Asymptotic Notations

Asymptotic Notations

- Asymptotic notations are mathematical tools to represent the time complexities of the algorithms.

- There are three different notations-
  - Big Omega (Ω) – lower bound
  - Big Theta (θ) – tight bound
  - Big Oh (O) – upper bound

Asymptotic Notations

- From the algorithm analysis, we found that the time complexity of the algorithm would be any of the follows-

  $1 < \log n < \sqrt{n} < n < n \log n < n^2 < n^3 < \ldots < 2^n < 3^n < \ldots < n^n$

- Suppose we have a function given as

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

The highest degree in above function is n^2, So all the time complexities after n^2 including itself will be the upper bound, all before including n^2 itself will be the lower bound of the function. The n^2 itself will be the tight bound.

tight bound

$1 < \log n < \sqrt{n} < n < n \log n < n^2 < n^3 < \ldots < 2^n < 3^n < \ldots < n^n$

lower bound

upper bound

Hence, $f(n) = O(n^2)$, $f(n) = O(n^3)$, $f(n) = O(2^n)$ // upper bound

Similarly,

$f(n)=\Omega(n^2)$, $f(n)=\Omega(n)$, $f(n)=\Omega(\log n)=\Omega(1)$

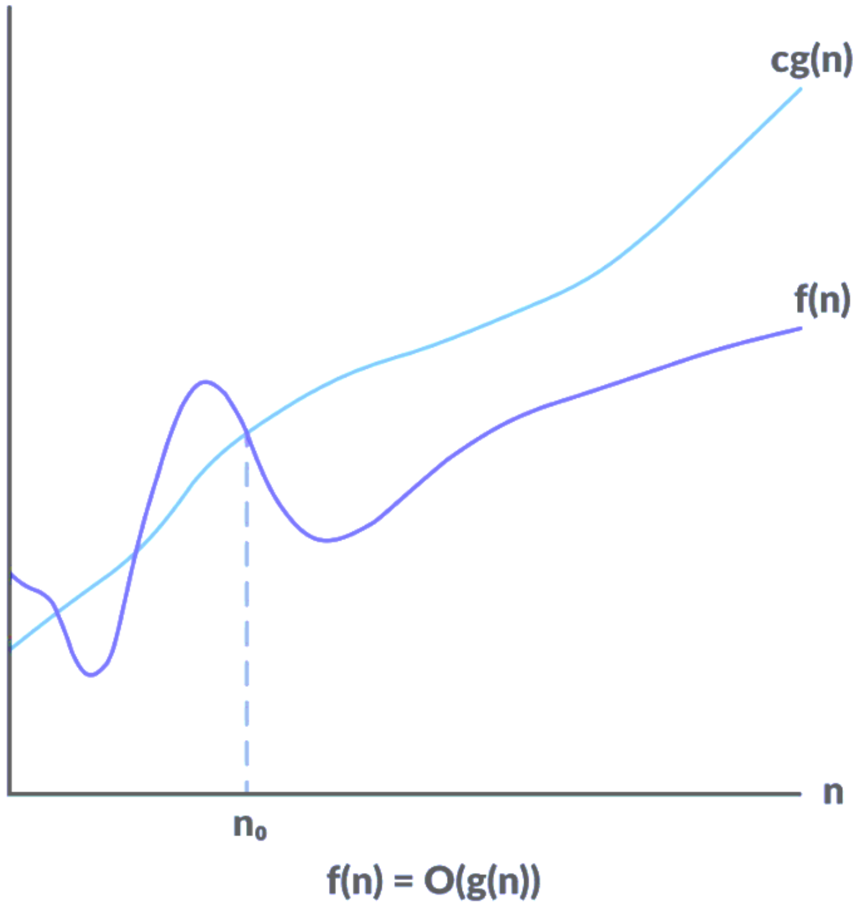
For any function if $f(n)=\Omega(n^2)$ and $f(n)=O(n^2)$ then $f(n)=\theta(n^2)$. [known as tight bound]

Asymptotic Notations: Big O notation

Formal Definition:

f(n)= O(g(n)) if there exist positive constant c and n_0 such that-

f(n)<= c*g(n) for all values of $n >= n_0$



Big-O gives the upper bound of a function

Asymptotic Notations: Big O notation

Formal Definition:

$f(n) = O(g(n))$ if there exist positive constant c and n_0 such that-

$f(n) <= c*g(n)$ for all values of $n >= n_0$

If $f(n) = 2n^2 + 3n+1$

If we make all statements n^2 then statement will be $2n^2 + 3n^2 + n^2 = 6n^2$

and $2n^2 + 3n+1 <= 6n^2$ for all values of n i.e., $n>=1$

$f(n) <= c*g(n)$ and $g(n) = 6n^2$, then it can be said that

$f(n) <= 6\ n^2$

So, we can say, $f(n) <= O(g(n))$ for $c=6$ and $n_0 = 1$

$f(n) = O(n^2)$

Asymptotic Notations: Big O notation

$f(n) = O(g(n))$ if there exist positive constant c and n_0 such that-

$f(n) <= c*g(n)$ for all values of $n >= n_0$

If $f(n) = 2n^2 + 3n+1$

If we make all statements 2^n then statement will be $2*2^n + 3*2^n + 2^n = 6*2^n$

and $2n^2 + 3n+1 <= 6*2^n$ for all values of n_0

$f(n) <= c*g(n)$ and $g(n) = 6*2^n$ then it can be said that

$f(n) <= 6 *2^n$

So, we can say, $f(n) <= O(g(n))$ for $c=6$ and $n_0 >=0$

$f(n) = O(2^n)$

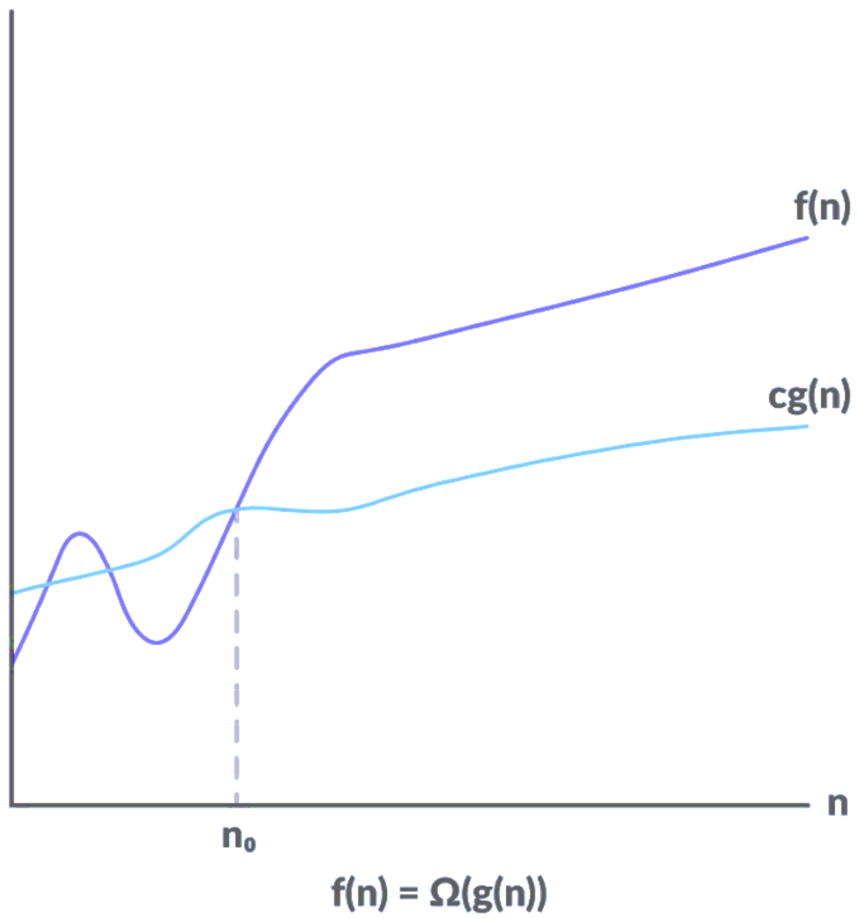
Always write the closest function. For the above function, the closest function is $g(n) = 6n^2$

Asymptotic Notations: 'Ω notation

Formal Definition:

The lower bound of $f(n)$ will be 'Ω(g(n)), i.e., $f(n) = $ 'Ω(g(n)), if there exist positive constant c and n_0 such that-

$f(n) >= c*g(n)$ for all values of $n >= n_0$



Omega gives the lower bound of a function

Asymptotic Notations: Ω notation

Formal Definition:

$f(n) = \Omega(g(n))$ if there exist positive constant c and n_0 such that-

$f(n) >= c*g(n)$ for all values of $n >= n_0$

If $f(n) = 2n^2 + 3n+1$

If we discard everything except n^2 then $g(n) = n^2$ and $f(n) = 2n^2 + 3n+1$. So,

$2n^2 + 3n+1 >= 1* n^2$ for all values of $n >= n_0$

$f(n) \quad >= c* g(n)$

So, we can say, $f(n) >= \Omega(g(n))$ for $c=1$ and $n_0 >=1$

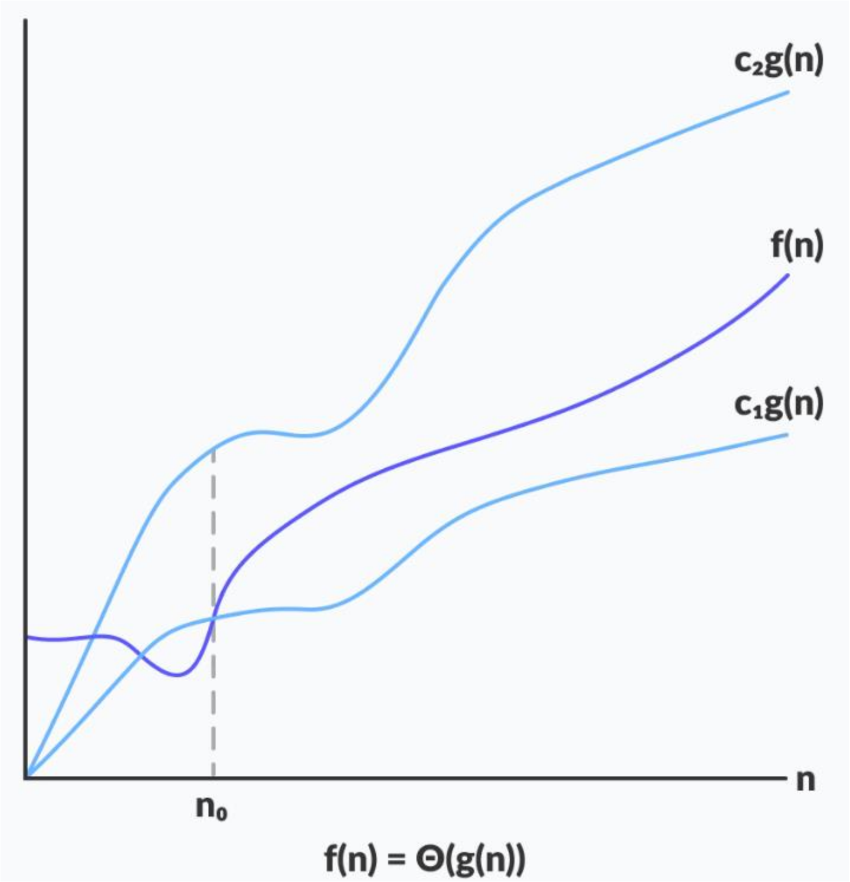
$= \Omega(n^2)$

Asymptotic Notations: θ notation

Formal Definition:

f(n)= θ(g(n)) if there exist positive constant c_1, c_2, and n_0 such that-

c1*g(n) <=f(n)<= c2*g(n) for all values of $n >= n_0$



Theta gives the tight bound of a function

Asymptotic Notations: θ notation

Formal Definition:

$f(n) = \theta(g(n))$ if there exist positive constant c_1, c_2, and n_0 such that-

$c1*g(n) <= f(n) <= c2*g(n)$ for all values of $n >= n_0$

If $f(n) = 2n^2 + 3n+1$, then

$1 * n^2 <= 2n^2 + 3n+1 <= 6 * n^2$

$C1 * g(n) <= f(n) <= c2 * g(n)$

Therefore, we can say $f(n) = \theta(g(n)) = \theta(n^2)$

This is the tight bound and only one statement will be satisfied for the tight bound. For the above we cannot say $f(n) = \theta(n)$ but for other asymptotic notation there may be other condition which may also satisfy.

Asymptotic Notations: Example

$f(n) = n^2 \log n + n$

Replace n by $n^2 \log n$ for upper bound, discard everything after first statement for lower bound.

$n^2 \log n \quad \le \quad n^2 \log n + n \quad \le \quad 2 * n^2 \log n$

Lower bound   Tight bound   Upper bound

Same term for both lower and upper, so, we can get the tight bound.

So, $f(n) = O(n^2 \log n)$, $f(n) = \Omega(n^2 \log n)$, $f(n) = \theta(n^2 \log n)$

Asymptotic Notations

Find asymptotic upper bound for the $f(n) = n!$.

Asymptotic Notations

Find asymptotic upper bound for the $f(n) = n!$.

As, $n! = n*(n-1)*(n-2)*(n-3)* \ldots *2*1$

To find the upper bound replace all statement by n

$g(n) = n*n*n*n* \ldots *n*n = \mathbf{n^n}$

As per the asymptotic upper bound $f(n) <= c*g(n)$, so

$n! <= 1*n^n$ for $n>=1$

Thus, we can say, $\mathbf{f(n)= O(n^n)}$

To find the lower bound replace all statement either by 1 or log n

$g(n) = 1*1*1*1* \ldots *1* = 1^n$ **or** $g(n) = \log n* \log n \ldots * \log n *= n\log n$

As per the asymptotic lower bound $f(n) >= c*g(n)$, so

$f(n) = n! >= 1*1^n$ for $n>=n_0$ Thus, we can say, $f(n)= \Omega(1)$

$f(n) = n! >= n\log n$ for $n>=n_0$ Thus, we can say, $f(n)= \Omega(n\log n)$

Asymptotic Notations: Example

For average or tight bound, we can write,

$1^n <= n! <= n^n$

It can be observed that there is no common asymptotic values for lower and upper bounds. Therefore, no tight bound will exists for n!.

For many functions, tight bound do not exist. In such case upper and lower bound can be used to represent the asymptotic bounds.

$f(n) = O(n^n)$

$f(n) = \Omega(1)$

Asymptotic Notations: Example

f(n)= log n!

We can write as-

log(1*1*1*…*1) <= log (1*2*3…*n) <= log (n*n*n….*n*n)

1<= log n! <= log n^n

1<= log n! <= nlog n

This also has no common terms for lower and upper bounds, so for this also no tight bounds exists. In such cases upper bounds are used for asymptotic bounds.

Please note, tight bound does not exist for the factorial functions.

Theta notations always tells the best value/ complexity while others may not give closest value/ complexity. This means Big O and Big Omega is used when we cannot find the exact figure/ complexity.

Properties of Asymptotic Notations

1. If f(n) is O(g(n)) then a* f(n) is also O(g(n)).

for e.g., $f(n) = 2n^2 + 3 = O(n^2)$ then

$5*f(n) = 5*(2n^2 + 3) = 10n^2 + 15 \Rightarrow O(n^2)$

The same property also satisfy for Ώ (n) and θ(n).

2. If f(n) is given then f(n) = O(f(n)).

for e.g., $f(n) = 2n^2 + 3 = O(f(n)) = O(n^2)$

The same property also satisfy for Ώ (n) and θ(n).

3. If f(n) is O(g(n)) and g(n) is O(h(n)), then f(n) = O(h(n))

for e.g., $f(n) = n$, $g(n) = n^2$ and $h(n) = n^3$

$n = O(n^2)$ and $n^2 = O(n^3)$, then

$n = O(n^3)$

The same property also satisfy for Ώ (n) and θ(n).

Properties of Asymptotic Notations

4. Symmetric property

If f(n) is θ(g(n)) then g(n) is θ(f(n)).

for e.g., $f(n) = 2n^2 + 3 = \theta(n^2)$

$g(n) = n^2 + 2 \Rightarrow \theta(n^2)$

As both functions are same, they will be symmetric.

This property only satisfy for θ(n).

5. Transpose Symmetric

If f(n) is O(g(n)) then g(n) is Ώ (f(n)).

for e.g., $f(n) = n + 3 = O(n)$,

$g(n) = 2n^2 + 1 = O(n^2)$

Then, $n = O(n^2)$ and $n^2 =$ Ώ(n)

This property satisfy for big O and big omega.

Properties of Asymptotic Notations

6. If $f(n) = O(g(n))$ then $f(n) = \Omega(g(n))$, i.e.,

$g(n) \;<=\; f(n) \;<=\; g(n)$, then

$f(n)=\theta(g(n))$

Find the asymptotic bound for the following

i. If $f(n)= O(g(n))$, and $d(n) =O(e(n))$, then $f(n)+ d(n)=?$

Assume $f(n)= n$ and $d(n)= n^2$ then

$f(n)+ d(n)= n+ n^2 = O(n^2)$

If we take $f(n)= n^2$ and $d(n)= n$ then

$f(n)+ d(n)= n^2+ n = O(n^2) =$ **$O(max(O(g(n), e(n)))$**

This means the asymptotic bound will be equal to the function having greater bound.

ii. If $f(n)= O(g(n))$, and $d(n) =O(e(n))$, then $f(n)*d(n)=?$

Comparison of two Functions

Which of the following is greater?

n^2 or n^3

Method 1

n	n^2	n^3
1	1	1
2	4	8
3	9	27
4	16	64

It can be observed from the above that $n^3 > n^2$.

Method 2

Take log for n^2 and n^3

$\text{Log } n^2 = 2\text{Log } n$	$\text{Log } n^3 = 3\text{Log } n$

It can be observed from the above that $2\text{Log } n < 3\text{Log } n$. So, $n^3 > n^2$.

Important log formulas

$a^{\wedge}\log_c^b = b^{\wedge}\log_c^a$

$a^b = n$ then $b = \log_a^n$

$\log a/b = \log a - \log b$

$\log ab = \log a + \log b$

$\log_a^b = b\log a$

Comparison of two Functions

Which of the following is greater?

$f(n) = n^2 \log n$ or $g(n) = n(\log n)^{10}$

Apply log both sides

$\log(n^2 \log n)$ $\log[n(\log n)^{10}]$

$\log n^2 + \log\log n$ $\log n + \log(\log n)^{10}$

$2\log n + \log\log n$ $\log n + 10\log\log n$

Bigger Term

Smaller Term

Here, $2\log n > \log n$ so, $n^2 \log n > n(\log n)^{10}$

Which of the following is greater?

$f(n) = 3n^{\sqrt{n}}$ or $g(n) = 2^{\sqrt{n}\log_2{}^n}$

Important log formulas

1. $a^\wedge\log_c{}^b = b^\wedge\log_c{}^a$
2. $a^b = n$ then $b = \log_a{}^n$
3. $\log a/b = \log a - \log b$
4. $\log ab = \log a + \log b$
5. $\log_a{}^b = b \log_a$

Comparison of two Functions

Which of the following is greater?

$f(n) = n^{\log n}$ or $g(n) = 2^{\sqrt{n}}$

Which of the following is greater?

$f(n) = 2^n$ or $g(n) = 2^{2n}$

Which of the following is greater?

$f(n) = 2n$ or $g(n) = 3n$

Comparison of two Functions

Which of the following is greater?

$f(n) = n^{\log n}$ or $g(n) = 2^{\sqrt{n}}$

Apply log

$\log n^{\log n}$ $\log 2^{\sqrt{n}}$

$\log n \times \log n$ $\sqrt{n} \log_2 2$

$\log^2 n$ $\sqrt{n}$

Apply log one more time

$2 \log \log n$ $\frac{1}{2} \log n$

So, $f(n) < g(n)$

Which of the following is greater?

$f(n) = 2^n$ or $g(n) = 2^{2n}$

Apply log

$\log 2^n$ $\log 2^{2n}$

$n\log_2 2$ $2n \log_2 2$

n $2n$

So, $f(n) < g(n)$

Which of the following is greater?

$f(n) = 2n$ or $g(n) = 3n$

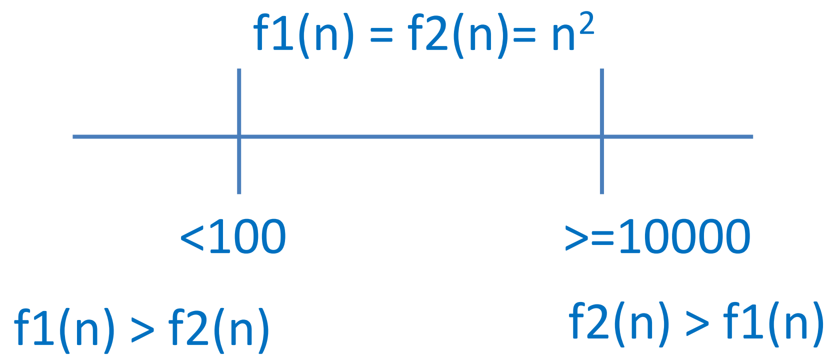
Both are asymptotically equal because we are not changing function by applying log

Comparison of two Functions

Which of the following is greater?

$$f1(n) = \begin{cases} n^3 & n<100 \\ n^2 & n>=100 \end{cases}$$

$$f2(n) = \begin{cases} n^2 & n<10000 \\ n^3 & n>=10000 \end{cases}$$



It can be observed from the analysis that f2(n) will always greater than f1(n) for n>=10000.

Best, Worst, and Average case Analysis

Let's take an example of linear search

A=

7	10	12	9	21	15	6	8

If we want to search a key =21, 5 comparison will be needed

For any key such as 75, it is not present in the list so search will be unsuccessful. As we are only interested in successful search so, check all possible cases for successful search.

Case 1: if the key is the first element (7) of the list [Best case]

Best case time = 1 (constant)=> B(n)=1

Case 2: if the key is the last element (8) of the list [Worst Case]

Worst case time = n => w(n)=n

Case 3: Average case time= all possible case time/no. of cases

It is difficult to find the average case time for any algorithm and generally, it is equivalent to worst case time.

= 1+2+3+… + n / n => [n(n+1)/2]/n= (n+1)/2

A(n) = (n+1)/2

Best, Worst, and Average case Analysis

Analyzing Asymptotic notations

Best case time

B(n)=1

B(n)=O(1)

B(n) = Ώ(1)

So, B(n) = ϑ(1)

Worst case time

w(n)=n

B(n)=O(n)

B(n) = Ώ(n)

So, B(n) = ϑ(n)

There is no fixed notation to represent best case or worst-case time. This can be represented by any of the asymptotic notations. It is not true that best case can always be represented by θ and worst case by O.