Mario Climb  
  
0 0 1 3  
1 1 1 0  
0 0 0 0  
2 1 1 1

Mario cần phải di chuyển từ vị trí có giá trị bằng 2 và ăn vàng ở ô có giá trị bằng 3  
0 là những ô Mario không thể qua  
1 là nhữngô Mario có thể qua  
2 là vị trícủa Mario  
3 là vị trí Mario cần di chuyển đến  
Các vị trí này được thể hiện bằng ma trận NxM( 2<=N,M<=50)  
Mario có thểdi chuyển theo hàng ngang hoặc hàng dọc  
Hàng ngang mario chỉ nhảy được tối đa 1 bước  
Hàng dọc mario có thể nhảy được “h” bước  
Tìm bước nhảy “h” tối thiểu để Mario có thể ăn được vàng  
Sample Input  
3  
5 8  
1 1 1 1 0 0 0 0  
0 0 0 3 0 1 1 1  
1 1 1 0 0 1 0 0   
0 0 0 0 0 0 1 0  
2 1 1 1 1 1 1 1  
5 6  
0 1 1 1 0 0  
3 1 0 1 1 0  
0 0 0 0 1 1  
0 0 0 0 0 1  
2 1 1 1 1 1  
9 13  
0 1 1 1 1 1 1 1 1 1 1 1 1   
1 1 0 0 0 0 0 0 0 0 0 1 1   
0 0 0 0 0 0 0 0 0 0 0 0 0   
0 0 0 0 0 0 0 0 0 0 0 0 0   
1 1 0 0 0 0 0 0 0 0 0 1 3   
0 0 0 0 0 0 0 0 0 0 0 0 0   
1 1 0 0 0 0 0 0 0 0 0 0 0   
0 0 0 0 0 0 0 0 0 0 0 0 0  
2 1 1 1 1 1 1 1 1 1 1 1 1  
Sample output  
Case #1  
2  
Case #2  
1  
Case #3  
3[#include](tg://search_hashtag?hashtag=include) <iostream>  
[#define](tg://search_hashtag?hashtag=define) SIZE 13  
using namespace std;  
  
int M,N;  
int Arr[SIZE][SIZE];  
int Visited[SIZE][SIZE];  
int Answer;  
  
  
int main()  
{  
int tc,TestCase, output = 0;  
  
freopen("Text.txt", "r", stdin);  
freopen("Text1.txt", "w", stdout);  
cin>>TestCase;  
  
for(tc=0; tc<TestCase; tc++)  
{  
cin >> N >> M;  
for(int i =0; i < N; i++){  
for(int j = 0; j < M; j++){  
cin >> Arr[i][j];  
}  
}  
bool flag = false;  
Answer = 0;  
  
while ( Answer < 3)  
{  
if(flag)  
break;  
Answer++;  
for(int i =0; i < N; i++){  
for(int j = 0; j < M; j++){  
Visited[i][j] = Arr[i][j];  
}  
}  
int Q[2000][2];  
int rear = 0, front = 0;  
  
Q[rear][0] = N-1;  
Q[rear][1] = 0;  
Visited[N-1][0] = 0;  
rear++;  
int a, b, x, y;  
  
int dX[4] = {0, -Answer, 0, Answer};  
int dY[4] = {-1, 0, 1, 0};\  
while(rear > front){  
a = Q[front][0];  
b = Q[front][1];  
front++;  
for(int k = 0; k < 4; k++){  
y = b + dY[k];  
if(dX[k] > 0){  
for(int t = 1; t <= dX[k]; t++){  
x = a + t;  
if(x >=0 && x < N && y >= 0 && y < M && Visited[x][y] != 0 ){  
if(Visited[x][y] == 3){  
rear = front;  
flag = true;  
//cout << " 123 "<< endl;  
break;  
}  
else{  
Visited[x][y] = 0;  
Q[rear][0] = x;  
Q[rear][1] = y;  
rear++;  
//cout << Answer <<" "<< x <<" " << y<< endl;  
}  
}  
}  
if(flag){  
break;  
}  
}  
else if(dX[k] < 0){  
for(int t = -1; t >= dX[k]; t--){  
x = a + t;  
if(x >=0 && x < N && y >= 0 && y < M && Visited[x][y] != 0 ){  
if(Visited[x][y] == 3){  
rear = front;  
flag = true;  
//cout << " 123 "<< endl;  
break;  
}  
else{  
Visited[x][y] = 0;  
Q[rear][0] = x;  
Q[rear][1] = y;  
rear++;  
//cout << Answer <<" "<< x <<" " << y<< endl;  
}  
}  
}  
if(flag){  
break;  
}  
}  
else{  
x = a;  
if(x >=0 && x < N && y >= 0 && y < M && Visited[x][y] != 0 ){  
if(Visited[x][y] == 3){  
rear = front;  
flag = true;  
//cout << " 123 "<< endl;  
break;  
}  
else{  
Visited[x][y] = 0;  
Q[rear][0] = x;  
Q[rear][1] = y;  
rear++;  
//cout << Answer <<" "<< x <<" " << y<< endl;  
}  
}  
}  
if(flag){  
break;  
}  
}  
}  
  
}  
  
  
  
////////////////////////////////  
  
  
  
////////////////////////////////  
  
  
  
cout<< "#" << tc+1 <<" "<< Answer << endl;  
}  
  
return 0;  
}

Code Java

package Mario;  
  
import java.io.FileInputStream;  
import java.util.Scanner;  
  
public class Mario {  
static int N, M;  
static int[][] maze, Vs;  
static int xS, yS, xE, yE;  
static int[][] Q;  
static int front, rear;  
static int[] Dx = { -1, 0, 0, 1 };  
static int[] Dy = { 0, 1, -1, 0 };  
  
public static void main(String args[]) throws Exception {  
System.setIn(new FileInputStream("src/Mario/input"));  
Scanner sc = new Scanner([System.in](http://system.in/));  
int T = sc.nextInt();  
for (int test = 1; test <= T; test++) {  
N = sc.nextInt();  
M = sc.nextInt();  
maze = new int[N][M];  
Vs = new int[N][M];  
for (int i = 0; i < N; i++) {  
for (int j = 0; j < M; j++) {  
maze[i][j] = sc.nextInt();  
if (maze[i][j] == 2) {  
xS = i;  
yS = j;  
}  
if (maze[i][j] == 3) {  
xE = i;  
yE = j;  
}  
}  
}  
Q = new int[100000][2];  
System.out.println("Case #" + test);  
  
BT(0);  
}  
}  
  
static void BT(int h) {  
  
front = 0;  
rear = 0;  
Q[rear][0] = xS;  
Q[rear][1] = yS;  
rear++;  
Vs[xS][yS] = 1;  
while (front != rear) {  
int x = Q[front][0];  
int y = Q[front][1];  
front++;  
if (x == xE && y == yE) {  
System.out.println(h);  
return;  
} else {  
for (int i = 0; i < 4; i++) {  
int b = y + Dy[i];  
if (i == 1 || i == 2) {  
int a = x + Dx[i];  
if (a >= 0 && a < N && b >= 0 && b < M && maze[a][b] != 0 && Vs[a][b] == 0) {  
Q[rear][0] = a;  
Q[rear][1] = b;  
Vs[a][b] = 1;  
rear++;  
}  
}  
if (i == 0 || i == 3) {  
int count = 0;  
while (h > count) {  
count++;  
int a = x + count \* Dx[i];  
if (a >= 0 && a < N && b >= 0 && b < M && maze[a][b] != 0 && Vs[a][b] == 0) {  
Q[rear][0] = a;  
Q[rear][1] = b;  
Vs[a][b] = 1;  
rear++;  
}  
}  
}  
}  
}  
}  
  
for (int i = 0; i < N; i++) {  
for (int j = 0; j < M; j++) {  
Vs[i][j] = 0;  
}  
}  
rear = 0;  
front = 0;  
BT(h + 1);  
}  
}

package luyende;  
  
import java.io.FileInputStream;  
import java.io.FileNotFoundException;  
import java.util.Scanner;  
  
public class MarioClimb {  
Scanner sc = new Scanner([System.in](http://system.in/));  
int t, n, m, st, en;  
int queueX[] = new int[1000000];  
int queueY[] = new int[1000000];  
int dx[] = { 1, 0, 0, -1 };  
int dy[] = { 0, 1, -1, 0 };  
int map[][], visit[][];  
int x2 = 0, y2 = 0, x3 = 0, y3 = 0;  
  
void init() {  
map = new int[n + 1][m + 1];  
visit = new int[n + 1][m + 1];  
}  
  
void resetVisit() {  
visit = new int[n + 1][m + 1];  
}  
  
void backTrack(int h) {  
st = en = 0;  
queueX[en] = x2;  
queueY[en++] = y2;  
visit[x2][y2] = 1;  
while (st != en) {  
int x = queueX[st];  
int y = queueY[st++];  
if (x == x3 && y == y3) {  
System.out.println(h);  
return;  
}  
for (int i = 0; i < 4; i++) {  
// di ngang  
if (i == 1 || i == 2) {  
int a = x + dx[i];  
int b = y + dy[i];  
if (a >= 0 && a < n && b >= 0 && b < m) {  
if (visit[a][b] == 0 && map[a][b] != 0) {  
queueX[en] = a;  
queueY[en++] = b;  
visit[a][b] = 1;  
}  
}  
}  
// di doc  
else if (i == 0 || i == 3) {  
int count = 0;  
while (h > count) {  
count++;  
// chú ý chỗ này: khi đi lên, xuống mà khi nhảy thì tăng độ cao h nghĩa là thay đổi tọa độ x  
// xem trong ví dụ thì nhảy từ x=3 lên x=1. trong ma trận này mình trục tọa độ mình để khác so với hệ tọa độ oxy trong toán học  
// !!!  
int a = x + dx[i] \* count;  
int b = y + dy[i];  
if (a >= 0 && a < n && b >= 0 && b < m) {  
if (visit[a][b] == 0 && map[a][b] != 0) {  
queueX[en] = a;  
queueY[en++] = b;  
visit[a][b] = 1;  
}  
}  
}  
  
}  
}  
}  
st = en = 0;  
resetVisit();  
backTrack(h + 1);  
}  
  
void solution() {  
t = sc.nextInt();  
for (int tc = 1; tc <= t; tc++) {  
n = sc.nextInt();  
m = sc.nextInt();  
init();  
for (int i = 0; i < n; i++) {  
for (int j = 0; j < m; j++) {  
map[i][j] = sc.nextInt();  
if (map[i][j] == 2) {  
x2 = i;  
y2 = j;  
} else if (map[i][j] == 3) {  
x3 = i;  
y3 = j;  
}  
}  
}  
System.out.println("Case #" + tc);  
backTrack(0);  
}  
}  
  
public static void main(String[] args) throws Exception {  
System.setIn(new FileInputStream("MarioClimb.txt"));  
MarioClimb m = new MarioClimb();  
m.solution();  
}  
  
}