컴퓨터공학실험1 12주차 예비보고서

전공: 컴퓨터공학과 학년: 2 학번: 20191559 이름: 강상원

1. DFS와 BFS 알고리즘에 대해 조사하고 간략히 요약한다.

   DFS(깊이 우선 탐색)은 그래프를 탐색할 때 현재 노드에서 이동할 수 있는 노드 중 아직 방문하지 않은 노드가 있으면 탐색하고, 없으면 다시 이전 노드로 돌아가 탐색을 계속하는 알고리즘이다. 현재 위치를 스택으로 기억하는 방식을 취한다. 한 방향으로 탐색하다가 진행할 수 없으면 다시 이전 노드로 돌아가 다른 방향으로 탐색을 계속하게 되는 것을 Backtracking이라고도 부른다.

   BFS(너비 우선 탐색)은 현재 노드에서 인접한 노드를 바로 방문하는 방식이다. 인접한 노드를 바로 바로 탐색하기 대문에 큐를 사용한다. 결과적으로 깊이가 얕은 노드부터 탐색이 진행되게 된다.

   DFS와 BFS 모두 시간복잡도는 $G=(V, E)$에서 $O(|V| + |E|)$이다.

2. 미로문제에서 DFS, BFS를 수행하기 위한 자료구조를 설계하고 이의 공간복잡도를 보인다.

   현재 칸에서 움직일 수 있는 방향을 생각한다면 위, 아래, 좌, 우 총 4가지이다. 현재 칸이 point[i][j]라 한다면, point[i-1][j], point[i+1][j], point[i][j-1], point[i][j+1]이다. (주어진 미로에서 대각선으로 움직일 수는 없다.) 따라서 이차원 배열을 만들어서, 이미 방문했는지 여부를 저장하면 된다. 이때, 단순 이차원 배열로 저장하면 상, 하, 좌, 우로의 이동 가능성을 저장하는데 문제가 생긴다. 따라서

   ```
   struct block {
       int up;
       int down;
       int left;
       int right;
   };
   ```

   와 같이 struct를 만들고, 이 struct를 담는 2차원 배열을 선언하면 된다.

   (ex: 이동 가능하면 1, 이동 불가 0 저장)

   따라서 이 자료구조의 공간복잡도는 $O(WIDTH*HEIGHT)$라 할 수 있다.

   앞서 언급했듯이 DFS를 이용한다면 스택이 필요하고, BFS를 이용한다면 큐가 필요할 수 있다. 하지만 스택이나 큐의 최대 크기는 노드의 개수이므로 위의 공간복잡도에는 영향을 미치지 않는다.

3. 설계한 자료구조에서 DFS, BFS를 어떻게 수행할지 간략히 보인다.

   - DFS

     start: s, end: e

     make blank stack

     stack.push(s)

     s: has visited

```
while( !stack.empty() )
    idx = stack.top()
    if idx == e
        break;
    stack.pop()
    if an adjacent path is not yet visited && move is possible
                stack.push(path)
                path: visited
    pop until stack is empty
```

- BFS

```
start: s, end: e
make blank queue
queue.enqueue(s)
while( !queue.empty() )
    idx = queue.front()
    if idx == e
        break
    for i = 0, 1, 2, 3
        iterate to find four paths
        if a path is not yet visited && move is possible
            queue.enqueue(path)
            path: visited
            idx = path's parent
```