CS314 – Greedy scheduling

Note Title 9/19/2013

- HW (written) due in 1 week

Strategy for greedy algorithms:

1. Figure out a greedy strategy.
2. Proof:
   - Assume optimal solution is different from the greedy solution.
   - Find the "first" place the 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:

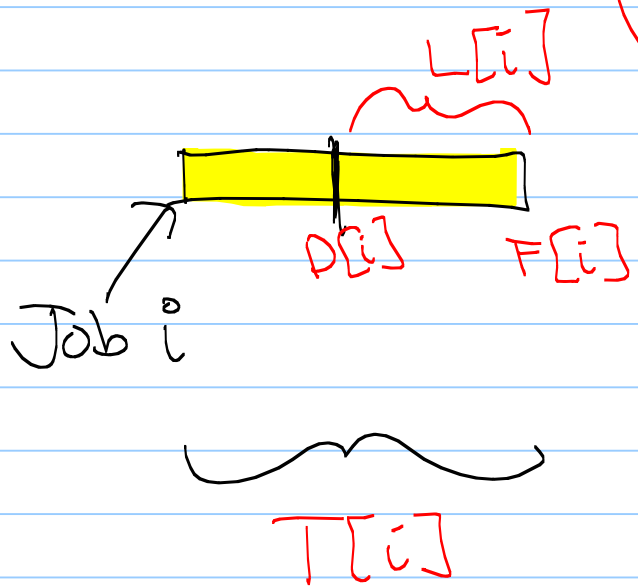
- A single resource (i.e. a processor)
- Input: n requests, each with
  - D[1..n] + deadline D[i] by which time request i must be completed
  - T[1..n] – length of time T[i] which request i will take.

Goal: Run everything, + minimize how late things are.
here – minimize the largest "lateness"

Lateness: Given a finish time $F[i]$,

lateness $L[i] = F[i] - D[i]$

schedule

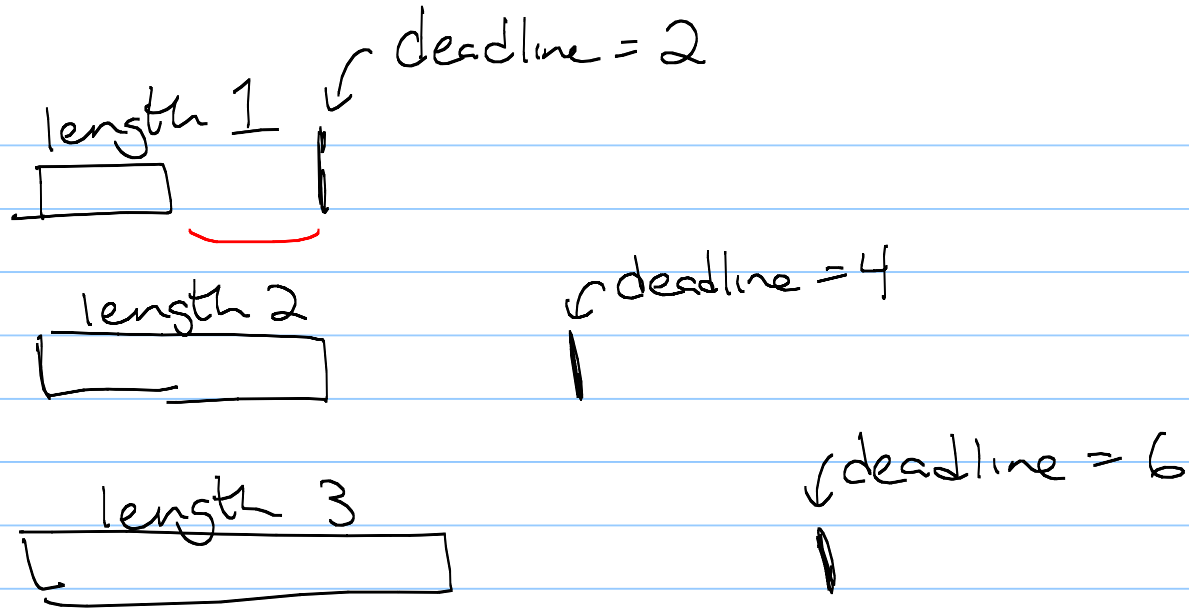


Goal: Minimize $\max_i L[i]$

Ex: Job 1:

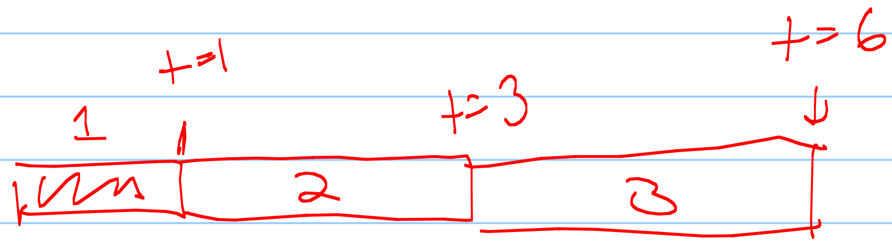
Job 2:

Job 3:



Input: D:
T:

Schedule:



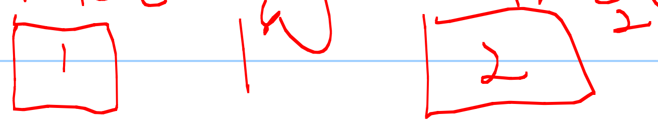
lateness = 0

Ideas for how to be greedy?

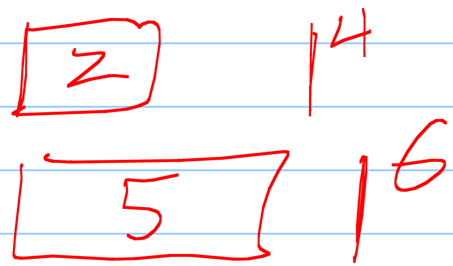
- earliest deadline first

  maybe?

- shortest job first



- "slack" — take smallest $D[i]-T[i]$



Earliest deadline first (EDF)

Sort by D[i], & schedule in this order.

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

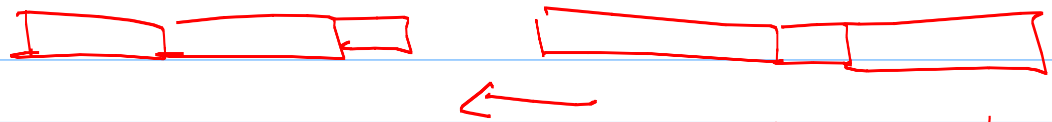
First: run time?

$O(n \log n)$

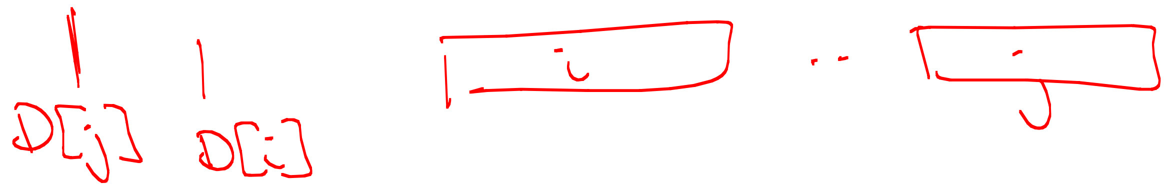
Proof of correctness:

First, note we can assume no idle time.

Why?



If I reschedule to eliminate idle time, max "lateness" only can decrease.

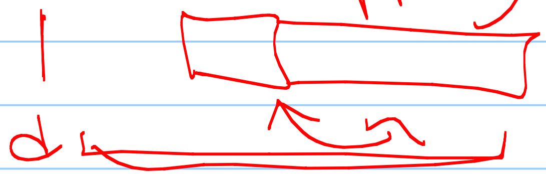


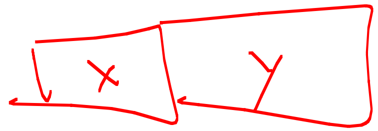
Dfn: Two jobs are *inverted* if job i goes before job j but: $D[i] > D[j]$

(Note: Our schedule has no inversions.)

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

pf: Only difference between 2 such schedules is jobs with same deadline. Swapping these won't change lateness.



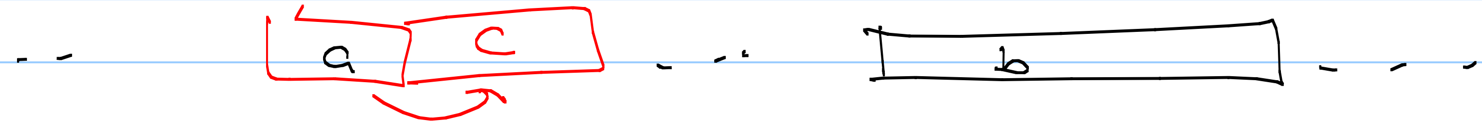


Thm: There is an optimal schedule with no inversions.

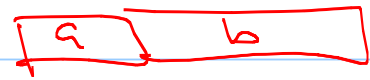
pf: Suppose opt has inversions.

Then $D[a] > D[b]$ but:

OPT: . . .



Find adjacent inversion:



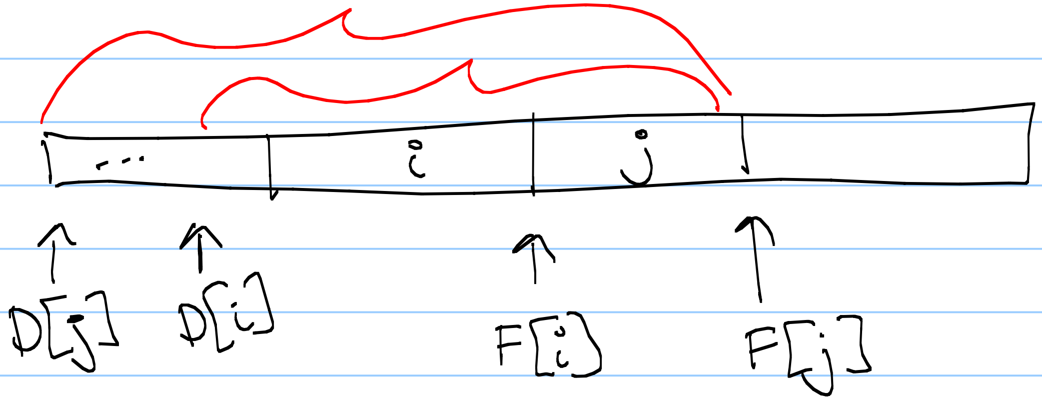
look at a's nbr, c. If inversion w/ a, done.

So assume not: $D[c] \geq D[a]$.

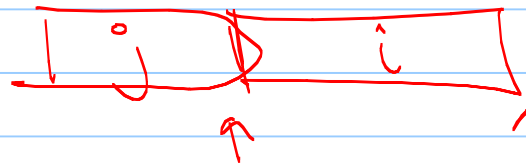
Know c is also inverted with b.

Goal: If we swap i & j, gets no worse.

Concern: did i get worse?



Swap:



j's finish time goes down

F[j] is i'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)

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

So swap could not have made maximum lateness worse.

Finally: How many inversions can there be?

$\binom{n}{2}$