**算法训练 传纸条**

时间限制：1.0s   内存限制：512.0MB

问题描述

　　小渊和小轩是好朋友也是同班同学，他们在一起总有谈不完的话题。一次素质拓展活动中，班上同学安排做成一个m行n列的矩阵，而小渊和小轩被安排在矩阵对角线的两端，因此，他们就无法直接交谈了。幸运的是，他们可以通过传纸条来进行交流。纸条要经由许多同学传到对方手里，小渊坐在矩阵的左上角，坐标(1,1)，小轩坐在矩阵的右下角，坐标(m,n)。从小渊传到小轩的纸条只可以向下或者向右传递，从小轩传给小渊的纸条只可以向上或者向左传递。  
　　在活动进行中，小渊希望给小轩传递一张纸条，同时希望小轩给他回复。班里每个同学都可以帮他们传递，但只会帮他们一次，也就是说如果此人在小渊递给小轩纸条的时候帮忙，那么在小轩递给小渊的时候就不会再帮忙。反之亦然。  
　　还有一件事情需要注意，全班每个同学愿意帮忙的好感度有高有低（注意：小渊和小轩的好心程度没有定义，输入时用0表示），可以用一个0-100的自然数来表示，数越大表示越好心。小渊和小轩希望尽可能找好心程度高的同学来帮忙传纸条，即找到来回两条传递路径，使得这两条路径上同学的好心程度只和最大。现在，请你帮助小渊和小轩找到这样的两条路径。

输入格式

　　输入第一行有2个用空格隔开的整数m和n，表示班里有m行n列（1<=m,n<=50）。  
　　接下来的m行是一个m\*n的矩阵，矩阵中第i行j列的整数表示坐在第i行j列的学生的好心程度。每行的n个整数之间用空格隔开。

输出格式

　　输出一行，包含一个整数，表示来回两条路上参与传递纸条的学生的好心程度之和的最大值。

样例输入

3 3  
0 3 9  
2 8 5  
5 7 0

样例输出

34

数据规模和约定

　　30%的数据满足：1<=*m*,*n*<=10  
　　100%的数据满足：1<=*m*,*n*<=50

锦囊1

动态规划。

锦囊2

把传两次纸条看成一次传两张纸条，则这两张纸条经过相同时间后达到同一条斜线，然后以斜线和斜线上的两个位置来建立状态进行动态规划。 方程的状态为：用F[i,j,k]表示到第i条斜线的第j和k个位置的最优值。

本题的C++参考代码如下：

#include<iostream>

#include<string.h>

using namespace std ;

int dis[101][101] ;

int f[105][52][52] ;

int rows , cols , step ;

#define MAX(a,b) ((a) > (b) ? (a) : (b))

int Max( int a , int b , int c , int d){

a = MAX ( a , b ) ;

a = MAX ( a , c ) ;

a = MAX ( a , d ) ;

return a ;

}

int main(){

std::ios::sync\_with\_stdio(false) ;

int i , j , k ;

memset( f , 0 , sizeof(f) ) ;

memset( dis , 0 , sizeof(dis) ) ;

cin >> rows >> cols ;

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

for( j = 1 ; j <= cols ; j++ )

cin >> dis[i][j] ;

step = rows + cols - 2 ;

f[1][2][1] = f[1][1][2]= dis[1][2] + dis[2][1] ;

for( k = 2 ; k <= (step - 1) ; k++ )

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

for( j = 1 ; j <= rows ; j++ )

if( i != j )

f[k][i][j] = dis[i][k - i + 2] + dis[j][k - j + 2] + Max( f[k - 1][i - 1][j - 1] , f[k - 1][i - 1][j] , f[k - 1][i][j - 1] , f[k - 1][i][j]) ;

cout << f[step - 1][rows - 1][rows] << endl ;

return 0 ;

}

本题的C参考代码如下：

#include<stdio.h>

int n,m;

int i,j,k;

int Map[51][51];

int F[111][51][51];

int Max(int a,int b,int c,int d)

{

if(a>=b&&a>=c&&a>=d)

return a;

if(b>=a&&b>=c&&b>=d)

return b;

if(c>=a&&c>=b&&c>=d)

return c;

if(d>=a&&d>=b&&d>=c)

return d;

}

int main()

{

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

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

for(j=1;j<=m;j++)

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

for(k=1;k<=n+m-2;k++)

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

for(j=1;j<=n;j++)

if(i==n&&j==n&&k==n+m-2)

F[k][i][j]=Max(F[k-1][i-1][j],F[k-1][i][j-1],F[k-1][i][j],F[k-1][i-1][j-1])+Map[i][k+2-i]+Map[j][k+2-j];

else if(i!=j&&k+2-i>=1&&k+2-j>=1)

F[k][i][j]=Max(F[k-1][i-1][j],F[k-1][i][j-1],F[k-1][i][j],F[k-1][i-1][j-1])+Map[i][k+2-i]+Map[j][k+2-j];

printf("%d",F[n+m-2][n][n]);

return 0;

}

本题的Java参考代码如下：

import java.io.BufferedReader;

import java.io.IOException;

import java.io.InputStreamReader;

import java.util.Arrays;

public class Main {

static int[][][] dp = new int[105][55][55];

static int[][] map = new int[100][100];

public static void main(String[] args) throws IOException {

BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

String order[] = br.readLine().split(" ");

int n = Integer.parseInt(order[0]), m = Integer.parseInt(order[1]);

for (int i = 1; i <= n; i++) {

String data[] = br.readLine().split(" ");

for (int j = 1; j <= m; j++) {

map[i][j] = Integer.parseInt(data[j - 1]);

}

}

int c = n + m - 2;

for (int k = 1; k < c; k++) {

int t = k + 2 > n ? n : k + 2;

for (int i = 1; i <= t; i++) {

for (int x = i + 1; x <= t; x++) {

dp[k][i][x] = Math.max(Math.max(dp[k - 1][i - 1][x], dp[k - 1][i][x - 1]),

Math.max(dp[k - 1][i - 1][x - 1], dp[k - 1][i][x]))

+ map[i][k + 2 - i] + map[x][k + 2 - x];

}

}

}

dp[c][n][n] = Math.max(dp[c - 1][n - 1][n], dp[c - 1][n][n - 1]);

System.out.println(dp[c][n][n]);

}

}