

1) Quick Sort

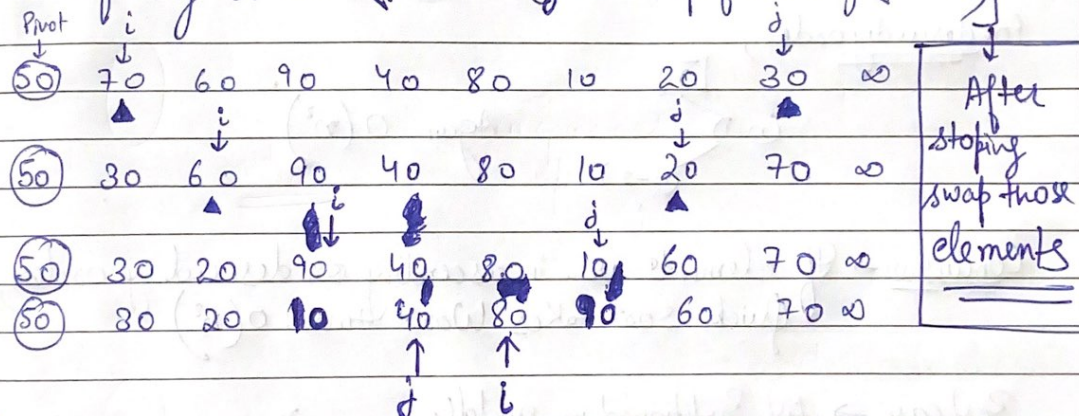
↳ An element is in a sorted position if all elements before that element are smaller & all elements after that element are greater.

i
 (50), 70, 60, 90, 40, 80, 10, 20, 80
 ↑
Pivot

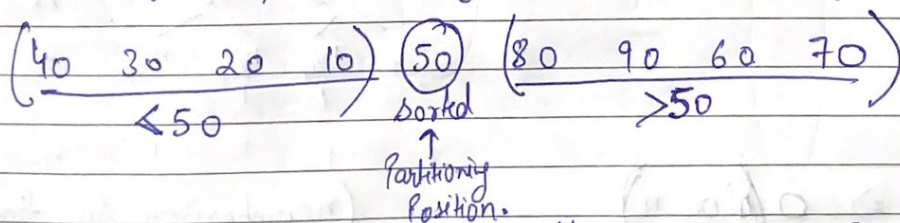
- Pivot will find out its place
 - all elements < 50 on one side
 - all elements > 50 on other side.

- i looks for any element $> \text{Pivot}$. (should stop if it is $> \text{Pivot}$)

- j looks for any element $\leq \text{Pivot}$. (should stop if element at $j \leq \text{Pivot}$)



• Here $j < i$, so interchange Pivot element with j th element.



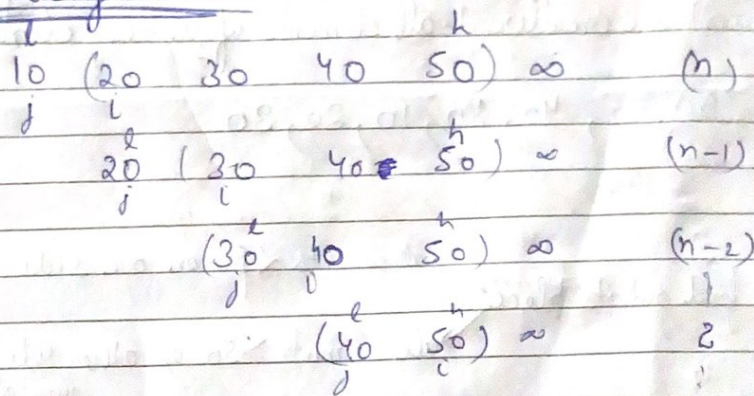
• Quick Sort will be Recursively applied on LS & RS.

• " " uses Partitioning Procedure to sort the elements.

* Quick Sort works if there are 2 elements.

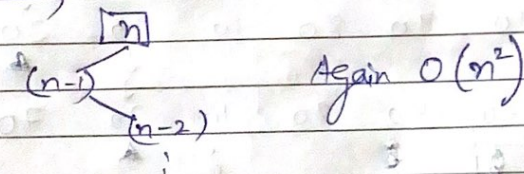
Ago/Pseudocode →

For already Sorted →



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

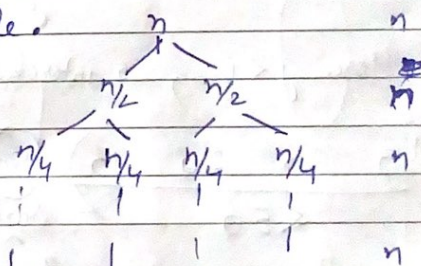
For descending order,



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.



$$\Rightarrow O(\log_2 n) \quad (n \text{ comparisons } \log_2 n \text{ times})$$

⇒ Best case → $O(n \log n)$

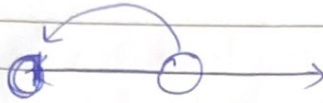
Worst case → $O(n^2)$ partitioning at any end.

Average case → $O(n \log n)$

→ No extra space needed

One element?
Stop Recursion

On 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, start, end)~~

<Pivot selected is last element>

QuickSort(A, start, end)
{

 Pindex = Partition(A, start, end)
 QuickSort(A, start, Pindex - 1)
 QuickSort(A, Pindex + 1, end)
}

//Rearrange array till start to end s.t
Left to Pivot < Pivot & Right element
≥ Pivot

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])

}