Factorial Time complexity using Substitution Method

Now, there are two ways to do :

<u>*1st Way*</u>

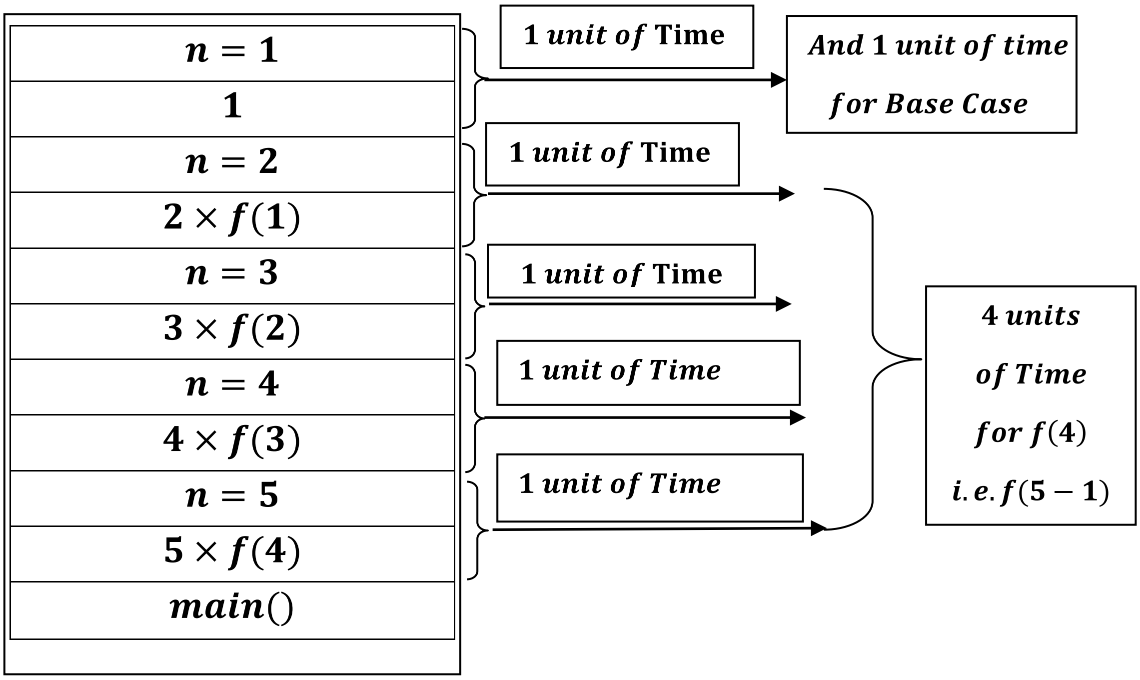
<u>*Base Cases*:</u>

$T(0) = 1$, *when* $n = 0$ *i.e. when* $n = 0$, *it returns* 1 *and it takes* 1 *unit of time* .

$T(1) = 1$, *when* $n = 1$ *i.e. when* $n = 1$, *it returns* 1 *and it takes* 1 *unit of time* .

<u>*and now for* $n > 0$,</u>

$F(n-1) \times n \Longrightarrow$ *Just take the last push for* $F(5)$:



$F(n-1) \times n$ *runs each at constant i.e. constant unit of time takes* $T(n-1)$ *times and as it reaches* $T(1)$ *it executes an extra constant time i.e.* 1 *unit of time* .

Hence total complexity is: $T(n) = T(n-1) + T(1)$ *or*,
$T(n) = T(n-1) + 1$.

As each time it multiples it takes a constant amount of time complexity.

$$T(n) = \begin{cases} 1 & ,for\ (n = 0) \\ 1 & ,for\ (n = 1) \\ T(n-1) + T(1) & ,for\ (n > 1) \end{cases}$$

$\boldsymbol{2nd\ way}$:

$\boldsymbol{According\ to\ some\ authors}$:

$\boldsymbol{Here\ say\ base\ case\ is\ 0\ only}$,

```
int factorial(int n)
{

    if (n == 0)
    {
        return 1;
    }

    else
    {
        return n * factorial(n - 1);
    }
}
```

$\boldsymbol{Factorial\ of\ (0) = 1\ and\ Factorial\ of\ (1) = 1 \times 1 = 1}$.

$Then\ with\ that\ approach$:

$n = 0$
1
$n = 1$
$1 \times f(0)$
$n = 2$
$2 \times f(1)$
$n = 3$
$3 \times f(2)$
$n = 4$
$4 \times f(3)$
$n = 5$
$5 \times f(4)$
$main()$

$4\ units$

$of\ Time$

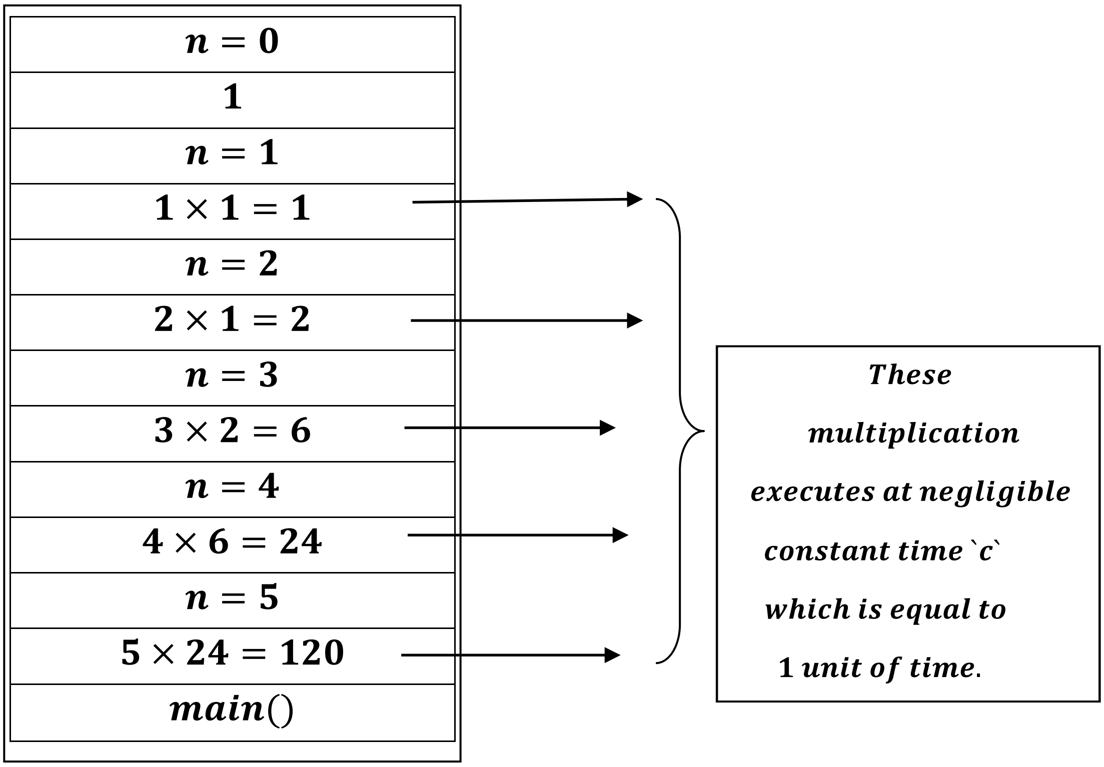
$for\ f(4)$

$i.e.\ f(5-1)$

$Hence\ F(n-1) takes\ T(n-1)\ times\ i.e.\ upto\ n > 0\ or,$

$upto\ F(1)\ i.e.\ F(4) \rightarrow F(3) \rightarrow F(2) \rightarrow F(1).$

$And,$



$$T(n) = \underbrace{T(n-1)}_{\boxed{\text{To Compute } F(n-1)}} + \underbrace{1}_{\boxed{\text{To multiply } F(n-1) \text{ by } n}} \qquad , for\ n > 0$$

***Which gives the following general recurrence equation*: −**

$$T(n) = \begin{cases} 1 & , for\ (n = 0) \\ T(n-1) + 1 & , for\ (n > 0) \end{cases}$$

$\boldsymbol{But\ we\ will\ move\ with\ the\ first\ way\ i.e.}$:

$$T(n) = \begin{cases} 1 & , for\ (n = 0) \\ 1 & , for\ (n = 1) \\ T(n-1) + T(1) & , for\ (n = 1) \end{cases}$$

$\boldsymbol{And\ we\ know\ T(1) = 1\ , hence\ rewriting\ the\ linear\ recurrence\ equation\ as}$:

$$T(n) = T(n-1) + 1$$

$$Therefore\ , T(n-1) = T(n-1-1) + 1 = T(n-2) + 1$$

$\boldsymbol{Substituting\ this\ in\ T(n)\ we\ get}$:

$$T(n) = (T(n-2) + 1) + 1$$

$$Now\ , T(n-2) = T(n-2-1) + 1 = T(n-3) + 1$$

$\boldsymbol{Substituting\ this\ in\ T(n)\ we\ get}$:

$$T(n) = \left((T(n-3)+1)\ +1\right)\ +1$$

$Now\ if\ it\ runs\ upto\ `i`\ times\ we\ get$:

$Now\ , T(n-i) = T(n-i-1)+1$

$Substituting\ this\ in\ T(n)\ we\ get$:

$$T(n) = T(n-i-1)+1+1+1+\cdots i\ times$$

$When\ i = n-1\ we\ get$:

$$T(n) = T(n-(n-1)-1)+1+1+1+\cdots(n-1)times$$

$$= T(n-n+1-1)+1+1+1\ \ldots.(n-1)times$$

$$= T(0)+1+1+1\ \ldots.\ \ (n-1)times$$

$And\ we\ know\ T(0) = 1\ along\ with\ T(1) is\ 1\ and\ as\ 1\ is$
$added\ to\ 1+1+1+\cdots(n-1)times\ \ it\ becomes$:

$$= 1+1+1+\cdots n\ times$$

$i.e.\ 1 \times n = n\ and\ time\ complexity\ is$:

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

****