CHANDIGARH UNIVERSITY
Discover. Learn. Empower.

NAAC GRADE A+
Accredited University

Worksheet No. - 1

Student Name: Devanand Utkarsh
UID: 24MCA20454
Branch: MCA
Section/Group: 6(B)
Semester: 2nd
Date of Performance: 10/01/2025
Subject Name: DAA LAB
Subject Code: 24CAP-612

Aim/Overview of the practical:

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 and plot a graph of the time taken versus n. The elements can be read from a file or can be generated using the random number generator.

Objective:

Implement Quick Sort algorithm for different values of n.

Input/Apparatus Used:

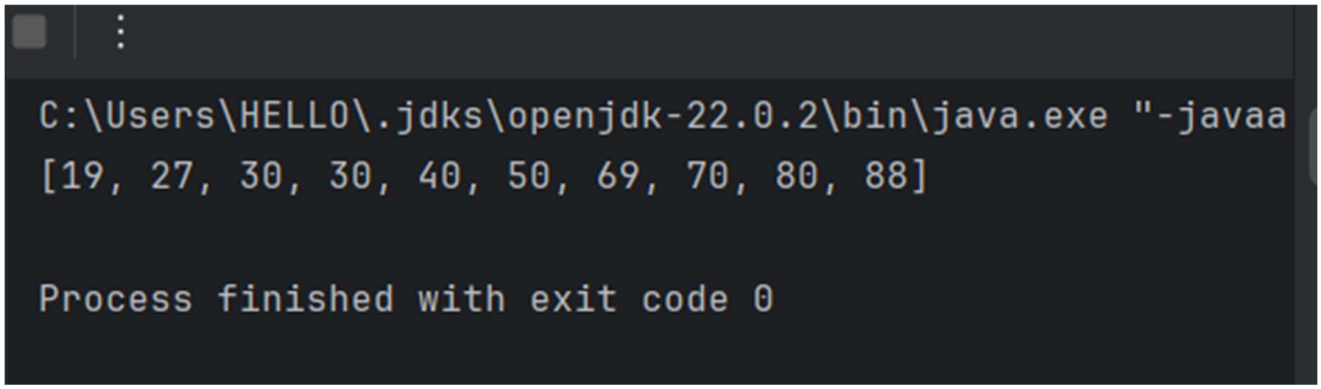
IntelliJ Idea as code editor.

Procedure/Algorithm/Code:

```
import java.lang.reflect.Array;
import java.util.Arrays;

public class quick_Sort {
    public static void main(String[] args) {
        int[] arr = {4, 3, 2, 1, 5};
        Qsort(arr, 0, arr.length - 1);
        System.out.println(Arrays.toString(arr));

    }
```

```
    static void Qsort(int[] arr, int low, int hi) {
        if (low >= hi) {
            return;
        }
        int s = low, e = hi, m = s + (e - s) / 2;
        int pivot = arr[m];
//        int pivot = arr[hi];
//        int pivot = (int) (Math.random() * arr.length);

        while (s <= e) {

            while (arr[s] < pivot) {
                s++;
            }
            while (arr[e] > pivot) {
                e--;
            }

            if (s <= e) {
                int temp = arr[s];
                arr[s] = arr[e];
                arr[e] = temp;
                s++;
                e--;
            }


            Qsort(arr, low, e);
            Qsort(arr, s, hi);

        }
    }
}
```

Output:



Learning outcomes (What I have learnt):

- Recognized the importance of selecting an optimal pivot (e.g., first, last, random, or median-of-three) to reduce the likelihood of encountering worst-case performance in Quick Sort.

- Explored how recursive calls break the array into smaller subarrays, emphasizing the efficiency of the divide-and-conquer strategy.

- Gained a clear understanding of the partitioning process, which rearranges elements around the pivot to create two subarrays: one with smaller values and the other with larger values.