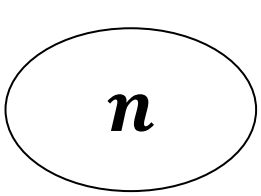
$C.1.$ *Recurrence Tree Method* $-$ *Example* 1

$$\textit{Example } 1: T(n) = 2T\left(\frac{n}{2}\right) + n$$

Solution:

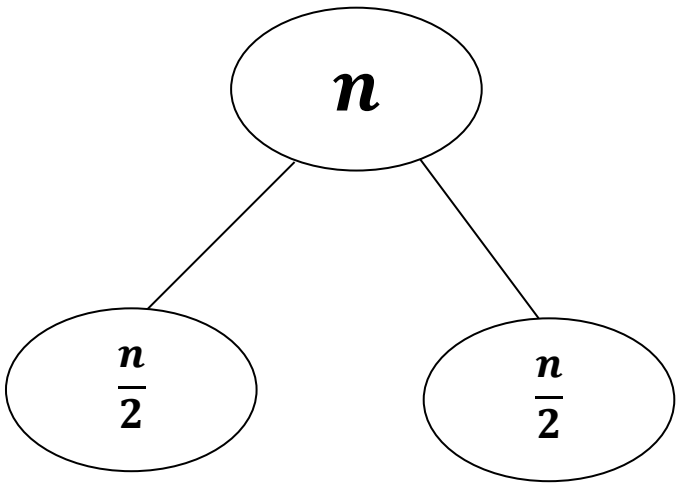
Size of the problem is: `$n$` . ***As it is*** $2T$ ***the division will be*** 2 ***of a single node of a recursive tree***.



Cost

Level 0 $\quad n$

And it divides into two sub problems, each of size $n/2$.

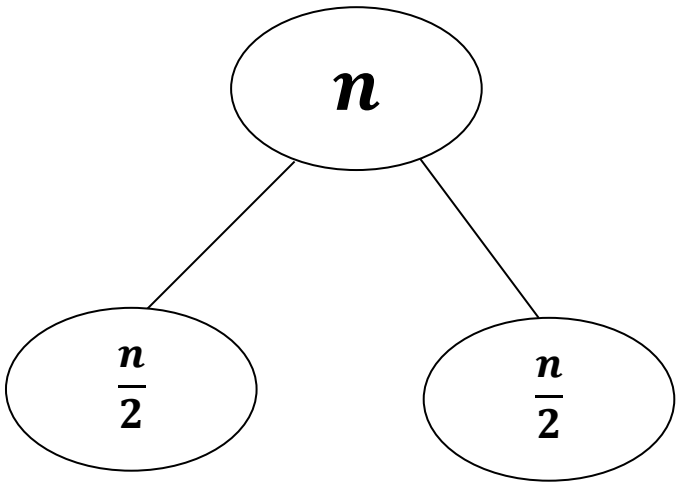


Cost

Level 0 $\quad n$

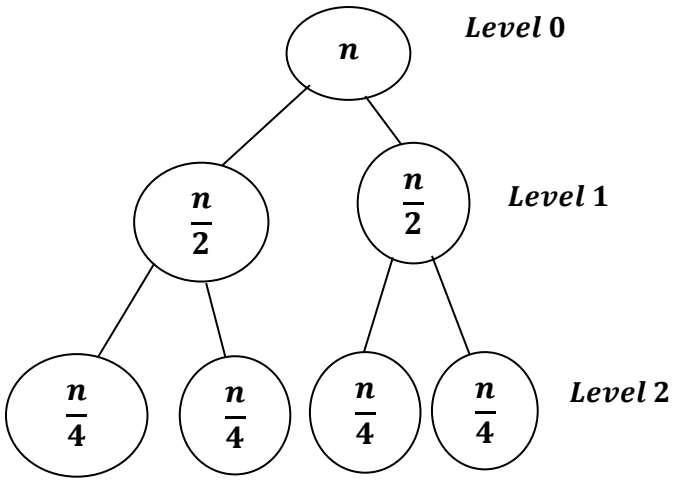
Level 1 $\quad \frac{n}{2} + \frac{n}{2} = n$

And it divides into two sub problems, each of size $n/2$.



	Cost
Level 0	n
Level 1	$\frac{n}{2}+\frac{n}{2}=n$

The next level of expansion, where original problem is divided into four sub problems, each with an input size of $\frac{n}{4}$.

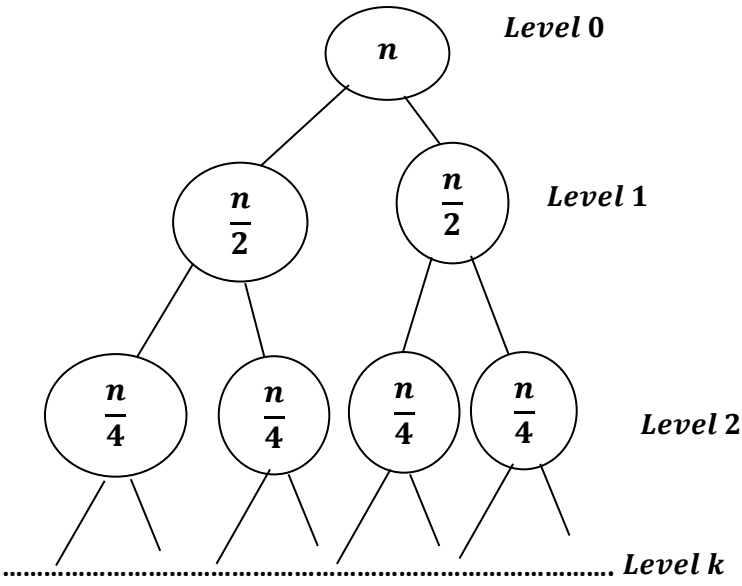


Cost

$\frac{n}{2} + \frac{n}{2} = n$

$\frac{n}{4} + \frac{n}{4} + \frac{n}{4} + \frac{n}{4} = n$

Now as it expands to Level k times:



Cost

n

$\frac{n}{2} + \frac{n}{2} = n$

$\frac{n}{4} + \frac{n}{4} + \frac{n}{4} + \frac{n}{4} = n$

$\frac{n}{2^k} + \frac{n}{2^k} + \frac{n}{2^k} + \cdots + \frac{n}{2^k} = n$

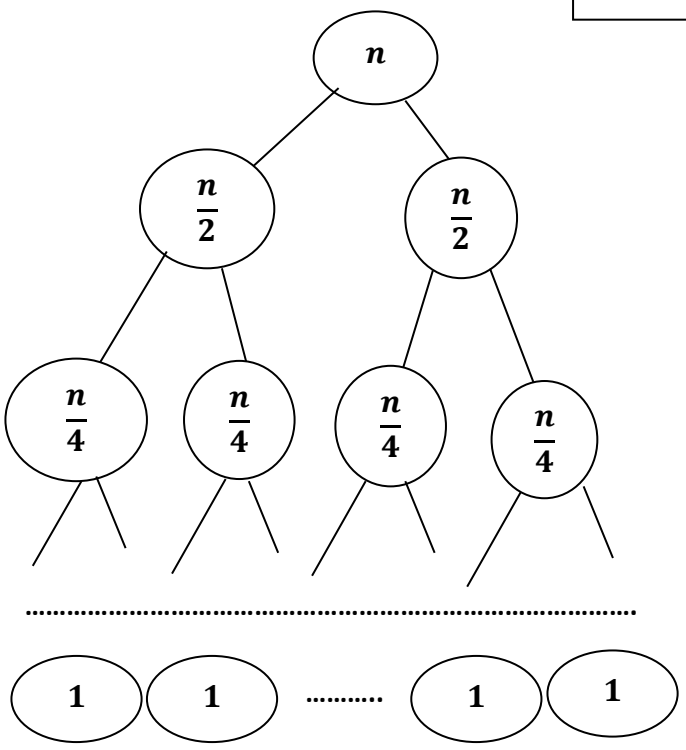
Now have we noticed that it gives us summation like:

$$\sum_{k=0}^{\log_2 n} \frac{n}{2^k}$$

$$= n + \frac{n}{2} + \frac{n}{4} + \cdots + \frac{n}{2^k}$$

$$= \frac{n}{2^0} + \frac{n}{2^1} + \frac{n}{2^2} + \cdots + \frac{n}{2^k}$$

$$= \frac{n}{2^{log_2 1}} + \frac{n}{2^{log_2 2}} + \frac{n}{2^{log_2 4}} + \cdots + \frac{n}{2^{log_2 n}}, where\ k = \log_2 n$$



LEVELS	No. of problems	Problem Size	Work Done
0	1	n	$1 \times n = n$
1	2	$\frac{n}{2}$	$2 \times \frac{n}{2} = n$
2	4	$\frac{n}{4}$	$4 \times \frac{n}{4} = n$
k	2^k	$\frac{n}{2^k}$	$2^k \times \frac{n}{2^k} = n$
$\log_2 n$	$2^{\log_2 n}$	1	$2^{\log_2 n} \times 1 = n$

$$(1 + \log_2 n) \times n$$

Now if we observe:

$$= \frac{n}{2^{log_2 1}} + \frac{n}{2^{log_2 2}} + \frac{n}{2^{log_2 4}} + \cdots + \frac{n}{2^{log_2 n}}$$

$$= \frac{n}{2^{log_2 1}} + \frac{n}{2^{log_2 2}} + \frac{n}{2^{log_2 4}} + \cdots + \frac{n}{n} \text{ , } \left(2^{\log_2 n} = n \text{ , as } a^{\log_a b} = b\right)$$

$$= \frac{n}{1} + \frac{n}{2} + \frac{n}{4} + \cdots + 2 + 1$$

$\left(\textit{We can also write } a^{\log_a b} = b^{\log_a a} = b\right)$

Thus , it can be concluded that the terms of a recurrence equation can be expanded as a tree .

At every level k, the amount of work done is computed.
In this case, the number of problems grows as a power of 2 and the size of the problem decreases in the following pattern:

$$= \ n + \frac{n}{2} + \frac{n}{4} + \cdots + 2 + 1 \text{ .}$$

One can observe that the amount of work done at every level is $: n$.

$$\textbf{can be calculated}$$

$$\textbf{as } 2^k \times \left(\frac{n}{2^k}\right) = n \textbf{ . Thus at the level of } \log_2 n \textbf{ , the problem}$$

$$\textbf{size would be reduced to } 1 \textbf{ , which is the base condition.}$$

$$\textbf{The work done by every leaf node}(\textbf{i.e., the node has no}$$

$$\textbf{children}) \textbf{ is } T(1) = 1 \textbf{ , which is a constant factor.}$$

$$\textbf{Therefore , the amount of work done at the last level}$$

$$\textbf{would be as follows:}$$

$$2^{\log_2 n} \times T(1) = n \times 1 = n.$$

$$\textbf{Alternatively,}$$

$$2^{\log_2 n} \times T(1) = n^{log_2 2} \times 1 = n$$

$$\textbf{Hence work done at levels is n. The total cost is the sum of}$$

$$\textbf{all level costs.}$$

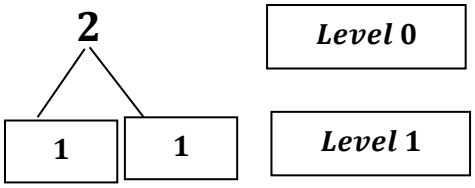
$$\textbf{As this is a complete tree and the level start from } 0,$$

$$\textbf{the number of levels would be :}$$

$$1 + \log_2 n \text{ .}$$

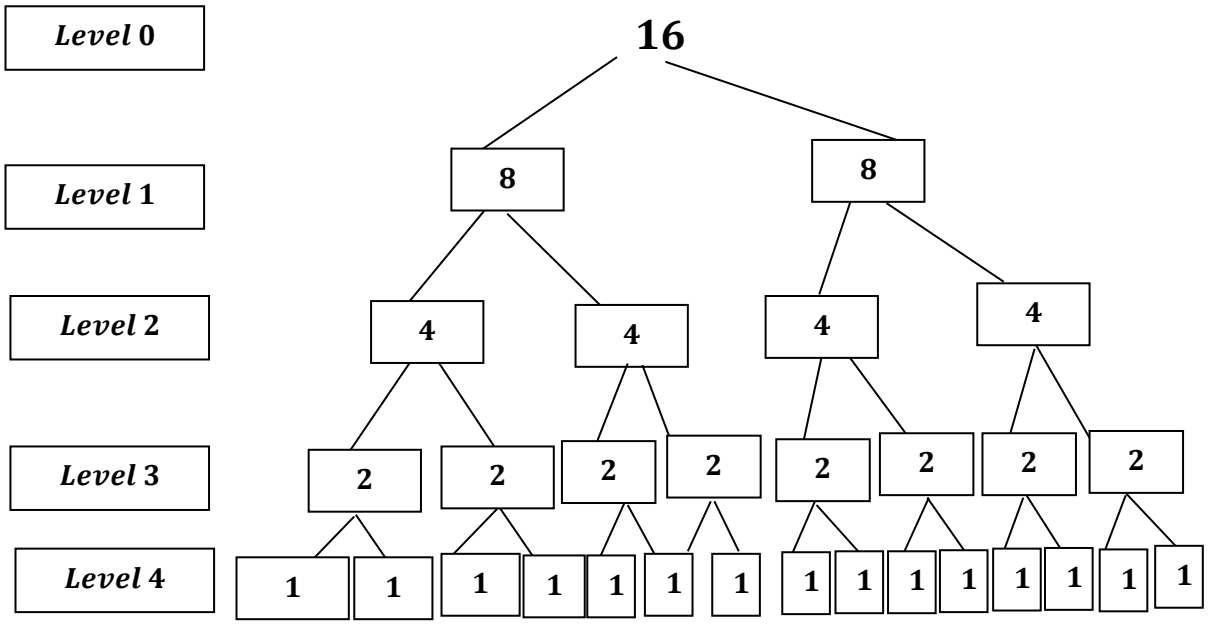
$\boldsymbol{How?}$

$\boldsymbol{lets\ take\ small\ node: n = 2\ with \frac{n}{2}\ division.}$



$\boldsymbol{No.\ of\ levels \Rightarrow \log_2 n + 1 = \log_2 2 + 1 = 1 + 1 = 2.}$

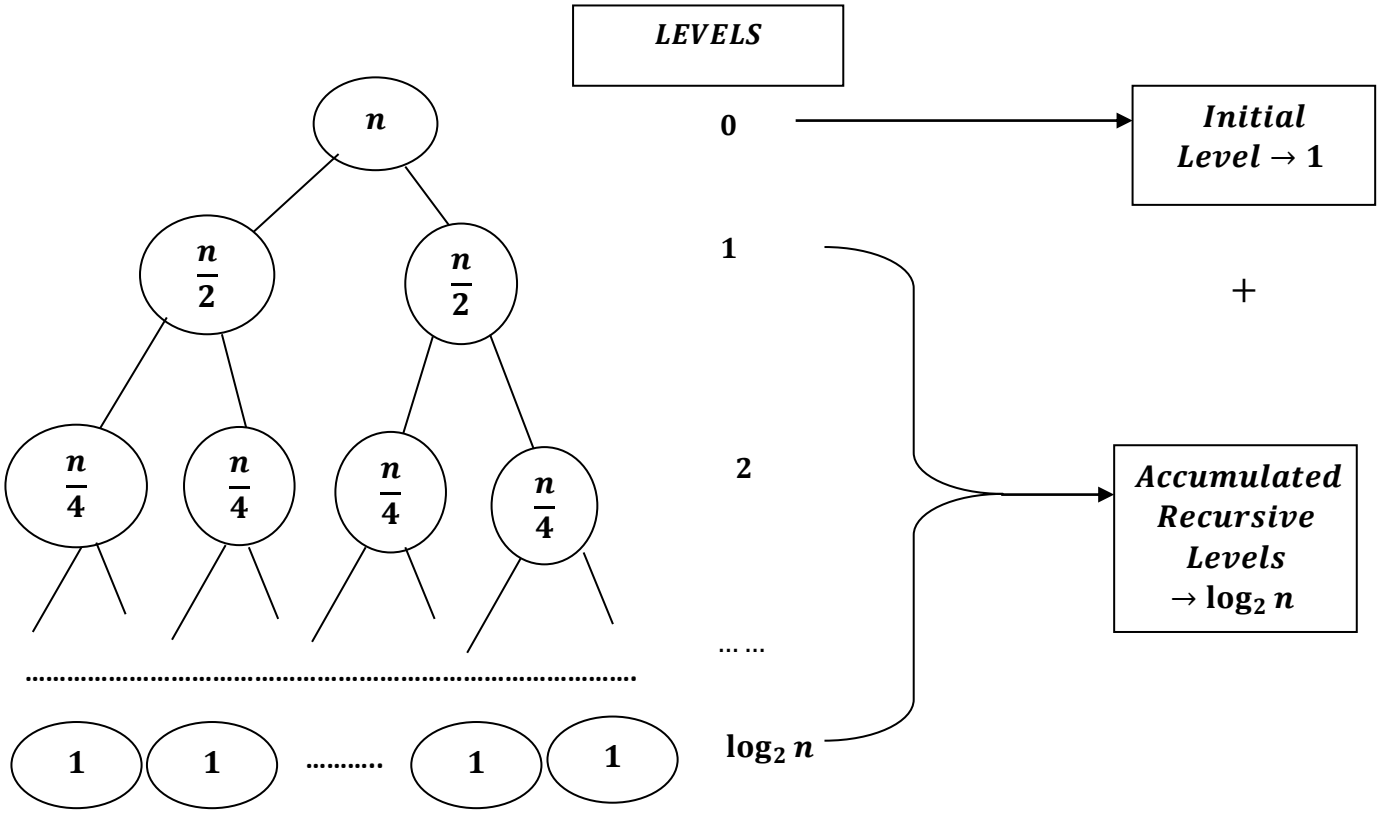
$\boldsymbol{Similarly,}$



$\boldsymbol{Hence, n = 16, Hence}$

$\boldsymbol{\text{No. of levels} = \log_2(16) + 1 = 4 + 1 = 5\ i.e.\ 0\ to\ 4.}$

$i.e.,$

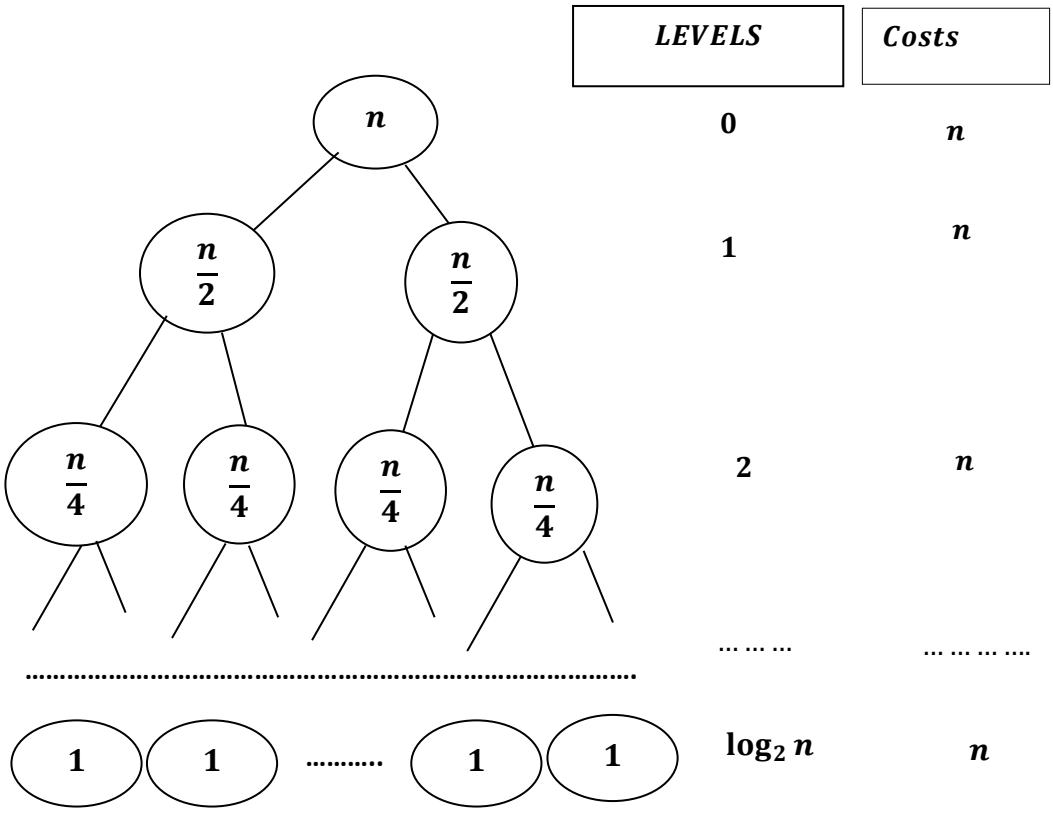


$Hence\ no.of\ levels = (1 + \log_2 n)\ for \frac{n}{2}\ divisions\ in\ a\ recursive\ tree.$

$i.e., \frac{n}{2}\ division\ of\ 16\ which\ is\ shown\ above : \log_2 16 = 4,$ $where\ 4\ is\ Accumulated\ recursive\ level.$

$And\ \log_2 16 + 1, here\ 1\ is\ the\ initial\ level\ i.e.\ level\ 0.$

Similarly, and now,



Therefore, the final cost is the level cost muliplied by the number of levels , which can be written as follows :

$$(\mathbf{1} + \log_2 n) \times n = n + n \log_2 n = \Theta(nlogn)$$

<u>Alternative approach</u>,

$$\textbf{Adding costs `n` upto } k = \log_2 n \textbf{ levels}$$

$$\Longrightarrow n + n + n + \cdots k \textbf{ times}.$$

$$= n + n + n + \cdots \log_2 n \textbf{ times } (\textbf{Where } k \textbf{ is } \log_2 n)$$

$$= n \times \log_2 n$$

$$= \Theta(n \log n)$$

Therefore , one can conclude that the asymptotic bound would be $n \times \log n \approx \Theta(n \log n)$.

This recursive tree demonstration : $T(n) = 2T\left(\frac{n}{2}\right) + n$ **is of Merge Sort algorithm.**

****