Analysis of Algorithms | Set 3 (Asymptotic Notations) - By GeeksForGeeks

We have discussed Asymptotic Analysis, and Worst, Average and Best Cases of Algorithms. The main idea of asymptotic analysis is to have a measure of efficiency of algorithms that doesn't depend on machine specific constants, and doesn't require algorithms to be implemented and time taken by programs to be compared. Asymptotic notations are mathematical tools to represent time complexity of algorithms for asymptotic analysis. The following 3 asymptotic notations are mostly used to represent time complexity of algorithms.
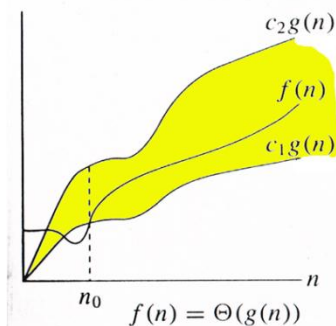
1) Θ Notation: The theta notation bounds a functions from above and below, so it defines exact asymptotic behavior.

A simple way to get Theta notation of an expression is to drop low order terms and ignore leading constants. For example, consider the following expression.

$3n^3 + 6n^2 + 6000 = \Theta(n^3)$

Dropping lower order terms is always fine because there will always be a n_0 after which $\Theta(n^3)$ has higher values than Θn^2) irrespective of the constants involved.

For a given function g(n), we denote Θ(g(n)) is following set of functions.

```
Θ(g(n)) = {f(n): there exist positive constants c1, c2 and n0 such

           that 0 <= c1*g(n) <= f(n) <= c2*g(n) for all n >= n0}
```

The above definition means, if f(n) is theta of g(n), then the value f(n) is always between c1*g(n) and c2*g(n) for large values of n (n >= n0). The definition of theta also requires that f(n) must be non-negative for values of n greater than n0.
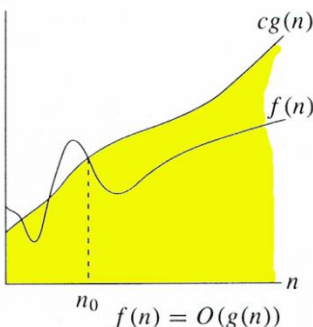
2) Big O Notation: The Big O notation defines an upper bound of an algorithm, it bounds a function only from above. For example, consider the case of Insertion Sort. It takes linear time in best case and quadratic time in worst case. We can safely say that the time complexity of Insertion sort is O(n^2). Note that O(n^2) also covers linear time.

If we use Θ notation to represent time complexity of Insertion sort, we have to use two statements for best and worst cases:
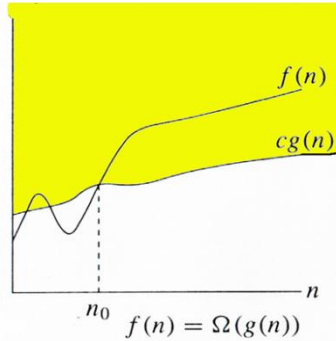
1. The worst case time complexity of Insertion Sort is Θ(n^2).
2. The best case time complexity of Insertion Sort is Θ(n).

The Big O notation is useful when we only have upper bound on time complexity of an algorithm. Many times we easily find an upper bound by simply looking at the algorithm.

```
O(g(n)) = { f(n): there exist positive constants c and
```

```
                    n0 such that 0 <= f(n) <= cg(n) for

                    all n >= n0}
```


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

3) Ω Notation: Just as Big O notation provides an asymptotic upper bound on a function, Ω notation provides an asymptotic lower bound. Ω Notation< can be useful when we have lower bound on time complexity of an algorithm. As discussed in the previous post, the best case performance of an algorithm is generally not useful, the Omega notation is the least used notation among all three.
For a given function g(n), we denote by Ω(g(n)) the set of functions.

```
Ω (g(n)) = {f(n): there exist positive constants c and

                    n0 such that 0 <= cg(n) <= f(n) for

                    all n >= n0}.
```

Let us consider the same Insertion sort example here. The time complexity of Insertion Sort can be written as Ω(n), but it is not a very useful information about insertion sort, as we are generally interested in worst case and sometimes in average case.

Exercise:
Which of the following statements is/are valid?
1. Time Complexity of QuickSort is $\Theta(n^2)$
2. Time Complexity of QuickSort is $O(n^2)$
3. For any two functions f(n) and g(n), we have f(n) = $\Theta(g(n))$ if and only if f(n) = $O(g(n))$ and f(n) = $\Omega(g(n))$.
4. Time complexity of all computer algorithms can be written as $\Omega(1)$

References:
Lec 1 | MIT (Introduction to Algorithms)
Introduction to Algorithms 3rd Edition by Clifford Stein, Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest