14. ASYMPTOTIC RULES

Asymptotic rules are useful for manipulating asymptotic notations and extracting new information about the behaviour of the functions.

1. **REFLEXIVITY RULE:** For any general complexity function $\boldsymbol{f(n)}$, $\boldsymbol{the}$ reflexive property is given as follows:

$$f(n) = O(g(n))$$
$$f(n) = \Omega(g(n))$$
$$f(n) = \Theta(g(n))$$

2. **TRANSITIVITY RULE:** The transitive property is defined as follows:

$$\boldsymbol{if\ f(n) = O(g(n))\ and\ g(n) = O(h(n)),}$$
$$\boldsymbol{then\ f(n) = O(h(n))}$$

$\boldsymbol{Where\ f(n), h(n)\ and\ g(n)\ are\ complexity\ functions\ of\ two\ algorithms.}$

$\boldsymbol{This\ property\ holds\ good\ for\ other\ notations\ also}$:

$2. \boldsymbol{f(n) = \Omega(g(n))\ and\ g(n) = \Omega(h(n))}$

$\boldsymbol{then, f(n) = \Omega(h(n))}$.

$3. f(n) = \Theta(g(n))$ *and* $g(n) = \Theta(h(n))$

then, $f(n) = \Theta(h(n))$.

3. LAW OF COMPOSITION: By law of composition, we mean that:

1. $O(O(g(n)) = O(g(n))$
2. $\Omega(\Omega(g(n)) = \Theta(g(n))$
3. $\Theta(\Theta(g(n)) = \Theta(g(n))$

Hence, we can say:

$$O\left(O\left(\dots\dots\left(O(g(n))\right)\dots\right)\right) = O(g(n))$$

Therefore, we can say that for other notations also=

$$\Omega\left(\Omega\left(\dots\dots\left(\Omega(g(n))\right)\dots\right)\right) = \Omega(g(n))$$

$$\Theta\left(\Theta\left(\dots\dots\left(\Theta(g(n))\right)\dots\right)\right) = \Theta(g(n))$$

4. SUMMATION RULE:

Assume that an algorithm A is written in such a way that some portions of it have complexity n, some portions have complexity have $logn$, and complexity n^2 :

Algorithm segment with complexity n

Algorithm segment with complexity $logn$.

Algorithm segment with complexity n^2.

Then what is the overall complexity of the algorithm A? The answer is provided by the summation rule.

The final complexity of the preceding segment is $n + logn + n^2$.

This is one of the most important rules. The law of addition states the following:

$$f(n) + g(n) = O(\max\{f(n), g(n)\})$$

$$f(n) + g(n) = \Omega(\max\{f(n), g(n)\})$$

$$f(n) + g(n) = \Theta(\max\{f(n), g(n)\})$$

As per this rule, the final complexity of this algorithm segment is:

$$\max\{n, \log(n), n^2\} = n^2.$$

The proof of this rule is given as follows:

$$Let\ f_1(n) = O(g_1(n))\ and\ f_2(n) = O(g_2(n))$$

$if\ so,$

$f_1(n) \le c_1 g_1(n)\ for\ n \ge n_1\ and\ f_2(n) \le c_2 g_2(n) for\ n \ge n_2$

Choose a number k bigger than c_1, c_2

Choose a number n bigger than $n_1, n_2 \Rightarrow n_0 = \max\{n_1, n_2\}$

This implies:

$\Rightarrow f_1(n) + f_2(n) \le c_1 g_1(n) + c_2 g_2(n)$

$\Rightarrow f_1(n) + f_2(n) \le k g_1(n) + k g_2(n) [k \ge \{c_1, c_2\}]$

$\Rightarrow f_1(n) + f_2(n) \le k g(n) + k g(n) [n \ge \{n_1, n_2\}, hence$

$$g(n) \ge \{g_1(n), g_2(n)\}]$$

$$\Rightarrow f_1(n) + f_2(n) \le 2kg(n)$$

Thus, $f_1(n) + f_2(n) = \mathrm{O}\big(\mathrm{g}(\mathrm{n})\big)$

And $g(n) \ge \{g_1(n), g_2(n)\}$

Therefore, $f_1(n) + f_2(n) = \mathrm{O}(\max\{\mathrm{g}_1(n), \mathrm{g}_2(n)\}$

Again,

As $f_1(n) = \mathrm{O}\big(g_1(n)\big)$ *and* $f_2(\mathrm{n}) = \mathrm{O}\big(\mathrm{g}_2(n)\big)$

Therefore, $f_1(n) + f_2(n) = \mathrm{O}(\max\{\mathrm{f}_1(n), \mathrm{f}_2(n)\}$

4. MULTIPLICATION RULE:

Consider the following segment:

$for\ i = 1\ to\ n\ do$ ← Executed n times.

$\quad perform\ operation\ \mathrm{O}(1)$ ← loop body getting executed n times.

$End\ for$

It is evident for that the loop is executed $\sum_{i=0}^{n} O(1) = \mathrm{O}(n)$ times.

i.e., $1 \times 2 \times 3 \times 4 \times \ldots \ldots \ldots . . \times n - 1 \times n$ times.

Now if there are two loops:

Then the inner loop would be executed $n \times n$ times.

This is called the multiplication rule.

In general, the multiplication of complexity of two functions equals the product of two complexity functions.

$$Let\ f_1(n) = O(g_1(n))\ and\ f_2(n) = O(g_2(n))$$

$$if\ so,$$

$$f_1(n) \leq c_1 g_1(n)\ for\ n \geq n_1\ and\ f_2(n) \leq c_2 g_2(n)\ for\ n \geq n_2$$

$$Let\ k\ be\ a\ number\ greater\ than\ c_1 \times c_2\ and\ n_0 = max\{n_1, n_2\}$$

$$\Rightarrow f_1(n) \times f_2(n) \leq c_1 g_1(n) \times c_2 g_2(n)$$

$$\Rightarrow f_1(n) \times f_2(n) \leq k g_1(n) \times k g_2(n) [k \geq \{c_1, c_2\}]$$

$$\Rightarrow f_1(n) \times f_2(n) = k^2 \left(g_1(n) \times g_2(n)\right), where\ n \geq n_0$$

Therefore, resultant is : $O\left(g_1(n) \times g_2(n)\right)$

Thus, we can write:

1. $f_1(n) \times f_2(n) = O\left(g_1(n) \times g_2(n)\right)$
2. $f_1(n) \times f_2(n) = \Omega\left(g_1(n) \times g_2(n)\right)$
3. $f_1(n) \times f_2(n) = \Theta\left(g_1(n) \times g_2(n)\right)$

That is:

$$\boldsymbol{for\ i = 1\ to\ n\ do} \xleftarrow{\boldsymbol{run\ n\ times}}$$

$$\quad\quad \boldsymbol{for\ i = 1\ to\ n\ do} \xleftarrow{\boldsymbol{run\ n\ times}}$$

Statement

$\boldsymbol{End\ for}$

Therefore, we have $\boldsymbol{time\ complexity : O(n^2)}$

4. TRANSPOSE SYMMETRY:

1. $\boldsymbol{f(n) = O\big(g(n)\big)\ if\ and\ only\ if\ g(n) = \Omega\big(f(n)\big)}$.
2. $\boldsymbol{f(n) = \Omega\big(g(n)\big)\ if\ and\ only\ if\ g(n) = O\big(f(n)\big)}$.
3. $\boldsymbol{f(n) = \Theta\big(g(n)\big)\ if\ and\ only\ if\ g(n) = \Theta\big(f(n)\big)}$.

6. CONSTANT RULE:

1. If $\boldsymbol{f(n) is\ in\ \mathrm{O}\left(c \times g(n)\right)\ for\ any\ c > 0, then}$

$$\boldsymbol{f(n) is\ in\ \mathrm{O}(n).}$$

2. If $\boldsymbol{f(n) is\ in\ \Omega\left(c \times g(n)\right)\ for\ any\ c > 0, then}$

$$\boldsymbol{f(n) is\ in\ \Omega(n).}$$

3. If $\boldsymbol{f(n) is\ in\ \Theta\left(c \times g(n)\right)\ for\ any\ c > 0, then}$

$$\boldsymbol{f(n) is\ in\ \Theta(n).}$$

****