11주차 예비보고서

전공: 수학/컴퓨터공학 학년: 3학년 학번: 20181288 이름: 윤성호

1. 세상에는 정말 많은 완전 미로 생성 알고리즘이 존재한다. 그 중 크루스칼 알고리즘에 흥미가 생겨 이를 이해해보았다. 먼저, 크루스칼 알고리즘은 총 3개의 과정을 반복하면서 미로를 생성해 나간다. N\*M 사이즈의 직사각형 안에 미로를 생성하는 과정은 아래와 같다.

1st step : 테두리가 아닌 벽을 무작위로 선택한다.

2nd step : 벽을 기준으로 있는 두 방이 이미 연결되어 있다면 다시 1st step으로, 연결되어 있지 않다면 둘을 연결시키고 1st step으로 간다.

3rd step ; 모든 방이 연결되었다면 종료한다.

먼저, 모든 방에 서로 다른 번호를 부여해준다. N\*M개의 방이 존재할 것이므로 시작할 때는 1번방, 2번방, … N\*M번방까지 있을 것이다. 그리고, 선택한 벽의 위치를 저장하는 배열을 하나 생성하여, 이미 선택한 벽은 다시 선택하지 않도록 한다. 방문처리를 해주지 않으면 시간복잡도가 기하급수적으로 커지기 때문에 이를 방지하기 위해 생성하였다. 준비과정이 끝난 이후에는 벽을 무작위로 선택하기 시작한다. 이후 벽을 기준으로 있는 두 방의 연결 상태를 확인해야 하는데, 이는 두 방에 배정되어있는 번호가 같은 지 다른 지를 확인하여 판별한다. 두 방의 번호가 같다면 연결되어 있는 것이고, 두 방의 번호가 다르다면 연결되어 있지 않다는 의미이다. 만약, 두 방이 연결되어 있다면(두 방의 번호가 같다면), 완전 미로의 특성 상 두 방을 연결하는 경로는 유일해야 하기 때문에 벽의 방문처리를 해주고, 다음 벽을 선택한다. 만약, 두 방이 연결되어 있지 않다면(두 방의 번호가 다르다면), 둘을 연결해주어야 한다. 두 방을 연결해주는 것은 각 방과 연결되어 있는 모든 방의 번호를 한가지로 만들어 주는 것을 의미한다. 예를 들어, 벽을 사이로 1번방과 2번방이 연결되어 있다면, 2번방의 번호를 1로 바꿔주는 것은 물론, 2번방과 연결되어있는 모든 방들 또한 1로 바꿔주어야 한다는 것이다. 이를 위해서는 DFS나 BFS 알고리즘이나 equivalent class를 이용한다. 먼저, DFS나 BFS 알고리즘을 이용하는 방법을 보자. 두 방의 번호 중 더 작은 번호로 통일시키는 것으로 생각한 다음, 큰 방부터 DFS나 BFS 알고리즘을 이용하여 그 방과 연결되어 있는 모든 방에 접근하여 번호를 바꿔준다. 그리고 벽이 선택되었으니 방문처리를 해준다면 연결되어있지 않은 방을 연결해주는 과정이 종료된다. DFS 알고리즘을 이용한다면 추가적인 메모리는 필요하지 않지만, 하나의 방의 번호를 여러번 업데이트 해주어야 하는 상황이 많이 발생할 수 있다. 이는 시간복잡도 측면에서 불리하다. 이를 개선한 것이 equivalent class를 이용하는 방법이다. DFS나 BFS 알고리즘에서와 다르게 모든 방의 번호를 직접 바꿔주는 과정은 없다. 여기서는 linked list를 이용하여 두 방의 번호를 동치관계로 만들어준다. 이렇게 하고, 두 방의 연결여부를 판단하기 위해서 두 방의 번호가 완전히 일치한다고 생각하지 않고, 두 방의 번호가 동치관계인지 아닌지를 보면서 연결시켜준다. 연결이 되었다면 벽을 방문처리 해주어 다시 선택되는 일이 없도록 한다. 이렇게 해서 모든 벽이 선택되었다면 크기가 N\*M의 완전 미로가 생성 된 것이다. 연결과정에서 DFS나 BFS 알고리즘을 이용하는 경우, 시간복잡도는 O(edges\*N\*M)이고 공간복잡도는 O(N\*M)이다. DFS나 BFS 알고리즘은 최악의 경우 모든 방을 방문해야 할 수도 있기 때문에 시간복잡도가 큰편이다. 여기서 edges는 벽의 개수, N은 세로, M은 가로이다. 연결과정에서 equivalent class를 이용하는 경우에는 시간복잡도는 O(edges)이고 공간복잡도는 O(N\*M)이다. 연결과정에서의 시간복잡도가 O(1)이기 때문이다. 그러나 linked list를 새롭게 생성해야되는 만큼 메모리가 추가적으로 필요하다.

1. 본 실험에서는 완전 미로를 구현하기 위해서 Eller’s algorithm을 이용한다. 크기가 N\*M인 이차원 배열을 생성한다음 모든 값을 0으로 초기화한다. 그리고 반복문을 이용하여 가장 윗 행부터 차례로 아래로 내려오며 Eller’s algorithm을 수행한다. roomNum변수를 이용하여, 아직 방번호가 배정되어 있지 않은 상황에서 방 번호를 배정해준다. 그 다음 인접한 방들의 번호가 다르다면, 연결을 할 지 말지 무작위로 결정한다. 만약, 인접한 방들의 번호가 같다면 완전 미로의 특성 상 두 방을 연결하는 경로는 유일해야 하므로 연결하지 않는다. 현재 행의 모든 인접한 방을 확인하였다면, 그 다음 행으로 넘어가서 이전 행의 같은 열과 연결을 할 지 말 지 결정한다. 즉, 수직경로를 결정하는 과정이다. 연결을 결정하는 기준은 이전 행에 존재하는 각 집합들과 적어도 하나는 연결되어 있어야한다는 것이다. 연결이 되어 있지 않다면 두 방 사이를 연결하는 경로가 존재하지 않을 수 있는 위험이 있다. 하나의 집합에 여러개의 방이 연결되는 것도 가능하다. 유일한 경로를 체크하는 과정은 다시 현재 행에서 인접한 방들을 확인할 때 진행하기 때문이다. 그리고 마지막 줄에서는 서로 다른 집합에 속해있는 방들을 모두 연결시켜준다. 수평 연결이나 수직 연결이 되었으면 파일에 벽을 그리지 말고, 연결되어 있지 않다면 벽을 그려준다. 연결 여부를 판단하기 위해서 총 3개의 배열을 추가적으로 생성한다. 크기가 N인 배열 2개와 크기가 N\*M인 배열 하나이다. 여기서 N은 미로의 가로길이, M은 미로의 세로길이이다. 그리고 현재 행의 집합의 종류의 개수를 저장하는 types 라는 int 형 변수를 선언하여 이용한다. 크기가 M인 배열은 현재 행을 탐색하며 등장하는 집합들을 저장하는 배열이다. 그리고, 또 다른 크기가 N인 배열은 그 집합의 등장 횟수를 저장한다. 그리고, 수직경로를 판단하기 위해 현재 체크중인 집합이 행에 몇개 남았는지 알아야 하기 때문에, 현재 탐색중인 집합의 배열 상의 인덱스를 저장하는 크기가 N\*M을 이용한다. 이렇게 하면 매 행마다, 수평 경로와, 수직 경로를 잘 판단할 수 있게 된다. 이와 같은 자료구조와 알고리즘을 이용하면 시간복잡도는 O(N\*M)이 된다. 각 행마다 모든 열을 탐색하는데, 각 열에서 해당하는 값을 참조하거나 변경하는 과정은 모두 O(1)이기 때문이다. 그리고 공간복잡도는 미로의 방 번호를 저장 할 이차원 배열(N\*M), 방번호의 등장횟수를 저장하는 배열(N), 방번호를 저장하는 배열(N), 방번호에 접근하기 위해 방번호의 배열상 인덱스를 저장하는 배열(N\*M)가 필요하므로 O(N\*M)이다.