Computer Science Fundamentals:

Intro to Algorithms, Systems, & Data Structures

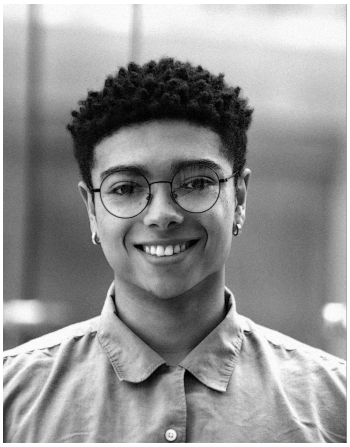
Christian J. Rudder

October 2024

Contents

This page is left intentionally blank.

Preface



Motivation: I thought I was bad at math, and terrible at taking notes. I passed all my classes through osmosis and forgot everything after the final exam; However, that wouldn't work at BU for Computer Science. I would go onto retake classes CS131 (Discrete Math), CS210 (Computer Systems), CS235 (Algebraic Algorithms w/ Leonid Levin).

What changed was when I became a course assistant for CS112 (Introduction to Computer Science II). I had taken multiple Java classes before BU and was fairly comfortable. What scared me to death was the thought of not being able to answer a student's question. This fear catapulted me into taking notes for the first time in a google doc. I would double study the material so that I would be prepared for questions in office hours, and post the notes on piazza (online forum). The positive feedback reinforced my note-taking. Knowing that others relied on my notes pushed me to improve them.

Then once it came to other classes, I thought, why not make notes for them too? I would go onto learn LaTeX to make them look nice, and use mathcha.io, to create diagrams. People began to use my notes more and more (often nagging me for when the next post was...). That pressure, combined with the pressure of passing, and the love I had for explaining and drawing the material compounded into these notes.

I was so completely done with the mountains of stress. I would rather study everything correctly the first time. After that, I never stressed for an exam again. I would actually never study, because the notes I took were a form of studying. I would always go to office hours and help others out. Once finals came around, I again, never studied, and got the best grades I've ever gotten in my life, and graduated feeling more accomplished than ever before.

Now after being through the thick of it, I truly believe anyone can be good at math, or any subject with the right amount of attention. All advanced topics are just basic definitions stacked together. If one can master the basics, everything else becomes possible.

Text-layout: We start at high-level understandings (admittedly somewhat dry with definitions), gaining familiarity with data-structures and algorithms (where the fun begins), number systems (e.g,. binary), which will allow us to dive deeper and create our own theoretical models of computation, i.e., a computer (plenty visuals, less text). After such, we revisit algorithms and data-structures, learning the classic methods and strategies that motivated our modern system today.

Big thanks to **Christine Papadakis-Kanaris**
for teaching Intro. to Computer Science II,
Dora Erdos and **Adam Smith**
for teaching BU CS330: Introduction to Analysis of Algorithms,
with contributions from:
S. Raskhodnikova, E. Demaine, C. Leiserson, A. Smith, and K. Wayne,
at Boston University

Please note: These are my personal notes, and while I strive for accuracy, there may be errors. I encourage you to always verify with your own class materials and other sources as well. Comments and suggestions for improvement are always welcome.

Support: If you want to support, give a star on github (https://github.com/Concise-Works/Algorithms), share with your friends!
Contribute: If you find an error, want to add something, or revise anything, consider forking and contributing, via the above link. Pull requests are welcome! Submit practice problems, engage with the text, take ownership of your learning.

Prerequisites

You can skip this: These prerequisites help, but they are not strictly necessary. Don't spend much if any time here. It may serve as a reference later on. Most if not all definitions are self contained and don't skip any logical steps.

However it would be helpful (not strictly) to be familiar with at least one programming language, specifically **Java** as we reference the language a few times. Not that we'll need to code, or will use them, but easily be able to read pseudocode (make-believe code that often strongly resembles real programming languages).

Use this entire text as a supplementary resource—If it doesn't make sense, cross reference other sources,and hopefully comeback and improve this one, as it's open-source.

Theorem 0.1: Common Derivatives

Power Rule: For $n \neq 0$	$\frac{d}{dx}(x^n) = n \cdot x^{n-1}$. E.g., $\frac{d}{dx}(x^2) = 2x$
Derivative of a Constant:	$\frac{d}{dx}(c) = 0$. E.g., $\frac{d}{dx}(5) = 0$
Derivative of ln:	$\frac{d}{dx}(\ln x) = \frac{1}{x}$
Derivative of $\log_a$:	$\frac{d}{dx}(\log_a x) = \frac{1}{x \ln a}$
Derivative of $\sqrt{x}$:	$\frac{d}{dx}(\sqrt{x}) = \frac{1}{2\sqrt{x}}$
Derivative of function $f(x)$:	$\frac{d}{dx}(x) = 1$. E.g., $\frac{d}{dx}(5x) = 5$
Derivative of the Exponential Function:	$\frac{d}{dx}(e^x) = e^x$

Theorem 0.2: L'Hopital's Rule

Let $f(x)$ and $g(x)$ be two functions. If $\lim_{x\to a} f(x) = 0$ and $\lim_{x\to a} g(x) = 0$, or $\lim_{x\to a} f(x) = \pm\infty$ and $\lim_{x\to a} g(x) = \pm\infty$, then:

$$\lim_{x\to a} \frac{f(x)}{g(x)} = \lim_{x\to a} \frac{f'(x)}{g'(x)}$$

Where $f'(x)$ and $g'(x)$ are the derivatives of $f(x)$ and $g(x)$ respectively.

Theorem 0.3: Exponents Rules

For $a, b, x \in \mathbb{R}$, we have:

$$x^a \cdot x^b = x^{a+b} \text{ and } (x^a)^b = x^{ab}$$

$$x^a \cdot y^a = (xy)^a \text{ and } \frac{x^a}{y^a} = \left(\frac{x}{y}\right)^a$$

Note: The $:=$ symbol is short for "is defined as." For example, $x := y$ means x is defined as y.

Definition 0.1: Logarithm

Let $a, x \in \mathbb{R}$, $a > 0$, $a \neq 1$. Logarithm x base a is denoted as $\log_a(x)$, and is defined as:

$$\log_a(x) = y \iff a^y = x$$

Meaning *log* is inverse of the exponential function, i.e., $\log_a(x) := (a^y)^{-1}$.

Tip: To remember the order $log_a(x) = a^y$, think, "base a," as a is the base of our *log* and y.

Theorem 0.4: Logarithm Rules

For $a, b, x \in \mathbb{R}$, we have:

$$\log_a(x) + \log_a(y) = \log_a(xy) \text{ and } \log_a(x) - \log_a(y) = \log_a\left(\frac{x}{y}\right)$$

$$\log_a(x^b) = b \log_a(x) \text{ and } \log_a(x) = \frac{\log_b(x)}{\log_b(a)}$$

Definition 0.2: Permutations

Let $n \in \mathbb{Z}^+$. Then the number of distinct ways to arrange n objects in order is $n! := n \cdot (n-1) \cdot (n-2) \cdot \ldots \cdot 2 \cdot 1$. When we choose r objects from n objects, it's Denoted:

$$^nP_r := \frac{n!}{(n-r)!}$$

Where $P(n, r)$ is read as "n permute r."

Definition 0.3: Combinations

Let n and k be positive integers. Where order doesn't matter, the number of distinct ways to choose k objects from n objects is it's *combination*. Denoted:

$$\binom{n}{k} := \frac{n!}{k!(n-k)!}$$

Where $\binom{n}{k}$ is read as "n choose k.", and $\binom{\cdot}{\cdot}$, the *binomial coefficient*.

Theorem 0.5: Binomial Theorem

Let a and b be real numbers, and n a non-negative integer. The binomial expansion of $(a+b)^n$ is given by:

$$(a+b)^n = \sum_{k=0}^{n} \binom{n}{k} a^{n-k} b^k$$

which expands explicitly as:

$$(a+b)^n = \binom{n}{0} a^n + \binom{n}{1} a^{n-1} b + \binom{n}{2} a^{n-2} b^2 + \cdots + \binom{n}{n-1} a b^{n-1} + \binom{n}{n} b^n$$

where $\binom{n}{k}$ represents the binomial coefficient, defined as:

$$\binom{n}{k} = \frac{n!}{k!(n-k)!}$$

for $0 \leq k \leq n$.

Theorem 0.6: Binomial Expansion of 2^n

For any non-negative integer n, the following identity holds:

$$2^n = \sum_{i=0}^{n} \binom{n}{i} = (1+1)^n.$$

Definition 0.4: Well-Ordering Principle

Every non-empty set of positive integers has a least element.

Definition 0.5: "Without Loss of Generality"

A phrase that indicates that the proceeding logic also applies to the other cases. i.e., For a proposition not to lose the assumption that it works other ways as well.

Theorem 0.7: Pigeon Hole Principle

Let $n, m \in \mathbb{Z}^+$ with $n < m$. Then if we distribute m pigeons into n pigeonholes, there must be at least one pigeonhole with more than one pigeon.

Theorem 0.8: Growth Rate Comparisons

Let n be a positive integer. The following inequalities show the growth rate of some common functions in increasing order:

$$1 < \log n < n < n \log n < n^2 < n^3 < 2^n < n!$$

These inequalities indicate that as n grows larger, each function on the right-hand side grows faster than the ones to its left.

Bibliography