Problem

Given a sequence of numbers $x_1, x_2, \cdots, x_n$. Find the longest increasing subsequence in $O(n^2)$ time

Solution

Definitions. Let O_j be the longest increasing subsequence ending at x_j.
Let P_j be the second last number of some longest increasing subsequence ending at x_j. (Set to -1 if $O_j = 1$)

Algorithm.

- For $j = 1, 2, \cdots n$
  - $O_j = 1$. $P_j = -1$.
  - For all $i < j$ with $x_i < x_j$
    - If $O_i \geq O_j$,
      - $O_j = \max(O_j, 1 + O_i)$
      - $P_j = i$.
- Let $k = \arg\max_k O_k$
- Let path $= \{k\}$.
- While $P_k \neq -1$
  - $k \leftarrow P_k$.
  - path.push_front(k).
- Return path.

Runtime. For each j, the algorithm takes $O(n)$ time and hence the total time is $O(n^2)$.

Correctness. Let $x_{i_1}, x_{i_2}, \cdots, x_{i_k}, x_j$ be the longest increasing sequence ending at j. Then, $x_{i_1}, x_{i_2}, \cdots, x_{i_k}$ is the longest increasing sequence ending at i_k and that $i_k < j$ and $x_{i_k} < x_j$. Hence, we have

$$O_j = 1 + O_{i_k} \leq 1 + \max_{i: x_i < x_j, i < j} O_i.$$

On the other hand, $O_j \geq 1 + \max_{i: x_i < x_j, i < j} O_i$ because we can extend the longest subsequence ends at x_i by appending x_j at the end. Hence, we have $O_j = 1 + \max_{i: x_i < x_j, i < j} O_i$, which matches with what the algorithm is doing.