TUTORIAL-2

Example-1: Suppose we are analyzing two types of algorithms (say algorithm-A & algorithm-B). The running time of algorithm-A is $0.01*(\log_2 n)^2$ ms and algorithm-B is 2 ms. We want to find out for what value of n (here n is input size) the algorithm-B is faster.

Faster algorithm means the algorithm which takes less amount of time to execute.
In this case to become the algorithm faster,

$2 < 0.01*(\log_2 n)^2$

$0.01*(\log_2 n)^2 > 2$

$\log_2 n > v2*10$

$n > 2^{v2*10}$

$n > 18080$

So, the algorithm-B is faster for the values greater than 18080.

Example-2: Suppose we have an algorithm with a complexity of $O(\log_{10}\log_{10} n)$. The running time of the algorithm is 10 ms for input size $n=10^3$. What should be the running time of that algorithm for $n=10^{17}$?

We can calculate the running time from the complexity of the algorithm using the function, $T(n) = k* \log_{10}\log_{10} n$

For $T(n) = 10$ ms and $n = 10^3$.

$k = 10/ (\log_{10}\log_{10} 10^3) = 10/3$

When $n=10^{17}$
$T(n) = k* \log_{10}\log_{10} n = (10/3) * \log_{10}\log_{10} 10^{17} = 25.76$ ms.

Example-3: Suppose we have an algorithm of running time $T(n) = \log_m n$. We have to select the value of m either 5 or 7. What value of m should we select to get lowest running time of the algorithm ?

We can write $\log_m n = \ln n/\ln m$

$T(n) = \ln n/\ln m$

For m=5, $T(n) = 0.6062*\ln n$

For m=7, T(n) =0.5139*ln n

So we will select m=7 to get lowest running time.

Example-4: We want to calculate the running time and Big-oh of the following algorithm.

```
for(int i= n; i > 0: i= i/2)
    for(int j= 0; j <i: j++)
        System.out.println("415.220");
```

Let us assume that $n = 2^m$ where m is a constant.

In this example we should calculate the number of execution of println() method considering both the loop at a time because the inner loop variable j is dependent on outer loop variable i.

For i= n the inner loop will execute for n ($=2^m$) times.
For i= n/2 the inner loop will execute for n/2 ($=2^{m-1}$) times.
For i= n/4 the inner loop will execute for n/4 ($=2^{m-2}$) times.
……………………………………………………………..

For i= 1 the inner loop will execute for 1 ($=2^0$) times.

Total number of execution of println() method is

$2^0 + 2^1 + 2^2 + \ldots\ldots\ldots + 2^{m-1} + 2^m = 2^{m+1} - 1 = 2 * 2^m - 1 = 2n - 1$

Running time of the above algorithm T(n) = k(2n-1) where k is the constant time required to execute println() method one time .

T(n) is O(n).