ABOUT ALGORITHMS

1. Important Quotations

1.1. **What is an algorithm?** "An algorithm is a sequence of unambiguous instructions for solving a problem, i.e., for obtaining a required output from any legitimate input in a finite amount of time."

Or, to be precise, an algorithm is a **finite** sequence of unambiguous instructions, etc.

— Anany Levitin, page 3.

1.2. **Algorithmic Thinking.** "Computer Science and Engineering is a field that attracts a different kind of thinker. I believe that one who is a natural computer scientist thinks algorithmically. Such people are especially good at dealing with situations where different rules apply in different cases; they are individuals who can rapidly change levels of abstraction, simultaneously seeing things in-the-large and in-the-small." — Donald Knuth, as quoted in J. Hartmanis *On computational complexity and the nature of computer science* published in *Communications of the ACM 37*, 10 (1994), 39.

1.3. **More than Answers to Interview Questions.** As one of the giants in the history of computer science put it:

" A person well-trained in computer science knows how to deal with algorithms: how to construct them, manipulate them, understand them, analyze them. This knowledge is preparation for much more than writing good computer programs; it is a general-purpose mental tool that will be a definite aid to the understanding of other subjects, whether they be chemistry, linguistics, or music, etc. The reason for this may be understood in the following way:

It has often been said that a person does not really understand something until after teaching it to someone else. Actually, a person does not **really** understand something until after teaching it to a **computer**, i.e., expressing it as an algorithm."

—Donald Knuth, *Computer Science and its relation to mathematics*, The American Mathematical Monthly, 81(4), 1974

1.4. **Why Study Algorithms?** "Two ideas lie gleaming on the jeweler's velvet. The first is the calculus, the second, the algorithm. The calculus and the rich body of mathematical analysis to which it gave rise made modern science possible; but it has been the algorithm that has made possible the modern world."

Date: April 17, 2020.

— David Berlinski, *The Advent of the Algorithm*, 2000

1.5. **Compelling Reasons and Limitations.** "There are compelling reasons to study algorithms. To put it bluntly, computer programs would not exist without algorithms. And with computer applications becoming indispensable in almost all aspects of our professional and personal lives, studying algorithms becomes a necessity for more and more people.

Another reason for studying algorithms is their usefulness in developing analytical skills. After all, algorithms can be seen as special kinds of solutions to problems — not answers but precisely defined procedures for getting answers. Consequently, specific algorithm design techniques can be interpreted as problem-solving strategies that can be useful regardless of whether a computer is involved.

Of course, the precision inherently imposed by algorithmic thinking limits the kinds of problems that can be solved with an algorithm.

You will not find, for example, an algorithm for living a happy life or becoming rich and famous. On the other hand, this required precision has an important. . . advantage."

— Anany Levitin, *Introduction to the Design and Analysis of Algorithms*, 2012

2. Basic, Commonly Known Algorithms

2.1. **Algorithms for Computing Euclid's GCD.** The Greatest Common Divisor (GCD) of two nonnegative, not-both-zero integers m and n, denoted $gcd(m, n)$, is defined as the largest integer that divides both m and n evenly, i.e., with a remainder of zero.

There are many algorithms that produce the gcd of two numbers. Not all are of the same quality. For the approaches shown below, $\exists m, n \in \mathbb{N}$.

2.1.1. *Approach 1.* Apply repeatedly the equality
$gcd(m, n) = gcd(n, m \pmod n)$ until $m \pmod n$ is equal to 0.
Since $gcd(m, 0) = m$, then m is the greatest common divisor of m and n.
Example: $gcd(60, 24) = gcd(24, 12) = gcd(12, 0) = 12$.
Here is a mathematical representation of this algorithm

$$gcd(m, n) = \begin{cases} m & m \pmod n = 0 \\ gcd(m, n) & otherwise \end{cases}$$

Here is this algorithm in Erlang.

```
1 -module(gcd).
2 -export([gcd/2]).
3
4
5 gcd(M, 0) -> M;
6 gcd(M, N) -> gcd(N, M rem N).
```

Now apply this algorithm to finding $gcd(31415, 14142)$.
How do we know Euclid's algorithm will stop?
Is each step non-ambiguous?
Was the range of inputs specified?

2.1.2. *Approach 2.*

(1) Assign the value of $min(m, n)$ to t
(2) Divide m by t. If the remainder of this division is 0, go to step 3; otherwise go to step 4.
(3) Divide n by t. If the remainder of this division is 0, return the value of t as the answer and stop; otherwise proceed to step 4
(4) Decrease the value of t by 1. Go to Step 2.

Apply this algorithm to finding $gcd(31415, 14142)$.
What is wrong with this algorithm?

2.1.3. *Approach 3.* 1. Find the prime factors of m. 2. Find the prime factors of n. 3. Identify all common factors in the two prime expansions found in steps 1 and 2. 4. Compute the product of all the common factors and return it as the greatest common divisor of m and n.

Example: Calculate $gcd(60, 24)$

$$60 = 2 \cdot 2 \cdot 3 \cdot 5 \tag{1}$$

$$24 = 2 \cdot 2 \cdot 2 \cdot 3 \tag{2}$$

$$gcd(60, 24) = 2 \cdot 2 \cdot 3 = 12 \tag{3}$$

Apply this algorithm to finding $gcd(31415, 14142)$.
Problems with this algorithm?