By Karan V (E0119039)

SRI RAMACHANDRA
INSTITUTE OF HIGHER EDUCATION AND RESEARCH
(Category - I Deemed to be University) Porur, Chennai
SRI RAMACHANDRA ENGINEERING AND TECHNOLOGY

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

**1. What is the time complexity of following function fun()?**

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

```
void fun()  //Θ(1) function declaration
{
   int i, j;  //Θ(1) variable declaration
for (i=1; i<=n; i++)  //Θ(n) ->for loop
      for (j=1; j<=log(i); j++) //Θ(n log(n))
         printf("Welcome to the course"); //Θ(1)
}
```

Answer:

Θ(n)xΘ(logn)

Time complexity: Θ(n logn)

**2. What is the time, space complexity of following code:**

```
int a = 0, b = 0;
for (i = 0; i < N; i++) {
    a = a + small();
}
for (j = 0; j < M; j++) {
    b = b + small();
}
```

Space complexity: Θ(1)

Time complexity: Θ(M + N)

**3. What is the time complexity of following code:**

```
int a = 0;
for (x = 0; i < N; x++)      //n times
{
    for (y = N; y > x; y--)      //(n+1)/2 times
        {
```

Answer:$\Theta(n^2)$

```
        a = a + x + y;
    }
}
```

**4. What is the time complexity of following code:**

```
int i, j, k = 0;
for (i = n / 2; i <= n; i++) {      //runs n/2
    for (j = 2; j <= n; j = j * 2) {     //logn
        k = k + n / 2;
    }
}
```

Answer: θ(nlogn)

**5) What is the complexity of the code given below?**

a.

```
for (int i = 1; i <=n; i *= c) {
      // some O(1) expressions
  }
```

Explanation: i increase exponentially
Answer: θ(log(n))

b.

```
  for (int i = n; i > 0; i /= c) {
      // some O(1) expressions
  }
```

Explanation: i decreases exponentially
Answer:θ(logn)

**6) What is the complexity of the code given below?**

a.

```
// Here d is a constant greater than 1
for (int i = 2; i <=n; i = pow(i, d)) {
      // some O(1) expressions
  }
```

Explanation: $2, 2^k, (2^k)^k = 2k^2, (2^{k^2})^k = 2^{k^3}, \ldots, 2^{k^{\log_k(\log(n))}}$

Answer: θ(log(logn))

b. //Here fun is sqrt or cuberoot or any other constant root

```
for (int i = n; i > 1; i = fun(i)) {
    // some O(1) expressions
}
```

Explanation: n, $n^{1/k}$, $(n^{1/k})^{1/k} = n^{1/k^2}$, $n^{1/k^3}$, …, $n^{1/k^{(\log k(\log(n)))}}$

Answer:0(log(logn))

7) **What is the complexity of the code given below?**

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

Explanation:i decreases exponentially
Answer:0(logn)

8) **What is the complexity of the code given below?**

```
function O_SQRT(n)
{
   count = 0;      //1 time
   for (var i = 1; i * i < n; i++)
        {

      count++;
   }
   return count
}
```

Explanation: i increases exponentialy
Answer:0(logn)

9**) Arrange the following order of complexity of algorithms in increasing order of growth.**

- Constant time
- Linear time
- Logarithmic time
- Polynomial time
- Exponential time
- Factorial time

Answer:

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