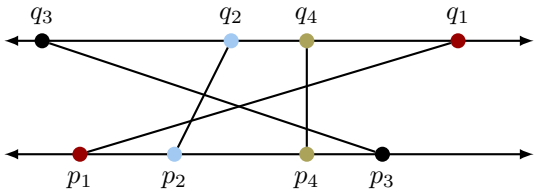
You are given two lists of n points, one list $P = [p_1, \ldots, p_n]$ lies on the line $y = 0$ and the other list $Q = [q_1, \ldots, q_n]$ lies on the line $y = 1$. We construct n line segments by connecting p_i to q_i for each $i = 1, \ldots, n$. You may assume that the numbers in P are distinct and the numbers in Q are also distinct. Describe an $O(n^2)$ algorithm to return the size of the largest subset L of line segments such that no pair of line segments in L intersect.

For example, the following instance



should return 2. The largest subset of pairwise-nonintersecting line segments is $\{(p_2, q_2), (p_4, q_4)\}$.

Note. *This is slightly different to the divide-and-conquer problem, which asked for the number of intersections of line segments. This problem asks for the size of the largest subset of* ***non-intersecting*** *line segments.*

There are at least two approaches.

Approach 1. *You can directly use dynamic programming...*

Approach 2. *Find a way to reduce this to the longest increasing subsequence problem.*

Solution. We first begin by sorting the points according to their starting points P, keeping track of their positions in Q during the sort. This sorting will take $O(n \log n)$. From here, we simply need to find the longest increasing subsequence in Q. A non-intersecting subset of segments corresponds to an increasing sequence on end points in Q as otherwise, there would exist an intersection point (where $p_i < p_j$ but $q_i > q_j$, there must be an intersection).

From here, the application is a standard usage of Dynamic Programming, where we make a list of n numbers DP, where $DP[i]$ represents the length of the LIS ending at q_i. For each i from 1 to n, we run for each j from 0 to $i - 1$, if $q_j < q_i$, we update $DP[i] = \max\{DP[i], DP[j] + 1\}$. The maximum value in DP will correspond to our desired answer. This is clearly done in $O(n^2)$ time.