**학과: 컴퓨터공학과 학년: 2 학번: 20191657 이름:최세은**

1. **DFS와 BFS의 시간 복잡도를 계산하고 그 과정을 설명한다.**

DFS: 깊이 우선 탐색으로, 스택을 사용하여 한 시작점에서 깊이 방향으로 갈 수 있을 때까지 계속 진행한다. 그러다 끝에 도달해 더 이상 갈 수 없으면 다시 다른 방문 가능한 노드로 돌아가서 깊이 진행하는 것을 재시작한다. 이를 방문 가능한 노드가 없어질 때까지 반복하는데, 전체 노드의 개수를 n이고 노드를 연결한 간선의 개수를 k개라고 하면 DFS의 시간 복잡도는 O(n + k)이다. DFS 함수가 n번 호출될 때, 함수 자체는 존재하는 간선의 개수만큼만 참조되므로 k의 복잡도를 가지기 때문이다.

BFS: 너비 우선 탐색으로, 큐를 사용하여 한 노드에서 수평 방향에 있는 노드들을 모두 접근하며 진행하는 방식이다. 한 노드에서 방문 가능한 노드를 고르고, 해당 노드에서 방문이 가능한 노드를 또 고려하며 진행한다. BFS는 DFS와 탐색 방법은 다르지만 목적이 같으므로 시간 복잡도는 둘이 동일하다. 둘 다 모든 노드를 한번씩 방문하고, 쓰는 자료구조(stack, queue)가 다르지만 노드가 한번씩 push 되었다 pop되는 것은 같기 때문이다.

1. **자신이 구현한 자료구조 상에서 DFS와 BFS 방법으로 실제 경로를 어떻게 찾는지 설명한다. 특히 DFS 알고리즘을 iterative한 방법으로 구현하기 위한 방법을 생각해보고 제시한 다.**

DFS: iterative한 방법으로 구현하기 위해서 stack을 사용하여 구현한다. 시작점을 root node로 하고, DFS 알고리즘을 따라 간선을 따라 노드들을 탐색하며 stack에 node를 push 하다가, 방문한 적이 없는 child node가 더 이상 존재하지 않는 노드에 도착했다면 stack의 노드를 pop하여 이전 node로 이동한다.

BFS: DFS와 비슷하게, queue를 사용해서 구현한다. 시작점을 root node로 하고, BFS 알고리즘을 따라 수평 방향의 node들을 탐색하며 queue에 탐색한 node들을 push 한다. 그러다 더 이상 탐색을 할 수 없게 되면 queue에 있던 node를 pop하여 이전 node로 이동해 다시 탐색을 시작한다.