# Greedy Algorithms

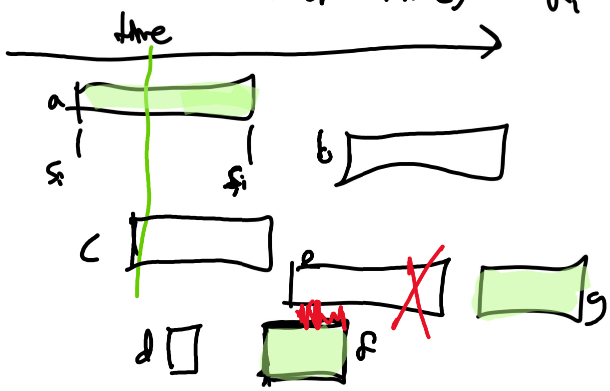
Thursday, November 5, 2020 1:45 PM

Simple algorithms....

tricky proofs.

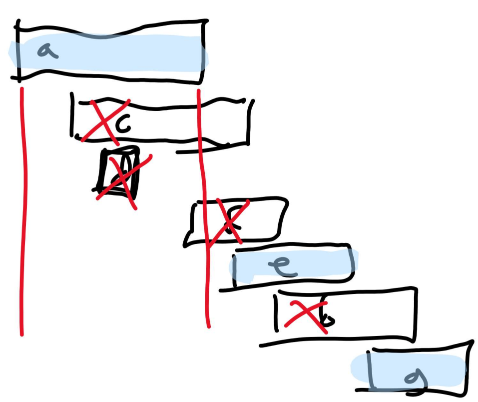
## Problem: Interval Scheduling

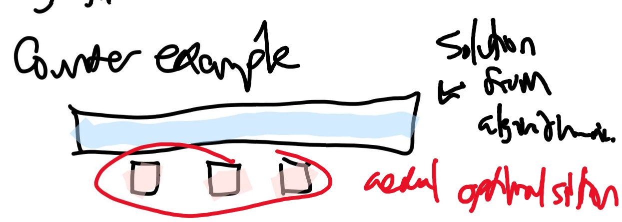
- Input: set of tasks w/ start times $s_i$ and finish times $f_i$




- Goal: find a maximum size subset of tasks that have no overlap

- Greedy alg approach.
  - take the earliest starting time first.



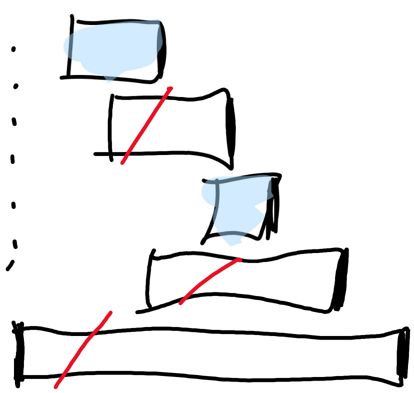




  - earliest finish time first
  - shortest interval first

- longest interval first.

Counterexample

# Earliest Finish Time first works.



## High level algorithm

- sort input intervals in order by $f_i$
  means $i < j$ then $f_i \le f_j$
- sched := {}
- for each $(s_i, f_i)$ in order,
  if $(s_i, f_i)$ does not overlap any interval in sched:
  add $(s_i, f_i)$ to sched
- return sched.

Prove this correct.

- Only gives valid (feasible) schedules.
  this only adds items to sched, if doing so does not create overlaps.

- Prove this is optimal.

General Framework for optimality proofs.

- let T be the output of my alg.
  - we know how it's created.
  - don't yet know it's optimal

- Let $O$ be an optimal solution to problem
  - we don't know much about $O$ except that it is optimal.

- Exchange argument.

– Find a measure of the difference between $O$ and $T$.

ex. let $r$ be the first place in which $O$ and $T$ differ.

– Construct another solution $O'$ $\quad O \cdots O' \cdots T$
which is more similar to $T$
and which is still optimal.

intervals in schedule

ex. $T = t_1 \; t_2 \; \ldots t_{r-1} \; t_r \; \ldots \; t_m$

$$\underbrace{t_1 \; t_2 \ldots t_{r-1}}_{=} \qquad t_r \neq o_r$$

$O \quad t_1 \; t_2 \ldots\ldots t_{r-1} \; o_r \; \ldots \; o_k$

$O' = \; , \ldots\ldots \; o_{r-1} \; t_r \; , \ldots$

– By induction, this show that $T$ is optimal.

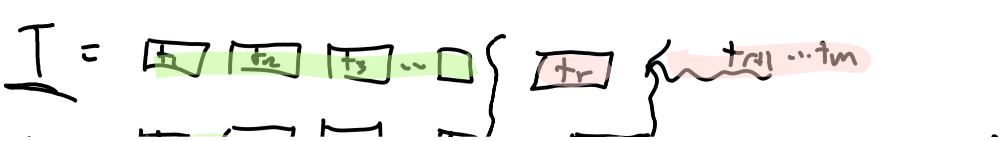
$O_r \ldots O_{r+1} \ldots O_m \qquad O_m = T$

optimal $\qquad$ optimal $\qquad$ optimal

Applying to interval scheduling.

$T = t_1 \ldots \; t_{r-1} \; t_r \; \ldots \; t_m$

$O = t_1 \ldots t_{r-1} \; o_r \ldots o_k \quad (t_r \neq o_r)$



$O = [t_1]\ [\ ]\ [\ ]\ [t_{r-1}]\ [o_r] \dots o_{r+1} \dots o_k$ $\quad k \geq m$

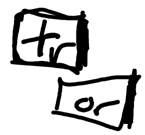
$O' = [t_1]\ [\ ]\ [\ ]\ [t_{r-1}]\ [t_r]\ | \dots o_{r+1} \dots o_k$

need to prove:

1. $O'$ is still feasible.
2. $O'$ is still optimal → direct $|O| = |O'|$

Cases to consider: $f_{t_r} > f_{o_r}$

I: 

II. 

T was constructed by Earliest Finish First.

O is feasible, so $o_r$ does not overlap any $t_1 \dots t_{r-1}$

EFT considered $o_r$ before $t_r$. so $o_r$ would included on

So case I is contradiction.

II: $t_r$ does not overlap any $t_1 \dots t_{r-1}$

Also, $t_r$ does not overlap any $o_{r+1}$ through $o_k$.

because $f_{t_r} \leq f_{o_r}$ and $o_r$ does not overlap any $o_{r+1} \dots o_k$.

Extra case to consider

$T = t_1 \dots t_m$ extra cases.

$O = t_1 \dots t_m\ o_{m+1} \dots o_k$ like case I.

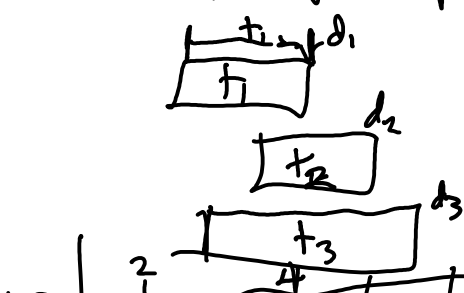
$o_{m+1}$ does not overlap $t_1 \dots t_m$.

So, EFT would have included it.

---

Minimizing max-lateness of jobs.

- Input: set of n jobs, with due time $d_i$ and time to complete $t_i$

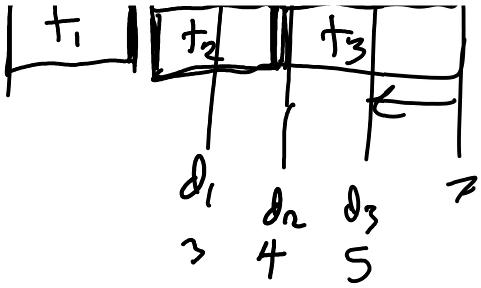


$t_1 = 2$

$t_2 = 2$

$t_3 = 3$

lateness




late by 2

| | |
|---|---|
| 1 | 0 |
| 2 | 0 |
| 3 | 2 |
| max | 2 |

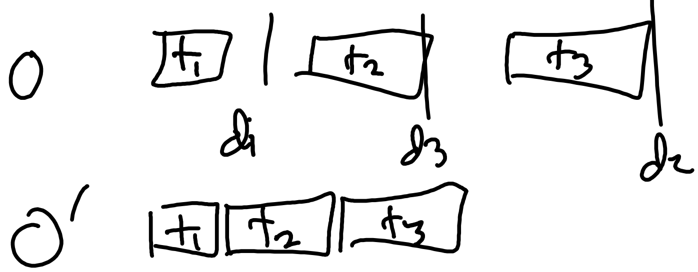
Output: output a job schedule that processes all jobs one one cpu and minimizes the max lateness.

---

Earliest Deadline first.

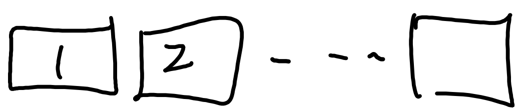
How to prove:

- An optimal solution has no idle time




- How do we make solution O closer to T?

Observations: T has all jobs in order by due time $d_i$




$i < j$ in T, $d_i \le d_j$

Metric: # of "Inversions" in a schedule.

An "inversion" is a pair $i<j$ in schedule, but $d_i > d_j$

- Suppose O is an optimal schedule.

Let r be # of inversions in O.

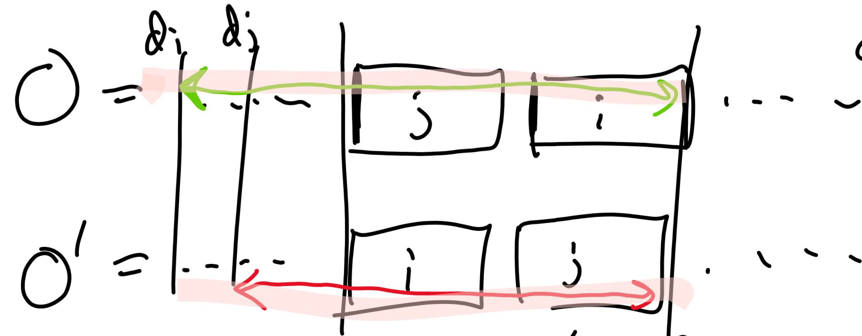
Construct O' that has r-1 (or fewer) inversions

and is still optimal

- T has has no inversions.

IF O has any inversions, then it must have some inversion of adjacent tasks

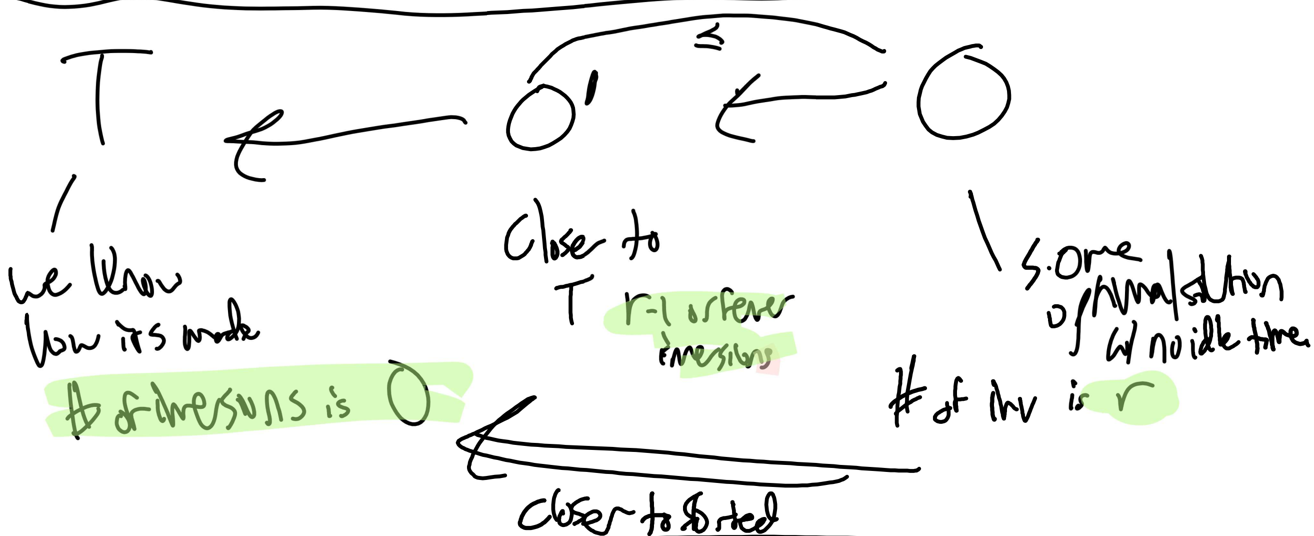
$d_i < d_j$




- O' is still optimal (doesn't increase max lateness.
  - none of jobs except i and j have finish times affected.
  - job i finishes earlier. It's lateness improves.
  - Job j finishes later in O'. It may be more late in O'. Lateness of j in O' is still better than the lateness of i in O

  So, O' max lateness is no worse than O.

---




---

- Every schedule w/ no inversions has same max lateness

  ...

- Simplifying is: all due times are unique.