1 Review of big-Oh notation

Definition: $f(n)$ is $O(g)$ iff $\exists n_0 \in \mathbb{N}, c \in \mathbb{R}: f(n) \leq c \cdot g(n) \forall n \geq n_0$

Intuition: $f(n)$ is $O(g(n))$ if $f(n)$ grows at most as fast as some constant times $g(n)$, for large $n$.

IMPORTANT: The running time of selection sort on an array of $n$ elements was $1 + 5n + 13n^2$, which is $O(n^2)$. But it is also $O(n^3)$, and $O(n^4)$, and $O($any function that grows at least as fast as $n^2$). However, we usually try to give the snuggest big-Oh description possible.

2 Hierarchy of big-Oh classes

$O(g(n))$ can be seen as the set of all functions $f(n)$ that are $O(g(n))$: $O(g(n)) = \{ f(n) : \exists c, n_0, \forall n \geq n_0, f(n) \leq c \cdot g(n) \}$.

Then we can write $n^2 + 10n + 2 \in O(n^2)$.

We have the following (incomplete) hierarchy of big-Oh classes:

- $O(1)$: functions bounded above by a constant. $f(n) = 100 \in O(1)$, $10 + \sin(n) \in O(1)$. All primitive operations can be executed in time $O(1)$.
- $O(\log n)$: logarithmic functions. Running time of binarySearch is $O(\log n)$.
- $O(\sqrt{n})$: square root of $n$. $2 + \sqrt{n} + 10 \in O(\sqrt{n})$
- $O(n)$: linear functions. Running time of findMin is $O(n)$
- $O(n^k)$, for some integer $k > 1$: polynomials. Running time of selectionSort is $O(n^2)$.
- $O(a^n)$, for some $a > 1$: exponential functions. Running time of Fibonacci recursive algorithm is $O(2^n)$.

3 Shortcuts

Always relying on $O()$ definition to prove statements is tiring... Instead, we can use the following rules:

1. Sum rule. If $f_1(n) \in O(g(n))$ and $f_2(n) \in O(g(n))$ then $f_1(n) + f_2(n) \in O(g(n))$.

   Proof:
Theorem: Let \( f_1(n) \in O(g(n)) \) and \( f_2(n) \in O(g(n)) \), there must exist constants \( c_1, n_1, c_2, n_2 \) such that \( f_1(n) \leq c_1 g(n) \) \( \forall n \geq n_1 \) and \( f_2(n) \leq c_2 g(n) \) \( \forall n \geq n_2 \). Thus, if we pick \( c_3 = c_1 + c_2 \) and \( n_3 = \max(n_1, n_2) \), we have that if \( n \geq n_3 \), then \( f(n) + g(n) \leq c_1 g(n) + c_2 g(n) = (c_1 + c_2) g(n) = c_3 g(n) \).

Thus \( f_1(n) + f_2(n) \) is \( O(g(n)) \).

2. Constant factors rule. if \( f_1(n) \in O(g(n)) \) then \( kf_1(n) \in O(g(n)) \) for any constant \( k \).

Example: \( n^3 + 10n^2 + \log(n) \in O(n^3) \) because \( 10n^2 \in O(n^3) \subset O(n^3) \) and \( \log(n) \in O(\log(n)) \subset O(n^3) \).

By rule (1), \( n^3 + 10n^2 + \log(n) \in O(n^3) \)

Example: for any polynomial \( p(n) = a_k n^k + a_{k-1} n^{k-1} + ... + a_1 n + a_0 n^0 \), we have \( p(n) \in O(n^k) \).

3. Product rule. if \( d(n) \in O(f(n)) \) and \( e(n) \in O(g(n)) \), then \( d(n) \cdot e(n) \in O(f(n) \cdot g(n)) \).

Example: \( (1 + 10n) \cdot (2\log(n) + 3) \in O(n \cdot \log(n)) \), because...

4. \( n^x \in O(a^n) \) for any fixed \( x > 0 \) and \( a > 1 \). However, \( a^n \notin O(n^x) \) for any fixed \( x > 0 \) and \( a > 1 \).

Example: \( n^{1000} \in O(1.0001^n) \). However, the constants \( c \) and \( n_0 \) for which \( n^{1000} \leq c \cdot 1.0001^n \) \( \forall n \geq n_0 \) are very large.

5. \( \log(n^x) \in O(\log(n)) \) for any fixed \( x > 0 \).

Proof: \( \log(n^x) = x \log(n) \in O(\log(n)) \) (by rule 2).

6. \( \log_a(n) \in O(\log_b(n)) \) for any fixed \( a > 1, b > 1 \).

Proof: \( \log_a(n) = \log_b(n)/\log_b(a) \in O(\log_b(n)) \) (by rule 2).

4 More shortcuts

Theorem: Let \( f(n) \) and \( g(n) \) be two non-negative functions. Then

1. if \( \lim_{n \to +\infty} f(n)/g(n) = 0 \), then \( f(n) \in O(g(n)) \) and \( g(n) \notin O(f(n)) \).

2. if \( \lim_{n \to +\infty} f(n)/g(n) = x \neq 0 \), then \( f(n) \in O(g(n)) \) and \( g(n) \in O(f(n)) \).

3. if \( \lim_{n \to +\infty} f(n)/g(n) = +\infty \), then \( g(n) \in O(f(n)) \) and \( f(n) \notin O(g(n)) \).

4. if \( \lim_{n \to +\infty} f(n)/g(n) \) does not exist, then we can’t say anything.

Reminder: l’Hospital rule:

\[ \lim_{n \to +\infty} f(n)/g(n) = \lim_{n \to +\infty} \frac{df(n)/dn}{dg(n)/dn} \]

Example: Prove that \( \log(n) \in O(\sqrt{n}) \).