Greedy Algorithms

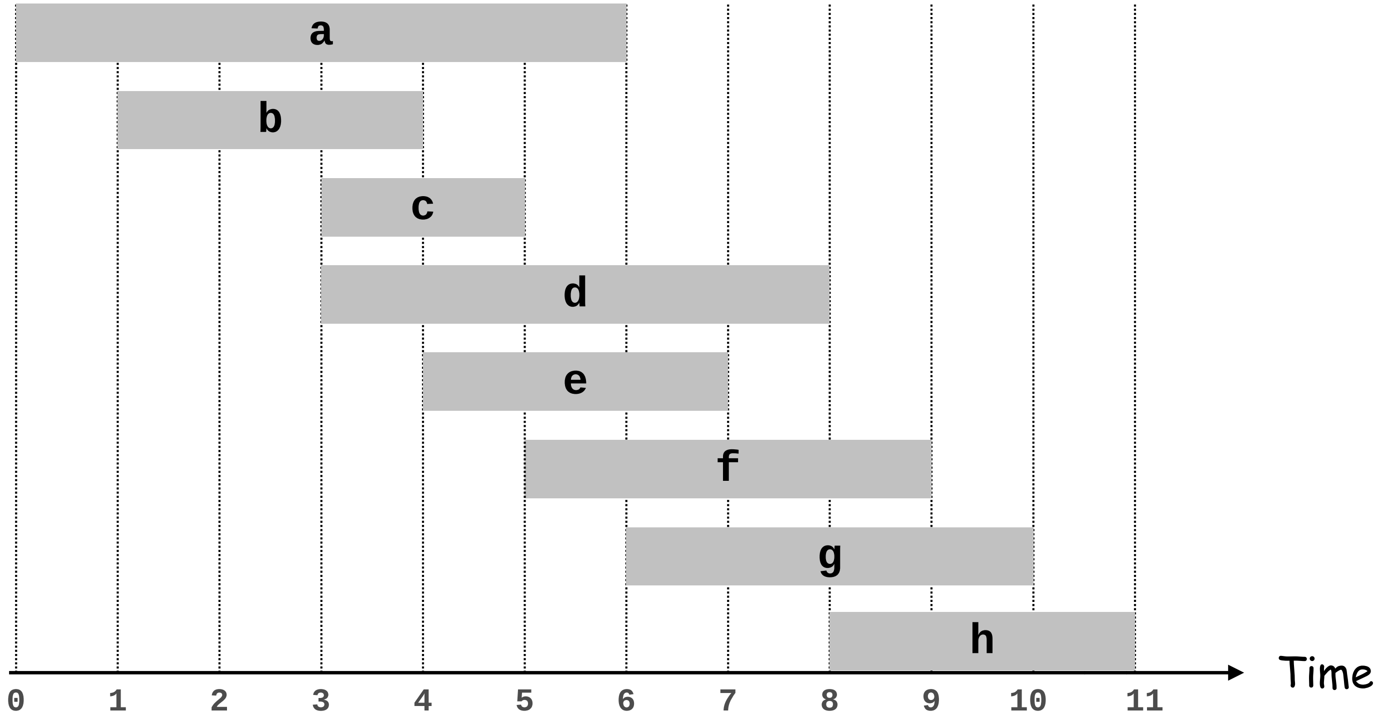
Based on slides by E. Demaine, C. Leiserson, S. Raskhodnikova, A. Smith, K. Wayne

Greedy Algorithms

- Build up a solution to an optimization problem at each step shortsightedly choosing the option that currently seems the best.
  - Sometimes good
  - Often does not work
- Key to designing greedy algorithms: find **structure** that ensures you don't leave behind other options

Interval Scheduling Problem

- Job j starts at s_j and finishes at f_j.
- Two jobs are **compatible** if they do not overlap.
- **Find**: maximum subset of mutually compatible jobs.



Possible Greedy Strategies

Consider jobs in some natural order. Take next job if it is compatible with the ones already taken.

- **Earliest start time:** ascending order of s_j.
- **Earliest finish time:** ascending order of f_j.
- **Shortest interval:** ascending order of $(f_j – s_j)$.
- **Fewest conflicts:** For each job j, count the number of conflicting jobs c_j. Schedule in ascending order of c_j.

Greedy: Counterexamples

for earliest start time

for shortest interval

for fewest conflicts

Formulating an Algorithm

- Input: arrays of start and finishing times
  - $s_1, s_2, \ldots, s_n$
  - $f_1, f_2, \ldots, f_n$

- Input length?
  - $2n = \Theta(n)$

Greedy Algorithm

- **Earliest finish time:** ascending order of f_j.

```
Sort jobs by finish times so that
f1 ≤ f2 ≤ ... ≤ fn.

A ← ϕ    // Set of jobs selected so far
for j = 1 to n
   if (job j compatible with A)
      A ← A ∪ {j}
return A
```

- Implementation:
  - How do we quickly test if j is compatible with A?
  - Store job j^* that was added last to A.
  - Job j is compatible with A if $s_j \geq f_{j^*}$.

Time and space analysis

O(n log n)

O(1)

$n \times O(1)$

```
Sort jobs by finish times so that
      f₁ ≤ f₂ ≤ ... ≤ fₙ.

A ← (empty)   // Queue of selected jobs
j* ← 0
for j = 1 to n
   if (f_j* ≤ s_j)
      enqueue(j onto A)
      j* ← j

return A
```

O(n log n) time; O(n) space.

Analysis: Greedy Stays Ahead

Theorem. Greedy algorithm's solution is optimal.

Proof strategy (by contradiction):

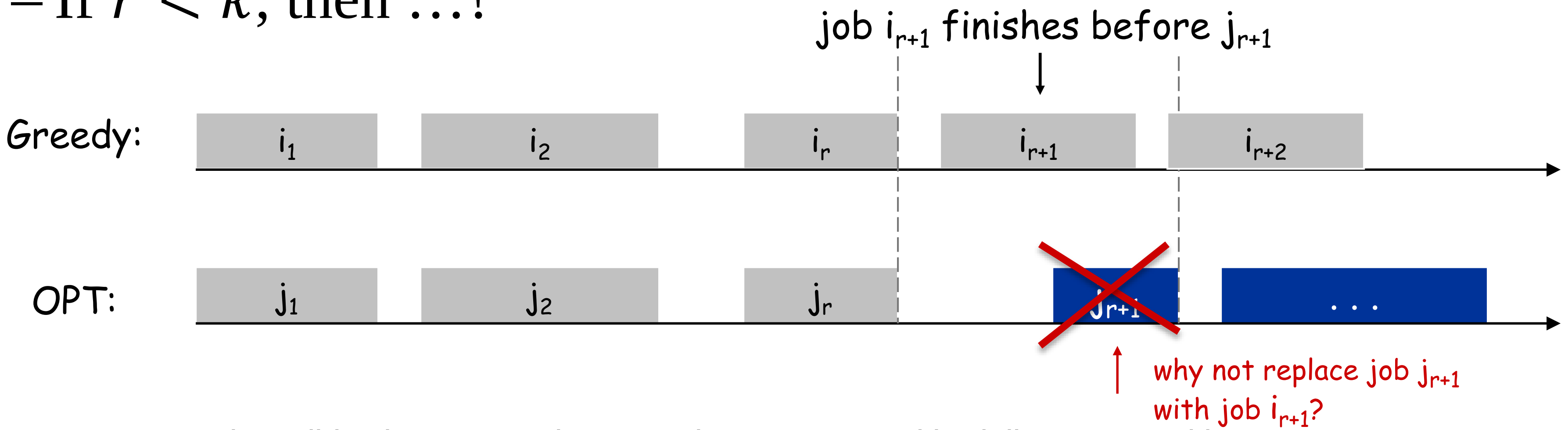
- Suppose greedy is not optimal.
- Consider an optimal solution…
  - which one?
  - optimal solution that agrees with the greedy solution for as many initial jobs as possible
- Look at the first place in the list where optimal solution differs from the greedy solution
  - Show a new optimal solution that agrees more w/ greedy
  - Contradiction!

Analysis: Greedy Stays Ahead

Theorem: Greedy algorithm's solution is optimal.

Proof (by contradiction): Suppose greedy not optimal.

- Let $i_1, i_2, \ldots\ i_k$ denote set of jobs selected by greedy.
- Let $j_1, j_2, \ldots\ j_m$ be the optimal solution with
$$i_1 = j_1, i_2 = j_2, \ldots, i_r = j_r$$
  for the largest possible value of r.
- If $r < k$, then …?



Analysis: Greedy Stays Ahead

Theorem: Greedy algorithm's solution is optimal.

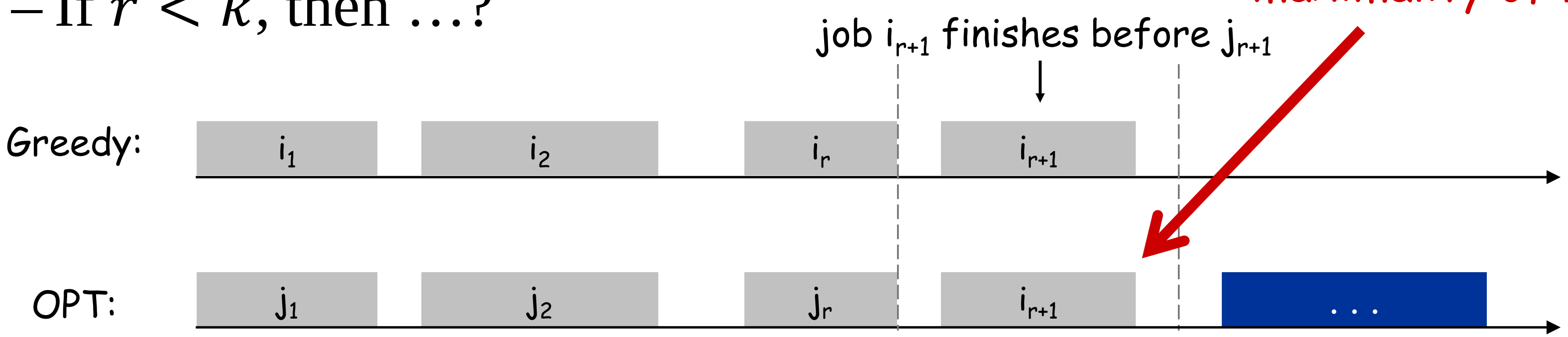
Proof (by contradiction): Suppose greedy not optimal.

- Let $i_1, i_2, \ldots\ i_k$ denote set of jobs selected by greedy.
- Let $j_1, j_2, \ldots\ j_m$ be the optimal solution with

$$i_1 = j_1, i_2 = j_2, \ldots, i_r = j_r$$

  for the largest possible value of r.
- If $r < k$, then …?

solution still feasible and optimal, but contradicts maximality of r.



Analysis: Greedy Stays Ahead

Theorem: Greedy algorithm's solution is optimal.

Proof (by contradiction): Suppose greedy not optimal.

– Let $i_1, i_2, \ldots\ i_k$ denote set of jobs selected by greedy.

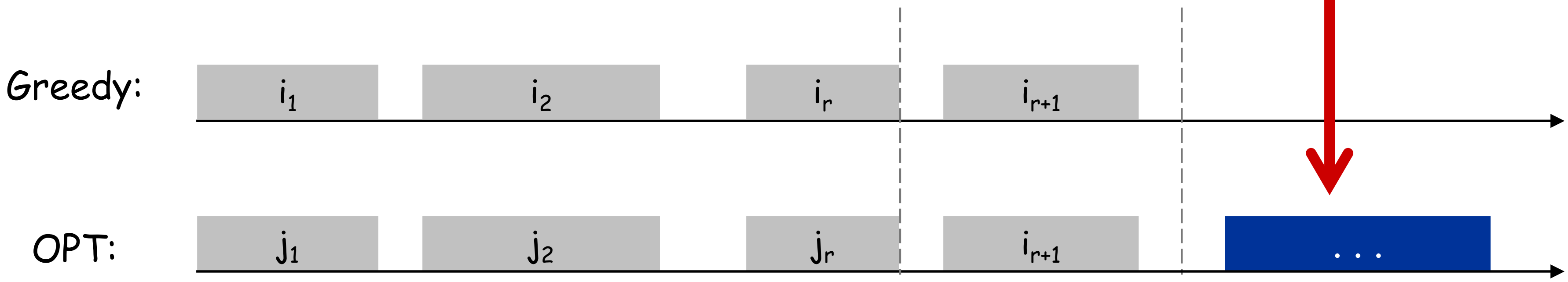
– Let $j_1, j_2, \ldots\ j_m$ be the optimal solution with

$$i_1 = j_1, i_2 = j_2, \ldots, i_r = j_r$$

for the largest possible value of r.

– If $r < k$, then we get contradiction.

Could it be that $r = k$ but $k < m$?

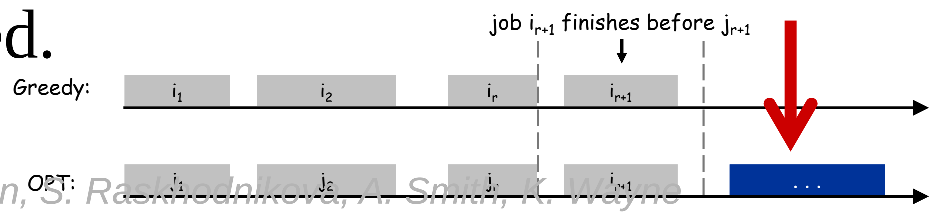


Analysis: Greedy Stays Ahead

Theorem: Greedy algorithm's solution is optimal.

Proof (by contradiction): Suppose greedy not optimal.

- Let $i_1, i_2, \ldots\ i_k$ denote set of jobs selected by greedy.
- Let $j_1, j_2, \ldots\ j_m$ be the optimal solution with
  $$i_1 = j_1, i_2 = j_2, \ldots, i_r = j_r$$
  for the largest possible value of r.
- If $r < k$, we get a contradiction by replacing j_{r+1} with i_{r+1} because we get an optimal solution with larger r.
- If $r = k$ but $m > k$, we get a contradiction because greedy algorithm stopped before all jobs were considered.



Alternate Way to See the Proof

- Induction statement

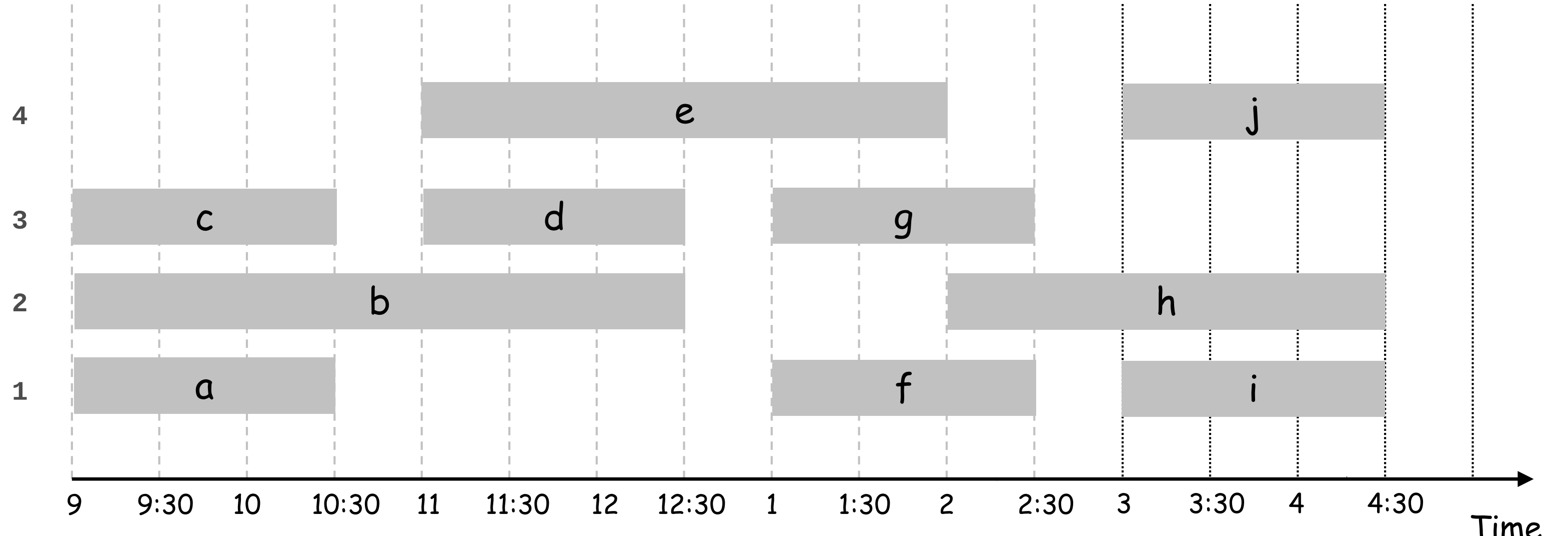
$P(k)$: There is an optimal solution that agrees with the greedy solution in the first k jobs.

- $P(n)$ is what we want to prove.
- Base case: $P(0)$
- We essentially proved the induction step…

Interval Partitioning

Interval Partitioning

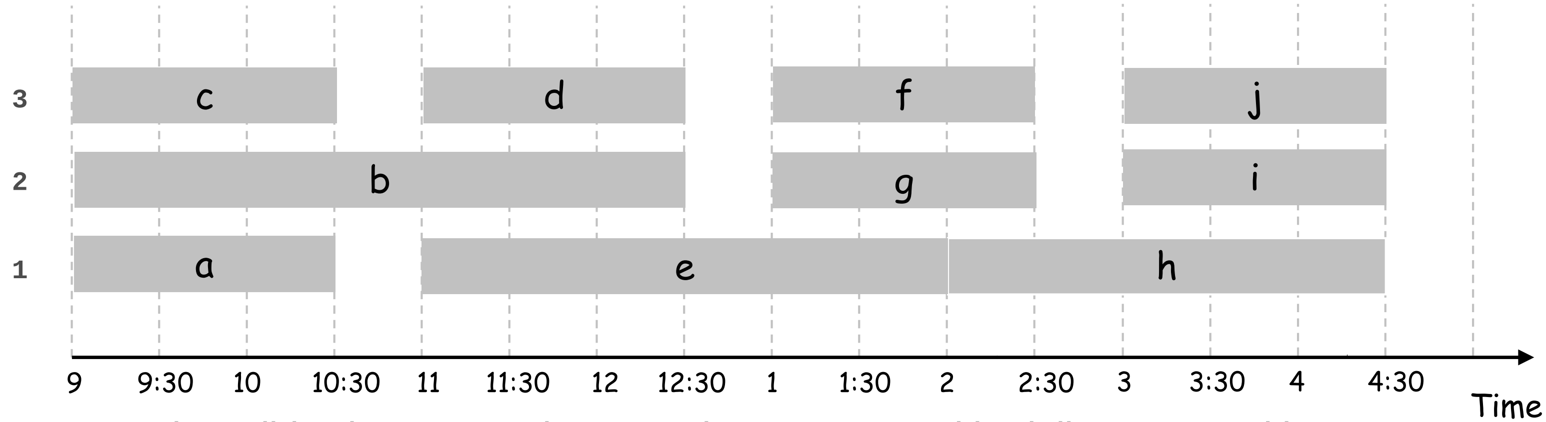
- Lecture j starts at s_j and finishes at f_j.
- **Input**: $s_1, \dots, s_n$ and $f_1, \dots, f_n$.
- **Goal**: find minimum number of classrooms to schedule all lectures so that no two occur at the same time in the same room.
- **E.g.**: 10 lectures are scheduled in **4** classrooms.



Based on slides by E. Demaine, C. Leiserson, S. Raskhodnikova, A. Smith, K. Wayne

Interval Partitioning

- Lecture j starts at s_j and finishes at f_j.
- **Input**: $s_1, \dots, s_n$ and $f_1, \dots, f_n$.
- **Goal**: find minimum number of classrooms to schedule all lectures so that no two occur at the same time in the same room.
- **E.g.**: Same lectures scheduled in **3** classrooms.

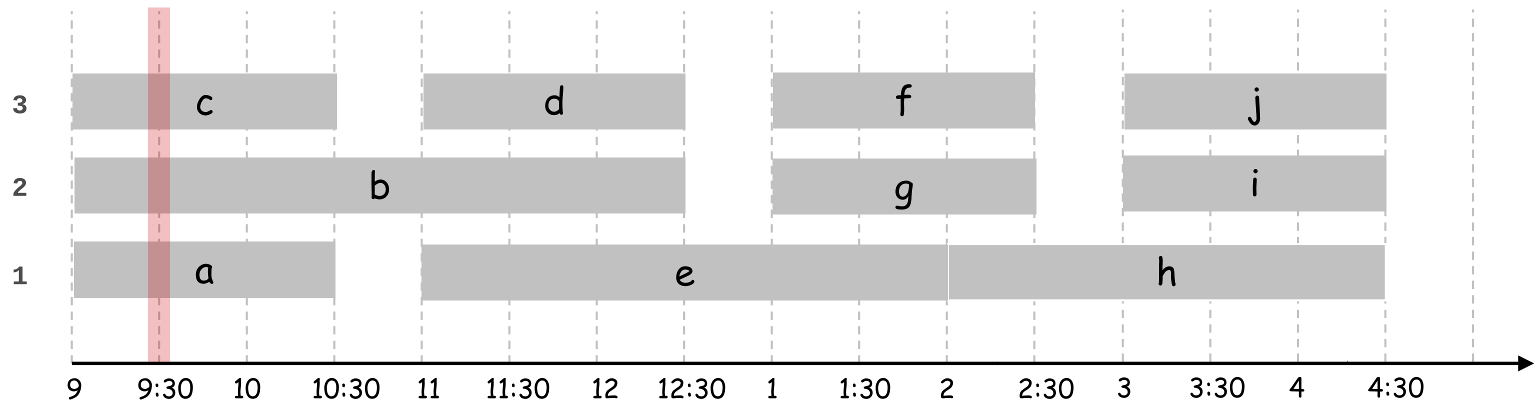


Based on slides by E. Demaine, C. Leiserson, S. Raskhodnikova, A. Smith, K. Wayne

Lower Bound

- **Definition**. The **depth** of a set of open intervals is the maximum number that contain any given time.
- **Key lemma**. Number of classrooms needed $\geq$ depth.
- **E.g.:** Depth of this schedule = 3 $\Rightarrow$ this schedule is optimal.

a, b, c all contain 9:30



- **Q:** Is it always sufficient to have number of classrooms = depth?

Greedy Algorithm

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

```
Sort intervals by starting time so that s1 ≤ s2 ≤ ... ≤ sn.
d ← 0   // Number of allocated classrooms
for j = 1 to n
   if (lecture j is compatible with some classroom k)
      schedule lecture j in classroom k
   else
      allocate a new classroom d + 1
      schedule lecture j in classroom d + 1
      d ← d + 1
```

- Implementation. $O(n \log n)$ time; $O(n)$ space.
  - For each classroom, maintain the finish time of the last job added.
  - Keep the classrooms in a priority queue
    - Using a heap, main loop takes $O(n \log d)$ time

Analysis: Structural Argument

Observation. Greedy algorithm never schedules two incompatible lectures in the same classroom.

- **Theorem.** Greedy algorithm is optimal.
- **Proof:** Let d = number of classrooms allocated by greedy.
  - Classroom d is opened because we needed to schedule a lecture, say j, that is incompatible with all $d-1$ last lectures in other classrooms.
  - These d lectures each end after s_j.
  - Since we sorted by start time, they start no later than s_j.
  - Thus, we have d lectures overlapping at time $s_j + \varepsilon$.
  - Key lemma $\Rightarrow$ all schedules use $\geq d$ classrooms. ▪

Duality

- Our first example of "duality"!
- High-level overview of proof of correctness:
  - Identify obstacles to scheduling in few classrooms
    - Sets of overlapping lectures
  - Show that our algorithm's solution matches some obstacle
    - If our solution uses d classrooms,
      then there is a set of d overlapping lectures
  - Conclude that our solution cannot be improved

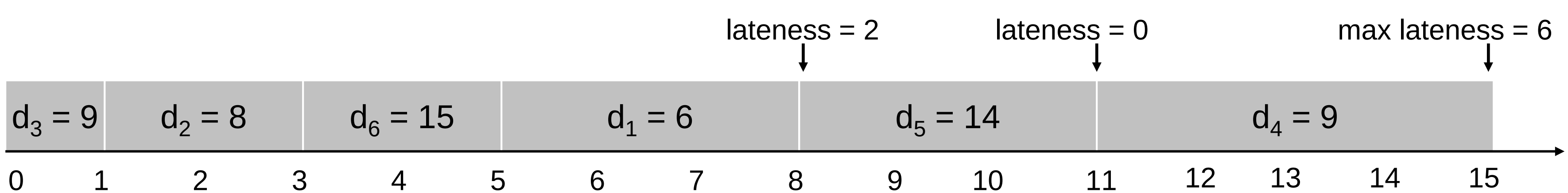
Scheduling to minimize lateness

Scheduling to Minimizing Lateness

Minimizing lateness problem.

- 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: $\ell_j = \max \{ 0, f_j - d_j \}$.
- Goal: schedule all jobs to minimize maximum lateness $L = \max \ell_j$.

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



Based on slides by E. Demaine, C. Leiserson, S. Raskhodnikova, A. Smith, K. Wayne

Greedy strategies?

Minimizing Lateness: Greedy Strategies

Greedy template: consider jobs in some order.

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

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

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

Minimizing Lateness: Greedy Strategies

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

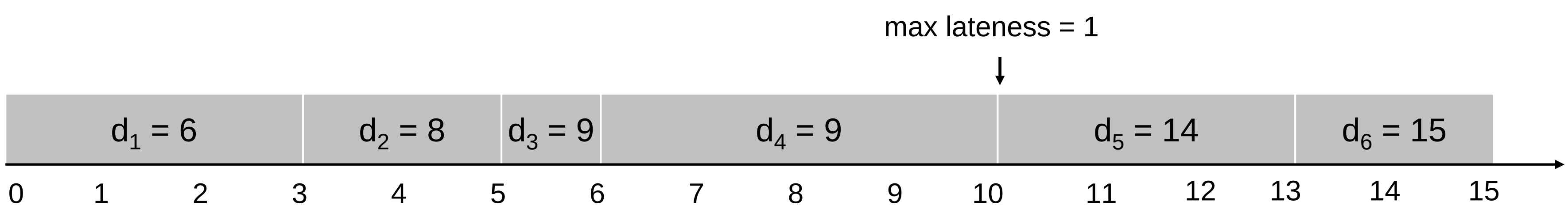
counterexample

Minimizing Lateness: Greedy Algorithm

- [Earliest deadline first]

Sort n jobs by deadline so that $d_1 \le d_2 \le \ldots \le d_n$

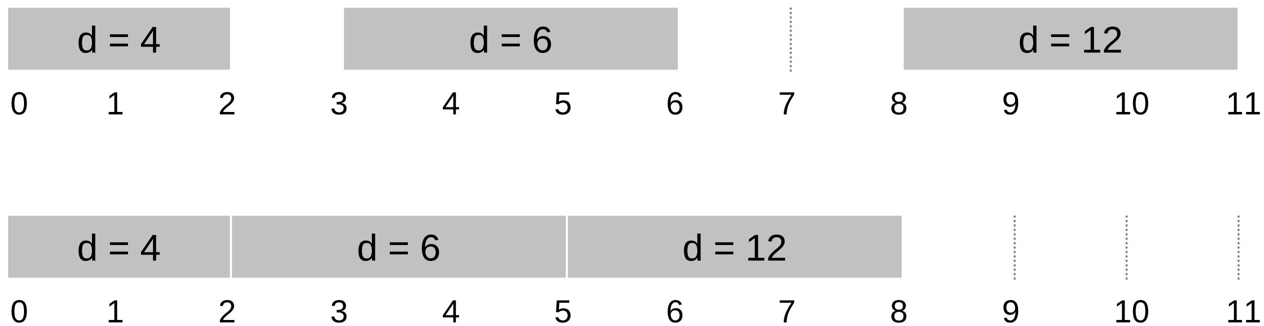
$t \leftarrow 0$
for $j = 1$ to n
    Assign job j to interval $[t, t + t_j]$
    $s_j \leftarrow t$, $f_j \leftarrow t + t_j$
    $t \leftarrow t + t_j$
output intervals $[s_j, f_j]$



Based on slides by E. Demaine, C. Leiserson, S. Raskhodnikova, A. Smith, K. Wayne

Minimizing Lateness: No Idle Time

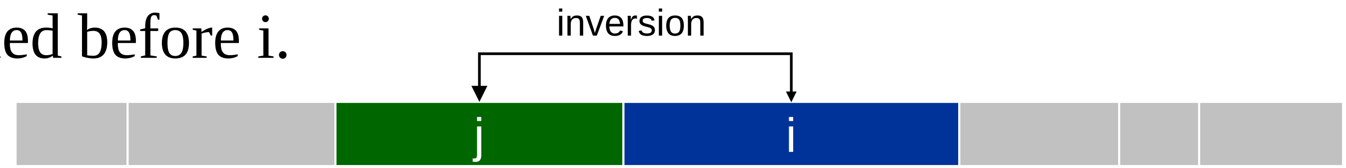
- **Observation.** There exists an optimal schedule with no idle time.



- **Observation.** The greedy schedule has no idle time.

Minimizing Lateness: Inversions

- An inversion in schedule S is a pair of jobs i and j such that $d_i < d_j$ but j scheduled before i.

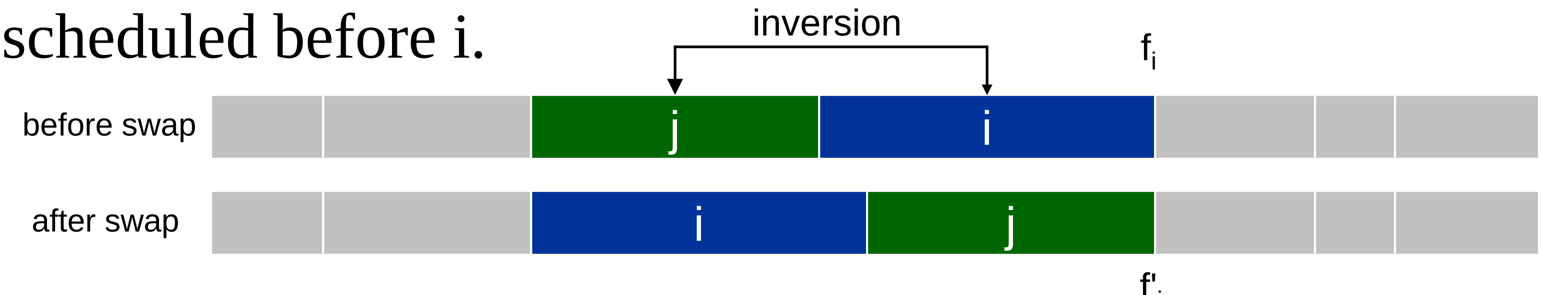


- **Observation.** Greedy schedule has no inversions.

- **Observation.** If a schedule (with no idle time) has an inversion, it has one with a pair of inverted jobs scheduled consecutively.

Minimizing Lateness: Inversions

- An inversion in schedule S is a pair of jobs i and j such that $d_i < d_j$ but j scheduled before i.



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

- **Proof:** Let ℓ be the lateness before the swap, and let ℓ' be the lateness afterwards.
  - $\ell'_k = \ell_k$ for all $k \neq i, j$
  - $\ell'_i \leq \ell_i$
  - If job j is late:

$$
\begin{aligned}
\ell'_j &= f'_j - d_j && \text{(definition)} \\
&= f_i - d_j && (j \text{ finishes at time } f_i) \\
&\leq f_i - d_i && (d_i < d_j) \\
&\leq \ell_i && \text{(definition)}
\end{aligned}
$$

Minimizing Lateness: Analysis

Theorem. Greedy schedule S is optimal.

Proof: Define S* to be an optimal schedule that has the fewest number of inversions.

- Can assume S* has no idle time.
- If S* has no inversions, then S = S*.
- If S* has an inversion, let i-j be an adjacent inversion.
  - Swapping i and j does not increase the maximum lateness and strictly decreases the number of inversions.
  - This contradicts the definition of S*. ▪

Alternative Proof

- Fix an input t, d.
- Define S* to be an optimal schedule that has the fewest number of inversions.
  - If S* has no inversions, then S = S*.
  - If S* has an inversion, let i-j be an adjacent inversion.
    - swapping i and j does not increase the maximum lateness and strictly decreases the number of inversions
    - this contradicts definition of S* ▪

Summary: 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.

- **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.

- **Exchange argument.** Gradually transform any solution to the one found by the greedy algorithm without hurting its quality.