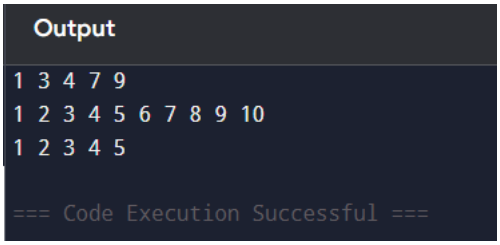
**Bubble Sort**

```
public class Main {
    public static void bubbleSort(int arr[]) {
        int n = arr.length;
        for (int i = 0; i < n - 1; i++) {
            boolean swapped = false;
            for (int j = 0; j < n - 1 - i; j++) {
                if (arr[j] > arr[j + 1]) {
                    int temp = arr[j];
                    arr[j] = arr[j + 1];
                    arr[j + 1] = temp;
                    swapped = true;
                }
            }
            if (!swapped) {
                break;
            }
        }
    }
    public static void printArray(int arr[]) {
        for (int i = 0; i < arr.length; i++) {
            System.out.print(arr[i] + " ");
        }
        System.out.println();
    }
    public static void main(String[] args) {
        int arr1[] = {4, 1, 3, 9, 7};
        bubbleSort(arr1);
        printArray(arr1);
        int arr2[] = {10, 9, 8, 7, 6, 5, 4, 3, 2, 1};
        bubbleSort(arr2);
```

```
        printArray(arr2);
        int arr3[] = {1, 2, 3, 4, 5};
        bubbleSort(arr3);
        printArray(arr3);
    }
}
```




Time Complexity : O(n^2)

**Quick Sort**

```
public class Main {
    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 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 void printArray(int arr[]) {
        for (int i = 0; i < arr.length; i++) {
            System.out.print(arr[i] + " ");
        }
        System.out.println();
    }
```

```
    public static void main(String[] args) {
        int arr1[] = {4, 1, 3, 9, 7};
        quickSort(arr1, 0, arr1.length - 1);
        printArray(arr1);
        int arr2[] = {2, 1, 6, 10, 4, 1, 3, 9, 7};
        quickSort(arr2, 0, arr2.length - 1);
        printArray(arr2);
        int arr3[] = {5, 5, 5, 5};
        quickSort(arr3, 0, arr3.length - 1);
        printArray(arr3);
    }
}
```

**Output**

```
1 3 4 7 9
1 2 3 4 5 6 7 8 9 10
1 2 3 4 5

=== Code Execution Successful ===
```

Time Complexity : O(n logn)

**Non Repeating Character**

```
public class Main {
    public static char firstNonRepeating(String s) {
        int[] frequency = new int[26];
        for (int i = 0; i < s.length(); i++) {
            frequency[s.charAt(i) - 'a']++;
        }
        for (int i = 0; i < s.length(); i++) {
            if (frequency[s.charAt(i) - 'a'] == 1) {
                return s.charAt(i);
            }
        }
        return '$';
    }
    public static void main(String[] args) {
        String s1 = "geeksforgeeks";
        String s2 = "racecar";
        String s3 = "aabbccc";
        System.out.println(firstNonRepeating(s1));
        System.out.println(firstNonRepeating(s2));
        System.out.println(firstNonRepeating(s3));
    }
}
```

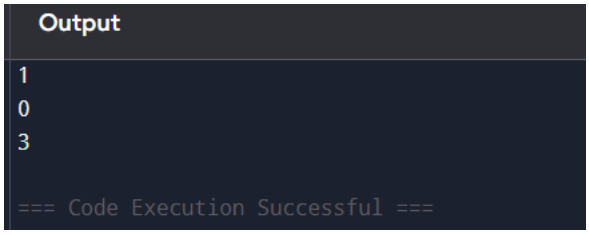
```
Output
f
e
$

=== Code Execution Successful ===
```

Time Complexity : O(n)

**Edit Distance**

```
public class Main {
    public static int minDistance(String s1, String s2) {
        int m = s1.length();
        int n = s2.length();
        int[][] dp = new int[m + 1][n + 1];
        for (int i = 0; i <= m; i++) {
            dp[i][0] = i;
        }
        for (int j = 0; j <= n; j++) {
            dp[0][j] = j;
        }
        for (int i = 1; i <= m; i++) {
            for (int j = 1; j <= n; j++) {
                if (s1.charAt(i - 1) == s2.charAt(j - 1)) {
                    dp[i][j] = dp[i - 1][j - 1];
                } else {
                    dp[i][j] = Math.min(dp[i - 1][j - 1], Math.min(dp[i - 1][j], dp[i][j - 1])) + 1;
                }
            }
        }
        return dp[m][n];
    }
    public static void main(String[] args) {
        String s1 = "geek";
        String s2 = "gesek";
        System.out.println(minDistance(s1, s2));
        String s1_2 = "gfg";
        String s2_2 = "gfg";
        System.out.println(minDistance(s1_2, s2_2));
        String s1_3 = "abc";
```

```
        String s2_3 = "def";
        System.out.println(minDistance(s1_3, s2_3));
    }
}
```




Time Complexity : O(m*n)

**K largest elements**

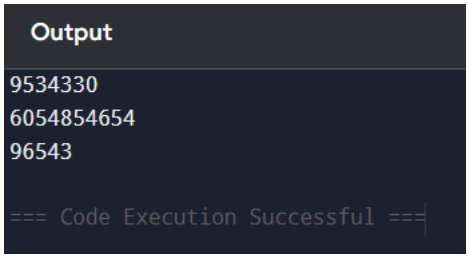
```
import java.util.Arrays;

public class Main {
    public static int[] kLargest(int[] arr, int k) {
        Arrays.sort(arr);
        int[] result = new int[k];
        for (int i = 0; i < k; i++) {
            result[i] = arr[arr.length - 1 - i];
        }
        return result;
    }
    public static void main(String[] args) {
        int[] arr1 = {12, 5, 787, 1, 23};
        int k1 = 2;
        System.out.println(Arrays.toString(kLargest(arr1, k1)));
        int[] arr2 = {1, 23, 12, 9, 30, 2, 50};
        int k2 = 3;
        System.out.println(Arrays.toString(kLargest(arr2, k2)));
        int[] arr3 = {12, 23};
        int k3 = 1;
        System.out.println(Arrays.toString(kLargest(arr3, k3)));
    }
}
```

```
Output
[787, 23]
[50, 30, 23]
[23]

=== Code Execution Successful ===
```

Time Complexity : O(n log n)

**Form the Largest Number**

```
import java.util.Arrays;
import java.util.Comparator;
public class Main {
   public static String largestNumber(int[] arr) {
      String[] strArr = new String[arr.length];
      for (int i = 0; i < arr.length; i++) {
         strArr[i] = String.valueOf(arr[i]);
      }
      Arrays.sort(strArr, new Comparator<String>() {
         public int compare(String x, String y) {
            return (y + x).compareTo(x + y);
         }
      });
      if (strArr[0].equals("0")) {
         return "0";
      }
      StringBuilder result = new StringBuilder();
      for (String num : strArr) {
         result.append(num);
      }
      return result.toString();
   }
   public static void main(String[] args) {
      int[] arr1 = {3, 30, 34, 5, 9};
      System.out.println(largestNumber(arr1));
      int[] arr2 = {54, 546, 548, 60};
      System.out.println(largestNumber(arr2));
      int[] arr3 = {3, 4, 6, 5, 9};
      System.out.println(largestNumber(arr3));
   }}
```




Time Complexity : O(n logn)