CS320 Algorithms: Theory and Practice
Fall 2022
(based on original slides by Wim Böhm)

Course Introduction

"For me, great algorithms are the poetry of computation. Just like verse, they can be terse, allusive, dense, and even mysterious. But once unlocked, they cast a brilliant new light on some aspect of computing." - *Francis Sullivan*

Welcome back!!

- We hope you are all alright! Welcome back to school
- This class is hybrid in-person and on-line. Lectures are live captured and available in Canvas via echo360.
  - Students in the online section will do everything remotely.
  - Students in the on-campus section will do many things remotely
    - (canvas quizzes, worksheets, programming assignments, etc.) but
    - will have proctored exams (in person)
    - And may optionally watch lecture remotely or in person
- TAs will do help desk and office hours in person (in CSB 130) and Teams.
- If you have issues (illness, uncertainties, timing, anything really) please don't hesitate to let me (Sanjay Rajopadhye) know through e-mail or office hours.
- This is a 3 credit course with no recitations. TAs will help you with quizzes and assignments using helpdesk. I will have office hours in person and on Teams.

Course Objectives

Algorithms:

- Design – strategies for algorithmic problem solving
- Reasoning about algorithm **correctness**
- Analysis of **time** and **space complexity**
- Implementation – create an implementation **that respects the runtime analysis.** In this class a program has to be correct and has to have the optimal complexity

Algorithmic Approaches / Classes:

- Greedy
- Divide and Conquer
- Dynamic programming

Parallel Algorithms:

- Dynamic Multi-threading

Problem Classes:

- Reduction, P, NP, NPC

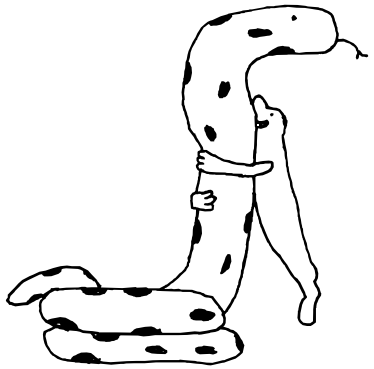
Grading (tentative)

- Prerequisites (quizzes+exam) 10%
- Programming Assignments 15%
- Worksheets 10%
- Quizzes 15%
- Exams 50%

See CS320 web site:

https://www.cs.colostate.edu/~cs320

Implementation



Programs will be written in Python:

- Powerful **data structures**
  - tuples, dictionaries, (array) lists
- Simple, easy to learn syntax
- Highly readable, compact code
- An extensive standard library
- Strong support for integration with other languages (C, C++, Java) and libraries (numpy, jupyter, CUDA)

We assume you are familiar with Python (CS220)!

Python vs. e.g. Java

What makes Python different from Java?

- Java is statically typed, i.e. variables are bound to types at compile time. This avoids run time errors, but makes java programs more rigid.
- Python is dynamically typed, i.e. a variable takes on some type at run time, and its type can change. A variable can be of one type somewhere in the code and of another type somewhere else

```
f = open(filename)
for line in f:
    # line is a String here, split it using " " as delimiter
    line = line.strip().split(" ")
    # line is an (Array)List of Strings here
```

- This makes python programs more flexible, but can cause strange run time errors, e.g. when a caller expects a return value but the called function does not return one.

Our approach to problem solving

- Formulate it with precision (usually using mathematical concepts, such as sets, relations, and graphs)
- Design an algorithm and its main data structures
- Prove its correctness
- Analyze its complexity (time, space)
  - Improve the initial algorithm (in terms of complexity), preserving correctness

- Implement it, **preserving the analyzed complexity!**
  In the lab PAs we will test for that. So in this course we check for **correctness and complexity** of your PAs.

Our first problem: matching

Two parties e.g., companies and applicants

- Each applicant has a preference list of companies
- Each company has a preference list of applicants
- A possible scenario:

  cA offers job to aA

  aA accepts, but now gets offer from cX

  aA likes cX more, retracts offer from cA

We would like a systematic method for assigning applicants to companies– stable matching

- A system like this is e.g. in use for matching medical residents with hospitals

Stable Matching

Goal. Given a set of preferences among companies and applicants, design a **stable matching algorithm**.

Unstable pair: applicant x and company y are an unstable pair **(not in the current matching)** if:

- **Both** x prefers y to its assigned company
- **And** y prefers x to one of its selected applicants.

Stable assignment. Assignment without unstable pairs.

- Natural and desirable condition.

Is some control possible?

Given the preference lists of applicants *A* and companies *C*, can we assign *A*s to *C*s such that

for each *C*
  for each *A* **not** scheduled to work for *C*
    either *C* prefers all its students to *A*
      or *A* prefers current company to *C*

Why **or, and not and.**
If this holds, then what?

Stable state

Given the preference lists of applicants *A* and companies *C*, can we assign *A*s to *C*s such that

for each *C*
  for each *A* not scheduled to work for *C*
    C prefers all its students to *A*
    or *A* prefers current company to *C*

or: Morgan's law not(*A* and B) = not *A* or not B

If this holds, there is no unstable pair, and therefore individual self interest will prevent changes in student / company matches: **Stable state**

Simplifying the problem

Matching students/companies problem messy:

- Company may look for multiple applicants, students looking for a single internship
- Maybe there are more jobs than applicants, or fewer jobs than applicants
- Maybe some applicants/jobs are equally liked by companies/applicants (partial orders)

Formulate a "bare-bones" version of the problem: match n men and n women

Stable Matching Problem: n women and n men

Perfect matching: Each man matched with exactly one woman, and each woman matched with exactly one man.

Stability: no incentive for some pair to undermine the assignment.

- A pair (m,w) NOT IN THE CURRENT MATCHING is an **instability** if BOTH **m** and **w** prefer each other to current partners in the matching, i.e.:

  BOTH m and w can improve their situation

Stable matching: perfect (i.e., complete) matching with no unstable pairs. Stable matching problem (Gale, Shapley 1962): Given the preference lists of n men and n women, find a stable matching if one exists.

The Stable Matching Problem

Problem: Given n men and n women where

- Each man lists women in **total order** of preference
- Each woman lists men in **total order** of preference
  - A total order (remember CS220?) allows the elements of the set to be linearly ordered.

favorite ↓ least favorite ↓

	1st	2nd	3rd
Xavier	Amy	Bertha	Clare
Yancey	Bertha	Amy	Clare
Zeus	Amy	Bertha	Clare

Men's Preference Profile

favorite ↓ least favorite ↓

	1st	2nd	3rd
Amy	Yancey	Xavier	Zeus
Bertha	Xavier	Yancey	Zeus
Clare	Xavier	Yancey	Zeus

Women's Preference Profile

find a stable matching of all men and women

Do it, Do it

favorite ↓ least favorite ↓

	1st	2nd	3rd
X	A	B	C
Y	B	A	C
Z	A	B	C

Men's Preference Profile

favorite ↓ least favorite ↓

	1st	2nd	3rd
A	Y	X	Z
B	X	Y	Z
C	X	Y	Z

Women's Preference Profile

Create all possible perfect matchings and check (in)stability

{ (X,A), (Y,B), (Z,C) } Stable (neither Z nor C can improve)

{ (X,A), (Y,C), (Z,B) } Instability: (Y,B) Y prefers B and B prefers Y

{ (X,B), (Y,A), (Z,C) } Stable

{ (X,B), (Y,C), (Z,A) } Instability: (X,A)

{ (X,C), (Y,A), (Z,B) } Instability: (X,B)

{ (X,C), (Y,B), (Z,A) } Instability: (X,A)

Formulation

Men: $M=\{m_1, \ldots, m_n\}$ Women: $W=\{w_1, \ldots, w_n\}$

The Cartesian Product MxW is the set of all possible ordered pairs.

A matching S is a set of pairs (subset of MxW) such that each m and w occurs in at most one pair

A perfect (complete) matching S is a set of pairs (subset of MxW) such that each individual occurs in exactly one pair

How many perfect matchings are there?

n	n-1	n-2		1
m_1	m_2	m_3	...	m_n

Instability

Given a perfect match, e.g.,

$S = \{ (m_1,w_1),\ (m_2,w_2) \}$

But m_1 prefers w_2 and w_2 prefers m_1
(m_1,w_2) is an instability for S

(notice again that (m_1,w_2) **is not in S**)

S is a stable matching if:

- S is perfect
- and there is no instability in S

Example 1

m_1: w_1, w_2 $\quad$ m_2: w_1, w_2

w_1: m_1, m_2 $\quad$ w_2: m_1, m_2

What are the perfect matchings?

Example 1

m_1: w_1, w_2 $\quad$ m_2: w_1, w_2

w_1: m_1, m_2 $\quad$ w_2: m_1, m_2

1. $\{ (m_1,w_1), (m_2,w_2) \}$
2. $\{ (m_1,w_2), (m_2,w_1) \}$

which is stable/instable?

Example 1

m_1: w_1, w_2 $\quad$ m_2: w_1, w_2

w_1: m_1, m_2 $\quad$ w_2: m_1, m_2

1. { (m_1,w_1), (m_2,w_2) } stable, WHY?

w2 prefers m1, but m1 prefers w1,
m2 prefers w1, but w1 prefers m1

2. { (m_1,w_2), (m_2,w_1) } instable, WHY?

(m1,w1)

Example 2

m_1: w_1, w_2 m_2: w_2, w_1

w_1: m_2, m_1 w_2: m_1, m_2

1. $\{ (m_1,w_1), (m_2,w_2) \}$
2. $\{ (m_1,w_2), (m_2,w_1) \}$

which is / are instable/stable?

both are stable!

1: w1 prefers m2 but m2 prefers w2, w2 prefers m1 but m1 prefers w1

2: m1 prefers w1 but w1 prefers m2, m2 prefers w2 but w2 prefers m1

Conclusion?

Sometimes there is more than 1 stable matching

Example 3

m_1: w_1, w_2, w_3 m_2: w_2, w_3, w_1 m_3: w_3, w_1, w_2

w_1: m_2, m_1, m_3 w_2: m_1, m_2, m_3 w_3: m_1, m_2, m_3

Is $\{ (m_1,w_1), (m_2,w_2), (m_3,w_3) \}$ stable?

Is $\{ (m_1,w_2), (m_2,w_1), (m_3,w_3) \}$ stable?

Do this one yourself.

Questions…

- Given a preference list, does a stable matching exist?
- Can we efficiently construct a stable matching if there is one?
- a naive algorithm:

```
for S in the set of all perfect matchings :
    if S is stable :  return S
return None
```

Is this algorithm correct?

What is its running time?

Towards an efficient algorithm

initially: no match

An unmatched man m proposes to the woman w highest on his list.

Will this be part of a stable matching?

Towards an efficient algorithm

initially: no match

An unmatched man m proposes to the woman w highest on his list.

Will this be part of a stable matching?

Not necessarily: w may like some m' better, AND?

m' likes w the most

So w and m will be in a temporary state of engagement.

w is prepared to change her mind when a man higher on her list proposes.

While not everyone is matched…

An unmatched man m proposes to the woman w highest on his list to whom he hasn't yet proposed.

Why is that important?

Termination

If w is free, they become engaged

If w is engaged to m':

If w prefers m' over m, w stays with m' and m stays free

If w prefers m over m', (m,w) become engaged and m' becomes free