最长公共子序列输出实现报告

王资 18214668 计算机技术

1.基本思想

1.1 利用步骤备忘录 s 的信息输出最长公共子序列

若 m[i][j]=1，则把该字符 x[i]放入待输出字符串中，且 i-1,j-1；若 m[i][j]=2，则 i-1；m[i][j]=3，则 j-1，直到 i=0 或 j=0。i, j 的初始值分别为 m, n. 这是把在构造 s 矩阵时反向读取利用 s 的信息。

1.2 仅使用备忘录 c 的信息输出最长公共子序列

对于 m[i][j]，若 x[i]=y[j]，则把字符 x[i]放入待输出字符串中，且 i-1, j-1；若否，则对比 m[i][j]与 m[i][j-1]，若相等，则 j-1；若不相等，对比 m[i][j]与 m[i-1][j]，若相等，i-1.直到 m[i][j]=0，i, j 的初始值分别为 m, n. 在构造 c 的过程中，已知有且只有以上三种情况的逆过程。

2.代码中的数据结构

char* x, char* y 为分别储存序列 x,y 的字符串，memo[][]为记录最长公共子序列长度的备忘录矩阵，s[][]为记录构造最长公共子序列长度过程的备忘录矩阵。

3.程序流程

1.分别输入序列 x, y

2.计算序列 x, y 的长度，将 memo[][]和 s[][]填充完毕

3.输出 memo[m][n]，为序列 x, y 的长度

4.利用 s 的信息，从 s[m][n]开始，按基本思想所描述，反推出序列 x, y 的的构造过程，并输出所构造的

5.利用 memo 的信息，从 memo[m][n]开始，按基本思想所描述，反推出序列 x, y 的的构造过程，并输出所构造的

4.调试过程出现的问题和解决方法

本次实验的问题主要出自课件提供的 LCSLength 函数代码有误。

```
int LCSLength(int m, int n, char *x,char *y,int **c,int **s) {
  int i,j;
  for(i=1;i<=m;i++) c[i][0]= 0;
  for(i=1;i<=m;i++) c[0][i]= 0;
  for(i=1;i<=m;i++)
     for(j=1;i<=n;i++)  {
        if (x[i]==y[j]) {
          c[i][j]= c[i-1][j-1]+1;  s[i][j]="1";
          }
         else   if (c[i-1][ j]>=c[i][j-1]) {
                  c[i][j]= c[i-1][j];  s[i][j]="2";
                  }
         else      {
                   c[i][j]= c[i][j-1];  s[i][j]="3";
                 }
       }
}
```

1.该函数不需要返回值

2.s[i][j]的赋值类型应为 int

3.第二个初始化循环应为 i=1:n

4.动态规划循环中的内循环中应是 j<=n;j++

5.对于 C++语言，字符串（数组）从 0 位开始计算，因此涉及到 x,y 的索引都应该-1

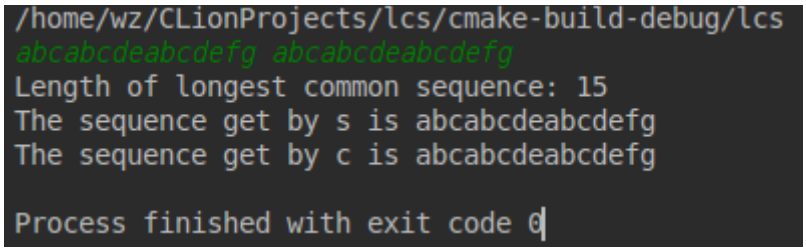
5.运行结果

输入 abedjiend apppend

```
/home/wz/CLionProjects/lcs/cmake-build-debug/lcs
abedjiend apppend
Length of longest common sequence: 4
The sequence get by s is aend
The sequence get by c is aend

Process finished with exit code 0
```

输入 abcabcdeabcdefg abcabcdeabcdefg. 该输入为课件提供的例子



两种得到最长公共子序列的方法的输出都相同且容易对比知道是正确的。

6.总结

该实验实现了使用步骤备忘录矩阵 s 找回找到最长公共子序列的长度的构造过程，从而得到该序列；以及仅使用备忘录矩阵 c 得到所构造最长公共子序列。使用矩阵 s 构造的时间复杂度是 O(n+m)因为索引 i 和 j 在每次循环至少有一者会减少 1，直到其中一者为 0.而使用矩阵 c 构造的时间复杂度也是 O(n+m)，因为每个元素 c[i][j]最多和 2 个相邻元素对比或 x[i]与 y[j]对比，根据对比结果 i,j 至少有一者会减少 1, 直到 c[i][j]为 0，也即 i,j 其中一者为 0（初始化定义）.所以对于该问题，总体来说不使用矩阵 s 在时间复杂度上不会变得特别大，却在空间上节省了 O(m*n)的空间，因而不使用矩阵 s 是更优的方案。

7.存在问题和改进设想

使用这种回溯的方法，当最长公共子序列存在多个的时候，仅会输出其中一种。如果要输出所有的情况，则对于 s 矩阵，需要关注全部元素为 1 的位置，以确定在哪些地方公共子序列会变长；对于 c 矩阵，需要关注所有 c[i][j]=c[i – 1][j - 1] + 1 的元素，以确定在哪些地方公共子序列会变长，同时要确定变化的点之间是否冲突，即是否最长公共子序列在同一个点有多种可能新增一个字符，但这个字符不同，最后还要确定得到的所有最长公共子序列是否有重复并将重复的去除，时间复杂度会达到 O(mn)。