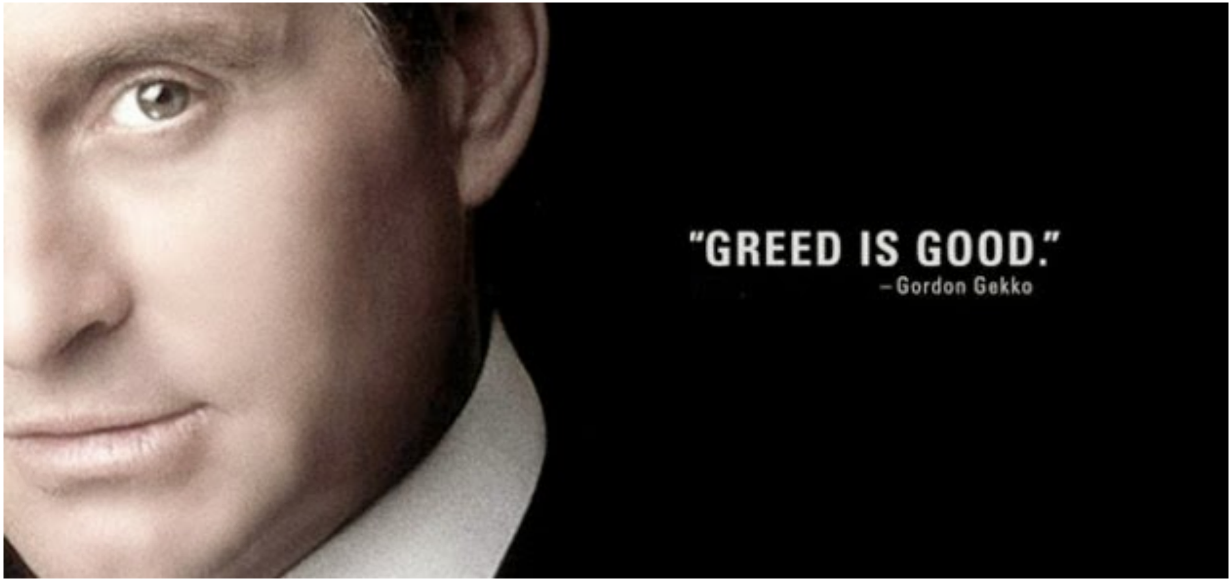
CS3000: Algorithms & Data
Jonathan Ullman

Lecture 18:

- Greedy Algorithms: Proof Techniques

March 30, 2020

Obligatory *Wall Street* Quotation



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

Greedy Algorithms

- What's a greedy algorithm?
  - I know it when I see it
  - Roughly, an algorithm that builds a solution myopically and never looks back (compare to DP)
  - Typically, make a single pass over the input (e.g. Kruskal)

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

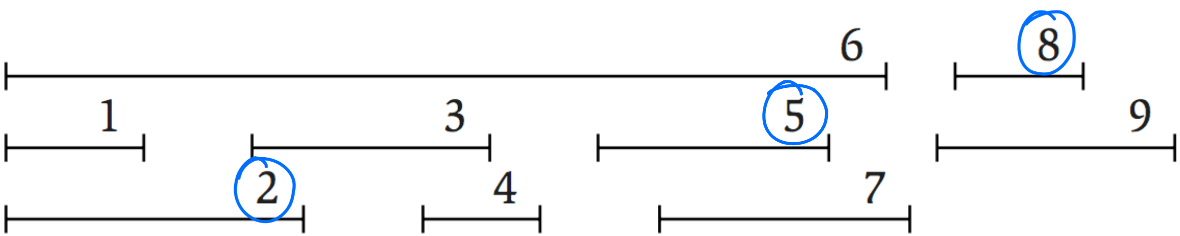
Interval Scheduling

(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 $S \subseteq \{1, \ldots, n\}$
  - A schedule S is compatible if no two $i, j \in S$ overlap
  - The total value of S is $\sum_{i \in S} v_i$

(Unweighted) Interval Scheduling

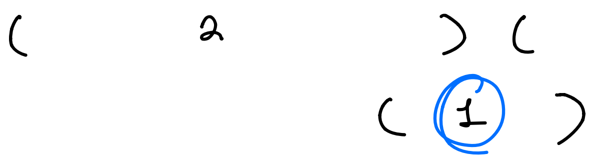
- Input: n intervals (s_i, f_i)
- Output: a compatible schedule S with the largest possible size
  - A schedule is a subset of intervals $S \subseteq \{1, \dots, n\}$
  - A schedule S is compatible if no two $i, j \in S$ overlap



A compatible subset of size 3

Possibly Greedy Rules

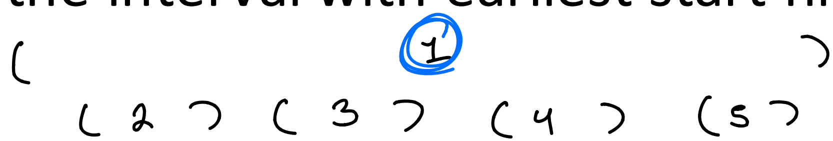
- Choose the shortest interval first



OPT = 2

GREEDY = 1

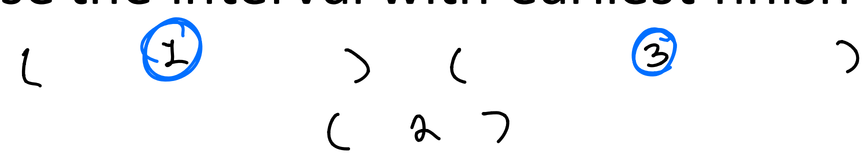
- Choose the interval with earliest start first



OPT = 4

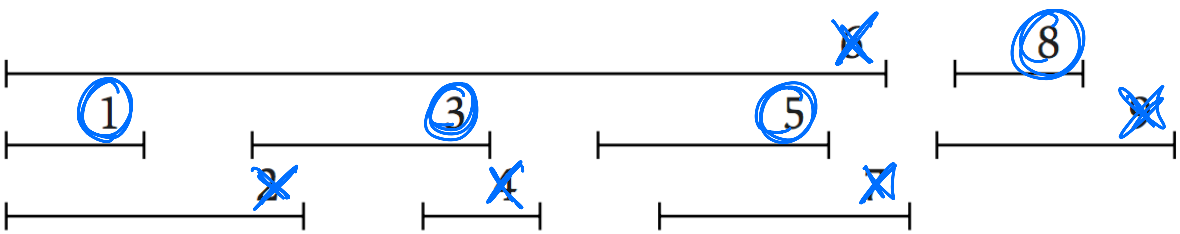
GREEDY = 1

- Choose the interval with earliest finish first



Greedy Algorithm: Earliest Finish First

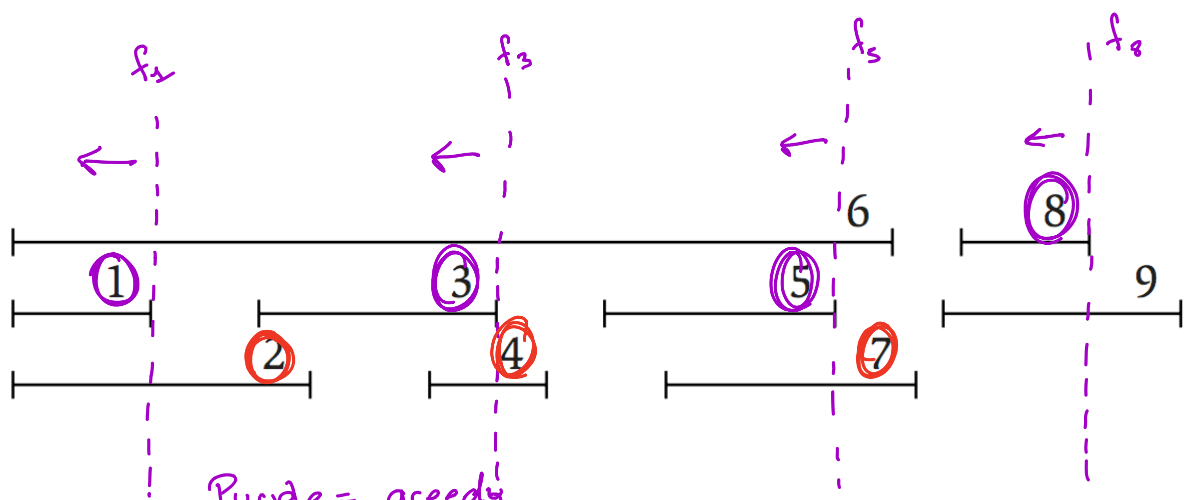
- Sort intervals so that $f_1 \leq f_2 \leq \cdots \leq f_n$
- Let S be empty
- For $i = 1, \ldots, n$:
  - If interval i doesn't create a conflict, add i to S
- Return S



Greedy Stays Ahead

Proof by Induction

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



Purple = greedy
Red = some other schedule

Greedy Stays Ahead

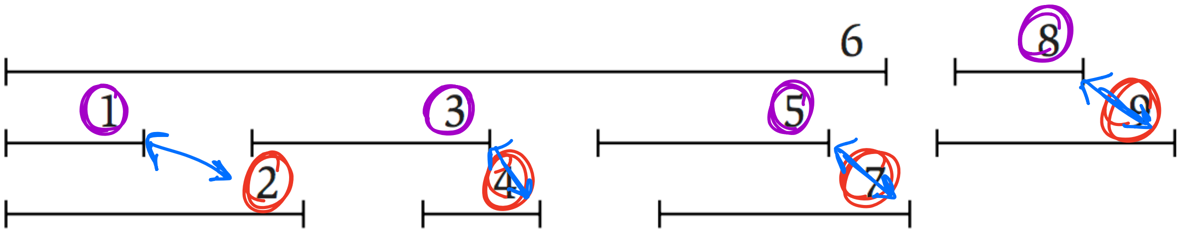
- Let $G = \{i_1, \dots, i_r\}$ be greedy's schedule
- Let $O = \{j_1, \dots, j_s\}$ be some optimal schedule
- **Key Claim:** for every $t = 1, \dots, r$, $f_{i_t} \le f_{j_t}$

e.x. $G = \{1, 3, 5, 8\}$

$i_1 = 1$
$i_2 = 3$
$i_3 = 5$
$i_4 = 8$

$O = \{2, 4, 7, 9\}$

$j_1 = 2$
$j_2 = 4$
$j_3 = 7$
$j_4 = 9$



$f_{i_1} \le f_{j_1}$ $f_{i_2} \le f_{j_2}$ $f_{i_3} \le f_{j_3}$ $f_{i_4} \le f_{j_4}$

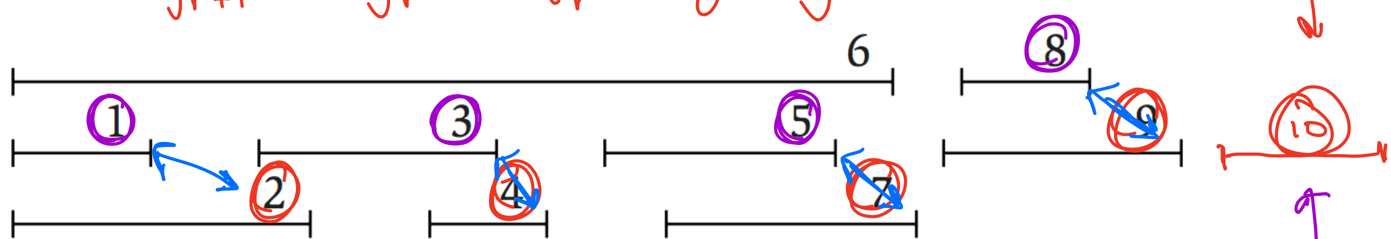
Greedy Stays Ahead

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

Claim $\Rightarrow$ Greedy is optimal

Suppose $s > r$

Then $s_{j_{r+1}} > f_{j_r} > f_{i_r} \Rightarrow$ greedy would also choose j_{r+1}



$f_{i_1} \le f_{j_1}$ $f_{i_2} \le f_{j_2}$ $f_{i_3} \le f_{j_3}$ $f_{i_4} \le f_{j_4}$

Would also be chosen by greedy

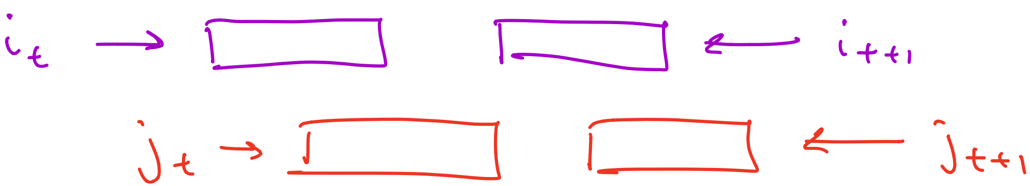
Greedy Stays Ahead

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

Proof by Induction:

- Base Case: $f_{i_1} \leq f_{j_1}$ (Because greedy always chooses the first interval to finish.)
- Inductive Step:

  If $f_{i_t} \leq f_{j_t}$ then $f_{i_{t+1}} \leq f_{j_{t+1}}$



Greedy Stays Ahead

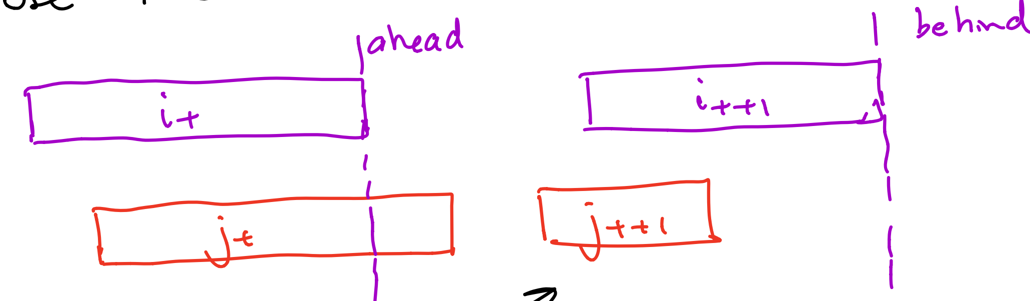
Subtle part is arguing why this picture is impossible.

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

Proof of Inductive Step:

If $f_{i_t} \le f_{j_t}$ then $f_{i_{t+1}} \le f_{j_{t+1}}$

Suppose this were false.



Greedy would have considered j_{t+1} before i_{t+1}, and chosen it.

Minimum Lateness Scheduling

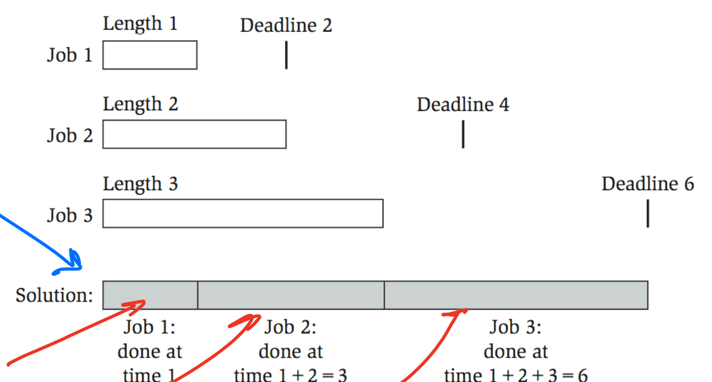
Minimum Lateness Scheduling

- Input: n jobs with length t_i and deadline d_i
  - Simplifying assumption: all deadlines are distinct
- 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 $\max_i\{\max\{f_i - d_i, 0\}\}$

$(s_1, f_1)\ (s_2, f_2)(s_3, f_3)$

no breaks

The schedule should never have gaps



This sched has 0 lateness

job1

job2

job3

Possible Greedy Rules

- Choose the shortest job first (min t_i)?

Job 1: $t_1 = 1$ $d_1 = 20$

Job 2: $t_2 = 10$ $d_2 = 10$

1 2 d_2 d_1

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

Job 1: $t_1 = 1$ $d_1 = 2$

Job 2: $t_2 = 10$ $d_2 = 10$

0 10 11

2 1 lateness = 9

d_1 d_2

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

Job 1: $t_1 = 1$ $d_1 = 20$

Job 2: $t_2 = 10$ $d_2 = 10$

Greedy would do 2 then 1 w/ lateness 0

Job 1: $t_1 = 1$ $d_1 = 2$

Job 2: $t_2 = 10$ $d_2 = 10$

Greedy would choose 1 then 2 w/ lateness 1

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
  - Therefore the lateness of G is at most that of O

$$G \leftarrow \cdots \leftarrow S_3 \leftarrow S_2 \leftarrow S_1 \leftarrow O$$

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

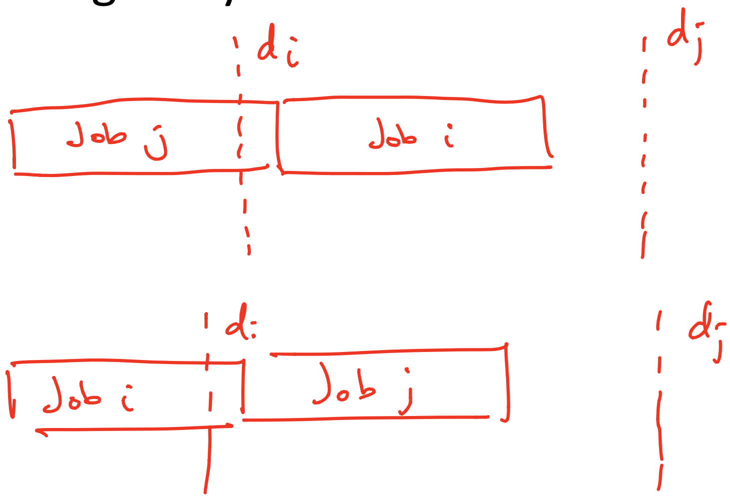
- G = greedy schedule, O = optimal schedule

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

Inversion

Inversion w/ consecutive jobs

Not Inversion



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

- Alternative Form: If a schedule has inversions, then there is a schedule that is at least as good without inversions

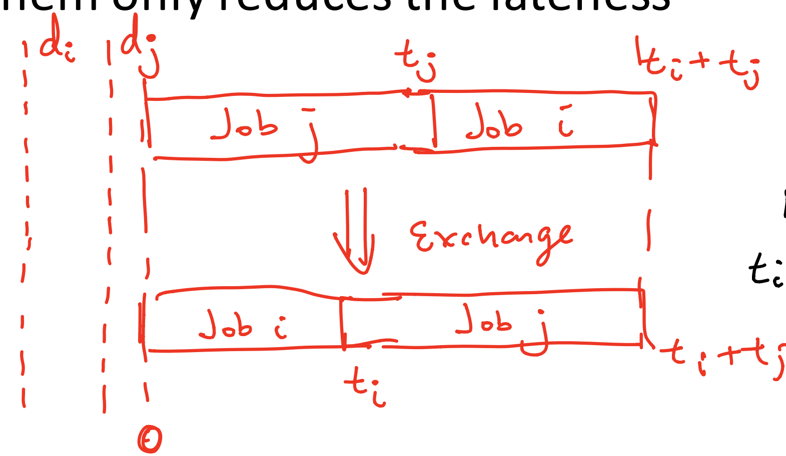
Exchange Argument

- 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

aka "exchanging"

lateness of i: $t_i + t_j - d_i$

lateness of j: $t_i + t_j - d_j$

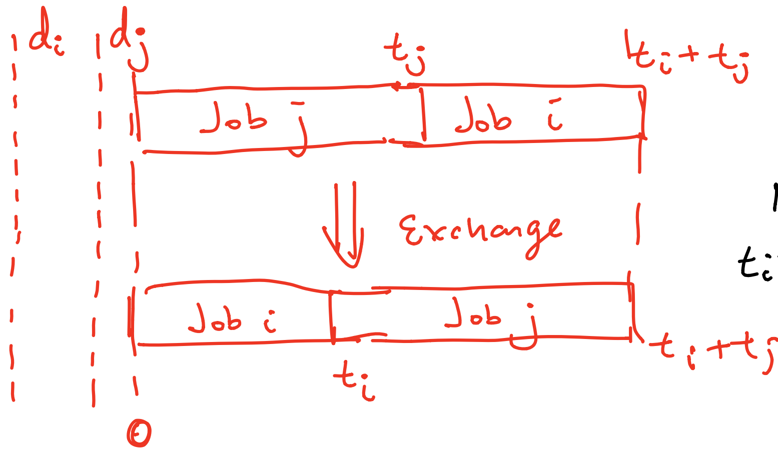


$t_i + t_j - d_i$

$> t_i + t_j - d_j$

Exchange Argument

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



lateness of i:

$t_i + t_j - d_i$

lateness of j:

$t_i + t_j - d_j$

$t_i + t_j - d_i$

$> t_i + t_j - d_j$

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, $G = O$