CSE 421
Algorithms

Richard Anderson (for Anna Karlin)
Winter 2006
Lecture 1

Course Introduction

- Instructor
  - Anna Karlin, karlin@cs.washington.edu
- Impostor
  - Richard Anderson, anderson@cs.washington.edu
- Teaching Assistants
  - Lloyd Parlee, parlee@cs.washington.edu
  - Vibhor Rastogi, vibhor@cs.washington.edu

Announcements

- It's on the web.
- Homework 1, Due Jan 12
  - It's on the web
- Subscribe to the mailing list
- Anna will have an office hour Monday, Jan 9, 11am-noon. CSE 594

Text book

- Algorithm Design
- Jon Kleinberg, Eva Tardos
- Read Chapters 1 & 2

All of Computer Science is the Study of Algorithms

How to study algorithms

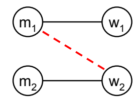
- Zoology
- Mine is faster than yours is
- Algorithmic ideas
  - Where algorithms apply
  - What makes an algorithm work
  - Algorithmic thinking

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



Examples

- m_1: w_1 w_2
- m_2: w_2 w_1
- w_1: m_1 m_2
- w_2: m_2 m_1

- m_1: w_1 w_2
- m_2: w_1 w_2
- w_1: m_1 m_2
- w_2: m_1 m_2

Examples

- m_1: w_1 w_2
- m_2: w_2 w_1
- w_1: m_2 m_1
- w_2: m_1 m_2

Intuitive 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
    - If w prefers m_2 to m, w rejects
- Unmatched m proposes to highest w on its preference list that m 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)
```

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
  - Once w is matched, w stays matched
  - w's partners get better

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

- Why?

Each m asks each w at most once

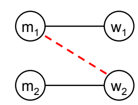
The algorithm terminates with a perfect matching

- Why?

If m is free, there is a w that has not been proposed to

The resulting matching is stable

- Suppose
  - m_1 prefers w_2 to w_1
  - w_2 prefers m_1 to m_2



- How could this happen?

m_1 proposed to w_2 before w_1
w_2 rejected m_1 for m_3
w_2 prefers m_3 to m_1
w_2 prefers m_2 to m_3

Result

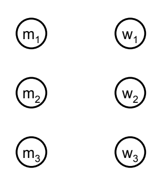
- Simple, $O(n^2)$ algorithm to compute a stable matching
- Corollary
  - A stable matching always exists

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



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

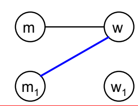
Proposal Algorithm finds the best possible solution for M

- And the worst possible for W
- (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*

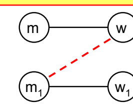
Proof

- Argument by contradiction
- Suppose the algorithm computes a matching S different from S*
- There must be some m rejected by a valid partner.
- Let m be the first man rejected by a valid partner w. w rejects m for m_1.
- w = best(m)

- S^+ stable matching including (m, w)
- Suppose m_1 is paired with w_1 in S^+
- m_1 prefers w to w_1
- w prefers m_1 to m
- Hence, (m_1, w) is an instability in S^+

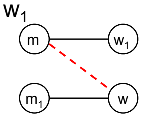


Since m_1 could not have been rejected by w_1 at this point, because (m, w) was the first valid pair rejected. (m_1, v_1) is valid because it is in S^+.



The proposal algorithm is worst case for W

- In S*, each w is paired with its worst valid partner
- Suppose (m, w) in S* but not m is not the worst valid partner of w
- S^- a stable matching containing the worst valid partner of w
- Let (m_1, w) be in S^-, w prefers m to m_1
- Let (m, w_1) be in S^-, m prefers w to w_1
- (m, w) is an instability in S^-



w prefers m to m_1 because m_1 is the wvp
w prefers w to w_1 because S* has all the bvp's

Could you do better?

- Is there a fair matching
- Design a configuration for problem of size n:
  - 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

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
- Underspecification of algorithm
- Establishing uniqueness of solution