Assignment No. 1

Aim: Implement Quick Sort using divide and conquer strategy.

Theory:

Quick Sort is an efficient in-place sorting algorithm that uses the divide and conquer strategy to sort an array. It works by selecting a pivot element from the array and partitioning the array into two sub-arrays, one with elements less than the pivot, and the other with elements greater than the pivot. This process is repeated recursively on the two sub-arrays until the entire array is sorted.

- Time Complexity:

Best case: O(nlogn)

Average case: O(nlogn)

Worst case: O(n^2)

- Space Complexity:

Worst case: O(n)

Code: -

```
public class QuickSort {
public static void quickSort(int[] arr, int left, int right) {
if (left < right) {
int partitionIndex = partition(arr, left, right);
quickSort(arr, left, partitionIndex - 1);
quickSort(arr, partitionIndex + 1, right);
}
}
private static int partition(int[] arr, int left, int right) {
int pivot = arr[right];
int i = left - 1;
for (int j = left; j < right; j++) {
if (arr[j] <= pivot) {
```

```
i++;
swap(arr, i, j);
}
}
swap(arr, i + 1, right);
return i + 1;
}
private static void swap(int[] arr, int i, int j) {
int temp = arr[i];
arr[i] = arr[j];
arr[j] = temp;
}
public static void main(String[] args) {
int[] arr = {4, 2, 6, 8, 1, 3, 5, 7};
int n = arr.length;
System.out.println("Original Array: " + Arrays.toString(arr));
quickSort(arr, 0, n - 1);
System.out.println("Sorted Array: " + Arrays.toString(arr));
}
}
```

Output: -

Original Array: [4, 2, 6, 8, 1, 3, 5, 7]

Sorted Array: [1, 2, 3, 4, 5, 6, 7, 8]