Asymptotic Notations: Big O, Big Omega and Big Theta Explained (With Notes)

codewithharry.com/videos/data-structures-and-algorithms-in-hindi-3

Asymptotic notation gives us an idea about how good a given algorithm is compared to some other algorithm.

Now let's look at the mathematical definition of 'order of.' Primarily there are three types of widely used asymptotic notations.

1. Big oh notation (O)
2. Big omega notation (Ω)
3. Big theta notation (θ) – Widely used one

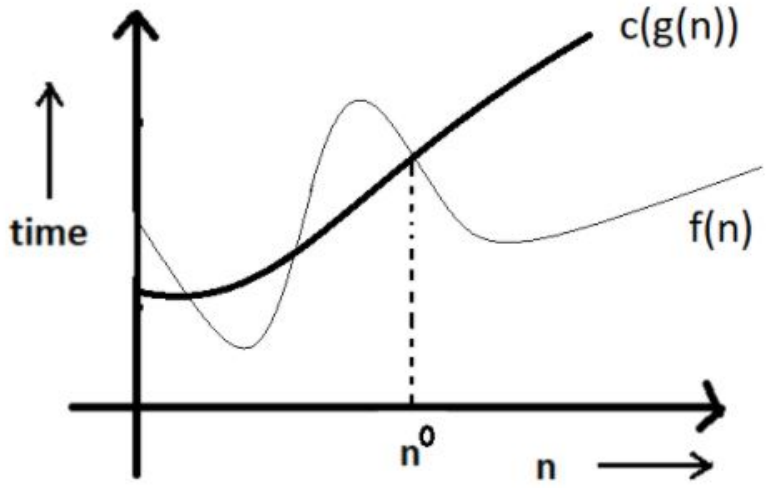
Bigoh notation (O):

- Big oh notation is used to describe an asymptotic upper bound.
- Mathematically, if f(n) describes the running time of an algorithm; f(n) is O(g(n)) if and only if there exist positive constants c and n° such that:

  $0 \leq f(n) \leq c\ g(n) \qquad \text{for all } n \geq n°.$

- Here, n is the input size, and g(n) is any complexity function, for, e.g. n, n2, etc. (It is used to give upper bound on a function)
- If a function is O(n), it is automatically O(n2) as well! Because it satisfies the equation given above.

Graphic example for Big oh (O):



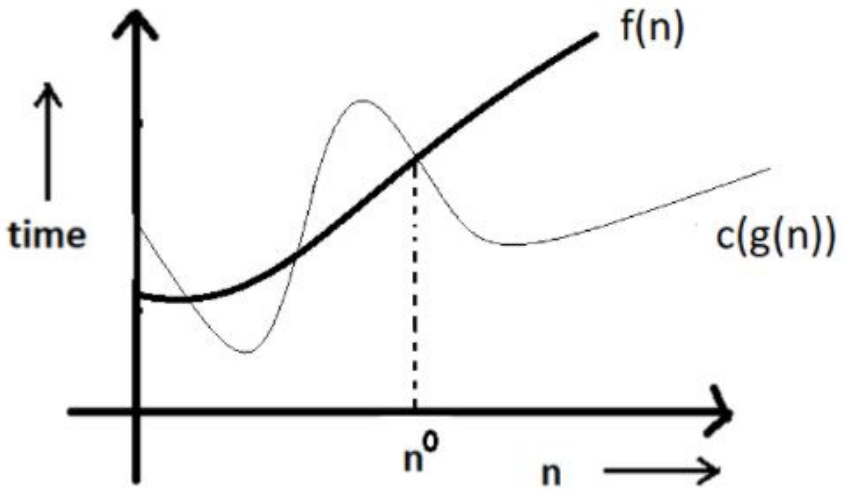
Big Omega Notation (Ω):

- Just like O notation provides an asymptotic upper bound, Ω notation provides an asymptotic lower bound.
- Let f(n) define the running time of an algorithm; f(n) is said to be Ω (g(n)) if and only if there exist positive constants c and n° such that:

```
0 ≤ c g(n) ≤ f(n)        for all n ≥ n°.
```

- It is used to give the lower bound on a function.
- If a function is Ω (n2) it is automatically Ω (n) as well since it satisfies the above equation.

Graphic example for Big Omega (Ω):



Big theta notation (θ):

- Let f(n) define the running time of an algorithm.
- F(n) is said to be θ (g(n)) if f(n) is O (g(n)) and f(x) is Ω (g(n)) both.

Mathematically,

$$0 \le f(n) \le c_1 g(n) \quad \forall \quad n \ge n_o$$

$$0 \le c_2 g(n) \le f(n) \quad \forall \quad n \ge n_o$$

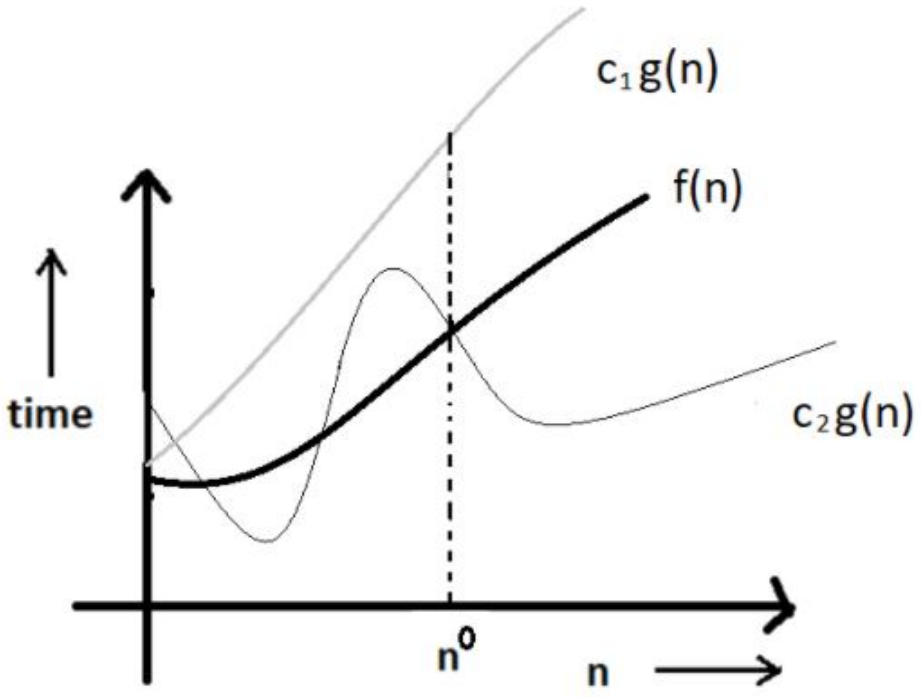
for sufficiently large value of n

Merging both the equations, we get:

```
0  ≤ c2 g(n)  ≤  f(n) ≤ c1 g(n)      ∀    n ≥ no.
```

The equation simply means that there exist positive constants c1 and c2 such that f(n) is sandwiched between c2 g(n) and c1 g(n).

Graphic example of Big theta (θ):



Which one of these to use?

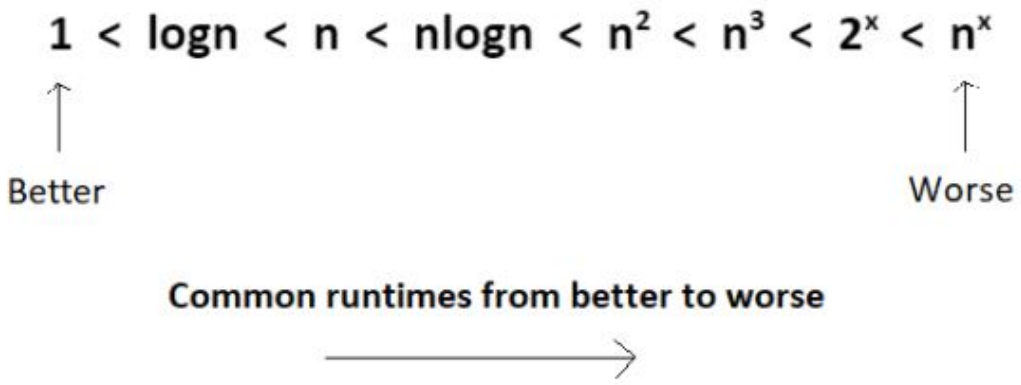
Big theta provides a better picture of a given algorithm's run time, which is why most interviewers expect you to answer in terms of Big theta when they ask "order of" questions. And what you provide as the answer in Big theta, is already a Big oh and a Big omega. It is recommended for this reason.

Quick Quiz: Prove that n2+n+1 is O(n3), Ω(n2), and θ(n2) using respective definitions.

Hint: You can approach this both graphically, making some rough graphs and mathematically, finding valid constants c1 and c2.

Increasing order of common runtimes:

Below mentioned are some common runtimes which you will come across in your coding career.



So, this was all about the asymptotic notations. We'll come across these a lot in future. However, there's no need for concern. Just stay with me.

Thank you for being with me throughout the tutorial. I hope you enjoyed it. If you appreciate my work, please let your friends know about this course too. Make sure to download the notes linked below. If you haven't checked out the whole playlist yet, move on to codewithharry.com or my YouTube channel to access it. See you all in the next tutorial here we'll learn analysing an algorithm and differentiating them on the basis of their time complexities. Till then keep learning.

Download notes here