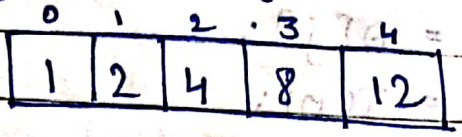
Quick Sort Analysis

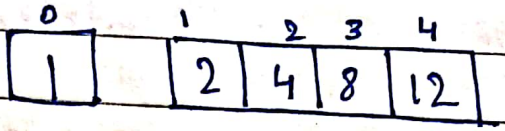
1) Time Complexity:

→ WORST-CASE: The worst-case in a quicksort algorithm happens when our array is already sorted.



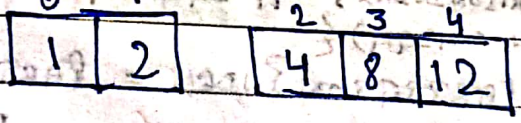
PARTITION → 1

US



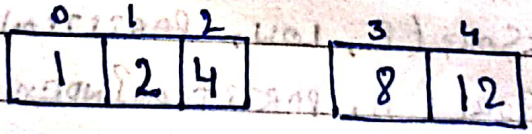
" → 2

US

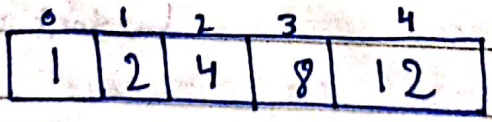


" → 3

US



" → 4

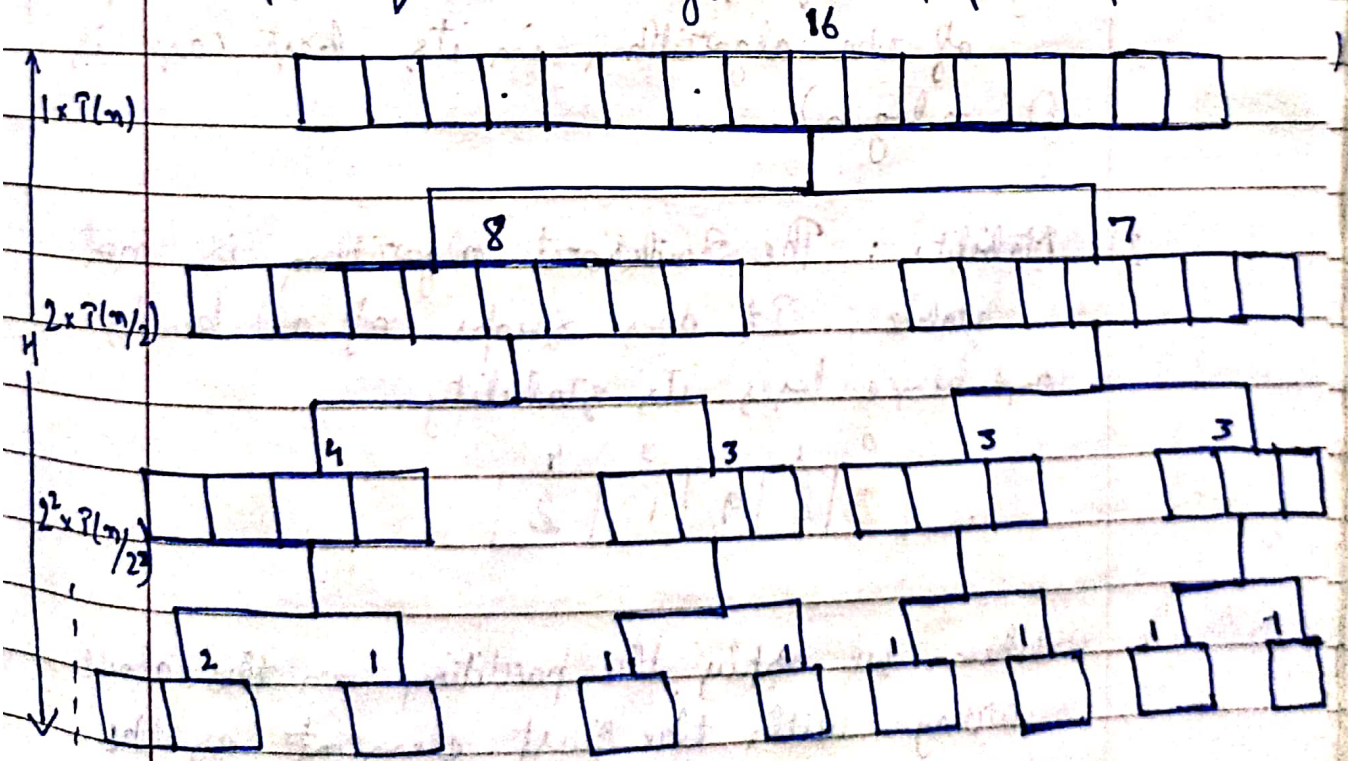


SORTED

So, if we calculate, for an array of size 5, we had to partition the subarray 4 times. That is, for an array of size n, there would be (n-1) partitions. Now, during each partitions, we made our two index variables, i & j and run from either direction towards each other

until they actually become equal or cross each other. And we do some swapping in b/w as well. These operations count to some linear function of n, contributing O(n) to the runtime complexity.
And since there are a total of (n-1) partitions, our total runtime complexity becomes n(n-1) which is simply $O(n^2)$. This is our ~~Hard~~ worst-case complexity.

→ BEST-CASE: The condition when our algorithm performs in its best possible time complexity is when our array gets divided into two almost equal subarrays at each partition.



Here, T(x) is the time taken during the partition of the array with x elements. And as we know, the partition takes

a linear function time, and we can assume $T(x)$ to be equal to x; hence the total time complexity becomes,

Total time = $1(n) + 2(n/2) + 4(n/4) + \ldots\ldots$
$\ldots\ldots$ + until the height of the tree (h)

$\therefore$ Total time = $n \times h$

→ Now H is the height of the tree, and the height of the tree, if n is $\log_2(n)$, where n is the size of the given array. In the above example, $h = 4$, since $\log_2(16)$ equals 4. Hence the time complexity of the algorithm in its best case is $O(n \log n)$.

2) Stability : The Quicksort algorithm is not stable. It does swaps of all kinds and hence loses its stability.

0	1	2	3	4
2	8	9	12	2

When we apply the partition on the above array with the first element as the pivot, our array becomes.

0	1	2	3	4
2	2	9	12	8

Page No.
Date

And the two 2s get their order reversed.
Hence quick sort is not stable.

3) Quicksort algorithm is an in-place algorithm. It doesn't consume any extra space in the memory. It does all kinds of operations in the same array itself.

4) There is no hard and fast rule to choose only the first element as the pivot; rather, you can have any random element as its pivot using the rand() function and that you shouldn't believe actually reduces the algorithm's complexity.