最长公共子序列（LCS）

SA20225085 朱志儒

实验内容

编程实现最长公共子序列（LCS）算法，并理解其核心思想。

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。例如，"ace"是"abcde"的子序列，但"aec"不是"abcde"的子序列。两个字符串的「公共子序列」是这两个字符串所共同拥有的子序列。

若这两个字符串没有公共子序列，则返回 0。

程序输入：

由控制台输入两个字符串 text1，text2。

其中：

1 <= text1.length <= 1000

1 <= text2.length <= 1000

输入的字符串只含有小写英文字符（a~z）

程序输出：

控制台打印 LCS 的长度及相应的序列，如不存在公共子序列则返回 0。

示例 1：

输入：text1 = “abcde”，text2 = “ace”

输出：LCS：“abc”，长度：3

示例 2：

输入：text1 = “abc”，text2 = “def”

输出：0

实验目的

通过本实验加深对最长公共子序列（LCS）算法的理解和运用。

算法设计思路

1. LCS 最优解的结构特征：

设序列$X=(x_1,x_2,\dots,x_m)$和$Y=(y_1,y_2,\dots,y_n)$，$Z=(z_1,z_2,\dots,z_k)$是 X 和 Y 的任意一个 LCS，则

（1） 若$x_m=y_n \Rightarrow z_k=x_m=y_n$且$Z_{k-1}$是$X_{m-1}$和$Y_{m-1}$的一个 LCS；

（2） 若$x_m\neq y_n, z_k\neq x_m \Rightarrow$ Z 是X_{m-1}和 Y 的一个 LCS；

（3） 若$x_m\neq y_n, z_k\neq y_n \Rightarrow$ Z 是 X 和Y_{n-1}的一个 LCS；

由此可见，2 个序列的最长公共子序列可由（1）（2）（3）算出，（2）（3）的解时对应子问题的最优解。

2. 子问题的递归解：

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

3. 数据结构设计：

$c[0\dots m,0\dots n]$ //存放最优解值，计算时行优先

$b[1\dots m,1\dots n]$ //解矩阵，存放构造最优解信息

$$b[i,j]=\begin{cases}\nwarrow, \text{如果 } c[i,j]\text{由 } c[i-1,j-1]\text{确定}\\ \uparrow, \text{ 如果 } c[i,j]\text{由 } c[i-1,j]\text{确定}\\ \leftarrow, \text{ 如果 } c[i,j]\text{由 } c[i,j-1]\text{确定}\end{cases}$$

当构造解时，从 b[m, n]出发，上溯至 i = 0 或 j = 0 为止，上溯过程中，当 b[i,j]包含“↖”时打印出$x_i(y_i)$。

4. 伪代码：

```
LCSlength(X, Y){
    m ← length[X]; n ← length[Y];
    for i ← 0 to m do c[i, 0] ← 0;
    for j ← 0 to n do c[0, j] ← 0;
    for i ← 1 to m do
        for j ← 1 to n do
            if x_i = y_i then
                {c[i, j] ← c[i − 1, j − 1] + 1; b[i, j] ← " ↖ ";}
```

else if c[i − 1, j] ≥ $c[i, j-1]$ *then*

$\{c[i,j] \leftarrow c[i-1,j]; b[i,j] \leftarrow$ "↑";}

else

$\{c[i,j] \leftarrow c[i,j-1]; b[i,j] \leftarrow \leftarrow;\}$

return b and c; }

PrintLCS(b, X, i, j) {

if i = 0 or j = 0 then return;

if b[i, j] = "↖" then {

PrintLCS(b, X, i − 1, j − 1);

print x_i; }

else if b[i, j] = "↑" then PrintLCS$(b, X, i-1, j)$;

else PrintLCS$(b, X, i, j-1)$; }

源码+注释

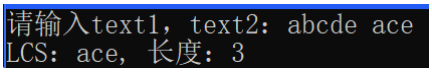
```
1.  void LCS_length(int** c, int** b, string x, string y) {
2.      int m = x.length() - 1, n = y.length() - 1;
3.      for (int i = 0; i <= m; ++i)
4.          c[i][0] = 0;
5.      for (int j = 0; j <= n; ++j)
6.          c[0][j] = 0;
7.      for (int i = 1; i <= m; ++i) {
8.          for (int j = 1; j <= n; ++j) {
9.              if (x[i] == y[j]) {
10.                 c[i][j] = c[i - 1][j - 1] + 1;
11.                 b[i][j] = 0;    //指向↖
12.             }
13.             else if (c[i - 1][j] >= c[i][j - 1]) {
14.                 c[i][j] = c[i - 1][j];
15.                 b[i][j] = 1;    //指向↑
16.             }
17.             else {
18.                 c[i][j] = c[i][j - 1];
19.                 b[i][j] = 2;    //指向←
20.             }
21.         }
```

```
    }
}

void get_LCS(int** b, string x, int i, int j, string& result) {
    if (i == 0 || j == 0)
        return;
    if (b[i][j] == 0) {          //指向↖
        get_LCS(b, x, i - 1, j - 1, result);
        result.push_back(x[i]);
    }
    else if (b[i][j] == 1) {     //指向↑
        get_LCS(b, x, i - 1, j, result);
    }
    else {                       //指向←
        get_LCS(b, x, i, j - 1, result);
    }
}

int main() {
    cout << "请输入 text1，text2：";
    string str1, str2;
    cin >> str1 >> str2;
    str1 = ' ' + str1;
    str2 = ' ' + str2;
    int m = str1.length() - 1, n = str2.length() - 1;
    //初始化二维数组 c
    int** c = new int* [m + 1];
    for (int i = 0; i <= m; ++i)
        c[i] = new int[n + 1];
    //初始化二维数组 b
    int** b = new int* [m + 1];
    for (int i = 0; i <= m; ++i)
        b[i] = new int[n + 1];
    LCS_length(c, b, str1, str2);
    int len = c[m][n];
    if (len > 0) {
        string result;
        get_LCS(b, str1, m, n, result);
        cout << "LCS：" << result << ", ";
    }
    cout << "长度：" << len << endl;
    return 0;
}
```

算法测试结果

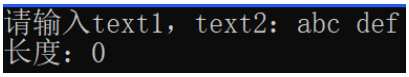
输入：text1 = “abcde”，text2 = “ace”

输出：



输入：text1 = “abc”，text2 = “def”

输出：



实验过程中遇到的困难及收获

在本次实验中，我遇到了一个小小的问题：伪代码中的下标是从 1 开始的，而我实现的 C++代码的下标是从 0 开始的，这导致我在测试程序正确性的时候出现了问题。其实解决方案很简单，只需把声明的数组的长度加 1，下标为 0 的位置不使用，实现算法时下标从 1 开始。