Activity Selection Problem

- Set $S = \{a_1, a_2, \dots, a_n\}$ 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 : $0 \le s_i < f_i < \infty$
    → 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)
  $\iff s_i \geq f_j$ or $s_j \geq f_i$
- want (ASP) : maximum size set of mutually compatible activities.

E.g.: S

i	1	2	3	4	5	6	7	8	9	10	11
s_i	1	3	0	5	3	5	6	8	8	2	12
f_i	4	5	6	7	8	9	10	11	12	13	14

(note) ↗ f_i

$\{a_3, a_9, a_{11}\}$ 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 $\mapsto$ greedy choice;
  - after greedy choice, one subproblem is empty!

Optimal substructure

- Define sets $S_{ij} = \{a_k \in S : f_i \le s_k < f_k \le s_j\}$
  - 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}

  $f_0 = 0$, $s_{n+1} = \infty$

  $\rightarrow S = S_{0,n+1}$

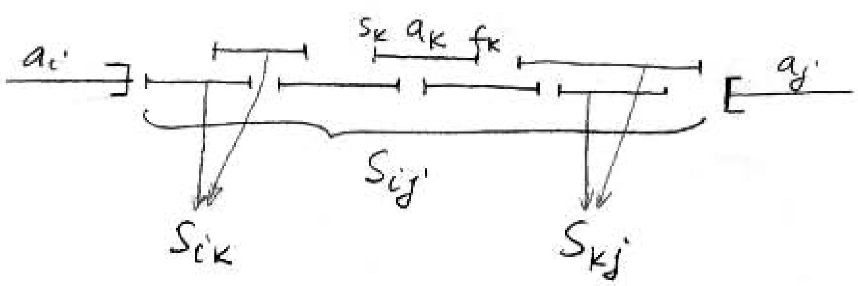
  $\rightarrow 0 \le i, j \le n+1$

Assume activities sorted ↑ by finish time:

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

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

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



- 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 → 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_{ij} (max-size subset of compatible activities in S_{ij})
  - $c[i,j] = 0$ if $i > j$

$\Rightarrow c[i,j] = c[i,k] + c[k,j] + 1$

- but we don't know the value of k

$$\Rightarrow c[i,j] = \begin{cases} 0 & , \text{ if } S_{ij} = \emptyset \\ \max\limits_{i<k<j} \{ c[i,k] + c[k,j] + 1 \}, & \text{if } S_{ij} \neq \emptyset \end{cases}$$

Compute an optimal solution by DP $\Rightarrow$ bottom-up.
$\Rightarrow$ easy

However:

Th: Let $S_{ij} \neq \emptyset$, $a_m \in S_{ij}$ with earliest finish time:

$f_m = \min \{ f_k \mid a_k \in S_{ij} \}$. Then:

1. a_m is used in "some" A_{ij}
2. S_{im} is empty $\mapsto$ choosing a_m leaves only S_{mj} as a possible non-empty (problem) subproblem

Proof:

2. By contradiction: if S_{im} 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 $A'_{ij} = A_{ij} - \{a_k\} \cup \{a_m\}$
  - activities in A'_{ij} are disjoint since $f_m \leq f_k$
  - $|A'_{ij}| = |A_{ij}| \Rightarrow A'_{ij}$ also optimal for S_{ij}.

$\Rightarrow$
- only one subproblem used in an optimal solution
- when solving S_{ij}, need consider only one choice: one with earliest finish time in S_{ij}.

$\Downarrow$

$\rightarrow$ can solve $S_{0,n+1}$ in a top-down way

$\rightarrow$ to solve S_{ij}, first choose a_m then solve $S_{mj} \Rightarrow \boxed{S_{i,n+1}}$

$\Rightarrow$ always pick activity with earliest finish time

$\rightarrow$ $\boxed{\text{greedy choice}}$

```
Greedy_Activity_Selector(s, f)
    n ← length[s]
    A ← {a_1}
    i ← 1                        | index of last activity selected

    for m ← 2 to n do
        if s_m ≥ f_i then        | f_i: max finishing time of
            A ← A ∪ {a_m}        |      any activity in A:
            i ← m                |      f_i = max{f_k : k ∈ A}
    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

    $A' = A - \{a_1\}$ optimal for $S' = \{a_i \in S : s_i \geq f_1\}$

    $\Downarrow$

    same problem but of smaller size

"Clasic" examples of Greedy algorithms

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

MST

- $G(V,E)$: connected, weighted, undirected graph.
  $(u,v) \in E \mapsto$ weight $w(u,v)$ (≥ 0 ?!)
- Find acyclic subset $T \subseteq E$ that connects all vertices in V such that:

$$w(T) = \sum_{(u,v)\in T} w(u,v) \text{ is minimized}$$

Note: $\left.\begin{array}{l} T: \text{acyclic} \\ \text{connects all vertices} \end{array}\right\} \Rightarrow$ tree $\rightarrow$ spanning tree

- Will discuss 2 greedy algorithms for finding MST:
  - Kruskal : $O(|E| \log |V|)$ using $<$ binary heap / disjoint set operations
  - 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:

  $A \cup \{(u, v)\}$ is a subset of some MST

  - (u, v) is called safe edge

```
Generic_MST(G, w)
    A ← ∅
    while A is not a spanning tree of G do
        find edge (u,v) safe for A
        A ← A ∪ {(u,v)}
    return A
```

Difficulty: find a safe edge.

- one must exist:

  $A \subseteq T$

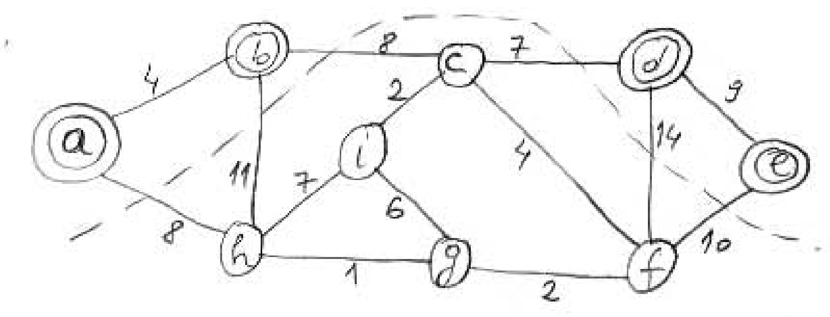
  if $\exists\, (u, v) \in T,\ (u, v) \notin A$

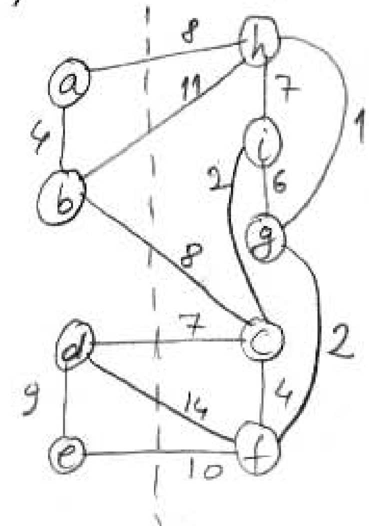
  $\Rightarrow (u, v)$ safe for A

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))

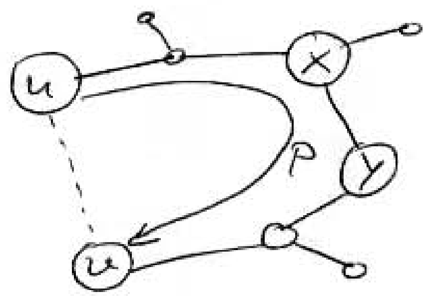
Theorem: Let $A \subset E$ included in some MST of G.
Let $(S, V-S)$ be any cut of G that respects A.
Let (u,v) be a light edge crossing $(S, V-S)$.
Then (u,v) is safe for A.

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

  $\Rightarrow \exists$ edge $(x,y) \in T$ crossing $(S, V-S)$
- $(x,y) \notin A$ since $(S, V-S)$ <u>respects</u> A
- removing (x,y) breaks T in 2 components; adding (u,v) reconnects them $\Rightarrow$ new <u>spanning tree</u> T':

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

  - $w(u,v) \le w(x,y)$ (both cross $(S, V-S)$, (u,v) is light)

  $\Rightarrow w(T') = w(T) - w(x,y) + w(u,v) \le w(T)$

  - but T is MST $\Rightarrow w(T') = w(T) \Rightarrow \boxed{T' \text{ is MST}}$

- <u>Remains</u> to show (u,v) <u>safe</u> for A
  - $A \subseteq T'$ since $A \subseteq T$ and $(x,y) \notin A$

  $\Rightarrow A \cup \{(u,v)\} \subseteq T'$, T' MST $\Rightarrow (u,v)$ <u>safe</u>

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 $A = \phi \mapsto n$ 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 $G_A = (V, A)$
forest

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.