풀이과정

기본적으로 최고의 만족도를 얻기 위해서는 모든 칸을 전부 지나가는 게 이상적이다. 다만 출발점과 도착점이 좌상/우하로 고정된 상태에서는 전부 지나가게 될 경우 도착점까지 이어지지 못하는 경우가 생긴다.

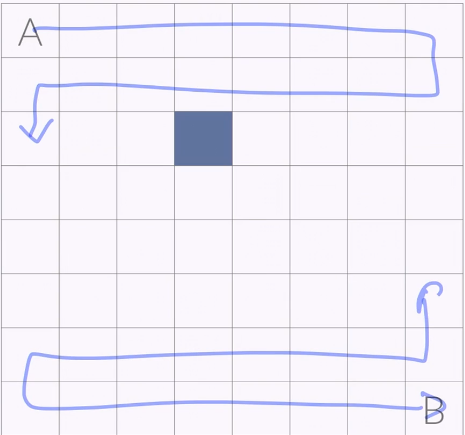
1. 행이 홀수인 경우 : 우측 끝->하강->좌측 끝->하강의 반복으로 전부 완주 가능
2. 열이 홀수인 경우 : 하강 끝->우측->상승 끝->우측의 반복으로 전부 완주 가능
3. 행과 열이 모두 홀수일 때 : 1),2)와는 달리 모든 곳을 전부 완주가 불가능하다.(가로로 완주하려하면 우하가 아니라 좌하가 도착점이 되버린다.) 완주를 하기 위해서는 출발/도착점과 대각에 있지 않은(체크무늬의 같은 영역에 있지 않은) 한 개의 좌표를 무시하고 진행해야한다. **최고의 만족도를 얻어야하기 때문에 이 문제에서는 가장 작은 만족도를 가진 위의 조건을 만족하는 위치를 무시해야한다.**

해당 좌표를 무시하기 위해서는 결국 2행은 기본적으로 필요하다.(좌표를 뛰어 넘을 수는 없으므로) 따라서 최저 만족도의 좌표 기준 2행만 남기고 1)을 통해 이동경로를 저장한다. 이후 2줄만 남은 부분은 이번에는 2행만 남도록(2x2가 되도록) 하고 나머지는 2)를 통해 이동경로를 저장한다.(어차피 2행짜리이기 때문에 하->우->상->우 4개의 경로만 반복한다.)

최저 만족도의 좌표는 출발/도착점과 대각에 있지 않은(체크무늬의 같은 영역에 있지 않은) 좌표이므로 나올 수 있는 경우의 수는 아래 2가지이다. 남은 2x2는 2가지 경우의 최저 만족도 좌표를 건드리지 않고 진행하도록 작성한다.



중간에 2줄만 남기는 방법은 시작점 기준으로 길을 뻗어 나가고 반대쪽에서도 마찬가지로 종료점을 기준으로 길을 뻗어 나가면 된다.



#include <iostream>

#include <algorithm>

using namespace std; int happy[1000][1000];

void append(string& ans, char c, int end) {

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

ans += c;

}

}

int main() {

int R, C;

cin >> R >> C;

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

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

cin >> happy[i][j];

if (R % 2 == 1) { //행이 홀수일 경우

string ans = "";

bool r = true; //진행방향 스위치(행마다 좌우로 왔다갔다 하면서 진행하기 때문에)

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

if (r) {

append(ans, 'R', C - 1);

r = !r;

}

else {

append(ans, 'L', C - 1);

r = !r;

}

if (i + 1 < R) //두 줄씩 진행하므로 우->아래->좌로 진행하는 듯

ans += 'D';

} cout << ans << '\n';

}

else if (C % 2 == 1) { //행이 짝수, 열이 홀수일 경우

string ans = "";

bool d = true; //진행방향 스위치(상/하)

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

if (d) {

append(ans, 'D', R - 1);

d = !d;

}

else {

append(ans, 'U', R - 1);

d = !d;

}

if (i + 1 < C) //두 줄씩 진행하므로 아래->옆으로->위로 진행하는 듯

ans += 'R';

}

cout << ans << '\n';

}

else { //열과 홀이 모두 짝수인 경우

int x = 0, y = 1; //시작점 주위의 좌표를 기준으로 체크무늬가 그려지는 위치 중에서 제일 만족도가 낮은 위치는 하나 걸러야 도착점(우하단)에 도달할 수 있다. 위처럼 하면 좌하단으로 내려가게 됨

int minVal = happy[0][1]; //최소 만족도를 가지는 좌표를 찾는다.(해당 좌표는 지나가지 않기 위해서)

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

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

if ((i + j) % 2 == 1 && minVal > happy[i][j]) {

minVal = happy[i][j];

x = i; y = j;

}

}

}

string ans = "", revAns = ""; //시작부분에서 진행, 도착부분에서 진행

int sr = 0, sc = 0, er = R - 1, ec = C - 1; //열과 행 시작/종료 변수인 듯

while (er - sr > 1) { //행이 2행보다 많이 남아있으면 계속 진행(최소 만족도 좌표가 영역에 포함되지 않는 선에서 진행)

if (sr + 1 < x) { //시작부분에서 뻗어야하는 진행라인에 최소만족도 좌표가 포함되지 않는 경우 진행

append(ans, 'R', C - 1); //우측 끝까지->하->좌측 끝까지->하강

append(ans, 'D', 1);

append(ans, 'L', C - 1);

append(ans, 'D', 1);

sr += 2;

}

if (er - 1 > x) { //끝부분에서 뻗어야하는 진행라인에 최소만족도 좌표가 포함되지 않는 경우 진행

append(revAns, 'R', C - 1); //하->왼->하->우(나중에 반전되면)

append(revAns, 'D', 1);

append(revAns, 'L', C - 1);

append(revAns, 'D', 1);

er -= 2;

}

}

while (ec - sc > 1) { //행이 2행만 남았을 경우 이제는 열단위로 진행한다.(최소 만족도 좌표가 있는 2x2배열이 될때까지 진행)

if (sc + 1 < y) { //시작부분에서 뻗어나감

append(ans, 'D', 1); //하->우->위->우

append(ans, 'R', 1);

append(ans, 'U', 1);

append(ans, 'R', 1);

sc += 2;

}

if (ec - 1 > y) { //끝부분에서 뻗어나감

append(revAns, 'D', 1); //우->위->우->하(나중에 반전되면)

append(revAns, 'R', 1);

append(revAns, 'U', 1);

append(revAns, 'R', 1);

ec -= 2;

}

}

//2x2에서 최소만족도 좌표가 나올수 있는 부분은 우상/좌하 둘 중 하나인데 그에 따라 진행방향을 결정한다.

if (x == sr + 1) {

append(ans, 'R', 1);

append(ans, 'D', 1);

}

else if (x == er - 1) {

append(ans, 'D', 1);

append(ans, 'R', 1);

}

reverse(revAns.begin(), revAns.end()); //끝에서부터 뻗은 경로는 합치기 위해서 반전시켜줘야 한다.

ans += revAns;

cout << ans << '\n';

}

return 0;

}