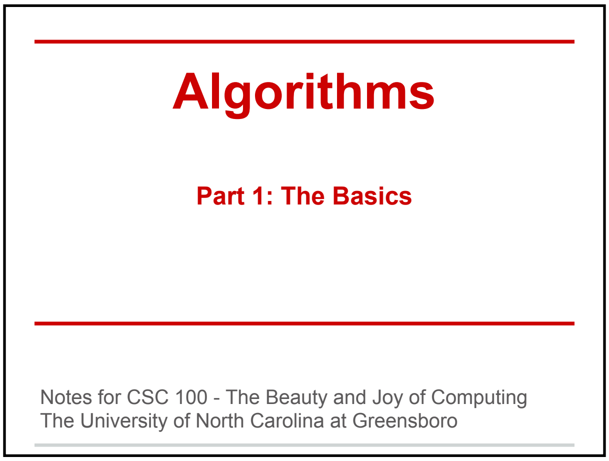
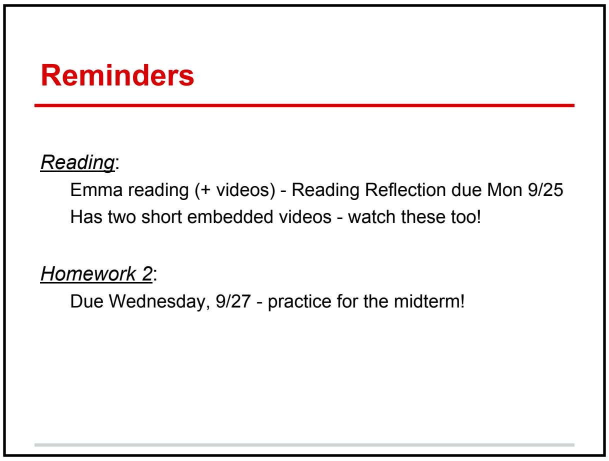


Algorithms

Part 1: The Basics

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



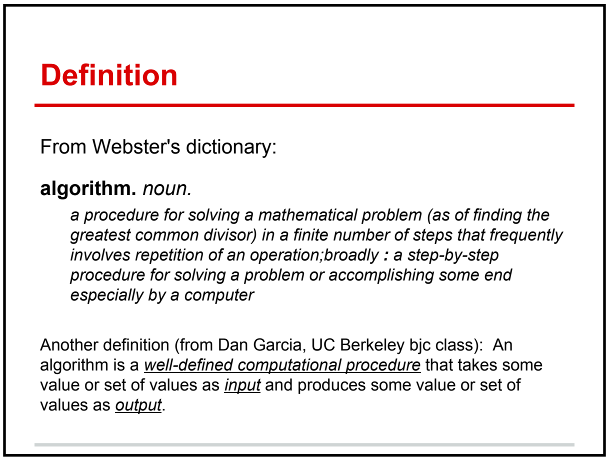
Reminders

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!



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) - “div”
Max of two (or three) numbers - Lab 3 (pre-lab reading)
Converting a value to a hexadecimal digit - Lab 4
Converting numbers to/from other bases - Lab 4
Adding up values in a list - Lab 5
Creating a list of a certain size (“range”) - Lab 5

Shuffling a list - Lab 5 (this one’s a little different!)

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": 154,000,000 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

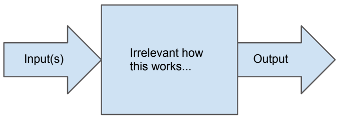
How can Mapquest find directions so quickly?

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.

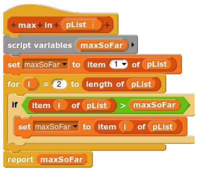
Problems vs Algorithms: Algorithms

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
```

Snap!



```
+max+in+ pList ⋮ +
script variables maxSoFar
set maxSoFar to item 1 of pList
for i = 2 to length of pList
    if item i of pList > maxSoFar
        set maxSoFar to item i of pList
report maxSoFar
```

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:

Challenge

Give an algorithm for computing greatest common divisors

For example: How would you compute the GCD of 10 and 15?

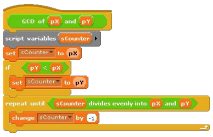
Algorithms for Problems

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

- Also say the algorithm *solves* the problem

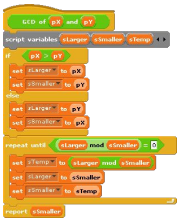
Problems often have multiple algorithms that solve them - example, GCD:

The obvious algorithm



```
GCD of pX and pY
script variables sCounter
set sCounter to pX
if pY < pX
    set sCounter to pY
repeat until sCounter divides evenly into pX and pY
    change sCounter by -1
```

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



```
GCD of pX and pY
script variables sLarger sSmaller sTemp
if pX > pY
    set sLarger to pX
    set sSmaller to pY
else
    set sLarger to pY
    set sSmaller to pX
repeat until sLarger mod sSmaller = 0
    set sTemp to sLarger mod sSmaller
    set sLarger to sSmaller
    set sSmaller to sTemp
report sSmaller
```

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

Comparing GCD Algorithms

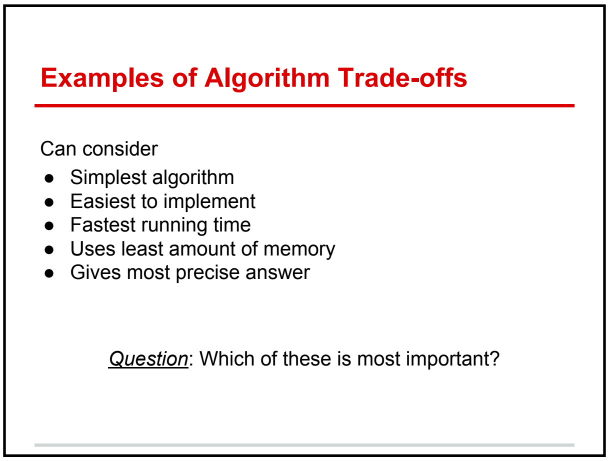
OK, let's try it in Snap!

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?

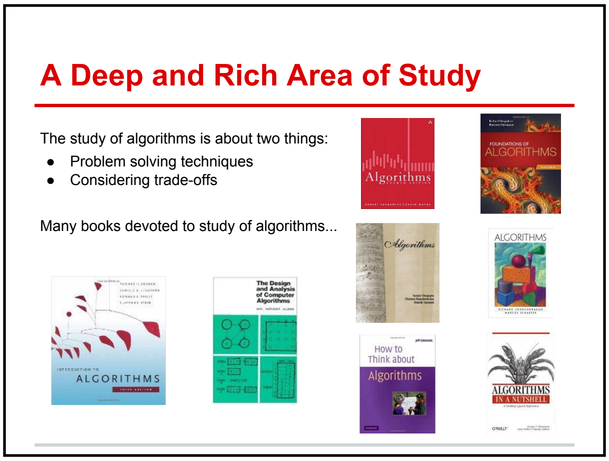


Examples of Algorithm Trade-offs

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?



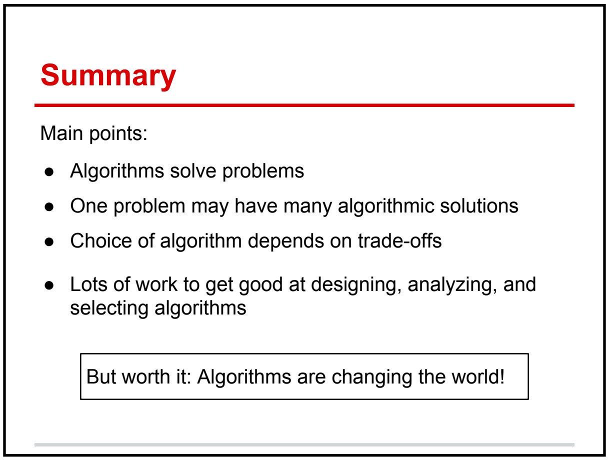
A Deep and Rich Area of Study

The study of algorithms is about two things:

- Problem solving techniques
- Considering trade-offs

Many books devoted to study of algorithms...

Algorithms
FOUNDATIONS OF ALGORITHMS
Algorithms
ALGORITHMS
INTRODUCTION TO ALGORITHMS
The Design and Analysis of Computer Algorithms
How to Think about Algorithms
ALGORITHMS IN A NUTSHELL

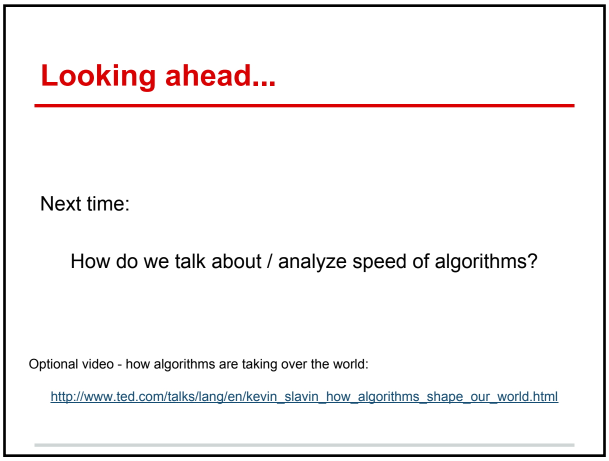


Summary

Main points:

- Algorithms solve problems
- One problem may have many algorithmic solutions
- Choice of algorithm depends on trade-offs
- 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