实验3、《动态规划算法实验》

**一、实验目的**

1. 掌握动态规划方法贪心算法思想

2. 掌握最优子结构原理

3. 了解动态规划一般问题

**二、实验内容**

1. 编写一个简单的程序，解决0-1背包问题。设N=5，C=10，w={2，2，6，5，4},v={6,3,5,4,6}

2. 合唱队形安排问题

【问题描述】N位同学站成一排，音乐老师要请其中的(N-K)位同学出列，使得剩下的K位同学排成合唱队形。合唱队形是指这样的一种队形：设K位同学从左到右依次编号为1，2…，K，他们的身高分别为T1，T2，…，TK， 则他们的身高满足T1<...<Ti>Ti+1>…>TK(1<=i<=K)。已知所有N位同学的身高，计算最少需要几位同学出列，可以使得剩下的同学排成合唱队形。

**三、算法思想分析**

1.0-1背包采用动规算法主要是动规方程的思考，之后就是确定边界条件即可。

2.合唱队形问题应用了分治与动态规划的算法，先将所有队员依次做中间最高的同学，将问题分为左右来做，接下来只需要求得左边的最长上升子序列数、右边的最长下降子序列数即可。

**四、实验过程分析**

1.0-1背包问题是背包问题的进一步条件限制，考虑清楚动规方程就不难，编程中对于m(i,j)的含义要清楚，搞混了就容易出错。

2.合唱队形问题的思想并不复杂，特别是如果已经掌握了最长上升子序列数的算法，在分别处理左右最长子序列时需要特别注意数组下标，一开始我用是i，j直接从0到左右的数据长度，但是一直出错，后来发现队员身高数组并不能完全用这些下标，特别是右边的函数，数组起始下标不是0，需要利用函数传递起始下标才能调用对应的数据段。

**五、算法源代码及用户屏幕**

1.（1）算法源码

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

0-1背包问题。

codeblocks C++

2018.11.2

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <iostream>

#include <iomanip>

using namespace std;

void knapSnack(int v[], int w[], int c, int n, int m[][11]);

int main()

{

int v[] = {6, 3, 5, 4, 6};

int w[] = {2, 2 ,6, 5, 4};

int c = 10;

int n = 5;

int m[5][11];

//初始化数组

for(int i=0; i<5; i++){

for(int j=0; j<11; j++){

m[i][j] = 0;

}

}

knapSnack(v, w, c, n, m);

//输出结果

cout<<setw(3)<<" ";

for(int i=0; i<11; i++){

cout<<setw(3)<<i;

}

cout<<endl;

for(int i=0; i<5; i++){

//输出行号

cout<<setw(3)<<i+1;

for(int j=0; j<11; j++){

cout<<setw(3)<<m[i][j];

}

cout<<endl;

}

return 0;

}

void knapSnack(int v[], int w[], int c, int n, int m[][11]){

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

for(int j=0; j<11; j++){

//边界条件

if(i == 0){

if(w[i] > j)

m[i][j] = 0;

else

m[i][j] = v[i];

}

/\*动规方程

j>w[i]

m(i,j) = max{m(i-1,j), m(i-1,j-w[i])+v[i]}

0<=j<w[i]

m(i,j) = m(i-1,j)

\*/

else{

if(w[i] > j)

m[i][j] = m[i-1][j];

else{

if(m[i-1][j] > (m[i-1][j-w[i]]+v[i]))

m[i][j] = m[i-1][j];

else

m[i][j] = m[i-1][j-w[i]]+v[i];

}

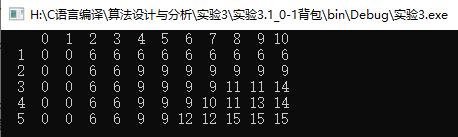
}

}//控制列数的for循环

}//控制行数的for循环

}

（2）用户屏幕



2.（1）算法源码

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

合唱队形问题

codeblocks C++

2018.11.2

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <iostream>

#include <string.h>

using namespace std;

//计算左端合唱队人数

int leftQueue(int a[], int \_start, int \_end);

//计算右端合唱队人数

int rightQueue(int a[], int \_start2, int \_end2);

int main()

{

cout<<"Please enter total number：";

int number;

cin>>number;

cout<<"Please input the height of each person (cm)："<<endl;

int a[number]; //记录每个人身高

//b数组分别记录当第n个人为合唱队中间人时,合唱队的总人数

int b[number];

int rightNumber[number]; //记录左端合唱队人数

int leftNumber[number]; //记录右端合唱队人数

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

b[i] = 0;

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

cin>>a[i];

int mostQueueNumber = b[0];

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

//设置a[i]为最高的同学

leftNumber[i] = leftQueue(a,0,i);

rightNumber[i] = rightQueue(a,i,number-1);

//计算合唱队总人数

b[i] = leftNumber[i] + rightNumber[i] - 1;

//计算合唱队最多的总人数

if(mostQueueNumber < b[i])

mostQueueNumber = b[i];

}

//计算最少出队人数

int leastDequeueNumber = number - mostQueueNumber;

cout<<"Minimum number of people out: "<<leastDequeueNumber<<endl;

return 0;

}

int leftQueue(int a[], int \_start, int \_end){

int leftMostNumber = 0;

int n = \_end-\_start+1;

//c数组记录i时的最长上升子序列数

int c[n];

int maxN;

//初始化最长上升子序列数为1

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

c[i] = 1;

}

for(int i=\_start; i<\_end+1; i++){

maxN = 0;

for(int j=i-1; j>=\_start; j--){

if(a[j]<a[i] && c[j]>maxN)

maxN = c[j];

c[i] = maxN + 1;

}

}

leftMostNumber = c[n-1];

return leftMostNumber;

}

int rightQueue(int a[], int \_start2, int \_end2){

int rightMostNumber = 0;

int n2 = \_end2-\_start2+1;

//c2数组记录i时的最长下降子序列数

int c2[n2];

int maxN2;

//初始化最长下降子序列数为1

for(int i=0; i<n2; i++){

c2[i] = 1;

}

for(int i=\_end2; i>=\_start2; i--){

maxN2 = 0;

for(int j=i+1; j<=\_end2; j++){

if(a[j]<a[i] && c2[j-\_start2]>maxN2)

maxN2 = c2[j-\_start2];

c2[i-\_start2] = maxN2 + 1;

}

}

rightMostNumber = c2[0];

return rightMostNumber;

}

（2）用户屏幕

