GUIDELINES FOR ASYMPTOTIC ANALYSIS- PART 1A

[BASED ON SOME SUB-CODES OF PROGRAMS]

TILL NOW WE HAVE LEARNT THAT:

Big − Oh or Worst Case Complexity:

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

Big − Omega or Best Case Complexity:

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

Big − Theta or Average Case Complexity:

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

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

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.

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

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

Evaluation:

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

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

That is:

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

$And\ we\ know\ by\ counting\ technique\ that :$

$When\ 1\ counted\ to\ n\ times\ gives\ 1 \times n = n$.

$Hence\ O(n).$

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

```
//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, 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:

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

Evaluation:

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

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

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

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

By arithmetic series(Arithmetic Progression
to find general term):
$$\Rightarrow S(n) = \frac{n}{2}\big((2 \times a) + ((n-1) \times (d))\big)$$

$Where\ , a = First\ Term.$

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

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

$T_n = Last\ Term \Rightarrow n.$

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

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

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

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

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

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

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

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

By addition rule (Outer Loop + Inner Loop):

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

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