More Greedy Algorithms

CSE 421 22AU
Lecture 7

Outline

Last Time

Practice with proofs for greedy algorithms (already knew the algorithm)

Today

Practice with generating greedy algorithms itself.

and maybe the proof at the end.

Section Tomorrow

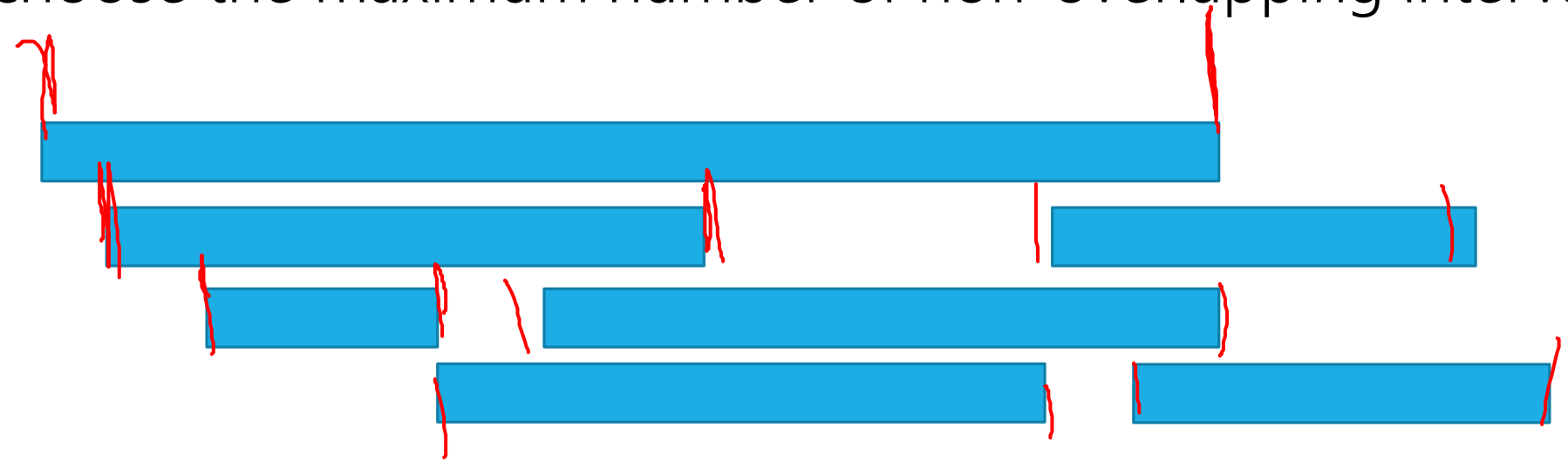
Practice yourself; general problem solving process you'll continue to use all quarter.

Interval Scheduling

You have a single processor, and a set of jobs with fixed start and end times.

Your goal is to maximize the number of jobs you can process.

I.e. choose the maximum number of non-overlapping intervals.

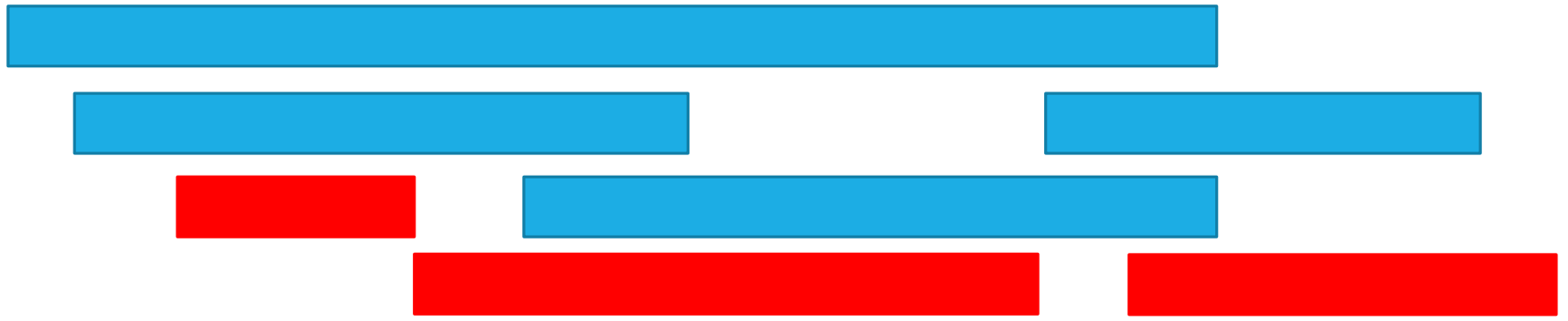


Interval Scheduling

You have a single processor, and a set of jobs with fixed start and end times.

Your goal is to maximize the number of jobs you can process.

I.e. choose the maximum number of non-overlapping intervals.



3 non-overlapping intervals

Interval Scheduling

You have a single processor, and a set of jobs with fixed start and end times.

Your goal is to maximize the number of jobs you can process.

I.e. choose the maximum number of non-overlapping intervals.

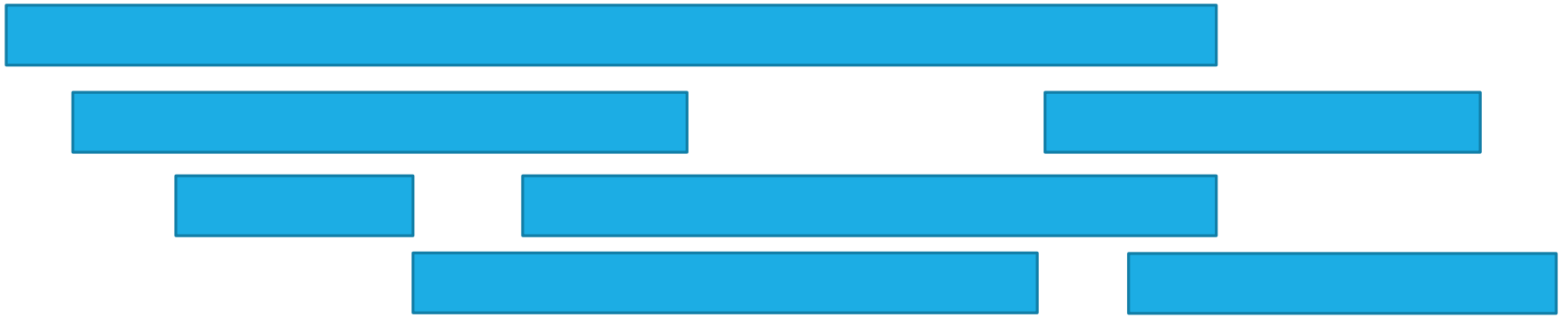
3 other non-overlapping intervals

Interval Scheduling

You have a single processor, and a set of jobs with fixed start and end times.

Your goal is to maximize the number of jobs you can process.

I.e. choose the maximum number of non-overlapping intervals.



OPT is 3 – there is no way to have 4 non-overlapping intervals;
both the red and purple solutions are equally good.

Greedy Ideas

To specify a greedy algorithm, we need to:

Order the elements (intervals)

Choose a rule for deciding whether to add.

Rule: Add interval as long as it doesn't overlap with those we've already selected.

What ordering should we use?

Think of **at least two** orderings you think might work.

Greedy Algorithm

Some possibilities

Earliest end time (add if no overlap with previous selected)

Latest end time

Earliest start time

Latest start time

Shortest interval

Fewest overlaps (with remaining intervals)

Greedy

That list slide is the real difficulty with greedy algorithms.

All of those look at least somewhat plausible at first glance.

With MSTs that was fine – those ideas all worked!

It's not fine here.

They don't all work.

As a first step – try to find counter-examples to narrow down

Greedy Algorithm

Earliest end time

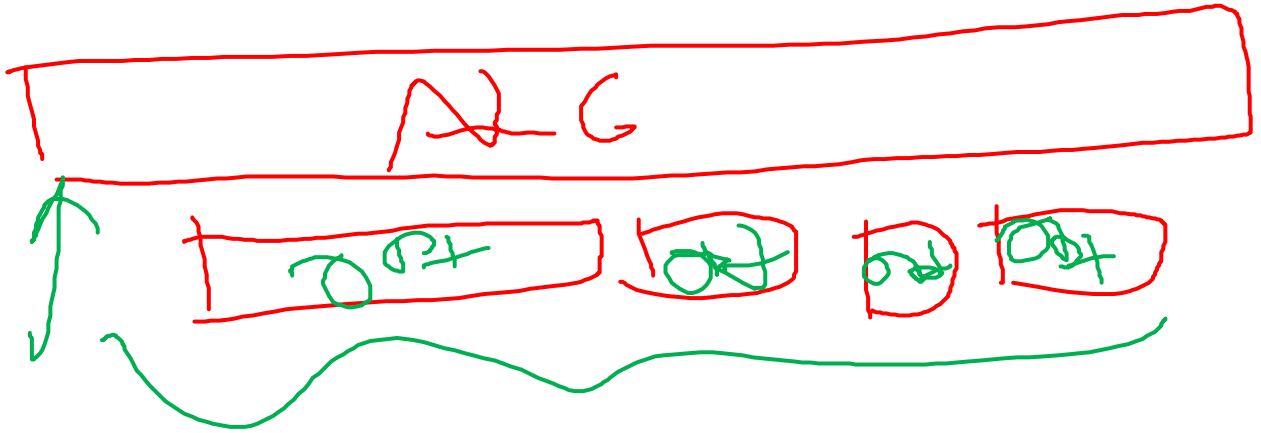
Latest end time

Earliest start time

Latest start time

Shortest interval

Fewest overlaps (with remaining intervals)

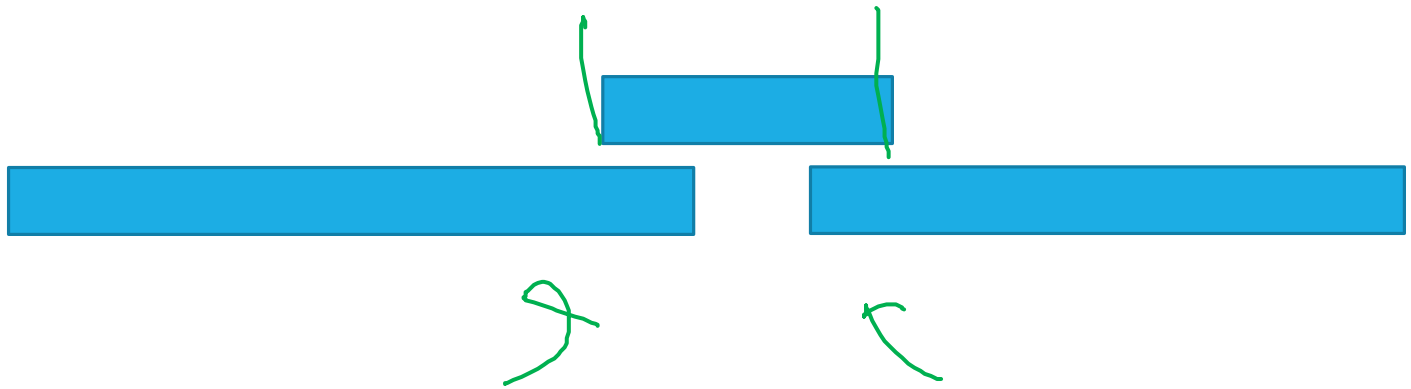


Take Earliest Start Time – Counter Example

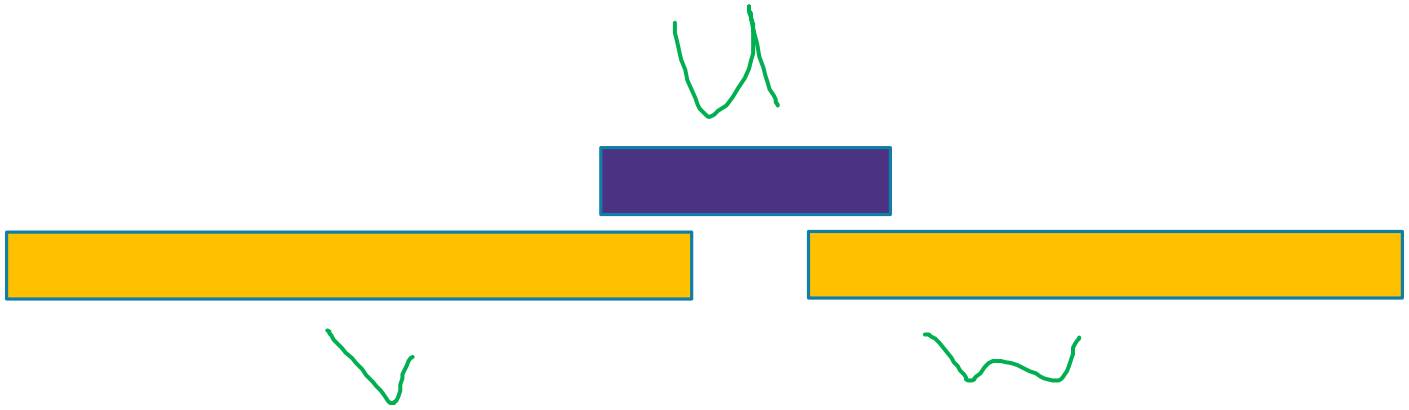
Take Earliest Start Time – Counter Example

Taking the one with the earliest start time doesn't give us the best answer.

Shortest Interval



Shortest Interval



Algorithm finds

Optimum

Taking the shortest interval first doesn't give us the best answer

Greedy Algorithm

Earliest end time

Latest end time ✖

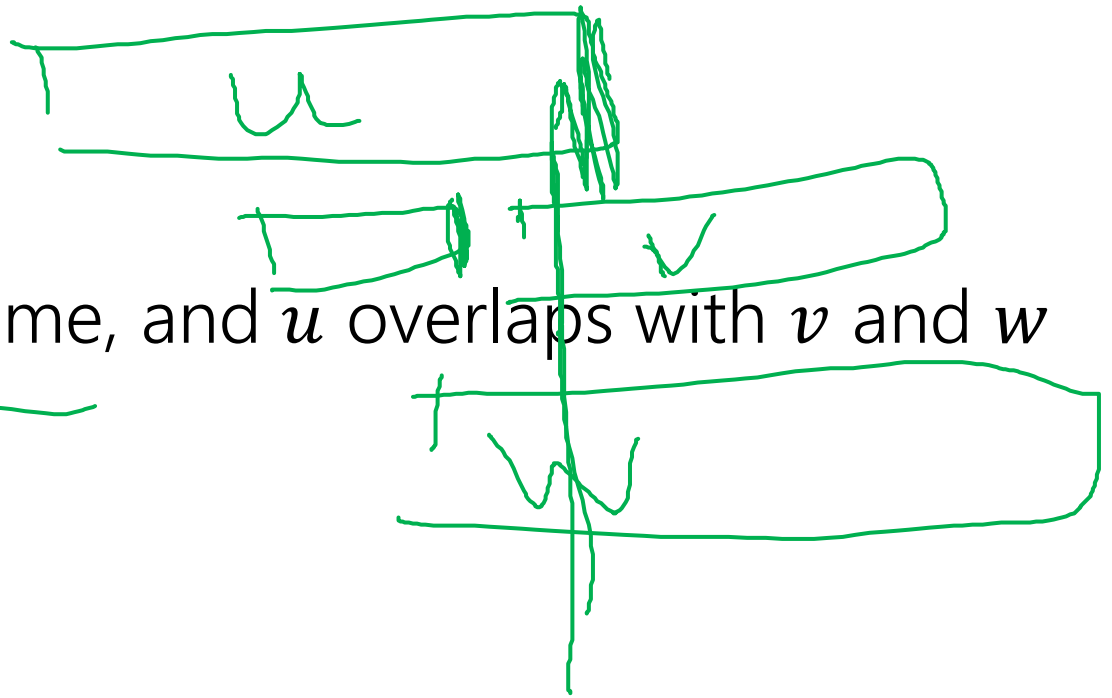
Earliest start time ✖

Latest start time

Shortest interval ✖

Fewest overlaps (with remaining intervals)

Earliest End Time



Intuition: If u has the earliest end time, and u overlaps with v and w then v and w also overlap.

Why?

If u and v overlap, then both are "active" at the instant before u ends (otherwise v would have an earlier end time).

Otherwise v would have an earlier end time than u! By the same reasoning, w is also "active" the instant before u ends. So v and w also overlap with each other.

Earliest End Time

Can you prove it correct?

Do you want to use

Structural Result

Exchange Argument

Greedy Stays Ahead

Exchange Argument

Let $A = a_1, a_2, \ldots, a_k$ be the set of intervals selected by the greedy algorithm, ordered by endtime

OPT= $o_1, o_2, \ldots, o_\ell$ be the maximum set of intervals, ordered by endtime.

Our goal will be to "exchange" to show A has at least as many elements as OPT.

Let a_i, o_i be the first two elements where a_i and o_i aren't the same. Since a_{i-1} and o_{i-1} are the same, neither a_i nor o_i overlaps with any of $o_1, \ldots, o_{i-1}$. And by the greedy choice, a_i ends no later than o_i so a_i doesn't overlap with o_{i+1}. So we can exchange a_i into OPT, replacing o_i and still have OPT be valid.

Exchange Argument

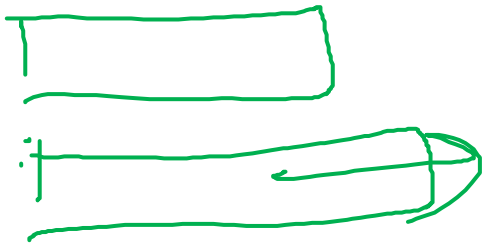
Repeat this argument until we have changed OPT into A.

Can OPT have more elements than A?

No! After repeating the argument, we could change every element of OPT to A. If OPT had another element, it wouldn't overlap with anything in OPT, and therefore can't overlap with anything in A after all the swaps. But then the greedy algorithm would have added it to A.

So A has the same number of elements as OPT does, and we really found an optimal

Greedy Stays Ahead



Let $A = a_1, a_2, \ldots, a_k$ be the set of intervals selected by the greedy algorithm, ordered by endtime

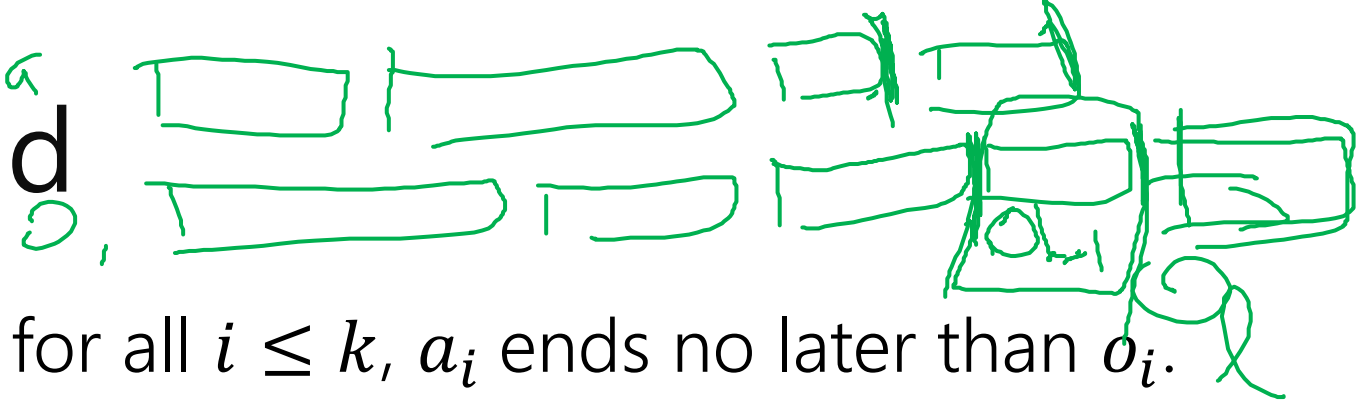
OPT= $o_1, o_2, \ldots, o_\ell$ be the maximum set of intervals, ordered by endtime.

Our goal will be to show that for every i, a_i ends no later than o_i.

Proof by induction:

Base case: a_1 has the earliest end time of any interval (since there are no other intervals in the set when we consider a_1 we always include it), thus a_1 ends no later then o_1.

Greedy Stays Ahead



Inductive Hypothesis: Suppose for all $i \leq k$, a_i ends no later than o_i.

IS: Since (by IH) a_k ends no later than o_k, greedy has access to everything that doesn't overlap with a_k. Since a_k ends no later than o_k, that includes o_{k+1}. Since we take the first one that doesn't overlap, a_{k+1} will also end before o_{k+1}.

Therefore a_{k+1} ends no later than o_{k+1}

Wrapping Up: Since every a_i ends no later than o_i, the last interval greedy selects (a_n) is no later than o_n. There cannot be an o_{n+1}, as if it didn't overlap with o_n it also wouldn't overlap with a_n and would have been added by greedy.

Greedy Algorithm

- Earliest end time ✓
- Latest end time ✗
- Earliest start time ✗
- Latest start time
- Shortest interval ✗
- Fewest overlaps (with remaining intervals)

Other Greedy Algorithms

It turns out latest start time and fewest overlaps also work.

Latest start time is actually the same as earliest end time (imagine "reflecting" all the jobs along the time axis – the one with the earliest end time ends up having the last start time).

What about fewest overlaps?

Easiest proof Robbie knows observes that fewest overlaps means you have the earliest finish time (among a certain subset of the intervals)

Greedy Algorithm

- Earliest end time ✓
- Latest end time ✗
- Earliest start time ✗
- Latest start time ✓
- Shortest interval ✗
- Fewest overlaps (with remaining intervals) ✗

Retcon!
Robbie was wrong about this in lecture --- fewest overlaps isn't optimal either!

Summary

Greedy algorithms

You'll probably have 2 (or 3…or 6) ideas for greedy algorithms. Check some simple examples before you implement!
Greedy algorithms rarely work.

When they work AND you can prove they work, they're great!

Proofs are often tricky

Structural results are the hardest to come up with, but the most versatile.

Greedy stays ahead usually use induction

Exchange start with the **first** difference between greedy and optimal.