**Analysis of Algorithm Efficiency, Asymptotic Notations, and Their Properties**

**1. Introduction to Algorithm Efficiency**

Algorithm efficiency refers to how well an algorithm performs in terms of time and space usage as the input size (n) increases. The efficiency of an algorithm is analyzed using:

* **Time Complexity:** The amount of time taken by an algorithm as a function of input size.
* **Space Complexity:** The amount of memory required by an algorithm.

**2. Asymptotic Notations**

Asymptotic notations help in describing the efficiency of an algorithm in terms of input size (n). The three most commonly used notations are:

**Big-O Notation (O)**

* Represents the **upper bound** of an algorithm’s complexity.
* Defines the worst-case scenario.
* Example: If an algorithm runs in at most **n²** time, we say it is **O(n²)**.
* **Example:** Bubble Sort has a worst-case complexity of **O(n²)**.



**If f(n) describes the running time of an algorithm, f(n) is O(g(n)) if there exist a positive constant C and n0 such that, 0 ≤ f(n) ≤ cg(n) for all n ≥ n0**

**Omega Notation (Ω)**

* Represents the **lower bound** of an algorithm’s complexity.
* Defines the best-case scenario.
* Example: If an algorithm takes at least **n log n** time, we write it as **Ω(n log n)**.
* **Example:** Quick Sort has a best-case complexity of **Ω(n log n)**.



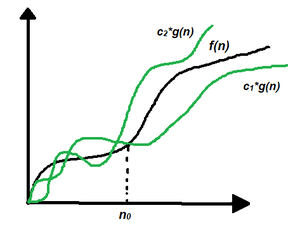
**Let g and f be the function from the set of natural numbers to itself. The function f is said to be Ω(g), if there is a constant c > 0 and a natural number n0 such that c\*g(n) ≤ f(n) for all n ≥ n0**

**Theta Notation (Θ)**

* Represents the **tight bound** (both upper and lower bound) of an algorithm.
* If an algorithm is both **O(f(n))** and **Ω(f(n))**, then it is **Θ(f(n))**.
* Example: If an algorithm always runs in **n log n** time, then it is **Θ(n log n)**.

**Let g and f be the function from the set of natural numbers to itself. The function f is said to be Θ(g), if there are constants c1, c2 > 0 and a natural number n0 such that c1\* g(n) ≤ f(n) ≤ c2 \* g(n) for all n ≥ n0**

* **Example:** Merge Sort runs in **Θ(n log n)** in all cases.



**3. Properties of Asymptotic Notations**

**1. Reflexive Property**

* For any function **f(n)**, **f(n) = O(f(n))**.
* Example: If **f(n) = n³**, then **f(n) = O(n³)**.

**2. Transitive Property**

* If **f(n) = O(g(n))** and **g(n) = O(h(n))**, then **f(n) = O(h(n))**.
* Example: If **f(n) = n³**, **g(n) = n⁴**, and **h(n) = n⁵**, then **f(n) = O(n⁵)**.

**3. Multiplicative Property**

* If **f(n) = O(g(n))**, then **a × f(n) = O(g(n))** (for any constant a).
* Example: If **f(n) = n²**, then **5 × f(n) = O(n²)**.

**Symmetric Properties:**If f(n) is Θ(g(n)) then g(n) is Θ(f(n))

**Transpose Symmetric Properties:**If f(n) is O(g(n)) then g(n) is Ω (f(n))

Example: If **f(n) = n**, and g(n)= **n²** then **f(n) = O(g(n))=O(n²)**and g(n)= Ω (f(n))= Ω (n)

**4. Order of Growth for Common Functions**

The increasing order of growth in time complexity is:

table:

|  |  |
| --- | --- |
| **Complexity** | **Name** |
| O(1) | Constant |
| O(log log n) | Double Logarithmic |
| O(log n) | Logarithmic |
| O(n^1/3) | Cube Root |
| O(n^1/2) | Square Root |
| O(n) | Linear |
| O(n log n) | Linearithmic |
| O(n^2) | Quadratic |
| O(n^3) | Cubic |
| O(2^n) | Exponential |
| O(n!) | Factorial |
| O(n^n) | Power of n |