전공 : 컴퓨터공학과 학년 : 4 학번 : 20212020 이름 : 박민준

1. 실험시간에 작성한 프로그램의 알고리즘과 자료구조를 요약하여 기술하시오. 완성한 알고리즘의 시간 및 공간 복잡도를 보이고 실험 전에 생각한 방법과 어떻게 다른지 아울러 기술하시오.

- 해당 프로그램은 Eller's Algorithm을 사용하여 주어진 너비(W)와 높이(H)로 미로를 생성하는 프로그램이다. 프로그램은 무작위로 벽을 제거하여 미로를 생성한다. 알고리즘과 자료구조를 요약하면 다음과 같다.

1) 자료구조

텍스트, 폰트, 스크린샷, 그래픽이(가) 표시된 사진

자동 생성된 설명

- 먼저, 미로의 너비와 높이 값을 저장할 int형 변수를 선언하고, 실제 미로의 형태를 저장할 2차원 배열 maze를 선언한다. maze 배열에서 벽은 '-', '|', '+'로 표현되고, 길은 ' '로 표현된다. 또한 해당 배열의 각 셀이 속한 집합을 저장하기 위한 2차원 배열 set을 선언한다. 이는 셀이 연결된 다른 셀들과의 그룹을 나타낸다.

- 미로의 너비와 높이는 사용자의 입력을 받아 값이 설정된다. 사용자 입력으로 6, 4를 입력 받으면 다음과 같은 5x5 크기의 미로가 생성된다.

스크린샷, 디자인이(가) 표시된 사진

자동 생성된 설명

2) 알고리즘

(1) 초기화 단계 (InitMaze 함수)

텍스트, 스크린샷, 디스플레이이(가) 표시된 사진

자동 생성된 설명

- InitMaze 함수는 미로의 초기 상태를 설정하고 각 셀에 고유한 집합 번호를 부여하는 역할을 한다. 이 함수는 미로의 형태를 초기화하는 부분과 집합 번호를 초기화하는 두 부분으로 나뉘어 있다.

- maze 벡터는 (H \* 2 + 1) x (W \* 2 + 1) 크기로 resize되며, '+'는 교차점, '-'는 가로 벽, '|'는 세로 벽을 나타낸다. 각 칸은 i와 j의 값에 따라 적절한 문자가 할당된다. set 벡터는 H x W 크기로 resize되며, 각 셀의 초기 집합 번호를 설정한다. 각 셀의 집합 번호는 i \* W + j로 설정되며, 0부터 시작하는 연속된 번호로 설정되기 때문에 각 셀은 고유한 집합 번호를 갖는다.

(2) 미로 생성 단계 (MakeMaze 함수)

텍스트, 스크린샷이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷, 디스플레이, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

- MakeMaze 함수는 미로를 생성하는 핵심 함수로, 벽을 무작위로 제거하여 미로를 만든다. 이 함수는 세로 벽 제거와 가로 벽 제거 두 부분으로 나뉜다. 이후 각 집합이 적어도 한 번은 아래로 연결되었는지 확인하고 이를 보장한다.

- 먼저, 현재 행의 각 셀을 순회하면서 세로 벽을 제거할지 결정한다. rand() % 2를 사용하여 무작위로 벽을 제거하며 마지막 행인 경우 무조건 벽을 제거한다. 만약 두 셀이 다른 집합에 속해 있다면, 벽을 제거하고 두 셀을 같은 집합으로 합친다. 합친 후, set 벡터를 업데이트하여 동일한 집합 번호를 할당한다. maze[i \* 2 + 1][j \* 2 + 2] 위치의 벽을 ' '로 설정하여 벽을 제거한다.

- 세로 벽을 제거한 이후, 각 행의 가로 벽을 무작위로 제거하여 다음 행과 연결한다. connected 벡터를 사용하여 각 집합이 적어도 한 번은 아래 행으로 연결되었는지 확인한다. 무작위로 벽을 제거하거나, 아직 연결되지 않은 집합이라면 벽을 제거한다. maze[i \* 2 + 2][j \* 2 + 1] 위치의 벽을 ' '로 설정하여 벽을 제거한다.

- 마지막으로 각 집합이 적어도 한 번은 다음 행과 연결되도록 보장한다. connected 벡터를 확인하여 연결이 보장되지 않은 집합은 강제로 벽을 제거하여 연결한다.

(3) 미로 출력 단계 (OutputMaze 함수)

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

- OutputMaze 함수는 maze 벡터를 파일로 출력한다. 각 행과 열을 순회하며 미로의 형태를 “maze.maz” 확장자 파일에 기록한다.

3) 시간 및 공간 복잡도

[시간 복잡도] : O(H \* W)

(1) 초기화 단계: O(H \* W)

- 각 셀에 대해 집합 번호를 설정하는 부분에서 H \* W의 시간이 소요된다. 미로의 초기 상태를 설정하는 부분에서 역시 (H \* 2 + 1) x (W \* 2 + 1) 크기의 벡터를 초기화한다.

(2) 미로 생성 단계: O(H \* W)

- 각 행에 대해 세로 벽을 제거하는 부분에서 W번의 반복이 일어난다. 가로 벽을 제거하는 부분에서도 W번의 반복이 일어난다. 따라서 전체적으로 H \* W번의 연산이 수행된다.

(3) 출력 단계: O(H \* W)

- 미로를 파일로 출력하는 부분에서 (H \* 2 + 1) x (W \* 2 + 1) 크기의 벡터를 순회하며 출력한다.

[공간 복잡도] : O(H \* W)

(1) 미로를 저장하는 벡터 maze: O(H \* W)

- 크기가 (H \* 2 + 1) x (W \* 2 + 1)인 벡터를 사용한다.

(2) 집합을 저장하는 벡터 set: O(H \* W)

- 크기가 H x W인 벡터를 사용한다.

4) 실험 전후 생각 차이점

- 실험 전 Eller's Algorithm이 간단하고 효율적인 알고리즘이라고 생각했으며, 각 행 별로 세로 벽과 가로 벽을 제거하여 미로를 생성하는 과정이 직관적이라고 생각했다. 그러나, 구현 과정에서 세로 벽 제거와 가로 벽 제거의 순서와 조건을 정확히 구현하는 것이 예상보다 까다로웠다. 세로 벽을 제거할 때, 두 셀이 다른 집합에 속하는지 확인하고 집합을 병합하는 과정에서 세밀한 조정이 필요했으며 가로 벽을 제거할 때, 각 집합이 적어도 한 번은 다음 행과 연결되도록 보장하는 부분에서 무작위성과 연결성을 적절히 조절해서 처리해야 했다.