C：

#include <stdio.h>

#define N 10

int num[N][N];

int tag[N][N] = {0};

int m, n;

int r = 100;

int find(int i, int j, int t, int ntag[][N])

{

int count = 0;

if (i < 0 || i >= n || j < 0 || j >= m || ntag[i][j] == 1)

return 0;

ntag[i][j] = 1;

if (tag[i][j] != t)

return 0;

count++;

count += find(i - 1, j, t, ntag);

count += find(i + 1, j, t, ntag);

count += find(i, j - 1, t, ntag);

count += find(i, j + 1, t, ntag);

return count;

}

int isbad()

{

int i, j, k = 0,ge2;

int t = tag[0][0];

int ntag1[N][N] = {0};

int ntag2[N][N] = {0};

int ge1 = find(0, 0, t, ntag1);

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

{

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

{

if (tag[i][j] != t)

{

k = 1;

break;

}

}

if (k == 1)

break;

}

if (i == n && j == m)

return 0;

ge2 = find(i, j, tag[i][j], ntag2);

return ge1 + ge2 != m \* n;

}

int bad(int i, int j)

{

int b;

if (i < 0 || i >= n || j < 0 || j >= m || tag[i][j] == 1)

return 1;

tag[i][j] = 1;

b = isbad();

tag[i][j] = 0;

return b;

}

void go(int i, int j, int k, int count)

{

if (bad(i, j) || count < num[i][j])

return;

k++;

if (count == num[i][j])

{

if (r > k)

r = k;

return;

}

tag[i][j] = 1;

count -= num[i][j];

go(i - 1, j, k, count);

go(i + 1, j, k, count);

go(i, j - 1, k, count);

go(i, j + 1, k, count);

tag[i][j] = 0;

}

int main()

{

int i, j;

int half = 0;

scanf("%d %d", &m, &n);

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

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

{

scanf("%d", &num[i][j]);

half += num[i][j];

}

if (half % 2 == 0 && half >= num[0][0] \* 2)

{

half /= 2;

go(0, 0, 0, half);

}

if (r == 100)

r = 0;

printf("%d", r);

return 0;

}

C++

#include<cstdio>

#include<cstring>

#include<cmath>

#include<cstdlib>

#include<algorithm>

#include<iostream>

#include<string>

#include<queue>

using namespace std;

int a[15][15];

int m,n;

int mx,sum;

int mi[][2]={{0,1},{0,-1},{1,0},{-1,0}};

struct ki{

int x,y;

int sum;

int tep;

};

void dfs(int x,int y,int s,int tep){

if(s>sum/(2.0))

return ;

else if(s==sum/(2.0)){

if(tep<mx)

mx=tep;

// printf("s=%d\n",s);

}

else{

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

int tx=x+mi[i][0];

int ty=y+mi[i][1];

if(tx<0||ty<0||tx>=n||ty>=m)

continue;

if(a[tx][ty]!=-1){

int tem=a[tx][ty];

a[tx][ty]=-1;

// printf("tx=%d ty=%d s=%d tep=%d\n",tx,ty,s,tep);

dfs(tx,ty,s+tem,tep+1);

a[tx][ty]=tem;

}

}

}

}

int main(){

while(cin>>m>>n){

sum=0;

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

for(int k=0;k<m;k++){

scanf("%d",&a[i][k]);

sum+=a[i][k];

}

mx=10000;

int tem=a[0][0];

a[0][0]=-1;

dfs(0,0,tem,1);

printf("%d\n",mx==10000?0:mx);

}

return 0;

}

Java：

import java.util.Scanner;

public class Main{

static int sum,half,m,n;

static int[][] map;

static boolean[][] flag;

static int count=100;

public static void main(String[] args) {

// TODO Auto-generated method stub

Scanner sc=new Scanner(System.in);

n=sc.nextInt();

m=sc.nextInt();

map=new int[m][n];

flag=new boolean[m][n];

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

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

map[i][j]=sc.nextInt();

sum+=map[i][j];

}

if(sum%2!=0)

System.out.println(0);

else{

half=sum/2;

dfs(0,0,1,0);

System.out.println(count==100?0:count);

}

}

private static void dfs(int i, int j, int step,int s) {

if(i<0 || i>=m || j<0 || j>=n ){

return;

}

if(flag[i][j]==true)

return;

if(s>half){

flag[i][j]=false;

return;

}

else{

s+=map[i][j];

flag[i][j]=true;

if(s==half){

if(step<count)

count=step;

}

else{

dfs(i+1,j,step+1,s);

dfs(i,j+1,step+1,s);

dfs(i-1,j,step+1,s);

dfs(i,j-1,step+1,s);

flag[i][j]=false;

}

}

}

}