

CSE 421

Greedy Algorithms / Interval Scheduling

Yin Tat Lee

Summary for last two classes

- Terminology: vertices, edges, paths, connected component, tree, bipartite...
- Vertices vs Edges: $m = O(n^2)$ in general, $m = n - 1$ for trees
- BFS: Layers, queue, shortest paths, all edges go to same or adjacent layer
- DFS: recursion/stack; all edges ancestor/descendant
- Algorithms: Connected Comp, bipartiteness, topological sort
- Techniques: Induction on vertices/layers
- Sorry that the algorithm 2 for topological sort is wrong.
See the email for the fix

Tentative Schedule

Today – Apr 16: Greedy Method

Apr 18 – Apr 27: Divide and Conquer

Apr 30: Midterm

May 2 – May 9: Dynamic Programming

May 11 – May 21: Network Flow

Rest: NP completeness

Greedy Algorithms

- Hard to define exactly but can give general properties
 - Solution is built in small steps
 - Decisions on how to build the solution are made to maximize some criterion without looking to the future
 - Want the ‘best’ current partial solution as if the current step were the last step
- May be more than one greedy algorithm using different criteria to solve a given problem

Greedy Strategy

Goal: Given currency denominations: 1, 5, 10, 25, 100, give change to customer using *fewest* number of coins.

Ex: 34¢.



Cashier's algorithm: At each iteration, give the *largest* coin valued \leq the amount to be paid.

Ex: \$2.89.



Greedy is not always Optimal

Observation: Greedy algorithm is sub-optimal for US postal denominations: 1, 10, 21, 34, 70, 100, 350, 1225, 1500.

Counterexample. 140¢.

Greedy: 100, 34, 1, 1, 1, 1, 1, 1.

Optimal: 70, 70.

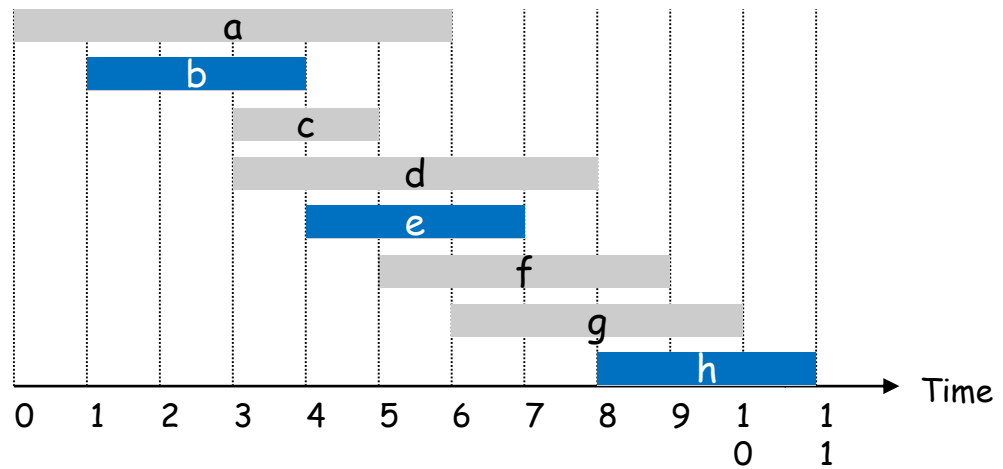


Lesson: Greedy is short-sighted. Always chooses the most attractive choice at the moment. But this may lead to a dead-end later.

Greedy Algorithms

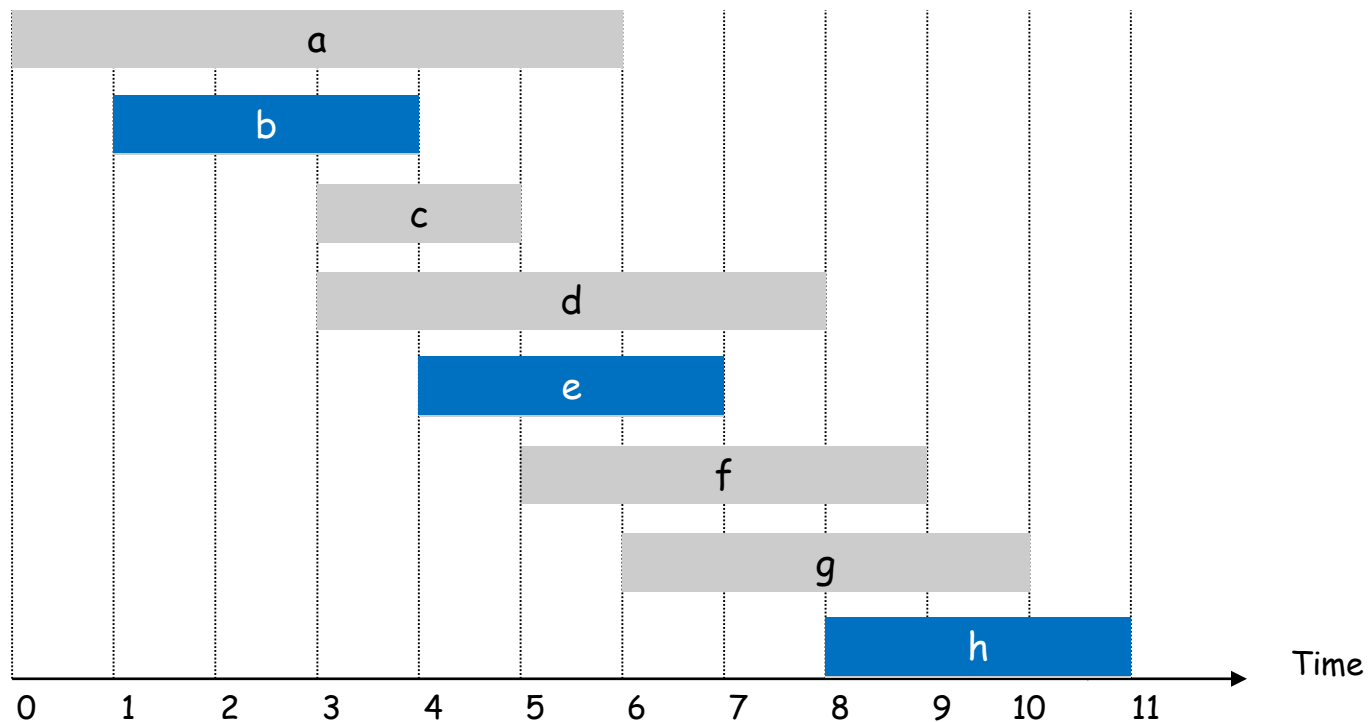
- Greedy algorithms
 - Easy to produce
 - Fast running times
 - Work only on certain classes of problems
 - Hard part is showing that they are correct
- Two methods for proving that greedy algorithms do work
 - Greedy algorithm stays ahead
 - At each step any other algorithm will have a worse value for some criterion that eventually implies optimality
 - Exchange Argument
 - Can transform any other solution to the greedy solution at no loss in quality

Interval Scheduling



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 Strategy

Sort the jobs in **some** order. Go over the jobs and take as much as possible provided it is compatible with the jobs already taken.

Main question:

- What order?
- Does it give the Optimum answer?
- Why?

Possible Approaches for Inter Sched

Sort the jobs in **some** order. Go over the jobs and take as much as possible provided it is compatible with the jobs already taken.

[Shortest interval] Consider jobs in ascending order of interval length $f(j) - s(j)$.

[Earliest start time] Consider jobs in ascending order of start time $s(j)$.

[Earliest finish time] Consider jobs in ascending order of finish time $f(j)$.

Greedy Alg: Earliest Finish Time

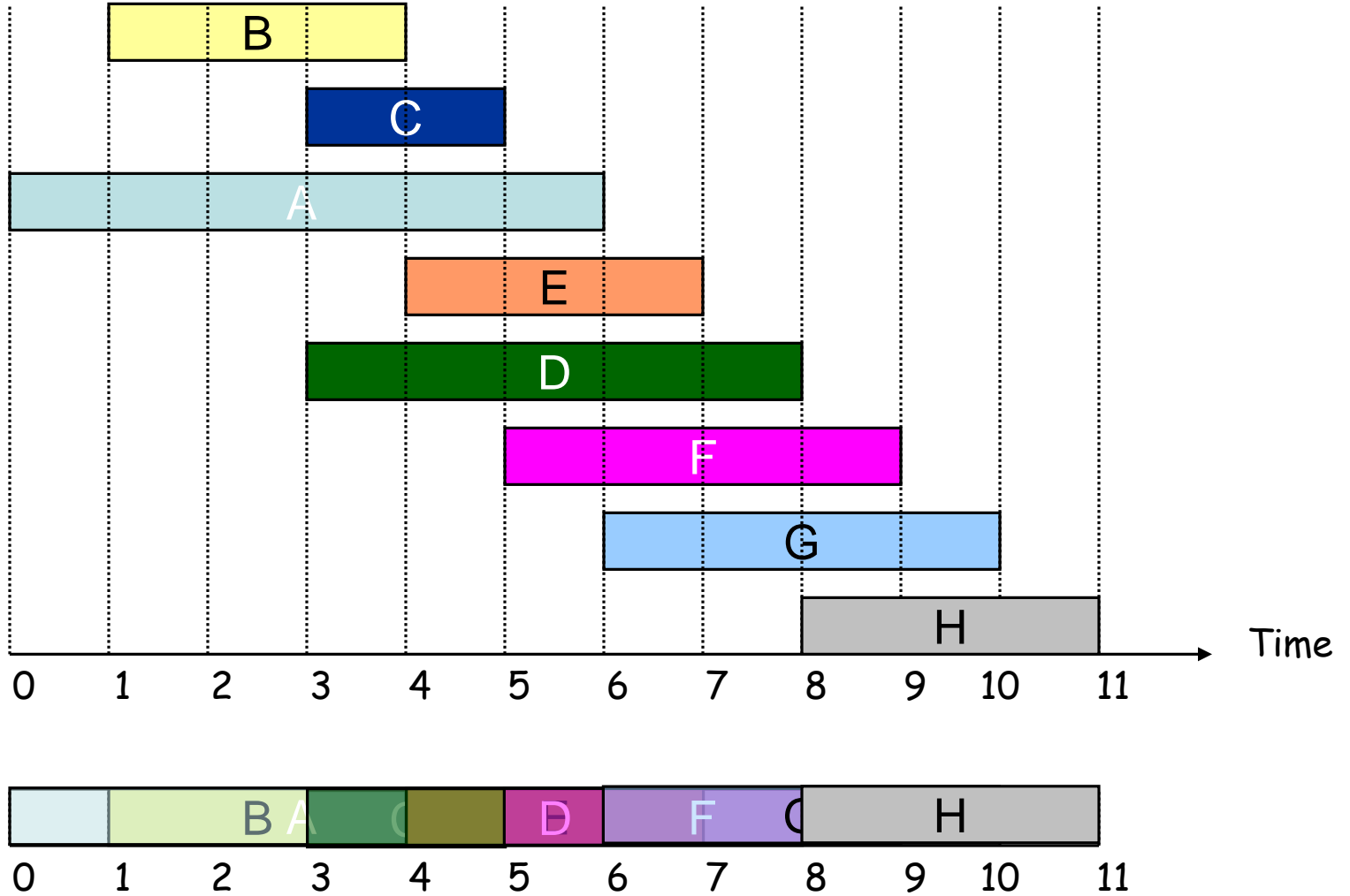
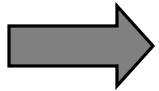
Consider jobs in increasing order of finish time. Take each job provided it's compatible with the ones already taken.

```
Sort jobs by finish times so that  $f(1) \leq f(2) \leq \dots \leq f(n)$ .  
 $A \leftarrow \emptyset$   
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 $s(j) \geq f(j^*)$.

Greedy Alg: Example



Correctness

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

How to prove it gives maximum number of jobs?

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

Let j_1, j_2, j_3, \dots 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.

Correctness

Theorem: Greedy algorithm is optimal.

Proof: (technique: “Greedy stays ahead”)

Let $i_1, i_2, i_3, \dots, i_k$ be jobs picked by greedy, $j_1, j_2, j_3, \dots, j_m$ 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 $f(i_{r+1}) \leq f(j_{r+1})$

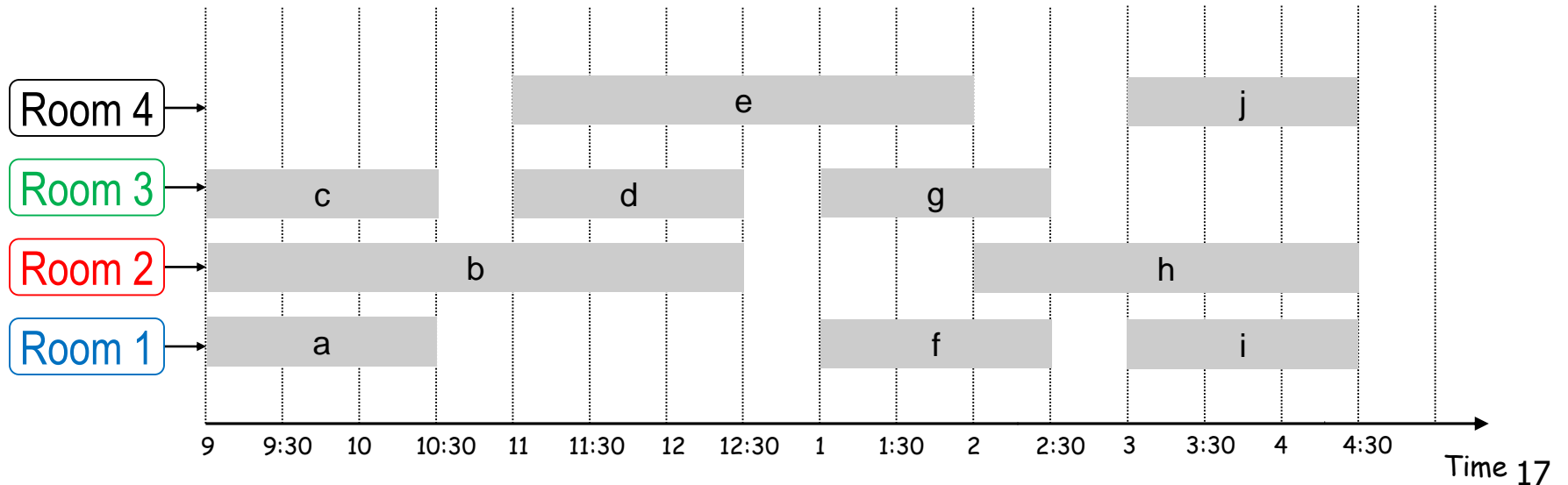
Observe that we must have $k \geq m$, else j_{k+1} is among (nonempty) set of candidates for i_{k+1} .

Interval Partitioning Technique: Structural

Interval Partitioning

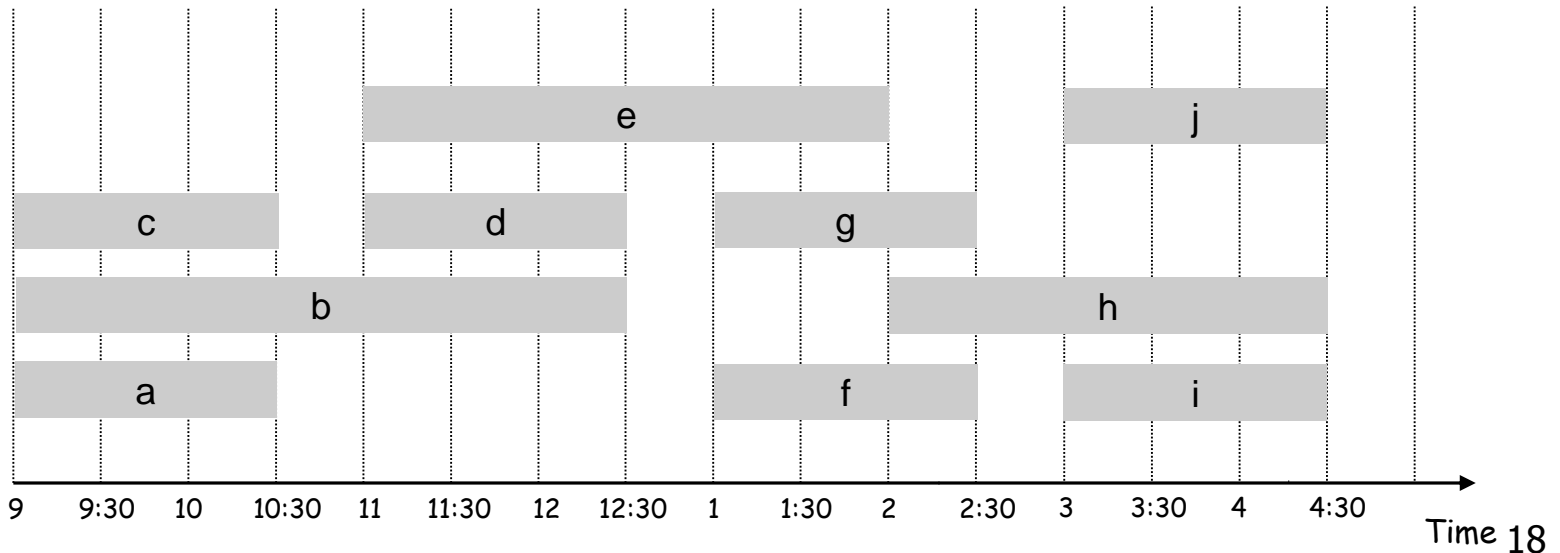
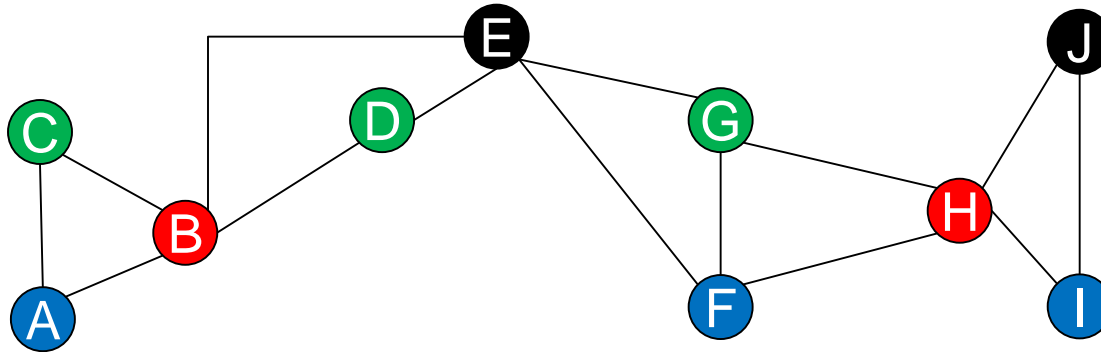
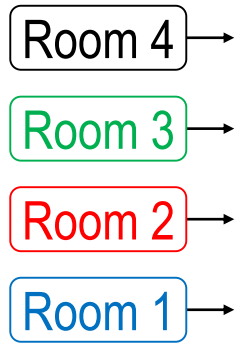
Lecture j starts at $s(j)$ and finishes at $f(j)$.

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



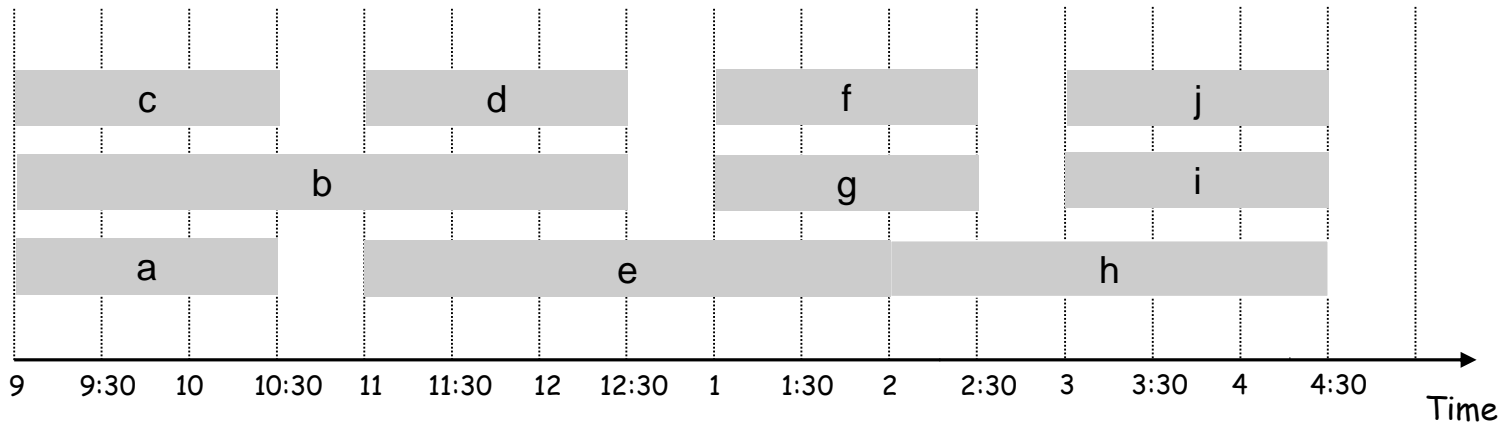
Interval Partitioning

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



A Better Schedule

This one uses only 3 classrooms



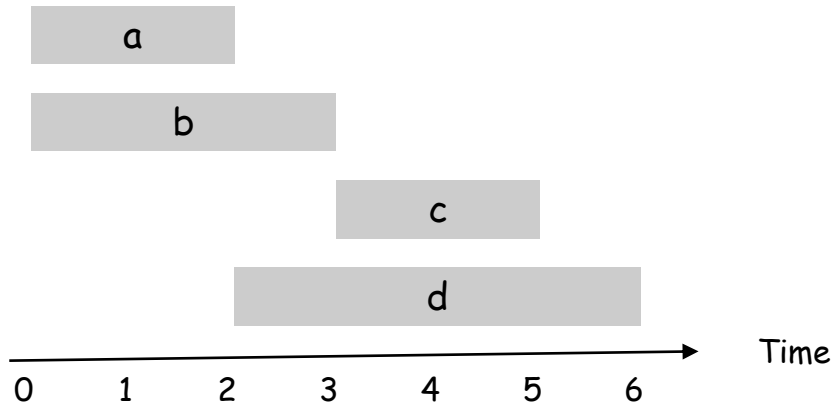
A Greedy Algorithm

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

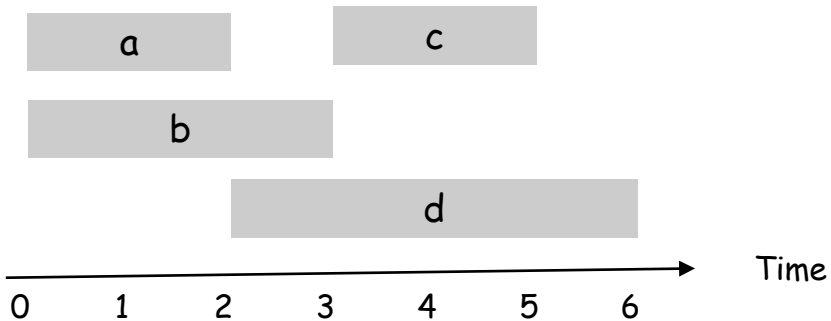
```
Sort intervals by finish time so that  $f_1 \leq f_2 \leq \dots \leq f_n$ .  
d  $\leftarrow$  0  
  
for j = 1 to n  
  if (lect j compatible with any classroom,  $1 \leq k \leq d$ )  
    schedule lect j in classroom k  
  else  
    allocate a new classroom  
    schedule lect j in this classroom  
    d  $\leftarrow$  d + 1  
}
```

Correctness: This is wrong!

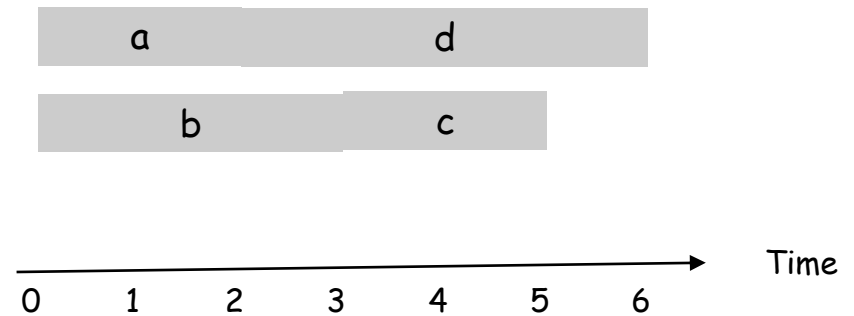
Example



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  $s_1 \leq s_2 \leq \dots \leq s_n$ .  
d  $\leftarrow$  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  $\leftarrow$  d + 1  
}
```

Implementation: Exercise!

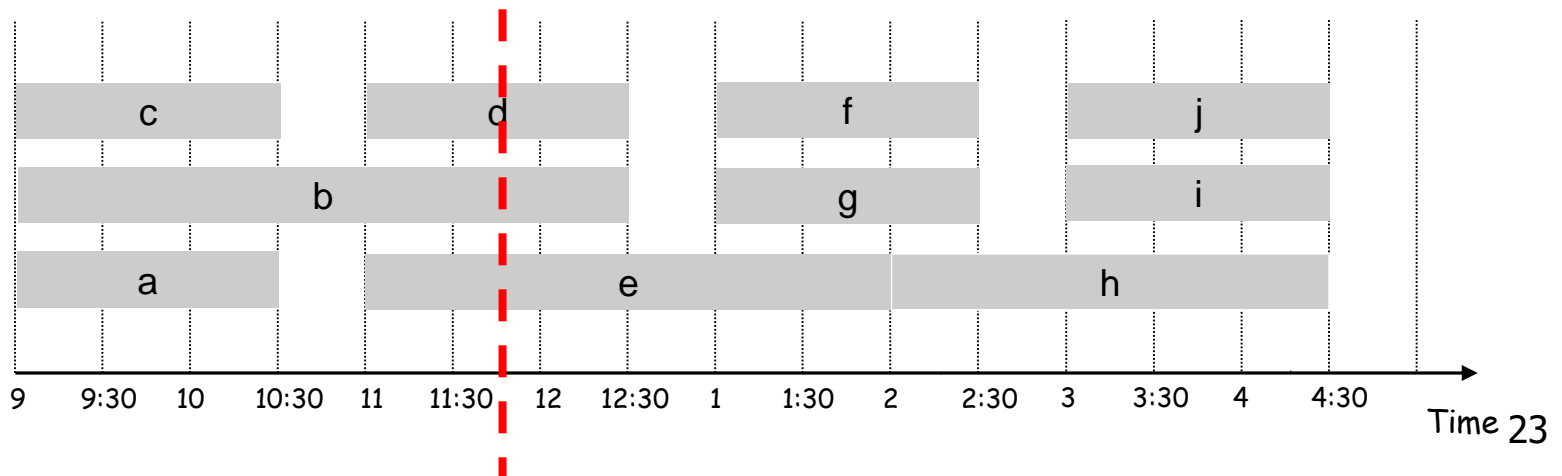
A Structural Lower-Bound on OPT

Def. The **depth** of a set of open 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. $\text{depth} \geq d$

“OPT Observation” \Rightarrow all schedules use $\geq \text{depth}$ classrooms,

so $d = \text{depth}$ and greedy is optimal ▪