We will start at 8:34

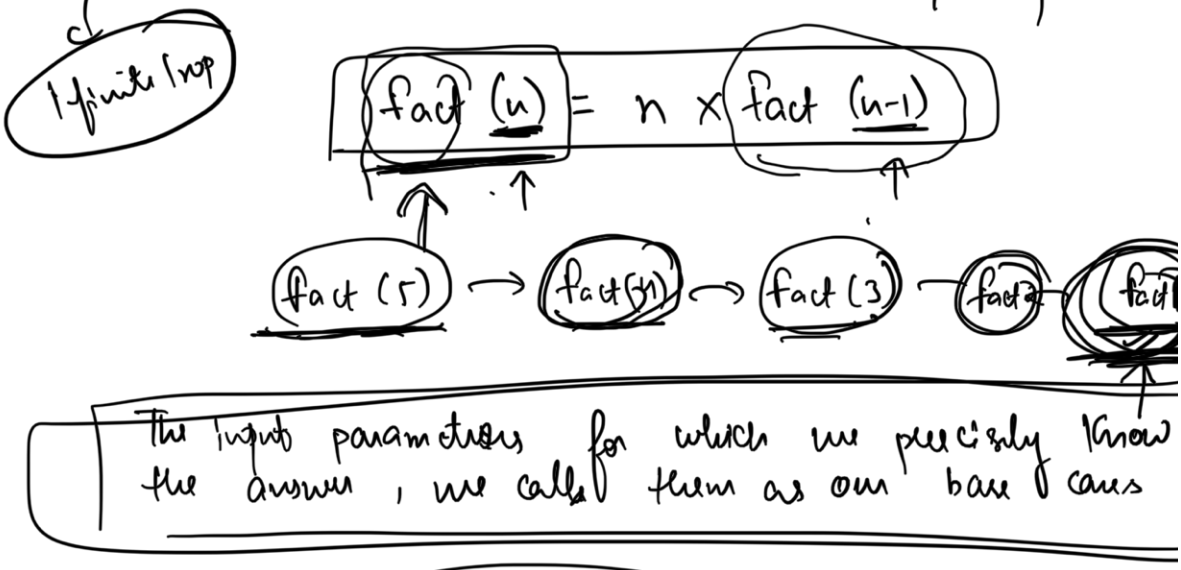
We need to find the number which occurs only once.

Agenda

→ Merge Sort Algorithms
→ Real life examples of merge Sort.

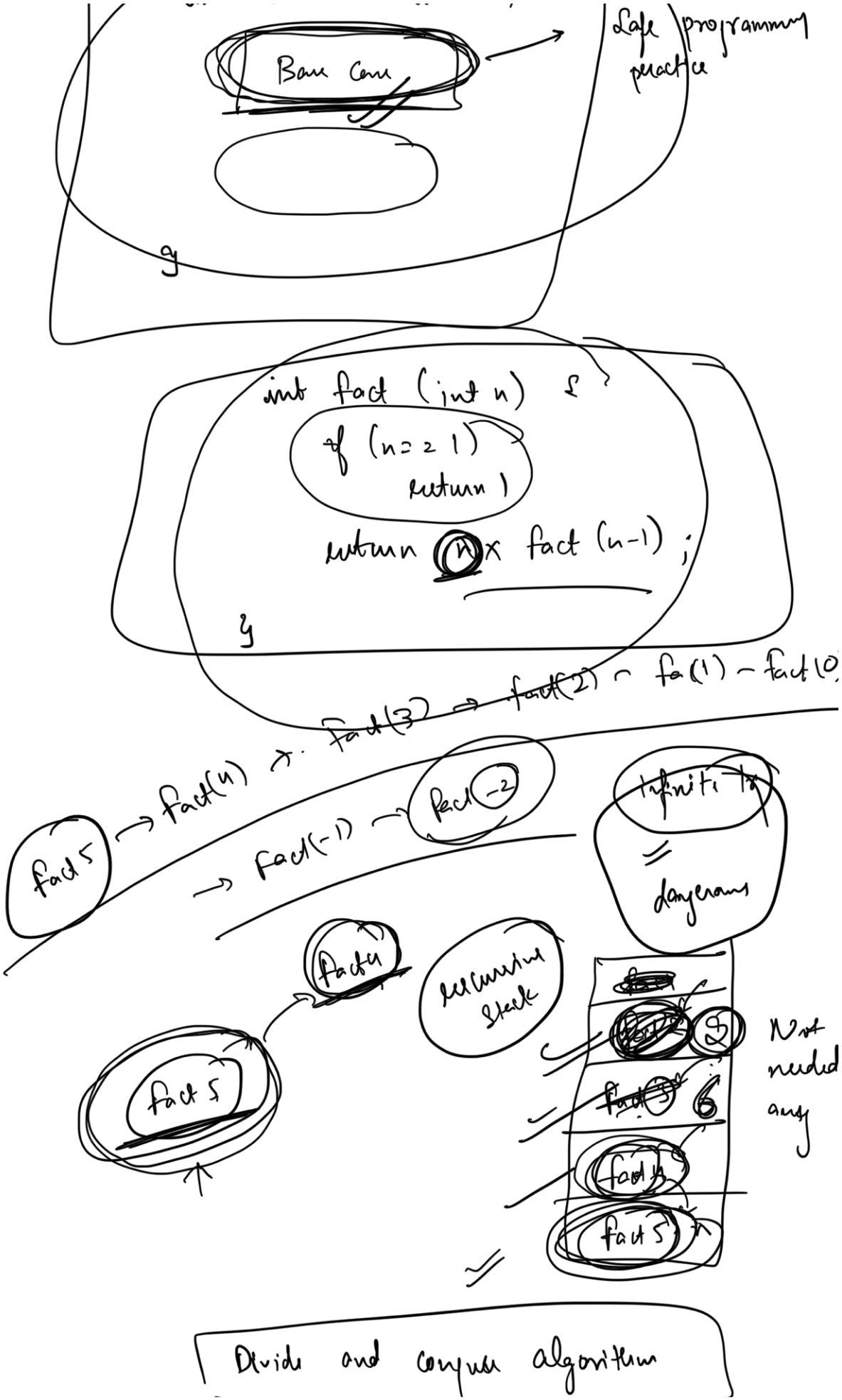
Recursion

→ Calling the same function again inside the func but with the reduced input paramete



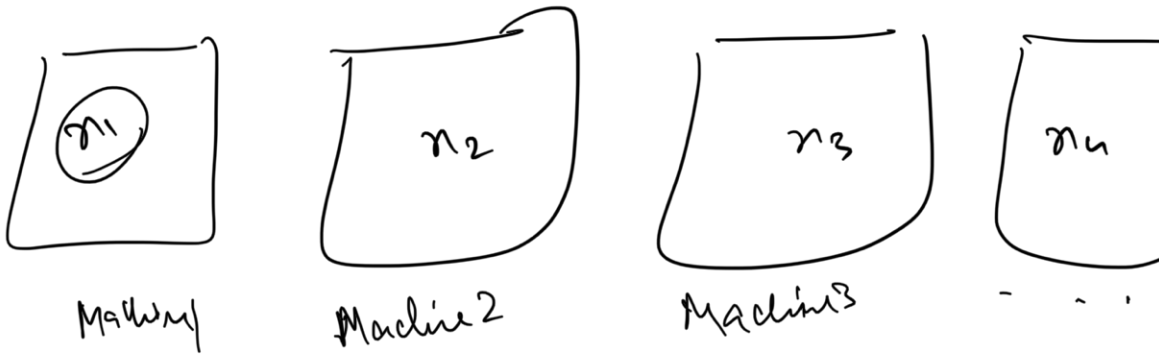
The input parameters for which we precisely know the answer, we call them as our base cases

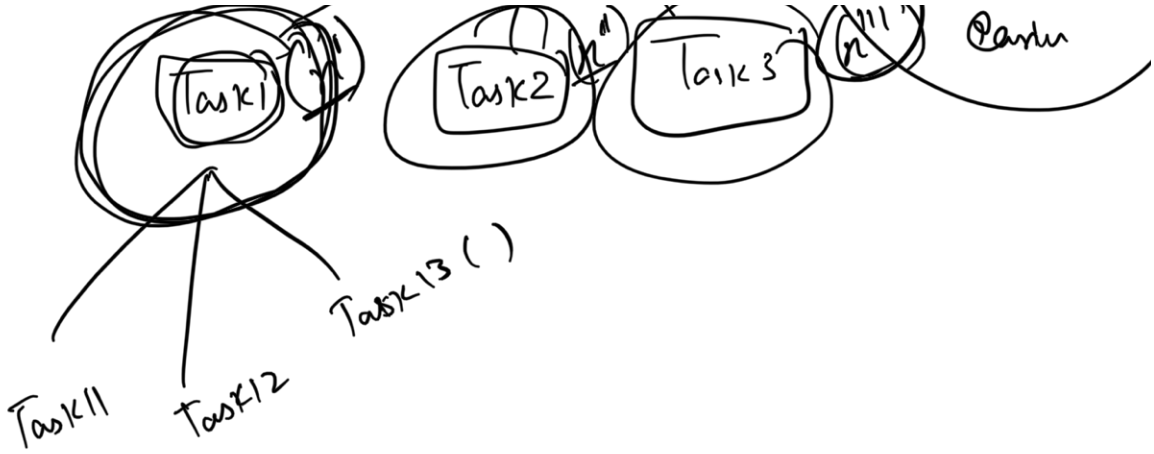
```
int fact ( int n ) {
```


Safe programming practice
Base case
g
int fact (int n) {
if (n == 1)
return 1
return n * fact (n-1) ;
}
fact 5
fact(4)
fact(3)
fact(2)
fact(1)
fact(0)
fact(-1)
fact(-2)
infinite
dangerous
recursive stack
fact 5
fact 5
fact 5
fact 5
fact 5
Not needed anyway
Divide and conquer algorithm

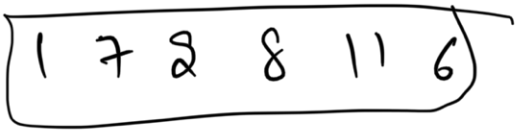
The question is you have a massive dictionary of words and we want to calculate the number of words in the dictionary !

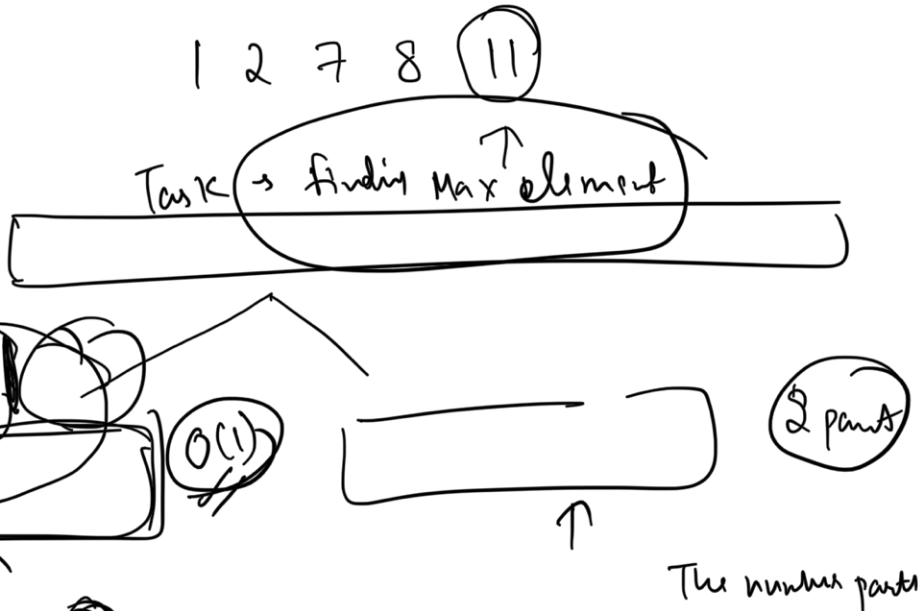
How will we able to calculate this





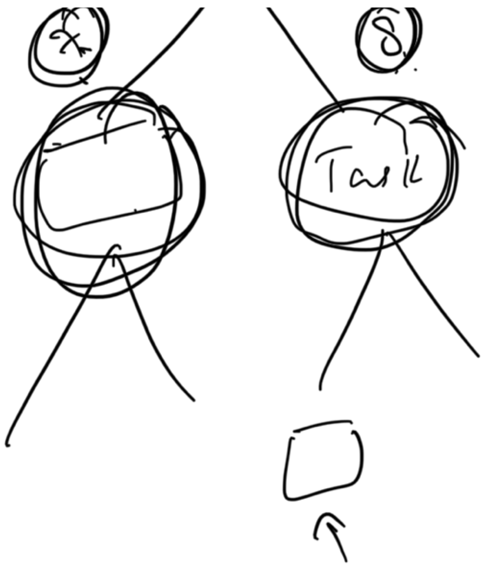
→ You will able to calculate max element in the array



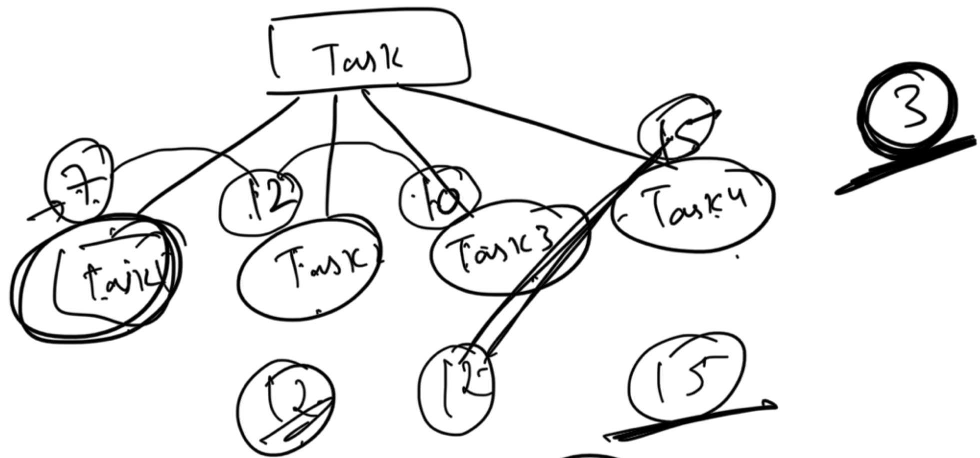


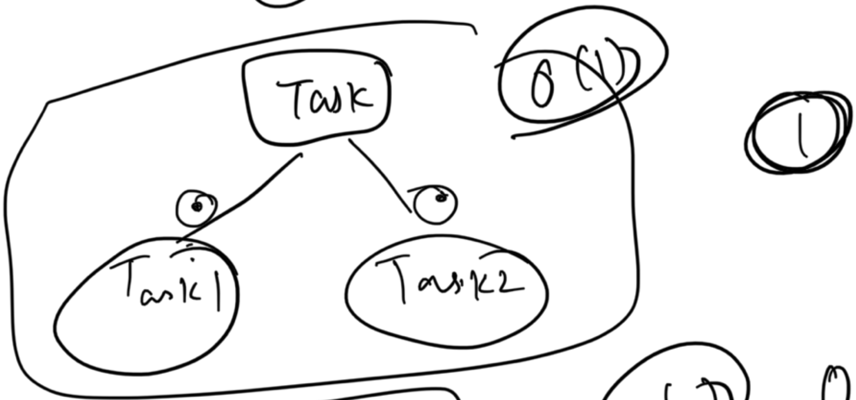
The number parts

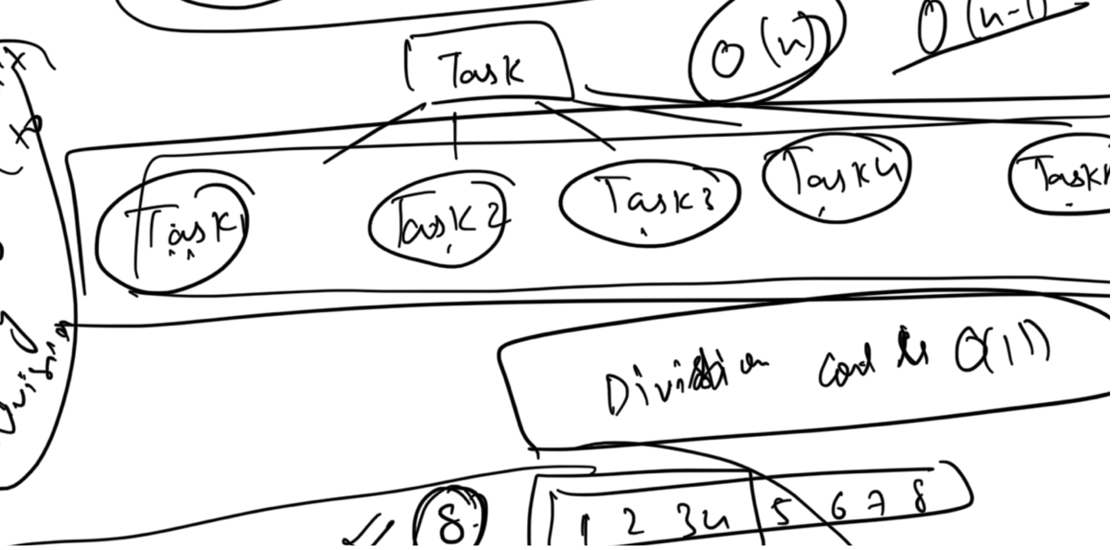
we need to divid
in, ?? decides t
complexity



How much

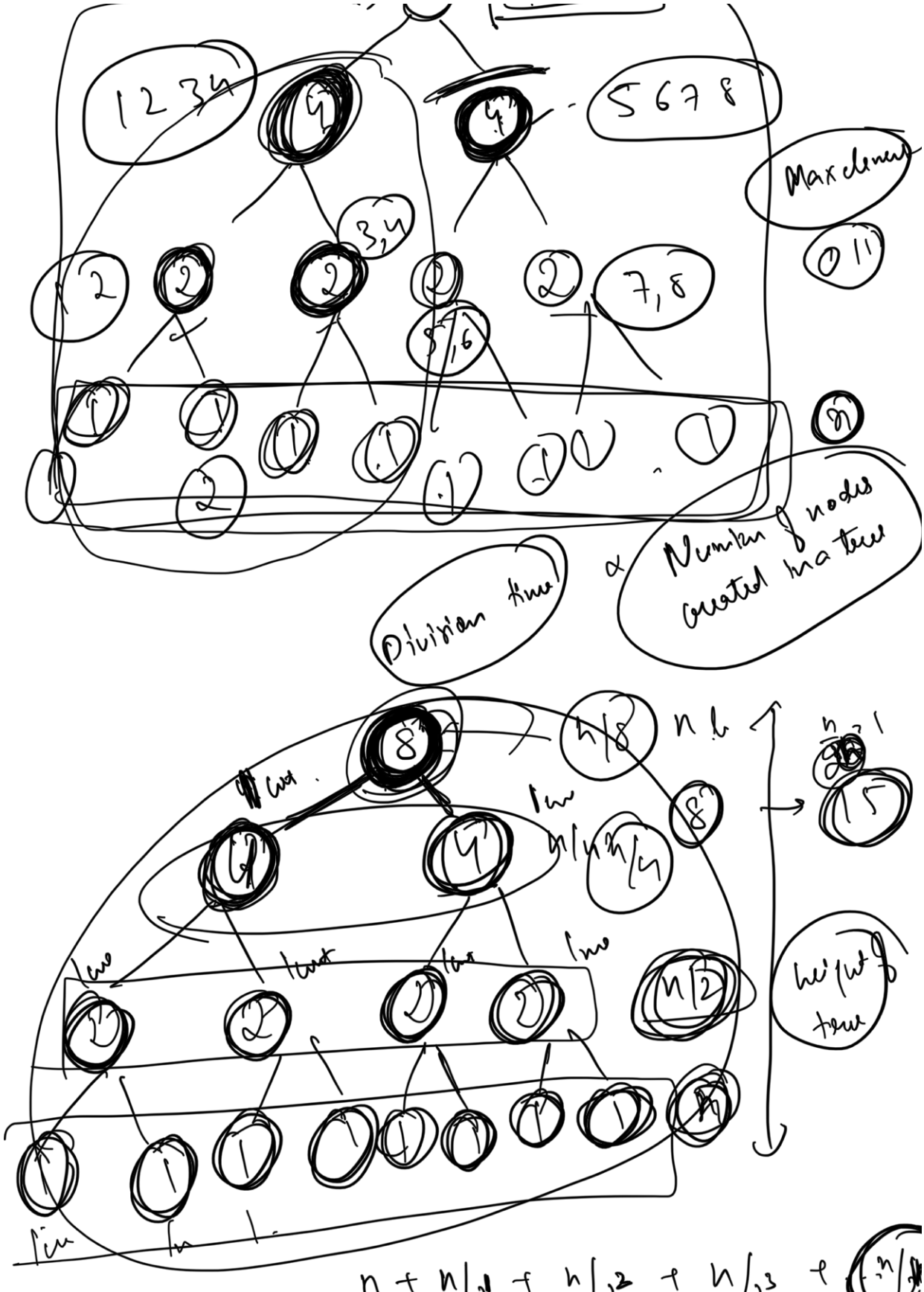






don't need to do any division

Division cost is O(1)



$$n + n/2^1 + n/2^2 + n/2^3 + \cdots + n/2^{\log n}$$

$$n\left(1 + \frac{1}{2} + \frac{1}{4} + \frac{1}{8}\right) + \cdots \frac{1}{n}$$

$\frac{n}{2^h} = 1$

$n \simeq 2^h$

$\log_2 n = h$

$n \in 1 + \frac{1}{2} \ldots \log n$

$n \left[\frac{1}{2} \left(\frac{n}{\sum_{k=0}^{1}} \right) \frac{1 - q^n}{1 - q} \right]$

$1 - \frac{1}{2^{\log_2 n}}$

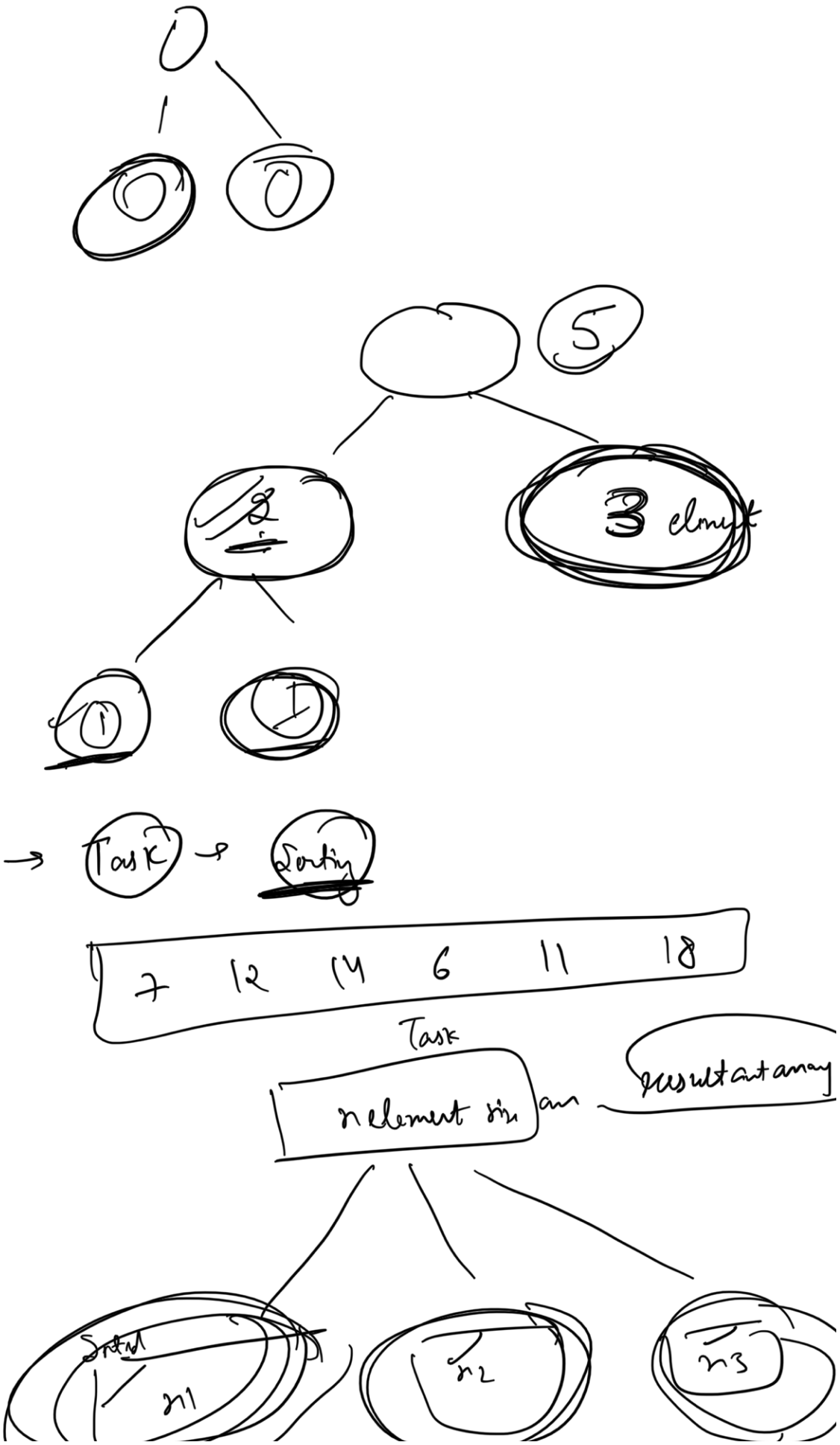
$1 - \frac{1}{n}$

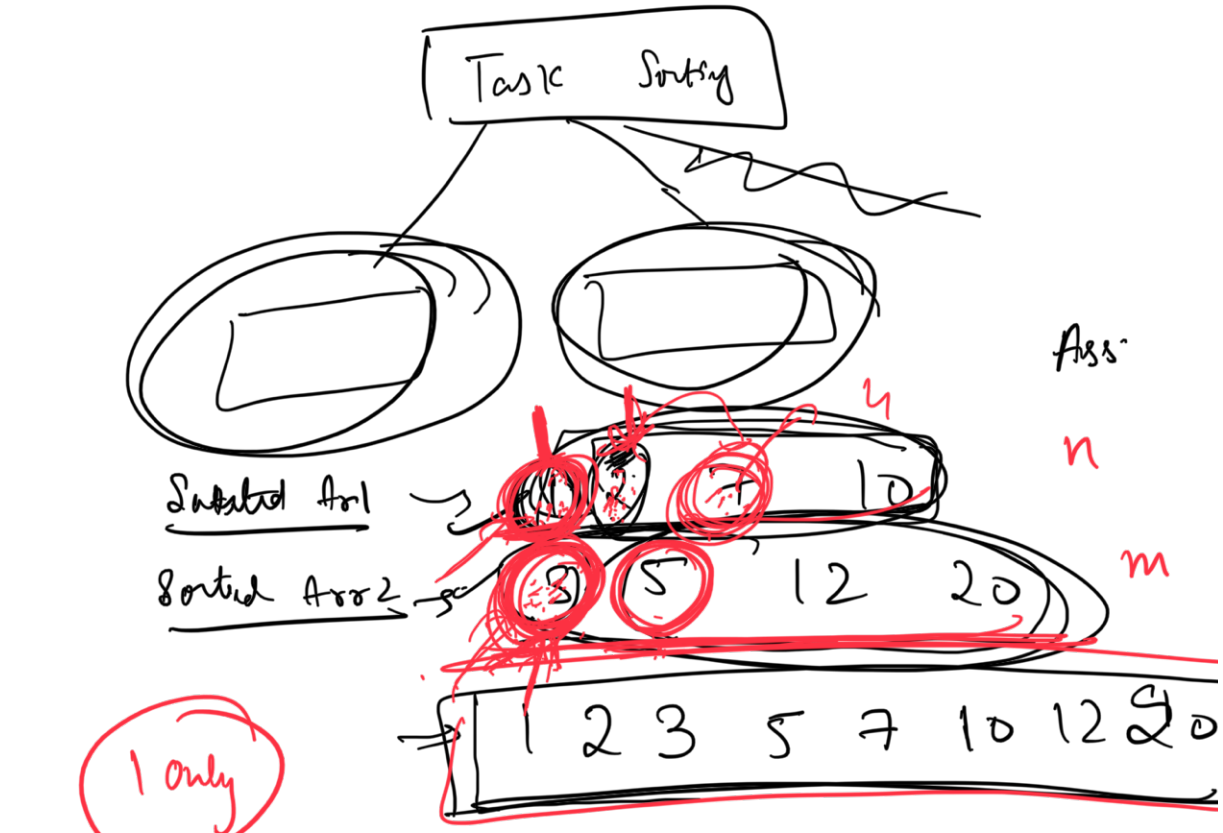
$O(n)$

n

$n \quad \frac{n-1}{n} \quad \frac{1}{2} \quad n-1$

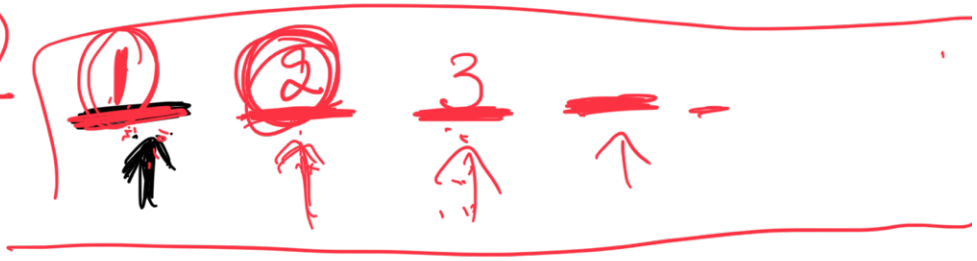
5


0
0
0
5
2
3 elmt
1
1
Task
Sorting
7 12 14 6 11 18
Task
n element in
result array
Sorted
n1
n2
n3



I only

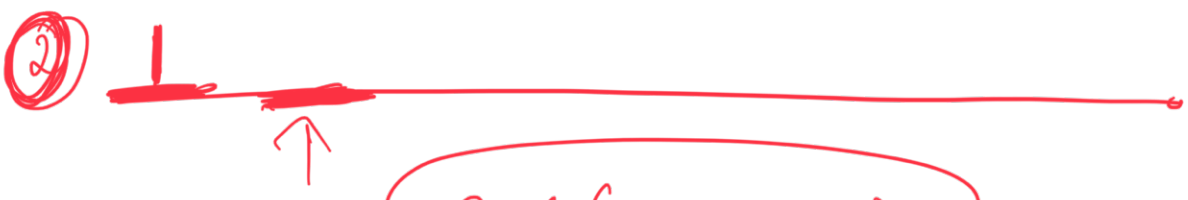
$O(n+m)$



Sorted Array 1 — 4 5 7 8

Sorted Array 2 — 1 12 16 20

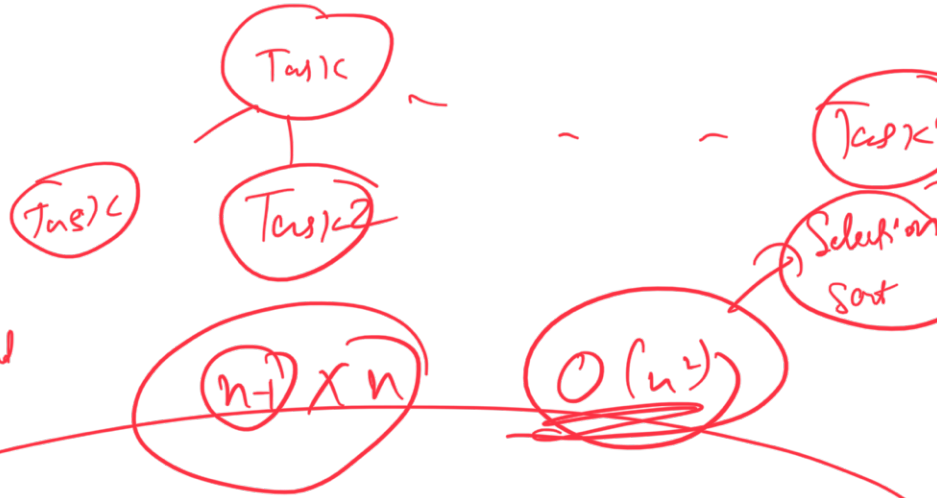
Sorted Array 3 — 3 4 5 6



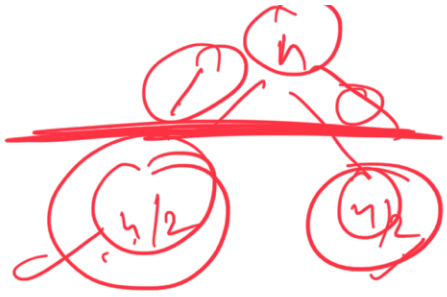
$2 \times (n_1 + n_2 + n_3)$

If I divided it in to 4 tasks

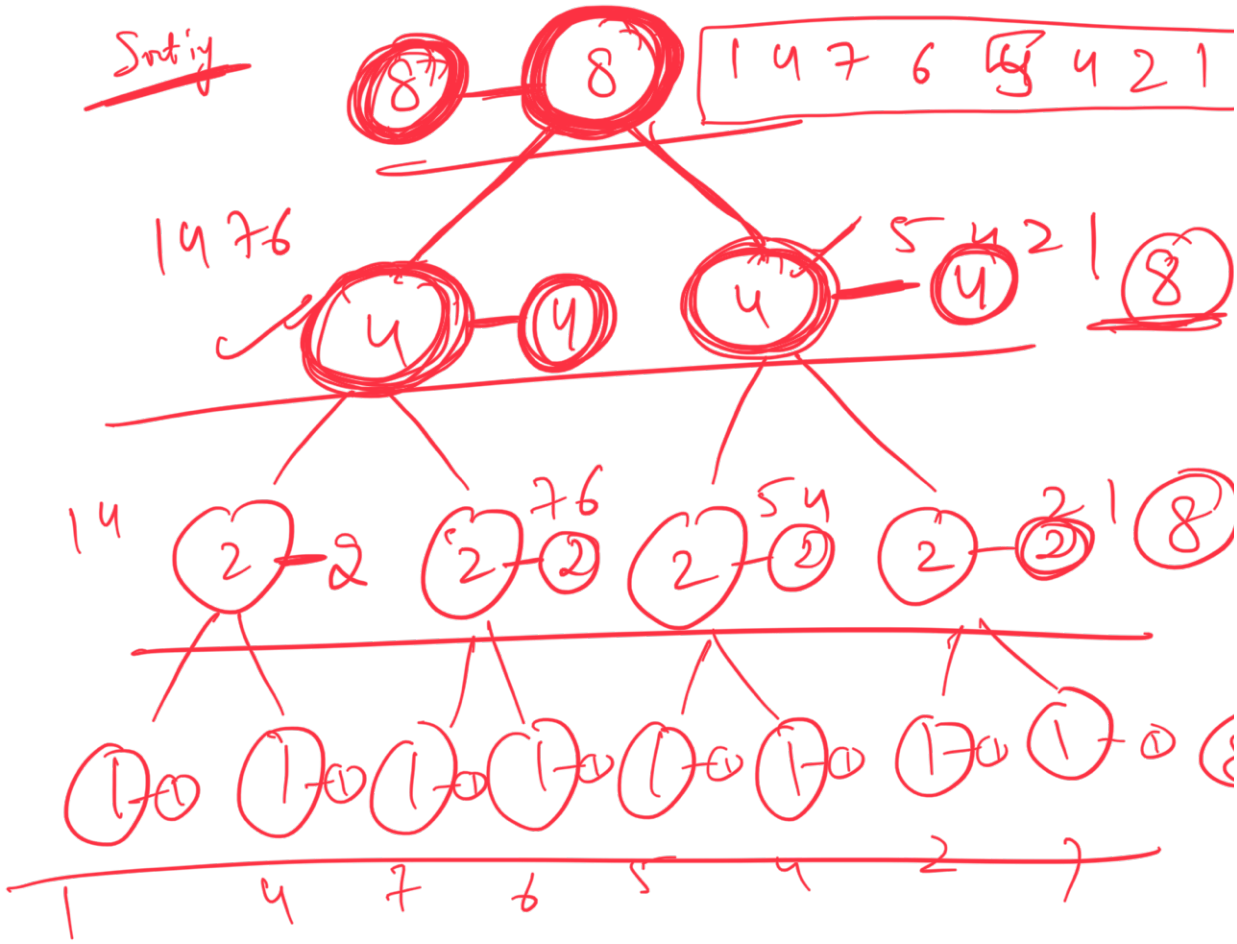
3 X Total



$$T(n) = 2 \times T(n/2) + O(n)$$



$$T(n) = 3 \times T(n/3) + 2n$$



$$T(n) = 2 \times T(n/2) + n$$

$$T(n) = 3 \times T(n/3) + 2n$$

$$T(n) = 4 \times T(n/4) + 4n$$

8 x number of levels

↓

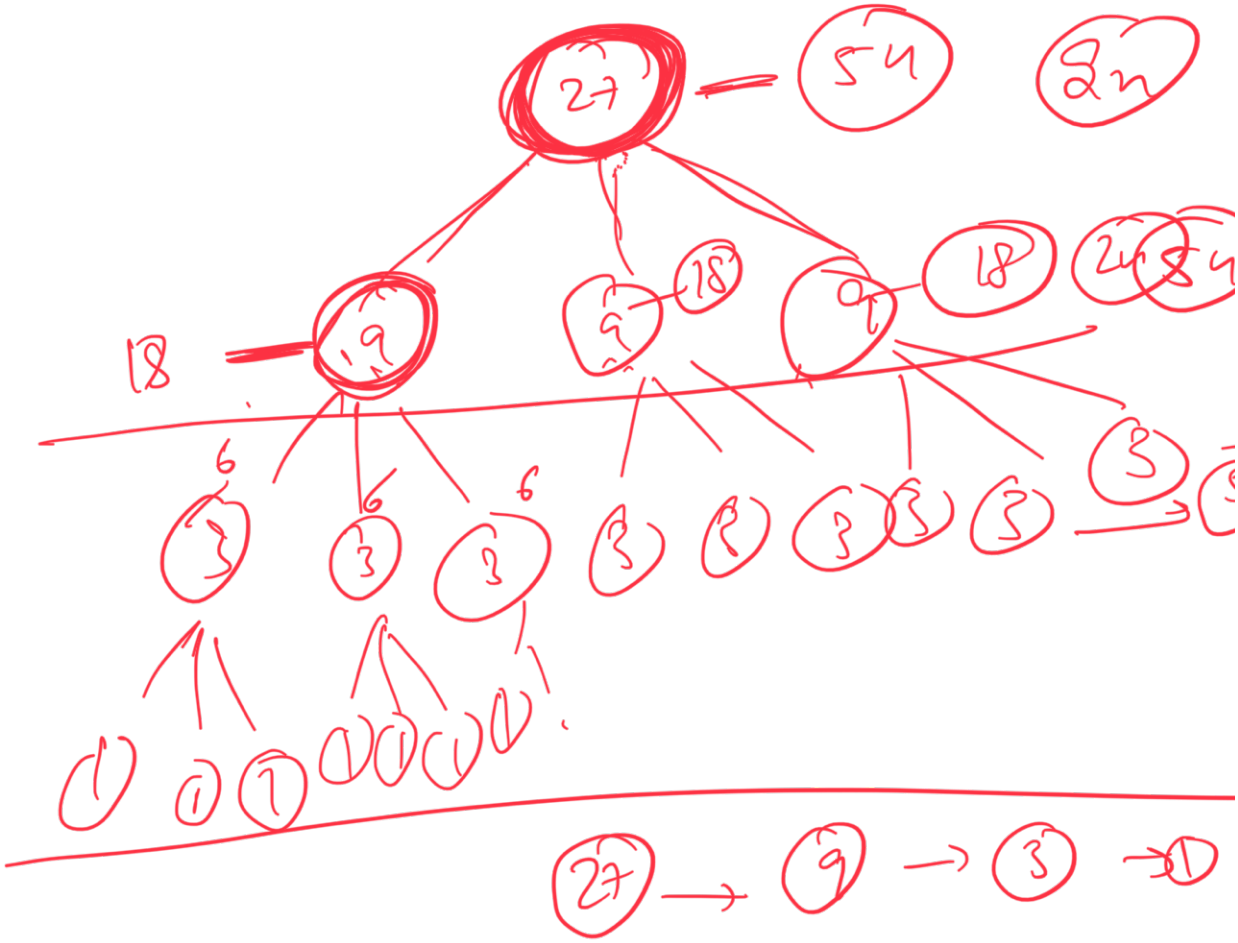
height of tree

8 x $\log_2 8$

$n \log n$

$n \times \log_2 n$

merge sort Algorithm



$n + n/3 + n/3^2 + \ldots n/3^k$

$n/3^k = 1$

$n = 3^h$

$\log_3 n$

$2n \log_3 n$

$3n \log_4 n$

$n \log_2 n$

$2n \log_3 n$

$3n \log_4 n$

$4n \log_5 n$

$f(k)$ (

$f(k) = (k-1)\, n \log_k n$

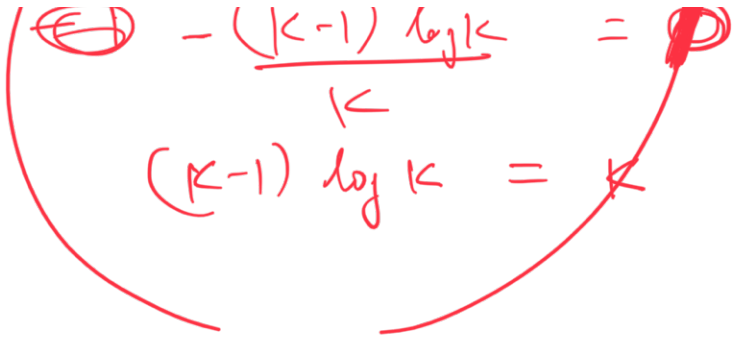
$= \dfrac{(k-1)\, n \log n}{\log k}$

$\log_k n = \dfrac{\log_3 n}{\log_3 k}$

$k = 2$

$f'(k) = \dfrac{(k-1)'}{\log k} =$

$\dfrac{-(k-1)\frac{1}{k}}{(\log k)^2}$



$n \log_2 n$ $\quad$ $2 \times 8 \log_3 n$

8×3 $\quad$ $2 \cdot \times 8 \log_3 8$

24 $\quad$ $16 \times 1.7 \sim 24$

$n \log n$ → merge sort

Netflix $\quad$ Youtube $\quad$ recommendation

⑥

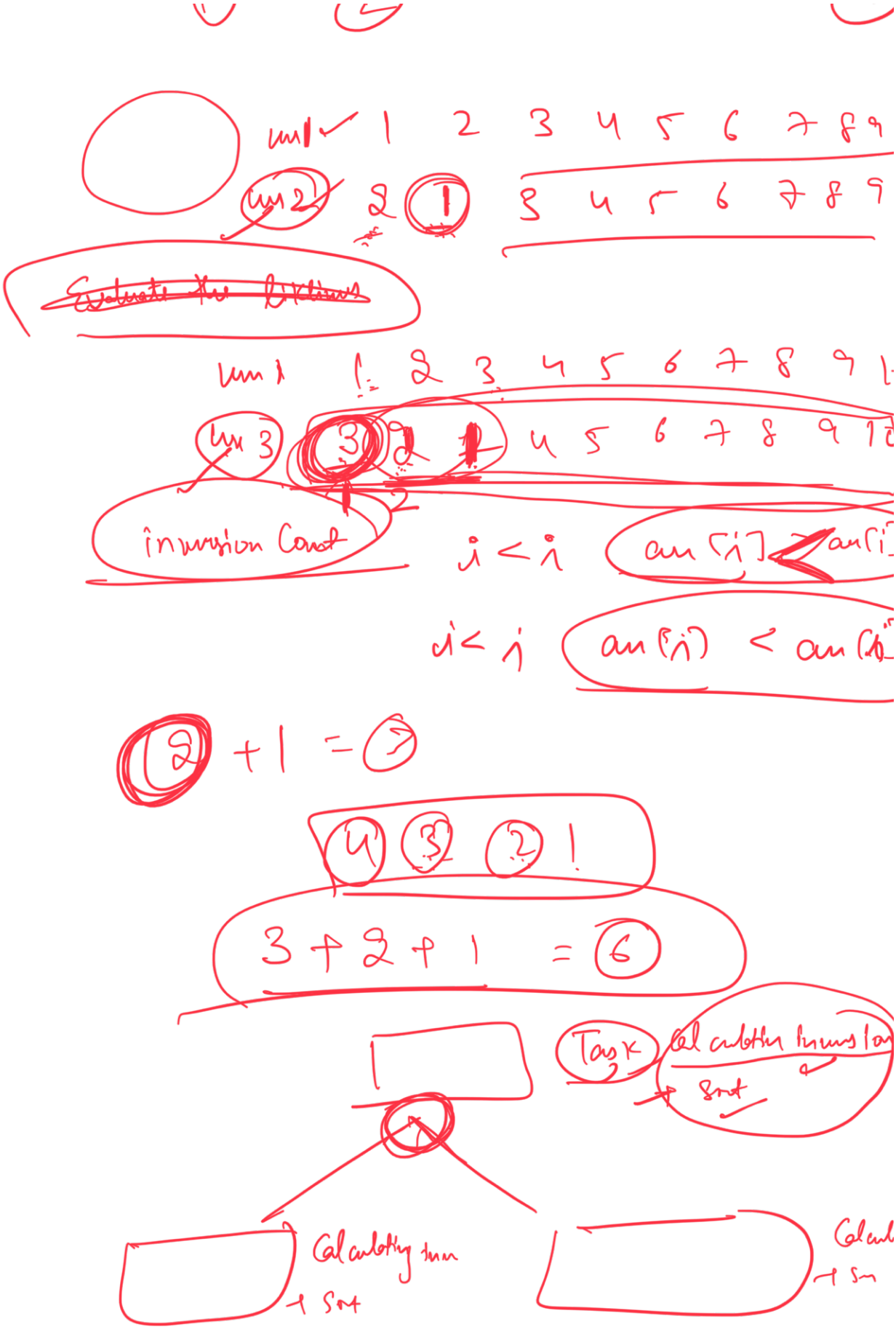
User1 → 1 2 3 4 5 6 7 8 9

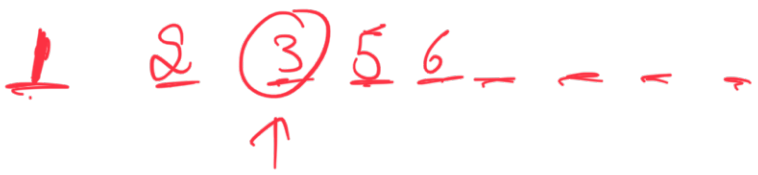
User2 → 10 9 8 7 6 5 4 3 2 1

Max known $\quad$ very diff

Movie1 $\quad$ Movie2 $\quad$ . . . $\quad$ 10

1 $\quad$ 1 $\quad$ 10


arr 1 2 3 4 5 6 7 8 9
arr 2 2 1 3 4 5 6 7 8 9
Evaluate the Options
arr 1 2 3 4 5 6 7 8 9
arr 3 3 2 1 4 5 6 7 8 9
inversion Count
i < j arr[i] < arr[j]
i < j arr[i] < arr[j]
2 + 1 = 3
4 3 2 1
3 + 2 + 1 = 6
Task
Calculating inversion
Sort
Calculating inv
Sort
Calcul
Sor



Ans

What is the Best case time complexity of Merge Sort Algorithm.

→ $O(n^2)$ (A)

→ $O(n^2 \log n)$ (B)

→ $O(n \log n)$ (C)

→ $O(n \log n^2)$ (D)

If your array is already sorted then which algorithm is fastest

→ Bubble sort

- Insertion Sort
- Selection Sort
- Merge Sort