# 算法知识之最长公共子序列问题(动态规划)



**最近朋友让帮做个关于动态规划的最长公共子序列的问题,翻看以前的笔记并完成该题后,顺便写这样一篇文章,希望对大家有所帮助,同时也帮助自己回顾该知识点.**

## 一.最长公共子序列的定义

**子序列: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.**

**公共子序列: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.**

**最长公共子序列: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.**

**如：序列ABCDEF和ADFGH的最长公共子序列为ADF**

**注意:最长公共子串(Longest Common Substirng)和最长公共子序列(Longest Common Subsequence,简称LCS)的区别为是最长公共子串的串是一个连续的部分,而最长公共子序列则是从不改变序列的顺序,而从序列中去掉任意的元素而获得新的序列;通俗的说就是子串中字符的位置必须是连续的而子序列则可以不必连续.**

## 二.最优子结构性质

**设序列X={x1,x2,...,xm}和Y={y1,y2,...,yn}的最长公共子序列为Z={z1,z2,...,zk} ,则**

**(1)若xm=yn,则zk=xm=yn,且z1,z2,..., zk-1是否为x1,x2,...,xm-1和y1,y2,...,yn-1的最长公共子序列.**

**(2)若xm≠yn且zk≠xm,则Z是x1,x2,...,xm-1和Y的最长公共子序列.**

**(3)若xm≠yn且zk≠yn,则Z是X和y1,y2,...,yn-1的最长公共子序列.**

**由此可见,2个序列的最长公共子序列包含了这2个序列的前缀的最长公共子序列.因此,最长公共子序列问题具有最优子结构性质.当问题具有最优子结构性质和子问题重叠性质时就可以用动态规划算法解决该问题.**

## 三.动态规划方法分析

**由最长公共子序列问题的最优子结构性质建立子问题最优值的递归关系.用c[i][j]记录序列和的最长公共子序列的长度.其中,Xi={x1,x2,...,xi},Yj={y1,y2,...,yj}.当i=0或j=0时,空序列是Xi和Yj的最长公共子序列.故此时C[i][j]=0.其它情况下,由最优子结构性质可建立递归关系如下:**

$$c[i][j]=\begin{cases}0 & i=0,j=0\\ c[i-1][j-1]+1 & i,j>0;x_i=y_j\\ \max\{c[i][j-1],c[i-1][j]\} & i,j>0;x_i\neq y_j\end{cases}$$

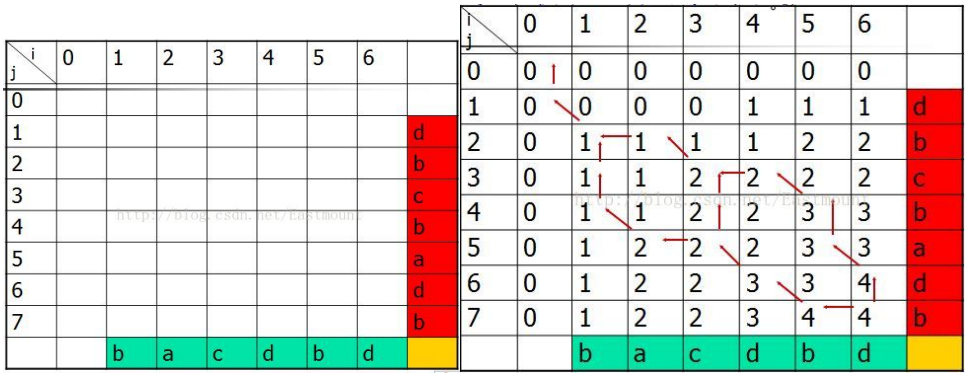


**其对应的核心代码如下:**

```
//参数:x字符串长度为m y字符串长度为n
void LCSLength(char x[], char y[],int m, int n)
{
        /* 计算最长公共子序列的长度 */
        int L[m][n],i, j;
        for (i = 0; i <= m; i++) L[i][0] = 0;
        for (i = 0; i <= n; i++) L[0][i] = 0;
        for (i = 1; i <= m; i++)
```

```
        {           for (j = 1; j <= n; j++)
           {
              if (x[i]==y[j])
                   L[i][j]=L[i-1][j-1]+1;
              else if (L[i-1][j]>= L[i][j-1])
                   L[i][j]= L[i-1][j];
              else
                   L[i][j]= L[i][j-1];
           }
        }
        return L[m][n];
}
```

**例如:输入字符串“bdcaba”和"abcbdab",求它们的最长公共子序列长度.在《算法设计与分析》课程中我们老师讲述的方法通常是使用动态规划填充表格方法解决.初始时,X字符串的长度为m,Y字符串的长度为n.c[m,n]二位数组如上面递归关系递归,最后的c[m,n]为最大数字即最长公共子序列的长度.**



| i\j | 0 | 1 | 2 | 3 | 4 | 5 | 6 | |
|---|---|---|---|---|---|---|---|---|
| 0 | | | | | | | | |
| 1 | | | | | | | | d |
| 2 | | | | | | | | b |
| 3 | | | | | | | | c |
| 4 | | | | | | | | b |
| 5 | | | | | | | | a |
| 6 | | | | | | | | d |
| 7 | | | | | | | | b |
| | | b | a | c | d | b | d | |

| i\j | 0 | 1 | 2 | 3 | 4 | 5 | 6 | |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |
| 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | d |
| 2 | 0 | 1 | 1 | 1 | 1 | 2 | 2 | b |
| 3 | 0 | 1 | 1 | 2 | 2 | 2 | 2 | c |
| 4 | 0 | 1 | 1 | 2 | 2 | 3 | 3 | b |
| 5 | 0 | 1 | 2 | 2 | 2 | 3 | 3 | a |
| 6 | 0 | 1 | 2 | 2 | 3 | 3 | 4 | d |
| 7 | 0 | 1 | 2 | 2 | 3 | 4 | 4 | b |
| | | b | a | c | d | b | d | |

**其中从表中找出最长公共子序列的方法**

**(1) 从(m,n)到(0,0)**

**(2) 若当前格与左边一格相同,则画" 一";若当前格与上边一格相同,则画"|";上两者都不符合,从当前格到左上格化斜线箭头"\";**

**(3) 从当前格向箭头方向前进一格,对此格进行(2)**

**(4) 从(m,n)到(0,0)的不同路径中,斜线箭头"\"相对应的格的元素构成最长公共子序列.如图bcbd、bcdb、badb.**

# 四.问题的提出与解决

## 1.问题

**题目:求两个字符串的最长公共子序列的长度.**

**输入:第一行字符串S1,第二行字符串 S2 (1<=字符串长度<=1000).输出:数字M,为最长公共子序列长度.测试用例如下:**

**输入**

**BDCABA**

**ABCBDAB**

**输出**

**4**

**输入**

**ABKLMNABCDI**

**ABCDEFGHIJKLMNOPQRSTUVWXYZ**

**输出**

**6**

## 2.代码

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
int *pln1 , *pln2;
char a[10010] , b[10010];
int main()
```

```
{         int i , j , lena , lenb ;
        gets(a);
        gets(b);
        lena = strlen(a);
        lenb = strlen(b);
        pln1 = (int*)calloc( lenb + 1 , sizeof(int) );
        memset( pln1 , 0 , sizeof(pln1) );
        pln2 = (int*)calloc( lenb + 1 , sizeof(int) );
        memset( pln2 , 0 , sizeof(pln2) );
        for( i = 1 ; i <= lena ; i++ )
        {
                for( j = 1 ; j <= lenb ; j++ )
                {
                        if( a[i-1] == b[j-1] )
                        {
                                pln2[j] = pln1[j-1] + 1;
                        }
                        else
                        if( pln1[j] >= pln2[j-1] )
                        {
                                pln2[j] = pln1[j];
                        }
                        else
                        {
                                pln2[j] = pln2[j-1];
                        }
                }
                free(pln1);
                pln1 = pln2;
                pln2 = (int*)calloc( lenb + 1 , sizeof(int) );
                memset( pln2 , 0 , sizeof(pln2) );
        }
        printf( "%d\n" , pln1[lenb] );
        return 0;
}
```

# 五.问题的升华与解决

## 1.升华问题

**输入:输入文件中的第1行是一个正整数T(0<T<=10),表示有T组测试数据.接下来是每组测试数据的描述,每组测试数据有3行.测试数据的第1行有2个正整数m、n,中间用一个空格隔开(0<m，n<50);第2、3行是长度分别为m、n的2个序列X和Y,每个序列的元素间用一个空格隔开.序列中每个元素由字母、数字等构成.**输入直到文件结束

**输出:对输入中的每组测试数据,先输出Case #表示第几组数据,在输出最长公共子序列,输出所有的最长公共子序列,并输出动态规划表格c表和b表.(测试用例见结果图)**

## 2.代码

**这里涉及到一个新的问题：就是使用上面所叙述的填充表格来实现动态规划,其中c[m,n]记录的是当前序列的最长子序列长度;还需要引用一个吧b[m,n]表来寻找所有最长公共子序列,并把结果存入到result[]数组中.其中最重要的代码就是两个实现的函数,如下:**
**第一个LSCLength函数是求最长公共子序列长度的函数,并在该函数中填充c[m][n]和b[m][n].**

```
//函数：计算最优值
//参数：m字符串X长 n字符串Y长 X字符串 Y字符串 b标志数组寻找所有字符串用
int LSCLength( int m, int n, char *X, char *Y, int b[][100] )
{
        /*计算最长公共子序列的长度*/
        int num[100][100];
```

```
int i,j;          int sum;

/*清零*/
for( i=0 ; i<=m ; i++ )
{
        for( j=0 ; j<=n ; j++ )
        {
                num[i][j]=0;
                b[i][j]=0;
        }
}

/* 递归结构-动态规划并输出 */
for( i=1 ; i<=m ; i++ )
{
        for( j=1 ; j<=n ; j++ )
        {
                if( X[i]==Y[j] ) {
                        num[i][j]=num[i-1][j-1]+1;
                        b[i][j]=1;
                }

                else if( num[i-1][j]>num[i][j-1] ) {
                        num[i][j]=num[i-1][j];
                        b[i][j]=2;
                }

                else if( num[i-1][j]<num[i][j-1] ){
                        num[i][j]=num[i][j-1];
                        b[i][j]=3;
                }
                else {
                        num[i][j]=num[i][j-1];
                        b[i][j]=4;
                }
        }
}

sum = num[m][n];
printf("最长公共子序列的长度：%d\n",sum);

//输出c[m][n]表
printf("\n");
for(i=0;i<=m;i++)
{
        for(j=0;j<=n;j++)
        {
                printf("%d ",num[i][j]);
        }
        printf("\n");
}
//输出b[m][n]表
printf("\n");
for(i=0;i<=m;i++)
{
        for(j=0;j<=n;j++)
        {
                printf("%d ",b[i][j]);
        }
        printf("\n");
}
```

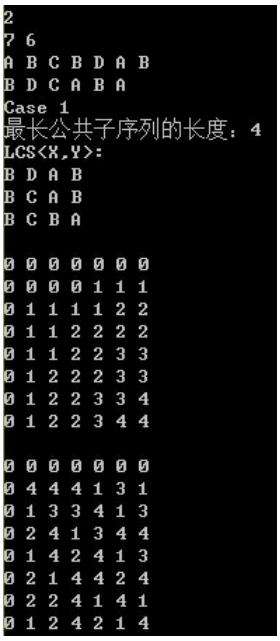
```
                return sum;
}
```

**第二个DisplayLSC函数是通过b[m][n]递归计算所有最长公共子序列,并存储至result数组中.**

```
//定义全局变量用于保存结果result 结果个数保存为count
char result[100];
int count=0;

//函数：计算所有最长公共子序列
//参数：m字符串X的长度 n字符串Y的长度 b标志数组 current_len当前长度 max_len最长公共子序列长度
void DisplayLSC(int i,int j,char *X,int b[][100],int current_len,int max_len)
{
        int s;

        //采用递归的算法求解所有长度
        if(i==0 || j==0)                //为0时输出结果并返回
        {
                for(s=0;s<max_len;s++)
                {
                        printf("%c ",result[s]);
                }
                printf("\n");
                count++;
                return;
        }

        if(b[i][j]==1)
        {
                current_len--;
                result[current_len]=X[i];
                DisplayLSC(i-1,j-1,X,b,current_len,max_len);
        }
        else
        {
                if(b[i][j]==2)
                {
                        DisplayLSC(i-1,j,X,b,current_len,max_len);
                }
                else
                {
                        if(b[i][j]==3)
                        {
                                DisplayLSC(i,j-1,X,b,current_len,max_len);
                        }
                        else
                        {
                                DisplayLSC(i,j-1,X,b,current_len,max_len);
                                DisplayLSC(i-1,j,X,b,current_len,max_len);
                        }
                }
        }
}
```

## 3.结果

**最后输出的结果如下图所示:**





**希望该文章对大家有所帮组,同时该文章参考了王晓东的《计算机算法设计与分析》,并引用了自己学校的PPT动态规划内容.同时感谢梦醒潇湘love博主的文章,希望大家也可以去见解该文章.**

**http://blog.chinaunix.net/uid-26548237-id-3374211.html**

**文章主要是对自己以前学过的知识的巩固以记录,如果有错误或不足之处,希望大家海涵.**

**(By:Eastmount 2013-11-5 中午3点http://blog.csdn.net/eastmount/)**