컴퓨터공학실험1 11주차 예비보고서

전공: 컴퓨터공학과 학년: 2 학번: 20191559 이름: 강상원

1. **문제 해결에서 언급한 미로 생성 알고리즘들에서 Eller's algorithm을 제외한 나머지 알고리즘 중 하나를 선택하여 이를 조사하고 이해한 후 그 방법을 기술하시오. (완전 미로 생성 알고리즘 들 중 하나를 선택하여 조사하시오)**

   완전 미로 생성 알고리즘들에 대해 조사해본 결과 Kruskal's algorithm, Prim's algorithm, Eller's algorithm 등이 존재한다는 사실을 알 수 있었다. 위 알고리즘들 중 Prim's algorithm에 대해 서술하고자 한다.

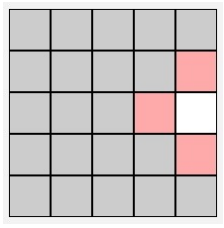
   Prim's algorithm은 자료구조 시간에 학습한 그래프 개념을 이용하게 된다. 미로 배열의 칸들을 정점으로, 칸들 사이의 관계(이어져 있는 여부)를 간선으로 생각한다. 그렇다면 전체 미로를 그래프(Graph) 구조 형태로 나타낼 수 있다. 이 그래프의 간선에는 가중치가 있고, 유향 그래프가 아닌 무향 그래프로 본다. (미로에서의 움직임은 벽 안이기만 하면 자유로우므로) 모든 정점을 포함하면서, 각 간선의 가중치의 합이 최소가 되는 부분 그래프를 찾는 방식으로 진행된다.

   임의의 한 노드를 골라 트리의 root로 만들고, 모든 노드가 트리에 추가되기 전까지 해당 트리와 연결된 간선 중 가중치가 가장 작은 간선을 추가로 연결한다.

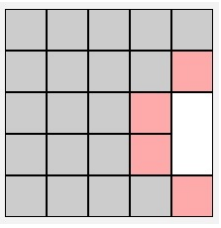
   시간 복잡도는 이진 힙으로 자료를 꾸몄을 때 $O(ElogV)$의 복잡도를 갖는다. (E: 간선의 수, V: 노드의 수) 피보나치 힙을 이용하면 $O(E + VlogV)$의 복잡도를 갖는다.

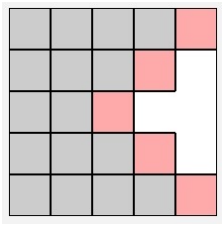
   미로의 구성 관점에서 본다면:

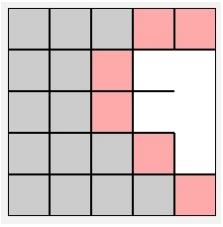
   (1) 먼저 기초가 되는 칸을 임의로 선택하고

   (2) 미로에 인접한 칸들 중 랜덤한 칸 하나를 선택하여 확장한다.

   (2)번 과정을 반복하여 미로를 랜덤으로 확장시킨다.

2. 본 실험에서 완전 미로(Perfect maze)를 만들기 위하여 선택한 알고리즘 구현에 필요한 자료구조를 설계하고 기술하시오. 설계한 자료구조를 사용하였을 경우 선택한 알고리즘 의 시간 및 공간 복잡도를 보이시오.

Eller's algorithm을 사용하였다.

자료구조 기술:

```
int WIDTH, HEIGHT;

vector<int> horizontal_wall; //가로 벽 그리는 배열->포인터
vector<int> vertical_wall; //세로 벽 그리는 배열->포인터
vector<int> maze; //전체 배열

int Id = 0; //방에 적힌 숫자 의미
```

int maze[WIDTH]: 미로의 전체 한 줄(열) 정보를 저장하는 배열. 각 칸 안의 Id (숫자)를 저장하는 역할을 한다.

int horizontal_wall[WIDTH]: 현재 열 (가로 줄)과 다음 열 사이에 있는 벽 정보 저장하는 동적 배열. 벽이 있을시 1, 없을 시 0 저장.

int vertical_wall[WIDTH-1]: 현재 열 안에서 서로 인접한 두 칸끼리의 벽을 의미한다. 사이의 벽이므로 WIDTH-1개 존재한다. horizontal_wall과 마찬가지로 벽이 존재하면 1, 존재하지 않으면 0을 저장한다.

위와 같이 구현할 경우 열마다 각 행에 대해 비교 연산을 진행해야 하므로 시간 복잡도는 O(WIDTH*HEIGHT)이다.

공간 복잡도는 O(WIDTH)가 될 것이다. (한 열씩만 저장하므로)

```
void first_rand() {
    srand(time(NULL));
    for (int i = 0; i < WIDTH - 1; i++) {
        vertical_wall[i] = rand() % 2; //0 아니면 1
        if (vertical_wall[i] == 1) { //1 일 경우, 막혀 있다는 의미이므로 그 이전
칸까지 모두 뚫는 처리.
            for (int j = (before_wall + 1); j <= i; j++)
                maze[j] = Id; //방에 적힌 숫자 통일

            before_wall = i; // 이전 벽 위치
            Id++; // 이제 다음 집합에 적힌 숫자는, 1 증가.
        }
    }
    for (int i = before_wall + 1; i < WIDTH; i++)
        maze[i] = Id; //마무리 처리->나머지 곳도 숫자 맞추기.
    Id++;
}
```