11주차 예비보고서

전공: 컴퓨터공학과 학년: 4학년 학번: 20212022 이름: 이예준

**1.**

미로를 생성하는 알고리즘 중에는 Binary Tree, Sidewinder 같은 알고리즘이 있지만 미로가

편향적이고, 일직선으로 뚫린 부분이 많은 단점이 있다.

그런 단점을 보완해주기 위한 알고리즘 중Recursive\_Backtracking라는 알고리즘이 있다.

사각형, 직사각형, 라인이(가) 표시된 사진

자동 생성된 설명 사각형, 라인, 직사각형, 스크린샷이(가) 표시된 사진

자동 생성된 설명

1. 먼저 모든 벽이 막혀있는 미로에 현재 셀인 초록색 셀을 시작 셀로 지정한다.

2. 그 다음 현재 셀과 이웃하는 셀들 중 하나를 임의로 선택한다.

(방문한 적이 없는 셀만 선택 가능하다)

3. 선택이 되면 현재 셀은 스택에 push한 뒤 현재 셀과 선택된 셀 사이의 벽을 뚫는다.

4. 선택한 셀을 현재 셀로 갱신한다.

1~4 과정을 현재 셀의 이웃한 셀들이 모두 방문한 셀일 때까지 반복해준다.

사각형, 직사각형, 라인이(가) 표시된 사진

자동 생성된 설명 사각형, 직사각형, 스크린샷, 라인이(가) 표시된 사진

자동 생성된 설명

이렇게 이동할 수 없는 상황이 되면 이름에서 보이는 것과 같이 Backtracking을 해준다.

스택에는 방문했었던 셀들이 push되어 있기 때문에 이동할 수 없는 상황에서는

스택에서 정보를 pop해주면서 경로를 되돌아 간다.

(경로를 되돌아가다가 이동할 수 있는 상황이 되면 다시 스택에 push를 해주면서 이동한다)

사각형, 직사각형, 라인이(가) 표시된 사진

자동 생성된 설명 사각형, 직사각형, 패턴이(가) 표시된 사진

자동 생성된 설명

직사각형, 사각형, 도표, 디자인이(가) 표시된 사진

자동 생성된 설명 직사각형, 사각형, 패턴, 디자인이(가) 표시된 사진

자동 생성된 설명

이렇게 스택에 있는 모든 경로 정보를 pop하면 미로가 완성된다.

**2.**

typedef struct {

int x, y;

} Position;

void Recursive\_Backtracking (char maze[Height][Width] , int x, int y) {

maze[y][x] = ‘ ‘;

for(int i = 3; i > 0; i--){

int j = rand() % (i+1);

Position tmp = array[i];

array[i] = array[j];

array[j] = tmp;

}

for(int i=0; i<4; i++){

int nx = x + directions[i].x;

int ny = y + directions[i].y;

int betweenX = x + directions[i].x / 2;

int betweenY = y + directions[i].y / 2;

}

if (isValid(nx, ny) && maze[ny][nx] == ‘#’) {

maze[betweenY][betweenX] = ‘ ‘;

recursiveBacktracking(maze, nx, ny);

}

}

시간 복잡도 : 각 셀을 한번씩 방문하며, 각 방문 시에는 최대 네 방향을 탐색하기 때문에

시간 복잡도는 O(n2)이다.

공간 복잡도 : 미로의 크기가 n2이고, 최악의 경우 재귀 호출 스택의 깊이가 n2가 되기 때문에

공간 복잡도가 O(n2)이다.