Algorithm Design and Analysis

CSE 565

LECTURE 5

Greedy Algorithms

- Interval Scheduling
- Interval Partitioning

Guest lecturer: Martin Furer

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

Review

- In a DFS tree of an undirected graph, can there be an edge (u,v)
  - where v is an ancestor of u? ("back edge")
  - where v is a sibling of u? ("cross edge")
- Same questions with a directed graph?
- Same questions with a BFS tree
  - directed?
  - undirected?

Design technique #1: Greedy Algorithms

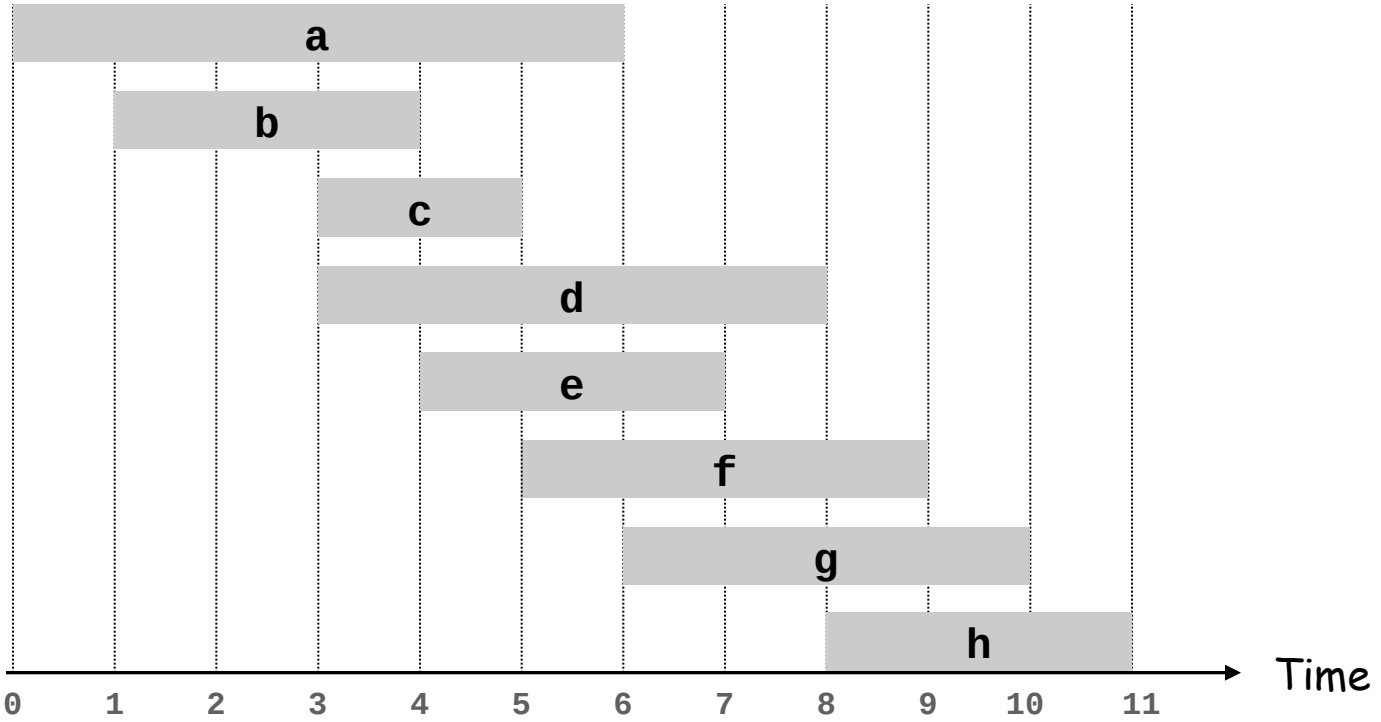
A. Smith; based on slides by E. Demaine, C. Leiserson, S. Raskhodnikova, K. Wayne

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

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 Algorithm

- 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 selected jobs
for j = 1 to n {
   if (job j compatible with A)
      A ← A ∪ {j}
}
return A
```

- Implementation. **O(n log n) time; O(n) space.**
  - Remember job j^* that was added last to A.
  - Job j is compatible with A if $s_j \geq f_{j^*}$.

Running time: O(n log n)

O(n log n)

O(1)

n × O(1)

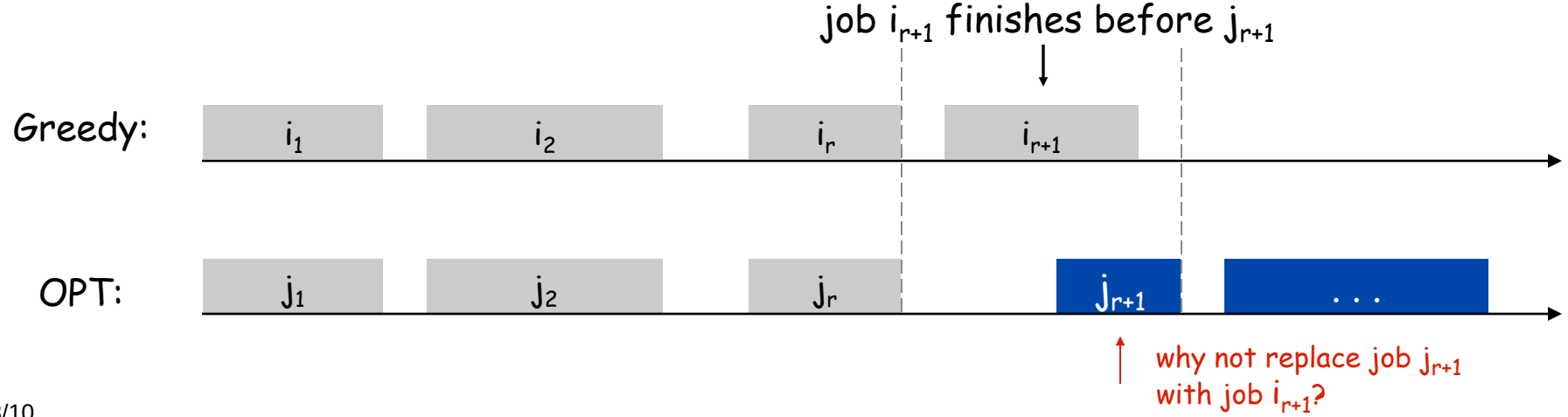
```
Sort jobs by finish times so that
      f1 ≤ f2 ≤ ... ≤ fn.

A ← (empty)    △ Queue of selected jobs
j* ← 0
for j = 1 to n {
   if (fj* <= sj)
      enqueue(j onto A)
}
return A
```

Analysis: Greedy Stays Ahead

- **Theorem**. Greedy algorithm is optimal.
- **Proof strategy** (by contradiction):
  - Suppose greedy is not optimal.
  - Consider an optimal strategy… which one?
    - Consider the optimal strategy that agrees with the greedy strategy for as many initial jobs as possible
  - Look at first place in list where optimal strategy differs from greedy strategy
  - Show a new optimal strategy that agrees more w/ greedy

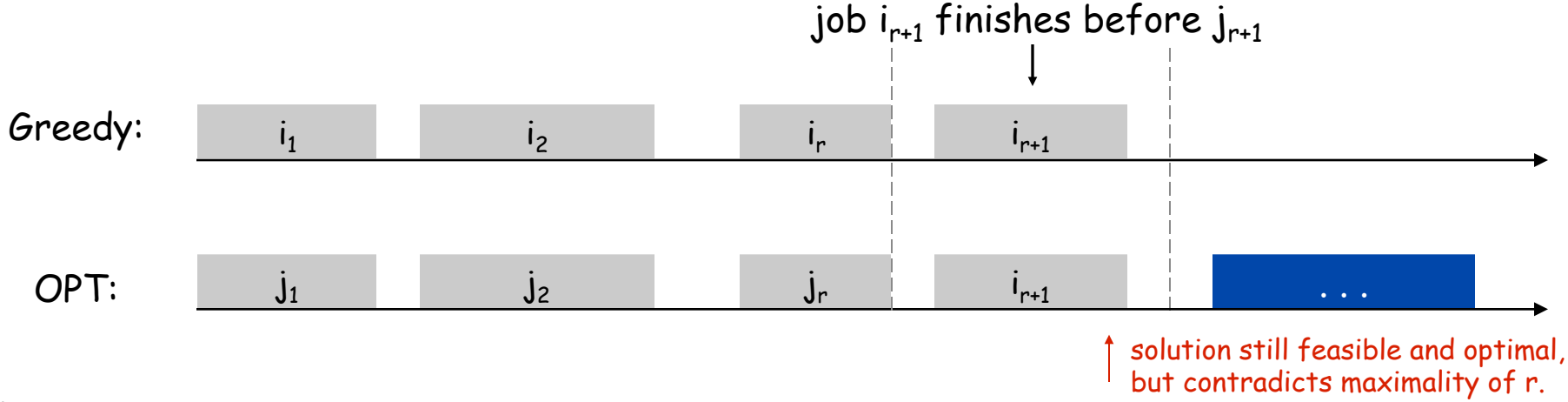
Analysis: Greedy Stays Ahead

- Theorem. Greedy algorithm is optimal.
- Pf (by contradiction): Suppose greedy is 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 set of jobs in the optimal solution with $i_1 = j_1, i_2 = j_2, \ldots, i_r = j_r$ for the largest possible value of r.



Analysis: Greedy Stays Ahead

- Theorem. Greedy algorithm is optimal.
- Pf (by contradiction): Suppose greedy is 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 set of jobs in the optimal solution with $i_1 = j_1, i_2 = j_2, \ldots, i_r = j_r$ for the largest possible value of r.

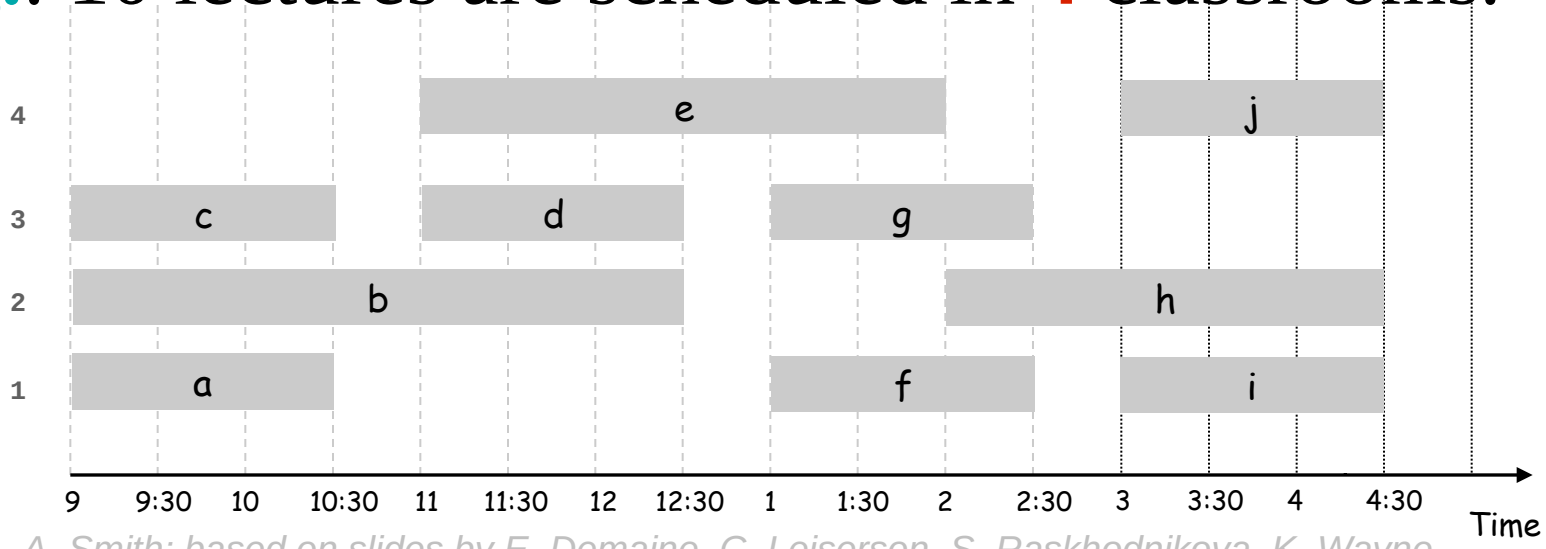


Interval Partitioning

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

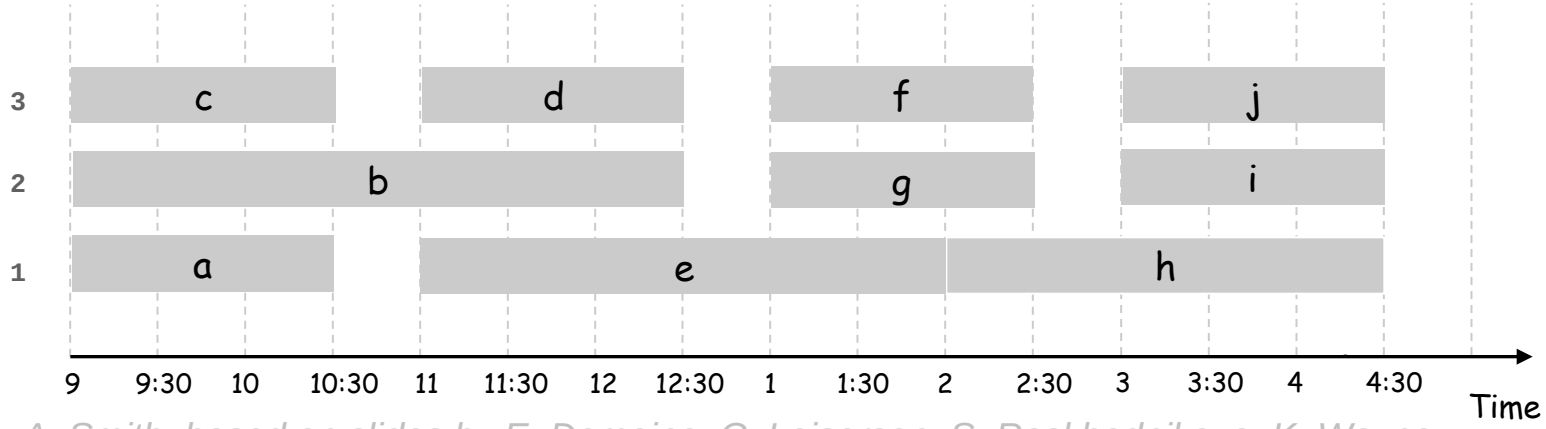
Interval Partitioning

- Lecture j starts at s_j and finishes at f_j.
- **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.



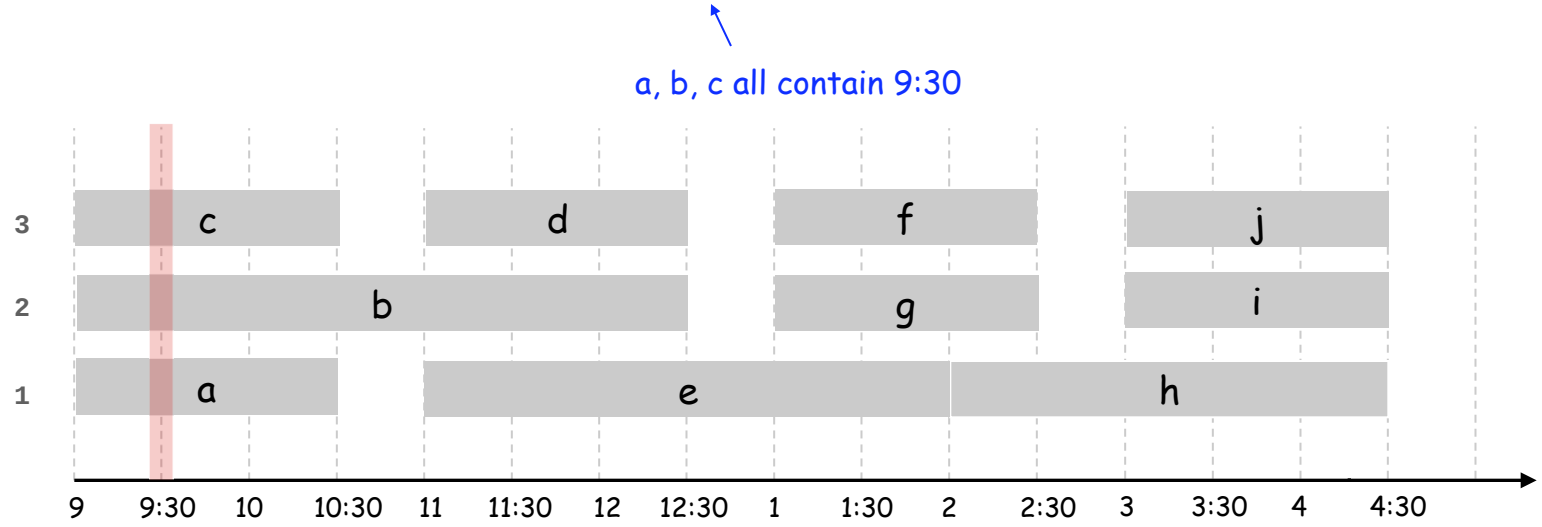
Interval Partitioning

- Lecture j starts at s_j and finishes at f_j.
- **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 are scheduled in **3** classrooms.



Lower Bound

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



- **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 (main loop 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 observation $\Rightarrow$ all schedules use $\geq$ d classrooms. ▪