|  |
| --- |
|  |
| **컴퓨터공학실험1 (1반) 12주차 결과보고서 (미로(1))** |
|  |
|  |
|  |
| 컴퓨터공학과 20161620 이수연 |

**컴퓨터공학실험1 // 미로(1) 결과보고서**

전공: 컴퓨터공학과 학년: 2학년 학번: 20161620 이름: 이수연

**1. 실험시간에 작성한 프로그램의 알고리즘과 자료구조를 요약하여 기술한다. 완성한 알고리즘의 시간 및 공간복잡도를 보이고 실험 전에 생각한 방법과 어떻게 다른지 아울러 기술한다.**

**<자료구조>**

#define WIDTH 300

#define HEIGHT 300

int maze[HEIGHT][WIDTH] = { 0 }; // 미로 2차원 배열 선언

int w[HEIGHT][WIDTH] = { 0 }; // maze의 복사판

int set[HEIGHT\*WIDTH] = { 0 }; // 저장된 id를 저장할 1차원 배열

**<알고리즘>**

for (i = 0; i<m - 1; i++)

{

for (j = 0; j<n - 1; j++)

{

if (maze[i][j] != maze[i][j + 1])

{

int isopen = rand() % 2;

if (isopen)

{

checklink(i, j + 1, maze[i][j + 1], maze[i][j]);

w[i][j] = 1;

}

}

}

for (j = 0; j<n; j++)

{

set[maze[i][j]] = 1;

}

int preset = maze[i][0];

for (j = 0; j <= n; j++)

{

int isopen = rand() % 2;

if (isopen)

{

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

set[maze[i][j]] = 0;

w[i][j] = w[i][j] ? 3 : 2;

}

if (preset != maze[i][j])

{

if (set[preset])

{

maze[i + 1][j - 1] = preset;

set[preset] = 0;

w[i][j - 1] = w[i][j - 1] ? 3 : 2;

}

preset = maze[i][j];

}

}

for (j = 0; j<n; j++)

{

if (maze[i + 1][j] == 0)

{

maze[i + 1][j] = setcnt++;

}

}

}

for (j = 0; j<n - 1; j++)

{

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

{

checklink(m - 1, j + 1, maze[m - 1][j + 1], maze[m - 1][j]);

w[m - 1][j] = 1;

}

}

벽을 뚫을 지 여부는 rand 함수를 이용하고 w[][]와 set[] 배열에 저장된 수를 보면서 조건문을 사용하여 같은 id의 방은 적어도 하나의 벽이 뚫리게 만든다.

\* 시간 복잡도: O(DRW000014145719 ) ; 이중 포문을 돈다.

\* 공간 복잡도: O(HEIGHT \* WDITH)

실험 전에는 미로를 완성한 뒤 한번에 프린트 해주려고 하였으나, 실습 때는 반복문을 이용하여 한 줄씩 미로를 연결해준 후 프린트 한 점이 달랐다. 그 외에 알고리즘의 측면에서는 크게 다른 점은 없었다.

**2. 숙제 문제를 해결하기 위한 알고리즘 및 자료구조를 요약하여 기술하시오. 시간 및 공간 복잡도를 보인다.**

실습 시간에 했던 완전 미로를 기반으로 하여 기존의 완전 미로를 불완전 미로로 만들어야 했다. 따라서 다음과 같은 코드를 추가하여 순환되는 경로가 생기게끔 하였다.

int min = n;

int count=0;

if (min > m)

min = m;

min /= 2;

while (count!= min)

{

int x = rand() % (n - 1);

int y = rand() % (m - 1);

if (w[y][x] == 0)

{

int wall = rand() % 2 + 1;

w[y][x] = wall;

count++;

}

else if (w[y][x] == 1 || w[y][x] == 2)

{

w[y][x]++;

count++;

}

}

실습 때 제출한 코드에 위의 코드를 추가한 것이므로 시간복잡도와 공간복잡도는 실험때와 크게 달라지지 않는다.

(끝)