## Objectives

- Finish implementation of Stable Matching
  - Get out your handouts
- Survey of common running times

Checking in on
- journal
- problem set → solved exercises in text book



## Review

- What does O(f(n)) mean?
  - Intuitive
  - More precise definition
- What are the other bounds we discussed?
- What are the differences between arrays and lists?



## Review: Gale-Shapley Stable Matching Algorithm

```
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
```



## Stable Matching Implementation

- What do we need to represent? How should we represent them?

| Data | How represented |
|---|---|
| Men, Women | |
| Preference lists | |
| Unmatched men | |
| Who men proposed to | |
| Engagements | |



## Stable Matching Implementation

- What do we need to represent? How should we represent them?

| Data | How represented |
|---|---|
| Men, Women | Integers (like ids) |
| Preference lists | Array of arrays (2d array) |
| Unmatched men | List |
| Who men proposed to | Integer for each man → Array of integers |
| Engagements | 2 Arrays |



## Asymptotic Analysis of Gale-Shapley Alg

```
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
```

What is the running time of each part of the algorithm?
What is the total running time of the algorithm?

## Efficient Implementation

- Women rejecting/accepting: determine does woman w prefer man m to man m'?
  - For each woman, create array of men with her preference
    - *inverse* of preference list
  - Constant time access for each query after O(n) preprocessing

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

Contains man's id

For each man, how does he rank?

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

Amy prefers man 3 to 6 since inverse[3] < inverse[6]

2 7

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



## Asymptotic Analysis of Gale-Shapley Alg

Not explicitly in the algorithm, but we need to make the inverse array before the while loop too.

```
Initialize each person to be free                    O(n)
while (some man is free and hasn't proposed to every woman)   O(n^2)
    Choose such a man m                              O(1)
    w = 1st woman on m's list to whom m has not yet proposed   O(1)
    if (w is free)                                   O(1)
        assign m and w to be engaged                 O(1)
    else if (w prefers m to her fiancé m')           O(1)   Using inverse array
        assign m and w to be engaged and m' to be free   O(1)
    else
        w rejects m                                  O(1)
```

Total: $O(n^2)$



## A SURVEY OF COMMON RUNNING TIMES



## Linear Time: O(n)

- Running time is at most a ***constant*** factor times the size of the input
- Example. Computing the maximum: Compute maximum of n numbers $a_1, \ldots, a_n$

```
max = a_1
for i = 2 to n
    if (a_i > max)
        max = a_i
```

**Constant** work for each input (does not depend on n)



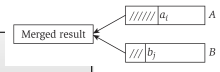
## Example Linear Time: O(n)

- Merge: Combine two sorted lists $A = a_1, a_2, \ldots, a_n$ with $B = b_1, b_2, \ldots, b_n$ into sorted whole



## Example Linear Time: O(n)

- Merge: Combine two sorted lists $A = a_1, a_2, \ldots, a_n$ with $B = b_1, b_2, \ldots, b_n$ into sorted whole
- Claim. Merging two lists of size *n* takes O(n) time

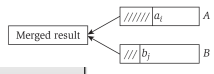



```
i = 1, j = 1
while (both lists are nonempty)
    if (a_i ≤ b_j)
        append a_i to output list and increment i
    else (a_i ≥ b_j)
        append b_j to output list and increment j

append remainder of nonempty list to output list
```

## Example Linear Time: O(n)

- Merge: Combine two sorted lists $A = a_1, a_2, \ldots, a_n$ with $B = b_1, b_2, \ldots, b_n$ into sorted whole
- Claim. Merging two lists of size n takes O(n) time
- Proof. After each comparison, the length of output list increases by 1




```
i = 1, j = 1
while (both lists are nonempty)
   if (a_i ≤ b_j)
        append a_i to output list and increment i
   else (a_i ≤ b_j)
        append b_j to output list and increment j

append remainder of nonempty list to output list
```



## O(n log n) Time

- Also referred to as ***linearithmic*** time
- Arises in divide-and-conquer algorithms
  - Splitting input into equal pieces, solve recursively, combine solutions in linear time

What well-known set of algorithms has an O(n logn) running time?



## O(n log n) Time Example

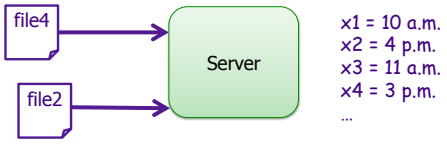
- Sorting: Mergesort and heapsort are sorting algorithms that perform O(n log n) comparisons
- Mergesort
  1. Break input into equal-sized pieces
  2. Sorts each half recursively
  3. Merges sorted halves into a sorted list

Talk about the bound on running time later…



## O(n log n) Time Example

- Largest empty interval. Given *n* (not necessarily ordered) time-stamps $x_1, \ldots, x_n$ at which copies of a file arrive at a server, what is largest interval of time when no copies of the file arrive?






## O(n log n) Time Example

- Largest empty interval. Given *n* (not necessarily ordered) time-stamps $x_1, \ldots, x_n$ at which copies of a file arrive at a server, what is largest interval of time when no copies of the file arrive?
- O(n log n) solution
  1. Sort time-stamps
  2. Scan sorted list in order, identifying the maximum gap between successive time-stamps



## Quadratic Time: $O(n^2)$

- Examples?

## Quadratic Time: $O(n^2)$

- Examples:
  - Enumerate all pairs of elements
  - Sometimes involves nested loops (n iterations)



## Quadratic Time: $O(n^2)$

- Closest pair of points. Given a list of n points in the plane $(x_1, y_1), \ldots, (x_n, y_n)$, find the pair that is closest
- $O(n^2)$ solution. Try all pairs of points

```
min = (x1 - x2)² + (y1 - y2)²
for i = 1 to n
   for j = i+1 to n
      d = (xi - xj)² + (yi - yj)²
      if (d < min)
         min = d
```

don't need to take square roots

$\Omega(n^2)$ seems inevitable, but Chapter 5 has an O(n logn) solution



## Polynomial Time: $O(n^k)$ Time

- To get all pairs, the algorithm is $O(n^2)$
- To get all triplets, the algorithm is $O(n^3)$

What is an example of an $O(n^k)$ algorithm?

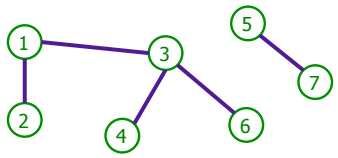
All subsets of size $k$

Stopped here …
Kept slides as a supplement to book reading



## Polynomial Time: $O(n^k)$ Time

- Independent set of size $k$. Given a graph, are there $k$ nodes such that no two are joined by an edge?
  - $k$ is a constant

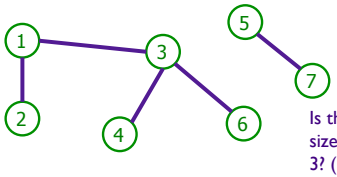



Is there an independent set of size 2? 3? 4? 5?



## Polynomial Time: $O(n^k)$ Time

- Independent set of size $k$. Given a graph, are there $k$ nodes such that no two are joined by an edge?
  - $k$ is a constant




Is there an independent set of size 2? Yes (2-3; 1-5; 6-7; …)
3? (5-6-7; 2-3-5; …)
4? (2-4-5-7; 1-4-6-7; …)
But not 5



## Polynomial Time: $O(n^k)$ Time

- Independent set of size $k$. Given a graph, are there $k$ nodes such that no two are joined by an edge?
  - $k$ is a constant

```
foreach subset S of k nodes
   check whether S in an independent set
   if (S is an independent set)
      report S is an independent set
```

- $O(n^k)$ solution
  1. Enumerate all subsets of k nodes

$$\binom{n}{k} = \frac{n!}{k!\,(n-k)!} = \frac{n\,(n-1)\,(n-2)\ldots(n-k+1)}{k\,(k-1)\,(k-2)\ldots(2)\,(1)} \leq \frac{n^k}{k!}$$

  2. Check whether S is an independent set = $O(k^2)$.

$O(k^2\, n^k / k!) = O(n^k)$

poly-time for k=17 but not practical

## Exponential Time

- Independent set. Given a graph, what is the *maximum size* of an independent set?
- $O(n^2 2^n)$ solution. Enumerate all subsets

```
S* = ϕ
foreach subset S of nodes
   check whether S in an independent set
   if (S is largest independent set seen so far)
      S* = S
```



## O(log n) Time

- ***Sublinear*** time
- Know any algorithms that take O(log n) time?



## O(log n) Time

- Example: Binary search
- Often requires some pre-processing or data structure that allows cheaper "querying" than *n* time



## Summary of Running Times

| Running Time | Example |
|---|---|
| O(log n) | Generally dividing problem in half on each iteration |
| O(n) | Operate on each input value |
| O(n log n) | Divide and conquer |
| $O(n^2)$ | Operate on each pair of inputs |
| O(n!) | Operate on each permutation of inputs |



## Looking ahead

- Problem Set Due Friday
  - Preferably typed
- Continue reading, summarizing Chapter 2
- Friday's class: 10:45 a.m.
  - ODK inductions