Space Complexity O(1)

Quick sort :-

* Quick sort, as the name suggest, sorts any list very quickly.
* quick sort is not stable, but it is very fast and requires very less additional space.
* It is based on divide and conquer algorithmic strategy.

Choosing the pivot element by following ways

* choose first element as pivot element
* Randomly choose
* choose middle element as pivot element.

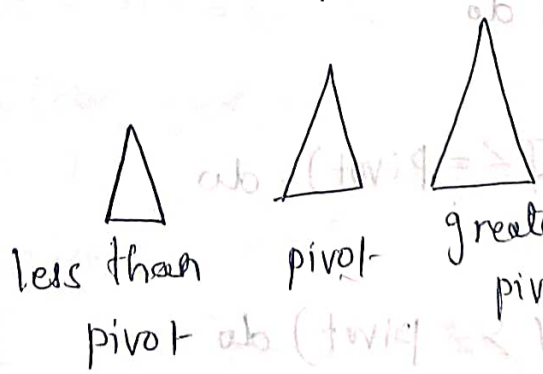
* In divide and conquer method division is dynamically carried out.

3 steps of quick sort are as follows

① Divide

split the array into two sub-arrays that each element in left sub-array is less than or equal to middle element ie) pivot & each element in right sub-array is greater than middle element.

② A splitting of array into sub array is based on pivot element



A[0] .... A[m-1] A[m] A[m+1] ... A[n-1]

Left Array. pivot Right Array

③ Conquer :- Recursively sort the two subarra combine all sorted elements in a group to form a list of sorted elements.

```
Algorithm quick(A[0..n-1], low, high)
{
  if (low < high)
  {
    m = partition(A[low ... high])
    quick(A[low ... m-1])
    quick(A[m+1 ... high]);
  }
}

Algorithm Partition(A[low ... high])
{
  Pivot = A[low];
  i = low;
  j = high;
  while (i <= j) do
  {
    while (A[i] <= pivot) do
    i = i+1;
    while (A[j] >= pivot) do
    j = j-1;
    if (i < j) then
    swap(A[i], A[j]);
  }
```

swap (A[low], A[j]);

return j;

}

Example

[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]
25	52	37	63	14	17	8	6
p, i							j

1) low = 0, high = 7

0 < 7

m = partition (A [0 .... 7])

quick ([0 ....

Partition

pivot = A[low];

pivot = A[0]; = 25

i = low = 0

j = 7

while (0 <= 7)

{

while (A[0] <= 25) True

25 <= 25

i = i+1 = 1

A[1] <= 25

52 <= 25 x

while (A[7] >= 25]

6 >= 25 x

1 <= 7 True

swap (A[1], A[7])

0	1	2	3	4	5	6	7
25	6	37	63	14	17	8	52
	i						j

②

$6 <= 25$

$i = i+1 = 2$

$37 <= 25$ (False)

$52 >= 25$ True

$j = j-1$;

$j = 7-1 = 6$

$8 >= 25$ (False)

25	6	37	63	14	17	8	52
		i					j

Swap (A[i], A[j])

Swap (37, 8)

0	1	2	3	4	5	6	7
25	6	8	63	14	17	37	52
		i				j	

③

$8 <= 25$ True

$i = i+1 = 3$

$63 <= 25$ (False)

$37 >= 25$ True

$j = j-1$

$j = 6-1 = 5$

$17 >= 25$ [False]

0	1	2	3	4	5	6	7
25	6	8	63	14	17	37	52
			i		j		

swap (3, 5)

0	1	2	3	4	5	6	7
25	6	8	17	14	63	37	52
			i		j		

④ 17 <= 25 True

i = i+1

i = 3+1 = 4

14 <= 25 (True)

i = i+1

i = 4+1 = 5

63 <= 25 (false)

63 >= 25 (True)

j = j-1

j = 5-1

j = 4

14 >= 25 false

0	1	2	3	4	5	6	7
25	6	8	17	14	63	37	52
				j	i		

i <= j 5 <= 4 false

swap(A[0], A[4])

0	1	2	3	4	5	6	7
14	6	8	17	25	63	37	52

---

m = 4

quick (A[0 ... 3])

quick (A[5 ... 7])

→ quick (A[0 ... 3]

if (0 < 3)

{

m = partition(A[0 ... 3])

}

pivot = A[low]

pivot = A[0] = 14

i = 0

j = 3

① (0 <= 3) True

0	1	2	3
14	6	8	17
i			j

14 <= 14

i = i+1

i = 0+1

6 <= 14 True

i = i+1

i = 1+1 = 2

8 <= 14 True

i = 2+1 = 3

17 <= 14 False

j 7 >= 14 True

j = j - 1

j = 2

8 >= 14 False

0	1	2	3
14	6	8	17
		j	i

swap (A[low], A[j]) j = 2

8 6 14 17

m = j m = 2

A[0....1]

6 8 14 17

⇒ quick (5 ... 7) (low, high)

5	6	7
63	37	52
i		j

pivot = A[5] = 63

i = 5

j = 7

63 <= 63 True

i = i+1

i = 6

37 <= 63 True

i = i+1 = 7

52 <= 63 True

i = i+1

i = 8

52 >= 63 false

5	6	7
63	37	52
		i j

i = 8

8 <= j

swap(A[low], A[j])

5	6	7
52	37	63
		j

37	52	63

Complexity Analysis of quick sort

- worst case time complexity $O(n^2)$
- Best case time complexity $O(n \log n)$
- Average case time complexity $O(n \log n)$
- space complexity $O(n \log n)$

→ Quick sort is not a stable sorting.