complexity analysis.
● Total time complexity of a program is equal to the summation of all the running time of disconnected fragments.

Need for Time Complexity

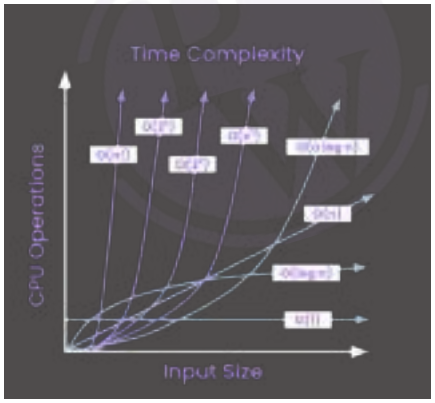
When analyzing any algorithm, we need to evaluate the effectiveness of that algorithm. Accordingly, we need to prefer the most optimized algorithm so as to save the time taken by it to execute. An example for the same could be linear search and binary search. Let's suppose we need to search a given value in a sorted array of size 8. This would take 8 iterations for linear search whereas it would just take log(8) ~ 3 iterations for binary search to search the same element, thereby saving a lot of time. Time Complexity for both binary search and linear search will be discussed in further lectures.

Types of Time Complexity Analysis:

a. Worst Case Time Complexity:
It is that case where the algorithm takes the longest time to complete its execution. It is represented by the Big-O notation. By default we calculate the worst case time complexity when we refer to the time complexity of a code.

b. Best Case Time Complexity: It is that case where the algorithm takes the least time to complete its execution. It is represented by the Omega Notation (Ω-Notation).

c. Average Case Time Complexity: As the name suggests, it gives average time for a program to complete its execution. It is represented by the theta notation (Θ-Notation).



The above graph represents how an increase in input size has an impact on the no. of CPU operations of a program for different time complexities of the program.

Constant Time Complexity	$O(1)$
Logarithmic Complexity	$O(\log_n)$
Linear Complexity	$O(n)$
Quadratic Complexity	$O(n^2)$
Cubic Complexity	$O(n^3)$
Polynomial complexity	$O(n^m), m>3$
Exponential Complexity	$O(a^n), a>1$

The above table shows different types of time complexities which are possible in a program. This list is not exhaustive and there are many more types of time complexities which are possible which is due to different parts of programs having different individual time complexities.

Let's now see some examples to understand how we calculate time complexity for different codes.

Example 1

Code:

```
for (int i = 0; i < n; i++) {
    // some operation of O(1) complexity
}
```

Explanation:
The above loop has a time complexity of O(n), where n is the number of iterations as the loop is running for n iterations.

Example 2

Code:

```
int i = 0;
while (i < n) {
    // some operation of O(1) complexity
    i++;
}
```

Explanation
The above while loop has a time complexity of O(n), where n is the number of iterations as the loop will run for n times.

Example 3

Code:

```
for (int i = 0; i < n; i++) {
    for (int j = 0; j < n; j++) {
        // some operation
    }
}
```

Explanation:
The above nested for loop has a time complexity of O(n^2), where n is the number of iterations of the outer loop. The reason for this complexity is that the inner loop is running for n times for each outer loop iteration making the whole time complexity quadratic.