13주차 결과 보고서

전공: 컴퓨터공학 학년: 2학년 학번: 20221559 이름: 박준우

1. 실습, 과제로 작성한 프로그램의 알고리즘

격자 그래프 형식으로 상하좌우 이동 가능한 지에 대한 여부를 bitMask로 가지고 있는 자료구조를 이용해 미로를 표현했다.

1-1. Iterative dfs using stack – 실습 시간에 작성한 프로그램의 알고리즘

우선 현재 위치를 담을 수 있는 스택을 선언한다.

그리고 각 위치에 대해 visited 여부를 체크할 수 있는 배열을 선언한다.

그 이후 시작 지점의 좌표를 스택에 넣고, 시작 지점을 방문하였다고 체크한다.

스택의 최상단에 있는 좌표가 현재 방문하고 있는 좌표이고, 이 좌표가 목표 지점이면 스택에 쌓여 있는 좌표들이 경로에 속해있는 좌표이므로, 이것을 별도의 컨테이너로 옮겨준다. 만약 목표 지점이 아닌 경우는 상하좌우로 진행할 수 있는지의 여부를 확인한다. 진행할 수 있는 방향이 있으면, 해당 좌표를 스택에 더해 주고, 다음 루프로 진행한다. 만약 더이상 진행할 수 없으면, 이 좌표는 경로 내의 좌표가 아니기 때문에 스택에서 빼 주고, 이전 좌표에서 다시 탐색할 수 있도록 한다.

스택에 최대로 쌓일 수 있는 원소의 개수는 미로의 칸 개수이고, 방문 여부 확인용 배열 또한 미로의 칸 개수만큼 필요하므로 공간 복잡도는 미로가 N행 M열 이라고 하면 O(NM)이 된다.

또, 방문 여부 확인을 하여서, 한 번 방문한 곳은 다시 방문하지 않기 때문에, 시간 복잡도가 loop 반복 횟수에 따르게 될 텐데, 이것 또한 최대 미로의 칸 개수가 된다. O(NM)

1-2. bfs using queue

Bfs는 시작 좌표를 루트 노드로 하는 spanning tree를 만드는 것과 비슷한 과정을 진행한다. Queue에 시작 좌표를 넣고 여기서부터 bfs를 진행한다. 또, 트리에서의 높이를 저장해 줄 배열을 만들어서, 이를 -1로 초기화 하면 -1인 곳은 방문하지 않은 곳, 방문했으면 높이를 양수로 저장해 주면 된다.

Queue의 맨 앞에 있는 곳에서, 주변으로 진행하지 않은 곳이 있다면 queue에 넣어 주고, 그 곳의 높이를 현재 위치의 높이 + 1로 넣어준 후, 현재 좌표(맨 앞에 있던 좌표)를 지워준다. 이 과정을 반복하다가 현재 좌표가 목표점이거나, queue가 비어있다면 종료하면 된다.

Dfs와 시간, 공간 복잡도가 같다.

2. 우선 내가 이용한 자료구조는 상하좌우의 벽이 있는지 저장하고 있는 2차원 배열로, 그래프 표현 등을 이용하지 않았기 때문에, 어떤 자료구조가 더 유리하고 불리함은 존재하지 않았다.

DFS, BFS 알고리즘 중 DFS가 조금 더 직관적이었던 것 같다. Iteration 방식을 사용하지 않고 재귀함수를 이용한 방식을 쓰면 경로를 취합하는 방식이 조금 더 복잡하고 이해하기 어려울 수 있지만, stack을 이용한 iteration 방식을 사용해서 구현하기 쉬웠던 것 같다.

내가 구현한 방법대로라면 시간, 공간 복잡도 면에서 별 차이가 없어서 크게 장단점도 존재하진 않았던 것 같다.