11주차 결과보고서

전공: 컴퓨터공학과 학년: 2학년 학번: 20231515 이름: 김다은

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

**1-1. 구현한 함수**

**(1) searchmaze(int x, int y, int target, int dst)**

미로의 벽을 제거한 경우, 새롭게 연결된 칸의 숫자들을 현재 칸의 숫자들로 바꾸어야 한다. searchmaze는 이 작업을 해주는 함수이다. target은 새롭게 연결된 칸의 숫자이고, dst는 현재 칸의 숫자이다. 또한 x, y는 새롭게 연결된 칸의 행렬 인덱스 값이다.

탐색을 구현하기 위해 DFS를 사용하였다. board[][] 값을 방문처리에 활용하였으며, 위에서 언급한 dx[], dy[]를 이용해 4방향(위, 아래, 왼쪽, 오른쪽)을 탐색하게 된다.

board[x][y] = dst; 를 통해 현재 x, y칸에 dst(현재 칸의 숫자)를 저장한다. 이는 해당 위치를 방문했다는 방문 처리 역할도 수행한다. 기준 칸으로부터 4방향(위, 아래, 왼쪽, 오른쪽)으로 탐색하여 해당 칸이 target 값을 가지는 경우 dst 값으로 바꾸어 저장하고 searchmaze함수를 재귀 호출한다.

void searchmaze(int x, int y, int target, int dst) {

    board[x][y] = dst;

    for(int k=0; k<4; k++) {

        int nx = x+dx[k];

        int ny = y+dy[k];

        if(0<=nx && nx<m && 0<=ny && ny<n && board[nx][ny] == target) {

            searchmaze(nx, ny, target, dst);

        }

    }

}

**(2) writefile(const char\* filename)**

Eller’s 알고리즘을 이용해 만든 미로를 filename의 파일로 내보내는 함수이다.

void writefile(const char\* filename) {

    ofstream fout;

    fout.open(filename, ios::out);

    for(int i=0; i<m\*2+1; i++) {

        for(int j=0; j<n\*2+1; j++) {

            fout << maze[i][j];

        } fout << "\n";

    }

    fout.close();

}

**(3). main 함수**

진행 순서는 다음과 같다.

1. rand 함수의 seed를 time으로 초기화한다.
2. n, m값을 입력받는다.
3. n, m값을 토대로 maze와 board에 메모리를 동적할당한다.
4. Eller’s Algorithm
   1. 미로의 뼈대(벽)을 maze[][]에 저장한다.
   2. board의 0번째 row를 1부터 row까지의 수로 각각 저장한다.
   3. 반복문
      1. 랜덤 값이 1이고 해당 row에서 기준 칸과 오른쪽 칸에 저장된 숫자가 서로 다른 경우, 오른쪽 벽을 삭제하고 searchmaze를 호출한다.
      2. 랜덤 값이 1인 경우 아래 벽을 삭제하고, 아래 칸의 숫자로 현재 칸의 숫자를 저장한다. 각 숫자별로 적어도 1개는 아래 벽을 삭제해야 한다.
   4. board의 마지막 row에서 기준 칸과 오른쪽 칸에 저장된 숫자가 서로 다른 경우 벽을 삭제하고 searchmaze함수를 호출한다.
5. writefile 함수를 호출하여, 완성된 미로를 maze.maz 파일로 저장한다.
6. maze와 board에 동적할당 했던 메모리를 해제한다.

int main() {

    int number = 1;

    int randnum;

    srand((unsigned int)time(NULL));

    cin >> n >> m;

*// 메모리 동적 할당*

    maze = new char\*[m\*2+1];

    for(int i=0; i<m\*2+1; i++) {

        maze[i] = new char[n\*2+1];

    }

    board = new int\*[m];

    for(int i=0; i<m; i++) {

        board[i] = new int[n];

    }

*// 미로 뼈대 그리기*

    for(int i=0; i<m\*2+1; i++) {

        for(int j=0; j<n\*2+1; j++) {

            if(i%2 == 0) {

                if(j%2 == 0) maze[i][j] = '+';

                else maze[i][j] = '-';

            }

            else {

                if(j%2 == 0) maze[i][j] = '|';

                else maze[i][j] = ' ';

            }

        }

    }

*// 0번째 row 초기화*

    for(int j=0; j<n; j++) {

        board[0][j] = number++;

    }

    for(int i=0; i<m-1; i++) {

*// 오른쪽 벽 뚫기*

        for(int j=0; j<n-1; j++) {

            randnum = rand()%2;

            if(randnum == 1 && board[i][j] != board[i][j+1]) {

                maze[i\*2+1][(j+1)\*2] = ' ';

                searchmaze(i, j+1, board[i][j+1], board[i][j]);

            }

        }

*// 아래 벽 뚫기*

        for(int j=0; j<n; j++) {

            randnum = rand()%2;

*// 각 숫자마다 적어도 1칸은 아래 벽을 뚫어야 하므로*

            if(board[i][j] != board[i][j+1]) {

                randnum = 1;

            }

            if(randnum == 1) {

                maze[(i+1)\*2][j\*2+1] = ' ';

                board[i+1][j] = board[i][j];

            }

            else {

                board[i+1][j] = number++;

            }

        }

    }

    for(int j=0; j<n-1; j++) {

        if(board[m-1][j] != board[m-1][j+1]) {

            maze[(m-1)\*2+1][(j+1)\*2] = ' ';

            searchmaze(m-1, j, board[m-1][j+1], board[m-1][j]);

        }

    }

*// 파일로 저장*

    writefile("maze.maz");

*// 메모리 할당 해제*

    for(int i=0; i<m\*2+1; i++) {

        delete[] maze[i];

    }

    delete[] maze;

    for(int i=0; i<m; i++) {

        delete[] board[i];

    }

    delete[] board;

}

**1-2. 시간복잡도**

반복문에서 n\*m의 반복이 이루어지고, searchmaze 함수에서 최악의 경우 시간복잡도가 O(n\*m)이므로 총 시간복잡도는 O((n\*m)^2)이다.

**1-3. 공간복잡도**

maze에 (2\*n+1)\*(2\*m+1)\*sizeof(char)의 메모리를 할당하였고, board에 n\*m\*sizeof(int)의 메모리를 할당하였다. 따라서 공간복잡도는 O(n\*m)이다.

실습한 내용은 예비보고서에서 계획했던 내용과 동일했다.