QUICK SORT

- Quick sort is also known as Partition-exchange sort based on the rule of Divide and Conquer.
- It is a highly efficient sorting algorithm.
- Quick sort is the quickest comparison-based sorting algorithm.
- It is very fast and requires less additional space, only O(n log n) space is required.
- Quick sort picks an element as pivot and partitions the array around the picked pivot.

Algorithm for Quick Sort:

Step 1: Choose the highest index value as pivot.

Step 2: Take two variables to point left and right of the list excluding

pivot.Step 3: Left points to the low index.

Step 4: Right points to the high index.

Step 5: While value at left < (Less than) pivot move right.

Step 6: While value at right > (Greater than) pivot move

left.

Step 7: If both Step 5 and Step 6 does not match, swap left and right.

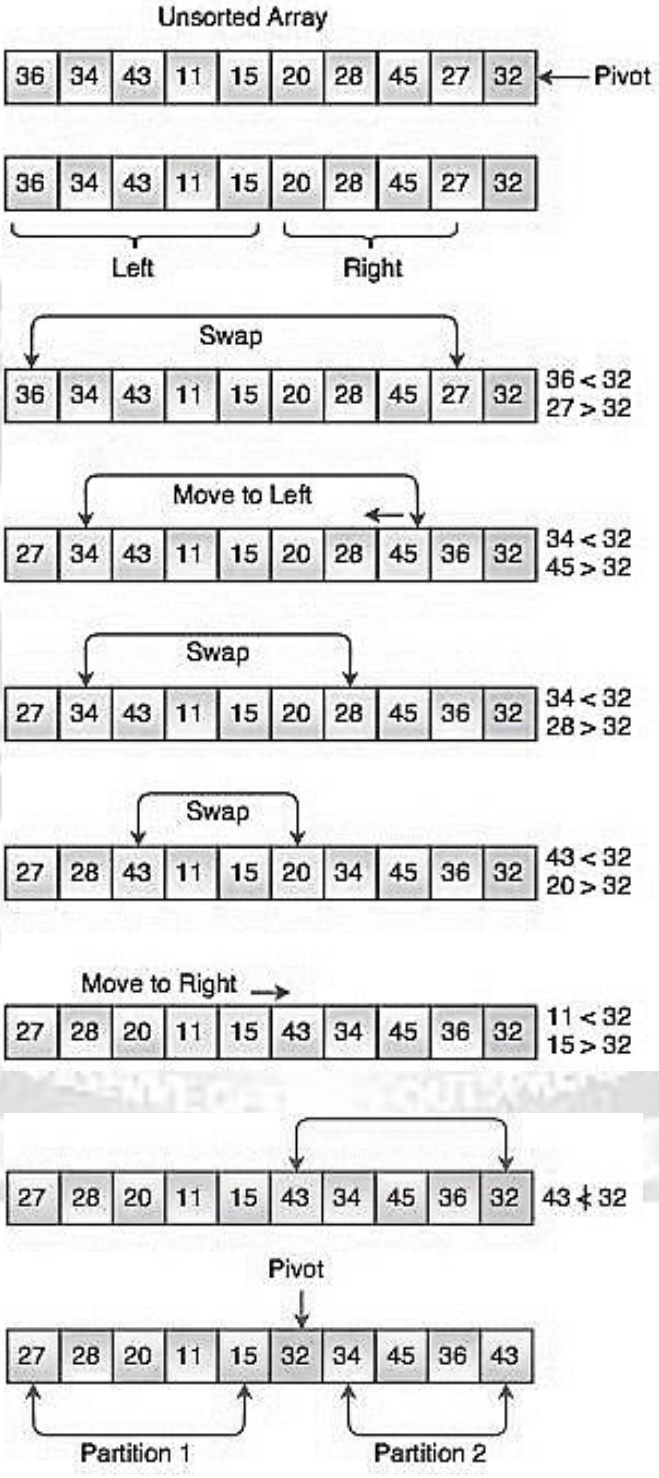
Step 8: If left = (Less than or Equal to) right, the point where they met is new pivot.

Working of Quick sort Algorithm

Consider an unsorted array as follows

36, 34, 43, 11, 15, 20, 28, 45, 27, 32

- The following steps represents how to find the pivot value in an array.
- As we see, pivot value divides the list into two parts (partitions) and then each part is processed for quick sort.
- Quick sort is a recursive function.
- We can call the partition function again.


Unsorted Array
36 34 43 11 15 20 28 45 27 32
Pivot
36 34 43 11 15 20 28 45 27 32
Left
Right
Swap
36 34 43 11 15 20 28 45 27 32
36 < 32
27 > 32
Move to Left
27 34 43 11 15 20 28 45 36 32
34 < 32
45 > 32
Swap
27 34 43 11 15 20 28 45 36 32
34 < 32
28 > 32
Swap
27 28 43 11 15 20 34 45 36 32
43 < 32
20 > 32
Move to Right
27 28 20 11 15 43 34 45 36 32
11 < 32
15 > 32
27 28 20 11 15 43 34 45 36 32
43 ≮ 32
Pivot
27 28 20 11 15 32 34 45 36 43
Partition 1
Partition 2

Quicksort Complexity

Time Complexity	
Best	O(n*log n)
Worst	$O(n^2)$
Average	O(n*log n)
Space Complexity	O(log n)
Stability	No

Applications of quick sort:

Quicksort algorithm is used when

- the programming language is good for recursion
- time complexity matters
- space complexity matters

Example Program 5.2: Program for implementing Quick Sort

```
#include<stdio.h>
#include<conio.h>
//quick Sort function to Sort Integer array list
void quicksort(int array[], int firstIndex, int lastIndex)
{
   //declaaring index variables
   int pivotIndex, temp, index1, index2;
   if(firstIndex < lastIndex)
   {
      //assigninh first element index as pivot element
      pivotIndex = firstIndex;
      index1 = firstIndex;
      index2 = lastIndex;
      //Sorting in Ascending order with quick sort
```

```
    while(index1 < index2)
    {
      while(array[index1] <= array[pivotIndex] && index1 < lastIndex)
      {
        index1++;
      }
      while(array[index2]>array[pivotIndex])
      {
        index2--;
      }
      if(index1<index2)
      {
        //Swapping opertation
        temp = array[index1];
        array[index1] = array[index2];
        array[index2] = temp;
      }
    }
    //At the end of first iteration, swap pivot element with index2 element
    temp = array[pivotIndex];
    array[pivotIndex] = array[index2];
    array[index2] = temp;
    //Recursive call for quick sort, with partiontioning
    quicksort(array, firstIndex, index2-1);
    quicksort(array, index2+1, lastIndex);
  }
}
int main()
{
```

```
//Declaring variables
int array[100],n,i;
//Number of elements in array form user input
printf("Enter the number of element you want to Sort : ");
scanf("%d",&n);
//code to ask to enter elements from user equal to n
printf("Enter Elements in the list : ");
for(i = 0; i < n; i++)
{
  scanf("%d",&array[i]);
}
 //calling quickSort function defined above
 quicksort(array,0,n-1);
 //print sorted array printf("Sorted elements: ");
 for(i=0;i<n;i++)
   printf(" %d",array[i]);getch();
 return 0;
 }
```

Output

Enter the number of element you want to sort: 5

Enter the elements in the list:

7

10

3

21

15

Sorted elements: 3 7 10 15 21