CSE 417
Algorithms and Computational Complexity

Richard Anderson
Winter 2020
Lecture 2
Announcements

• Course website
  – https://courses.cs.washington.edu/courses/cse417/20wi/

• Homework due Wednesdays (strict)
  – HW 1, Due Wednesday, January 15, 9:29 AM.
  – Submit solutions on canvas

• You should be on the course mailing list
  – But it will probably go to your uw.edu account
Course Mechanics

• Homework
  – Due Wednesdays
  – About 5 problems, sometimes programming
  – Programming – your choice of language
  – Target: 1 week turnaround on grading

• Exams (In class)
  – Midterm, TBD
  – Final, Wednesday, March 18, 8:30-10:20 am

• Approximate grade weighting
  – HW: 50, MT: 15, Final: 35

• Course web
  – Slides, Handouts

• Instructor Office hours (CSE2 344):
  – Monday 2:30-3:30, Wednesday 2:30-3:30
TA Office Hours

Yuqing Ai, Tuesday, 3:00-4:00, CSE2 131
Alex Fang, Thursday, 1:30-2:30, CSE2 151
Anny Kong, Monday, 3:30-4:30, TBA
Zhichao Lei, Monday, 4:30-5:30, CSE1 007
Ansh Nagda, Tuesday, 11:30-12:30, CSE2 152
Chris Nie, Friday, 3:30-4:30, CSE2 121
Stable Matching: 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 (e.g., no instabilities):

For all $m’, m”’, w’, w”’$

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 m₂
    If w prefers m to m₂, w accepts m, dumping m₂
    If w prefers m₂ 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

\[ m_1: w_1 \ w_2 \ w_3 \]
\[ m_2: w_1 \ w_3 \ w_2 \]
\[ m_3: w_1 \ w_2 \ w_3 \]
\[ w_1: m_2 \ m_3 \ m_1 \]
\[ w_2: m_3 \ m_1 \ m_2 \]
\[ w_3: m_3 \ m_1 \ m_2 \]

Order: \( m_1, \ m_2, \ m_3, \ m_1, \ m_3, \ m_1 \)
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

Hence, the algorithm finds a perfect matching
The resulting matching is stable

Suppose

\[(m_1, w_1) \in M, (m_2, w_2) \in M\]
\[m_1 \text{ prefers } w_2 \text{ to } w_1\]

How could this happen?
Result

• Simple, $O(n^2)$ algorithm to compute a stable matching

• Corollary
  – A stable matching always exists
A closer look

Stable matchings are not necessarily fair

\[ m_1: \ w_1 \ w_2 \ w_3 \]
\[ m_2: \ w_2 \ w_3 \ w_1 \]
\[ m_3: \ w_3 \ w_1 \ w_2 \]
\[ w_1: \ m_2 \ m_3 \ m_1 \]
\[ w_2: \ m_3 \ m_1 \ m_2 \]
\[ w_3: \ m_1 \ m_2 \ m_3 \]

How many stable matchings can you find?
Algorithm under specified

• Many different ways of picking m’s to propose
• Surprising result
  – All orderings of picking free m’s give the same result

• Proving this type of result
  – Reordering argument
  – Prove algorithm is computing something mores specific
    • Show property of the solution – so it computes a specific stable matching
M-rank and W-rank of matching

- **m-rank**: position of matching w in preference list
- **M-rank**: sum of m-ranks
- **w-rank**: position of matching m in preference list
- **W-rank**: sum of w-ranks

What is the M-rank?

What is the W-rank?
Suppose there are $n$ m’s, and $n$ w’s

- What is the minimum possible M-rank?

- What is the maximum possible M-rank?

- Suppose each m is matched with a random w, what is the expected M-rank?
Random Preferences

Suppose that the preferences are completely random

\[ m_1: w_8, w_3, w_1, w_5, w_9, w_2, w_4, w_6, w_7, w_{10} \]
\[ m_2: w_7, w_{10}, w_1, w_9, w_3, w_4, w_8, w_2, w_5, w_6 \]
\[ \ldots \]
\[ w_1: m_1, m_4, m_9, m_5, m_{10}, m_3, m_2, m_6, m_8, m_7 \]
\[ w_2: m_5, m_8, m_1, m_3, m_2, m_7, m_9, m_{10}, m_4, m_6 \]
\[ \ldots \]

If there are \( n \) m’s and \( n \) w’s, what is the expected value of the M-rank and the W-rank when the proposal algorithm computes a stable matching?
Stable Matching Algorithms

• M Proposal Algorithm
  – Iterate over all m’s until all are matched

• W Proposal Algorithm
  – Change the role of m’s and w’s
  – Iterate over all w’s until all are matched
Generating a random permutation

```csharp
public static int[] Permutation(int n, Random rand) {
    int[] arr = IdentityPermutation(n);

    for (int i = 1; i < n; i++) {
        int j = rand.Next(0, i + 1);
        int temp = arr[i];
        arr[i] = arr[j];
        arr[j] = temp;
    }
    return arr;
}
```
What is the run time of the Stable Matching Algorithm?

Initially all $m$ in $M$ and $w$ in $W$ are free
While there is a free $m$      \hspace{2cm} \text{Executed at most } n^2 \text{ times}
    \begin{align*}
    &w \text{ highest on } m\text{'s list that } m \text{ has not proposed to} \\
    &\text{if } w \text{ is free, then match } (m, w) \\
    &\text{else} \\
    &\text{suppose } (m_2, w) \text{ is matched} \\
    &\text{if } w \text{ prefers } m \text{ to } m_2 \\
    &\text{unmatch } (m_2, w) \\
    &\text{match } (m, w)
    \end{align*}
O(1) time per iteration

- Find free m
- Find next available w
- If w is matched, determine m_2
- Test if w prefer m to m_2
- Update matching
What does it mean for an algorithm to be efficient?
Key ideas

• Formalizing real world problem
  – Model: graph and preference lists
  – Mechanism: stability condition

• Specification of algorithm with a natural operation
  – Proposal

• Establishing termination of process through invariants and progress measure

• Under specification of algorithm

• Establishing uniqueness of solution