9.A.1 BIG O NOTATION – EXAMPLES

Example 1

Let $f(n) = 3n^3$ for an algorithm. Prove that $f(n)$ of the algorithm is in $O(n^3)$.

Solution

The definition of the Big -Oh notation is that $f(n) \leq c \times g(n)$.

In order to prove that we know:

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

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

Where $g(n)$ is in n^3.

Hence, we can show that:

$3n^3 \leq c \times n^3$, holds good for a positive number c and sufficiently large values of n.

$f(n) = 3n^3$ *can also be written as*:

$f(n) = 3n^3 + 0$

We can write it as:

$3n^3 + 0 \leq 3n^3 + n^3$

$\approx 3n^3 + 0 \leq 4n^3$ [Note $3n^3 = 3n^3$ *but* $3n^3$ *is always* $\leq 4n^3$]

[As highest degree of polynomial n *is* 3]

$\approx 0 \leq 4n^3 - 3n^3$

$\approx 0 \leq n^3$

Or, Divide n^2 in both side:

$$\approx \frac{0}{n^2} \leq \frac{n^3}{n^2}$$

$$\approx 0 \leq n$$

or, $n \geq 0$

Therefore $n_0 = 0$.

We can again write it as:

$3n^3 + 0 \leq 4 \times n^3$

Hence $c \geq 4$.

Therefore, $f = O(g)$

or in other words, the algorithm is $O(n^3)$.

Hence proved.

Example 2

Let $f(n) = 3n + 8$ for an algorithm. Prove that $f(n)$ of the algorithm is in $O(n)$.

Solution

The definition of the Big -Oh notation is that $f(n) \leq c \times g(n)$.

In order to prove that we know:

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

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

Where $g(n)$ is in n.

Hence, we can show that:

$3n + 8 \leq c \ \times n$, holds good for a positive number c and sufficiently large values of n.

$$3n + 8 \leq 3n + n$$

$$\approx 3n + 8 \leq 4n \, [\, 3n \; is \; always \; \leq 4n]$$

[As highest degree of polynomial $n \; is \; 1$]

$$\therefore 3n + 8 \leq 4n$$

$$\approx 8 \leq 4n - 3n$$

$$\approx 8 \leq n$$

$$or, n \geq 8$$

$$\boxed{\therefore n_0 = 8}$$

$$3n + 8 \leq 4n$$

Can be written as:

$$3n + 8 \leq 4 \ \times n$$

$$Hence \; , c \geq 4 \; .$$

Therefore, f is O(g)

or in other words $O(n)$.

Hence proved.

Example 3

Let $f(n) = n^2 + 1$ for an algorithm. Prove that $f(n)$ of the algorithm is in $O(n^2)$.

Solution

The definition of the Big -Oh notation is that $f(n) \leq c \times g(n)$.

In order to prove that we know:

$f(n) \in O\big(g(n)\big)$

or, $f(n) = O\big(g(n)\big)$

Where $g(n)$ is in n^2.

Hence, we can show that:

$n^2 + 1 \leq c \times n^2$, holds good for a positive number c and sufficiently large values of n.

$n^2 + 1 \leq n^2 + n^2$

$\approx n^2 + 1 \leq 2n^2[\,n^2 \; is \; always \; \leq 2n^2]$

[As highest degree of polynomial $n \; is \; 2$]

$\therefore n^2 + 1 \leq 2n^2$

$\approx 1 \leq 2n^2 - n^2$

$\approx 1 \leq n^2$

$\approx -n^2 + 1 \leq 0$

$\approx -(n^2 + 1) \leq 0$

$\approx -(n^2 - (-1)^2) \leq 0$

$As \; we \; know: x^2 - y^2 = (x + y)(x - y)$

$\approx -1 \times (\; \big(n + (-1)\big) \; \big(n - (-1)\big) \;) \leq 0$

$$\approx \mathbf{-1} \times (\ (\boldsymbol{n} - \mathbf{1}) \ \ (\boldsymbol{n} + \mathbf{1}) \ \ ) \leq \mathbf{0}$$

$$\approx (\boldsymbol{n} - \mathbf{1})(\boldsymbol{n} + \mathbf{1}) \leq \frac{\mathbf{0}}{\mathbf{-1}}$$

$$\approx (\boldsymbol{n} - \mathbf{1})(\boldsymbol{n} + \mathbf{1}) \leq \mathbf{0}$$

[From Quadratic Inequalities]

$$Say\ n = 1$$

$\approx (1 - 1) \times (1 + 1) \leq 0$

$\approx 0 \times 2 \leq 0$

$\approx 0 \leq 0[True]$

Also,

$\approx 0 \geq 0[True]$

$$Say\ n = 2$$

$\approx (2 - 1) \times (2 + 1) \leq 0$

$\approx 1 \times 3 \leq 0$

$\approx 3 \leq 0[False]$

But,

$\approx 3 \geq 0[True]$

Hence, we can say $n \geq 1$

$$Say\ n = 0$$

$\approx (0-1) \times (0+1) \leq 0$

$\approx -1 \times 1 \leq 0$

$\approx -1 \leq 0[True]$

$$Say\ n = -1$$

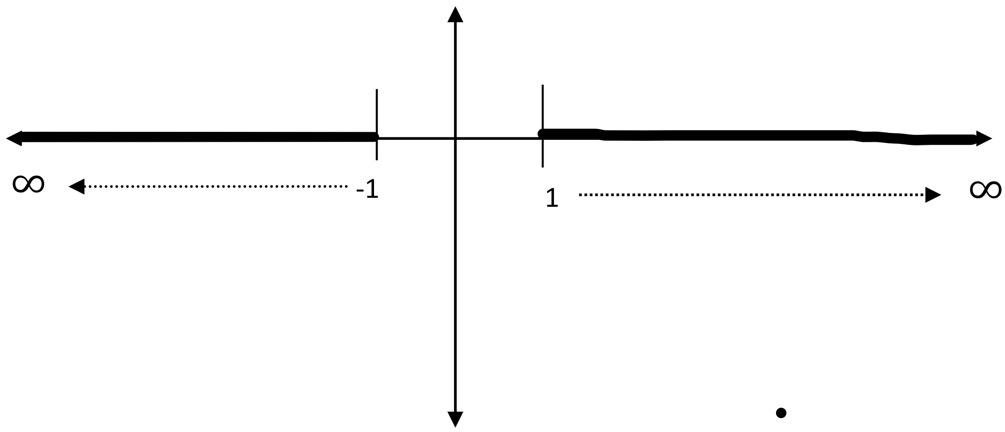
$\approx (-1-1) \times (-1+1) \leq 0$

$\approx -2 \times 0 \leq 0$

$\approx 0 \leq 0[True]$

That implies $n \leq -1$

$$\therefore \boldsymbol{-1 \geq n \geq 1}$$



$(-\infty, -1] \cup [1, \infty)$ *and they are in Sem − open and semi closed intervals.*

Hence now we get:

$$n^4 + 1 \leq 2n^2 \; for \; all \; n \geq 1$$

As from the definition of Big – O notation, all should be positive numbers but not negative numbers.

Therefore , we get $n_0 = 1$.

And,

$n^4 + 1 \leq 2 \times n^2$ *in terms of* $\boldsymbol{f(n) \leq c \times g(n)}$

Hence:

$\boldsymbol{c \geq 2}$ **.**

Therefore, f is O(g)

or in other words $\boldsymbol{O(n^2)}$.

Hence proved.

Example 4

Let $\boldsymbol{f(n) = n^4 + 100n^2 + 50}$ for an algorithm. Prove that $\boldsymbol{f(n)}$ of the algorithm is in $\boldsymbol{O(n^4)}$.

Solution

The definition of the Big -Oh notation is that $\boldsymbol{f(n) \leq c \times g(n)}$.

In order to prove that we know:

$\boldsymbol{f(n) \in O(g(n))}$

or, $\boldsymbol{f(n) = O(g(n))}$

Where $\boldsymbol{g(n)}$ is in $\boldsymbol{n^4}$.

Hence, we can show that:

$n^4 + 100n^2 + 50 \leq c \times n^4$

$\approx n^4 + 100n^2 + 50 \leq n^4 + n^4$

$\approx n^4 + 100n^2 + 50 \leq 2n^4$ [n^4 *is always* $\leq 2n^4$]

[As highest degree of polynomial n *is* 4]

Now,

$\approx -2n^4 + n^4 + 100n^2 + 50 \leq 0$

$\approx -n^4 + 100n^2 + 50 \leq 0$

We can write the above equation as:

$\approx -n^4 + 100n^2 + 50 = 0$ *and* $-n^4 + 100n^2 + 50 < 0$

Taking the equation:

$\approx -n^4 + 100n^2 + 50 = 0$

Rewriting the equation, $u = n^2$ *and* $u^2 = n^4$:

$\approx -u^2 + 100u + 50 = 0$

Solving with quadratic equation formula:

Quadratic Equation of the form $ax^2 + bx + c = 0$:

$$x_{1,2} = \frac{\left(-b \pm \sqrt{b^2 - 4ac}\right)}{2a}$$

For a $= -1$, b $= 100$, c $= 50$

$$u_{1,2} = \frac{\left(-100 \pm \sqrt{100^2 - 4(-1)(50)}\right)}{2(-1)}$$

$$u_{1,2} = \frac{\left(-100 \pm \sqrt{100^2 - 4(-1)(50)}\right)}{-2}$$

$$u_{1,2} = \frac{\left(-100 \pm \sqrt{100^2 + 200}\right)}{-2}$$

$$u_{1,2} = \frac{\left(-100 \pm \sqrt{10000 + 200}\right)}{-2}$$

$$u_{1,2} = \frac{\left(-100 \pm \sqrt{10200}\right)}{-2}$$

$Using\ prime\ factorization\ of\ 10200 = 2^3 \times 3 \times 5^2 \times 17$

$$u_{1,2} = \frac{\left(-100 \pm \sqrt{2^3 \times 3 \times 5^2 \times 17}\right)}{-2}$$

Applying exponent rule: $a^{b+c} = a^b \times a^c$

$$u_{1,2} = \frac{\left(-100 \pm \sqrt{2^2 \times 2 \times 3 \times 5^2 \times 17}\right)}{-2}$$

Applying radical rule: $\sqrt[n]{ab} = \sqrt[n]{a} \times \sqrt[n]{b}$

$$u_{1,2} = \frac{\left(-100 \pm \sqrt{2^2} \times \sqrt{5^2} \times \sqrt{2 \times 3 \times 17}\right)}{-2}$$

Applying radical rule: $\sqrt[n]{\boldsymbol{a}^{\boldsymbol{n}}} = \boldsymbol{a}$

$$\boldsymbol{u}_{1,2} = \frac{(-\mathbf{100} \pm \mathbf{2} \times \mathbf{5} \times \sqrt{\mathbf{2} \times \mathbf{3} \times \mathbf{17}})}{-\mathbf{2}}$$

$$\boldsymbol{u}_{1,2} = \frac{(-\mathbf{100} \pm \mathbf{10} \times \sqrt{\mathbf{2} \times \mathbf{3} \times \mathbf{17}})}{-\mathbf{2}}$$

$$\boldsymbol{u}_{1,2} = \frac{(-\mathbf{100} \pm \mathbf{10}\sqrt{\mathbf{102}})}{-\mathbf{2}}$$

$$\boldsymbol{u} = \frac{(-\mathbf{100} + \mathbf{10}\sqrt{\mathbf{102}})}{-\mathbf{2}}$$

$$= \frac{\mathbf{10}(-\mathbf{10} + \sqrt{\mathbf{102}})}{-\mathbf{2}}$$

$$= -\mathbf{5}(-\mathbf{10} + \sqrt{\mathbf{102}}) - -\boldsymbol{i}$$

$$\boldsymbol{u} = \frac{(-\mathbf{100} - \mathbf{10}\sqrt{\mathbf{102}})}{-\mathbf{2}}$$

$$= \frac{-\mathbf{10}(\mathbf{10} + \sqrt{\mathbf{102}})}{-\mathbf{2}}$$

$$= \mathbf{5}(\mathbf{10} + \sqrt{\mathbf{102}}) - -\boldsymbol{ii}$$

Substituting back $u = n^2$ and solving for n,

$n^2 = -\mathbf{5}(-\mathbf{10} + \sqrt{\mathbf{102}})$

$(g(x))^2$ $cannot\ be\ negative\ for\ x \in R$, $hence\ no\ solution.$

$n^2 = \mathbf{5}(\mathbf{10} + \sqrt{\mathbf{102}})$

We know $(g(x))^2 = f(a)$ $the\ solutions\ are$ $\sqrt{f(a)}$, $-\sqrt{f(a)}$

$\therefore \boldsymbol{n} = \sqrt{\mathbf{5}(\mathbf{10} + \sqrt{\mathbf{102}})}$ **and** $\boldsymbol{n} = -\sqrt{\mathbf{5}(\mathbf{10} + \sqrt{\mathbf{102}})}$

Now we can easily understand,

$\approx -n^4 + 100n^2 + 50 = 0$

putting n= $-\sqrt{5(10+\sqrt{102})}$ in the above equation we will get a negative value while putting n= $\sqrt{5(10+\sqrt{102})}$ in the above equation we will get a positive value,

Hence:

$$n \leq -\sqrt{5(10+\sqrt{102})}$$

or

$$n \geq \sqrt{5(10+\sqrt{102})}$$

Now what does $\sqrt{5(10+\sqrt{102})}$ $stand\ for : 10.028484537$

As by definition: the function f and g should be set of natural numbers and it should grow by time (growth rate) we take $n \geq 11$.

Or by analysis:

$\approx -n^4 + 100n^2 + 50 \leq 0$

$\approx 100n^2 + 50 \leq n^4$

$or, n^4 \geq 100n^2 + 50$

if we take $n = 10$

$\approx\ 10^4 \geq 100 \times 10^2 + 50$

$\approx\ 10000 \geq 10000 + 50$

$\approx\ 10000 \geq 10050[Not\ \ True]$

if we take $\boldsymbol{n} = \mathbf{11}$

$\approx \mathbf{11^4 \geq 100 \times 11^2 + 50}$

$\approx \mathbf{14641 \geq 1210 + 50}$

$\approx \mathbf{14641 \geq 1260}[\ \boldsymbol{True}]$

Hence, we confirm in both the ways that:

$\boldsymbol{n^4 + 100n^2 + 50 \leq 2n^4\ for\ all\ n \geq 11}$

Therefore , we get $n_0 = 11$.

And,

$\boldsymbol{n^4 + 100n^2 + 50 \leq 2\ \times n^4}\ in\ terms\ of\ \boldsymbol{f(n) \leq c\ \times g(n)}$

Hence:

$\boldsymbol{c \geq 2}$.

Therefore, f is O(g)

or in other words $\boldsymbol{O(n^4)}$.

Hence proved.

Example 5

Let $\boldsymbol{f(n) = n}$ for an algorithm. Let $\boldsymbol{g(n) = n}$. Prove that $\boldsymbol{f(n)\ of\ this\ algorithm\ is\ in\ O(n)}$.

Solution

$\boldsymbol{f(n) \leq c\ \times g(n)}$

$\Rightarrow \boldsymbol{n \leq 1 \times n, for\ all\ n \geq 1}$

$\Rightarrow \boldsymbol{n =\ O(n), c \geq 1\ and\ n_0 = 1}$

Example 6

Let $f(n) = 410$ for an algorithm. Let $g(n) = 410$. Prove that $f(n)\ of\ this\ algorithm\ is\ in\ O(1)$.

Solution

$f(n) \leq c \times g(n)$

$\Rightarrow 410 \leq 1 \times 410, for\ all\ n \geq 1$

$\Rightarrow 410 = O(1), c \geq 1\ and\ n_0 = 1$

NO UNIQUENESS in Above Method

There is no unique set of values for n_0 and c in proving the asymptotic bounds.

Let us consider, $100n + 5 = O(n)$. For this function there are multiple n_0 and c values possible.

Solution1:

100n+5 ≤100n+n

≈100n+5 ≤101n, for all $n \geq 5, n_0 = 5\ and\ c \geq 101$ is a solution.

Solution2:

100n+5 ≤100n+5n

≈100n+5 ≤105n, for all $n \geq 1, n_0 = 1\ and\ c \geq 105$ is also a solution.

Example 7

Let $f(n) = 3n^3 + 2n^2 + 3$ for an algorithm. Let $g(n) = n^3$. Prove that $f(n)\ of\ this\ algorithm\ is\ in\ O(n^3)$.

Solution

The definition of Big-Oh notation is that $f(n) \leq c \times g(n)$. Therefore, one must show that $3n^3 + 2n^2 + 3 \leq cn^3$ holds good for a positive number c and for sufficiently large values of n.

$f(n) = 3n^3 + 2n^2 + 3$

$f(n) \leq 3n^3 + 2n^3 + 3$ $(as\ growth\ of\ functions\ n^2\ to\ n^3)$

$f(n) \leq 3n^3 + 2n^3 + 3n^3$ $(3\ is\ less\ than\ n^3)$

$f(n) \leq 8n^3$

It can be observed that $c = 3 + 2 + 3 = 8$ (one can approximate $2n^2$ and 3 to $2n^3\ and\ 3n^3\ respectively$) . This condition holds good for any values of $c \geq 8$.

Let the polynomial be

$$f(n) = \sum_{i=0}^{m} a_i n^i$$

whose degree is m. Then one can show that $f(n) = O(n^m)$.

$|f(n)| \leq |a_m|n^m + |a_{m-1}|n^{m-1} + \cdots + |a_1|n + |a_0|$

$\approx |f(n)| \leq |a_m|n^m + |a_{m-1}|n^m + \cdots + |a_1|n^m + |a_0|n^m\ \ for\ all\ n \geq 1$

$$\approx \left(\sum_{i=0}^{m} |a_i|\right) n^m$$

$\approx c \times n^m$

$\approx O(n^m)$

Hence the above algorithm has $\boldsymbol{O(n^3)}$.

This is another way we can prove the algorithm has $\boldsymbol{the\ complexity}$.

Example 8

Let $\boldsymbol{f(n) = \frac{(2x^3+13\log_2 x)}{7n^2}\ for\ an\ algorithm\ A. Prove\ that\ f(n)}$ $\boldsymbol{of\ algorithm\ A\ is\ O(n)}$.

Solution

It can be observed that $\boldsymbol{logx < x}$ is always true. Therefore, one can argue that $\boldsymbol{13\log_2 x \leq 13x}$ and as $\boldsymbol{13x \leq 13x^3}$ always , one can rewrite $\boldsymbol{f(n)}$ as follows:

$$\boldsymbol{f(n) \leq \frac{2x^3 + 13x^3}{7n^2}}$$

$$\boldsymbol{f(n) \leq \frac{15x^3}{7n^2}}$$

$$\boldsymbol{\cong 2n^{3-2}}$$

$\boldsymbol{\cong 2n}$ for all n > 1

$$\boldsymbol{\therefore f(n) = O(n)}$$

Example 9

Prove that $n \in O(n^2)$

Solution

This implies that $n \leq c \times n^2. This\ is\ true\ for\ n \geq n_0, where\ n_0 = 0\ and\ c > 0$.

Therefore, $n \in O(n)$.