20.3. TIME COMPLEXITY CALCULATION

FOR LOOP (EG-2).

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

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

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

$\quad for(j = 1; j \leq i/2; j++)\{$

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

$\}$

SOLUTION:

$1. Inner\ most\ loop's\ statement \Longrightarrow c = c + 1\ which\ runs\ at$

$O(1)\ time\ i.e.1\ unit\ of\ time\ \ .$

$2. No.of\ iterations\ \ in\ outer\ for\ loop\ takes\ 1\ to\ n\ times.$

$lets\ see\ the\ inner\ loop\ and\ runtime\ of$

$inner\ loop's\ statement.$

$f(1) \leq c \times n \Longrightarrow when\ i = 1$

$f(1) \leq c \times \frac{1}{2} \times i \Longrightarrow when\ j = 1$

$c = c + 1\ executes\ in\ \frac{1}{2} \times 1\ unit\ of\ time\ .$

$[Hence\ , total\ amount\ of\ taken\ to\ run\ (c = c + 1\ )$

$is\ 1\ unit\ of\ time]$

$f(2) \leq c \times n \Longrightarrow when\ i = 2$

$f(1) \leq c \times \frac{1}{2} \times i \Longrightarrow when\ j = 1$

$c = c + 1\ executes\ in\ \frac{1}{2} \times 1\ unit\ of\ time\ .$

$f(2) \leq c \times \frac{1}{2} \times i == \Longrightarrow when\ j = 2$

$c = c + 1\ executes\ in\ \frac{1}{2} \times 1\ unit\ of\ time$

$[Hence\ , total\ amount\ of\ taken\ to\ run\ (c = c + 1\ )$

$is \frac{1}{2} + \frac{1}{2} = \frac{2}{2}\ unit\ of\ time]$

....

$f(n) \leq c \times n \Longrightarrow when\ i = n$

$f(1) \leq c \times \frac{1}{2} \times i \Longrightarrow when\ j = 1$

$c = c + 1\ executes\ in\ \frac{1}{2} \times 1\ unit\ of\ time\ .$

$f(2) \leq c \times \frac{1}{2} \times i \Longrightarrow when\ j = 2$

$c = c + 1\ runs \frac{1}{2} \times 1\ \ unit\ of\ time.$

....

$f(n) \leq c \times \frac{1}{2} \times i \Longrightarrow when\ j = n$

$c = c + 1\ runs \frac{1}{2} \times 1\ unit\ of\ time.$

$[Hence\ ,total\ amount\ of\ taken\ to\ run\ (c = c + 1\ )$

$is\frac{1}{2} + \frac{1}{2} + \cdots + \frac{1}{2}\ to\ n\ times = \frac{n}{2}\ unit\ of\ time]$

$No.of\ units\ of\ time\ taken\ to\ run\ the\ inner\ most\ statement$

$$\frac{1}{2} + \frac{2}{2} + \frac{3}{2} + \cdots + \frac{n}{2} = \frac{1}{2}(1 + 2 + 3 + \cdots + n) =$$

$By\ arithmetic\ series(Arithmetic\ Progression$

$to\ find\ general\ term)$:

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

$Where\ ,a = First\ Term.$

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

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

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

$$\Rightarrow S(n) = \frac{1}{2}(1 + 2 + 3 + \cdots + (n-1) + n)$$

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

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

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

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

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

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

$$\textit{Therefore time complexity of the program is}:$$

$$= O(n^2)$$

SOME OBSERVATION:

$c = c + 1$ *inner most statement will execute depending upon the upper bound of inner most loop* j *i.e.* $\frac{i}{2}$ *and we can too add up upper bound* $g(n)$ *of inner most loop as i increment at each iteration i.e.* $\frac{1}{2} + \frac{2}{2} + \frac{3}{2} + \cdots + \frac{i}{2}$ *as we are looking for upper bound and i will execute till `n` time*

hence at $i = n$ *, we have* : $\frac{1}{2} + \frac{2}{2} + \frac{3}{2} + \cdots + \frac{n}{2}$. *This is also correct.*