CS314 - Greedy Scheduling

- HW (written) due in 1 week
Strategy for greedy algorithms:

1. Figure out a greedy strategy.
   Proof:
2. Assume optimal solution is different from the greedy solution.
   * Find the "first" place they differ.
   * Argue that we can exchange the two without making optimal any worse

⇒ there is no "first place" they differ, so greedy is optimal.
Setting:

- A single resource (i.e., a processor)
- Input: \( n \) requests, each with 
  \( D_1, \ldots, D_n \) deadline \( D[i] \) by which 
  time request \( i \) must be 
  completed 
  \( T[1..n] \) – length of time \( T[i] \) 
  which request \( i \) will take.

Goal: Run everything, and minimize how 
late things are.

Here: minimize the largest “lateness”
Lateness: Given a finish time $F_i$, the lateness is defined as:

$$L_i = F_i - D_i$$

**Goal:** Minimize $\max_i L_i$.
Ex: Job 1: \[\text{length 1} \quad \text{deadline} = 2\]

Job 2: \[\text{length 2} \quad \text{deadline} = 4\]

Job 3: \[\text{length 3} \quad \text{deadline} = 6\]

Input: \[
\frac{D}{T} = \frac{1}{1} + \frac{3}{3} + \frac{6}{6} = 6
\]

Schedule: \[
\text{1} \quad \text{2} \quad \text{3}
\]

\[\text{lateness} = 0\]
Idea for how to be greedy?

* earliest deadline first
  maybe?

* shortest job first

* "slack"—take smallest \( D[j] - T[i] \)

\[
\begin{array}{c}
1 & 2 \\
\hline
2 & 14 \\
5 & 16
\end{array}
\]
Earliest deadline first (EDF)
Sort by $D[i]$, & schedule in this order.

(Hard to believe this works - that's why the proof is key)

First: run time?

$O(n \log n)$
Proof of correctness:

First, note we can assume no idle time. Why?

If I reschedule to eliminate idle time, max "lateness" only can decrease.
**Definition:** Two jobs are inverted if job $i$ goes before job $j$ but: $D[i] > D[j]$

(Note: Our schedule has no inversions.)

**Key:** All schedules with no inversions and no idle time have same max lateness.

**Proof:** Only difference between 2 such schedules is jobs with same deadline. Swapping these won't change lateness.
Thm: There is an optimal schedule with no inversions.

pf: Suppose opt has inversions.
Then \( D[a] > D[b] \) but:

OPT: 

Find adjacent inversion: look at c's nbr c. If inversion w/a, done.
So assume not: \( D[c] > D[a] \).
Know c is also inverted with b.
**Goal**: If we swap $i$ and $j$, gets no worse.

**Concern**: did $i$ get worse?

**Swap**:

- $i$'s finish time goes down
- $F[j]$ is $j$'s new finish
What if job i's lateness increased?

After swap, i finishes at $F[j]$ from OPT.

New lateness for i: $F[j] - D[i]$

But j's lateness in OPT was:

(before swap)

$F[j] - D[i] \leq F[j] - D[j] \leq \text{max lateness}$

So swap could not have made maximum lateness worse.
Finally: How many inversions can there be?