**5. Quick Sort:**

- Like Merge Sort, Quick Sort also follows the **Divide and Conquer** mechanism to sort a given set of numbers recursively.
- The quick sort algorithm attempts to separate the list of elements into two parts and then sort each part recursively.
- The partition of the list is performed based on a selected **pivot** element from the list.
- The list is divided into two partitions such that-

  all elements to the left of pivot are smaller than the pivot and

  all elements to the right of pivot are greater than or equal to the pivot.

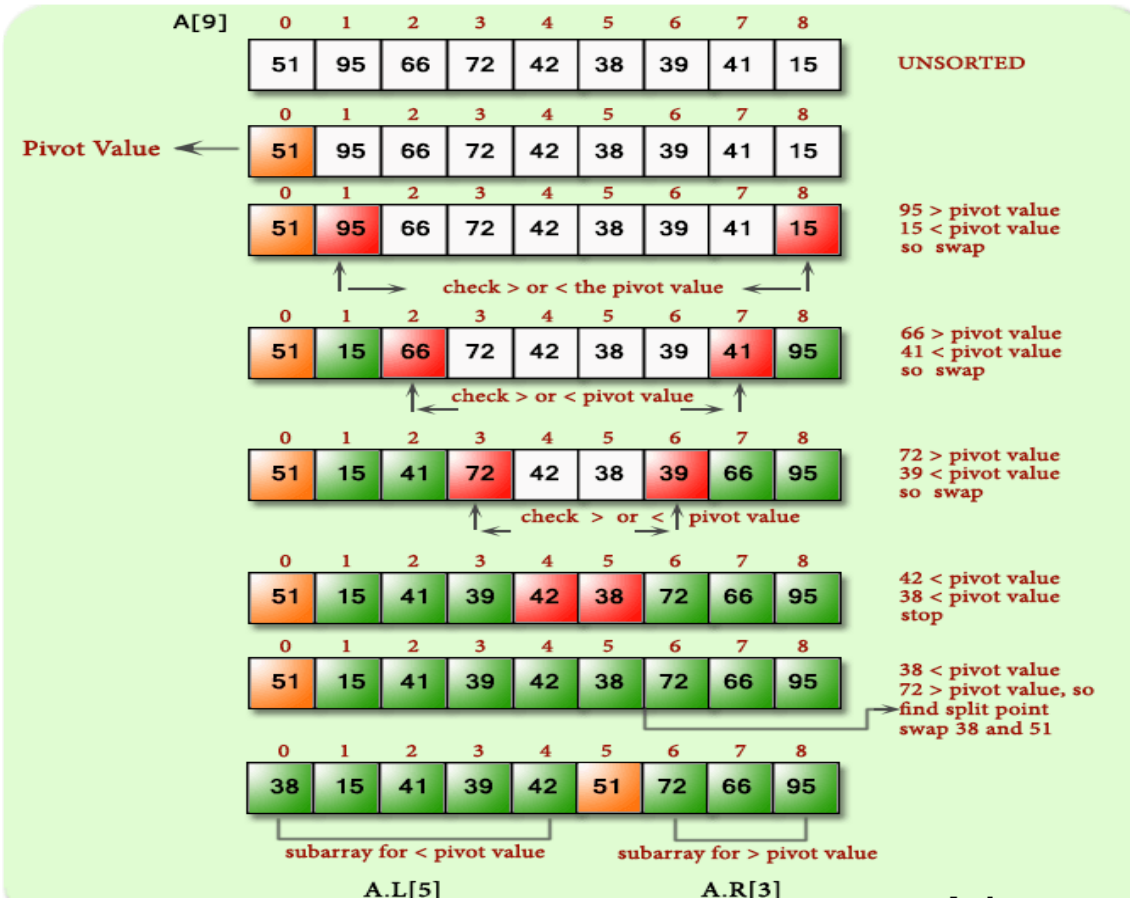
- Pivot element can be any element from the array:

  it can be the first element, or

  the last element or

  any random element.

**Example-1:** Considering first element as pivot




Applying the same process of partitioning recursively on left and right sub arrays, finally the obtained sorted array will be:

| 15 | 38 | 39 | 41 | 42 | 51 | 66 | 72 | 95 |
|---|---|---|---|---|---|---|---|---|

**Algorithm:**

```
void quicksort(int a[], int p, int r)
{
    int q;
    if(p < r)
    {
        q = partition(a, p, r);
        quicksort(a, p, q);
        quicksort(a, q+1, r);
    }
}

int partition(int a[], int p, int r)
{
     int pivot, i, j, temp;
     pivot = a[p];  // selecting first element as pivot
     i=p-1;
     j=r+1
     while(1)
     {
          do
          {
               j=j-1;
          }while(a[j]>pivot);
          do
          {
               i=i+1;
          }while(a[i]<pivot);
          if(i<j)
          {
               temp=a[i];
               a[i]=a[j];
               a[j]=temp;
          }
          else
               return(j);
     }
}
```