**算法训练 采油区域**

时间限制：2.0s   内存限制：512.0MB

　　采油区域　　Siruseri政府决定将石油资源丰富的Navalur省的土地拍卖给私人承包商以建立油井。被拍卖的整块土地为一个矩形区域，被划分为*M*×*N*个小块。  
　　Siruseri地质调查局有关于Navalur土地石油储量的估测数据。这些数据表示为*M*×*N*个非负整数，即对每一小块土地石油储量的估计值。  
　　为了避免出现垄断，政府规定每一个承包商只能承包一个由*K*×*K*块相连的土地构成的正方形区域。  
　　AoE石油联合公司由三个承包商组成，他们想选择三块互不相交的*K*×*K*的区域使得总的收益最大。  
　　例如，假设石油储量的估计值如下：

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 8 | 8 | 8 | 8 | 8 | 1 | 1 | 1 |
| 1 | 8 | 8 | 8 | 8 | 8 | 1 | 1 | 1 |
| 1 | 8 | 8 | 8 | 8 | 8 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 8 | 8 | 8 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 8 | 8 | 8 |
| 1 | 1 | 1 | 1 | 1 | 1 | 9 | 9 | 9 |
| 1 | 1 | 1 | 1 | 1 | 1 | 9 | 9 | 9 |

　　如果*K* = 2, AoE公司可以承包的区域的石油储量总和为100, 如果*K* = 3, AoE公司可以承包的区域的石油储量总和为208。  
　　AoE公司雇佣你来写一个程序，帮助计算出他们可以承包的区域的石油储量之和的最大值。

输入格式

　　输入第一行包含三个整数*M*, *N*, *K*，其中*M*和*N*是矩形区域的行数和列数，*K*是每一个承包商承包的正方形的大小（边长的块数）。接下来*M*行，每行有*N*个非负整数表示这一行每一小块土地的石油储量的估计值。

输出格式

　　输出只包含一个整数，表示AoE公司可以承包的区域的石油储量之和的最大值。

数据规模和约定

　　数据保证*K*≤*M*且*K*≤*N*并且至少有三个*K*×*K*的互不相交的正方形区域。其中30%的输入数据，*M*, *N*≤ 12。所有的输入数据, *M*, *N*≤ 1500。每一小块土地的石油储量的估计值是非负整数且≤ 500。

样例输入

9 9 3  
1 1 1 1 1 1 1 1 1  
1 1 1 1 1 1 1 1 1  
1 8 8 8 8 8 1 1 1  
1 8 8 8 8 8 1 1 1  
1 8 8 8 8 8 1 1 1  
1 1 1 1 8 8 8 1 1  
1 1 1 1 1 1 8 8 8  
1 1 1 1 1 1 9 9 9  
1 1 1 1 1 1 9 9 9

样例输出

208

本题的C++参考代码如下：

#include <cstdio>

#include <cstdlib>

#include <algorithm>

#include <string>

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

#define max3(a, b, c) (max((a), max(b, c)))

const char fi[] = "oil.in";

const char fo[] = "oil.out";

const int maxN = 1510;

const int MAX = 0x3f3f3f3f;

const int MIN = ~MAX;

int sum[maxN][maxN];

int zoxx[maxN][maxN]; // Left and below.

int zouh[maxN][maxN]; // Left and above.

int yzxx[maxN][maxN]; // Right and below.

int yzuh[maxN][maxN]; // Right and above.

int hh[maxN]; // Horizon.

int lp[maxN]; // Vertical.

int n,m,K;

void init\_file()

{

return;

}

inline int getint()

{

int res = 0; char tmp;

while(!isdigit(tmp = getchar()));

do res = (res << 3) + (res << 1) + tmp - '0';

while(isdigit(tmp = getchar()));

return res;

}

void readdata()

{

n = getint(); m = getint(); K = getint();

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

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

(sum[i][j] = getint()) += sum[i][j - 1];

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

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

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

//预处理出二维的前缀和。

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

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

{

zouh[i][j] = sum[i][j]

+ sum[i - K][j - K]

- sum[i][j - K]

- sum[i - K][j];

hh[i] = max(hh[i],zouh[i][j]); //Horizon.

lp[j] = max(lp[j],zouh[i][j]); //Vertical.

zouh[i][j] = max3(zouh[i][j],

zouh[i - 1][j],

zouh[i][j - 1]);

} // Left and above.

for(int i = n - K + 1; i; --i)

for(int j = m - K + 1; j; --j)

yzxx[i][j] = max3(sum[i - 1][j - 1]

+ sum[i + K - 1][j + K - 1]

- sum[i - 1][j + K - 1]

- sum[i + K - 1][j - 1],

yzxx[i + 1][j],

yzxx[i][j + 1]);

//Right and below.

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

for(int j = m - K + 1; j; --j)

yzuh[i][j] = max3(sum[i][j + K - 1]

+ sum[i - K][j - 1]

- sum[i][j - 1]

- sum[i - K][j + K - 1],

yzuh[i - 1][j],

yzuh[i][j + 1]);

//Right and above.

for(int i = n - K + 1; i; --i)

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

zoxx[i][j] = max3(sum[i - 1][j - K]

+ sum[i + K - 1][j]

- sum[i - 1][j]

- sum[i + K - 1][j - K],

zoxx[i + 1][j],

zoxx[i][j - 1]);

//Left and below

return;

}

void work()

{

int ans = 0;

for(int i = K; i < n - (K << 1); ++i)

ans = max(ans,yzuh[i][1]

+ hh[i + K]

+ yzxx[i + K + 1][1]);

// Case 1.

for(int j = K; j < m - (K << 1); ++j)

ans = max(ans,zoxx[1][j]

+ lp[j + K]

+ yzxx[1][j + K + 1]);

// Case 2.

for(int i = K; i < n - K + 1; ++i)

for(int j = K; j < m - K + 1; ++j)

{

ans = max3(ans,zouh[i][j]

+ yzuh[i][j + 1]

+ yzxx[i + 1][1], //Case 3.

yzuh[i][1]

+ zoxx[i + 1][j]

+ yzxx[i + 1][j + 1]); //Case 4.

ans = max3(ans,zouh[i][j]

+ zoxx[i + 1][j]

+ yzxx[1][j + 1], //Case 5.

zoxx[1][j]

+ yzuh[i][j + 1]

+ yzxx[i + 1][j + 1]); //Case 6.

}

printf("%d\n",ans);

return;

}

int main()

{

init\_file();

readdata();

work();

return 0;

}

#undef max

本题的C参考代码如下：

#include <stdio.h>

#include<stdlib.h>

long n,m,K;

long map[1510][1510];

long sum1[1510][1510];//left and above

long sum2[1510][1510];//right and above

long sum3[1510][1510];//left and below

long sum4[1510][1510];//right and below

long rect1[1510][1510];//left and above

long rect2[1510][1510];//right and above

long rect3[1510][1510];//left and below

long rect4[1510][1510];//right and below

long rectA[1510][1510];

long rectB[1510][1510];

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

int main()

{

long i,j;

long ans = 0;

scanf("%ld%ld%ld",&n,&m,&K);

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

{

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

{

scanf("%ld",map[i]+j);

sum1[i][j] = sum1[i][j-1]+map[i][j];

sum3[i][j] = sum1[i][j];

}

}

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

{

for (j=m;j>0;j--)

{

sum2[i][j] = sum2[i][j+1]+map[i][j];

sum4[i][j] = sum2[i][j];

}

}

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

{

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

{

sum1[i][j] += sum1[i-1][j];

sum2[i][j] += sum2[i-1][j];

}

}

for (i=n;i>0;i--)

{

for (j=m;j>0;j--)

{

sum3[i][j] += sum3[i+1][j];

sum4[i][j] += sum4[i+1][j];

}

}

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

{

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

{

if (i-K>=0 && j-K>=0)

{

rect1[i][j] = sum1[i][j]-sum1[i][j-K]-sum1[i-K][j]+sum1[i-K][j-K];//left and above

rectA[i][j] = rect1[i][j];

}

if (i-K>=0 && j+K-1<=m)

rect2[i][j] = sum2[i][j]-sum2[i][j+K]-sum2[i-K][j]+sum2[i-K][j+K];//right and above

if (i+K-1<=n && j-K>=0)

{

rect3[i][j] = sum3[i][j]-sum3[i][j-K]-sum3[i+K][j]+sum3[i+K][j-K];//left and blow

rectB[i][j] = rect3[i][j];

}

if (i+K-1<=n && j+K-1<=m)

rect4[i][j] = sum4[i][j]-sum4[i][j+K]-sum4[i+K][j]+sum4[i+K][j+K];//right and blow

}

}

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

{

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

{

rect1[i][j] = MAX(rect1[i][j],rect1[i-1][j]);

rect1[i][j] = MAX(rect1[i][j],rect1[i][j-1]);

rectA[i][j] = MAX(rectA[i][j],rectA[i][j-1]);

rectB[i][j] = MAX(rectB[i][j],rectB[i-1][j]);

}

}

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

{

for (j=m;j>0;j--)

{

rect2[i][j] = MAX(rect2[i][j],rect2[i-1][j]);

rect2[i][j] = MAX(rect2[i][j],rect2[i][j+1]);

}

}

for (i=n;i>0;i--)

{

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

{

rect3[i][j] = MAX(rect3[i][j],rect3[i+1][j]);

rect3[i][j] = MAX(rect3[i][j],rect3[i][j-1]);

}

}

for (i=n;i>0;i--)

{

for (j=m;j>0;j--)

{

rect4[i][j] = MAX(rect4[i][j],rect4[i+1][j]);

rect4[i][j] = MAX(rect4[i][j],rect4[i][j+1]);

}

}

for (i=1+K;i<n-K+2;i++)

{

for (j=1+K;j<m-K+2;j++)

{

ans = MAX(ans,rect1[i-1][j-1]+rect2[i-1][j]+rect4[i][1]);

ans = MAX(ans,rect3[i][j-1]+rect4[i][j]+rect2[i-1][1]);

ans = MAX(ans,rect1[i-1][j-1]+rect3[i][j-1]+rect4[1][j]);

ans = MAX(ans,rect2[i-1][j]+rect4[i][j]+rect3[1][j-1]);

}

}

for (i=1+K;i<n-2\*K+1;i++)

{

ans = MAX(ans,rect2[i-1][1]+rect4[i+K][1]+rectA[i+K-1][m]);

}

for (j=1+K;j<m-2\*K+1;j++)

{

ans = MAX(ans,rect3[1][j-1]+rect4[1][j+K]+rectB[n][j+K-1]);

}

printf("%ld",ans);

system("pause");

return 0;

}

本题的Java参考代码如下：

import java.io.BufferedReader;

import java.io.IOException;

import java.io.InputStreamReader;

public class Main{

static int[][] cl;

static int[][] cbcl,clqs,clqx,clhs,clhx;

static int m=0,n=0,k=0,fin=0;

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

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

String s=reader.readLine();

String[] sa = s.split(" ");

m=Integer.parseInt(sa[0]);

n=Integer.parseInt(sa[1]);

k=Integer.parseInt(sa[2]);

cl=new int[m+1][n+1];

cbcl=new int[m-k+2][n-k+2];

clqs=new int[m-k+2][n-k+2];

clqx=new int[m-k+2][n-k+2];

clhs=new int[m-k+2][n-k+2];

clhx=new int[m-k+2][n-k+2];

for(int i=1;i<=m;i++) cl[i][0]=0;

for(int i=1;i<=n;i++) cl[0][i]=0;

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

s=reader.readLine();

sa = s.split(" ");

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

cl[i][j]=cl[i][j-1]+Integer.parseInt(sa[j-1]);

}

}

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

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

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

}

}

for(int i=m-k+1;i>=1;i--){

for (int j=n-k+1;j>=1;j--){

cbcl[i][j]=cl[i+k-1][j+k-1]-cl[i-1][j+k-1]-cl[i+k-1][j-1]+cl[i-1][j-1];

}

}

for (int i=1,l=m-k+1;i<=l;i++){

for (int j=1,o=n-k+1;j<=o;j++){

int a=cbcl[i][j];

if (cbcl[i][0]<a) {

cbcl[i][0]=a;

}

}

}

for (int j=1,o=n-k+1;j<=o;j++){

for (int i=1,l=m-k+1;i<=l;i++){

int a=cbcl[i][j];

if (cbcl[0][j]<a) {

cbcl[0][j]=a;

}

}

}

for (int i=1;i<=m-k+1;i++){

for(int j=1;j<=n-k+1;j++){

if (j==1) clqs[i][j]=cbcl[i][j]; else clqs[i][j]=Math.max(cbcl[i][j],clqs[i][j-1]);

clqx[i][j]=clqs[i][j];

}

}

for (int i=1;i<=m-k+1;i++){

for(int j=n-k+1;j>=1;j--){

if (j==n-k+1) {

clhs[i][j]=cbcl[i][j];

clhx[i][j]=cbcl[i][j];

}else {

clhx[i][j]=Math.max(cbcl[i][j],clhx[i][j+1]);

clhs[i][j]=clhx[i][j];

}

}

}

for(int j=1;j<=n-k+1;j++){

for (int i=m-k;i>=1;i--){

clqx[i][j]=Math.max(clqx[i][j],clqx[i+1][j]);

clhx[i][j]=Math.max(clhx[i][j],clhx[i+1][j]);

}

}

for(int j=1;j<=n-k+1;j++){

for (int i=2;i<=m-k+1;i++){

clqs[i][j]=Math.max(clqs[i][j],clqs[i-1][j]);

clhs[i][j]=Math.max(clhs[i][j],clhs[i-1][j]);

}

}

for (int i=1;i<=m-3\*k+1;i++){

for(int j=i+k;j<=m-2\*k+1;j++){

fin=Math.max(fin, clhs[i][1]+cbcl[j][0]+clhx[j+k][1]);

}

}

for (int i=1;i<=n-3\*k+1;i++){

for(int j=i+k;j<=n-2\*k+1;j++){

fin=Math.max(fin, clqs[1][i]+cbcl[0][j]+clhx[1][j+k]);

}

}

for (int i=1,l=m-2\*k+1;i<=l;i++){

for (int j=1,o=n-2\*k+1;j<=o;j++){

fin=Math.max(fin,clqs[i][n-k+1]+clqx[i+k][j]+clhx[i+k][j+k]);

fin=Math.max(fin,clqx[i+k][n-k+1]+clqs[i][j]+clhs[i][j+k]);

fin=Math.max(fin,clqx[1][j]+clhs[i][j+k]+clhx[i+k][j+k]);

fin=Math.max(fin,clhx[1][j+k]+clqs[i][j]+clqx[i+k][j]);

}

}

System.out.println(fin);

}

}