CS314 - Longest Increasing Subsequence

Note Title 3/1/2010

Announcements

- HW due ~~Monday~~ Tuesday after break - slide under my door before 4pm.
- Midterm is Friday after break (in-class)
- Look for sample midterm early in break. (Will post to website.)

Longest Increasing Subsequence (LIS)

Input: $A[1..n]$ (integers)

Goal: Find longest sequence of indices $1 \le i_1 < i_2 < \dots < i_k \le n$ s.t. $A[i_j] \le A[i_{j+1}]$ $\forall j$.

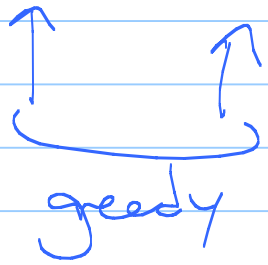
Ex: 3 5 2 1 6 10 9 4 -2 8

LIS: 3, 5, 6, 10

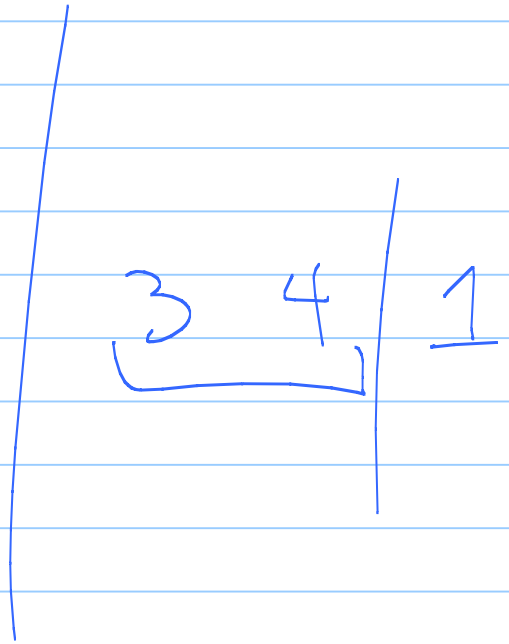
indices: 1, 2, 5, 6

Side note - why doesn't greedy work?

Ex: 1 4 2 3



LIS: 1, 2, 3



Recursive dfn of subsequence:

(Base case) - Subsequence of empty sequence is the empty sequence.

- Subsequence of $A[1..n]$ is either:
  - a subsequence of $A[2..n]$
  - or $A[1]$ followed by subsequence of $A[2..n]$

either $A[1]$ is in subsequence,
or it isn't.

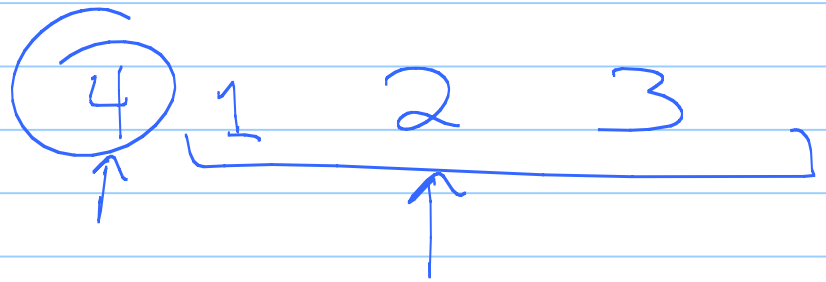
LIS (recursively)

An LIS of $A[1..n]$ is either

- LIS of $A[2..n]$ (no $A[1]$)

or

- $A[1]$ followed by LIS of $A[2..n]$ where all values are $> A[1]$



So- incorporate idea of computing LIS where everything is $> x$ (for some x).

A[1] is too small

If $A[1]$ is $< x$, then just compute LIS of $A[2..n]$ where all values are $> x$.

If $A[1]$ is $> x$, might include $A[1]$ or might not, so take max of:

- LIS of $A[2..n]$ w/ values $> x$
- 1 + LIS of $A[2..n]$ w/ values $> A[1]$.

↑ take $A[1]$

Pseudocode (in lecture notes on Recursion)

```
LIS(A[1..n]):
    return LISBIGGER(−∞, A[1..n])
```

```
LISBIGGER(prev, A[1..n]):
    if n = 0
        return 0
    else
        max ← LISBIGGER(prev, A[2..n])
        if A[1] > prev
            L ← 1 + LISBIGGER(A[1], A[2..n])
            if L > max
                max ← L
        return max
```

skip A[1]

Take A[1]

Runtime? $T(n) = 2T(n-1) + O(1)$

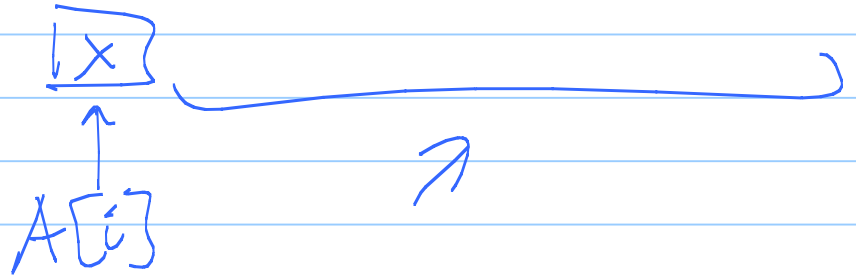
$= O(2^n)$

Smart recursion

We are computing LIS of array $A[1..n]$ with elements larger than x.

What are the possible values for x?

x will always be $A[i]$ for some i



Let $L(i,j)$ = length of LIS of $A[j..n]$ with elements larger than $A[i]$.

For any $i < j$,

$$L(i,j) = \begin{cases} 0 & \text{if } j > n \\ L(i,j+1) & \text{if } A[i] \geq A[j] \\ \max\{L(i,j+1),\ 1 + L(j,j+1)\} & \text{otherwise} \end{cases}$$

$A[j]$ is too small!

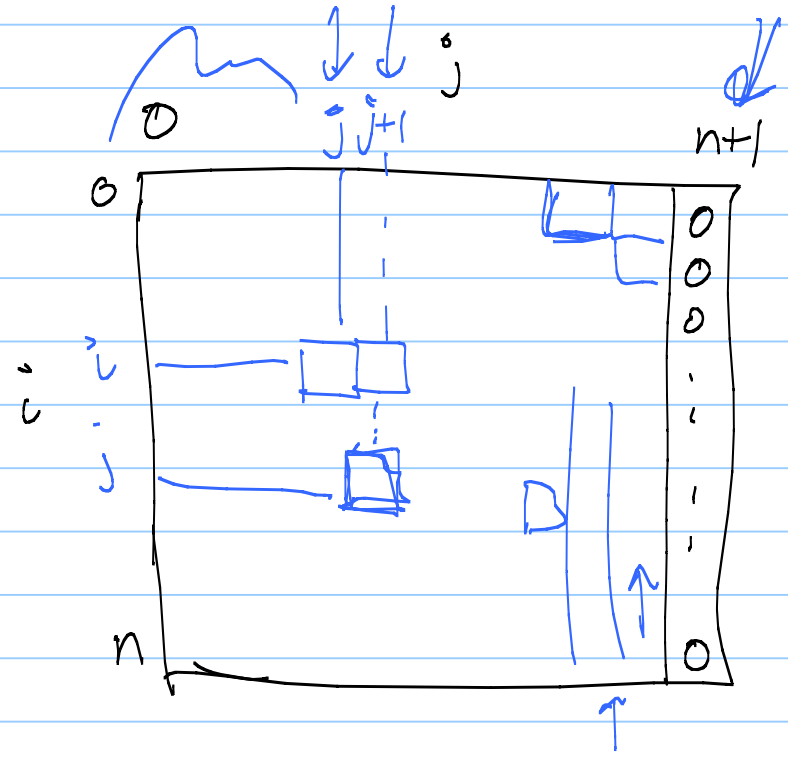
Could include $A[j]$ or not

So think of this as n×n table.

To fill in $L(i,j)$, which entries do we need?

$L(i,j+1)$ –

$L(j,j+1)$ – $i<j$



Pseudocode

```
LIS(A[1..n]):
  A[0] ← −∞                          ⟨⟨Add a sentinel⟩⟩
  for i ← 0 to n                     ⟨⟨Base cases⟩⟩
      L[i, n+1] ← 0

  for j ← n downto 1
      for i ← 0 to j − 1
          if A[i] ≥ A[j]
              L[i, j] ← L[i, j+1]
          else
              L[i, j] ← max{L[i, j+1], 1 + L[j, j+1]}

  return L[0, 1] − 1                 ⟨⟨Don't count the sentinel⟩⟩
```

O(1)

$$\sum_{j=1}^{n} \sum_{i=0}^{j-1} O(1)$$

Runtime?

$$\sum_{j=1}^{n} \sum_{i=0}^{j-1} 1 = \sum_{j=1}^{n} (j-1) = 0+1+2+\cdots+n-1$$

$$= \sum_{j=0}^{n-1} j = \Theta(n^2)$$

$O(n^2)$

Space?

$O(n^2)$ – lookup table

Improving the space

Do we need the whole table?

$$L(i,j) = \begin{cases} 0 & \text{if } j > n \\ L(i, j+1) & \text{if } A[i] \geq A[j] \\ \max\{L(i, j+1),\ 1 + L(j, j+1)\} & \text{otherwise} \end{cases}$$

No - only need prev. column $(j+1)$ (plus current column j).

$O(2n) = O(n)$