Merge Sort

→ Comparison based sorting algo.

→ DNC

→ Sorting is defined as the process by which we can arrange the given list of objects in a certain permutation

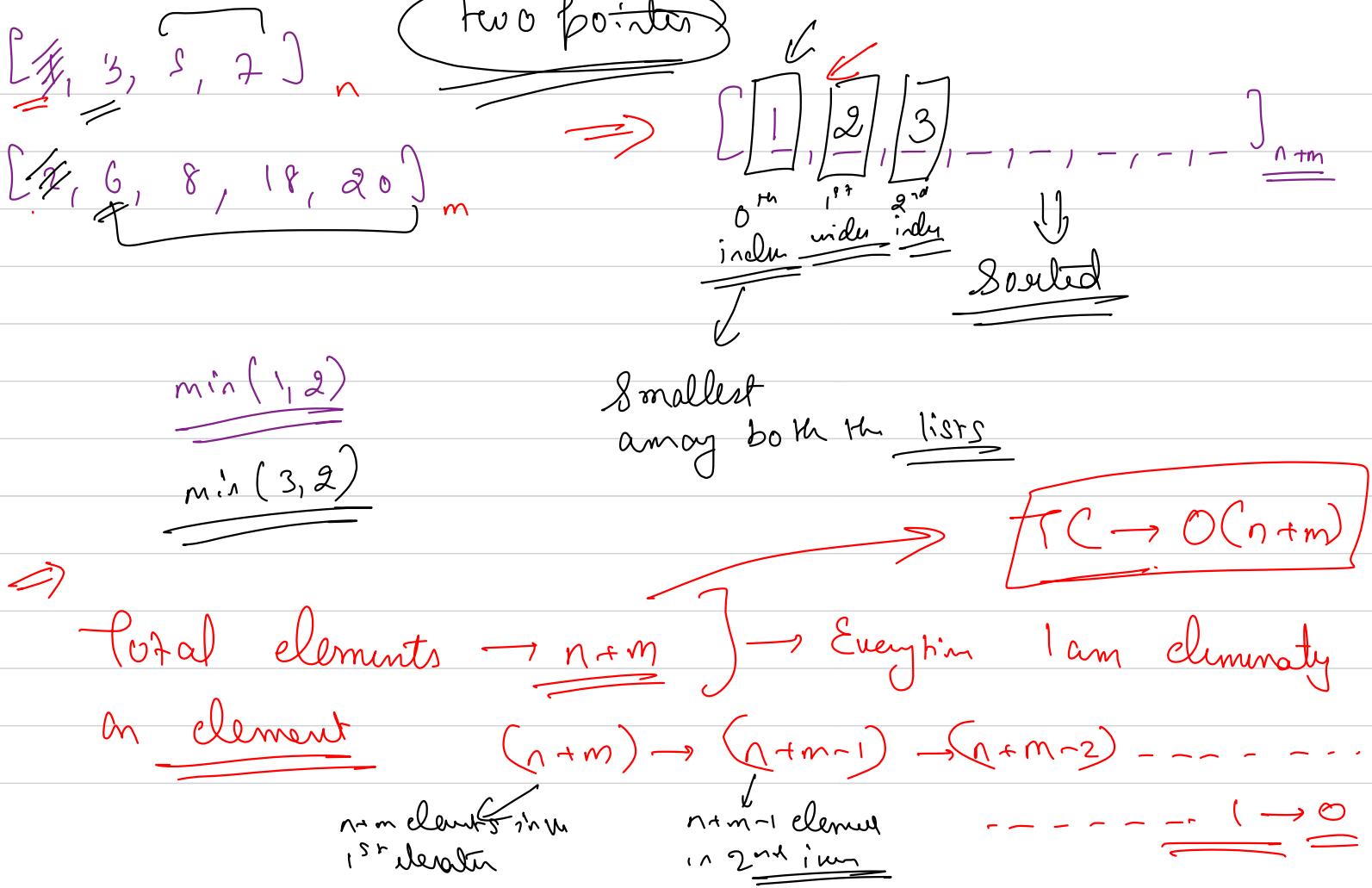
→ Merge Sort is dependent on a Subroutine →

Given two sorted arrays, merge them to give a new sorted array.

↳ [1, 3, 5, 7] → n = 4

[2, 6, 8, 18, 20] → n = 5

ans → [1, 2, 3, 5, 6, 7, 8, 18, 20]


[1, 3, 5, 7] n
[2, 6, 8, 18, 20] m
two pointers
[1, 2, 3, ...,] n+m
0th index
1st index
2nd index
Sorted
Smallest among both the lists
min(1, 2)
min(3, 2)
TC → O(n+m)
Total elements → n+m → Everytime I am eliminating an element
(n+m) → (n+m-1) → (n+m-2) ... 1 → 0
n+m elements in the 1st iteration
n+m-1 element in 2nd iteration

[1, 3, 5, 7]

[2, 6, 8, 18, 20]

i

j

2

2^k

[1 2 3 _ _ _ _ _ _]

1+m

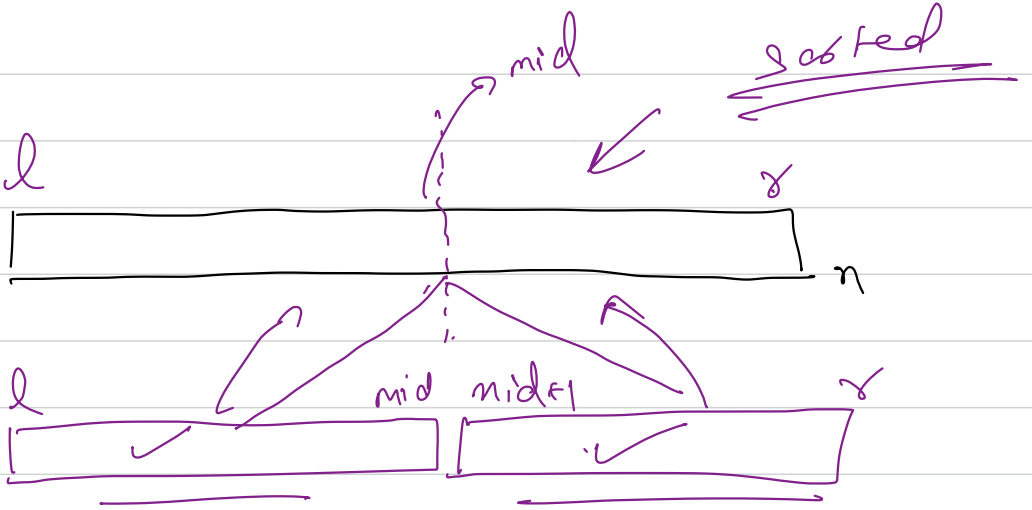
i = ~~0~~ ~~1~~ 2

j = ~~0~~ 1

k = ~~0~~ ~~1~~ ~~2~~ 3

min(i, j)

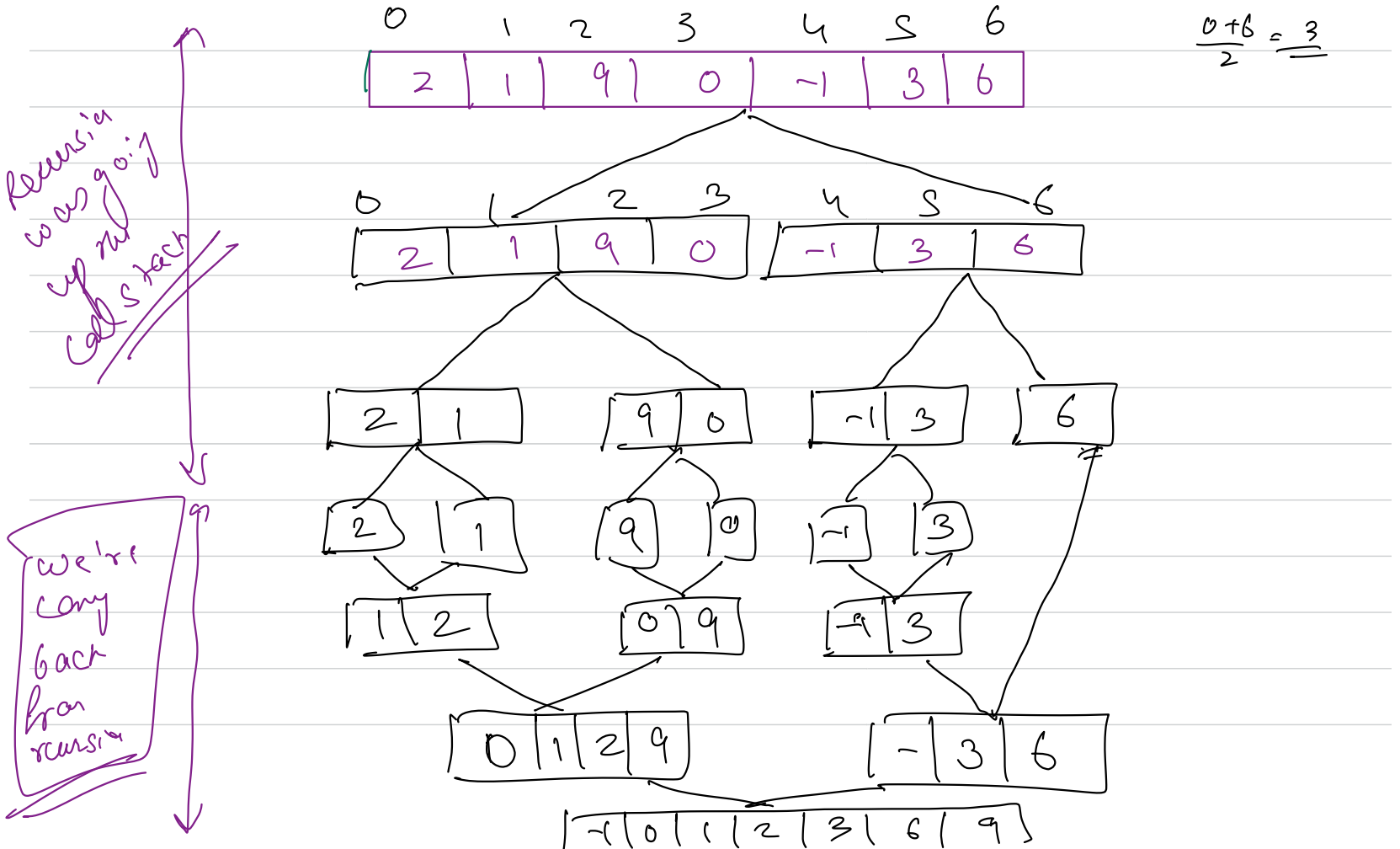
Merge Sort



→ Base Case

→ Self work

→ Recursive Intuition


0 1 2 3 4 5 6
2 1 9 0 -1 3 6
0+6 / 2 = 3
Recursion is going up the call stack
We're coming back from recursion
0 1 2 3 4 5 6
2 1 9 0 -1 3 6
2 1
9 0
-1 3
6
2
1
9
0
-1
3
1 2
0 9
-1 3
0 1 2 9
-1 3 6
-1 0 1 2 3 6 9

TC

no. of operation for merge sort on left half

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

$T(n)$: func^{n} which calculates no. of operations done to merge sort a list of size n

$T\left(\frac{n}{2}\right)$: no of operation on merge sort of right half

$O(n)$: + merge the left & right half

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

$T(n) = 2T\left(\frac{n}{2}\right) + O(n)$ — $2 \times \frac{\log n}{2}$ $2n$

$2T\left(\frac{n}{2}\right) = 4T\left(\frac{n}{4}\right) + O(n)$ — $\times 2$

$4T\left(\frac{n}{4}\right) = 8T\left(\frac{n}{8}\right) + O(n)$ — $\times 4$ — k levels

$2^k T(2) = 2^{k+1} T(1) + O(n)$

$T(n) = 2^{k+1} \times 1 + k \times O(n) \rightarrow 2^{k+1} + \log_2 n \times O(n)$

$\Rightarrow 2n + n \log n$

$n \longrightarrow \frac{n}{2} \longrightarrow \frac{n}{4} \longrightarrow \frac{n}{8} \cdots\cdots\cdots \frac{n}{2^k}$

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

$k = \log_2 n$

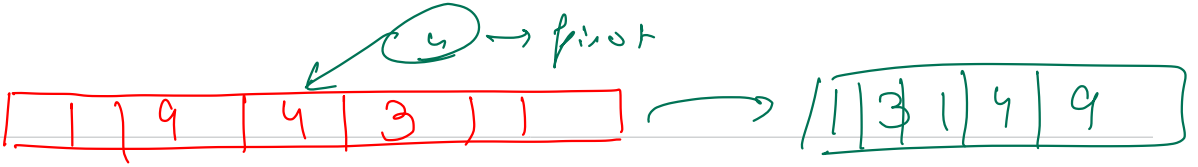
$T(n) = O(n \log n)$

SANKETIO

Quick Sort → Comparison based sort

→ DNC

→ subroutine → partition also

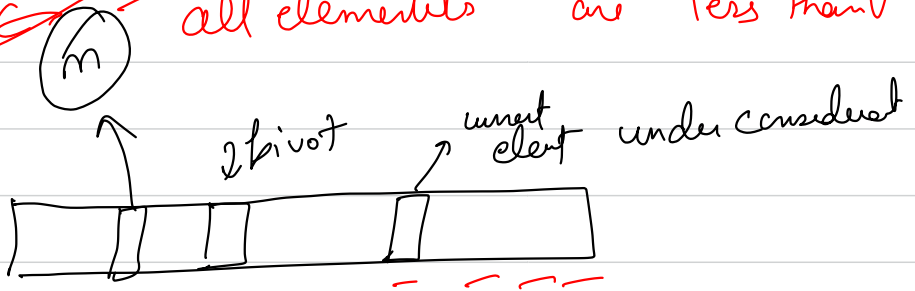


Given an array of elements, and a pivot element, rearrange the elements of the original array such that all the elements <= pivot are on the left side & greater than pivot are on right side.

Two pointer approach

Boundary elem

m denotes the index before which all elements are less than or equal to pivot



n = 0

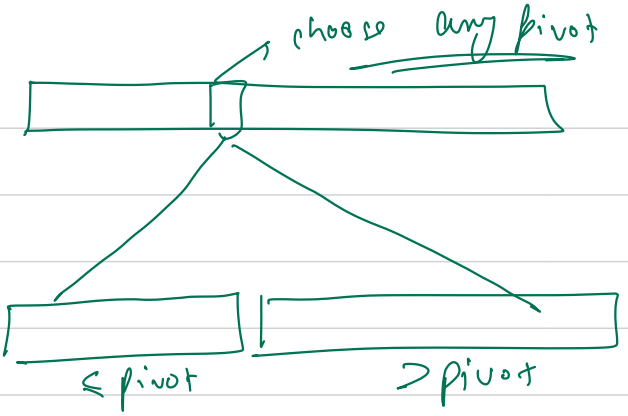
```
if current element ≤ pivot
    swap (arr(m), current, elem)
    m++
```

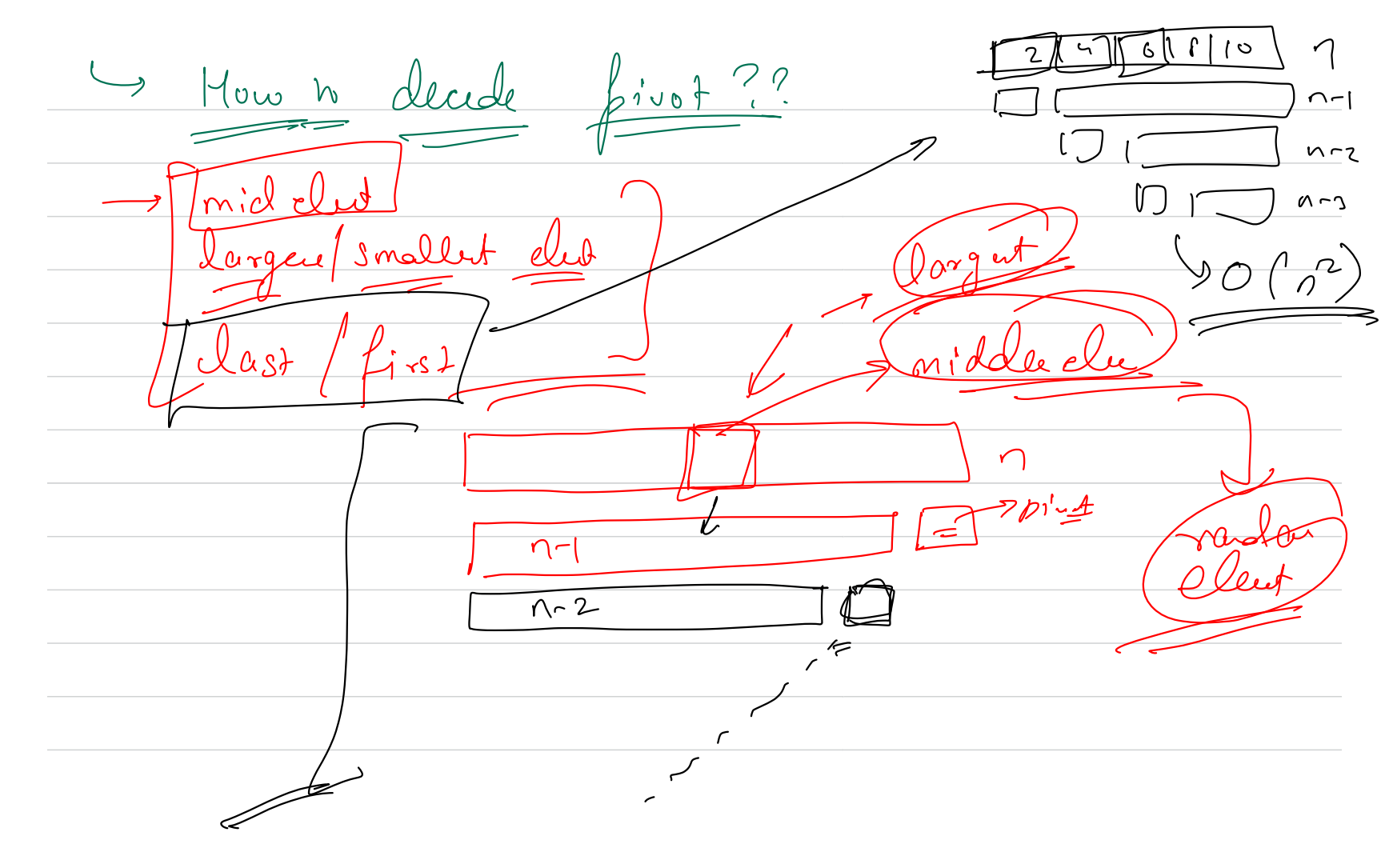
[7, 5, 1, 8, 10, 9]

m, i

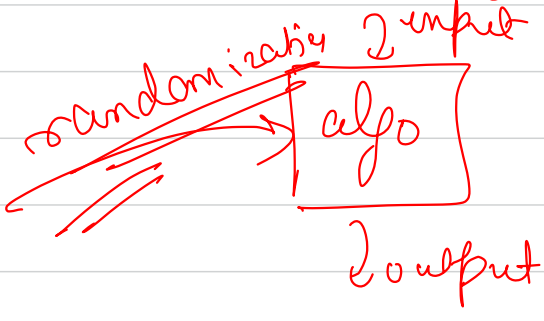
pivot = 8

8 ≤ 8


choose any pivot
≤ pivot
> pivot


How to decide pivot ??
mid elmt
largest/smallest elmt
last / first
2 4 6 8 10
n
n-1
n-2
n-3
O(n²)
largest
middle elmt
random elmt
n
n-1
n-2
= pivot

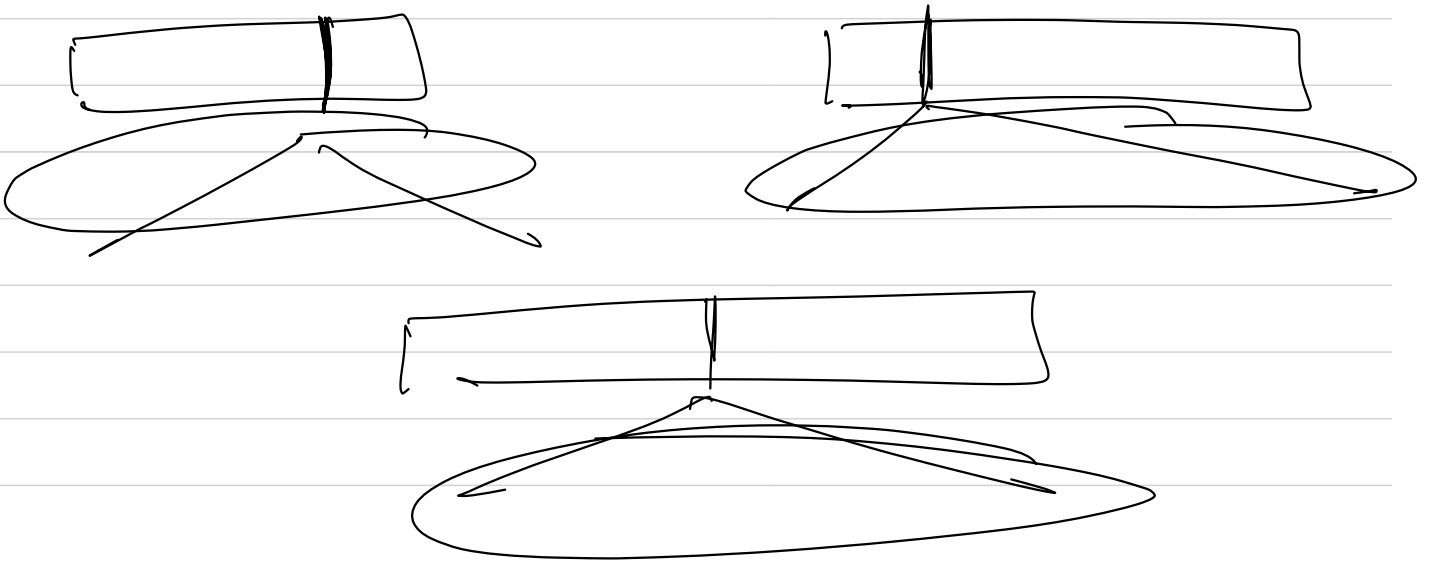
Randomized algo



→ generally in quick sort we choose random pivot

→ Our claim is that we should pick elements randomly as the pivot.

→

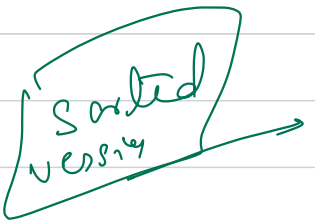


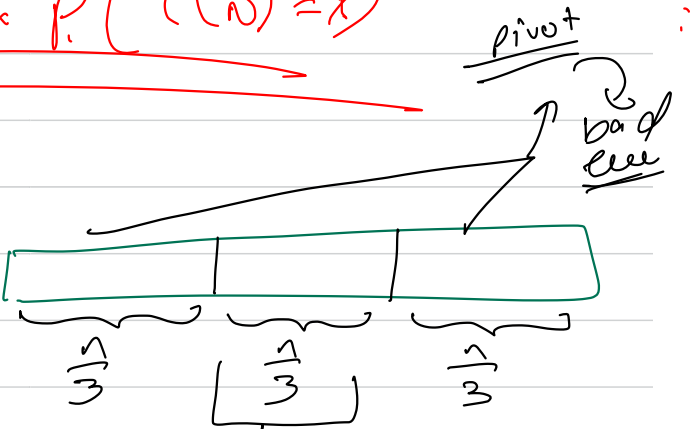
→ If the pivot is chosen time complexity can also change.

$\alpha \rightarrow$ avg estimated no.

$$\Rightarrow \alpha(T(n)) = \sum x \times P(T(n) = x)$$

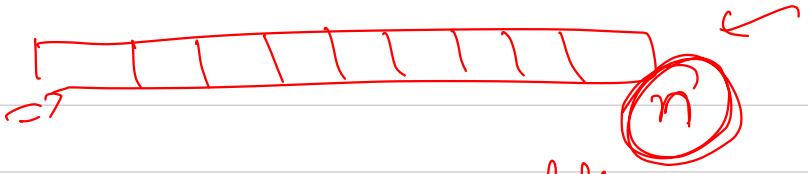
It is a function that gives you no. of operations to apply quicksort on array of size n





If my pivot comes from here → good case

tightest upper bound



$$\alpha(\tau(n)) = \sum_{k=0}^{n-1} \left(\alpha(\tau(k)) + \alpha(\tau(n-k)) + n \right) \times \frac{1}{n}$$

left · right · partition

$$\tau(n) \leq n + \frac{1}{2} \times \left(\tau\left(\frac{n}{3}\right) + \tau\left(\frac{2n}{3}\right) \right) + \frac{2}{3}\left(\tau(n)\right)$$

multiply 3 on both sides

$$\frac{1}{3}\tau(n) \leq n + \frac{1}{3}\left(\tau\left(\frac{n}{3}\right) + \tau\left(\frac{2n}{3}\right) \right)$$

$$T(n) \leq 3n + T\left(\frac{n}{3}\right) + T\left(\frac{2n}{3}\right)$$

assume $T(n) \leq cn\log n$

$$T(n) \leq 3n + c\frac{n}{3}\log\frac{n}{3} + c\frac{2n}{3}\log\frac{2n}{3}$$

$$T(n) \leq n\left(3 + \frac{c}{3}\log\frac{n}{3} + \frac{2c}{3}\log\frac{2n}{3}\right)$$

$$T(n) \leq n\left(3 + \frac{c}{3}\log n - \frac{c}{3}\log 3 + \frac{2c}{3}\log n - \frac{2c}{3}\log\frac{3}{2}\right)$$

$$T(n) \leq n\left(3 + \log n - \frac{c}{3}\log 3 - \frac{2c}{3}\log\frac{3}{2}\right)$$

$$T(n) \leq 3n + n\log n - \frac{cn}{3}\log 3 - \frac{2cn}{3}\log\frac{3}{2}$$

$$T(n) = \Theta(n\log n) \longrightarrow \text{Avg Case}$$

$$T(n) = 2T\left(\frac{n}{2}\right) + O(n) \Longrightarrow \Omega(n\log n)$$

Best Case

$$T(n) = T(n-1) + T(1) + O(n) \longrightarrow O(n^2)$$

Space Complexity $\longrightarrow O(\log n) \longrightarrow$ due to recursion

sanket.singh__