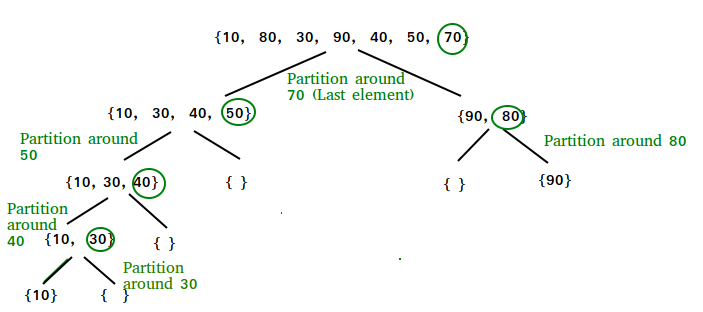
***QuickSort***

***QuickSort****is a sorting algorithm based on the Divide and Conquer algorithm that picks an element as a pivot and partitions the given array around the picked pivot by placing the pivot in its correct position in the sorted array.*

**How does QuickSort work?**

*The key process in****quickSort****is a****partition()****. The target of partitions is to place the pivot (any element can be chosen to be a pivot) at its correct position in the sorted array and put all smaller elements to the left of the pivot, and all greater elements to the right of the pivot.*

*Partition is done recursively on each side of the pivot after the pivot is placed in its correct position and this finally sorts the array.*

**

There are many different choices for picking pivots.

* Always pick the first element as a pivot.
* Always pick the last element as a pivot (implemented below)
* Pick a random element as a pivot.
* Pick the middle as the pivot.

**Partition Algorithm:**

*The logic is simple, we start from the leftmost element and keep track of the index of smaller (or equal) elements as****i****. While traversing, if we find a smaller element, we swap the current element with arr[i]. Otherwise, we ignore the current element.*

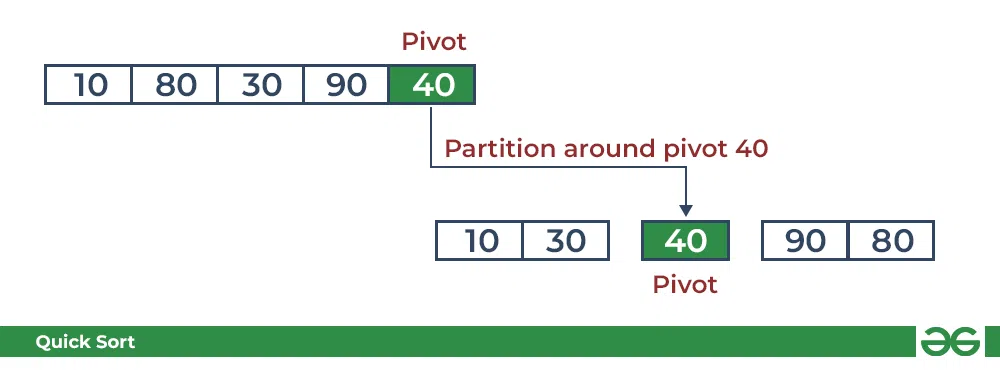
Let us understand the working of partition and the Quick Sort algorithm with the help of the following example:

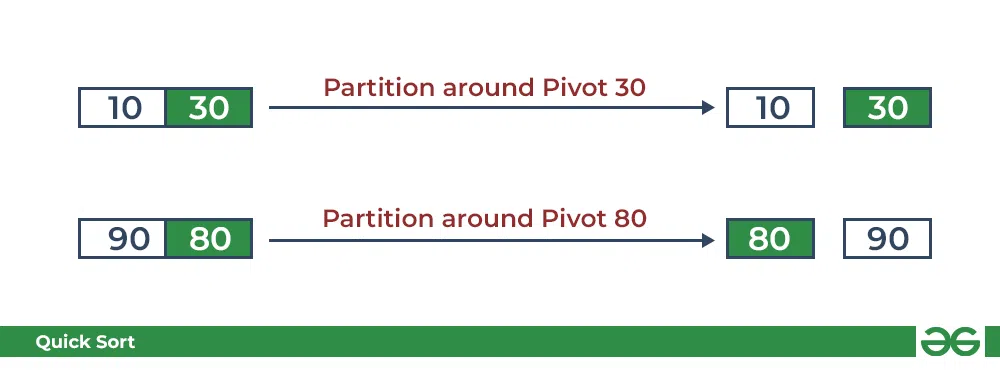
*Consider: arr[] = {10, 80, 30, 90, 40}.*

**CONSIDER FOLLOWING e.g. pivot at last**

**Illustration of Quicksort:**

As the partition process is done recursively, it keeps on putting the pivot in its actual position in the sorted array. Repeatedly putting pivots in their actual position makes the array sorted.





Complexity Analysis of Quick Sort**:**

**Time Complexity:**

* **Best Case: Ω (N log (N))**
* **Average Case:** **θ ( N log (N))**
* **Worst Case:** O(N2)

**Auxiliary Space:** O(1) as no extra space is used

**Advantages of Quick Sort:**

* It is a divide-and-conquer algorithm that makes it easier to solve problems.
* It is efficient on large data sets.
* It has a low overhead, as it only requires a small amount of memory to function.

**Disadvantages of Quick Sort:**

* It has a worst-case time complexity of O(N2), which occurs when the pivot is chosen poorly.
* It is not a good choice for small data sets.
* It is not a stable sort, meaning that if two elements have the same key, their relative order will not be preserved in the sorted output in case of quick sort, because here we are swapping elements according to the pivot’s position (without considering their original positions).