GUIDELINES FOR ASYMPTOTIC ANALYSIS- PART 1A

[BASED ON SOME SUB-CODES OF PROGRAMS]

TILL NOW WE HAVE LEARNT THAT:

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

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

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

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

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

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

$$\boldsymbol{then\ f(n) = \Theta\big(g(n)\big)}$$

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.

$$
\begin{array}{l}
//\boldsymbol{executes\ n\ times} \\
\boldsymbol{for}(\boldsymbol{i} = \mathbf{1}; \boldsymbol{i} \leq \boldsymbol{n}; \boldsymbol{i}++)\{ \\
\quad \boldsymbol{m} = \boldsymbol{m} + 2; //\boldsymbol{constant\ time, c} \\
\}
\end{array}
$$

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

Evaluation:

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

Again, we can say inner loop runs $\boldsymbol{or\ iterates\ \ n\ times}\ \mathbf{1}\ \boldsymbol{to\ n} =$

That is:

$\boldsymbol{i} = \mathbf{1}\ , \boldsymbol{i\ terates}\ \mathbf{1}\ \boldsymbol{time}\ , \boldsymbol{i} = \mathbf{2}\ , \boldsymbol{i\ terates}\ \mathbf{1}\ \boldsymbol{time}$
$\boldsymbol{i} = \mathbf{3}\ , \boldsymbol{i\ terates}\ \mathbf{1}\ \boldsymbol{time}\ , \boldsymbol{i} = \mathbf{4}\ , \boldsymbol{i\ terates}\ \mathbf{1}\ \boldsymbol{time}$
Similarly, up to n time $\boldsymbol{i} = \boldsymbol{n}\ , \boldsymbol{i\ terates}\ \mathbf{1}\ \boldsymbol{time}.$

$\boldsymbol{And\ we\ know\ by\ counting\ technique\ that}:$

$\boldsymbol{When}\ \mathbf{1}\ \boldsymbol{counted\ to\ n\ times\ gives}\ \mathbf{1} \times \boldsymbol{n} = \boldsymbol{n}\ .$

$\boldsymbol{Hence}\ \boldsymbol{O}(\boldsymbol{n}).$

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

$$
\begin{array}{l}
//\boldsymbol{outer\ loop\ executed\ \ n\ times} \\
\boldsymbol{for}(\boldsymbol{i}=\mathbf{1};\boldsymbol{i}\le\boldsymbol{n};\boldsymbol{i}++)\{ \\
\quad //\boldsymbol{inner\ loop\ executes\ n\ times} \\
\quad \boldsymbol{for}(\boldsymbol{j}=\mathbf{1};\boldsymbol{j}\le\boldsymbol{n};\boldsymbol{j}++)\{ \\
\qquad \boldsymbol{k}=\boldsymbol{k}+\mathbf{1}\ ;//\ \boldsymbol{constant\ time}. \\
\}
\end{array}
$$

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	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:

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

Evaluation:

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

Again, we can say inner loop
runs $or\ iterates\ \ n\ times\ 1\ to\ n =$

$i = 1\ , k = \{1, 2, 3..n\}\ i.e\ k\ iterates\ 1\ time\ 1\ to\ n\ .$
$i = 2\ , k = \{\{1, 2, 3..n\}, \{1, 2, 3, .., n\}\}$
$i.e\ k\ iterates\ \ 2\ times\ 1\ to\ n\ .$
.....
$i = n\ ,$
$k = \{\{1, 2, 3..n\}, \{1, 2, 3, \dots, n\} \dots., \{1, 2, 3, \dots, n\}\}$
$i.e\ k\ iterates\ n\ times\ 1\ to\ n.$

$We\ have\ to\ see\ the\ number\ of\ times\ to\ calculate$

$time\ complexity.$

$1 + 2 + 3 + 4 + \cdots + n - 1 + n\ times$

$By\ arithmetic\ series(Arithmetic\ Progression$

$to\ find\ general\ term):$

$$\Longrightarrow S(n) = \frac{n}{2}\left((2 \times a) + (\ (n - 1) \times (d)\ \right)$$

$Where\ , a = First\ Term.$

$d = (\ T_n - T_{n-1})\ or\ it\ can\ be\ 2^{nd}\ \ term -$
$(minus) 1^{st}\ term.$

$i.e. the\ common\ difference.$

$T_{n-1} = Second\ Last\ term\ \Rightarrow n-1.$
$T_n = Last\ Term \Rightarrow n.$
$n-1 = Second\ last\ term\ i.e. T_{n-1}.$

$$\Rightarrow S(n) = \frac{n}{2}\big((2 \times 1) + (\,(n-1) \times (1)\,)\big)$$

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

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

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

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

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

Hence inner loop iterates dominates over outer loop i.e.

By addition rule (Outer Loop + Inner Loop):

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

Also, $how\ much\ inner\ loop\ will\ be\ iterated$
$,it\ is\ depended\ over\ outer\ loop.$