Algorithms

Part 2: Measuring Time

Notes for CSC 100 - The Beauty and Joy of Computing
The University of North Carolina at Greensboro

Reminders

Reading:

Emma reading (+ videos) - Reading Reflection due Mon 9/25

Has two short embedded videos - watch these too!

Homework 2:

Due Wednesday, 9/27 - practice for the midterm!

Lab 6:

Pre-Lab work before Friday

Importance of Understanding Algorithms

Algorithms have been studied for thousands of years

Intensity of study has exploded in last few decades

Why?

Speed of Electronic Computers

People compute at 1-2 medium-sized multiplications (5 digit) per minute

In 1965, IBM shipped the first IBM System/360 (model 40):

- 133,300 fixed-point additions/sec
- 12,000 fixed-point multiples/sec

Project manager was Fred Brooks - Professor at UNC
(was chair of UNC Dept of Computer Science for 20 years)

Question: How fast are the fastest computers now?

Speed of Electronic Computers

People compute at 1-2 medium-sized multiplications (5 digit) per minute

In 1965, IBM shipped the first IBM System/360 (model 40):

- 133,300 fixed-point additions/sec
- 12,000 fixed-point multiples/sec

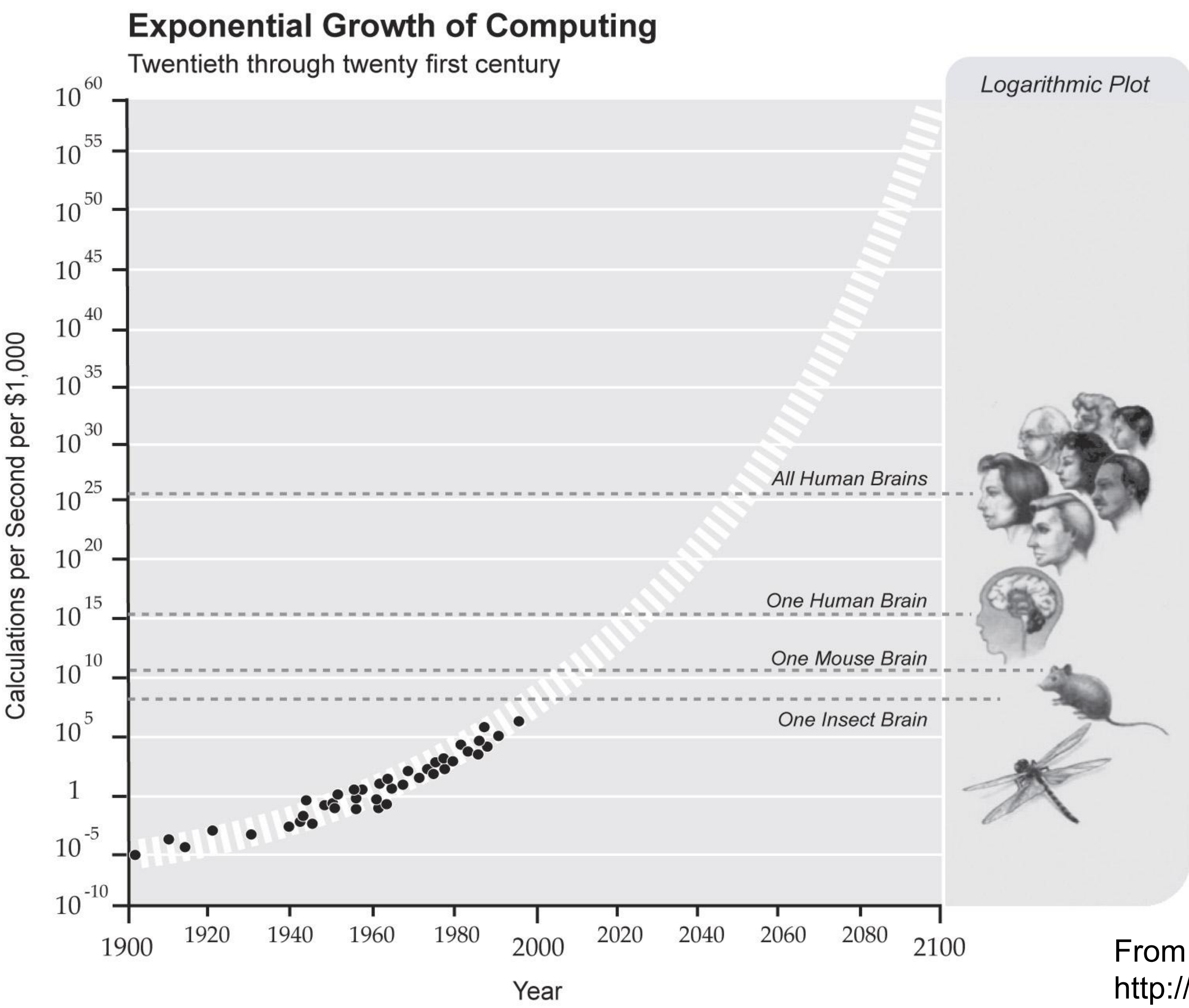
Project manager was Fred Brooks - Professor at UNC
(was chair of UNC Dept of Computer Science for 20 years)

In June 2017 the most powerful computer on earth could do 93,015,000,000,000,000 calculations per second (93.015 petaflops).

See http://www.top500.org/

Thinking about computations on this scale is incredibly different from thinking about computations at a few calculations per minute.

How Computing Power Has (and Will) Grow



Example from 2012:

$400 computer, 2.71 Gflops

Approx. 6.8 x 10^9 ops/$1000

... and
... can be more cost effective

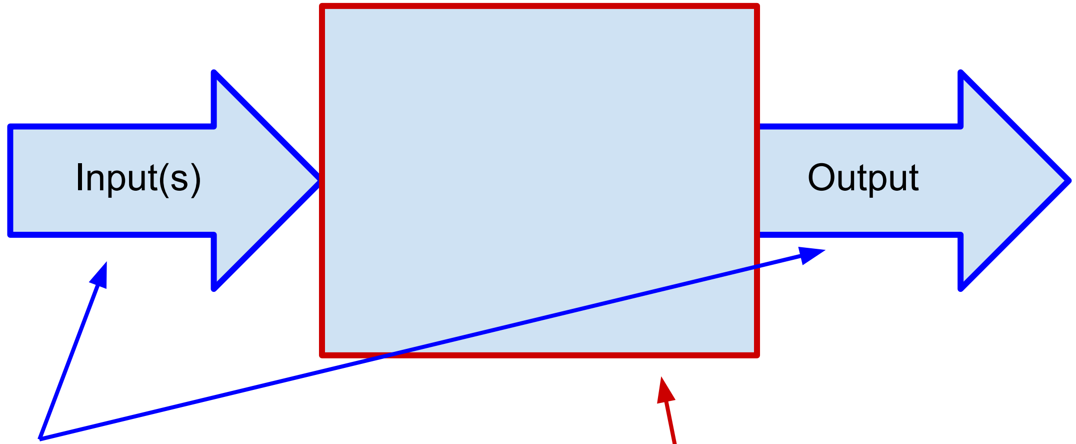
Moore's Law: Computing power doubles every 1.5 years (or 2 years, depending on version)

From:
http://www.singularity.com/charts/page70.html

Last Time We Saw...

Problems are defined by input/output relation, with no reference to how they are solved (*focus is on what*)

Algorithms are well-defined computational procedures that solve problems (*focus is on how*)



Problem specifier worries about input and outputs

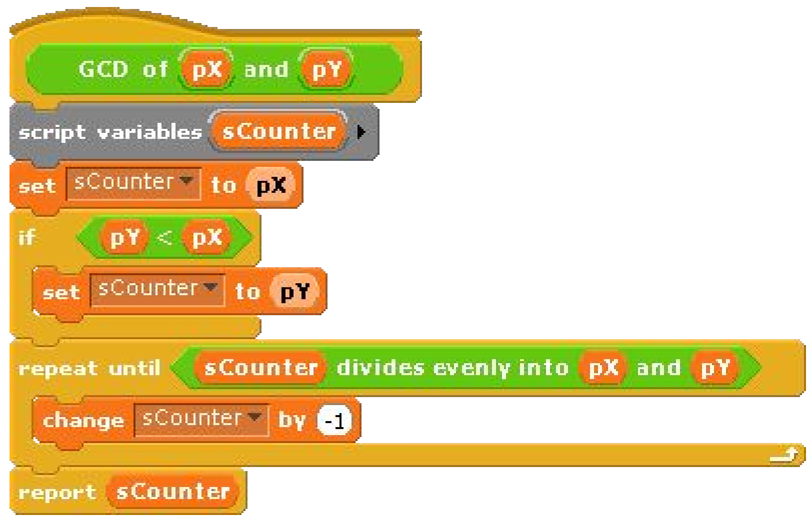
Implementer / algorithm designer worries about the computational process

In Snap!

Problem Focus

With a well-chosen name, that may define the problem well enough for the user!

Algorithm Focus



This is an over-simplification: Sometimes the user wants to know some properties of the block implementation.

Question: What kinds of properties?

Algorithm Characteristics

- Does the algorithm work correctly (does it solve the problem)?
- Is the answer provided precise?
- How confident are you in the correctness of the algorithm and implementation (simpler algorithms are easier to verify)?
- How much memory does the algorithm require?
- How fast is the algorithm?

Algorithm Characteristics

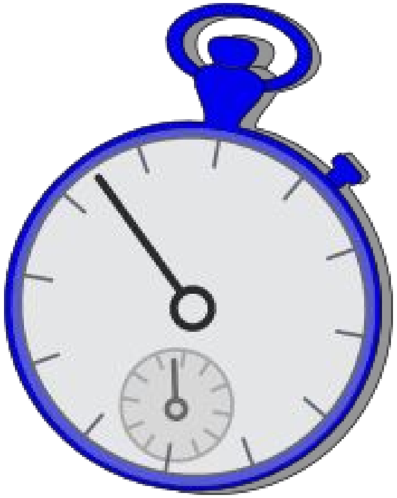
- Does the algorithm work correctly (does it solve the problem)?
- Is the answer provided precise?
- How confident are you in the correctness of the algorithm and implementation (simpler algorithms are easier to verify)?
- How much memory does the algorithm require?
- How fast is the algorithm?

Assume no problems with correctness or precision for now.

Memory is a problem for some algorithms, but not as common a limiting factor as...

Time is usually the most interesting and limiting characteristic, whether talking about running a big computation for a week, or calculating a new graphics frame in 1/30 of a second.

What is "time" for an Algorithm?



Time is time, right?

But...

- Does time depend on things other than the algorithm?
- If run many times (on the same input), is time always the same?
- If QuickSort runs in 20 seconds on my old IBM PC, and SelectionSort runs in 0.5 seconds on my current computer, is SelectionSort a faster algorithm?
- Can we give clock time without implementing the algorithm?

Correcting for vagueness of timing

Wall-clock times depend on:

- Speed of computer that it's run on
- What else is happening on the computer
- ... and a few other things we'll address later

But... these are not differences in algorithms!

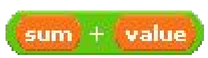
Solution: Algorithms are sequences of steps, so count steps!

Question: We discussed steps earlier - so what's a step?

Snap! blocks and "steps"

Which of these should not be treated as "one step"?

a) 

b) 

c) 

d) 

e) 

Experimenting with timing Snap! scripts

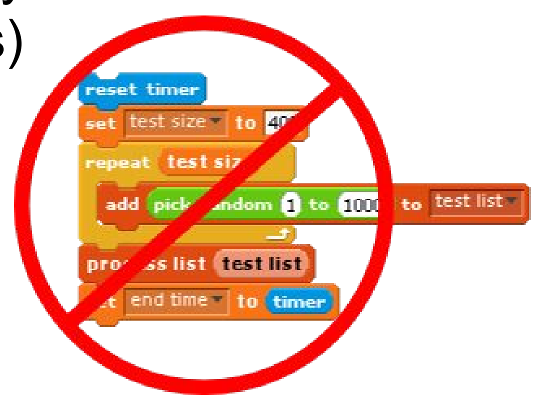
Timer is available to help test things out

- Reset timer to start it at zero

- Save current timer value into a variable for "lap timer"

- Watch variable shows limited precision - for more use "say

- Tip: surround only what you're interested in timing with reset/set blocks (not initializations)



Summary

Time is one of the most important algorithm characteristics

An “algorithm” should be independent of what runs it

→ So measure time in steps, not seconds

But - when you want time in seconds for a specific implementation, Snap! gives you tools to measure that.