Time Complexity

Compiled by - Nazmus Sakib Akash

Time Complexity

In computer science, the **time complexity** is the computational complexity that describes the amount of computer time it takes to run an algorithm.

The time complexity of algorithms is most commonly expressed using the **big O notation**. It's an asymptotic notation to represent the time complexity.

Time Complexity is most commonly estimated by **counting the number of elementary steps** performed by any algorithm to finish execution.

Asymptotic Analysis

Asymptotic Analysis is the idea that handles above issues in analyzing algorithms.

In Asymptotic Analysis, we evaluate the performance of an algorithm in terms of **input size** (we don't measure the actual running time).

We calculate, **how the time taken by an algorithm increases with the input size.**

Asymptotic Analysis (Example)

Consider the search problem **(searching a given item)** in a sorted array and use **Linear Search** and **Binary Search** as the methods to solve this.

Assume that we run the Linear Search on a **fast computer A** and Binary Search on a **slow computer B.**

For **small values of input array size n**, the fast computer may take less time.

But, after **a certain value of input array size**, the Binary Search will definitely start taking less time compared to the Linear Search even though the Binary Search is being run on a slow machine.

The reason: the order of growth of Binary Search with respect to input size is **logarithmic** while the order of growth of Linear Search is linear. So the machine dependent constants can always be ignored after a certain value of input size.

Asymptotic Notations of Time Complexity

1. **Big Oh** denotes **"*fewer than or the same as*"** <expression> iterations.

   O(expression) is the set of functions that grow slower than or at the same rate as expression. It indicates the maximum required by an algorithm for all input values. It represents the worst case of an algorithm's time complexity.

2. **Big Omega** denotes **"*more than or the same as*"** <expression> iterations.

   Ω (expression) is the set of functions that grow faster than or at the same rate as expression. It indicates the minimum time required by an algorithm for all input values. It represents the best case of an algorithm's time complexity.

3. **Big Theta** denotes **"*the same as*"** <expression> iterations.

   Θ (expression) consist of all the functions that lie in both O(expression) and Omega(expression). It indicates the average bound of an algorithm. It represents the average case of an algorithm's time complexity.

Formula

1. O(1) = C *[Constant time complexity]*
2. n * O(1) = O(n) *[n is the input size]*
3. n * C = n * O(1) = O(n) *[Linear]*
4. q * O(p) = O(pq) *[Quadratic]*
5. O(n^p) *[Polynomial]*
6. O(m) + O(n) != O(m+n)
7. C * O(x) = O(Cx) = O(x)