Greedy Algorithm - I

Activity Selection Problem

Instructor: Ashok Singh Sairam

Lecture Plan

- Greedy algorithm definition
- Activity selection problem
  - DP solution
  - Greedy solution

Optimization problems

- Algorithms for optimization problems
  - Sequence of steps, with a set of choices at each step
- Types of algorithms we can consider
  - Brute force algorithms
  - Simple recursive algorithms
  - Divide and conquer algorithms
  - Dynamic programming algorithms
  - Greedy algorithms
  - Branch and bound algorithms
  - Randomized algorithms

Greedy Algorithms

- A greedy algorithm works in steps. At each step:
  - You take the best choice, without regard for future consequences
  - You hope that by choosing a *local* optimum at each step, you will end up at a *global* optimum

Activity-selection problem

- Consider a set of activities S={a_1, a_2,…, a_n}
  - They use resources, such as lecture hall, one lecture at a time
  - Each a_i, has a start time s_i, and finish time f_i, with $0 \leq s_i < f_i < \infty$.
  - a_i and a_j are compatible if $[s_i, f_i)$ and $[s_j, f_j)$ do not overlap
- Goal: select maximum-size subset of mutually compatible activities.
- Start from dynamic programming, then greedy algorithm, see the relation between the two

Activity-selection problem

- The activity-selection problem is to select a maximum-size subset of mutually compatible activities.

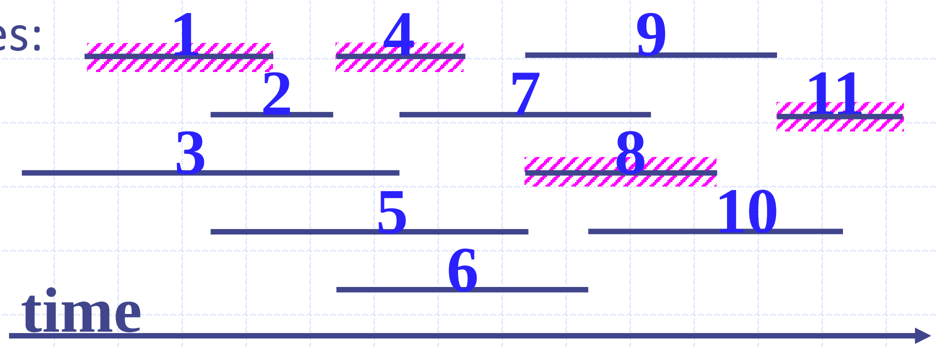
Example:

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	9	9	10	11	12	14	16

Some mutually compatible activities:

$\{a_3, a_9, a_{11}\}$ --- not the largest

$\{a_1, a_4, a_8, a_{11}\}$



DP solution: Notations

- Optimal substructure of activity-selection problem

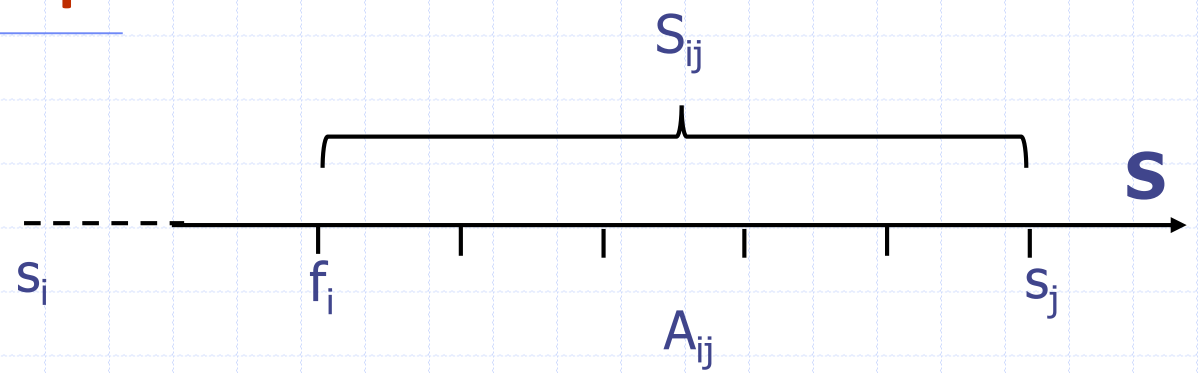
  $\boldsymbol{S_{ij}}=\{a_k \in S : f_i \leq s_k < f_k \leq s_j\}$,

  is the subset of activities in S that can start after activity a_i finishes and finish before activity a_j starts.

- Problem:
  - Find maximum set of mutually compatible activities in S_{ij}

$\boldsymbol{A_{ij}}$: Maximum set of mutually compatible activities in S_{ij}

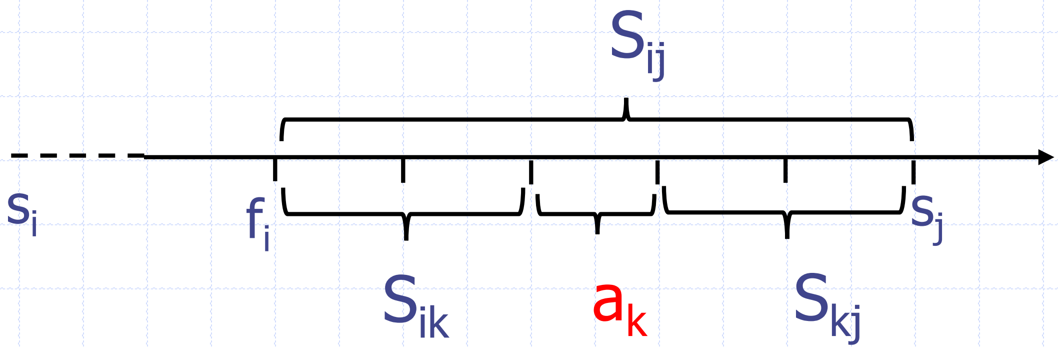
Example



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	9	9	10	11	12	14	16

- $S_{1,11}$= ??
- $A_{1,11}$= ??

Illustration:
DP solution – Optimal Substructure



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

DP solution –step 1

- Optimal substructure
  - Suppose an optimal solution A_{ij} includes a_k
  - We are left with two subproblems - S_{ik} and S_{kj}
- Let $A_{ik} = A_{ij} \cap S_{ik}$
  - Contain the activities in A_{ij} that finish before a_k
- Let $A_{kj} = A_{ij} \cap S_{kj}$
  - Contain the activities in A_{ij} that start after a_k
- $A_{ij} = A_{ik} \cup \{a_k\} \cup A_{kj}$

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

DP Solution – step 2

- Recursive cost

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

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

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

- Need to solve all sub-problems

Greedy Choice

- Choose an activity to add to the optimal solution without having to solve all the subproblems
- What is a greedy choice?
  - One possibility: Choose an activity such that it will leave maximum resources available for others
    - Choose an activity a_1 with the earliest finish time
- Making a greedy choice leaves us with only **one** subproblem to choose
  - Find activities that start after a_1

Greedy choice (2)

- Set of activities that start after a_k finishes

  $\boldsymbol{S_k} = \{a_i \in S: s_i \geq f_k\}$

  - Ex: If a_1 is greedy choice, only subproblem is S_1

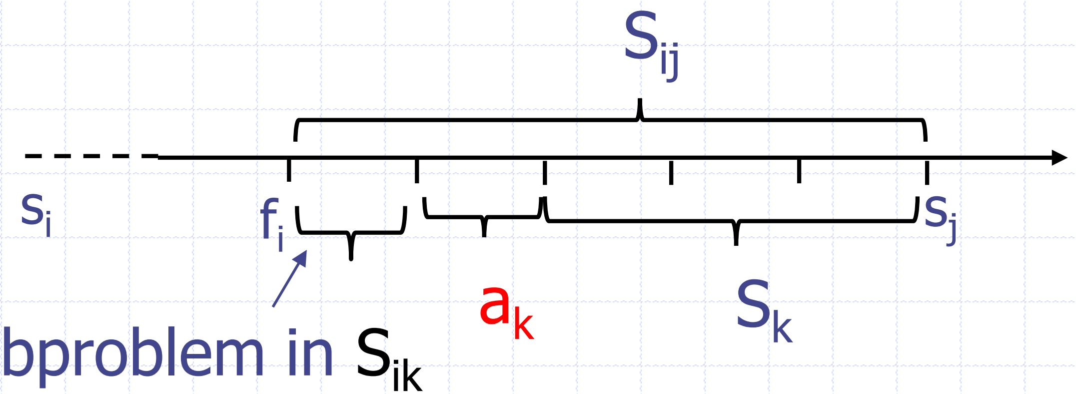
  $\boldsymbol{A_k}$ is optimal solution of S_k

- We have shown that Activity-Selection problem exhibit optimal substructure
  - That is , if a_1 is in the optimal solution, then

    $$\mathrm{A_{ij}} = a_1 \cup A_1$$

- We need to prove a_1 in the optimal solution

Illustration: Greedy choice



$$A_{ij} = \{a_k\} \cup A_k$$

a_k: Activity with earliest finish time

S_k: Activities that start after a_k finishes

Illustration of Theorem

Theorem 16.1

Consider any nonempty subproblem S_k, and let a_m be an activity in S_k with the earliest finish time. Then a_m is included in some maximum-size subset of mutually compatible activities of S_k.

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	9	9	10	11	12	14	16

- Consider S_0 [all activities that start after a_0, f_0=0]
  - The activity with earliest finish time is a_1
  - a_1 must be included in some optimal solution
- Optimal solutions: $\{a_1, a_4, a_8, a_{11}\}$, $\{a_2, a_4, a_9, a_{11}\}$

Recursive Greedy Algorithm

- Assume that *n* input activities are already ordered by monotonically increasing finish times

$$f_1 \leq f_2 \leq \ldots \leq f_n$$

- Add fictitious activities a_0 with f_0=0
  - Subproblem S_0 is the entire set of activities

Recursive Greedy Algorithm

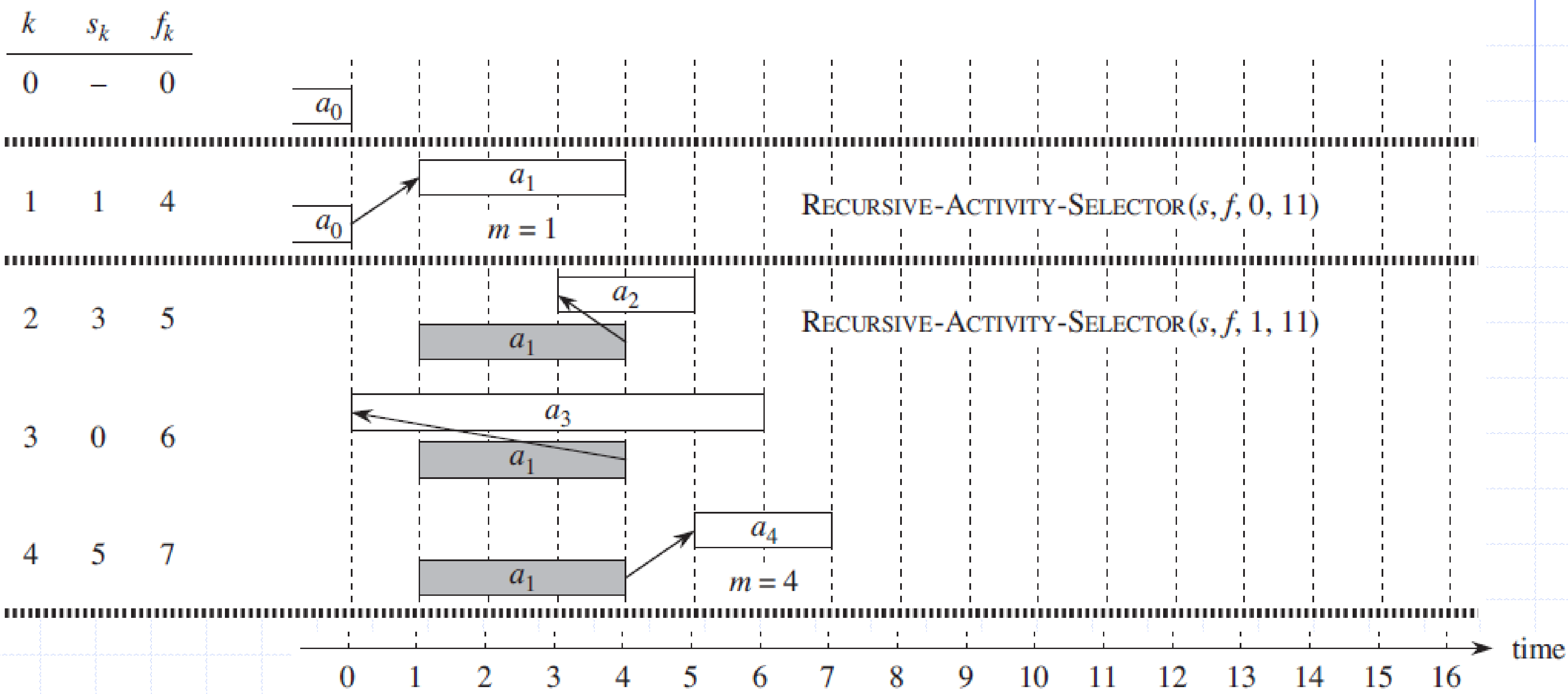
- Input: s[]: start time of a_i; f[]: finish time of a_i
- Initial call: Recursive-Activity-Selector(s,f,0,n)

```
RECURSIVE-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} ∪ RECURSIVE-ACTIVITY-SELECTOR(s, f, m, n)
else return Ø
```

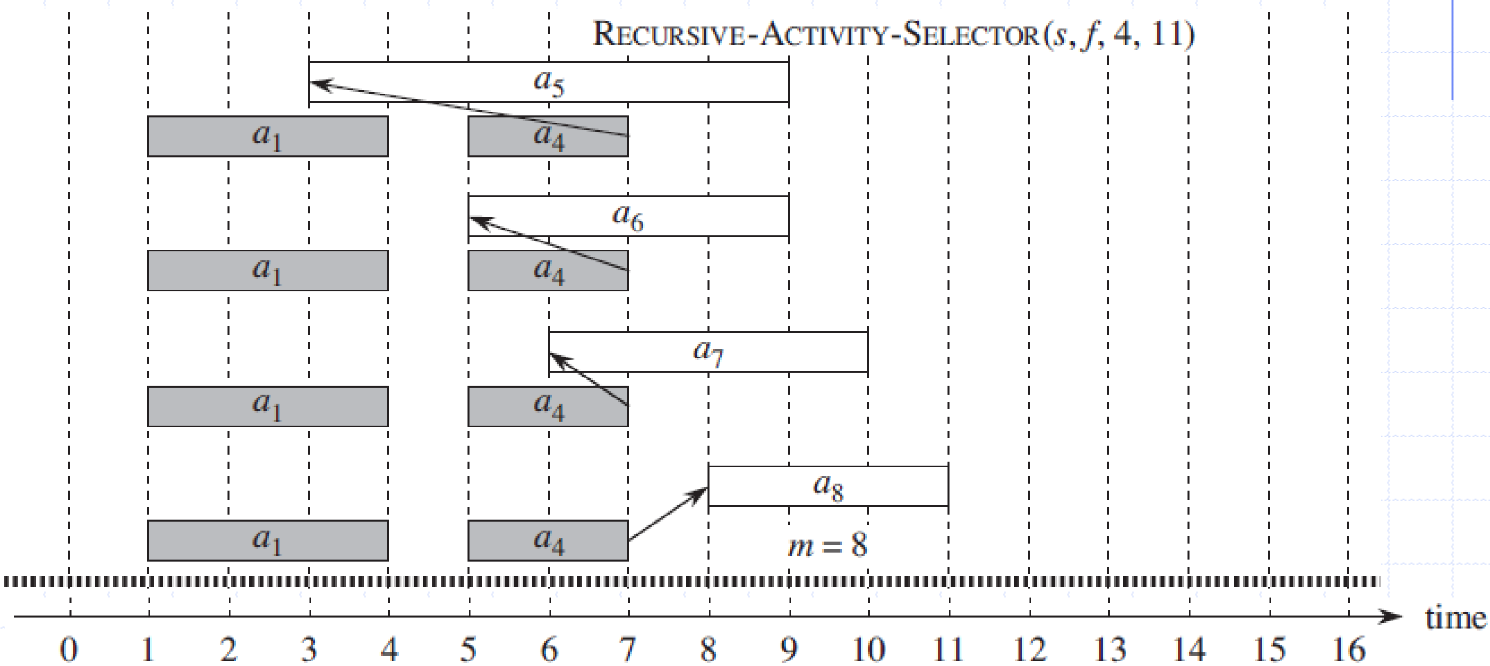
- While loop returns first activity a_m compatible with a_k

Running time: $\Theta(n)$

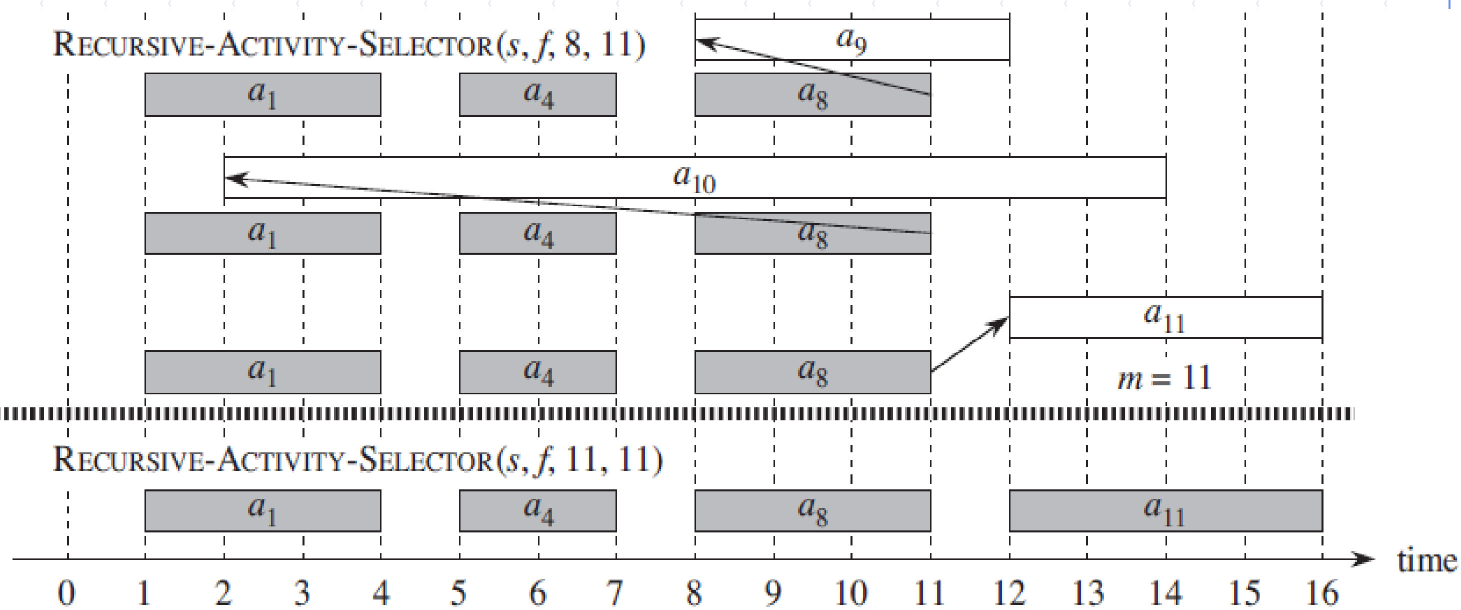
RECURSIVE-ACTIVITY-SELECTOR:
Illustration on the 11 activities given earlier



k	s_k	f_k
5	3	9
6	5	9
7	6	10
8	8	11



k	s_k	f_k
9	8	12
10	2	14
11	12	16



Iterative version: Greedy Algo

- Write an iterative version of the Greedy algorithm (recursive) given below.

```
RECURSIVE-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} ∪ RECURSIVE-ACTIVITY-SELECTOR(s, f, m, n)
else return ∅
```

Exercise

16.1-1

Give a dynamic-programming algorithm for the activity-selection problem, based on recurrence (16.2). Have your algorithm compute the sizes $c[i, j]$ as defined above and also produce the maximum-size subset of mutually compatible activities.