Algorithm Design and Analysis

CSE 565

LECTURE 2

- Analysis of Stable Matching
- Asymptotic Notation

Adam Smith

Stable Matching Problem

- **Goal:** Given n men and n women, find a "suitable" matching.
  - –Participants rate members of opposite sex.
  - –Each man lists women in order of preference from best to worst.
  - –Each woman lists men in order of preference from best to worst.

	1st (favorite)	2nd	3rd (least favorite)
Xavier	Amy	Bertha	Clare
Yancey	Bertha	Amy	Clare
Zeus	Amy	Bertha	Clare

Men's Preference Profile

	1st (favorite)	2nd	3rd (least favorite)
Amy	Yancey	Xavier	Zeus
Bertha	Xavier	Yancey	Zeus
Clare	Xavier	Yancey	Zeus

Women's Preference Profile

Stable Matching Problem

- **Unstable pair:** man ***m*** and woman ***w*** are unstable if
  - ***m*** prefers ***w*** to his assigned match, and
  - ***w*** prefers ***m*** to her assigned match
- **Stable assignment**: no unstable pairs.

	1st (favorite)	2nd	3rd (least favorite)
Xavier	Amy	Bertha	Clare
Yancey	Bertha	Amy	Clare
Zeus	Amy	Bertha	Clare

Men's Preference Profile

	1st (favorite)	2nd	3rd (least favorite)
Amy	Yancey	Xavier	Zeus
Bertha	Xavier	Yancey	Zeus
Clare	Xavier	Yancey	Zeus

Women's Preference Profile

Review Questions

- In terms of n, what is the length of the input to the Stable Matching problem, i.e., the number of entries in the tables?

(Answer: $2n^2$ list entries, or $2n^2 \log n$ bits)

Review Question

- **Brute force algorithm:** an algorithm that checks every possible solution.

- In terms of n, what is the running time for the brute force algorithm for Stable Matching Problem? (Assume your algorithm goes over all possible perfect matchings.)

(Answer: $n! \times$ (time to check if a matching is stable) $= \Theta(n!\, n^2)$)

Propose-and-Reject Algorithm

- Propose-and-reject algorithm. [Gale-Shapley 1962]

```
Initialize each person to be free.
while (some man is free and hasn't proposed to every woman) {
    Choose such a man m
    w = 1st woman on m's list to whom m has not yet proposed
    if (w is free)
        assign m and w to be engaged
    else if (w prefers m to her fiancé m')
        assign m and w to be engaged, and m' to be free
    else
        w rejects m
}
```

Proof of Correctness: Termination

- **Claim.** Algorithm terminates after at most n^2 iterations of while loop.
- **Pf.** Each time through the loop a man proposes to a new woman. There are only n^2 possible proposals. ▪

	1st	2nd	3rd	4th	5th
Victor	A	B	C	D	E
Wyatt	B	C	D	A	E
Xavier	C	D	A	B	E
Yancey	D	A	B	C	E
Zeus	A	B	C	D	E

	1st	2nd	3rd	4th	5th
Amy	W	X	Y	Z	V
Bertha	X	Y	Z	V	W
Clare	Y	Z	V	W	X
Diane	Z	V	W	X	Y
Erika	V	W	X	Y	Z

An instance where n(n-1) + 1 proposals required

Propose-and-Reject Algorithm

- **Observation 1.** Men propose to women in decreasing order of preference.

- **Observation 2.** Once a woman is matched, she never becomes unmatched; she only "trades up."

CSE 565 Proof of Correctness: Perfection

- **Claim.** All men and women get matched.
- **Proof:** (by contradiction)
  - Suppose, for sake of contradiction, some guy, say Zeus, is not matched upon termination of algorithm.
  - Then some woman, say Amy, is not matched upon termination.
  - By Observation 2, Amy was never proposed to.
  - But Zeus proposes to everyone, since he ends up unmatched. ▪

Proof of Correctness: Stability

- **Claim.** No unstable pairs.
- **Proof:** (by contradiction)
  - Suppose A-Z is an unstable pair: they prefer each other to their partners in Gale-Shapley matching S*.
  - *Case 1:* Z never proposed to A.
    - ⇒ Z prefers his GS partner to A. ← men propose in decreasing order of preference
    - ⇒ A-Z is stable.
  - *Case 2:* Z proposed to A.
    - ⇒ A rejected Z (right away or later)
    - ⇒ A prefers her GS partner to Z. ← women only trade up
    - ⇒ A-Z is stable.
  - In either case A-Z is stable, a contradiction. ▪

Efficient Implementation

- We describe $O(n^2)$ time implementation.
- Assume men have IDs 1,…, n, and so do women.
- Engagements data structures:
  - a list of free men, e.g., a queue.
  - two arrays `wife[m]`, and `husband[w]`.
    - set entry to `0` if unmatched
    - if m matched to w then `wife[m]=w` and `husband[w]=m`
- Men proposing data structures:
  - an array `men-pref[m,i]=` `i`th women on `m`th list
  - an array `count[m]` = how many proposals `m` made.

Efficient Implementation

- Women rejecting/accepting data structures
  - Does woman `w` prefer man `m` to man `m'`?
  - For each woman, create **inverse** of preference list of men.
  - Constant time queries after ***O(n)*** preprocessing per woman.

Amy	1st	2nd	3rd	4th	5th	6th	7th	8th
Pref	8	3	7	1	4	5	6	2

Amy	1	2	3	4	5	6	7	8
Inverse	4th	8th	2nd	5th	6th	7th	3rd	1st

```
for i = 1 to n
   inverse[pref[i]] = i
```

Amy prefers man 3 to 6 since `inverse[3]` < `inverse[6]`
2 7

Summary

- **Stable matching problem.** Given ***n*** men and ***n*** women, and their preferences, find a stable matching if one exists.
- **Gale-Shapley algorithm.** Guarantees to find a stable matching for **every** problem instance.
  - (Also proves that stable matching always exists)
- **Time and space complexity:**

  $O(n^2)$, linear in the input size.

Brief Syllabus

- Reminders
  - Worst-case analysis
  - Asymptotic notation
  - Basic Data Structures
- Design Paradigms
  - Greedy algorithms, Divide and conquer, Dynamic programming, Network flow and linear programming
- Analyzing algorithms in other models
  - Parallel algorithms, Memory hierarchies (?)
- P, NP and NP-completeness

Measuring Running Time

- Focus on scalability: parameterize the running time by some measure of "size"
  - (e.g. n = number of men and women)
- Kinds of analysis
  - Worst-case
  - Average-case (requires knowing the distribution)
  - Best-case (how meaningful?)
- Exact times depend on computer; instead measure asymptotic growth