2020년도 2학기 컴퓨터공학설계및실험I

11주차 결과 보고서

20170175 김태안

1. 실습 목적

미로는 복잡한 길을 찾아 출발점부터 시작해 도착점까지 도달하는 퍼즐로 직사각형으로 배열된 방과 이러한 방들 사이를 막고 있는 벽으로 구성된다. 벽은 각 방의 상하좌우 사면에 존재할 수 있는데 인접한 두 방 사이에 벽이 없을 경우 두 방 사이를 자유로이 지날 수 있다. 본 프로젝트에서는 다음에 설명하는 완전미로를 만들 수 있는 여러 가지 방법을 알아보고 이를 직접 구현하기 위한 자료구조 설계 및 프로그래밍을 통해 미로를 생성해보도록 한다.

1. 실습 구현 내용

void MazeAlloc();

main에서 입력 받은 미로의 WIDTH와 HEIGHT에 맞게 cell 구조체의 2차원 배열 maze을 동적 할당한다. 그리고, cell 구조체의 변수 right, bottom, setNum을 -1로 초기화한다.

void MazeDealloc();

2차원 배열 maze의 동적 할당을 해제한다.

void MazeEllers();

Eller’s Algorithm을 통해 미로를 생성한다. 줄 단위로 집합 할당-오른쪽 경계선 그리기-집합 병합-아래쪽 경계선 그리기를 HEIGHT-1번만큼 실행한다. 집합 할당 단계에서는 위의 칸의 아래쪽 경계선이 없으면 위의 칸의 집합 번호를, 칸에 할당된 집합이 없으면 새 집합 번호를 부여한다. 오른쪽 경계선 그리기 단계에서는 다음 칸의 집합 번호가 현재 칸의 집합 번호와 같으면 경계선을 그리고, 아니면 랜덤하게 그리거나 그리지 않는다. 이때. 맨 오른쪽 벽의 오른쪽 경계선은 항상 존재해야 하도록 예외처리 한다 그리고, SetNumAdj 함수를 사용해 오른쪽 경계선이 없는 칸과 인접한 칸의 집합 번호를 통일한다. 아래쪽 경계선은 랜덤하게 그리되, 각 집합은 하나 이상의 칸이 아래쪽 경계선이 없도록 한다. 마지막 줄은 아래쪽 경계선은 모두 그리고, 인접한 칸과의 집합 번호 차이에 따라 오른쪽 경계선을 그리거나 지워 미로를 완성한다. 알고리즘은 줄 단위로 진행되며, 각 줄에서 실행되는 각 단계는 O(WIDTH)의 시간 복잡도를 가진다. 예외로 집합 병합 단계는 각 칸에 대해 시간 복잡도가 O(WIDTH)인 SetNumAdj 함수가 실행되므로 O(WIDTH2)의 시간 복잡도를 가진다. 따라서, Eller’s Algorithm의 시간 복잡도는 O(HEIGHT\* WIDTH2 +a\*HEIGHT\*WIDTH) = O(HEIGHT\*WIDTH2)이다, (a는 함수 병합 이외의 단계의 개수) 그리고, Eller’s Algorithm은 처음 할당된 미로 maze이외의 저장공간이 필요하지 않으므로 공간 복잡도는 O(HEIGHT\*WIDTH)이다.

void SetNumAdj(int height, int to, int from);

Eller’s Algorithm은 오른쪽 경계선을 그리지 않는다면 현재 칸과 다음 칸의 집합을 일치시켜 칸이 연결됨을 기록한다. SetNumAdj는 현재 줄을 탐색하며 다음 칸의 집합 번호를 가진 칸을 찾아 현재 칸의 집합 번호로 변경한다. maze의 모든 칸이 아닌 현재 줄만 변경하는 이유는 현재 줄에 존재하는 집합만이 다음 줄의 미로 생성에 사용되기 때문이다. 현재 줄을 모두 탐색하기 때문에 O(WIDTH)의 시간 복잡도를 가진다.

void Maze1D();

Eller’s Algorithm에서 가장 오른쪽의 칸과 가장 아래쪽의 칸은 다른 단계를 거친다. 따라서 가로나 세로가 1인 미로를 생성할 때 에러가 발생할 수 있다. 이를 막기 위해 가로나 세로가 1인 경우의 미로를 생성하는 함수를 따로 둔다. 이 경우. 미로는 항상 내부 경계선이 없는 일자 미로가 형성된다.

void MazePrint();

디버깅을 위해 사용한 함수로, 미로와 각 칸의 setNum을 콘솔에 출력한다.

void MazeOut();

maze.maz 파일을 생성하고, 생성한 미로를 파일에 쓴다. maze 2차원 배열을 모두 탐색하며 각 칸의 right, bottom 정보에 따라 미로를 쓰며, 코너는 +, 벽은 |, -로 표시한다.

1. 실습 환경

cspro.sogang.ac.kr

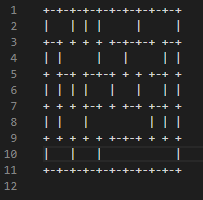
OS: Ubuntu 16.04.2 LTS (GNU/Linux 4/4/0-184-generic x86\_64)

Compiler: gcc version 5.4.0 20160609 (Ubuntu 5.4.0-6ubuntu1~16.04.12)

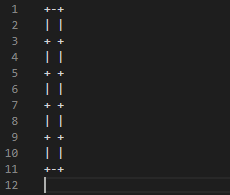
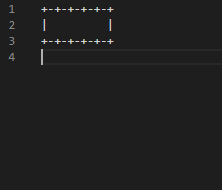
1. 실습 결과 및 분석



프로그램을 실행하면 다음과 같이 가로와 세로를 입력 받는다. maze.maz 파일이 생성되며, 내용은 다음과 같다.



이때, 미로는 순환하는 경로가 존재하지 않으며, 모든 임의의 두 점을 연결하는 경로가 하나인 완전 미로이다. 미로에 가로나 세로가 1인 입력을 넣은 결과는 아래와 같다.

1. 과제

**불완전 미로**

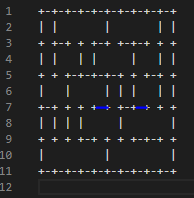
void MazeImperfect();

Eller’s 알고리즘을 통해 생성한 완전 미로를 불완전 미로로 바꾼다. 불완전 미로로 바꾸기 위해서는 미로의 개의 경계선을 랜덤하게 삭제해야 한다. rand 함수를 사용해 임의의 칸에 접근한 뒤, 오른쪽 경계선이나 아래쪽 경계선이 존재한다면 둘 중 하나를 지운다. 이를 번 반복해 불완전 미로를 생성한다.

int EraseWallNum();

WIDTH와 HEIGHT를 통해 를 계산해 반환한다.

프로그램을 실행하면 다음과 같이 순환 경로가 있는 미로가 생성된다. 이때. 파란색의 경로를 막으면 완전 미로가 생성되며, 이므로 알고리즘이 정상적으로 작동함을 확인할 수 있다.



1. 결론

이번 실험에서는 완전 미로와 불완전 미로를 생성해보았다. 완전 미로란 모든 임의의 두 구간을 연결하는 경로가 하나뿐인 미로로, Eller’s Algorithm을 사용해 구현할 수 있다. 불완전 미로란 임의의 두 구간을 연결하는 경로가 둘 이상인 경우가 존재하는 미로로 순환 구간이 존재한다. 이는 완전 미로에서 개의 경계선을 제거하여 구현할 수 있다.