3-2(D)

范潇　2254298

2024 年 4 月 24 日

题目 1. (最长单调子序列) 设计一个 $O(n\log n)$ 时间的算法，找出由 n 个数组成的序列的最长单调递增子序列。

解答. 在 $O(n^2)$ 时间的算法中，dp 数组用于记录以输入序列中的各个数为结尾的最长递增子序列，而在 $O(n\log n)$ 时间的算法中，dp 数组用于记录各个长度的递增子序列的末尾元素的最小值。

假设输入序列中的元素为互异的正数。可以证明，在扫描输入序列时，总是存在 i，使得 dp[1..i] 为非零单调递增序列，且 dp 中的其余元素均为 0。事实上，如果存在 $j<i$，使得 dp[j]>dp[i]，由于 dp[i] 对应的子序列中的第 j 个元素 x 小于 dp[i] 对应的元素，同时，由其前 j 个元素组成的子序列仍是单调递增子序列，因此产生矛盾。同时由于递增子序列的任意长度的子序列仍是递增子序列，所以 dp 中的非零元素是连续的。扫描完输入序列后，dp 中非零元素的个数便是最长递增子序列的长度。

为了维护 dp，扫描到的当前元素应该写入的位置是当前 dp 数组中第一个大于它的元素的位置，如果没有，则写入第一个 0 的位置（下标从 1 开始)。为了能够还原最长递增子序列，dp 数组每次更新时，更新位置对应的递增子序列以新增元素为结尾，且倒数第二个元素便是前面一个位置中的当前元素。因此只需要额外用一个数组记录这个关系即可。

伪代码在下一页中给出。其中在遍历输入序列时，使用了二分查找，所以时间复杂度为 $O(n\log n)$。

Algorithm 1: getLIS

Input: A<$x_1, \cdots, x_n$>($x_i > 0, i = 1, \cdots, n; x_i \neq x_j, i \neq j$)

Output: 最长递增子序列之一

初始化 dp// 用于记录各个长度的递增子序列的末尾元素的最小值以及在输入序列中的下标
初始化 pre// 用于回溯形成最长递增子序列
dp[0].val = -1// 哨兵
dp[0].no = -1
i = 1
while $i \leq n$ **do**
    pos = `bisect` (dp,A[i])
    // 当前 `dp` 数组中第一个大于它的元素的位置，若没有，则返回第一个 `0` 的位置
    dp[pos].val = A[i]
    dp[pos].no = i
    pre[i] = dp[pos-1].no
    i += 1
end while
l = 0
while *dp[l+1]≠0* **do** l += 1 // 求最长递增子序列长度
cnt = l
now = dp[l].no
初始化 LIS
while *now ≠ -1* **do**
    LIS[cnt] = A[now]
    cnt -= 1
    now = pre[now]
end while
return *LIS*