Week 5 PSO: Just For Fun

Mukul Agarwal

September 29, 2026

In this note, we provide an algorithm for the "Just-for-fun" extension from PSO for Longest Increasing[1] Subsequence; we achieve an algorithm which takes $O(n \log n)$ time (in fact, this runtime is optimal in the decision tree/ comparison model).

The key idea is to keep the smallest possible last element for each subsequence length, so that we can find the longest subsequence an element can extend using binary search.

Algorithm. We have the following algorithm.

LONGEST-INCREASING-SUBSEQUENCE($A[1..n]$):

/* Given an array A with elements belonging to $\mathbb{R}$, return the length of the longest nondecreasing subsequence of A. */

1. Create an array $T[1..n]$.
2. $L \leftarrow 0$
3. **for** $i = 1, ..., n$
   A. $E \leftarrow \min\left\{j \in [L] : T[j] > A[i]\right\}$ by Binary Search
      (if no such index exists, $E \leftarrow L + 1$)
   B. $T[E] \leftarrow A[i]$
   C. $L \leftarrow \max(L, E)$
4. return L

Running time. The i-th iteration takes $O(\log i)$ work. Summing across n iterations and accounting for $O(n)$ initialization work, we find that the overall runtime is $O(n \log n)$.

Proof of correctness. Let T_i, L_i, E_i denote the values of the corresponding variables at the end of iteration i. Thus T_0, L_0 are their initial values, and E_i is the index chosen during iteration i.

[1]Really, it's a nondecreasing sequence.

At the end of iteration i of the loop (where $i \in \{0, ..., n\}$)[2],

$(\mathsf{Inv1})_i$	L_i is the length of the longest nondecreasing subsequence of $A[1..i]$.
$(\mathsf{Inv2})_i$	For each $j \in [L_i]$, $T_i[j]$ is the smallest last element of any length-j nondecreasing subsequence of $A[1..i]$.
$(\mathsf{Inv3})_i$	$T_i[1..L_i]$ is initialized and sorted.

By $(\mathsf{Inv1})_n$, the returned value L_n is correct. Thus it suffices to prove the invariants.

Base Case ($i = 0$). The input prefix and active part of T_0 are empty, and $L_0 = 0$, so $(\mathsf{Inv1})_0$, $(\mathsf{Inv2})_0$, and $(\mathsf{Inv3})_0$ hold.

Inductive Case ($1 \le i \le n$). Assume the invariants at $i - 1$; we prove them at i. Write $x = A[i]$.

By $(\mathsf{Inv2})_{i-1}$, there is a length-j subsequence of $A[1..i-1]$ that can be extended by x exactly when $T_{i-1}[j] \le x$. Since these entries are sorted by $(\mathsf{Inv3})_{i-1}$, binary search finds that the largest such length is $E_i - 1$ (or zero if none exists). Thus the longest subsequence ending at $A[i]$ has length E_i. A longest subsequence of $A[1..i]$ either ends at $A[i]$, in which case its length is E_i, or ends earlier, in which case it is a longest subsequence of $A[1..i-1]$ and has length L_{i-1} by $(\mathsf{Inv1})_{i-1}$. Thus its length is $\max(L_{i-1}, E_i) = L_i$, proving $(\mathsf{Inv1})_i$.

Every newly available subsequence ends at x. For lengths below E_i, the old minimum is already at most x; for greater lengths, no new subsequence exists. At length E_i, x either improves the old minimum or creates the first subsequence of that length. Hence T_i contains exactly the minima required by $(\mathsf{Inv2})_i$.

Only that entry changes. Its left neighbor is at most x, and its right neighbor is greater than x (whenever these neighbors exist). Thus the updated prefix remains sorted and initialized, proving $(\mathsf{Inv3})_i$.

[2]Here $i = 0$ means "before the loop begins".