QuickSort is a Divide and Conquer algorithm. It picks an element as a pivot and partitions the given array around the pivot. There are many different versions of quickSort that pick the pivot in different ways.

- Always pick the first element as a pivot.
- Always pick the last element as a pivot.
- Pick a random element as a pivot.
- Pick the median as the pivot.

recursive QuickSort function:

```
/ low –> Starting index,  high  –> Ending index
quick Sort(arr[], low, high) {
   if (low < high) {

      // pi is partitioning index, arr[pi] is now at right place
      pi = partition(arr, low, high);
      quick Sort(arr, low, pi – 1);  // Before pi
      quick Sort(arr, pi + 1, high); // After pi
   }
}
```

partition() function

```
/* This function takes first element as pivot, places the pivot element at its correct position in
sorted array, and places all smaller (smaller than or equal to pivot) to left of pivot and all
greater elements to right of pivot */
partition (arr[], low, high) {
   // first element as pivot
   pivot = arr[low]
   k = high
   for (i = high; i > low; i–) {
      if (arr[i] > pivot){
         swap arr[i] and arr[k];
         k–;
      }
   }
   swap arr[k] and arr[low]
   return k-1;
}
```

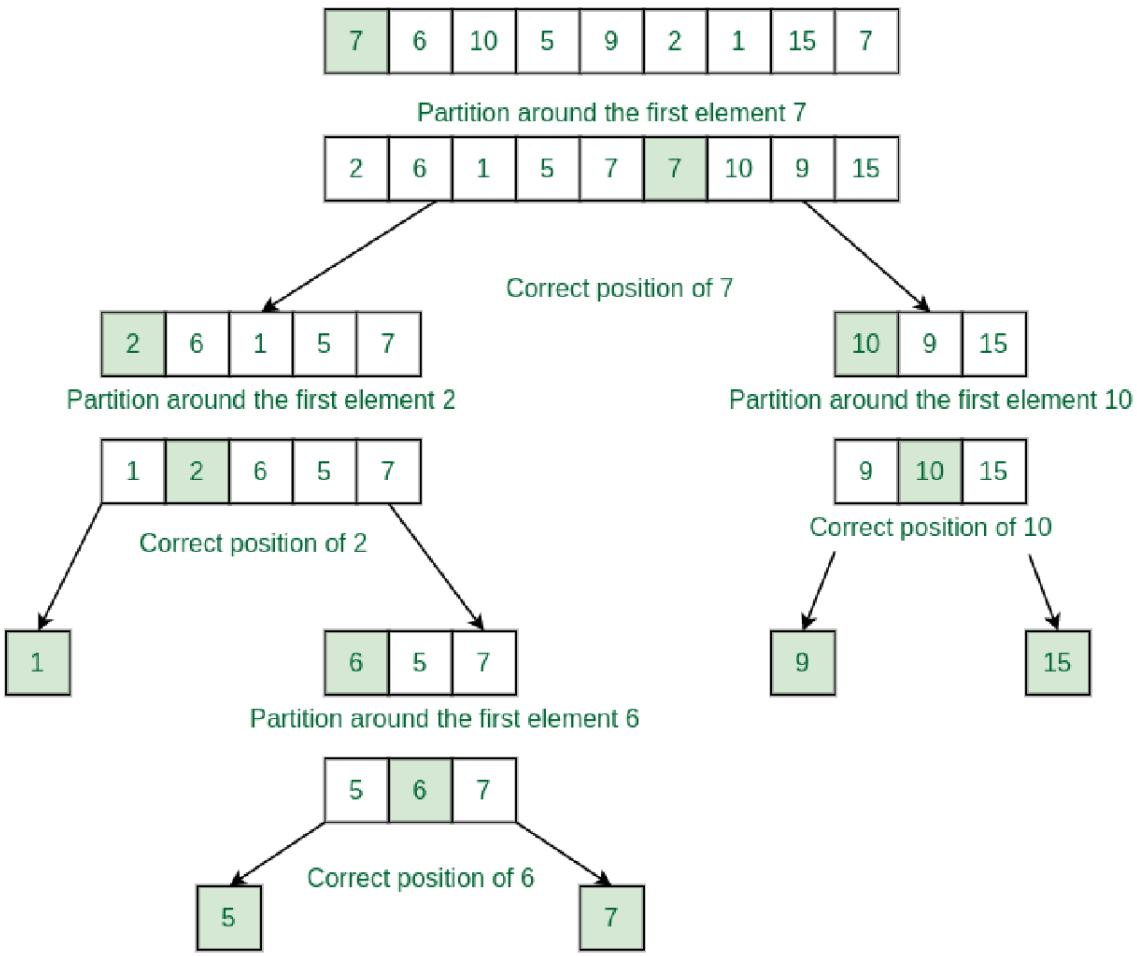
illustration

Consider: arr[] = { 7, 6, 10, 5, 9, 2, 1, 15, 7 }

First Partition: *low = 0, high = 8, pivot = arr[low] = 7*

Initialize index of right most element k = high = 8.

- *Traverse from i = high to low:*
  - *if arr[i] is greater than pivot:*
    - *Swap arr[i] and arr[k].*
    - *Decrement k;*
- *At the end swap arr[low] and arr[k].*



Complexity Analysis:

- **Time Complexity:**
  - **Average Case**: O(N * logN), where N is the length of the array.
  - **Best Case:** O(N * logN)
  - **Worst Case:** $O(N^2)$
- **Auxiliary Space:** O(1)