1 Activities Selection

Given: \( a = [s_1, f_1], [s_2, f_2], \ldots, [s_n, f_n] \)

1. Select the largest set of non-overlapping activities.
2. Observation: OK to start with activity ending first.
3. Then, take the activity whose starting time is closer to the finishing time on first activity.

**Recursive (already sorted with finishing time)**

\[
\text{Rec-As}(s, f, i) \begin{cases} i \text{ is initially 0} \\
1 & m \leftarrow i + 1 \\
2 & \text{while } m \leq n \land s_m < f_i \text{ do} \\
3 & \quad m \leftarrow m + 1 \\
4 & \quad \text{if } m \leq n \text{ then} \\
5 & \quad \quad \text{return } a_m \cup \text{Rec-As}(s, f, m) \\
6 & \quad \text{else} \\
7 & \quad \quad \text{return } \emptyset
\end{cases}
\]

**Iterative**

\[
\text{Top-Down-As}(s, f) \\
\begin{array}{l}
1 & i \leftarrow 1 \\
2 & \text{for } m = 2 \text{ to } n \text{ do} \\
3 & \quad \text{if } s_m \geq f_i \text{ then} \\
4 & \quad \quad A = A \cup a_m \\
5 & \quad \quad i = m \\
6 & \text{return } A
\end{array}
\]

- Greedy is often Dynamic Programming with 1 known subproblem
- Iterative version of Greedy Algorithm is top-down
□ **Greedy Stays Ahead:** Show (by induction) \( \forall i \), the first \( i \) "steps" of the greedy algorithm are "better" than the first \( i \) "steps" of any other algorithms (including an optimal algorithm)

A. Do it by induction

B. Since it is true \( \forall i \) to the last step \( \Rightarrow \) greedy algorithm is optimal

C. Tricky part: Defining "step" and "better"

With Activities Selection: \( \forall i, \forall \text{schedules } Z \), define:

\[ F_i(Z) = \text{finishing time of } i^{th} \text{ scheduled activity in } Z \]

**Inductive Claim:** \( \forall i \forall Z, F_i(\text{greedy}) \leq F_i(Z) \)

**Proof:** Induction on \( i, i = 1 \) (just the definition of greedy algorithm)

Assume true for \( i = m \):

<table>
<thead>
<tr>
<th>Greedy</th>
<th>#1</th>
<th>#2</th>
<th>...</th>
<th>#m</th>
<th>#m+1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Z</td>
<td>#1</td>
<td>#2</td>
<td>...</td>
<td>#m</td>
<td>#m+1</td>
</tr>
</tbody>
</table>

\[ f_m \leq f^Z_m \leq s^Z_{m+1} \]

So, \( f_{m+1} \leq f^Z_{m+1} \)

□ Corollary: \( |\text{Greedy}| \leq |Z| \)

□ **Local Swap:** Argue that it is always safe to include (one) greedy choice in the optimal solution.

(*) \( \forall OPT \), construct \( OPT' \) that is not worse than \( OPT \) which has one greedy choice (and reduces to the same subproblem as greedy).

□ **Key Step:** Argue subproblem of \( OPT' = \text{subproblem of greedy} \)

□ Essentially, it is induction on the size of activity selection.
2 Huffman Codes

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
<th>e</th>
<th>f</th>
</tr>
</thead>
<tbody>
<tr>
<td>%</td>
<td>45</td>
<td>13</td>
<td>12</td>
<td>16</td>
<td>9</td>
<td>5</td>
</tr>
<tr>
<td>fixed</td>
<td>000</td>
<td>001</td>
<td>010</td>
<td>011</td>
<td>100</td>
<td>101</td>
</tr>
<tr>
<td>opt</td>
<td>0</td>
<td>101</td>
<td>100</td>
<td>111</td>
<td>1101</td>
<td>1100</td>
</tr>
</tbody>
</table>

- Prefix-Free Encoding:
  \[ \forall c_1, c_2, \text{encoding of } c_1 \text{ named } E(c_1) \text{ is not prefix of } E(c_2) \]
  \[ \text{cost}(E) = \sum f(c') |E(c')| \]

  \[ \text{cost(fixed)} = 3 \]

  \[ \text{cost OPT} = 2.24 \]

- Correctness
  
  Claim 1: \( \text{OPT} \) never has nodes of degree 1 (one child)
  
  Claim 2: Let \( T_1, T_2 \) be solid trees which are identical except for 2 swapped frequencies:

  \[ T_1: \quad l_1 \quad l_2 \quad T_2: \quad l_1 \quad l_2 \]

  \[ B \quad A \quad A \quad B \]

  \[ l_1 \geq l_2 \land A \leq B \]

  \[ \text{cost}(T_2) \leq \text{cost}(T_1) \]

  
  Corollary: If we sort frequencies in ascending order, then optimal lengths will decrease.