Announcements

• Homework 1, due Wednesday Oct 5
  – in class, paper turn in
  – pay attention to making explanations clear and understandable

• Reading
  – Chapter 1, Sections 2.1, 2.2
Office Hours

• Richard Anderson
  – Monday, 2:30 pm - 3:30 pm, CSE 582
  – Wednesday, 2:30 pm - 3:30 pm, CSE 582
• Deepali Aneja
  – Monday, 5:30 pm - 6:30 pm, CSE 220
• Max Horton
  – Monday, 4:30 pm – 5:30 pm, CSE 220
  – Tuesday, 2:00 pm – 3:00 pm, CSE 218
• Ben Jones
  – Tuesday, 1:00 pm – 2:00 pm, CSE 218
  – Friday, 2:30 pm – 3:30 pm, CSE 220
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

\[(m_1, w_1) \in M, (m_2, w_2) \in M\]

\[m_1\] 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
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

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

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?
Best choices for one side may be bad for the other

Design a configuration for problem of size 4:

M proposal algorithm:
All m’s get first choice, all w’s get last choice

W proposal algorithm:
All w’s get first choice, all m’s get last choice
But there is a stable second choice

Design a configuration for problem of size 4:

M proposal algorithm:
All m’s get first choice, all w’s get last choice

W proposal algorithm:
All w’s get first choice, all m’s get last choice

There is a stable matching where everyone gets their second choice
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 $m$
    $w$ 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 $(m, w)$

Executed at most $n^2$ times
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