Advanced DP Topics

LIS: Longest Increasing Subsequence

\[ x_1, \ldots, x_n \]

NaiveDP:

\[
LIS(i) = \begin{cases} 
LIS \text{ ending at } i 
\end{cases}
\]

\[
LIS(i) = 1 + \max \{ LIS(j) \mid j < i \land x_j < x_i \}
\]

Answer = \max \{ LIS(i) \}

Time: \(O(n^2)\)

Fancier:

Lowest \( (i, k) \) = smallest last value of length-\( k \) \text{ IS ending before } i.

\[
\text{Lowest (i, k) = min} \left( \begin{array}{c}
\text{Lowest (i-1, k)} \\
x_i, \text{ if length } k \text{ sequence ending at } i
\end{array} \right)
\]
Homework:

\[ \min \text{ cost interval cover} \]

Sort by finish time.
Build from left.

\[ \begin{array}{c}
\text{cost} \\
\vdots \\
\text{amount covered}
\end{array} \]

\[ \begin{array}{c}
\text{Cost } [i_j] \\
\text{Cover } [i_j]
\end{array} \]

For new \((s, t, c)\):

Let \( j = \min \{j \mid s \leq i_j \leq t, \text{cover } i_j \geq s \} \).

New option \((\text{cost} + [i_j]c, \leq t + c)\).

Add to list (undominated since \(+ \) bigger).
But first, pop until \(\text{cost} + [i_j]c \leq \text{cost} + [i_j]c\).
Sliding Window + Finding Set

Recall Knapsack:

$$\text{Val}(i, v) = \max \left( \text{Val}(i-1, v), \text{Val}(i-1, v-s_i) + v_i \right)$$

$O(nC)$ time
$O(C)$ space if only track 2 columns

How to find set $S$?

- Normally, Back pointers.
- But: that's $O(nC)$ space.
- Just need the path. $O(n)$.

Idea: only keep back pointers into column $n/2$.

$$[\text{create at } i = n/2; \text{just copy over}]$$

$O(nC)$ time to get 1 point on path.

$c_{n/2} = \text{cap used after first } n/2$.

Recurse:

Knapsack ($\frac{n}{2}$ items, $0 \to n/2$, $c_{n/2}$)
Knapsack ($\frac{n}{2}$ items, $n/2 \to n$, $c$, $C - c_{n/2}$)

will get whole path $(Q^n \to (n, C))$

How long:

$$T(n, C) = nC + T\left(\frac{n}{2}, c_{n/2}\right) + T\left(\frac{n}{2}, C - c_{n/2}\right)$$
Time in next round = \( \frac{nC}{2} \).

\[ nC + \frac{nC}{2} + \frac{nC}{4} + \ldots = O(nC). \]