Algorithm Analysis - Math CheatSheet

Logarithm properties

- Definition: $base^{log_{base}x} = x$
- Base change: $log_a(x) = \frac{log_b(x)}{log_b(a)}$
- Product rule: $log_b(x * y) = log_b(x) + log_b(y)$
- Quotient rule: $log_b(x/y) = log_b(x) - log_b(y)$
- $a^{log_B(b)} = b^{log_B(a)}$

Common derivatives

F(x)	F(x)’
x^n	$n * x^{(n-1)}s$
b^x	$b^x * ln(b)$
$log_b(x)$	$\frac{1}{x * ln(b)}$
$f(x) * g(y)$	$f(x)' * g(y) + f(x) * g(y)'$
$\frac{f(x)}{g(y)}$	$\frac{f(x)' * g(y) - f(x) * g(y)'}{g(x)^2}$
$f(g(x))$	$f'(g(x)) * g'(x)$

L’Hospital’s Rule

Assuming:

- $lim_{x \to a} f(x)$ and $lim_{x \to a} g(x)$ are either 0 or ∞
- $lim_{x \to a} \frac{f'(x)}{g'(x)}$ exists

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

Useful sums

- $\sum_{i=0}^{n} i = \frac{n*(n+1)}{2}$
- $\sum_{i=0}^{n} i^2 = \frac{n*(n+1)(2n+1)}{6}$
- $\sum_{i=0}^{n} i^p \in \theta(n^{p+1})$
- $\sum_{i=0}^{n} 2^i = 2^{(n+1)} - 1$
- $\sum_{i=1}^{n} a * r^{i-1} = \frac{a*(1-r^n)}{1-r}$
- $\sum_{i=0}^{n} i * x^i = \frac{x-(n+1)*x^{n+1}+n*x^{n+2}}{(x-1)^2}$
- $\sum_{i=1}^{n} \frac{1}{x} \in \theta(log(n))$
- $\sum_{i=1}^{n} ln(j) = ln(n!) = n * ln(n) - n + O(ln(n))$

Asymptotic Complexities

Assuming: $\lim_{n \to \infty}(\frac{f(n)}{g(n)}) = L$

- $0 \le L < \infty \Rightarrow f(n) \in O(g(n))$
- $0 < L < \infty \Rightarrow f(n) \in \Theta(g(n))$
- $0 < L \le \infty \Rightarrow f(n) \in \Omega(g(n))$
- $L = 0 \Rightarrow f(n) \in o(g(n))$
- $L = \infty \Rightarrow f(n) \in \omega(g(n))$

Master Theorem

Suppose: $T(n) = a * T(n/b) + f(n), T(1) = \theta(1)$.
Subject to: $a \ge 1, b > 1$, f is an asymptotically positive function.
Then: $T(n) = \sum_{i=0}^{log_b(n)-1} a^i * f(n/b^i) + \theta(n^{log_b(a)})$
Proof using recursion trees:

- $log_b(n)$ - height of the recursion tree
- $a^i * f(n/b^i)$ - cost of all internal nodes at depth i
- $\theta(n^{log_b(a)})$ - cost associated with the leaves

Master Theorem

1. $f(n) = O(n^{log_b(a)-\epsilon})$, for some constant $\epsilon > 0$ then $T(n) = \theta(n^{log_b(a)})$
2. $f(n) = \theta(n^{log_b(a)})$, then $T(n) = \theta(n^{log_b(a)} log(n))$
3. $f(n) = \Omega(n^{log_b(a)+\epsilon})$, for some constant $\epsilon > 0$ and if $a * f(n/b) \le c * f(n)$, for some constant $c < 1$ and all sufficiently large n, then $T(n) = \theta(f(n))$

Extension of Master Theorem
Assuming: $f(n) = O(n^{log_b(a)} * log_b(n)^{\alpha})$. Then:

1. $\alpha < -1 \to T(n) = \theta(n^{log_b(a)})$
2. $\alpha = -1 \to T(n) = \theta(n^{log_b(a)} log_b(log_b(n)))$
3. $\alpha > -1 \to T(n) = \theta(n^{log_b(a)} log_b(n)^{\alpha+1})$

- Sometimes we may obtain a simpler expression by changing variables:

  Example: $T(n) = 2 * T(\sqrt{n}) + log(n)$

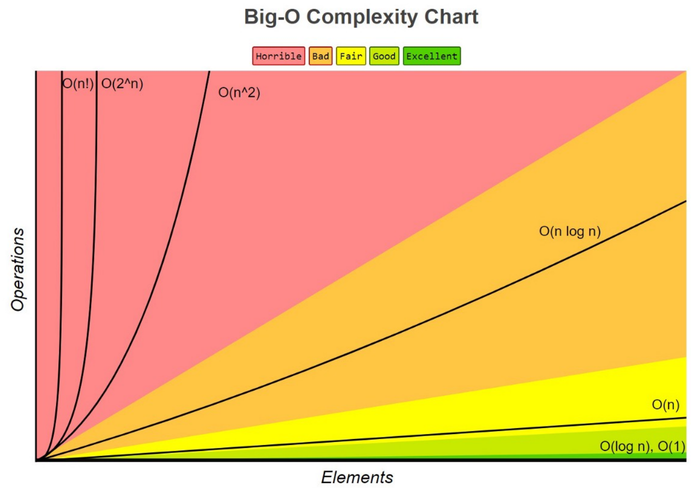
  1. Let $n = 2^m \to m = log(n) \to \sqrt{n} = 2^{m/2}$
  2. Let $S(m) = T(2^m)$

  Then: $T(2^m) = 2 * T(2^{m/2}) + m$
  $\to S(m) = 2 * S(m/2) + m$
  $\to S(m) = \theta(m * log(m))$ – Master Th.
  $\to T(n) = T(2^m) = S(m) = \theta(m * log(m))$
  $\to T(n) = \theta(log(n) * log(log(n)))$

- It may be useful to search for a lower and an upper bound that is easier to compute.

  Example: $T(n) = T(n/2 - log_2(n)) + 1$

  Consider:
  $T_1(n) = T_1(n/4) + 1 \to T_1(n) = \theta(log(n))$
  $T_2(n) = T_2(n/2) + 1 \to T_2(n) = \theta(log(n))$
  Since: $n/4 \le n/2 - log_2(n) \le n/2$
  $\to T_1(n) \le T(n) \le T_2(n)$
  $\to T(n) \in \theta(log(n))$



(source)