# Characterising Running Times through Big-O Notations

## Understanding the Big-O notation mathematically

**Jidneya Desale**
Y12 CSSOC member

In this article I hope to expand your idea of what Big O notation really is and introduce it a more mathematical sense. We will not talk in depth about its practical applications, but rather we will explore how Big O notation really works and also give some graphical interpretations. Through our newly gained understanding of the topic we will apply it to some examples and this will lead us to understand why Big O notation works the way that we have learnt in class, instead of jut accepting that this is the way it works.

I'm sure that you would have heard of the Big-O notation and its usage in analysing the time complexity of algorithms. However a phrase that you may not have heard is how it is used to describe the asymptotic behaviour of functions in a mathematical context. In this article I want to go over the Big-O notation in a new light.

# THE O-NOTATION

We know O-notation as being something that we use to analyse the time complexity of an algorithm. However let me present to you another definition:

> O-notation characterizes an upper bound on the asymptotic behaviour of a function.

## Definition (intuitive)

What this is telling is that the function grows no faster than a certain rate, where this rate is determined by the highest order term of that function.

$$f(x) = 9n^3 + 6n^2 + 10n + 7$$

For example we have this function $f$, we would write the time complexity in Big-O notation as $O(n^3)$. What may surprise you is that we could also write the time complexity for the function as $O(n^4)$, $O(n^5)$, or even $O(n^6)$. This is because the function grows slowly than anything above $n^3$.

Thus we can generalise this by saying

$$O(n^c) \qquad \text{where } c \geq 3$$

But of course in practice we would only use $O(n^3)$ as it is more useful than any higher power of $n$.

## Definition (formal)

What we just saw is a more conceptual understanding of the Big-O notation, however now it is time we formalise this definition into a more concrete mathematical one:

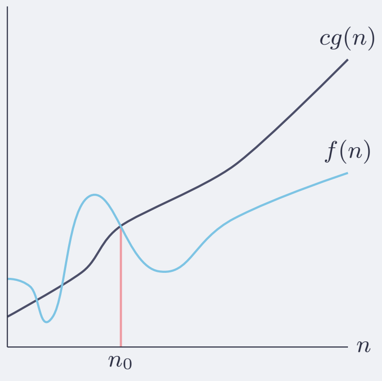
For a given function $g(n)$, we denote by $O(g(n))$ the set of all the functions:

$$O(g(n)) = f(n) : \text{there exist positive constants } c \text{ and } n_0$$

such that $0 \leq f(n) \leq cg(n)$ for all $n \geq n_0$

### Visualisation

The below graph is a very good visualisation of the Big-O notation:



$$f(n) = O(g(n))$$

One important restriction that we should be aware of is that the definition of $O(g(n))$ requires every function $f(n)$ in the set $O(g(n))$ must be *asymptotically non-negative*: which is where $f(n)$ must be non-negative whenever n is sufficiently large. This also implies that the function $g(n)$ must also be asymptotically non-negative, or the set of $O(g(n))$ would be empty.

Great, now we know that we define Big O notation in term of sets, but you must now be confused as to why we usually write $f(n) = O(g(n))$, instead of using the proper set syntax of $f(n) \in O(g(n))$.

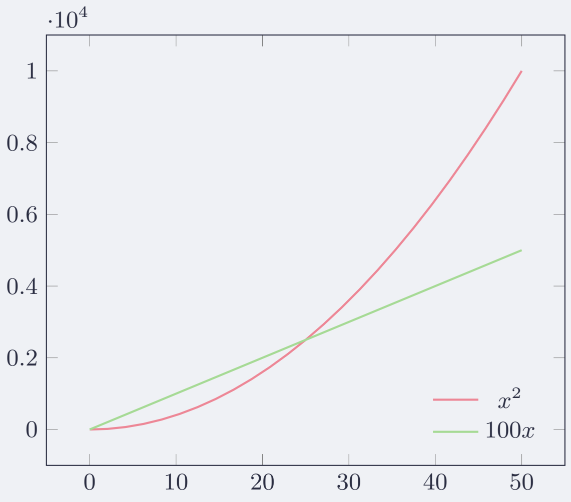
It might seem a bit confusing at first, but by abusing the equality this way, we can its advantages.

## Worked example

$$f(n) = 4n^2 + 100n + 500$$

Let us now conceptually understand is why for this example, why the Big-O notation is $O(n^2)$, rather than $O(n)$, where the coefficient of $n$ is much larger (25 times) than the coefficient of $n^2$.

When we graph both the graphs of $4n^2$ and $100n$, we shall see that eventually $4n^2 > 100n$



Let us now use our formal definition to explore some examples and why we use equality this way.

Take the example of $4n^2 + 100n + 500 = O(n^2)$, despite the coefficient of $n^2$ being comparatively much smaller than the others. To do this we must first find positive constants $c$ and $n_0$ such that

$$4n^2 + 100n + 500 \leq cn^2 \qquad \text{for all } n \geq n_0$$

By dividing both sides of the equation by $n^2$, we can see that we now get the inequality

$$4 + 100/n + 500/n^2 \leq c$$

There are many values of $c$ and $n_0$ that satisfy the equation, such as $c = 604$ and $n_0 = 1$. Another thing that we should see is that as $n$ grows, the value of the expression decreases, and thus the inequality is never violated.

Now let us see an example where the inequality is violated, and why some functions do not belong to the Big-O notation of functions of smaller exponents.

$$f(n) = n^3 - 100n^2$$

Why does it not belong to $O(n^2)$ even though the coefficient of the $n^2$ term is a much larger negative number? One conceptual way to understand this is that at some point $n^3 \geq 100n^2$, so the output of the function will basically only be determined by $n^3$ as the function grows very big - which is what our Big-O notation works for. This uses the same concept that we have used above.

What we can also say is that

- If $n^3 - 100n^2 = O(n^2)$,
- Implies $n^3 - 100n^2 \leq cn^2$ for all $n \geq n_0$

Some simplification lead us to $n - 100 \leq c$. So no matter what value we choose for $c$, this inequality will not hold if $n > c + 100$, thus it is invalid.

## EXTENSIONS

Three questions are presented below for yours to try:

a. Explain why the statement, *the running time of $A$ is at least $O(n^2)$*, is meaningless.

b. Is $2^{n+1} = O(2^n)$?

c. Is $2^{2n} = O(2^n)$?