Quick Sort

↳ Worst case Time complexity: Already sorted array is worst case input.

→ Pivot is maximum or minimum
- one partition is empty
- Other is size n-1

→ n-1 pass required to sort n data.

example → I/P: 2 4 8 10 12 24 30

Pivot element = 2

Pass 1: ⌊2⌋ | 4 8 10 12 24 30 |, No. of comparision = 6

Pivot element = 4

Pass 2: ⌊2⌋ ⌊4⌋ | 8 10 12 24 30 |, No. of comparision = 5

Pivot element = 8

Pass 3: ⌊2⌋ ⌊4⌋ ⌊8⌋ | 10 12 24 30 |, No. of comparision = 4

Pass 4: ⌊2⌋ ⌊4⌋ ⌊8⌋ ⌊10⌋ | 12 24 30 |, No. of Comparision = 3

Pass 5: ⌊2⌋ ⌊4⌋ ⌊8⌋ ⌊10⌋ ⌊12⌋ | 24 30 |, No. of comparision = 2

Pass 6: ⌊2⌋ ⌊4⌋ ⌊8⌋ ⌊10⌋ ⌊12⌋ ⌊24⌋ | 30 |, No. of Comparision = 1

Output: 2 4 8 10 12 24 30

In the above example

No. of data = 7

Required passes = 6

Total No. of Comparision = 6 + 5 + 4 + 3 + 2 + 1

Worst case time complexity

if No. of data = n.

Pass 1 : n-1 comparisions

Pass 2 : n-2 "

Pass 3 : n-3 "

Pass 4 : n-4 "

- - - - - - - - - -

Pass n-2 - - - 2 - comparision.

Pass n-1 : 1 comparision

Total No. of comparisions in all passes = $n-1+n-2+n-3+\cdots+2+1$

or

$1+2+3+\cdots+n-3+n-2+n-1$

As we know that,

Sum of first n-natural number :

$$1+2+3+\cdots+n-2+n-1+n = \frac{n(n+1)}{2} \longrightarrow (1)$$

we just put n-1 instead of n in equation 1 to calculate the Sum $1+2+3+\cdots+n-3+n-2+n-1$

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

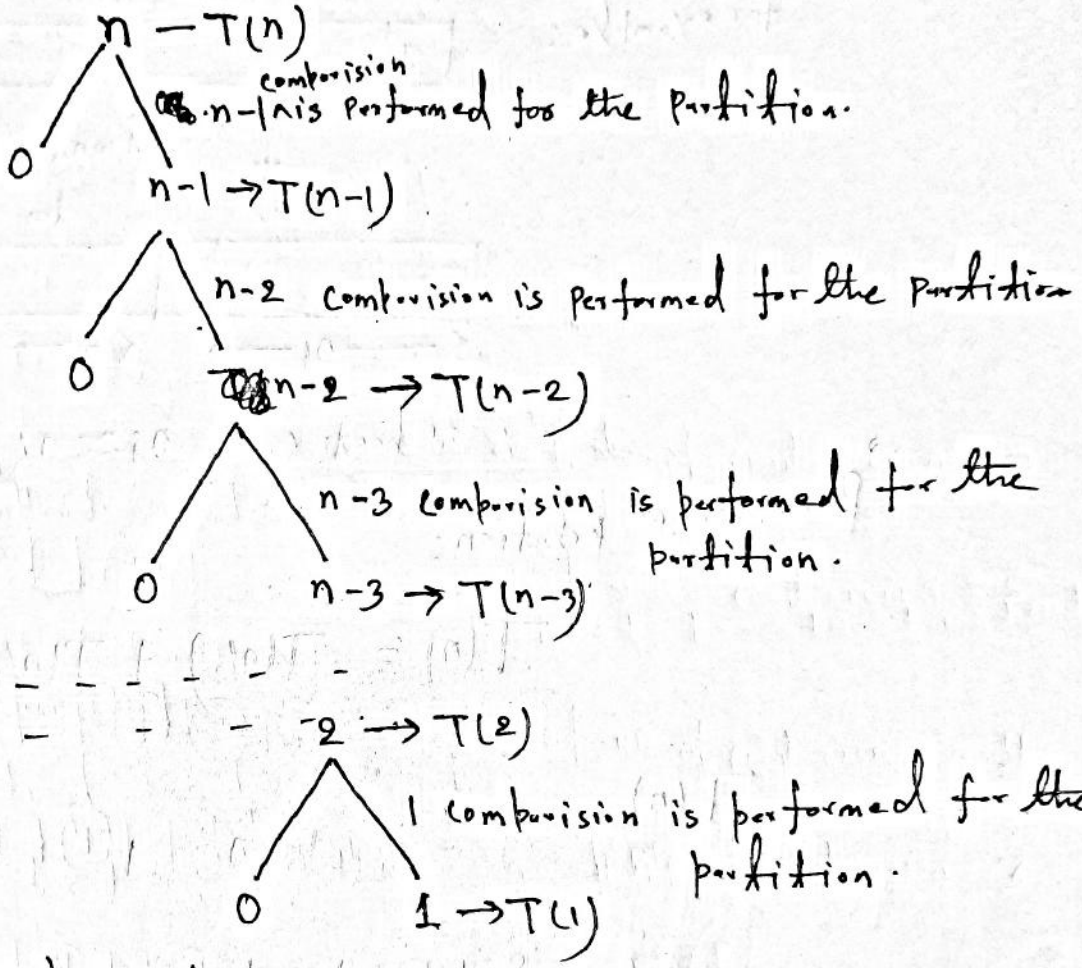
$$= \frac{n^2}{2} - \frac{n}{2}$$

$= O(n^2)$, where n is too large.

$n \longrightarrow \infty$.

Recurrence Relation:

$$T(n) = \begin{cases} 1, & \text{if } n = 1 \\ T(n-1) + \Theta(n), & n > 1 \end{cases}$$



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

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

$$= T(n-3) + n-1 + n-2 + n-3$$

$$= T(1) + n-1 + n-2 + n-3 + \cdots + 1$$

$T(1) = 1$.

So, $T(n) = n-1 + n-2 + n-3 + \cdots + 1$

or $1 + 2 + 3 + \cdots + n-3 + n-2 + n-1 = \frac{n(n-1)}{2}$

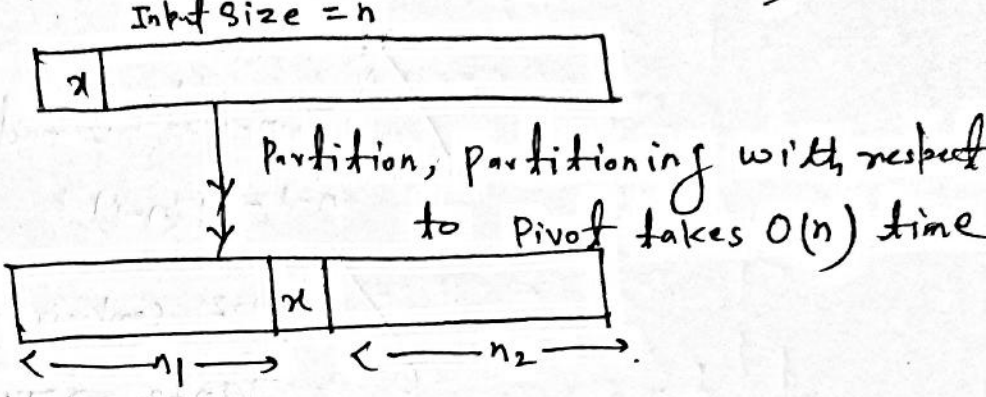
$= O(n^2)$

↳ Best case example →

→ If Pivot is median

* Each partition is of size $n/2$ (almost $n/2$).

for example:



So, for best case $n_1 \simeq n/2 \simeq n_2$

Recurrence Relation:

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

$$T(n) = \begin{cases} 1, & \text{if } n = 1 \\ 2T(n/2) + O(n), & n > 1 \end{cases}$$

$$\boxed{T(n) = O(n \log_2 n)}$$

* In base case $\log_2 n$ no. of passes required to sort n data.

Time Complexity = $O(n \log_2 n)$.