Notations for Complexity: Ω, ω, θ, O, o

O Big-O Notation (O) - Upper Bound

Big-Omega Notation (Ω) - Lower Bound

Theta Notation (θ) - Tight Bound

Little-o Notation (o) - Strictly Upper Bound

Little-Omega Notation (ω) - Strictly Lower Bound

Complexity notations are used to analyze the performance of algorithms. They describe the upper, lower, and exact bounds of an algorithm's running time or space usage.

---

1. Big-O Notation (O) - Upper Bound

- **Definition**: Describes the worst-case scenario, the maximum time or space the algorithm can take.
- **Use Case**: Ensures the algorithm won't take more resources than specified, even in the worst case.

  ### Example:

  Suppose an algorithm processes n elements in a loop:

  ```
  void processArray(int n) {
     for (int i = 0; i < n; i++) {
        // O(1) operation
     }
  }
  ```

  Time Complexity:
  Each operation inside the loop is O(1).
  Total: O(n) (linear complexity).

2. Big-Omega Notation (Ω) - Lower Bound

- **Definition**: Describes the best-case scenario, the minimum time or space the algorithm will take.
- **Use Case**: Guarantees the algorithm will take at least this much time.

Example:

For a linear search:

```
int linearSearch(int arr[], int n, int key) {
   for (int i = 0; i < n; i++) {
      if (arr[i] == key)
         return i; // Found
   }
   return -1; // Not found
}
```

Time Complexity:

- Best-case: Key is at the first position. Ω(1).

3. Theta Notation (θ) - Tight Bound
- **Definition**: Describes the exact bound when the upper and lower bounds are the same.
- **Use Case**: Indicates the growth rate of an algorithm in all cases.

Example:
For a simple summation loop:

```
int sumArray(int arr[], int n) {
   int sum = 0;
   for (int i = 0; i < n; i++) {
      sum += arr[i];
   }
   return sum;
}
```

Time Complexity:
- The loop always runs n times.
- Best-case = Worst-case = θ(n).

4. Little-o Notation (o) - Strictly Upper Bound
- **Definition**: Describes the upper bound of the growth rate but is not tight. It excludes the possibility of the actual growth rate matching the bound.
- **Use Case**: Used for theoretical analysis when an algorithm grows slower than a certain rate.

Example:
Suppose an algorithm has n^2 log n operations.
- If we say T(n) = o(n^3), it means the algorithm grows slower than n^3, but it is not equal to n^3.

---

5. Little-Omega Notation (ω) - Strictly Lower Bound
- **Definition**: Describes the lower bound of the growth rate but is not tight. It excludes the possibility of the actual growth rate matching the bound.
- **Use Case**: Indicates an algorithm grows faster than a given rate.

Example:
If T(n)=n^2, we can say:
- T(n)=ω(nlogn) meaning it grows strictly faster than nlogn.

Summary Table

Notation	Definition	Use Case	Example
O	Upper bound	Worst-case performance	O(n^2) Sorting algorithms like Bubble Sort
OmegaΩ	Lower bound	Best-case performance	Ω(1) Key found in first position in linear search
thetaθ	Tight bound (upper = lower)	Exact growth rate	θ(n): Summation of array
o	Strictly upper bound	Growth slower than a rate	o(n^3): Algorithm slower than n^3

Notation	Definition	Use Case	Example
omegaω	Strictly lower bound	Growth faster than a rate	ω(nlogn): Algorithm grows faster than nnlogn

Real-world Analogy

- **Big-O**: "The maximum speed limit is 100 km/h."
- **Big-Omega**: "The minimum speed limit is 30 km/h."
- **Theta**: "The speed is always between 30 and 100 km/h."
- **Little-o**: "The speed is less than 100 km/h but never exactly 100 km/h."
- **Little-omega**: "The speed is more than 30 km/h but never exactly 30 km/h."

Complexity Notations as Cooking Times 🍳

1. **Big-O (O)**:
   - "It will take at most 30 minutes to cook the dish, even on a bad day."
2. **Big-Omega (OmegaΩ)**:
   - "It will take at least 10 minutes to cook this dish, even if you're super fast."
3. **Theta (thetaθ)**:
   - "This dish always takes exactly 20 minutes to cook, no matter what."
4. **Little-o (o)**:
   - "Cooking time is less than 30 minutes, but never exactly 30 minutes."
5. **Little-omega (omegaω)**:
   - "Cooking time is more than 10 minutes, but never exactly 10 minutes."