Tutorial 1

1 Big-O notation

Prove or disprove each of the following statements.

(a) For every constant $b > 1$, the functions $f : n \mapsto \log_2(n)$ and $g : n \mapsto \log_b(n)$ satisfy $f = \Theta(g)$.

Solution. The statement is true.

Proof. For every $n \geq 1$,

$$f(n) = \log_2(n) = \frac{\log_b(n)}{\log_b(2)} = \frac{1}{\log_b(2)} \cdot g(n).$$

So $f = O(g)$ by definition with $n_0 = 1$ and $c = \frac{1}{\log_b(2)}$. And $g = O(f)$ by definition with $n_0 = 1$ and $c = \log_b(2)$, so $f = \Omega(g)$ and, therefore, $f = \Theta(g)$. □

Moral: The base of the logarithm doesn't matter when we are working with asymptotic notation!

(b) For every integer $k \geq 0$ and every constant $b > 1$, the functions $f : n \mapsto n^k$ and $g : n \mapsto b^n$ satisfy $f = o(g)$.

Solution. The statement is true.

The cleanest proof of the statement uses an alternative definition for $o(\cdot)$ notation.

Fact. *Two functions $f, g : \mathbb{N} \to \mathbb{R}^+$ satisfy $f = o(g)$ if and only if $\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0$.*

Proof that the statement is true. We prove the statement by induction on k. In the base case, when $k = 0$, for every $c > 0$ we have $n^0 = 1 < cb^n$ for every $n > \log_b(c)$ so $f = o(g)$.

For the induction step, the induction hypothesis $n^{k-1} = o(b^n)$ implies that $\lim_{n\to\infty} \frac{n^{k-1}}{b^n} = 0$. Then by L'Hopital's rule,

$$\lim_{n\to\infty} \frac{n^k}{b^n} = \lim_{n\to\infty} \frac{k}{\ln b} \cdot \frac{n^{k-1}}{b^n} = 0$$

and $n^k = o(b^n)$. □

Moral: (Any) polynomial time complexity is always better than (any) exponential time complexity! And, similarly, poly-logarithmic time complexity is always better than any polynomial time complexity.

(c) The functions $f : n \mapsto \sqrt{\log n}$ and $g : n \mapsto \log(\sqrt{n})$ satisfy $f = O(g)$ but not $f = \Theta(g)$.

Solution. The statement is true.

Proof. For every $n \geq 1$,

$$f(n) = \sqrt{\log n} \leq \log n = 2(\tfrac{1}{2}\log n) = 2\log(n^{1/2}) = 2g(n).$$

so $f = O(g)$ by definition with $n_0 = 1$ and $c = 2$.

Conversely, for any $c > 0$, whenever $n > 2^{4c^2}$ then $c < \frac{1}{2}\sqrt{\log n}$ and

$$g(n) = \log(\sqrt{n}) = \tfrac{1}{2}\log n = (\tfrac{1}{2}\sqrt{\log n}) \cdot \sqrt{\log n} > cf(n)$$

so $g \neq O(f)$. Therefore, $f \neq \Omega(g)$, so $f \neq \Theta(g)$ either. □

(d) The functions $f : n \mapsto (n+2)!$ and $g : n \mapsto n!$ satisfy $f = \Theta(g)$.

Solution. The statement is false.

Proof. The function f satisfies $f(n) = (n+2)(n+1)n! > n^2 g(n)$. So for any constant $c > 0$, we have $f(n) > c\,g(n)$ for every $n > \sqrt{c}$. Therefore, $f \neq O(g)$, which also implies that $f \neq \Theta(g)$. □

(e) For every functions $f, g : \mathbb{N} \to \mathbb{R}^{\geq 2}$, if $f = \Theta(g)$ then the functions $F : n \mapsto \log(f)$ and $G : n \mapsto \log(g)$ (with logarithms over base 2) satisfy $F = \Theta(G)$.

Solution. The statement is true.

Proof. When $f = O(g)$, there exist $c > 0$ and $n_0 \in \mathbb{N}$ such that for all $n \geq n_0$, $f(n) \leq c\,g(n)$. For these values of n, we also have

$$F(n) = \log(f) \leq \log(c\,g(n)) = \log(g(n)) + \log(c) = G(n) + \log(c).$$

Define $c' = (1 + \log(c))$. Then for all $n \geq n_0$, $F(n) \leq c'G(n)$ and so $F = O(G)$.

The same argument with the roles of f and g exchanged shows that $G = O(F)$ and so $F = \Omega(G)$. Therefore, $F = \Theta(G)$. □

(f) Arrange the following functions in increasing order of growth rate. All bases of the logarithms are 2.:

$$n2^{\sqrt{\log(n)}}, n\sqrt{\log(n)}, n^{1.59}, 2^n, n\log(\log(n)), n^{2013}1.99^n$$

Solution.

$$n\log(\log(n)), n\sqrt{\log(n)}, n2^{\sqrt{\log(n)}}, n^{1.59}, n^{2013}1.99^n, 2^n$$

Best way to see the order is to first separate the exponentials, which are big, $B = \{2^n, n^{2013}1.99^n\}$ from the polynomials, which are small, $S = \{n\sqrt{\log(n)}, n^{1.59}, n\log(\log(n))\}$ and order B and S separately. One function $n2^{\sqrt{\log(n)}}$ is not clear and we will look at that separately.

The order of S is $n\log(\log(n)), n\sqrt{\log(n)}, n^{1.59}$: ignoring the common n component: we see that $\log\log(n)$ is a much more slowly growing function that $\log(n)$, which grows much more slowly than $n^{0.59}$.

The order of B is $B = \{n^{2013}1.99^n, 2^n\}$. This can be seen by applying L'Hopital's rule 2013 times to $\frac{n^{2013}1.99^n}{2^n}$, we get:

$$lim_{n\to\infty}2013!(\frac{1.99}{2})^n$$

This limit goes to 0, so 2^n is larger.

For the order of $n2^{\sqrt{\log_2(n)}}$: Call this function $h(n)$. It may not be clear if $h(n)$ is a polynomial or an exponential. If you consider a larger function $n2^{\log_2(n)}$, you'll notice that $h(n)$ is a polynomial because $n2^{\log_2(n)}$ is equal to n^2 (because $2^{\log_2(n)} = n$). We can see $h(n)$ will be greater than $n\sqrt{\log(n)}$, because it contains the exponential of $\sqrt{\log(n)}$ but how it compares to $n^{1.59}$ is not clear. We can make the following observation: $2^{\sqrt{\log_2(n)}} = o(2^{\frac{\log_2(n)}{2}} = \sqrt{n})$. This can be seen by taking the logarithm of both $2^{\sqrt{\log_2(n)}}$ and $2^{\frac{\log_2(n)}{2}}$ and comparing the growth of the logarithms *(Side exercise: prove that if* $\log(f(n)) = o(\log(g(n)))$*, then* $f(n)$ *is also* $o(g(n))$*. Note that the reverse is not necessarily true. As a second side exercise, show a counter example for the reverse too.)* So $n2^{\sqrt{\log_2(n)}} = o(n\sqrt{n} = n^{1.5})$. Therefore $n2^{\sqrt{\log_2(n)}}$ is asymptotically smaller than $n^{1.5}$, so also $n^{1.59}$.

In general if ordering of two functions is not clear: (1) use L'Hopital's rule; and/or (2) modify one of it to make it look more like the other, such that the order becomes clear.