Objectives

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

Checking in on 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	Integers (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)                                   O(n²)
while (some man is free and hasn't proposed to every woman)
    Choose such a man m   O(1)                                              O(1)
    w = 1st woman on m's list to whom m has not yet proposed
    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 = $a_i$
```

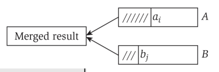
Constant work for each input (does not depend on n)

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

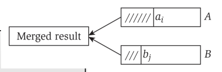


```
i = 1, j = 1
while (both lists are nonempty)
    if ($a_i \le b_j$)
        append $a_i$ to output list and increment i
    else ($a_i \le b_j$)
        append $b_j$ to output list and increment j

append remainder of nonempty list to output list
```

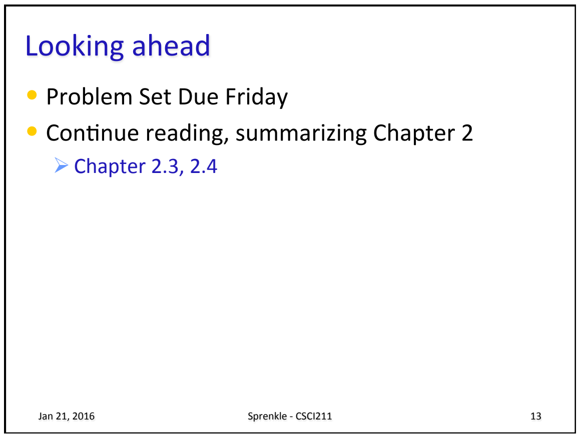
Example Linear Time: O(n)

- 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 ($a_i \le b_j$)
        append $a_i$ to output list and increment i
    else ($a_i \le b_j$)
        append $b_j$ to output list and increment j

append remainder of nonempty list to output list
```



Looking ahead

- Problem Set Due Friday
- Continue reading, summarizing Chapter 2
  - Chapter 2.3, 2.4

Jan 21, 2016 Sprenkle - CSCI211 13