CSCI 3100: Algorithms

Today:
- Syllabus
- Intro to algorithms

---

**algorithm**

*noun*

Word used by programmers when they do not want to explain what they did.
Question: What is an algorithm?

(Side question: What is a program?)
Origins

- Not Greek "algos"?

- 9th century writer and mathematician Abu Abd Allah Muhammad ibn Mūsā Al-Khwārizmī

(also where "algebra" came from, invented 0 as a placeholder)

- Later known as "algorism", popularized by Fibonacci
The usual silly examples

**BottlesOfBeer(n):**
For \( i \leftarrow n \) down to 1
- Sing “\( i \) bottles of beer on the wall, \( i \) bottles of beer,"
- Sing “Take one down, pass it around, \( i - 1 \) bottles of beer on the wall.”
- Sing “No bottles of beer on the wall, no bottles of beer,"
- Sing “Go to the store, buy some more, \( n \) bottles of beer on the wall.”
of Arabic numerals; it was still taught in elementary schools in England in the 19th century. This algorithm was also commonly used by early digital computers for integer multiplication directly in hardware.

```
PeasantMultiply(x, y):

prod ← 0
while x > 0
    if x is odd
        prod ← prod + y
    x ← ⌊x/2⌋
    y ← y + y
return p
```

<table>
<thead>
<tr>
<th></th>
<th>x</th>
<th>y</th>
<th>prod</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>123</td>
<td>+ 456</td>
<td>456</td>
</tr>
<tr>
<td></td>
<td>61</td>
<td>+ 912</td>
<td>1368</td>
</tr>
<tr>
<td></td>
<td>30</td>
<td>1824</td>
<td></td>
</tr>
<tr>
<td></td>
<td>15</td>
<td>+ 3648</td>
<td>5016</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>+ 7296</td>
<td>12312</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>+ 14592</td>
<td>26904</td>
</tr>
<tr>
<td>1</td>
<td>+ 29184</td>
<td>56088</td>
<td></td>
</tr>
</tbody>
</table>

The peasant multiplication algorithm breaks the difficult task of general multiplication into a process of repeated doubling and halving.
\textbf{Construct the line perpendicular to }\ell\text{ and passing through }P.\textbf{)}

\textbf{\textsc{RightAngle}}(\ell, P):
- Choose a point }A \in \ell\text{ }
- \(A, B \leftarrow \text{Intersect}\text{\} (\text{Circle}(P,A), \ell)\text{\)}
- \(C, D \leftarrow \text{Intersect}\text{\} (\text{Circle}(A, B), \text{Circle}(B, A))\text{\)}
- \text{return Line}(C, D)

\textbf{\textsc{Construct a point }Z\text{ such that }|AZ| = |AC||AD|/|AB|.)\textbf{)}

\textbf{\textsc{MultiplyOrDivide}}(A, B, C, D):
- \(\alpha \leftarrow \text{\textsc{RightAngle}}\text{\} (\text{Line}(A, C), A)\text{\)}
- \(E \leftarrow \text{\textsc{Intersect}}\text{\} (\text{Circle}(A, B), \alpha)\text{\)}
- \(F \leftarrow \text{\textsc{Intersect}}\text{\} (\text{Circle}(A, D), \alpha)\text{\)}
- \(\beta \leftarrow \text{\textsc{RightAngle}}\text{\} (\text{Line}(E, C), F)\text{\)}
- \(\gamma \leftarrow \text{\textsc{RightAngle}}\text{\} (\beta, F)\text{\)}
- \text{return Intersect}(\gamma, \text{\} Line}(A, C))\text{\)}

Multiplying or dividing using a compass and straightedge.

\textsuperscript{4}In fact, some medieval English sources claim the Greek prefix “algo-” meant “art” or “introduction.
Now a bad example:

**BECOMEAMILLIONAIREANDNEVERPAYTAXES:**

Get a million dollars.
If the tax man comes to the door and says, “You have never paid taxes!”
Say “I forgot.”

Why is this bad?
Some tips for this class:

- Don’t write actual code!
  - essentially, goal is to write the comments that should be in the program

- But– don’t write English prose either!
  - use loops + data structures

- Indent and avoid brackets
More tips:

- Meaningful variable names, please!
- Keep a statement on a single line
- Find a good balance between words and math:
  
  Insert $x$ into $A$
  
  $\text{INSERT}(x, A)$
  
  $X \leftarrow X \cup \{a\}$
3 parts to every algorithm:

1.

2.

3.

+ sometimes 4:
This week: Why you should have paid attention in discrete math & data structures!

Topics to recall:
Runtimes:
What is big-O analysis?

Why use it?
Formal def:
Let $f$ and $g$ be functions $\mathbb{R} \to \mathbb{R}$ (or $\mathbb{Z} \to \mathbb{R}$). We say that:

$f(n) = O(g(n))$

if $\exists$ constants $C > 0$ such that:

$|f(n)| \leq C |g(n)|$

$\forall n > n_0$
Big-O: functions ranking

**BETTER**
- $O(1)$: constant time
- $O(\log n)$: log time
- $O(n)$: linear time
- $O(n \log n)$: log linear time
- $O(n^2)$: quadratic time
- $O(n^3)$: cubic time
- $O(2^n)$: exponential time

**WORSE**

---

Big-O Complexity

- $O(1)$
- $O(\log n)$
- $O(n)$
- $O(n \log n)$
- $O(n^2)$
- $O(n^3)$
- $O(2^n)$
Example proof:

\[ f(x) = x^2 + 2x + 1 \text{ is } O(x^2) \]

pf:
Key theorem:

Let \( f(x) \) be a polynomial of degree \( n \), so \( f(x) = \sum_{i=0}^{n} a_i x^i \)

where each \( a_i \in \mathbb{R} \).

Then \( f(x) = O(x^n) \).

pf sketch:
Induction: recursion’s twin

A method of proving a statement which depends on the statement being true for smaller values.

Required pieces:
Aside: I think of this as “automating” a proof:

Show true for $n=1$.

Show if $n$ holds, then $n+1$ must also.

$\Rightarrow$ Get all $n$ for free!
Example: \[ \sum_{i=0}^{n} i = \]
Next time:
- Recursion as an algorithmic technique
- Even more induction