

Lecture01

CSE 417 Algorithms and Computational Complexity

Richard Anderson
Winter 2023
Lecture 1

CSE 417 Course Introduction

- CSE 417, Algorithms and Computational Complexity
 - MWF 10:30-11:20 AM
 - CSE2 G10
- **Instructor**
 - Richard Anderson, anderson@cs.washington.edu
 - Office hours:
 - Office hours: TBD
- **Teaching Assistants**
 - Nickolay Perezhogin, Artin Tajdini, Tom Zhaoyang Tian, Michael Wen, Albert Weng, Yilin Zhang

Announcements

- **It's on the course website**
- **Homework weekly**
 - Usually due Fridays
 - HW 1, Due Friday, January 13.
 - It's on the website (or will be soon)
- **Homework is to be submitted electronically**
 - Due at 11:59 pm, Fridays. Five late days.
- **You should be on the course mailing list**
 - But it will probably go to your uw.edu account

Textbook

- Algorithm Design
- Jon Kleinberg, Eva Tardos
 - Only one edition
- Read Chapters 1 & 2
- Expected coverage:
 - Chapter 1 through 7
- Book available at:
 - UW Bookstore (\$191.00/\$74.99)
 - Ebay (\$16.87 to \$582.30)
 - Amazon (\$154.66/\$33.28)
 - Electronic (\$74.99)
 - PDF



Course Mechanics

- Homework
 - Due Fridays
 - Mix of written problems and programming
 - Target: 1-week turnaround on grading
- Exams
 - Midterm, Tentatively, Wednesday, February 8
 - Final, Monday, March 13, 8:30-10:20 AM
 - Approximate grade weighting:
 - HW: 50, MT: 15, Final: 35
- Course web
 - Slides, Handouts, Discussion Board
- Canvas
 - Panopto videos

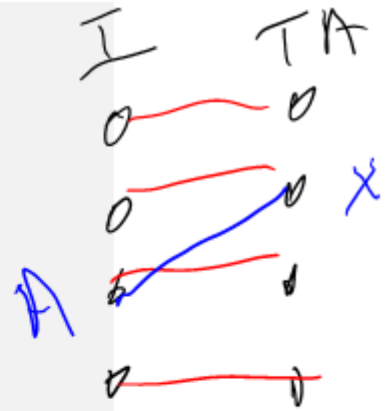
All of Computer Science is the Study of Algorithms

How to study algorithms

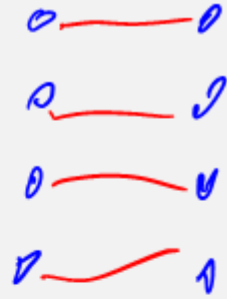
- Zoology
- Mine is faster than yours is
- Algorithmic ideas
 - Where algorithms apply
 - What makes an algorithm work
 - Algorithmic thinking
- Algorithm practice

Introductory Problem: Stable Matching

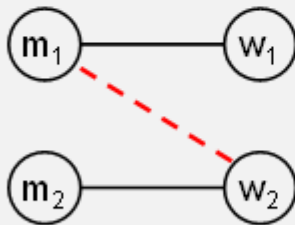
- Setting:
 - Assign TAs to Instructors
 - Avoid having TAs and Instructors wanting changes
 - E.g., Prof A. would rather have student X than her current TA, and student X would rather work for Prof A. than his current instructor.



Formal notions



- Perfect matching
- Ranked preference lists
- Stability $m_1 \succ w_2 \succ w_1$



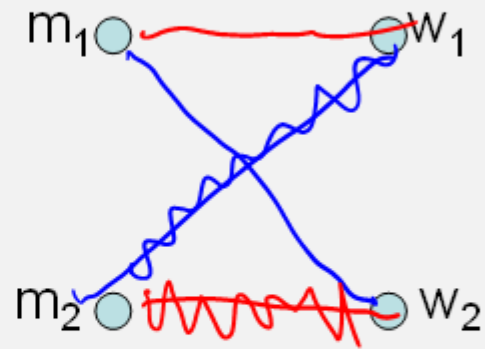
Example (1 of 3)

$m_1: w_1 w_2$

$m_2: w_2 w_1$

$w_1: m_1 m_2$

$w_2: m_2 m_1$



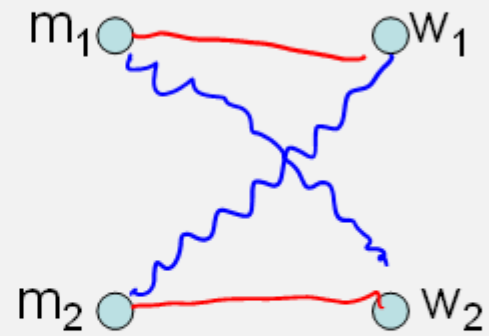
Example (2 of 3)

$m_1: w_1 w_2$

$m_2: w_1 w_2$

$w_1: m_1 m_2$

$w_2: m_1 m_2$



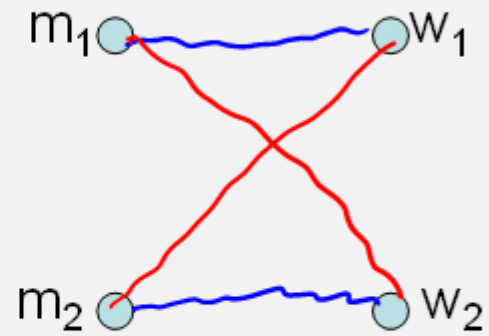
Example (3 of 3)

$m_1: w_1 w_2$

$m_2: w_2 w_1$

$w_1: m_2 m_1$

$w_2: m_1 m_2$



Formal Problem

- **Input**

- Preference lists for m_1, m_2, \dots, m_n
- Preference lists for w_1, w_2, \dots, w_n

- **Output**

- Perfect matching M satisfying stability property:

If $(m', w') \in M$ and $(m'', w'') \in M$ then
(m' prefers w' to w'') or (w'' prefers m'' to m')

Gale, Shapley 1956

Idea for an Algorithm

m proposes to w

If w is unmatched, w accepts

If w is matched to m_2

If w prefers m to m_2 w accepts m , dumping m_2

If w prefers m_2 to m , w rejects m

Unmatched m proposes to the highest w on its preference list **that it has not already proposed to**

m — w

m — w

m_2 — w

m — w

m_2 — w

Algorithm

Initially all m in M and w in W are free

While there is a free m

w highest on m 's list that m has not proposed to

 if w is free, then match (m, w)

 else

 suppose (m_2, w) is matched

 if w prefers m to m_2

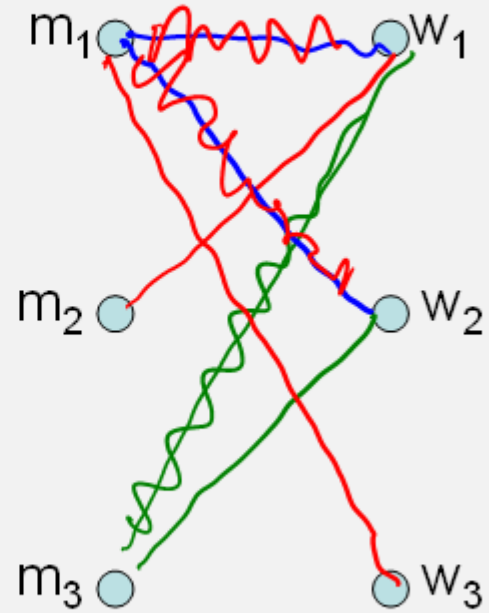
 unmatch (m_2, w)

 match (m, w)

\emptyset \emptyset
 \emptyset \emptyset
 \emptyset \emptyset
 \emptyset \emptyset
 M W

$m_1 \quad m_2 \quad m_3 \quad m_1 \quad m_2 \quad m_1$

Example

 $m_1: w_1 w_2 w_3$
 $m_2: w_1 w_3 w_2$
 $m_3: w_1 w_2 w_3$
 $w_1: m_2 m_3 m_1$
 $w_2: m_3 m_1 m_2$
 $w_3: m_3 m_1 m_2$


Does this work?

- Does it terminate?
- Is the result a stable matching?
- **Begin by identifying invariants and measures of progress**
 - m's proposals get worse (have higher m-rank)
 - Once w is matched, w stays matched
 - w's partners get better (have lower w-rank)

Claim: If an m reaches the end of its list, then all the w 's are matched

Claim: The algorithm stops in at most n^2 steps

When the algorithm halts, every w
is matched

Why?

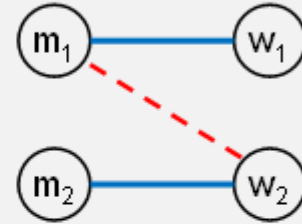
Hence, the algorithm finds a perfect
matching

The resulting matching is stable

Suppose

$(m_1, w_1) \in M, (m_2, w_2) \in M$

m_1 prefers w_2 to w_1



How could this happen?

Result

- Simple, $O(n^2)$ algorithm to compute a stable matching
- Corollary
 - A stable matching always exists