2nd Topic

Divide and Conquere (DnC)

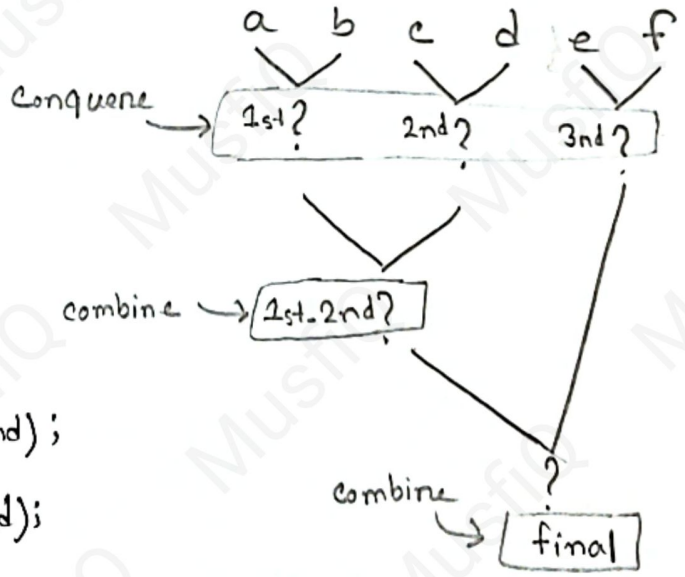
3 steps

- Divide
- Conquere (result)
- Combine (merge)

a, b, c, d, e, f এগুলার ভেতর যদি a max কিনা check করতে চাইলে,

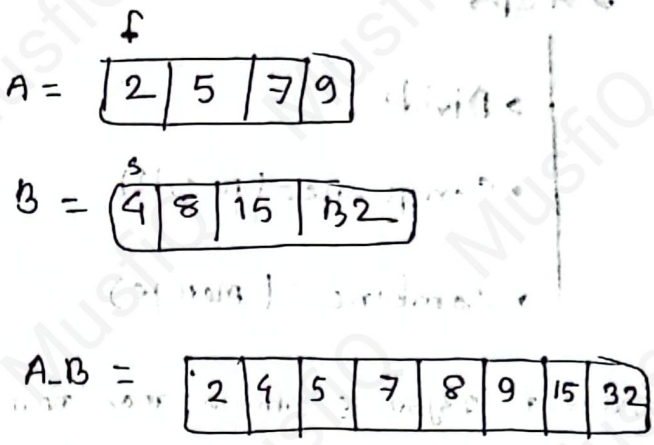
```
if(a>=b && a>=c && a>=d && a>=e && a>=f)
 {
  return a;
 }
else
  {

  }
```

কিন্তু এটা efficient way না




```
int max(a,b,c,d,e,f){
  int first = max(a,b);
  int second = max(c,d);
  int third = max(e,f);
  int first_second = max(first, second);
  int final = max(first_second, third);
  return final;
}
```

Menge sort s recap:

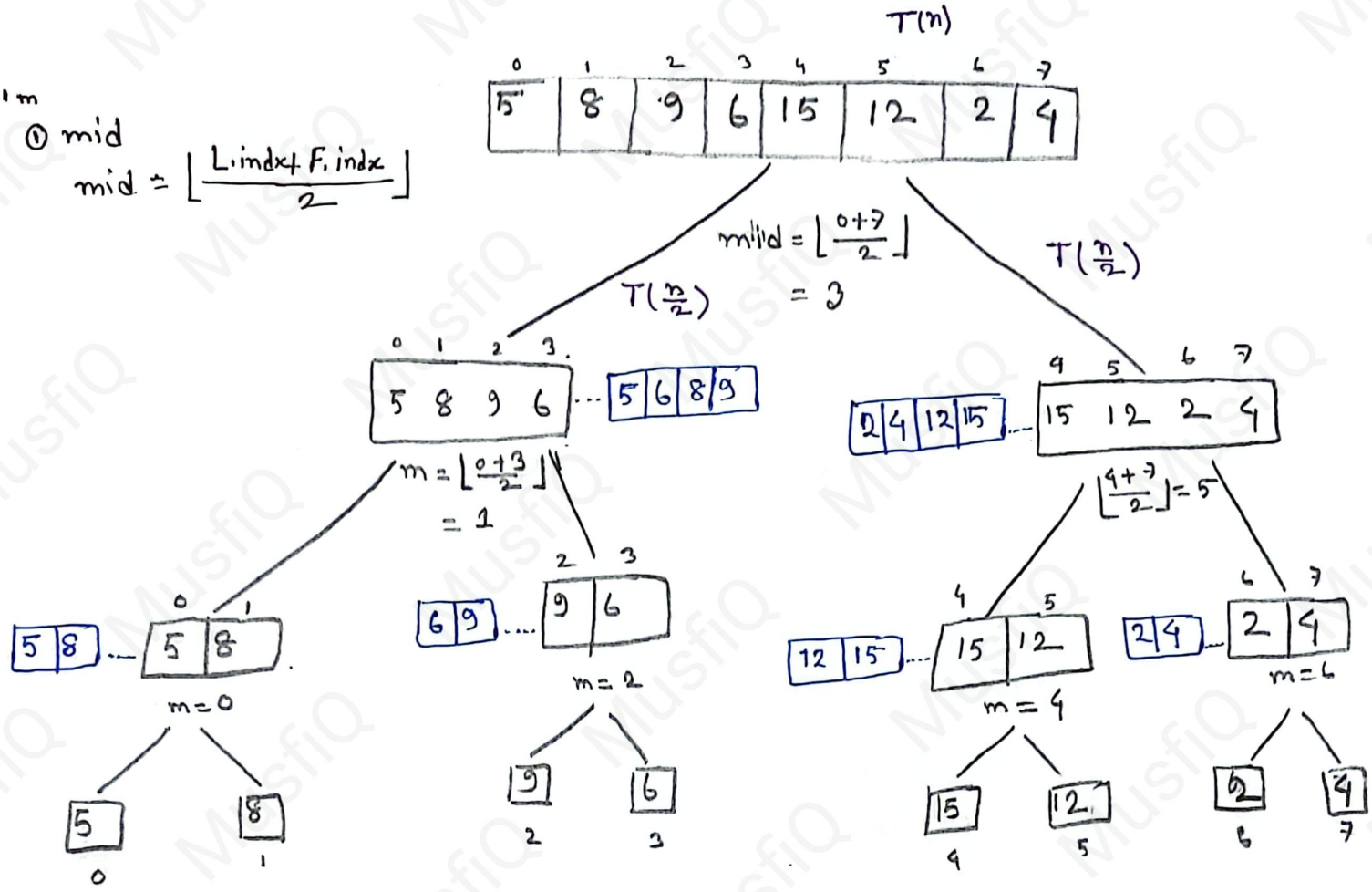



```
if (A[f] <= B[s]){

    A_B[i] = A[f];
    i = i+1;
    f = f+1;
}
else
  {
    A_B[i] = B[s];
      i = i+1;
      s = s+1;
  }
```

merge sort simulation (ascending) + T.C -

① mid

$$mid = \left\lfloor \frac{L.indx + F.indx}{2} \right\rfloor$$



A = | 5 | 6 | 8 | 9 |

B = | 2 | 4 | 12 | 15 |

A_B = | 2 | 4 | 5 | 6 | 8 | 9 | 12 | 15 |

$$T(n) = T(\tfrac{n}{2}) + T(\tfrac{n}{2}) + n$$

$$= 2T(\tfrac{n}{2}) + n$$

$$= O(n)$$

$$T(n) \begin{cases} 2T(\frac{n}{2}) + n, & \text{when } n > 1 \\ 1, & \text{when } n = 1 \end{cases}$$

Conquere time complexity

07-10-23

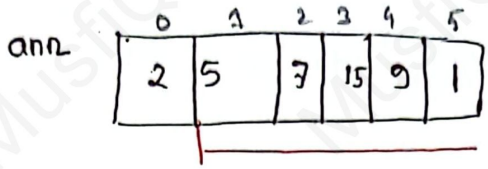
Min Max Problem

Using DnC approach

- Devide
- conquer
- Combine

" যদি Array টি 1 size এর হয় তাহলে min = max "

Example - 2



max = 2 5 7 15 বাকি গুলোর 15 বড় তাই

min = 2 1

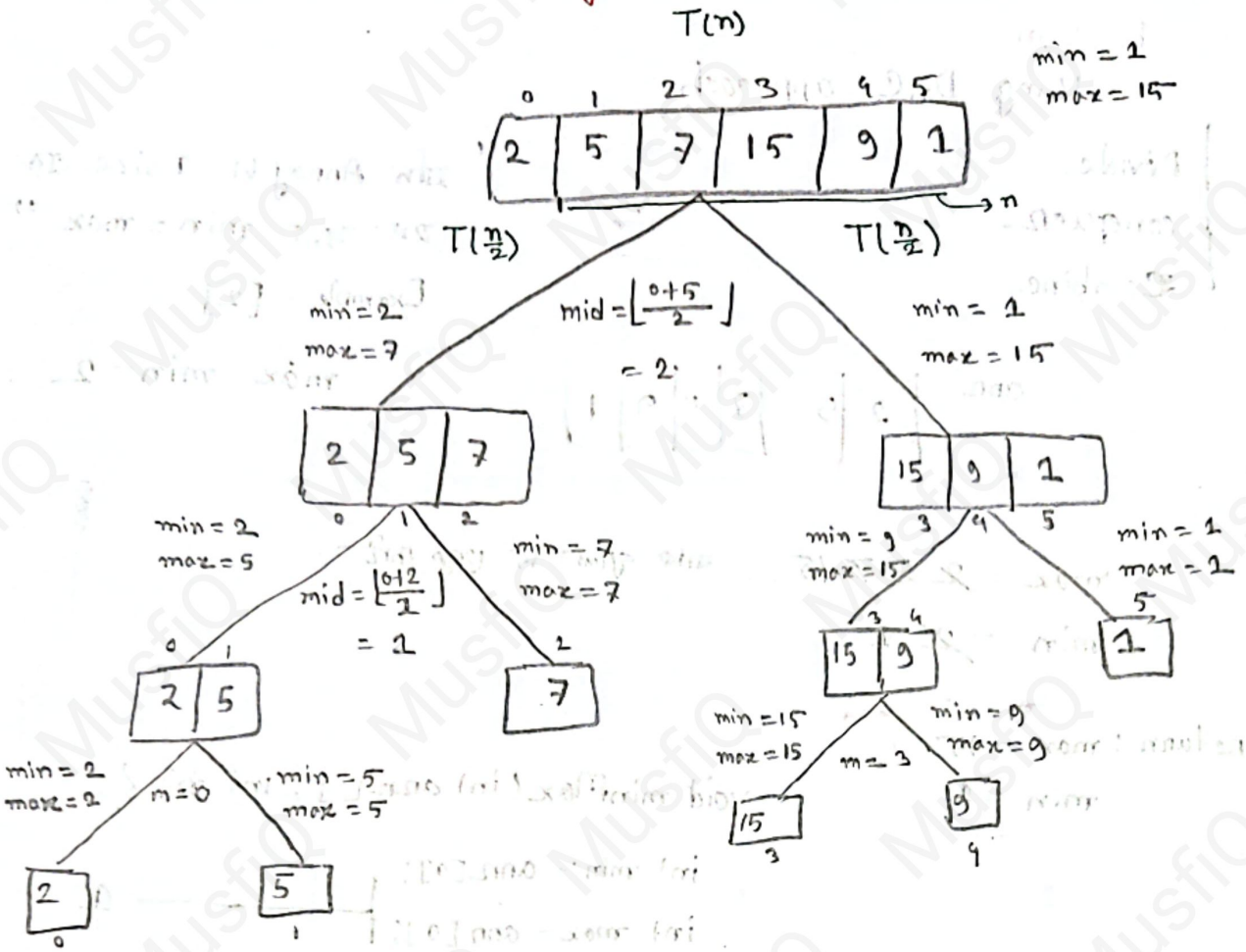
max = min = 2

return : max = 15
min = 1

" Find the minimum and the maximum of given array "

```
void minMax(int ann[], int n) {
    int min = ann[0];
    int max = ann[0];                    ---- 1

    for(int i=1; i<n; i++) {             ---- n
        if(ann[i] > max) {
            max = ann[i];
        }
        if(ann[i] < min) {               ---- 1
            min = ann[i];
        }
    }
    printf("%d %d\n", min, max);         ---- 1
}
```

Simulation + Time Complexity :



$$T(n) = T\left(\frac{n}{2}\right) + T\left(\frac{n}{2}\right) + c$$

$$= 2T\left(\frac{n}{2}\right) + c$$

$$T(n) = \begin{cases} 2T\left(\frac{n}{2}\right) + c & ; n > 1 \\ 1 & ; n = 1 \end{cases}$$

Maximum Sum Subarray:

Using DnC approach

- Subarray
- Sub-sequence

all subarray are subsequence, but all subsequence are not subarray

arr | 5 | 9 | 8 | 1 |

5 | 9 → sub array (পর পর 2টি 2টি)

9 | 1 → sub sequence (পর পর না, কিন্তু order maintain করছে 2টি)

* $\frac{n(n+1)}{2}$ → sub array find

5	= 5		9	= 9		8	= 8		2	= 2 (1)
5 | 9	= 14		9 | 8	= 17		8 | 2	= 10 (2)			
5 | 9 | 8	= 22		9 | 8 | 2	= 19 (3)						
5 | 9 | 8 | 2	= 24 (4)									

কোন sub array কিন্তু maximum sum এর value পাবে সেটাই

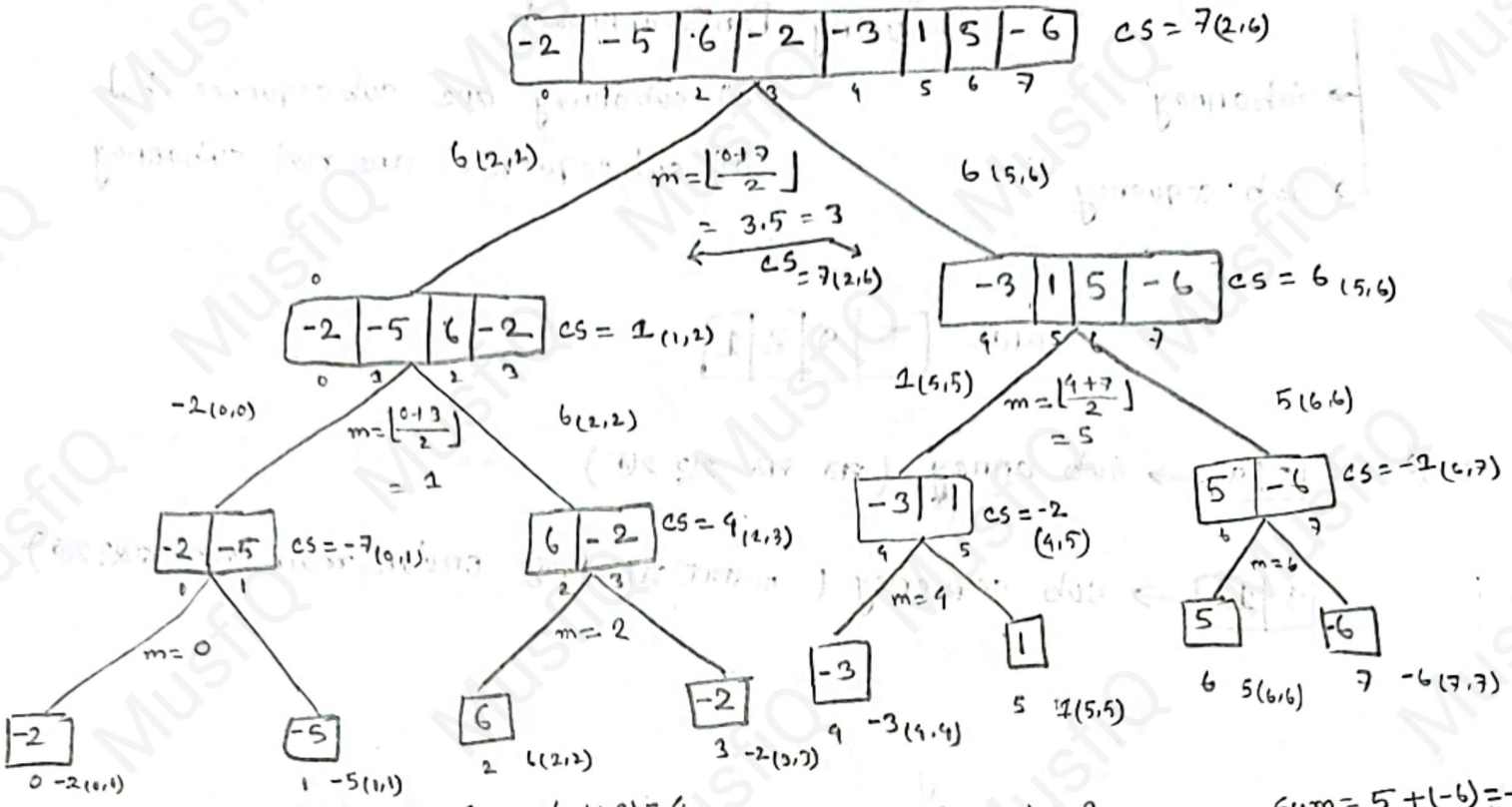
Max sum subarray. -10 | 17 | -15 | 50 = (52)

-10	= -10		17	= 17		-15	= -15		50	= 50
-10 | 17	= 7		17 | -15	= 2		-15 | 50	= 35			
-10 | 17 | -15	= -8		17 -15 50	= (52)						
-10 | 17 | -15 | 50	= 42									

O(n³) → optimize → O(n²) → DnC → O(nlog₂n)

kadOnes algorithm O(n)

Simulation + Time complexity



Sum = -5 + (-2) = -7

Lsum = $-5_{(1,1)}$

Sum = 6 + (-2) = 4

Rsum = $6_{(2,2)}$

Sum = 1 + (-3) = -2

Lsum = $1_{(5,5)}$

Sum = 5 + (-6) = -1

RS = $5_{(6,6)}$

crossing sum = LS + RS

= 1 + 5 = 6 (5,6)

Sum = -2 + 6 = 4 + (-5) = -1 + (-2) = -3

Lsum = $-2\ 4_{(2,2)}$

Sum = -3 + 1 = -2 + 5 = 3 + (-6) = -3

RS = $-3\ -2\ 3_{(6,6)}$

crossing sum = 4 + 3

= 7 (2,6)

Mange sort

$$T(n) = \begin{cases} 2T\left(\frac{n}{2}\right) + n & ; n > 1 \\ 1 & ; n = 1 \end{cases}$$

min max problem

$$T(n) = \begin{cases} 2T\left(\frac{n}{2}\right) + 1 & ; n > 1 \\ 1 & ; n = 1 \end{cases}$$

maximum sub array sum

$$T(n) = \begin{cases} 2T\left(\frac{n}{2}\right) + n & ; n > 1 \\ 1 & ; n = 1 \end{cases}$$

Binary Search

$$T(n) = \begin{cases} T\left(\frac{n}{2}\right) + 1 & ; n > 1 \\ 1 & ; n = 1 \end{cases}$$

solving.

Recursion Solving

① Substitution

② master theorm

③ Recursion tree.

Recursion tree methods:

$$T(n) = \begin{cases} bT\left(\frac{n}{a}\right) + \text{combine complexity} \\ \text{conerne} \end{cases}$$

Example 1:

$$T(n) = \begin{cases} 2T\left(\frac{n}{2}\right) + n & ; n > 1 \\ 1 & ; n = 1 \end{cases}$$

$$a\log_k b = b\log_k a$$

$$5\log_2 4 = 4\log_2 5$$

(i) Divide $= \frac{n}{a} =$

(ii) No of child of (each node) $= b =$

(iii) Combine complexity $=$

(iv) Conerne complexity $=$

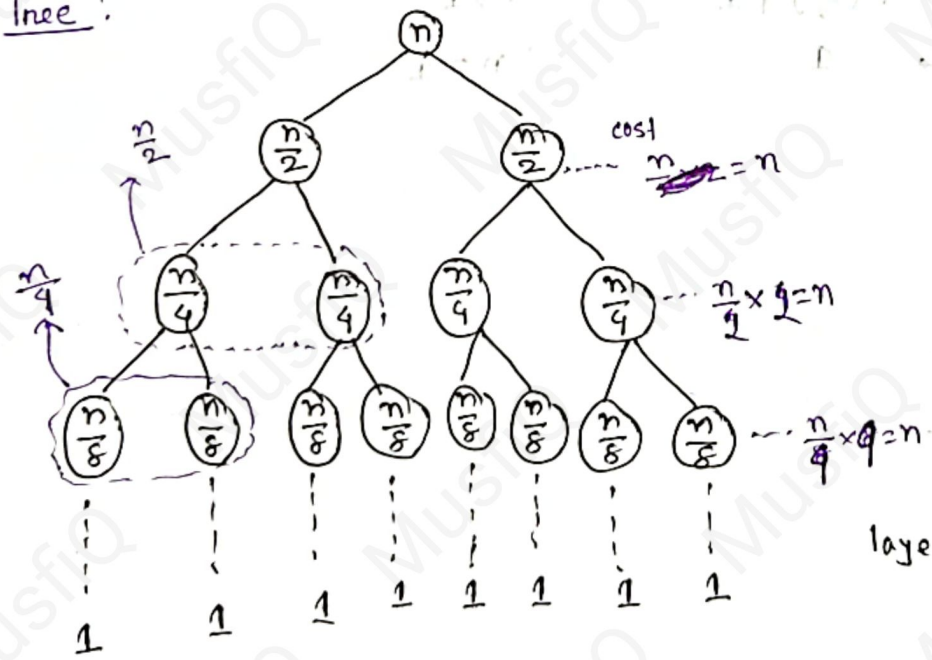
(v) number of leaf node $= b \log_a n$

$T(n)$ = internal layer complexity + leaf layer complixity

(i) Leaf layer complexity = (nuber of leaf node * conerne complexity)

(ii) Internal layer complexity = Layer * cost

Tree:



(i) Divide $= \frac{n}{2}$

(ii) No. of. ch(e.N) $= b = 2$

(iii) Combine complexity $= n$

(iv) Conerne $= n$ $= 1$

(v) No of leaf node $= 2\log_2 n$

$T(n) = n\log_2 n + n$

$= O(n \log_2 n)$

layer $= \log_2 n$

Indennal layerc $= \log_2 n \times n$

$= n \log_2 n$

leaf layer c. $= 2\log_2 n * 1$

$= n \log_2 2$

$= n$

Example-2 (minMax)

$$T(n) = \begin{cases} 2T\left(\frac{n}{2}\right) + 1 & ; n > 1 \\ 1 & ; n = 1 \end{cases}$$

$\frac{a}{n-1}$	$n > 1$
$\frac{a}{1-n}$	$n < 1$

Divide $= \frac{n}{a} = \frac{n}{2}$

No of child (each node) $= b = 2$

Combine complexity $= 1$

Conquer complexity $= 1$

Num of leaf node $= b^{\log_a n} = 2^{\log_2 n}$

leaf layer complexity $= 2^{\log_2 n} \times 1$

$= n^{\log_2 2} \times 1$

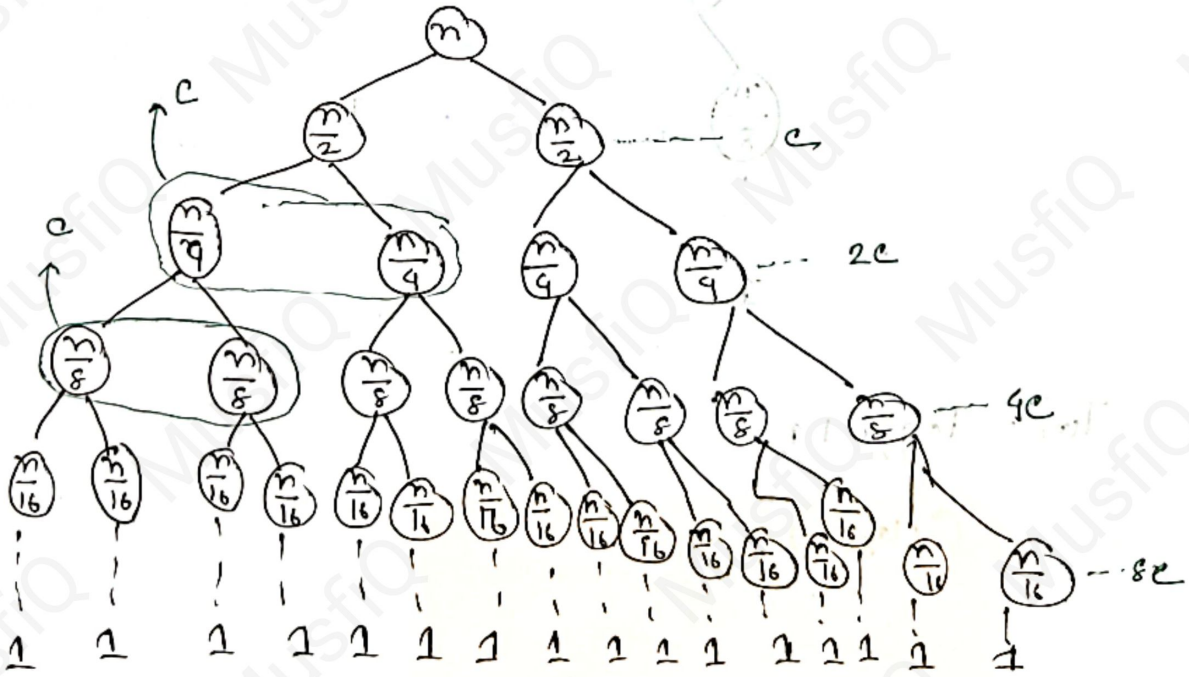
$= n$

Internal layer complexity $= c + 2c + 4c + 8c + 16c \cdots$

$= c(2 + 4 + 8 + 16 \cdots)$

$T(n) = c(2+4+8+16) + n$

$= O(n)$



Example - 3

$$T(n) = \begin{cases} T(\frac{n}{2}) + 1 & ; n > 1 \\ 1 & ; n = 1 \end{cases}$$

Divide : $\frac{n}{a} = \frac{n}{2}$

number of child (each node) $= b = 1$.

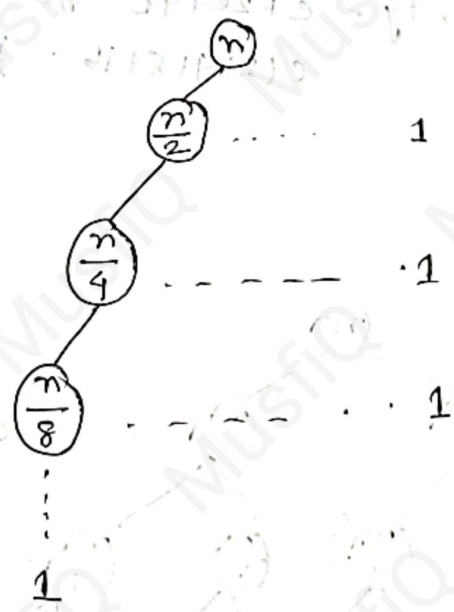
Combine complexity $= 1$

Concure complexity $= 1$

no of leaf node $= b\log_a n = 1^{\log_2 n} = n^{\log_2 1} = n^0 = 1$

Internal layer complexity $= \log_2 n * 1$

leaf layer complexity $= 1 \times 1 = 1$



$T(n) = \log_2 n + 1$

$= O(\log_2 n)$.

Example-4

$$T(n) = \begin{cases} 4T\left(\frac{n}{2}\right) + n & ; n > 1 \\ 1 & ; n = 1 \end{cases}$$

Divide $= \frac{n}{a} = \frac{n}{2}$

no of child (EN) $= b = 4$

Combine complexity $= n$

Conquer complexity $= 1$

no of leaf node $= b \log_a n = 4 \log_2 n = n \log_2 2^2 = n^2$

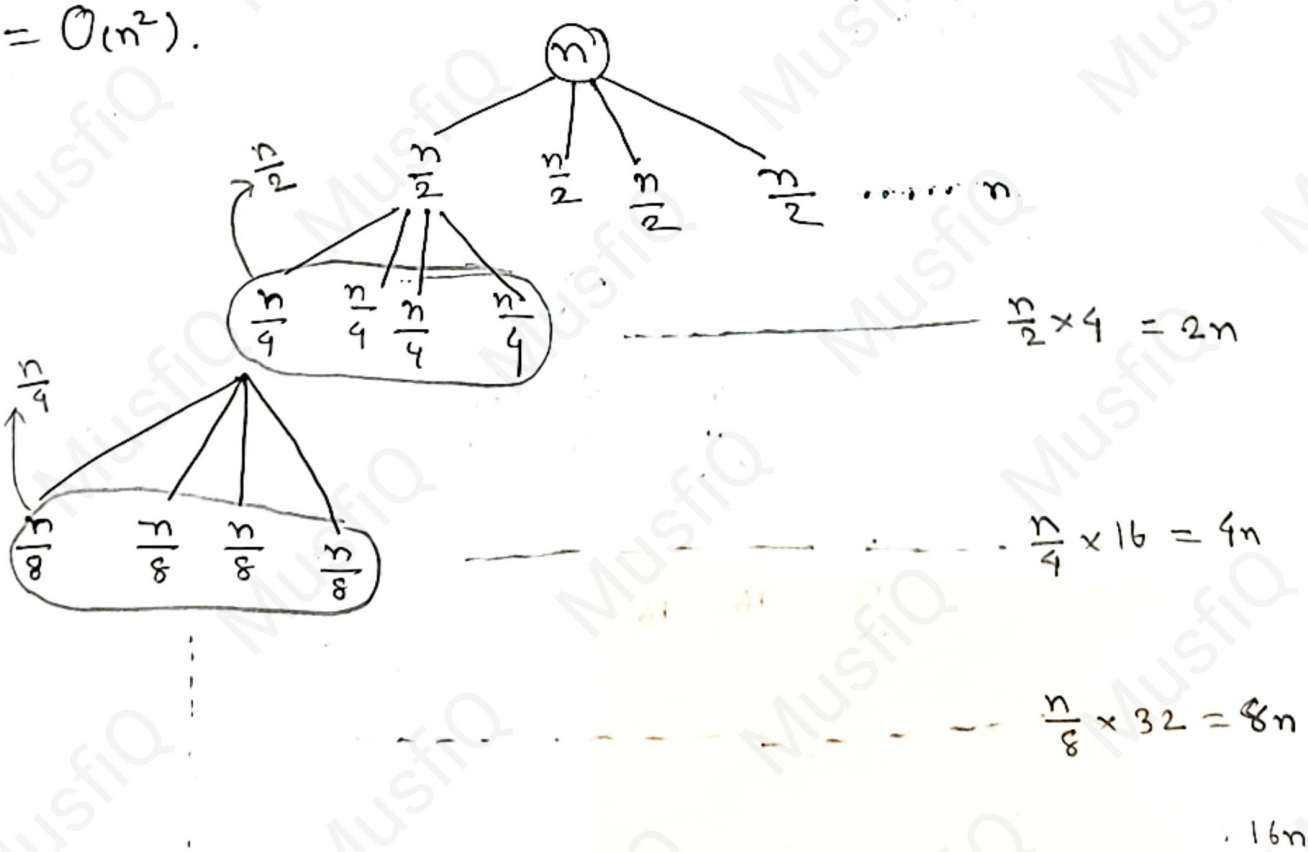
Leaf layer complexity $= (n^2 \times 1) = n^2$

Internal layer complexity $= n + 2n + 4n + 8n + 16n \cdots$

$= n(2 + 4 + 8 + 16 \cdots)$

$T(n) = n(2+4+8+16\cdots) + n^2$

$= O(n^2)$.



Example - 5

$$T(n) = \begin{cases} 3T\left(\frac{n}{4}\right) + n^2 & ; n > 1 \\ 1 & ; n = \end{cases}$$

Divide $= \frac{n}{a} = \frac{n}{4}$

no. of child (em) $= b = 3$

Combine complexity $= n^2$

Conquer complexity $= 1$

no of leaf node $= b^{\log_a n} = 3^{\log_4 n} = n^{\log_4 3} = n^{0.79}$

leaf layer complexity $= n^{0.79} * 1 = n^{0.79}$

Internal layer complexity $= n^2 + n^2\left(\frac{3}{16}\right) + n^2\left(\frac{3}{16}\right)^2 + n^2\left(\frac{3}{16}\right)^3 \cdots$

$= n^2(c)$

$T(n) = n^2 c + n^{0.79}$

$= O(n^2)$