Algorithms

Part 1: The Basics

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

Definition

From Webster's dictionary:

algorithm. *noun.*

> *a procedure for solving a mathematical problem (as of finding the greatest common divisor) in a finite number of steps that frequently involves repetition of an operation;broadly : a step-by-step procedure for solving a problem or accomplishing some end especially by a computer*

Another definition (from Dan Garcia, UC Berkeley bjc class): An algorithm is a *well-defined computational procedure* that takes some value or set of values as *input* and produces some value or set of values as *output*.

Algorithms we've seen...

Integer division (division without remainder) - Lab 2
Max of two numbers - Lab 3 (pre-lab reading)
Adding a sequence of numbers - Lab 4 (pre-lab reading)
Greatest common divisor - Lab 4
Adding numbers in a list - Lab 5
Swapping values in a list - Lab 5
Searching for a value in a list - Lab 5
Finding the largest value in a list (high score) - Lab 5

Some Famous algorithms

JPEG Image Compression

10 megapixel image
- 30 megabytes "raw"

6.5 MBytes at high quality
2.7 MBytes at reasonable q

Google Page Rank

Orders search results
- "algorithm": 112,000,000 results
- how did it find that so fast?
- why are first listed so relevant?

Fast Fourier Transform

Signal processing
Time to frequency domain
Used in MP3
- 100x faster than "obvious" alg

Dijkstra's Shortest Path

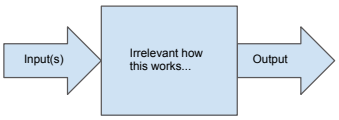
How can Mapquest find directions so quickly?

Imagine how many possible paths - can't try them all!

Problems vs Algorithms: Problems

A *problem* describes input/output behavior

- For each input, what are correct output(s)?
- How outputs are obtained is irrelevant - could be magic!



Examples:

- Given two integers, what is greatest common divisor?
- Given a list of numbers, what is the largest element of a list?
- Given a set of points in space, which two points are closest?
- Given a list of numbers, output the same numbers in sorted order.

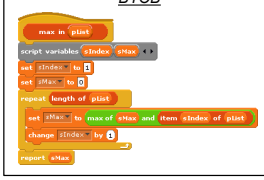
Problems vs Algorithms: Algorithms

An *algorithm* gives a well-defined sequence of steps to solve a problem (e.g., finding largest value in a list):

Pseudo-code

Start at the first element
For each element:
    Compare the element to the largest seen so far
    If larger, replace largest seen so far
Report largest seen so far

BYOB



Two representations of the same abstract concept

Algorithm Representations/Implementations

More examples:

Java

```
public int getMaxOfList(List<Integer> values)
{
    int maxValue = 0;
    for (int current : values)
        if (current > maxValue)
            maxValue = current;

    return maxValue;
}
```

Python

```
def maxValue(list):
   maxValue = 0
   for i in range(len(v)):
      if v[i] > maxValue: maxValue = v
[i]
   return maxValue
```

Disclaimer: Not how a Python programmer would do this....

Notes:

- Some representations are good for people to understand (pseudo-code)
- Some are good for computers to understand
- Efficient translation between representations is important!

But all the same algorithm!

- Algorithms are fundamental and transcend any fad in languages or representation

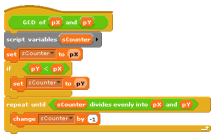
Algorithms for Problems

An algorithm is *correct* if the output it produces satisfies the problem definition.

- Also say the algorithm *solves* the problem

Problems often have multiple algorithms that solve them - example, GCD:

From Lab 4



Euclid's GCD algorithm (300 B.C.)

Both of these algorithms solve the GCD problem
Should we prefer one over the other?

Comparing GCD Algorithms

OK, let's try it in BYOB!

Algorithms: Choices, choices, choices...

Different algorithms for a problem have different properties
- Choosing the right algorithm is a matter of trade-offs

Question: What trade-offs can you think of for algorithms?

Examples of Algorithm Trade-offs

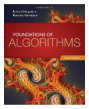
Can consider
- Simplest algorithm
- Easiest to implement
- Fastest running time
- Uses least amount of memory
- Gives most precise answer

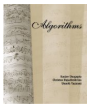
Question: Which of these is most important?

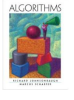
A Deep and Rich Area of Study

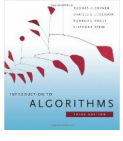
The study of algorithms is about two things:
- Problem solving techniques
- Considering trade-offs

Many books devoted to study of algorithms...









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

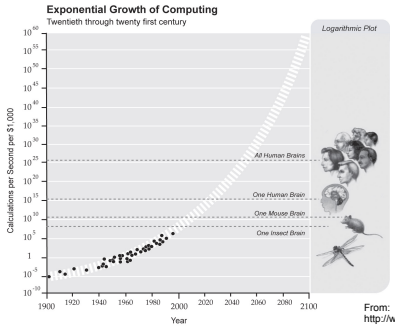
Project manager was Fred Brooks - Professor at UNC
(was chair of UNC Dept of Computer Science for 20 years)

In June 2012 the most powerful computer on earth could do 16,320,000,000,000,000 calculations per second (16.32 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)

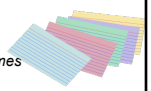
From:
http://www.singularity.com/charts/page70.html

A Flood of Data

It's not all about calculations per second either...

Consider the amount of data we deal with:

- Human genome: Just over 3 billion base pairs
  - *Typing in 12pt on 8.5x11 paper fits 2880 characters*
  - *So the human genome would be over a million pages (printed two-sided, an 86 foot high stack of paper)*
- Facebook (source: http://thesocialskinny.com/100-social-media-statistics-for-2012/)
  - *Around a billion users*
  - *Around 420 million status updates per day*
  - *On index cards, would be a stack 53 miles high!*
  - *... or end-to-end would stretch around the world 1.3 times*
- Large Synoptic Survey Telescope
  - *16 terabytes (16,000,000,000,000 bytes) will be captured per day*
  - *No human being will ever see most of this data*



Summary

Main points:

- Algorithms solve problems
- One problem may have many algorithmic solutions
- Choice of algorithm depends on trade-offs
- Scale of algorithm use is like nothing before in history
- Lots of work to get good at designing, analyzing, and selecting algorithms

But worth it: Algorithms are changing the world!

Looking ahead...

Next time:

How do we talk about / analyze speed of algorithms?

Optional video - how algorithms are taking over the world:

http://www.ted.com/talks/lang/en/kevin_slavin_how_algorithms_shape_our_world.html