CSE 421
Algorithms

Richard Anderson

Autumn 2006

Lecture 2

Announcements

- It's on the web.
- Homework 1, Due October 4
  - It's on the web
- Subscribe to the mailing list

- Richard's office hours:
  - Tuesday, 2:30-3:20 pm, Friday, 2:30-3:20 pm.
- Ning's office hours:
  - Monday, 12:30-1:20 pm, Tuesday, 4:30-5:20 pm.

A closer look

- Stable matchings are not necessarily fair

m_1: w_1 w_2 w_3

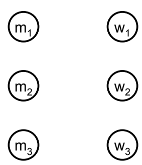
m_2: w_2 w_3 w_1

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

Proposal Algorithm finds the best possible solution for M

- Formalize the notion of best possible solution

- (m, w) is valid if (m, w) is in some stable matching
- best(m): the highest ranked w for m such that (m, w) is valid
- S* = {(m, best(m)}
- Every execution of the proposal algorithm computes S*

Proof

- See the text book – pages 9 – 12

- Related result: Proposal algorithm is the worst case for W
- Algorithm is the M-optimal algorithm
- Proposal algorithms where w's propose is W-Optimal

Best choices for one side are 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

m_1:

m_2:

m_3:

m_4:

w_1:

w_2:

w_3:

w_4:

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

m_1:

m_2:

m_3:

m_4:

w_1:

w_2:

w_3:

w_4:

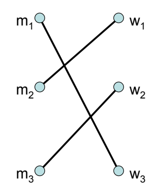
Key ideas

- 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

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

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



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
…
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
…

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?

Expected Ranks

- Expected M rank
- Expected W rank

Guess – as a function of n

Expected M rank

- Expected M rank is the number of steps until all M's are matched
  - (Also is the expected run time of the algorithm)
- Each steps "selects a w at random"
  - O(n log n) total steps
  - Average M rank: O(log n)

Expected W-rank

- If a w receives k random proposals, the expected rank for w is n/(k+1).
- On the average, a w receives O(log n) proposals
  - The average w rank is O(n/log n)

Probabilistic analysis

- Select items *with replacement* from a set of size n. What is the expected number of items to be selected until every item has been selected at least once.
- Choose k values at random from the interval [0, 1). What is the expected size of the smallest item.

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 **Executed at most n^2 times**
    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)

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?