Definition We say function $f(n)$ belongs to a **class** $O(g(n))$ iff

$$\exists C, n_0 \text{ so that } \forall n \geq n_0 \; f(n) \leq Cg(n)$$

It is pronounced *there exist numbers C and n_0 so that for all values n greater or equal than n_0 it is true that $f(n) \leq Cg(n)$.*

A short-hand notation is $f(n) \in O(g(n))$, this notation shows that $O(g(n))$ is not a single function, but a collection, i.e. class. In Computer Science a less precise notation is used $f(n) = O(g(n))$

Examples

1. Prove $\sqrt{n} = O(n)$: so our goal is to find to numbers C and n_0 so that $\sqrt{n} \leq n$ starting at n_0 (and values greater). Let's make a guess – say $C = 1$ and $n_0 = 1$, then we look at the corresponding inequality to see if it holds:

$$\sqrt{n} \leq n$$

is it true for all n starting at $n_0 = 1$?

Let's rewrite – cancel $\sqrt{n}$ on both sides:

$$1 \leq \sqrt{n}$$

This is obviously true, for n is 1 or greater.

2. Prove $2n + 5 = O(n)$.

- Attempt 1 (will fail): fix $C = 1$ and $n_0 = 1$, then see if

$$2n + 5 \leq n$$

for all n starting at $n_0 = 1$. False! Actually the inequality is never true.

- Attempt 2 (will also fail): fix $C = 2$ and $n_0 = 1$, then see if

$$2n + 5 \leq 2n$$

same problem - always false.

- Attempt 3 (will also fail): fix $C = 3$ and $n_0 = 1$, then see if

$$2n + 5 \leq 3n$$

the above inequality is not true for $n \geq 1 = n_0$, it fails for $n = 1, 2, 3, 4$. Actually if we set n_0 to be 5, we would have proved it.

- Attempt 4 (best): fix $C = 3$, but do not fix n_0 for know – wait till we see the inequality:

$$2n + 5 \leq 3n$$

simplify (subtract $2n$ on both sides):

$$5 \leq n$$

so if we fix $n_0 = 5$, the above inequality holds and we have a proof.

3. Prove $9n - 18 = O(n^2)$. Let's fix $C = 1$. Then we want inequality

$$9n - 18 \leq n^2$$

to be true starting at some integer, which we will call n_0. Let's simplify the above

$$n^2 - 9n + 18 \geq 0$$

or

$$(n-3)(n-6) \geq 0$$

it is easy to see that the inequality will hold if $n \geq 6$. So just $n_0 = 6$. Done.

4. Prove $\log_a n = O(\log_b n)$. This example shows that base of the logarithm is not important in big-O notations. This is why it is often omitted – for example we just say that binary search complexity is $O(\log n)$ – no base specified.

   Use change of base formula for logarithms

$$\log_a n = \frac{\log_b n}{\log_b a}$$

   Notice that term $\log_b a$ is a constant, therefore $\log_a n$ and $\log_b n$ belong to the same big-O class.

5. A harder example show $\ln n = O(\sqrt{n})$. Reminder $\ln n = \log_e n$ – natural logarithm. Let's fix $C = 1$. Then we want inequality

$$\ln n < \sqrt{n}$$

to be true starting at some integer, which we will call n_0. Unfortunately it is not easy to deal with formulas that involve both logarithms and powers, therefore instead of trying to solve the inequality (as we were doing before) we will analyze derivatives of the two function:

$$(\ln n)' = \frac{1}{n}$$
$$(\sqrt{n})' = \frac{1}{2\sqrt{n}}$$

Note that starting with 4 derivative of $\ln n$ is stricly less than the derivative of $\sqrt{n}$:

$$\frac{1}{n} < \frac{1}{2\sqrt{n}}$$
$$2 < \sqrt{n}$$

Also notice $\ln 4 < \sqrt{4}$. Therefore funtion $\ln n$ is smaller then $\sqrt{n}$ at 4 and logarithm's derivative is smaller then the derivative of square root, thus $\ln n$ will continue to be smaller:

$$\ln n < \sqrt{n} \text{ for all } n > n_0 = 4$$

6. Actually log is growing slower than any positive power of n, that is

$$\log_a n = O(n^\epsilon), \text{ where } \epsilon > 0$$

To show the opposite, i.e. $f(n)$ is not in $O(g(n))$-class we have to negate the definition of big-O:

Definition We say function $f(n)$ does not belongs to a **class** $O(g(n))$ iff

$$\forall C, n_0 \exists n \geq n_0 \text{ so that } f(n) > Cg(n)$$

The statement above says that "no matter how you choose c and n_0, there will always be a value n that breaks the inequality.

1. Prove n^2 is not in $O(n)$. Since C, n_0 are universally quantified, we keep them as variables and look for an appropriate n **given** C and n_0 – so that n which breaks the inequality becomes a function of C and n_0. Note that the value of n has to satisfy two conditions:

$$\begin{cases} n & \geq n_0 \\ n^2 & \geq Cn \end{cases}$$

   simplify to

$$\begin{cases} n & \geq n_0 \\ n & > C \end{cases}$$

   so that $n = \max(n_0, C+1)$. The "+1" is because second inequality is strict.