Objectives

- Greedy Algorithms
  - Interval scheduling
  - Interval partitioning

Review

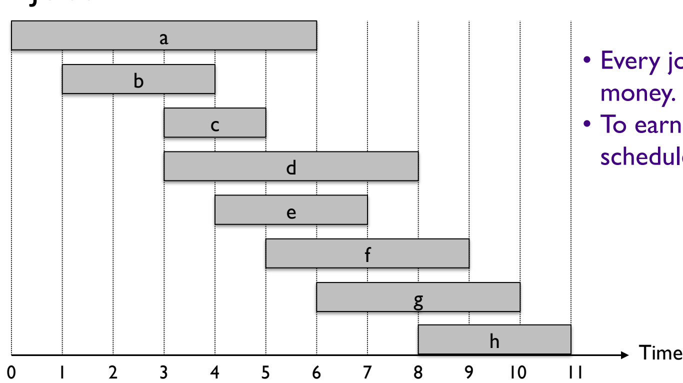
- What is a greedy algorithm?
- What is the template for a greedy algorithm?
- What problem were we trying to solve?
- What orders did we come up with?
  - What approaches didn't work?
    - How did they prove they didn't work?
  - Can you "break" any of the other orders?
    - Find a counterexample to finding the optimal (not necessarily based on our example)

Review: Greedy Algorithms

- Template
  1. Consider candidates in some order
     - Decision: What order is best?
  2. Take each candidate provided it's compatible with the ones already taken
- At each step, take as much as you can get
  - Feasible – satisfy problem's constraints
  - Locally optimal – best local choice among available feasible choices
  - Irrevocable – after decided, no going back

Review: Interval Scheduling

- 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



- Every job is worth equal money.
- To earn the most money → schedule the most jobs

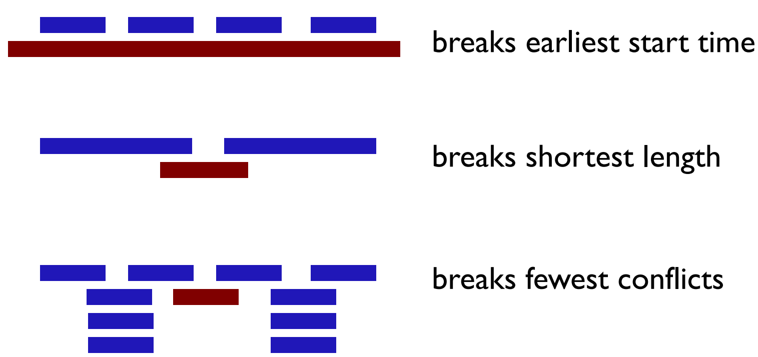
Interval Scheduling

- Earliest start time. Consider jobs in ascending order of start time s_j
  - Utilize CPU as soon as possible
- Earliest finish time. Consider jobs in ascending order of finish time f_j
  - Resource becomes free ASAP
  - Maximize time left for other requests
- Shortest interval. Consider jobs in ascending order of interval length $f_j - s_j$
- Fewest conflicts. For each job, count the number of conflicting jobs c_j. Schedule in ascending order of conflicts c_j

Can we "break" any of these?
i.e., prove they're not optimal?

Counterexamples to Optimality of Various Job Orders

Not optimal when …



Interval Scheduling: Greedy Algorithm

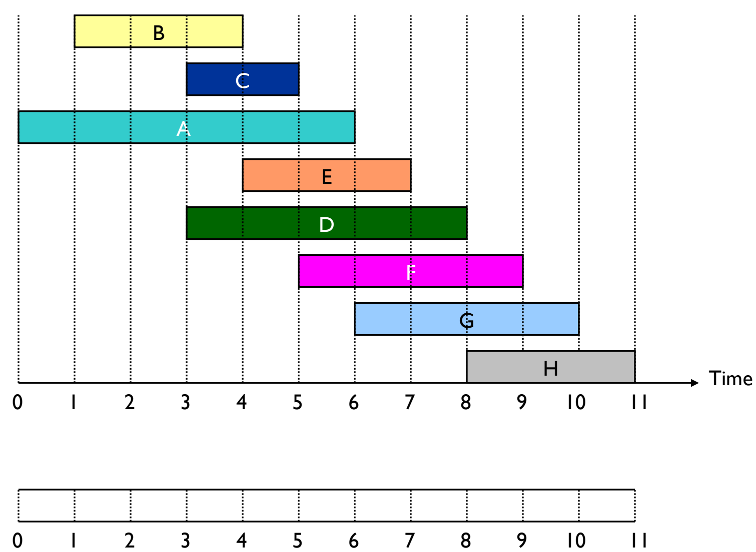
- Consider jobs in increasing order of finish time
- Take each job provided it's compatible with the ones already taken

jobs selected

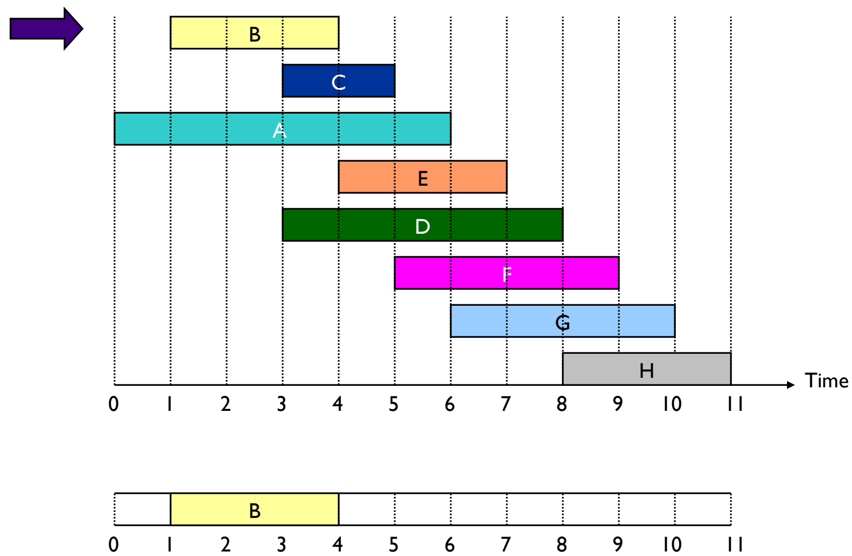
```
Sort jobs by finish times so that f1 ≤ f2 ≤ ... ≤ fn

G = {}
for j = 1 to n
   if job j compatible with G
      G = G ∪ {j}
return G
```

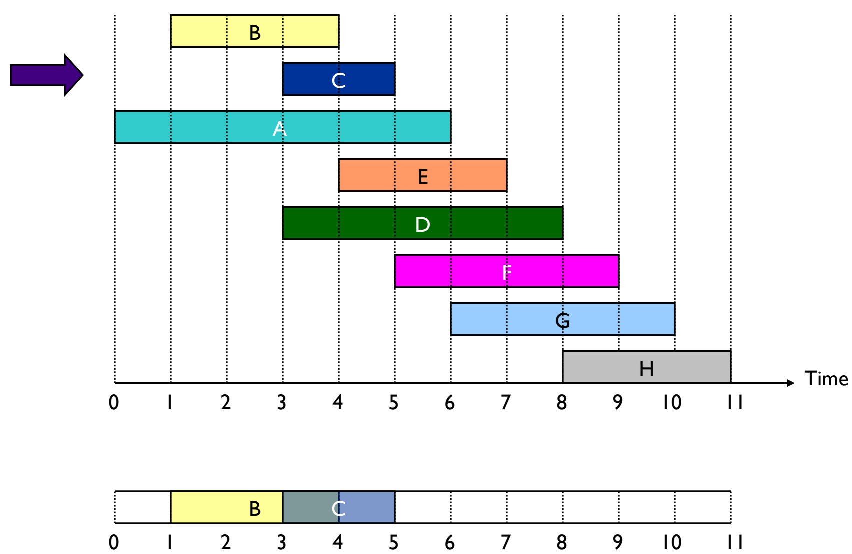

Interval Scheduling



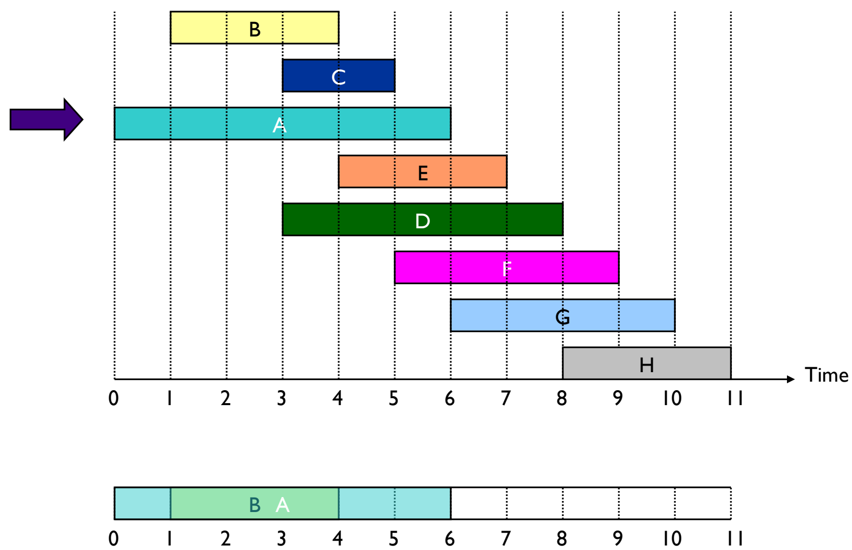
Interval Scheduling



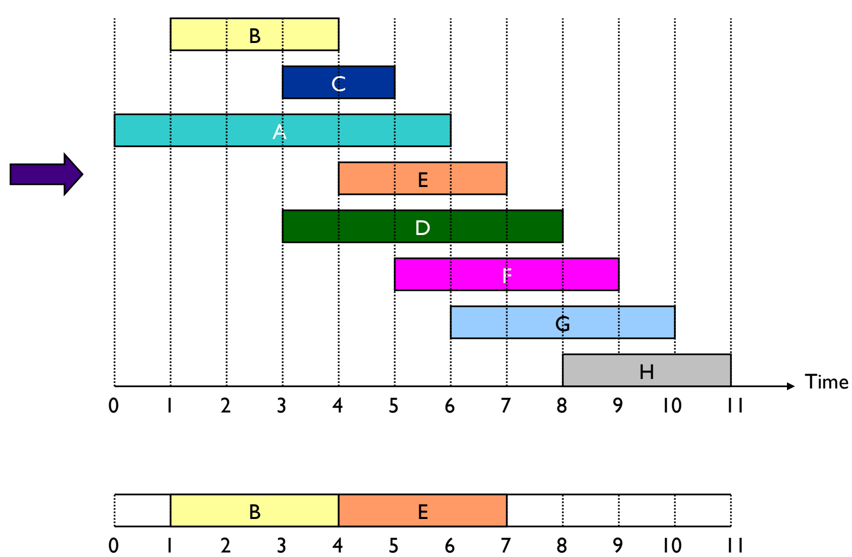
Interval Scheduling



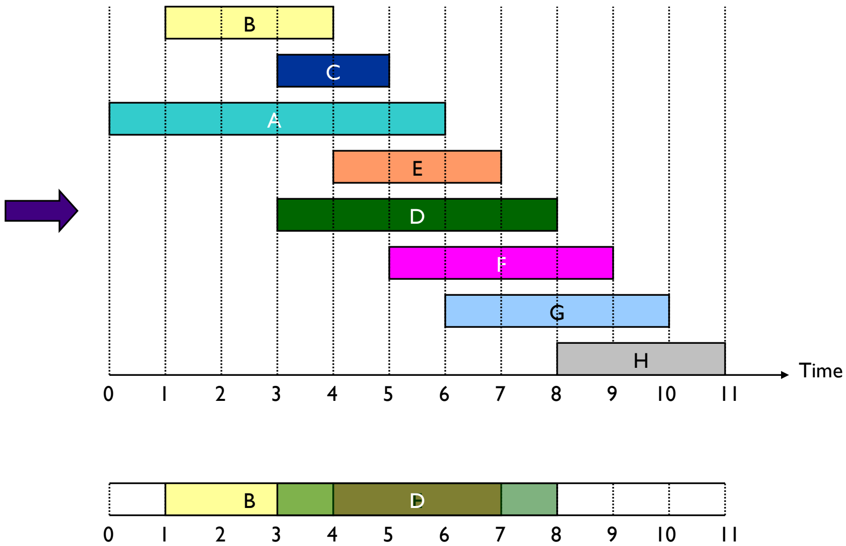
Interval Scheduling



Interval Scheduling

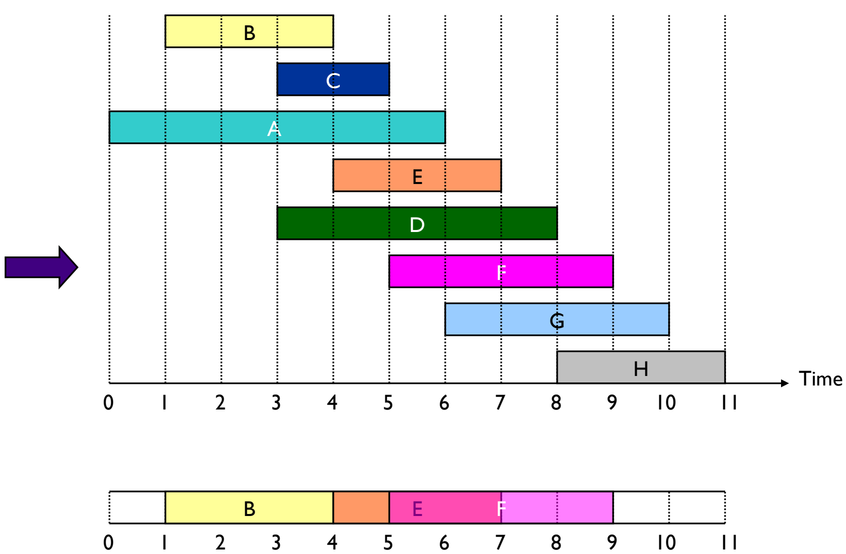


Interval Scheduling



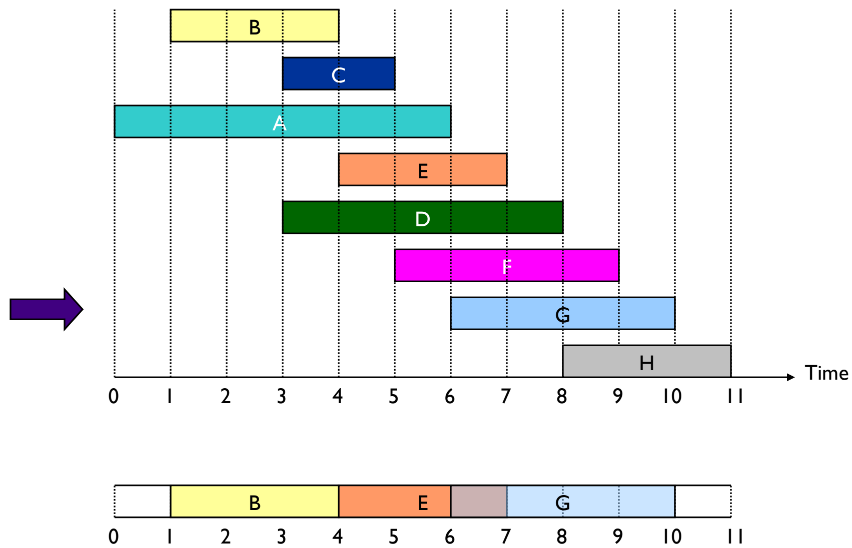
Feb 8, 2019 CSCI211 - Sprenkle 13

Interval Scheduling



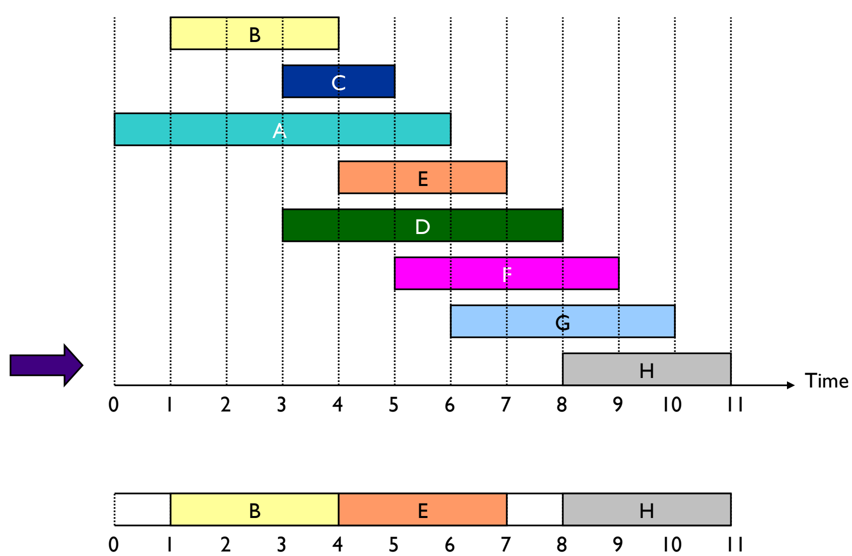
Feb 8, 2019 CSCI211 - Sprenkle 14

Interval Scheduling



Feb 8, 2019 CSCI211 - Sprenkle 15

Interval Scheduling



Feb 8, 2019 CSCI211 - Sprenkle 16

Interval Scheduling: Greedy Algorithm

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

jobs selected

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

G = {}
for j = 1 to n
   if job j compatible with G
      G = G ∪ {j}
return G
```

Runtime of algorithm?
- Where/what are the costs?

Interval Scheduling: Greedy Algorithm

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

O(n logn)

jobs selected

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

G = {}
for j = 1 to n
   if job j compatible with G   O(1)  } O(n)
      G = G ∪ {j}
return G
```

- Implementation. O(n log n)
  - Remember job j* that was added last to G
  - Job j is compatible with G if $s_j \geq f_j$*

Analyzing Interval Scheduling

- Correctness: Know that the intervals are compatible
  - Handled by the if statement
- But is it optimal?
  - What does it mean to be optimal?
  - Recall our goal for maximization

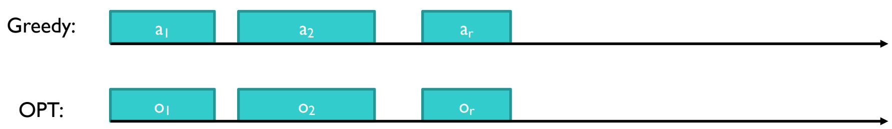
Greedy Stays Ahead Proofs

1. Define your solutions
   - Describe the form of your greedy solution (**A**) and of some other solution (possibly the optimal solution, **O**)
2. Find a measure
   - Find a measure by which greedy ***stays ahead*** of the optimal solution
     - Ex: Let $a_1, \ldots, a_k$ be the first *k* measures of greedy algorithm and $o_1, \ldots, o_m$ be the first *m* measures of optimal solution (sometimes $m = k$)
3. Prove greedy stays ahead
   - Show that greedy's partial solutions constructed are always just as good as the optimal solution's initial segments based on the measure
     - Ex: for all indices $r \leq \min(k,m)$, prove by induction that $a_r \geq o_r$ or $a_r \leq o_r$
   - Use the greedy algorithm to help you argue the inductive step
4. Prove optimality
   - Prove that since greedy stays ahead of the other solution with respect to the measure, then the greedy solution is optimal

→ Make sure maps back to measure of optimality

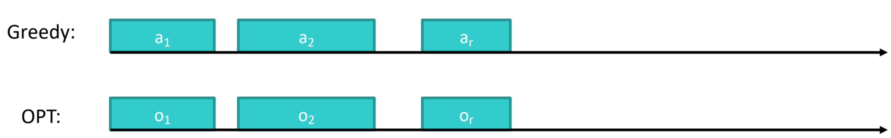
Interval Scheduling: Optimality Analysis

- Theorem. Greedy algorithm is optimal, i.e., schedules the most jobs possible
- Pf. (by contradiction)
  - Assume greedy is not optimal
  - Let $a_1, a_2, \ldots, a_k$ denote set of jobs selected by *greedy* (*k* jobs)
  - Let $o_1, o_2, \ldots, o_m$ denote set of jobs in *optimal* solution (*m* jobs)
  - Both sets ordered by finish time for comparison ordering
  - Want to show that $k = m$



Interval Scheduling: Optimality Analysis

- Theorem. Greedy algorithm is optimal, i.e., schedules the most jobs possible
- Pf. (by contradiction)
  - Assume greedy is not optimal
  - Let $a_1, a_2, \ldots, a_k$ denote set of jobs selected by *greedy* (*k* jobs)
  - Let $o_1, o_2, \ldots, o_m$ denote set of jobs in *optimal* solution (*m* jobs)
  - Both sets ordered by finish time for comparison ordering
  - Want to show that $k = m$



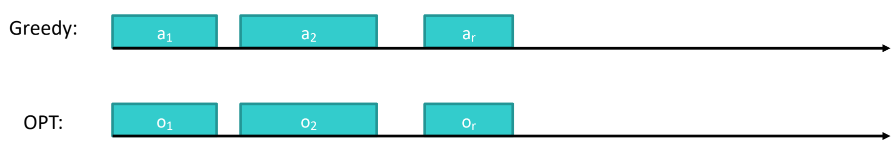
What can we say about a_1 and o_1?

$f(a_1) \leq f(o_1)$

Interval Scheduling: Optimality Analysis

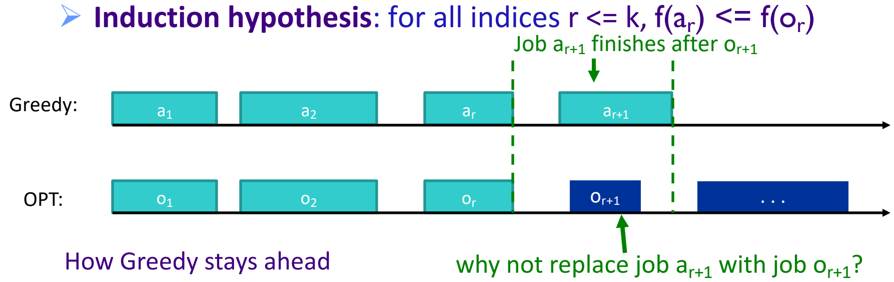
- Theorem. Greedy algorithm is optimal
  - i.e., schedules the most jobs possible
- Pf. (by contradiction)
  - Since we picked the first job to have the first finishing time, we know that $f(a_1) <= f(o_1)$
  - Want to show that Greedy "stays ahead"
    - Each interval finishes at least as soon as Optimal's
  - **Induction hypothesis**: for all indices $r <= k$, $f(a_r) <= f(o_r)$

Prove for r+1



Interval Scheduling: Analysis

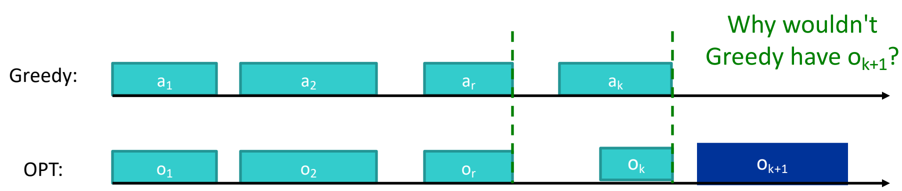
- Theorem. Greedy algorithm is optimal
  - i.e., schedules the most jobs possible
- Pf. (by contradiction)
  - Since we picked the first job to have the first finishing time, we know that $f(a_1) <= f(o_1)$
  - Want to show that Greedy "stays ahead"
    - Each interval finishes at least as soon as Optimal's
  - **Induction hypothesis**: for all indices $r <= k$, $f(a_r) <= f(o_r)$



How Greedy stays ahead

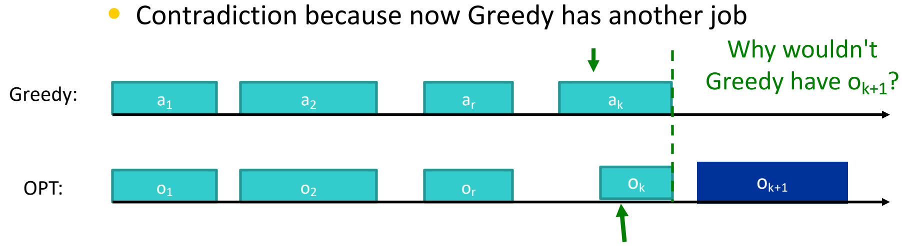
Interval Scheduling: Analysis

- Theorem. Greedy algorithm is optimal.
  - i.e., schedules the most jobs possible
- Pf. (by contradiction)
  - Assume Greedy is not optimal (i.e., m > k)
    - Optimal solution has more jobs than Greedy
  - We already showed that for all indices $r \le k$, $f(a_r) \le f(o_r)$
  - Since m > k, there is a request o_{k+1} in Optimal



Interval Scheduling: Analysis

- Theorem. Greedy algorithm is optimal.
  - i.e., schedules the most jobs possible
- Pf. (by contradiction)
  - Assume Greedy is not optimal (i.e., m > k)
  - We already showed that for all indices $r \le k$, $f(a_r) \le f(o_r)$
  - Since m > k, there is a request o_{k+1} in Optimal
    - Starts after o_k ends → after a_k ends
  - So, Greedy could *also* add o_{k+1}
    - Contradiction because now Greedy has another job



Problem Assumptions

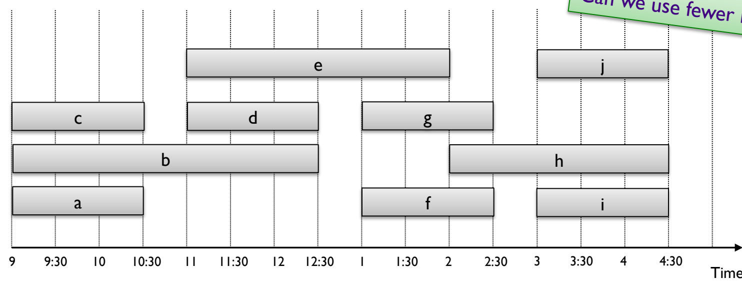
- All requests were known to scheduling algorithm
  - Online algorithms: make decisions without knowledge of future input
- Each job was worth the same amount
  - What if jobs had *different* values?
    - E.g., scaled with size
- Single resource requested
  - Rejected requests that didn't fit

INTERVAL PARTITIONING

Interval Partitioning

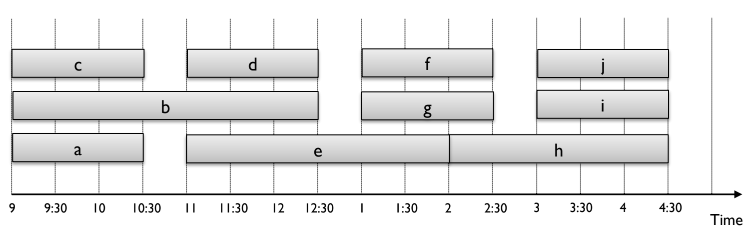
- Lecture j starts at s_j and finishes at f_j
- Goal: find minimum number of classrooms to schedule all lectures so that no two occur at the same time in the same room.
- Ex: 10 lectures in 4 classrooms

What are our constraints?
Can we use fewer rooms?



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.
- Alternative schedule uses only 3 classrooms

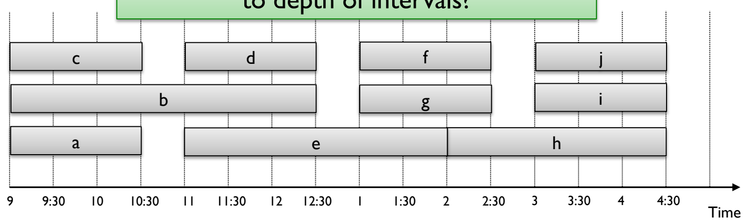


Interval Partitioning: Lower Bound on Optimal Solution

- Def. The depth of a set of open intervals is the maximum number that contain any given time.
- Key observation. # of classrooms needed ≥ depth.
- Ex: Depth of schedule below = 3 $\Rightarrow$ schedule below is optimal.

a, b, c all contain 9:30

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



Interval Partitioning Discussion

- Does there always exist a schedule equal to depth of intervals?
- Can we make decisions locally to get a global optimum?
  - Or are there long-range obstacles that require more resources?

Interval Partitioning: 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
```

Analyze algorithm

Interval Partitioning: 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: $O(n \log n)$
  - For each classroom k, maintain the finish time of the last job added.
  - Keep the classrooms in a priority queue by last job finish time.

Exam 1 – due next Friday at 5 p.m.

- Open
  - Your brain
  - Your notes, wiki
  - Handouts
  - My posted slides, course web site
  - Sakai forum for our class (posted solutions)
  - Me (more limited than with problem sets)
- Closed – everything else
- Start early! No extensions

Next Week

- No class on Wednesday
  - Work on exam
- No wiki
- Office Hours
  - M: 2:30-5 p.m.
  - W: 9:45 – 11:30 a.m. (class time), 2:30-5 p.m.
  - R: 2:30-5 p.m.
  - And by appointment
- Try to rotate – limit of 10 minutes