Quick Sort Algorithm

Quick Sort, as the name suggests, sorts any list very quickly. Quick sort is not stable search, but it is very fast and requires very less aditional space. It is based on the rule of **Divide and Conquer**(also called *partition-exchange sort*). This algorithm divides the list into three main parts :

1. Elements less than the Pivot element
2. Pivot element
3. Elements greater than the pivot element

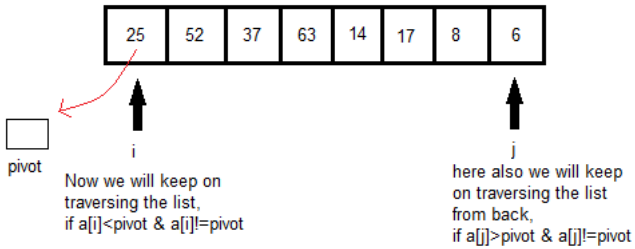
In the list of elements, mentioned in below example, we have taken **25** as pivot. So after the first pass, the list will be changed like this.

```
6 8 17 14 25 63 37 52
```

Hnece after the first pass, pivot will be set at its position, with all the elements smaller to it on its left and all the elements larger than it on the right. Now `6 8 17 14` and `63 37 52` are considered as two separate lists, and same logic is applied on them, and we keep doing this until the complete list is sorted.

How Quick Sorting Works

Coming Soon



if both sides we find the element not satisfying their respective conditions, we swap them. And keep repeating this.

DIVIDE AND CONQUER - QUICK SORT

Sorting using Quick Sort Algorithm

```
/*  a[] is the array, p is starting index, that is 0,
and r is the last index of array.  */

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

int partition(int a[], int p, int r)
{
  int i, j, pivot, temp;
  pivot = a[p];
  i = p;
  j = r;
  while(1)
  {
   while(a[i] < pivot && a[i] != pivot)
   i++;
   while(a[j] > pivot && a[j] != pivot)
   j--;
   if(i < j)
   {
    temp = a[i];
    a[i] = a[j];
    a[j] = temp;
   }
   else
   {
    return j;
   }
  }
}
```

Complexity Analysis of Quick Sort

Worst Case Time Complexity : $O(n^2)$

Best Case Time Complexity : O(n log n)

Average Time Complexity : O(n log n)

Space Complexity : O(n log n)

- Space required by quick sort is very less, only O(n log n) additional space is required.
- Quick sort is not a stable sorting technique, so it might change the occurence of two similar elements in the list while sorting.

← Prev (selection-sorting)

Next → (merge-sort)