Activity Selection Problem

- Set \( S = \{ a_1, a_2, \ldots, a_n \} \) of \( n \) activities
  - all want to use same resource (lecture hall)
    - can be used by only one activity at a time

- activity \( a_i \):
  - start time: \( s_i \)
  - finish time: \( f_i \): \( 0 \leq s_i < f_i < \alpha \)
  - if selected take place: \([ s_i, f_i)\)

- \( a_i, a_j \): compatible if \( [s_i, f_i) \cap [s_j, f_j) = \emptyset \) (no overlap)
  \[ \iff s_i \geq f_j \text{ or } s_j \geq f_i \]

- Want (ASP): maximum size set of mutually compatible activities.

\[
\begin{array}{l|cccccccccc}
  & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \\
  s_i & 1 & 3 & 0 & 5 & 3 & 5 & 6 & 8 & 8 & 2 \\
  f_i & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 \\
\end{array}
\]

\( \text{(note)} \)

- \( \{ a_3, a_9, a_{11} \} \) mutually compatible
- \( \{ a_1, a_4, a_8, a_{11} \} \), \( \{ a_2, a_4, a_9, a_{11} \} \)
First formulate a DP solution

- combine optimal solution to 2 subproblems to form optimal solution to original problem.
- consider a few choices to find which 2 subproblems to use
- but need consider only one choice (greedy choice)
  - after greedy choice, one subproblem is empty!

Optimal substructure

- define sets $S_{ij} = \{a_k \in S : f_i \leq s_k < f_k \leq s_j\}$
  - set of activities in $S$ that start after $a_i$ finishes and finish before $a_j$ starts.
  - compatible with all activities that finish (before $a_i$) no later than $a_i$ and all that start no earlier than $a_j$.
- to represent entire problem, add fictitious $a_0, a_{n+1}$
  - $f_0 = 0$, $s_{n+1} = \infty$
  - $S = S_0, n+1$
  - $0 \leq i, j \leq n+1$
Assume activities sorted $A$ by finish time:

$$f_0 < f_1 < f_2 < \cdots < f_n < f_{n+1}$$

$$\Rightarrow S_{ij} = \emptyset \text{ if } i > j$$

- Space of subproblems: $S_{ij}$, $0 \leq i < j \leq n + 1$
  - select maximum-size subset of compatible activities from $S_{ij}$
  - assume solution uses $A_k$
  - using $A_k \in S_{ij}$ generates 2 subproblems:
    - $S_{ik}$, $S_{kj}$
    - each is subset of activities in $S_{ij}$

<table>
<thead>
<tr>
<th>$A_i$</th>
<th>$S_{ik}$</th>
<th>$S_{kj}$</th>
<th>$E_{aj}$</th>
</tr>
</thead>
</table>

- Solution to $S_{ij}$: union of solutions to $S_{ik}, S_{kj}, A_k$
- # of activities: $|A_{ik}| + |A_{kj}| + 1$
- Optimality of subproblems follows from "standard" proof-by-contradiction

$$A_{ij} = A_{ik} \cup \{A_k\} \cup A_{kj}$$

Optimal solution to overall problem $\Rightarrow$ solution to $S_0, n+1$
Recursive Solution (second step of DP)

- recursively define the value of an optimal solution

- \( C[i,j] \) : # activities in \( A_{i,j} \) [max-size subset of compatible activities in \( S_{i,j} \)]
  - \( C[i,j] = 0 \) if \( i > j \)

  \[ C[i,j] = C[i,k] + C[k,j] + 1 \]

  - but we don't know the value of \( k \)

  \[ C[i,j] = \begin{cases} 0, & \text{if } S_{i,j} = \emptyset \\ \max \{ C[i,k] + C[k,j] + 1 \}, & \text{if } S_{i,j} \neq \emptyset \end{cases} \]

Compute an optimal solution by DP \( \rightarrow \) bottom-up. Easy

However:

Th: Let \( S_{i,j} \neq \emptyset \), \( a_m \in S_{i,j} \) with earliest finish time:

\[ f_m = \min \{ f_k \mid a_k \in S_{i,j} \} \]

Then:

1. \( a_m \) is used in "some" \( A_{i,j} \)
2. \( S_{i,m} \) is empty \( \rightarrow \) choosing \( a_m \) leaves only \( S_{m,j} \) as a possible non-empty subproblem
Proof:

2. By contradiction: if $S_{in}$ contains some $a_k$, $a_k$ has earlier finish time than $a_m$

1. Order activities in $A_{ij}$ by finish time.
   Let $a_k$ be first activity in sorted $A_{ij}$.
   - if $a_k = a_m$ \( \rightarrow \) done
   - if $a_k \neq a_m$ consider subset $A'_{ij} = A_{ij} \setminus \{a_k\} \cup \{a_m\}$
     - activities in $A'_{ij}$ are disjoint since $f_m \leq f_k$
     - $|A'_{ij}| = |A_{ij}| \Rightarrow A'_{ij}$ also optimal for $S_{ij}$

\[\Rightarrow\]\[
\begin{align*}
\text{• only one subproblem used in an optimal solution} \\
\text{• when solving $S_{ij}$, need consider only one choice:}
\end{align*}
\]
- one with earliest finish time in $S_{ij}$

\[\downarrow\]
- can solve $S_{0,n+1}$ in a top-down way
- to solve $S_{ij}$, first choose $a_m$ then
- solve $S_{mj} \Rightarrow S_{i',n+1}$

\[\Rightarrow\] always pick activity with earliest finish time

\[\rightarrow\] greedy choice
Greedy Activity Selector \( (s, f) \)

\[ n \leftarrow \text{length}[S] \]

\[ A \leftarrow \{a_1\} \]

\[ i \leftarrow 1 \quad \text{index of last activity selected} \]

\[
\text{for } m = 2 \text{ to } n \text{ do}
\]

\[
\text{if } s_m \geq f_i \text{ then}
\]

\[
A \leftarrow A \cup \{a_m\}
\]

\[
i \leftarrow m
\]

\[ \text{return } A \]

- \( \Theta(n) \) time if activities already sorted by finish time.
- **Greedy choice**: maximize the amount of unused time remaining.
- After greedy choice: problem reduces to similar one over activities compatible with \( A_1 \):
  - if \( A \) optimal solution for \( S \) then
    \[
    A' = A - \{a_1\} \text{ optimal for } S' = \{a \in S : s_i \geq f_1\}
    \]
    \[
    \text{Same problem but of smaller size}
    \]
"Classic" examples of Greedy algorithms

- Minimum Spanning Tree (MST)
- Single Source Shortest Path (Dijkstra’s)

**MST**

- $G(V, E)$: connected, weighted, undirected graph
  - $(u, v) \in E \rightarrow$ weight $w(u, v) (\geq 0)$
- Find acyclic subset $T \subseteq E$ that connects all vertices in $V$ such that:
  $$\text{w}(T) = \sum_{(u,v) \in T} w(u,v) \text{ is minimized}$$

*Note: $T$: acyclic connects all vertices $\rightarrow$ tree $\rightarrow$ spanning tree

- Will discuss 2 greedy algorithms for finding MST:
  - Kruskal: $O(|E| \log |V|)$ using binary heap
  - Prim: $O(|E| + |V| \log |V|)$ using Fibonacci heaps
*Note: both greedy strategies lead to optimal solution (MST)*
Generic MST algorithm

- grows a spanning tree by adding one edge at a time.
- captures the greedy strategy (greedy choice).
- maintains a set \( A \) that is always a subset of some MST (MST is not unique in general)
- at each step, find edge \((u, v)\) and add to \( A \), without changing the invariant:
  \[ A \cup \{ (u, v) \} \text{ is a subset of some MST} \]

\((u, v)\) is called a safe edge

\[
\text{Generic-MST}(G, w) \]

\[
\begin{align*}
A & \leftarrow \emptyset \\
\text{while } & \text{ A is not a spanning tree of } G \text{ do} \\
& \text{ find edge } (u, v) \text{ safe for } A \\
& \quad A \leftarrow A \cup \{ (u, v) \} \\
\text{return } & A
\end{align*}
\]

Difficulty: find a safe edge.

- one must exist:
  \[ A \subseteq T \]
  \[ \text{if } \exists (u, v) \in T, (u, v) \notin A \]
  \[ \Rightarrow (u, v) \text{ safe for } A \]
Need rule to recognize safe edges

Definitions:

- A cut \((S, V-S)\) of \(G=(V,E)\) is a partition of \(V\)

\((u,v)\in E\) crosses the cut if
\(u\in S, v \in (V-S)\)

A cut \((S, V-S)\) respects \(A\) (set of edges) if no edge in \(A\) crosses the cut.

Edge \((u,v)\) crossing the cut is called light edge if
\(w(u,v) = \min \{ w(x,y) \mid (x,y) \text{ crosses the cut} \}\)

Note: not unique! (see fig. above without \((d,c)\))

Theorem: Let \(A \subseteq E\) included in some MST of \(G\).
Let \((S, V-S)\) be any cut of \(G\) that respects \(A\).
Let \((u,v)\) be a light edge crossing \((S, V-S)\).
Then \((u,v)\) is safe for \(A\).
**Proof:** Let \( T \) be a MST, \( A \subset T \). Assume \((u,v) \notin T\).

We shall construct another MST, \( T' \) s.t.
\[(A \cup \{(u,v)\}) \subset T' \Rightarrow (u,v) \text{ safe for } A.\]

- \((u,v) \) not in \( T \); consider path \( P(u,v) \) in \( T \), from \( u \) to \( v \).
- \((u,v) \) forms a cycle with edges on \( P(u,v) \).

\[u \xrightarrow{} \{x,y\} \xrightarrow{} v \]

- \( u, v \) on opposite sides of cut \( (S, V - S) \)

\[\Rightarrow \exists \text{ edge } (x, y) \in T \text{ crossing } (S, V - S)\]

- \((x, y) \in A \) since \((S, V - S) \) respects \( A \)

- removing \((x, y) \) breaks \( T \) in 2 components; adding \((u,v) \) reconnects them \( \Rightarrow \) new spanning tree \( T' \):

\[T' = T - \{(x,y)\} \cup \{(u,v)\}\]

- \( w(u,v) \leq w(x,y) \) \( \left( \text{ both cross } (S, V - S), \right. \)

\[\left( (u,v) \text{ is light} \right)\]

\[\Rightarrow w(T') = w(T) - w(x,y) + w(u,v) \leq w^*(T)\]

- but \( T \) is MST \( \Rightarrow w(T') = w(T) = \) \( T' \) is MST

- Remains to show \((u,v) \) safe for \( A \)
- \( A \subset T' \) since \( A \subset T \) and \((x,y) \notin A\)

\[\Rightarrow A \cup \{(u,v)\} \subseteq T' \Rightarrow T' \text{ MST } \Rightarrow (u,v) \text{ safe} \]
Note:
- A always acyclic
- at any step of the algorithm, graph $G_A = (V, A)$ is a forest.
  - each connected component of $G_A$ is a tree
  - at first, $n$ components, each a 1 node tree.
- a safe edge $(u, v)$ connects 2 distinct components of $G_A$
- the loop of Generic MST
  - executed $n-1$ times (MST has $n-1$ edges)
  - when $A = \emptyset \Rightarrow n$ trees in $G_A$
  - each iteration reduces # trees by 1.
  - terminates when forest has 1 tree
- MST algorithms use the following:

Corollary: Let $A \subseteq E$ included in some MST of $G$.
Let $C$ be a connected component (tree) in $G_A = (V, A)$. If $(u, v)$ is a light edge connecting $C$ to some other tree of $G_A$, then $(u, v)$ is safe for $A$.

Proof: cut $(C, V - C)$ respects $A$, $(u, v)$ light edge for it $\Rightarrow (u, v)$ safe.