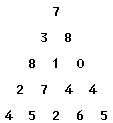
**算法训练 数字三角形**

时间限制：1.0s   内存限制：256.0MB

问题描述

　　（图３.１－１）示出了一个数字三角形。 请编一个程序计算从顶至底的某处的一条路  
　　径，使该路径所经过的数字的总和最大。  
　　●每一步可沿左斜线向下或右斜线向下走；  
　　●1＜三角形行数≤100；  
　　●三角形中的数字为整数0，1，…99；  
  
  
　　.  
　　（图３.１－１）

输入格式

　　文件中首先读到的是三角形的行数。  
  
　　接下来描述整个三角形

输出格式

　　最大总和（整数）

样例输入

5  
7  
3 8  
8 1 0  
2 7 4 4  
4 5 2 6 5

样例输出

30

本题的C++参考代码如下：

#include<iostream>

#include<algorithm>

#include<vector>

using namespace std;

int main(){

int n;

cin >> n;

vector<vector<int> >a(n + 1, vector<int>(n + 1));

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

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

cin >> a[i][j];

}

}

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

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

a[i][j] = max(a[i - 1][j - 1], a[i - 1][j]) + a[i][j];

}

}

int max = a[n][1];

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

if (a[n][i] > max) max = a[n][i];

}

cout << max << endl;

system("pause");

return 0;

}

本题的C参考代码如下：

#include<stdio.h>

int n,a[100][100];

int sum();

int main(){

int i,num,j;

scanf("%d",&n);

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

{

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

{

scanf("%d",&num);

a[i][j]=num;

}

}

printf("%d",sum());

return 0;

}

/\*

int sum(int i,int j)

{

if(i==n)

{

return a[i][j];

}

if(sum(i+1,j)>sum(i+1,j+1))

{

return sum(i+1,j)+a[i][j];

}else

{

return sum(i+1,j+1)+a[i][j];

}

}

\*/

int sum()

{

int i,j;

for(i=n-1;i>0;i--)

{

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

{

if(a[i][j]>a[i][j+1])

{

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

}

else

{

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

}

}

}

return a[0][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 NumberFormatException, IOException {

BufferedReader br=new BufferedReader(

new InputStreamReader(System.in));

int n=Integer.parseInt(br.readLine().trim()),i,j;

int[][] arr=new int[n][n];

arr[0][0]=Integer.parseInt(br.readLine().trim());

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

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

for(j=0;j<=i;j++){

arr[i][j]=Integer.parseInt(str[j]);

}

}

for(i=n-2;i>=0;i--){

for(j=0;j<=i;j++){

if(arr[i+1][j]>arr[i+1][j+1]) arr[i][j]+=arr[i+1][j];

else arr[i][j]+=arr[i+1][j+1];

}

}

System.out.println(arr[0][0]);

}

}