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.

CLAIM: if I pick this, I can still make OPT.

Suppose that a solution doesn’t contain H.
Then replace the earliest ending interval in sol. w/ H.

Since H ends even earlier, solution is still valid!

So there exists an optimum solution containing H!
Greedy Algorithm for Maximum Disjoint Intervals

Earliest end works!

Is it fast? Yes!

O(n)
Break for 5 Minutes
Paying the fewest coins

Your total: $2.88.
Greedy algorithm for paying the fewest coins...

Keep picking the \underline{Largest} possible coin!

\underline{Greedy rule}
Does this always work?

Your 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

\[ C_k : \text{Fewest coins to make } k \]

Use the solutions to \( C_{k-1}, C_{k-2}, \ldots \) to help!
<table>
<thead>
<tr>
<th>k</th>
<th>C_k</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td></td>
</tr>
</tbody>
</table>
Recurrence Relation for Fewest Coins

\[ C_k = \min(C_{k+1}, C_{k+4}, C_{k+5}, C_{k+6}). \]

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

Pretty fast! \( O(nC) \)
Greedy and DP have something in common...

Both exploit **optimal substructure.**

Optimal solutions to smaller subproblems help find optimum of target problem.
**Greedy vs DP**

**Greedy** makes a greedy decision to form a single subproblem.

**DP** considers multiple choices and subproblems, and chooses the best one.