Asymptotic Notation

William Schultz

September 15, 2022

If we want to describe the runtime of a Turing machine (i.e. a program), we describe it based on the size of the input. That is, how does the runtime scale as the input size scales. If we were to look at the performance on a single input, it would be difficult to compare one Turing machine against another since its specific runtime could be affected by many other factors e.g. speed of the machine running the program, etc. So, we look at the *asymptotic* behavior of the machine. That is, how does its performance scale as input sizes get very large. The notations below describes the asymptotic behavior of a function $f(n)$ in terms of a function $g(n)$. In the case of describing the runtime of a Turing machine, f typically represents the runtime (i.e. the number of steps taken) of a machine as a function of the input size, n.

Basic Asymptotic Notation

For the notations listed below, the $\in$ and $=$ symbols are, in practice, often used interchangeably, even though the former is technically correct. For example, $O(g(n))$ describes an entire family of functions (i.e. all functions asymptotically bounded above by g), so saying $f = O(g(n))$ is really an abuse of notation. Writing $f \in O(g(n))$ is more precise. See [1] for a good treatment of asymptotic notation.

- **(O) Big O (Upper Bound, analogous to $\leq$)**

  $$f(n) = O(g(n))$$
  $$f(n) = \mathcal{O}(g(n)) \quad (\textit{alternate notation})$$

  Establishes an upper bound. Formally, $\exists c > 0$ such that as n approaches ∞, $f(n) \leq c{*}g(n)$. That is, f is bounded above by g, within some constant factor, as n approaches infinity.

- **(Ω) Big Omega (Lower Bound, analogous to $\geq$)**

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

  Establishes a lower bound i.e. best case complexity. More formally, $\exists c > 0$ such that as n approaches ∞, $f(n) \geq c * g(n)$. That is, f is bounded below by g, within some constant factor, as n approaches infinity.

- **(Θ) Big Theta (analogous to $\approx$)**

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

  f is bounded asymptotically both above and below by $g(n)$. That is, $f = O(g(n))$ and $f = \Omega(g(n))$.

- **(o) Small O (analogous to $<$)**

  $$f(n) = o(g(n))$$

  f is dominated by g asymptotically. Can think of this as a variant of Big O notation but saying something stronger i.e. $f = o(g(n))$ implies $f = O(g(n))$. The function f is not only bounded above by g but is dominated by g. In other words, f's asymptotic growth is strictly less than g's growth. Formally, we can state this by saying that $\forall c > 0$, as n goes to infinity, $f(n) \leq c * g(n)$. As a concrete example, $x^2 \in o(x^3)$.

- **(ω) Small Omega (analogous to $>$)**

  $$f(n) \in \omega(g(n))$$

  f dominates g asymptotically. Similar to $o(g(n))$ but establishes a strict lower bound. It is a stronger statement than $f = \Omega(g(n))$, saying that f's asymptotic growth is strictly greater than g's.

Additional Notation

In addition to the above standard notations, there are some additional common asymptotic notations:

- $f(n) = \text{poly}(g(n))$
  This is equivalent to
  $$f(n) = g(n)^{O(1)}$$
  which basically means that f is bounded above by g to some constant power.
- $f(n) = \tilde{O}(g(n))$
  This is equivalent to
  $$\begin{aligned} f(n) &= g(n) \cdot \text{poly}(\log(g(n))) \\ &= O(g(n) \cdot \log^{O(1)}(g(n))) \end{aligned}$$
  which is basically just like saying that f is nearly the same as g but with some extra factor that is logarithmic in g.

References

[1] Ronald L. Graham, Donald E. Knuth, and Oren Patashnik. *Concrete Mathematics: A Foundation for Computer Science*. Addison-Wesley Longman Publishing Co., Inc., USA, 2nd edition, 1994.