QuickSort Algorithm

↳ Dividing → mid = i + (j − i) // 2
↳ Other application

Dividing ↓ QuickSort

Partition Algorithm

* Partition Algorithm

	0	1	2	3	4	5	6	7	8
Pivot Element ←	(70)	50	27	~~99~~ 47	~~85~~ ~~34~~ 70	~~47~~ 99	123	~~34~~ 85	147
	p	j	j	j	j	j	j	j	j / q

i → i → i → i → i

J → > pivot → move on

i → < pivot i = p

```
for (j = i+1; j < n; j++)
{
    if arr[j] <= pivot :
    {
        i = i+1
        swap(arr(i), arr(j))
    }
}
swap(arr(i), arr(p))
```

Smaller than Pivot Element ⇑ | pivot | ⇑ Larger than pivot element

n = 4

0	1	2	3	4	5	6	7	8
34	50	27	47	(70)	99	123	85	147

QuickSort(0, 3) — QS(p, m−1)

QuickSort(5, 8) — QS(m+1, q)

0	1	2	3	4	5	6	7	8
27	(34)	47	50	(70)	85	99	123	147

→ final sorted array

Psuedocode of Partition Algo

```
Partition(arr, p, q):
    i = p
    pivot = arr(p)
    for j = i+1 to n-1:
        if arr(j) <= pivot:
            i = i+1
            swap(arr(i), arr(j))
    swap(arr(i), arr(p))
    return i
```

O(n)

j → > pivot
↳ move on
i → ≤ pivot

pivot → Pivot

Psuedocode QuickSort Algorithm

```
QuickSort(arr, p, q):
    if p < q:
        m = Partition(arr, p, q)
        QuickSort(arr, p, m-1)
        QuickSort(arr, m+1, q)
```

Divide

Left side Recursion ← QuickSort(arr, p, m-1) — T(m-p)

Right side Recursion ← QuickSort(arr, m+1, q) — T(q-m)

Conquer

* No need of combine part in QuickSort algo

Recursive Tree

27, 34, 47, 50

QuickSort(arr, 0, 8) mid = 4

QuickSort(arr, 0, 3)

QuickSort(arr, 5, 8)

34, 50, 27, 47 m = 1

P J → J → J

i i

(27) 34 (50, 47)

QuickSort(arr, 0, 0)

27

27

(47, 50)

QuickSort(2, 3)

47 50

50, 47 m = 3

P Q

i J

i

QuickSort(2, 2)

47

QuickSort (3, 3)

50

27, 19, 49, 57, 29, 46, 37

0 1 2 3 4 5 6

19, 27, 49, 57, 29, 46, 37

i = 1 → m = 1

Recurrence Relation

QuickSort

$$T(n) = \begin{cases} 1 & n = 1 \\ T(n-p) + T(q-n) + n & n > 1 \end{cases}$$

Best case scenario

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

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

Master's Theorem or Substitution Method;

$$T(n) = O(n \log n)$$

Worst case scenario

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

$$T(n) = O(n^2)$$

array is highly unsorted

⇑

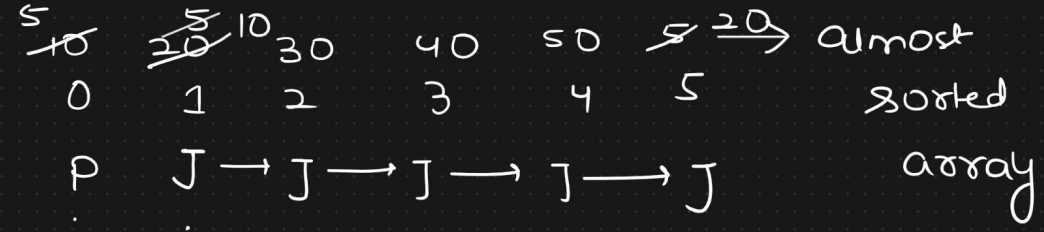
Best/average case scenario
↳ $O(n \log n)$

QuickSort

Insertion Sort → $O(n)$

array is almost/fully sorted

Worst case scenario
↳ $O(n^2)$



↳ Worst case scenario

⇓

QuickSort

⇓

$O(n^2)$

Note:

↳ Inplace sorting algorithm (not using any extra space)

↳ Not a stable algorithm (QuickSort, HeapSort)

10^2 10^2 10^3 10^4 10^5

sorting

10^1 10^2 10^3 10^4 10^5

↳ sequence is not retained

↳ Real time scenario

(Actually the array is unsorted)