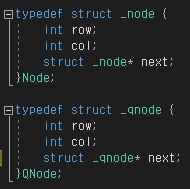
13주차 예비보고서

전공: 기계공학과 학년: 3학년 학번: 20191820 이름: 김형준

**1.**

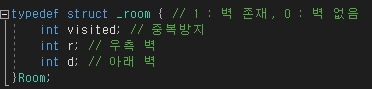
DFS와 BFS는 그래프 탐색의 한 방법으로써, 그래프의 모든 노드를 한 번씩만 방문하는 탐색 알고리즘이다. 그래프 G = (V,E)에서 (V : 노드, E : 간선) 모든 노드를 방문하는데 걸리는 시간복잡도는 O(||V||)이다. 또한, 모든 노드를 방문하기 위해, 한 노드에서 다른 노드로 건너가야할 때 인접한 노드들을 탐색하는데, 이 경우 구현할 때 사용하는 자료구조에 따라 시간복잡도가 달라진다. 인접 행렬을 사용하는 경우, 한 노드에서 그래프상의 모든 다른 노드들을 순회하면서, 해당 노드와 연결되어있는지를 확인해야 하므로 O(||V||^2)의 시간복잡도를 가지고, 인접 리스트를 사용하는 경우, 한 노드에서 다른 노드를 순회할 때 해당 노드에 직접 연결되어있는 모든 간선을 체크하므로 O(||E||)의 시간복잡도를 추가로 소요하게 된다. 따라서 DFS와 BFS의 전체 시간 복잡도는 인접 행렬로 구현할 경우 O(||V||^2)이고, 인접 리스트로 구현할 경우 O(||V||+||E||)이다.

**2.**





위와 같은 구조체와 전역변수들을 사용해 DFS와 BFS를 구현한다. (아래는 2주차에 만든 자료구조)





DFS :

iterative한 방법으로 구현하기 위해 연결리스트를 사용한 스택 자료구조를 직접 만들어 DFS를 구현한다.

DFS 탐색 실행시 Maze에 저장된 미로 데이터의 모든 원소의 visited를 0으로 초기화 한다. (0 : 방문안함, 1 : 방문함 이므로 탐색 시작시 모든 방을 방문 안한 상태로 초기화 함)

미로의 시작 위치가 맨 왼쪽 위이므로 스택에 해당 좌표를 넣고 0,0 지점의 visited를 1로 설정한다. 그 다음 스택의 맨 위쪽 원소의 값으로 i, j의 값을 갱신하고, 상( (i-1,j)의 아래 벽이 없는 경우), 하( (i,j)의 아래 벽이 없는 경우), 좌( (i,j-1)의 우측 벽이 없는 경우), 우( (i,j)의 우측 벽이 없는 경우) 방향을 각각 탐색하면서 탐색하는 방향의 방이 아직 방문하지 않은 방인 경우 (visited가 0인 경우) 해당 방의 위치를 스택에 Push하고, visited의 값을 1로 설정한다. 모든 탐색 방향에서 더 이상 진행 할 수 없는 경우 (현재 방에서 상하좌우 방향으로 ‘벽으로 막혀있지 않고, 이미 방문하지 않은 방’이 하나도 없는 경우) Pop하여 스택에서 원소를 하나 제거한다.

이를 스택에 남은 노드가 없거나 현재 위치가 도착점 (Row-1,Col-1)에 도달할 때까지 반복한다.

BFS :

BFS 탐색방법을 구현하기 위해 연결리스트를 사용한 큐 자료구조를 직접 만들어 BFS를 구현한다.

BFS 탐색 실행시 Maze에 저장된 미로 데이터의 모든 원소의 visited를 0으로 초기화 한다.

(0 : 방문안함, 1 : 방문함 이므로 탐색 시작시 모든 방을 방문 안한 상태로 초기화 함)

미로의 시작위치가 맨 왼쪽 위이므로 큐에 해당 좌표를 넣고 0,0 지점의 visited를 1로 설정한다.

그 다음 Dequeue로 큐의 원소를 하나 제거하고, 제거된 원소의 값으로 i, j의 값을 갱신한 다음 상( (i-1,j)의 아래 벽이 없는 경우), 하( (i,j)의 아래 벽이 없는 경우), 좌( (i,j-1)의 우측 벽이 없는 경우), 우( (i,j)의 우측 벽이 없는 경우) 방향을 각각 탐색하면서 탐색하는 방향의 방이 아직 방문하지 않은 방인 경우 (visited가 0인 경우) 해당 방의 위치를 큐에 추가하고, visited의 값을 1로 설정한다. (DFS와 다르게 한 탐색방향에서 조건을 만족해 큐에 원소를 추가하더라도 나머지 탐색방향으로 계속 탐색을 진행함)

이를 큐에 남은 노드가 없거나 현재 위치가 도착점 (Row-1,Col-1)에 도달할 때까지 반복한다.