GUIDELINES FOR ASYMPTOTIC ANALYSIS- PART 1

[BASED ON SOME SUB-CODES OF PROGRAMS]

TILL NOW WE HAVE LEARNT THAT:

$\underline{\boldsymbol{Big - Oh\ or\ Worst\ Case\ Complexity}}$:

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

$\underline{\boldsymbol{Big - Omega\ or\ Best\ Case\ Complexity}}$:

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

$\underline{\boldsymbol{Big - Theta\ or\ Average\ Case\ Complexity}}$:

$$c_1 g(n) \leq f(n) \leq c_2 g(n), where\ c\ , n_1\ and\ n_2\ are\ constants\ and\ n \geq \{n_{1\,;} n_2\},$$

$$then\ f(n) = \Omega\big(g(n)\big)$$

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.

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

$$Total\ time = contant\ time\ c \times n = O(n).$$

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

1ST TYPE

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

So, 1st 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:

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

Findings:

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

Again, we can say inner loop runs $\boldsymbol{n\ times\ 1\ to\ n =}$ $\boldsymbol{1 + 2 + 3 + 4 + \cdots + n - 1 + n\ times}$

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

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

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

By addition rule:

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

Hence inner loop dominates over outer loop i.e.

By addition rule(Outer Loop + Inner Loop) :

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

2ND TYPE

$//\boldsymbol{outer\ loop\ executed\ \ n\ times}$

$\boldsymbol{for}(\boldsymbol{i} = \mathbf{1}; \boldsymbol{i} \leq \boldsymbol{n}; \boldsymbol{i} + +)\{$

$//\boldsymbol{inner\ loop\ executes\ n\ times}$

$\boldsymbol{for}(\boldsymbol{j} = \mathbf{1}; \boldsymbol{j} \leq \boldsymbol{i}; \boldsymbol{j} + +)\{$

$\boldsymbol{k} = \boldsymbol{k} + \mathbf{1}\ ;\ //\ \boldsymbol{constant\ time}.$

$\}$

So, 1st For loop will be executed $\boldsymbol{n\ times}$.

$\boldsymbol{Or, 2nd\ For\ Loop\ will\ be\ executed\ as\ depending\ upon\ the\ first\ for\ loop\ \ as\ shown\ below}$:

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:

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

Findings:

We can say $\boldsymbol{outer\ loop\ executes\ 1\ to\ n\ times} =$ $\mathbf{O}(\boldsymbol{n})\boldsymbol{for\ outer\ loop}.$

Again, we can say inner loop runs $\boldsymbol{n\ times\ 1\ to\ i} =$ $\mathbf{1+2+3+4+\cdots+}\boldsymbol{n}\mathbf{-1+}\boldsymbol{n\ times}$

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

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

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

By addition rule:

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

Hence inner loop dominates over outer loop i.e.

By addition rule(Outer Loop + Inner Loop) :

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

3RD TYPE

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

So, 1st 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/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:

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

Findings:

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

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

Hence total number of iterations that inner look will run:

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

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

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

By addition rule:

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

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

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

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

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[RTH] TYPE

$//outer\ loop\ executed\ \ n\ times$

$for(i = 1; i \leq n; i++)\{$

$\quad //inner\ loop\ executes\ n\ times$

$\ for(j = 1; j \leq n - 1; j++)\{$

$\qquad k = k + 1\ ;\ //\ constant\ time.$

$\}$

So, 1st 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-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:

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

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:

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

$By\ arithmetic\ series$:

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

$Where\ , a = First\ Term.$
$d = (\,T_{n+1} - T_n)$
$T_{n+1} = Second\ Last\ term\ \Rightarrow n-1.$
$T_n = Last\ Term \Rightarrow n.$
$n-1 = Second\ last\ term\ i.e. T_{n+1}.$

$We\ can\ rewrite\ the\ formula\ as$:

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

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

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

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

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

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

Therefore:

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

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

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

By addition rule:

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

Hence inner loop dominates over outer loop i.e.

By addition rule (Outer Loop + Inner Loop):

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

5TH 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, 1st For loop will be executed $\boldsymbol{n\ times}$.

$\boldsymbol{Or, 2nd\ For\ Loop\ will\ be\ executed\ as\ depending\ upon\ the\ first\ for\ loop\ \ as\ shown\ below}$:

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:

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

Findings:

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

Again, we can say inner loop runs $\boldsymbol{n\ times\ n - k}$.
Hence total number of iterations that inner look will run:

$$\boldsymbol{\Rightarrow (n - k + n - k - 1 + n - k - 2 + \cdots + 3 + 2 + 1)\ times}$$

$\boldsymbol{By\ arithmetic\ series}$:

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

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

$$d = (\,T_{n+1} - T_n)$$
$$T_{n+1} = Second\ Last\ term$$
$$T_n = Last\ Term$$

$We\ can\ rewrite\ the\ formula\ as$:

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

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

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

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

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

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

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

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

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

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

$$\approx \mathbf{O}(n) + O(n^2) + O(k)[\ Here\ k\ is\ constant]$$
$$\approx \mathbf{O}(n) + O(n^2) + O(1)$$
$$\approx \mathbf{O}(n) + \mathbf{O}(n^2) + O(1) = \mathbf{O}\{max(n^2 + n + 1)\} = \mathbf{O}(n^2)$$

Hence inner loop dominates over outer loop i.e.

By addition rule (Outer Loop + Inner Loop):

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

6TH 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, 1st 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/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:

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

Findings:

We can say $\boldsymbol{outer\ loop\ executes\ 1\ to\ n\ times} =$ $\mathbf{O}(\boldsymbol{n})\boldsymbol{for\ outer\ loop}$.

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

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

$\boldsymbol{By\ arithmetic\ series}$:

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

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

$$\boldsymbol{d = (T_{n+1} - T_n) = (\tfrac{n}{2} - 1) - \tfrac{n}{2} = \tfrac{n-2}{2} - \tfrac{n}{2} = \tfrac{-2}{2} = -1}$$

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

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

$We\ can\ rewrite\ the\ formula\ as$:

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

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

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

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

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

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):

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

7TH 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 $O(kn)$.

****