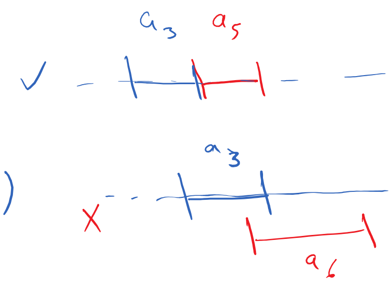
Greedy Algorithms

Problem: Activity Selection

Set of activities $S = \{a_1, \ldots a_n\}$



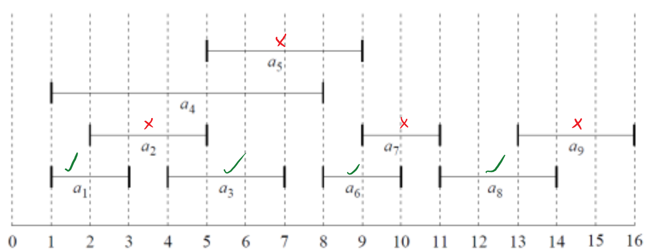
each activity a_i has period $[s_i, f_i)$

Goal: select the largest subset of S containing nonoverlapping activities

number of activities maximized — mutually compatible

Let's sort our activities: $f_1 \leq f_2 \ldots \leq f_n$

i	1	2	3	4	5	6	7	8	9
s_i	1	2	4	1	5	8	9	11	13
f_i	3	5	7	8	9	10	11	14	16



$\{a_1, a_3, a_6, a_8\}$ optimal

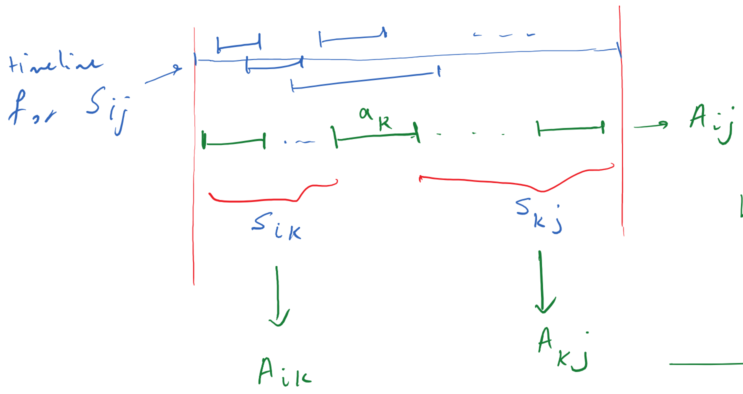
$\{a_2, a_5, a_7, a_9\}$ optimal

$S_{ij} = \{a_k \in S : f_i \leq s_k, f_k \leq s_j\}$ activities that start after a_i finishes and before a_j starts

Is S_{ij} compatible with a_i?

↳ S_{ij} is compatible with $\begin{cases} \text{all activities that finish by } f_i \\ \text{all activities that start no earlier than } s_j \end{cases}$

Let's assume A_{ij} be a max-size set of nonoverlapping activities within S_{ij}



Let $a_k \in A_{ij}$

$\longrightarrow$ finding these optimal subsets are our subproblems

$$\begin{cases} A_{ik} = A_{ij} \cap S_{ik} \\ A_{kj} = A_{ij} \cap S_{kj} \end{cases} \Longrightarrow A_{ij} = A_{ik} \cup \{a_k\} \cup A_{kj}$$

$$|A_{ij}| = |A_{ik}| + |A_{kj}| + 1$$

<u>Claim</u>: Optimal solution A_{ij} must include optimal solutions for S_{ik} and S_{kj}.

<u>recursive solutions</u>

$c[i,j]$: size of optimal solution for S_{ij}

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

which a_k? $\longrightarrow$ try all potential a_ks

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

$\longrightarrow$ dynamic programming

<u>Greedy strategy</u>:

- Can we choose activity part of optimal set before solving the subproblems?

- greedy choice in this problems pick first activity (activities are ordered by finish time)

- if S_k notation: $S_k = \{a_i \in S : s_i \geq f_k\}$ (activities that start no earlier simplify S_{ij} notation: $S_k = \{ a_i \in S : s_i \geq f_k \}$ (activities that start no earlier than when a_k finishes)

if we choose a_1 greedily $\Longrightarrow$ need to optimize S_1

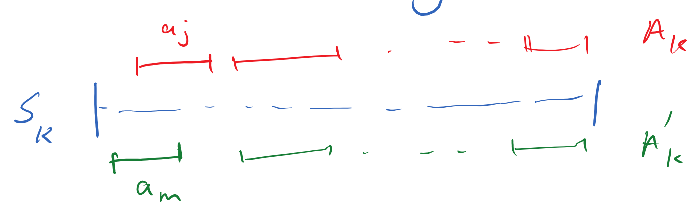
<u>theorem</u>: if S_k is non-empty and a_m has the earliest finish time in S_k, then a_m is included in some optimal solution.

proof: Let A_k be optimal solution, and a_j have the earliest finish time

$$\begin{cases} a_j = a_m \longrightarrow \text{Done} \\ a_j \neq a_m \longrightarrow \text{Let } A'_k = A_k \setminus \{a_j\} \cup \{a_m\} \end{cases}$$

Note: $|A'_k| = |A_k|$

Show A'_k is non-overlapping. ✓



initially call$(s, f, 0, n)$

indicate current subproblem S_k

```
REC-ACTIVITY-SELECTOR(s, f, k, n)
  m = k + 1
  while m ≤ n and s[m] < f[k]          // find the first activity in S_k to finish
      m = m + 1
  if m ≤ n
      return {a_m} ∪ REC-ACTIVITY-SELECTOR(s, f, m, n)
  else return ∅
```

complexity? $\Theta(n)$

iterative solution

```
GREEDY-ACTIVITY-SELECTOR(s, f)
```

iterative Solution

```
GREEDY-ACTIVITY-SELECTOR(s, f)
  n = s.length
  A = {a_1}
  k = 1
  for m = 2 to n
      if s[m] ≥ f[k]
          A = A ∪ {a_m}
          k = m
  return A
```

Dynamic Programming vs. Greedy

first solve subproblems, then choose

first choose then solve subproblems

Solve Bottom-Up

Solve Top-Down