

Longest Increasing Subsequence Extension

最长递增子序列扩展问题

问题:

在 Longest Increasing Subsequence 问题的基础上, 额外求出最长递增子序列的数量。假设序列 s 的递增子序列中, s_1 、 s_2 、 s_3 的长度都是 5, 并且在所有递增子序列中最长, 那么最长递增子序列的数量就是 3。

本问题的原型是 USACO4.3 的 “Buy Low, Buy Lower”, 本问题对其进行了简化, 不考虑子序列相同的情况。

解法:

仍然使用 <Longest Increasing Subsequence> 中的方法: 序列 s 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为 $s[1, i]$ 。设 $f(i)$ 是以 $s[i]$ 作为最后一个元素的最长递增子序列的长度, 则有如下状态转移方程:

$$f(i) = \begin{cases} 0 & \text{(初始化) } i = 0 \\ 1 & \text{(初始化) } i \in [1, n] \\ \max\{f(k) + 1\} & i > 0 \text{ 且 } s[i] > s[k], \text{ 其中 } k \in [1, i] \end{cases}$$

在此基础上, 设 $g(i)$ 是以 $s[i]$ 作为最后一个元素, 且最长递增子序列长度为 $f(i)$ 的子序列个数, 有如下状态转移方程:

$$g(i) = \begin{cases} 0 & \text{(初始化) } i = 0 \\ 1 & \text{(初始化) } i \in [1, n] \\ g[k] & i > 0 \text{ 且 } s[i] > s[k], f[k] + 1 > f[i], \text{ 其中 } k \in [1, i] \\ g[i] + g[k] & i > 0 \text{ 且 } s[i] > s[k], f[k] + 1 = f[i], \text{ 其中 } k \in [1, i] \end{cases}$$

- (1) 前 0 个元素的最长递增子序列的数量为 0 个, $g(0) = 0$;
 - (2) 长度为 $f(i)$ 的最长递增子序列的数量为 1;
 - (3) 若 $s[i] > s[k]$, 且 $f[k] + 1 > f[i]$ (即 $f[k] > f[i]$ 且 $f[k] + 1 \neq f[i]$), 这说明 $s[i]$ 作为末尾元素的最长递增子序列 (简称为 sub_i) 中, $s[i]$ 并不和 $s[k]$ 相邻, 因为如果是相邻元素则必然有 $f[k] + 1 = f[i]$ 。因此 $g[i] = g[k]$;
 - (4) 若 $s[i] > s[k]$, 且 $f[k] + 1 = f[i]$, 这说明 $s[i]$ 作为末尾元素的最长递增子序列 (简称为 sub_i) 中, $s[i]$ 和 $s[k]$ 相邻。因此 $g[i] = g[i] + g[k]$;
- 最后返回 $\max\{g(i)\}$ (其中 $i \in [1, n]$), 即 $g(i)$ 中的最大值。该算法的时间复杂度是 $O(n^2)$ 。

Buy Low, Buy Lower:

<http://intercontineo.com/article/6713331759/>

<http://jackneus.com/programming-archives/buy-low-buy-lower/>

<http://poj.org/problem?id=1952>