4) Quick Sort

↳ An element is in a sorted Position if all elements before that element are smaller & all elements after that element are greater.

50, 70, 60, 90, 40, 80, 10, 20, 80

(i → 50, j → 80)

Pivot

- Pivot will find out its place. → all elements < 50 on one side
  → all elements > 50 on other side.
- i looks for any element > Pivot. (i should stop if element at i is > Pivot)
- j looks for any element ≤ Pivot. (j should stop if element at j ≤ Pivot)

Pivot	i						j	
50	70	60	90	40	80	10	20	30	∞
50	30	60	90	40	80	10	20	70	∞
50	30	20	90	40	80	10	60	70	∞
50	30	20	10	40	80	90	60	70	∞

(j → 40, i → 80)

After stoping swap those elements

- Here $j < i$, so interchange Pivot element with jth element.

(40 30 20 10) 50 (80 90 60 70)

(40 30 20 10) < 50 ; (80 90 60 70) > 50

50 sorted

↑ Partitioning Position.

- Quick Sort will be Recursively applied on LS & RS.
- " " uses Partitioning Procedure to sort the elements.

* Quick Sort works if there are 2 elements.

Algo/Pseudocode →

For already Sorted →

10 (20 30 40 50) ∞ (n)

20 (30 40 50) ∞ (n-1)

(30 40 50) ∞ (n-2)

(40 50) ∞ 2

50 ∞ 1

$1+2+3+\dots n \Rightarrow \frac{n(n+1)}{2} \Rightarrow O(n^2)$ time complexity

For descending order,

n

(n-1)

(n-2)

Again $O(n^2)$

Conclusion → If elements are in ascending or descending order, Quick sort takes Worst time $O(n^2)$

Best case → List Partitioned in middle.

n

n/2 n/2

n/4 n/4 n/4 n/4

n

n

(n comparisons $\log_n$ times)

$\Rightarrow O(n \log_2 n)$

$\Rightarrow$ Best case → $O(n \log n)$

Worst case → $O(n^2)$ partitioning at any end.

Average case → $O(n \log n)$

→ No extra space needed

One element? Stop Recursion

In case we select middle element as Pivot :→

- Bring middle element ~~at~~ first position
- After sorting it will go and sit in middle only in case of sorted list so partitioning will always happen in the middle!

We can also select any random element.

{ then, Best Case = Sorted list $O(n \log n)$
Worst case = Partitioning at any end $O(n^2)$

~~Partition (A, h)~~ <Pivot selected is last element>

```
QuickSort(A, start, end)
{
Pindex = Partition(A, start, end)   //Rearrange array till start to end. s.t
                                    Left to Pivot less than Pivot & Right element
                                    >= Pivot
Quick Sort(A, Start, Pindex - 1)
Quick sort(A, Pindex+1, end)
}
```

```
Base Case → if(start >= end){
                return;
            }
```

→ only 1 element exit
→ if segment not valid then also exit.

```
Partition(A, start, end){          //At any instance elements < Pivot will be left of Pindex.
    Pivot = A[end];
    Pindex = start;
    for(i = start to end-1){
        if(A[i] <= Pivot){
            swap(A[i], A[pindex]);
            Pindex++;
        }
    }
    swap(A[Pindex], A[end])
}
```

FREEMIND