Longest Common Subsequence

最长公共子序列

定义：

对于序列$s_1 = \{1, 2, 3\}$来说，$\{1, 2, 3\}, \{1, 2\}, \{1, 3\}, \{2, 3\}, \{1\}, \{2\}, \{3\}, \{\}$都是$s_1$的子序列，但$\{2, 1\}$不是。子序列是由序列的子集组成的，并且相对顺序不变的序列。

问题：

查找两个序列s_1和s_2中最长公共子序列 s 的长度，其中 s 既是s_1的子序列，也是s_2的子序列，并且是所有子序列中最长的。

解法：

简单的假设序列s_1和s_2的长度为n（数组从 1 开始，范围为$[1, n]$），前i个元素组成的子序列分别为$s_1[1, i]$和$s_2[1, i]$。设$f(i, j)$为$s_1[1, i]$和$s_2[1, j]$的最长公共子序列的长度，则有如下状态转移方程：

$$f(i,j) = \begin{cases} 0 & \text{（初始化）} i = 0 \text{ 或} j = 0 \\ f(i-1, j-1) + 1 & i > 0 \text{ 且} j > 0 \text{ 且} s_1[i] = s_2[j] \\ max\big(f(i, j-1), f(i-1, j)\big) & i > 0 \text{ 且} j > 0 \text{ 且} s_1[i] \neq s_2[j] \end{cases}$$

(1) 我们用数组中的下标 0 来存储初始的固定值，对于s_1, s_2序列的前 0 个元素，他们的最长公共子序列显然是空的，即$\{\}$，因此$f(i, j) = 0$，其中$i = 0$或$j = 0$；
(2) 若$s_1[i] = s_2[j]$，则显然两个序列的这个部分是公共的，所以$f(i, j)$在$f(i-1, j-1)$的基础上加 1；
(3) 若$s_1[i] \neq s_2[j]$，则两个序列的这个部分不是公共的，所以$f(i, j)$仍然保持之前的值，为了获取最大值我们会在$f(i, j-1)$和$f(i-1, j)$中选取最大的那个；

$f(n, n)$即为序列s_1和s_2的最长公共子序列的长度值。遍历该算法的时间复杂度是$O(n^2)$。