11주차 결과보고서

전공: 경영학과 학년: 3학년 학번: 20190963 이름: 한다현

**1.**

실험시간에 작성한 프로그램의 알고리즘은 Eller의 알고리즘이고, 이 알고리즘에 사용된 자료구조는 정수형 이차원 배열이다. 이 이차원 배열은 미로를 저장하기 위한 자료구조이다. 전체 코드는 main 함수와 미로를 초기화하는 init\_maze 함수, 랜덤으로 완전 미로를 생성하는 random\_maze 함수, 그리고 미로를 maze.maz 파일에 출력하는 print\_maze 함수로 구성되어 있다. 먼저 main 함수에서는 사용자에게 미로의 행과 열의 개수를 입력 받는다. 이렇게 입력 받은 행과 열의 개수는 m\*n 미로에서 m과 n을 의미한다. 하지만 실제 미로를 출력하기 위해서는 미로의 모서리나 벽 또한 저장해야 하므로 이차원 배열 maze의 크기는 (col\*2+1, row\*2+1)이다.

사용자에게 미로의 크기를 입력 받은 후에는 미로를 초기화하는 init\_maze 함수를 호출한다. 이 함수에서 height와 width 크기의 이차원 배열인 maze를 동적 할당한다. 이 후 이중 for loop를 실행하여 maze의 초기 값을 저장한다. 첫 번째 for loop은 i가 0부터 heigth보다 작을 때까지 반복한다. 두 번째 for loop은 j가 0부터 width보다 작을 때까지 반복한다. maze의 짝수 행은 maze의 모서리와 벽을 저장해야하고, 홀수 행은 maze의 방과 벽을 저장해야 한다. 따라서 i와 j 모두 짝수이면 방의 모서리를 나타내는 ‘+’를, i는 짝수, j는 홀수이면 ‘-‘를 maze[i][j]에 저장한다. 또한 i는 홀수, j는 짝수이면 벽을 나타내는 ‘|’를 저장하고, i와 j가 모두 홀수이면 maze[i][j]에 방의 번호를 저장한다.

init\_maze 함수가 종료되면 main함수에서 랜덤한 완전 미로를 생성하는 random\_maze 함수를 호출한다. 이 함수에서는 먼저 maze의 방의 번호가 저장되어 있는 첫 번째 행에 대한 작업을 실행한다. 이 때 for loop이 사용되는데, i가 1부터 width-2보다 작을 때까지 i를 2씩 증가시키는 반복문이다. 반복문 내에서는 rand 함수를 사용해 0부터 9까지 임의의 숫자를 생성하고, 이 임의의 숫자가 5보다 작고 maze[1][i]와 maze[1][i+2]가 같지 않을 때 두 방의 번호를 같게 하고, 두 방 사이의 벽을 제거한다. 이 작업이 종료되면 나머지 작업을 수행한다. 나머지 작업은 현재 행의 아래쪽 벽을 제거하고, 다음 행의 측면 벽을 제거하는 작업이다. 이 작업을 수행하기 위해 3중 for loop을 사용하였다. 첫 번째 for loop은 i가 1부터 height-2보다 작을 때까지 i를 2씩 증가시킨다. 이 반복문 내에는 두 개의 for loop이 존재한다. 이 중 첫 번째 for loop은 i가 1이 아닐 때 실행되는데, k가 1부터 width-2보다 작을 때까지 k를 2씩 증가시킨다. 이 반복문은 두 방 사이의 벽을 제거하는데, 제거되는 과정에서 for 반복문을 통해 방의 번호가 동일한 방의 번호를 모두 수정한다. 그리고 이 과정이 끝나면 count 배열의 값을 초기화한다. 두 번째 for loop은 j가 1부터 width-1보다 작을 때까지 j를 2씩 증가시킨다. 이 반복문은 현재 행의 아래쪽 벽을 제거하는 반복문이다. 먼저 0부터 9 사이의 임의의 수를 생성하고, 이 수가 5보다 작으면 아래쪽 벽을 제거하고, 방 번호에 해당하는 count의 수를 1 증가시킨다. 임의의 수가 5보다 작은 경우에는 j+2가 width 이상이거나 maze[i][j]가 maze[i][j+2]와 동일하지 않을 때 방 번호에 해당하는 count의 값이 0이면 아래쪽 벽을 제거하고 count를 1 증가시킨다. 이와 같은 과정을 3중 for loop이 종료될 때까지 반복한다. 이 반복문이 종료되면 random\_maze 함수의 마지막 반복문이 실행된다. 이 반복문은 i가 1부터 width-2보다 작을 때까지 i를 2씩 증가시키며 이는 미로의 마지막 행의 측면 벽을 제거시키는 반복문이다. 먼저 maze[height - 2][i]이 maze[height - 2][i + 2] 같지 않으면 둘 사이의 벽을 제거하고, maze[height-2][i+2]와 동일한 번호를 모두 수정한다.

random\_maze 함수가 종료되면 main 함수에서 print\_maze 함수를 호출한다. print\_maze 함수에서는 먼저 maze.maz 파일을 쓰기 권한으로 오픈한다. 그 후 이중 for loop을 실행하는데, 첫 번째 for loop은 i가 0부터 height 미만까지 1씩 증가시키며, 두 번째 for loop은 j가 0부터 width 미만까지 1씩 증가시킨다. 이중 for loop 내에서는 maze[i][j]가 방 번호를 저장하고 있다면 ‘ ‘를 출력하고, 그렇지 않다면 ‘+’, ‘-‘, ‘|’를 각각 출력한다. 한 개의 행이 출력된 후에는 줄바꿈 문자를 출력하고 이중 반복문이 종료되면 파일을 닫고 함수를 종료한다.

이 프로그램에서는 3중 for loop이 사용되었기 때문에 시간 복잡도는 O(n^3)이고, 미로를 저장하기 위해 2차원 배열을 사용하였기 때문에 공간 복잡도는 O(n^2)이다. 실험 전에는 Kruskal의 알고리즘을 조사하였지만 실습에서는 Eller의 알고리즘을 사용하였다. Kruskal 알고리즘은 임의의 방을 선택하고, 4개의 방향에 있는 벽 중 하나를 제거한 후 두 방의 번호를 일치시킨다. 이 과정을 반복하며 최종적으로 하나의 번호만 남을 때 종료된다. Eller의 알고리즘은 첫 번째 행에 대한 작업을 먼저 수행한다. 첫 번째 방에서부터 순차적으로 측면 벽의 제거 여부를 결정하고, 그 후 아래쪽 벽의 제거 여부를 결정한다. 이 과정을 첫 번째 행부터 마지막 행까지 반복한다.