20.2. TIME COMPLEXITY CALCULATION FOR LOOP (EG-1).

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

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

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

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

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

$\quad \}$

$\}$

SOLUTION:

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

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

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

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

$inner\ loop's\ statement.$

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

$f(1) \leq c \times i \Rightarrow when\ j = 1$

$c = c + 1\ runs\ 1\ unit\ of\ time.$

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

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

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

$f(1) \leq c \times i \Rightarrow when\ j = 1$

$c = c + 1\ runs\ 1\ unit\ of\ time.$

$f(2) \leq c \times i \Rightarrow when\ j = 2$

$c = c + 1\ runs\ 1\ unit\ of\ time.$

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

$is\ (1 + 1 = 2\ )unit\ of\ time]$

$f(3) \leq c \times n \Rightarrow when\ i = 3$

$f(1) \leq c \times i \Rightarrow when\ j = 1$

$c = c + 1\ runs\ 1\ unit\ of\ time.$

$f(2) \leq c \times i \Rightarrow when\ j = 2$

$c = c + 1\ runs\ 1\ unit\ of\ time.$

$f(3) \leq c \times i \Rightarrow when\ j = 2$

$c = c + 1\ runs\ 1\ unit\ of\ time.$

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

$is\ (1 + 1 + 1 = 3\ )unit\ of\ time]$

....

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

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

$c = c + 1\ runs\ 1\ unit\ of\ time.$

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

$c = c + 1\ runs\ 1\ unit\ of\ time.$

$f(3) \leq c \times i \Longrightarrow when\ j = 2$

$c = c + 1\ runs\ 1\ unit\ of\ time.$

......

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

$c = c + 1\ runs\ 1\ unit\ of\ time.$

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

$is\ (1 + 1 + 1 + \cdots n\ times = n\ )unit\ of\ time]$

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

$time\ complexity.$

$$1 + 2 + 3 + 4 + \cdots + n - 1 + n\ times = \sum_{i=1}^{n} i\ \ (Arithmetic\ Series)$$

VERY IMPORTANT

****

$By\ arithmetic\ series(Arithmetic\ Progression$

$to\ find\ \ Sum\ of\ first\ 'n'\ \ term):$

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

$Where , a = First\ Term.$

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

$T_{n-1} = Second\ Last\ term \Longrightarrow n-1.$
$T_n = Last\ Term \Longrightarrow n.$

PROOF OF ABOVE EQUATION

if \` $l$ \` which is last term i.e. $l = a + (n-1)d$ and the equation actually is:
$Sn = a + (a+d) + (a+2d) + (a+3d) + \cdots + (l-2d) + (l-d) + l$

Rewriting the series in reverse additive order:

$Sn = l + (l-d) + (l-2d) + \cdots + (a+2d) + (a+d) + a$

Adding columnwise we get:

$2Sn = (a+l) + (a+l) + (a+l) + \cdots n\ times = n(a+1)$

$$\therefore S_n = \frac{n}{2}[a+l]$$

$$\therefore S_n = \frac{n}{2}[a+a+(n-1)d]$$

$$\therefore S_n = \frac{n}{2}[2a+(n-1)d]$$

AND HOW WE ARE GETTING $T_n = a + (n-1)d$ or $l = a + (n-1)d.$

ARITHMETIC PROGRESSION: A sequence (finitite or infinite) is called an arithmetic progession abbreviated as A.P iff the difference of any term from its preceeding term is finite.

Say: we have $1 + 2 + 3 + \cdots n$, then $T_{n+1} - T_n = 1,$
where T_{n+1} is preceeding term and T_n is term substracted i.e. $2 - 1 = 1,$
$3 - 2 = 1 \ldots\ldots etc.$

if there is a sequence of $a_1, a_2, a_3, \ldots, a_n$ then $a_1 + a_2 + a_3 + \cdots + a_n$ is called an

$Arithmetic\ Series.$

GENERAL TERM OF A.P:

$Let\ a\ be\ the\ first\ term\ and\ d\ be\ the\ common\ difference\ of\ Arithmetic\ Progression(A.P).Let\ T_1, T_2, T_3, \dots, T_{n-1}, T_n = 1st, 2nd, 3rd, \dots, nth\ terms\ respectively, then\ we\ have:$

$T_2 - T_1 = d$
$T_3 - T_2 = d$
$T_4 - T_3 = d$
.....
$T_n - T_{n-1} = d$

ADDING THESE $n-1\ equations\ we\ get:$
$T_n - T_1 = (n-1)d \implies T_n = T_1 + (n-1)d, where\ T_1 = a, then$
$T_n = a + (n-1)d.$

LAST TERM OF A.P
$if\ T_n = l\ (last\ term), then\ l = a + (n-1)d.$

$Series: if\ the\ terms\ are\ connected\ by\ +(signs) we\ get\ a\ series.$
$T_1 + T_2 + \cdots + T_n\ is\ called\ a\ series.$

****

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

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

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

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

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

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

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

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

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