Objectives

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

Wiki Feedback

- Is coming!
- Edited the wikis – fixing sidebars, adding pages
- **Content – For each section**
  - Brief summary of the section
    - ~1 paragraph of about 5-10 sentences per section
    - feel free to write more if that will help you
  - Include motivations for the given problem, as appropriate
  - For algorithms, brief sketch of algorithm, intuition, and implementation
    - Include runtime for algorithms
  - Questions you have about motivation/solution/proofs/analysis
  - Discuss anything that makes more sense after reading it again, after it was presented in class (or vice versa)
  - Anything that you want to remember, anything that will help you
  - Say something about how readable/interesting the section was on scale of 1 to 10

Review

Get out handouts from last time

- Run times:
  - What does O(f(n)) mean?
    - Intuitive
    - More precise definition
  - What are the other bounds we discussed?
- Implementing Stable Matching Algorithm
  - What do we need to model/represent?
  - What are the differences between a list and an array?
    - What is the cost to convert from an array to a list?
  - Which data structure(s) makes the most sense for the Stable Matching Problem?

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 (ids)
Preference lists	2 Array of arrays (2D array)
Unmatched men	List
Who men proposed to	Integer for each man → Array of integers (maps man's id to "next" woman)
Engagements	2 Arrays, length n

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

```
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?

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: $O(n^2)$

Looking Ahead

- Problem Set 1 due Friday, before class