# Lecture 9

CSE 331

# Please have a face mask on

**Masking requirement**


*UB requires all students, employees and visitors – regardless of their vaccination status – to wear face coverings while inside campus buildings.*

https://www.buffalo.edu/coronavirus/health-and-safety/health-safety-guidelines.html

# Proof Details of Lemma 1

# Gale Shapley algorithm terminates

This page collects material from Fall 17 incarnation of CSE 331, where we proof details for the claim that the Gale-Shapley algorithm terminates in $O(n^2)$ iterations.

## Where does the textbook talk about this?

`Section 1.1` in the textbook has the argument (though not in as much detail as below).

## Fall 2017 material

Here is the lecture video (it starts from the part where we d the proof details):

# Proof by contradiction

Assume the negation of what you want to prove

After some reasoning


Source: 4simpsons.wordpress.com

# Two obervations

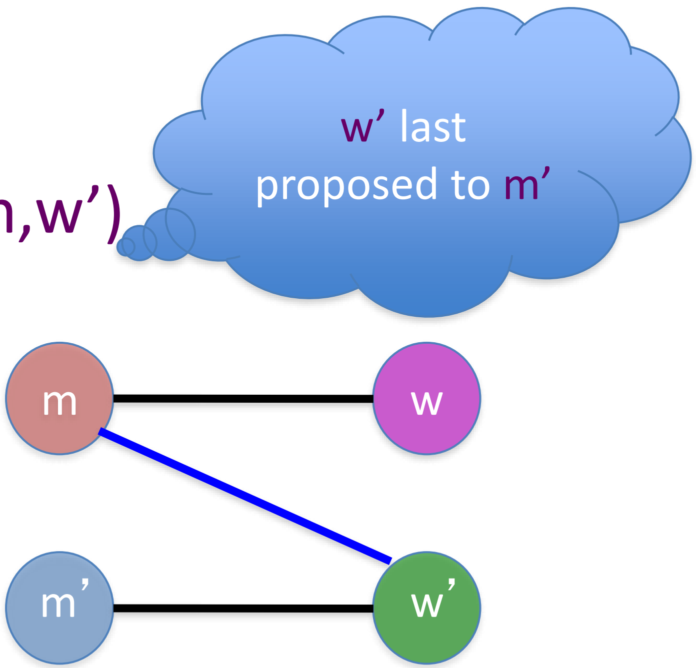
**Obs 1**: Once m is engaged he keeps getting engaged to "better" women

**Obs 2**: If w proposes to m' first and then to m (or never proposes to m) then she prefers m' to m

# Proof of Lemma 3

By contradiction

Assume there is an instability (m,w')




m prefers w' to w

w' prefers m to m'

# Contradiction by Case Analysis

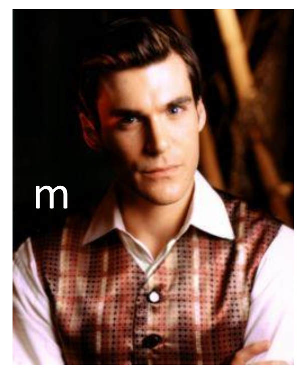
Depending on whether w’ had proposed to m or not

Case 1: w’ never proposed to m

w’ prefers m’ to m

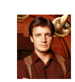
By Obs 2

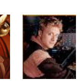
Assumed w’ prefers m to m’

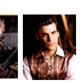





Source: 4simpsons.wordpress.com







# Case 2: w’ had proposed to m



## Case 2.1: m had accepted w’ proposal

m is finally engaged to w

Thus, m prefers w to w’

By Obs 1


4simpsons.wordpress.com

## Case 2.2: m had rejected w’ proposal

m was engaged to w’’ (prefers w’’ to w’)

By Algo def

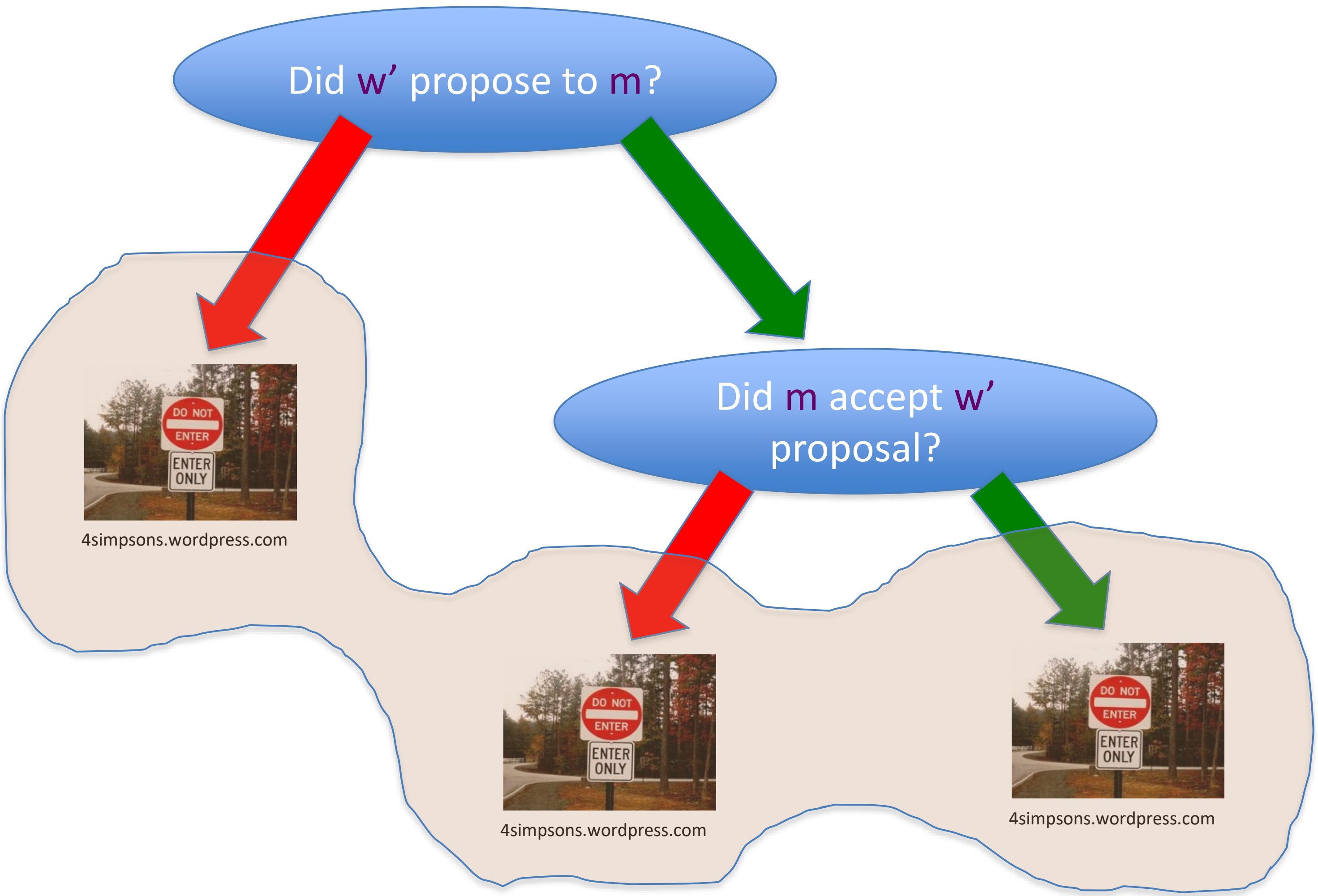
m is finally engaged to w (prefers w to w’’)

By Obs 1

m prefers w to w’


4simpsons.wordpress.com

# Overall structure of case analysis



# Questions?

# Extensions

Fairness of the GS algorithm

Different executions of the GS algorithm

# Main Steps in Algorithm Design

Problem Statement

↓

Problem Definition

↓

Algorithm

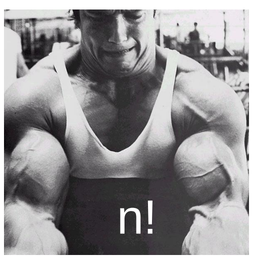
↓

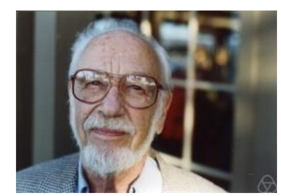
“Implementation”

↓

Analysis

NRMP
National Resident Matching Program




Correctness Analysis

# Definition of Efficiency

An algorithm is efficient if, when implemented, it runs quickly on real instances

Implemented where?



Platform independent definition

What are real instances?

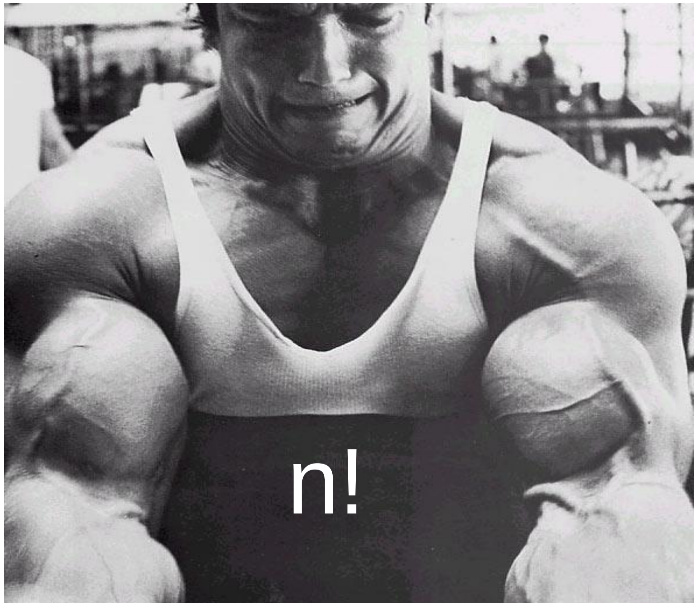
Worst-case Inputs

$N = 2n^2$ for SMP

Efficient in terms of what?

Input size N

# Definition-II




Analytically better than brute force

How much better? By a factor of 2?

# Definition-III

Should scale with input size

If N increases by a constant factor, so should the measure

Polynomial running time

At most $c \cdot N^d$ steps ($c>0$, $d>0$ absolute constants)

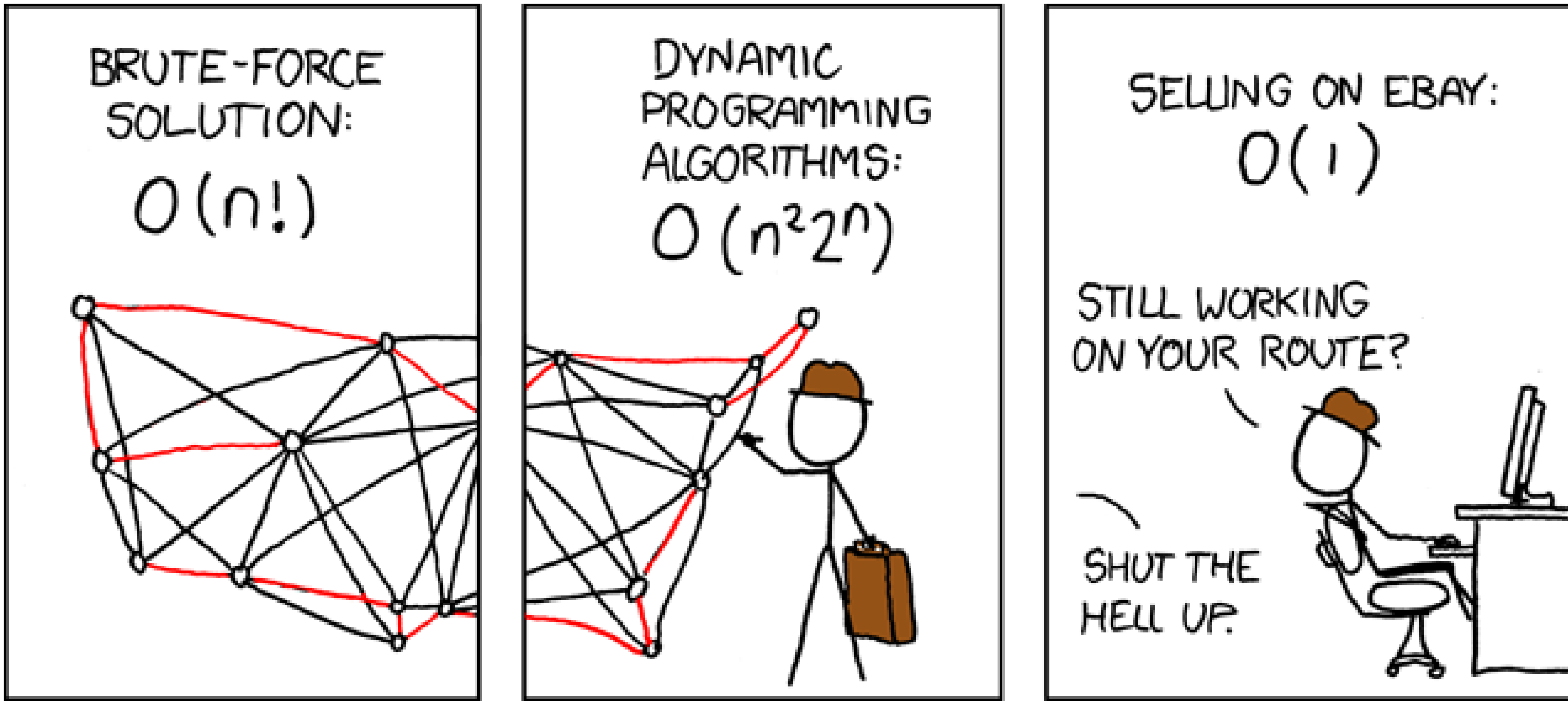
Step: “primitive computational step”

# More on polynomial time

Problem centric tractability

Can talk about problems that are not efficient!

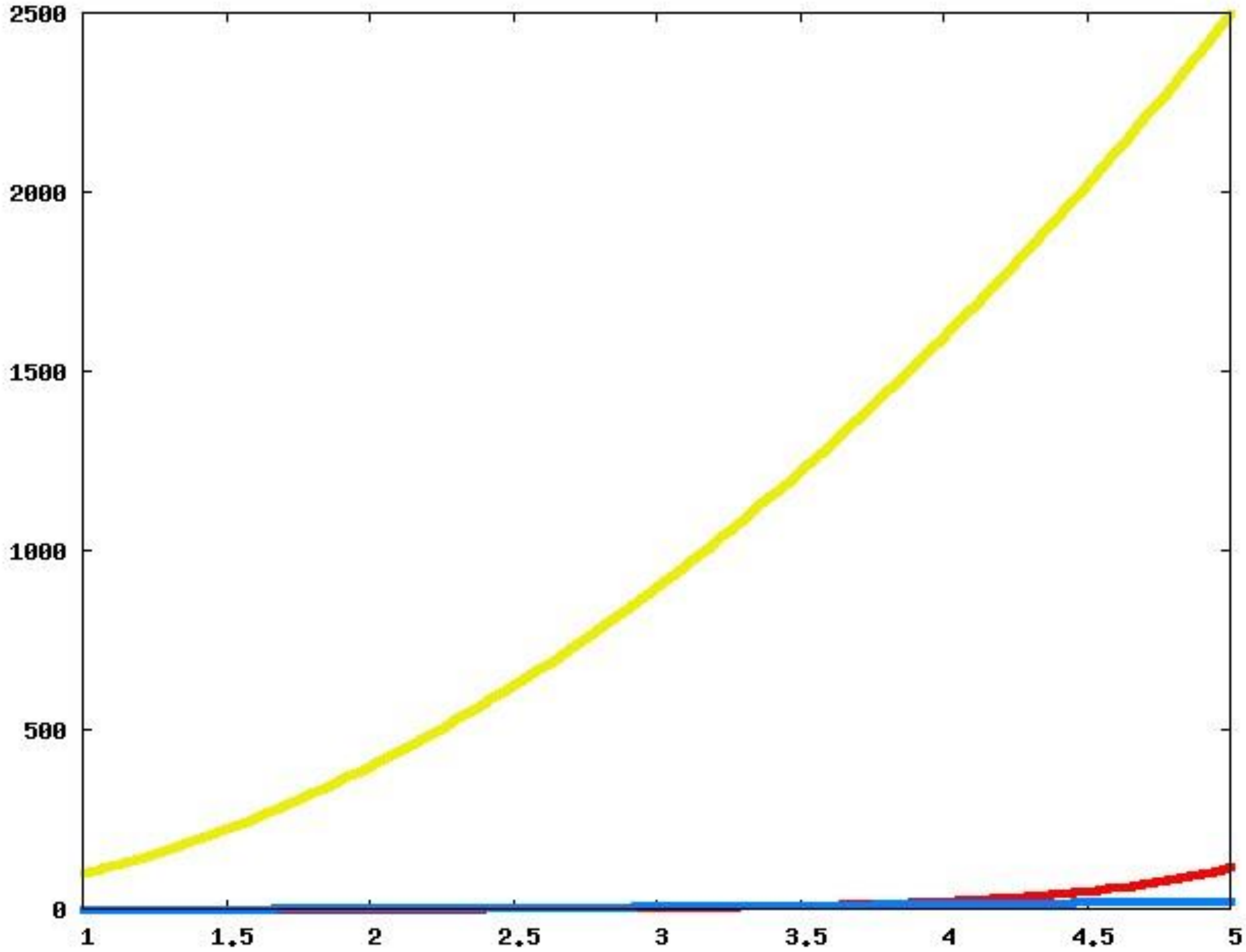
# Asymptotic Analysis



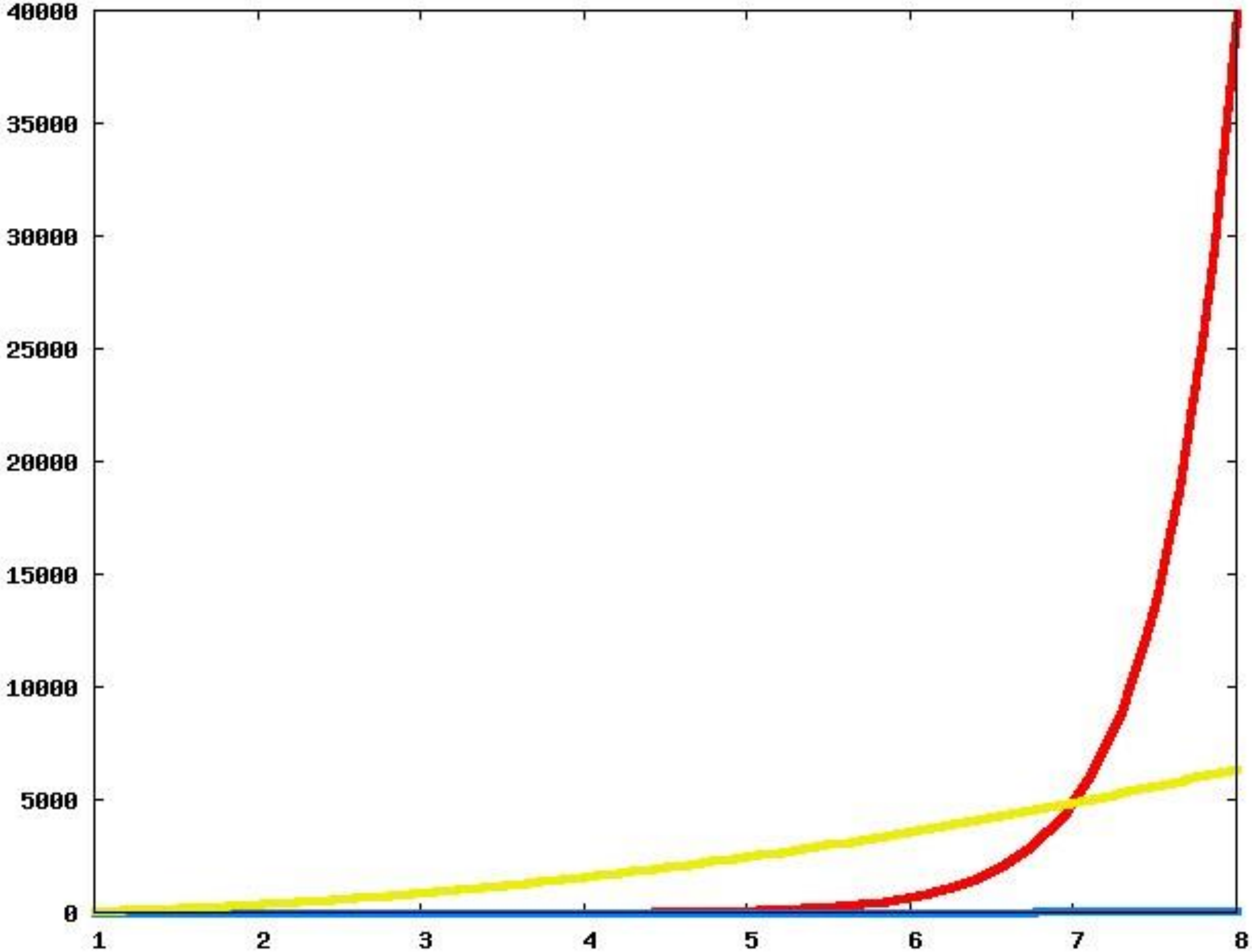

Travelling Salesman Problem

(http://xkcd.com/399/)

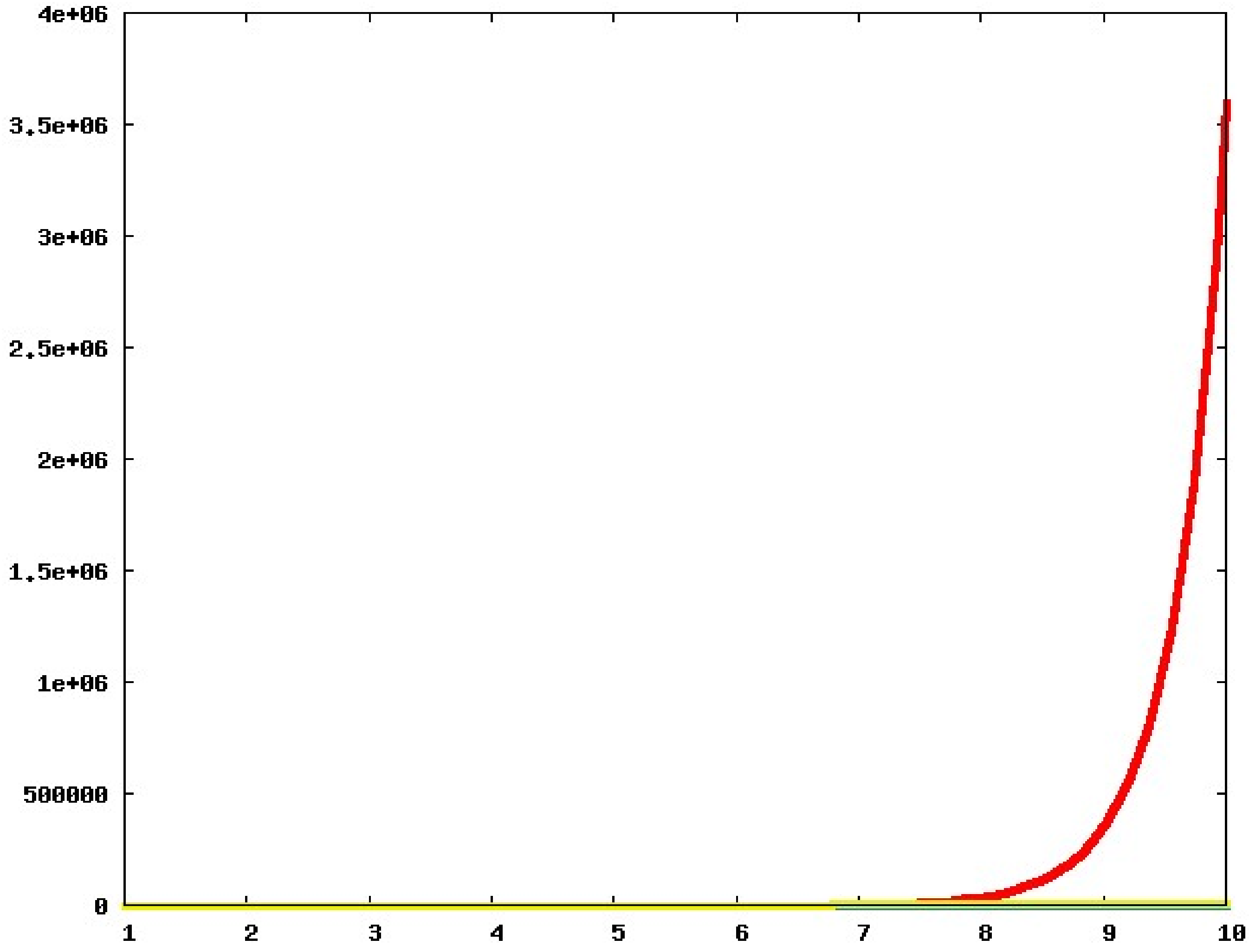
# Which one is better?



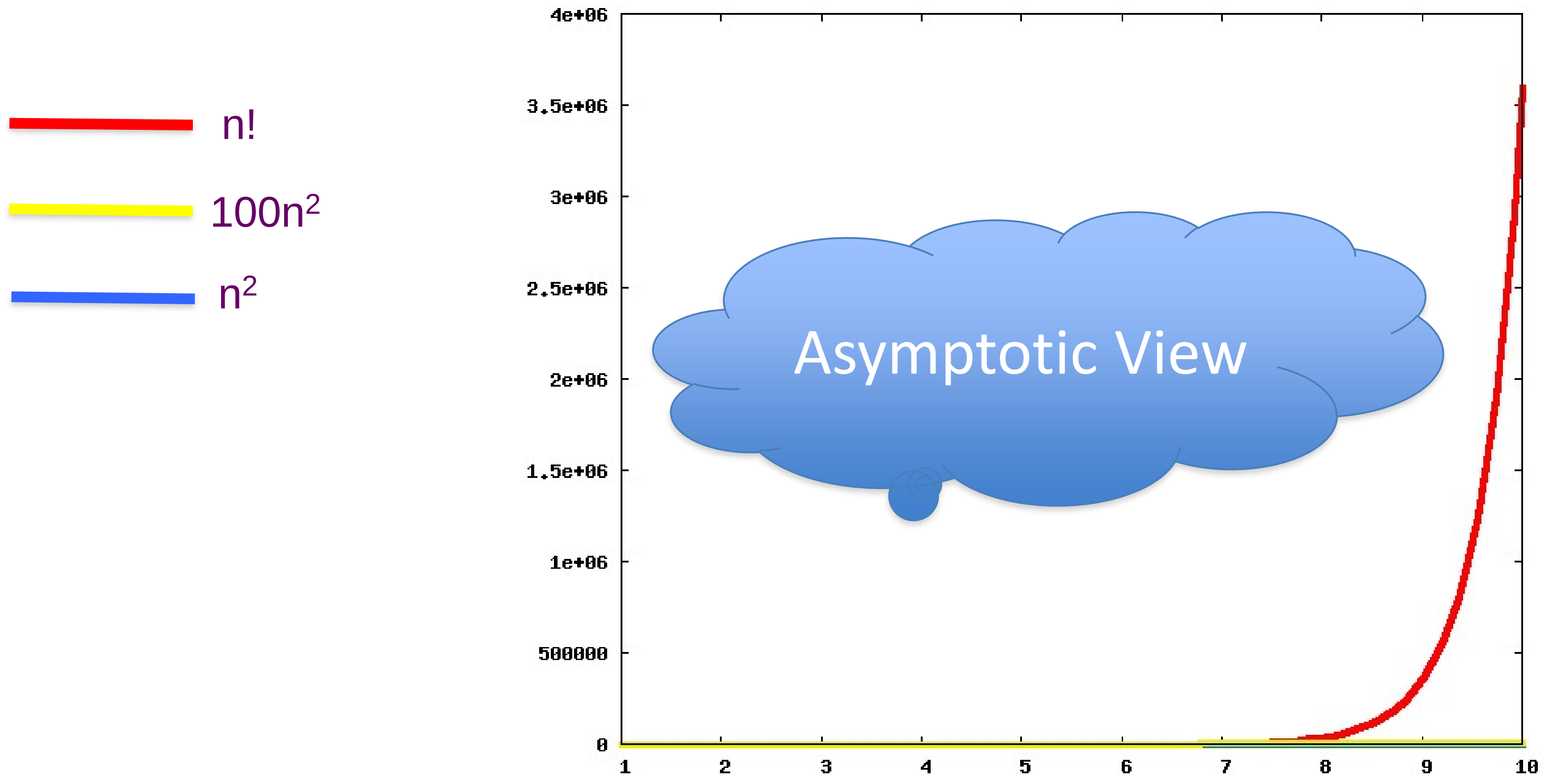
# Now?



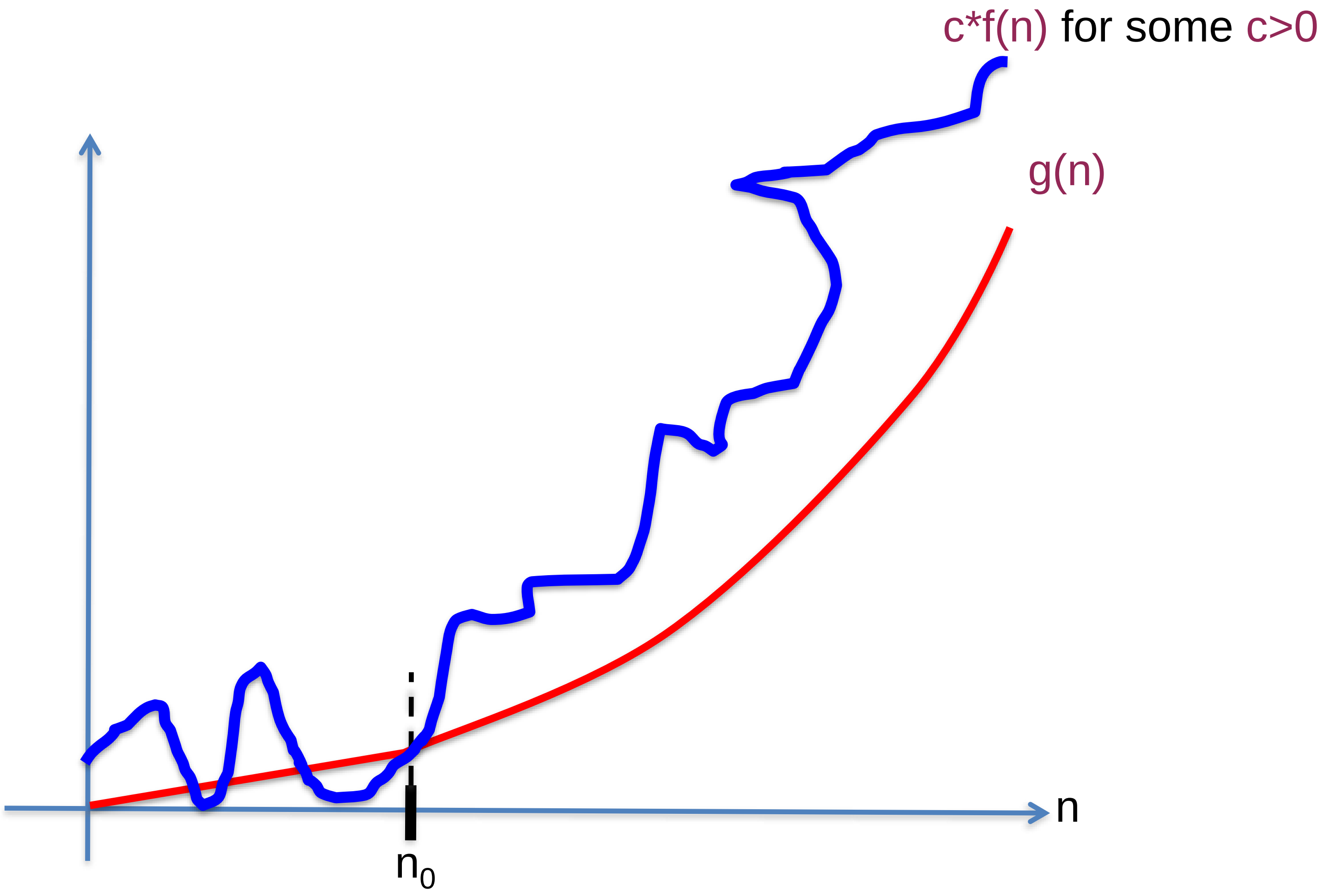
# And now?



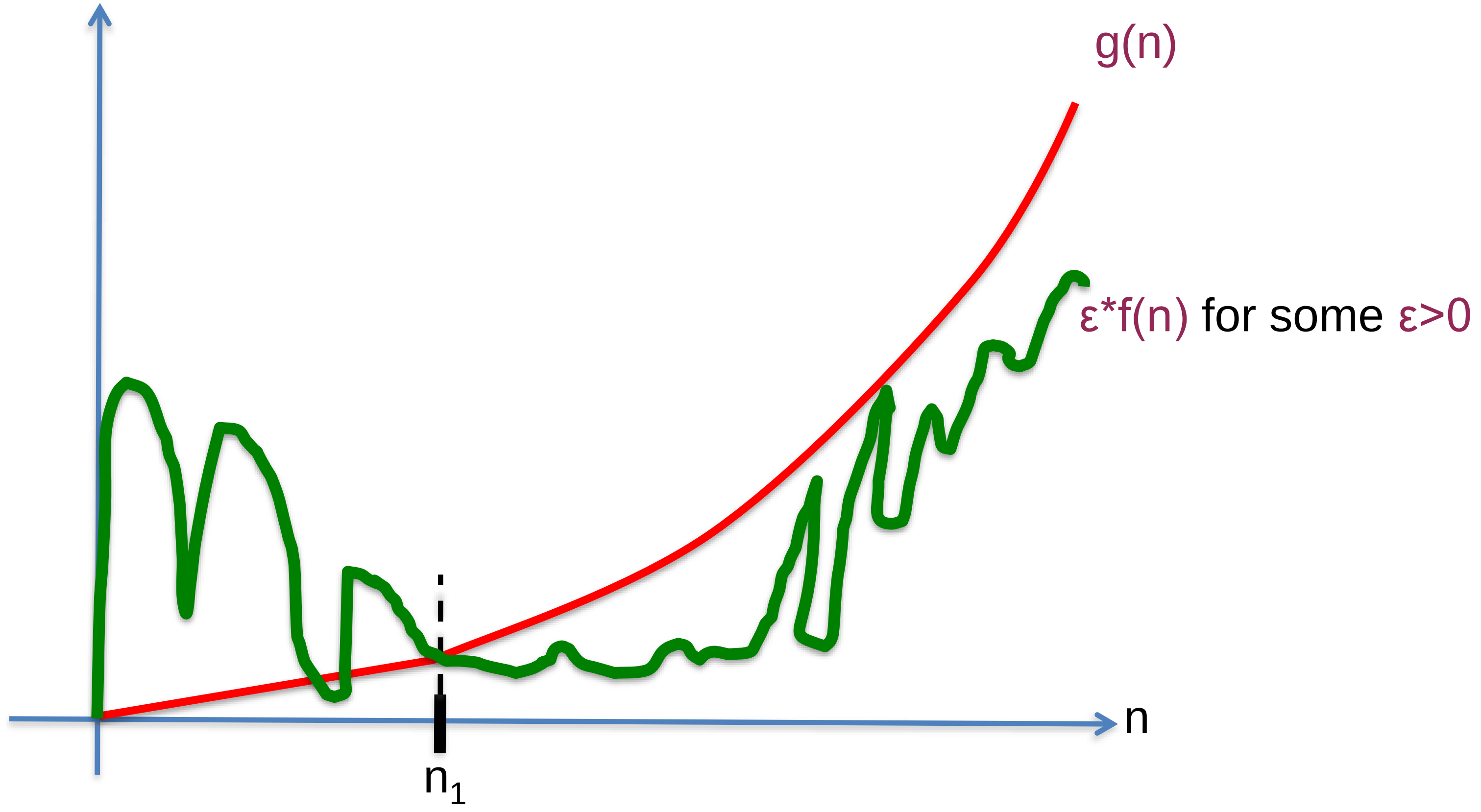
# The actual run times



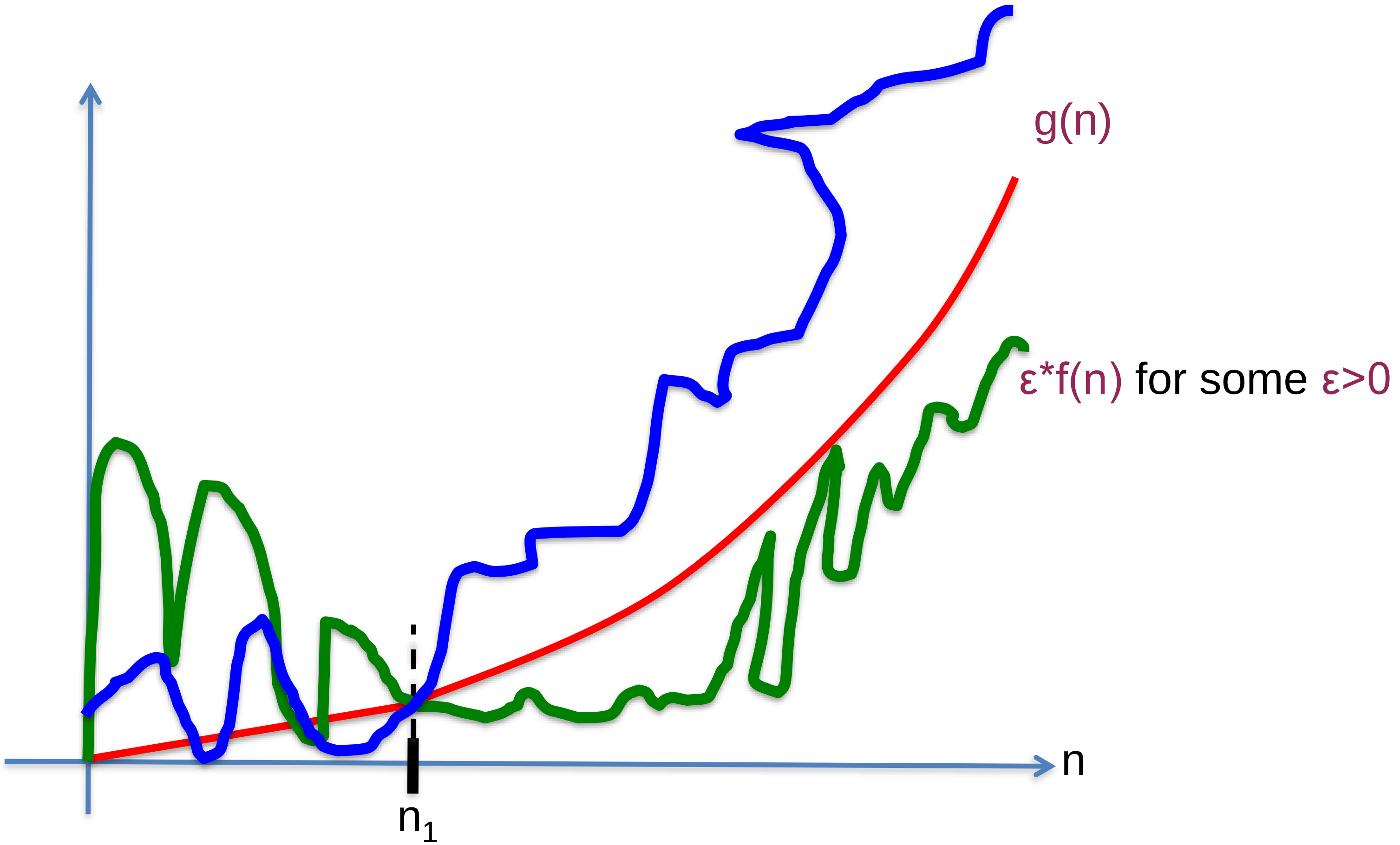
# g(n) is O(f(n))



# g(n) is Ω(f(n))



# g(n) is Θ(f(n))



# Properties of O (and Ω)

**Transitive**

g is O(f) and f is O(h) then
g is O(h)

Step 1 // O(n) time
Step 2 // O(n) time

Overall:
O(n) time

**Additive**

g is O(h) and f is O(h) then
g+f is O(h)

**Multiplicative**

g is $O(h_1)$ and f is $O(h_2)$ then
g*f is $O(h_1*h_2)$

While (loop condition) // $O(n^2)$ iterations
Stuff happens // O(1) time

Overall:
$O(n^2)$ time

# Another Reading Assignment

CSE 331 Support Pages

# Analyzing the worst-case runtime of an algorithm

Some notes on strategies to prove Big-Oh and Big-Omega bounds on runtime of an algorithm.

## The setup

Let $\mathcal{A}$ be the algorithm we are trying to analyze. Then we will define $T(N)$ to be the worst-case run-time of $\mathcal{A}$ over all inputs of size $N$. Slightly more formally, let $t_{\mathcal{A}}(\mathbf{x})$ be the number of steps taken by the algorithm $\mathcal{A}$ on input $\mathbf{x}$. Then

$$T(N) = \max_{\mathbf{x}:\mathbf{x} \text{ is of size } N} t_{\mathcal{A}}(\mathbf{x}).$$

In this note, we present two useful strategies to prove statements like $T(N)$ is $O(g(N))$ or $T(N)$ is $\Omega(h(N))$. Then we will analyze the run time of a very simple algorithm.

## Preliminaries

We now collect two properties of asymptotic notation that we will need in this note (we saw these in class today).

Sections 1.1, 1.2, 2.1, 2.2 and 2.4 in [KT]

# Gale-Shapley Algorithm

Intially all men and women are free

While there exists a free woman  who can propose

Let w be such a woman and m be the best man she has not proposed to

    w proposes to m

    If m is free

        (m,w) get engaged

    Else (m,w') are engaged

        If m prefers w' to w

            w remains free

        Else

            (m,w) get engaged and w' is free

Output the engaged pairs as the final output

# Implementation Steps

How do we represent the input?

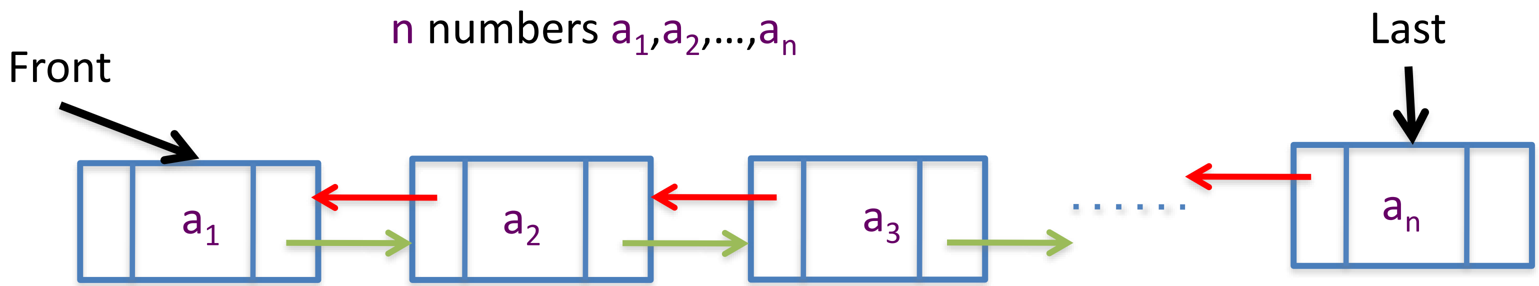
How do we find a free woman w?

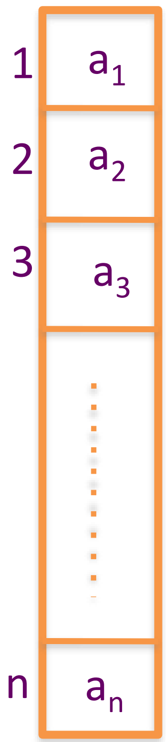
How would w pick her best unproposed man m?

How do we know who m is engaged to?

How do we decide if m prefers w' to w?

# Arrays and Linked Lists




| | Array | Linked List |
|---|---|---|
| Access ith element | O(1) | O(i) |
| Is e present? | O(n) (O(log n) if sorted) | O(n) |
| Insert an element | O(n) | O(1) given pointer |
| Delete an element | O(n) | O(1) given pointer |
| Static vs Dynamic | Static | Dynamic |

# Rest on the board…