sorted in incr. order
of $f_i$

main problem: input $\mathcal{S}$, $a_1, a_2, \ldots, a_k, \ldots, a_n$ denote $\mathcal{P}$

(Set of $n$ activities)

optimal solution: $\mathcal{S}^\ast$ denote $\mathcal{S}^\ast$

note $a_1$ here

Want to show that a sub-solution of $\mathcal{S}^\ast$ is optimal for a sub-problem of $\mathcal{P}$

What should be the sub-problem / sub-solution?

Can we do what we did for Fractional KS and remove some activity, $a_k$, from both $\mathcal{P}$ and $\mathcal{S}^\ast$?

No! Why not?

$\mathcal{P}$:

$$\begin{array}{ccc}
    a_1 & a_2 & a_3 & \cdots & a_{k-1} & a_k & a_{k+1} & \cdots & a_n \\
    a_2 & a_3 & \cdots & a_k & \cdots & a_n \\
    a_3 & \cdots & a_k & \cdots & a_n \\
    \vdots & \ddots & \vdots & \ddots & \ddots & \vdots & \ddots & \ddots & \ddots \\
    a_k & \cdots & a_{k-1} & a_k & a_{k+1} & \cdots & a_{k+m} & \cdots & a_n \\
    a_k & \cdots & a_{k-1} & \cdots & a_n \\
    a_{k+1} & \cdots & a_{k+m} & \cdots & a_n \\
    \vdots & \ddots & \vdots & \ddots & \ddots & \vdots & \ddots & \ddots & \ddots \\
    a_n & \cdots & a_2 & a_3 & \cdots & a_k & a_{k+1} & \cdots & a_n \\
\end{array}$$

see next page for concrete example.

$\mathcal{S}^\ast$:

$$\begin{array}{ccc}
    a_1 & a_2 & a_3 & \cdots & a_{k-1} & a_k & a_{k+1} & \cdots & a_n \\
    a_2 & a_3 & \cdots & a_k & \cdots & a_n \\
    a_3 & \cdots & a_k & \cdots & a_n \\
    \vdots & \ddots & \vdots & \ddots & \ddots & \vdots & \ddots & \ddots & \ddots \\
    a_k & \cdots & a_{k-1} & a_k & a_{k+1} & \cdots & a_{k+m} & \cdots & a_n \\
    a_k & \cdots & a_{k-1} & \cdots & a_n \\
    a_{k+1} & \cdots & a_{k+m} & \cdots & a_n \\
    \vdots & \ddots & \vdots & \ddots & \ddots & \vdots & \ddots & \ddots & \ddots \\
    a_n & \cdots & a_2 & a_3 & \cdots & a_k & a_{k+1} & \cdots & a_n \\
\end{array}$$

new solution may include $a_k'$

Cannot simply remove some activity $a_k$ from both $\mathcal{P}$ and $\mathcal{S}^\ast$.

$\mathcal{S}^\ast$ may exclude some activity $a_k$ that overlaps with $a_k$.

But optimal solution to $\mathcal{P} - a_k$ may include $a_k$. 

Remove from $S^*$ (first activity) $a_1$.
"Remove from $P$ $a_1$ and any activity that overlaps $a_1"$
(Could also remove $S^*$ $a_k$ and remove from $P$ $a_k$ and
any activity that overlaps $a_k$)

$P: \quad a_k \quad \quad \Rightarrow \text{sub-problem}$

$S^*: \quad \quad \quad \quad \Rightarrow \text{sub-solution}$

Remove from $P$ any activity $a_i$ s.t. $s_i < f_i$ | $s_k < f_i < f_k$ or
(including $a_1$)

sub-solution: $S^* - a_1$
sub-problem: $P - a_i$ for all $a_i$ s.t. $s_i < f_i$ ⇒ shorten to "P - $a_i"

Claim: $S^* - a_1$ is optimal solution to $P - a_i$ for all $a_i$ s.t. $s_i < f_i$

Proof: Suppose hore that $S^* - a_1$ is not optimal.
Then there must be another solution $S'$ for $P - a_i$
that contains more activities than $S^* - a_1$
Example of why it's incorrect to remove any one activity:

\[ P: \begin{array}{ccc}
a_1 & a_2 & a_3 \\ 0 & 22 & 44 \\
\end{array} \quad S^*: \begin{array}{ccc}
a_1 & a_2 & a_3 \\ 0 & 44 & 6 \\
\end{array} \]

Remove \( a_3 \) from \( P \) and \( S^* \):

New \( P: \begin{array}{cc}
a_1 & a_2 \\
\end{array} \quad \) New \( S^*: \begin{array}{cc}
a_1 & a_2 \\
\end{array} \)

But \( S^* \) is not optimal for new \( P \)!
Will eventually show that this false assumption implies that $S^*$ was not optimal for $P$

What does $S'$ look like?

\[ \begin{array}{c}
\text{2} \\
\text{start} \geq f_i \\
\end{array} \]

$S'$:

\[ \begin{array}{c}
\text{1} \\
\text{a} \\
\uparrow \\
\text{additional activity} \\
\end{array} \]

1. What else do we know about $S'$?
   \[ \downarrow \downarrow \downarrow \downarrow \]
   • If $S'$ is optimal for $P-a_i$ then $S'$ cannot contain any activity that overlaps with $a_i$.
   • So start time of first activity in $S'$ is $\geq f_i$.

3. How can we modify $S'$ to show a contradiction to our original assumption that $S^*$ is optimal for $P$?

5. Can add $a_i$ to $S'$ (we can do this b/c no activities in $S'$ overlap $a_i$) to get a better solution than $S^*$ for $P$.
   • So $S^*$ not optimal for $P$, contradiction!
\[ S^* - a_i \text{ is optimal for } P - a_i \]

\[ \text{GreedyActSel has o.s.p.} \]
Huffman's Encoding

Algorithm for data compression.

Pictures, songs, videos, text files all stored as binary strings.

Example: text \rightarrow \text{ASCII} \rightarrow \text{binary}

picture? \rightarrow each pixel has 3 ASCII codes (R, G, B) \rightarrow binary

Simplest way - just store as long binary string (8 bits for ASCII code).

Better: "use" data compression - encode information using fewer bits than original.

We will focus on text files but the algorithm applies to all file types.
<table>
<thead>
<tr>
<th>char</th>
<th>freq</th>
<th>code</th>
<th>#bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>10</td>
<td>000</td>
<td>30</td>
</tr>
<tr>
<td>e</td>
<td>15</td>
<td>001</td>
<td>45</td>
</tr>
<tr>
<td>i</td>
<td>12</td>
<td>010</td>
<td>36</td>
</tr>
<tr>
<td>s</td>
<td>3</td>
<td>011</td>
<td>9</td>
</tr>
<tr>
<td>t</td>
<td>4</td>
<td>100</td>
<td>12</td>
</tr>
<tr>
<td>sp</td>
<td>13</td>
<td>101</td>
<td>39</td>
</tr>
<tr>
<td>nl</td>
<td>1</td>
<td>110</td>
<td>3</td>
</tr>
</tbody>
</table>

174 ≤ Can we do better?

1. (Skip ASCII) will show: text → binary

   **Simplest way:**
   - fixed-length encoding - codes have same length.
   - For n character types, need \( \lceil \log_2(n) \rceil - \lceil \log_2(3) \rceil = 3 \) bits per code
   - (Need \( \log_2(n) \) bits to encode value n in binary)

4. **Goal**: Given alphabet A of n character types, find an encoding that uses the fewest number of total bits.

   Helps to visualize with binary tree (Natural)
   - characters at leaves
   - every left branch: 0
   - "right": 1
"Make large enough for 7 chars"

Can easily decode

\[
011010\overline{100}
\]

Number of bits in terms of tree?

\[
\#\text{bits} = \text{tree "cost"} = \sum_{c \in \mathcal{A}} d_c \times f_c
\]

Cheaper tree?

\* Can use variable length encoding - code lengths differ

Yes!

\[
\begin{align*}
\text{nl} & = 11 \\
\text{new cost} & = 173
\end{align*}
\]
How to use variable length encoding to get cheaper tree?

- Assign less freq chars longer codes
  "more" "shorter"

Not clear how.
What should be shortest/longest?
What if there is a tie?
Less obvious problem

Let's try: e: 0 sp: 1 i: 00

Problem? Decode 000: 000 e i i e e e

ambiguous encoding!

We want: unambiguous encoding - unique decoding for every encoded string.
characters at leaves only!

How to ensure unambiguous encoding? characters at leaves only

New goal: find cheapest tree that yields unambiguous encoding.

Huffman's tree: start with single-node tree for each char, cost = freq