Monday, Jan 30, 2023
Lecture #6
MSSC 6000



## Announcements

* Monday office hours are moved to 1pm-2pm for the rest of the semester
* HW1 due a week from today, 11:59pm

# Topic 3 - Greedy Algorithms (continued)

## Problem #1: Interval Scheduling

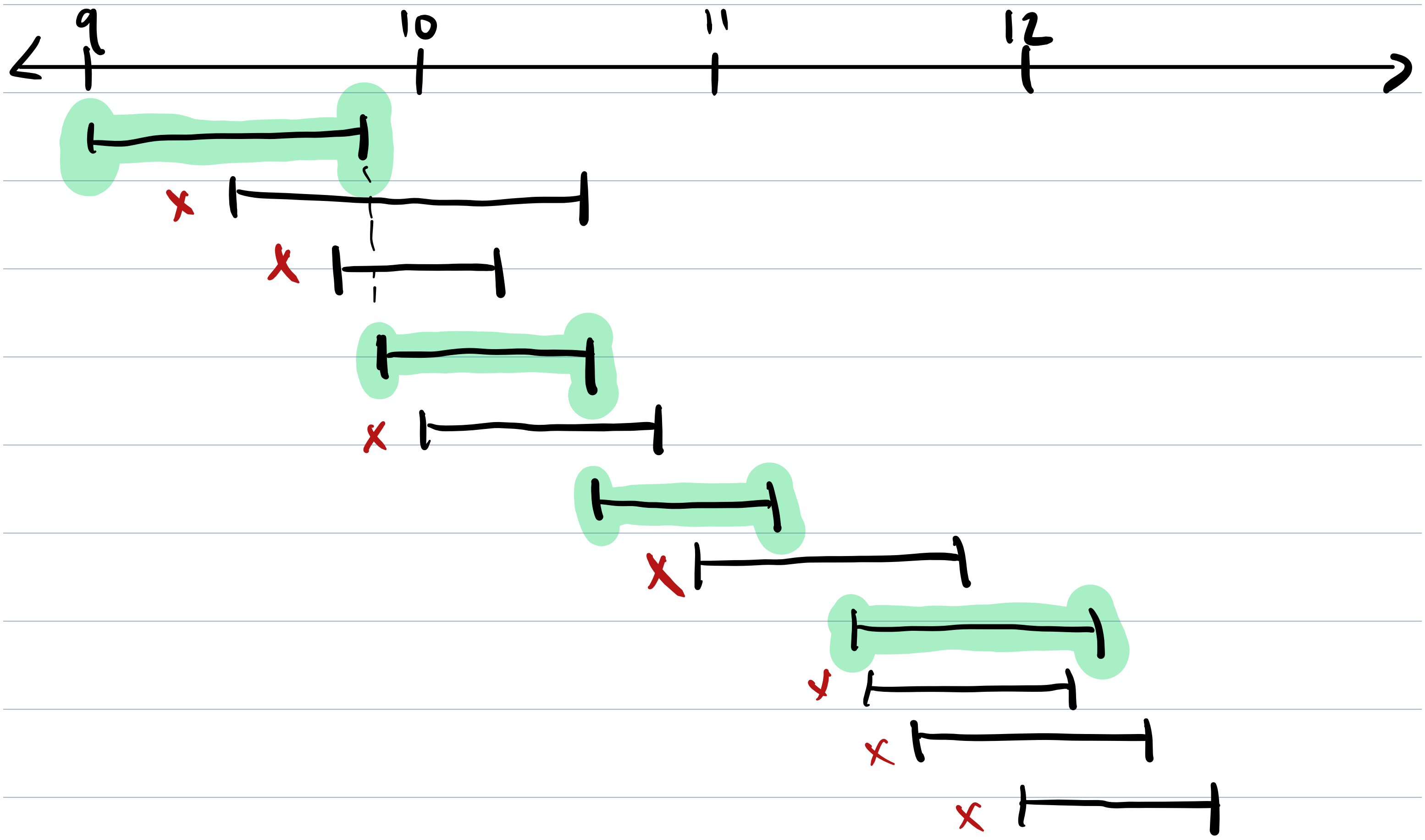
(Algorithm Design, by Kleinberg + Tardos)

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



Ex: Requested times

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




What is the largest # of meetings that we can book?

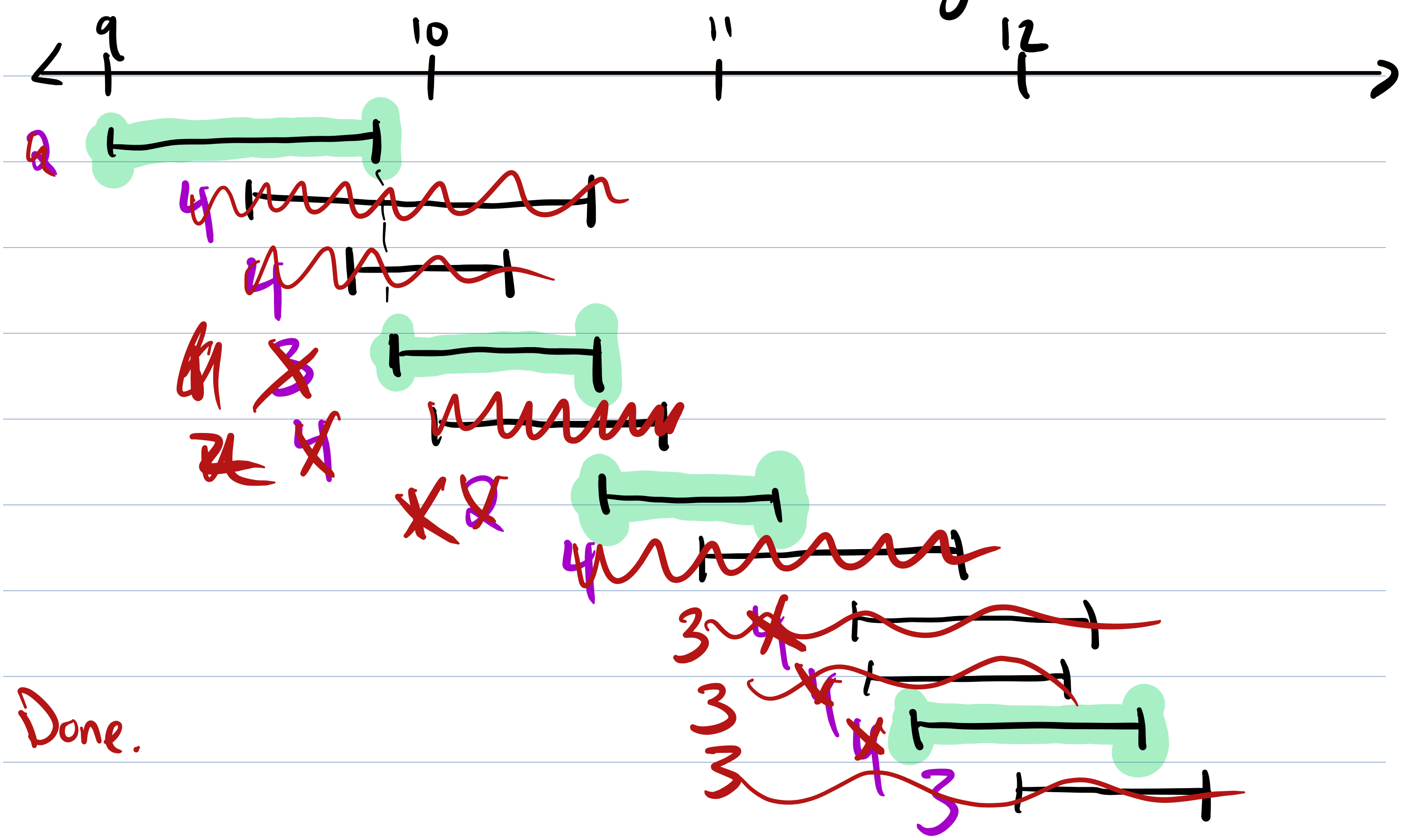
Best = 4 meetings, there are many ③

Let's think about possible greedy approaches.

General idea:

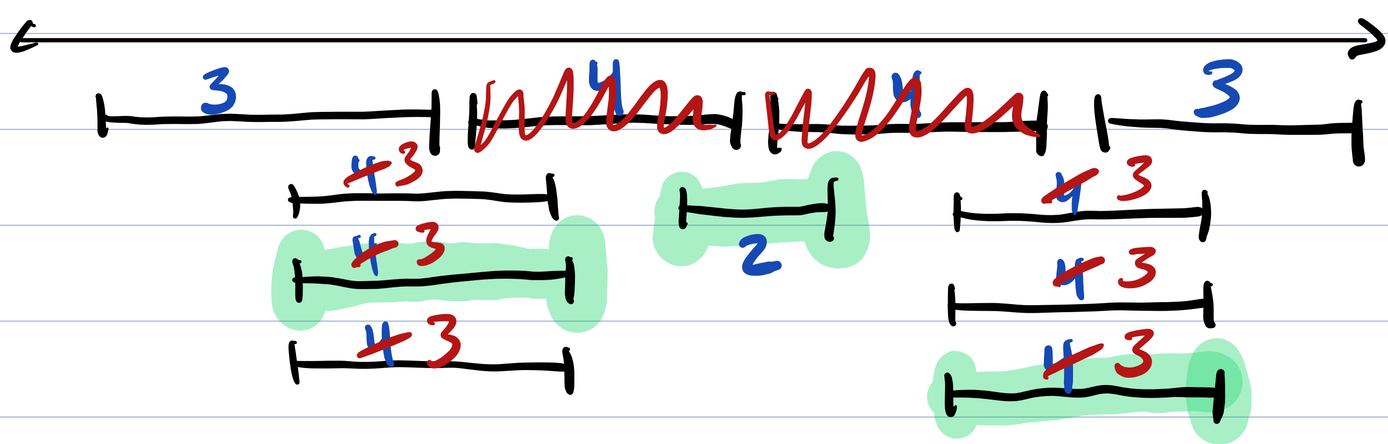
* decide on a rule for which meeting is "best"
* pick it, eliminate conflicts, repeat

Idea #1: best = overlaps with the fewest other meetings



For this example input, using best = "fewest conflicts" tied the optimal solution.

Is this greedy algo actually optimal? Let's try to find a set of requests where this gives a non-optimal sol.

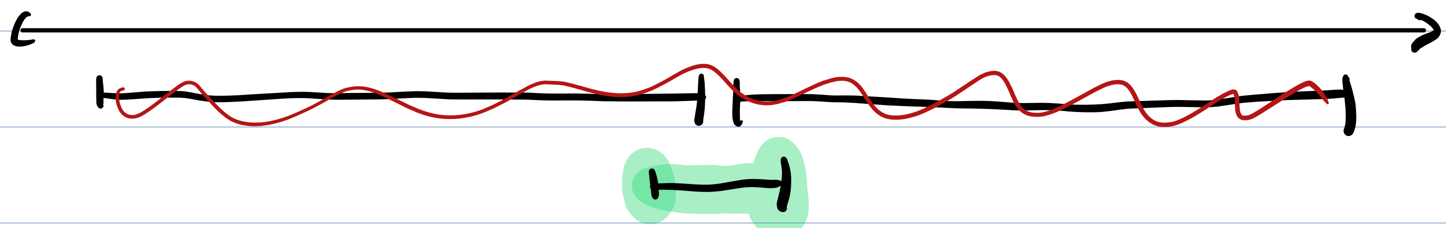



11 requests - with this input, the greedy algorithm gives a solution with 3 meetings.
The optimal solution for this input is <u>4</u> meetings.

This greedy algo. is not optimal.

Idea #2: best = "shortest"

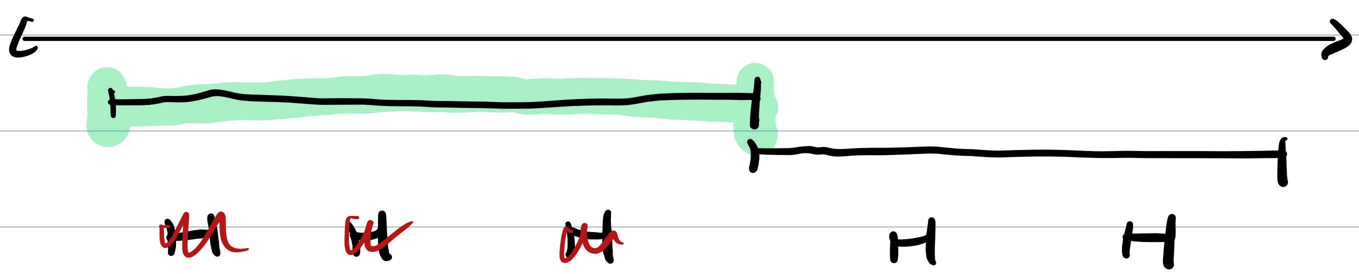
Can we break it?



We get 1 meeting, but the optimal solution has 2.

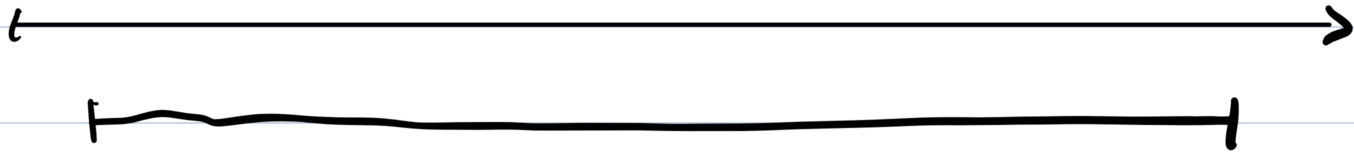
Idea #3: best = minimum gap between meetings

Best = the meeting that has another meeting most closely after it ends



Greedy algo gives 2 or 3 depending how you define it, optimal is 5.

Idea #4: best = least time overlap (whatever that means)



HW 2, try to break this

Idea #5: best = earliest start time

not optimal

Idea #6: best = earliest end time

Can we break it? No.

Good practice to try and see why you can't.

Intuition: Picking the one that ends earliest gets you credit for the 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 the earliest end time
    Add it to A
    Remove it and all other meeting that conflict with it from R
A is the solution

if a tie, doesn't matter which one you pick

Theorem: This greedy algorithm always produces an optimal solution.

Note: There can be many optmal

solutions - different sets of meeting but the same # 

<u>Proof:</u> Let $R$ be a set of requests and let $A$ be the output of our greedy algo. Let $\mathcal{O}$ be an optimal solution.

We want to show that $|A| = |\mathcal{O}|$

(not necessarily $A = \mathcal{O}$)

Obvious: Since $\mathcal{O}$ is optimal, we know $|A| \leq |\mathcal{O}|$. We want to show $|A| \geq |\mathcal{O}|$.

A common strategy when proving your greedy algo. is optimal is to show that the answer it produces <u>stays ahead</u> of any optimal solution.

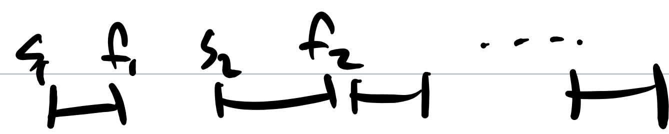
Suppose the requests in $A$ are:

$A = \{(s_1, f_1), (s_2, f_2), \ldots, (s_k, f_k)\}$

and in $\mathcal{O}$:

$\mathcal{O} = \{(s_1', f_1'), (s_2', f_2'), \ldots, (s_m', f_m')\}$

and assume we've written them in chronological order:




$$s_1 < f_1 \le s_2 < f_2 \ldots\ldots$$

$$s_1' < f_1' \le s_2' < f_2' \ldots\ldots$$

Note that $k \le m$ because $|A| \le |\mathcal{O}|$.

Now we'll prove that A "stays ahead" of $\mathcal{O}$:

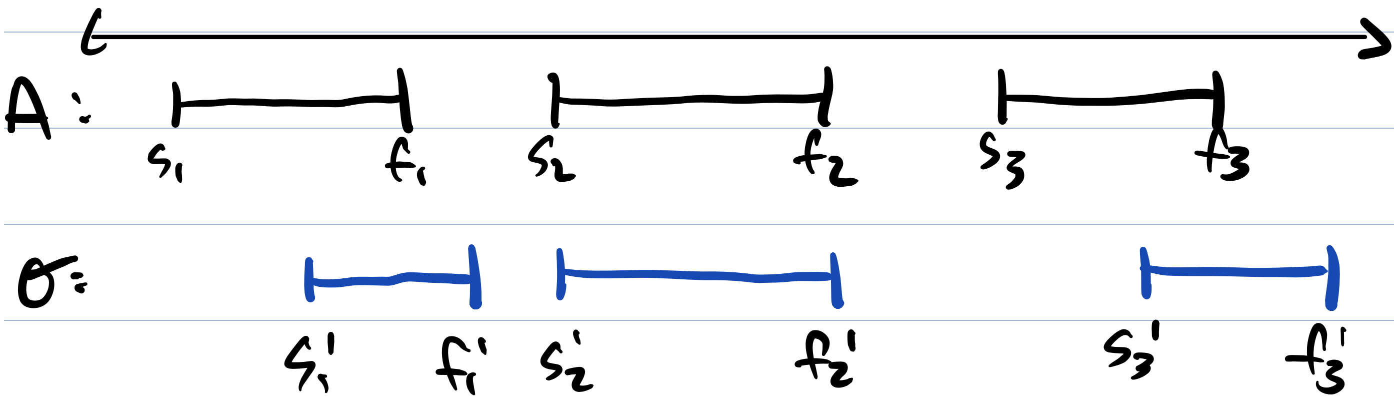
$$f_r \le f_r' \quad \text{for } r = 1, 2, \ldots, k$$

In English: the $r^{th}$ meeting of A finishes before the $r^{th}$ meeting of $\mathcal{O}$.

We'll prove this by induction.

Base Case: $r=1$, want to prove $f_1 \le f_1'$

Why "$\le$" and not "$=$"?




Our first meeting ends earlier (or the same) as the first meeting in any other optimal solution.

The way we know that $f_1 \le f_1'$ is that our algo. by definition picks the meeting with the earliest end time.

Next time: induction step

Assume $f_i \le f_i'$ for $i = 1, 2, \dots, r-1$

Prove $f_r \le f_r'$