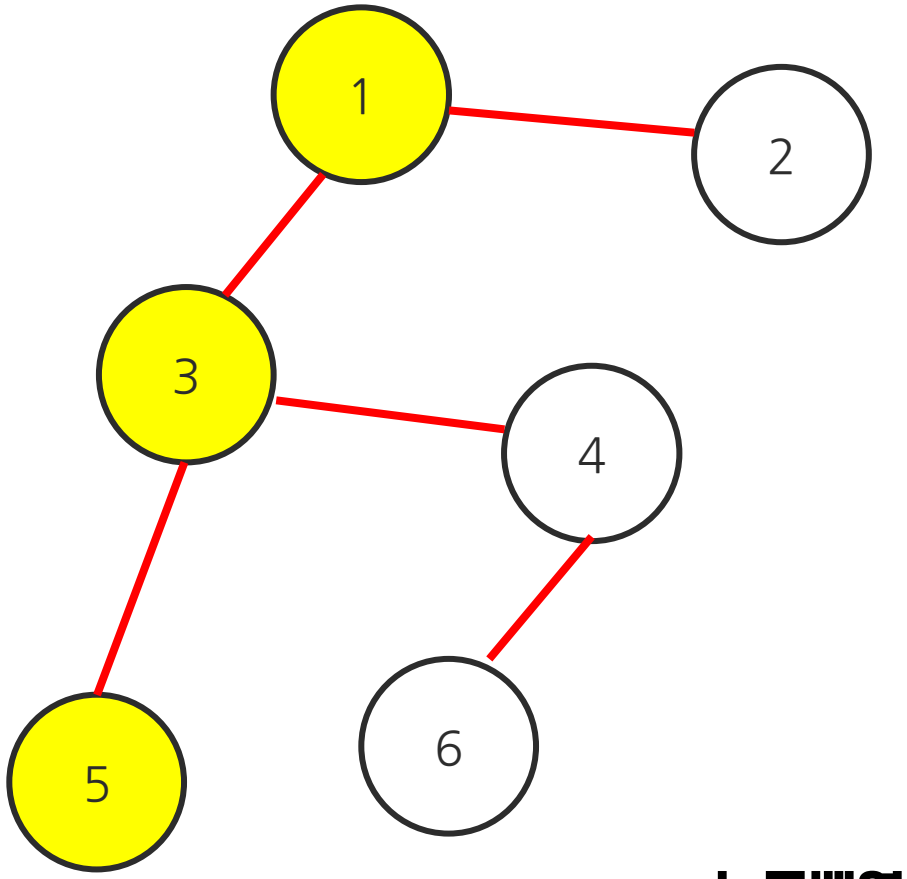


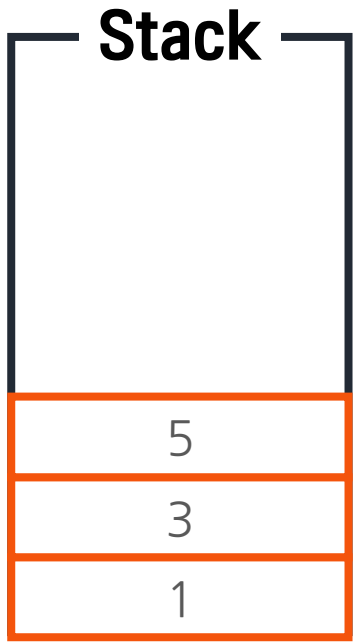
노드배열: [0, -1, 1, -1, -1, -1]

간선배열: [[3,2], [1], [1,5,4], [3,6], [3], [4]]

순서: 1 -> 3 ->

DFS



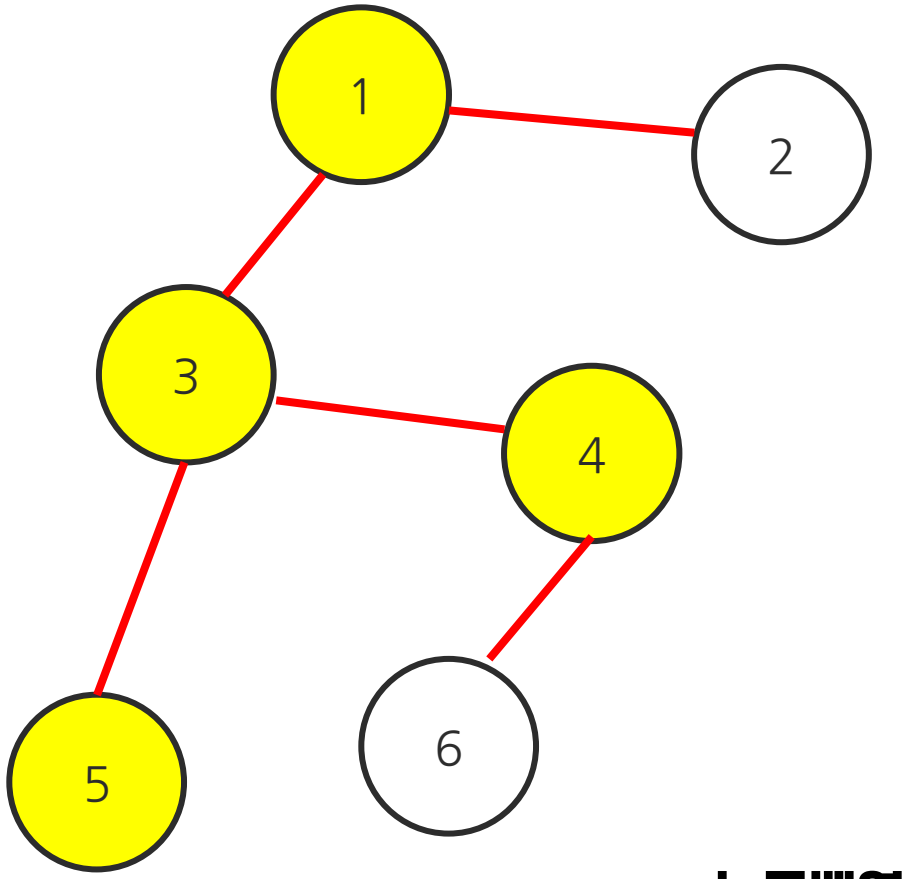


노드배열: [0, -1, 1, -1, 2, -1]

간선배열: [[3,2], [1], [1,5,4], [3,6], [3], [4]]

순서: 1 -> 3 -> 5

DFS

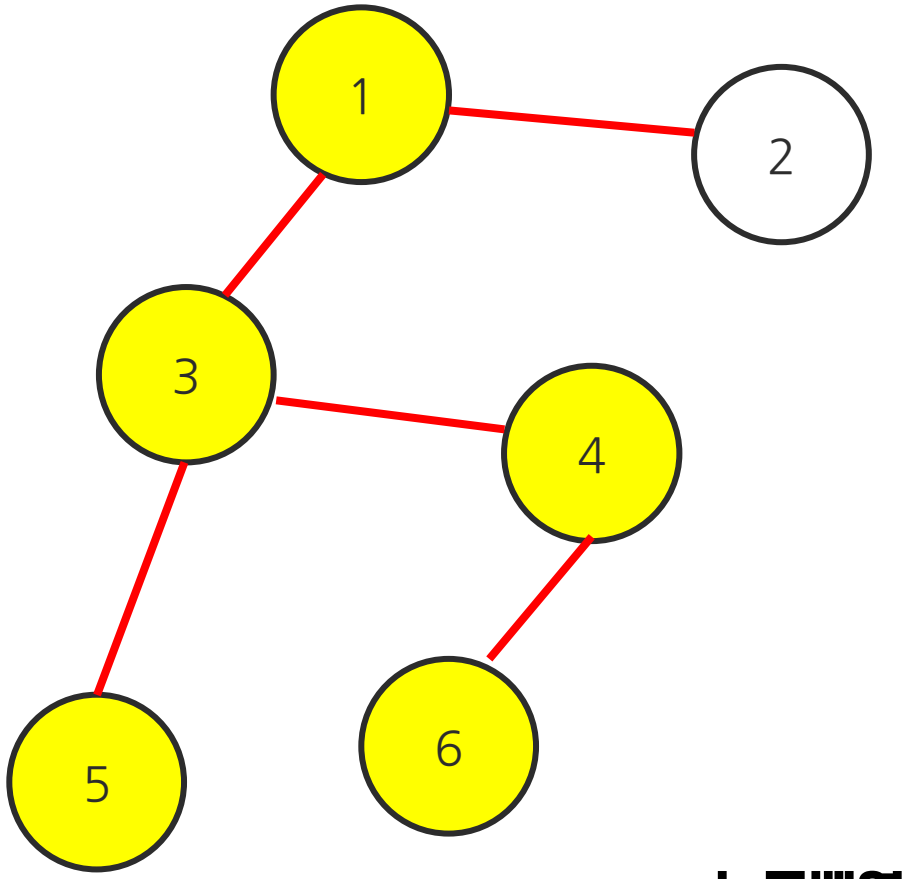


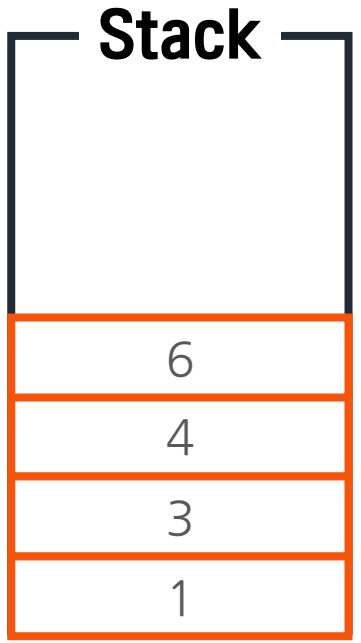
노드배열: [0, -1, 1, 2, 2, -1]

간선배열: [[3,2], [1], [1,5,4], [3,6], [3], [4]]

순서: 1 -> 3 -> 5 -> 4

DFS



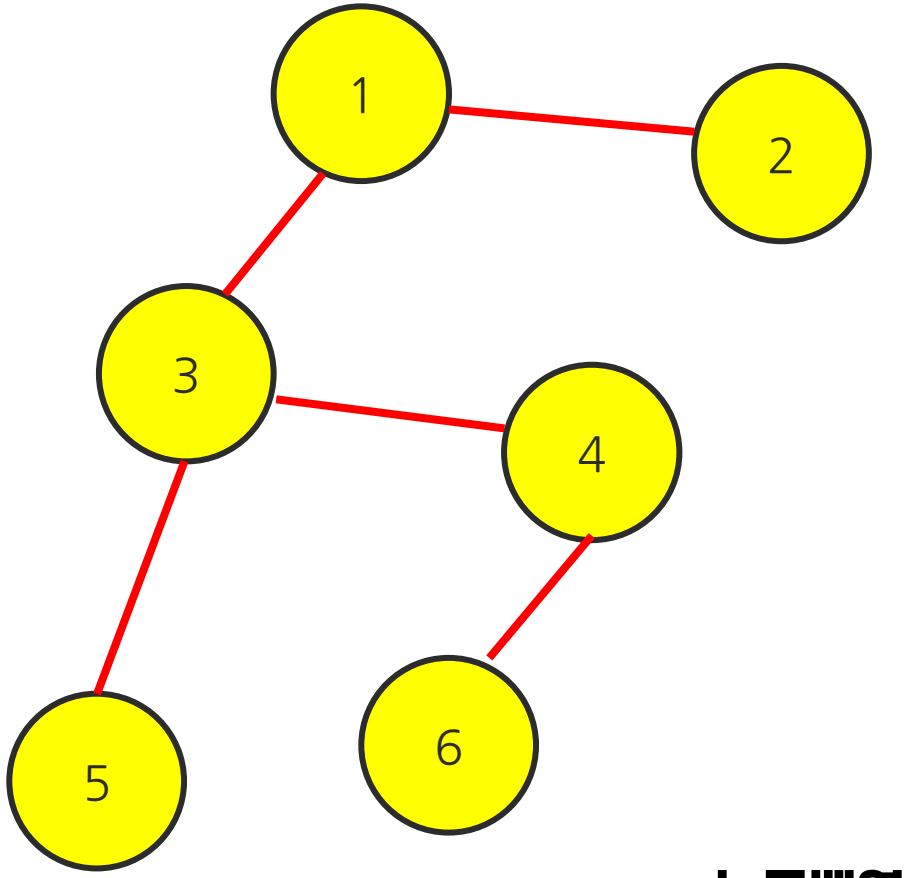


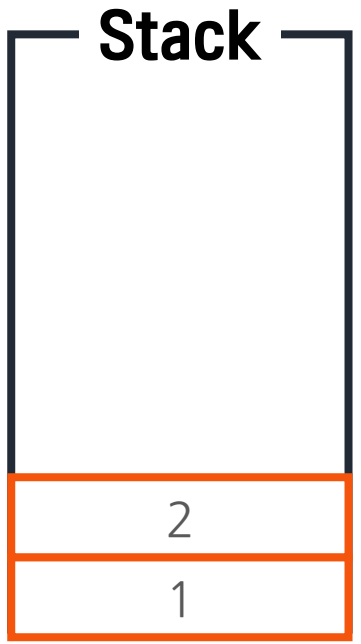
노드배열: [0, -1, 1, 2, 2, 3]

간선배열: [[3,2], [1], [1,5,4], [3,6], [3], [4]]

순서: 1 -> 3 -> 5 -> 4 -> 6

DFS





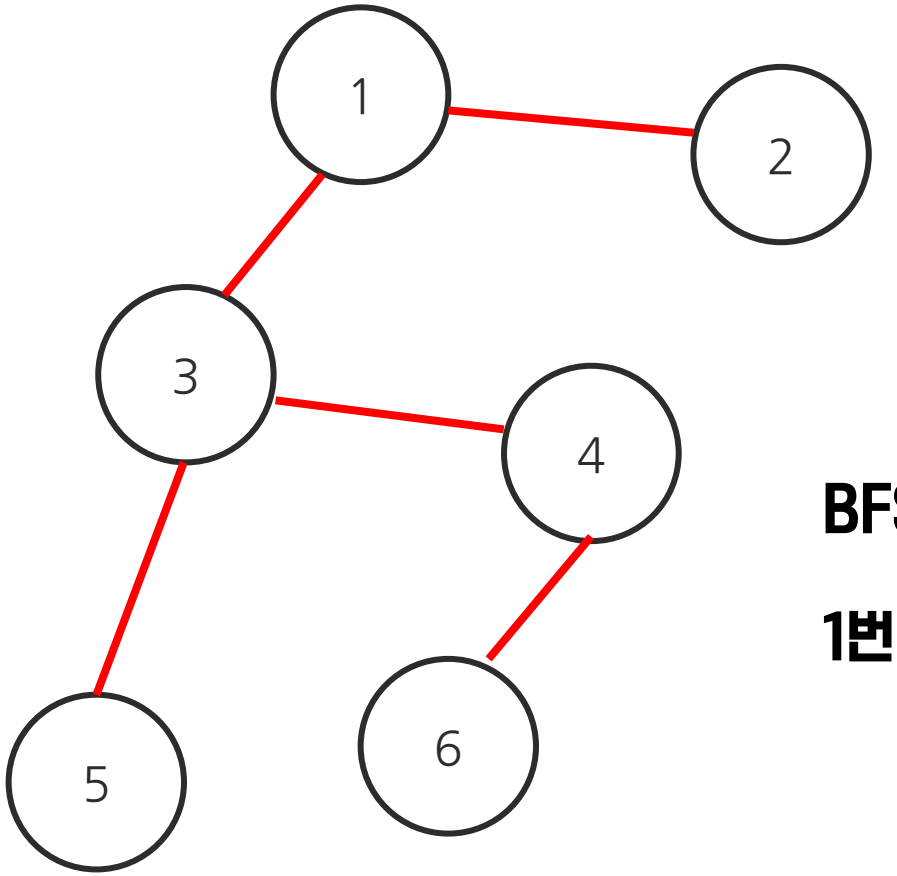
노드배열: [0, 1, 1, 2, 2, 3]

간선배열: [[3,2], [1], [1,5,4], [3,6], [3], [4]]

순서: 1 -> 3 -> 5 -> 4 -> 6 -> 2

03. BFS

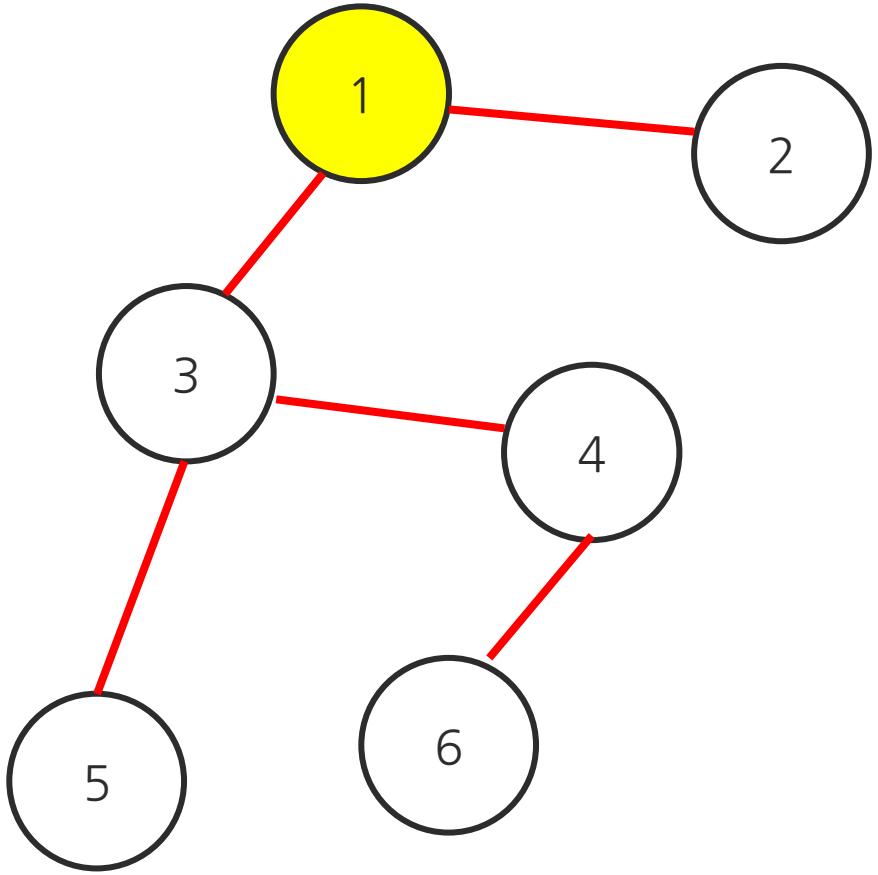
BFS



BFS: 현재 정점에서 연결된 정점을 방문하고 다음으로 넘어감

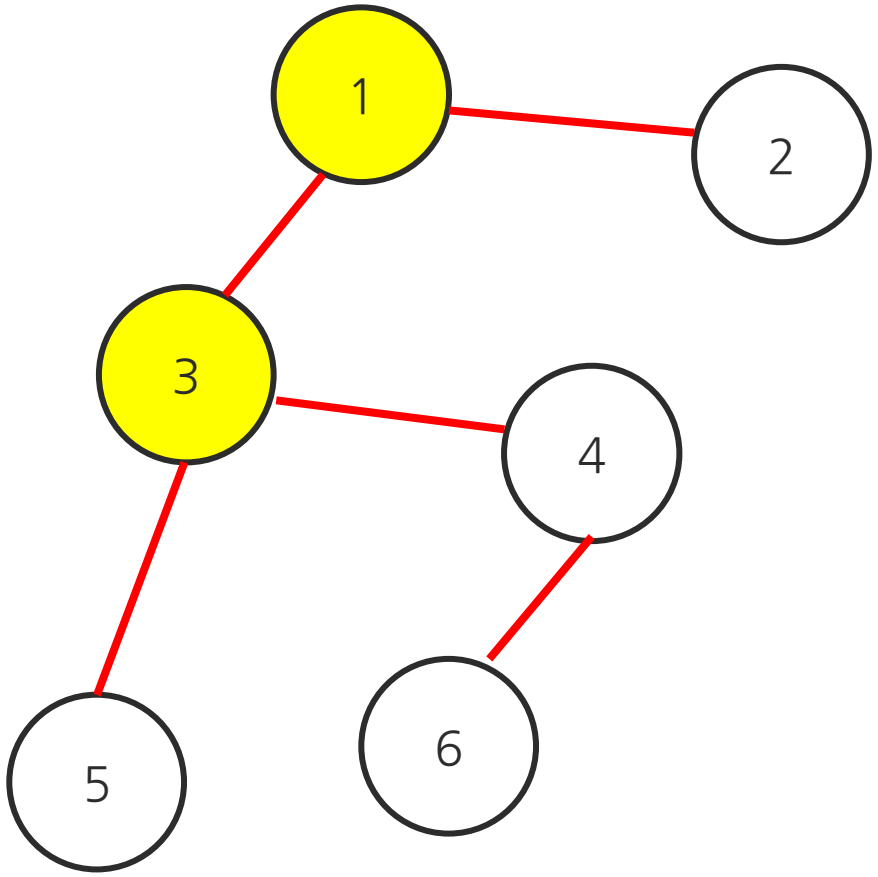
1번부터 시작한다고 가정

BFS



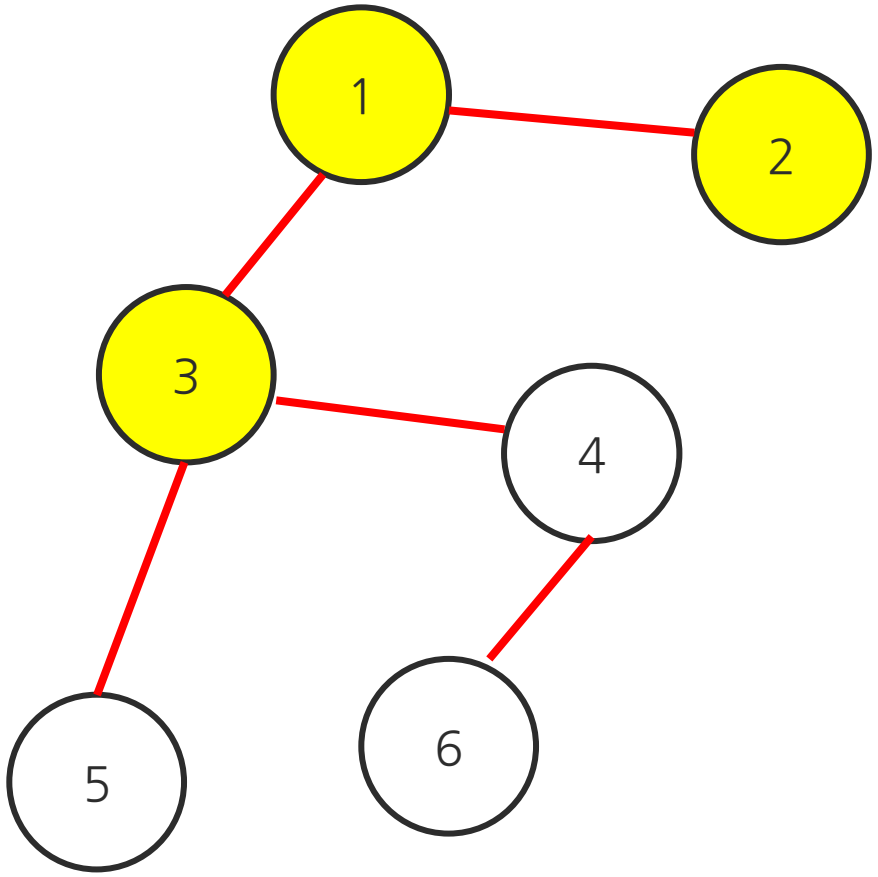
순서: 1 ->

BFS



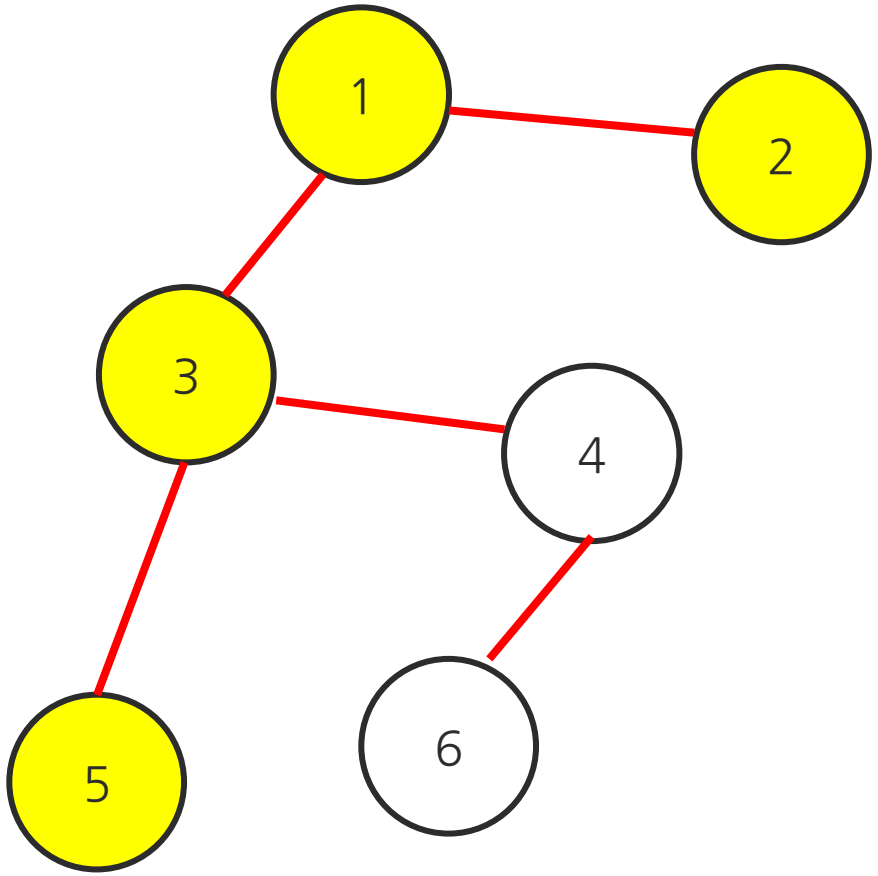
순서: 1 -> 3 ->

BFS



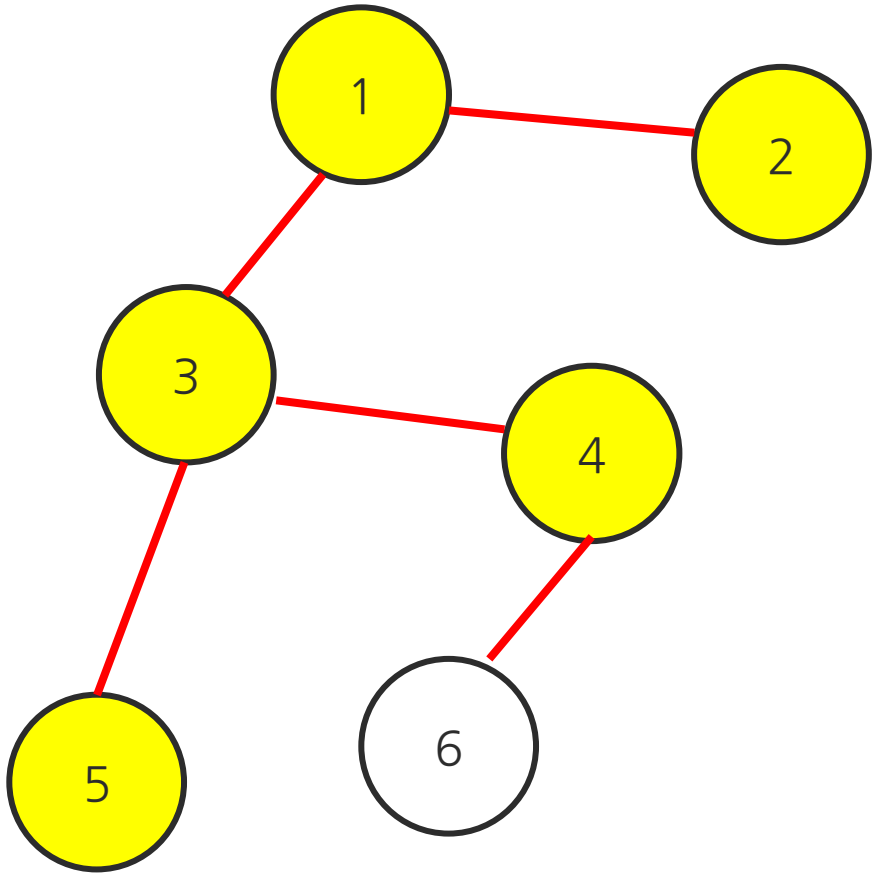
순서: 1 -> 3 -> 2 ->

BFS



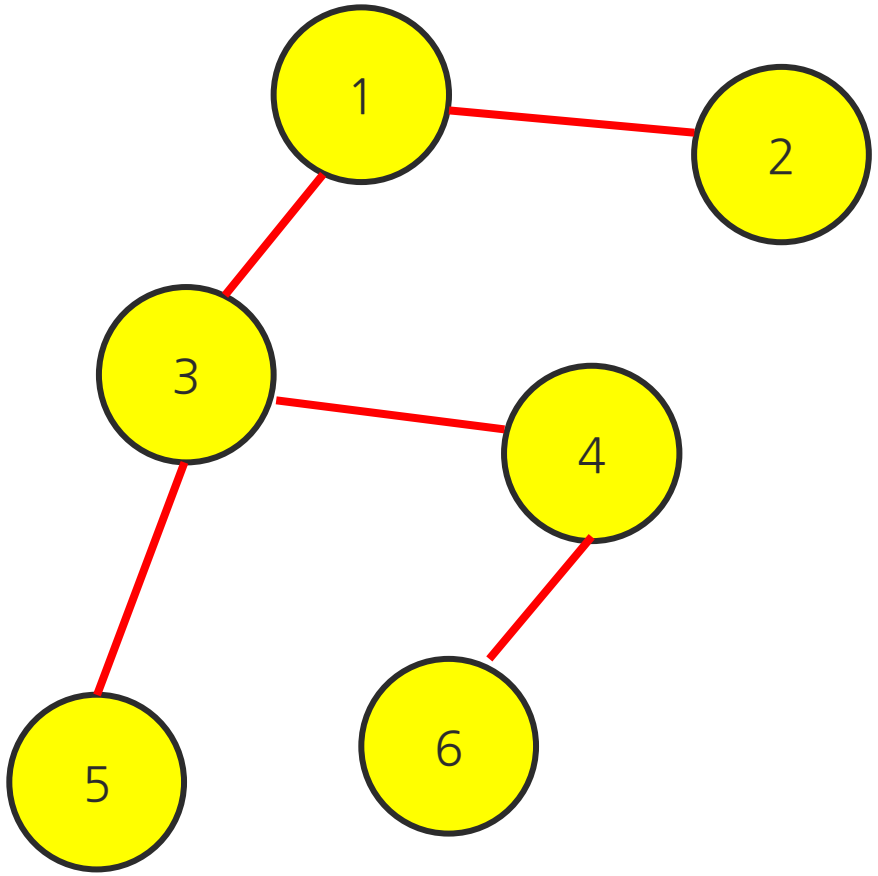
순서: 1 -> 3 -> 2 -> 5 ->

BFS



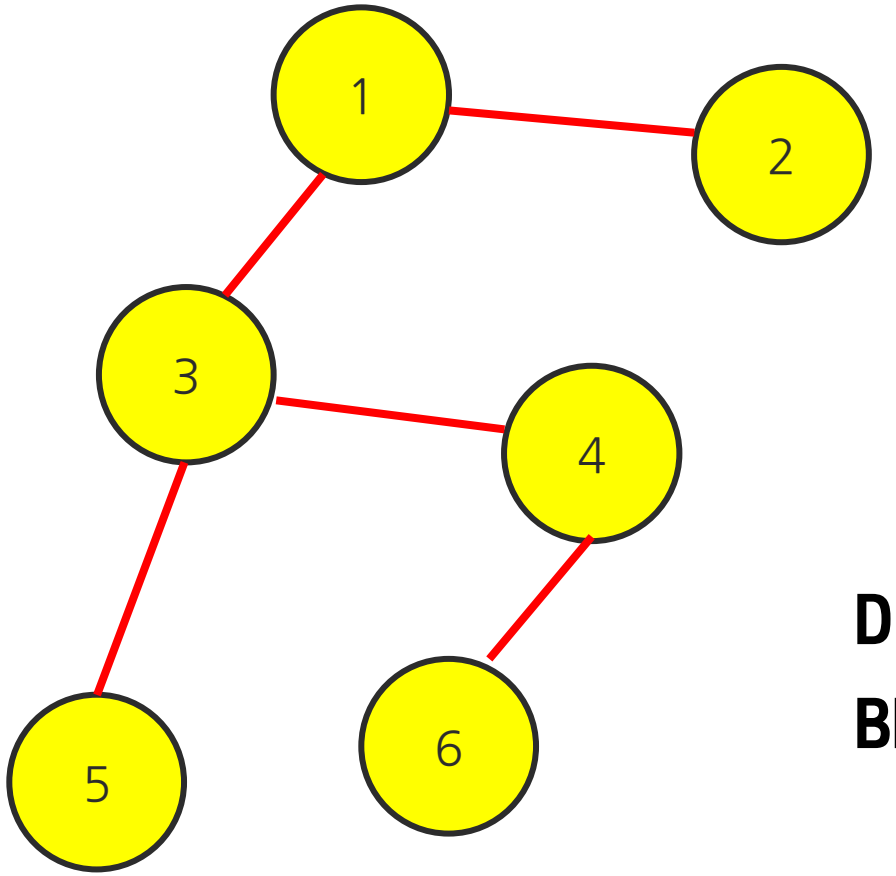
순서: 1 -> 3 -> 2 -> 5 -> 4 ->

BFS



순서: 1 -> 3 -> 2 -> 5 -> 4 -> 6

BFS



DFS순서: 1 -> 3 -> 5 -> 4 -> 6 -> 2

BFS 순서: 1 -> 3 -> 2 -> 5 -> 4 -> 6

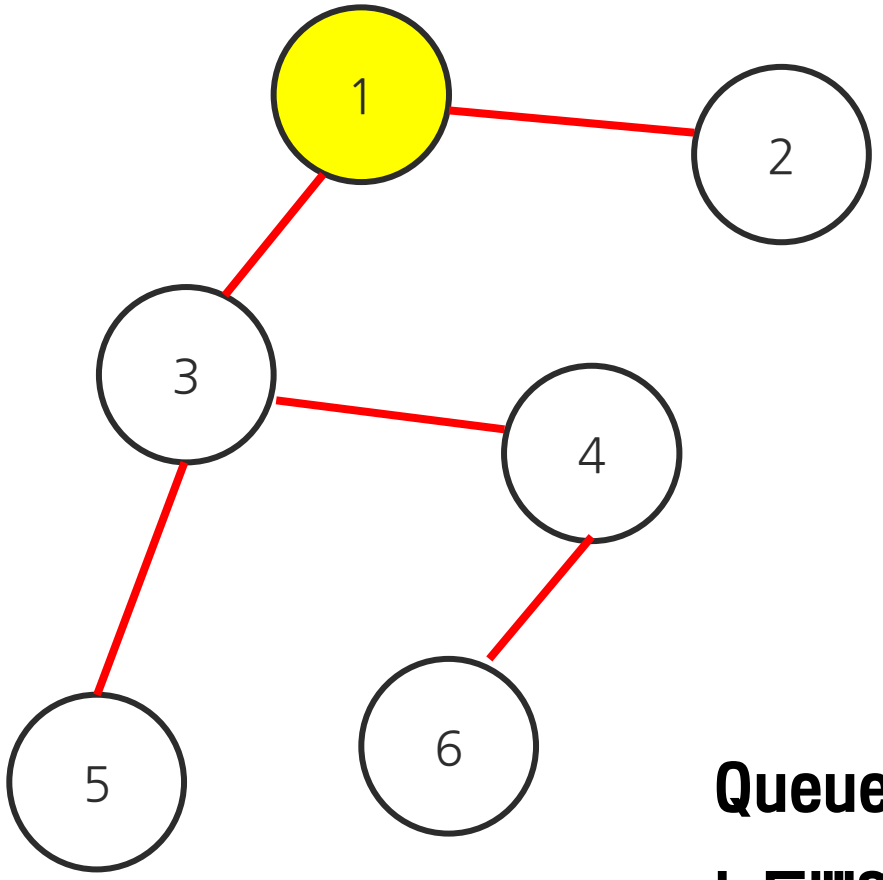
03. BFS

어떻게 구현???

Queue를 활용해서

```
노드배열 = [-1] * 노드갯수
간선배열 # 입력정보
현재위치 = 1
현재비용 = 0

queue.push((현재위치, 현재비용))

while !queue.empty():
    현재위치, 현재비용 = queue.top()
    queue.pop()
    if 노드배열[현재위치] >= 0:
        continue
    노드배열[현재위치] = 현재비용
    for 다음방문노드 in 간선배열[현재위치]:
        queue.push((다음방문노드, 현재비용 + 1))
```

BFS



Queue: [(3, 1) , (2,1)]

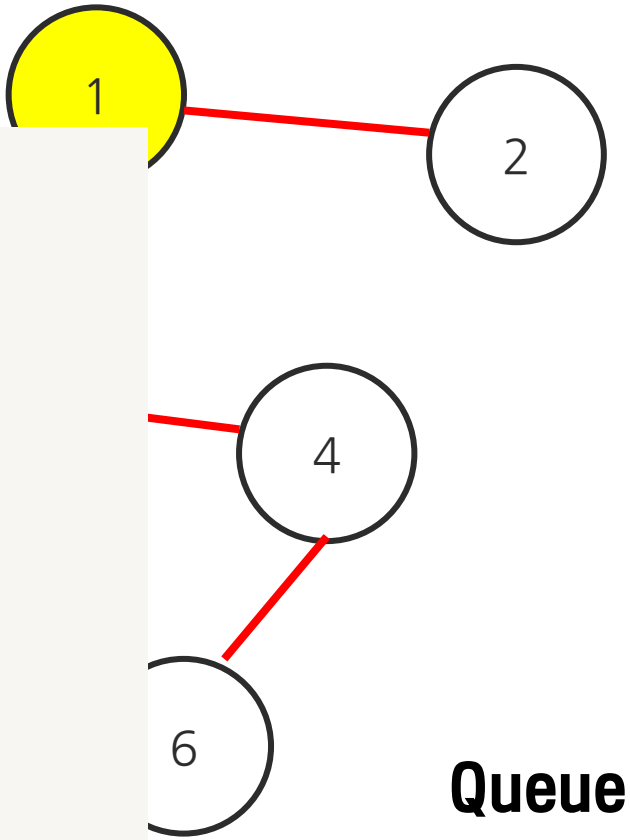
노드배열: [0, -1, -1, -1, -1, -1]

간선배열: [[3,2], [1], [5,4,1], [6,3], [3], [4]]

순서: 1 ->

BFS

```
노드배열 = [-1] * 노드갯수
간선배열 # 입력정보
현재위치 = 1
현재비용 = 0

queue.push((현재위치, 현재비용))

while !queue.empty():
    현재위치, 현재비용 = queue.top()
    queue.pop()
    if 노드배열[현재위치] >= 0:
        continue
    노드배열[현재위치] = 현재비용
    for 다음방문노드 in 간선배열[현재위치]:
        queue.push((다음방문노드, 현재비용 + 1))
```

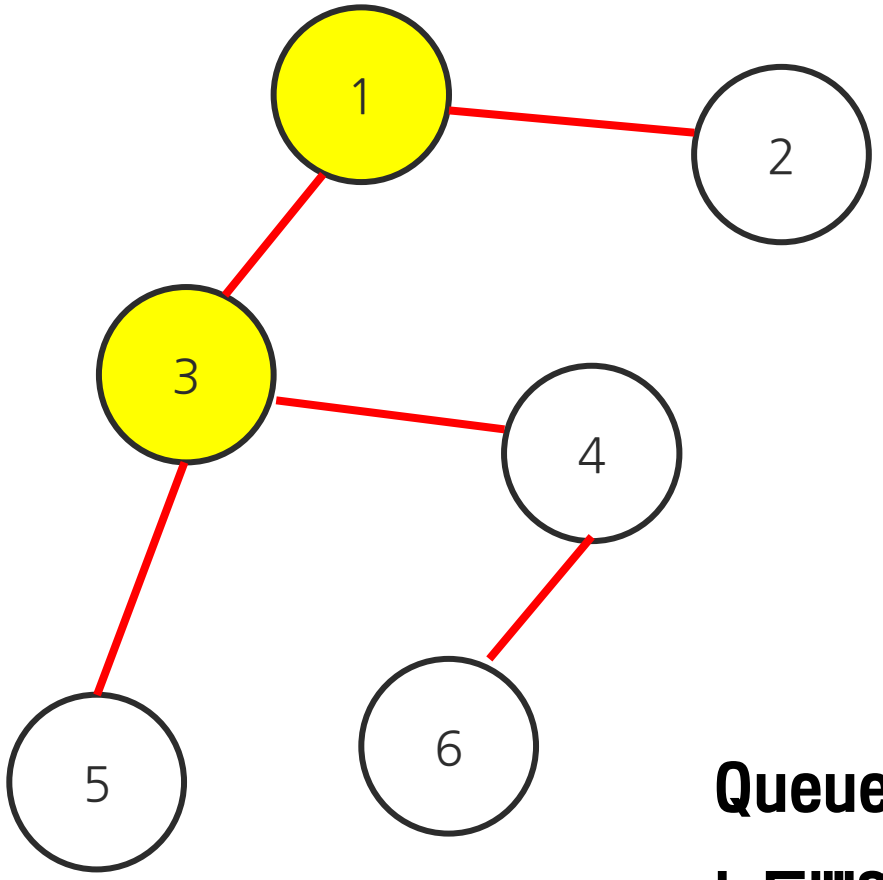


Queue: [(3, 1) , (2,1)]

노드배열: [0, -1, -1, -1, -1, -1]

간선배열: [[3,2], [1], [5,4,1], [6,3], [3], [4]]

BFS



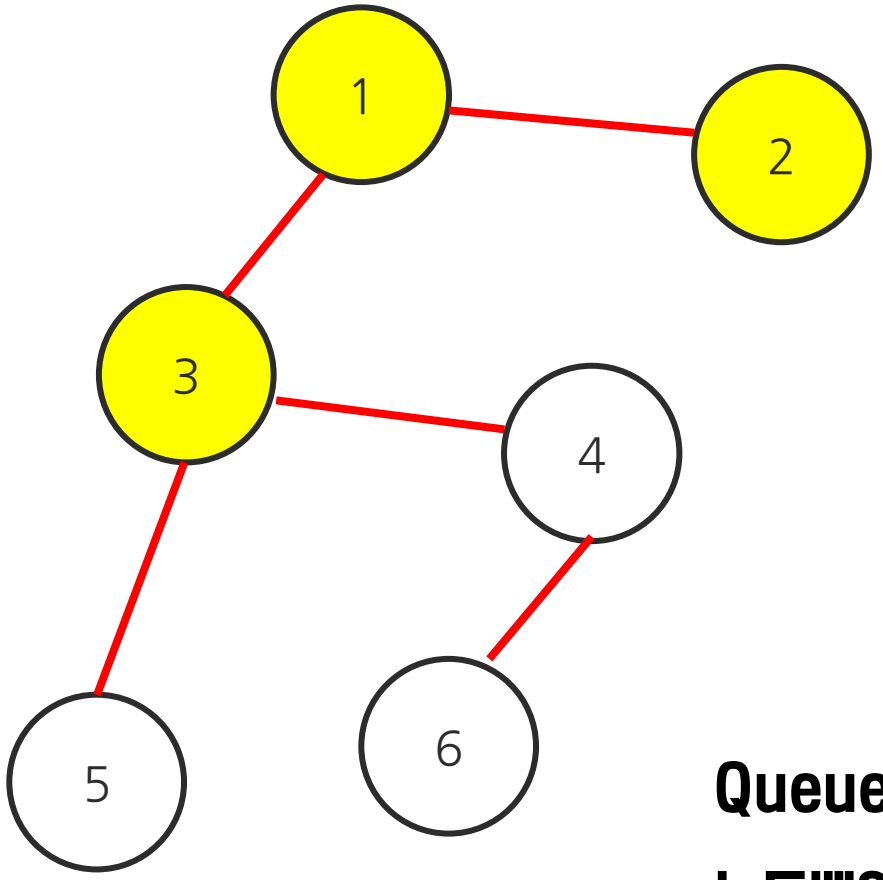
Queue: [(2,1) , (5,2), (4,2)]

노드배열: [0, -1, 1, -1, -1, -1]

간선배열: [[3,2], [1], [5,4,1], [6,3], [3], [4]]

순서: 1 -> 3 ->

BFS



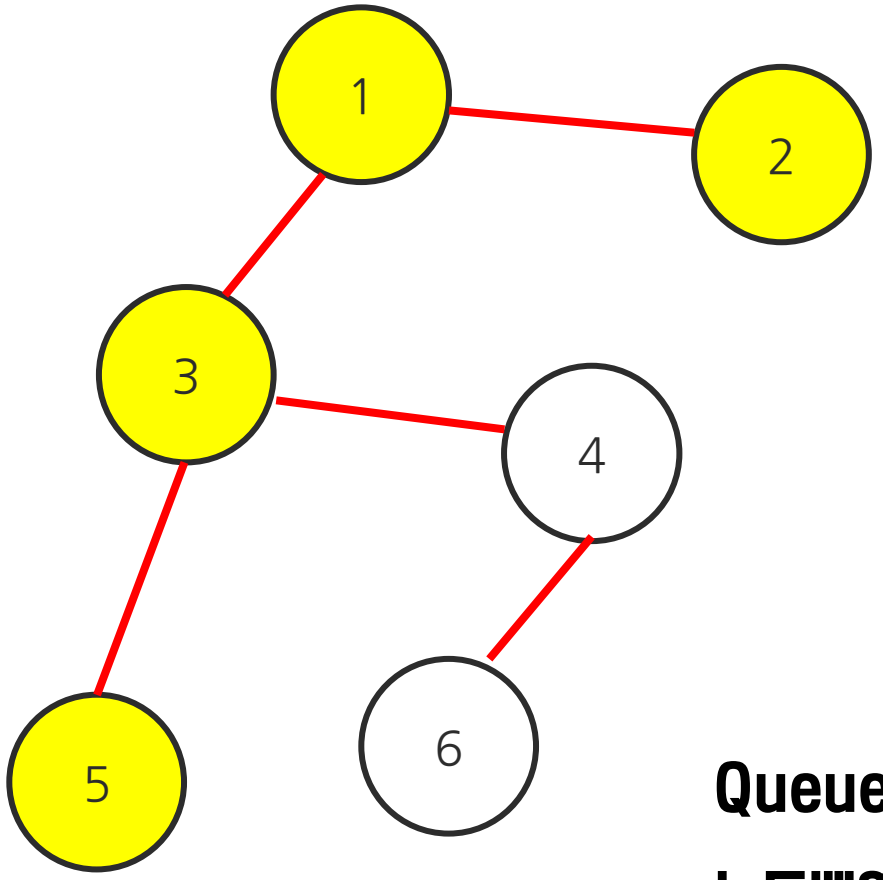
Queue: [(5,2), (4,2)]

노드배열: [0, 1, 1, -1, -1, -1]

간선배열: [[3,2], [1], [5,4,1], [6,3], [3], [4]]

순서: 1 -> 3 -> 2 ->

BFS



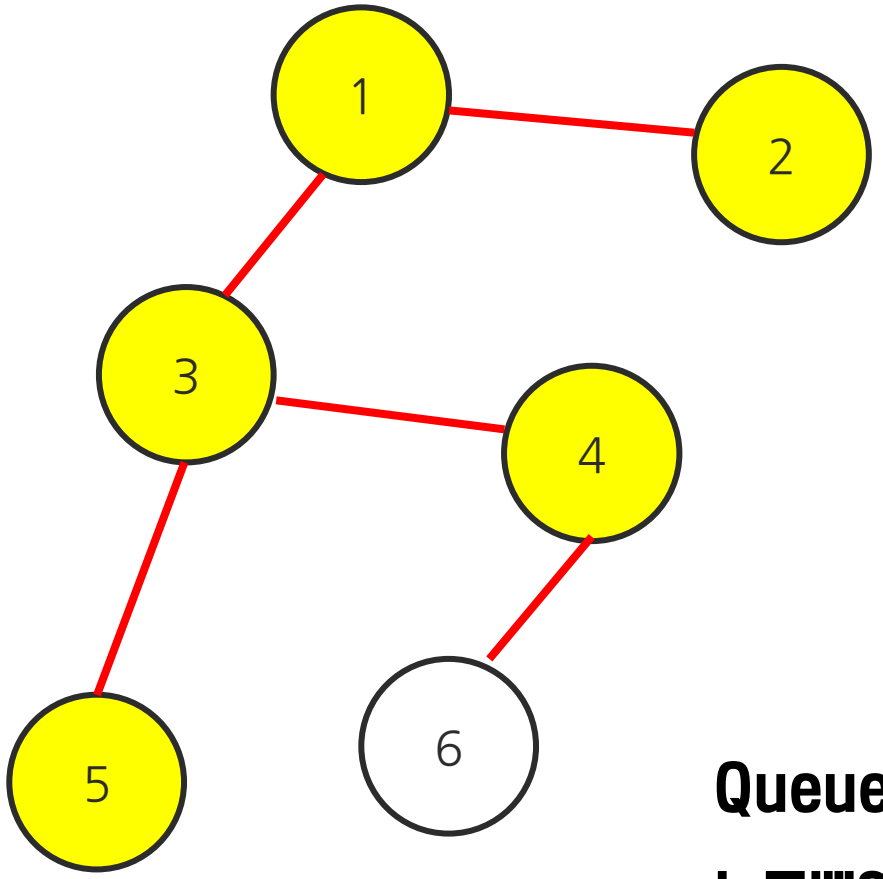
순서: 1 -> 3 -> 2 -> 5 ->

Queue: [(4,2)]

노드배열: [0, 1, 1, -1, 2, -1]

간선배열: [[3,2], [1], [5,4,1], [6,3], [3], [4]]

BFS



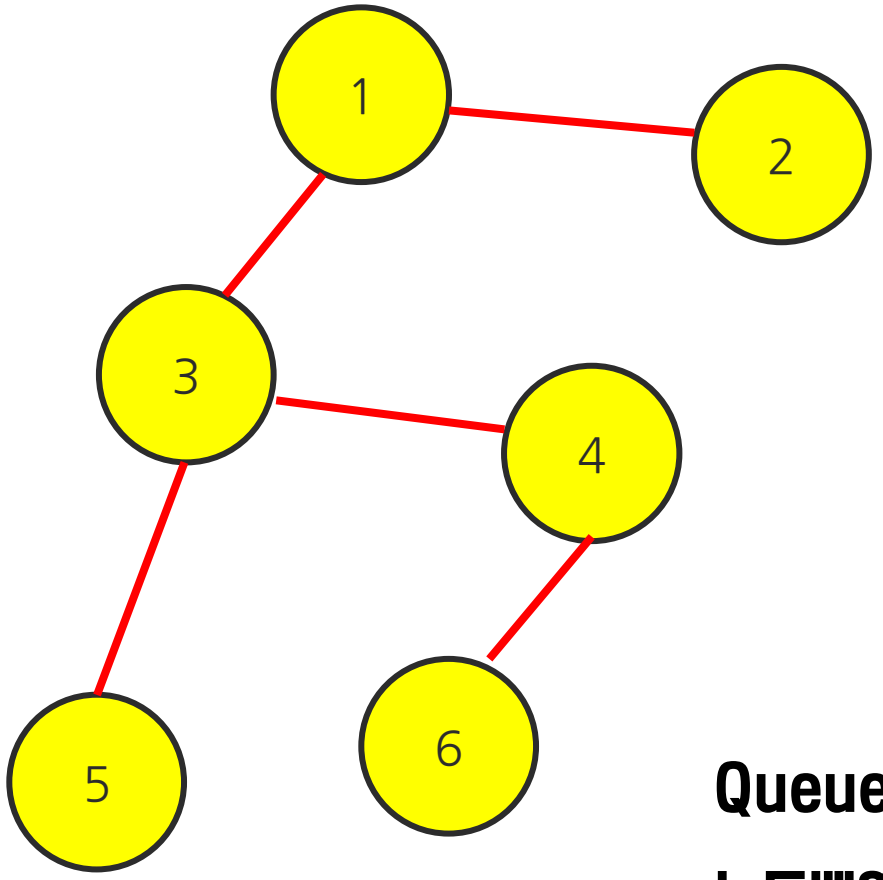
순서: 1 -> 3 -> 2 -> 5 -> 4 ->

Queue: [(6,3)]

노드배열: [0, 1, 1, 2, 2, -1]

간선배열: [[3,2], [1], [5,4,1], [6,3], [3], [4]]

BFS



Queue: [ ]

노드배열: [0, 1, 1, 2, 2, 3]

간선배열: [[3,2], [1], [5,4,1], [6,3], [3], [4]]

순서: 1 -> 3 -> 2 -> 5 -> 4 -> 6

04. 점검

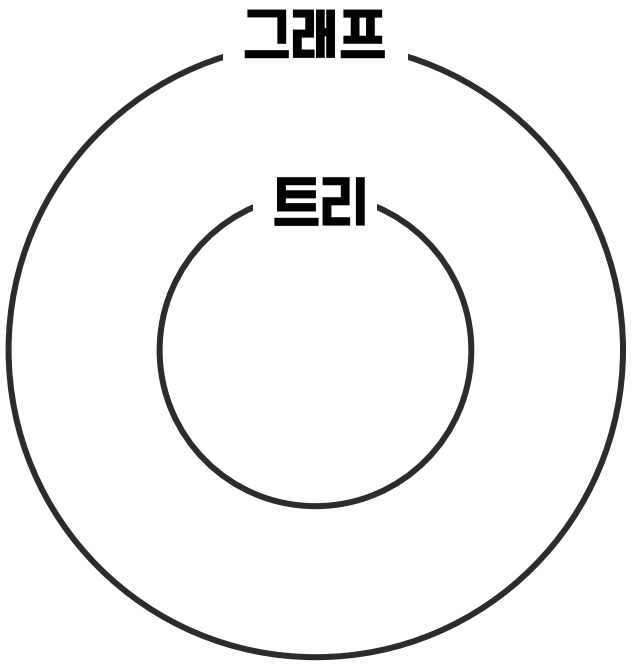
점검

Q1: 그래프와 트리란 무엇인가? 둘은 어떻게 다른가?

Q2: DFS와 BFS는 어떻게 구현할 수 있는가?

점검

Q1: 그래프와 트리란 무엇인가? 둘은 어떻게 다른가?



그래프: 정점(Vertex)과 간선(Edge)으로 이루어진 자료구조

트리: 그래프의 일부, Cycle을 허용하지 않음 (계층구조)

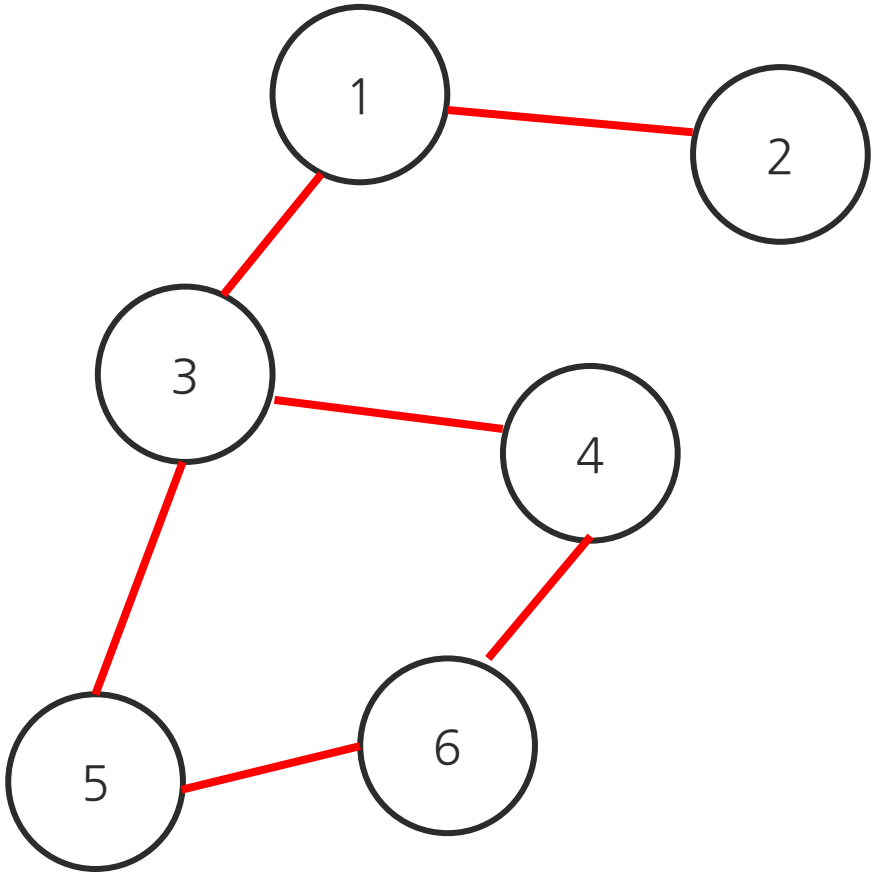
Cycle이 있다? -> 임의의 두 정점에 대해 경로가 2개 이상 존재한다.

점검

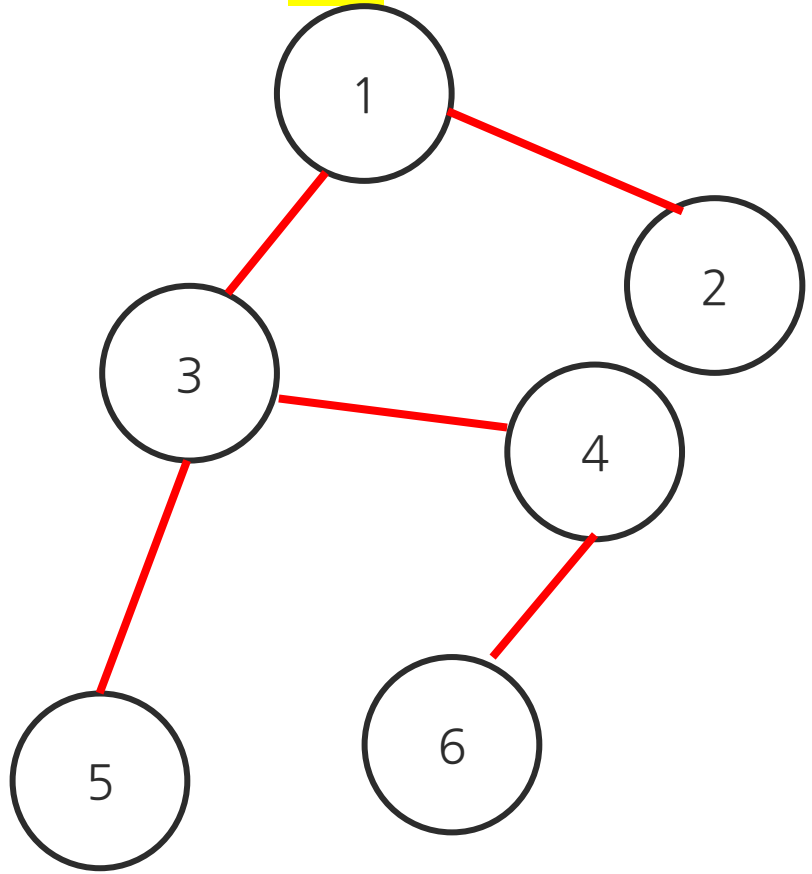
Q1: 그래프와 트리란 무엇인가? 둘은 어떻게 다른가?

Cycle이 있다? -> 임의의 두 정점에 대해 경로가 2개 이상 존재한다.

그래프



트리



점검

Q2: DFS와 BFS는 어떻게 구현할 수 있는가?

점검

Q2: DFS와 BFS는 어떻게 구현할 수 있는가?

DFS => 재귀

```
def Vertex방문(노드배열, 현재위치, 간선배열, 비용):
    if 노드배열[현재위치] >= 0: #현재 위치에 방문 했었나???
        return

    노드배열[현재위치] = 비용
    for 다음방문노드 in 간선배열[현재위치]:
        Vertex방문(노드배열, 다음방문노드, 간선정보배열, 비용 + 1)

#main
노드배열 = [-1] * 노드갯수
현재위치 = 1
간선배열 #입력으로 받음

Vertex방문(노드배열, 현재위치, 간선배열, 0)
```

BFS => 큐

```
노드배열 = [-1] * 노드갯수
간선배열 # 입력정보
현재위치 = 1
현재비용 = 0

queue.push((현재위치, 현재비용))

while !queue.empty():
    현재위치, 현재비용 = queue.top()
    queue.pop()
    if 노드배열[현재위치] >= 0:
        continue
    노드배열[현재위치] = 현재비용
    for 다음방문노드 in 간선배열[현재위치]:
        queue.push((다음방문노드, 현재비용 + 1))
```

추천문제

백준

https://www.acmicpc.net/step/24

프로그래머스

https://programmers.co.kr/learn/courses/30/parts/12421