Robustness and speed

Prof. Noah Snavely
CS1114
http://cs1114.cs.cornell.edu

Cornell University
Computer Science

Administrivia

- Assignment 1 is due next Friday by 5pm
  - Lots of TA time available (check the web)
- For grading, please sign up for a demo slot
  - These will be posted to CMS soon

Administrivia

- Please be careful with the robots
  - As you can tell, they are easy to break, cause smoke to come out of, etc.
  - Proper care and feeding manual coming soon…

What's the difference between = and == ?

- Answer: a lot

- = : assignment

```
x = 2;
```

- == : test for equality

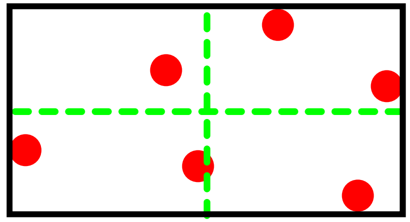
```
if x == 2
    y = 4;
end
```

- … very important not to get these mixed up

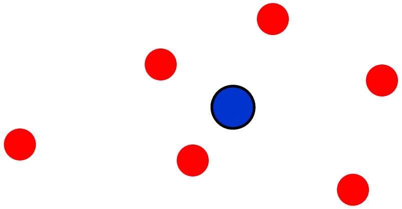
Finding the lightstick center

- Last time: two approaches

Bounding box



Centroid



- Both have problems…

How can we do better?

- What is the average weight of the people in this kindergarten class photo?

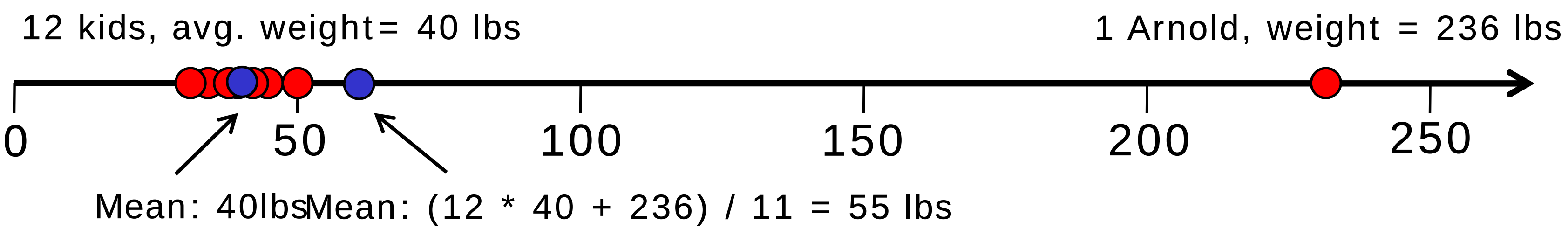
12 kids, avg. weight= 40 lbs

1 Arnold, weight = 236 lbs

Mean: (12 x 40 + 236) / 13 = 55 lbs

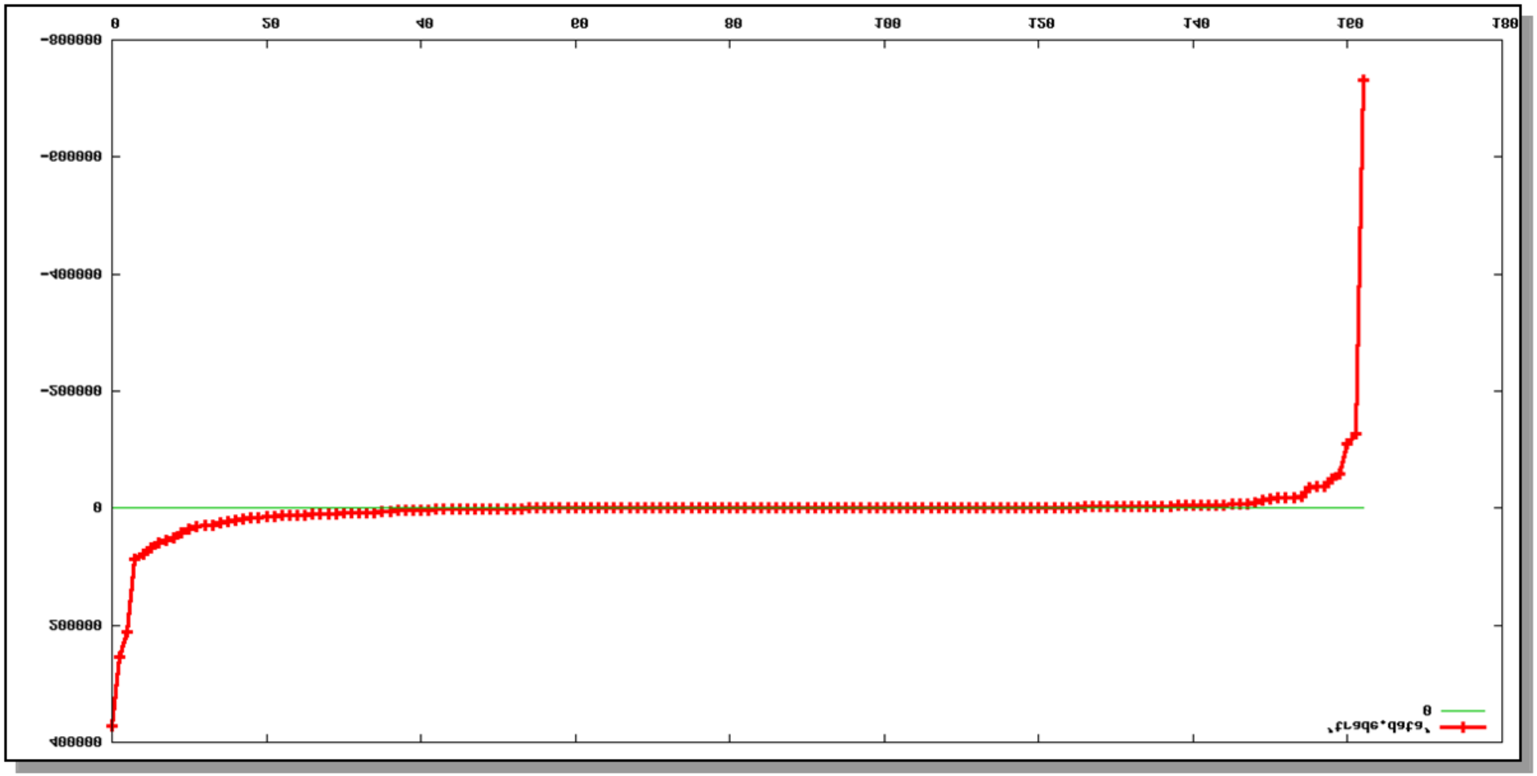
How can we do better?

- Idea: remove maximum value, compute average of the rest



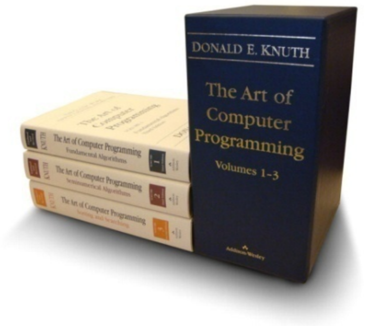
How can we do better?

- Trade deficit by country



How can we avoid this problem?

- Consider a simple variant of the mean called the “trimmed mean”
  - Simply ignore the largest 5% and the smallest 5% of the values
  - Q: How do we find the largest 5% of the values?



D.E. Knuth, *The Art of Computer Programming*
Chapter 5, pages 1 – 391

Easy to find the min (or max)

```
A = [11 18 63 15 22 39 14 503 20];
m = -1;  % Why -1?
for i = 1:length(A)
    if (A(i) > m)
        m = A(i);
    end
    % Alternatively: m = max(m,A(i));
end
```

How to get top 5%?

- First, we need to know how many cells we're dealing with
  - Let's say there are 100 → remove top 5
- How do we remove the biggest 5 numbers from an array?

Removing the top 5% -- Take 1

```
% A is a vector of length 100
for i = 1:5
    % Find the maximum element of A
    %     and remove it
end
```

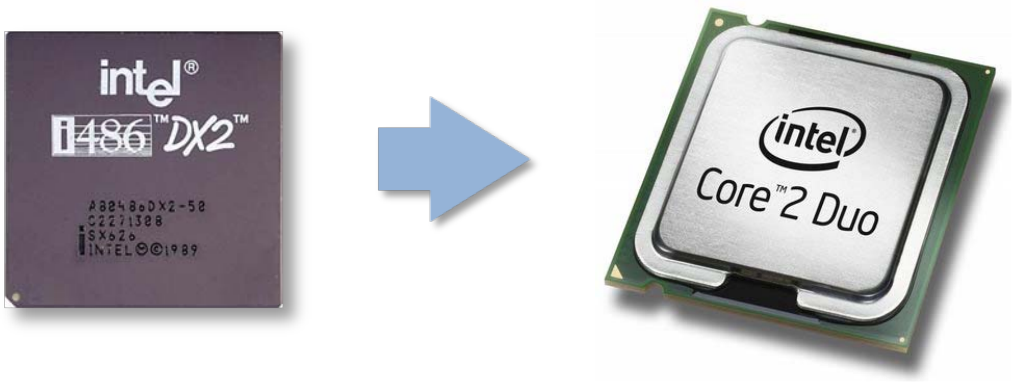
How good is this algorithm?

```
% A is a vector of length 100
 for i = 1:5
     % Find the maximum element of A
     %    and remove it
 end
```

- Is it correct?
- Is it fast?
- Is it *the fastest* way?

How do we define fast?

- It's fast when **`length(A) = 20`**
- We can make it faster by upgrading our machine



- So why do we care how fast it is?
- What if **`length(A) = 6,706,993,152`** ?

How do we define fast?

- We want to think about this issue in a way that doesn't depend on either:
  A. Getting really lucky input
  B. Happening to have really fast hardware

Where we are so far

- Finding the lightstick
  - Attempt 1: Bounding box (not so great)
  - Attempt 2: Centroid isn't much better
  - Attempt 3: Trimmed mean
    - Seems promising
    - But how do we compute it quickly?
    - The obvious way doesn't seem so great...
    - But do we really know this?

How fast is this algorithm?

- An elegant answer exists
- You will learn it in later CS courses
  - But I'm going to steal their thunder and explain the basic idea to you here
  - It's called "big-O notation"

- Two basic principles:
  - Think about the average / worst case
    - Don't depend on luck
  - Think in a hardware-independent way
    - Don't depend on Intel!

A more general version of trimmed mean

- Given an array of n numbers, find the k^{th} largest number in the array
- Strategy:
  - Remove the biggest number
  - Do this k times
  - The answer is the last number you found

Performance of our algorithm

- What value of *k* is the worst case?
  - ~~$k = n$~~ we can easily fix this
  - $k = n/2$

- How much work will we do in the worst case?

1. Examine *n* elements to find the biggest
2. Examine *n*-1 elements to find the biggest

   ... keep going ...

n/2. Examine *n*/2 elements to find the biggest

How much work is this?

- How many elements will we examine in total?

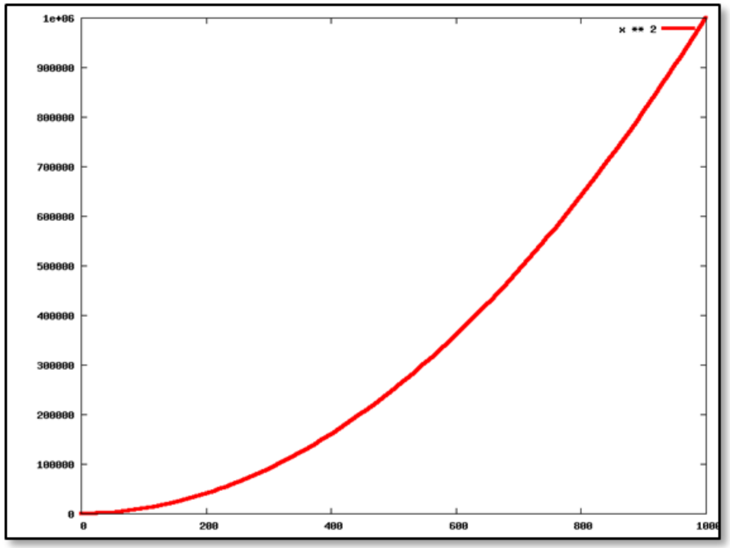
$$\underbrace{n + (n-1) + (n-2) + \ldots + n/2}_{n\,/\,2 \text{ terms}}$$

$$= ?$$

- We don't really care about the exact answer
  - It's bigger than $(n / 2)^2$

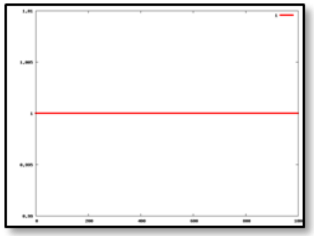
How much work is this?

- The amount of work grows *in proportion* to n^2

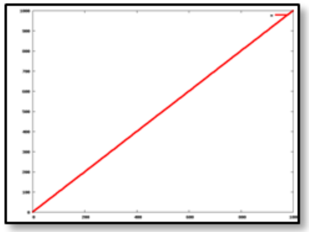


- We say that this algorithm is $O(n^2)$

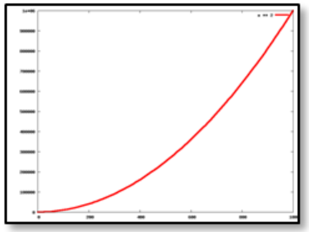
Classes of algorithm speed



- Constant time algorithms, $O(1)$
  - Do not depend on the input size
  - Example: find the first element

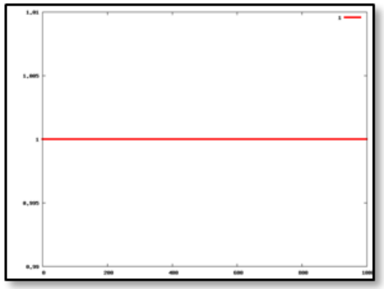


- Linear time algorithms, $O(n)$
  - Constant amount of work for every input item
  - Example: find the largest element

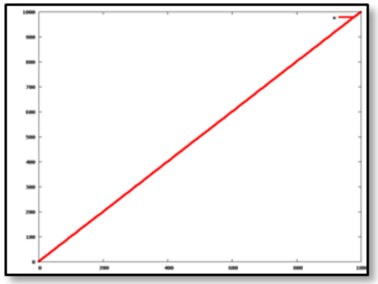


- Quadratic time algorithms, $O(n^2)$
  - Linear amount of work for every input item
  - Example: dumb median method

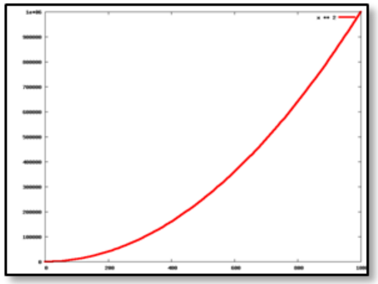
Asymptotic analysis picture



$O(1)$



$O(n)$



$O(n^2)$

- Different hardware only affects the parameters (i.e., line slope)
- As n gets big, the “dumber” algorithm by this measure always loses eventually