Quick SORT

i) Time Complexity :-

A $\boxed{\text{'n'}}$

$O(n)$

I) $\langle n, a_1, a_2, \ldots a_n \rangle$

f partition : $O(n)$

$I_1 \langle n, (\overset{L}{\tfrac{n}{2}}) \boxed{a_1} (\overset{R}{\tfrac{n}{2}}) \rangle$

Case-1

$I_2 \langle (n-1) \boxed{a_1} \rangle$ Dec.

Inc. $I_3 \langle \boxed{a_1} (n-1) \rangle$

Smiliear. Case-2

$\langle n\, (n/2) \boxed{a_1} (n/2) \rangle$

2nd pass $(\overset{L}{\ }) \boxed{\cdot} (\overset{R}{\ }) \boxed{a_1} (\overset{L}{\ }) \boxed{\cdot} (\overset{R}{\ })$

3 Element fix.

Best case

$I_1 \langle (n-1) \boxed{a_1} \rangle$

$(n-2) \boxed{\cdot} \boxed{a_1}$

2 Element fix.

wrost case.

Note :- Quick sort Behave worst case, when elements are sorted.

Quick Best ⇒ unsorted.
Quick wrost ⇒ sorted.

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

$= \boxed{\cancel{O(n)}} \boxed{O(n\log n)}$

Time complexity
Best case. $\boxed{O(n\log n)}$

- unsorted List.

Sorted List

$T(n) = O(n) + T(n-1)$

$= \boxed{O(n^2)}$

derivation

$T(n) = O(n) + T(n-1)$

$T(n-1) = (n-1) + (n-2)$

$= (n-1) + (n-2) + n$

= Arithmetic Prog.

$= \frac{n(n-1)}{2} = \frac{n^2-n}{2} = n^2$

Space complexity :-

① Best case :-

↳ You Need Recursion Tree-Stack. = height of Tree = Logn = $\boxed{O(\log n)}$

② Wrost case :-

n element traversing.

Stack = $\boxed{O(n)}$