알고리즘 스터디 10 - DFS

DFS의 기본 원리는 갈 수 있는 만큼 최대한 깊이가고, 더이상 갈곳이 없으면 이전 정점으로 돌아간다는 것이다. 쉽게 말해서 그냥 세로로 한줄씩 읽어 내려간다고 생각하면 된다. stack과 재귀함수를 사용하여 구현한다.

DFS 구현 방법

- **인접 행렬**

장점 : 구현이 간단함. 직접 접근할 경우(노드가 서로 연결되었는지 확인할경우), O(1)의 시간복잡도를 가진다.

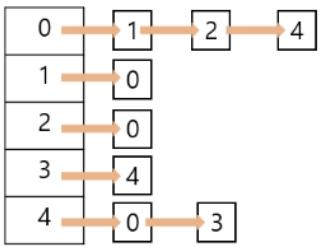
단점

1) 연결된 모든 노드 확인 시간복잡도 O(N)

i 노드에 연결된 모든 노드를 확인하고자할때 array[i][0] ~ array[i][N-1] 모두 확인해야하기 때문에 O(N)의 시간복잡도를 가진다.

2) 공간복잡도 O(N^2)

- **인접 리스트**



장점

1) 연결된 모든 노드 확인 시간복잡도 O(N)

항상 인접행렬보다 빠르다.

2) 공간복잡도 O(N) → N:간선의 개수

실제 연결된 노드만 저장

단점

1) 노드가 서로 연결되었는지 확인하는데 오래 걸린다. array[y][x]가 y를 갖는지 확인해야 한다. **시간복잡도 O(N)**

사용하는 경우

1. 하나하나 모든 경우의 수를 탐색해야 할때 → DFS, BFS 둘다 가능
2. 경로의 특징을 저장하는 문제

   각 정점에 숫자가 있고 a부터 b까지 경로를 구하는데 경로에 같은 숫자가 있으면 안된다는 문제. 각각의 경로마다 그 특징을 저장해두어야하는 문제