Today

- Binary!
- Intro to algorithms and pseudocode
- Discussion of readings

Reminders:

- Leading discussions
- Blog posts!
- Bring laptops with face project ready to show off on Thursday.

Telling a computer how to behave:
intro to algorithms + pseudocode

Many slides from Sanjeev Arora

Steps in solving a computational task

- Design an algorithm: A precise,unambiguous procedure for solving a computational task. (We will express our algorithms in pseudocode.)
- Turn pseudocode into computer program.

Programming a robot

- Your robot needs to get ready for a big date… and wants to purchase a bottle of perfume.

- Specifically, it needs to identify the cheapest bottle. (Say it can scan prices)

Rules

- Can pick up a cup
- Can compare the price on the cup in hand with the price of a cup on the table.
- Can swap the cup in hand with a cup on the table.
- Should stop when the cup in hand is guaranteed to be the minimum priced one.

- give all the instructions at the beginning.
- should work no matter how many cups there are.

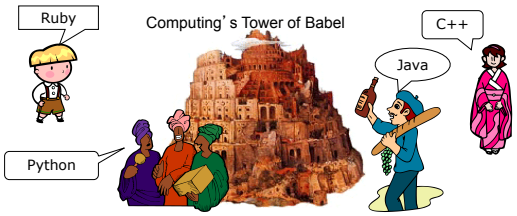
Solution

- Pick up first bottle, check price
- Walk down aisle. For each bottle, do this:
  - If price on bottle is less than price in hand, exchange for one in hand.

How can we describe an algorithm precisely enough so there is no ambiguity?

We will do this using "pseudocode".

Processing language illustrates essential features of all computer languages

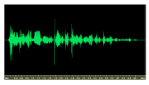


- Fundamental features of human languages:nouns/verbs/adjectives, subjects/objects, pronouns, etc.
- Computer languages also share fundamental features, e.g. conditional and loop statements, variables, **ability to perform arithmetic**, etc.

Why is arithmetic so important for computers?

Because to a computer **everything is a number!**

Audio waveform



Sequence of Numbers representing frequency, amplitude, etc.

Image

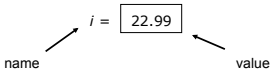
Sequence of Numbers representing red/green/blue color value of each pixel.

Now let's solve the problem as it would be done on a computer

- There are *n* prices stored in memory
- Want to find minimum price

Memory

- A simplified view of computer memory: a scratchpad that can be perfectly erased and re-written any number of times
- A variable: a piece of memory with a name; stores a "value"



Examples

$i = 5$	Sets *i* to value 5
$j = i$	Sets *j* to whatever value is in *i*. Leaves *i* unchanged
$i = j + 1$	Sets *i* to *j* + 1. Leaves *j* unchanged
$i = i + 1$	Sets *i* to 1 more than it was.

Arrays

- *A* is an array of *n* values, *A*[*i*] is the *i*th value

A =	40.99	62.99	52.99	···	22.99

- Example: *A*[3] is 52.99

Pseudocode

- Variables and arrays
- Assignment
- Simple instructions: involve +, -, ×, ÷ ,...
- Tests for ==, < , > ,...
- Compound instructions
  - Conditionals
  - Loops

Now we can express our solution in pseudocode

- Pick up first bottle, check price
- Walk down aisle. For each bottle, do this:
  - If price on bottle is less than price in hand, exchange for one in hand.

Procedure findmin (in pseudocode)

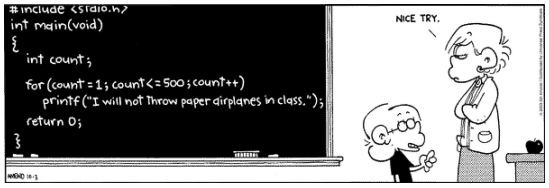
- *Input: n* values, stored in array *A*
- Variables are *i*, *best*
- *best* ← 1
- for (*i* = *2* to *n*)

```
{
    if ( A[ i ] < A[best] ) then
        { best = i }
}
```

Output *A[best]*.

Another way to do the same

```
best = 1;
i = 1
while (i < n)
{
        i = i + 1;
        if ( A[ i ] < A[best] ) then
                { best ← i }
}
```



New problem for robot: sorting

Arrange them so prices increase from left to right (or top to bottom).

Solution

Do for *i*=1 to *n*-1
{
  Find cheapest bottle among those numbered *i* to *n*

  Swap that bottle and the *i*' th bottle.
}

"selection sort"

Exercise for Thursday: Try to write pseudocode for selection sort.

Swapping

- Suppose *x* and *y* are variables. How do you swap their values?
- Need extra variable!

tmp ← *x*
x ← *y*
y ← *tmp*

Efficiency of this sorting algorithm?

Measure the efficiency of an algorithm in terms of the

number of elementary operations it performs

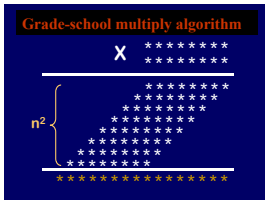
Find a number in a sorted list, if it's there

- Underneath each cup is a ping pong ball.
- On each ping pong ball is a number.
- The numbers are in sorted order.
- Goal: to look at as few ping pong balls as possible!

Efficiency Matters

Table 2.1 The running times (rounded up) of different algorithms on inputs of increasing size, for a processor performing a million high-level instructions per second. In cases where the running time exceeds 10^{25} years, we simply record the algorithm as taking a very long time.

	n	$n \log_2 n$	n^2	n^3	1.5^n	2^n	$n!$
$n = 10$	< 1 sec	< 1 sec	< 1 sec	< 1 sec	< 1 sec	< 1 sec	4 sec
$n = 30$	< 1 sec	< 1 sec	< 1 sec	< 1 sec	< 1 sec	18 min	10^{25} years
$n = 50$	< 1 sec	< 1 sec	< 1 sec	< 1 sec	11 min	36 years	very long
$n = 100$	< 1 sec	< 1 sec	< 1 sec	1 sec	12,892 years	10^{17} years	very long
$n = 1{,}000$	< 1 sec	< 1 sec	1 sec	18 min	very long	very long	very long
$n = 10{,}000$	< 1 sec	< 1 sec	2 min	12 days	very long	very long	very long
$n = 100{,}000$	< 1 sec	2 sec	3 hours	32 years	very long	very long	very long
$n = 1{,}000{,}000$	1 sec	20 sec	12 days	31,710 years	very long	very long	very long



Aside: History of Algorithm

- Named for Abu Abdullah Muhammad bin Musa al-Khwarizmi (780-850AD)
  - His book "Al-Jabr wa-al-Muqabilah" evolved into today's high school algebra text.
- Notion of algorithm has existed for at least 2000 years (in Hindu, Chinese, and Greek traditions)
- "Variables" in algebra come from the same tradition.

Algorithm defn; revisited

"Pseudocode for turning a set of inputs into outputs in a finite amount of time"

Questions to think about:

- What class of computational tasks can be solved by algorithms?
- How dependent is this class on the exact definition of pseudocode?