**实验二 动态规划算法（4学时）**

**基本题一：最长公共子序列问题**

**一、实验目的与要求**

1、熟悉最长公共子序列问题的算法；

2、初步掌握动态规划算法；

**二、实验题**

    若给定序列X={x1,x2,…,xm}，则另一序列Z={z1,z2,…,zk}，是X的子序列是指存在一个严格递增下标序列{i1,i2,…,ik}使得对于所有j=1,2,…,k有：zj=xij。例如，序列Z={B，C，D，B}是序列X={A，B，C，B，D，A，B}的子序列，相应的递增下标序列为{2，3，5，7}。

给定2个序列X和Y，当另一序列Z既是X的子序列又是Y的子序列时，称Z是序列X和Y的公共子序列。

给定2个序列X={x1,x2,…,xm}和Y={y1,y2,…,yn}，找出X和Y的最长公共子序列。

**三、实验提示**

include "stdlib.h"

#include "string.h"

void LCSLength(char \*x ,char \*y,int m,int n, int \*\*c, int \*\*b)

{

       int i ,j;

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

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

       for (i = 1; i <= m; i++)

          for (j = 1; j <= n; j++)

          {

            if (x[i]==y[j])

            {

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

                 b[i][j]=1;

            }

            else if (c[i-1][j]>=c[i][j-1])

            {

                 c[i][j]=c[i-1][j];

                 b[i][j]=2;

            }

            else

            {    c[i][j]=c[i][j-1];

                 b[i][j]=3;

            }

         }

}

void LCS(int i ,int j, char \*x ,int \*\*b)

{

      if (i ==0 || j==0) return;

      if (b[i][j]== 1)

      {

           LCS(i-1,j-1,x,b);

           printf("%c",x[i]);

      }

      else if (b[i][j]== 2)

           LCS(i-1,j,x,b);

      else LCS(i,j-1,x,b);

}

四：实验代码:

#include <iostream>

#include<cstring>

using namespace std;

#define max 100

void LCS(int i,int j,char \*x,int b[][max])

{

if(i==0||j==0) return;

if(b[i][j]==1)

{

LCS(i-1,j-1,x,b);

cout<<x[i-1];

}

else if(b[i][j]==2)

LCS(i-1,j,x,b);

else LCS(i,j-1,x,b);

}

int main()

{

int m;

int n;

char x[100];

char y[100];

cout<<"请输入两个字符串:"<<endl;

cin>>x>>y;

m=strlen(x);

n=strlen(y);

int c[m+1][n+1];

int b[max][max];

int i,j;

for(i=0;i<=m;i++) c[i][0]=0;

for(i=0;i<=n;i++) c[0][i]=0;

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

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

{

if(x[i-1]==y[j-1])

{

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

b[i][j]=1;

}

else if(c[i-1][j]>=c[i][j-1])

{

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

b[i][j]=2;

}

else

{

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

b[i][j]=3;

}

}

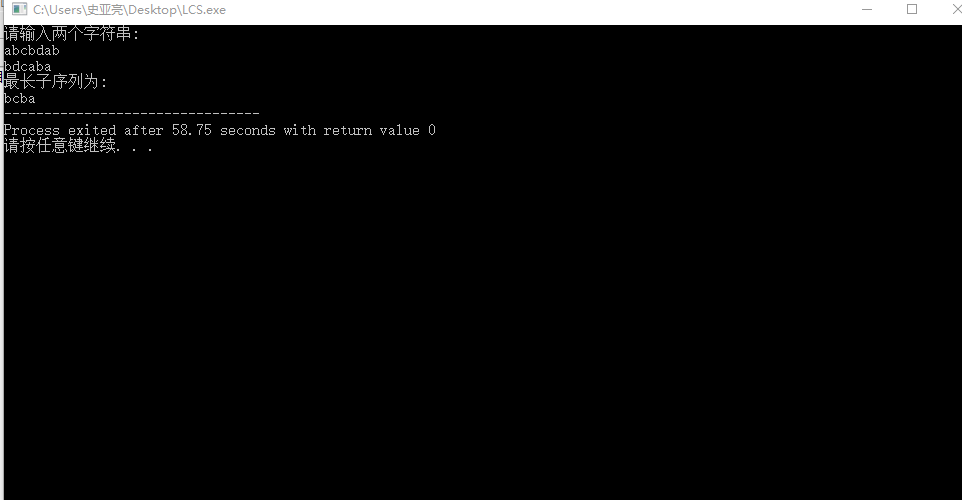
cout<<"最长子序列为:"<<endl;

LCS(m,n,x,b);

return 0;

}

五．实验结果:



**基本题二：最大字段和问题**

**一、实验目的与要求**

1、熟悉最长最大字段和问题的算法；

2、进一步掌握动态规划算法；

**二、实验题**

    若给定n个整数组成的序列a1，a2，a3，……an，求该序列形如ai＋ai＋1＋……＋an的最大值。

**三、实验提示**

int MaxSum(int n,int \*a,int &besti,int &bestj)

{

  intsum=0;

  for(int i=1;i<=n;i++)

  for(int j=i;j<=n;j++)

   {

     int thissum=0;

     for(int K=i;k<=j;k++)thissum+=a[k];

     if(thissum>sum)

       {

         sum=thissum;

         besti=i;

         bestj=j;

        }

     }

    return sum;

 }

int MaxSum(int n,int \*a,int &besti,int &bestj)

 {

   intsum=0;

   for(int i=1;i<=n;i++)

   {

     int thissum=0;

     for(intj=i;j<=n;j++)

      {

        thissum+=a[j];

        if(thissum>sum)

        {

          sum=thissum;

           besti=i;

           bestj=j;

          }

         }

    }

    return sum;

}

四：实验代码:

#include<iostream>

using namespace std;

int maxsum(int n,int \*a)

{

int sum=0,b=0;

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

{

if(b>0) b+=a[i];

else b=a[i];

if(b>sum)

sum=b;

}

return sum;

}

int main()

{

int n;

int a[100];

cout<<"请输入数组长度:"<<endl;

cin>>n;

cout<<"请输入数组"<<endl;

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

{

cin>>a[i];

}

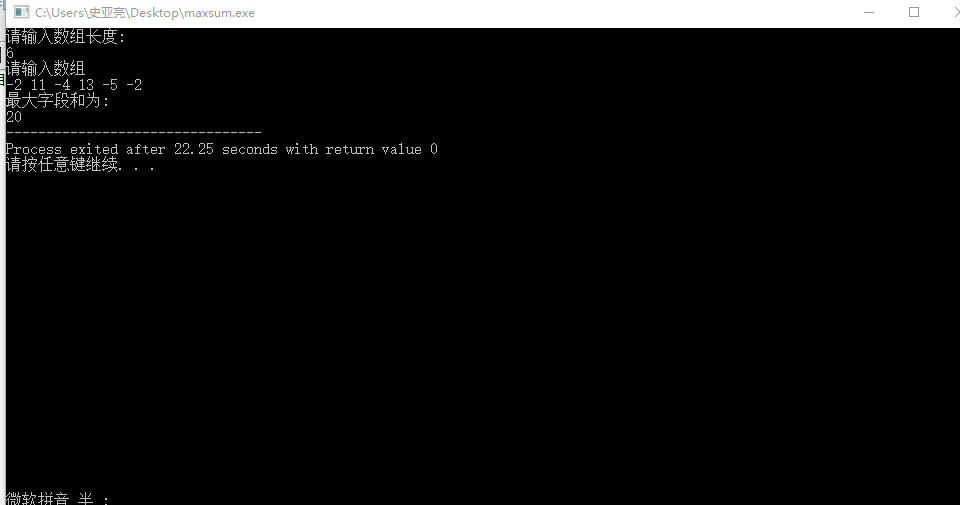
cout<<"最大字段和为:"<<endl;

cout<<maxsum(n,a);

return 0;

}

五：实验结果:



**提高题一： 用动态规划法求解0/1背包问题**

一、实验要求与目的

1. 掌握动态规划算法求解问题的一般特征和步骤。
2. 使用动态规划法编程，求解0/1背包问题。

二、实验内容

1. 问题描述：给定n种物品和一个背包，物品i的重量是Wi，其价值为Vi，问如何选择装入背包的物品，使得装入背包的物品的总价值最大？
2. 算法描述。
3. 程序实现；给出实例测试结果。

三；实验代码:

#include<iostream>

using namespace std;

#define max1 100

int min(int a,int b)

{

if(a>=b)return b;

else return a;

}

int max(int a,int b)

{

if(a>=b)return a;

else return b;

}

void bag(int \*v,int \*w,int c,int n,int m[][max1])

{

int jmax=min(w[n]-1,c);

for(int j=0;j<=jmax;j++) m[n][j]=0;

for(int j=w[n];j<=c;j++) m[n][j]=v[n];

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

{

jmax=min(w[i]-1,c);

for(int j=0;j<=jmax;j++)

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

for(int j=w[i];j<=c;j++)

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

}

m[1][c]=m[2][c];

if(c>=w[1])

m[1][c]=max(m[1][c],m[2][c-w[1]]+v[1]);

}

void traceback(int m[][max1],int \*w,int c,int n,int \*x)

{

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

if(m[i][c]==m[i+1][c])

x[i]=0;

else

{

x[i]=1;

c-=w[i];

}

x[n]=(m[n][c])?1:0;

}

int main()

{

int v[max1];

int w[max1];

int c;

int n;

int m[max1][max1];

int x[max1];

cout<<"输入物品种类:"<<endl;

cin>>n;

cout<<"输入重量数组:"<<endl;

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

{

cin>>w[i];

}

cout<<"请输入价值数组:"<<endl;

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

{

cin>>v[i];

}

cout<<"请输入背包容量:"<<endl;

cin>>c;

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

traceback(m,w,c,n,x);

cout<<"结果如下:"<<endl;

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

cout<<x[i];

return 0;

}

四：实验结果：

