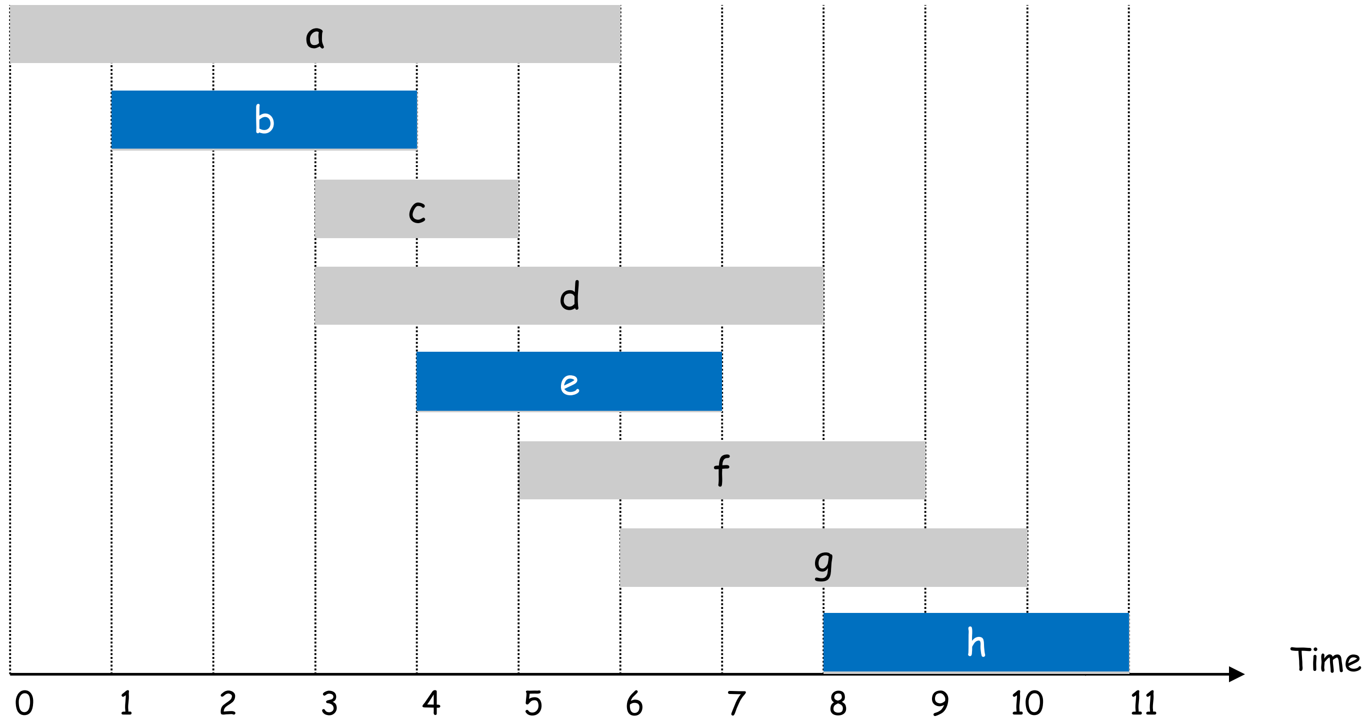
CSE 421

Greedy Algorithms / Interval Scheduling

Yin Tat Lee

Interval Scheduling

- Job j starts at $s(j)$ and finishes at $f(j)$.
- Two jobs compatible if they don't overlap.
- Goal: find maximum subset of mutually compatible jobs.



Greedy Alg: Earliest Finish Time

Consider jobs in increasing order of finish time. Take each job provided it’s compatible with the ones already taken.

```
Sort jobs by finish times so that f(1) ≤ f(2) ≤ ... ≤ f(n).
$A \leftarrow \emptyset$
for j = 1 to n {
   if (job j compatible with $A$)
      $A \leftarrow A \cup \{j\}$
}
return $A$
```

Implementation. $O(n \log n)$.

- Remember job j^* that was added last to A.
- Job j is compatible with A if $s(j) \geq f(j^*)$.

Correctness

Theorem: Greedy algorithm is optimal.

Proof: (technique: "Greedy stays ahead")

Let $i_1, i_2, i_3, \cdots, i_k$ be jobs picked by greedy, $j_1, j_2, j_3, \cdots, j_m$ those in some optimal solution in order.

We show $f(i_r) \leq f(j_r)$ for all r, by induction on r.

Base Case: i_1 chosen to have min finish time, so $f(i_1) \leq f(j_1)$.

IH: $f(i_r) \leq f(j_r)$ for some r

IS: Since $f(i_r) \leq f(j_r) \leq s(j_{r+1})$, j_{r+1} is among the candidates considered by greedy when it picked i_{r+1}, & it picks min finish, so $f(i_{r+1}) \leq f(j_{r+1})$

Observe that we must have $k \geq m$, else j_{k+1} is among (nonempty) set of candidates for i_{k+1}.

CSE 421

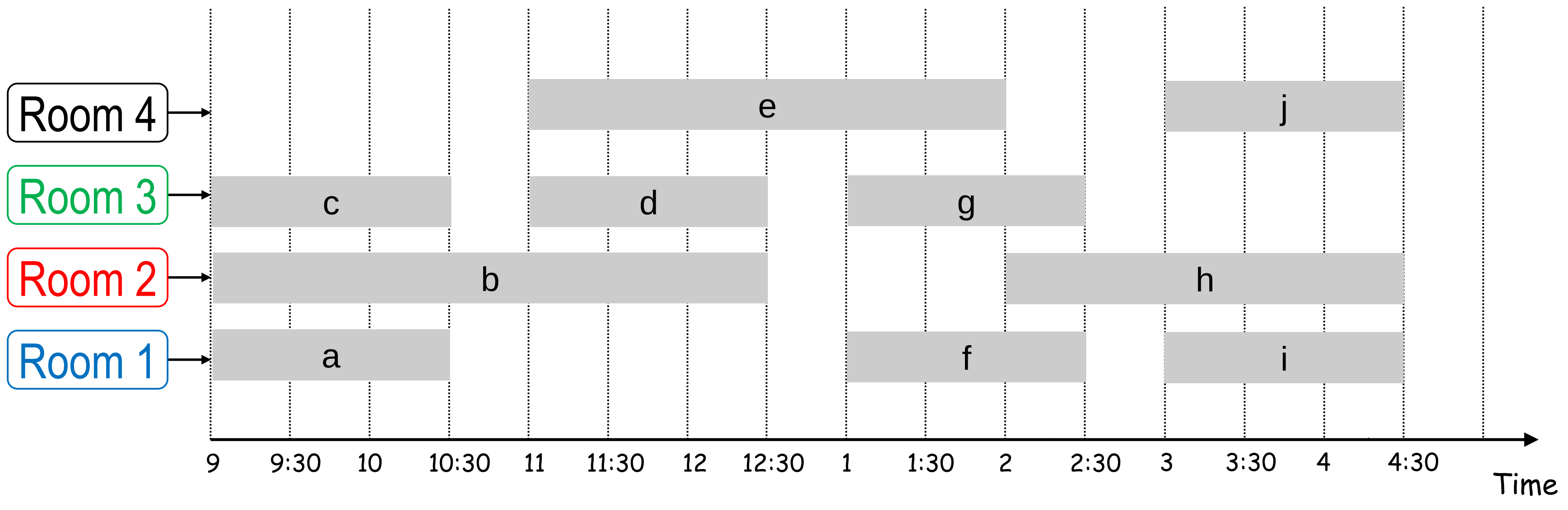
Greedy Algorithms / Interval Partitioning

Yin Tat Lee

Interval Partitioning

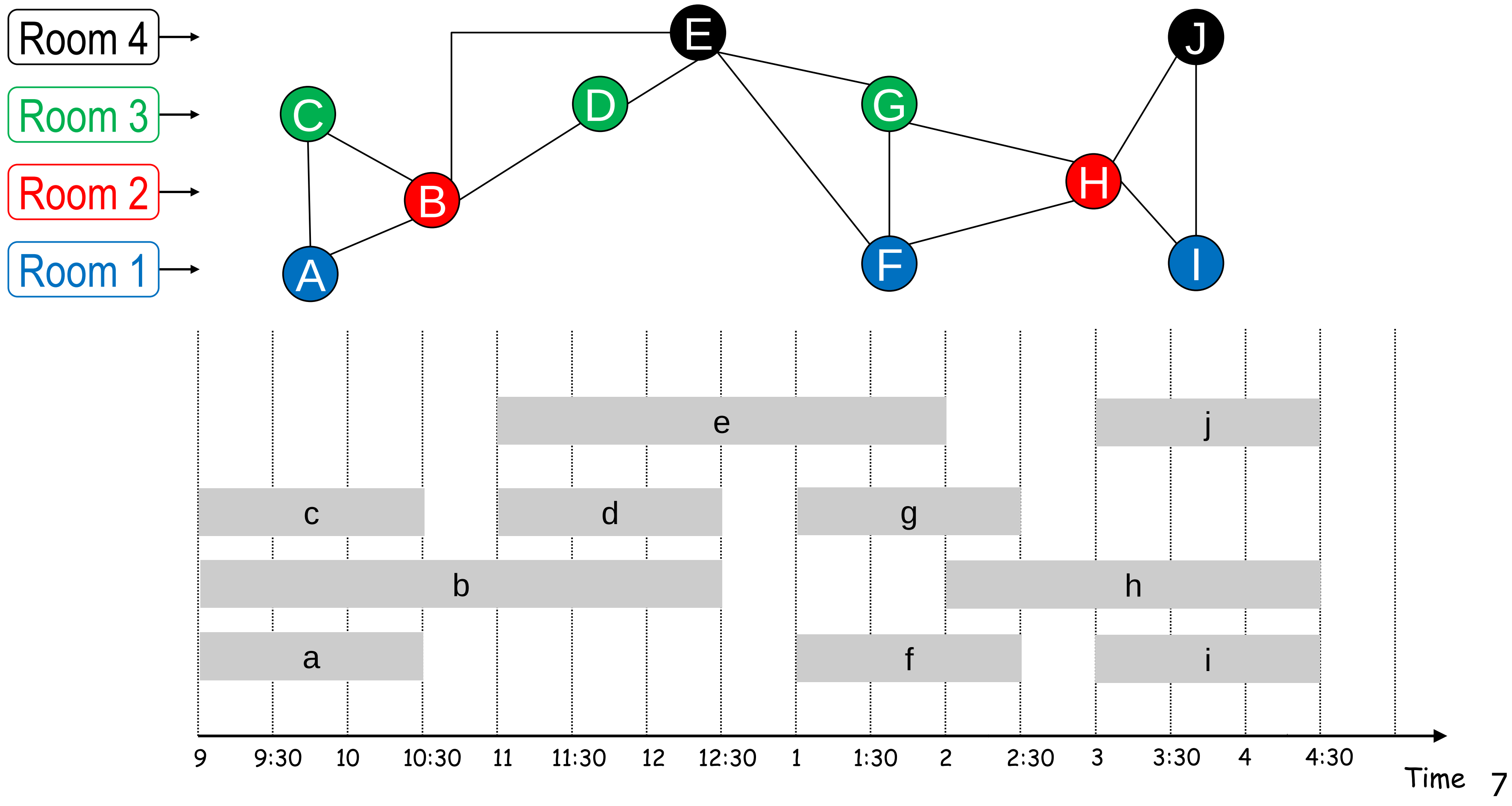
Lecture j starts at $s(j)$ and finishes at $f(j)$.

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



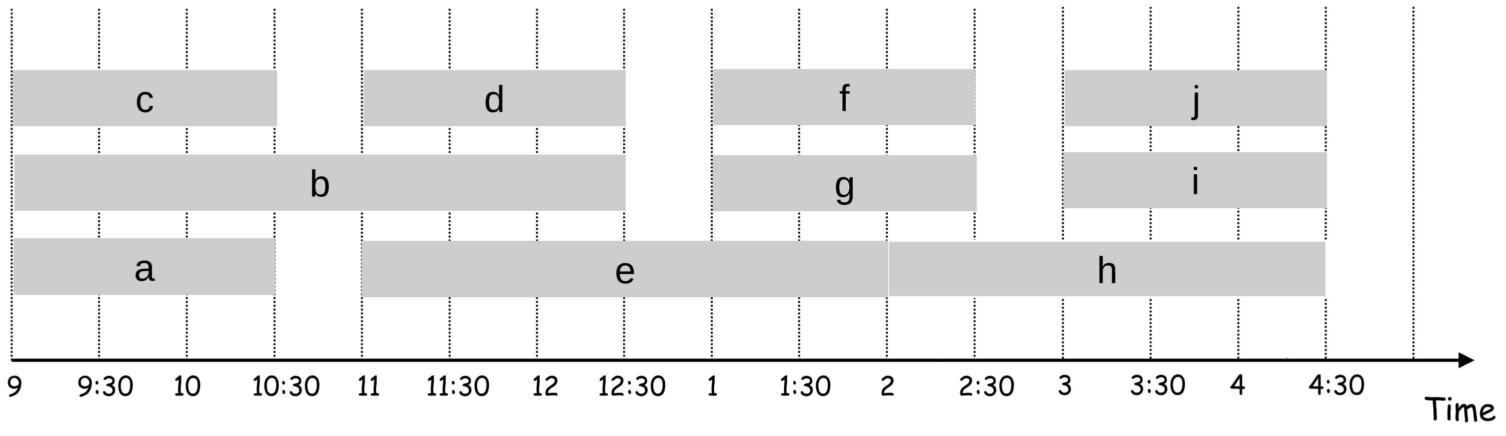
Interval Partitioning

Note: graph coloring is hard in general, but graphs corresponding to interval intersections are simpler.



A Better Schedule

This one uses only 3 classrooms



A Greedy Algorithm

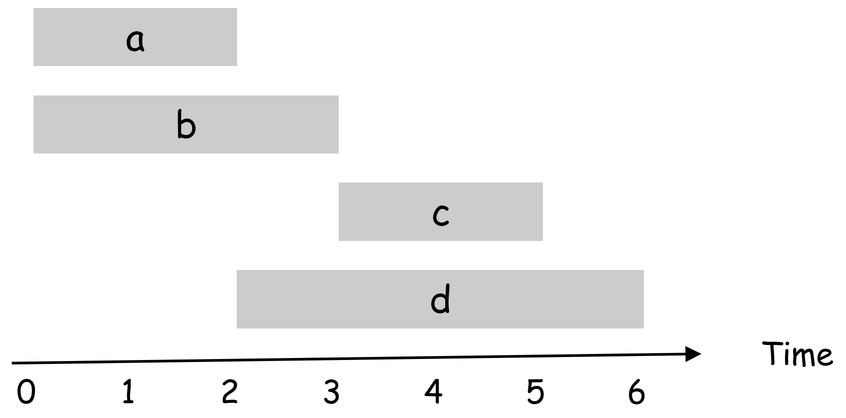
Greedy algorithm: Consider lectures in increasing order of finish time: assign lecture to any compatible classroom.

```
Sort intervals by finish time so that $f_1 \le f \le ... \le f_n$.
d ← 0

for j = 1 t
   if (lect j                              oom k, $1 \le k \le d$)
      schedule le
   else
      allocate a
      schedu
      d ← d
}
```

Correctness: This is wrong!

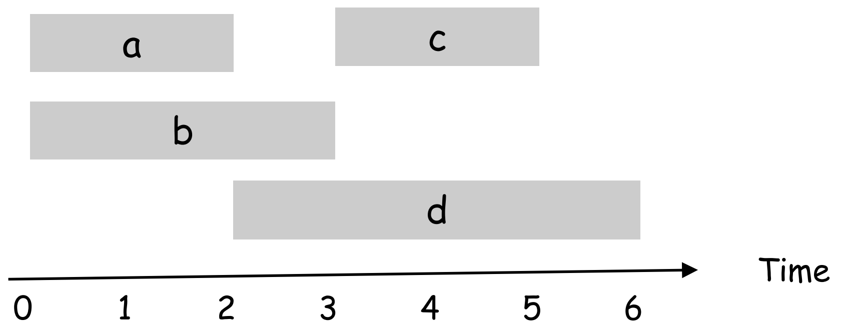
Example



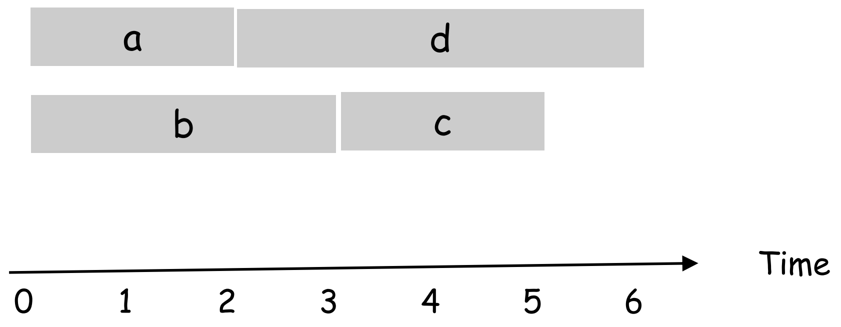
In the interval scheduling problem, we want to ignore super long job.

In this problem, we need to schedule all the jobs.
Picking them tightly is important.

Greedy by finish time gives:



OPT:



A Greedy Algorithm

Greedy algorithm: Consider lectures in increasing order of start time: assign lecture to any compatible classroom.

```
Sort intervals by starting time so that $s_1 \leq s_2 \leq \ldots \leq s_n$.
d ← 0

for j = 1 to n {
   if (lect j is compatible with some classroom k, $1 \leq k \leq d$)
      schedule lecture j in classroom k
   else
      allocate a new classroom d + 1
      schedule lecture j in classroom d + 1
      d ← d + 1
}
```

Implementation: Exercise!

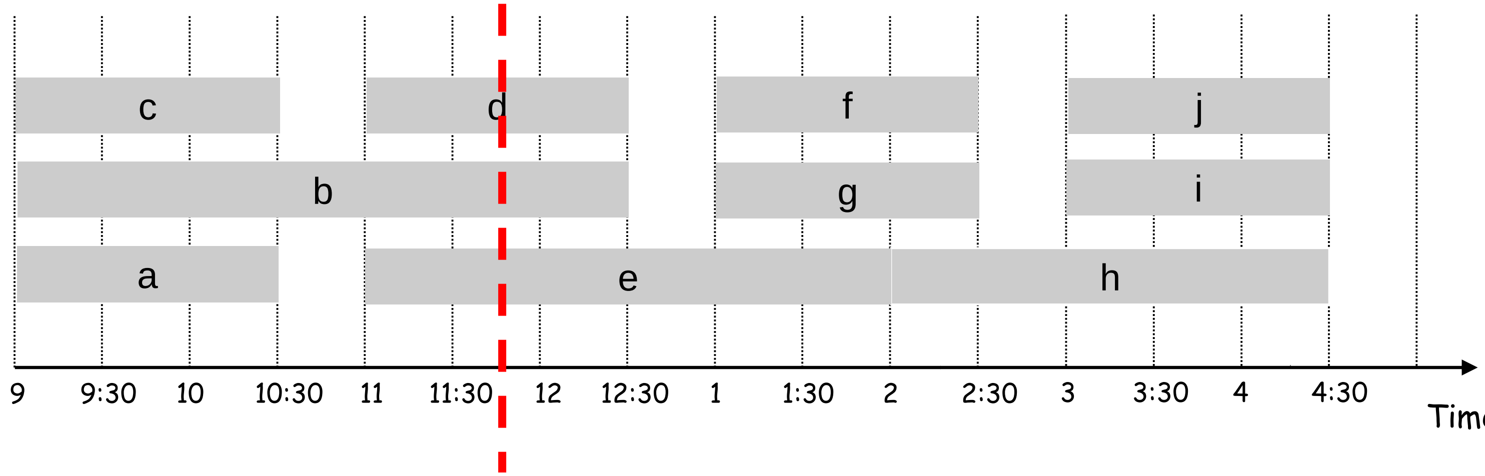
A Structural Lower-Bound on OPT

Def. The depth of a set of open intervals is the maximum number that contains any given time.

Key observation. Number of classrooms needed $\geq$ depth.

Ex: Depth of schedule below $= 3 \Rightarrow$ schedule below is optimal.

Q. Does there always exist a schedule equal to depth of intervals?



Correctness

Theorem: Greedy algorithm is optimal.

Proof (exploit structural property).

Let $d = \#$ classrooms greedy allocates.

Classroom d is opened because

there are $d - 1$ classrooms are in use.

So, d lectures overlapping at time $s(j)$, i.e. depth $\geq d$.

So, all schedules use ≥ d classrooms,

So, greedy is optimal ▪

CSE 421

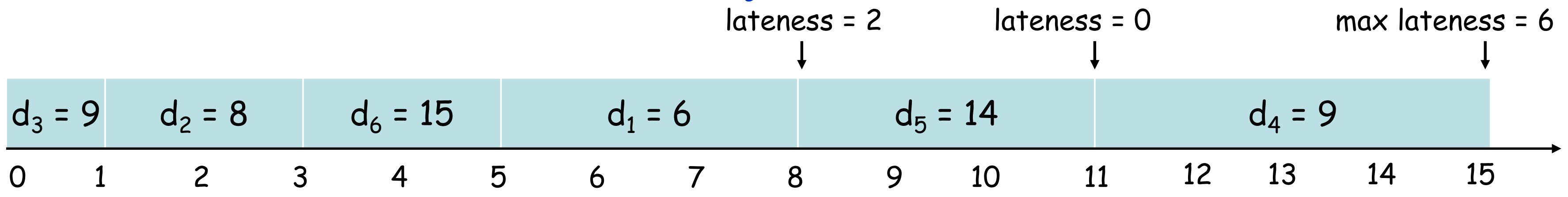
Greedy Algorithms / Minimizing Lateness

Yin Tat Lee

Scheduling to Minimizing Lateness

- Similar to interval scheduling.
- Instead of start and finish times, request $\boldsymbol{i}$ has
  - ➢ Time Requirement $\boldsymbol{t_i}$ which must be scheduled in a contiguous block
  - ➢ Deadline $\boldsymbol{d_i}$ by which time the request would like to be finished
- Requests are scheduled into time intervals $[\boldsymbol{s_i}, \boldsymbol{f_i}]$ s.t. $\boldsymbol{t_i = f_i - s_i}$.
- Lateness for request $\boldsymbol{i}$ is
  - If $\boldsymbol{d_i < f_i}$ then request $\boldsymbol{i}$ is late by $\boldsymbol{L_i = f_i - d_i}$ otherwise its lateness $\boldsymbol{L_i = 0}$
- **Goal:** Find a schedule that minimize the Maximum lateness $\boldsymbol{L = \max_i L_i}$

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



Minimizing Lateness: Greedy Algorithms

Greedy template. Consider jobs in some order.

- [Shortest processing time first]
  Consider jobs in ascending order of processing time t_j.

	1	2
t_j	1	10
d_j	100	10

counterexample

- [Smallest slack]
  Consider jobs in ascending order of slack $d_j - t_j$.

	1	2
t_j	1	10
d_j	2	10

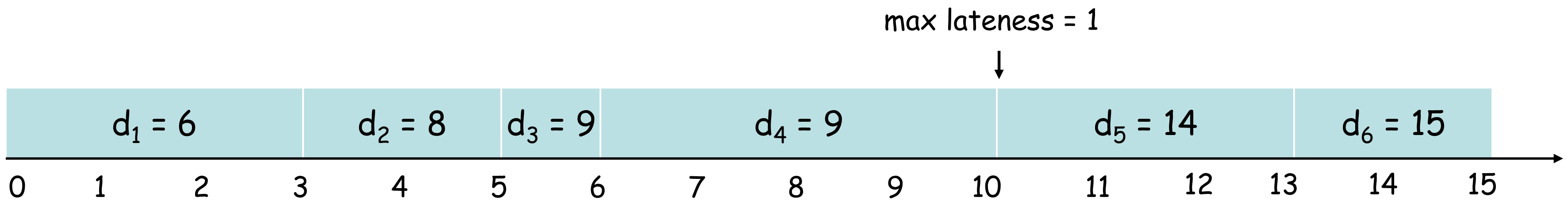
counterexample

- [Earliest deadline first]
  Consider jobs in ascending order of deadline d_j.

Greedy Algorithm: Earliest Deadline First

```
Sort deadlines in increasing order  $(d_1 \le d_2 \le \cdots \le d_n)$
$f \leftarrow 0$
for $i \leftarrow 1$ to $n$ to
        $s_i \leftarrow f$
        $f_i \leftarrow s_i + t_i$
        $f \leftarrow f_i$
end for
```

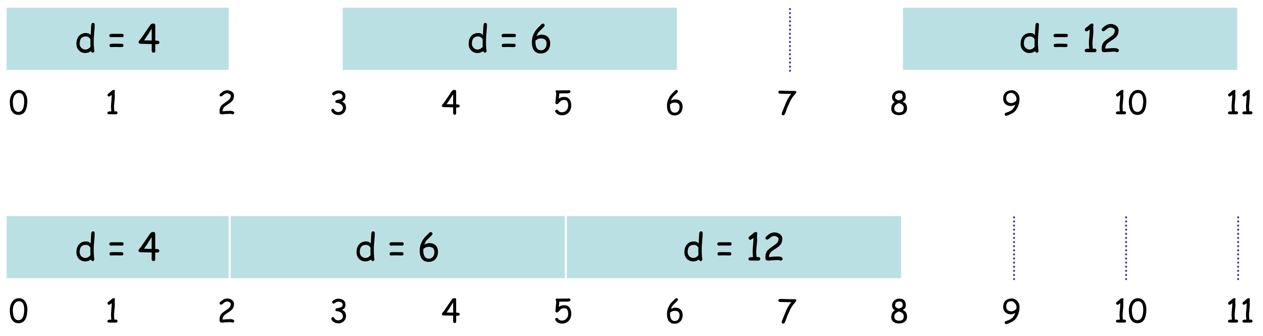
	1	2	3	4	5	6
t_j	3	2	1	4	3	2
d_j	6	8	9	9	14	15



Minimizing Lateness: No Idle Time

Observation.

- There exists an optimal schedule with no idle time.



Observation.

- The greedy schedule has no idle time.

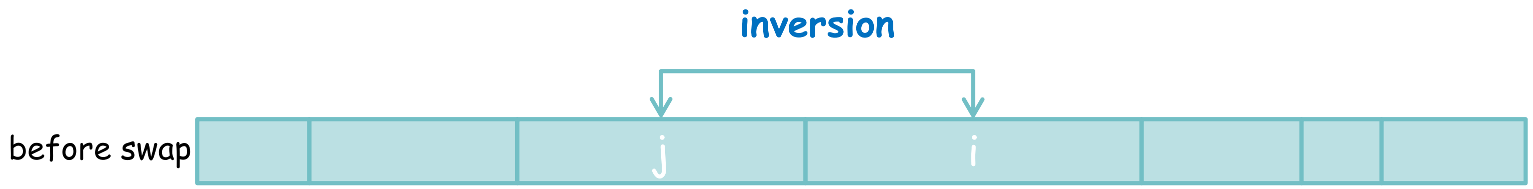
Proof for Greedy Algorithm: Exchange Argument

- We will show that if there is another schedule $\boldsymbol{O}$ (think optimal schedule) then we can gradually change $\boldsymbol{O}$ so that
  - at each step the maximum lateness in $\boldsymbol{O}$ never gets worse.
  - it eventually becomes the same cost as $\boldsymbol{A}$.

Minimizing Lateness: Inversions

Definition

- An adjacent inversion in schedule S is a pair of jobs i and j such that
  - $d_i < d_j$
  - Job i is scheduled immediately after Job j



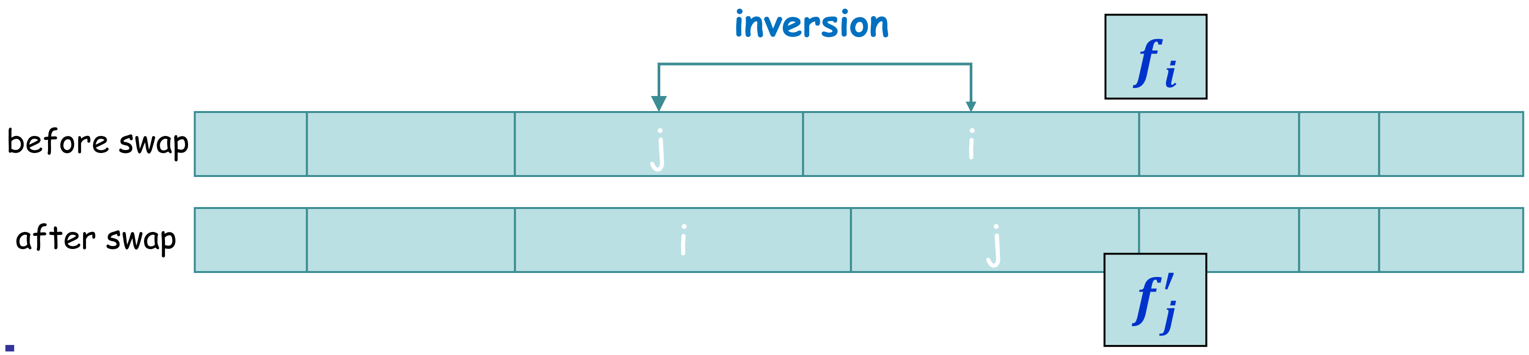
Observation

- Greedy schedule has no adjacent inversions.

Minimizing Lateness: Inversions

Definition

- An adjacent inversion in schedule S is a pair of jobs i and j such that
  - $d_i < d_j$
  - Job i is scheduled immediately after Job j

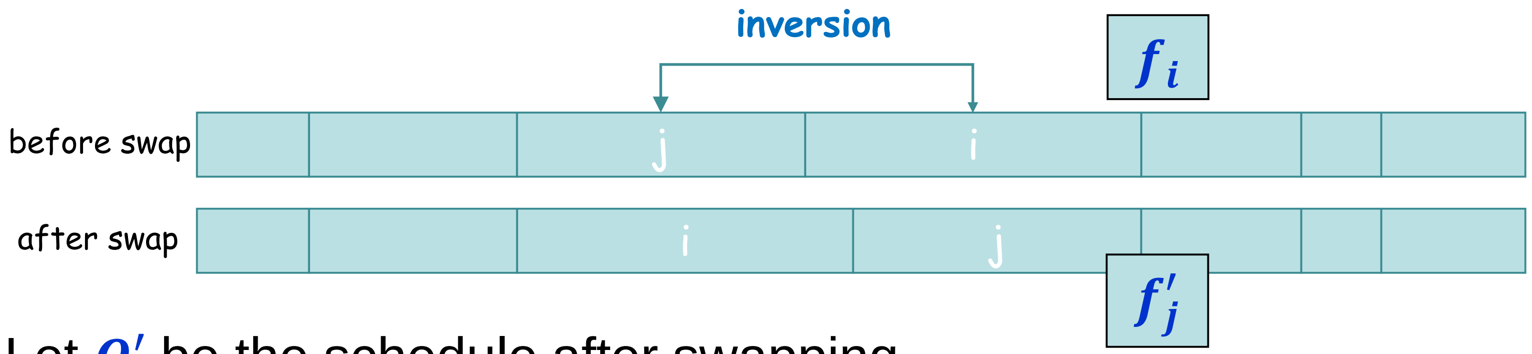


Claim

- Swapping two adjacent, inverted jobs reduces the number of inversions by one and does not increase the max lateness.

Minimizing Lateness: Inversions

Lemma: Swapping two adjacent, inverted jobs does not increase the maximum lateness.



Proof: Let O' be the schedule after swapping.

- Lateness $L_i' \leq L_i$ since i is scheduled earlier in O' than in O
- Requests i and j together occupy the same total time slot in both schedules
    - All other requests $k \neq i, j$ have $L_k' = L_k$
    - $f_j' = f_i$ so $L_j' = f_j' - d_j = f_i - d_j < f_i - d_i = L_i$
- Maximum lateness has not increased!

Optimal schedules and inversions

Claim: There is an optimal schedule with no idle time and no inversions

Proof:

- By previous argument there is an optimal schedule $\boldsymbol{O}$ with no idle time
- If $\boldsymbol{O}$ has an inversion then it has a **consecutive** pair of requests in its schedule that are inverted and can be swapped without increasing lateness
- Eventually these swaps will produce an optimal schedule with no inversions
  - Each swap decreases the number of inversions by **1**
  - There are at most $\boldsymbol{n(n-1)/2}$ inversions.
    (we only care that this is finite.)

Idleness and Inversions are the only issue

Claim: All schedules with no inversions and no idle time have the same maximum lateness

Proof:

- Schedules can differ only in how they order requests with equal deadlines
- Consider all requests having some common deadline d
- Maximum lateness of these jobs is based only on the finish time of the last of these jobs but the set of these requests occupies the same time segment in both schedules
  - Last of these requests finishes at the same time in any such schedule.