Piazza poll re: midterm review

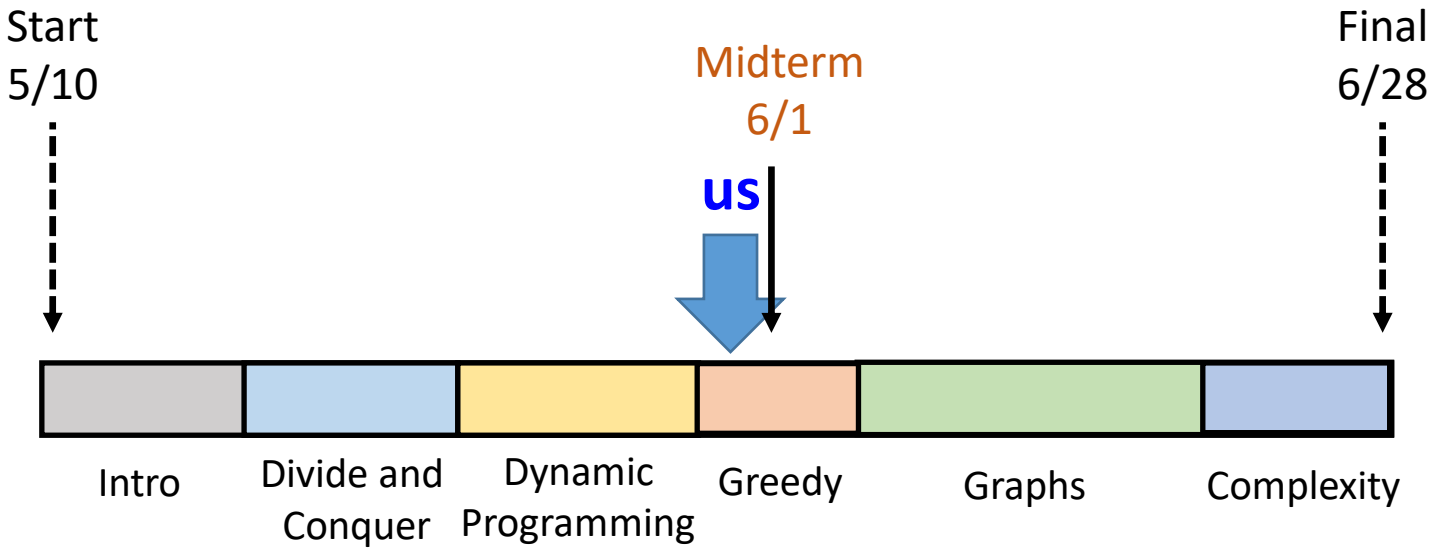
CS3000: Algorithms & Data
Drew van der Poel

Lecture 11

- Greedy algorithms: Proof techniques

May 26, 2021

Outline



Last class: dynamic programming: SLS

Next class: midterm review

Greedy Algorithms

- What's a greedy algorithm?

Greedy Algorithms

- What's a greedy algorithm?
  - Roughly, an algorithm that builds a solution by doing what is currently "best" and never looks back (compare to DP)
  - Typically, make a single pass over the input (recall Knapsack – "bang for your buck") → rule: pick items w/ largest $\frac{v_i}{w_i}$
  - You know it when you see it

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

- Problems: counting students, stable matching, sorting, n-digit mulitiplication, array searching, selection, weighted interval scheduling, segmented least squares, knapsack

- Alg. techniques: divide & conquer, dynamic programming, **greedy**

- Analysis: asymptotic analysis, recursion trees, Master Thm.

- Proof techniques: (strong) induction, contradiction

Interval Scheduling

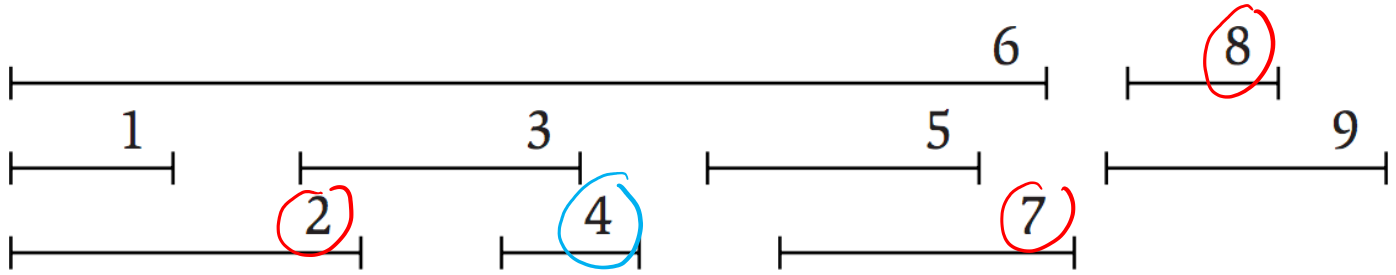
(Weighted) Interval Scheduling

- 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 $\sum_{i \in S} v_i$

DP

(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



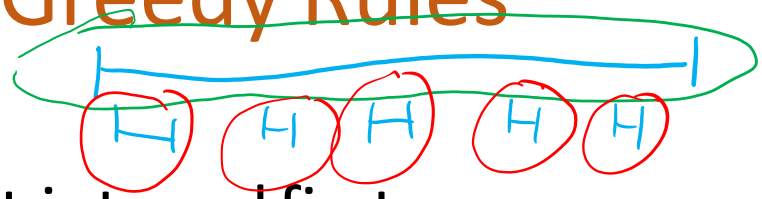
$S = \{2, 7, 8\}$ compatible not optimal

Opt is > 3

Possibly Correct Greedy Rules

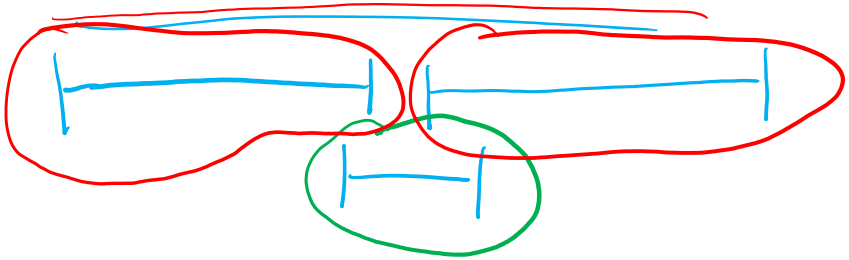
- Select the shortest intervals first
- Select the interval w earliest start
- " " " finish

Possibly Correct Greedy Rules



- Choose the shortest interval first

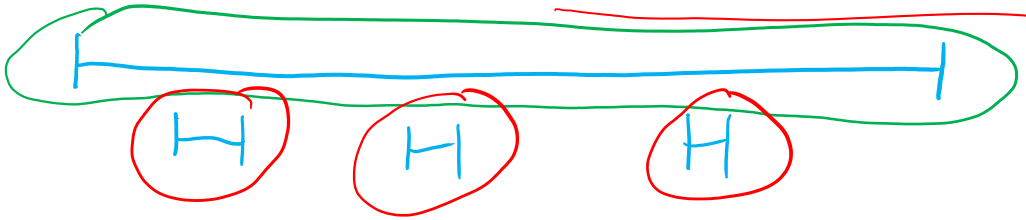
Attempt 1



OPT = 2 >

greedy = 1

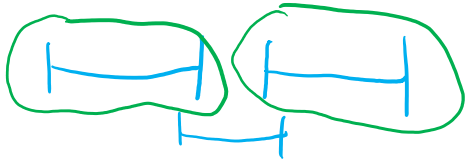
- Choose the interval with earliest start first



OPT = 3 >

greedy = 1

- Choose the interval with earliest finish first



greedy = 2

= OPT!

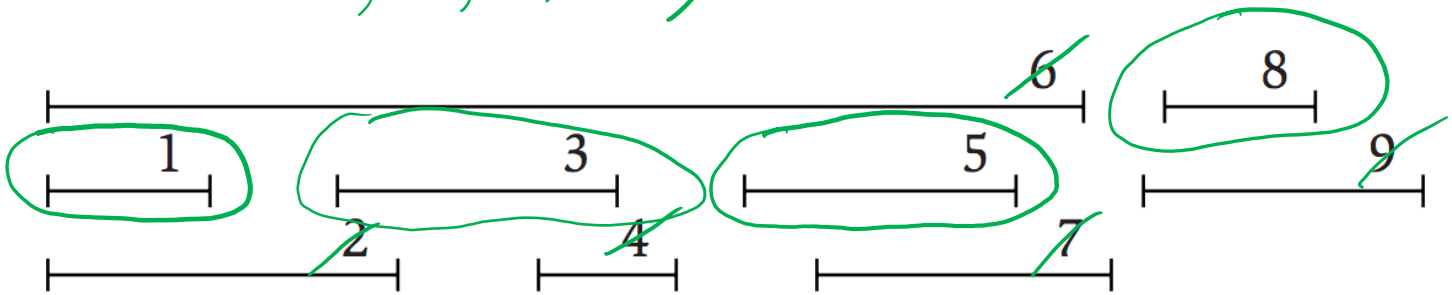
greedy = 3

= OPT!

Greedy Algorithm: Earliest Finish First

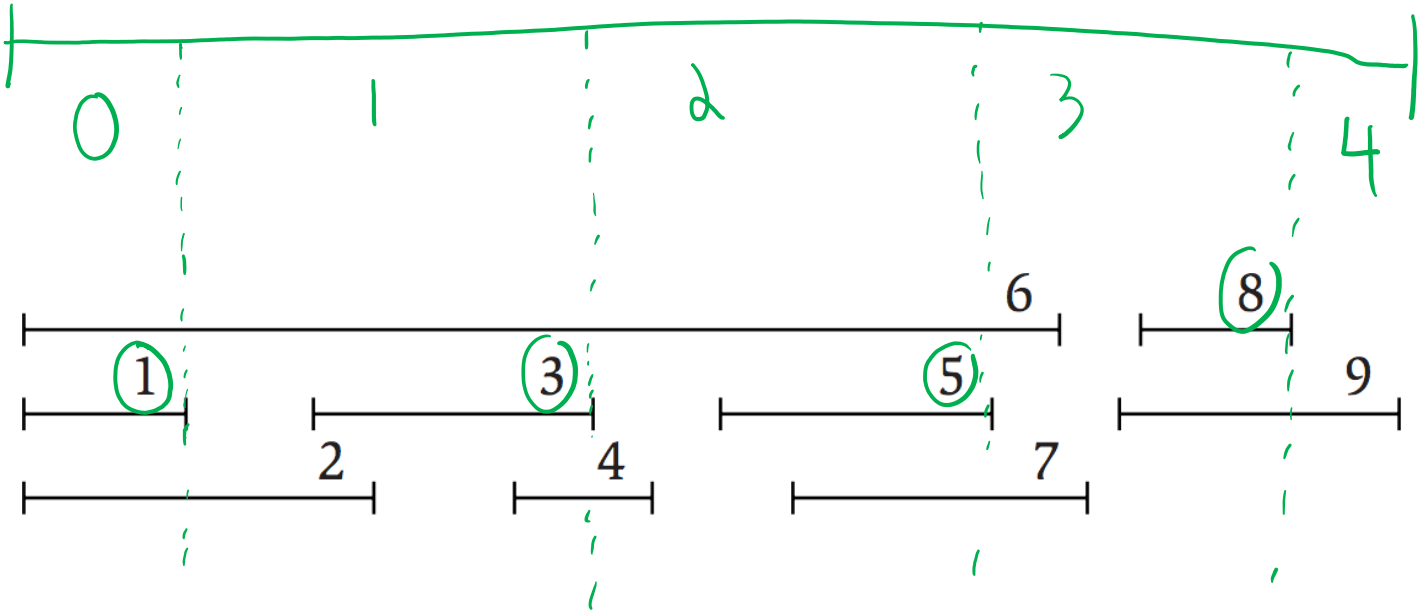
- Sort intervals so that $f_1 \leq f_2 \leq \cdots \leq f_n$
- Let S be empty
- For $i = 1, \ldots, n$:
  - If interval i doesn't create a conflict, add i to S
- Return S

$S = \{1, 3, 5, 8\}$



Greedy Stays Ahead

- 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



Greedy Stays Ahead

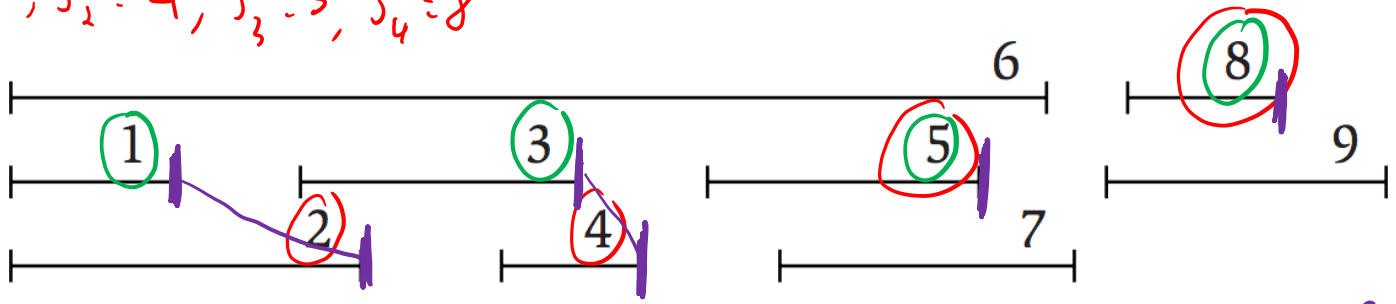
$G = \{1, 3, 5, 8\}$ $O = \{2, 4, 5, 8\}$

$r = 4$ $s = 4$

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

$i_1 = 1, \; i_2 = 3, \; i_3 = 5, \; i_4 = 8$

$j_1 = 2, \; j_2 = 4, \; j_3 = 5, \; j_4 = 8$



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

Greedy Stays Ahead

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

Weak Induction

$H(k)$: $f_{i_k} \leq f_{j_k}$ $\quad 1 \leq k \leq r$

Base: $H(1)$ — greedy alg. chooses the first interval to finish

(direct)

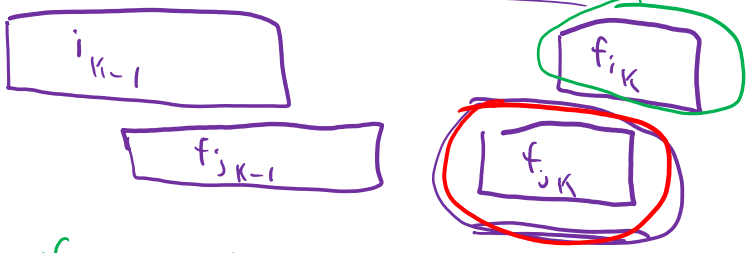
$f_{i_1} \leq f_{j_1}$

Greedy Stays Ahead

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

Ind. $H(k-1) \rightarrow H(k)$

by IH, $f_{i_{k-1}} \leq f_{j_{k-1}}$



Suppose $f_{i_k} > f_{j_k}$

$f_{i_{k-1}} \leq f_{j_{k-1}} < s_{j_k}$ (compatibility)

if $f_{i_k} > f_{j_k}$, greed would have selected j_k instead!

$\therefore$ G is not actually greedy $\rightarrow\leftarrow$ $f_{i_k} \leq f_{j_k}$

Greedy Stays Ahead

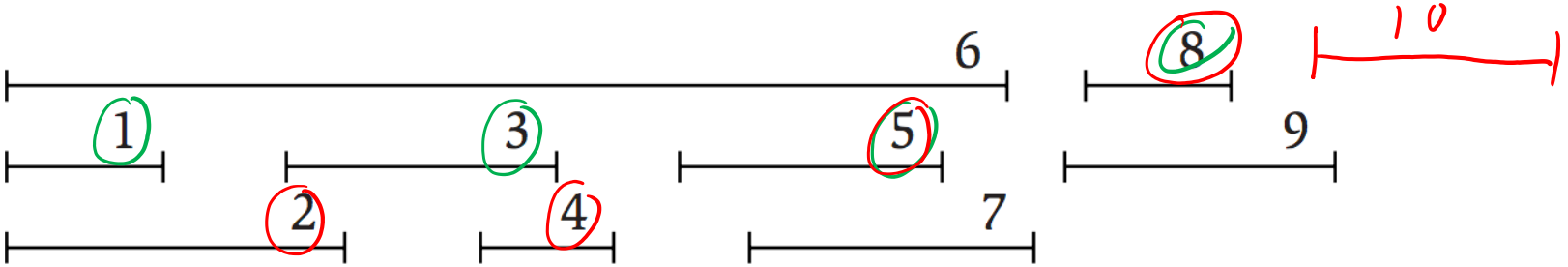
- Let $G = \{i_1, \ldots, i_r\}$ be greedy's schedule
- Let $O = \{j_1, \ldots, j_s\}$ be some other schedule
- **Lemma.** for every $t = 1, \ldots, r$, $f_{i_t} \leq f_{j_t}$ → G is optimal

Assume False, G is not optimal. $r < s$

$s_{j_{r+1}} > f_{j_r} \geq f_{i_r}$ ∴ greedy can also select interval j_{r+1}

↑ compatibility — by Lemma

G is not actually the greedy soln. →



- Problems: counting students, stable matching, sorting, n-digit mulitiplication, array searching, selection, weighted interval scheduling, segmented least squares, knapsack

- Alg. techniques: divide & conquer, dynamic programming, greedy

- Analysis: asymptotic analysis, recursion trees, Master Thm.

- Proof techniques: (strong) induction, contradiction, **greedy stays ahead**

Minimum Lateness Scheduling

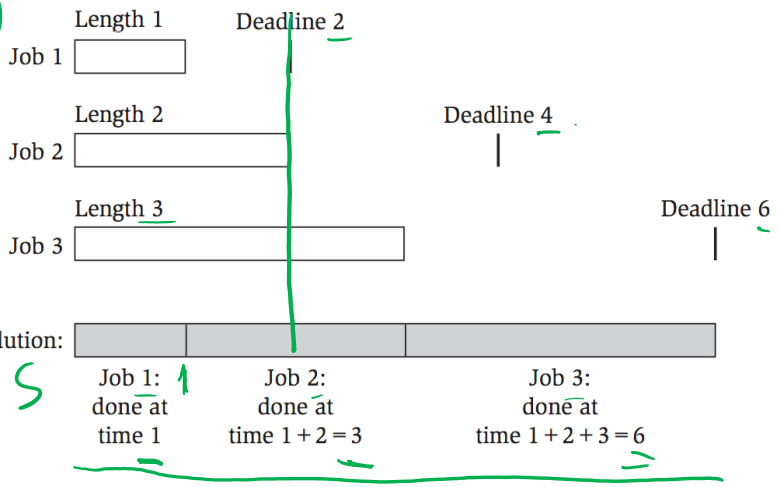
Minimum Lateness Scheduling

- 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 $\max_i\{\max\{f_i - d_i, 0\}\}$

Lateness(1) = max(1-2, 0) = 0

Lateness(2) = max(3-4, 0) = 0

Lateness(3) = max(6-6, 0) = 0



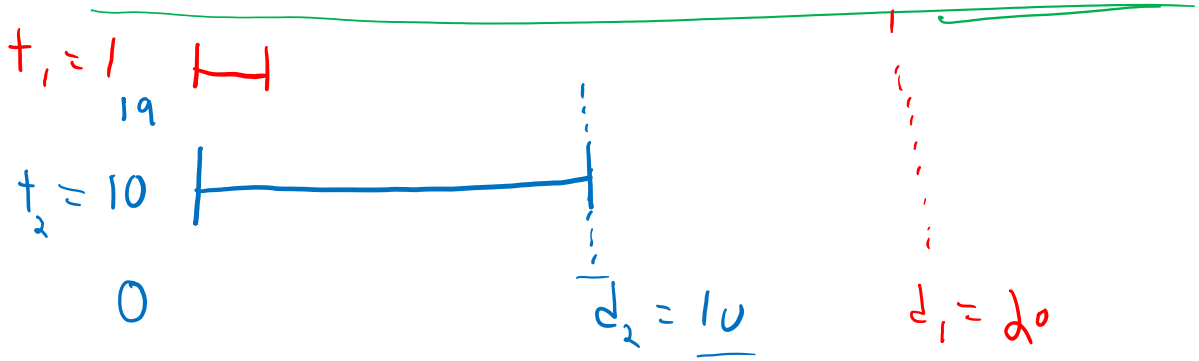
lateness(S) = 0

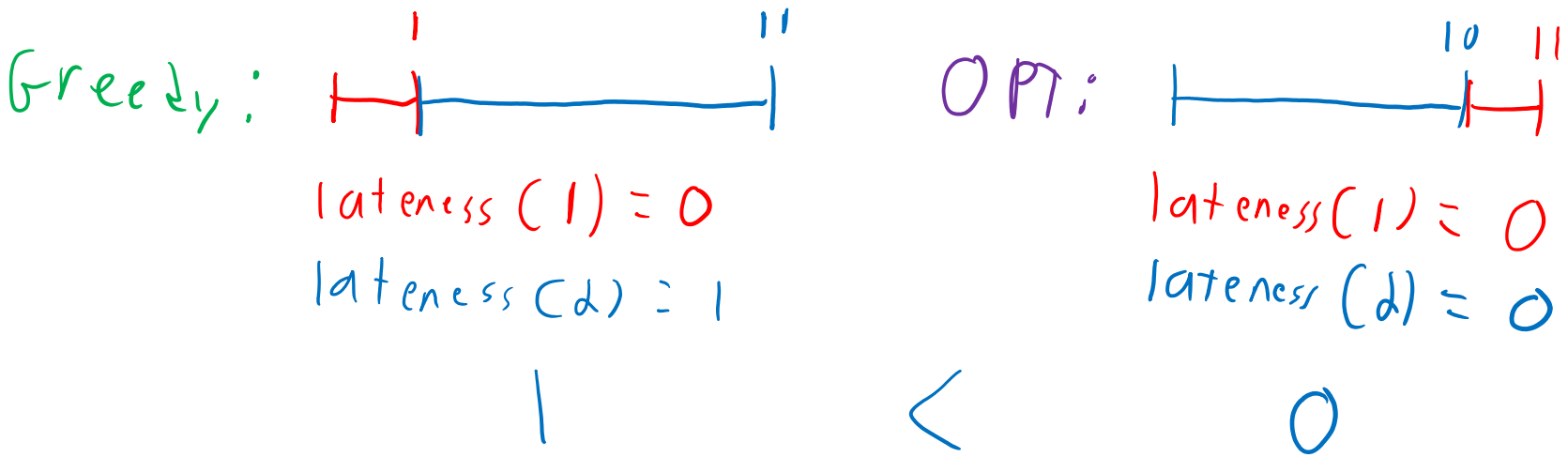
Possible Greedy Rules

- Shortest time
- Shortest d_i/t_i
  - ↳ Shortest $d_i - t_i$
- Earliest Deadline First

Possible Greedy Rules

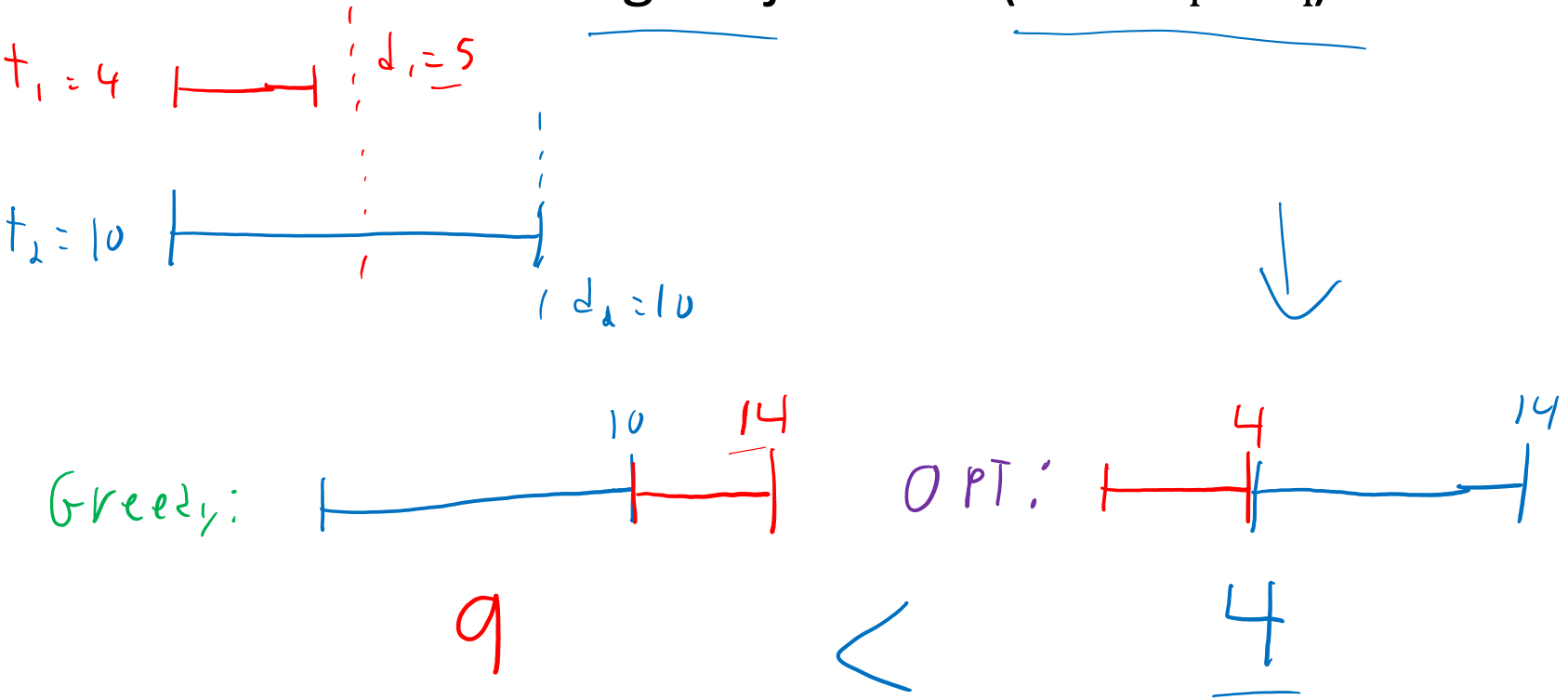
- Choose the shortest job first ($\min t_i$)?





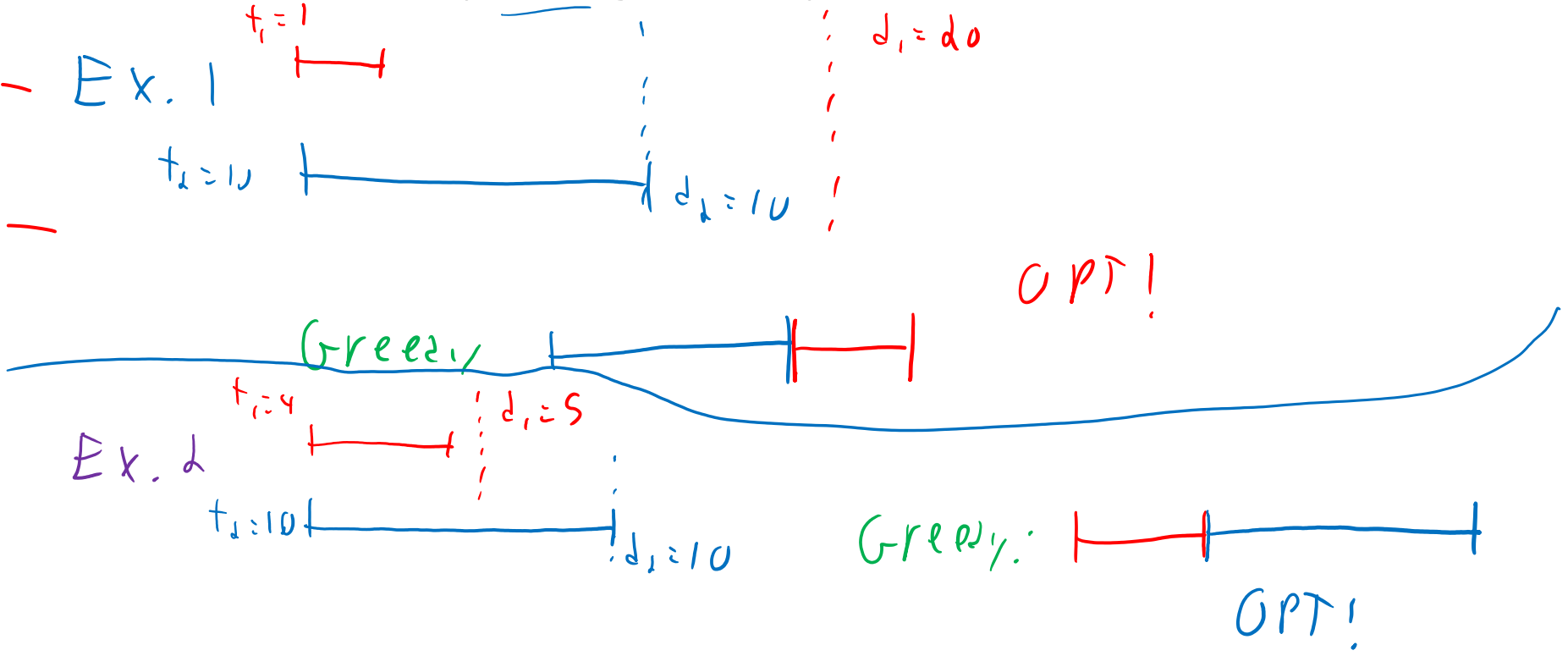
Possible Greedy Rules

- 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



Exchange Argument

- G = greedy schedule, O = (supposedly) 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

Exchange Argument

- G = greedy schedule, O = (supposedly) 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 = (supposedly) optimal schedule

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

Exchange Argument

- We say that two jobs i, j are inverted in O if $d_i < d_j$ but j comes before i
- Claim: an 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

Exchange Argument

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

Exchange Argument

- We say that two jobs i, j are inverted in O if $d_i < d_j$ but j comes before i
- Claim: an 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 consecutive jobs that are inverted then flipping them only reduces the lateness
- G is the unique schedule with no inversions, lateness(G) $\leq$ lateness(O)

- Problems: counting students, stable matching, sorting, n-digit multiplication, array searching, selection, weighted interval scheduling, segmented least squares, knapsack

- Alg. techniques: divide & conquer, dynamic programming, greedy

- Analysis: asymptotic analysis, recursion trees, Master Thm.

- Proof techniques: (strong) induction, contradiction, greedy stays ahead, **exchange argument**