11주차 예비보고서

전공: 수학/컴퓨터공학 학년: 3학년 학번: 20181294 이름: 임승섭

**1.**

Eller’s algorithm을 제외한 미로 생성 알고리즘으로는 Recursive Backtracking algorithm이 있다.

Backtracking 이란 조건을 만족하는 모든 조합을 탐색하여 해를 찾아내는 방법을 의미한다. 이 과정에서 조합을 탐색하는 방법으로 DFS (깊이 우선 탐색)을 이용한다.

편의를 위해 미로의 방 한 칸을 하나의 노드라고 하면, 우선 미로를 시작할 시작점 노드를 정한다. 그 노드를 기준으로 4가지 방향(동서남북) 중 하나를 선택해 그 방향의 노드를 아직 방문한 적이 없다면 그 노드로 이동한다. 이 과정에서 두 방 사이에 있는 벽을 없앤다. 새로운 노드에 대해 이 과정을 계속 반복한다. 만약 4 방향의 노드를 모두 방문했다면, 방문하지 않은 방이 있는 이전 노드로 돌아가서 다른 방향으로 이 과정을 반복한다. 여기서 recursive의 개념이 들어가게 된다. 여러 번의 recursive 실행을 통해 결국 시작점까지 다시 돌아오게 되면, 더 이상 제거할 벽이 없다는 의미이므로 미로가 완성되고 프로그램은 종료된다.

Recursive Backtracking algorithm의 장점은 알고리즘 자체가 직관적이기 때문에 이해하기에 큰 어려움이 없다는 점이다. 개인적으로 이번 프로젝트에서 사용할 Eller’s algorithm보다 금방 이해할 수 있었다. 또한 아직 실제로 구현해보진 않았지만, 자료구조만 적절히 사용한다면 코드로 구현하는 데에도 큰 어려움이 없을 것으로 보인다. 하지만 단점으로는 메모리 공간을 많이 사용한다는 점이 있다. 아무래도 미로의 size가 커질수록 스택에 넣어야 하는 양도 많아지기 때문에 시간적으로도 오래 걸릴 수가 있다. 이런 경우는 시작점을 여러 개로 설정해서 두 경로가 중간에 만나도록 한다면 시간은 줄일 수 있을 것이다.

**2.**

Eller’s algorithm을 사용한다. 위의 Recursive Backtracking algorithm과 다르게 여기서는 모든 방의 정보를 저장할 필요가 없이 한 줄씩 아래로 내려가면서 정보를 업데이트해주면 된다. 그래서 우선 필요한 정보들을 저장하기 위한 자료구조로는 가장 단순한 배열을 사용한다. 필요한 상수는 미로의 가로 WIDTH와 세로 HEIGHT 이고, 이를 scanf 함수로 받아온다.

우선 현재 수정하고 있는 행의 정보(집합 이름)을 담고 있는 배열 set[WIDTH]를 만든다. 각 인덱스에 해당하는 원소로는 방의 집합 이름이 들어간다. ppt 상에서는 1, 2, 3 등 자연수를 집합 이름으로 사용하였다. 또 현재 수정하고 있는 행의 정보(인접한 방 사이의 벽 유무)를 담고 있는 배열 row\_wall[WIDTH - 1]을 만든다. 방이 총 WIDTH개이므로 벽은 WIDTH - 1개가 된다. 인접한 방 사이에 벽이 있으면 값을 1, 없으면 0으로 설정한다. 마지막으로 현재 행과 다음(아래) 행 사이의 벽 유무를 알려주는 배열 column\_wall[WIDTH]을 만든다. 위와 마찬가지로 벽이 있으면 1, 없으면 0으로 값을 넣어준다. 이렇게 3개의 배열을 사용하면 Eller’s algorithm을 구현할 수 있을 것으로 보인다.

실제 실행을 하면 행의 모든 방을 한번씩 탐색하고 이를 모든 열에 대해 반복한다. 따라서 시간복잡도는 O(WIDTH \* HEIGHT)라고 할 수 있다. 또한 가장 큰 배열의 size가 WIDTH이기 때문에 공간복잡도는 O(WIDTH)라고 할 수 있다.