Level 4

Cleaning Robot

We have to plan a path for a cleaning robot to clean a rectangular room floor of size NxM. The room floor paved with square tiles whose size fits the cleaning robot (1 × 1). There are clean tiles and dirty tiles, and the robot can change a dirty tile to a clean tile by visiting the tile. Also there may be some obstacles (furniture) whose size fits a tile in the room. If there is an obstacle on a tile, the robot cannot visit it. The robot moves to an adjacent tile with one move. The tile onto which the robot moves must be one of four tiles (i.e., east, west, north or south) adjacent to the tile where the robot is present. The robot may visit a tile twice or more.

Your task is to write a program which computes the minimum number of moves for the robot to change all dirty tiles to clean tiles, if ever possible.

Time limit: 1s (C/C++), 2s (Java)

Submit limit: 10 times

Example:

The following is a room of size 5x7, with 3 dirty tiles, and 0 furniture. The answer for this case is 8.

Input

The input consists of multiple maps, the first line is the number of test case T (T < = 50).

Each test case begins with N and M representing the size of the room. ( 5 =< N, M <= 100)

The next N line representing the arrangement of the room with following describe:

0 : a clean tile

1 : a dirty tile

2 : a piece of furniture (obstacle)

3 : the robot (initial position)

In the map the number of dirty tiles does not exceed 10 and there is only one robot.

Output

Print each test case on two lines, the first line of each test case is "Case #x", where x is the test case number. The next line is the minimum number of moves for the robot to change all dirty tiles to clean tiles. If the map includes dirty tiles which the robot cannot reach, your program should output -1.

Sample

Input

5

5 7

0 0 0 0 0 0 0

0 3 0 0 0 1 0

0 0 0 0 0 0 0

0 1 0 0 0 1 0

0 0 0 0 0 0 0

5 15

0 0 0 0 2 0 2 0 0 0 0 1 2 0 1

0 0 0 1 0 2 0 2 2 0 1 2 0 0 0

2 1 0 2 0 1 0 2 0 0 0 0 0 0 0

0 0 0 1 0 2 0 0 1 2 0 0 2 0 0

0 2 1 0 2 0 0 0 0 0 3 0 0 0 0

...............

Output

Case #1

8

Case #2

38

Case #3

37

Case #4

-1

Case #5

49

============================

#include<iostream>

using namespace std;

const int maxn = 305;

int map[maxn][maxn];

int N, M;

int dx[] = {-1,0,0,1};

int dy[] = {0,-1,1,0};

bool visit[maxn][maxn] = {false};

int k;//bien dem luu vi tri robot va o ban

int dis[maxn][maxn];//mang luu khoag cach ngan nhat giua cac o ban

int check[20];

struct cell{

int x, y;

int distance;

cell(){};

cell(int x, int y, int distance): x(x), y(y), distance(distance){}

};typedef struct cell Cell;

const int Q = 90009;

Cell queue[Q];

Cell disty[11]; // queue luu vi tri robot va vi tri o ban

int front = -1;

int rear = -1;

void Init(){

front = -1;

rear = -1;

}

bool isEmpty(){

return rear == front;

}

void Push(Cell x){

if(rear == Q - 1)rear = -1;

rear++;

queue[rear] = x;

}

Cell Pop(){

if(front == Q - 1)front = -1;

front++;

return queue[front];

}

bool CheckVtri(int x, int y){

if(x <= N && x > 0 && y > 0 && y <= M)return true;

return false;

}

void Reset(){

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

{

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

{

visit[i][j] = false;

}

}

}

int BFS(int x\_start, int y\_start, int x\_end, int y\_end){

Init();

Reset();

Push(Cell(x\_start, y\_start, 0));

visit[x\_start][y\_start] = true;

while (!isEmpty())

{

Cell temp = Pop();

//check neu o hien tai bang voi o dich, return khoang cach

if(temp.x == x\_end && temp.y == y\_end)return temp.distance;

for (int i = 0; i < 4; i++)

{

int x = temp.x + dx[i];

int y = temp.y + dy[i];

if(CheckVtri(x, y)){

if(visit[x][y] == false && map[x][y] != 2){

visit[x][y] = true;

Push(Cell(x, y, temp.distance+1));

}

}

}

}

return -1;

}

int kq = 1000000;

void backTracking(int count, int parent, int sum){

int i;

if(count == k){

if(kq > sum)

{

kq = sum;

}

return;

}

else{

for (i = 1; i < k; i++)

{

if(check[i] == 0 && kq > sum){

check[i] = 1;

count++;

backTracking(count, i, sum + dis[parent][i]);

//Backtrack

check[i] = 0;

count --;

}

}

}

}

int main(){

int T;

cin>>T;

int flag;

for (int tc = 1; tc <= T; tc++)

{

k = 1;

flag = 1;

kq = 1000000;

cin>>N>>M;

for (int i = 1; i <= N; i++)

{

for (int j = 1; j <= M; j++)

{

cin>>map[i][j];

if(map[i][j] == 1){

disty[k].x = i;

disty[k].y = j;

k++;

}

else if(map[i][j] == 3){

disty[0].x = i;

disty[0].y = j;

}

}

}

cout<<"Case #"<<tc<<endl;

//BFS tinh truoc matrix khoang cach ngan nhat giua cac dinh

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

{

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

{

if(i<j && flag == 1){

dis[i][j] = BFS(disty[i].x, disty[i].y, disty[j].x, disty[j].y);

dis[j][i] = dis[i][j];

if(dis[i][j] == -1){

cout<<"-1"<<endl;

flag = 0;

break;

}

}

if(flag == 0)break;

}

}

//Backtrack de tim thu tu di den cac vi tri ban, sao cho duong di la nho nhat

if(flag == 1){

for (int i = 1; i < k; i++)

{

check[i] = 0;

}

backTracking(1, 0, 0);

cout<<kq<<endl;

}

}

return 0;

}

1