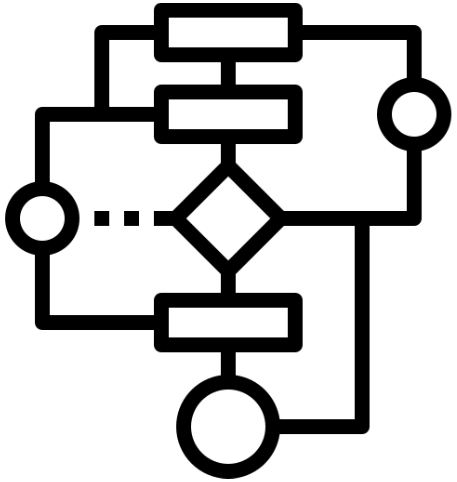


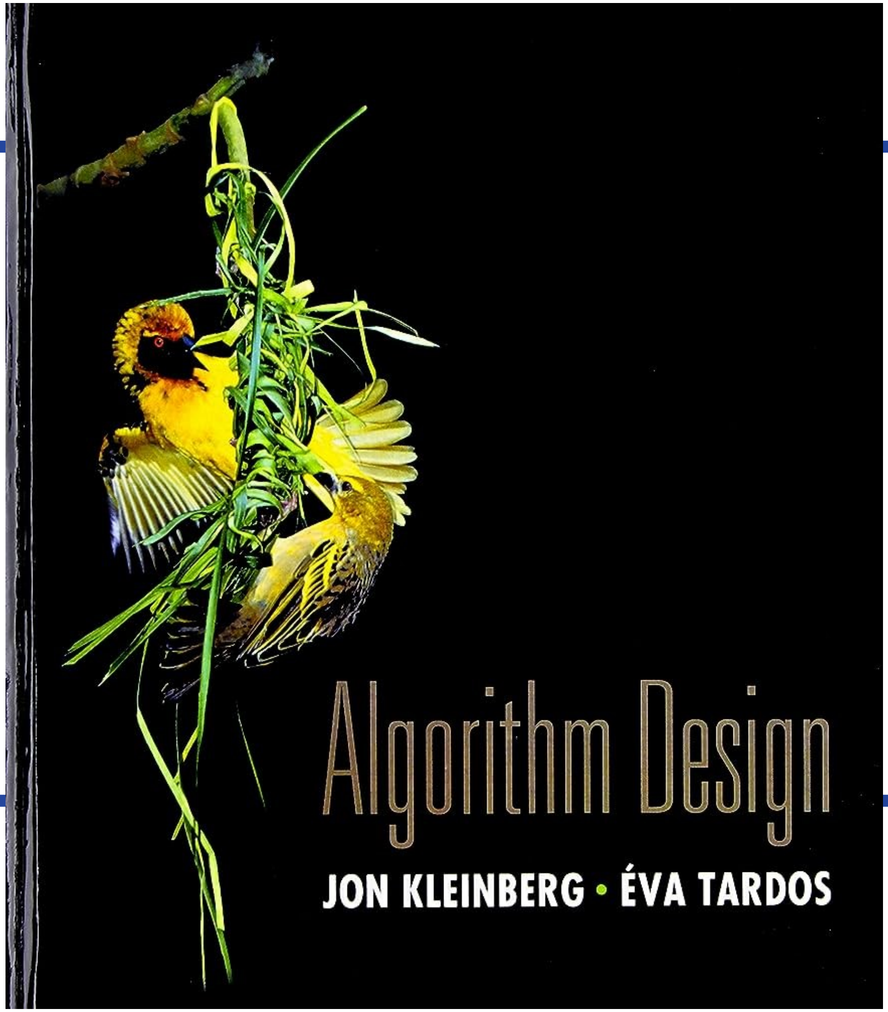
CS 310: Algorithms

Lecture 14

Instructor: Naveed Anwar Bhatti

Administrivia

- Three HWs (***Time Complexities***, ***Graphs*** and ***Divide & Conquer***)
- Solutions will be released on Friday
- Tutorial in A1 – Thursday, 6:00-7:30 pm
- ***Midterm Exam*** finalized



Chapter 4:
Greedy Algorithms

Greed is Goooood

Scenario: The Last of Us

Scenario: The Last of Us

Scene: We are in LUMS maze at night (limited visibility)

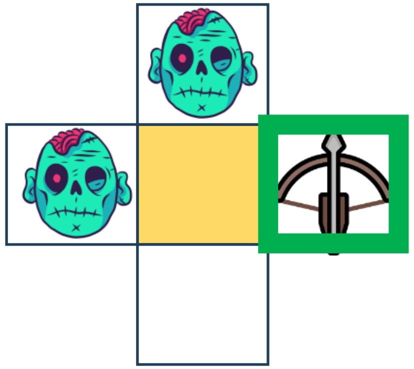
Objective: Collect the immediate weapon in sight and reach to safe zone

Rules:

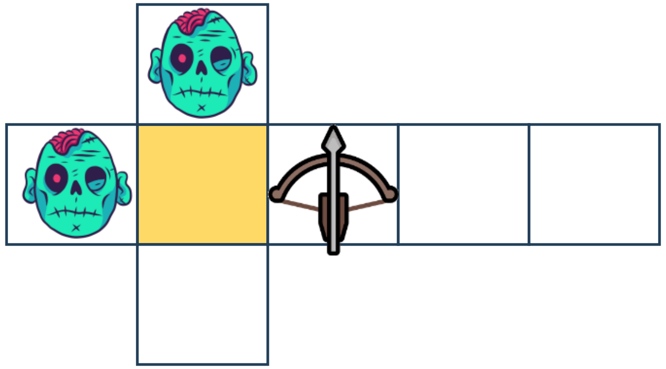
- Can move in any direction: Up, Down, Left and Right
- Once a path is taken, there is no going back
- Game ends when you reach safe zone (or die)

Scenario 1

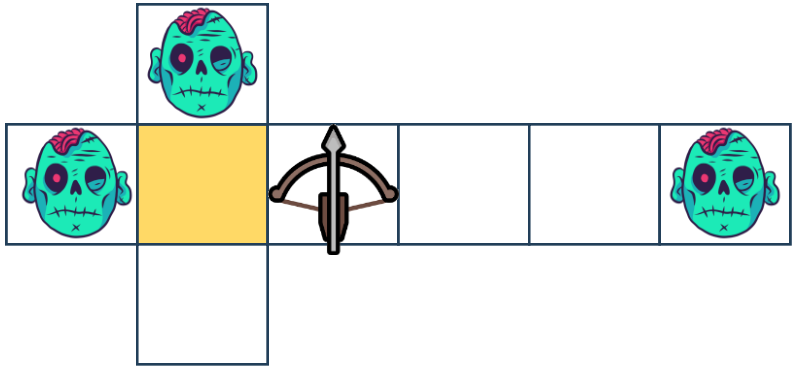
Scenario 1



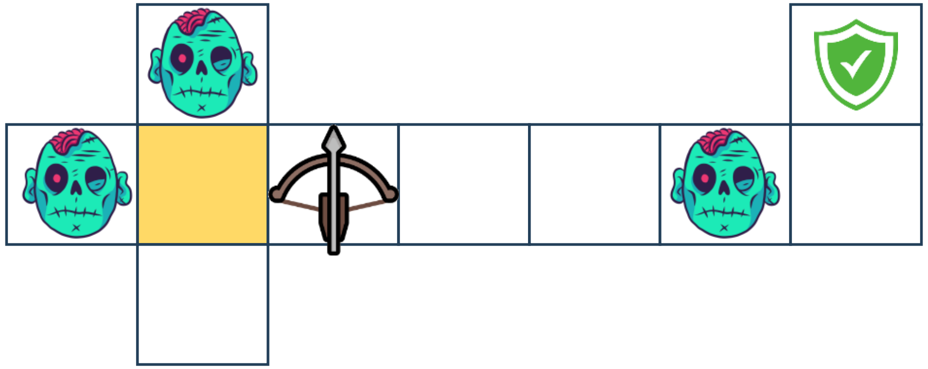
Scenario 1



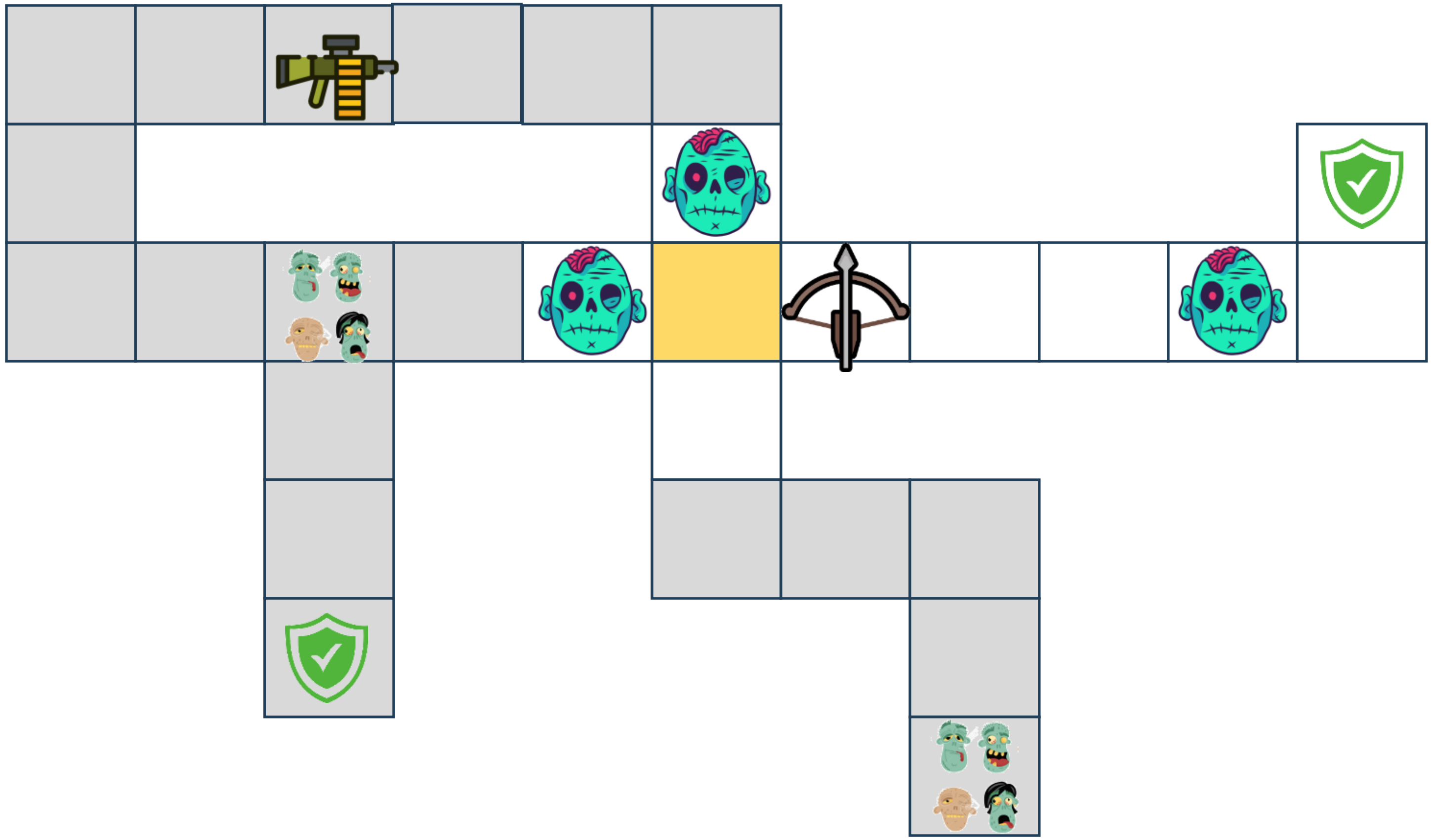
Scenario 1



Scenario 1

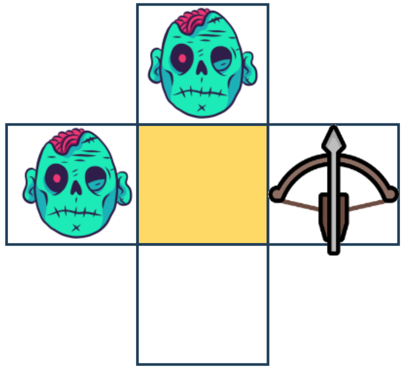


Scenario 1

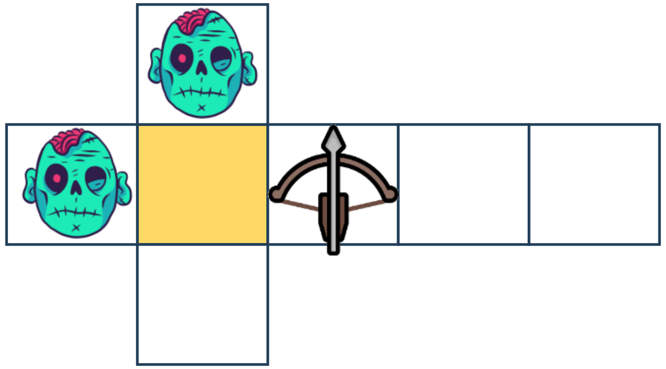


Scenario 2

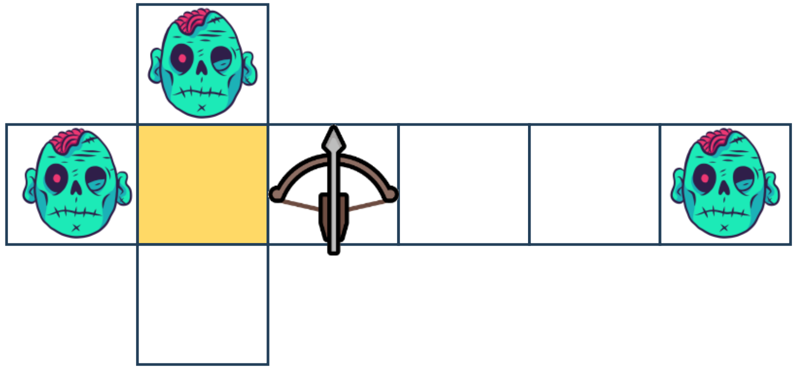
Scenario 2



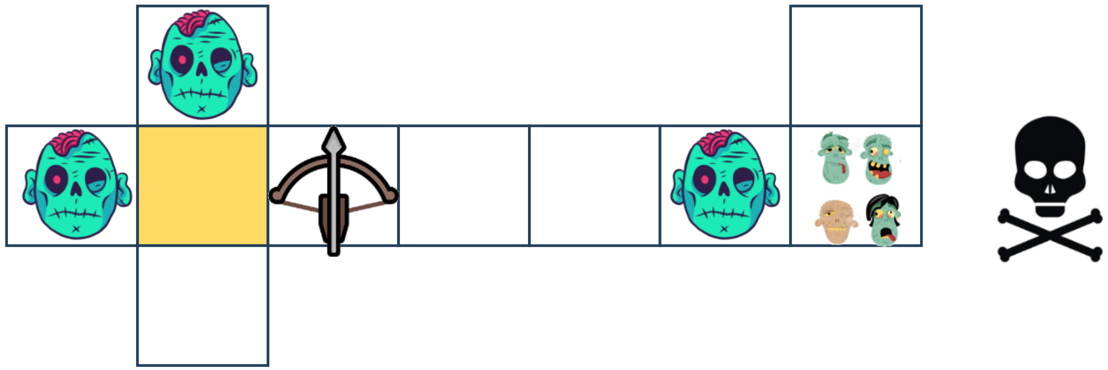
Scenario 2



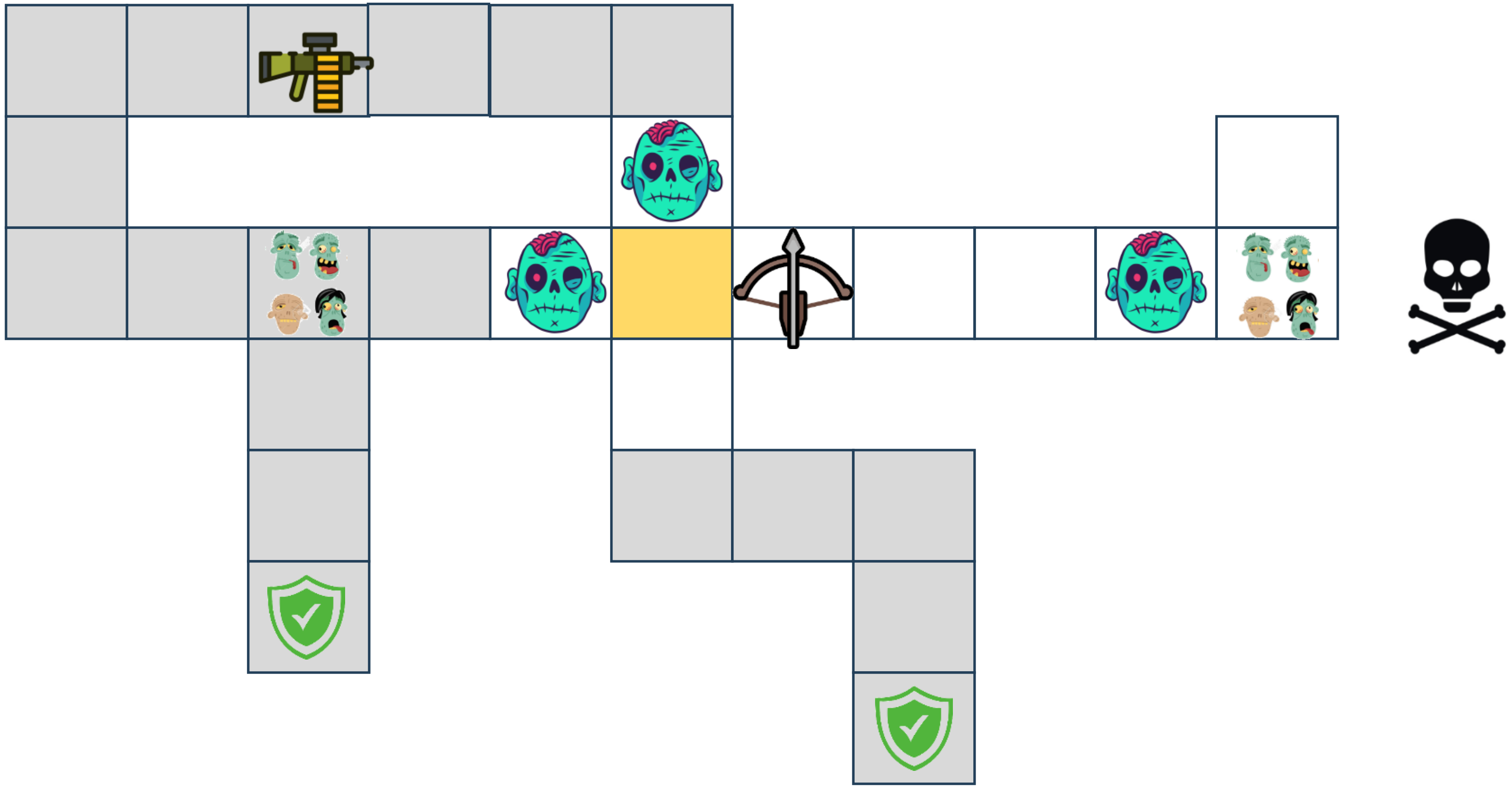
Scenario 2



Scenario 2



Scenario 2



Greedy Algorithms

- An algorithm is greedy if it builds up solution in small steps

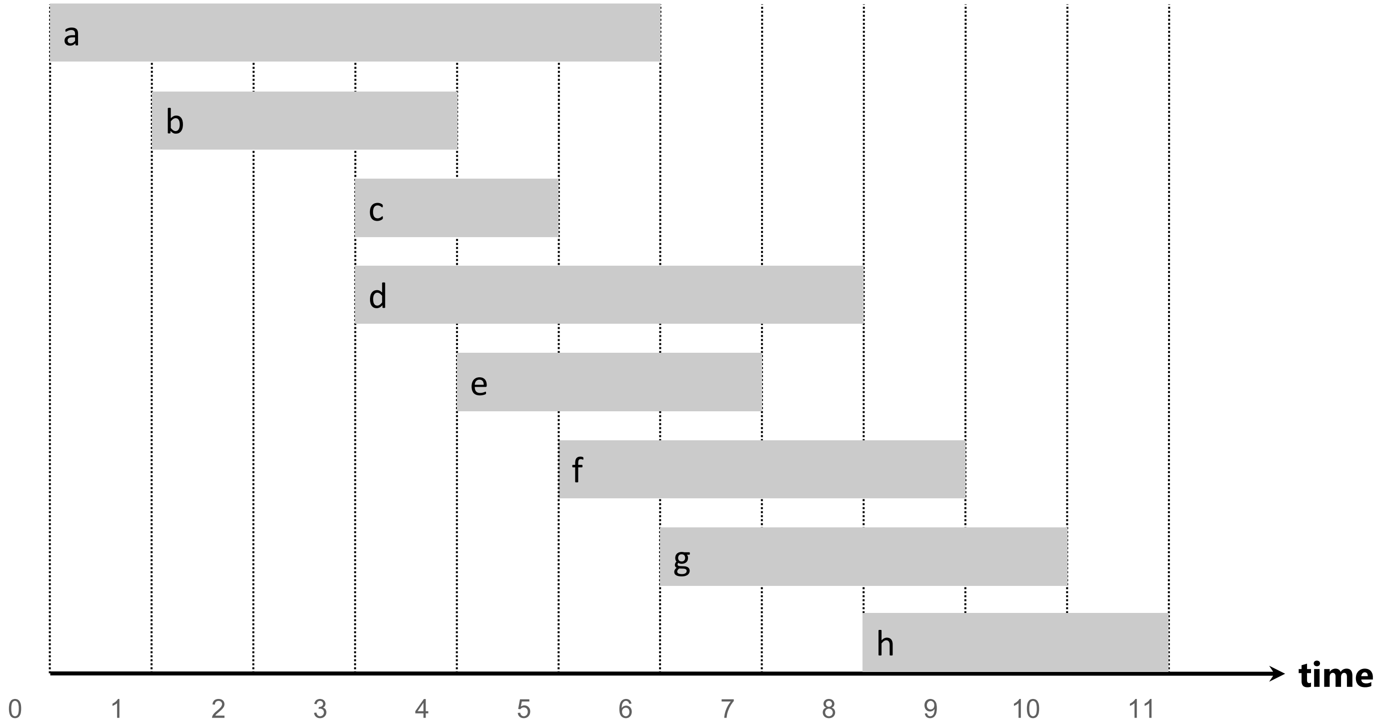
- In each step:
  - make a choice that looks best at the moment (greedy choice)
  - incrementally optimize the solution

- For many problems, greedy algorithms yield **global optimal** solution
  - Interval scheduling
  - Interval partitioning
  - Shortest paths in a graph
  - Minimum Spanning Tree

  - Perfect and stable matching

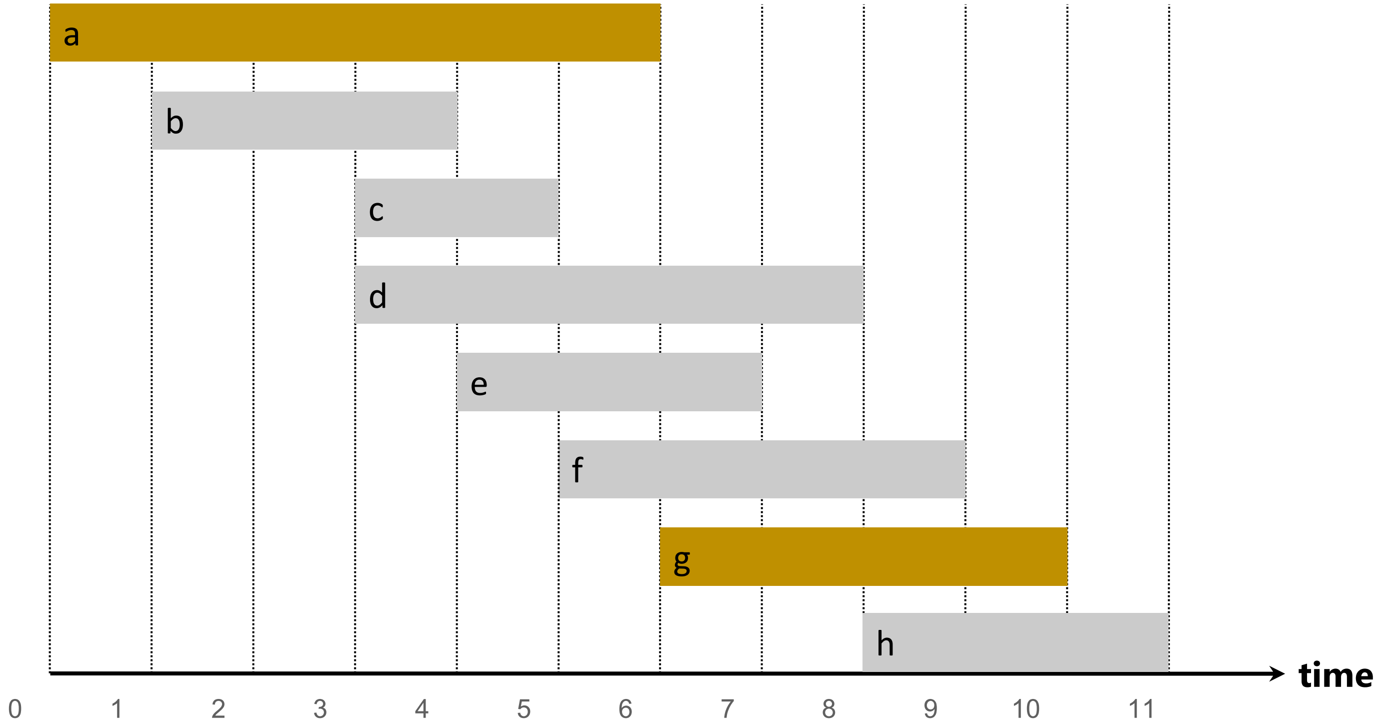
Interval Scheduling Problem

- Job j starts at s_j and finishes at f_j.
- Two jobs are compatible if they don't overlap.
- **Goal:** find maximum subset of mutually compatible jobs.



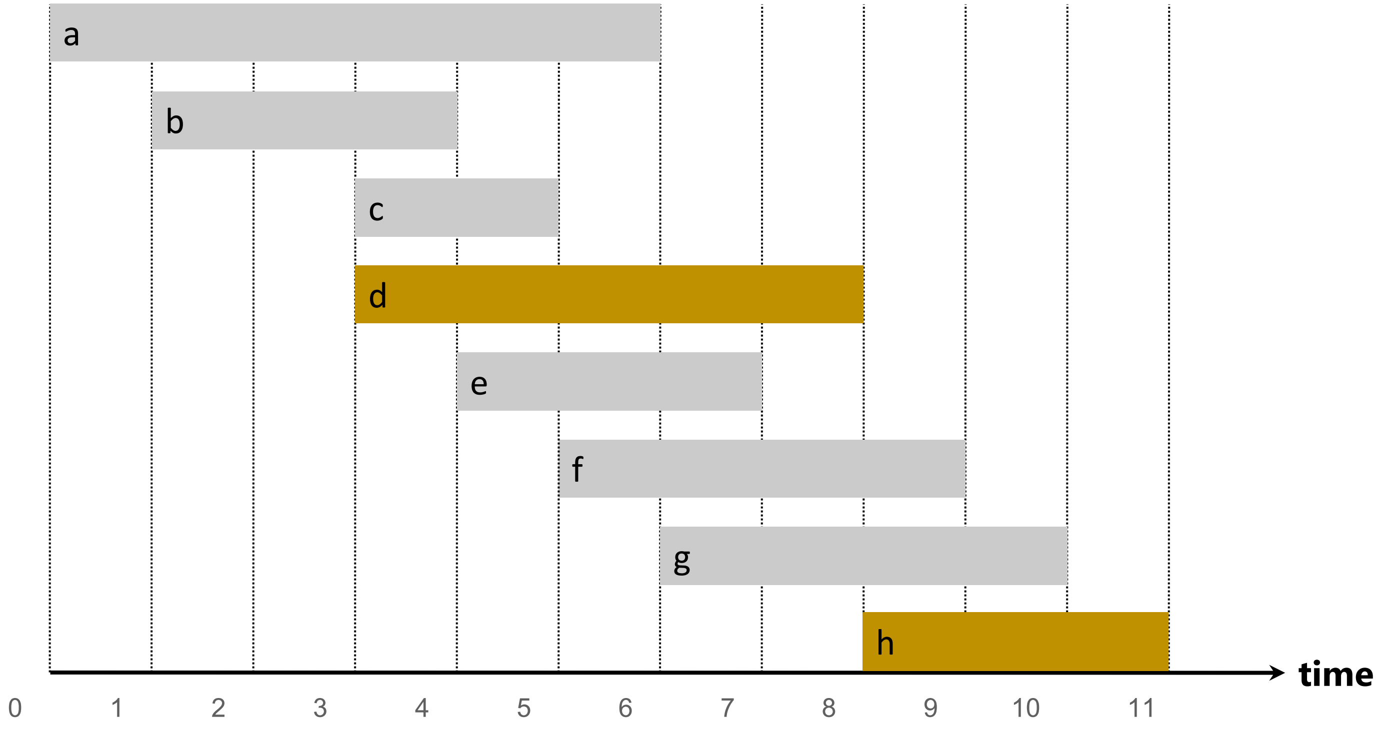
Interval Scheduling Problem

- Job j starts at s_j and finishes at f_j.
- Two jobs are compatible if they don't overlap.
- **Goal:** find maximum subset of mutually compatible jobs.



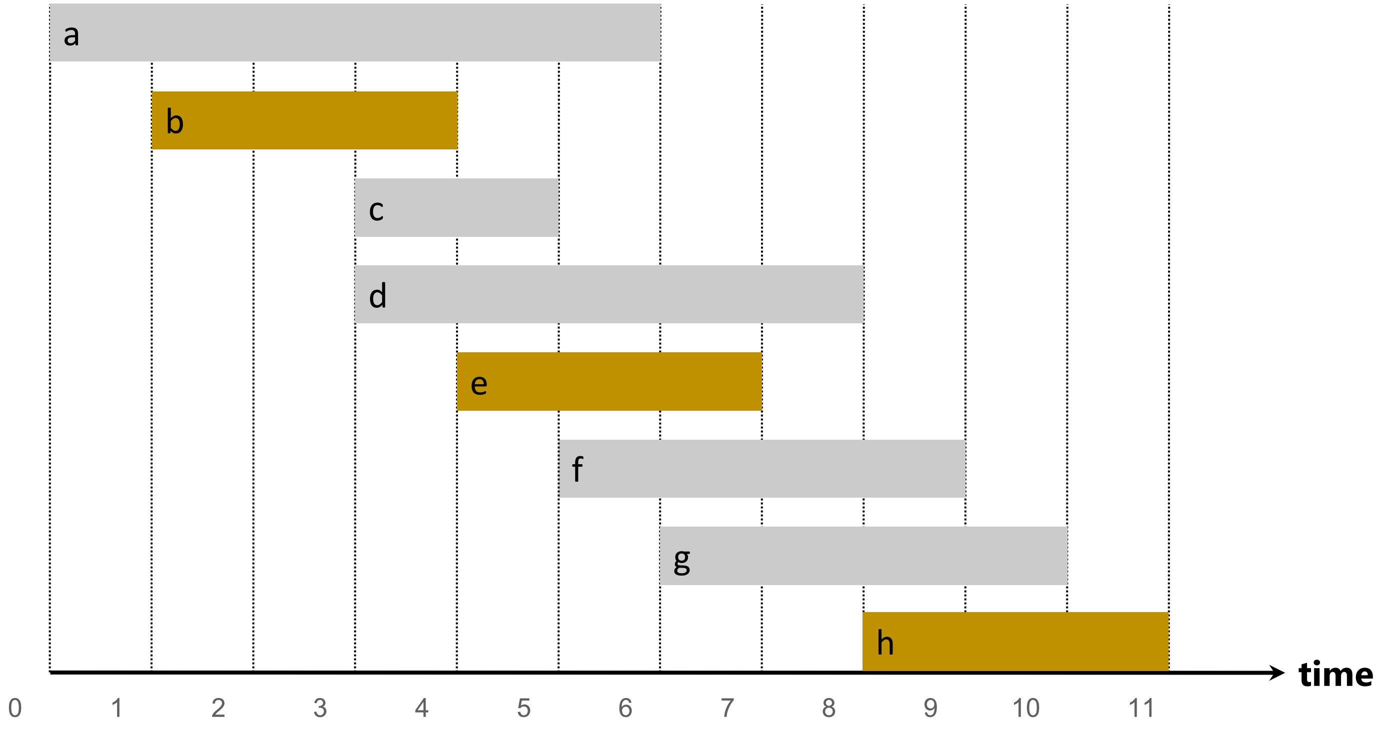
Interval Scheduling Problem

- Job j starts at s_j and finishes at f_j.
- Two jobs are compatible if they don't overlap.
- **Goal:** find maximum subset of mutually compatible jobs.



Interval Scheduling Problem

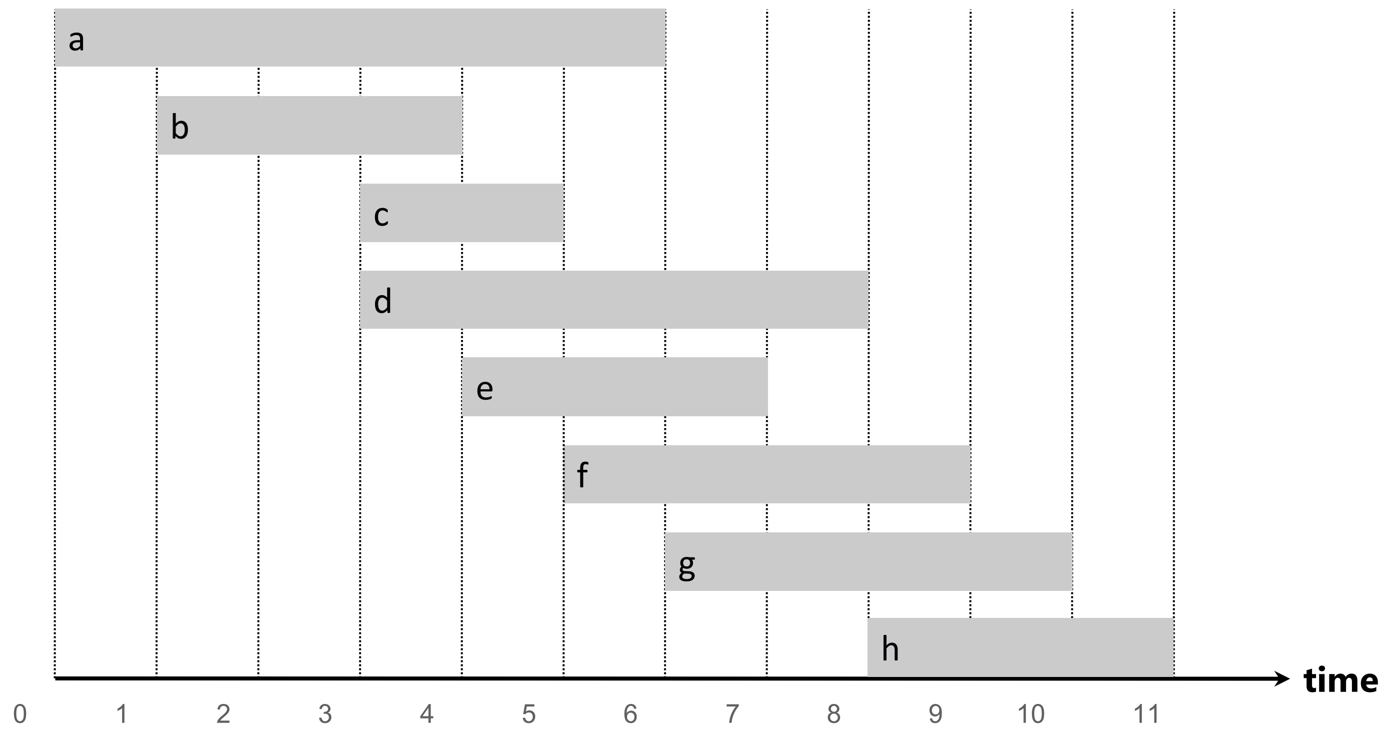
- Job j starts at s_j and finishes at f_j.
- Two jobs are compatible if they don't overlap.
- **Goal:** find maximum subset of mutually compatible jobs.



Interval Scheduling Problem

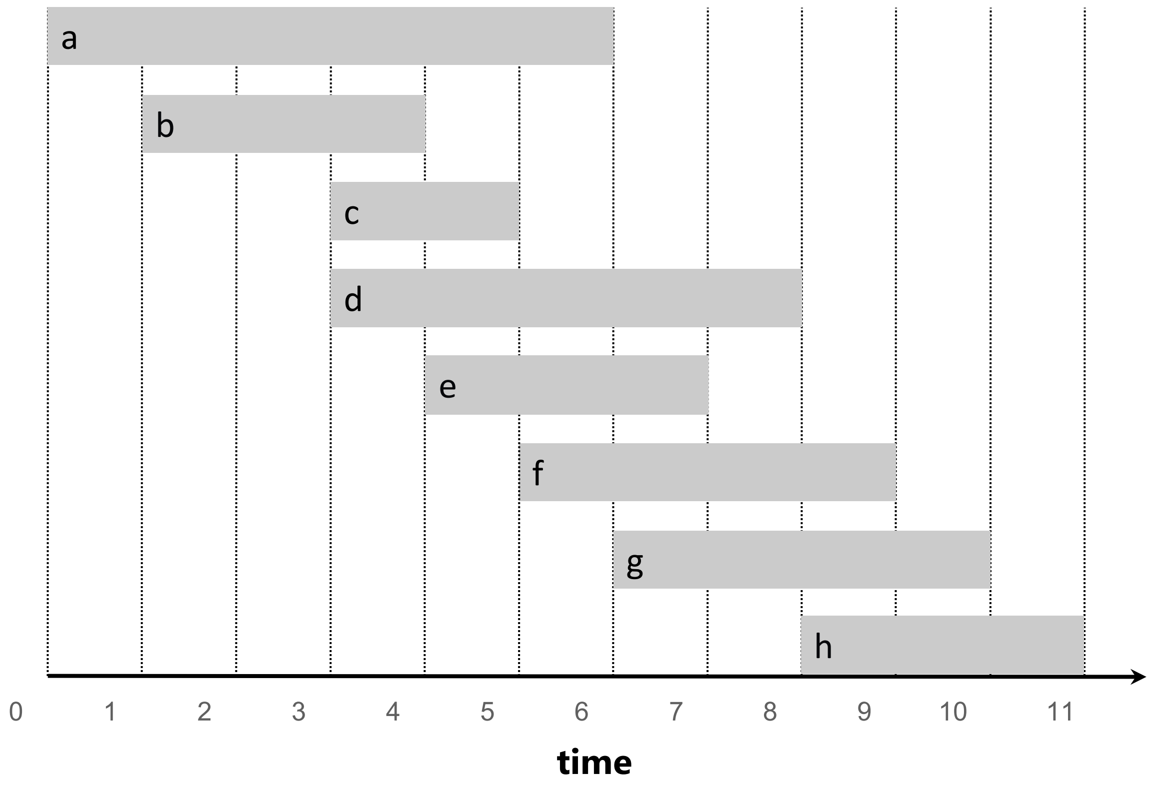
- Earliest Starting Request First
- Latest Finishing Request First
- Shortest Duration Request First
- Earliest Finish Time First

Sub-optimal



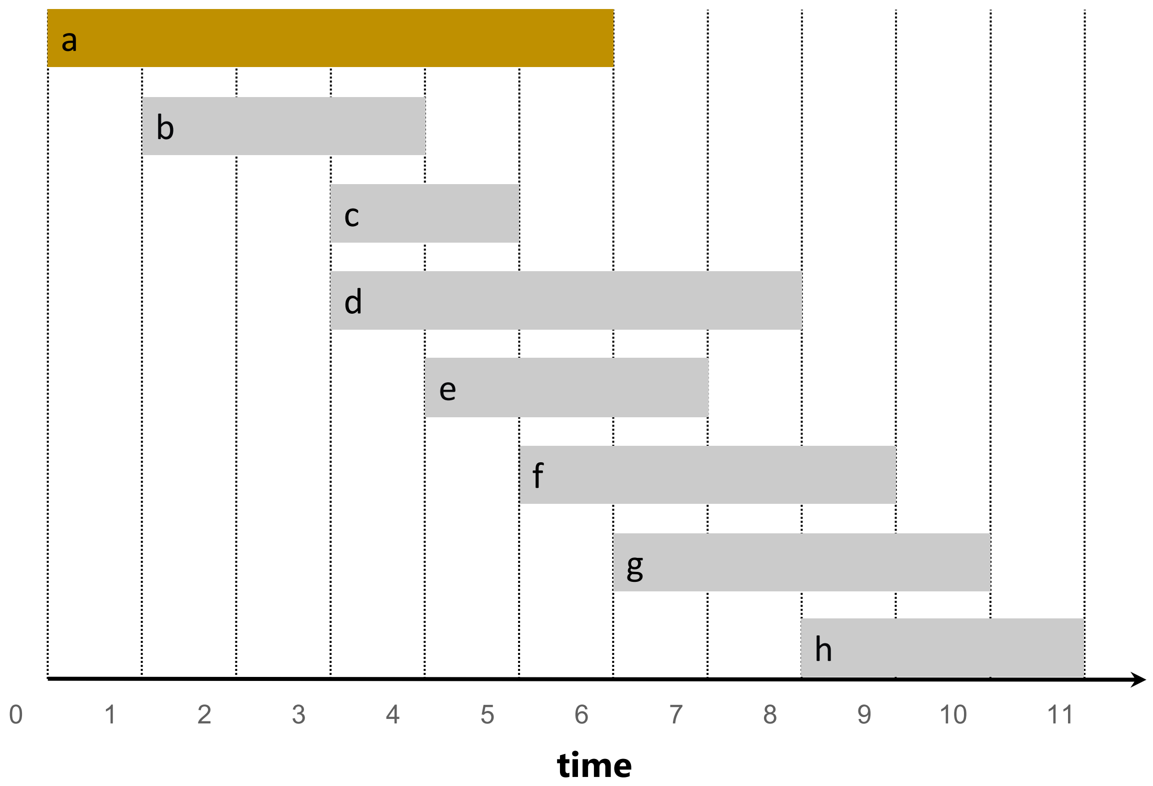
Interval Scheduling Problem

- **Earliest Starting Request First**
  - Select the request with the earliest start time
  - Eliminate any conflicting intervals (those that overlap with the chosen interval)
  - Continue this process until there are no more requests left.



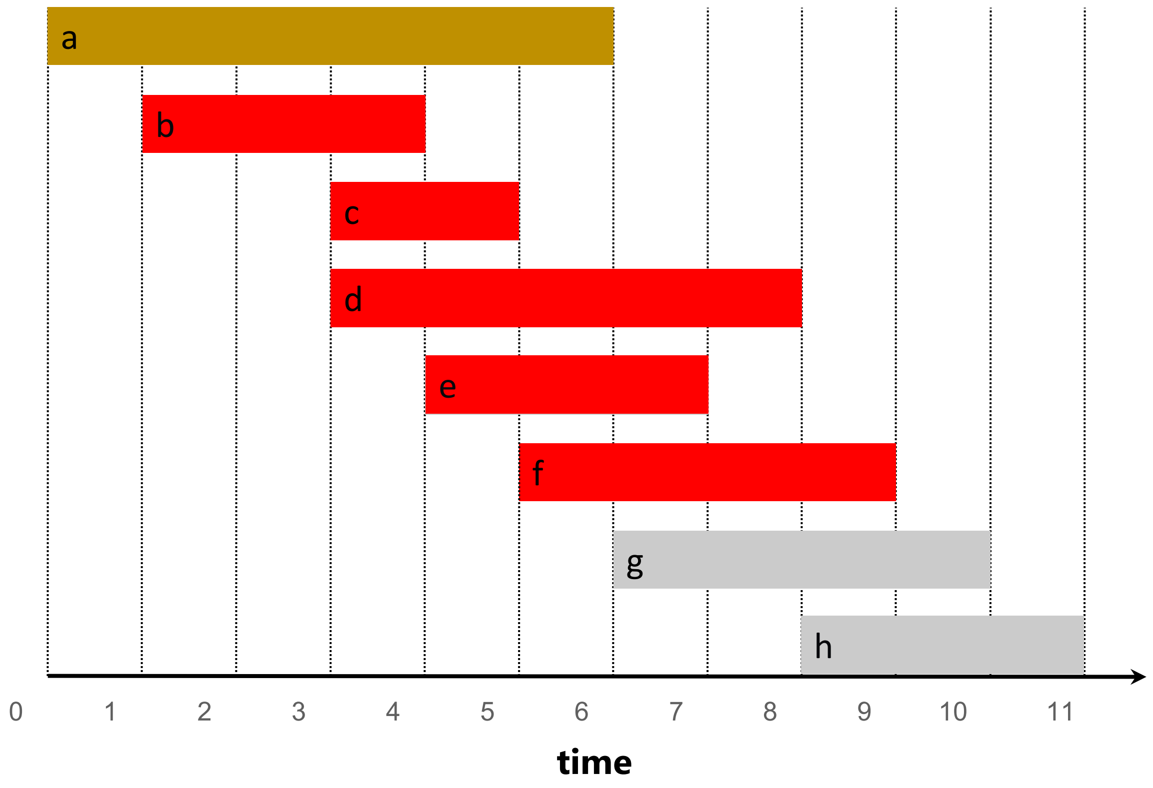
Interval Scheduling Problem

- **Earliest Starting Request First**
  - Select the request with the earliest start time
  - Eliminate any conflicting intervals (those that overlap with the chosen interval)
  - Continue this process until there are no more requests left.



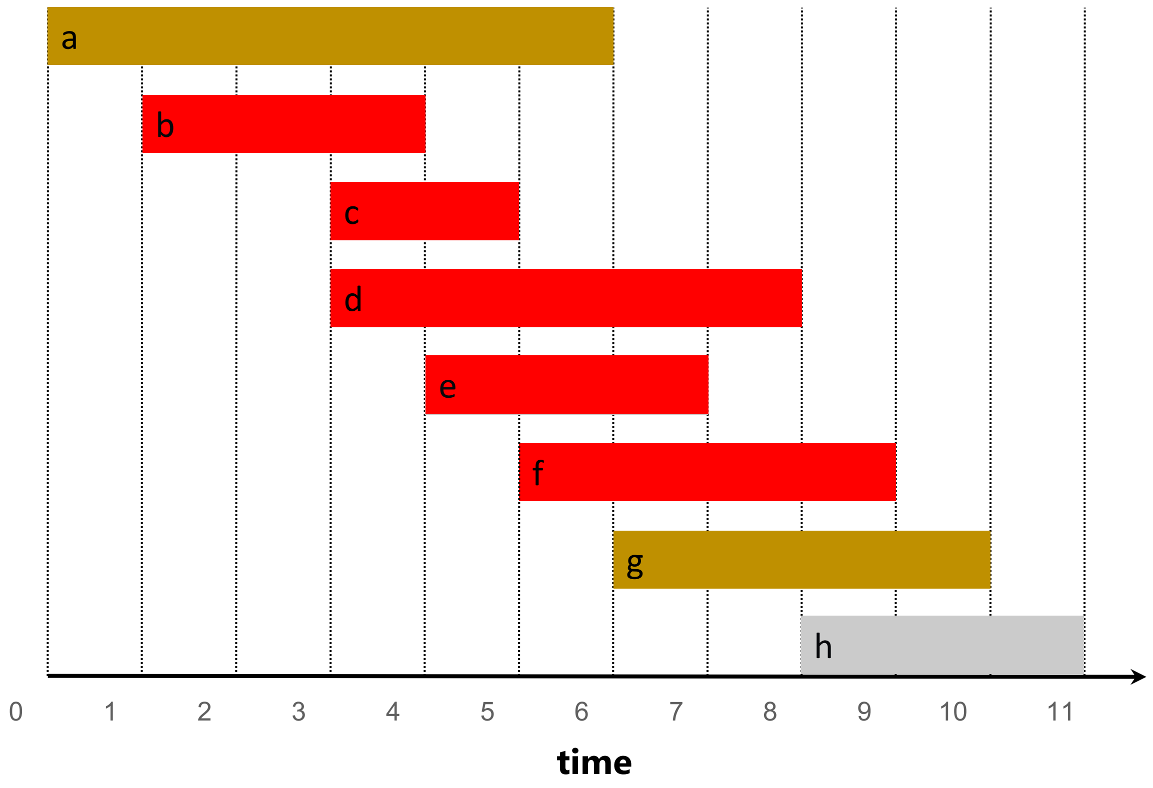
Interval Scheduling Problem

- **Earliest Starting Request First**
  - Select the request with the earliest start time
  - Eliminate any conflicting intervals (those that overlap with the chosen interval)
  - Continue this process until there are no more requests left.



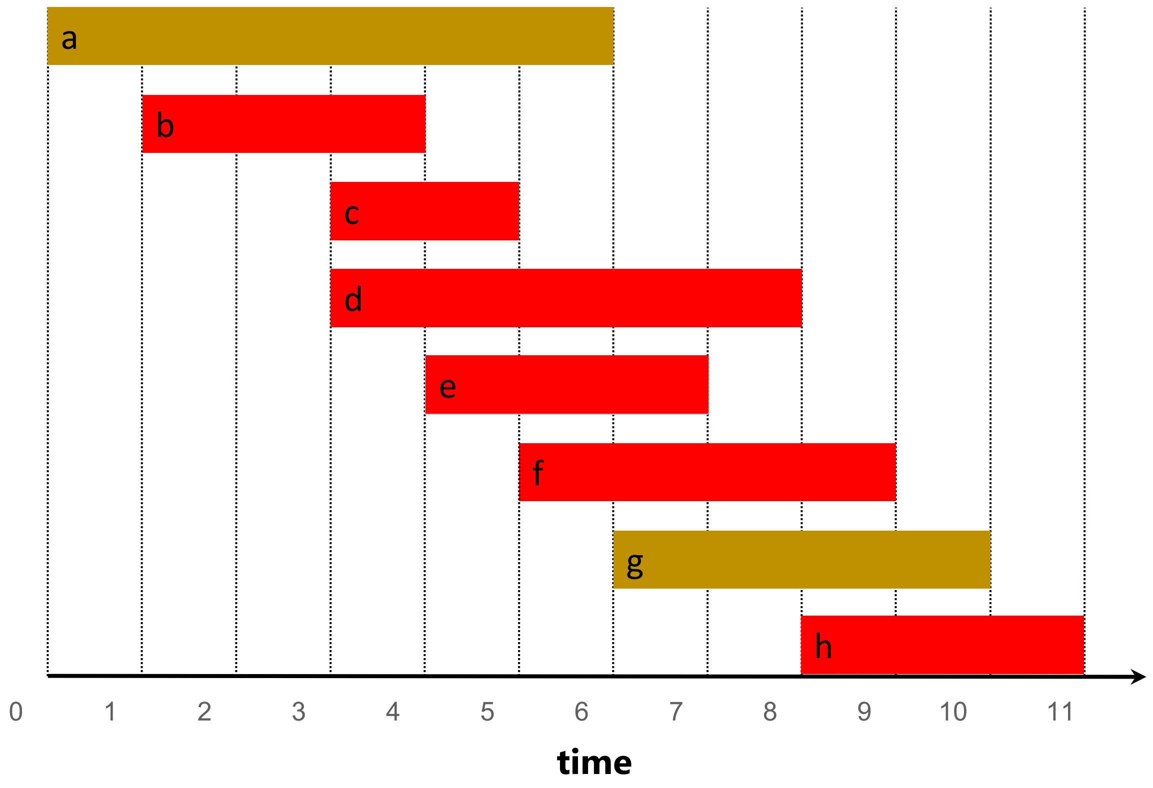
Interval Scheduling Problem

- **Earliest Starting Request First**
  - Select the request with the earliest start time
  - Eliminate any conflicting intervals (those that overlap with the chosen interval)
  - Continue this process until there are no more requests left.



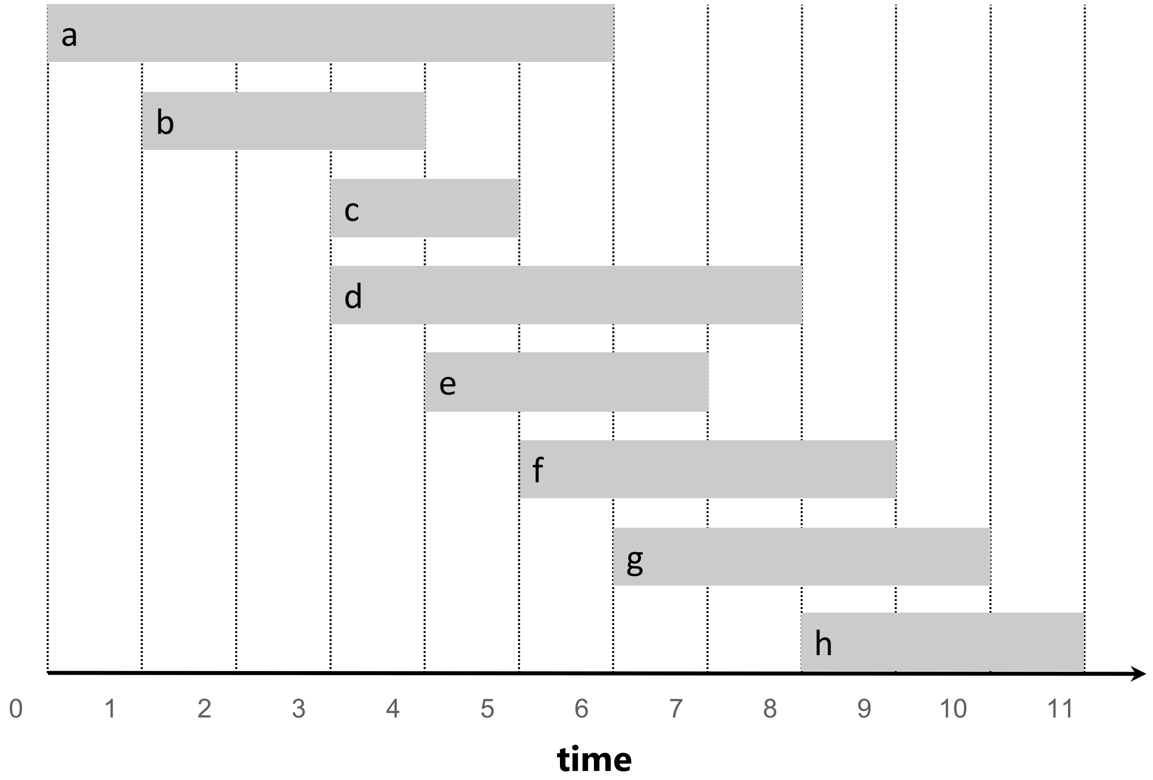
Interval Scheduling Problem

- **Earliest Starting Request First**
  - Select the request with the earliest start time
  - Eliminate any conflicting intervals (those that overlap with the chosen interval)
  - Continue this process until there are no more requests left.



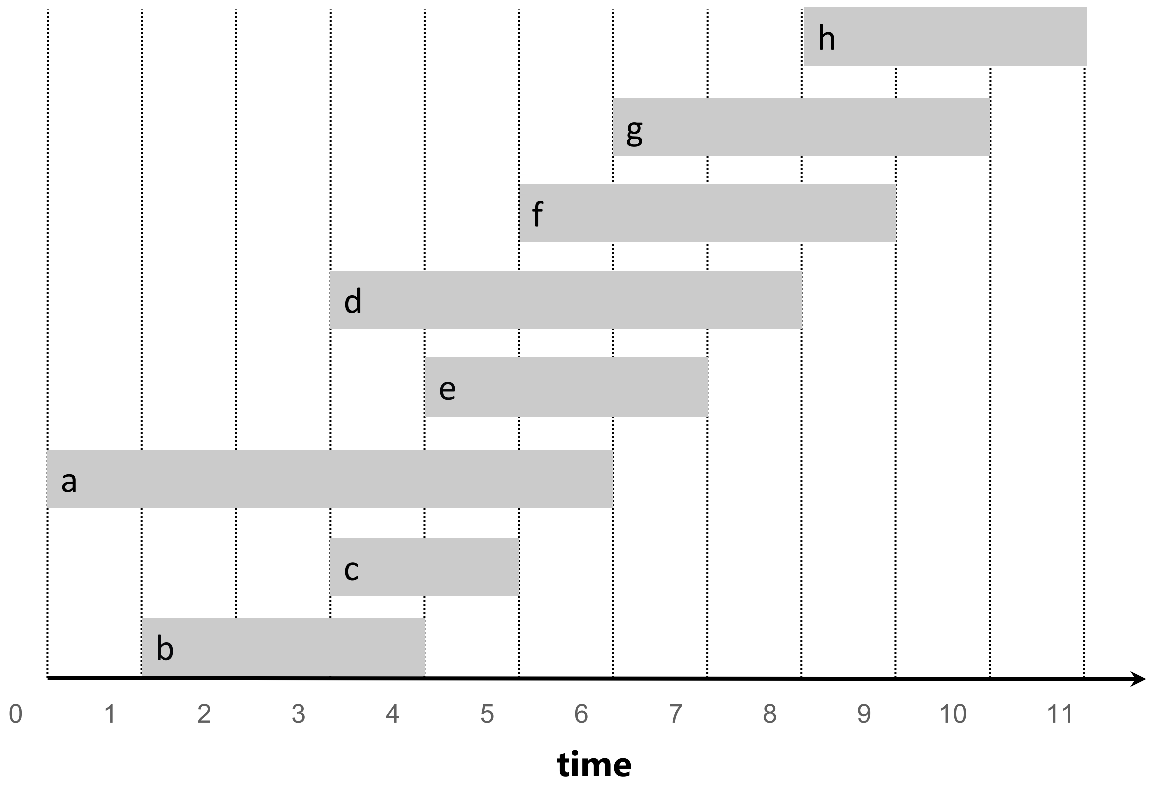
Interval Scheduling Problem

- **Latest Finishing Request First**
  - Select the request with the latest finish time
  - Eliminate any conflicting intervals
  - Continue until there are no more requests left.



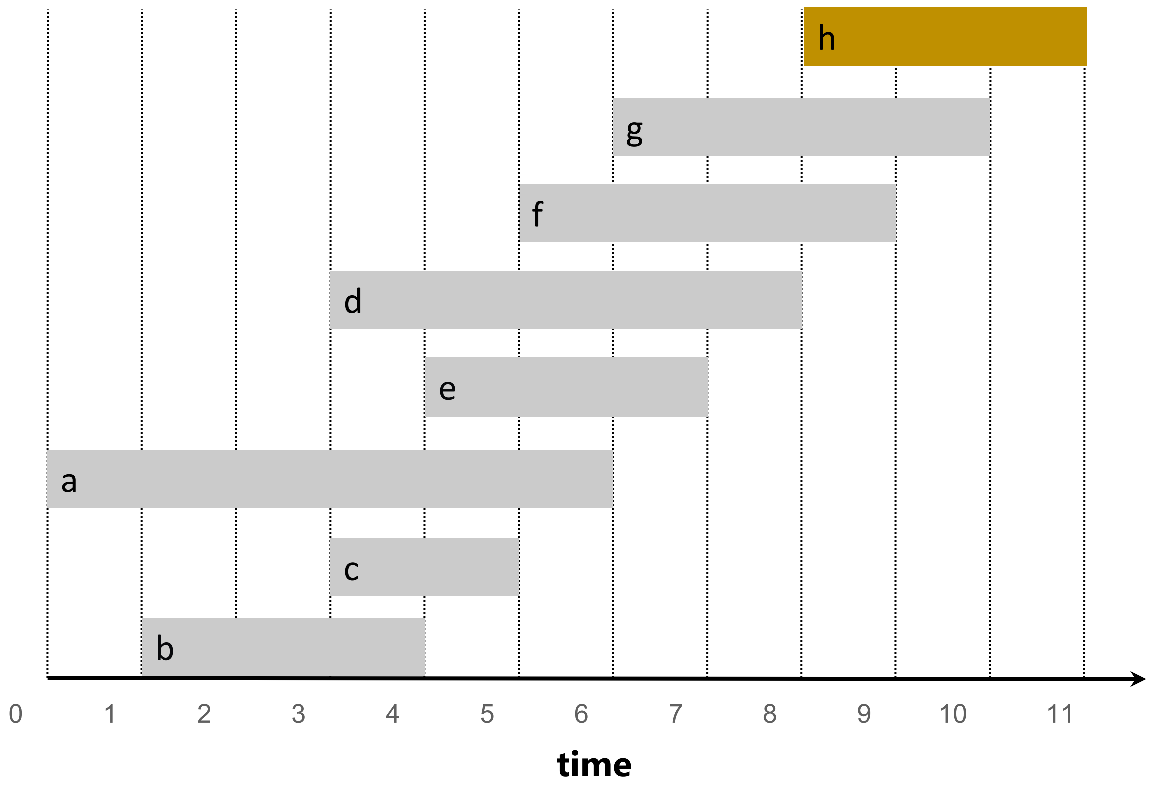
Interval Scheduling Problem

- **Latest Finishing Request First**
  - Select the request with the latest finish time
  - Eliminate any conflicting intervals
  - Continue until there are no more requests left.



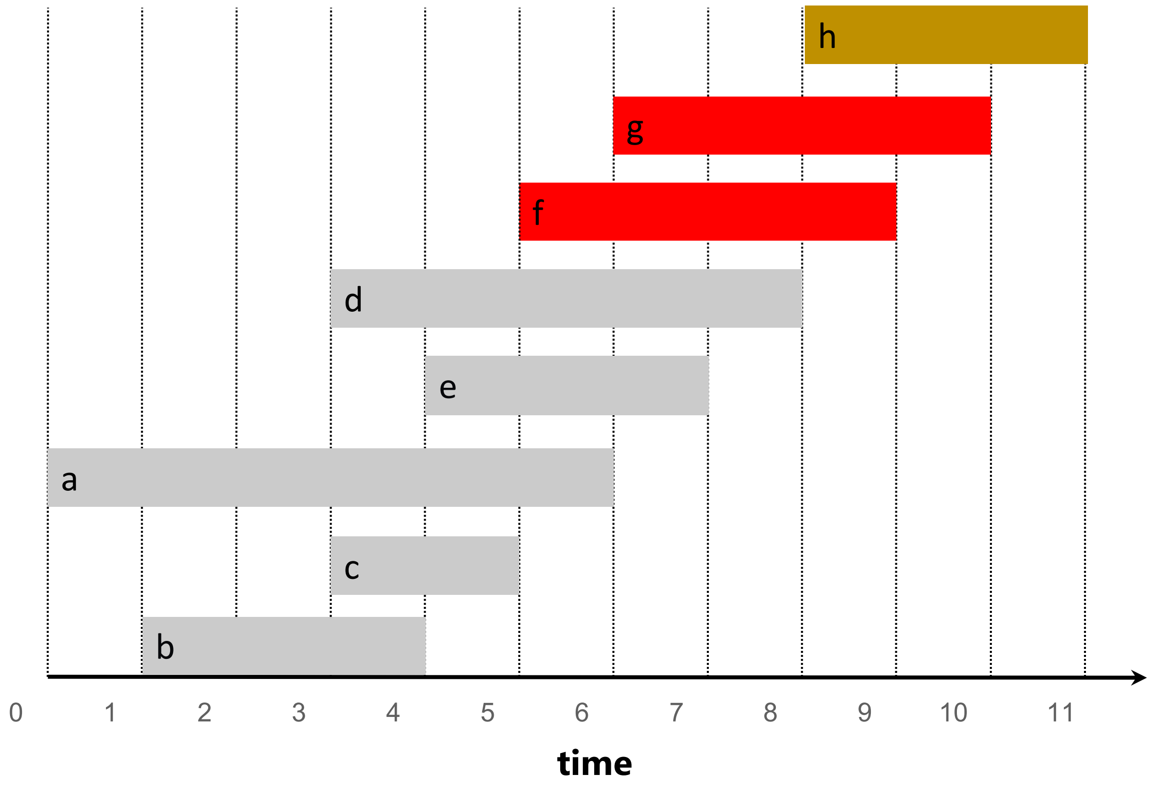
Interval Scheduling Problem

- **Latest Finishing Request First**
  - Select the request with the latest finish time
  - Eliminate any conflicting intervals
  - Continue until there are no more requests left.



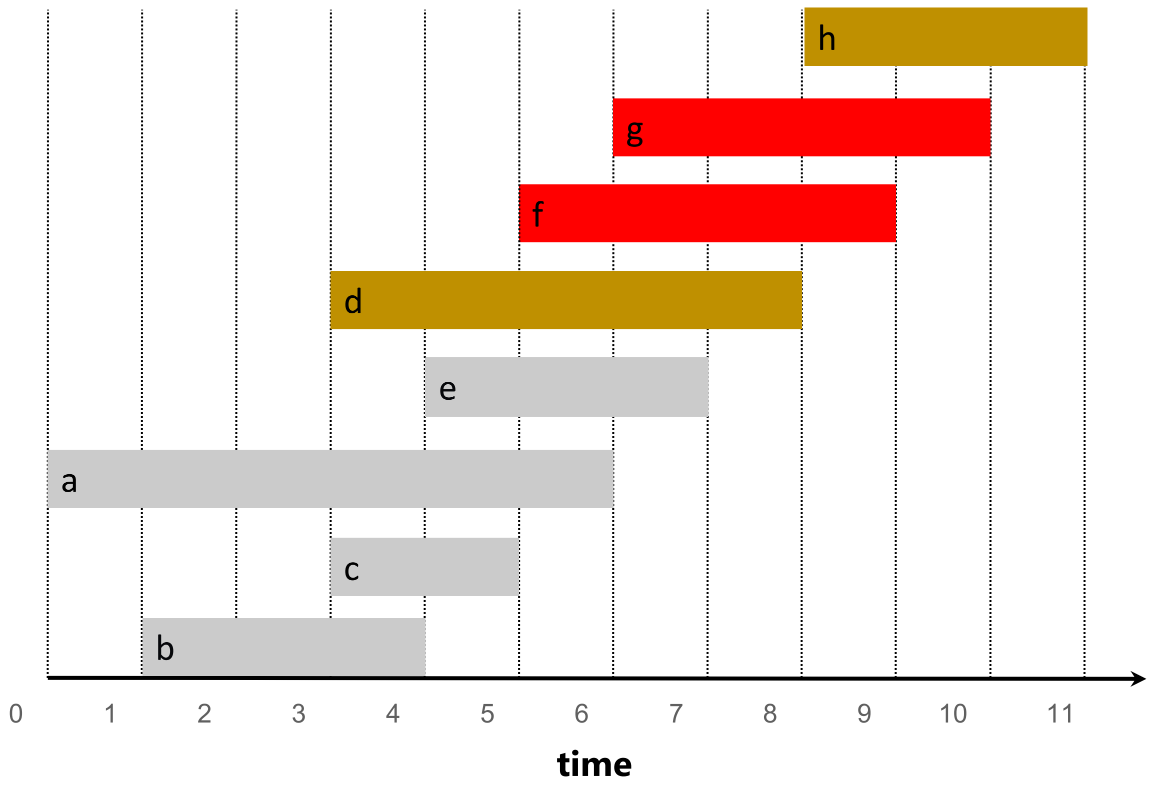
Interval Scheduling Problem

- **Latest Finishing Request First**
  - Select the request with the latest finish time
  - Eliminate any conflicting intervals
  - Continue until there are no more requests left.



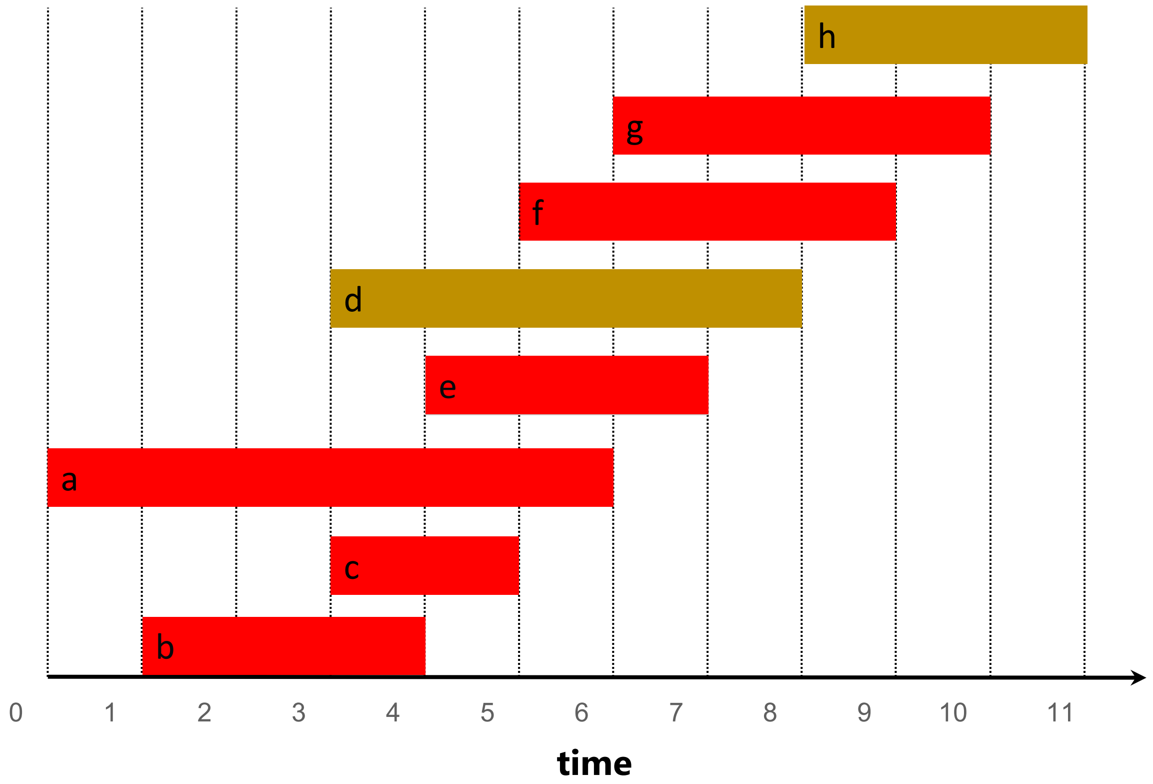
Interval Scheduling Problem

- **Latest Finishing Request First**
  - Select the request with the latest finish time
  - Eliminate any conflicting intervals
  - Continue until there are no more requests left.



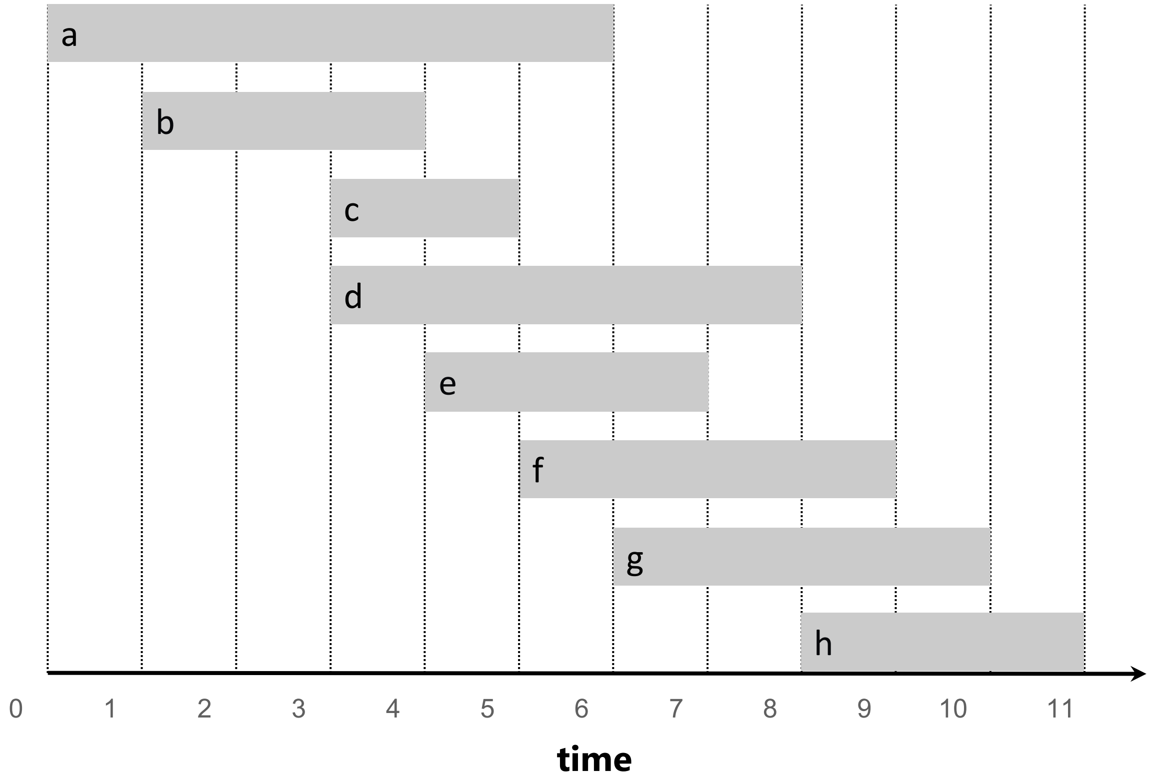
Interval Scheduling Problem

- **Latest Finishing Request First**
  - Select the request with the latest finish time
  - Eliminate any conflicting intervals
  - Continue until there are no more requests left.



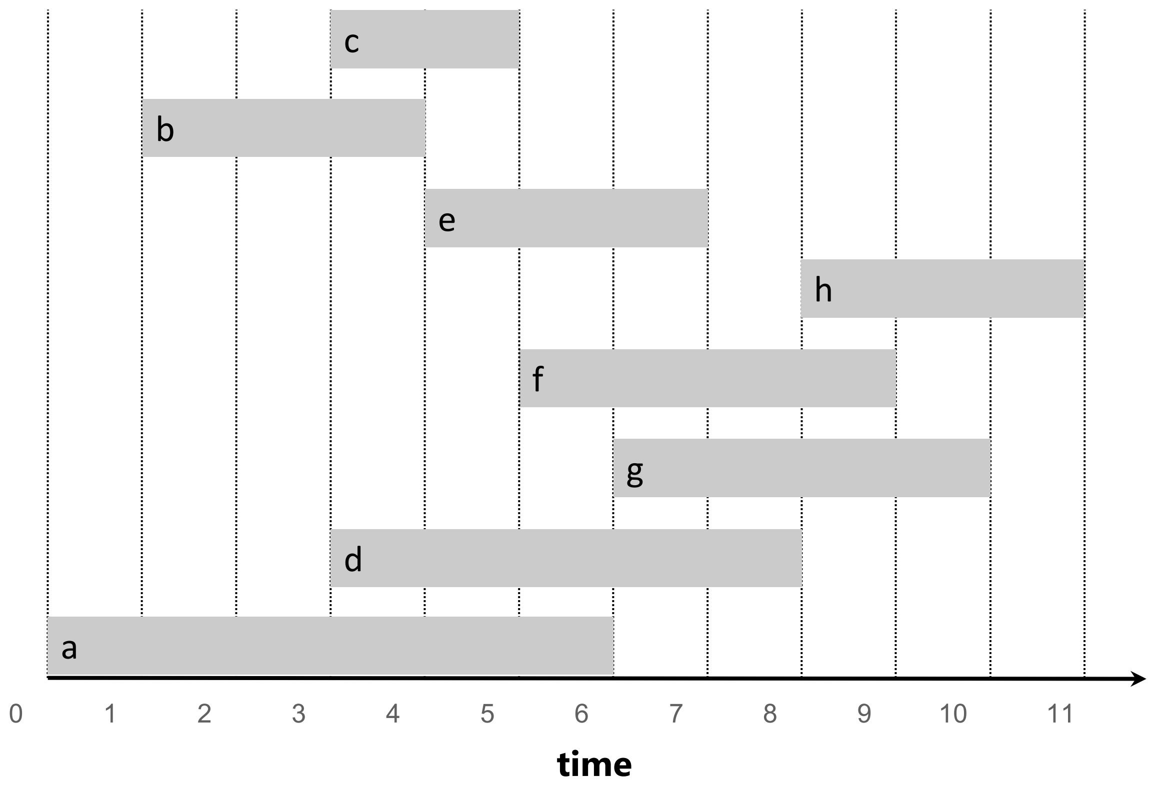
Interval Scheduling Problem

- **Shortest Duration Request First**
  - Select the shortest request
  - Eliminate any conflicting intervals
  - Continue this process until no more requests are left.



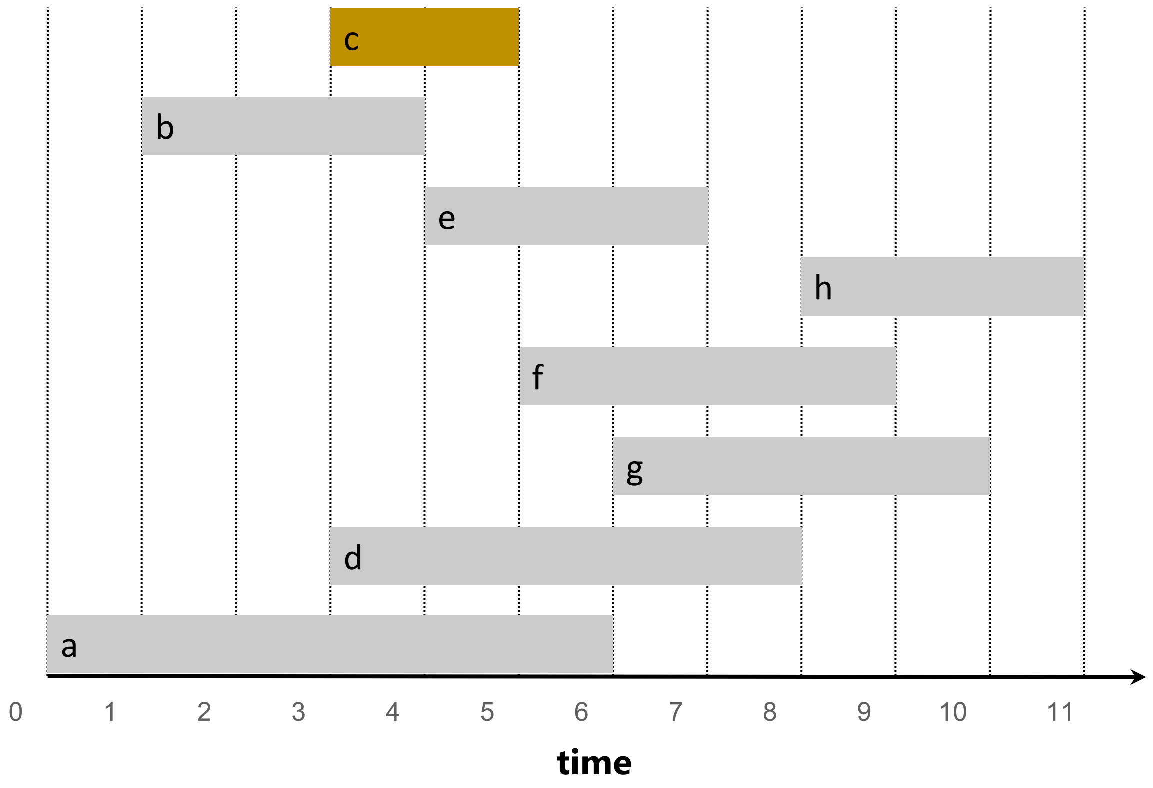
Interval Scheduling Problem

- Shortest Duration Request First
  - Select the shortest request
  - Eliminate any conflicting intervals
  - Continue this process until no more requests are left.



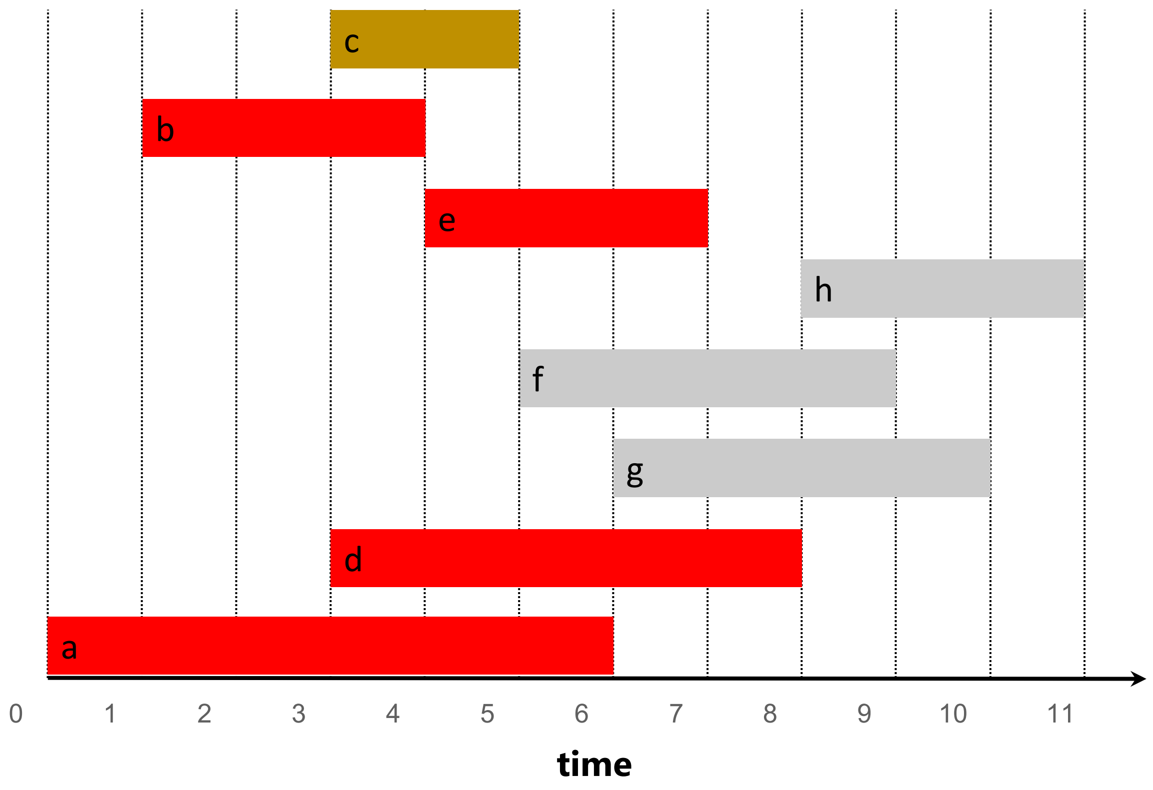
Interval Scheduling Problem

- Shortest Duration Request First
  - Select the shortest request
  - Eliminate any conflicting intervals
  - Continue this process until no more requests are left.



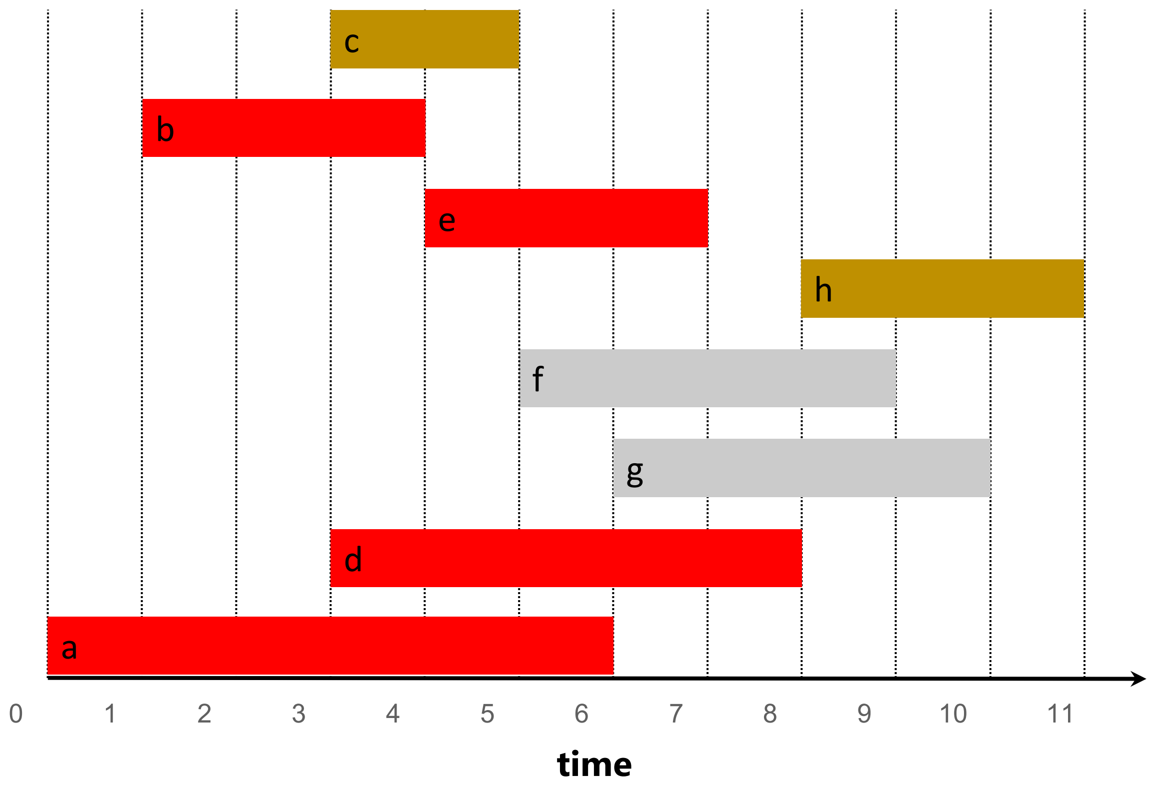
Interval Scheduling Problem

- Shortest Duration Request First
  - Select the shortest request
  - Eliminate any conflicting intervals
  - Continue this process until no more requests are left.



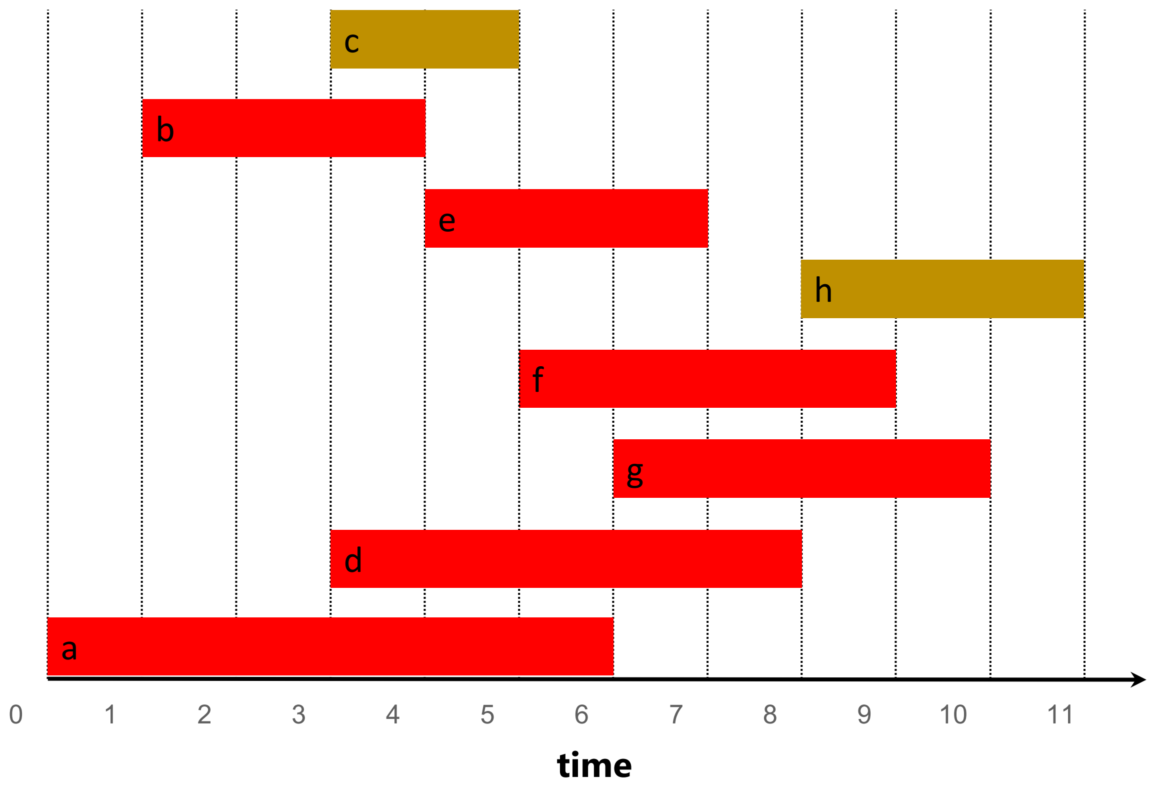
Interval Scheduling Problem

- **Shortest Duration Request First**
  - Select the shortest request
  - Eliminate any conflicting intervals
  - Continue this process until no more requests are left.



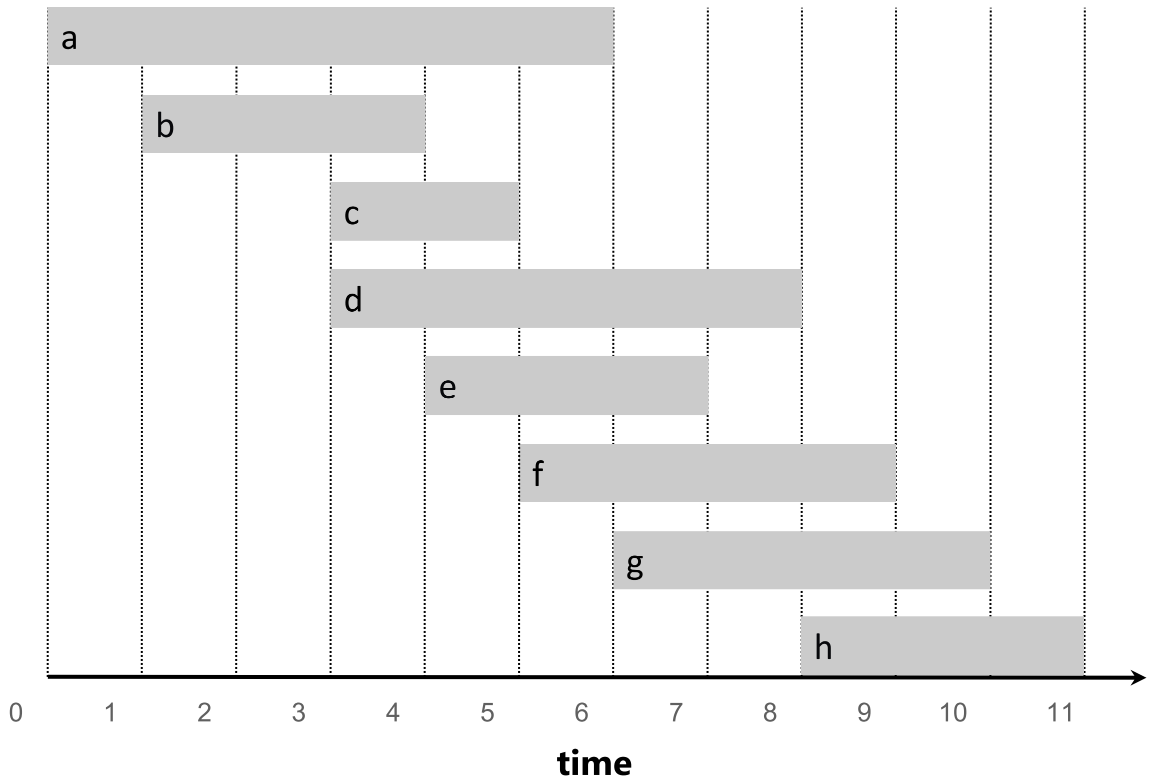
Interval Scheduling Problem

- **Shortest Duration Request First**
  - Select the shortest request
  - Eliminate any conflicting intervals
  - Continue this process until no more requests are left.



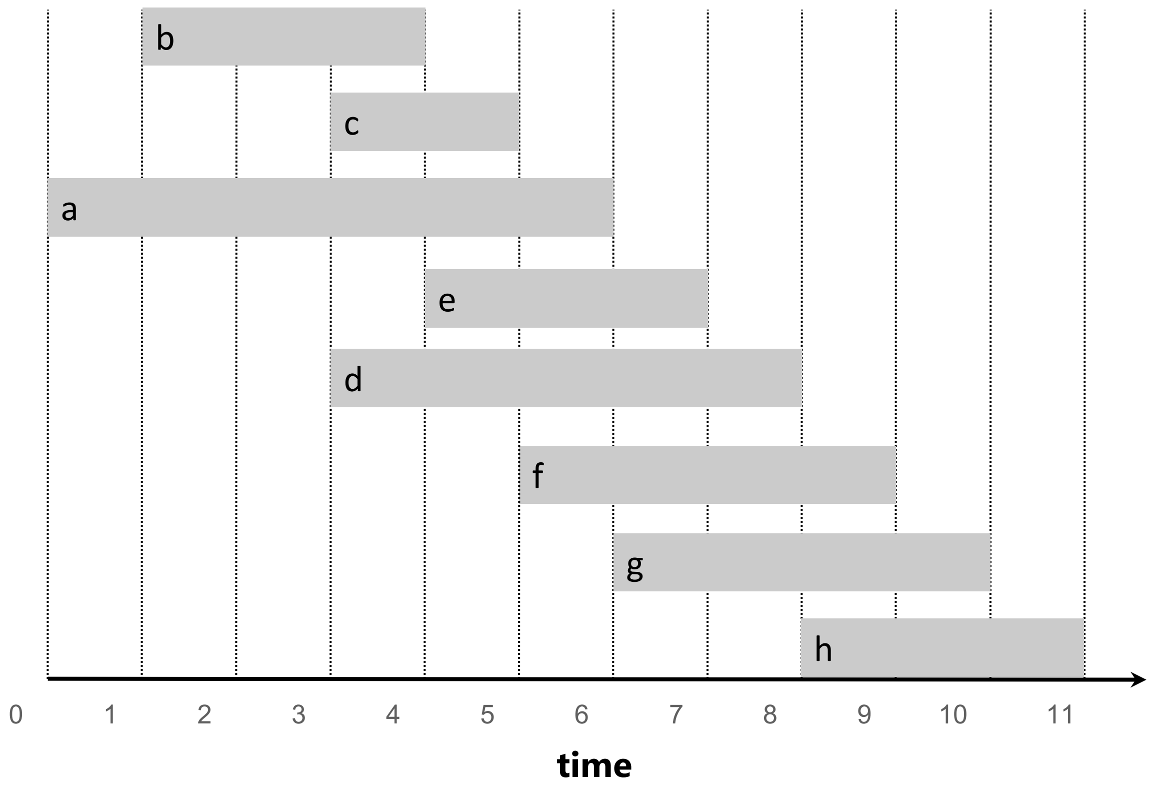
Interval Scheduling Problem

- **Earliest Finish Time First**
  - Select the shortest request
  - Eliminate any conflicting intervals
  - Continue this process until no more requests are left.



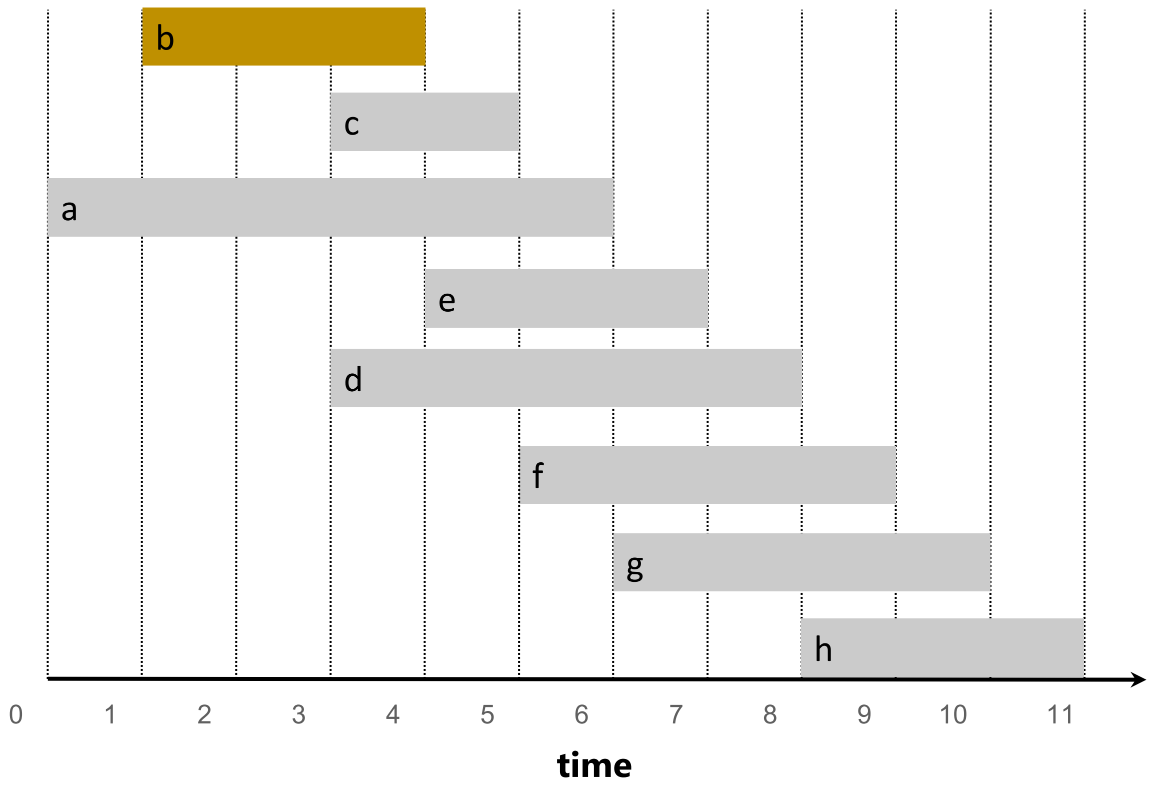
Interval Scheduling Problem

- **Earliest Finish Time First**
  - Select the shortest request
  - Eliminate any conflicting intervals
  - Continue this process until no more requests are left.



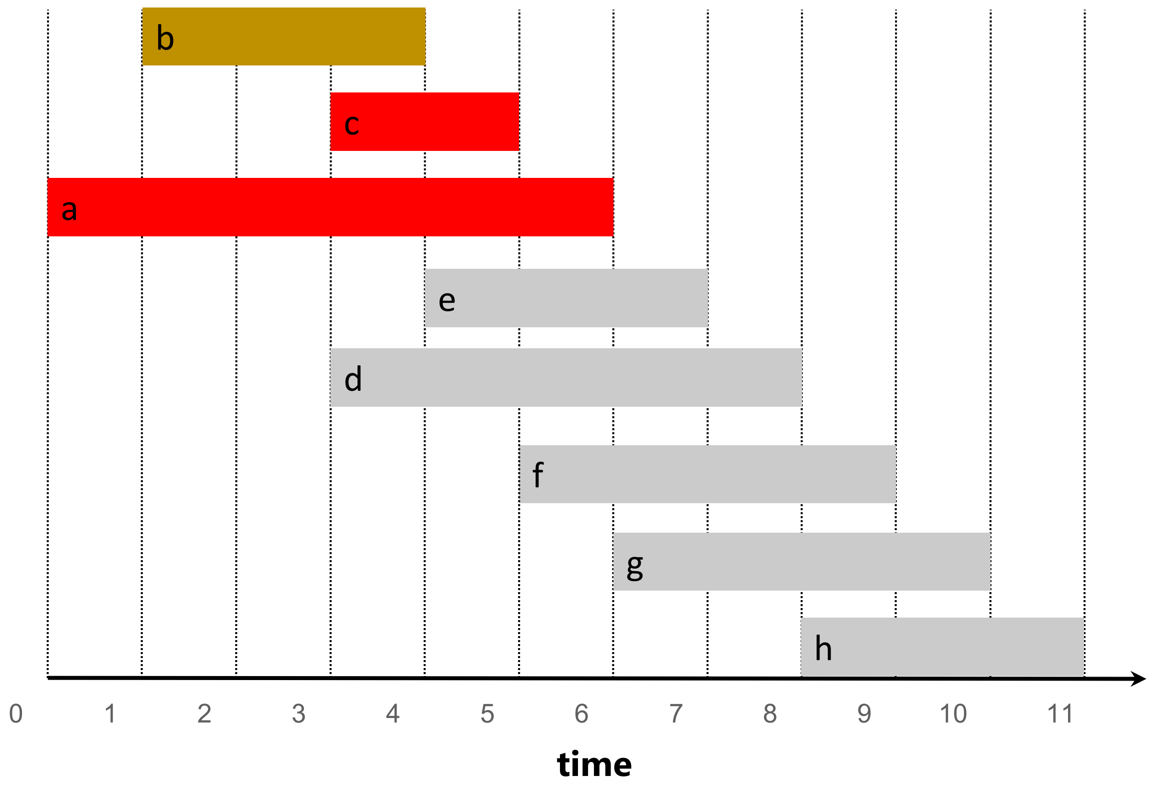
Interval Scheduling Problem

- **Earliest Finish Time First**
  - Select the shortest request
  - Eliminate any conflicting intervals
  - Continue this process until no more requests are left.



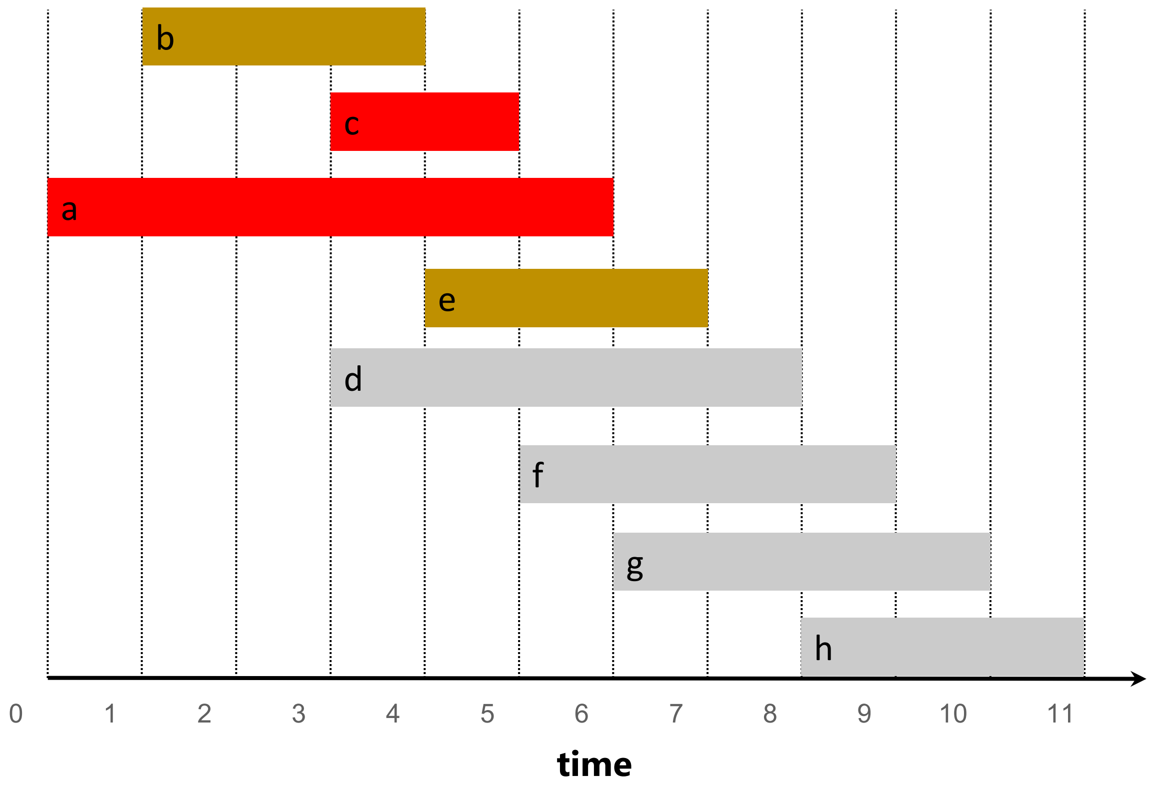
Interval Scheduling Problem

- **Earliest Finish Time First**
  - Select the shortest request
  - Eliminate any conflicting intervals
  - Continue this process until no more requests are left.



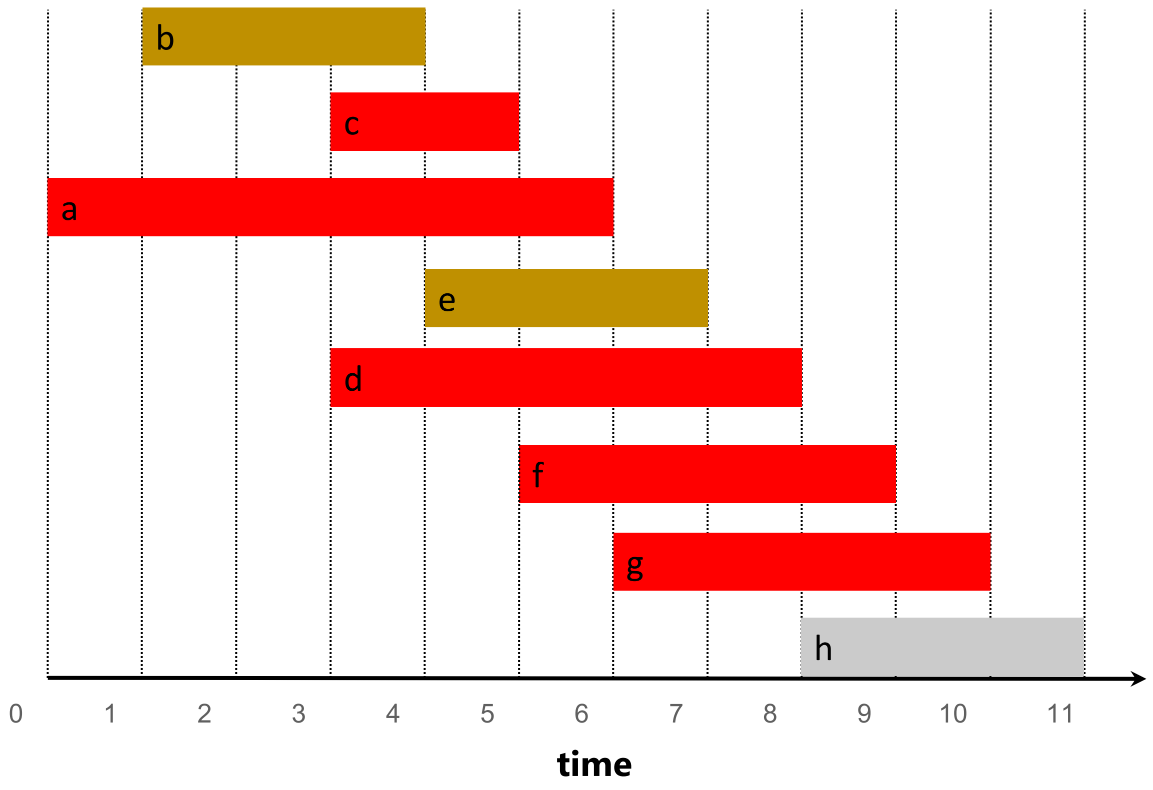
Interval Scheduling Problem

- **Earliest Finish Time First**
  - Select the shortest request
  - Eliminate any conflicting intervals
  - Continue this process until no more requests are left.



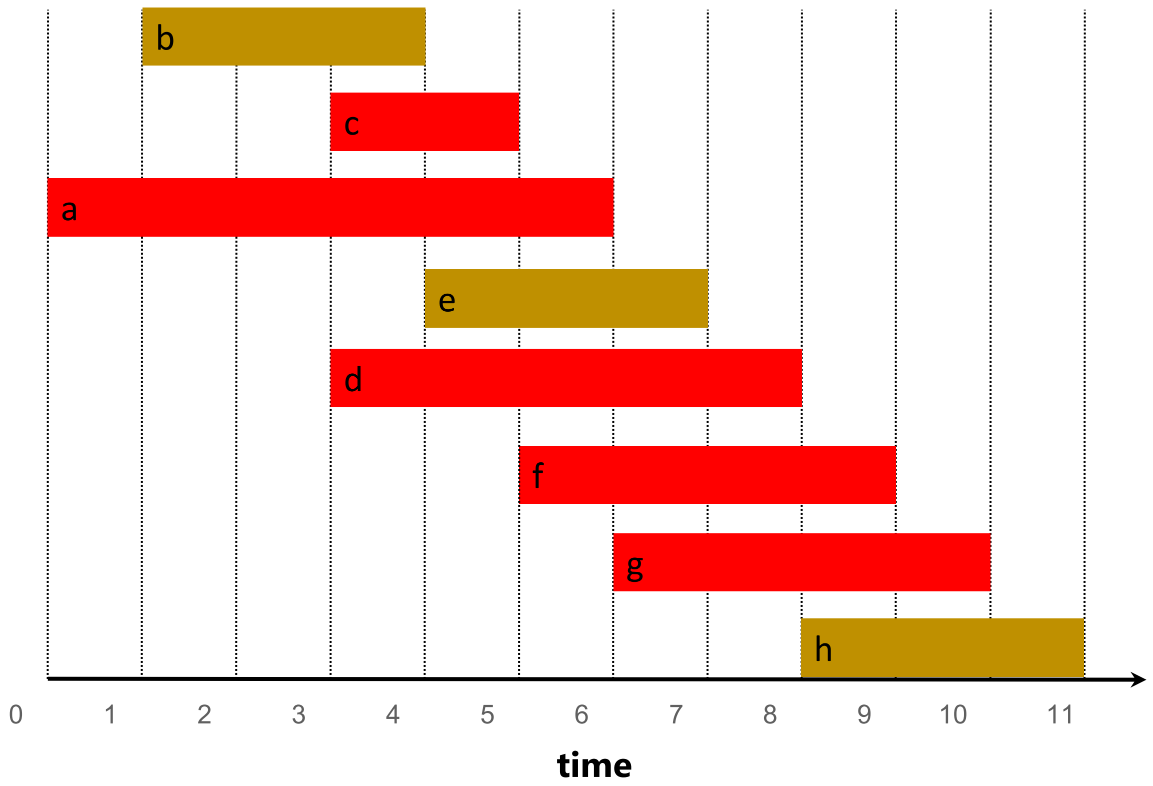
Interval Scheduling Problem

- **Earliest Finish Time First**
  - Select the shortest request
  - Eliminate any conflicting intervals
  - Continue this process until no more requests are left.



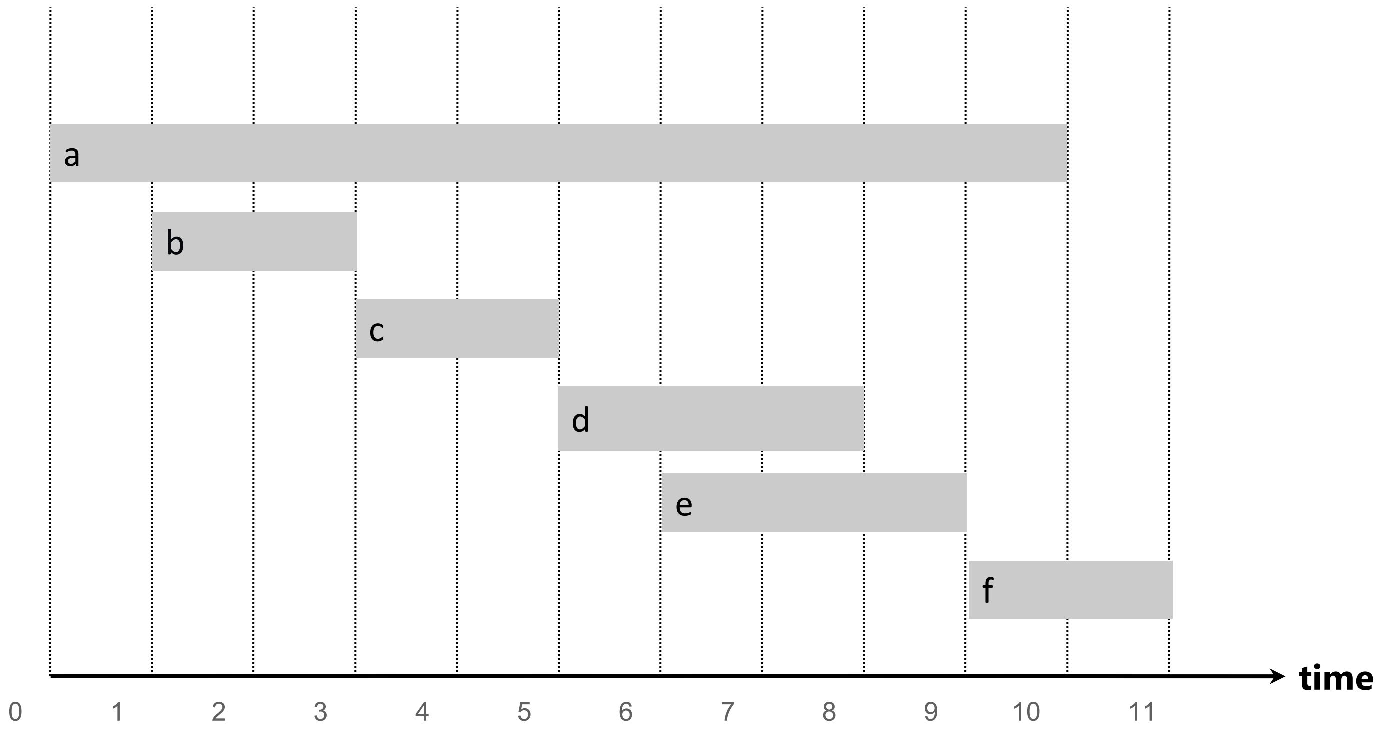
Interval Scheduling Problem

- **Earliest Finish Time First**
  - Select the shortest request
  - Eliminate any conflicting intervals
  - Continue this process until no more requests are left.



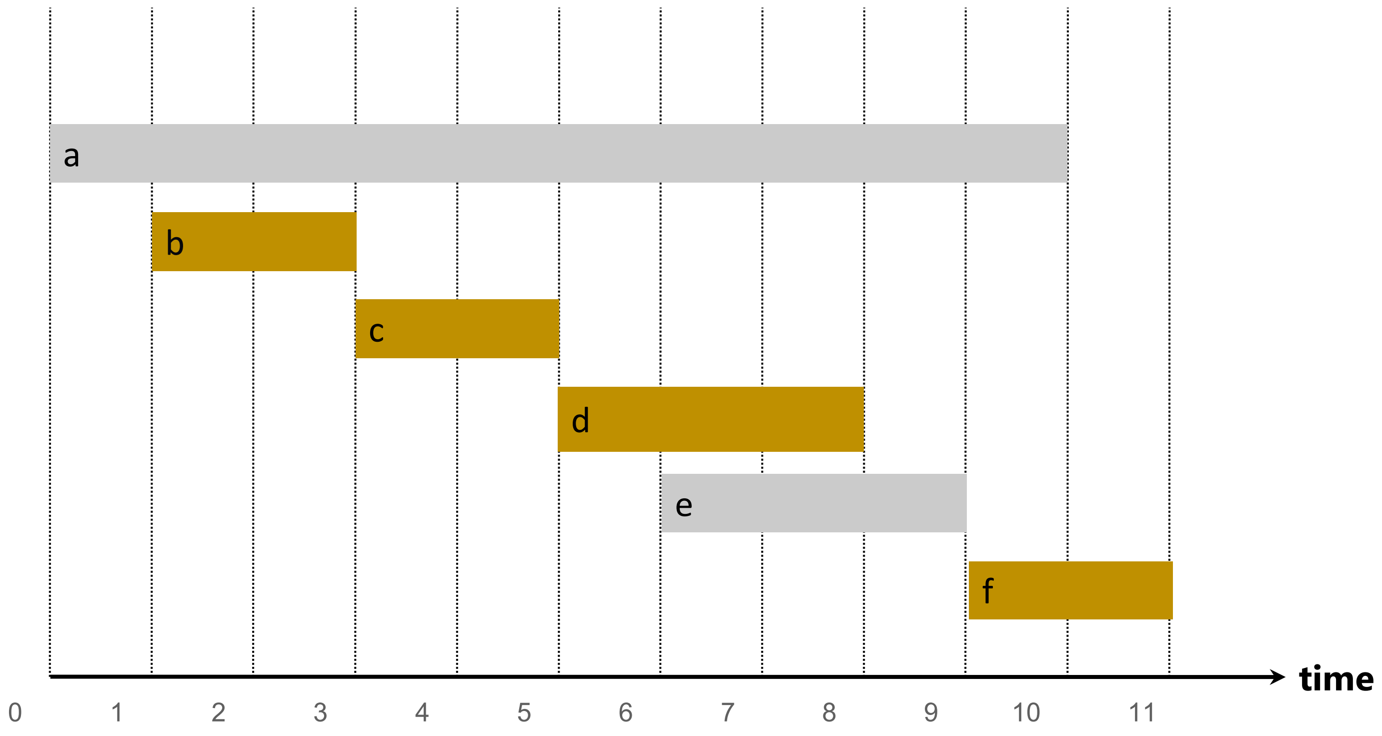
Interval Scheduling Problem

- **Earliest Finish Time First**
  - Select the shortest request
  - Eliminate any conflicting intervals
  - Continue this process until no more requests are left.



Interval Scheduling Problem

- **Earliest Finish Time First**
  - Select the shortest request
  - Eliminate any conflicting intervals
  - Continue this process until no more requests are left.



Interval Scheduling: EFTF Algorithm Analysis

EARLIEST-FINISH-TIME-FIRST $(n, s_1, s_2, \ldots, s_n, f_1, f_2, \ldots, f_n)$

SORT jobs by finish times and renumber so that $f_1 \leq f_2 \leq \ldots \leq f_n$.

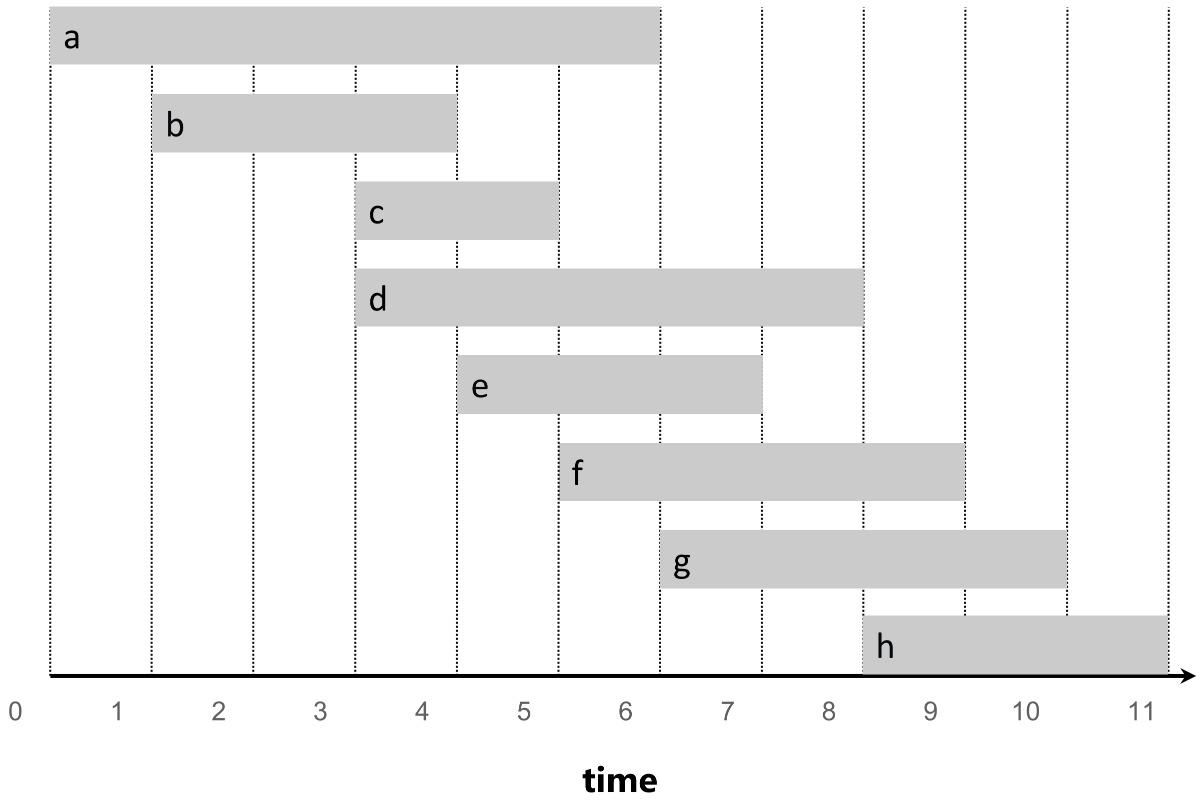
$S \leftarrow \varnothing$. ⟵ set of jobs selected

FOR $j = 1$ TO n

IF (*job j is compatible with S*)

$S \leftarrow S \cup \{ j \}$.

RETURN S.



Interval Scheduling: EFTF Algorithm Analysis

EARLIEST-FINISH-TIME-FIRST $(n, s_1, s_2, \ldots, s_n, f_1, f_2, \ldots, f_n)$

SORT jobs by finish times and renumber so that $f_1 \leq f_2 \leq \ldots \leq f_n$.

$Set \leftarrow \varnothing$. set of jobs selected

FOR $j = 1$ TO n

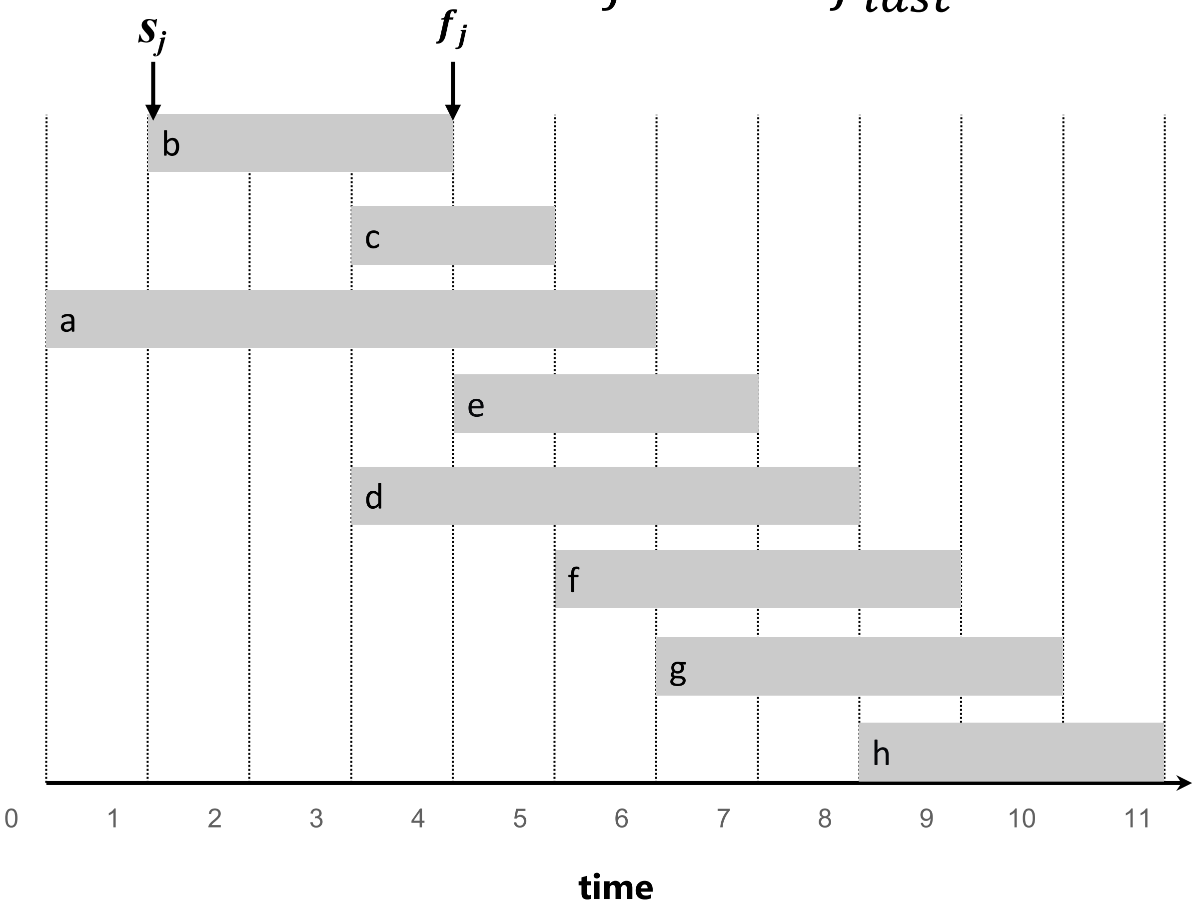
IF (job j is compatible with Set)

$Set \leftarrow Set \cup \{ j \}$.

RETURN S.

$$\boldsymbol{\Theta(n \log n) + \Theta(n)}$$

$$s_j \geq Set_{f_{last}}$$



Interval Scheduling: Correctness

Theorem: The earliest-finish-time-first (EFTF) algorithm is optimal.

Let A= $i_1, i_2, i_3, i_4, \ldots i_k$ ⟵ **EFTF**

Assume sorted: f(i_1) ≤ f(i_2) ≤ f(i_3) …

Let B= $j_1, j_2, j_3, j_4, \ldots j_k, j_{k+1}, \ldots j_m,$ ⟵ **Magic Optimal**

Assume sorted: f(j_1) ≤ f(j_2) ≤ f(j_3) …

where **m** > **k**

We need to show m ≤ k

Interval Scheduling: Correctness

Let A= $i_1, i_2, i_3, i_4, \dots i_k$

Assume sorted: f(i_1) ≤ f(i_2) ≤ f(i_3) …

Let B= $j_1, j_2, j_3, j_4, \dots j_k, j_{k+1}, \dots j_m,$

Assume sorted: f(j_1) ≤ f(j_2) ≤ f(j_3) …

First, we need to show that for each **r ≤ k**, f(i_r) ≥ f(j_r) [by Induction]

Then, we need to show that for each **m ≤ k**, [by Contradiction]

Interval Scheduling: Correctness

Proof: [by Induction]

Base Case:

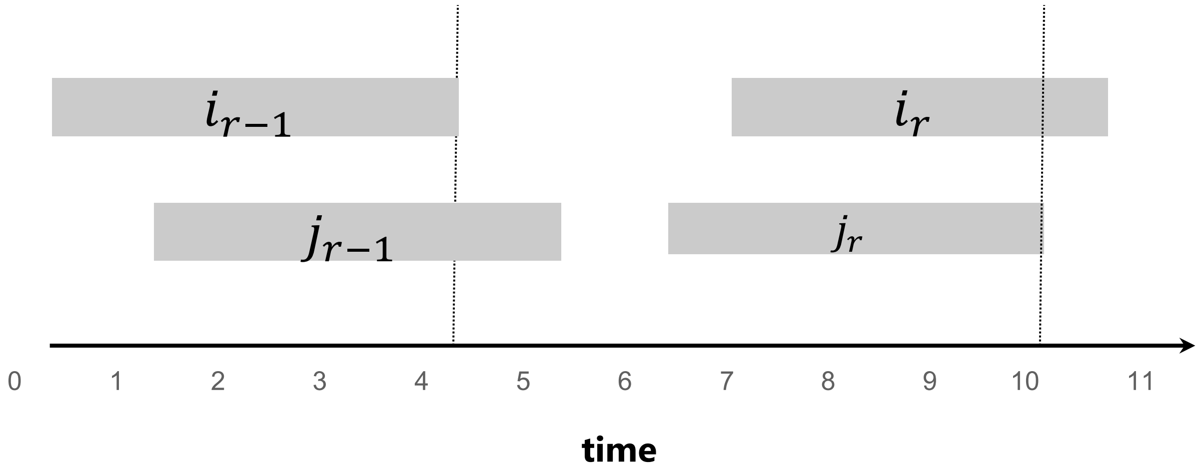
- r=1: EFTF chooses booking i_1 with earliest overall finish time, i.e., $\mathbf{f}(\boldsymbol{i_1}) \leq \mathbf{f}(\boldsymbol{j_1})$

Inductive Hypothesis:

- r> 1: Assume, by induction that $\mathbf{f}(\boldsymbol{i_{r-1}}) \leq \mathbf{f}(\boldsymbol{j_{r-1}})$

Then

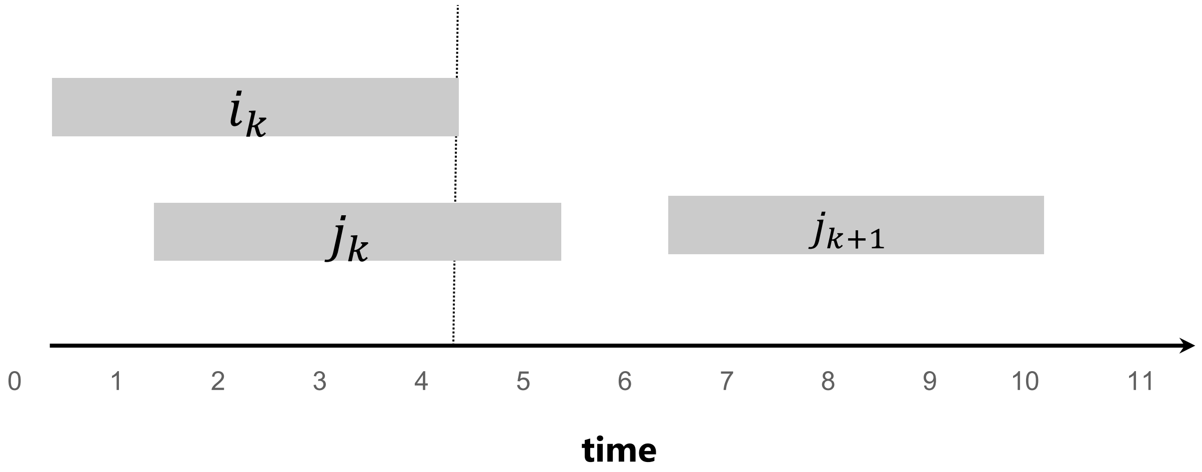
- It must be the case that $\mathbf{f}(\boldsymbol{i_r}) \leq \mathbf{f}(\boldsymbol{j_r})$



Interval Scheduling: Correctness

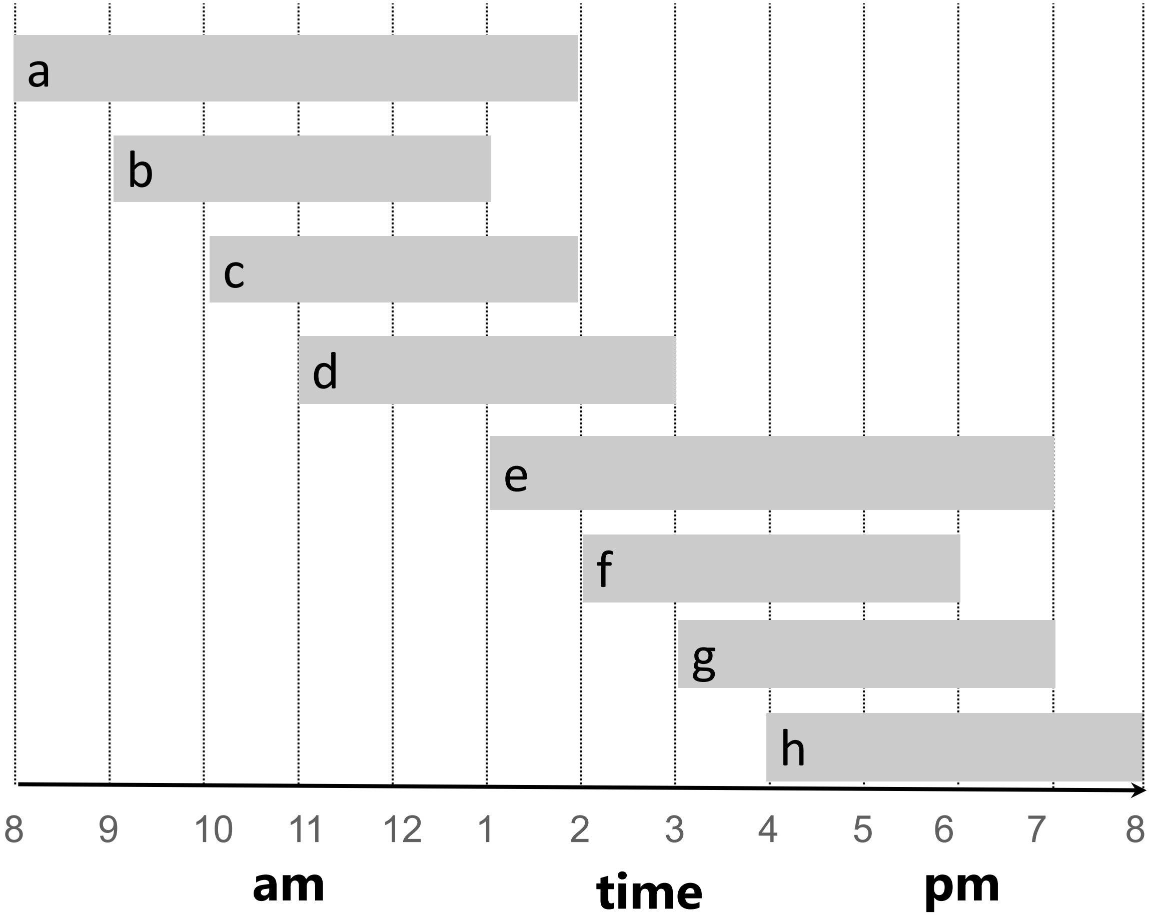
Proof: [by Contradiction]

- We know that $\mathbf{f}(\boldsymbol{i_r}) \leq \mathbf{f}(\boldsymbol{j_r})$
- Consider j_{k+1} in “Magic Optimal”
- Greedy algorithms terminates only when no more jobs are left or all remaining jobs overlaps
- **Contradiction**

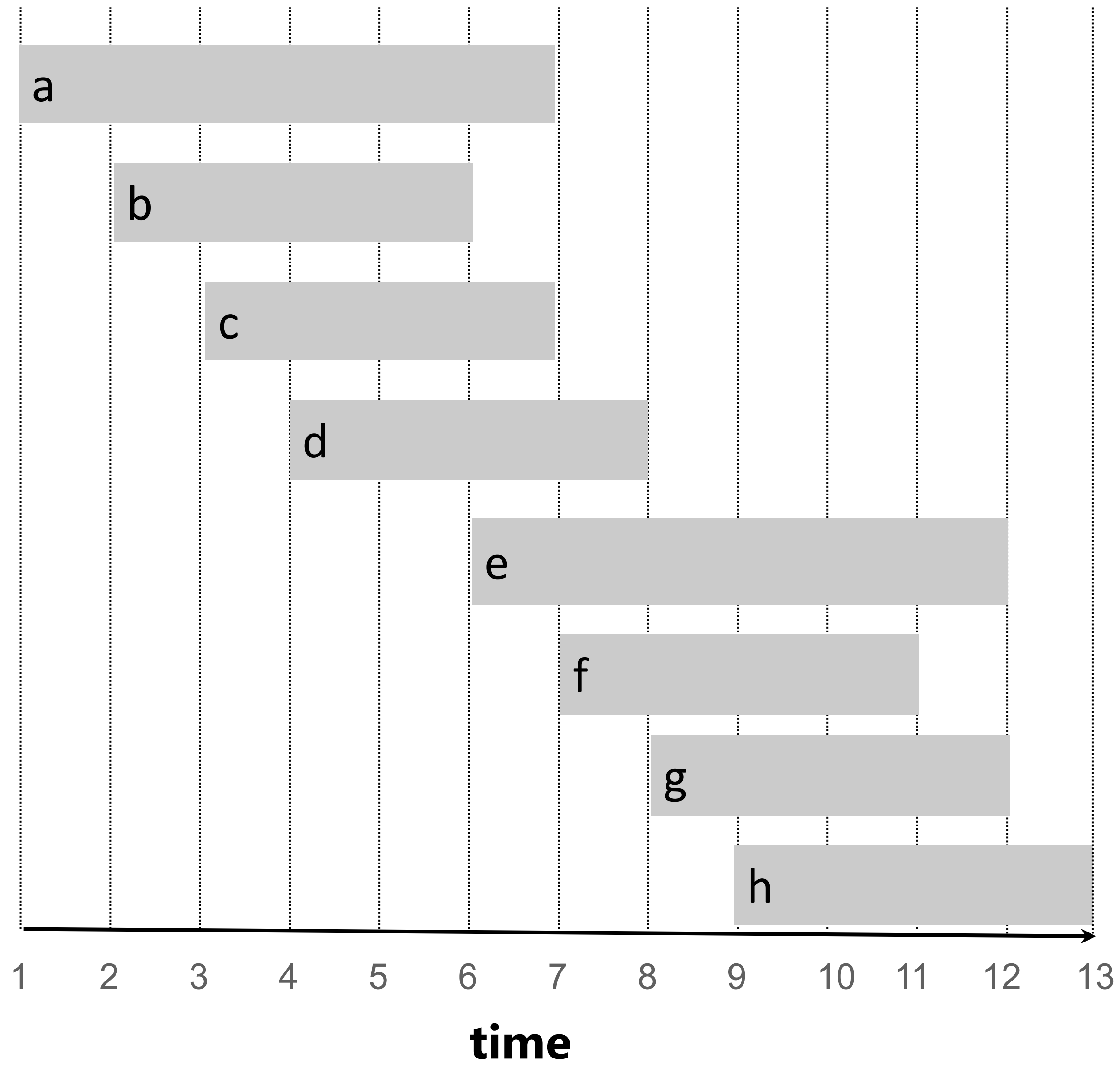


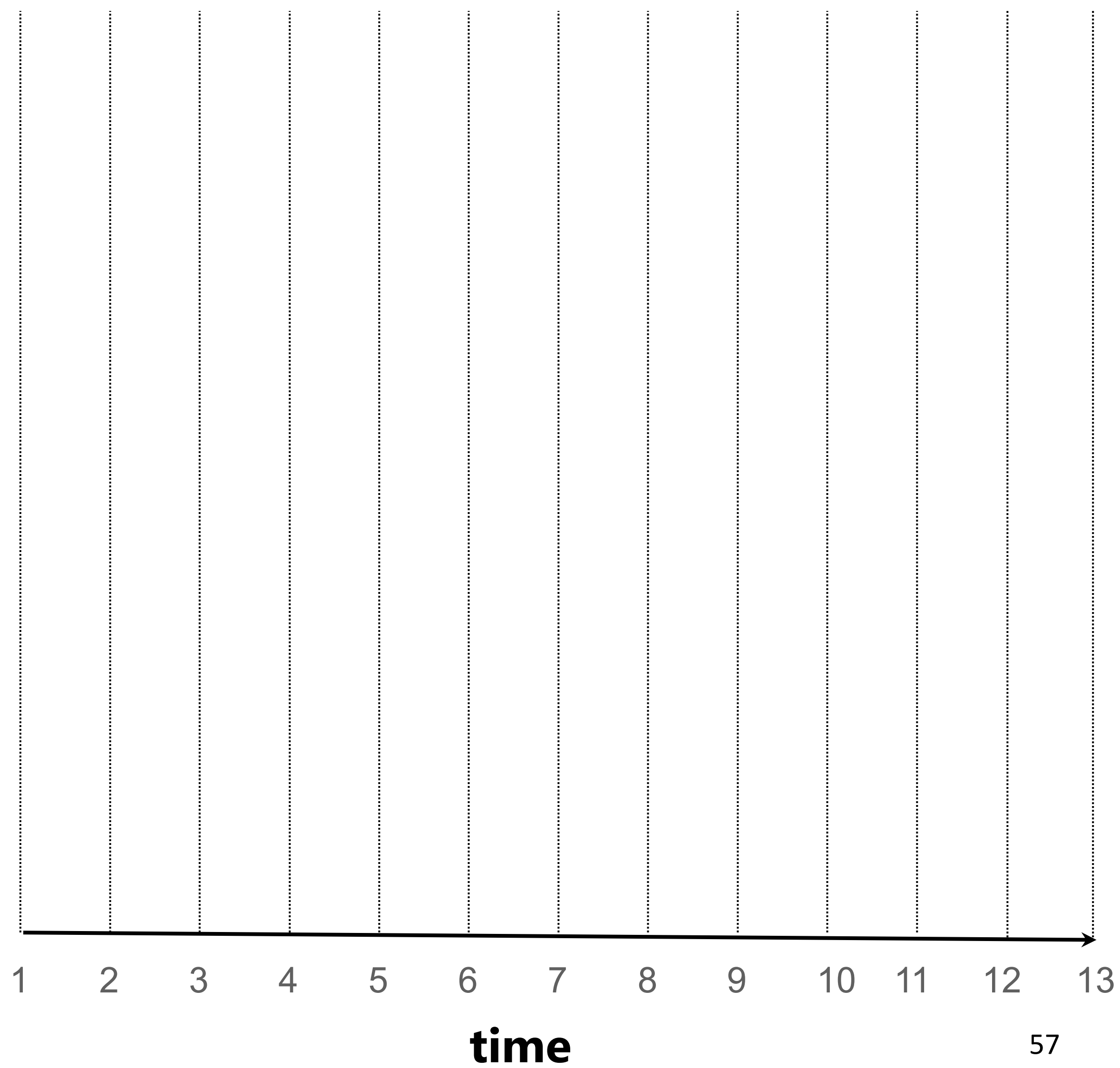
Interval partitioning

- Lecture j starts at s_j and finishes at f_j.
- **Goal:** find **minimum** number of classrooms to schedule all lectures so that no two lectures occur at the same time in the same room.

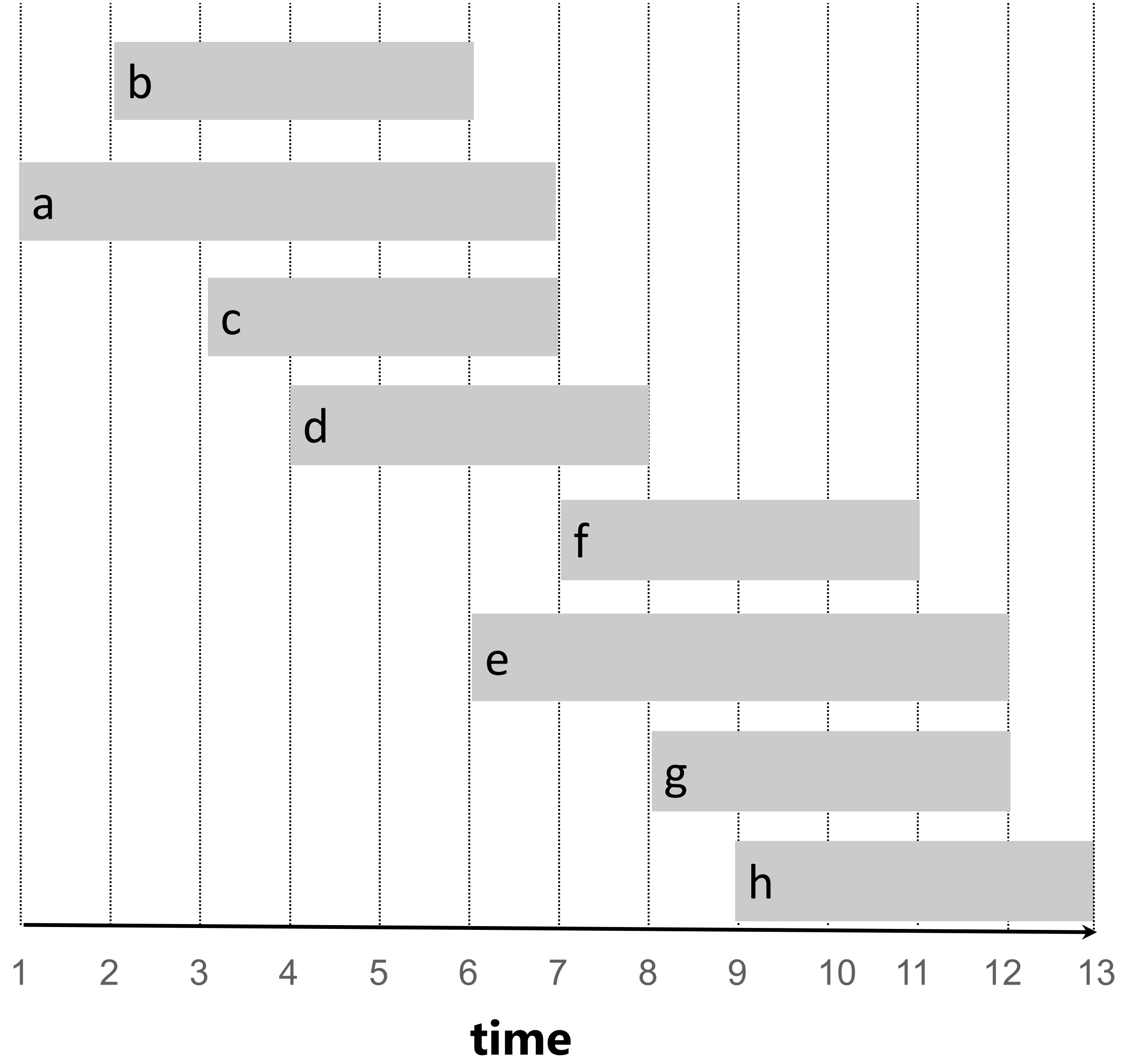


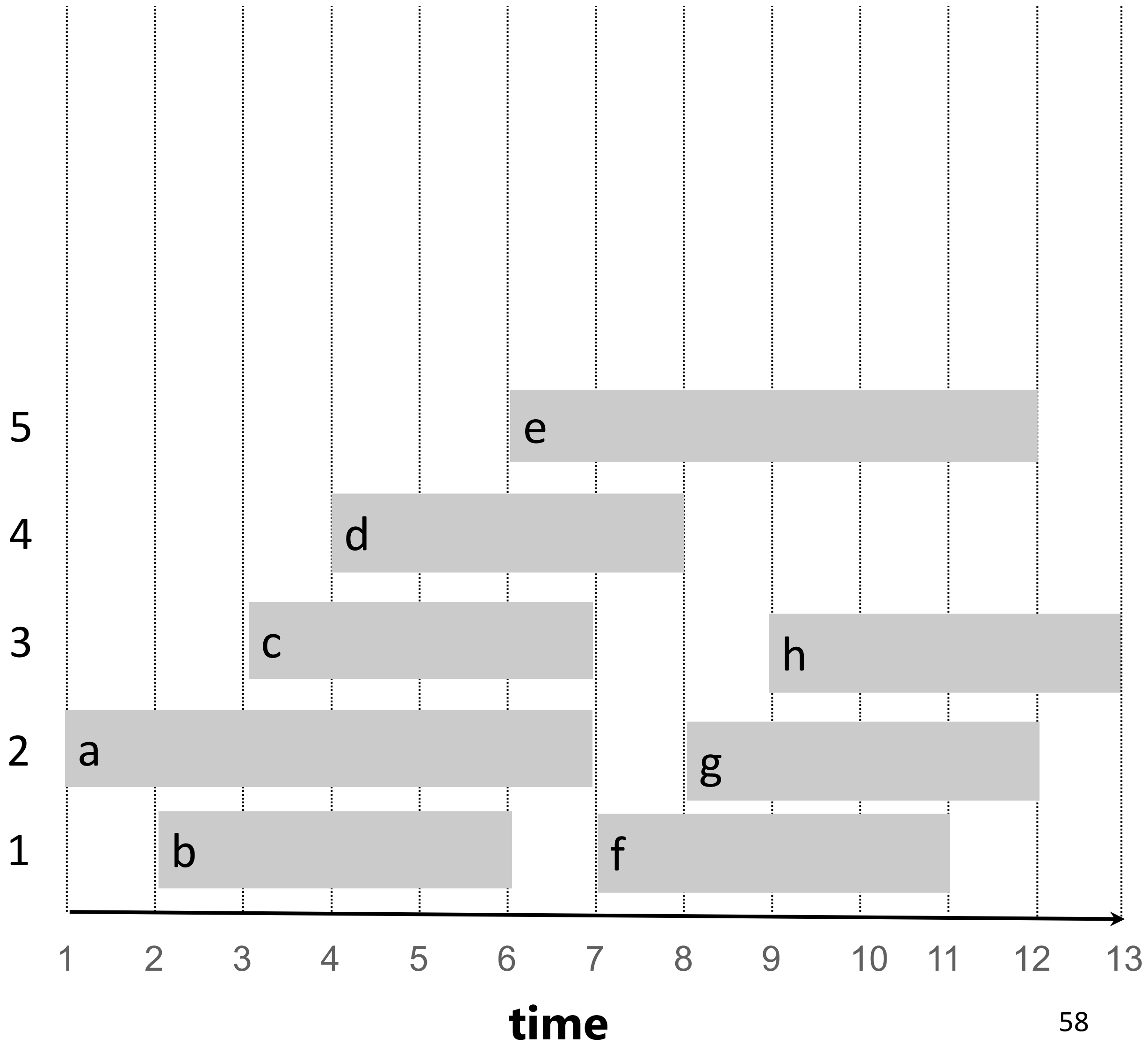
Earliest Finish Time First



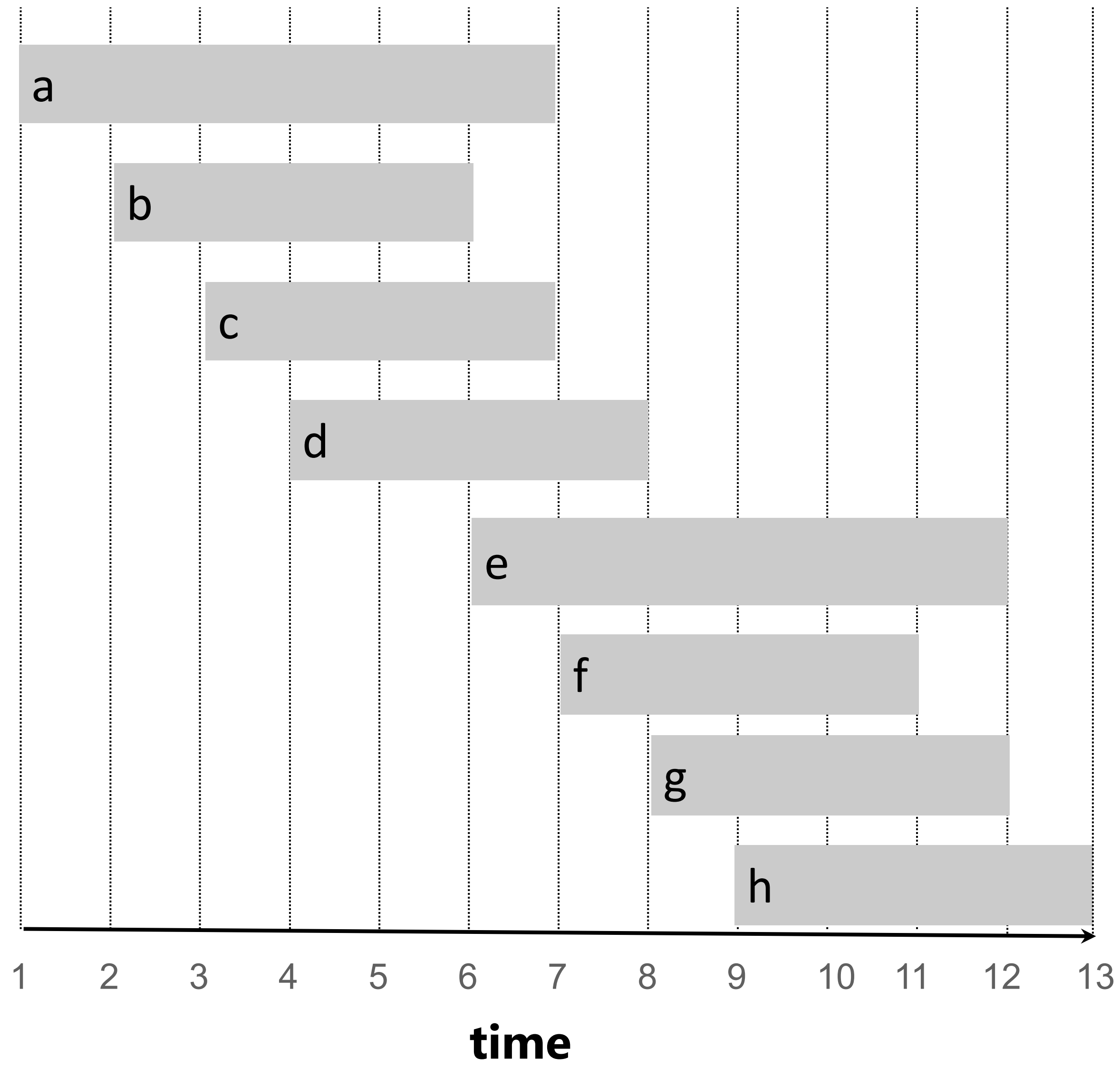


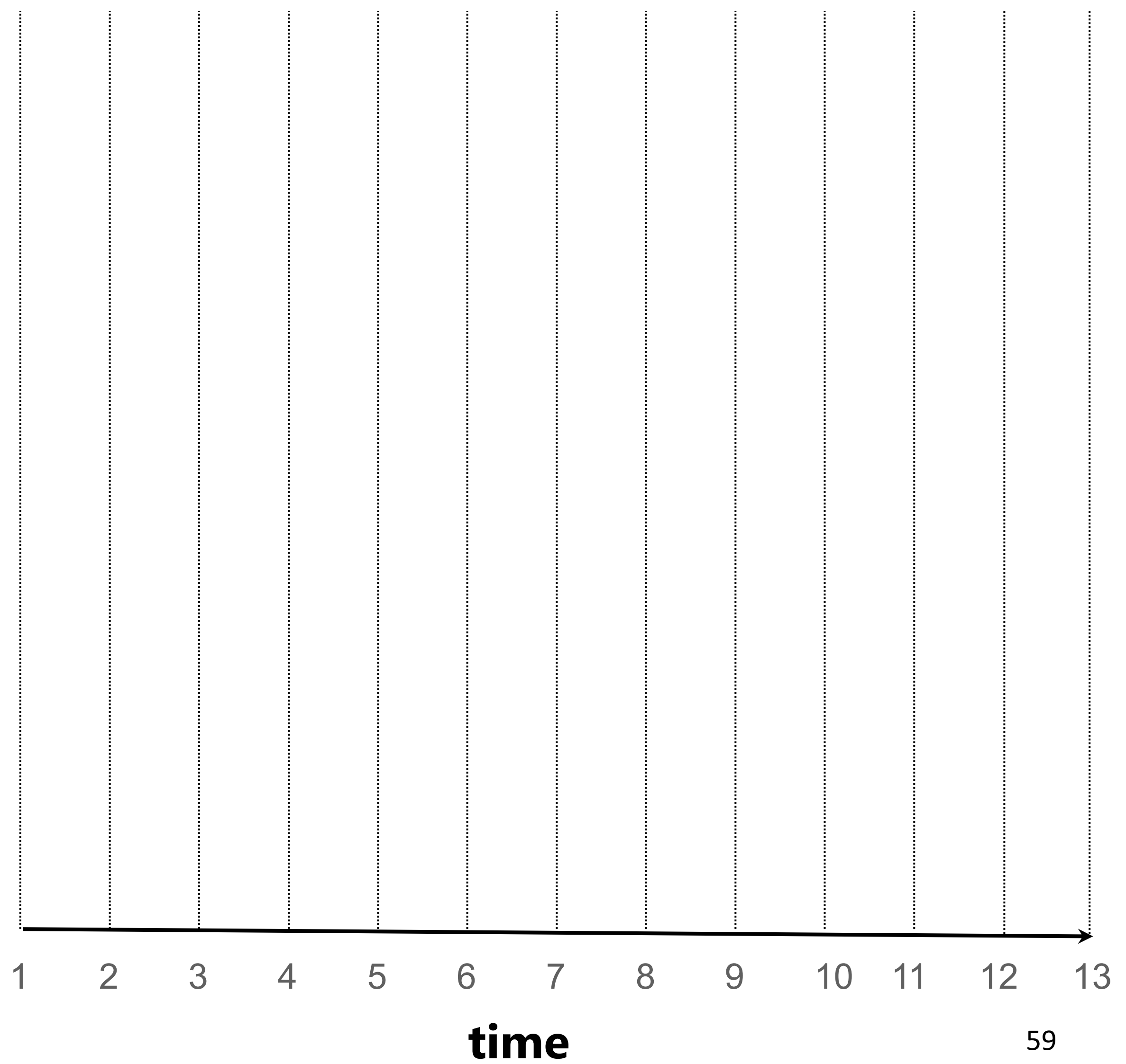
Earliest Finish Time First



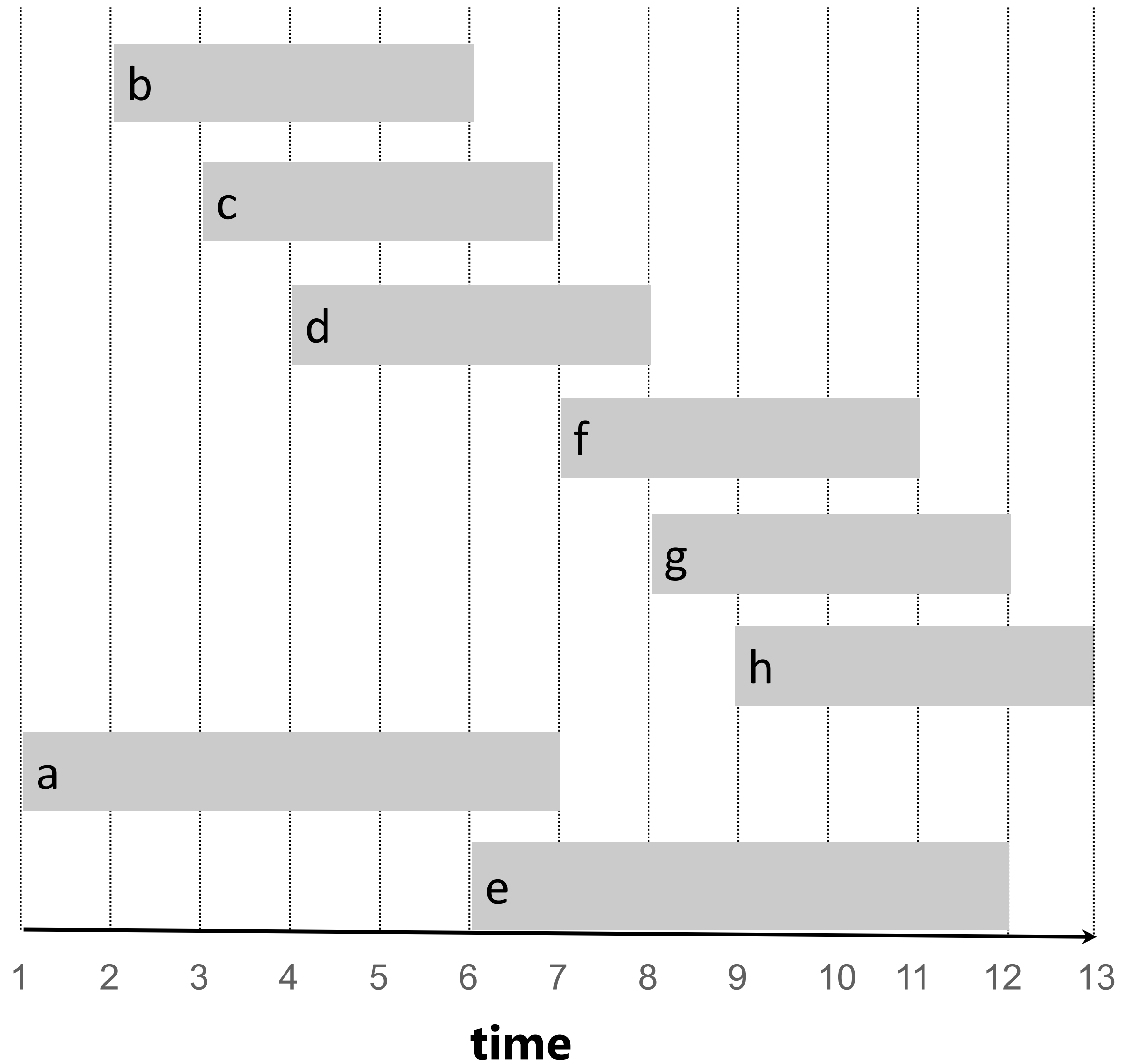


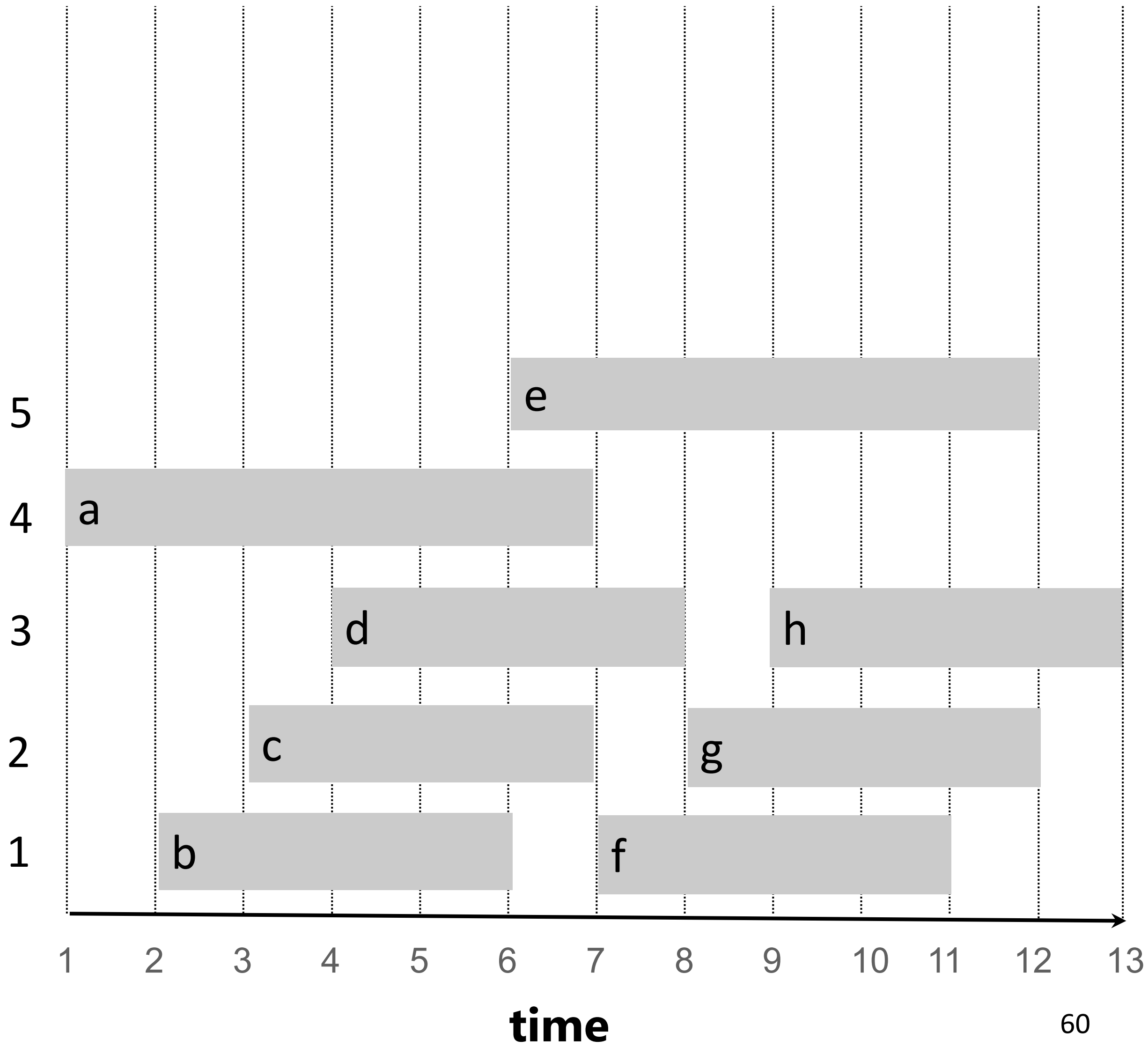
Smallest Interval First



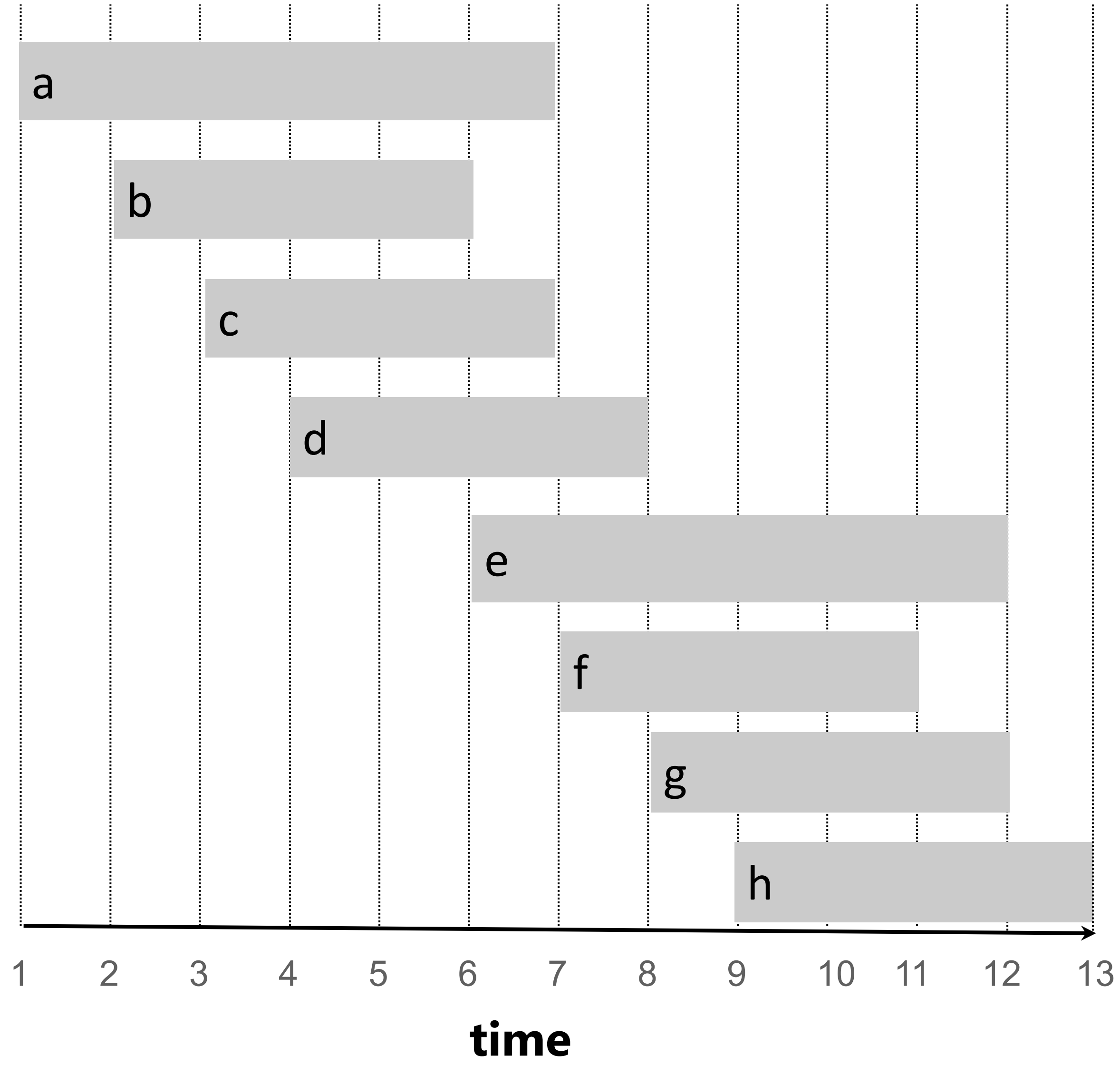


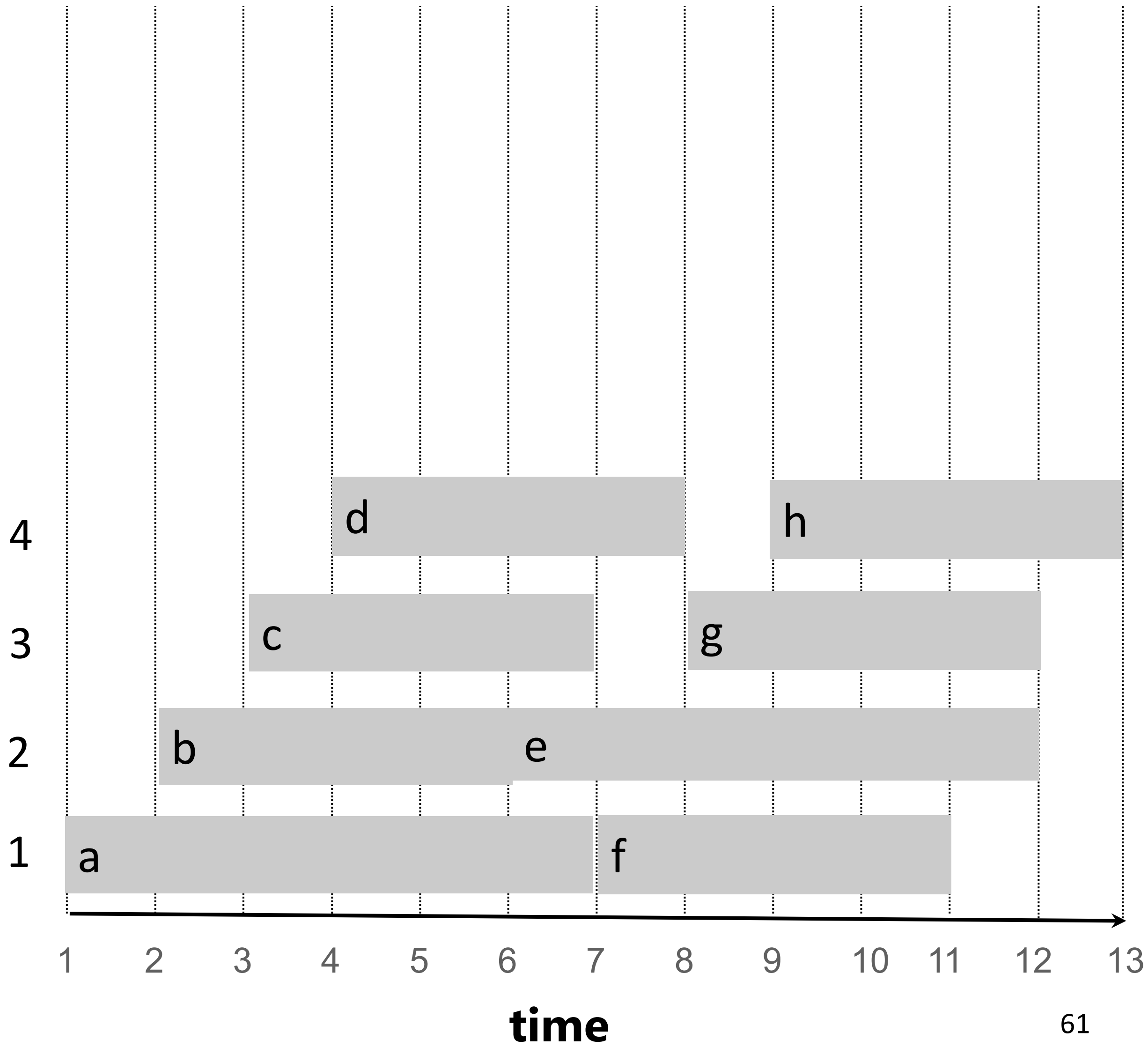
Smallest Interval First





Earliest Start Time First





Interval Partitioning: ESTF Algorithm Analysis

EARLIEST-START-TIME-FIRST $(n, s_1, s_2, \ldots, s_n, f_1, f_2, \ldots, f_n)$

SORT lectures by start times and renumber so that $s_1 \leq s_2 \leq \ldots \leq s_n$. $\boldsymbol{\Theta(n \log n)}$

$d \leftarrow 1$ ⟵ number of allocated classrooms

FOR $j = 1$ TO n

  IF (lecture j is compatible with some classroom $k \in \{1, 2, .., d\}$)

    Schedule lecture j in any such classroom k

  ELSE

    Allocate a new classroom $d + 1$.

    Schedule lecture j in classroom $d + 1$.

    $d \leftarrow d + 1$.

RETURN schedule.

$\boldsymbol{\Theta(n)}$

$\boldsymbol{\Theta(\log n)}$

$\boldsymbol{\Theta(n \log n)}$

Interval Partitioning: ESTF Algorithm Analysis

Proposition: The earliest-start-time-first algorithm can be implemented in $O(n \log n)$ time.

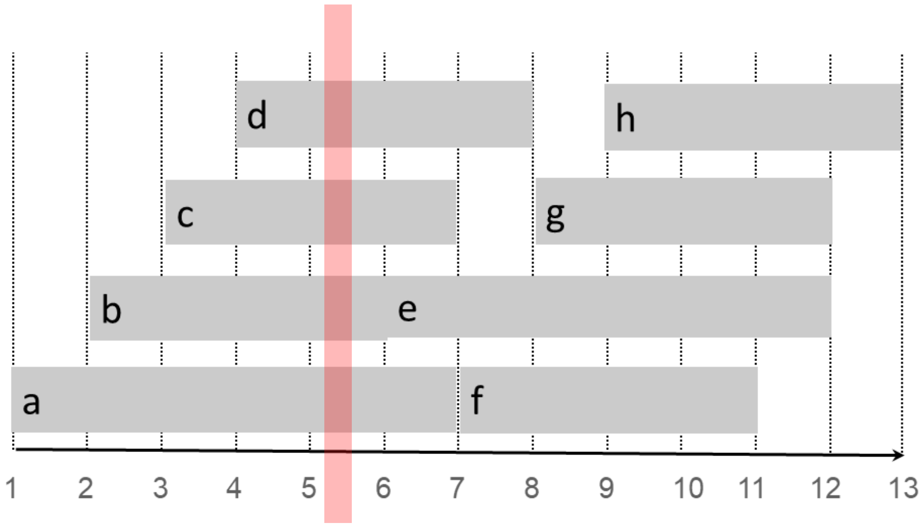
Proof:

- Sorting by start times takes $O(n \log n)$ time.
- Store classrooms in a priority queue (key = finish time of its last lecture).
  - to allocate a new classroom, INSERT classroom onto priority queue.
  - to schedule lecture j in classroom k, INCREASE-KEY of classroom k to f_j.
  - to determine whether lecture j is compatible with any classroom, compare s_j to FIND-MIN
- Total # of priority queue operations is $O(n)$; each takes $O(\log n)$ time. ▪

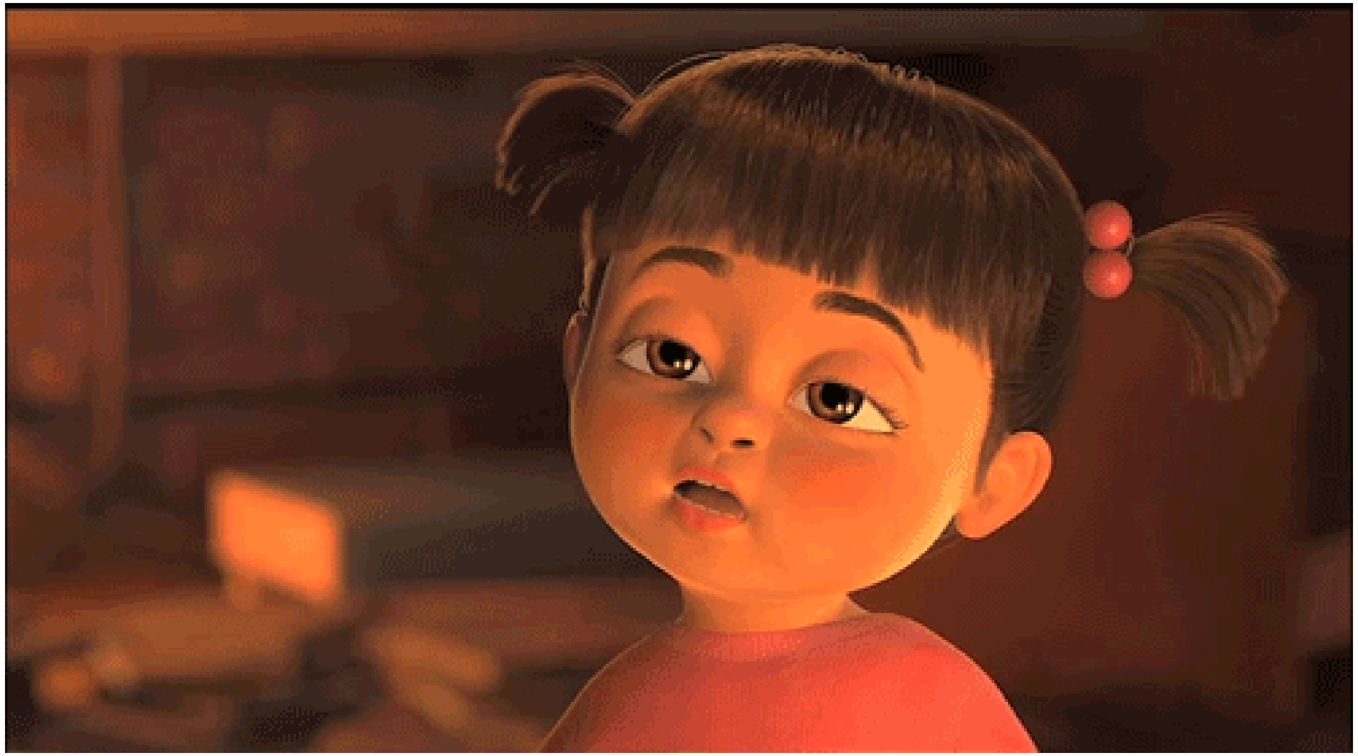
Remark: This implementation chooses a classroom k whose finish time of its last lecture is the earliest.

Interval Partitioning: ESTF Algorithm Analysis

- **Def:** The depth of a set of open intervals is the maximum number of intervals that contain any given point.
- **Key observation:** Number of classrooms needed $\geq$ depth.
- **Q.** Does minimum number of classrooms needed always equal depth?
- **A.** Yes! Moreover, earliest-start-time-first algorithm finds a schedule whose number of classrooms equals the depth.



Thanks a lot



If you are taking a Nap, **wake up**........Lecture Over