

DAY-1: 19-10-2020 **MODULE -1 : ASSIGNMENT -1**

## What is the time complexity of following function fun()?

Assume that log(x) returns log value in base 2. void fun()

{

int i, j;

**Time complexity = O (n) \* O (log(n)) = O (n log(n))**

for (i=1; i<=n; i++)

for (j=1; j<=log(i); j++) printf("Welcome to the course");

}

## What is the time, space complexity of following code:

int a = 0, b = 0;

for (i = 0; i < N; i++) { a = a + small();

**Time complexity = O(N) + O(M) = O(n) Space complexity = O(1)**

}

for (j = 0; j < M; j++) { b = b + small();

}

## What is the time complexity of following code:

int a = 0;

**Time complexity = n(n+1)/2 = O(n2)**

for (x = 0; i < N; x++)

{

for (y = N; y > x; y--)

{

a = a + x + y;

}

}

## What is the time complexity of following code:

int i, j, k = 0;

**Time complexity = n/2 \* log(n) = O (n log(n))**

for (i = n / 2; i <= n; i++) { for (j = 2; j <= n; j = j \* 2) {

k = k + n / 2;

}

}

## What is the complexity of the code given below?

a.

**Time complexity = log(n) (‘i’ increase exponentially) Space complexity= O (1) (since only one integer)**

for (int i = 1; i <=n; i \*= c) {

// some O(1) expressions

}

b.

**Time complexity = log(n) (‘i’ decrease exponentially) Space complexity= O (1) (since only one integer)**

for (int i = n; i > 0; i /= c) {

// some O(1) expressions

}

## What is the complexity of the code given below?

1. // Here d is a constant greater than 1 for (int i = 2; i <=n; i = pow(i, d)) {

**Time complexity = log(log(n)**

**(‘i’ increase exponentially by constant ) Space complexity= O (1)**

**(only one integer are declared**

// some O(1) expressions

}

1. //Here fun is sqrt or cuberoot or any other constant root for (int i = n; i > 1; i = fun(i)) {

**Time complexity = log(log(n))**

**(the value of ‘i’ decrease exponentially)**

**Space complexity= O (1) (only one integer are declared)**

// some O(1) expressions

}

## What is the complexity of the code given below?

while (x > 0) { x /= 2;

**Time complexity = log(n)**

**(if the value of ‘i’ decrease exponentially) Time complexity = log(1)**

**(if the value of x is zero)**

}

## What is the complexity of the code given below?

function O\_SQRT(n)

{

count = 0;

for (var i = 1; i \* i < n; i++)

**Time complexity = log(n) (i increases exponentially)**

{

count++;

}

return count

}

## Arrange the following order of complexity of algorithms in increasing order of growth.

1. **Constant time**
2. **Logarithmic time**
3. **Linear time**
4. **Polynomial time**
5. **Exponential time**
6. **Factorial time**
   * Constant time
   * Linear time
   * Logarithmic time
   * Polynomial time
   * Exponential time
   * Factorial time

# M.Sathishkumar



# E0119052

# B.Tech CSE – AI & ML (II year)

# Design and Analysis of Algorithms