Asymptotic notations
Big-Oh
Big-Omega
Big-theta

Big-Oh # asymptotic upper bound
f(n) = O(g(n)): there exists constant c > 0 and n0 such that f(n)⩽cg(n) for n⩾n0

$$\log_{10} n = \frac{\log_2 n}{\log_2 10} = O(\log_2 n) = \log\ n$$

$$\sum_{i=1}^{n} i \leq n \cdot n = O(n^2)$$

$$\log(n!) = \log(n) + \cdots + log1 = O(n\ \log n)$$

Harmonic Series:

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

Big-Omega # asymptotic lower bound
f(n) = Ω(g(n)): there exists constant c > 0 and n0 such that f(n)⩾cg(n) for n⩾n0

$$\log(n!) = \log(n) + \log(n-1) + \cdots + log1$$

$$\geq \log(n) + \log(n-1) + \cdots + \log\left(\frac{n}{2}\right)$$

$$\geq \frac{\mathrm{n}}{2} \cdot \log\left(\frac{n}{2}\right)$$

$$= \frac{\mathrm{n}}{2} \cdot (\mathrm{logn} - 1) = \Omega(\mathrm{n\ logn})$$

Harmonic Series:

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

Big-Theta # asymptotic tight bound
f(n) = Θ(g(n)): f(n) = O(g(n)) and f(n) = Ω(g(n))

An interesting fact about logarithm:

$$\log_{b_1} n = O(\log_{b_2} n)$$

For any constant b1 > 1 and b2 > 1
Obviously, Ω and Θ also have this property

Solving Recurrences
- Recursion-tree Method
- Substitution Method
- Master Method and Master Theorem