**西 安 邮 电 大 学**

**（计算机学院）**

算法实验报告

**专业名称： 软件工程**

**班 级： 1401班**

**学生姓名： 代栋**

**学号（8位）： 04143036**

**指导教师 陈琳**

一：矩阵连乘（动态规划）

思想:每次计算结果都会依赖前一个的计算结果（前提矩阵个数大于3）。先计算相邻两个矩阵相乘的次数，放在对应二维数组m中，然后计算相邻三个矩阵相乘的次数（依赖两个矩阵相乘的次数），然后在不同的断开位置计算出不同的最小次数，然后取最小值，修改m数组和断开位置的数组s，依次重复上面的操作，直到设计到全部矩阵相乘为止！m数组存放矩阵相乘的最小次数，s矩阵存放记录加括号的位置。

#include <stdio.h>

#define N 20

void MatrixChain(int p[N],int n,int m[N][N],int s[N][N]){

int i,j,t,k;

int r; //记录相乘的矩阵个数变量

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

m[i][i]=0; //当一个矩阵相乘时，相乘次数为 0

}

//矩阵个数从两个开始一次递增

for(r=2;r<=n;r++){

//从某个矩阵开始

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

//到某个矩阵的结束

j=i+r-1;

//拿到从 i 到 j 矩阵连乘的次数

m[i][j]=m[i+1][j]+p[i-1]\*p[i]\*p[j];

//拿到矩阵连乘断开的位置

s[i][j]=i;

//寻找加括号不同，矩阵连乘次数的最小值，修改 m 数组，和断开的位置 s 数组

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

t=m[i][k]+m[k+1][j]+p[i-1]\*p[k]\*p[j];

if(t<m[i][j]){

m[i][j]=t;

s[i][j]=k;

}

}

}

}

}

int main(void){

int n,n1,m1,i,j=2;

int p[N]={0}; //存储矩阵的行和列数组

int m[N][N]={0}; //存储矩阵与矩阵相乘的最小次数

int s[N][N]={0}; //存储矩阵与矩阵相乘断开的位置

printf("请输入矩阵个数:\n");

scanf("%d",&n);

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

printf("请输入第%d个矩阵的行和列(n1\*m1 格式):",i);

scanf("%d\*%d",&n1,&m1);

if(i==1){

p[0]=n1;

p[1]=m1;

}

else{

p[j++]=m1;

}

}

printf("\n记录矩阵行和列:\n");

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

printf("%d ",p[i]);

}

printf("\n");

MatrixChain(p,n,m,s);

printf("\n矩阵相乘的最小次数矩阵为:\n");

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

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

printf("%d ",m[i][j]);

}

printf("\n");

}

printf("\n矩阵相乘断开的位置矩阵为:\n");

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

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

printf("%d ",s[i][j]);

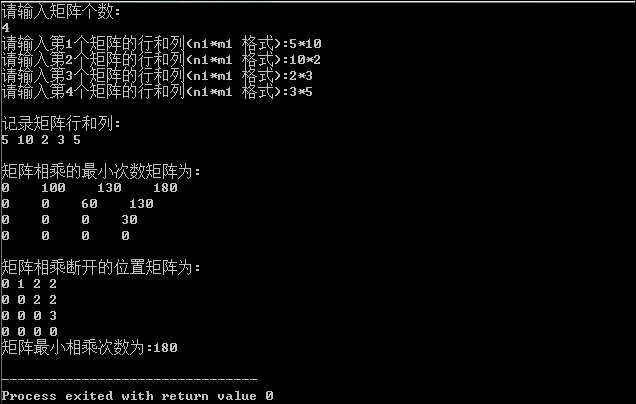
}

printf("\n");

}

return 0;

}



二：找出由n个数组成的序列的最长单调递增子序列

思想:将数组Len的第一个数放在L数组里面，然后依次遍历Len数组，采用 二分查找法依次遍历L数组，采用left和right下标控制，当L[mid]的值比Key 小，left=mid+1,大的话，right=mid-1,直到循环结束，修改Len数组里的值， 计算K值。

//测试数据 {2,6,-5,4,-2,8,1,3} ,结果为 4

#include <stdio.h>

#include <string.h>

#define N 100

//函数的声明

int FindKey(int p[],int Key,int n);

int GetLength(int p[],int n){

//记录最长公共子序列的长度（存的是原数组中的值）

int L[N];

int k=0;

int i,index;

//初始化数组 L

memset(L,0,sizeof(L));

L[0]=p[0];

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

index=FindKey(L,p[i],k);

L[index]=p[i];

if(index+1>k){

k++;

}

}

return k;

}

//二分查找关键值

int FindKey(int p[],int Key,int n){

int left=0;

int right=n-1;

int mid;

while(left<=right){

mid=(left+right)/2;

if(p[mid]<=Key){

left=mid+1;

}

else{

right=mid-1;

}

}

return left;

}

int main(void){

int Len[N];

int i,n,Length;

printf("请输入数组长度:\n");

scanf("%d",&n);

printf("请输入数组元素:\n");

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

scanf("%d",Len+i);

}

Length=GetLength(Len,n);

printf("最长公共子序列的长度为:%d\n",Length);

return 0;

}



三：数字三角形问题

思路：先将第一竖列和右斜线的元素求和相加依次赋值，然后再将其余元素按规则求和相加赋值，最后遍历二维数组，找出求和后的最大值sum

#include <stdio.h>

#define N 101

int max(int x,int y){

if(x>y){

y=x;

}

return y;

}

int main(void){

int a[N][N]={0};

int i,j,n,sum;

scanf("%d",&n);

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

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

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

}

}

//先给第一竖列和右斜线的元素赋值...

for(i=0;i<n-1;i++){

a[i+1][0]=a[i+1][0]+a[i][0];

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

}

//给其他元素赋值

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

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

//寻找最大值

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

}

}

//找出各元素相加的最大和

sum=a[0][0];

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

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

if(sum<a[i][j]){

sum=a[i][j];

}

}

}

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

return 0;

}

