CS 401: Computer Algorithm I

Interval Scheduling / Interval Partitioning

Xiaorui Sun

Stuff

Homework 2 is out (due at 11:59pm February 23 Friday)

Submit your source code for each problem to gradescope via blackboard

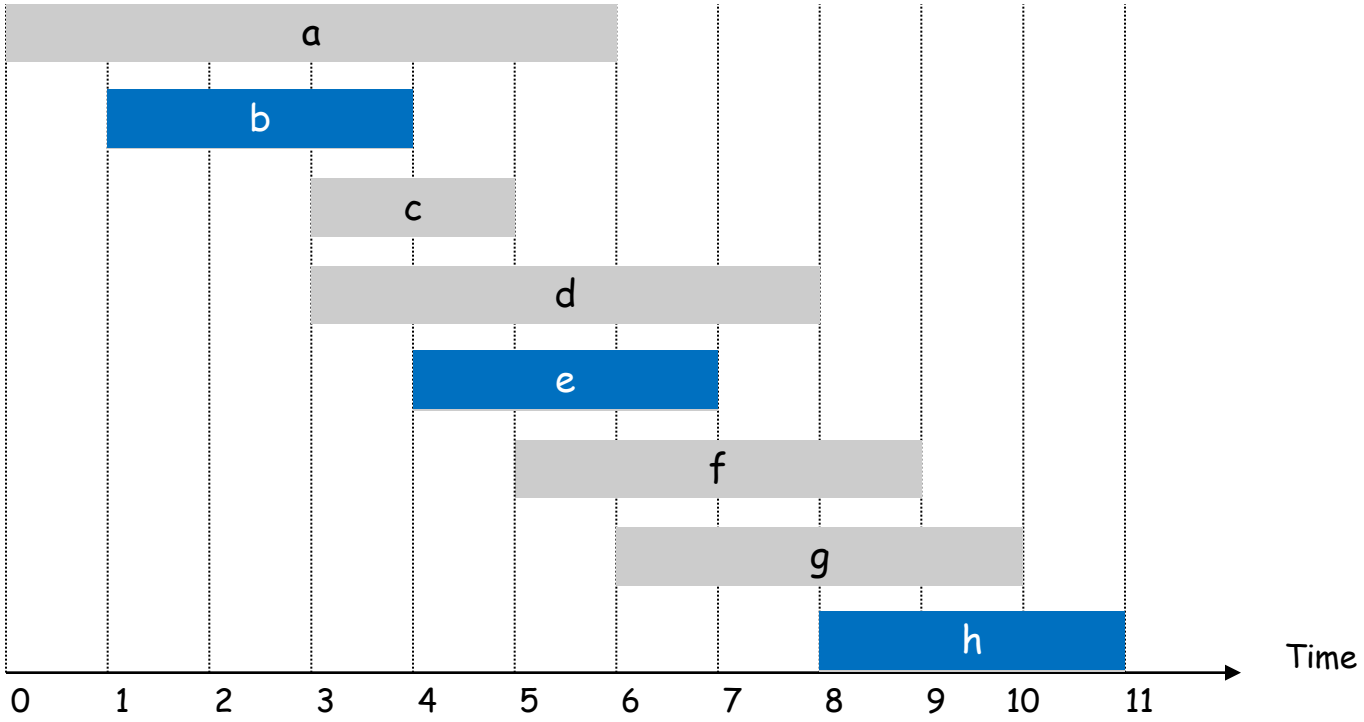
Greedy Algorithm

- Greedy algorithm:
  - Divide solution construction into small steps
  - Decision in each step: ‘best’ current partial solution at each step

- Recipe:
  - Order the input in some way (the most ‘important’ element will be considered first)
  - Go through the input according to the order
  - Determine the strategy to construct best current partial solution in each step

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 ← ∅
for j = 1 to n {
   if (job j compatible with A)
      A ← A ∪ {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

- The output is compatible. (This is by construction.)

How to show it gives maximum number of jobs?

Let $i_1, i_2, i_3, \cdots$ be jobs picked by greedy (ordered by finish time)

Let $j_1, j_2, j_3, \cdots$ be an optimal solution (ordered by finish time)

How about proving $i_k = j_k$ for all k?

No, there can be multiple optimal solutions.

Idea: Prove that greedy outputs the "best" optimal solution.

Given two compatible orders, which is better?

The one finish earlier.

How to prove greedy gives the "best"?

Induction: it gives the "best" during every iteration.

Greedy stays ahead: At each step any other solution has a worse value for some criterion that eventually implies optimality

This example: criterion = finish time

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}.

Lesson

Order is important for greedy algorithms

- In general, the order gives priorities to different elements (the most important element is ordered first)
- This example: the job can be finished earliest is the most important job because finishing this job gives more freedom to finish other jobs
- If you want to solve a problem by greedy, first think about what is the “right” order of the elements

Greedy stays ahead

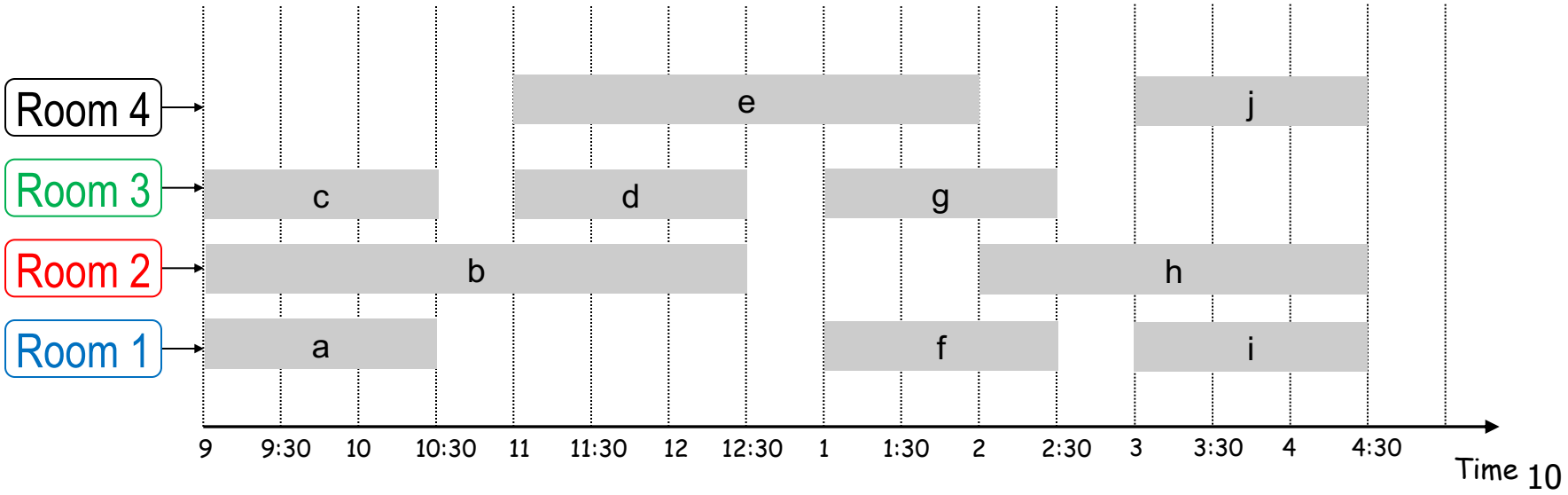
- A useful strategy to argue why the solution is the best

Interval Partitioning Technique: Structural

Interval Partitioning

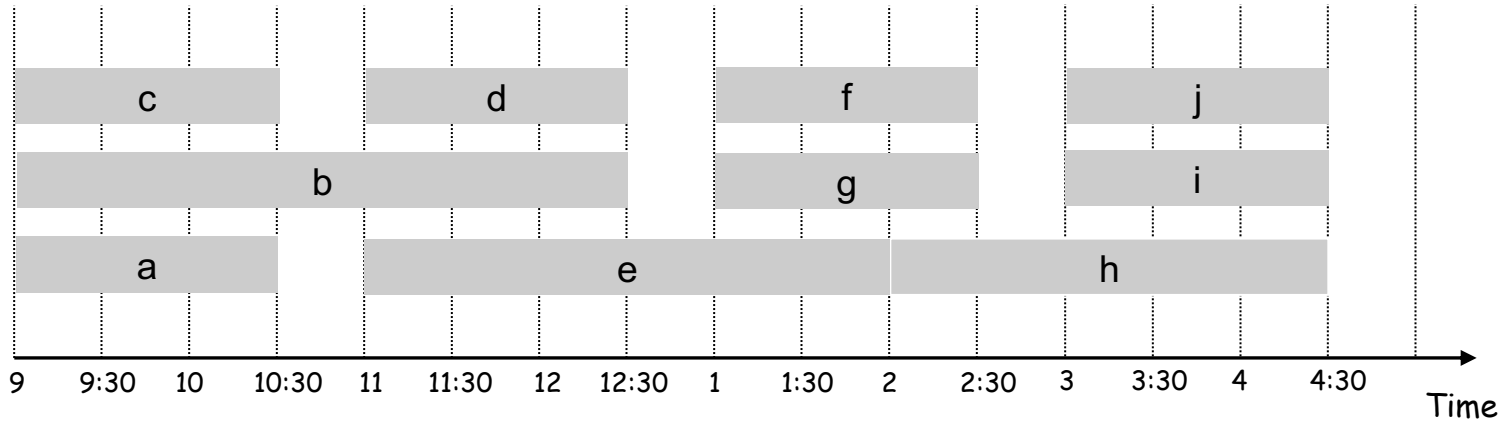
Lecture j starts at $s(j)$ and finishes at $f(j)$.

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



A Better Schedule

This one uses only 3 classrooms

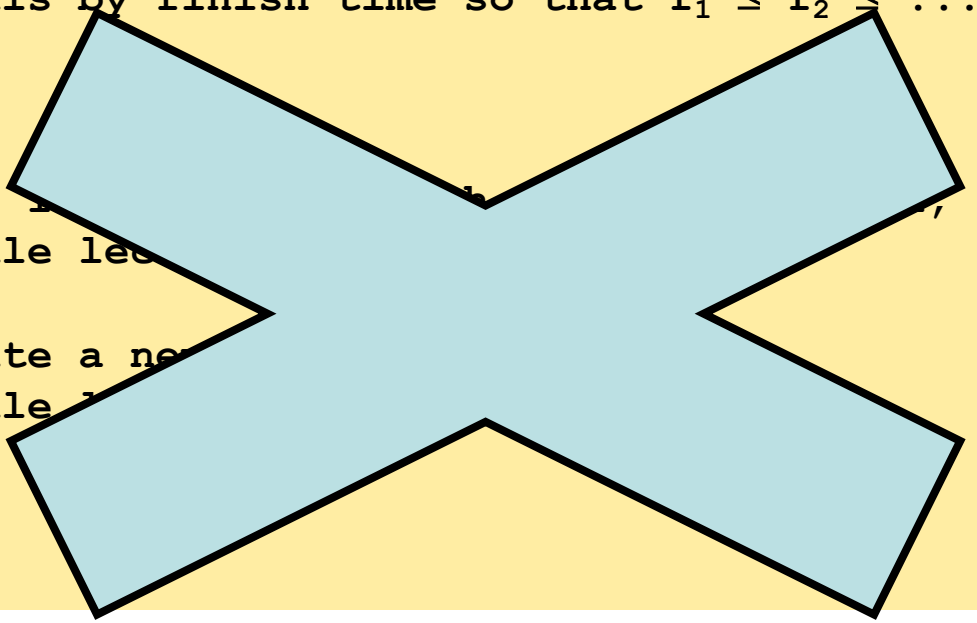


A Greedy Algorithm

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

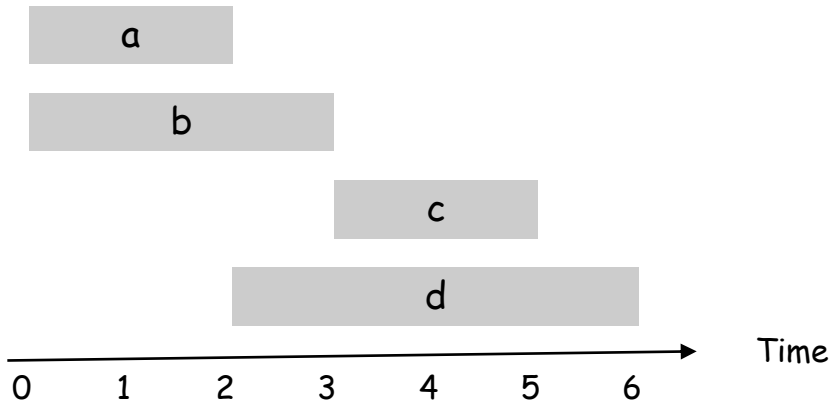
```
Sort intervals by finish time so that f1 ≤ f2 ≤ ... ≤ fn.
d ← 0

for j = 1 to
   if (let j
      schedule lec
   else
      allocate a ne
      schedule
      d ← d
}
```

$1 \leq k \leq d$)

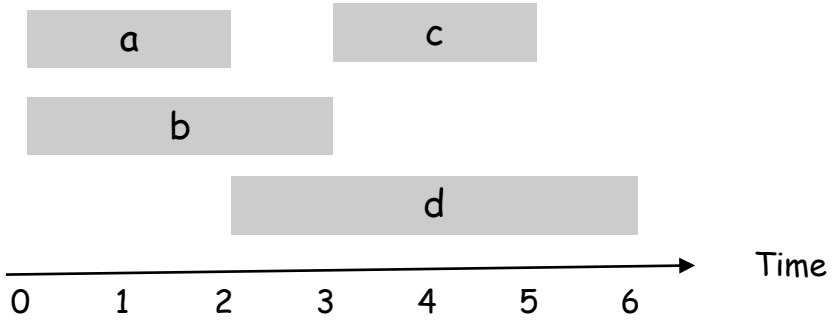


Correctness: This is wrong!

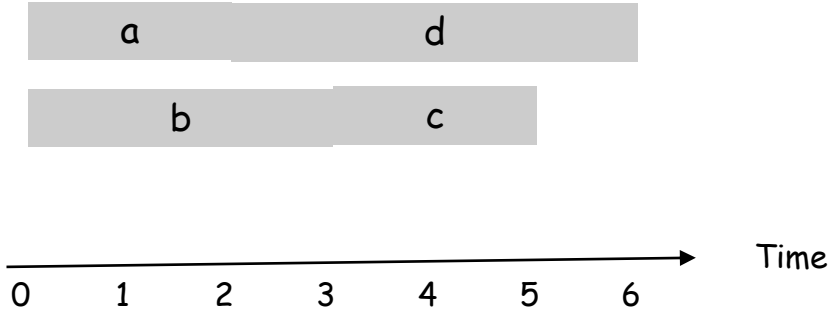
Example



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 s1 ≤ s2 ≤ ... ≤ sn.
d ← 0

for j = 1 to n {
   if (lect j is compatible with some classroom k, 1 ≤ k ≤ 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!

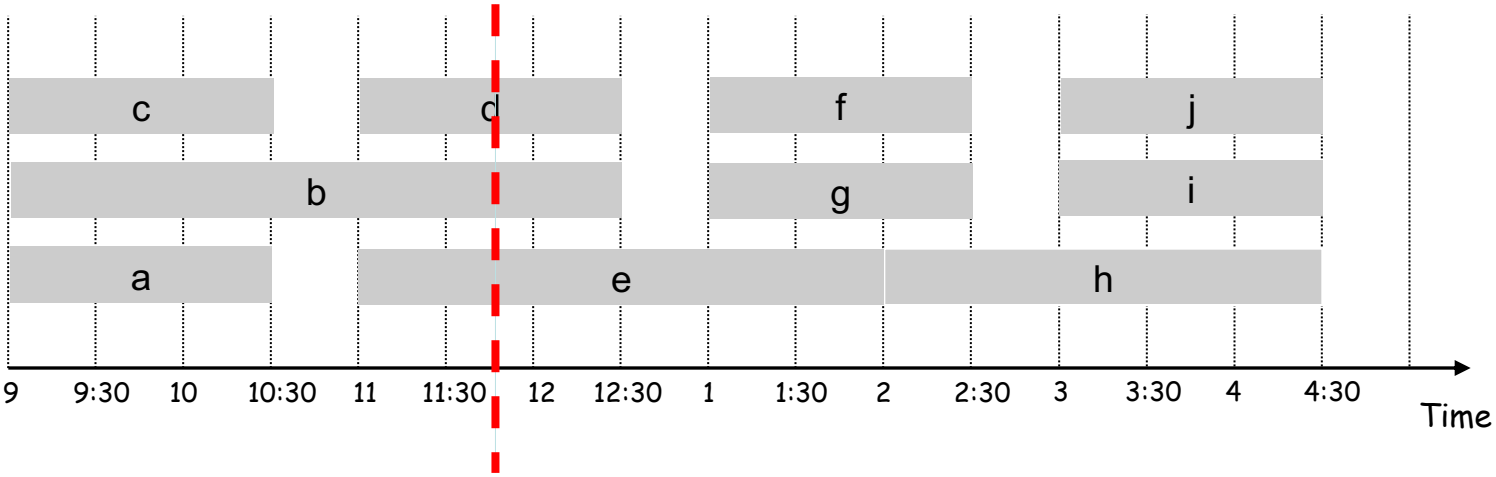
A Structural Lower-Bound on OPT

Def. The depth of a set of intervals is the maximum number that contain 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

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

Theorem: Greedy algorithm is optimal.

Proof (exploit structural property).

Let d = number of classrooms that the greedy algorithm allocates.

Classroom d is opened because we needed to schedule a job, say j, that is incompatible with all $d - 1$ previously used classrooms.

Since we sorted by start time, all these incompatibilities are caused by lectures that start no later than $s(j)$.

Thus, we have d lectures overlapping at time $s(j) + \epsilon$, i.e. depth $\geq d$

"OPT Observation" $\Rightarrow$ $d \geq$ depth,

so d = depth and greedy is optimal