Scientific Computing

Jan 24, 2025

Announcements

→ Office Hours: Mondays + Fridays, 9:30 – 10:30
Cudahy 307

→ HW 1 assigned
On D2L → Dropbox
Due Friday, Jan 31

* Acceptable Sources: Online searches for how to do things in Python **cite!**

Unacceptable: Searching for the questions, AI Tools

Today

→ Greedy Algorithms

Ex: Giving change - How does a cashier give change? Suppose you owe \$3.27 and pay with \$20. They start giving you bills and coins from largest to smallest.

$$\$20 - \$3.27 = \$16.73$$

~~\$100~~	~~\$50~~	~~\$20~~	\$10	5	1	0.25	0.10	~~0.05~~	0.01
			1	1	1	2	2		3
			\$6.73	\$1.73	\$0.73	\$0.23	\$0.03		\$0.

$$\boxed{\$10,\ \$5,\ \$1,\ 2Qs,\ 2Ds,\ 3Ps}$$

= 10 items

This is a greedy algorithm!

At every step, the cashier gives you the largest possible bill.

Is the cashier's algorithm optimal?

fewest # of things

yes, for US denominations

(what are some other versions of "optimal"?)

Theorem: For the US currency denominations listed, the cashier's algorithm is optimal.

Proof: To simplify things, let's just assume the denominations 1, 5, 10, 25 cents. Suppose we are making x cents.

Lemma: The optimal solution will have ≤ 4 P.

Proof: If it had ≥ 5 Ps, we could replace with a nickel.

Lemma: The optimal solution will have ≤ 1 N.
(same proof)

Lemma: The optimal solution will have $\#N + \#D \leq 2$.

Proof: 2N : bad because

1N+2D : bad $\rightsquigarrow$ $= 1Q$

0N+3D : bad $\rightsquigarrow$ $= 1N + 1Q$.

Main proof by induction:

Base case: 0¢ with 0 coins. Optimal ✓

Now assume we're making x ¢.

Case 1: $x < 5$: x pennies, only option

Case 2: $5 \le x < 10$: must have 1 N (or else >4 P)

$$1N + \underbrace{(\text{sol for } x-5)}_{\text{optimal by ind.}}$$

Case 3: $10 \le x < 25$: must have 1D (or else >4 P or >1 N)

$$1D + \underbrace{(\text{sol for } x-10)}_{\text{optimal by ind}}$$

Case 4: $x \geq 25$: must have 1Q (or else >4 P
or >1 N
or 1N+2D
or 3D)

Why? The best you can do without a quarter is 4P+1N+1D or 4P+2D.

So $1Q + \underbrace{(\text{sol for } x-25)}_{\text{optimal by ind.}}$ $\blacksquare$

To include dollar denominations, more cases are needed.

Q: Is the cashier's algorithm optimal for <u>any</u> set of denominations?

Ex: US Postage Denominations

1, 2, 3, 5, 10, 20, 35, 36, 55, 65, 75, 95, 100, 120, 200, 500, 795, 1000, 2635

72¢ optimal: 36¢ 36¢ $\boxed{2}$

greedy: 65¢ + 5¢ + 2¢ $\boxed{3}$

Ex: US Postage Denominations

1, 2, 3, 5, 10, 20, 35, 36, 55, 65, 75, 95, 100, 120, 200, 500, 795, 1000, 2635

To make 72¢: greedy solution = 65 + 5 + 2

optimal solution = 36 + 36

Greedy ≠ Optimal!

Throughout this course we're going to learn about a catalog of problems that model all kinds of real-world problems you might face.

Problem #1: Interval Scheduling (Algorithm Design, by Kleinberg + Tardos)

Suppose you are in charge of a conference room that a lot of people want to use to hold meetings. A bunch of people tell you the times they want to book the room for, and your goal is to accomodate as many groups as possible.

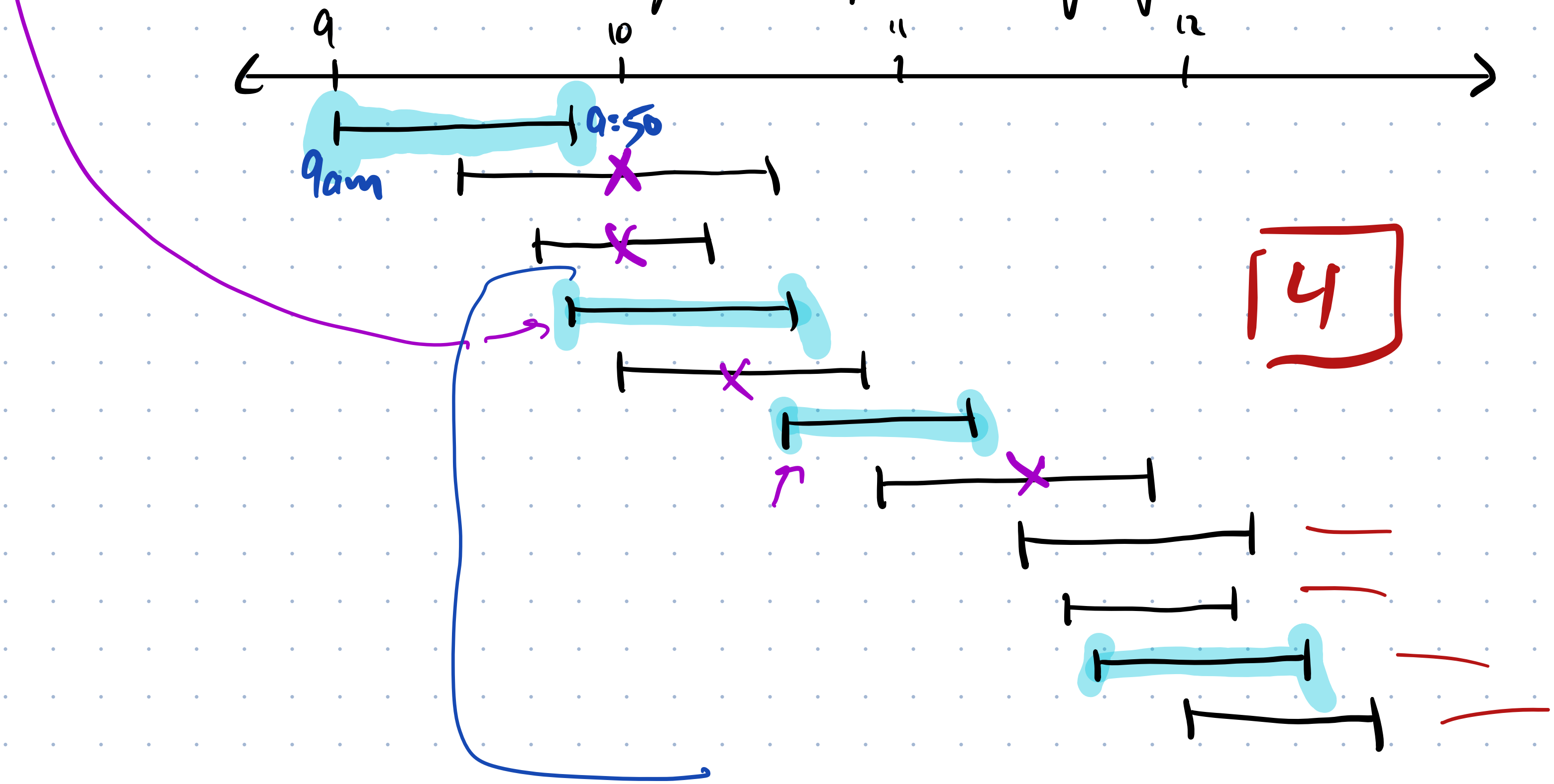
Optimal: Accepting as many as possible

Ex:

Reservations:

- 9am – 9:50am
- 9:30am – 10:30am
- 9:45am – 10:15am
- 9:50am – 10:30am
- 10:00am – 10:50am
- 10:30am – 11:15am
- 11:00am – 11:50am
- 11:30am – 12:15pm
- 11:35am – 12:10pm
- 11:40am – 12:20pm
- 12:00pm – 12:30pm

What is the largest # of meetings you can book?



$\boxed{4}$

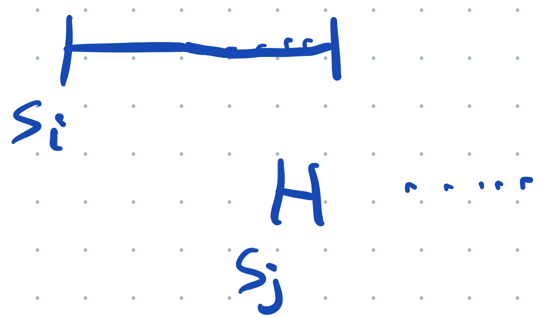
Formal setup:

2-tuple (x, y)

- n requests
- each request has a start time s_i and a finish time f_i (real numbers), with $\boxed{s_i < f_i.}$

Goal: find a maximal size subset of nonoverlapping requests

Two requests (s_i, f_i) and (s_j, f_j) overlap if: Assume $s_j > s_i$



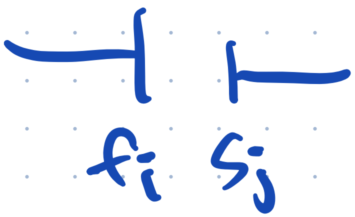
when

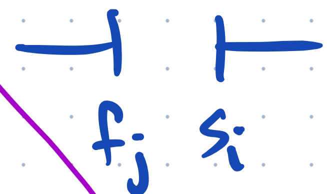
(s_i, f_i) $\quad$ (s_j, f_j)

when do they overlap?

when do they not overlap?

$(f_i \le s_j)$ or $(f_j \le s_i)$





Do overlap if: $(f_i > s_j)$ and $(f_j > s_i)$

Formal setup:

- n requests
- each request has a start time s_i and a finish time f_i (real numbers), with $s_i < f_i$.

Goal: find a maximal size subset of nonoverlapping requests

Two requests (s_i, f_i) and (s_j, f_j) overlap if:

$$s_j < f_i \quad \text{and} \quad s_i < f_j$$

Let's think about possible greedy approaches.

General idea:
* decide on a rule for which meeting is "best"
* pick it, then eliminate conflicts
* repeat

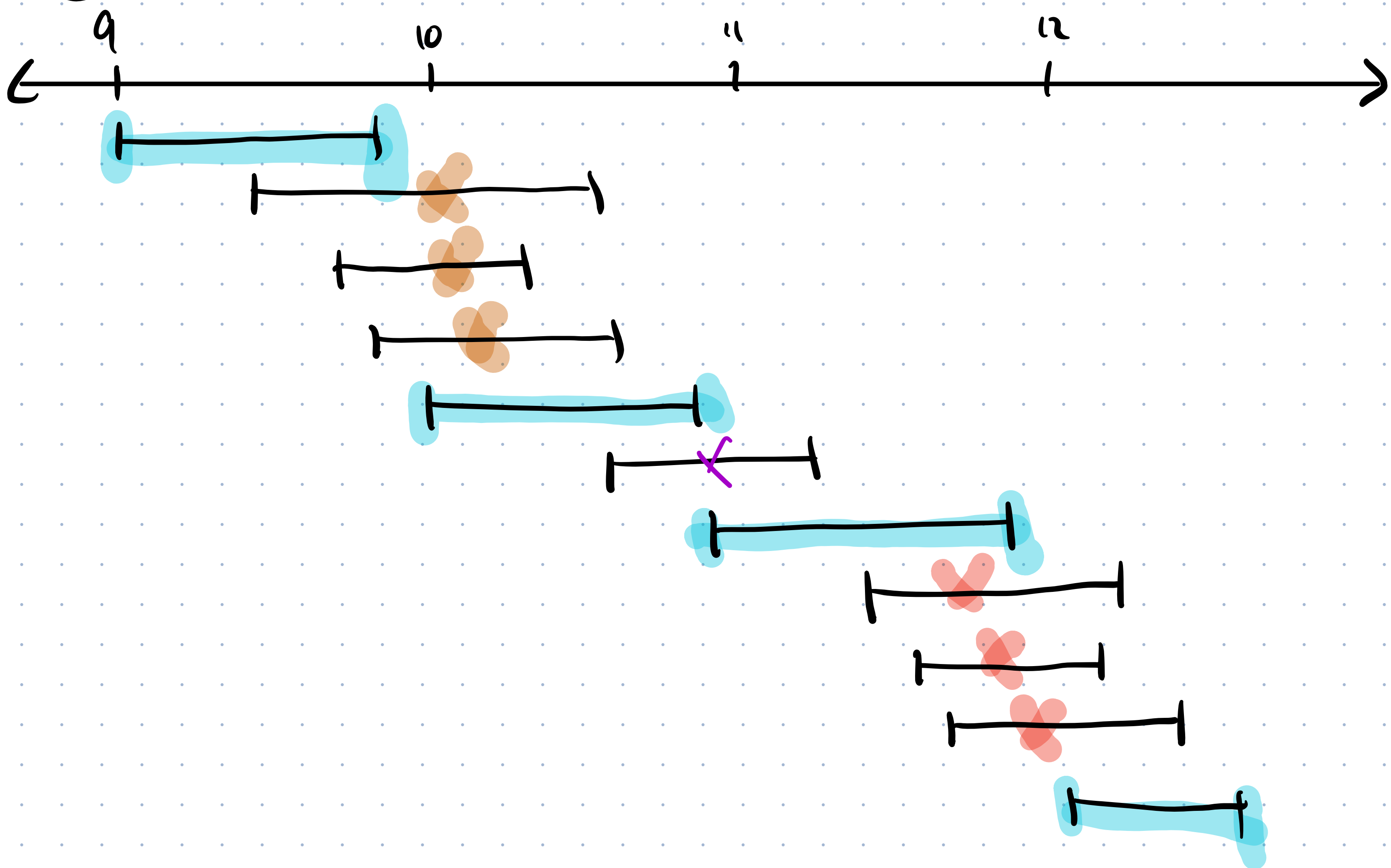
Possibilities: best = shortest

best = earliest start time

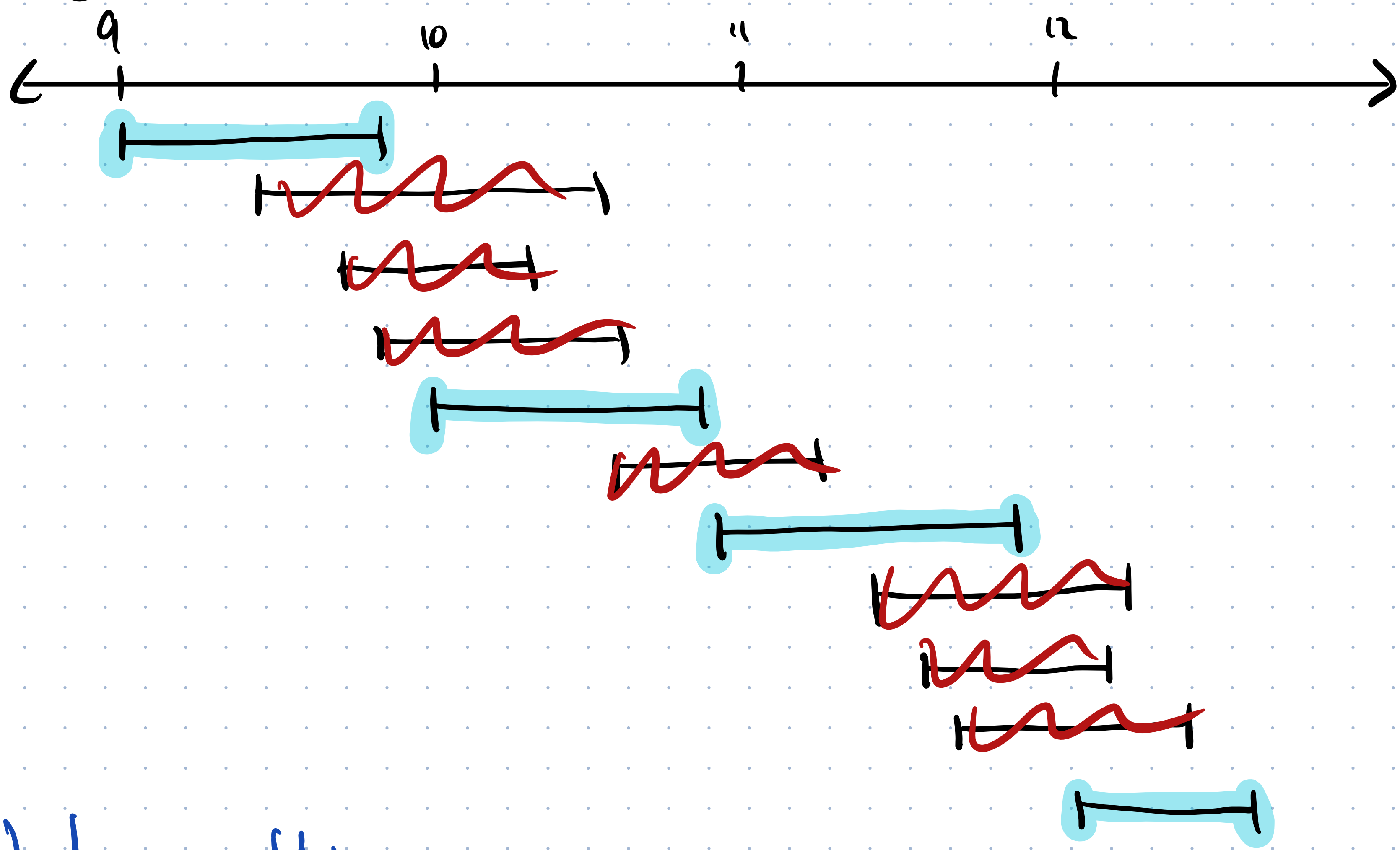
best = latest finish time

best = overlaps with the fewest other meetings

Idea #1: best = earliest start time



Idea #1: best = earliest start time

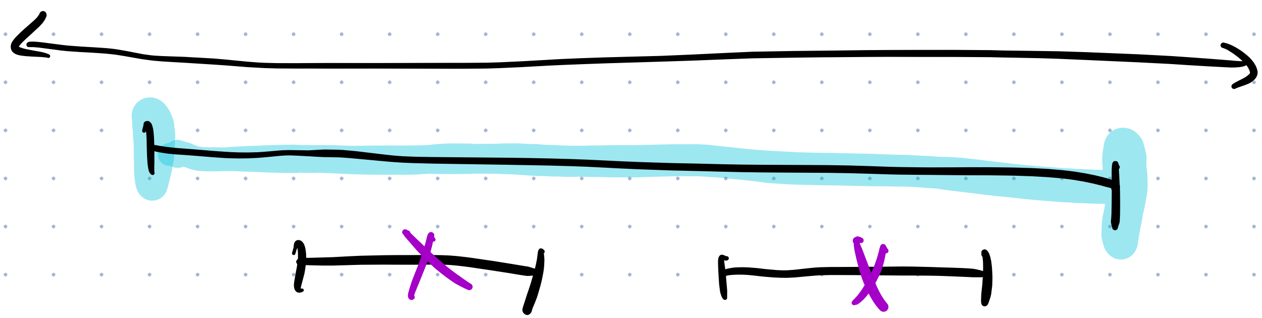


Works in this case.

Can we break it?

Idea #1: best = earliest start time

Can we break it?



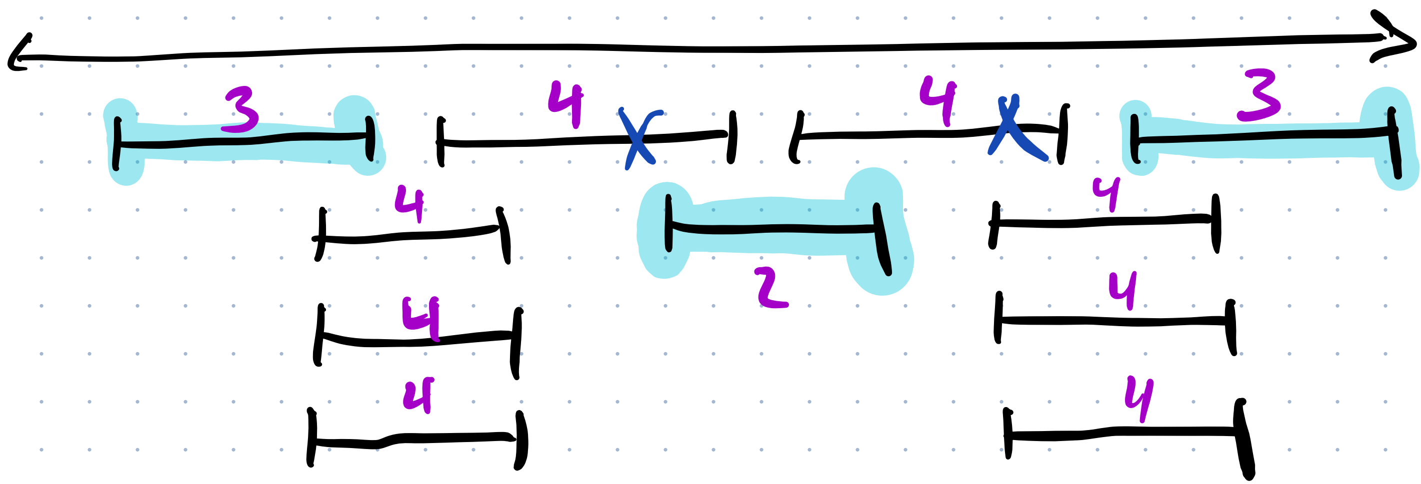
Greedy: 1

Optimal: 2

Idea #2: best = "shortest"

Greedy: 1
Optimal: 2

Idea #3: best = "least conflicts"



Greedy: 3

Optimal: 4

Idea #4: best = "earliest ending time"

This works on all our previous examples.

Can we break it?

Idea #4: best = "earliest ending time"

This works on all our previous examples.
Can we break it?

Intuition: Picking the one that ends earliest gets you credit for a meeting that gets out of the room as quickly as possible.

Algorithm:

Let R be the set of requests.
Let A be the empty set.
While R is non-empty:
    Find the request with earliest end time.
    Add it to A.
    Remove it from R and remove all other requests that are not compatible.
A is the solution

__Theorem:__ The greedy algorithm produces an optimal solution.

Note: There could be other optimal solutions too.