实验五：最长公共子序列（LCS）

学号：P21000224　　姓名：陈鸿绪　　日期：11.5

实验内容：给定两个字符串 text1 和 text2，返回这两个字符串的最长公共子序列的长度。 一个字符串的子序列是指这样一个新的字符串：它是由原字符串在不改变字符的相对 顺序的情况下删除某些字符（也可以不删除任何字符）后组成的新字符串。若这两个字符串没有公共子序列，则返回 0。

实验要求：

1. 编程实现最长公共子序列（LCS）算法，并理解其核心思想。
2. 时间复杂度 O(mn)，空间复杂度 O(mn)，求出 LCS 及其长度。
3. 时间复杂度 O(mn)，空间复杂度 O(2*min(m,n))，求出 LCS 的长度。
4. 时间复杂度 O(mn)，空间复杂度 O(min(m,n))，求出 LCS 的长度。

算法设计思路：

1. 对于普通 LCS，可以直接设置一个 m×n 的矩阵，直接动态规划即可。
2. 对于时间复杂度 O(mn)，空间复杂度 O(m×n)的 LCS，考虑到每做一次动态规划的一层，只会依赖于上一层，所以不需要 O(mn)的空间，只需要在做一行的同时再加上一层的空间，所以有空间复杂度 O(2*min(m,n))。
3. 对于时间复杂度 O(mn)，空间复杂度 O(2*min(m,n))的 LCS，考虑到每做一次动态规划的时候，当前位置的值只会和上方、左上方以及左边有关，所以只需要每次拿出 O(1)的空间存储左上方即可。

主要源码以及注释：

```
void LCS_1(int m,int n,char *p,char *q){
//时间复杂度 O(mn)，空间复杂度 O(mn)，求出 LCS 及其长
    int temp[m+1][n+1];//n<=m,设定动态规划所用矩阵
    for(int i=0;i<=n;i++) temp[0][i]=0;
```

```
    for(int i=0;i<=m;i++) temp[i][0]=0;
    for(int i=1;i<=m;i++){
        for(int j=1;j<=n;j++){
            if(p[i]==q[j]) temp[i][j]=temp[i-1][j-1]+1;
            else temp[i][j]=(temp[i-1][j]>temp[i][j-1])?temp[i-1][j]:temp[i][j-1];
        }
    }//动态规划过程
    if(temp[m][n]==0){
        printf("NULL\n");
        return;
    }//如果没有打印出 NULL
    else{
        printf("%d\n",temp[m][n]);
        char s[N];int x=m,y=n,s_len=0;
        while(x&&y){
            if(p[x]==q[y]) s[s_len++]=q[y],x--,y--;
            else if(temp[x][y-1]>=temp[x-1][y]) y--;
            else x--;
        }
        printf("LCS1: ");
        for(int i=s_len-1;i>=0;i--) printf("%c ",s[i]);
        printf("\n");
    }//否则打印出最长公共子串
    printf("\n");
}
void LCS_2(int m,int n,char *p,char *q){
//时间复杂度 O(mn)，空间复杂度 O(2*min(m,n))，求出 LCS 的长度
    int temp1[n+1]={0},temp2[n+1]={0};//n<=m
    //一个数组是上一层的状态，一个数组是当前层的状态。
    for(int i=1;i<=m;i++){
        for(int j=1;j<=n;j++){
            if(p[i]==q[j]) temp1[j]=temp2[j-1]+1;
            else temp1[j]=(temp1[j-1]>temp2[j])?temp1[j-1]:temp2[j];
        }
        for(int k=1;k<=n;k++) temp2[k]=temp1[k];
    }//动态规划
    if(temp1[n]==0){
        printf("NULL\n");
        return;
    }//如果为 0 打印 NULL
    else printf("LCS2: %d\n",temp1[n]);
    printf("\n");
}
void LCS_3(int m,int n,char *p,char *q){
```

```
//时间复杂度 O(mn)，空间复杂度 O(min(m,n))，求出 LCS 的长度
    int temp[n+1]={0},s=0;//n<=m //空间复杂度 O(min(m,n))
    for(int i=1;i<=m;i++){
        for(int j=1;j<=n;j++){
            if(p[i]==q[j]) temp[j]=s+1;
            else{
                s=temp[j];
                temp[j]=(temp[j-1]>temp[j])?temp[j-1]:temp[j];
            }
        }
    }//动态规划
    if(temp[n]==0){
        printf("NULL\n");
        return;
    }//如果为 0 打印 NULL
    else printf("LCS3: %d\n",temp[n]);
    printf("\n");
}
```

算法测试结果:

```
26
LCS1: b c d e f g h i g k l m n o p q r s t u v w x y z

LCS2: 26

LCS3: 26
```

对读入文件中的两个字符串，程序有如上的结果，经过直接比较不难验证程序的正确性。