**地宫取宝**

问题描述

　　X 国王有一个地宫宝库。是 n x m 个格子的矩阵。每个格子放一件宝贝。每个宝贝贴着价值标签。  
  
　　地宫的入口在左上角，出口在右下角。  
  
　　小明被带到地宫的入口，国王要求他只能向右或向下行走。  
  
　　走过某个格子时，如果那个格子中的宝贝价值比小明手中任意宝贝价值都大，小明就可以拿起它（当然，也可以不拿）。  
  
　　当小明走到出口时，如果他手中的宝贝恰好是k件，则这些宝贝就可以送给小明。  
  
　　请你帮小明算一算，在给定的局面下，他有多少种不同的行动方案能获得这k件宝贝。

输入格式

　　输入一行3个整数，用空格分开：n m k (1<=n,m<=50, 1<=k<=12)  
  
　　接下来有 n 行数据，每行有 m 个整数 Ci (0<=Ci<=12)代表这个格子上的宝物的价值

输出格式

　　要求输出一个整数，表示正好取k个宝贝的行动方案数。该数字可能很大，输出它对 1000000007 取模的结果。

样例输入

2 2 2  
1 2  
2 1

样例输出

2

样例输入

2 3 2  
1 2 3  
2 1 5

样例输出

14

本题的C++参考代码如下：

#pragma comment(linker,"/STACK:102400000,102400000")

#include <algorithm>

#include <iostream>

//#include <fstream>

#include <climits>

#include <cstdlib>

#include <cstring>

#include <string>

#include <vector>

#include <cstdio>

#include <queue>

#include <stack>

#include <cmath>

#include <list>

#include <set>

#include <map>

#define sf scanf

#define pf printf

#define fst first

#define scd second

#define pb push\_back

#define mkp make\_pair

#define cls(a,x) memset(a,x,sizeof a)

#define dt(x) cout<<#x<<"="<<x<<" ";

#define dte(x) cout<<#x<<"="<<x<<endl;

#if(defined(\_WIN32)||defined(\_\_WIN32\_\_))

typedef \_\_int64 LL;

typedef unsigned \_\_int64 ULL;

#define ll I64

#else

typedef long long LL;

typedef unsigned long long ULL;

#define I64 ll

#endif

using namespace std;

template<class T>inline void scaf(T &v)

{

char ch;

while(ch=getchar())

if(ch<='9' && ch>='0') break;

v=ch-'0';

while(ch=getchar())

if(ch<='9' && ch>='0') v=(v<<1)+(v<<3)+ch-'0';

else break;

}

typedef pair<int,int > PII;

const int MX=50100;

const int mod=1000000007;

int sp[51][51];

int g[51][51];

LL dp[51][51][13];

inline void run()

{

int n,m,k,i,j,ii,jj,kk;

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

for(i=1;i<=50;++i)

sp[i][1]=sp[1][i]=1;

for(i=2;i<=n;++i)

for(j=2;j<=m;++j)

sp[i][j]=sp[i-1][j]+sp[i][j-1];

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

{

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

{

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

dp[i][j][1]=sp[i][j];

}

}

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

{

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

{

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

{

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

{

if(g[ii][jj]<g[i][j])

{

for(kk=2;kk<=k;++kk)

{

dp[i][j][kk]+=dp[ii][jj][kk-1]\*sp[i-ii+1][j-jj+1];

dp[i][j][kk]%=mod;

}

}

}

}

}

}

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

// {

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

// {

// pf("%d ",dp[i][j][k]);

// }

// putchar(10);

// }

LL Ans=0;

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

{

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

{

Ans+=dp[i][j][k]\*sp[n-i+1][m-j+1];

Ans%=mod;

}

}

pf("%I64d\n",Ans);

}

int main()

{

run();

return 0;

}

本题的C参考代码如下：

#include<stdio.h>

#include<string.h>

#define N 55

#define MOD 1000000007

int map[55][55];

int dp[55][55][15][15];

int main(void)

{

int n, m, k;

int i, j, c, val, aMax;

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

aMax = 0;

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

{

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

{

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

// map[i][j]++;

if(aMax < map[i][j])

{

aMax = map[i][j];

}

}

}

memset(dp, 0, sizeof(dp));

dp[1][1][0][0] = 1;

dp[1][1][1][map[1][1]] = 1;

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

{

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

{

dp[i][j][0][0] += dp[i][j - 1][0][0] + dp[i - 1][j][0][0];

dp[i][j][0][0] %= MOD;

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

{

for(val = 0; val <= aMax; val++)

{

dp[i][j][c][val] += dp[i][j - 1][c][val] + dp[i - 1][j][c][val];

dp[i][j][c][val] %= MOD;

}

if(c == 1)

{

dp[i][j][1][map[i][j]] += dp[i][j - 1][0][0];

dp[i][j][1][map[i][j]] %= MOD;

dp[i][j][1][map[i][j]] += dp[i - 1][j][0][0];

dp[i][j][1][map[i][j]] %= MOD;

}

else

{

for(val = 0; val < map[i][j]; val++)

{

dp[i][j][c][map[i][j]] += dp[i][j - 1][c - 1][val];

dp[i][j][c][map[i][j]] %= MOD;

dp[i][j][c][map[i][j]] += dp[i - 1][j][c - 1][val];

dp[i][j][c][map[i][j]] %= MOD;

}

}

}

}

}

int sum = 0;

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

{

sum += dp[n][m][k][i];

sum %= MOD;

}

printf("%d", sum);

return 0;

}

本题的Java参考代码如下：

import java.io.InputStreamReader;

import java.io.OutputStreamWriter;

import java.io.PrintWriter;

import java.io.StreamTokenizer;

import java.util.Arrays;

public class Main

{

private static StreamTokenizer tokenizer = new StreamTokenizer(

new InputStreamReader(System.in));

private static PrintWriter outWriter = new PrintWriter(

new OutputStreamWriter(System.out));

private static int n, m, k;

private static int[][] table;

private static final int MOD = 1000000007;

private static long[][][][] state;

private static long dfs(int i, int j, int num, int max)

{

if (state[i][j][num][max] != -1)

return state[i][j][num][max];

long currentAns = 0;

if (i == n - 1 && j == m - 1)

{

if (num == k || max < table[i][j] && num + 1 == k)

currentAns++;

state[i][j][num][max] = currentAns;

return currentAns;

}

if (i + 1 < n)

{

currentAns += dfs(i + 1, j, num, max);

if (max < table[i][j] && num + 1 <= k)

currentAns += dfs(i + 1, j, num + 1, table[i][j]);

}

if (j + 1 < m)

{

currentAns += dfs(i, j + 1, num, max);

if (max < table[i][j] && num + 1 <= k)

currentAns += dfs(i, j + 1, num + 1, table[i][j]);

}

state[i][j][num][max] = currentAns;

return currentAns;

}

public static void main(String[] args) throws Exception

{

tokenizer.nextToken();

n = (int) tokenizer.nval;

tokenizer.nextToken();

m = (int) tokenizer.nval;

tokenizer.nextToken();

k = (int) tokenizer.nval;

table = new int[n][m];

state = new long[n][m][k + 1][14];

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

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

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

Arrays.fill(state[i][j][t], -1);

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

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

{

tokenizer.nextToken();

table[i][j] = (int) tokenizer.nval;

table[i][j]++;

}

long ret = dfs(0, 0, 0, 0);

outWriter.println(ret % MOD);

outWriter.flush();

}

}