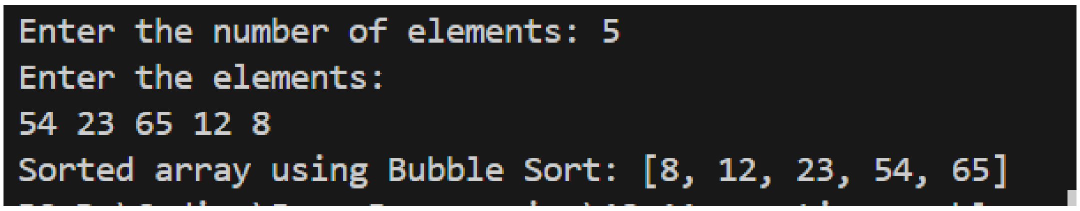
GATE TECHNICAL TRAINING DSA QUESTIONS 18-11-24

1.Bubble Sort

```java
import java.util.Scanner;
import java.util.Arrays;

public class BubbleSort {
    public static void bubbleSort(int[] arr) {
        int n = arr.length;
        for (int i = 0; i < n - 1; i++) {
            for (int j = 0; j < n - 1 - i; j++) {
                if (arr[j] > arr[j + 1]) {
                    // Swap arr[j] and arr[j+1]
                    int temp = arr[j];
                    arr[j] = arr[j + 1];
                    arr[j + 1] = temp;
                }
            }
        }
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter the number of elements: ");
        int n = scanner.nextInt();

        int[] arr = new int[n];
        System.out.println("Enter the elements:");
        for (int i = 0; i < n; i++) {
            arr[i] = scanner.nextInt();
        }

        bubbleSort(arr);
        System.out.println("Sorted array using Bubble Sort: " + Arrays.toString(arr));

        scanner.close();
    }
}
```

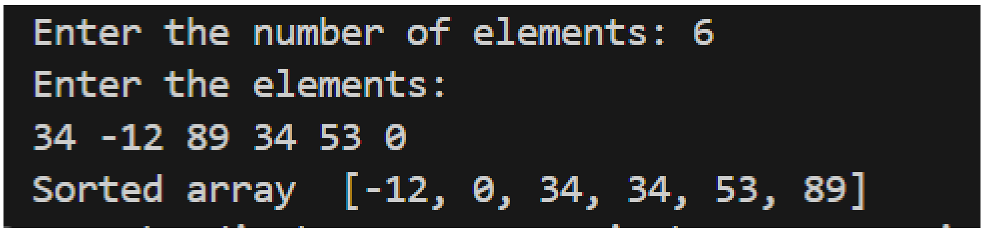


```
Enter the number of elements: 5
Enter the elements:
54 23 65 12 8
Sorted array using Bubble Sort: [8, 12, 23, 54, 65]
```

Time Complexity : O(n^2)

2.Quick Sort

```java
import java.util.Arrays;
import java.util.Scanner;

public class QuickSort {
    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]; // Pivot element
        int i = (low - 1); // Index of smaller element
        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) {
        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter the number of elements: ");
        int n = scanner.nextInt();

        int[] arr = new int[n];
        System.out.println("Enter the elements:");
        for (int i = 0; i < n; i++) {
            arr[i] = scanner.nextInt();
        }

        quickSort(arr, 0, n - 1);
        System.out.println("Sorted array  " + Arrays.toString(arr));

        scanner.close();
    }
}
```

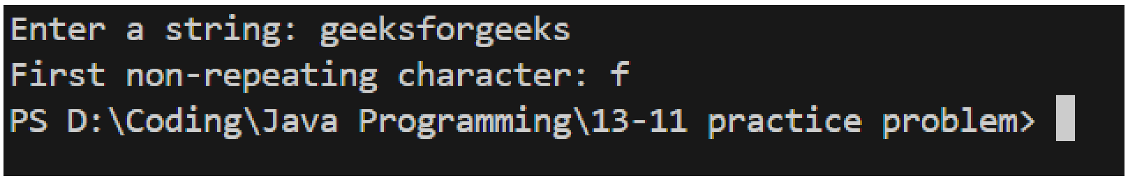


```
Enter the number of elements: 6
Enter the elements:
34 -12 89 34 53 0
Sorted array  [-12, 0, 34, 34, 53, 89]
```

Time Complexity:O(n log n)

3.Non Repeating Character

```
import java.util.Scanner;

public class FirstNonRepeatingCharacter {
    public static char firstNonRepeatingChar(String s) {
        int[] frequency = new int[26];

        for (char c : s.toCharArray()) {
            frequency[c - 'a']++;
        }

        for (char c : s.toCharArray()) {
            if (frequency[c - 'a'] == 1) {
                return c;
            }
        }

        return '$';
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter a string: ");
        String input = scanner.nextLine();

        System.out.println("First non-repeating character: " + firstNonRepeatingChar(input));
    }
}
```



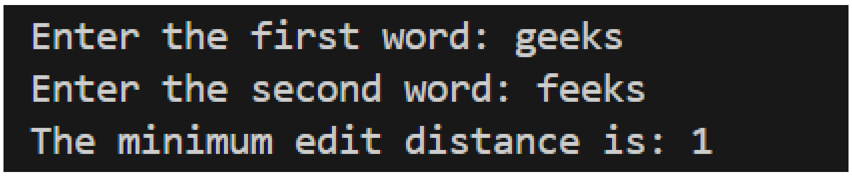
```
Enter a string: geeksforgeeks
First non-repeating character: f
PS D:\Coding\Java Programming\13-11 practice problem>
```

Time Complexity:O(n)

4.Edit Distance

```
import java.util.Scanner;

public class EditDistance {

    public static int minDistance(String word1, String word2) {
        int m = word1.length();
        int n = word2.length();

        int dp[][] = new int[m + 1][n + 1];
        for (int i = 0; i <= m; i++) {
            for (int j = 0; j <= n; j++) {
                if (i == 0) {
                    dp[i][j] = j;
                } else if (j == 0) {
                    dp[i][j] = i;
                } else if (word1.charAt(i - 1) == word2.charAt(j - 1)) {
                    dp[i][j] = dp[i - 1][j - 1];
                } else {
                    dp[i][j] = 1 + Math.min(Math.min(dp[i][j - 1], dp[i - 1][j]), dp[i - 1][j -
1]);
                }
            }
        }
```

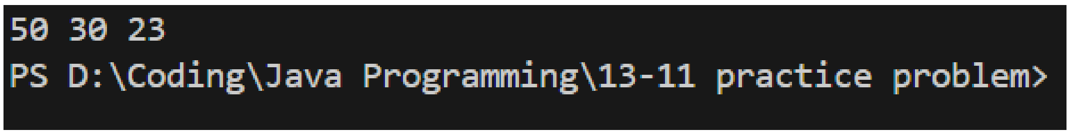
```java
        return dp[m][n];
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter the first word: ");
        String word1 = scanner.nextLine();

        System.out.print("Enter the second word: ");
        String word2 = scanner.nextLine();

        System.out.println("The minimum edit distance is: " + minDistance(word1, word2));

        scanner.close();
    }
}
```



```
Enter the first word: geeks
Enter the second word: feeks
The minimum edit distance is: 1
```

5. KLargest Element

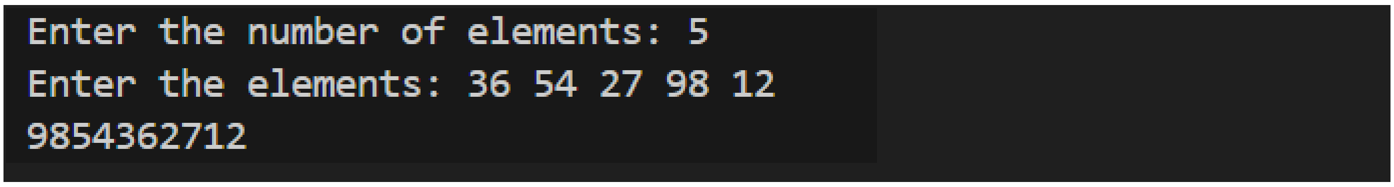
```java
import java.util.PriorityQueue;

public class KLargestElement {
    public static int[] findKLargestElements(int[] arr, int k) {
        PriorityQueue<Integer> minHeap = new PriorityQueue<>();

        for (int i = 0; i < k; i++) {
            minHeap.offer(arr[i]);
        }

        for (int i = k; i < arr.length; i++) {
            if (arr[i] > minHeap.peek()) {
                minHeap.poll();
                minHeap.offer(arr[i]);
            }
        }

        int[] result = new int[k];
        for (int i = k - 1; i >= 0; i--) {
            result[i] = minHeap.poll();
        }

        return result;
    }

    public static void main(String[] args) {
        int[] arr = {1, 23, 12, 9, 30, 2, 50};
        int k = 3;

        int[] kLargest = findKLargestElements(arr, k);
        for (int num : kLargest) {
            System.out.print(num + " ");
        }
    }
}
```



```
50 30 23
PS D:\Coding\Java Programming\13-11 practice problem>
```

6. Form Largest Number

```java
import java.util.Arrays;
import java.util.Comparator;
import java.util.Scanner;

public class LargestNumber {
    public static String largestNumber(int[] nums) {
        String[] strNums = new String[nums.length];
        for (int i = 0; i < nums.length; i++) {
            strNums[i] = String.valueOf(nums[i]);
        }

        Arrays.sort(strNums, new Comparator<String>() {
            @Override
            public int compare(String a, String b) {
                return (b + a).compareTo(a + b);
            }
        });

        if (strNums[0].equals("0")) {
            return "0";
        }

        StringBuilder sb = new StringBuilder();
        for (String str : strNums) {
            sb.append(str);
        }

        return sb.toString();
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter the number of elements: ");
        int n = scanner.nextInt();

        int[] nums = new int[n];
        System.out.print("Enter the elements: ");
        for (int i = 0; i < n; i++) {
            nums[i] = scanner.nextInt();
        }

        System.out.println(largestNumber(nums));}}
```


```
Enter the number of elements: 5
Enter the elements: 36 54 27 98 12
9854362712
```