Table of Contents

What Is Theta Notation

Theta notation (Θ) is a mathematical notation used in computer science to describe the tight bound or the asymptotically tight behavior of an algorithm or function.

It represents both the upper bound and the lower bound of the algorithm's time complexity or space complexity.

In Theta notation, we use the symbol "Θ" followed by a function.

The function typically represents the number of operations performed by the algorithm or the amount of space required.

Some Commonly Used Theta Notations:

1. Θ(1):

This notation represents a constant tight bound. It indicates that the algorithm takes a

constant amount of time, regardless of the input size. Both the upper and lower bounds are the same.

2. Θ(log n):

This notation represents a logarithmic tight bound. It indicates that the algorithm takes time proportional to the logarithm of the input size. Both the upper and lower bounds grow logarithmically.

3. Θ(n):

This notation represents a linear tight bound. It indicates that the algorithm takes time proportional to the input size. Both the upper and lower bounds grow linearly.

4. Θ(n log n):

This notation represents a tight bound that grows faster than linear but slower than quadratic time. It indicates that the algorithm takes time proportional to n log n with respect to the input size. Both the upper and lower bounds are the same.

5. Θ(n^2):

This notation represents a quadratic tight bound. It indicates that the algorithm takes time proportional to the square of the input size. Both the upper and lower bounds are the same.

Analyze The Time Complexity Of The Algorithm, Using Theta Notation.

Example 1:

C

```c
#include <stdio.h>

int find_max(int arr[], int length) {
    int max_value = arr[0];  // Assume the first element is the
maximum

    for (int i = 1; i < length; i++) {
        if (arr[i] > max_value) {
            max_value = arr[i];
        }
    }

    return max_value;
}

int main() {
    int arr[] = {5, 8, 2, 10, 3};
    int length = sizeof(arr) / sizeof(arr[0]);

    int max = find_max(arr, length);
    printf("Maximum value: %d\n", max);

    return 0;
```

```
}
```

1. The sizeof operator used to calculate the length of the array takes constant time and can be considered Θ(1).
2. The find_max function iterates through the array elements to find the maximum value. The loop runs for length – 1 iterations, where length represents the number of elements in the array.
3. Inside the loop, the comparison operation arr[i] > max_value and the assignment operation max_value = arr[i] both take constant time. Thus, they can be considered Θ(1).
4. The printf function used to print the maximum value also takes constant time and can be considered Θ(1).

Therefore, the time complexity of the find_max function is Θ(length).

Example 2:

C

```c
#include <stdio.h>

int main() {
    int i;
    for (i = 1; i <= 10; i++) {
        printf("%d\n", i);
```

```
    }
    return 0;
}
```

1. The for loop in the code snippet always runs for a fixed number of iterations, specifically from 1 to 10. It does not depend on any variable or input size. Therefore, the loop has a constant time complexity, Θ(1).
2. Inside the loop, the printf function is called to print the value of i. The printf function takes constant time as it performs a fixed number of operations. Thus, it can be considered Θ(1).

As a result, the time complexity using Theta notation (Θ) in the given code is Θ(1).

Example 3:

C

```c
#include <stdio.h>

int main() {
    int i,n;
    printf("Enter a number");
    scanf("%d",&n);
    for (i = 1; i <= n; i++) {
        printf("%d\n", i);
    }
    return 0;
```

```
}
```

1. The scanf function for getting user input takes constant time and can be considered Θ(1).
2. The for loop iterates from 1 to n, where n represents the user input. The loop runs n times.
3. Inside the loop, the printf function prints the value of i. The printf function takes constant time as it performs a fixed number of operations. Thus, it can be considered Θ(1).

Therefore, the time complexity of the for loop is Θ(n).