Longest Increasing Subsequence

Wednesday, 30 August 2023 11:21 AM

I/P: Array of n distinct integers

$A = (a_1, a_2, \ldots, a_n)$

A subsequence S is a sorted array of indices $1 \ldots n$

O/P: Length of longest subsequence S of $1 \ldots n$ so that, if $i < j \in S$, then $a_i < a_j$

Eg. 18 3 6 1 14 8 17 [10 15 2 3 4 22 23 16 31]

3 components for dynamic programming:

① Optimal substructure: i.e., optimal solution can be found by first obtaining soln. to an independent smaller problem.

$LIS(r, i)$ = length of longest increasing subsequence whose first elt. is a_r & remaining elts. are for $a_i, a_{i+1}, \ldots, a_n$

(for $r \geq i$, $LIS(r, i) = LIS(r, r+1)$)

if we include a_i / if we exclude a_i

$$LIS(r, i) = \begin{cases} \text{if } a_r < a_i : \max\{1 + LIS(i, i+1),\ LIS(r, i+1)\} \\ \text{if } a_r > a_i : LIS(r, i+1) \quad \text{can't include } a_i \end{cases}$$

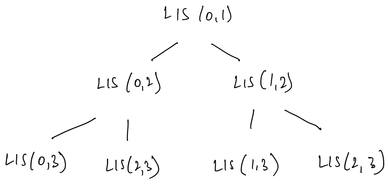
$(r < i)$

② Recursion

Assume $a_0 = -\infty$

```
LIS(r,i)        \\ assume r < i
{ if (a_r < a_i)
      return max{ 1 + LIS(i,i+1), LIS(r,i+1) }
  else
      return LIS(r,i+1)
}
```

Call LIS(0,1)

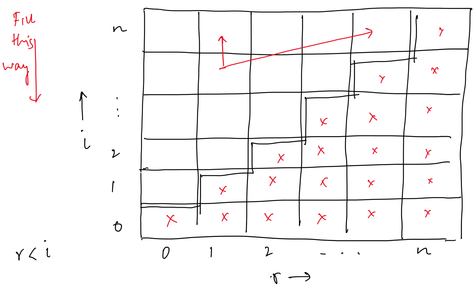


Basically LIS(0,1) calls LIS(2,3) twice

LIS(2,3) calls LIS(4,5) twice...

So LIS(n, n+1) gets called $2^{n/2}$ times!

③ Table-Filling: Do it inductively, instead of recursively.



Easy to see this takes time $O(n^2)$

PROBLEM 1: Write down algorithm yourself.

PROBLEM 2: $O(n \log n)$ time algorithm?