Time Complexity

hitbullseye

Definition

- The time complexity for a program or algorithm defines as the amount of time it takes for each statement to complete.
- The time taken for a program to complete its execution.
- It's the number of times each statements gets executed to get the desired output.

Time-complexity can be expressed in three Notations.

a) Big -O Notation (O)
b) Big – Omega(Ω)
c) Big – Theta(θ)

- **Time Complexity depends upon three factors:**

1. Best-Case
2. Worst –Case
3. Average-Case

Best Case defined as minimum number of operations to be executed in a program.

Worst Case defined as maximum number of operations to be executed in a program.

Average-Case defined as all possible inputs and calculate the computing time for all of the inputs. It is calculated as sum of all values which is calculated divide by total number of inputs.

a. **Big O notation** is used most of the times, because it will give us the execution time in worst case inputs. So, we consider worst-case complexity because it requires the maximum amount of time for inputs of a given size. It represents upper bound of the run time of program.
b. **Big – Omega(Ω)** notation is used to describe the best case running time for a given algorithm. It represents lower bound of the run time of program.
c. **Big – Theta(θ)** is used to define consist of all the functions that lie in both Big O and Big Ω notations. It represents the average case of an algorithm's time complexity. It denotes upper and lower bound of the run time of the program.

Program:

```
#include <stdio.h>
void main()
{
int i, n = 15;
for (i = 1; i <= n; i++) {
printf(“Time Complexity");
} }
```

Here , the loop will execute n times . So, the time complexity of this program is O(n) where O is denoted as Big O notation.

Time complexity for loops:

- Time complexity of a loop when the loop variable is incremented or decremented.

e.g. int i = 1;

```
do
  {
      i++;
  }while(i<=n);
```

Time complexity is : O(n) where n denotes number of times the loop will execute and i represents loop variable.

- When the loop variable is divided or multiplied by a constant.

e.g. int i=1;

```
do
{
  i = i*a;
 }while(i<=n);
```

Time complexity is : O(log n) where n denotes number of times the loop will execute, i represents loop variable and a denotes constant.

- For Nested Loop

e.g. int j=0;

```
do{
      do{  int k =0;
                  k++;
           }while(k<=i);
       j++;
      }while(j<=n-1);
```

Here, j is outer loop and k is inner loop.Time complexity = number of times the innermost statement is to be executed.

No. of times inner loop executed= sum of integers(0,1……..n-1)=n^2/2 - n/2

Time complexity=O(n

Exercise:

Find the time complexity of:

I.
```
for(var i=0;i<n;i++)
for(var j=0;j<i;j++)
value += 1;
```

II.
```
for (int i = 0; i < N; i++)
{ printf(Time Complexity"); }
```

References URL:

- https://en.wikipedia.org/wiki/Best,_worst_and_average_case
- https://www.javatpoint.com/big-o-notation-in-c
- https://www.hackerearth.com/practice/basic-programming/complexity-analysis/
- https://www.geeksforgeeks.org/miscellaneous-problems-of-time-complexity/

THANK YOU

hitbullseye