# Greedy Algorithms + Dynamic Programming

July 31st, 2021 (Class #4)

# Scheduling Lectures

The lecture hall is open all day!
But everyone wants to use it...

Professor A wants to lecture from 1:00 to 2:30,

B wants 1:30 to 2:45,

C wants 12:30 to 1:15,

D wants 1:00 to 1:20...

and so on for hundreds of professors!

# What might we want to know?

Can we make all the professors happy?

Algorithm:

And if not...

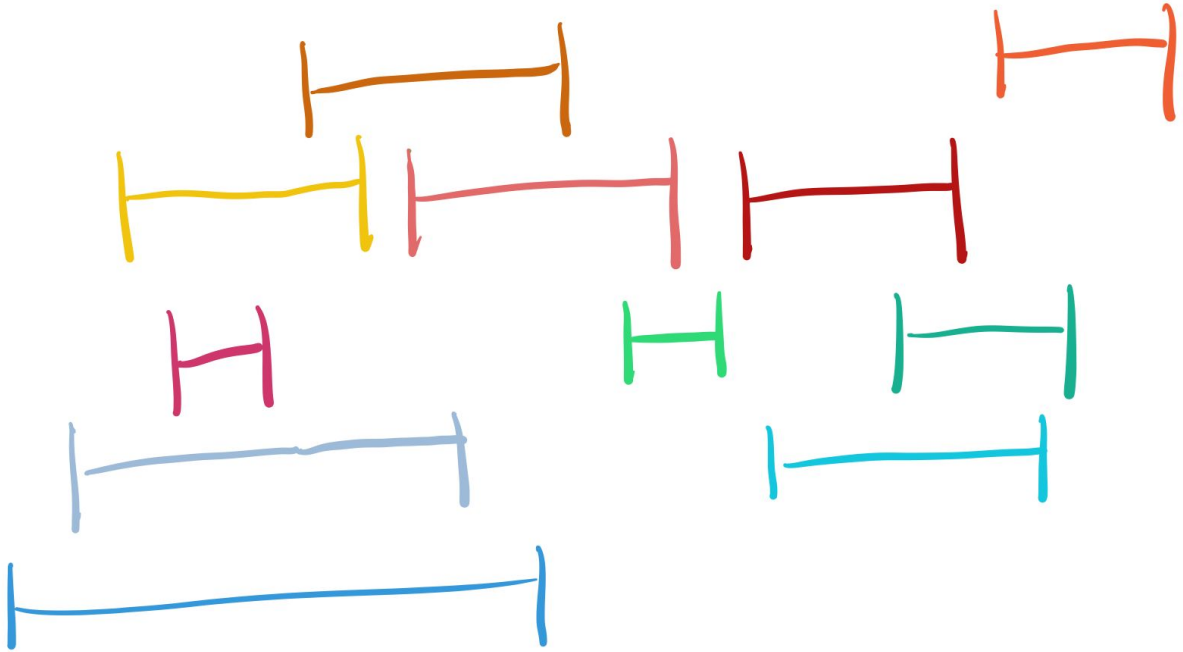
What is maximum possible number of professors we can satisfy?

# Maximum Disjoint Intervals

**Input:** A set of intervals!

**Output:** Maximum size subset of intervals that are disjoint.

# Can you think of a good algorithm?



# Greedy algorithm...

Just keep choosing intervals until you can't anymore...

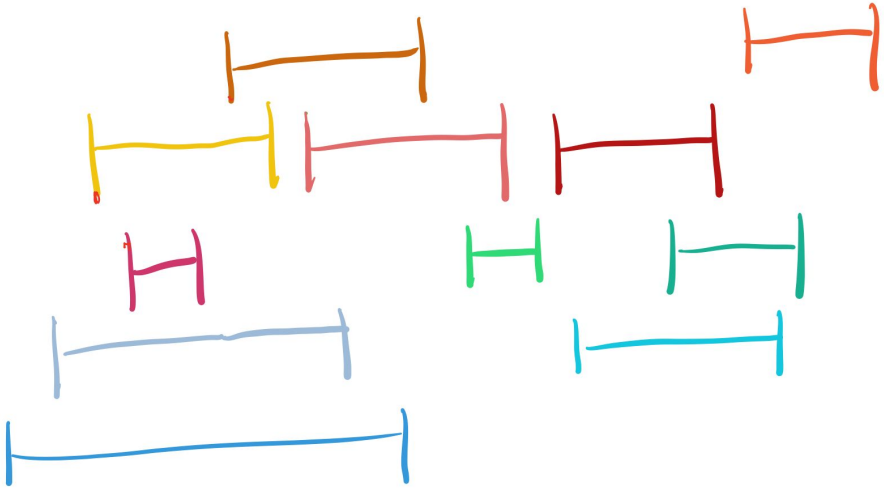
and hope you never make a mistake!

## Designing a greedy algorithm...

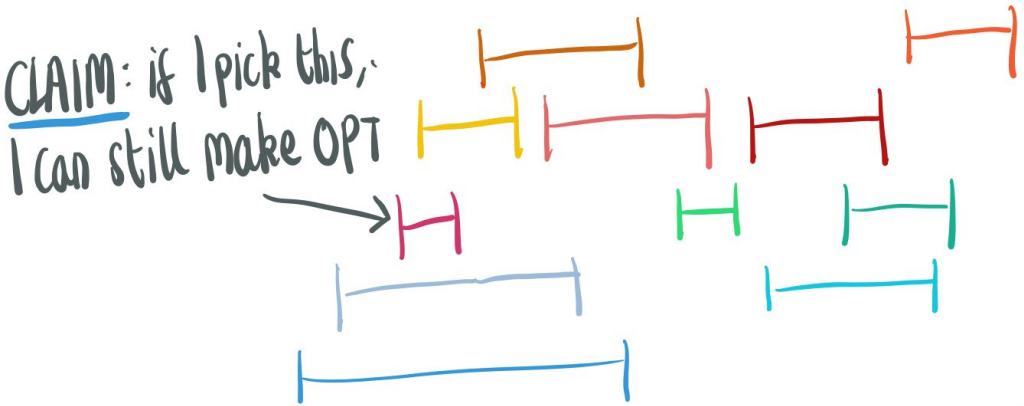
We need a rule for picking intervals.

- shortest?
- fewest conflicts?
- earliest start?
- earliest end?

# Trying out greedy algorithms...



# We need an algorithm that is **provably correct.**




Suppose that a solution doesn't contain ⊢.

Then replace the earliest ending interval in Sol. w/ ⊢.

Since ⊢ ends even earlier, solution is still valid! So there exists an optimum solution containing ⊢!

# Greedy Algorithm for Maximum Disjoint Intervals

Earliest end works!

Is it fast? Yes!

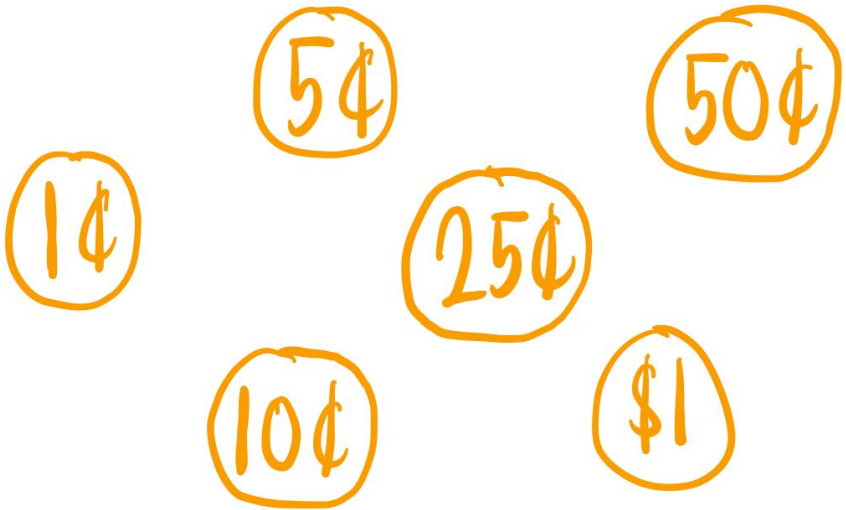
$O(n)$

↑

number of intervals

# Break for 5 Minutes

# Paying the fewest coins



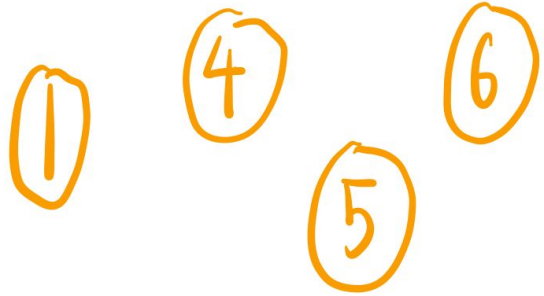

Your total: $2.88.

## Greedy algorithm for paying the fewest coins...

Keep picking the largest possible coin!

greedy rule

# Does this always work?




Yout total: 9.

Is there a method that works for all coin systems?

# Dynamic Programming (DP)

Build up from smaller problems to larger problems.

# Dynamic Programming for Fewest Coins

$C_k$: Fewest coins to make $k$?

Use the solutions to $C_{k-1}, C_{k-2}, \ldots$ to help!

Building Bottom Up

# Recurrence Relation for Fewest Coins

$$C_k = \min\left(C_{k-1}, C_{k-4}, C_{k-5}, C_{k-6}\right).$$

Work from bottom up,
until we get to desired value.

Pretty fast! $O(nC)$

- $n$: number of coin types
- $C$: desired value

# Optimal Substructure

Greedy and DP have something in common...

Both exploit optimal substructure.

optimal solutions to smaller subproblems help find optimum of larger problem.

# Greedy vs DP

Greedy makes a greedy decision to form a single subproblem.

DP considers multiple choices + subproblems and chooses the best one.