Understanding Algorithm Complexity

Cracking Coding Interview @ Ostad
- Partharaj Deb -

Complexity

Complexity refers to the measure of how the resources required by an algorithm (such as time and space) grow as the input size increases. It is a crucial aspect of algorithm analysis that helps us understand the efficiency and scalability of algorithms.

- Time Complexity
- Space Complexity

Time Complexity

Time Complexity is a measure of the amount of time an algorithm takes to complete as a function of the input size.

It provides an upper bound on the running time of an algorithm in the worst-case scenario. Time complexity is typically expressed using Big O notation.

Notations:

1. Big O notation (O)
2. Big Omega notation (Ω)
3. Big Theta notation (Θ)

Space Complexity

Space Complexity is a measure of the amount of memory an algorithm uses as a function of the input size.

It describes the growth in memory consumption as the input size increases. Similar to time complexity, space complexity is expressed using Big O notation.

For example:

1. O(1) represents constant space complexity.
2. O(n) represents linear space complexity.
3. O(n^2) represents quadratic space complexity.

Big O Notation

Big O (O): Represents the upper bound of the algorithm's running time.

O(1): Constant time complexity.

O(log n): Logarithmic time complexity.

O(n): Linear time complexity.

O(n log n): Linearithmic time complexity.

O(n^2), O(n^3), ... : Polynomial time complexity.

O(2^n), O(n!): Exponential time complexity.

Examples of
Time Complexity

Examples - Time Complexity

O(1): Constant time complexity.

```
void constantTimeExample() {
    printf("Hello, World!\n");
}
```

Examples - Time Complexity

O(n): Linear time complexity.

```
void linearTimeExample(int n) {
    for (int i = 0; i < n; i++) {
        printf("%d ", i);
    }
    printf("\n");
}
```

Examples - Time Complexity

O(log n): Logarithmic time complexity.

```
void logarithmicTimeExample(int n) {
    int i = 1;
    while (i < n) {
        printf("%d ", i);
        i *= 2;
    }
    printf("\n");
}
```

Examples - Time Complexity

O(n^2) : Polynomial time complexity

```
void quadraticTimeExample(int n) {
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            printf("(%d, %d) ", i, j);
        }
        printf("\n");
    }
}
```

Examples - Time Complexity

O(n log n) : Linearithmic time complexity

```
void linearithmicTimeExample(int n) {
    for (int i = 0; i < n; i++) {
        for (int j = 1; j <= n; j*=2) {
            printf("(%d, %d) ", i, j);
        }
        printf("\n");
    }
}
```

Examples of
Space Complexity

Examples - Space Complexity

O(1) - Constant Space Complexity

```
void constantSpaceExample() {
    int x = 5;
    printf("Value of x: %d\n", x);
}
```

Examples - Space Complexity

O(n) - Linear Space Complexity

```
void linearSpaceExample(int n) {
    int array[n];
    for (int i = 0; i < n; i++) {
        array[i] = i;
        printf("%d ", array[i]);
    }
    printf("\n");
}
```

Examples - Space Complexity

O(n^2) - Quadratic Space Complexity:

```
void quadraticSpaceExample(int n) {
    int matrix[n][n];
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            matrix[i][j] = i + j;
            printf("%d ", matrix[i][j]);
        }
        printf("\n");
    }
}
```