CS 7800: Advanced Algorithms

Class 2: Greedy Algorithms

Jonathan Ullman
September 9, 2025

Administrivia

- Office hours

Current proposal: Wed 2-4pm

- Piazza

- Get desk fixed

Optimization

Defined by the "data"

real numbers

Objective function $f: \mathcal{X} \to \mathbb{R}$

Set of feasible solutions $\mathcal{X}$

Goal: find $x \in \mathcal{X}$ that maximizes $f(x)$

Goal': find $\max_{x \in \mathcal{X}} f(x)$

Types of problems:

- Discrete vs. continuous
- Convex objectives / linear objectives
- Approximate / exact solution

Greedy Algorithms

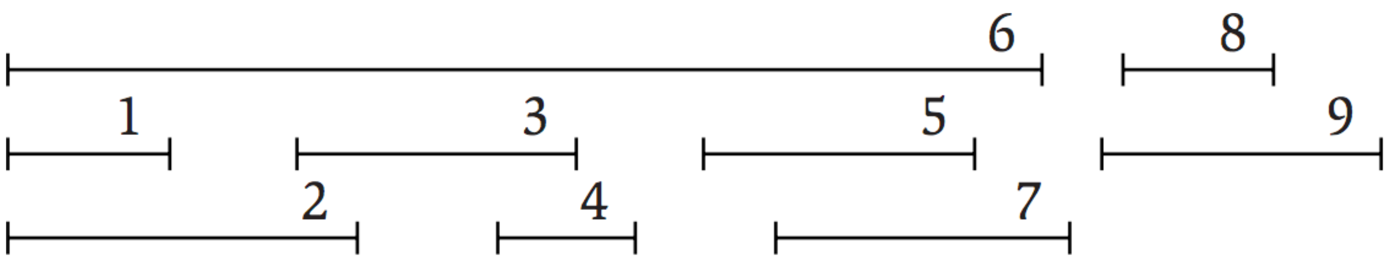
- What's a greedy algorithm?
  - You know it when you see it
  - Typically builds a solution in one "pass" over the data
- Why care about greedy algorithms?
  - Fastest and simplest algorithms imaginable
  - Greedy algorithms are often useful heuristics
  - Greedy algorithms often arise naturally
  - Interesting proof techniques
    - Induction ("Greedy Stays Ahead")
    - Exchange Argument
    - Duality
    - …

Interval Scheduling

- Input: n intervals (s_i, f_i) Assume $s_i < f_i$
- Output: a compatible schedule S with the largest possible size

feasible solutions
  - 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

objective to maximize
  - The size of the schedule is $|S|$

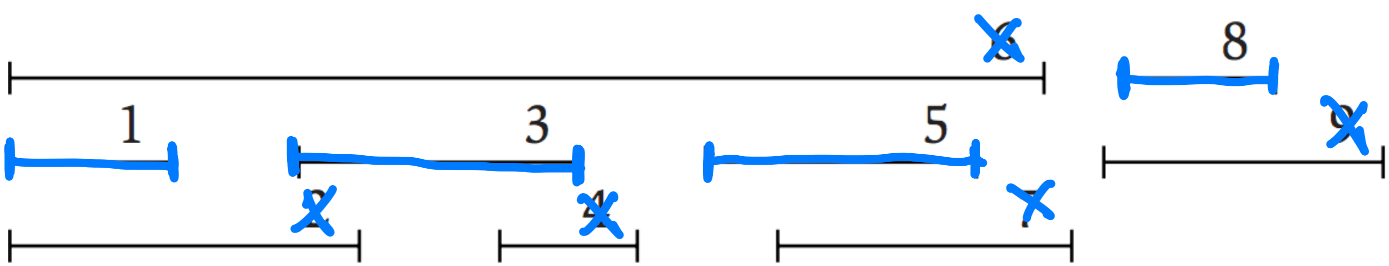


$S = \{1, 3, 8\}$ is compatible/feasible

$S = \{1, 2\}$ is not compatible/feasible

Generic Greedy Algorithm

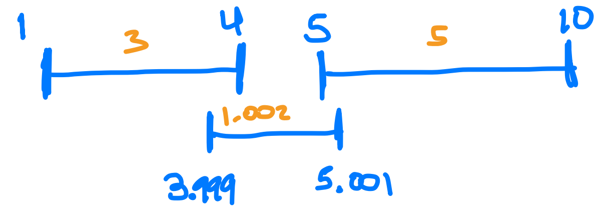
- Sort intervals by [...]
- Let S be empty
- For $i = 1, \dots, n$:
  - If interval i doesn't create a conflict, add i to S
- Return S



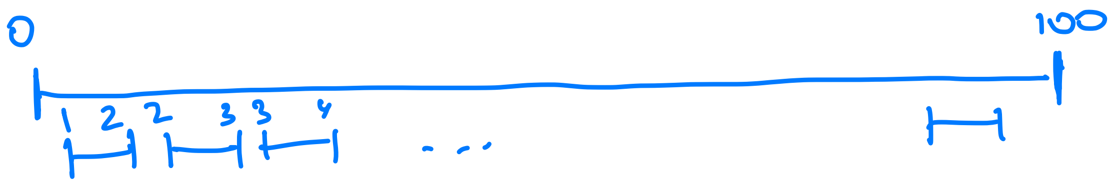
$S = \{1, 3, 5, 8\}$

Possibly Correct Greedy Rules

- Choose the shortest interval first   Doesnt work



- Choose the interval with earliest start first   Doesnt work



- 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

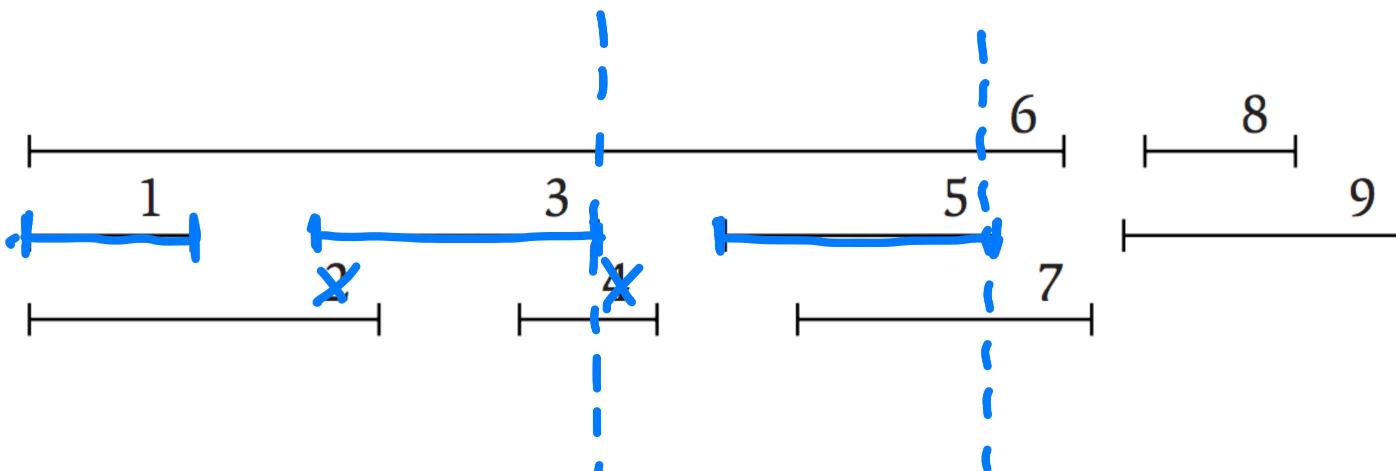
Time

$O(n \log n)$

Loop n tim

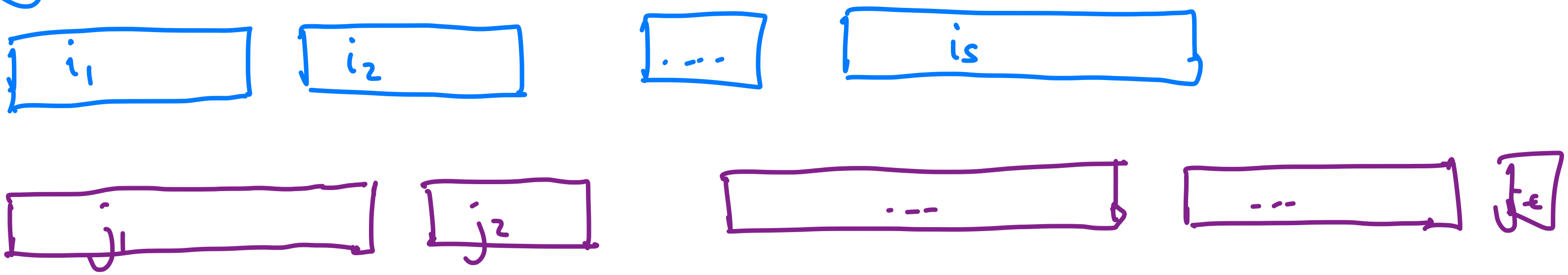
$O(1)$

$O(n \log n)$



Greedy Stays Ahead

Greedy returns a feasible schedule: $\{i_1, i_2, \ldots, i_s\}$

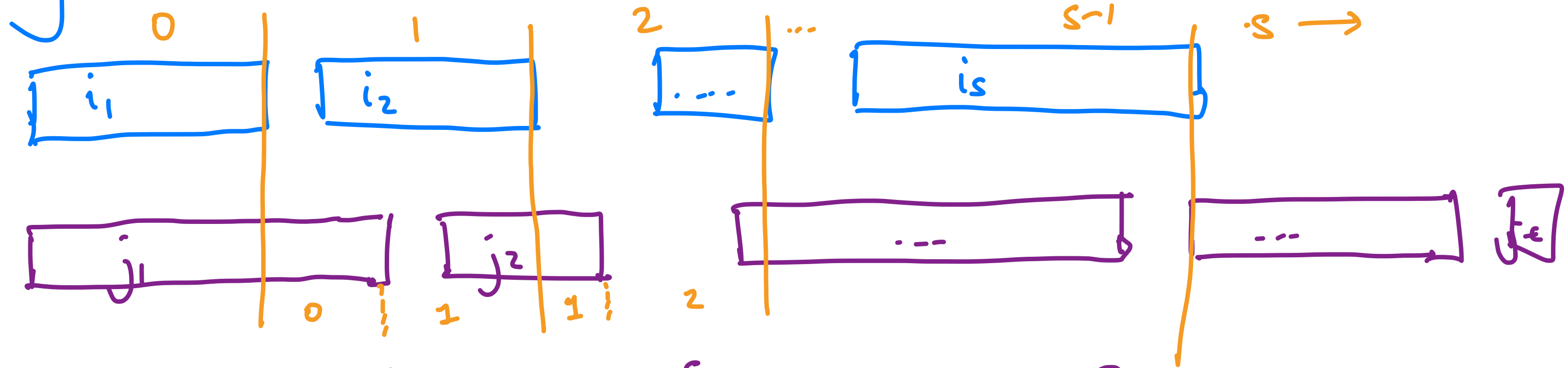


There is an "optimal" solution $\{j_1, j_2, \ldots, j_t\}$

<u>Prove:</u> $s \geq t$ "greedy finds at least as many intervals as any other solution"

Greedy Stays Ahead

Greedy returns a feasible schedule: $\{i_1, i_2, \dots, i_s\}$

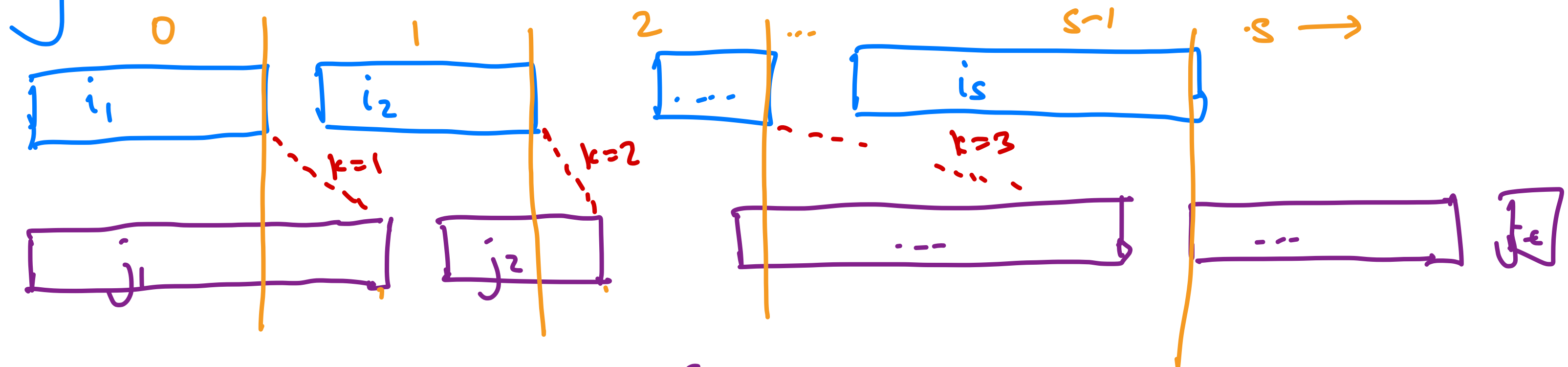


There is an "optimal" solution $\{j_1, j_2, \dots, j_t\}$

Prove: $s \geq t$ "greedy finds at least as many intervals as any other solution"

Greedy Stays Ahead

Greedy returns a feasible schedule: $\{i_1, i_2, \dots, i_s\}$



There is an "optimal" solution $\{j_1, j_2, \dots, j_t\}$

Claim:

"Whenever greedy finishes a new interval, it has completed as many as the optimal schedule"

- for every $k = 1, \dots, s$, $\quad f_{i_k} \le f_{j_k}$

Greedy Stays Ahead

Claim:

"Whenever greedy finishes a new interval, it has completed as many as the optimal schedule"

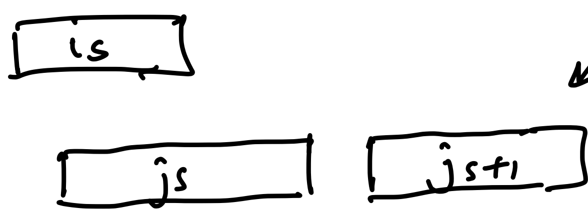
- for every $k = 1, \dots, s$, $\quad f_{i_k} \le f_{j_k}$

Why is this enough?

- $f_{i_s} \le f_{j_s}$
- Suppose $t > s$ "optimal has more intervals than greedy"

greedy:

optimal:



greedy would have added j_{s+1}

... contradiction!

Greedy Stays Ahead

Claim:

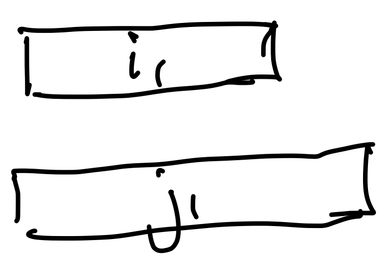
"Whenever greedy finishes a new interval, it has completed as many as the optimal schedule"

- for every $k = 1, \ldots, s$, $f_{i_k} \le f_{j_k}$

Proof: (By induction)

(Base case) for $k = 1$ $f_{i_1} \le f_{j_1}$

True because greedy picks earliest finishing interval first



Greedy Stays Ahead

Claim:

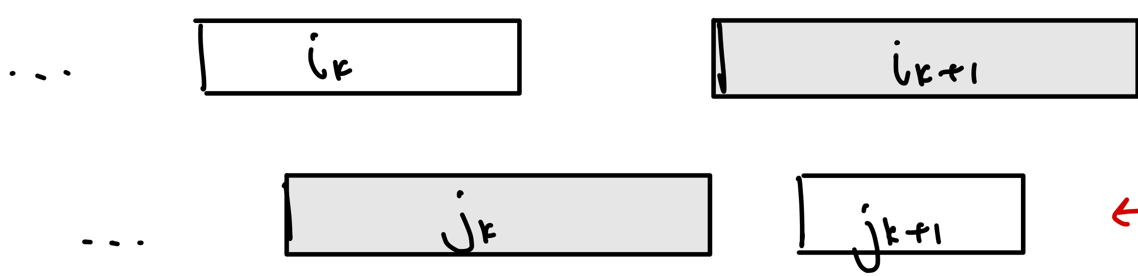
"Whenever greedy finishes a new interval, it has completed as many as the optimal schedule"

- for every $k = 1, \dots, s$, $\quad f_{i_k} \le f_{j_k}$

Proof: (By induction)

(Inductive step) If $f_{i_k} \le f_{j_k}$ then $f_{i_{k+1}} \le f_{j_{k+1}}$

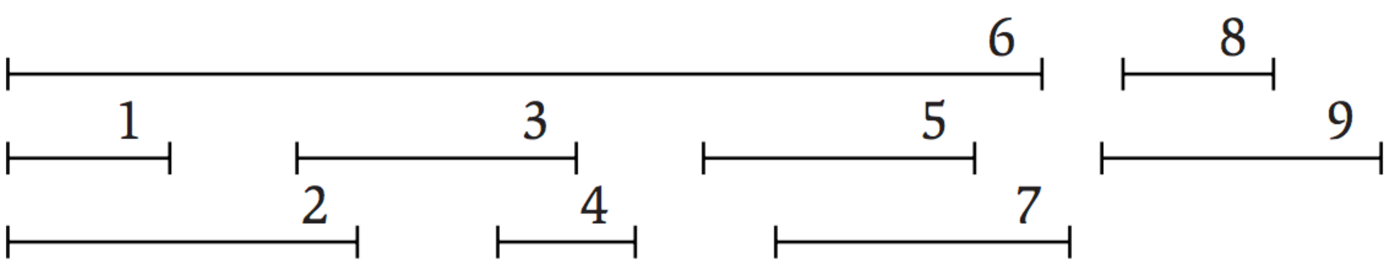
What would it look like if this stmt were false?



greedy looked at j_{k+1} before i_{k+1} and would have picked it if

Interval Scheduling Recap

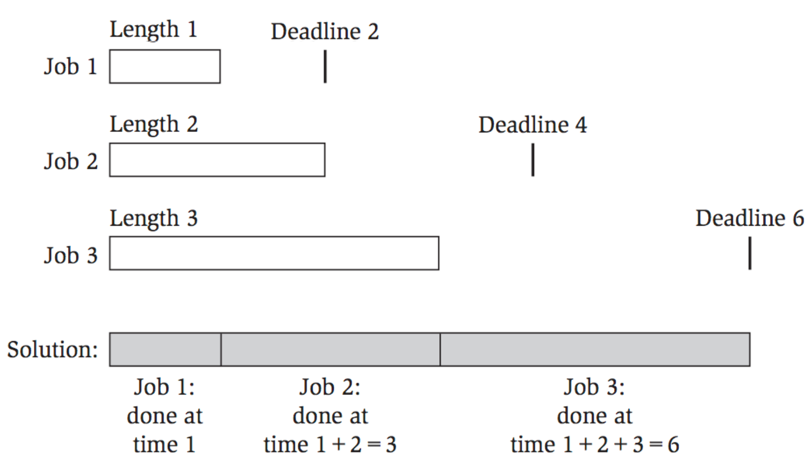
- There is an $O(n \log n)$ time greedy algorithm for the interval scheduling problem
  - Sort intervals by finish time, make one pass over the intervals, and add every compatible interval
  - Analyze using induction (“greedy stays ahead”)



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
  - $f_i = s_i + t_i$
  - Job i starts at s_i finishes f_i, no jobs overlap } Feasible solutions
  - The lateness of job i is $\max\{f_i - d_i, 0\}$
  - The lateness of a schedule is $\max_i\{\max\{f_i - d_i, 0\}\}$

} Objective

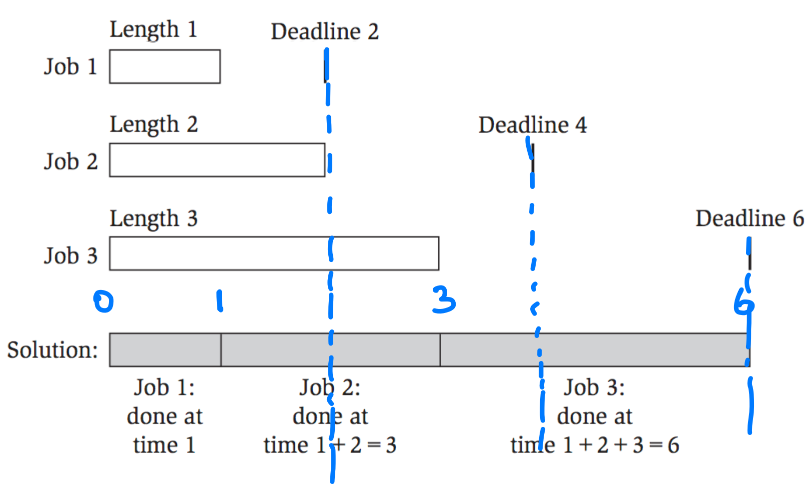


Quick observation:
there is always an optimal solution where jobs are scheduled back-to-back

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
  - $f_i = s_i + t_i$
  - Job i starts at s_i finishes f_i, no jobs overlap } Feasible solutions
  - The lateness of job i is $\max\{f_i - d_i, 0\}$
  - The lateness of a schedule is $\max_i\{\max\{f_i - d_i, 0\}\}$

} Objective



→ schedule with lateness 0

Generic Greedy Algorithm

- Sort jobs by [. . .]

- Schedule jobs consecutively

Possible greedy rules:

- sort by length (longest first)
- sort by "urgency" ($d_i - t_i$)
- sort by deadline (earliest first)