

# **GUIDELINES FOR ASYMPTOTIC ANALYSIS- PART 1**

**[ BASED ON SOME SUB-CODES OF PROGRAMS]**

**TILL NOW WE HAVE LEARNT THAT:**

**Big – Oh or Worst Case Complexity:**

*$f(n) \leq cg(n)$ , where  $c$  and  $n_0$  are constants and  $n \geq n_0$ , then  $f(n) = O(g(n))$*

**Big – Omega or Best Case Complexity:**

*$f(n) \geq cg(n)$ , where  $c$  and  $n_0$  are constants and  $n \geq n_0$ , then  $f(n) = \Omega(g(n))$*

**Big – Theta or Average Case Complexity:**

*$c_1g(n) \leq f(n) \leq c_2g(n)$ , where  $c, n_1$  and  $n_2$  are constants and  $n \geq \{n_1, n_2\}$ ,  
then  $f(n) = \Theta(g(n))$*

And from an algorithm the priority is to find Worst Case Time Complexity.

- 1. LOOPS:** The running time of a loop is, at most, the running time of the statements inside the loop (including tests) multiplied by the number of iterations.

```
//executes n times  
for(i = 1; i ≤ n; i ++){  
    m = m + 2; //constant time, c  
}
```

$$\text{Total time} = \text{constant time } c \times n = O(n).$$

- 2. NESTED LOOPS:** Analyse from the inside out. Total running time is the product of the sizes of all the loops.

### **1<sup>ST</sup> TYPE**

```
//outer loop executed n times  
for(i = 1; i ≤ n; i ++){  
    //inner loop executes n times  
    for(j = 1; j ≤ n; j ++){  
        k = k + 1 ; // constant time.  
    }  
}
```

So, 1<sup>st</sup> For loop will be executed  $n$  times.

*Or, 2nd For Loop will be executed as depending upon the first for loop as shown below:*

i=1	i=2	i = 3	i = 4	.....	i =n
j= 1 to n	j= 1 to n	j= 1 to n	j= 1 to n	.....	j= 1 to n

Hence inner loop also executes n times:

Which gives us the Multiplication rule:

$$\text{Total Time} = c \times n \times n = cn^2 = O(n^2)$$

## Findings:

We can say *outer loop executes 1 to n times =  $O(n)$  for outer loop.*

Again, we can say inner loop runs *n times 1 to n =  $1 + 2 + 3 + 4 + \dots + n - 1 + n$  times*

$$= \frac{n(n+1)}{2} = \frac{n^2 + n}{2} = O\left(\frac{1}{2} \times n^2 + \frac{1}{2} \times n\right)$$

By Constant Rule:  $O(k \times n) = O(n)$ , where *k is constant.*

$$O\left(\frac{1}{2} \times n^2 + \frac{1}{2} \times n\right) = O(n^2 + n)$$

By addition rule:

$$O(n) + O(n^2) = O\{\max(n^2 + n)\} = O(n^2)$$

Hence inner loop dominates over outer loop i.e.

By addition rule(Outer Loop + Inner Loop) :

$$O(n) + O(n^2) = O\{\max(n^2 + n)\} = O(n^2)$$

## 2<sup>ND</sup> TYPE

```
//outer loop executed n times
for(i = 1; i ≤ n; i++){
    //inner loop executes n times
    for(j = 1; j ≤ i; j++){
        k = k + 1 ; // constant time.
    }
```

So, 1<sup>st</sup> For loop will be executed  $n$  times.

*Or, 2nd For Loop will be executed as depending upon the first for loop as shown below:*

i=1	i=2	i = 3	i = 4	.....	i =n
j= 1 to i	j= 1 to i	j= 1 to i	j= 1 to i	.....	j= 1 to i

Hence inner loop executes  $n$  times:

$$\text{Total Time} = c \times n \times n = cn^2 = O(n^2)$$

## Findings:

We can say *outer loop executes 1 to n times* =  $O(n)$  for outer loop.

Again, we can say inner loop runs *n times 1 to i* =  $1 + 2 + 3 + 4 + \dots + n - 1 + n$  times

$$= \frac{n(n+1)}{2} = \frac{n^2 + n}{2} = O\left(\frac{1}{2} \times n^2 + \frac{1}{2} \times n\right)$$

By Constant Rule:  $O(k \times n) = O(n)$ , where  $k$  is constant.

$$O\left(\frac{1}{2} \times n^2 + \frac{1}{2} \times n\right) = O(n^2 + n)$$

By addition rule:

$$O(n) + O(n^2) = O\{\max(n^2 + n)\} = O(n^2)$$

Hence inner loop dominates over outer loop i.e.

By addition rule(Outer Loop + Inner Loop) :

$$O(n) + O(n^2) = O\{\max(n^2 + n)\} = O(n^2)$$

### 3<sup>RD</sup> TYPE

```
//outer loop executed n times
for(i = 1; i ≤ n; i++){
    //inner loop executes n times
    for(j = 1; j ≤ i/2; j++){
        k = k + 1 ; // constant time.
    }
}
```

So, 1<sup>st</sup> For loop will be executed  $n$  times.

Or, 2<sup>nd</sup> For Loop will be executed as depending upon the first for loop as shown below:

i=1	i=2	i = 3	i = 4	.....	i =n
j= 1 to i/2	j= 1 to i/2	j= 1 to i/2	j= 1 to i/2	.....	j= 1 to i/2

Hence inner loop executes  $n$  times:

$$\text{Total Time} = c \times n \times n = cn^2 = O(n^2)$$

### Findings:

We can say outer loop executes 1 to  $n$  times =  $O(n)$  for outer loop.

Again, we can say inner loop runs  $n$  times  $\frac{i}{2}$ .

Hence total number of iterations that inner loop will run:

$$\Rightarrow \left( \frac{1}{2} + \frac{2}{2} + \frac{3}{2} + \cdots + \frac{n}{2} \right) \text{ times}$$

$$\Rightarrow \frac{1}{2} (1 + 2 + 3 + \cdots + n) \text{ times}$$

$$= \frac{1}{2} \left( \frac{n(n+1)}{2} \right) = \frac{1}{2} \left( \frac{n^2 + n}{2} \right) = \frac{(n^2 + n)}{4}$$

By addition rule:

$$\Rightarrow O\left(\frac{1}{4} \times n\right) + O\left(\frac{1}{4} \times n^2\right)$$

By Constant Rule:  $O(k \times n) = O(n)$ , where  $k$  is constant.

$$\Rightarrow O(n) + O(n^2)$$

$$\Rightarrow O(n) + O(n^2) = O\{\max(n^2 + n)\} = O(n^2)$$

Hence inner loop dominates over outer loop i.e.

By addition rule (Outer Loop + Inner Loop):

$$O(n) + O(n^2) = O\{\max(n^2 + n)\} = O(n^2)$$

#### 4<sup>TH</sup> TYPE

```
//outer loop executed n times
for(i = 1; i ≤ n; i++){
    //inner loop executes n times
    for(j = 1; j ≤ n - 1; j++){
        k = k + 1 ; // constant time.
    }
```

So, 1<sup>st</sup> For loop will be executed  $n$  times.

Or, 2<sup>nd</sup> For Loop will be executed as depending upon the first for loop as shown below:

i=1	i=2	i = 3	i = 4	.....	i =n
j= 1 to n-1	j= 1 to n-1	j= 1 to n-1	j= 1 to n-1	.....	j= 1 to n-1

Hence inner loop executes  $n$  times:

$$\text{Total Time} = c \times n \times n = cn^2 = O(n^2)$$

### Findings:

We can say *outer loop executes 1 to n times* =  $O(n)$  for outer loop.

Again, we can say inner loop runs  $n$  times  $n - 1$ .

Hence total number of iterations that inner look will run:



$\Rightarrow (1 + 2 + 3 + \cdots + n - 1)$  *times*

*By arithmetic series:*

$$\Rightarrow S(n) = \frac{n}{2} \left( (2 \times \text{First Term}) - ((n - 1) \times (T_{n+1} - T_n)) \right)$$

*Where ,  $a = \text{First Term}$ .*

$$d = (T_{n+1} - T_n)$$

$$T_{n+1} = \text{Second Last term} \Rightarrow n - 1.$$

$$T_n = \text{Last Term} \Rightarrow n.$$

$$n - 1 = \text{Second last term i. e. } T_{n+1}.$$

*We can rewrite the formula as:*

$$S_n = \frac{n}{2} (2a - (n - 1)d)$$

$$d = (n - 2) - (n - 1) = -1$$

$$a = 1$$

$$S_n = \frac{n}{2} \left( (2 \times 1) - ((n - 1) \times -1) \right)$$

$$= \frac{n}{2} (2 - (-n + 1))$$

$$= \frac{n}{2} (2 + n - 1)$$

$$= \frac{n(n + 1)}{2}$$

Therefore:

$$= \frac{n(n+1)}{2} = \frac{n^2 + n}{2} = O\left(\frac{1}{2} \times n^2 + \frac{1}{2} \times n\right)$$

By Constant Rule:  $O(k \times n) = O(n)$ , where  $k$  is constant.

$$O\left(\frac{1}{2} \times n^2 + \frac{1}{2} \times n\right) = O(n^2 + n)$$

By addition rule:

$$O(n) + O(n^2) = O\{\max(n^2 + n)\} = O(n^2)$$

Hence inner loop dominates over outer loop i.e.

By addition rule (Outer Loop + Inner Loop):

$$O(n) + O(n^2) = O\{\max(n^2 + n)\} = O(n^2)$$

## 5<sup>TH</sup> TYPE

```
//outer loop executed n times
for(i = 1; i ≤ n; i++){
    //inner loop executes n times
    for(j = 1; j ≤ n - k; j++){
        k = k + 1 ; // constant time.
    }
```

So, 1<sup>st</sup> For loop will be executed  $n$  times.

*Or, 2nd For Loop will be executed as depending upon the first for loop as shown below:*

i=1	i=2	i = 3	i = 4	.....	i =n
j= 1 to n-k	j= 1 to n-k	j= 1 to n-k	j= 1 to n-k	.....	j= 1 to n-k

Hence inner loop executes  $n$  times:

$$\text{Total Time} = c \times n \times n = cn^2 = O(n^2)$$

## Findings:

We can say *outer loop executes 1 to  $n$  times* =  $O(n)$  for outer loop.

Again, we can say inner loop runs  $n$  times  $n - k$ .

Hence total number of iterations that inner look will run:

$$\Rightarrow (n - k + n - k - 1 + n - k - 2 + \dots + 3 + 2 + 1) \text{ times}$$

*By arithmetic series:*

$$\Rightarrow S(n) = \frac{n}{2} ((2 \times \text{First Term}) - ((n - 1) \times (T_{n+1} - T_n)))$$

Where ,  $a = \text{First Term}$ .

$$d = (T_{n+1} - T_n)$$

$$T_{n+1} = \textit{Second Last term}$$

$$T_n = \textit{Last Term}$$

*We can rewrite the formula as:*

$$S_n = \frac{n}{2}(2a - (n - 1)d)$$

$$d = (n - k - 1) - (n - k) = -1$$

$$a = 1$$

$$S_n = \frac{n}{2}((2 \times 1) - (n - k - 1) \times -1)$$

$$S_n = \frac{n}{2}((2 \times 1) - (n - k - 1) \times -1)$$

$$= \frac{n}{2}((2) - (-n + k + 1))$$

$$= \frac{n}{2}(2 + n - k - 1)$$

$$= \frac{n}{2}(1 + n - k)$$

$$= \frac{n + n^2 - k}{2}, \textit{for } k \textit{ is some constant}$$

$$\approx O\left(\frac{1}{2} \times n + \frac{1}{2} \times n^2 - \frac{1}{2} \times k\right)$$

By Constant Rule:  $O(k \times n) = O(n)$ , where  $k$  is constant.

$$\approx O(n) + O(n^2) + O(k) \text{ [ Here } k \text{ is constant ]}$$

$$\approx O(n) + O(n^2) + O(1)$$

$$\approx O(n) + O(n^2) + O(1) = O\{\max(n^2 + n + 1)\} = O(n^2)$$

Hence inner loop dominates over outer loop i.e.

By addition rule (Outer Loop + Inner Loop):

$$O(n) + O(n^2) = O\{\max(n^2 + n)\} = O(n^2)$$

## 6<sup>TH</sup> TYPE

```
//outer loop executed n times
for(i = 1; i ≤ n; i++){
    //inner loop executes n times
    for(j = 1; j ≤ n/2; j++){
        k = k + 1 ; // constant time.
    }
}
```

So, 1<sup>st</sup> For loop will be executed  $n$  times.

Or, 2<sup>nd</sup> For Loop will be executed as depending upon the first for loop as shown below:

i=1	i=2	i = 3	i = 4	.....	i =n
j= 1 to n/2	j= 1 to n/2	j= 1 to n/2	j= 1 to n/2	.....	j= 1 to n/2

Hence inner loop executes n times:

$$\text{Total Time} = c \times n \times n = cn^2 = O(n^2)$$

## Findings:

We can say *outer loop executes 1 to n times =  $O(n)$  for outer loop.*

Again, we can say inner loop runs *n times 1 to n/2.*  
Hence total number of iterations that inner look will run:

$$\Rightarrow \left( \frac{n}{2} + \frac{n}{2} - 1 + \dots + 3 + 2 + 1 \right) \text{ times}$$

*By arithmetic series:*

$$\Rightarrow S(n) = \frac{n}{2} \left( (2 \times \text{First Term}) - \left( \left( \frac{n}{2} - 1 \right) \times (T_{n+1} - T_n) \right) \right)$$

*Where , a = First Term.*

$$d = (T_{n+1} - T_n) = \left( \frac{n}{2} - 1 \right) - \frac{n}{2} = \frac{n-2}{2} - \frac{n}{2} = \frac{-2}{2} = -1$$

$$T_{n+1} = \text{Second Last term} = \frac{n}{2} - 1 = \frac{n-2}{2}.$$

$$T_n = \text{Last Term} = \frac{n}{2}$$

*We can rewrite the formula as:*

$$\Rightarrow S(n) = \frac{n}{2} \left( (2 \times 1) - \left( \left( \frac{n-2}{2} \right) \times (-1) \right) \right)$$

$$\Rightarrow S(n) = \frac{n}{2} \left( 2 - \left( \frac{-n+2}{2} \right) \right)$$

$$\Rightarrow S(n) = \frac{n}{2} \left( \frac{2+n-2}{2} \right)$$

$$\Rightarrow S(n) = \frac{n}{2} \left( \frac{n}{2} \right)$$

$$\Rightarrow S(n) = \frac{n^2}{2}$$

**By Constant Rule:**  $O(k \times n) = O(n)$  , *where k is constant.*

$$O\left(\frac{1}{2} \times n^2\right) = O(n^2)$$

Hence inner loop dominates over outer loop i.e.

**By addition rule (Outer Loop + Inner Loop):**

$$O(n) + O(n^2) = O\{\max(n^2 + n)\} = O(n^2)$$

## 7<sup>TH</sup> TYPE

```
//outer loop executed n times
for(i = 1; i ≤ n; i++){
    //inner loop executes n times
    for(j = 1; j ≤ k; j++){
        k = k + 1 ; // constant time.
    }
}
```

Now from above prove we already got that if inner loop upper bound say here  $k \leq n$  (outer loop's upper bound) then

we have  $n$  times 1 to  $k$  giving run:

Then it will be :  $O(c \times n \times n) = O(cn^2) = O(n^2)$ .

If  $k > n$  then it will be  $O(c \times k \times n) = O(kn)$

But if  $k > n$ ; then programming perspective it must throw out of bound exception, hence not possible to compile, Though complexity will remain  $O(kn)$ .

\*\*\*\*\*