11주차 결과보고서

전공: 컴퓨터공학과 학년: 2학년 학번: 20191619 이름: 이동석

**1. Eller’s 알고리즘**

자료구조는 미로를 구현하기 위해 너비와 높이만큼의 2차원 포인터 배열을 사용했다.

int\*\* map = malloc(sizeof(int\*) \* h);

for (int i = 0; i < h; i++)

map[i] = malloc(sizeof(int) \* w);

for (int i = 0; i < h; i++)

for (int j = 0; j < w; j++)

map[i][j] = (w+1)\*i + 1 + j;

우선 각 배열에 각자 다른 집합임을 표현하기 위해서 차례대로 값을 할당해준다. 값을 할당해준 후, time.h 라이브러리의 rand 함수를 이용해 1이면 TRUE로, 0이면 FALSE로 설정했다. Eller’s 알고리즘에 따라 첫번째 배열의 set을 랜덤하게 합쳐준다.

for (int i = 0; i < h; i++) {

for (int j = 0; j < w; j++) {

ran = rand() % 2;

if (i == 0) { // 첫시행시 set 랜덤으로 합치기

if (map[i][j] != map[i][j + 1] && j + 1 < w) {

if (ran == TRUE) {

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

}

}

else {

if (ran == TRUE) { // true일 경우 수직경로 생성

map[i][j] = map[i - 1][j];

flag = TRUE;

}

else if (map[i - 1][j] != map[i - 1][j + 1] && flag == FALSE) { // 수직경로가 아직 생성되지는 않았는데 다음 set과 현재 set이 다를 경우

map[i][j] = map[i - 1][j];

}

else if (j + 1 == w - 1 && flag == FALSE) { // 마지막 칸에 왔는데도 아직 수직경로가 없을 경우

map[i][j] = map[i - 1][j];

}

}

if (flag == TRUE && map[i][j] != map[i][j + 1]) flag = FALSE; // flag 초기화

}

}

첫번째 줄을 랜덤하게 합쳐주고 나면, 각 set에 대해서 최소1개의 수직경로를 랜덤하게 생성해주어야 한다. Int 형 변수 flag를 이용해 최소 1개인지 아닌지를 판별했다. 마찬가지로, rand 함수를 통해 랜덤함을 표현했다. 수직경로는 기본적으로 TRUE일 경우 생성하며, 이때 flag의 값 역시 TRUE로 바꾸어준다. 만약, 다음 집합set과 현재 집합 set이 다르지만, 현재 flag가 FALSE 인 경우에 수직경로를 생성해준다. 또한, 마지막 칸이지만, 역시 flag가 FASLE일 경우 생성해준다.

수직경로를 모두 생성해주고 나면, 다시 랜덤하게 set을 합쳐주어야한다. 인접한 집합 set이 다르면서 랜덤값이 TRUE일 경우에만 서로 셀을 합쳐준다. 하지만, 마지막 줄의 경우에는 인접한 셀이 다르면 모두 합쳐준다.

for (int k = 0; k < w - 1; k++) {

int ran = rand() % 2;

if (i == h - 1) {

if (map[i][k] != map[i][k + 1]) { update\_field(map, i, w, map[i][k + 1], map[i][k]);

}

}

else {

if (map[i][k] != map[i][k + 1] && ran == TRUE) { update\_field(map, i, w, map[i][k + 1], map[i][k]);

}

}

}

}

update\_field 함수는 인접한 셀을 합치게 되면, 현재 미로상에 설정되어있는 모든 집합set들을 같은 값으로 만들어주는 함수이다. 하지만, 굳이 모든 배열을 확인할 필요가 없다. 미로는 현재줄과 그 다음줄에만 영향을 주기 때문에 현재 줄만 바꾸어주면 된다. 따라서, update\_field의 함수의 경우 시간복잡도가 O(w)가 된다. (w는 미로의 너비)

위 Eller’s 알고리즘의 경우 공간복잡도는 당연하게도 O(h\*w) 이다. 시간복잡도의 경우에는 모든 배열을 확인해주고, 한 줄이 끝날 때마다 update함수를 호출하게 되므로 O(h\*w\*w)가 된다.

실험전에 생각한 알고리즘과 완벽하게 같은 알고리즘으로 작성하였고, 시간복잡도와 공간복잡도 역시 변함 없이 작성되었다.