CS 314 - Longest Increasing Subsequence

Announcements

- HW due Monday 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 $1 \leq i_1 < i_2 < \ldots < i_k \leq n$ s.t. $A[i_j] < A[i_{j+1}] \quad \forall j$.

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

3 4 1
Recursive dfn of subsequence:

- **Base case**: Subsequence of empty sequence is the empty sequence.
- Subsequence of $A[1..n]$ is either:
- either $A[i]$ is in subsequence or it isn't.
LIS (recursively)

An LIS of \( A[1..n] \) is either


or

- \( A[i] \) followed by LIS of \( A[2..n] \)

where all values are \( > A[i,j] \)
So incorporate idea of computing LIS where everything is $>x$ (for some $x$).

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

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

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

Take $A[i]$
Pseudo code

\[ \text{LIS}(A[1..n]) : \]
\[ \text{return LISbigger}(\infty, A[1..n]) \]

\[ \text{LISbigger}(\text{prev}, A[1..n]) : \]
\[ \text{if } n = 0 \]
\[ \text{return 0} \]
\[ \text{else} \]
\[ \text{max} \leftarrow \text{LISbigger}(\text{prev}, A[2..n]) \]
\[ \text{if } A[1] > \text{prev} \]
\[ L \leftarrow 1 + \text{LISbigger}(A[1], A[2..n]) \]
\[ \text{if } L > \text{max} \]
\[ \text{max} \leftarrow L \]
\[ \text{return max} \]

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 $i$.

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

For any \( i < j \),

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

Could include \( A[j] \) or not

\( A[j] \) is too small
So think of this as an nxn table.

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

$L(i, j+1)$

$L(j, j+1)$ - $i_j$
Pseudocode

\[
\text{LIS}(A[1..n]):
\]
\[
A[0] \leftarrow -\infty \quad \langle \text{Add a sentinel} \rangle
\]
\[
\text{for } i \leftarrow 0 \text{ to } n \quad \langle \text{Base cases} \rangle
\]
\[
L[i, n+1] \leftarrow 0
\]
\[
\text{for } j \leftarrow n \text{ downto } 1
\]
\[
\quad \text{for } i \leftarrow 0 \text{ to } j - 1
\]
\[
\quad \quad \begin{cases} 
0 & \text{if } A[i] \geq A[j] \\
1 & \text{else}
\end{cases}
\]
\[
\quad L[i, j] \leftarrow L[i, j + 1]
\]
\[
\quad L[i, j] \leftarrow \max\{L[i, j + 1], 1 + L[j, j + 1]\}
\]
\[
\text{return } L[0, 1] - 1 \quad \langle \text{Don’t count the sentinel} \rangle
\]

\[
\sum_{i=0}^{n} j^{-1} \leq O(1)
\]

\[
\sum_{j=1}^{n} i = \frac{n(n+1)}{2}
\]
Runtime? \( O(n^2) \)

\[
\sum_{j=1}^{n} \sum_{i=0}^{j-1} = \sum_{i=0}^{\frac{n^2}{2}} = O\left(1+2+\cdots+n-1\right) = O\left(\sum_{i=0}^{n-1} i\right) = \Theta(n^2)
\]

Space? \( O(n^2) \) — look-up 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 \(j\)).

\(O(2n) = O(n)\)