Worksheet-6

Student Name: Arpit Anand
UID: 23BCS12710
Branch: BE - CSE
Section/Group :KGR-3(A)
Semester: 5
Subject Name: DAA

Date of Performance: 09/09/2026
Subject Code: 23CSH-301

1.Aim:
Sort a given set of elements using the Quick sort method and determine the time required to sort the elements. Repeat the experiment for different values of n, the number of elements in the list to be sorted. The elements can be read from a file or can be generated using the random number generator.

2.Objective:
To study and analyze the performance of the Quick Sort algorithm by measuring the time taken to sort arrays of different sizes (n).

3.Requirements (Hardware/Software):
Online Java compiler.

4.Algorithm :
1. Create an array of size n (read from file or generate randomly).
2. Apply Quick Sort using divide and conquer.
3. In Quick Sort, choose a pivot, partition the array, and recursively sort left and right parts.
4. Measure the time before and after sorting.
5. Print the time taken.
6. Repeat for different values of n.
7. End.

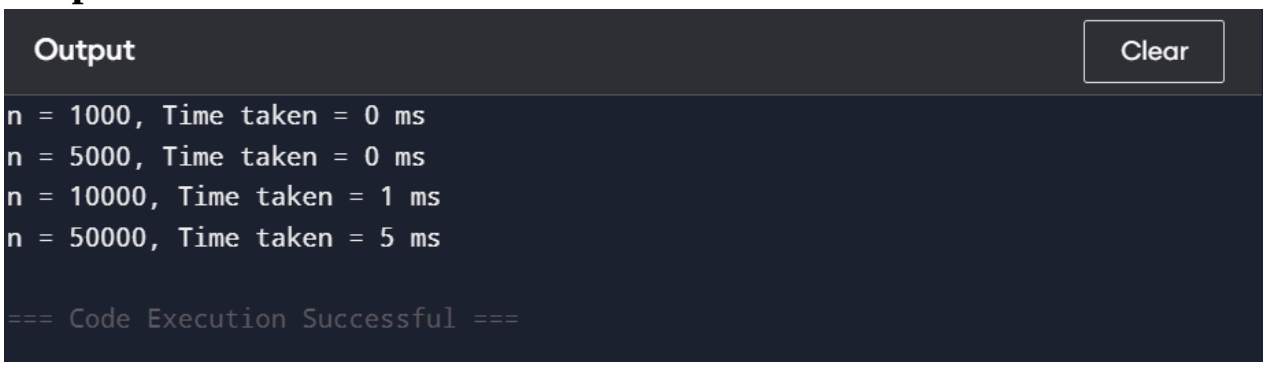
5.Procedure:

```
import java.util.*;
import java.time.*;
public class Main {
    static void quickSort(int[] arr, int low, int high) {
        if (low < high) {
            int p = partition(arr, low, high);
            quickSort(arr, low, p - 1);
            quickSort(arr, p + 1, high);
        }
    }
    static int partition(int[] arr, int low, int high) {
        int pivot = arr[high], i = low - 1;
        for (int j = low; j < high; j++) {
            if (arr[j] <= pivot) {
                i++;
                int temp = arr[i]; arr[i] = arr[j]; arr[j] = temp;
            }
        }
        int temp = arr[i + 1]; arr[i + 1] = arr[high]; arr[high] = temp;
        return i + 1;
    }
    public static void main(String[] args) {
        Random rand = new Random();
        int[] sizes = {1000, 5000, 10000, 50000};
        for (int n : sizes) {
            int[] arr = new int[n];
            for (int i = 0; i < n; i++) arr[i] = rand.nextInt(100000);
            Instant start = Instant.now();
            quickSort(arr, 0, n - 1);
            Instant end = Instant.now();
            long timeMs = Duration.between(start, end).toMillis();
            System.out.println("n = " + n + ", Time taken = " + timeMs + "
ms");
        }
    }
}
```

Time Complexity : Best Case: O(n log n)
Average Case: O(n log n)
Worst Case: $O(n^2)$.

Space complexity : O(log n) (In the best/average case).
O(n) (in the worst case due).

Output:



Learning Outcomes :

1. Understand the working of Quick Sort using the divide and conquer approach.
2. Analyze the time required for sorting different input sizes.
3. Compare the efficiency of Quick Sort based on time and space complexity.