Algorithmic Thinking

Chapter 10

Exercise

- Write instructions to make a peanut butter and jelly sandwich.

- You have:
  - jar of peanut butter
  - jar of jelly
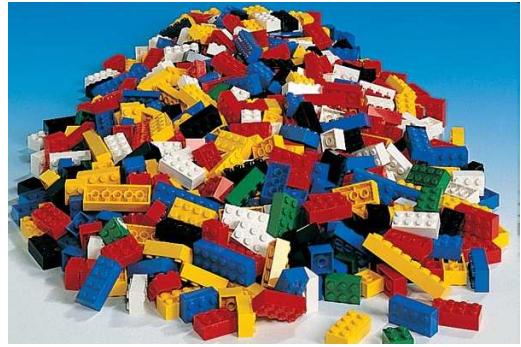 - bagged loaf of bread
  - butter knives

Algorithm

- **al·go·rithm**: a step-by-step procedure for solving a problem

- **program**: instructions for a computer; an algorithm expressed in a *programming language* (vs. *natural language*)
  - The programming language we will use in this class is called JavaScript.

Programming is just like Legos...

Why Programming Is Hard

- Programming is a fairly detail-oriented task. There is no room for mistakes.
  - Did you forget a semi-colon?
  - Did you use the wrong capitalization?
  - If you keep making the same small mistakes, you will get easily frustrated.
    - Learn from your mistakes and make sure to minimize them.

- Programming requires the ability to think logically.

8

Logic Test

- Computers are stupid.
- Computers can't read minds.
- Computers don't make mistakes.

- Therefore, if your computer program is not doing what you want …

  it is because **YOU** made a mistake!

9

Exercise

- You have a list of items and their retail prices. Write an algorithm to determine which item is the most expensive.

10

Exercise

- Write an algorithm to sort a list of one million numbers.
  - How do you start thinking up a solution?
    - Start small: How do you sort a list of four numbers?
    - Experience helps: How were similar problems solved?

I think the bubble sort would be the wrong way to go.

Barack Obama, when asked by Google CEO
Eric Schmidt about "the most efficient way to
sort a million 32-bit integers", 11/14/2007

11

Algorithmic Efficiency

- Not all algorithms are created equal.
- The most obvious solution is not necessarily the most efficient.
  - Could be the difference between an instant and waiting a lifetime (or more!)
  - If we wait long enough, will the question of algorithm efficiency disappear as faster and faster computers come into existence?
    - No. There will always be bigger problems to solve with more and more data.
- In this class, do not consider efficiency when writing programs—stick with the most obvious algorithm.

12

2