HW9 due tonight

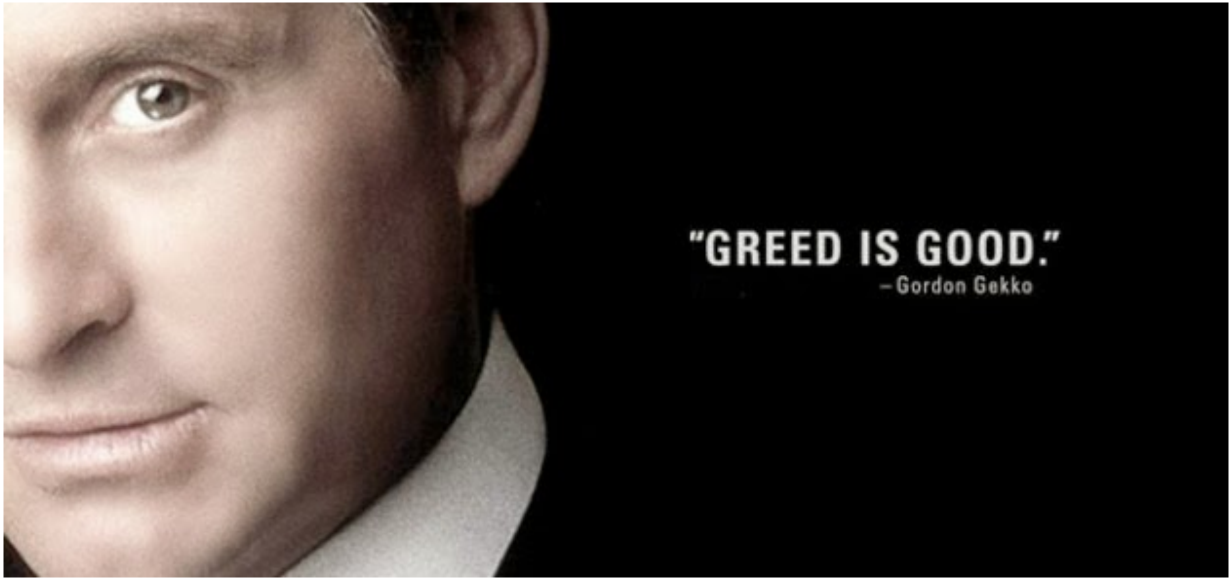
HW10 will be due Apr 20th (last HW)

CS4800: Algorithms & Data
Jonathan Ullman

Lecture 21:
- Greedy Algorithms: Scheduling Problems

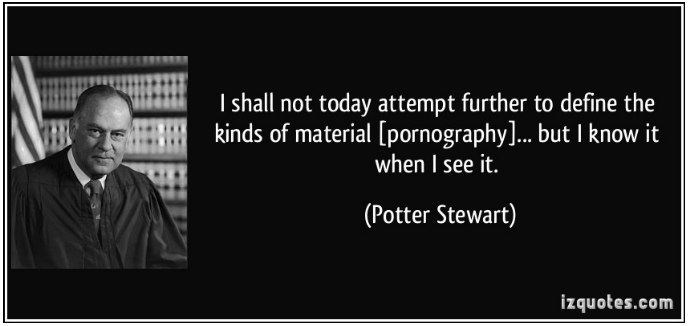
Apr 6, 2018

Obligatory *Wall Street* Quotation



The movie *Wall Street*, however, is not.

Greedy Algorithms



- What's a greedy algorithm?

- Roughly: an algorithm that builds a solution myopically and never looks back
  - Compare to dynamic programming
- Typically: make a single pass over the input
  - Example: Kruskal's MST algorithm

Greedy Algorithms

- Why do care about greedy algorithms?
  - Greedy algorithms are the fastest and simplest algorithms imaginable, and sometimes they work!
  - Simplicity makes them easy to adapt to different models
  - Sometimes useful heuristics when they don't

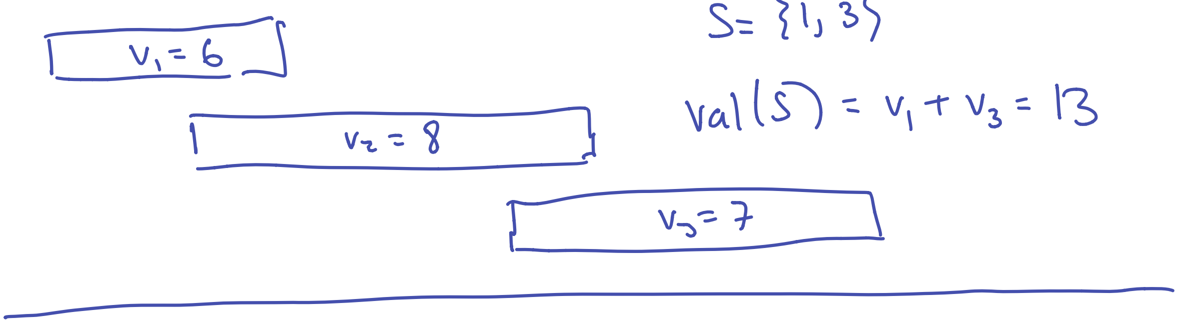
Interval Scheduling

(Weighted) Interval Scheduling

- Input: n intervals (s_i, f_i) with values v_i
- Output: a compatible schedule S with the largest possible total value
    - A schedule is a subset of intervals $S \subseteq \{1, \dots, n\}$
    - A schedule S is compatible if no two $i, j \in S$ overlap
    - The total value of S is $\sum_{i \in S} v_i$

$S = \{1, 3\}$

$\mathrm{val}(S) = v_1 + v_3 = 13$

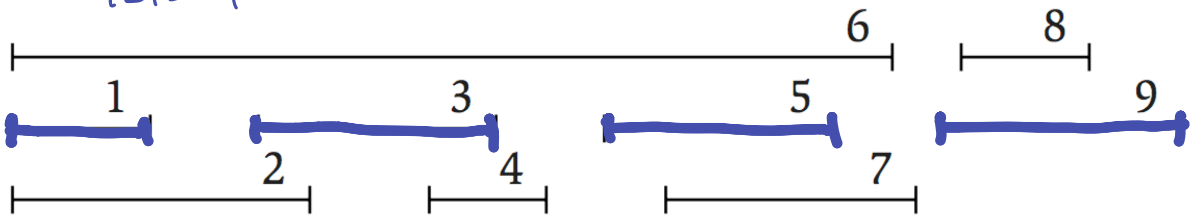


(Unweighted) Interval Scheduling

- Input: n intervals (s_i, f_i)
- Output: a compatible schedule S with the largest possible size
  - A schedule is a subset of intervals $S \subseteq \{1, \ldots, n\}$
  - A schedule S is compatible if no two $i, j \in S$ overlap

$S = \{1, 3, 5, 9\}$

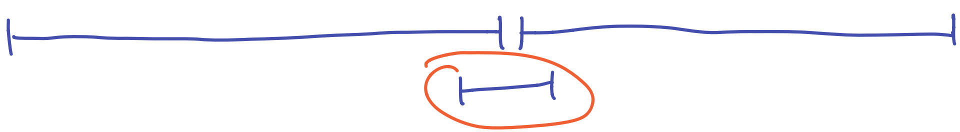
$|S| = 4$



Possibly Greedy Rules

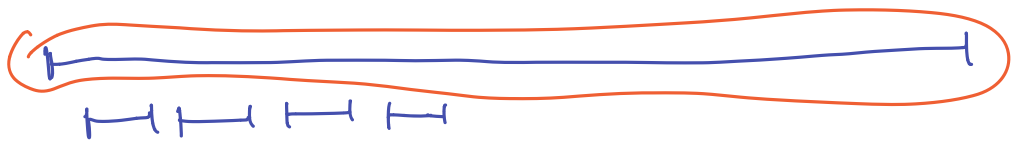
- Choose the shortest interval first

Fail



- Choose the interval with earliest start first

Fail



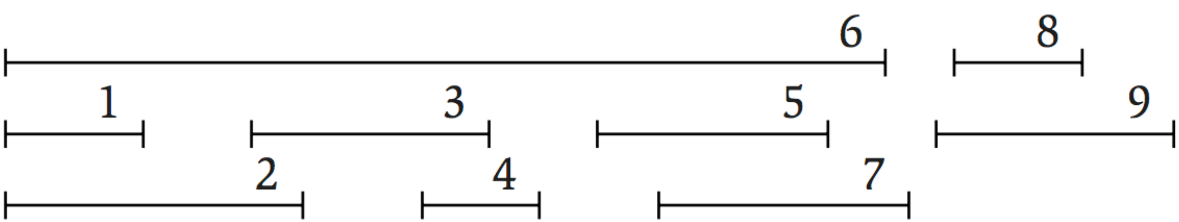
- Choose the interval with earliest finish first

Succeeds

We'll prove this from first principles

Greedy Algorithm: Earliest Finish First

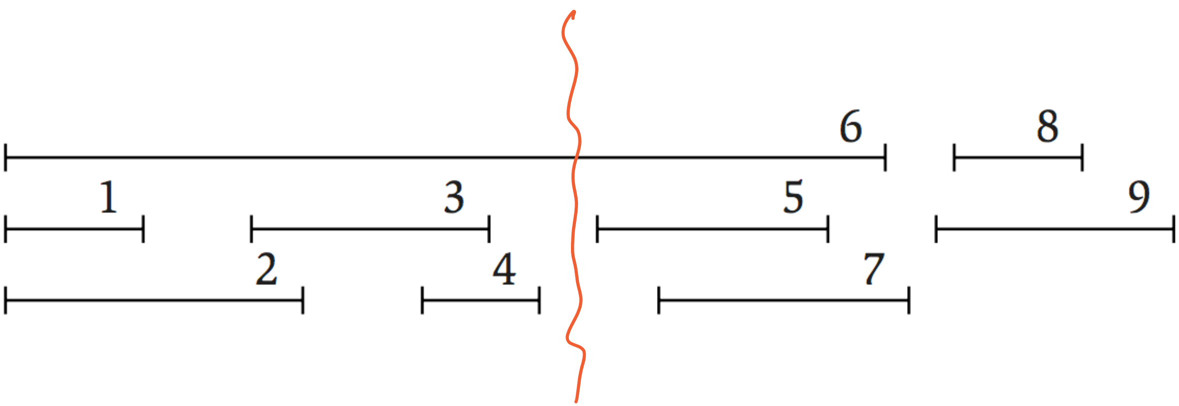
- Sort intervals so that $f_1 \leq f_2 \leq \cdots \leq f_n$ $O(n \log n)$ time
- Let S be empty   end $\leftarrow 0$ $O(1)$
- For $i = 1, \ldots, n$:
  - If interval i doesn't create a conflict, add i to S $n \times O(1)$
    (is $s_i \geq$ end?)   end $\leftarrow f_i$ $= O(n)$
- Return S



Total time is $O(n \log n)$

Greedy Stays Ahead (Proof by Induction)

- How do we know we found an optimal sched.
- "Greedy Stays Ahead" strategy
  - We'll show that at every point in time, the greedy schedule does better than any other schedule



Greedy Stays Ahead

- Let $G = \{i_1, \dots, i_r\}$ be greedy's schedule
- Let $O = \{j_1, \dots, j_s\}$ be some optimal schedule
- Main Claim: for every $t = 1, \dots, r$, $f_{i_t} \leq f_{j_t}$

(My interval t finishes before your interval t.)

Greedy Stays Ahead

- Let $G = \{i_1, \ldots, i_r\}$ be greedy's schedule
- Let $O = \{j_1, \ldots, j_s\}$ be some optimal schedule
- Main Claim: for every $t = 1, \ldots, r$, $f_{i_t} \leq f_{j_t}$

Base Case $(t=1)$:

By construction, f_{i_1} is the smallest finish time

$\Longrightarrow \quad f_{i_1} \leq f_{j_1}$

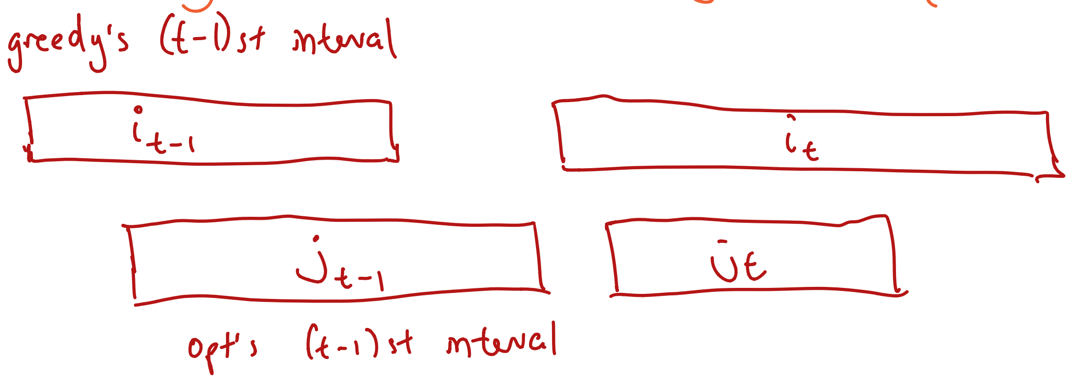
Greedy Stays Ahead

- Let $G = \{i_1, \dots, i_r\}$ be greedy's schedule
- Let $O = \{j_1, \dots, j_s\}$ be some optimal schedule
- Main Claim: for every $t = 1, \dots, r$, $f_{i_t} \le f_{j_t}$

Inductive Step: Assume $f_{i_{t-1}} \le f_{j_{t-1}}$

Assume for contradiction that $f_{i_t} > f_{j_t}$

$\Rightarrow$ greedy would have chosen j_t over i_t



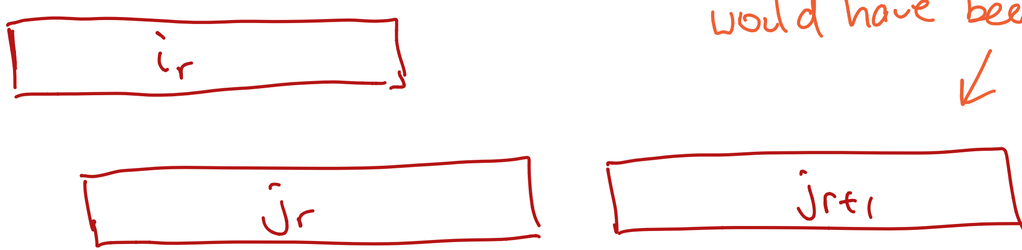
Greedy Stays Ahead

- Let $G = \{i_1, \ldots, i_r\}$ be greedy's schedule
- Let $O = \{j_1, \ldots, j_s\}$ be some optimal schedule
- Finishing the Proof. Claim: $r \geq s$

Assume for the sake of contradiction that $s > r$.

But then greedy stopped early for no reason.

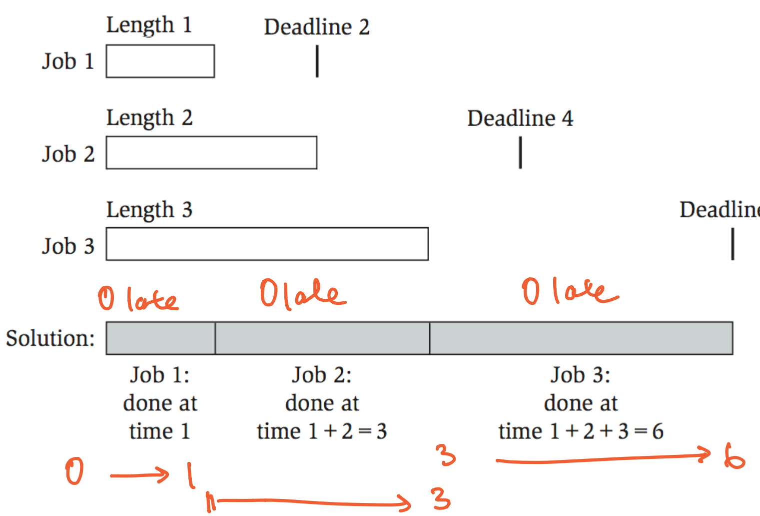
would have been in greedy



Minimum Lateness Scheduling

Minimum Lateness Scheduling

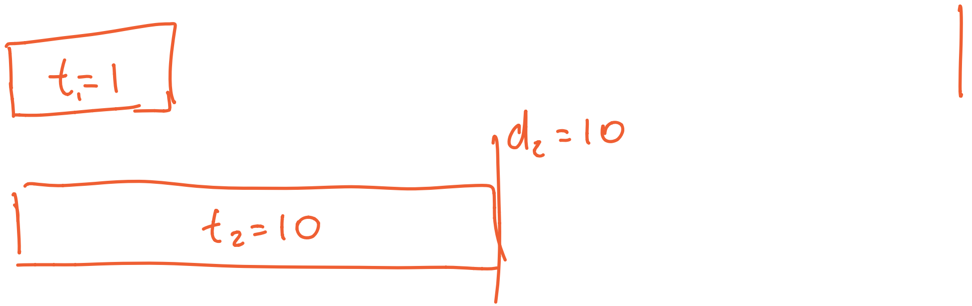
- Input: n jobs with length t_i and deadline d_i
- Output: a minimum-lateness schedule for the jobs
  - Can only do one job at a time, no overlap
  - The lateness of job i is $\max\{f_i - d_i, 0\}$
  - The lateness of a schedule is $\max_i\{\max\{f_i - d_i, 0\}\}$



Possible Greedy Rules (Ask the Audience)

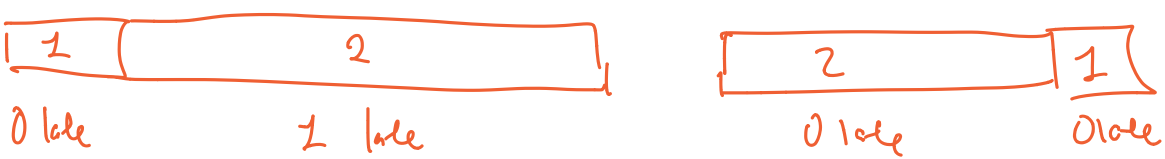
- Choose the shortest job first ($\min t_i$)?

$d_1 = 100$



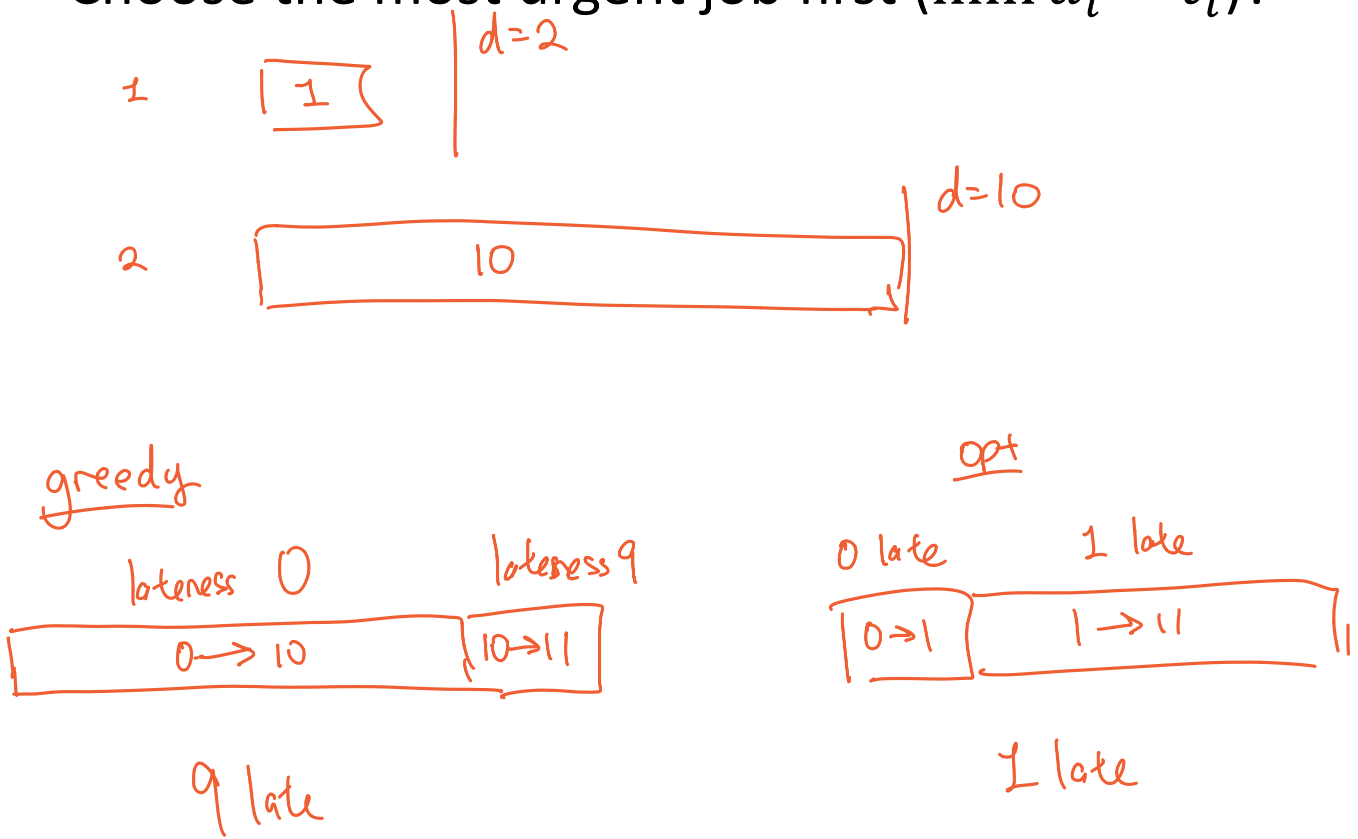
greedy is

opt is



Possible Greedy Rules (Ask the Audience)

- Choose the most urgent job first ($\min d_i - t_i$)?



Greedy Algorithm: Earliest Deadline First

- Sort jobs so that $d_1 \leq d_2 \leq \cdots \leq d_n$
- For $i = 1, \ldots, n$:
  - Schedule job i right after job $i - 1$ finishes

- $O(n \log n)$ time algorithm
- We can easily give start and finish times so that our schedule has no overlaps

Exchange Argument

- G = greedy schedule, O = optimal schedule

- Exchange Argument:
  - We can transform O to G by exchanging pairs of jobs
  - Each exchange only reduces the lateness of O

$O \rightsquigarrow O' \rightsquigarrow O'' \rightsquigarrow O''' \rightsquigarrow \ldots G$

- lateness never increases along the chain
- therefore lateness$(G) \leq$ lateness(O)

Exchange Argument

- G = greedy schedule, O = optimal schedule
- Observation: the optimal schedule has no gaps
  - A schedule is just an ordering of the jobs, with jobs scheduled back-to-back

Exchange Argument

- G = greedy schedule, O = optimal schedule

- We say that two jobs i, j are inverted in O if $d_i < d_j$ but j comes before i
  - Simplifying Assumption: all deadlines are unique
  - Observation: greedy has no inversions

there is a unique sched
w/ no inversions and it's the
greedy sched.

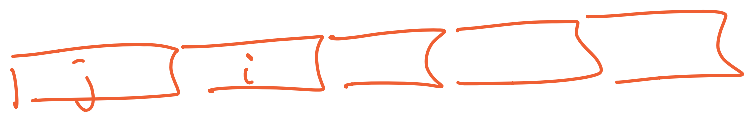
Exchange Argument

If O is not greedy then I can improve O by eliminating an inversion

- We say that two jobs i, j are inverted in O if $d_i < d_j$ but j comes before i
- Claim: the optimal schedule has no inversions
  - Step 1: suppose O has an inversion, then it has an inversion i, j where i, j are consecutive

no

yes



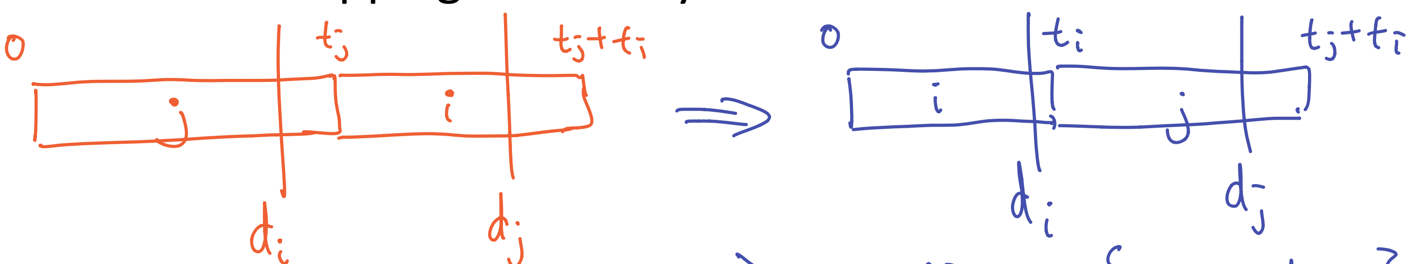
if i,j are inverted, some $k, k+1$ in the middle

of them must be inverted

Exchange Argument

if we flip i, j then we reduce the lateness, therefore O was not optimal.

- We say that two jobs i, j are inverted in O if $d_i < d_j$ but j comes before i
- Claim: the optimal schedule has no inversions
    - Step 1: suppose O has an inversion, then it has an inversion i, j where i, j are consecutive
    - Step 2: if i, j are a consecutive jobs that are inverted then flipping them only reduces the lateness



$\text{late}(j) = \max\{t_j - d_j, 0\}$ $>$ $\text{late}(j) = \max\{t_j + t_j - d_j, 0\}$

$\boxed{\text{late}(i) = \max\{t_j + t_i - d_i, 0\}}$ $>$ $\text{late}(i) = \max\{t_i - d_i, 0\}$

Exchange Argument

- If i, j are a consecutive jobs that are inverted then flipping them only reduces the lateness

- Choose some ordering for greedy.
- Argue that if a solution doesn't respect that ordering then there are two points consecutively that dont.
- Argue that flipping those points only helps.

Part that depends on your problem

Exchange Argument

- We say that two jobs i, j are inverted in O if $d_i < d_j$ but j comes before i
- Claim: the optimal schedule has no inversions
  - Step 1: suppose O has an inversion, then it has an inversion i, j where i, j are consecutive
  - Step 2: if i, j are a consecutive jobs that are inverted then flipping them only reduces the lateness

- G is the unique schedule with no inversions, O is the unique schedule with no inversions, $G = O$

Classroom Assignment

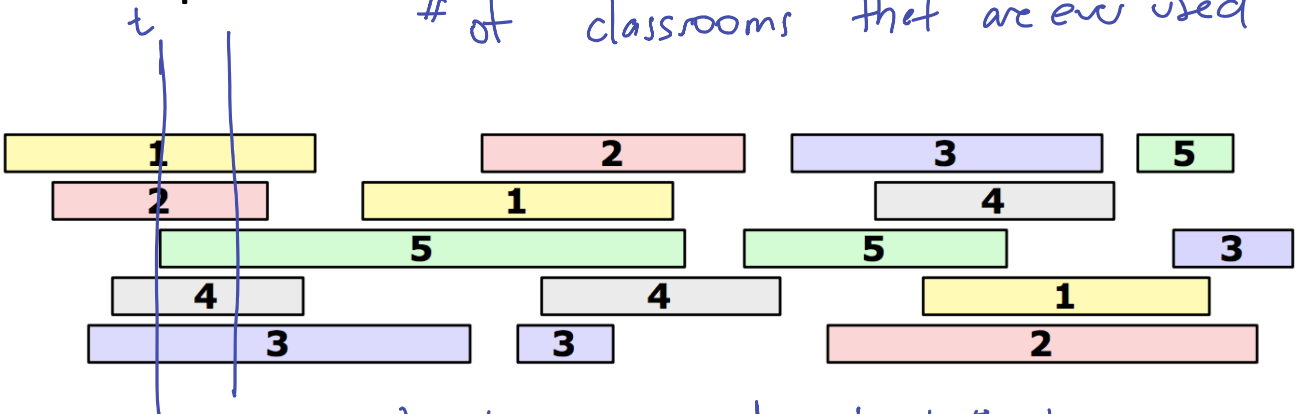
Classroom Assignment

- Input: n classes (s_i, f_i)
- Output: an assignment of intervals to classrooms using the smallest number of classrooms
  - classrooms can hold any number of classes
  - but no two classes can share a classroom

Classroom Assignment

- Example

score of an assigment is the total # of classrooms that are ever used



at this time we need at least 5 classrooms

- Is this an optimal packing of these classes?

The only type of obstruction is that there is a time t w/ k classes in session

Greedy: First Available Classroom

- Sort classes by start time so $s_1 \leq s_2 \leq \cdots \leq s_n$
- For $i = 1, \ldots, n$
  - Let c be the smallest # classroom available at time s_i
  - Assign class i to room c

Duality

- Let G be the greedy assignment
- Claim: if G uses k classrooms, then no assignment can use fewer than k classrooms
  - Let t be the time when we first used classroom k
  - There must be at least k classes that are in session at time t (so all assignments use $\geq k$)

Suppose the first use of room k was class i

There must have been k classes s.t. $s_i \in [s, f]$

- Clearly class i is one of them
- Since we didnt use room 1, ..., k-1, those rooms must have been assigned classes j s.t. $s_i \in [s_j, f_j]$