Algorithm Analysis: Useful Equalities, Definitions, and Theorems

A list of useful mathematical definitions for algorithm analysis and computer science in general. Much of this was copied with minor modification from our textbook and:

- Cormen, Thomas H., et. al.. *Introduction to Algorithms*. Second Edition. MIT Press. Cambridge, MA. 2001.

Big Oh

Definition 1 (Big Oh)**.**
Where $f(n) = \mathcal{O}(g(n))$, then there exists some constant c such that $f(n) \leq c * g(n)$ for all $n \geq n_0$.

$g(n)$ is an upper bound on $f(n)$. $g \geq f$ (ish). "I can draw a $g(n)$ above $f(n)$"

Definition 2 (Big Omega)**.**
Where $f(n) = \Omega(g(n))$, then there exists some constant c such that $f(n) \geq c * g(n)$ for all $n \geq n_0$.

$g(n)$ is a lower bound on $f(n)$. $g \leq f$ (ish). "I can draw a $g(n)$ below $f(n)$"

Definition 3 (Big Theta)**.**
Where $f(n) = \Theta(g(n))$, then there exists constants c_1 and c_2 such that $c_1 * g(n) \leq f(n) \leq c_2 * g(n)$ for all $n \geq n_0$.

$f(n)$ is the same class of function as $g(n)$. "$f(n)$ is a $g(n)$ (ish)."**We can now classify functions based on Θ. Think $f(n) \in \Theta(g(n))$. $\Theta(g(n))$ is the class. This means we can classify algorithms in terms of their Time Complexity (work done).**

Theorem 1 (Transitivity)**.**
If $f(n) = \mathcal{O}(g(n))$ and $g(n) = \mathcal{O}(h(n))$, then $f(n) = \mathcal{O}(h(n))$.

If $a \leq b$ and $b \leq c$, then $a \leq c$. Ordering still works how you think it does.

Definition 4 (Dominance Relations)**.**
Where $f(n) \neq \Theta(g(n))$, we say that $f(n)$ dominates $g(n)$, or $f(n) \gg g(n)$, when $f(n) = \Omega(g(n))$ or similarly $g(n) = (O)(g(n))$.

$\mathcal{O}$, Ω, and Θ impose an ordering on *classes* of functions.

Theorem 2 (Dominance Relations of Common Functions)**.**

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

Theorem 3 (Adding Functions)**.**

$f(n) + g(n) \to \Theta(max(f(n), g(n)))$

The dominant term of a sum determines the class of the sum.

Algorithms can always be deconstructed into a series of independent blocks of code. The work done by the algorithm is the sum of the work done by each block. **You can analyze each block independently of the others and reduce the algorithm's complexity to the dominant block.**

Theorem 4 (Multiplying Functions)**.**

$\mathcal{O}(f(n) * g(n)) = \mathcal{O}(f(n)) * \mathcal{O}(g(n)))$

$\Omega(f(n) * g(n)) = \Omega(f(n)) * \Omega(g(n)))$

$\Theta(f(n) * g(n)) = \Theta(f(n)) * \Theta(g(n)))$

Multiplying can result in reclassification.

Repetition of work happens through loops (loop body is repeated) and recursion (function is repeated). Both forms of repetition induce a multiplication of work. "Repeat these operations, this number of times." **Multiplication is impactful, but orderly. You can simplify the terms (see R.H.S.) but cannot ignore one for the other like with addition.**

Sums

Definition 5 (Summation Notation).

$$\sum_{i=1}^{n} f(i) = f(1) + f(2) + \ldots + f(n-1) + f(n)$$

It's a concise way of writing the sum of n things,
sum([f(i) for i in range(1,n+1)])

Theorem 5 (Sum Closed-Forms and Equivalents).

$\sum_{i=1}^{n} a = a \times n$

$\sum_{i=k}^{n} i = \sum_{i=1}^{n} f(i) - \sum_{i=1}^{k-1} f(i)$

$\sum_{i=1}^{n} i = \frac{n(n+1)}{2} = \Theta(n^2)$

$\sum_{i=1}^{n} i^2 = \frac{n(n+1)(2n+1)}{6} = \Theta(n^3)$

$\sum_{i=1}^{n} i^3 = \frac{n^2(n+1)^2}{4} = \Theta(n^4)$

a is independent of i. Its just multiplication disguised as addition.

Sums can always be made to start from 1 (or zero if needed)

Theorem 6 (*Geometric Progression*. Sum of Exponential Sequence).
For $a \geq 1$,

$$\sum_{i=0}^{n} a^i = \frac{(a^{n+1}-1)}{(a-1)} = \Theta(a^{n+1})$$

We really like $a = 2$, where
$\sum_{i=0}^{n} 2^i = (2^{n+1}-1) = \Theta(2^{n+1})$

Theorem 7 (Sum of Increasing Logarithms).

$$\sum_{i=1}^{n} \log i = \log n! = \Theta(n \log n)$$

$\log 1 + \log 2 + \ldots + \log n - 1 + \log n$

Theorem 8 (Sum of p^{th} power of an Integer Sequence).
For $p \geq 0$,

$$\sum_{i=1}^{n} i^p = \Theta(n^{p+1})$$

.

Greatest hits: $\sum_{i=1}^{n} i$, $\sum_{i=1}^{n} i^2$, $\sum_{i=1}^{n} i^3$

Theorem 9 (*Harmonic Numbers*. Sum of a Fraction Sequence).

$$\sum_{i=1}^{n} \frac{1}{i} = \sum_{i=1}^{n} i^{-1} = \Theta(\log n)$$

.

$\frac{1}{1} + \frac{1}{2} + \frac{1}{3} + \ldots$

Theorem 10 (Sum of the multiplicative inverse of p^{th} power of an Integer Sequence).
For $p < -1$,

$$\sum_{i=1}^{n} i^p = \sum_{i=1}^{n} \frac{1}{i^{-p}} = \Theta(1)$$

.

for $p = -2$,
$\sum_{i=1}^{n} i^{-2} = 1 + \frac{1}{4} + \frac{1}{9} + \frac{1}{16} + \ldots$

Theorem 11 (Sum of Exponential Sequence with Fractional Base).
For $|a| < 1$,

$$\sum_{i=0}^{n} a^i = \Theta(1)$$

We really like $a = \frac{1}{2}$, where
$\sum_{i=0}^{n} (\frac{1}{2})^i = 1 + \frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \ldots < 2$

Exponentials

Definition 6 (Product Notation).

$\sum$ is to addition as $\prod$ is to multiplication

$$\prod_{i=1}^{n} f(i) = f(1) * f(2) * \ldots * f(n-1) * f(n)$$

Definition 7 (Exponentiation with Integer Exponents).

For integer $n \geq 1$, $a^n = \prod_{i=1}^{n} a$

$a^2 = a * a$

For integer $n \leq -1$, $a^n = \prod_{i=1}^{n} \frac{1}{a^{-n}}$

$a^{-2} = \frac{1}{a^2} = \frac{1}{a} * \frac{1}{a}$

Theorem 12 (Properties of Exponents).

For all real $a > 0, m, n$:

$a^0 = 1$
$a^m a^n = a^{m+n}$
$a^{-1} = \frac{1}{a}$
$(a^m)^n = a^{mn} = (a^n)^m = a^{nm}$

Logarithms

Definition 8 (Logarithm base b.).

Where $b^y = x$, $\log_b x = y$. *For notational convenience:* $\log_b^k n = (\log_b n)^k$. *Also,* $\log_{10}$ *is* $\log$, $\log_2$ *is* lg *and* $\log_e$ *is* ln.

log in the inverse of exponential. "The $\log_b x$ is the number you'd need to raise b to in order to get x."

Theorem 13 (Properties of Logarithms).

Think in terms of exponent properties. Like, "With $\log a$, a is an exponent and operates by the same rules."

For all $a > 0$,$b > 0$, $c > 0$, n,

$\log_c(ab) = \log_c a + \log_c b$
$\log_b \frac{1}{a} = -\log_b a$
$\log_b \frac{c}{a} = \log_b c - \log_b a$
$\log_b a^n = n \log_b a$
$a^{\log_b c} = c^{\log_b a}$

Theorem 14 (Relationships between Log bases).

Logs of different bases differ by a constant multiple.

$\log_b a = \frac{\log_c a}{\log_c b} = \frac{1}{\log_c b} \log_c a$
$\log_b a = \frac{1}{\log_a b}$

Roots

Definition 9 (Roots). *Where $r^n = x$, then the n^{th} root of x is r and is denoted by $\sqrt[y]{x} = x^{\frac{1}{y}} = r$*

Theorem 15 (Properties of Roots).

$$\sqrt[n]{ab} = \sqrt[n]{a}\sqrt[n]{b}$$
$$\sqrt[n]{\frac{a}{b}} = \frac{\sqrt[n]{a}}{\sqrt[n]{b}}$$
$$\sqrt[n]{a^m} = (a^m)^{\frac{1}{n}} = a^{\frac{m}{n}}$$

Floors and Ceilings

Definition 10 (Floor Function).

For real number x, the ***floor*** *of x, $\lfloor x \rfloor$, is the greatest integer less than x.*

Round down to the nearest integer.

Definition 11 (Ceiling Function).

For real number x, the ***ceiling*** *of x, $\lceil x \rceil$, is the least integer greater than x.*

Round up to the nearest integer.

Theorem 16 (Rounding Reals).

For any real number x,

$$x - 1 < \lfloor x \rfloor \leq x \leq \lceil x \rceil < x + 1$$

For any real $n \geq 0$ and integers $a, b > 0$:

$$\lceil \lceil n/a \rceil / b \rceil = \lceil n/ab \rceil$$
$$\lfloor \lfloor n/a \rfloor / b \rfloor = \lfloor n/ab \rfloor$$
$$\lceil a/b \rceil \leq (a + (b - 1))/b$$
$$\lfloor a/b \rfloor \geq (a - (b - 1))/b$$

Theorem 17 (Rounding Integers).

For any integer n,

$$\lceil n/2 \rceil + \lfloor n/2 \rfloor = n$$