**Design and Analysis of Algorithm**

**Experiment No. : 3**

**Write a program to implement Quick sort and Merge sort**

Experiment No. 3

1. **Aim:** Write a program to implement Quick sort and Merge sort.
2. **Algorithm**

**Quicksort** is the widely used sorting algorithm that makes **n log n** comparisons in average case for sorting an array of n elements. It is a faster and highly efficient sorting algorithm. This algorithm follows the divide and conquers approach. Divide and conquer is a technique of breaking down the algorithms into subproblems, then solving the subproblems, and combining the results back together to solve the original problem.

**Divide:** In Divide, first pick a pivot element. After that, partition or rearrange the array into two sub-arrays such that each element in the left sub-array is less than or equal to the pivot element and each element in the right sub-array is larger than the pivot element.

**Conquer:** Recursively, sort two subarrays with Quicksort.

**Combine:** Combine the already sorted array.361Hello Java Program for Beginner

**Quick Sort**

1. QUICKSORT (array A, start, end)
2. {
3. 1 **if** (start < end)
4. 2 {
5. 3 p = partition(A, start, end)
6. 4 QUICKSORT (A, start, p - 1)
7. 5 QUICKSORT (A, p + 1, end)
8. 6 }
9. }

**Partition Algorithm:**

The partition algorithm rearranges the sub-arrays in a place.

1. PARTITION (array A, start, end)
2. {
3. 1 pivot ? A[end]
4. 2 i ? start-1
5. 3 **for** j ? start to end -1 {
6. 4 **do** **if** (A[j] < pivot) {
7. 5 then i = i + 1
8. 6 swap A[i] with A[j]
9. 7  }}
10. 8 swap A[i+1] with A[end]
11. 9 **return** i+1
12. }

**Best Case Complexity -** In Quicksort, the best-case occurs when the pivot element is the middle element or near to the middle element. The best-case time complexity of quicksort is **O(n\*logn)**.

**Average Case Complexity -** It occurs when the array elements are in jumbled order that is not properly ascending and not properly descending. The average case time complexity of quicksort is **O(n\*logn)**.

**Worst Case Complexity -** In quick sort, worst case occurs when the pivot element is either greatest or smallest element. Suppose, if the pivot element is always the last element of the array, the worst case would occur when the given array is sorted already in ascending or descending order. The worst-case time complexity of quicksort is **O(n2)**.

**Merge sort** is similar to the quick sort algorithm as it uses the divide and conquer approach to sort the elements. It is one of the most popular and efficient sorting algorithm. It divides the given list into two equal halves, calls itself for the two halves and then merges the two sorted halves. We have to define the **merge()** function to perform the merging.

The sub-lists are divided again and again into halves until the list cannot be divided further. Then we combine the pair of one element lists into two-element lists, sorting them in the process. The sorted two-element pairs is merged into the four-element lists, and so on until we get the sorted list

**Merge Sort**

1. MERGE\_SORT(arr, beg, end)
3. **if** beg < end
4. set mid = (beg + end)/2
5. MERGE\_SORT(arr, beg, mid)
6. MERGE\_SORT(arr, mid + 1, end)
7. MERGE (arr, beg, mid, end)
8. end of **if**
10. END MERGE\_SORT

**Best Case Complexity -** It occurs when there is no sorting required, i.e. the array is already sorted. The best-case time complexity of merge sort is **O(n\*logn)**.

**Average Case Complexity -** It occurs when the array elements are in jumbled order that is not properly ascending and not properly descending. The average case time complexity of merge sort is **O(n\*logn)**.

**Worst Case Complexity -** It occurs when the array elements are required to be sorted in reverse order. That means suppose you have to sort the array elements in ascending order, but its elements are in descending order. The worst-case time complexity of merge sort is **O(n\*logn)**.

1. **Conclusion and Discussion:** Hence we have implemented Quick sort and Merge sort algorithm