Max Weight Talk Scheduling

In HW0 we saw the problem of scheduling the max # of talks that were not overlapping. Now we see how to solve a weighted version of that.

<table>
<thead>
<tr>
<th>Input:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Collection of intervals of the Real line</td>
</tr>
<tr>
<td>Each interval i has start $s_i$, end $t_i$, weight $w_i$</td>
</tr>
<tr>
<td>Two intervals i and j conflict if $(s_i, t_i) \cap (s_j, t_j) \neq \emptyset$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Goal:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pick a collection of intervals that do not conflict and have largest total weight</td>
</tr>
</tbody>
</table>

HW0: greedy algorithm is optimal if all weights are same
Exercise: greedy fails if weights are not identical.

Solution for general case via DP:

Sort by right endpoint, so $t_1 \leq t_2 \leq t_3 \leq \ldots \leq t_n$

For interval i, let $p(i)$ be last interval ending by time $s_i$

$$T(i) = \max \left\{ T(j), w_i + T(p(i)) \right\}$$

$$T(0) = 0$$
Two ways to write this \((\text{assume computed } p(i) \neq i)\)

1. **Bottom up / Table filling**

   \[ T(0) = 0 \]
   
   **for** \(i = 1\) **to** \(n\) **do** :
   
   \[ T(i) = \max (T(i-1), w_i + T(p(i))) \]

   \(O(n)\) time

2. **Top down / Memorization**

   \[ T(i) \neq \]
   
   **if** \(i = 0\) **return** 0

   **if** memo\((i)\) **undefined**

   \[ \text{memo}(i) \leftarrow \max (T(i-1), w_i + T(p(i))) \]

   **return** memo\((i)\)

**Note**: each time call \(T(i)\) recursively, fill in one entry of memo table

\(\Rightarrow\) total \# of calls \(\leq n \times 2\)

Each such call takes \(O(1)\) time \(\Rightarrow O(n)\) total time

---

Finally, compute \(p(i)\) for all \(i\):

1. **Sort** \(t_i\) **times** \((\text{take } O(n \log n)\) time\)

2. For each \(s_i\), do binary search to find largest \(t_i \leq s_i\). \(\{\text{takes } O(n)\) time per search\)

\(\Rightarrow\) Overall \(O(n \log n)\) time