$[\longrightarrow]$

Q: Given a list of integers, arrange them in an order

such that $\longrightarrow$ $a_1 >= a_2 <= a_3 >= a_4 <= a_5 \ldots$

Ex $\rightarrow$ $[1, 2, 3, 4]$ $\qquad$ $n \leq 10^7$

ans $\rightarrow$ $[2, 1, 4, 3]$ or $[4, 1, 3, 2]$

[1, 0, 25, 12, 16, 15, 30, 40, 50]

$\uparrow i$

warm-up

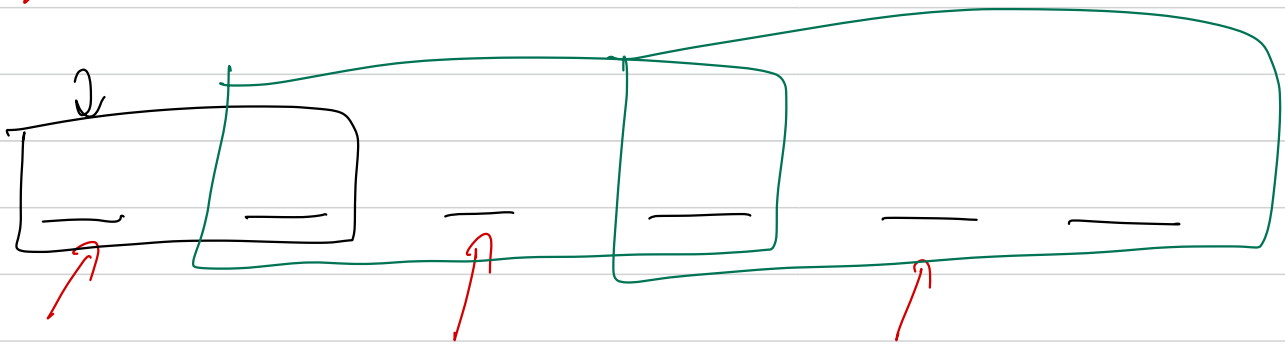
$a, b \rightarrow a > b$

$a, b \rightarrow a < b \rightarrow$ swap (a, b)

$a, b \rightarrow a = b$

scene the 0th pos

→ We can just arrange all the odd or all the even values, the counter part will be already arranged.



2 1 4 3 5

3 2 9 4 6 1

5 2 9 2 3

```python
a1 = [1,2,6,8,19,22] # sorted
a2 = [1,5,7,8,10,12,13,13] # sorted

# output = [1,1,2,5,6,7,8,8,10,12,13,13,19,22]
```

$a_1 \rightarrow$ ~~1~~, 2, 6, 8, 19, 22 $\longrightarrow$ already sorted

$a_2 \rightarrow$ 1, 5, 7, 8, 10, 12, 13, 13

[1, 1

$a_1[0] <= a_1[1]$

$a_2[0] <= a_2[1]$

final o/p $\rightarrow$ array to be sorted

$C[0] \longrightarrow$ min element from both arrays

$C[1]$

$a \rightarrow [a_1 \; a_2 \; a_3 - - - - -]$

$b \rightarrow [b_1 \; b_2 \; b_3 - - - - -]$

$\rightarrow$ sorted

$c[0] \rightarrow \underline{\underline{\min(a_1, b_1)}} \rightarrow a_1$

→ 2 pointer approach

```
a1 = [1,2,6,8,19,22] # sorted
a2 = [1,5,7,8,10,12,13,13] # sorted

# output = [1,1,2,5,6,7,8,8,10,12,13,13,19,22]
```

at max every element is chosen once

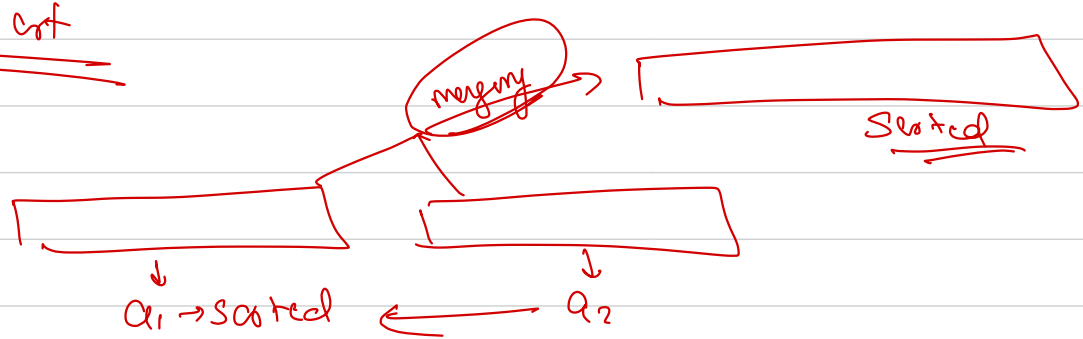
Total comparison → n + m

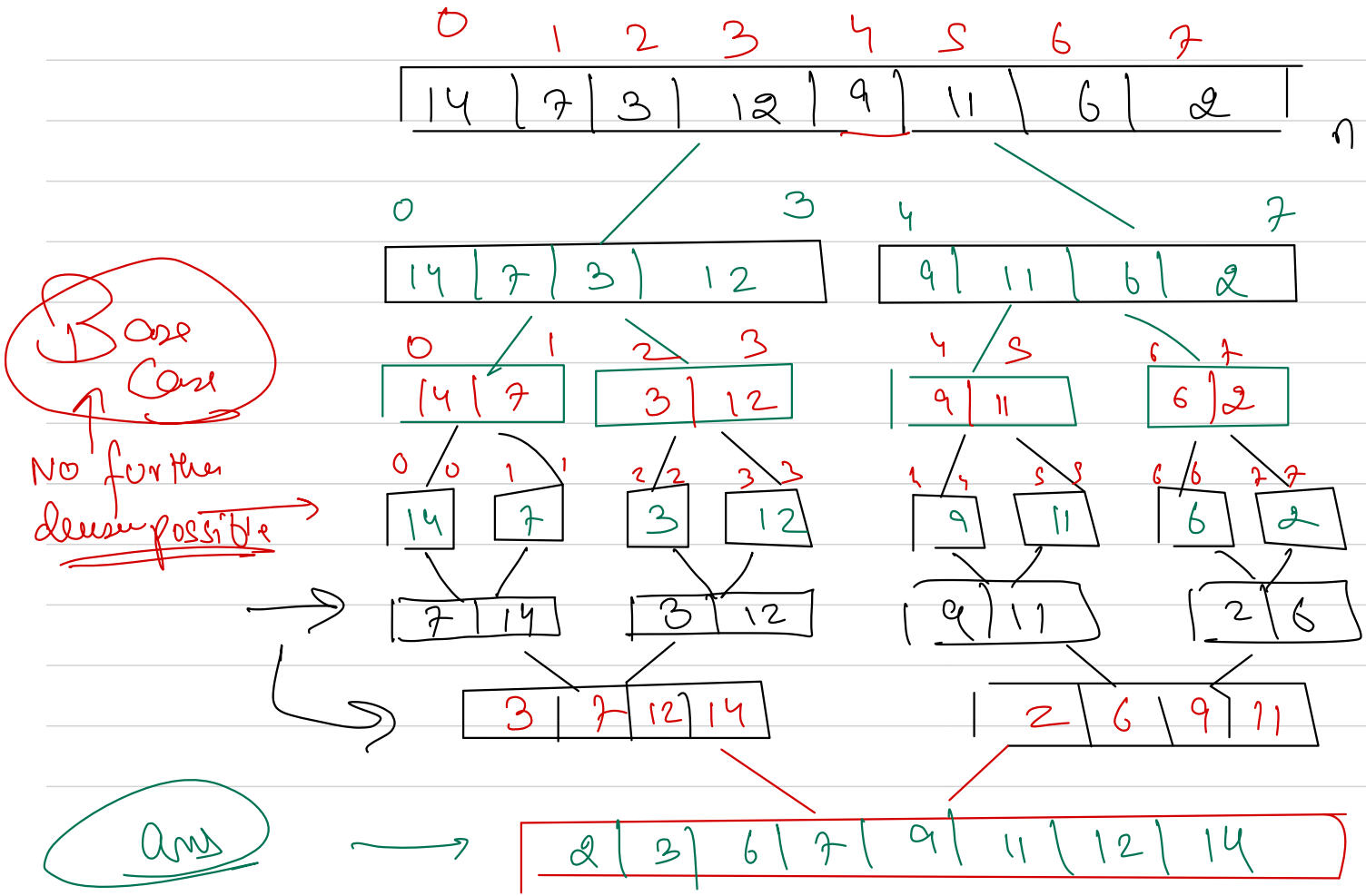
→ TC → O(n+m)

SC → O(n+m)

[1, 1, 2, 5, 6, 7, 8, 8, 10, 12, 13, 13, 19, 22]

Merge Sort




0 1 2 3 4 5 6 7
14 7 3 12 9 11 6 2
n
0 3 4 7
14 7 3 12
9 11 6 2
0 1 2 3 4 5 6 7
14 7
3 12
9 11
6 2
0 0 1 1 2 2 3 3 4 4 5 5 6 6 7 7
14
7
3
12
9
11
6
2
7 14
3 12
9 11
2 6
3 7 12 14
2 6 9 11
Base Case
No further decomposition possible
Ans
2 3 6 7 9 11 12 14

1. Divide your unsorted array into 2 halves
2. then recursively merge sort the left half & recursively merge sort the right half
3. Now u have 2 sorted arrays merge them.

$$f(a, i, j) = \text{merge}\big(f(a, i, mid),\ f(a, mid+1, j)\big)$$

func^n that sorts $a[i..j]$ <u>perfectly</u>

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

$T(n)$ → func^n that represents no of steps reqd to mergesort an array of size n.

$T(n/2)$ → steps reqd for left half with mgesort

$T(n/2)$ → steps reqd for right half using mergesort

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

general eq^n of TC
DnC algo

$$T(n) = a\,T\left(\frac{n}{b}\right) + O(n^d \log^p n)$$

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

using Master theorem

$a = 2 \quad b = 2 \quad d = 1 \quad p = 0$

$a = b^d$

$\rightarrow O(n \times \log n)$

# level	# problems	# no of problem	# time taken
0	$\boxed{n}$	1	$O(n)$
1	$\boxed{n/2}$ $\boxed{n/2}$	2	$2\,O(\frac{n}{2})$
2	$\boxed{n/4}$ $\boxed{n/4}$ $\boxed{n/4}$ $\boxed{n/4}$	4	$4 \times O(\frac{n}{4})$
$\vdots$			
i	$\boxed{n/i}$	2^i	$\rightarrow 2^i \times O\left(\frac{n}{2^i}\right)$

Time $\rightarrow \sum_{i=0}^{\log n} 2^i\, O(n/2^i)$

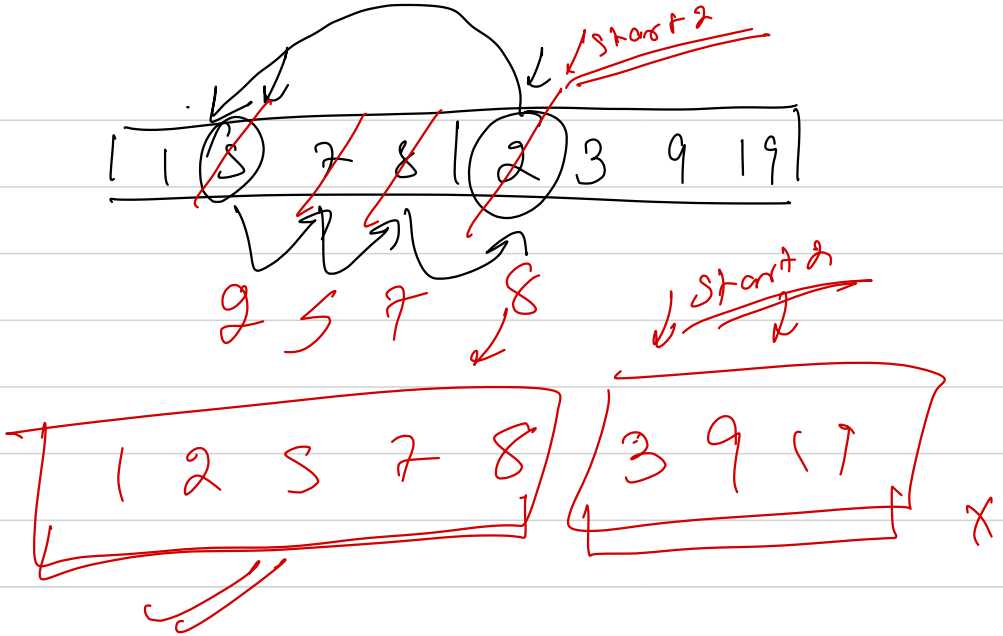
$$\sum_{i=0}^{\log n} \cancel{2^i} \times O\left(\frac{n}{\cancel{2^i}}\right)$$

$$\sum_{i=0}^{\log n} O(n)$$

$$O(n)(1+\log n)$$

$$\rightarrow \underline{\underline{O(n\log n)}}$$

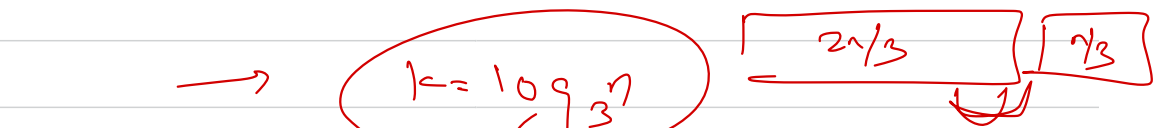
$$n\log_2 n$$


start+2
1 1 5 7 8 1 2 3 9 11
2 5 7 8
start+2
1 2 5 7 8
3 9 11

$$n \longrightarrow \frac{n}{3} \longrightarrow \frac{n}{3^2} \text{ - - - - - - } \frac{n}{3^k}$$

$$\frac{2n}{3} + n$$

$$\frac{n}{3^k} = 1 \longrightarrow k = \log_3 n$$



$$\log_3 n < \log_2 n$$

$$\frac{5n}{3}$$

$$\xrightarrow{n \log_3 n}$$

$$T(n) = 3T\left(\frac{n}{3}\right) + \frac{5n}{3}$$

$$= \cancel{3^i} \times \frac{5}{3} O\left(\frac{n}{\cancel{3^i}}\right)$$

$$\frac{5}{3} \sum_{i=0}^{\log_3 n} O(n)$$

$$\frac{5}{3} n \times \log_3 n \longrightarrow O(n \log n)$$

$$\frac{5}{3} n \cancel{\log_3 n} \approx \cancel{n \log_2 n}$$

$$\frac{5}{3} \log_3 n \longleftrightarrow \log_2 n$$

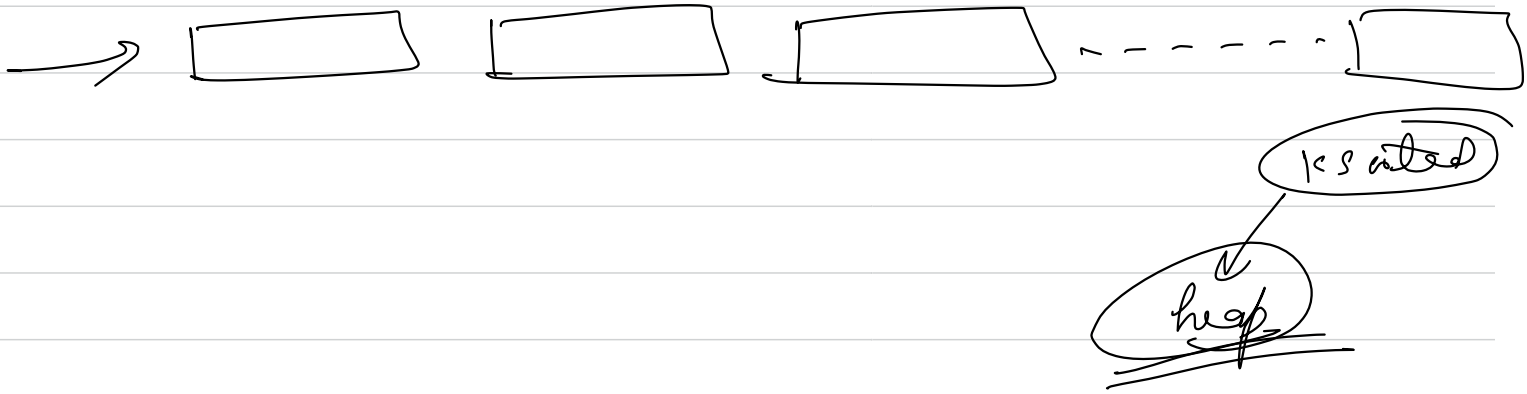
$$\frac{5}{3}\log_3 n \longleftrightarrow \log_2 n$$

$$\boxed{\frac{5}{3} \times \frac{\log_2 n}{\log_2 3}} \longleftrightarrow \log_2 n$$

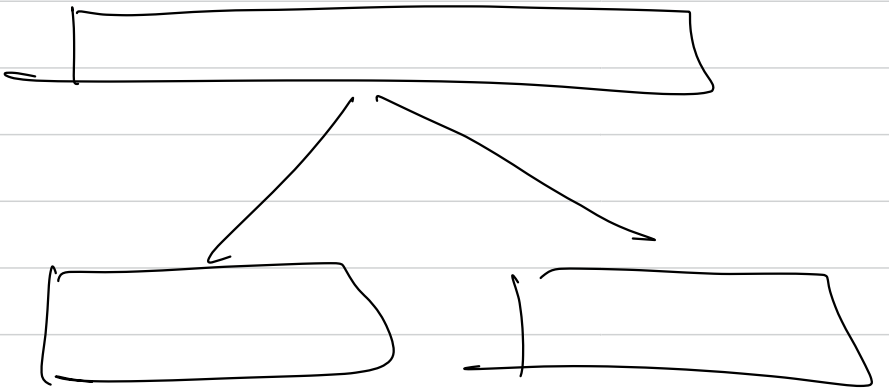
$c > 1$

$$c \times \log_2 n \longleftrightarrow \log_2 n$$

LHS > RHS


k sorted
heap

DNC



algorithms

- Brute force
- Greedy
- Dynamic prog
- Divide N Conquer