CS340 - Analysis of Algorithms

Introduction

Agenda

1. Course logistics

2. Stable Matching Problem

What is an algorithm?

An excavated triangular cistern has been found.

The width is a. The height is b.

Take the square of the width.

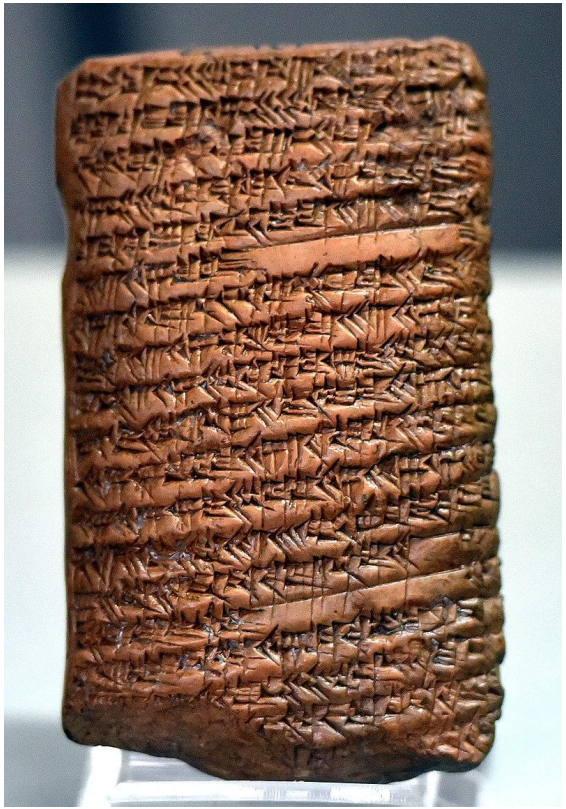
Take the square of the height.

Add the squared values.

Take the square root of the result.

This is the length of the diagonal.

This is the Procedure.



What is an algorithm?

- Any well-defined computational procedure that takes some values as input and produces some values as output.
- A step-by-step instruction for solving a computational problem
- Enough details so that a competent computer scientist can implement without questions

Can you name some algorithms you are already familiar with?

What you'll learn

1. How to analyze the run-time performance of algorithms
2. Write rigorous correctness proofs of algorithms
3. Demonstrate a familiarity with major algorithms and data structures
4. Apply important algorithmic design paradigms
5. Synthesize efficient algorithms in common situations

Why is learning algorithms valuable?

1. If we can identify broad categories of algorithms and solve them, then when we see those problems again, they're easy to solve.

2. If we encounter a new problem, it's probably similar to a solved one, or a few solved problems put together.

3. Tech interviews!

Course Logistics

Prerequisites

1. Basic programming
    a. pointers, structures/classes, recursion
2. Discrete Math
    a. induction, recurrence relations, counting, probability and graph theory
3. Understanding of basic data structures
    a. lists, stacks, queues, trees, graphs and heaps
4. Knowledge of basic sorting algorithms

Course Logistics

- Lectures: Monday and Wednesday
  - Required readings
- Labs: Thursdays 1:10-2:30pm Park 230
  - Graded on completeness
- Textbook: Algorithm Design by Jon Kleinberg and Eva Tardos
  - Really good textbook!
- Assignments:
  - Due Monday before class
  - Start early!
  - Expect 10 hours+ per week
  - Submit via gradescope
  - Learn Latex!

Logistics

- Course Website: bmc-cs-340.github.io
  - Assignments, labs, lecture slides, recordings
- Gradescope
  - Entry code: ZY6WN7
- Piazza
  - Discussion form
  - You should already be enrolled
  - https://piazza.com/brynmawr/fall2025/cmscb340
- Goldengate server
  - Needed for project
  - Email David Diaz ddiaz1@brynmawr.edu if you encounter issues

Grade Breakdown

- Homeworks: 15%
- Project: 15%
- Labs: 5%
- Midterm: 25%
- Final: 40%

Policies

- Come to class and actively participate
- No late work will be accepted
- Extensions should be requested at least 24hours ahead of deadline

Study Groups and Discussions

- Discussions are vital
- Study groups
  - Formation is required
- Collaboration policy:
  - Allowed sources:
    - Your classmates in this course! Work together, then write it up seperately
    - The textbook
    - The professor and TAs
    - Notes / content from previous courses (discrete math and data structures will be helpful)
  - Disallowed sources (for solution lookup):
    - The internet and AI tools
    - Students work from previous semesters
  - Cite anything that contributed to your solutions

Stable Matching Problem

Our first algorithm

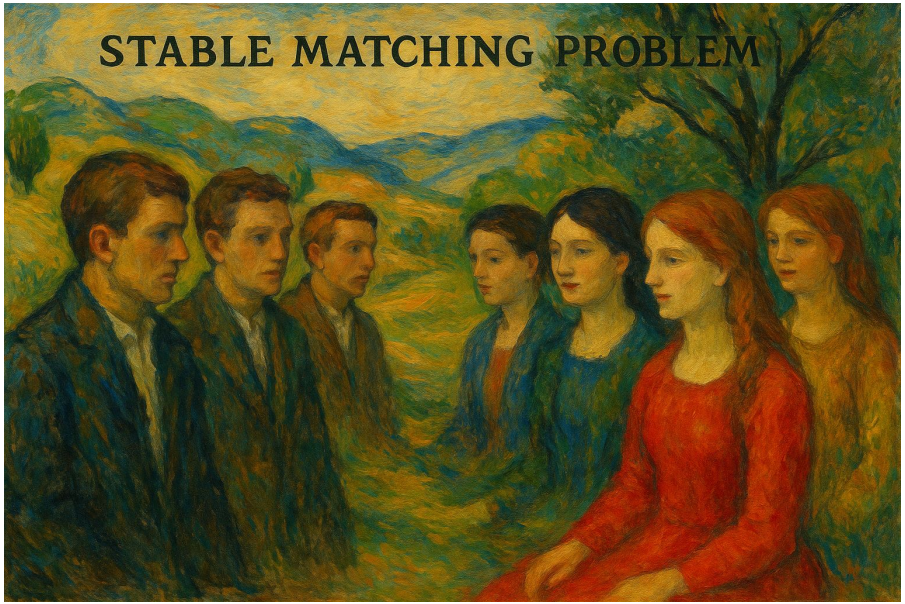
Stable Matching Problem

Could one design a college admissions process or a job recruiting process that is *self-enforcing*? (proposed in 1962)

Applicants and the Institutions (companies or colleges) have preference orderings.

Based on the companies preferences, they extend offers and applicants can choose which offer to accept.

How can this process go wrong?



Stable Matching Problem

What can go wrong?

- Sophia accepts a summer internship at Meta
- Isabella accepts an internship at Open AI
- A few days later, Perplexity AI calls her and offers her a position
  - She would rather work at Perplexity and retracts her position at Meta
- Meta now has a position open
  - They call their waitlisted applicant Isabella and she retracts her previous position at OpenAI
- OpenAI now has a position open
- ....

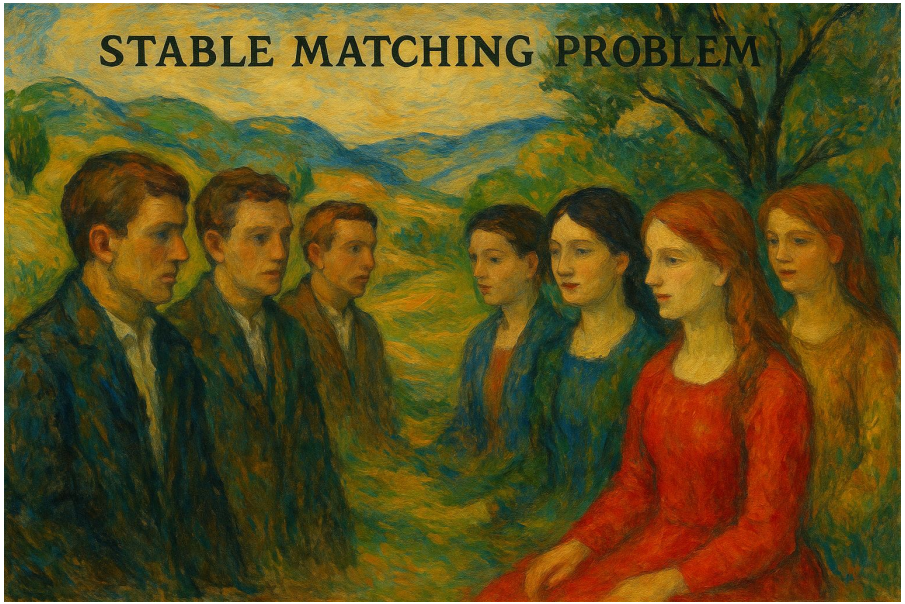
Stable Matching Problem

Could one design a college admissions process or a job recruiting process that is *self-enforcing*? (proposed in 1962)

Self enforcing: self interest prevents offers from being retracted and redirected

If any **student contacts a company**, they will say “no, I prefer my current intern”

If any **company contacts a student**, they will say “no I prefer my current internship”



Stable Matching Problem

Real world applications:

1. **Residency** - assignment of graduating medical students to their first hospital appointments
2. **Rabbi assignment** - assignment of rabbis who graduate from Hebrew Union College to Jewish congregations
3. **French University Applicants** - assignment of students to French Universities
4. **Panhellenic Sorority Rush** - assignment of women to sororities
5. ...

Stability Definition

For every employer E and every applicant A who is not assigned to intern at E,

at least one of the following should be true:

- E prefers its accepted intern to A or
- A prefers her current internship over working for employer E

Individual self interest will lead to stability

A Note on Definitions

- Good definitions are vital
  - The problems usually come with definitions. Use them, do not make up new ones unnecessarily
  - Learn to write definitions like the ones you were given

- Outline / formulate your inputs and outputs with the definitions

Employer-Applicant Matching

Input: preference lists for companies and preference lists for applicants

Output: matching of companies to applicants

Goal: output a matching between companies and applicants that is stable

Employers			Applicants		
Google(G)	Intel(I)	Apple(A)	Kate(K)	Clara(C)	Lisa(L)
K	K	L	A	A	G
C	L	K	G	I	I
L	C	C	I	G	A

Employer-Applicant Matching Example

Employers			Applicants		
Google(G)	Intel(I)	Apple(A)	Kate(K)	Clara(C)	Lisa(L)
K	K	L	A	A	G
C	L	K	G	I	I
L	C	C	I	G	A

Q: How many possible matchings are there?
A: 6 (n!)

Employer-Applicant Matching Example

Employers			Applicants		
Google(G)	Intel(I)	Apple(A)	Kate(K)	Clara(C)	Lisa(L)
K	K	L	A	A	G
C	L	K	G	I	I
L	C	C	I	G	A

Matching 1: (G, C), (I, L), (A, K)

Q:Is this stable?
A: Yes

Employer-Applicant Matching Example

Employers			Applicants		
Google(G)	Intel(I)	Apple(A)	Kate(K)	Clara(C)	Lisa(L)
K	K	L	A	A	G
C	L	K	G	I	I
L	C	C	I	G	A

Matching 2: (G, L), (I, K), (A, C)

Q:Is this stable?
A: No

Employer-Applicant Matching Example

Employers			Applicants		
Google(G)	Intel(I)	Apple(A)	Kate(K)	Clara(C)	Lisa(L)
K	K	L	A	A	G
C	L	K	G	I	I
L	C	C	I	G	A

Matching 3: (G, L), (I, C), (A, K)

Q:Is this stable?
A: Yes

An algorithm for deciding if a matching is stable

We just did one by hand... can you formalize it? (Lab tomorrow)

Even if you have a pretty good idea on how an algorithm is supposed to work, writing it up cleanly and succinctly is still not an easy task!

Steps to finding an algorithm

1. Construct a good example (not too small).
2. Solve the example and check your solution (by hand, without worrying about the algorithm)
3. Think about how you solved and/or checked the problem and how you can use that to solve a general instance.
4. Formalize an algorithm (might have to formalize the problem too)
5. Construct a new and somehow different example and run your algorithm on it and check the solution.
6. Repeat until you are confident it works.

More Definitions

Matching: Given a pair of sets X and Y, a matching is a collection of pairs (x, y) where $x \in X$ and $y \in Y$, and each element of X and Y appears in at most one pair

Are the following matchings?

1. (Google,Clara), (Apple,Lisa), (Intel,Kate)
2. (Google,Clara), (Google,Lisa), (Google, Kate)
3. (Intel,Kate)
4. (Apple,Kate), (Intel,Kate), (Google, Lisa)
5. (Google,Clara), (Apple,Lisa)

More Definitions

Perfect Matching: A matching is perfect if every element of X and Y occurs in some pair

Note: Perfect != Stable

Are the following perfect matchings?

1. (Google,Clara), (Apple,Lisa), (Intel,Kate)
2. (Google,Clara), (Google,Lisa), (Google, Kate)
3. (Intel,Kate)
4. (Apple,Kate), (Intel,Kate), (Google, Lisa)
5. (Google,Clara), (Apple,Lisa)

More Definitions

Stable Matching: Given sets X and Y of equal size and a preference ordering for each element of each set, a perfect matching is stable if there is no pair (x, y) that is *not* in the matching where x prefers y to its current match and y prefers x to its current match

Summary

1. Logistics
    a. Gradescope
    b. Piazza
    c. Goldengate account

2. Office hours poll on piazza

3. Start the reading before next class

4. Lab tomorrow: writing up an algorithm to decide if a matching is stable
    a. Due before next lab (Sep 11)

5. HW1 due Sep 15th (next next Monday)