

COL 351:

Analysis and Design of Algorithms

Lecture 2

Algorithm Paradigms

1. Divide and Conquer

- Divide the problem into smaller problems
- Solve the smaller problems
- Combine

2. Dynamic Programming

Reduce the problem on an input of size n into problems of size $n - 1, n - 2, n - 3, \dots$ etc.

3. Greedy Strategy

Build solution greedily.

Job-Scheduling

Computer Science Fest



Goal: Attend maximum number of seminars in CS Fest

$|opt| = 5$

Job Scheduling

$$J_i = [s_i, t_i]$$

Formal Definition

Given: A collection n jobs, $\{J_1 = [s_1, t_1], \dots, J_n = [s_n, t_n]\}$.
A single server.

Constraint: If job J_i is scheduled on server, then it occupies the server for time-interval $[s_i, t_i]$

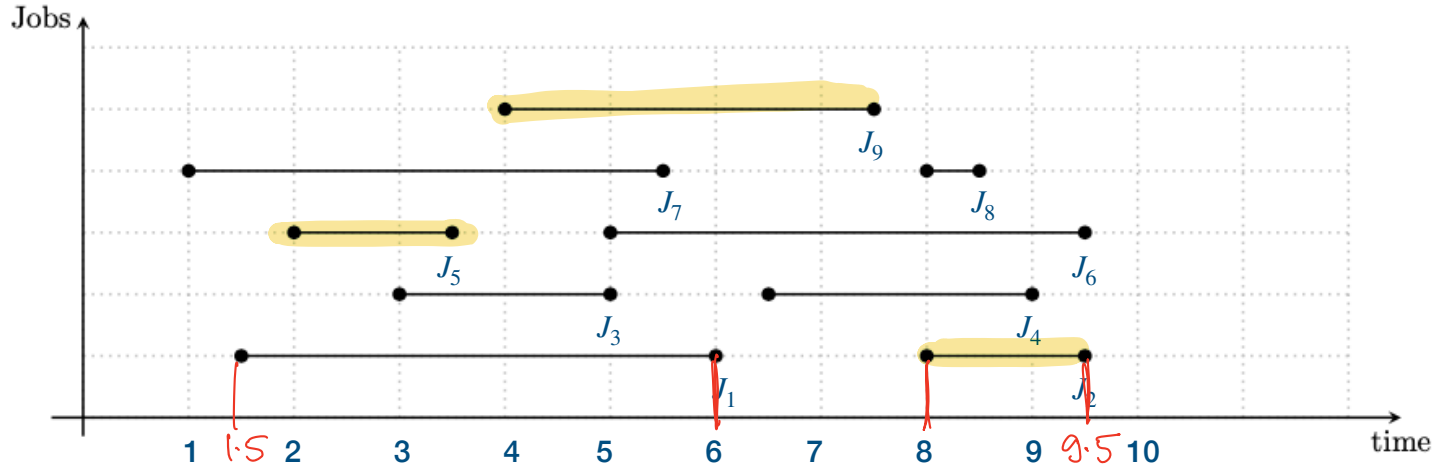
Aim: Find a maximum subset $S \subseteq \{J_1, \dots, J_n\}$ of non-overlapping jobs.

$$J_{set} := \{J_1, \dots, J_n\}$$

Example

Goal - $O(n^c)$, $c \geq 0$

$n=9$



All 2^n subsets

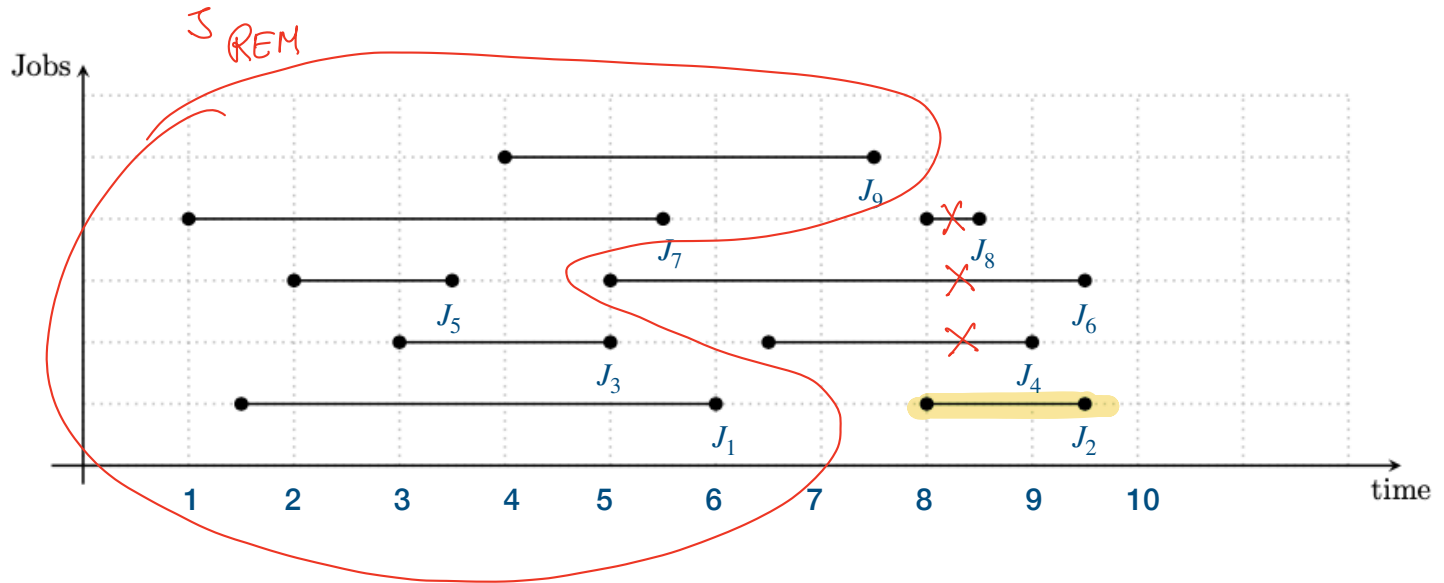
$$\text{Time} = \Omega(2^n)$$

TRIVIAL
APPROACH?



Example

$\exists$ an opt solⁿ containing job J_2

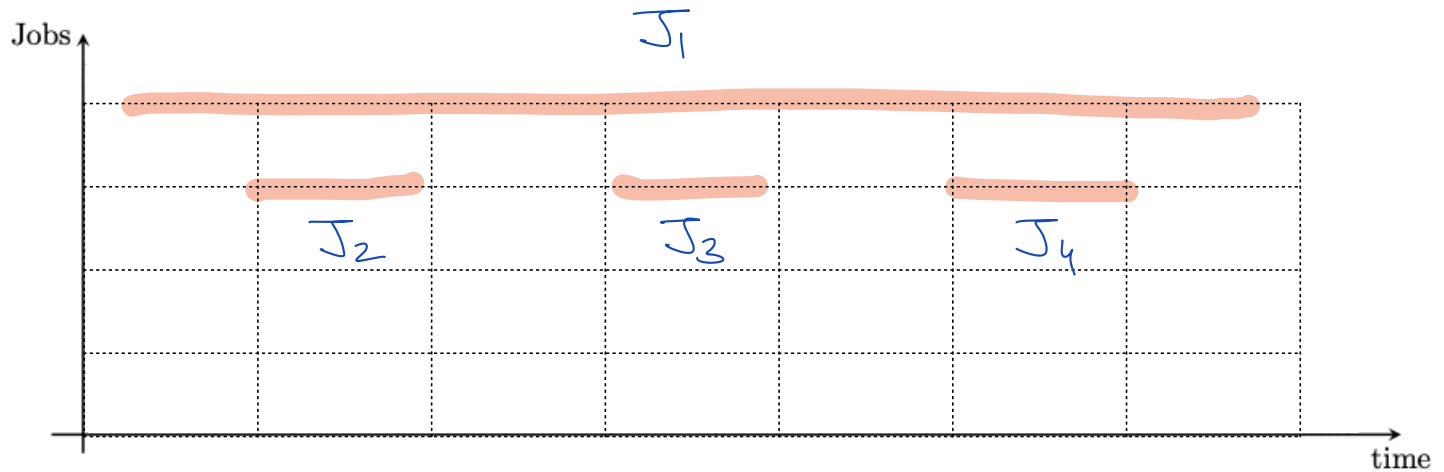


Main Idea: Greedily find one job that lies in an optimal solution.

Ques: How to find a job in the set $\{J_1 \dots J_n\}$
that lies in the opt solⁿ.

Which Greedy Strategy works?

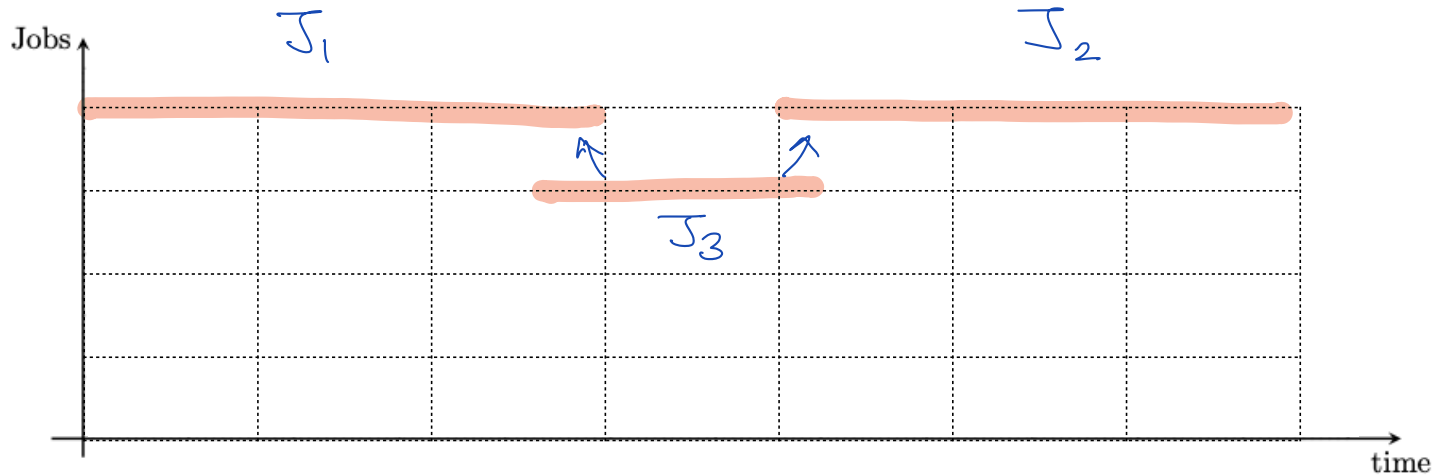
Ques. Can we select a job that **arrives first**, i.e., has smallest s_i ? No



$$\text{opt} = \{ J_2 \ J_3 \ J_4 \}$$

Which Greedy Strategy works?

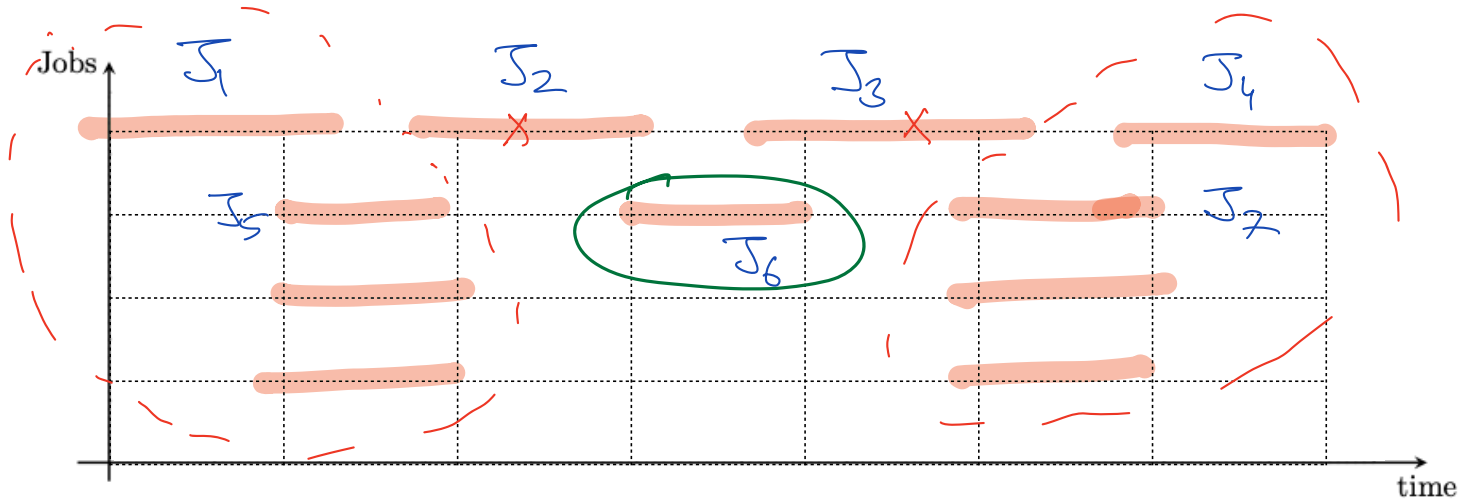
Ques. Can we select a job with **smallest duration**, i.e., with smallest value of $(t_i - s_i)$? *No*



$$\text{opt} = \{ J_1, J_2 \}$$

Which Greedy Strategy works?

Ques. Can we select a job with **minimum overlap**?



$$\text{opt} = \{ J_1, J_2, J_3, J_4 \}$$

Which Greedy Strategy works?

Ques. Can we select a job that **finishes earliest**, i.e., has smallest value of t_i ?

Yes.

Greedy Strategy - Choose job with earliest finish time

Lemma: Let $J_0 \in J_{set}$ be job with earliest finish time. Then there exists an optimal solution containing J_0 .

$\equiv$ set of all jobs.

Proof:



$S = \text{any opt sol}^n$

$J^* = \text{job with earliest F.T. in } S$

$$\text{F.T.}(J_0) \leq \text{FT}(J^*)$$

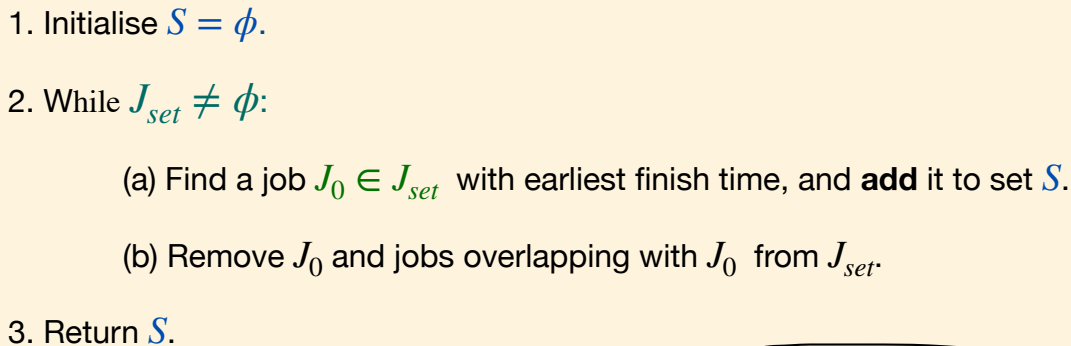
$$S_{\text{new}} = (S \setminus J^*) + J_0$$

Jobs in S_{new} are NON-OVERLAPPING

$|S_{\text{new}}| = |S| \Rightarrow S_{\text{new}}$ must be opt.

J_0 = Job with earliest F.T.

$$J_{rem} = J \setminus \text{OVERLAP}(J_i)$$



What will
be
implementation
time?

Schedule(J_{set})

$$\text{opt}(\mathcal{I}_{\text{set}}) * O(n)$$

Correctness

Theorem : Let $\underline{J_0} \in J_{set}$ be job with earliest finish time, and $J_{rem} = J_{set} \setminus \text{Overlap}(J_0)$. Then,

$$OPT(J_{set}) = OPT(J_{rem}) + 1.$$

$$LHS \leq RHS \quad | \quad LHS \geq R.H.S.$$

Correctness

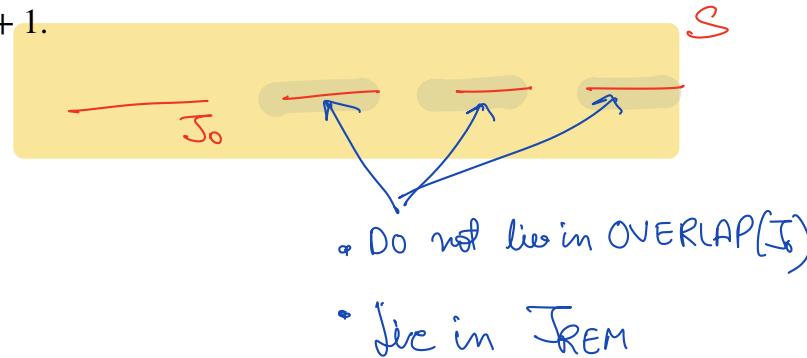
$$J_{set} = \{J_1, \dots, J_n\}$$

Theorem : Let $J_0 \in J_{set}$ be job with earliest finish time, and $J_{rem} = J_{set} \setminus \text{Overlap}(J_0)$. Then,

$$OPT(J_{set}) = OPT(J_{rem}) + 1.$$

Proof Part 1: We will show $OPT(J_{set}) \leq OPT(J_{rem}) + 1$.

$S = \text{opt sol}^n$ containing job J_0



CLAIM(1) $S \setminus J_0 \subseteq J_{rem}$

claim (2) $S \setminus J_0$ is non-overlapping.

$$\text{opt}(J_{rem}) \geq |S \setminus J_0| = |S| - 1 = \text{opt}(J_{set}) - 1$$

HINT - $\exists$ an opt solⁿ of J_{set} containing job J_0

Correctness

Theorem : Let $J_0 \in J_{set}$ be job with *earliest finish time*, and $J_{rem} = J_{set} \setminus \text{Overlap}(J_0)$. Then,

$$OPT(J_{set}) = OPT(J_{rem}) + 1.$$

Proof Part 2: We will show $OPT(J_{set}) \geq \underline{OPT(J_{rem}) + 1}$.

S^* = opt solⁿ of J_{rem}

$S^* \cup \{J_0\}$ — Non overlapping

$$\underline{opt(J_{set})} \geq S^* \cup \{J_0\} = |S^*| + 1 = \underline{opt(J_{rem}) + 1}$$

Homework Exercises

- Suppose the n jobs $J_1, \dots, J_n$ satisfy that $t_1 \leq \dots \leq t_n$.

Then design an $O(n)$ time algorithm to compute an optimal scheduling.

① $O(n \log n)$ time

② $\ell_i, t_i \in \{1, 2, 3, \dots, cn\}$ — Bucket sorting

-
- Design an algorithm to find an optimal scheduling with **two servers**. $\Rightarrow O(n)$

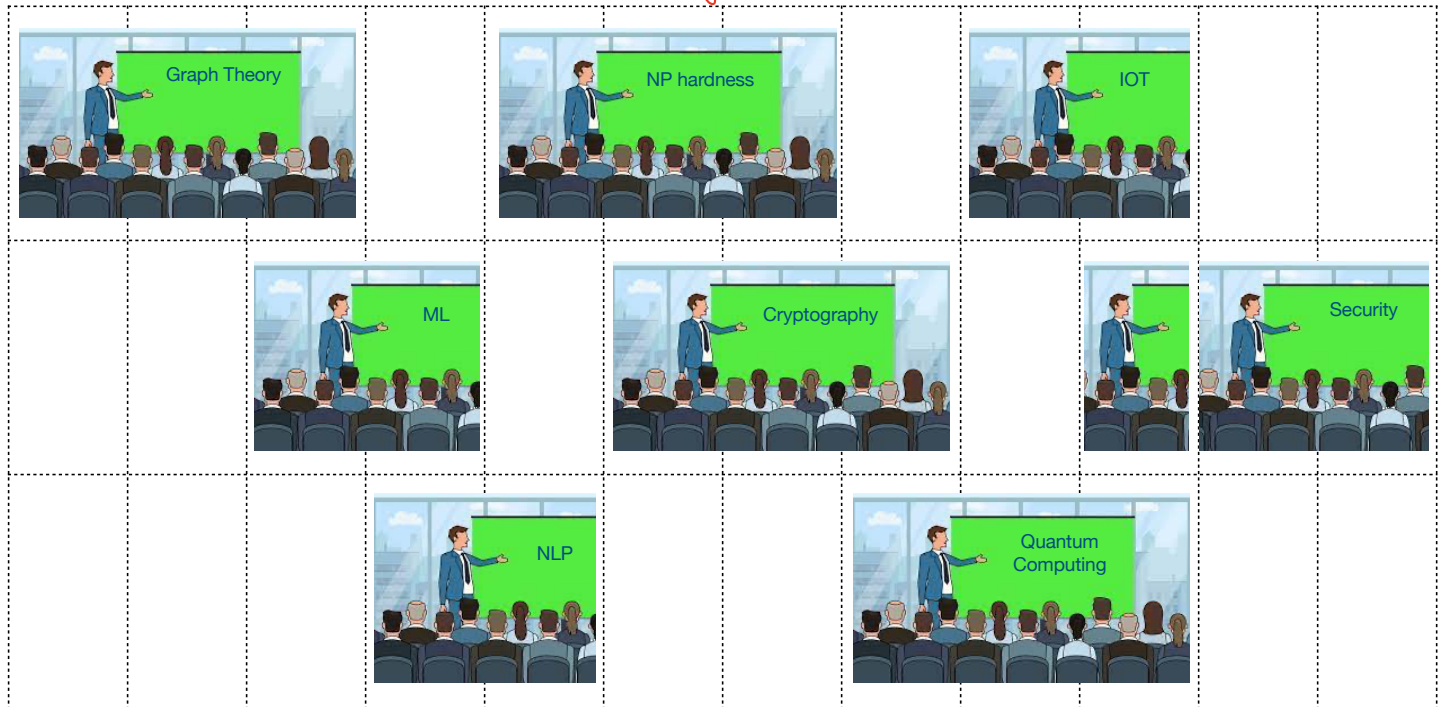
k servers.

$k = 5, 7$

Scheduling with two servers:

How you and your friend can in total attend maximum possible number of seminars in CS Fest?

Maximum possible coedicty $\rightarrow |sem(you) \cup sem(friend)|$



10 AM

11 AM

12 PM

1 PM

2PM

3 PM

4 PM

$$n=9$$

$$|\text{set } \mathcal{S}^n| = 8$$

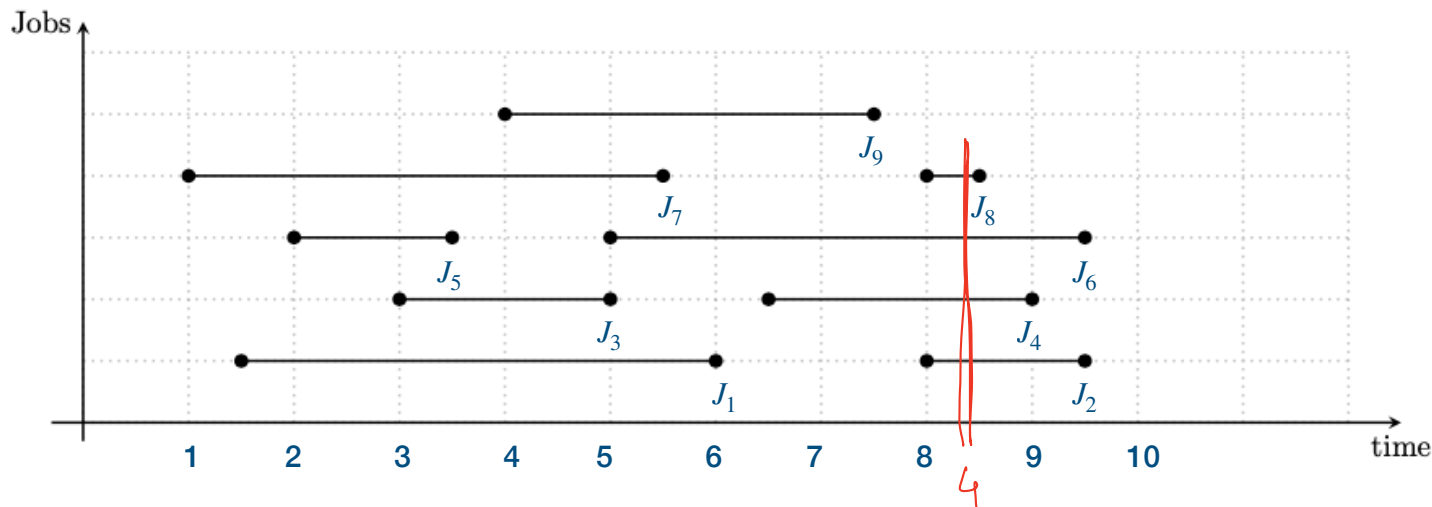
Unique Solution??

- Design an algorithm to check whether a collection of n given jobs has a **unique** optimal scheduling, with respect to **one given server**.

Challenge Problem (EASY)

Given : A collection of n jobs, $\{J_1 = [s_1, t_1], \dots, J_n = [s_n, t_n]\}$.

Find : Minimum number of servers required to schedule all jobs. $\geq$ max-overlap
?
=



What is the best possible time complexity?