**DAAOA Assignment 1**

**Name: Atharva Salitri**

**Roll No.: 029**

**Branch: TY CSAI-B**

**Batch: B2**

**PRN: 12310120**

**Title: Implement Bubble and Quick Sort with Time and Space Complexity Analysis**

import java.util.Scanner;

public class SortingDemo {

    public static void bubbleSort(int[] arr) {

        int n = arr.length;

        for (int i = 0; i < n - 1; i++) {

            for (int j = 0; j < n - i - 1; j++) {

                if (arr[j] > arr[j + 1]) {

                    int temp = arr[j];

                    arr[j] = arr[j + 1];

                    arr[j + 1] = temp;

                }

            }

        }

    }

    public static void quickSort(int[] arr, int low, int high) {

        if (low < high) {

            int pi = partition(arr, low, high);

            quickSort(arr, low, pi - 1);

            quickSort(arr, pi + 1, high);

        }

    }

    public static int partition(int[] arr, int low, int high) {

        int pivot = arr[high];

        int 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 printArray(int[] arr) {

        for (int num : arr) {

            System.out.print(num + " ");

        }

        System.out.println();

    }

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        System.out.println("Enter no. of elements:");

        int n = sc.nextInt();

        int[] arr = new int[n];

        System.out.println("Enter elements:");

        for (int i = 0; i < n; i++) {

            arr[i] = sc.nextInt();

        }

        int[] arrBubble = arr.clone();

        int[] arrQuick = arr.clone();

        System.out.println("Bubble Sort:");

        bubbleSort(arrBubble);

        printArray(arrBubble);

        System.out.println("Quick Sort:");

        quickSort(arrQuick, 0, arrQuick.length - 1);

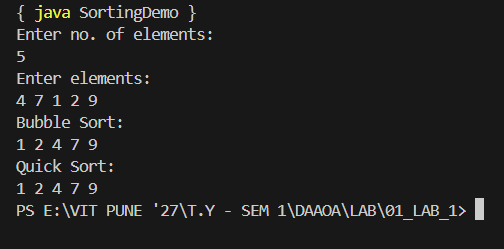
        printArray(arrQuick);

        sc.close();

    }

}

**OUTPUT:**

****

**Time and Space Complexity Analysis:**

**Bubble Sort:**

* Best Case Time Complexity: **O(n)** (when the array is already sorted)
* Average Case Time Complexity: **O(n²)**
* Worst Case Time Complexity: **O(n²)** (when the array is sorted in reverse order)
* Space Complexity: **O(1)** (in-place sorting with constant extra space)

**Quick Sort:**

* Best Case Time Complexity: **O(n log n)** (when partitioning divides array into nearly equal halves)
* Average Case Time Complexity: **O(n log n)**
* Worst Case Time Complexity: **O(n²)** (when partitioning is highly unbalanced, e.g., already sorted or reverse sorted with poor pivot choice)
* Space Complexity: **O(log n)** in best/average case (due to recursion stack)
* Space Complexity: **O(n)** in worst case (due to skewed recursion tree)