Greedy Algorithms

CS366: Design & Analysis of Algorithms

Tuesday, October 28, 2025
Prof. Declan Gray-Mullen

Today's Agenda

1. What are greedy algorithms?
2. Interval scheduling problem
3. Proving greedy correctness
4. When greedy works (and when it doesn't)
5. Preview: PA4 - Greedy (Connecting Sticks)

What is a Greedy Algorithm?

Definition: An algorithm that builds a solution iteratively by:

1. Making the choice that looks best at each step
2. Never reconsidering or backtracking from previous choices
3. Know that local optimal choices that lead to a global optimum

Question: For what problems does greedy actually work?

Real-World Example

Scenario: You're at the grocery store with 20 items and need to get your melting ice cream home ASAP

Greedy Strategy: Always go to the shortest checkout line right now

Question: Is this optimal?

Depends! Lines can change, new cashiers can open...

Greedy makes a locally optimal choice, but may not be globally optimal

When Greedy Works

A greedy algorithm is optimal when:

1. Greedy Choice Property

    A locally optimal choice at each step leads to a globally optimal solution

2. Optimal Substructure

    An optimal solution to the problem contains optimal solutions to subproblems

Both properties must hold!

Today's Problem - Course Scheduling

Problem: Maximize the number of courses you can attend when courses have time conflicts

Example Courses:

Course A: 9:00 AM - 10:30 AM

Course B: 9:30 AM - 11:00 AM

Course C: 10:45 AM - 12:15 PM

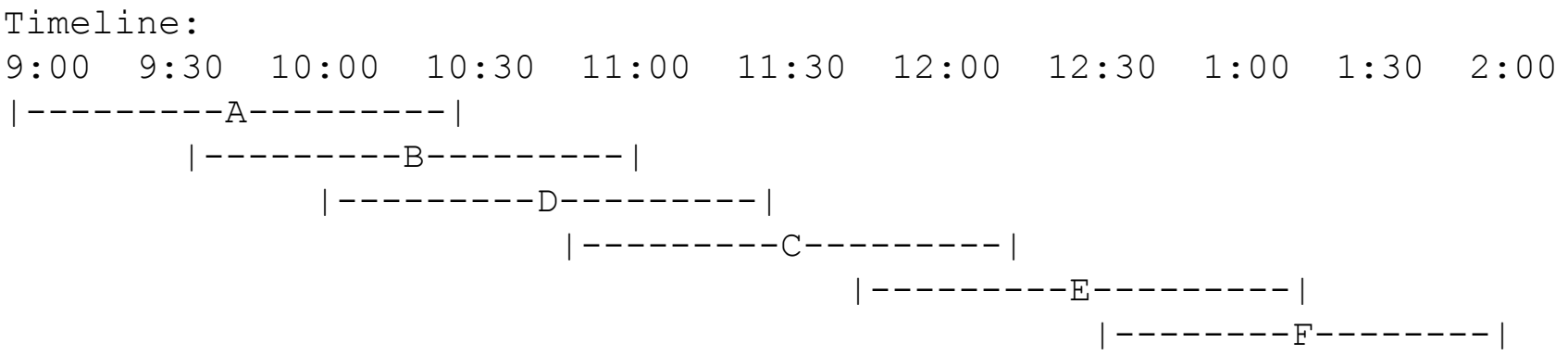
Course D: 10:00 AM - 11:30 AM

Course E: 11:45 AM - 1:15 PM

Course F: 12:30 PM - 2:00 PM

Goal: Select maximum number of non-overlapping courses

Visualizing the Problem



Questions:

- Which courses overlap?
- What's the maximum number you think we can attend?

Activity!

Course A: 9:00 AM - 10:30 AM
Course B: 9:30 AM - 11:00 AM
Course C: 10:45 AM - 12:15 PM
Course D: 10:00 AM - 11:30 AM
Course E: 11:45 AM - 1:15 PM
Course F: 12:30 PM - 2:00 PM

Your Task:

1. Get a partner
2. Draw the timeline
3. Try different greedy strategies:
    - Shortest course first?
    - Earliest start time first?
    - Earliest finish time first?
4. Count how many courses each strategy gives you

Let's Compare Strategies

Strategy 1: Shortest duration first

All courses are 1.5 hours!

This doesn't help us 🤷

Strategy 2: Earliest start time (A → B → D → C → E → F)

Select A, then C, then F

Result: 3 courses ✓ (can you imagine an input that won't work?)

Strategy 3: Earliest finish time (A → B → D → C → E → F)

Select A, then C, then F

Result: 3 courses ✓

Why Earliest Finish Time?

Intuition:

Finishing early leaves more time for future courses

Think: 'Get done early, have more time for other things'

But this is just intuition - we need a PROOF!

Proving Greedy Correctness

Challenge: How do we prove our greedy algorithm is optimal?

Three Main Techniques:

1. Exchange Argument ← Most common
2. Greedy Stays Ahead
3. Structural Induction

Today: Focus on exchange arguments

The Exchange Argument

Basic Idea:

1. Assume there's an optimal solution O that differs from greedy G
2. Find where they differ
3. 'Exchange' O to match G without making it worse
4. Repeat until O = G
5. Conclusion: G must be optimal!

Think of it as: 'Any difference from greedy can be fixed'

Course Scheduling Proof - Setup

Claim: Earliest Finish Time First maximizes the number of courses attended

Proof:

Let G = greedy schedule (sorted by finish time)

Suppose there exists an optimal schedule O where O ≠ G

Since O ≠ G, there must be a first position where they differ

Course Scheduling Proof - The Exchange

At position k, suppose:

Greedy picks course g (finishes at time t_1)

Optimal picks course o (finishes at time t_2)

Key observation: g finishes earlier than o

(that's how greedy works!) So $t_1 < t_2$

What if we swap them in O?

Replace o with g in the optimal schedule: g finishes earlier (at t_1 instead of t_2)

Any course after o that didn't overlap with o won't overlap with g!

We haven't lost any courses

Course Scheduling Proof - Conclusion

What we showed:

Any time optimal chooses a later-finishing course

We can swap it for greedy's earlier-finishing course

This swap doesn't reduce the number of courses

Keep swapping until optimal matches greedy

Therefore: The greedy schedule must be optimal

Key Insight

The exchange argument shows:

We can transform ANY optimal solution into the greedy solution

Without making it worse

Therefore the greedy solution is also optimal

This technique works for many greedy problems!

But Greedy Doesn't Always Work!

Counterexample - Weighted Intervals:

Course A: 9:00-11:00, worth 15 credits

Course B: 9:00-10:00, worth 5 credits

Course C: 10:00-11:00, worth 5 credits

Greedy (earliest finish): Pick B, then C $\longrightarrow$ 10 credits

Optimal: Pick A alone $\longrightarrow$ 15 credits

Greedy fails!

Problem lacks greedy choice property

When courses have weights, we need Dynamic Programming

Try This Challenge Problem!

Given these activities (start, finish):

```
activities = [(1, 3) (2, 5) (4, 7) (1, 8) (5, 9) (8, 10)
              (9, 11) (11, 14) (13, 16)]
```

Questions:

1. What does the greedy algorithm (earliest finish time) select?
2. How many activities can you attend?
3. Is there any other valid schedule with the same number?
4. What is the runtime the greedy interval scheduling?

Hint: Sort by finish time first, then apply the greedy strategy

Problem 3 - Connecting Sticks

Preview of PA4!

Problem: Connect n sticks into one stick

Cost of connecting two sticks = sum of their lengths

What's the minimum total cost?

Example: Sticks [2, 4, 3]

One way: (2+3=5) + (5+4=9) = 14

Another: (2+4=6) + (6+3=9) = 15

Question: What greedy strategy minimizes cost?

Connecting Sticks - Hint

Think about:

Which sticks should you combine first?

What happens to combined sticks?

How many times does each stick get 'counted'?

Data structure hint: What if you needed to repeatedly find

the two smallest items then add the sum value dynamically?

This is PA4 - I won't give away the answer!

Common Greedy Patterns

Problem Type	Greedy Strategy	Example
Scheduling	Shortest first	Job scheduling
Intervals	Earliest finish	Activity selection
Combining	Smallest first	Connecting sticks
Ratios	Best ratio	Fractional knapsack
Deadlines	Earliest deadline	Task scheduling

Pattern recognition helps!

Proof Techniques Summary

Exchange Argument (most common):

1. Assume optimal solution differs from greedy
2. Find the difference
3. Show you can exchange to match greedy
4. Prove exchange doesn't worsen solution
5. Conclude greedy is optimal

PA4 Preview

Due: November 6 (same day as Exam 2)

Tasks:

1. Write informal proof of correctness (exchange argument!)
2. Implement greedy (and naive) algorithm for connecting sticks
3. Analyze time complexity (expected vs actual speedup)

You now have the tools you need!

PA4 Proof Expectations

Your proof should include:

- Clear statement of what you're proving
- Exchange argument or greedy-stays-ahead
- Explanation of why exchange doesn't worsen solution
- Logical conclusion

Quality over length - a clear, concise proof is better

than a rambling one