COMP 3011
DESIGN AND ANALYSIS OF ALGORITHMS
FALL 2024

# Greedy Algorithms

LI Bo
Department of Computing
The Hong Kong Polytechnic University

# GREEDY ALGORITHMS

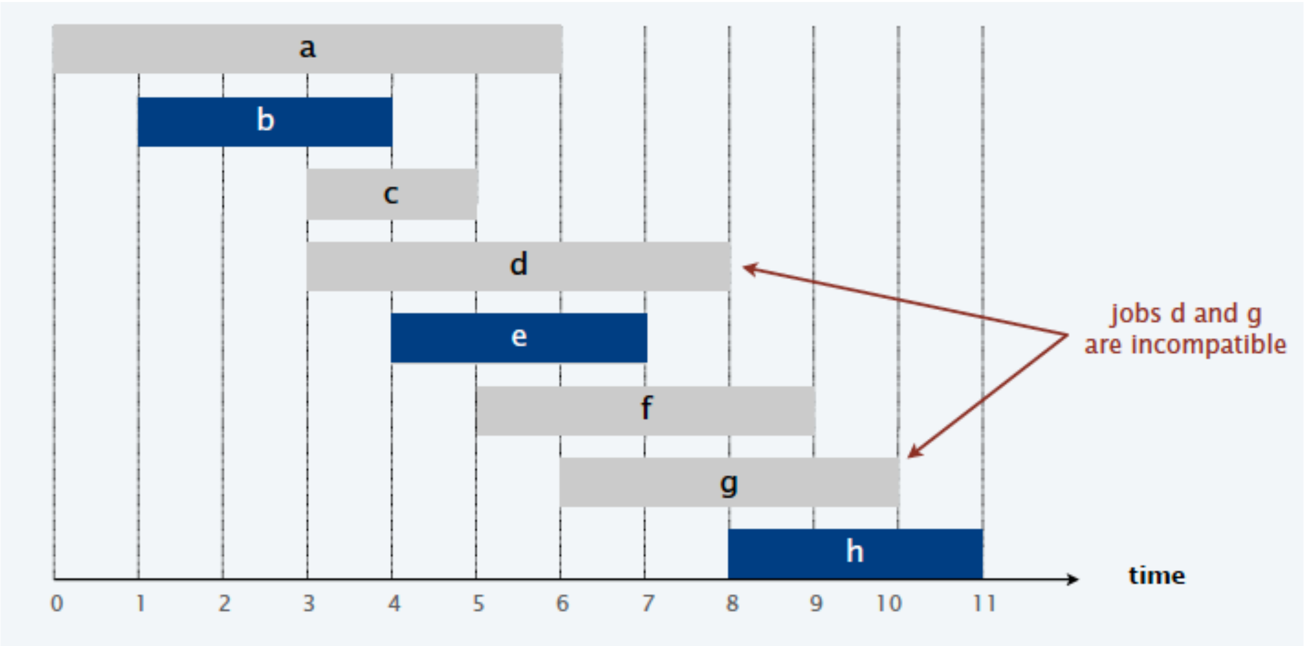
# INTERVAL SCHEDULING PROBLEM

# INTERVAL SCHEDULING PROBLEM


Given a set of jobs $J = \{1, 2, \ldots, n\}$

- Job $j$ starts at $s_j$ and finishes at $f_j \geq s_j$.
- Two jobs (open intervals) are compatible if they don't overlap.

***Goal***: find maximum subset of mutually compatible jobs.




***Intuition:*** shorter is better

# INTERVAL SCHEDULING PROBLEM

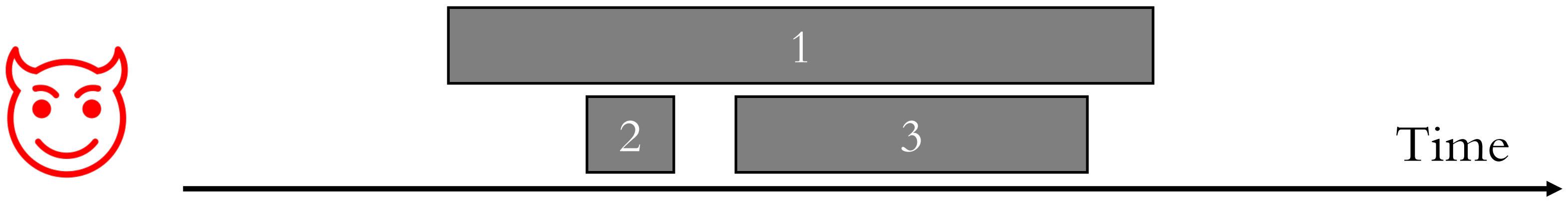
***Idea 1:***

- Repeatedly pick shortest compatible, unscheduled job (i.e. that does not conflict with any scheduled job).


***Idea 2:***

***Intuition:*** earlier is better

- Repeatedly pick compatible job with earliest starting time.



# GREEDY ALGORITHM

- Repeatedly pick an item until no more feasible choices.
- Among all feasible choices, we always pick the one that minimizes (or maximizes) ***some property***.
  - length, starting time, …
- Such algorithms are called ***greedy***.

- Greedy algorithms may not be optimal.
- But maybe we have been using the wrong property!

# INTERVAL SCHEDULING PROBLEM

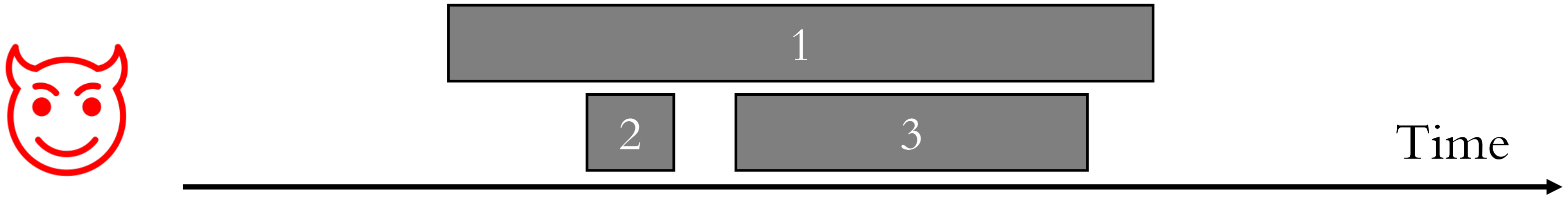
***What about earliest-finish-time-first?***

***Idea 1:***

- Repeatedly pick shortest compatible, unscheduled job (i.e. that does not conflict with any scheduled job).


***Idea 2:***

- Repeatedly pick compatible job with earliest starting time.



# EARLIEST-FINISH-TIME-FIRST ALGORITHM

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 $S$.

---

***Proposition***. Can implement earliest-finish-time first in $O(n \log n)$ time.

# EARLIEST-FINISH-TIME-FIRST ALGORITHM

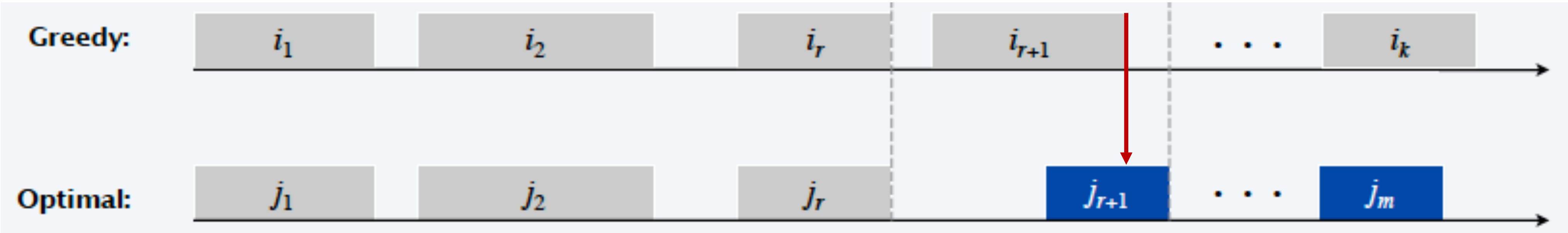
***Theorem***. The earliest-finish-time-first algorithm is optimal.

Switching $j_{r+1}$ by $i_{r+1}$ in $O$:
Still ***feasible*** and ***optimal***!

***Proof***. [by contradiction]

- Assume Greedy is not optimal.
- Let $A = \{i_1, i_2, \dots, i_k\}$ be set of jobs selected by Greedy.
- Let $O = \{j_1, j_2, \dots, j_m\}$ be set of jobs in an optimal solution. Then $m > k$.
- Let $r + 1$ be first index such that $i_{r+1} \neq j_{r+1}$.

such a job exists ➡ $f_{i_{r+1}} \leq f_{j_{r+1}}$




$i_1 = j_1$ $\quad$ $i_2 = j_2$ $\quad$ $i_r = j_r$ $\quad$ $i_{r+1} \neq j_{r+1}$

# INTERVAL PARTITIONING

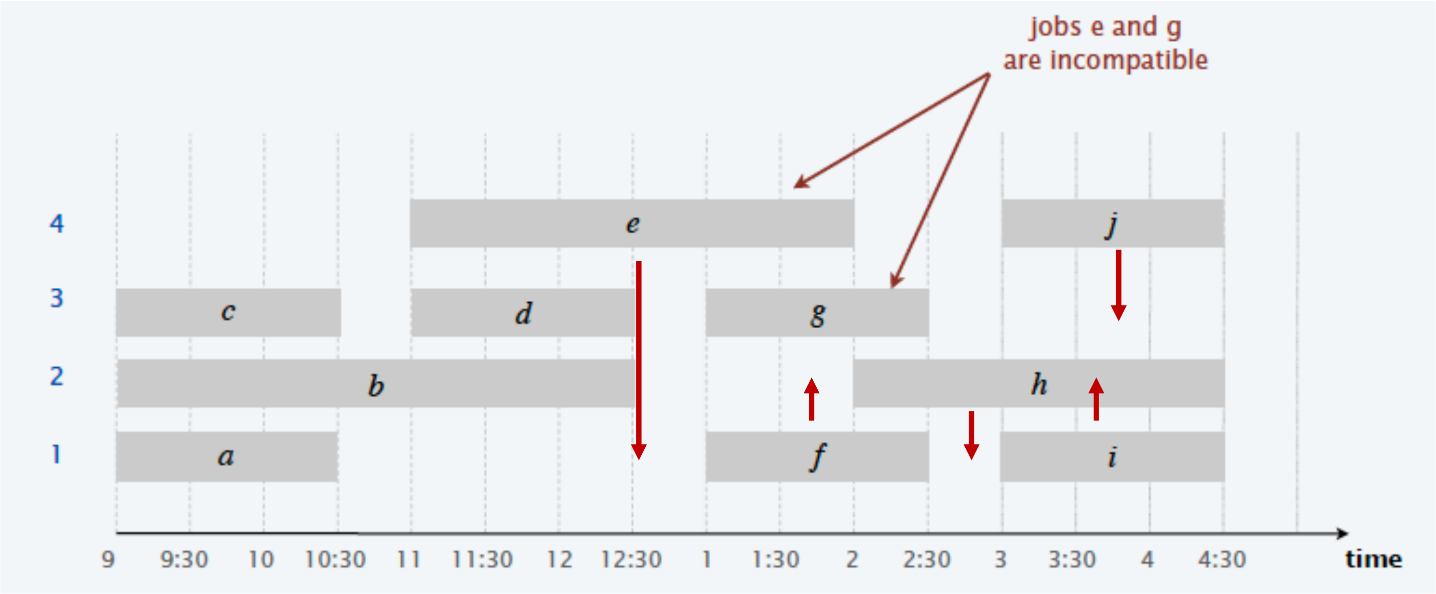
# INTERVAL PARTITIONING

Given a set of lectures (jobs) $L = \{1, 2, \ldots, n\}$;

- Lecture $j$ starts at $s_j$ and finishes at $f_j \geq s_j$.
- Two lectures are compatible if they don't overlap.

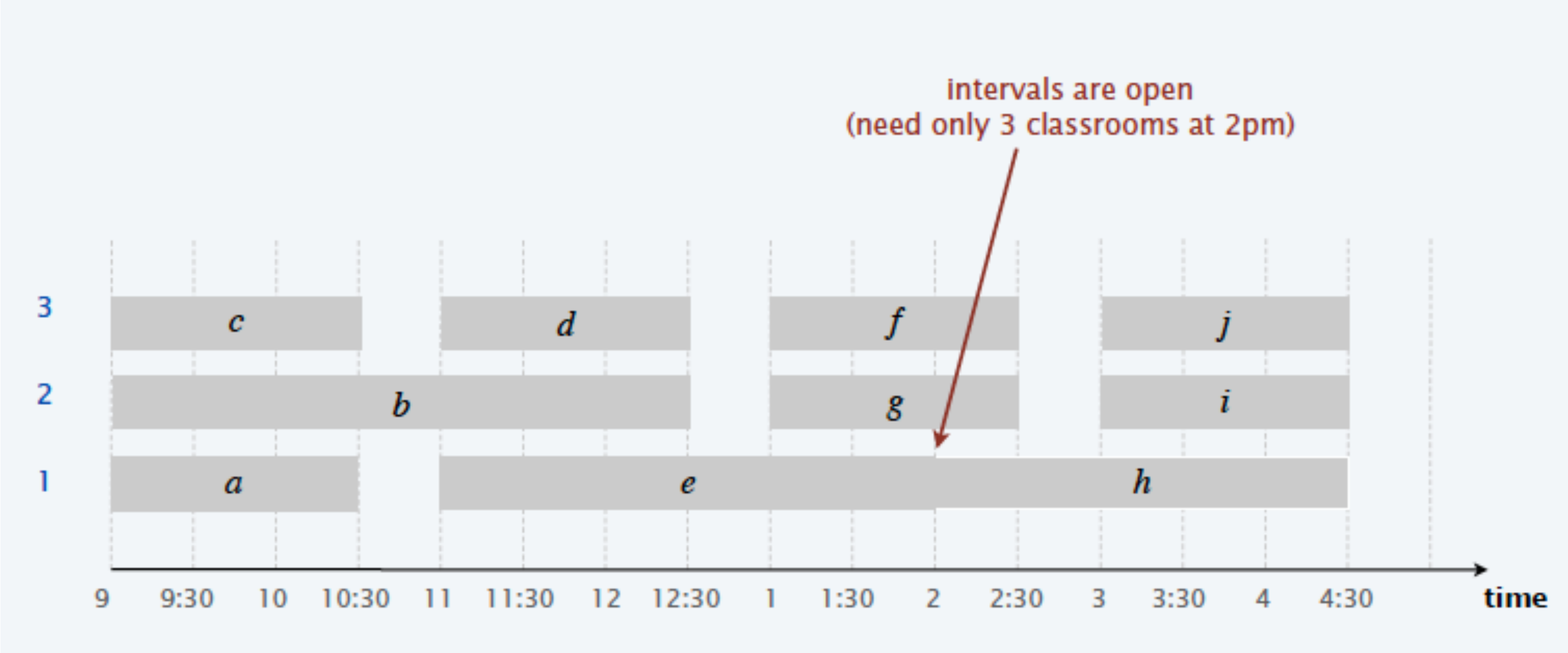

***Goal***: find minimum number of classrooms to schedule all lectures so that no two lectures occur at the same time in the same room





# INTERVAL PARTITIONING

- Optimal is 3 classrooms.



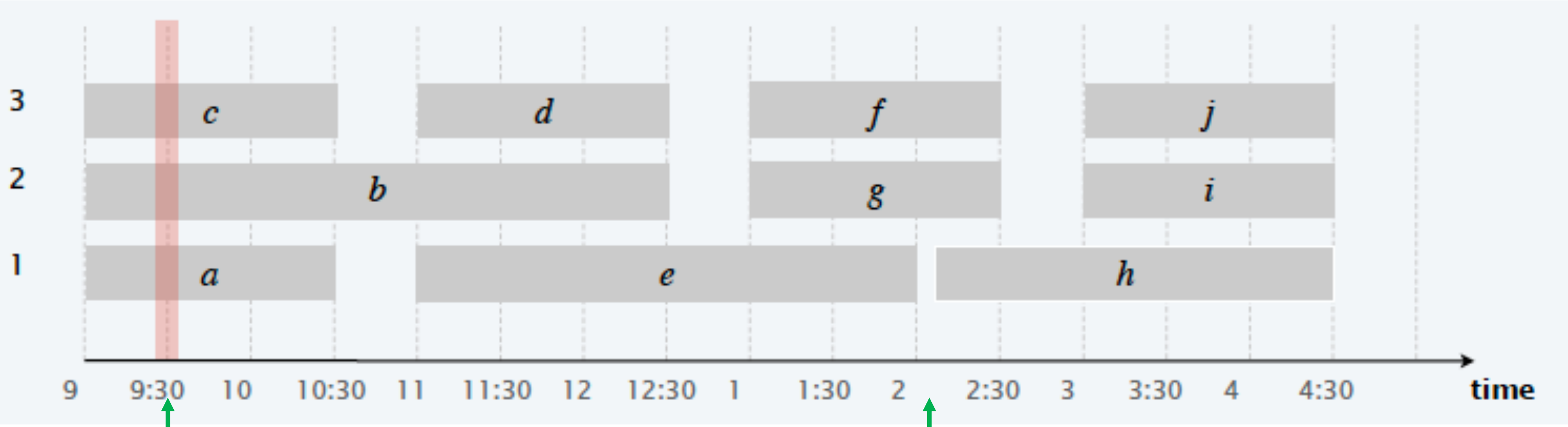
# INTERVAL PARTITIONING

***Definition***. The depth of a set of open intervals is the maximum number of intervals that contain any given point.

***Key observation***. #rooms needed ≥ depth.

Is depth enough???



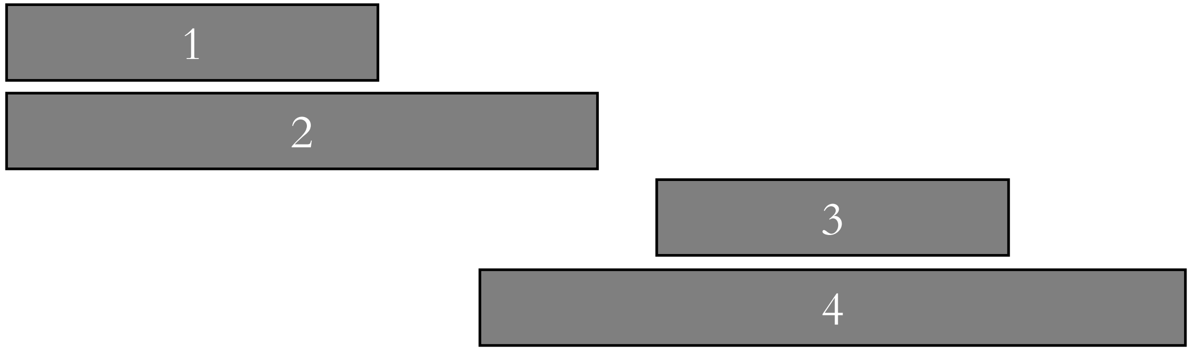


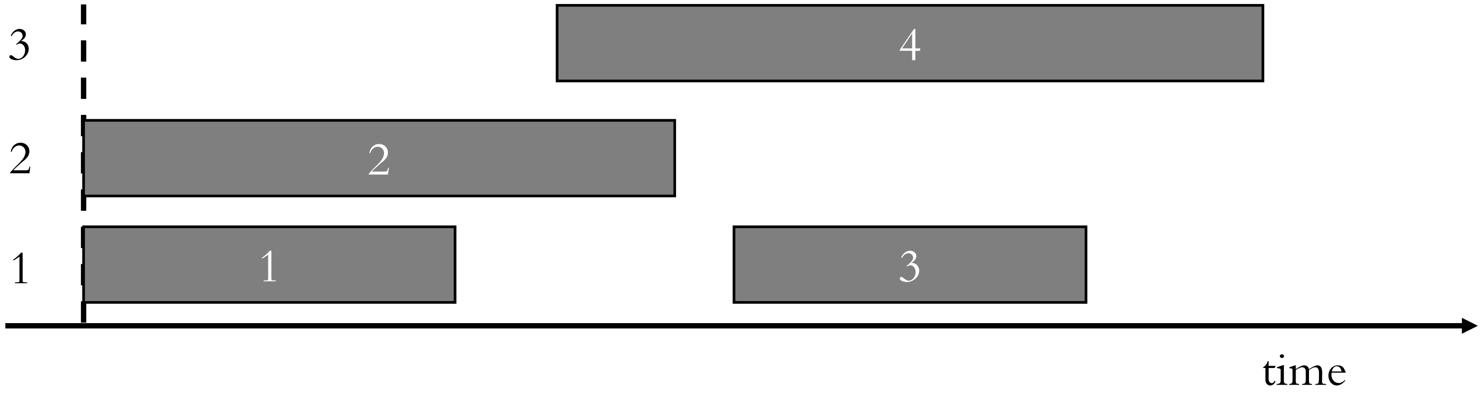

3 classrooms are needed

2 classrooms are needed

# INTERVAL PARTITIONING

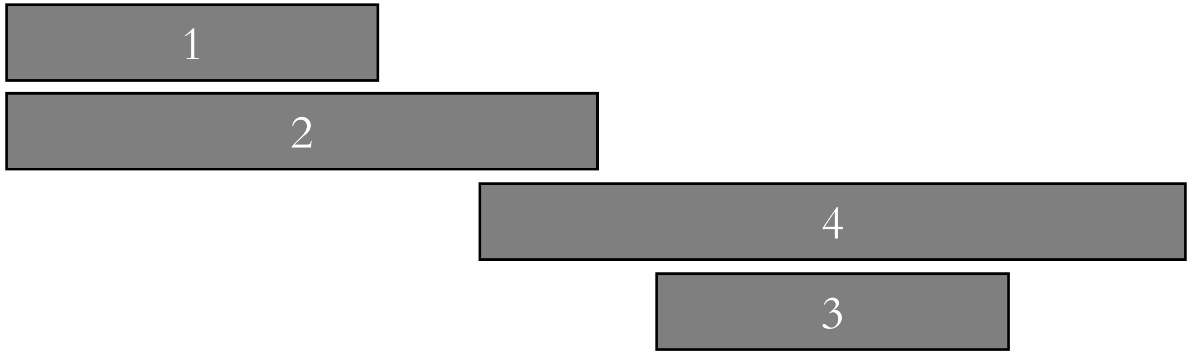
Can we do earliest-finish-time-first?

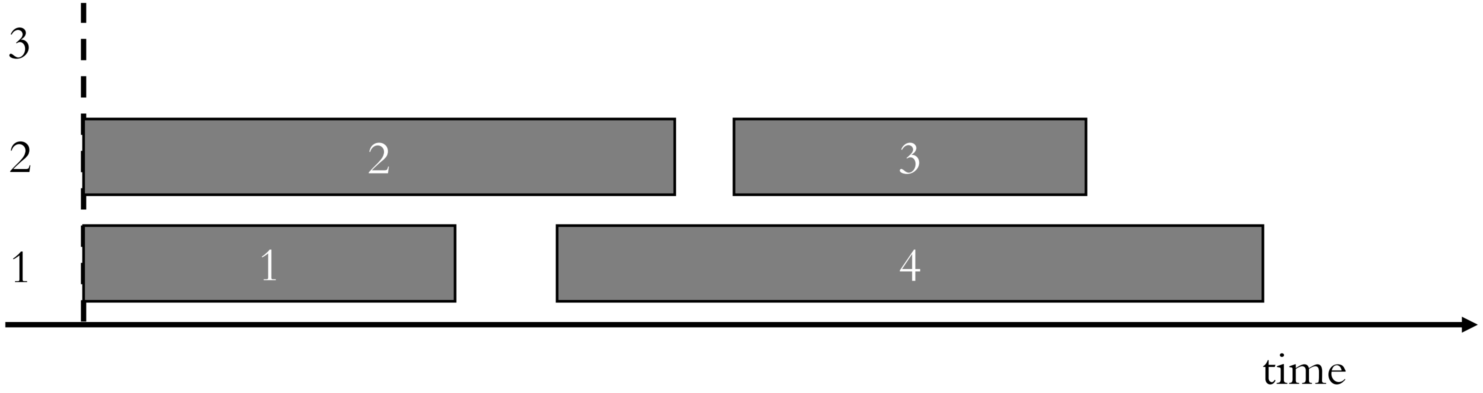


# INTERVAL PARTITIONING

Can we do earliest-start-time-first?



# INTERVAL PARTITIONING: EARLIEST-START-TIME-FIRST ALGORITHM

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 $s_1 \leq s_2 \leq \ldots \leq s_n$.

$d \leftarrow 0$. ⟵ number of allocated classrooms

FOR $j = 1$ TO $n$

  IF (lecture $j$ is compatible with some classroom)

    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.

***Lemma*.**

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

***Lemma*.**

The earliest-start-time first algorithm never schedules two incompatible lectures in the same classroom.

# INTERVAL PARTITIONING: EARLIEST-START-TIME-FIRST ALGORITHM

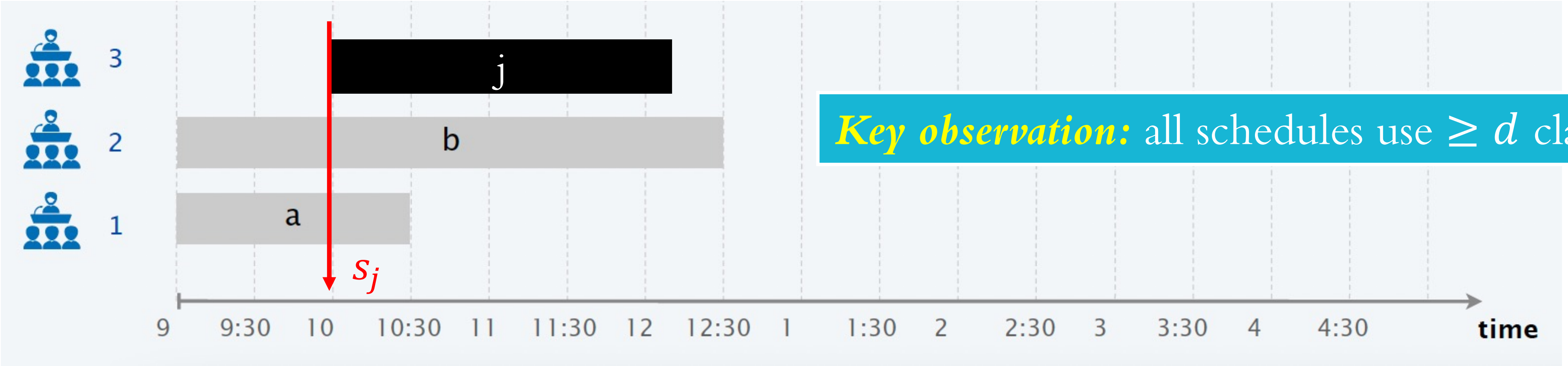
***Theorem***. Earliest-start-time-first algorithm uses #depth rooms and thus is optimal.

- Let $d$ = number of classrooms that the algorithm allocates.
- Classroom $d$ is opened because we needed to schedule a lecture, say $j$, that is incompatible with a lecture in each of $d-1$ other classrooms.
- Thus, these $d$ lectures each end after $s_j$.
- Since we sorted by start time, each of these incompatible lectures start no later than $s_j$. ■

The $d$ lectures are incompatible.



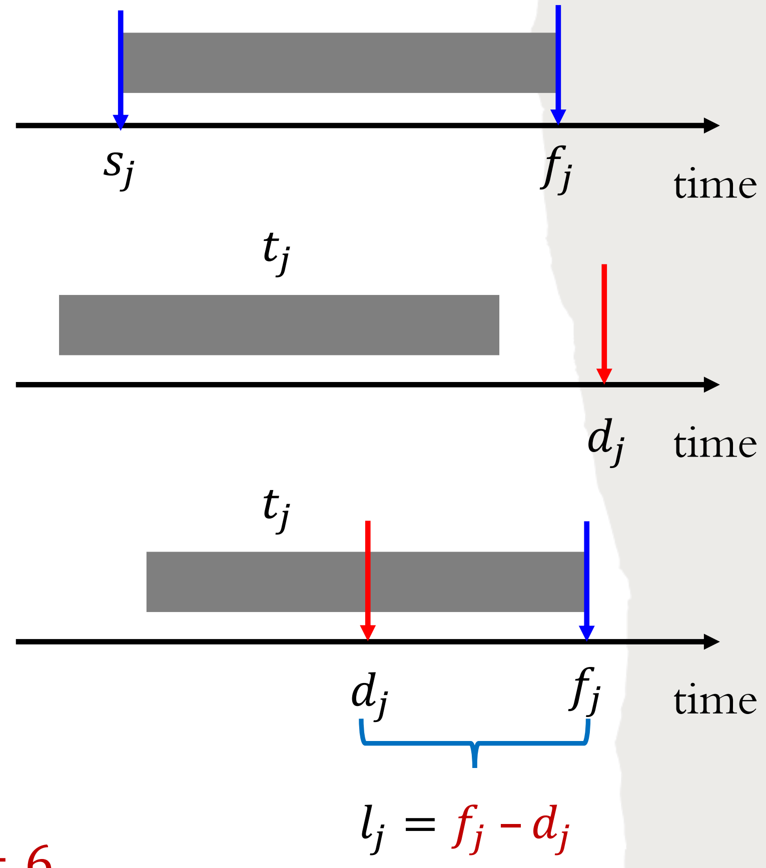
$d-1$




***Key observation:*** all schedules use $\geq d$ classrooms.

# SCHEDULING TO MINIMIZING LATENESS

# SCHEDULING TO MINIMIZING LATENESS

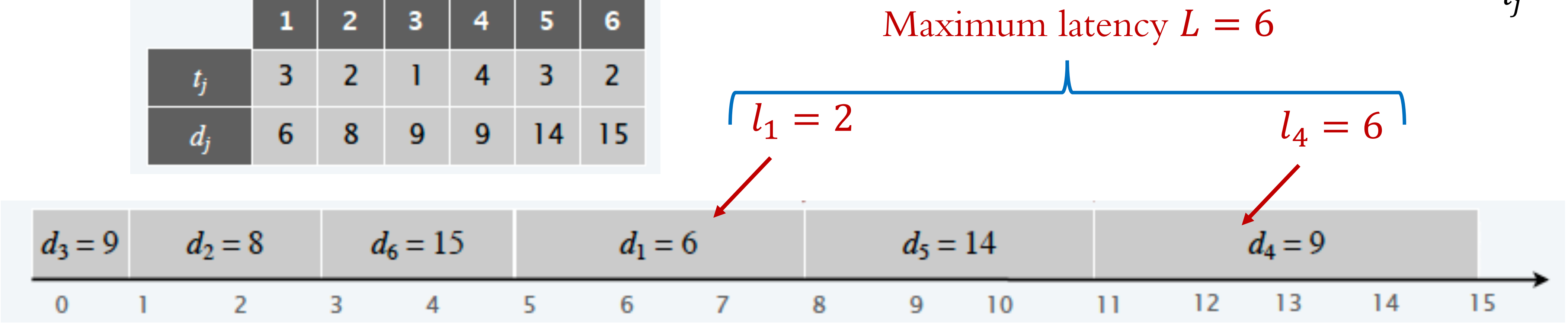



Single resource processes one job at a time.

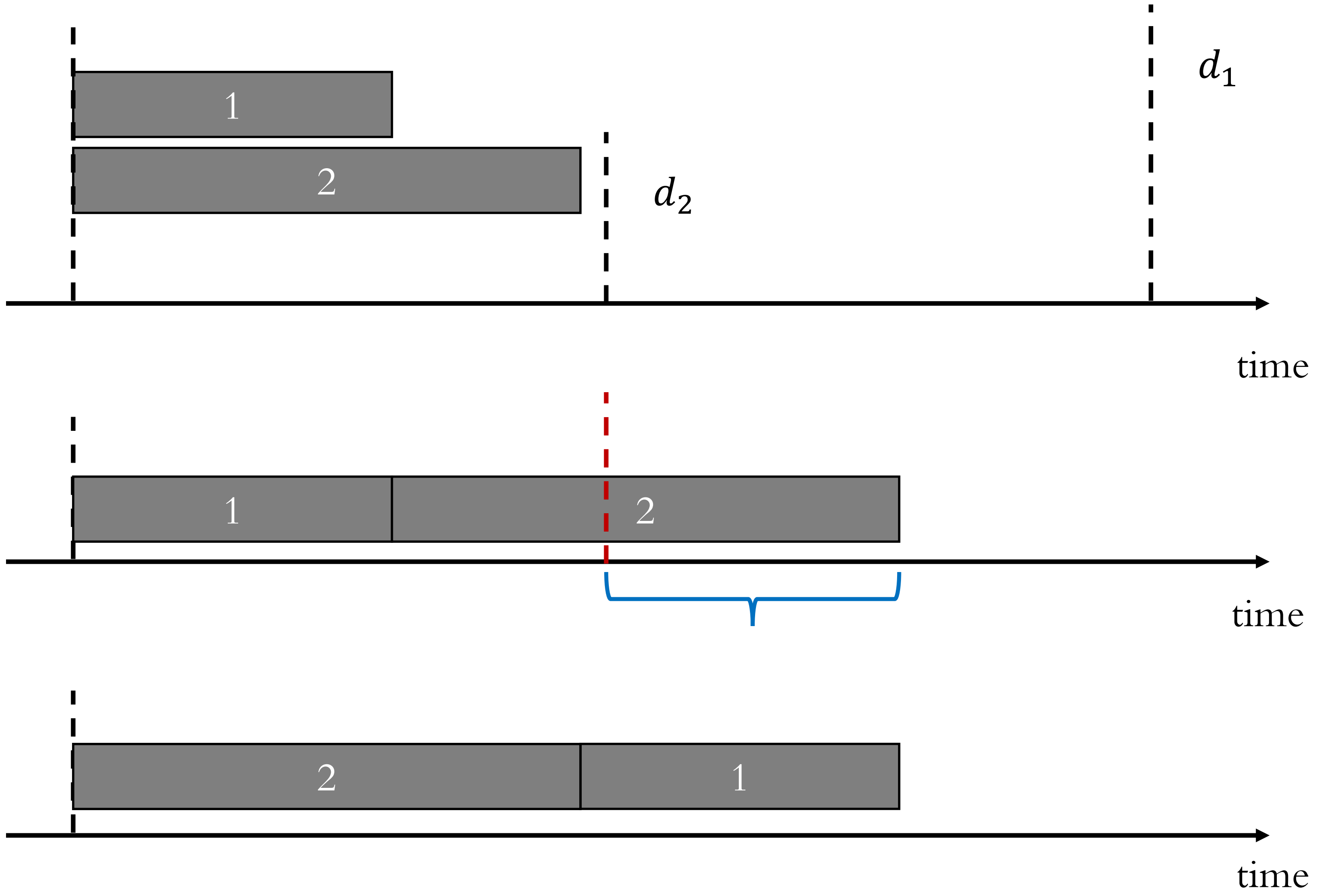
- Job $j$ requires $t_j$ units of processing time and is due at time $d_j$.
- If $j$ starts at time $s_j$, it finishes at time $f_j = s_j + t_j$.
- Lateness: $l_j = \max\{0, f_j - d_j\}$.

***Goal***: schedule all jobs to minimize maximum lateness $L = \max_j l_j$.

| | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| $t_j$ | 3 | 2 | 1 | 4 | 3 | 2 |
| $d_j$ | 6 | 8 | 9 | 9 | 14 | 15 |



# SCHEDULING TO MINIMIZING LATENESS




20

# SCHEDULING TO MINIMIZING LATENESS

EARLIEST-DEADLINE-FIRST $(n, t_1, t_2, \ldots, t_n, d_1, d_2, \ldots, d_n)$

---

SORT jobs by due times and renumber so that $d_1 \leq d_2 \leq \ldots \leq d_n$.

$t \leftarrow 0$.

FOR $j = 1$ TO $n$ ← Process the ordered jobs one by one (immediately)

  Assign job $j$ to interval $[t, t + t_j]$.

  $s_j \leftarrow t$; $f_j \leftarrow t + t_j$.

  $t \leftarrow t + t_j$.

RETURN intervals $[s_1, f_1], [s_2, f_2], \ldots, [s_n, f_n]$.

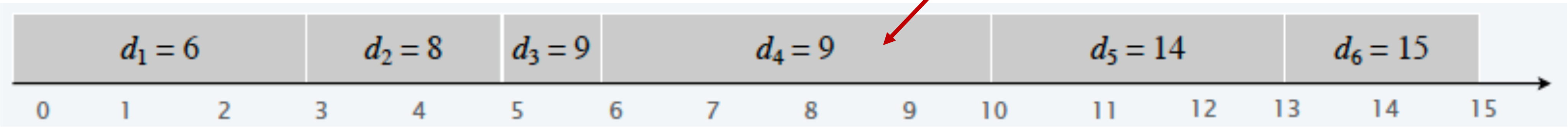
---


$\Downarrow$

$t$

| | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| $t_j$ | 3 | 2 | 1 | 4 | 3 | 2 |
| $d_j$ | 6 | 8 | 9 | 9 | 14 | 15 |

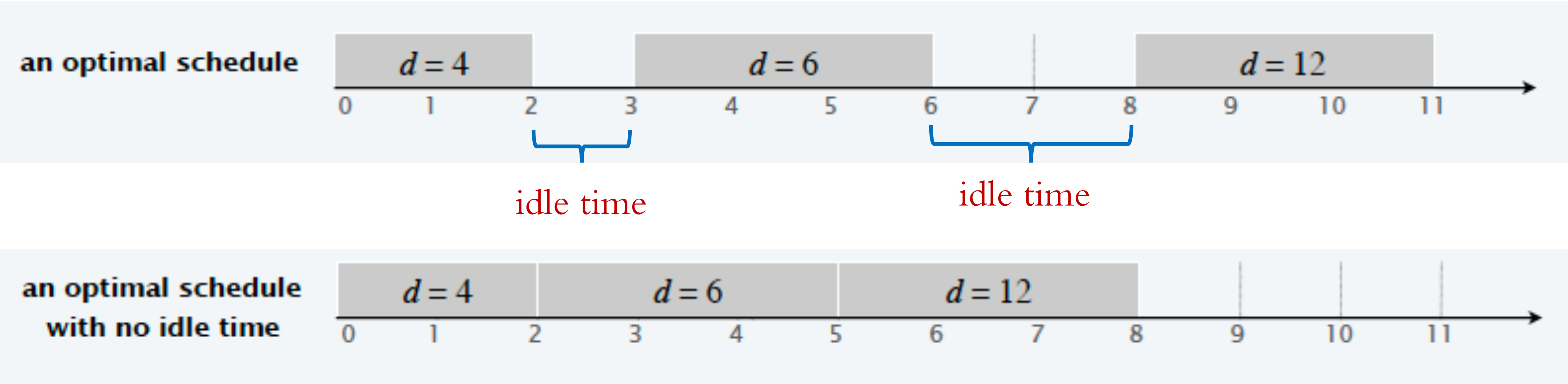
$l_4 = 1$



# SCHEDULING TO MINIMIZING LATENESS

*Properties for optimal schedules.*

*Observation 1*. There exists an optimal schedule with no idle time.



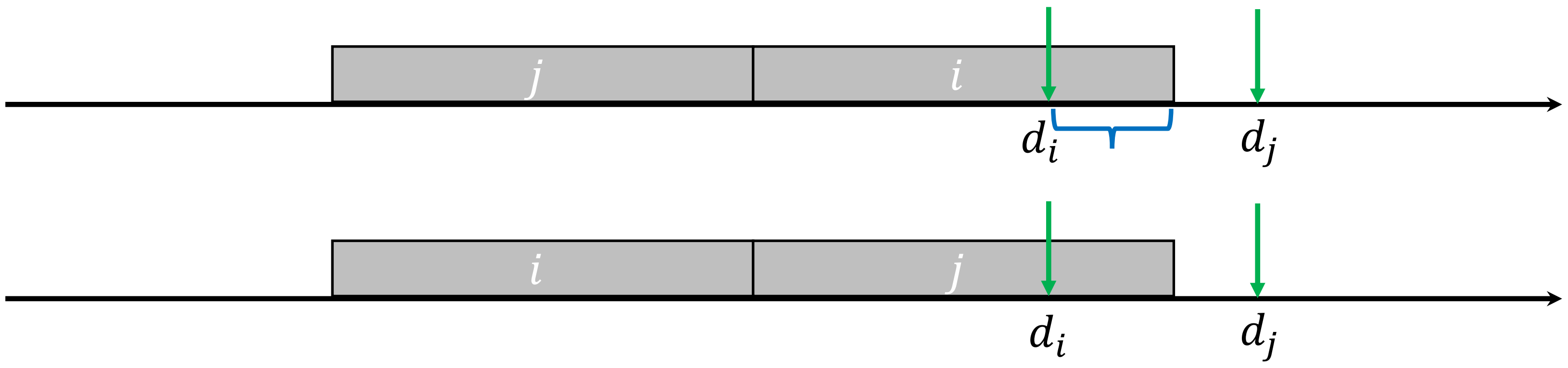

*Observation 2*. The earliest-deadline-first schedule has no idle time.

# SCHEDULING TO MINIMIZING LATENESS

or $i < j$ for ordered jobs

***Definition***. Given a schedule $S$, an inversion is a pair of jobs $i$ and $j$ such that: $d_i < d_j$ but $j$ is scheduled before $i$.

***inversion***




swap makes the schedule better!

***Observation 3***. The earliest-deadline-first schedule is the ***unique*** idle-free schedule with no inversions.

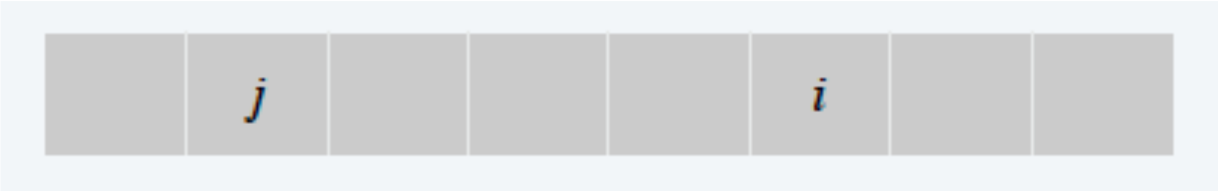
# SCHEDULING TO MINIMIZING LATENESS

***Observation 4***. If an idle-free schedule has an inversion, then it has an adjacent inversion.

two inverted jobs scheduled consecutively

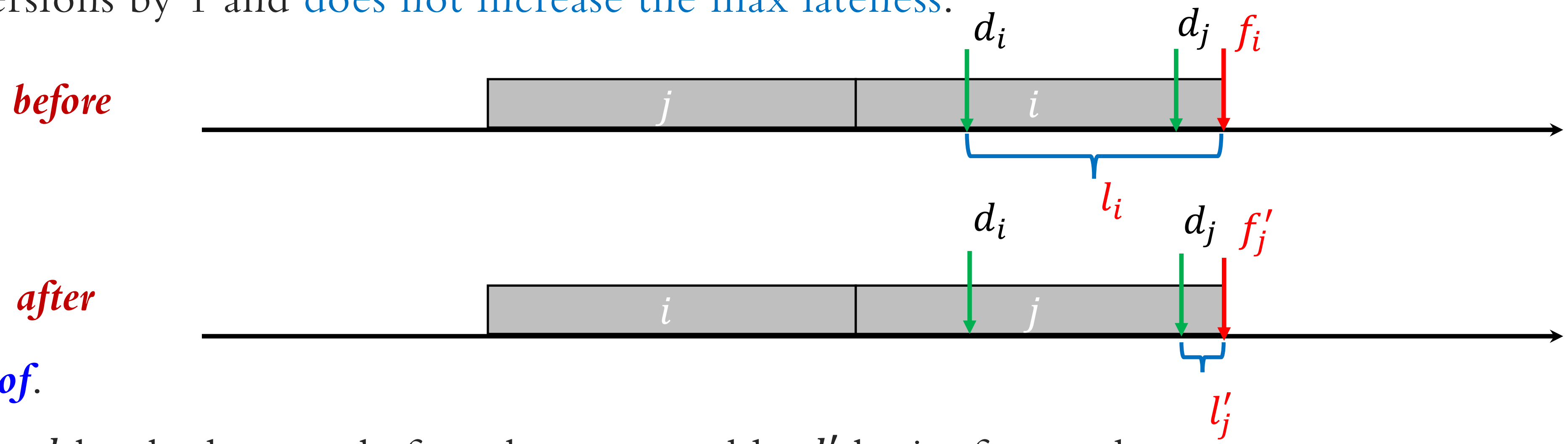
***Proof***.

- Let $i$–$j$ be a closest inversion. $d_j > d_i$
- Let $k$ be element immediately to the right of $j$.
  - Case 1: $d_j > d_k$. Then $j - k$ is an adjacent inversion.
  - Case 2. $d_j < d_k$. Then $i - k$ is a closer inversion. ■



# SCHEDULING TO MINIMIZING LATENESS

***Key Claim***. Exchanging two adjacent, inverted jobs $i$ and $j$ reduces the number of inversions by 1 and does not increase the max lateness.




***Proof***.

- Let $l$ be the lateness before the swap, and let $l'$ be it afterwards.
- $l_k' = l_k$ for all $k \neq i, j$.
- $l_i' \leq l_i$
- If job $j$ is late, $l_j' = f_j' - d_j \overset{f_j' = f_i}{=} f_i - d_j \overset{i<j:\ d_i \leq d_j}{\leq} f_i - d_i \leq l_i$. ■

# SCHEDULING TO MINIMIZING LATENESS

***Theorem***. The earliest-deadline-first schedule $S$ is optimal.

***Proof***. [by contradiction]

- Define $S^*$ to be an optimal schedule with the fewest inversions.
- Can assume $S^*$ has no idle time. ⟶ Observation 1
- Case 1: $S^*$ has no inversions. Then $S = S^*$. ⟶ Observation 3
- Case 2: $S^*$ has an inversion.
  - Let $i - j$ be an adjacent inversion ⟶ Observation 4
  - Exchanging jobs $i$ and $j$ decreases the number of inversions by 1 without increasing the max lateness ⟶ Key Claim
  - Contradicts "fewest inversions" part of the definition of $S^*$. ∎

# GREEDY ANALYSIS STRATEGIES

## *Greedy algorithm stays ahead.*

- Show that after each step of the greedy algorithm, its solution is at least as good as any other algorithm's.
- [Interval scheduling]

## *Structural.*

- Discover a simple "structural" bound asserting that every possible solution must have a certain value. Then show that your algorithm always achieves this bound.
- [Interval partitioning]

## *Exchange argument.*

- Gradually transform any solution to the one found by the greedy algorithm without hurting its quality.
- [Minimizing lateness, Interval scheduling]

# Thank You!