DS 320 Algorithms for Data Science — **Lecture #9**
Spring 2026 — Prof. Kira Goldner and Jeffrey Considine

Greedy Exchange II: Scheduling to Minimize Lateness

In the problem of scheduling to minimize lateness, we have n scheduling requests. Request i has a deadline d_i and requires time t_i to process the job. We'll assign start time s_i and finish time f_i to job i. Let lateness $\ell_i := f_i - d_i$. The goal is to minimize the maximum lateness $\max_i \ell_i$.

Ideas for Greedy Metrics:

- increasing length

  counterexample: $(t_1 = 1, d_1 = 100), (t_2 = 10, d_2 = 10)$

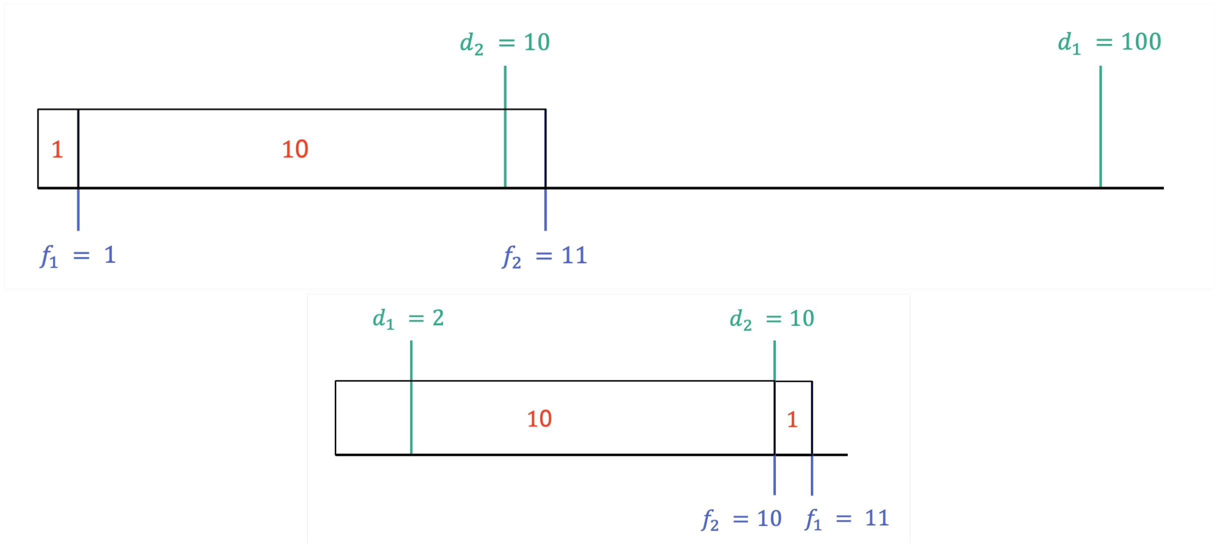

Figure 1: Counterexample depictions.

- slack time $d_i - t_i$

  counterexample: $(t_1 = 1, d_1 = 2), (t_2 = 10, d_2 = 10)$

- earliest deadline

Our algorithm "earliest deadline" will order the tasks from earliest to latest deadline and complete the tasks in this order.

Lemma 1. *There is an optimal schedule with no idle time.*

Algorithm 1 EarliestDeadline(d, t)

```
Renumber the jobs such that $d_1 \leq d_2 \leq \cdots \leq d_n$
Initialize schedule end time $\hat{f} = 0$
for job $j$ from 1 to $n$ do
    Set $i$'s start time $s_i = \hat{f}$ and finish time $f_i = \hat{f} + t_i$
    Update $\hat{f} = \hat{f} + t_i$
end for
return arrays $s, f$
```

Proof of correctness. By Greedy Exchange.

Step 1: Label your algorithm's solution ($A = \{a_1, a_2, \cdots, a_n\}$) and a general solution ($O = \{o_1, o_2, \cdots, o_n\}$).

Label the jobs by nondecreasing (weakly increasing) deadline. This is the order in our algorithm. Let $\pi(i)$ be the position of job i in some arbitrary ordering. ($\pi(i) = 3$ means the job is 3rd.)

Step 2: Compare greedy with the other solution. Assume they're not the same and isolate some difference.

Assume there is an *inversion* in the arbitrary solution somewhere. That is, there exists some i, j such that $i < j$ but $\pi(i) > \pi(j)$.

Step 3: Exchange. Swap the elements in O without making the solution worse. Argue that swapping a finite number of times will result in A.

Within the other solution, at some point between the $\pi(j)^{th}$ job and $\pi(i)^{th}$ job, there must be adjacent jobs in π, i' and j' such that such that $\pi(j) \leq \cdots \leq \pi(j') < \pi(i') \leq \cdots \leq \pi(i)$, but $i' < j'$. Exchange these.

The lateness of i' only decreases. The lateness of j' may increase, but it must be less than i''s was before the swap. Thus the swap can only improve the solution. Continue until there are no inversions.

Hence, greedy is just as good as *any* optimal or arbitrary solution.

Then it is optimal to order by deadline with no idle time. □

Runtime: $O(n \log n)$—just sorting.