Longest Increasing Subsequence

input: A sequence $A[1..n]$ of numbers (or objects from any totally ordered set).

output: The length of the longest increasing subsequence of A. A subsequence $x = (x_1, x_2, \ldots, x_m)$ is *increasing* if $x_1 \leq x_2 \leq \cdots \leq x_m$.

Here is the algorithm. Fix some arbitrary input instance $A[1..n]$.

subproblems. For each $i \in \{0, 1, \ldots, n\}$ define $L[i]$ to be the longest increasing subsequence that ends with $A[i]$. The solution to the given problem is $\max_{i=1}^{n} L[i]$.

recurrence. $L[0] = 0$ and, for $j \geq 1$, $L[j] = \max\{1 + L[i] : i < j, A[j] \leq A[i]\}$.

The intuition is as follows. An increasing subsequence x that ends with $A[j]$ can be obtained by taking any increasing subsequence x' that ends at some $A[i]$ (where $i < j$ and $A[i] \leq A[j]$) then adding $A[j]$. (So x' is x with the last element $A[j]$ of x removed.) So

$$\begin{aligned} L[j] &= \max\{|x| : x \text{ is an increasing subsequence ending in } A[j]\} \\ &= \max\big\{ \max\{1 + |x'| : x' \text{ is an incr. subs. ending in } A[i]\} : i < j, A[i] \leq A[j]\big\} \\ &= \max\{1 + L[i] : i < j, A[i] \leq A[j]\big\}. \end{aligned}$$

time. There are $n + 1$ subproblems, and for each, the right-hand side of the recurrence can be evaluated in $O(n)$ time. So the total time is $O(n^2)$.

correctness. To show correctness, we would use the ideas sketched above to carefully prove that the recurrence relation is correct. Correctness of the algorithm would follow by induction (on i) over the subproblems.

This algorithm is equivalent to running the Longest Common Subsequence algorithm on the pair (A, B) where B is A in sorted order.

External resources on Longest Increasing Subsequence

- Dasgupta et al. Chapter 6
- Jeff Edmond's Lecture 5
  http://jeffe.cs.illinois.edu/teaching/algorithms/notes/05-dynprog.pdf