QUICK SORT :

$A[0] \ldots A[s-1]$ $A[s]$ $A[s+1] \ldots A[n-1]$

all are $\leq A[s]$. all are $> A[s]$

eg:- 2, 1, 3, 4, 5 Sort using quick sort.

Choose 1 elt as pivot elt.
A/c to algorithm we choose the 1st elt as pivot elt.

Pivot elt = 2.

→ 2 1 3 4 5
i↑ j↑

i ↑ increments
j ↓ decrements.

i - pointing to 1st elt
j - pointing to last elt.

Any comparision is made with PIVOT.

→ ① st comparision - 2 = 2

If elt pointed by i ~~compared~~ is less than or equal to pivot → increment i.

2 = 2 so $i \leftarrow i+1$

→ ② nd - 2 1 3 4 5
i↑ ↑j

$i \leftarrow i+1$

Pivot
2 > 1

→ ③ rd comparision - 2 1 3 4 5
↑i ↑j

3 > Pivot
2

If the elt pointed by j is greater than pivot → decrement j.

→ ④ th comparision 2 1 3 4 5
↑i ↑j

5 > 2

→ ⑤th comparision 4 > 2   2 1 3 4 5
↑i ↑j

→ ⑥th comparision -

3 > 2   2 1 3 4 5
↑j ↑i

→ ⑦th comparision -

2 = 2   2 1 3 4 5
↑j ↑i

Compare positions of i & j
If i<j
Swap a[i] & a[j]
else
Swap a[j] & pivot
So, swap 1 & pivot.

1 2 3 4 5.

1 subdivision   ↑ pivot   another subdivision.

eg:- 12, 14, 13, 9, 5, 4

12 14 13 9 5 4
↑i   ↑j

→ 12 ≤ 12
i ← i+1

12 14 13 9 5 4
↑i   ↑j

→ 14 > 12   Compare a[j] & pivot.
~~decrement j~~.   4 < 12.

So, No decrement j.

Now compare i & j positions.

Since, i < j

a[i] and a[j] are swapped.

12 4 13 9 5 14.

↑i ↑j

→ 4 < 12

increment i.

12 4 13 9 5 14

↑i ↑j

→ 13 > 12

Stop i. Look for j.

14 > 12.

12 4 13 9 5 14

↑i ↑j.

→ 5 < 12.

Stop decrementing j.

Compare i & j positions.

i < j

Swap a[i] & a[j]

12 4 5 9 13 14

↑i ↑j

→ 5 < 12. increment i.

12 4 5 9 13 14

↑i ↑j

→ 9 < 12 increment i

12 4 5 9 13 14

↑i↑j.

→ 13 > 12 look for j.

13 > 12. decrement j.

12 4 5 9 13 14

↑j ↑i

→ 9 < 12 stop.

Compare i & j positions.

j < i → so swap a[j] & pivot.

~~9 12 4 5~~ 9 4 5 12 13 14

pivot ↑ ↑j ↑i.

When pivot is swapped,

the left array becomes 1 subproblem

& the right array becomes 1 supproblem.

$\underbrace{9\ 4\ 5}_{①}$ [12] $\underbrace{13\ 14}_{②}$

Solve ① & ② by Quick sort.

Algorithm Quicksort (A[l -- r])

Sorts a subarray by quicksort.

input - Subarray of array A[0 --- n-1], defined by its left and right indices l and r.

output - Subarray A[l -- r] sorted in nondecrea-sing order.

if l < r

  s ← Partition(A[l -- r]) // s is a split position

  Quicksort(A[l -- s-1])

  Quicksort(A[s+1 --- r]).

→ Partition(A[l -- r])

  pivot ← A[l]

  i ← l

  j ← r+1

  repeat.

pivot = 3

3	2	1	4	5
↑i				↑j.

~~while A[i] ≤ pivot & i < j~~

~~i ← i+1~~

repeat i ← i+1 until i >= j or A[i] >= pivot.

repeat j ← j-1 until A[j] <= pivot

if i < j

  Swap a[i] & a[j]

until i >= j.

  swap a[j] & pivot

return j.

8/9/21 : 3 4 5 1 2

3 - pivot

3 4 5 1 2
↑i ↑j.

→ 3 ≤ 3   i → i+1 - 3 4 5 1 2
↑i ↑j

4 ≯ 3 × stop i → verify j.

2 < 3 → stop

Compare i & j

i < j.

→ 3 2 5 1 4
↑i ↑j

→ 2 < 3   i = i+1

→ 5 > 3   stop i.

4 > 3   j = j-1   3 2 5 1 4
↑i ↑j

1 < 3 stop j

Compare i & j

i < j. swap a[i] & a[j]

3 2 1 5 4
↑i ↑j

→ 1 < 3   i = i+1

5 > 3 stop i.

Verify j → 5 > 3   j = j-1   3 2 1 5 4
↑j ↑i

1 < 3 stop j.

Compare i & j

i > j.

swap. a[j] & pivot

$\underbrace{1\ 2}$ [3] $\underbrace{5\ 4}$

Analysis-

Basic Operation- Comparision

(These occur in partition)

Best Case- Sets are partitioned into subsets of equal size.

$\Rightarrow T(n) = a \cdot T(n/b) + f(n)$

No of comparisions = No. of elts (n) + const.

atleast 'n' comparisions.

So,

$f(n) \in \theta(n^d)$

$f(n) = \theta(n)$

$c(n) = 2 \cdot c(n/2) + f(n)$

$$\boxed{c(n) = 2 \cdot c(n/2) + n} \quad n > 1$$

Applying master's theorem,

$a = 2, \; b = 2, \; d = 1$

$a = b^d \quad 2 = 2^1$

$\theta(n^d \log_b n)$

$= \theta(n^1 \log_2 n)$

$= \theta(n \log n)$

$$\boxed{C\_best(n) = \theta(n \log_2 n)}$$

$$\boxed{C\_best(1) = 1}$$

eg:- 3 4 5 1 2

① 4 < 3 stop i

② 2 < 3 stop j

∴ Compare i & j

i < j

3 2 5 1 4
↑i ↑j

③ 2 < 3 i = i+1

④ 5 > 3 stop i

⑤ 4 > 3 j = j−1

⑥ 1 < 3 stop j

Compare i & j

i < j

3 2 1 5 4
↑j ↑i

⑦ 1 < 3 i = i+1

⑧ 5 > 3 stop i

⑨ 1 > 3 j = j−1

i > j

swap a[j] & pivot.

1 2 [3] 5 4.

9 = 5 + 4

Comparisions = (n) + const.

Worst Case - Array is already sorted.

1 2 3 4 5 Pivot = 1

↑i ↑j

→ 1 ≤ 1 i = i+1

→ 2 > 1 stop i

5 > 1 j = j-1

4 > 1 j = j-1

3 > 1 j = j-1

2 > 1 j = j-1

1 > 1 stop j

1 2 3 4 5

↑j ↑i

Compare i & j : i > j

Swap a[j] & pivot.

$[1] \underbrace{2 \quad 3 \quad 4 \quad 5}_{\text{sort it by quicksort.}}$

→ Now pivot is taken as 2 and solved as usual.

$1 \ [2] \ \underbrace{3 \quad 4 \quad 5}_{\text{solve by quicksort.}}$

→ Pivot - 3

$1 \ 2 \ [3] \ \underbrace{4 \quad 5}_{\text{solve by quicksort.}}$

→ Pivot - 4

1 2 3 [4] 5

1. 2 > 1 stop i
2. 5 > 1 j = j-1
3. 4 > 1 j = j-1
4. 3 > 1 j = j-1
5. 2 > 1 j = j-1
6. 1 > 1 x stop j

1 2 3 4 5

↑j ↑i

i > j swap a[j] & pivot.

→ ① 3 > 2 stop i
② 5 > 2 j-j-1
③ 4 > 2 j-j-1
④ 3 > 2 j-j-1
⑤ 2 > 2 × stop j

Compare i & j.

j < i

swap a[j] & pivot.

2 3 4 5
↑j ↑i

Array size - 4
No. of comparisions - 6

So,

When an array of n elements and are sorted the no. of comparisions will be n+1.

1 2 3 4 5 → 6 comparisions.

1 [2 3 4 5] → 5 comparisions

1 2 [3 4 5] → 4 comparisions

1 2 3 [4 5] → 3 comparisions.

1 2 3 4 5 → No comparisions.

Generalising, No. of comparisions -

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

$= \frac{(n+1)(n+2)}{2} - 1 - 2 \quad = 3 + 4 + \cdots + n + (n+1)$

$= \frac{(n+1)(n+2) - 6}{2} \quad = 3 + 4 + \cdots + n + (n+1)$

No. of comparisions $\simeq O(n^2)$

$C\text{-}worst(n) = C\text{-}worst(n-1) + 1 \quad n > 1$

Bcz as the array is sorted u will have only 1 subarray when u swap with pivot.

So, $a = 1$

$$\boxed{\begin{aligned} C\text{-}worst(n) &= (n+1) + (n) + (n-1) + \cdots 3 \\ &= \frac{(n+1)(n+2)}{2} - 3 \simeq \theta(n^2). \end{aligned}}$$

$C\text{-}worst(1) = 0.$

$C\text{-}worst(2) = C\text{-}worst + 3.$