Algorithm Design and Analysis

CSE 565

LECTURE 3

An Example Problem

- Stable matching problem

Asymptotic Notation

- *O*-, Ω-, Θ-, o-, ω-notation

Sofya Raskhodnikova

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: $\Omega(n!)$)

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."

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.

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.
- **Time and space complexity:**

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

Asymptotic Order of Growth

- **Upper bound.** ***T(n)* is *O(f(n))*** if there exist constants $c > 0$ and $n_0 \ge 0$ such that for all $n \ge n_0$: $0 \le T(n) \le c \cdot f(n)$.
- **Lower bound.** ***T(n)* is *Ω(f(n))*** if there exist constants $c > 0$ and $n_0 \ge 0$ such that for all $n \ge n_0$: $T(n) \ge c \cdot f(n)$.
- **Tight bound.** ***T(n)* is *Θ(f(n))*** if T(n) is both O(f(n)) and Ω(f(n)).
- Example: $T(n) = 32n^2 + 17n + 32$.
  - T(n) is $O(n^2)$, $O(n^3)$, $\Omega(n^2)$, $\Omega(n)$, and $\Theta(n^2)$.
  - T(n) is not $O(n)$, $\Omega(n^3)$, $\Theta(n)$, or $\Theta(n^3)$.

Notation

- **One-sided equality:** T(n) = O(f(n)).
  - Not transitive:
    - $f(n) = 5n^3$; $g(n) = 3n^2$
    - $f(n) = O(n^3) = g(n)$
    - but $f(n) \neq g(n)$.
  - Alternative notation: $T(n) \in O(f(n))$.
- **Meaningless:** Any comparison-based sorting algorithm requires at least O(n log n) comparisons.
  - Use Ω for lower bounds.

Properties

- Transitivity.
  - If f = O(g) and g = O(h) then f = O(h).
  - If f = Ω(g) and g = Ω(h) then f = Ω(h).
  - If f = Θ(g) and g = Θ(h) then f = Θ(h).
- Additivity.
  - If f = O(h) and g = O(h) then f + g = O(h).
  - If f = Ω(h) and g = Ω(h) then f + g = Ω(h).
  - If f = Θ(h) and g = O(h) then f + g = Θ(h).

Common Functions: Asymptotic Bounds

- **Polynomials.** $a_0 + a_1 n + \ldots + a_d n^d$ is $\Theta(n^d)$ if $a_d > 0$.
- **Polynomial time.** Running time is $O(n^d)$ for some constant d independent of the input size n.
- **Logarithms.** $\log_a n = \Theta(\log_b n)$ for all constants a, b > 0.

  can avoid specifying the base

  log grows slower than every polynomial

  For every x > 0, $\log n = O(n^x)$.

  every exponential grows faster than every polynomial
- **Exponentials.** For all r > 1 and all d > 0, $n^d = O(r^n)$.
- **Factorial.** By Sterling's formula, $n! = 2^{\Theta(n \log n)}$

CSE 565

Overview

Notation	… means …	Think…	E.g.	Lim $f(n)/g(n)$
$f(n)=O(n)$	$\exists\ c>0, n_0>0, \forall\ n > n_0$: $0 \le f(n) < cg(n)$	Upper bound	$100n^2 = O(n^3)$	If it exists, it is $< \infty$
$f(n)=\Omega(g(n))$	$\exists\ c>0, n_0>0, \forall\ n > n_0$: $0 \le cg(n) < f(n)$	Lower bound	$n^{100} = \Omega(2^n)$	If it exists, it is > 0
$f(n)=\Theta(g(n))$	both of the above: $f=\Omega(g)$ **and** $f = O(g)$	Tight bound	$\log(n!) = \Theta(n \log n)$	If it exists, it is > 0 and $< \infty$
$f(n)=o(g(n))$	$\forall\ c>0, n_0>0, \forall\ n > n_0$: $0 \le f(n) < cg(n)$	Strict upper bound	$n^2 = o(2^n)$	Limit exists, $=0$
$f(n)=\omega(g(n))$	$\forall\ c>0, n_0>0, \forall\ n > n_0$: $0 \le cg(n) < f(n)$	Strict lower bound	$n^2 = \omega(\log n)$	Limit exists, $=\infty$

CSE 565

Some functions sorted by asymptotic growth

- $\log(n)$
- $\sqrt{n}$
- n
- $n \log(n)$
- n^2
- $n^{1,000,000}$
- 2^n (beats n^k for any fixed k)
- $n!$