**最大子阵**

问题描述

　　给定一个n\*m的矩阵A，求A中的一个非空子矩阵，使这个子矩阵中的元素和最大。  
  
　　其中，A的子矩阵指在A中行和列均连续的一块。

输入格式

　　输入的第一行包含两个整数n, m，分别表示矩阵A的行数和列数。  
　　接下来n行，每行m个整数，表示矩阵A。

输出格式

　　输出一行，包含一个整数，表示A中最大的子矩阵中的元素和。

样例输入

3 3  
-1 -4 3  
3 4 -1  
-5 -2 8

样例输出

10

样例说明

　　取最后一列，和为10。

数据规模和约定

　　对于50%的数据，1<=n, m<=50；  
　　对于100%的数据，1<=n, m<=500，A中每个元素的绝对值不超过5000。

本题的C++参考代码如下：

#include "stdio.h"

#include "string.h"

int a[500][500],b[500];

int f(int n,int m)

{int i,j,k,t,max=-999999;

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

{memset(b,0,m\*sizeof(int));

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

{t=-999999;

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

{b[k]+=a[j][k];

t+=b[k];

if(t<b[k])t=b[k];

if(max<t)max=t;

}

}

}

return max;

}

int main()

{

int i,j,n,m;

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

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

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

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

printf("%d",f(n,m));

return 0;

}

本题的C参考代码如下：

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

int a[500][500],b[500];

int maxsub(int n,int m){

int i,j,k,t,max=-999999;

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

{

memset(b,0,m\*sizeof(int));

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

{

t=-999999;

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

{

b[k]+=a[j][k];

t+=b[k];

if(t<b[k])t=b[k];

if(max<t)max=t;

}

}

}

return max;

}

int main()

{

int i,j,n,m;

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

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

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

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

printf("%d",maxsub(n,m));

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 {

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

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

String[] s = sc.readLine().split(" ");

int n = Integer.parseInt(s[0]);

int m = Integer.parseInt(s[1]);

int[][] f = new int[n][m];

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

s = sc.readLine().split(" ");

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

f[i][j] = Integer.parseInt(s[j]);

}

}

System.out.print(cc(f));

}

static int csum(int[] s) {

int b = 0;

int max = Integer.MIN\_VALUE;

for (int i = 0; i < s.length; i++) {

if (b > 0)

b += s[i];

else

b = s[i];

if (b > max)

max = b;

}

return max;

}

static int cc(int[][] s) {

int[] c = new int[s[0].length];

int max = Integer.MIN\_VALUE;

int n = s.length, m = s[0].length;

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

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

c[k] = 0;

}

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

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

c[k] += s[j][k];

}

int d = csum(c);

if (d > max)

max = d;

}

}

return max;

}

}