13주차 예비보고서

전공: 컴퓨터공학과 학년: 2학년 학번: 20231515 이름: 김다은

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

미로의 가로 길이를 COL, 세로 길이를 ROW라고 표현하겠다. DFS, BFS 모두 미로의 모든 칸을 탐색하게 되는 경우가 최악의 경우이다. 따라서 DFS와 BFS 모두 시간복잡도로 O(ROW\*COL)을 가진다.

**2. 자신이 구현한 자료구조 상에서 DFS와 BFS 방법으로 실제 경로를 어떻게 찾는지 설명한 다. 특히 DFS 알고리즘을 iterative한 방법으로 구현하기 위한 방법을 생각해보고 제시한다.**

12주차에서 아래와 같은 inputmaze vector<string> 변수에 미로를 저장하였다.

vector<string> inputmaze;

**2-1. DFS 방법으로 실제 경로를 찾는 방법**

필요한 변수는 다음과 같다.

- 미로의 각 칸에 방문했는지 여부를 체크할 bool \*\*visited

- 또한 현재 칸의 좌표를 나타낼 int x, y

- 이동할 목표 칸의 좌표를 나타낼 int nx, ny

- 미로의 출구를 나타낼 int destx, desty

- 경로를 담아 둘 stack<tuple<int, int, short>>. 이때 stack에 저장되는 원소는 {x좌표, y좌표, direction} 꼴이다.

- 탐색한 모든 경로를 저장할 vector<pair<int,int>> path

- 정답 경로를 저장할 vector<pair<int,int>> anspath

또한 DFS 알고리즘을 iterative한 방법으로 구현하기 위해서는 재귀함수 대신 stack과 반복문을 사용할 수 있다. iterative한 DFS 알고리즘의 작동 흐름은 다음과 같다.

1. 출발지점 (1,1)을 stack에 push하고 visited 변수에 방문처리 한다.
2. stack.top()의 지점에서 상하좌우로 이동가능한 경우를 찾는다. (벽에 막히지 않았고, 미로 밖으로 나가지 않고, 방문하지 않은 칸인 경우)
3. 이동이 가능한 좌표를 stack에 push하고 visited 변수에 방문처리 한다. 이후 2번으로 되돌아간다. 만약 이동가능한 좌표가 목적지의 좌표와 같다면 반복을 종료한다.
4. 만약 상하좌우로 이동가능한 경우가 없다면 stack.pop() 이후 2번으로 되돌아간다.

**2-2. BFS 방법으로 실제 경로를 찾는 방법**

queue를 이용해 BFS로 미로의 탈출 경로를 찾아낼 수 있다.

BFS 알고리즘의 작동 흐름은 다음과 같다.

1. 출발지점 (1,1)을 queue에 push하고 visited 변수에 방문처리 한다.
2. queue.front()의 좌표를 저장한 뒤 queue.pop()을 해준다.
3. 저장해 놓은 좌표에서 상하좌우로 이동가능한 좌표를 모두 queue에 push한다. (벽에 막히지 않았고, 미로 밖으로 나가지 않고, 방문하지 않은 칸인 경우) 만약 이동가능한 좌표가 목적지의 좌표와 같다면 반복을 종료한다. 아직 목적지에 도달하지 않은 경우 2번 과정으로 되돌아간다.