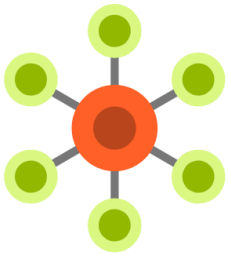
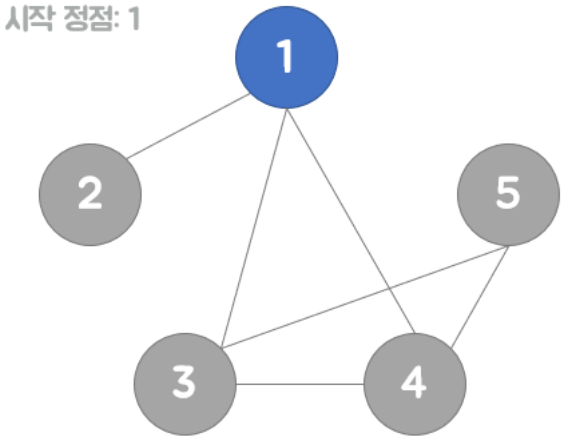
깊이 우선 탐색(DFS)



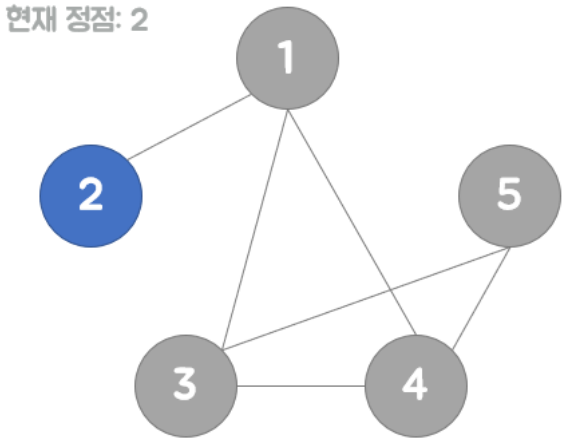
'깊이 우선 탐색(Depth First Search)'은 BFS 와 마찬가지로 **그래프를 방문하거나 탐색하는 방법의 하나**이다. DFS 는 주로 **완전 탐색이나 백트래킹과 같이 탐색의 횟수, 즉 그래프의 최대 경로가 정해져 있거나 예측 가능한 경우에 주로 이용**한다. 왠지 BFS 와 이름이 비슷하지만 다른 느낌이 드실 텐데요, DFS 는 어떤 절차로 진행되는지, BFS 와는 어떤 차이점이 있는지 살펴보자.

1. 선택한 정점에서 해야 할 작업 진행
2. 선택한 정점과 연결된 정점 중 아직 방문하지 않은 정점 방문
3. 만약 더는 방문할 정점이 없다면 이전 정점으로 되돌아감
4. 1~3 과정 반복

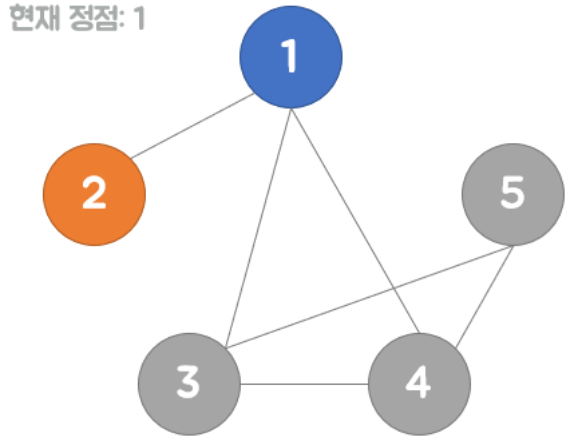
DFS 는 선택한 정점을 저장하기 위한 도구로 큐(Queue) 대신 스택(Stack)을 이용한다는 점에서 BFS 와 차이를 보이다. DFS 의 진행 절차 중 3 단계를 좀 더 쉽게 구현하기 위해서 스택을 사용하는데, 이때 스택을 직접 사용하지 않고 주로 스택의 원리를 이용하는 재귀 함수를 통해 구현하는 편이다. DFS 역시 그림을 통해 살펴보면,.



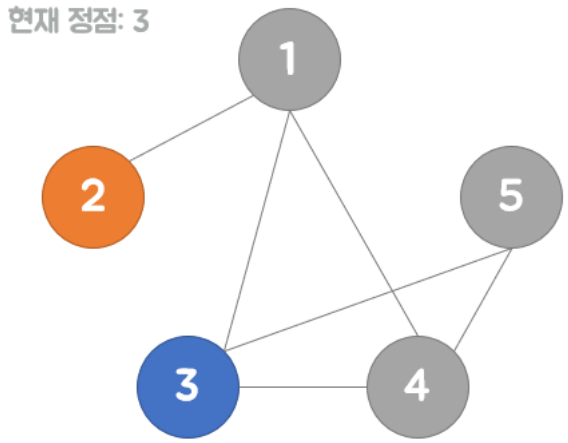
위와 같은 그래프가 있을 때 1 번 정점부터 탐색을 시작하겠습니다. 1 번 정점과 연결된 정점 중, 아직 방문하지 않은 점은 2, 3, 4 번 정점이다. 그중 2 번 정점을 택하여 이동한다.



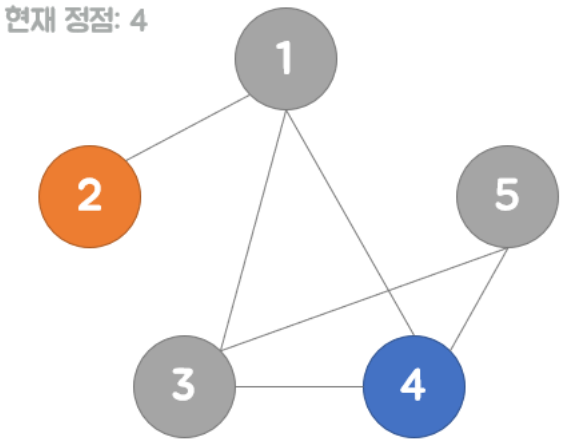
2 번 정점에는 연결된 정점이 없으므로 다시 1 번 정점으로 되돌아간다.



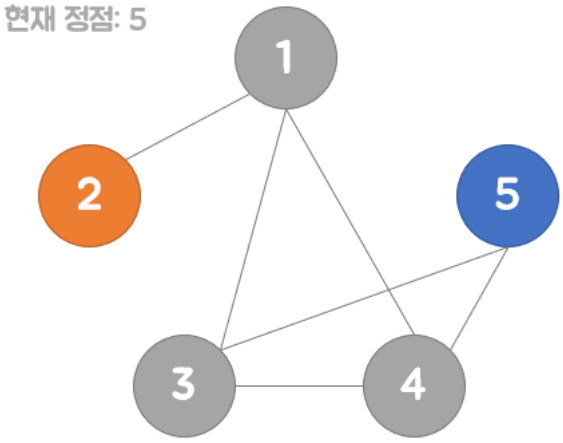
그다음, 1 번 정점과 연결된 정점 중 아직 방문하지 않은 정점인 3 번 정점으로 이동한다.



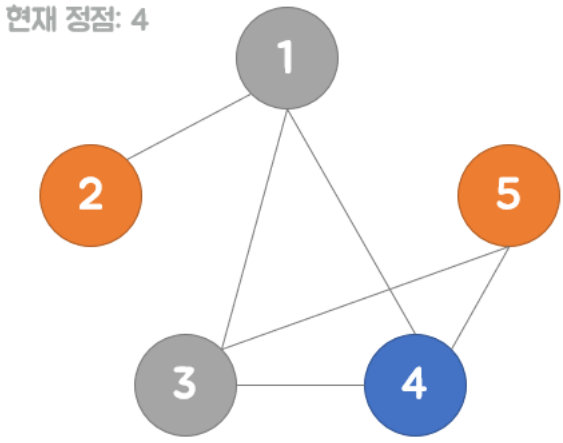
3 번 정점과 연결된 정점 중 아직 방문하지 않은 정점인 4 번 정점으로 이동한다.



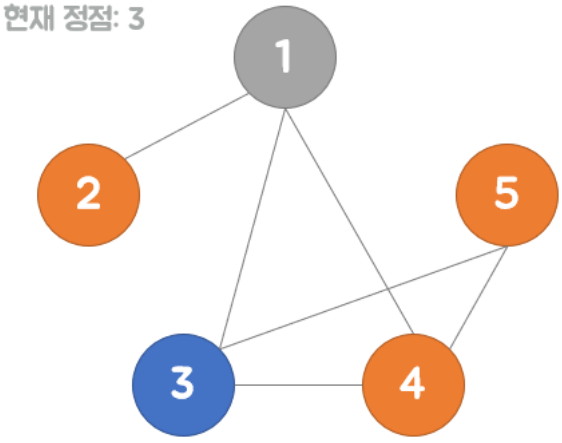
4 번 정점과 연결된 정점 중 아직 방문하지 않은 정점인 5 번 정점으로 이동한다.



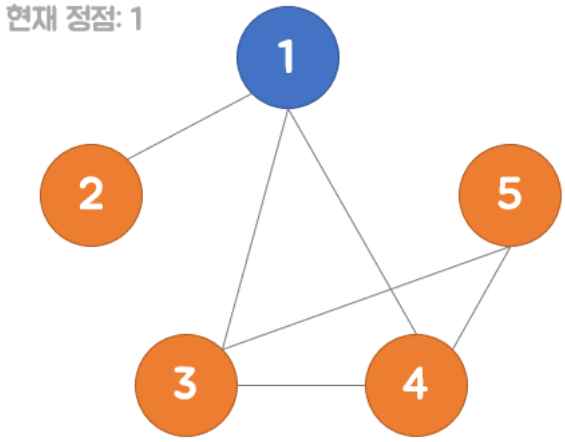
5 번 정점에는 연결된 정점이 없으므로 다시 4 번 정점으로 되돌아간다.



4 번 정점 역시 연결된 정점이 없으므로 다시 3 번 정점으로 되돌아간다.



3 번 정점 역시 연결된 정점이 없으므로 다시 1 번 정점으로 되돌아간다.



1 번 정점 역시 연결된 정점이 없습니다. 이로써 모든 정점을 방문했음을 확인했으니 탐색을 종료한다. 아래의 코드는 C++를 이용하여 구현한 DFS 예제 코드이다.

```swift
var check = [Bool](repeating: false, count: 1010)
var adj = [[Int]]()

func dfs(_ start: Int) {
    if check[start] { // 이미 방문한 정점이라면 함수를 종료
        check[start] = true // 현재 정점을 방문했다고 표시
    }

    // 인접 리시트를 이용한 그래프
    for i in 0..<adj[start].count {
        let next = adj[start][i]
        dfs(next)
    }

    // 인접 행렬을 인용한 그래프
    for i in 0..<v { // 현재 정점과 연결된 모든 정점을 확인
        if adj[start][i] == 1 { // start와 v가 연결된 경우에만 방문
            dfs(v)
        }
    }
}
```