

Project Start + Heuristic Selection

CS/MCS 401: Computer Algorithms I

Lecture 17

Professor: Jonathan Liu

Icebreaker: We're supposed to have a milder winter this year. How do you feel about that?

Announcements

Homework 4 and worked examples have been released.

Everything on gradescope should be graded by the end of the day tomorrow. If you resubmit today, I will try to have it graded by end of day.

Tomorrow, I plan to make two logistical improvements to this course:

- Add all future assignments onto Canvas, so you have a calendar for everything
- Import gradescope grades onto Canvas, to help you see your current overall score in the course

Today's Overview

Today, we're going to talk more about the project problem, and use it as a backdrop to have some quick discussion about heuristic selection as well as the limits of greedy algorithms.

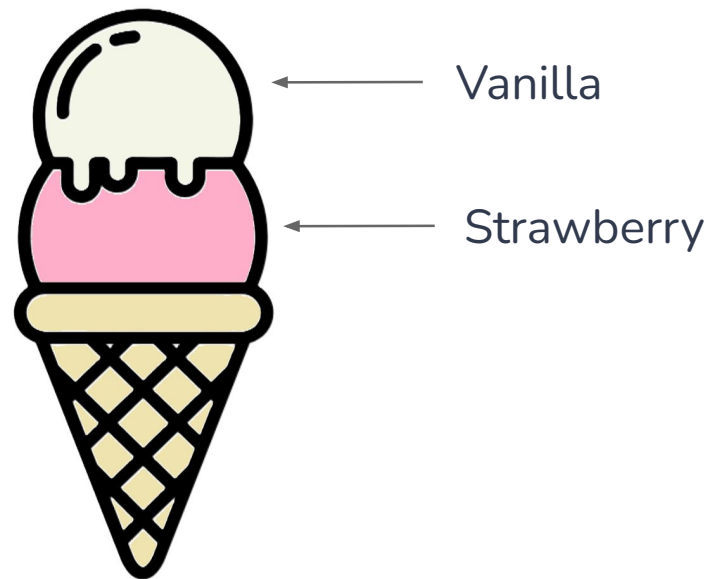
We're also going to talk in more detail about the project structure and requirements.

Project: Cone-flict of Interest



The Story

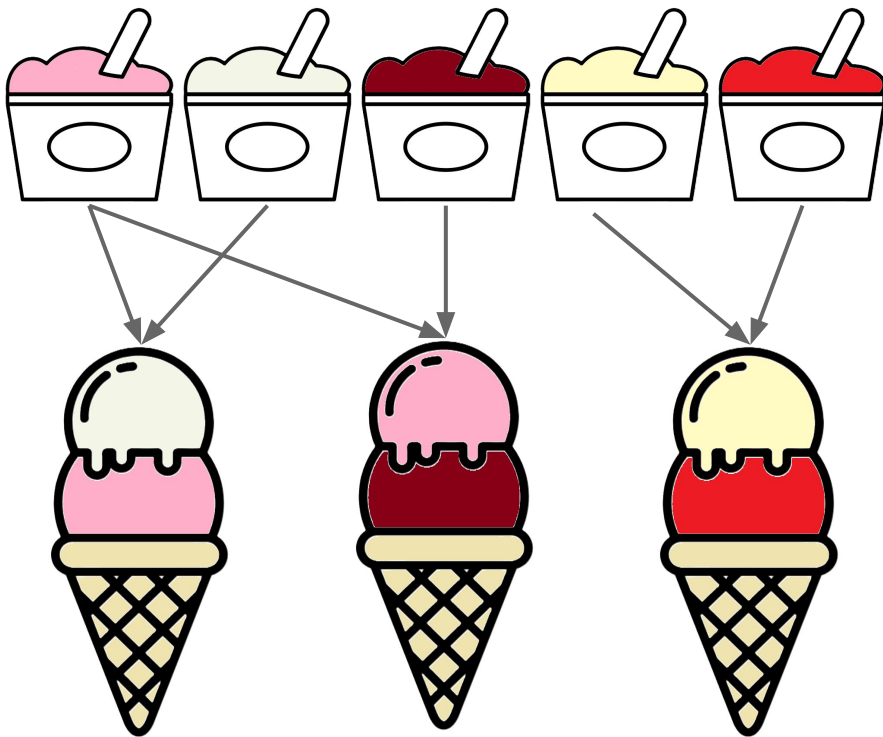
Your ice cream shop specializes in **flavor combination cones**: cones with two different flavors that go well together.



The Story

Your ice cream shop specializes in **flavor combination cones**: cones with two different flavors that go well together.

You have $2n$ flavors in your inventory, and you'd like to use them to offer a menu with as many **good** pairings as you can, without repeating any flavors.

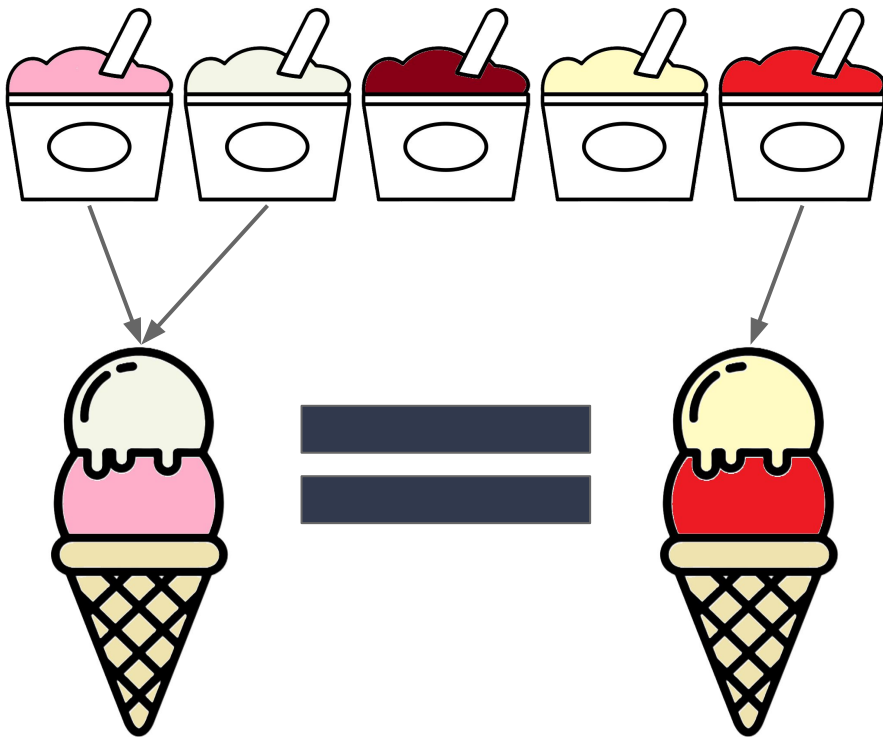


The Story

Your ice cream shop specializes in **flavor combination cones**: cones with two different flavors that go well together.

You have $2n$ flavors in your inventory, and you'd like to use them to offer a menu with as many **good** pairings as you can, without repeating any flavors.

However, some pairings are too similar to one another. Each similar pairing will get a “similarity penalty.”



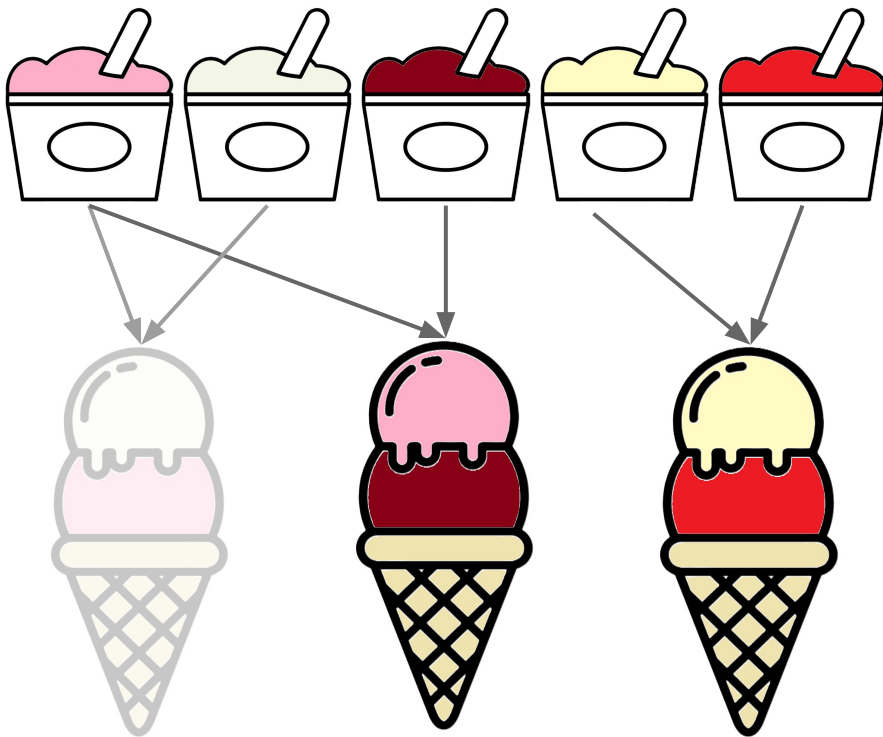
The Story

Your ice cream shop specializes in **flavor combination cones**: cones with two different flavors that go well together.

You have n flavors in your inventory, and you'd like to use them to offer a menu with as many **good** pairings as you can, without repeating any flavors.

However, some pairings are too similar to one another. Each similar pairing will get a “similarity penalty.”

Goal: Find the menu that offers as many flavors as possible, while trying to also minimize similarity between pairs.



What greedy heuristics should we try?

- Pick the cone with the fewest penalties first
 - Find the cone pairs that have the least penalties
 - Add the cone that gives you the most points (after considering similarity penalties)
 - Remove the pairs that have the highest penalty
 - Group up cones with the fewest connections to other cones
 - Take the highest pairs, choose one to pick and toss the other
 - Randomly pick!
-
- Fewest similar pairs: Pick the cone that has similarity with fewest other cones
 - Most similar pairs: Pick the cone that has similarity with most other cones
 - Rarest flavors: Pick the flavor that has the fewest potential cones
 - Cheapest Cone: Add the cone that gives you the most points (after considering similarity penalties)

Try it yourself!

<https://tinyurl.com/401project>

What should we take away from this?

Instance	Fewest Similar Pairs	Most Similar Pairs	Rarest Flavors First	Cheapest Cone
1	109	113	107	109
2	89	88	89	89
3	89	87	87	89
4	88	89	89	89
5	117	117	117	89

Project Details



Project Overview

Phase	Deliverable	Due	%
0	Team Selection	Tomorrow!	0
1	Inputs	10/30	10
2	Checkpoint	11/13	10
	Final Outputs	11/25	40
	Your Code + Replication Audit		
	Design Doc + Reflection		
3	Peer Presentation	12/3	15
	Professor Presentation	Weeks 15-16	25

Project Overview

Phase	Deliverable	Due	%
0	Team Selection	Tomorrow!	0
1	Inputs	10/30	10
2	Checkpoint	11/13	10
	Final Outputs	11/25	40
	Your Code + Replication Audit		
	Design Doc + Reflection		
3	Peer Presentation	12/3	15
	Professor Presentation	Weeks 15-16	25

You will submit a problem instance. You'll get to choose which pairs make valid cones and which cones are similar to each other.

You'll be limited to 250 flavors.

You'll get full credit for submitting a valid instance and a short justification.

For each team that you beat on your instance, you'll get 1% extra credit on the project.

Project Overview

Phase	Deliverable	Due	%
0	Team Selection	Tomorrow!	0
1	Inputs	10/30	10
2	Checkpoint	11/13	10
	Final Outputs	11/25	40
	Your Code + Replication Audit		
	Design Doc + Reflection		
3	Peer Presentation	12/3	15
	Professor Presentation	Weeks 15-16	25

Your ~20 inputs will be added to a collection of 45 inputs that I have created. These range from $n=10$ to $n=1000$.

At the end of the project, you'll need to submit a valid menu of cones for all 65 of these inputs.

There are three other shops you're competing with, and each has written their own algorithm to find menus. **You'll be scored based on how many you beat, with extra credit per input based on how much you beat them by.**

Project Overview

Phase	Deliverable	Due	%
0	Team Selection	Tomorrow!	0
1	Inputs	10/30	10
2	Checkpoint	11/13	10
	Final Outputs	11/25	40
	Your Code + Replication Audit		
	Design Doc + Reflection		
3	Peer Presentation	12/3	15
	Professor Presentation	Weeks 15-16	25

At any point during the project, if you submit a menu to Gradescope, it'll tell you how many points you've earned per input, how many shops you beat, and how many points you need to beat the next shop. You'll also see everyone else's scores per input, in case that's helpful information for you to find better inputs.

You'll also be given code that scores your menu for you to run locally or use in your own algorithms. This code matches the autograder's code.

Project Overview

Phase	Deliverable	Due	%
0	Team Selection	Tomorrow!	0
1	Inputs	10/30	10
2	Checkpoint	11/13	10
	Final Outputs	11/25	40
	Your Code + Replication Audit		
	Design Doc + Reflection		
3	Peer Presentation	12/3	15
	Professor Presentation	Weeks 15-16	25

You'll also need to submit the code that you ran to find these menus, and a description of how you ran the code to find them.

You may, for example, write two algorithms, compare the outputs, and submit the better menu. That's completely valid, but you'll need to describe that in your replication audit. If I can't replicate your menus with your code, you won't get the points.

You'll also write up a design document with what you did and some reflections on the project.

Project Overview

Phase	Deliverable	Due	%
0	Team Selection	Tomorrow!	0
1	Inputs	10/30	10
2	Checkpoint	11/13	10
	Final Outputs	11/25	40
	Your Code + Replication Audit		
	Design Doc + Reflection		
3	Peer Presentation	12/3	15
	Professor Presentation	Weeks 15-16	25

1.5 weeks before the deadline, there's going to be a checkpoint that's a lighter version of the final submission.

You'll earn full points if you:

- Submit a valid menu for each input.
- Submit a short design doc describing what you've tried so far and what you plan to do moving forward.

You'll also get feedback from me on what I think you should try!

Project Overview

Phase	Deliverable	Due	%
0	Team Selection	Tomorrow!	0
1	Inputs	10/30	10
2	Checkpoint	11/13	10
	Final Outputs	11/25	40
	Your Code + Replication Audit		
	Design Doc + Reflection		
3	Peer Presentation	12/3	15
	Professor Presentation	Weeks 15-16	25

On 12/3, lecture will consist of 6 presentation rounds: I'll pair you with another person/group, and you'll either present or be presented to (but not both!).

You'll present 3 times and listen 3 times. Each presentation is 10 minutes long, with a 5-7 minute prepared talk and the rest of the time you'll take questions. Make whatever materials will help you present, but you have to speak – no videos.

The listeners will also be graders. Rubric is in the project spec.

Project Overview

Phase	Deliverable	Due	%
0	Team Selection	Tomorrow!	0
1	Inputs	10/30	10
2	Checkpoint	11/13	10
	Final Outputs	11/25	40
	Your Code + Replication Audit		
	Design Doc + Reflection		
3	Peer Presentation	12/3	15
	Professor Presentation	Weeks 15-16	25

It's important to be able to communicate about the decisions you're making with your algorithms, especially in high-stakes settings. After all, your algorithm won't be perfect, but you have to be able to defend it.

You'll book a 20-minute appointment with me to present your work. You'll give the same 5-7 minute presentation, but I'll have read your code and design doc, so I'll be asking more specific questions about your algorithm and your process.

Project Overview

Phase	Deliverable	Due	%
0	Team Selection	Tomorrow!	0
1	Inputs	10/30	10
2	Checkpoint	11/13	10
	Final Outputs	11/25	40
	Your Code + Replication Audit		
	Design Doc + Reflection		
3	Peer Presentation	12/3	15
	Professor Presentation	Weeks 15-16	25

Overall Rules

Your algorithm does NOT need to be efficient. However, it does need to finish running before 11/25. **Starting early is really, really, really helpful.**

You may use AI to code your algorithm. However, I will be reading your code and asking you about it. So I think telling AI what you want and having it implement that is safe, but having it design the algorithm for you is likely not worth it.

You may not pay for anything, but you may use any service that is free to all UIC students.

Project Overview

Phase	Deliverable	Due	%
0	Team Selection	Tomorrow!	0
1	Inputs	10/30	10
2	Checkpoint	11/13	10
	Final Outputs	11/25	40
	Your Code + Replication Audit		
	Design Doc + Reflection		
3	Peer Presentation	12/3	15
	Professor Presentation	Weeks 15-16	25

Overall Rules

Deadlines will be enforced quite strictly. The gradescope assignments are all equipped with autograders, so you will know very quickly whether or not your submission is valid format. Submit early to make sure you don't run into this issue!

Lecture Recap

AI Statement

I use AI for very few things in this class, and I take immense pride in knowing that everything you see in this class (lectures, homeworks, solutions, feedback, etc.) is hand-written by me.

That said, the autograders for this project are AI-generated, as was the demonstration today. So were the ice cream cones.

Actual Recap

Today we saw that, when dealing with problems that have no efficient solution, good heuristics can often (but not always) get use close to good answers. We also learned about the project.