Objectives

- Review: Asymptotic running times
- Implementing Gale-Shapley algorithm
- Classes of running times

Review Asymptotic Bounds

- What does O(f(n)) mean?
- What are the other two bounds we discussed?
- We discussed three classes of running times
  - What are they?
  - Order them by their growth rates

A Fashion Analogy

- O == Hammer pants
  - Loose and baggy with plenty of room for the pants to shrink or the body to grow
- Ω == The pants you plan to fit in this summer after working off the snacks from Christmas
- Θ == Katy Perry's skin tight jeans in a teenage dream
  - Can't make them any smaller, and no extra room to even fit a cell phone in the pocket

Courtesy Andy Danner, Swarthmore

Answer to Tight Bound Question

- Yes, there can be more than one tight bound
  - But not really
- Example with same asymptotic bounds
- For non-equivalent asymptotic bounds
  - There is "slop" on at least one side of the bound
  - Can make one of the functions tighter→ not tight to begin with

Review: Our Process

1. Understand/identify problem
   - Simplify as appropriate
2. Design a solution
3. Analyze
   - Correctness, efficiency
   - May need to go back to step 2 and try again
4. Implement
   - Within bounds shown in analysis

IMPLEMENTING GALE-SHAPLEY ALGORITHM

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
```


How Can We Implement The Algorithm Efficiently?

- What is our goal for the implementation's runtime?
- What do we need to model?
- How should we represent them?

How Can We Implement The Algorithm Efficiently?

- What is our goal for the implementation's runtime?
  - $O(N^2)$
- What do we need to model?
- How should we represent them?

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	

What's the difference between an array and a list?

Arrays

- *Fixed* number of elements
- What is the runtime of
  - Determining the value of the i^{th} item in the array?
  - Determining if a value *e* is in the array?
  - Determining if a value *e* is in the array if the array is sorted?

Array Operations' Running Times

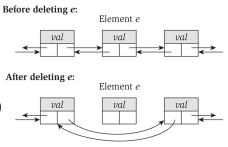
Operation	Running Time
Value of i^{th} item	O(1) → direct access
If *e* is in the array	O(n) → look through all the elements
If *e* is in the array if sorted	O(log n) → binary search

Limitation of arrays?

Fixed size, so difficult to add/delete elements

Lists

- Dynamic set of elements
  - Linked list
  - Doubly linked list
- What is the running time to
  - Add an element to the list?
  - Delete an element from the list?
  - Find an element *e* in the list?
  - Find the i^{th} element in the list?



List Operations' Running Time

Operation	Running Time
Add element	O(1)
Delete element	O(1)
Find element	O(n)
Find i^{th} element	O(i)

Disadvantage of list instead of array?

Finding i^{th} element is slower

Converting between Lists and Arrays (and Vice Versa)

- What is the running time of converting a list to an array?
- An array to a list?

O(n)

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
Preference lists	Array of arrays
Unmatched men	List
Who men proposed to	Array of integers
Engagements	2 Arrays

Efficient Implementation

- Women rejecting/accepting: determine does woman `w` prefer man `m` to man `m'`?
  - For each woman, create *inverse* of preference list of men
  - 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

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

```
Initialize each person to be free                    3n
while (some man is free and hasn't proposed to every woman)   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)
        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)

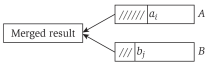
- Running time is at most a ***constant*** factor times the size of the input
- Example. Computing the maximum: Compute maximum of n numbers $a_1, \ldots, a_n$

```
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)

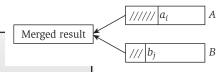
Example Linear Time: O(n)

- 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

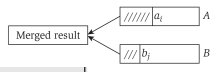



```
i = 1, j = 1
while (both lists are nonempty)
    if (a_i ≤ b_j)
        append a_i to output list and increment i
    else (a_i ≤ b_j)
        append b_j to output list and increment j

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
```

25

Assignments

- Review Chapter 2
  - Finishing up on Wednesday
  - ODK: 9:50 a.m.-10:35 a.m.
- Journal due Wednesday before class
- Problem Set 1 due Friday in class
  - FAQ of commonly asked questions on course web page