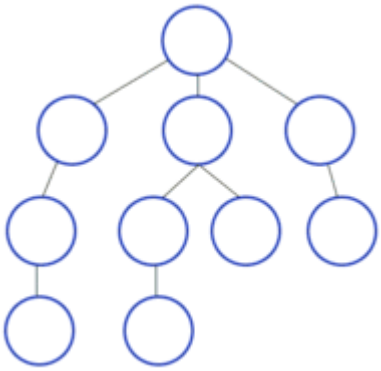
깊이 우선 탐색 (DFS)



💡 **깊이 우선 탐색** : 루트, 시작 노드에서 시작해 가장 멀리 있는 정점을 방문, 더 방문할 정점이 없거나 원하는 노드가 아니라면 다시 돌아가서 탐색

`깊이 우선 탐색`의 경우 한 층을 다 방문한 다음 넘어가는 BFS와는 달리 한번 결정이 된 방향을 계속해서 탐색하는 알고리즘입니다. 위 그림에서 볼 수 있든 탐색하는 모습이 가장 깊은 곳을 찍는게 목표라 '깊이 우선' 이라고 합니다.

BFS와의 차이

구현 방법

가장 큰 차이는 **구현 방법**입니다. BFS에서는 큐를 사용해서 구현을 했다고 하면 DFS는 스택을 사용하는게 맞습니다. 다만 함수 CALL 방식들이 스택 방식으로 되어 있으므로 주로 **재귀를 통해 구현**하게 됩니다.

(이것 덕분에 공간적으로 이득을 볼 수 있습니다)

적합한 곳

BFS는 정답이 루트 근처에 있을 때 유용하게 작동되고 최단 경로임을 알아내기 쉽습니다. (간선 길이가 같은 경우)

대신 DFS는 정답이 리프 노드(트리에서 자식이 없는 노드)일때 사용하기 좋은 알고리즘으로 이 경우 BFS 보다 쉽게 탐색할 수 있습니다.

⇒ 이 이유로 인해서 깊이가 깊어질 우려가 있는 경우 DFS로 하게 되면 상당한 시간이 걸리게 됩니다.

백트래킹 (backtracking)

트리에서 탐색을 하는 과정중 **'이 경로는 무조건 정답이 나올 수 없다'** 하는 경우가 있습니다. 예를 들어 돈은 10000원 있는데 이 경로로가면 10만원을 써야 한다던지 등등... 이럴 경우 더이상 탐색을 할 필요가 없으니 다시 루트 노드로 돌아가 다른 경로를 탐색하면 됩니다.

주로 문제에서 '여행에 쓸 수 있는 돈의 수', '시간' 등으로 제약이 있는 경우 백트래킹을 많이 사용하게 되며 이런 경우 DFS와 함께 사용하는 편입니다. 다만 DFS와 백트래킹은 항상 함께하는게 아니니 문제의 특성을 잘 고려해 **트리의 끝까지 가야 한다면 DFS만을 사용**해야 합니다!

DFS 의사 코드

```
void DFS(int node){
  // 해당 노드에 대한 방문 처리
    visit[node] = true;

  // 해당 노드에서 원하는 일(출력)을 함
    printf("%d ", node);

  // 이곳과 연결된 노드들 중 방문하지 않은 노드 탐색하기
    for(int i = 1; i <= n; i++){
        if(!visit[i] && edge[node][i])
            DFS(i);   // 재귀 호출로 구현
    }
}
```

추천 문제

1260번: DFS와 BFS

그래프를 DFS로 탐색한 결과와 BFS로 탐색한 결과를 출력하는 프로그램을 작성하시오. 단, 방문할 수 있는 정점이 여러 개인 경우에는 정점 번호가 작은 것을 먼저 방문하고, 더 이상 방문할 수 있는 점이 없는

https://www.acmicpc.net/problem/1260

13023번: ABCDE

https://www.acmicpc.net/problem/13023

2668번: 숫자고르기

세로 두 줄, 가로로 N개의 칸으로 이루어진 표가 있다. 첫째 줄의 각 칸에는 정수 1, 2, ..., N이 차례대로 들어 있고 둘째 줄의 각 칸에는 1이상 N이하인 정수가 들어 있다. 첫째 줄에서 숫자를 적절히 뽑으면, 그

https://www.acmicpc.net/problem/2668