算法基础 Lab 5

最长公共子序列

张芷苒

PB21081601

November 9, 2023

Part 1: 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。

Part 2: 算法设计思路

此问题可以用动态规划的思想解决:

$$c[i,j] = \begin{cases} 0 & \text{若} i=0 \text{ 或} j=0 \\ c[i-1,j-1] & \text{若} i,j>0 \text{ 且} x_i = y_i \\ \max(c[i,j-1], c[i-1,j]) & \text{若} i,j>0 \text{ 且} x_i \neq y_i \end{cases}$$

方法一: 时间复杂度 $O(mn)$ ，空间复杂度 $O(mn)$ ，维护两个表 $c[0..m0..n], b[1..m, 1..n]$ 分别用来记录子问题的解以及构造子问题解的方向。

方法二: 时间复杂度 $O(mn)$ ，空间复杂度 $O(2 * \min(m,n))$, 每一步计算只需要上一行和本行的数据，只需要两行记录即可。

方法三: 时间复杂度 $O(mn)$ ，空间复杂度 $O(\min(m,n))$, 事实上，计算 $c[i][j]$ 仅需要 $c[i-1][j], c[i][j-1], c[i-1][j-1]$ 即可，所以可以仅用一个数组 a 记录，a 中元素储存的内容如下:

$$\begin{cases} a[k] = c[i,k], & \text{if } 1 \leq k < j-1 \\ a[k] = c[i-1,k] & \text{if } \quad k \geq j-1 \\ a[0] = c[i,j-1] \end{cases}$$

而计算 $c[i][j]$ 需要的三个值在 a 中分别为: $c[i-1][j]=a[j], c[i][j-1]=a[0], c[i-1][j-1]=a[j-1]$.
算好之后将 $a[0]$ 放入 $a[j-1], c[i,j]$ 放入 $a[0]$ 即可。

Part 3: 算法实现过程

空间复杂度 O(m*n) 的算法:

```
// 空间复杂度 O(m*n) 的算法
int LCS1(const string& text1, const string& text2)
{
    int m, n;
    m = text1.size();
    n = text2.size();

    c.resize(m + 1, vector<int>(n + 1)); // 分配内存
    b.resize(m, vector<Direction>(n)); // 分配内存

    for (int i = 1; i < m + 1; i++)
    {
        c[i][0] = 0; // 初始化第一行
    }
    for (int j = 0; j < n + 1; j++)
    {
        c[0][j] = 0; // 初始化第一列
    }
    for (int i = 0; i < m; i++)
    {
        for (int j = 0; j < n; j++)
        {
            // 转移方程
            if (text1[i] == text2[j])
            {
                c[i + 1][j + 1] = c[i][j] + 1;
                b[i][j] = Up_Left;
            }
            else if (c[i][j + 1] >= c[i + 1][j])
            {
                c[i + 1][j + 1] = c[i][j + 1];
                b[i][j] = Up;
            }
```

```
            else
            {
                c[i + 1][j + 1] = c[i + 1][j];
                b[i][j] = Left;
            }
        }
    }
    if (c[m][n]) // 若c[m][n]不为0, 则打印LCS
    {
        cout << "LCS1:\"";
        Print_LCS(text1, m, n);
        cout << "\"";
    }

    return c[m][n];
}
```

空间复杂度 O(2*min(m,n)) 的算法:

```
// 空间复杂度 O(2*min(m,n)) 的算法
int LCS2(const string& text1, const string& text2)
{
    auto m = text1.size();
    auto n = text2.size();
    auto min = std::min(m, n); // 找最小值
    lcs2_1.resize(min + 1); // 分配内存
    lcs2_2.resize(min + 1); // 分配内存
    for (int i = 0; i < min + 1; i++)
    {
        lcs2_1[i] = 0; // 初始化第一行
    }
    lcs2_2[0] = 0; // 将最左侧为0当做已知
    if (m > n)
    {
        for (int i = 0; i < m; i++)
        {
            for (int j = 0; j < n; j++)
            {
                // 转移方程
                if (text1[i] == text2[j])
                {
                    lcs2_2[j + 1] = lcs2_1[j] + 1;
                }
                else
```

```
                {
                    lcs2_2[j + 1] = std::max(lcs2_2[j], lcs2_1[j + 1]);
                }
            }
            std::swap(lcs2_1, lcs2_2); // 动态滚动数组
        }
    }
    else
    {
        for (int i = 0; i < n; i++)
        {
            for (int j = 0; j < m; j++)
            {
                if (text1[j] == text2[i])
                {
                    lcs2_2[j + 1] = lcs2_1[j] + 1;
                }
                else
                {
                    lcs2_2[j + 1] = std::max(lcs2_2[j], lcs2_1[j + 1]);
                }
            }
            std::swap(lcs2_1, lcs2_2);
        }
    }
    return lcs2_2[min];
}
```

空间复杂度 O(min(m,n)) 的算法：

```
// 空间复杂度 O(min(m,n)) 的算法
int LCS3(const string& text1, const string& text2)
{
    auto m = text1.size();
    auto n = text2.size();
    auto min = std::min(m, n);
    lcs3.resize(min + 1); // 分配内存
    for (int i = 0; i < min + 1; i++)
    {
        lcs3[i] = 0;
    }
    if (m > n)
    {
        for (int i = 0; i < m; i++)
```

```
15          {
16              lcs3[0] = 0;
17              for (int j = 0; j < n; j++)
18              {
19                  // 转移方程
20                  if (text1[i] == text2[j])
21                  {
22                      std::swap(lcs3[j], lcs3[0]);
23                      lcs3[0]++;
24                  }
25                  else
26                  {
27                      lcs3[j] = lcs3[0];
28                      lcs3[0] = std::max(lcs3[j + 1], lcs3[0]);
29                  }
30              }
31              lcs3[n] = lcs3[0];
32          }
33      }
34      else
35      {
36          for (int i = 0; i < n; i++)
37          {
38              lcs3[0] = 0;
39              for (int j = 0; j < m; j++)
40              {
41                  if (text1[j] == text2[i])
42                  {
43                      std::swap(lcs3[j], lcs3[0]);
44                      lcs3[0]++;
45                  }
46                  else
47                  {
48                      lcs3[j] = lcs3[0];
49                      lcs3[0] = std::max(lcs3[j + 1], lcs3[0]);
50                  }
51              }
52              lcs3[m] = lcs3[0];
53          }
54      }
55      return lcs3[min];
56  }
```

Part 4: 实验结果分析

运行程序，可以得到以下输出结果：

```
                                    >  & 'c:\Users\Miner\.vscode\extensions\ms-vscode.cpptools-1.17.5-win32
-x64\debugAdapters\bin\WindowsDebugLauncher.exe' '--stdin=Microsoft-MIEngine-In-o4cbjbvy.2mu' '--stdout=Micro
soft-MIEngine-Out-rlkmgoar.1jj' '--stderr=Microsoft-MIEngine-Error-zfdpxtl0.awk' '--pid=Microsoft-MIEngine-Pi
d-vlilrfuv.4r0' '--dbgExe=D:\settings\mingw64\bin\gdb.exe' '--interpreter=mi'
text1=  abcvgjkjgdfkjhgfghkhtdkr    text2=  defshkhgfjgdfkjhgliytces
LCS1:"gjgdfkjhgt"    Length:    10
LCS2    Length:    10
LCS3    Length:    10
PS D:\code\cc\single\23algorithms\lab5>  & 'c:\Users\Miner\.vscode\extensions\ms-vscode.cpptools-1.17.5-win32
-x64\debugAdapters\bin\WindowsDebugLauncher.exe' '--stdin=Microsoft-MIEngine-In-3jctx34g.tpg' '--stdout=Micro
soft-MIEngine-Out-wvfwgkoo.pym' '--stderr=Microsoft-MIEngine-Error-t1ugdudi.kc4' '--pid=Microsoft-MIEngine-Pi
d-jczm5mzd.und' '--dbgExe=D:\settings\mingw64\bin\gdb.exe' '--interpreter=mi'
text1=  abcfg    text2=  defg
LCS1:"fg"    Length:    2
LCS2    Length:    2
LCS3    Length:    2
PS D:\code\cc\single\23algorithms\lab5>  & 'c:\Users\Miner\.vscode\extensions\ms-vscode.cpptools-1.17.5-win32
-x64\debugAdapters\bin\WindowsDebugLauncher.exe' '--stdin=Microsoft-MIEngine-In-xzspyjkh.usp' '--stdout=Micro
soft-MIEngine-Out-axupqnoc.2g2' '--stderr=Microsoft-MIEngine-Error-sol0xxsl.t4u' '--pid=Microsoft-MIEngine-Pi
d-p0gzr2ih.jpf' '--dbgExe=D:\settings\mingw64\bin\gdb.exe' '--interpreter=mi'
text1=  abcfg    text2=  dghj
LCS1:"g"    Length:    1
LCS2    Length:    1
LCS3    Length:    1
PS D:\code\cc\single\23algorithms\lab5>
```

图 1: 输出结果

以上结果符合预期，由此可以认为算法正确。

Part 5 实验总结

在实验过程中获得了以下收获：

- 掌握了最长公共子序列算法的核心思想，即利用动态规划的方法，将一个大问题分解为多个小问题，保存并利用子问题的最优解，避免重复计算，提高效率。
- 学习了使用 C++ 语言实现最长公共子序列算法的技巧，例如如何定义和初始化二维数组，如何使用循环和条件判断，如何使用递归和迭代等。