Longest Increase Subsequence

最长递增子序列

定义：

对于序列$s = \{3, 0, 2, 1, 4\}$来说，$\{\}$，$\{3\}$，$\{0\}$，$\{2\}$，$\{1\}$，$\{4\}$，$\{3, 4\}$，$\{0, 2\}$，$\{0, 4\}$，$\{2, 4\}$，$\{1, 4\}$，$\{0, 2, 4\}$，$\{0, 1, 4\}$都是 s 的递增子序列。递增子序列是递增的，相对顺序不变，不必是连续的。

问题：

查找序列 s 中的最长递增子序列s_{sub}的长度。

解法：

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：

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

(1) 用数组中的下标 0 来存储初始的固定值，对于 s 序列的前 0 个元素，最长递增子序列显然是空的，即$\{\}$，因此$f(0) = 0$；
(2) 对于序列 s 中所有由 1 个元素组成的序列$s[i, i]$（其中$i \in [1, n]$）只有一个元素，也算是递增子序列，因此$f(i) = 1$；
(3) 对于序列s中第i个数字$s[i]$，若$s[i] \geq s[k]$（其中$k \epsilon [1, i - 1]$）则$s[i]$与$s[1, k]$之间的部分可以组成一个更长的递增子序列，因此$f(i) = f(k) + 1$，k需要遍历$[1, i - 1]$中的所有可能的子序列；

最后返回$max\{f(i)\}$（其中$i \in [1, n]$），即$f(i)$中的最大值。该算法的时间复杂度是$O(n^2)$。