Longest Monotone Subsequence

Problem: Given a sequence $\sigma = x[1], \ldots x[n]$, find the longest strictly monotone increasing subsequence of σ. (The Greek letter sigma.) To simplify our notation, we write "monotone" to mean strictly monotone increasing.

$\tau[t]$, the sequence $x[1], \ldots x[t]$. (The Greek letter tau.) Note that $\sigma = \tau[n]$.
$\upsilon[t]$, the longest monotone subsequence of $\tau[t]$. Defaults to 0. (The Greek letter upsilon, or ypsilon.)
$U[t]$, the length of $\upsilon[t]$
$j[t]$ the index of the last term of $\upsilon[t]$
$\varrho[t]$, the longest monotone subsequence of $\tau[t]$ ending at $x[t]$. (The Greek letter rho.)
$R[t]$, the length of $\varrho[t]$. Defaults to 0.
$p[t]$, a back pointer, the index of the second-to-last term of $\varrho[t]$. Defaults to 0.

Here is the code:

$x[0] = -\infty$
$U[0] = 0$
$R[0] = 0$
For all t from 1 to n
  {
    Pick $0 \leq i < t$ such that $R[i]$ is maximum subject to $x[i] < x[t]$
    $R[t] = R[i] + 1$
    $p[t] = i$
  }
if($R[t] > U[t-1]$)
  {
    $U[t] = R[t]$
    $j[t] = t$
  }
else
  {
    $U[t] = U[t-1]$
    $j[t] = j[t-1]$
  }

The longest monotone subsequence is found by following the back pointers starting at index $j[n]$, and has length $U[n]$. Line 6 of the code (starting with the word "Pick") dominates the time complexity. If the search is done linearly, the time complexity of the algorithm is $O(n^2)$. However, i can be found in $O(\log n)$ time using binary search, yielding time complexity $O(n \log n)$.