# CSE 421
# Introduction to Algorithms

Richard Anderson

Winter 2024

Lecture 1



# CSE 421 Course Introduction

- CSE 421, Introductions to Algorithms
  - MWF 1:30-2:20 PM, CSE2 G10
  - Thursday Section
- Instructor
  - Richard Anderson, anderson@cs.washington.edu
  - Office hours:
    - Office hours: TBD, CSE2 344
- Teaching Assistants
  - Raymond Gao, Sophie Robertson, Aman Thukral, Kaiyuan Liu, Albert Weng, Tom Zhaoyang Tian



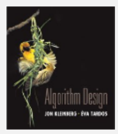
# Announcements

- It's on the course website
  - https://courses.cs.washington.edu/courses/cse421/24wi/
- Homework weekly
  - Due Wednesdays
  - HW 1, Due Wednesday, January 10, 2024.
  - It's on the website
- Homework is to be submitted electronically
  - Due at 11:59 pm, Fridays. Five late days.
- Edstem Discussion Board
- Panopto Videos



# 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 ($197.50/$79.99)
  - Ebay ($11.27 to $192.70)
  - Amazon ($156.95/$28.76)
  - Electronic ($10.99 per month)
  - PDF




# Course Mechanics

- Homework
  - Due Wednesdays
  - Mix of written problems and programming
  - Target: 1-week turnaround on grading
- Exams
  - Midterm, Friday, February 9
  - Final, Monday, March 11, 2:30-4:20 PM
  - Approximate grade weighting:
    - HW: 50, MT: 15, Final: 35
- Course web
  - Slides, Handouts, Discussion Board
- Canvas
  - Panopto videos
- Section on Thursdays
  - Recent addition for CSE421



# 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



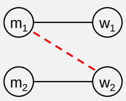
## Introductory Problem: Stable Matching

- 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

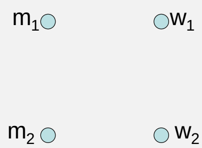
- Perfect matching
- Ranked preference lists
- Stability






## Example (1 of 3)

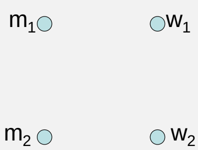
$m_1$: $w_1$ $w_2$
$m_2$: $w_2$ $w_1$
$w_1$: $m_1$ $m_2$
$w_2$: $m_2$ $m_1$






## Example (2 of 3)

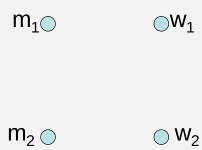
$m_1$: $w_1$ $w_2$
$m_2$: $w_1$ $w_2$
$w_1$: $m_1$ $m_2$
$w_2$: $m_1$ $m_2$






## 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 $m_1, m_2, \ldots, m_n$
  - Preference lists for $w_1, w_2, \ldots, w_n$
- 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 $m'$)



## 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



## Algorithm

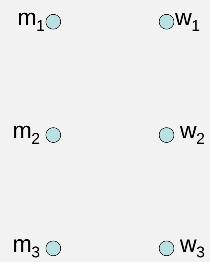
```
Initially all m in M and w in W are free
While there is a free m
        w highest on m's list that m has not proposed to
        if w is free, then match (m, w)
        else
                suppose (m2, w) is matched
                if w prefers m to m2
                        unmatch (m2, w)
                        match (m, w)
```



## 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 algorithms halts, every w is matched

Why?

Hence, the algorithm finds a perfect matching

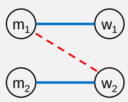


## The resulting matching is stable

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