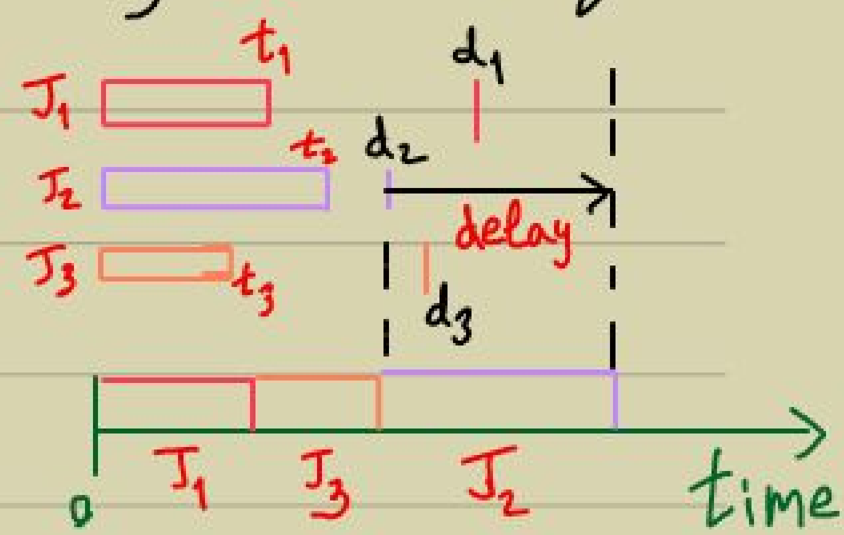
Job Scheduling

Input:
- There are n jobs $\{J_1, \ldots, J_n\}$.
- Job J_i takes t_i time to complete.
- Job J_i has deadline d_i.

Output: Schedule them on a single server such that the maximum delay is minimized.

- **Qn:** Is this a hard problem?



- **Ideas:** 1) Schedule in the order of t_i's.
  2) " " " " d_i's.

- Counterexample for Idea-1:
  $\{J_2, J_3\}$ above. It is better if J_2 is scheduled first.

▷ For two jobs, Idea-2 always works.

Proof: • Let $\{J_1, J_2\}$ be the jobs.
• If J_1 is scheduled first, the delay is $t_1+t_2-d_2$.
• If J_2 is scheduled first, then the delay is $t_1+t_2-d_1$.
$\Rightarrow$ delay is minimized if we schedule according to the earliest deadline. $\square$

- **Qn:** What to do with $n>2$ jobs?

- Consider a scheduling in the order $(J_1, J_2, \dots, J_n)$. Focus on $\{J_i, J_{i+1}\}$.
  • If $d_i > d_{i+1}$ then swapping them gives us a better scheduling.

$\Rightarrow$ Thus, in an optimal schedule we have $d_1 \leq d_2 \leq \dots \leq d_n$.

Theorem: Job Scheduling (min max delay) can be done in $O(n \lg n)$ time.

Greedy Paradigm

- In the last algorithm we used a local approach to get a global one.
  (From $n=2$ to $n>2$.)

- Given an optimization problem P, with instance A of size n:
  - Greedy step identifies an instance A' of size $n' < n$.
  - In the Proof you are required to formally show that:

A lemma is needed → $OPT(A')$ follows from $OPT(A)$,
& $OPT(A)$ " " $OPT(A')$.

This is what gives the pseudocode.

- Greedy paradigm is a very powerful technique.
  In the end, you only need sorting.