13주차 예비보고서

전공: 컴퓨터공학과 학년: 4학년 학번: 20192135 이름: 윤영인

**1.**

DFS (Depth First Search)는 깊이 우선 탐색으로 한 정점에서 시작해서 해당 정점과 연결된 모든 정점들을 간선을 따라가면서 재귀적으로 탐색한다. 탐색하면서 더 이상 탐색할 정점이 없으면 이전 정점으로 돌아가서 이 과정을 반복한다. 따라서 DFS의 시간 복잡도는 모든 정점과 간선을 확인하므로 O(V+E) 로 V는 정점, E는 간선의 수이다.

BFS(Breadth-First Search)는 너비 우선 탐색으로 한 정점과 연결된 인접한 정점부터 탐색하는 알고리즘이다. 인접한 정점을 모두 탐색한 후 다음 정점의 인접한 정점을 탐색한다. 따라서 BFS의 시간 복잡도 역시, 모든 정점과 간선을 확인하므로 O(V+E) 로 V는 정점, E는 간선의 수이다.

**2.**

12주차에서 BFS와 DFS를 구현하기 위해 미리 정점을 만들어 인접리스트를 구현하였다. 하지만 이번 주차에선 인접리스트 대신 방문 체크 배열과 백트래킹을 활용하여 재귀적으로 DFS를 구현하고 큐를 사용하여 BFS를 구현하고자 한다. 우선 가장 빠른 경로와 경로를 체크할 벡터 구조체인 shortestPath, currentPath와 탐색한 모든 경로를 담을 allPath 벡터 구조체를 생성한다. 벽은 0, 방은 1로 변경하여 미로의 정보를 저장한 map 배열을 탐색하면서 방이고 아직 방문하지 않았을 경우 방문 체크를 한다. 이때 currentPath와 allPath에 현재 위치를 넣어주고 주변 방을 탐색하면서 해당 위치로 재귀를 시작한다. 재귀를 하면서 짧은 거리를 측정하기 위해 인자로 경로의 수를 하나씩 증가하면서 측정하도록 한다. 만약 재귀가 끝났다면 currentPath에서 현재 위치를 pop하여 이전 위치로 돌아가 다시 탐색하도록 한다. 또한 재방문하여 탐색이 가능하도록 하기 위해 방문 배열을 초기화한다. 만약 위치가 탈출 위치여서 재귀를 종료하게 될 때, 짧은 경로라면 shortestPath에 currentPath에 담긴 위치들을 담도록 한다.