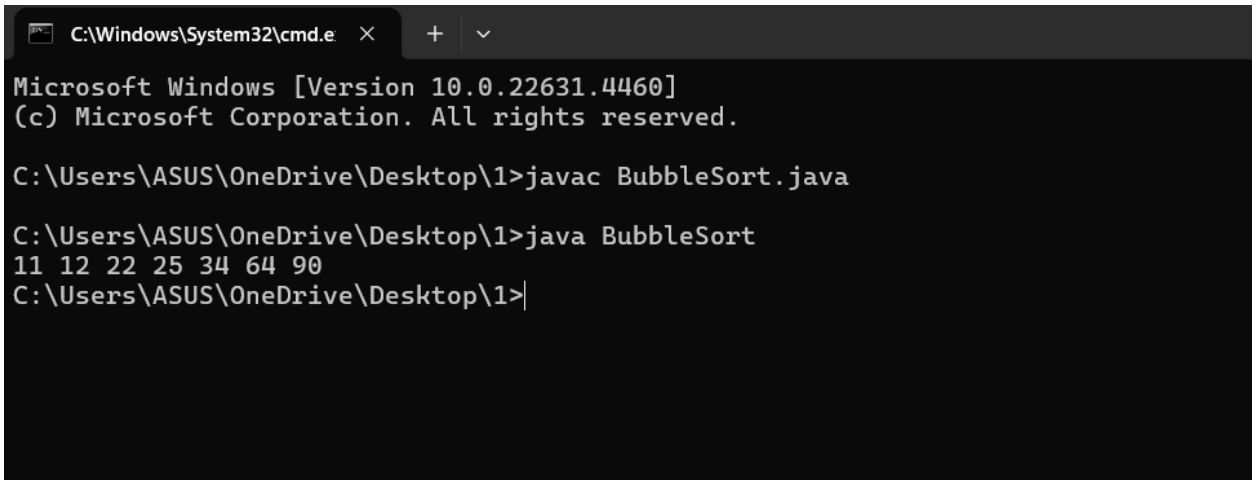
1)Bubble Sort:
Code:

```
public class BubbleSort {
    public static void main(String[] args) {
        int[] arr = {64, 34, 25, 12, 22, 11, 90};
        for (int i = 0; i < arr.length - 1; i++) {
            for (int j = 0; j < arr.length - 1 - i; j++) {
                if (arr[j] > arr[j + 1]) {
                    int temp = arr[j];
                    arr[j] = arr[j + 1];
                    arr[j + 1] = temp;
                }
            }
        }
        for (int i : arr) System.out.print(i + " ");
    }
}
```

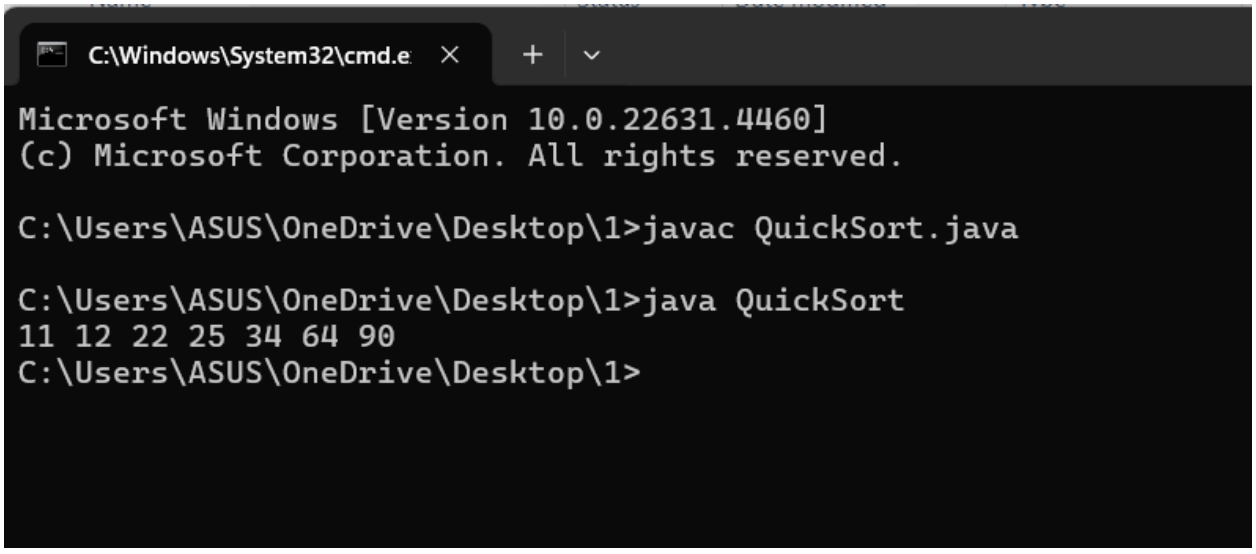


Time complexity:O(n^2)

2)Quick Sort:
Code:

```
public class QuickSort {
    public static void main(String[] args) {
        int[] arr = {64, 34, 25, 12, 22, 11, 90};
        quickSort(arr, 0, arr.length - 1);
```

```
        for (int i : arr) System.out.print(i + " ");
    }

    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;
    }
}
```

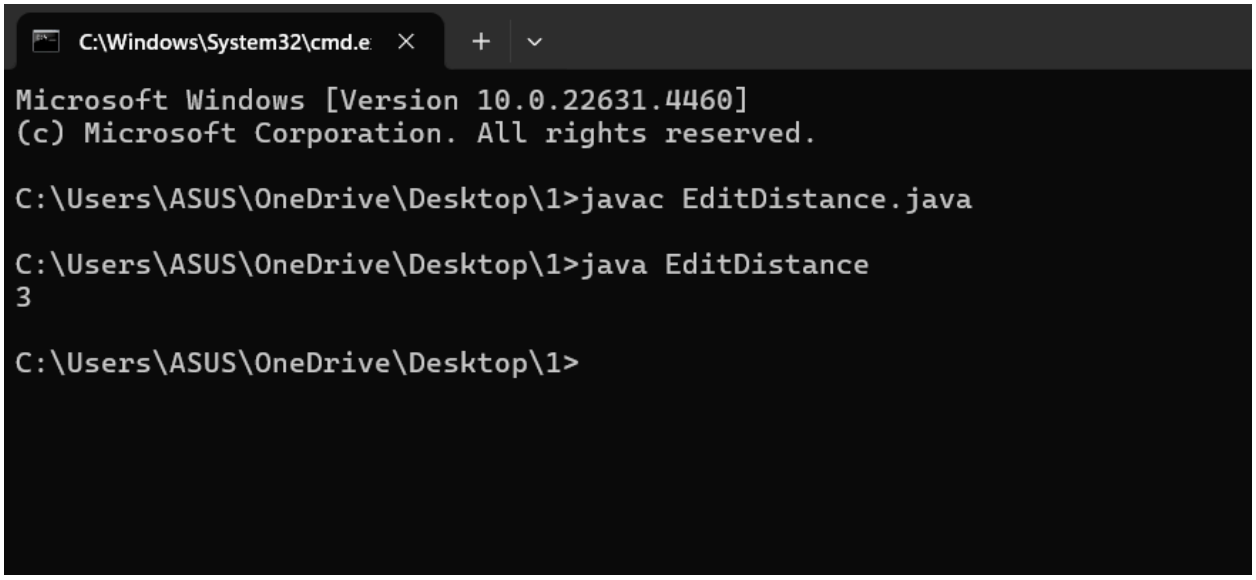


Time complexity:O(n^2)

3)Edit Distance:

```
public class EditDistance {
    public static void main(String[] args) {
        String str1 = "horse";
        String str2 = "ros";
        System.out.println(minDistance(str1, str2));
    }

    public static int minDistance(String word1, String word2) {
        int m = word1.length(), 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(dp[i - 1][j - 1], Math.min(dp[i - 1][j], dp[i][j
- 1]));
```

```
            }
        }

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

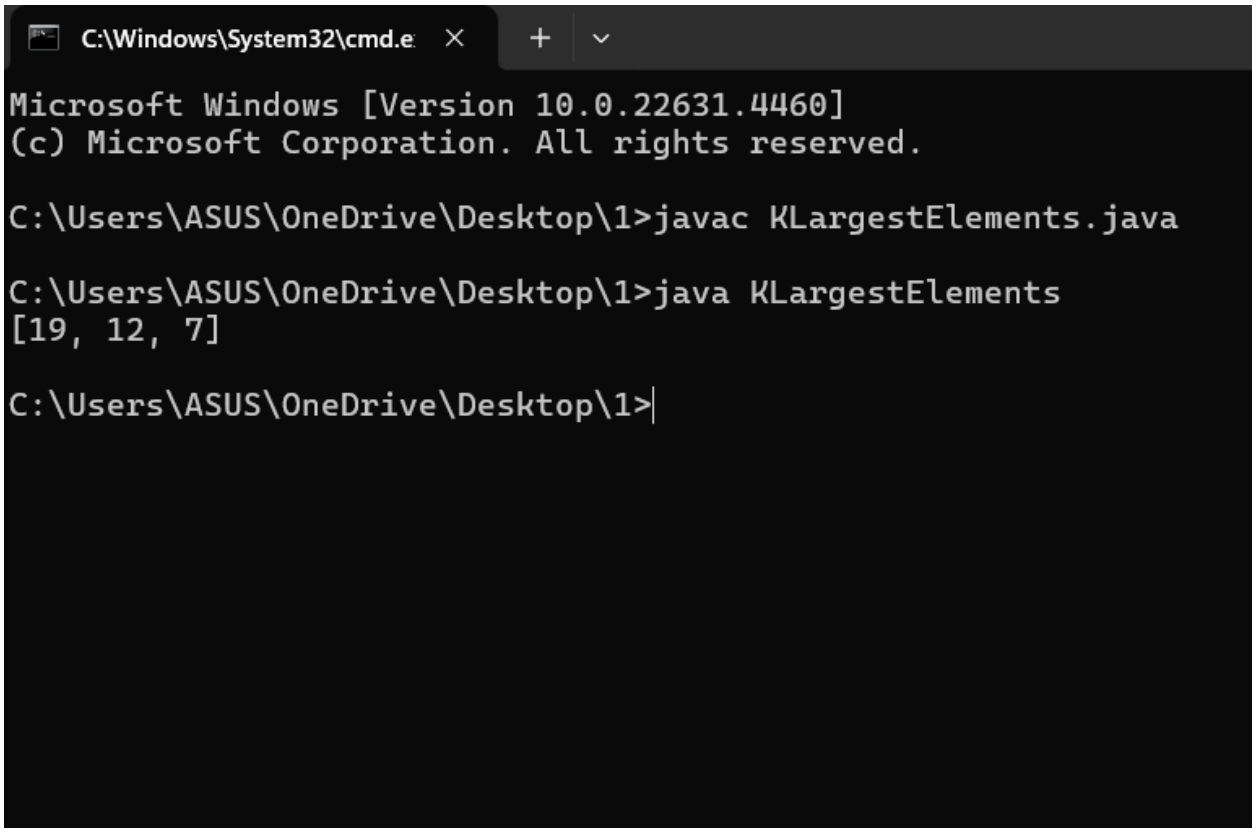


Time complexity:O(mxn)

4)k largest elements

```
import java.util.*;

public class KLargestElements {
    public static void main(String[] args) {
        int[] arr = {12, 3, 5, 7, 19};
        int k = 3;
        findKLargest(arr, k);
    }

    public static void findKLargest(int[] arr, int k) {
        PriorityQueue<Integer> minHeap = new PriorityQueue<>(k);
        for (int num : arr) {
```

```
            minHeap.add(num);
            if (minHeap.size() > k) {
                minHeap.poll();
            }
        }
        List<Integer> result = new ArrayList<>(minHeap);
        Collections.sort(result, Collections.reverseOrder());
        System.out.println(result);
    }
}
```

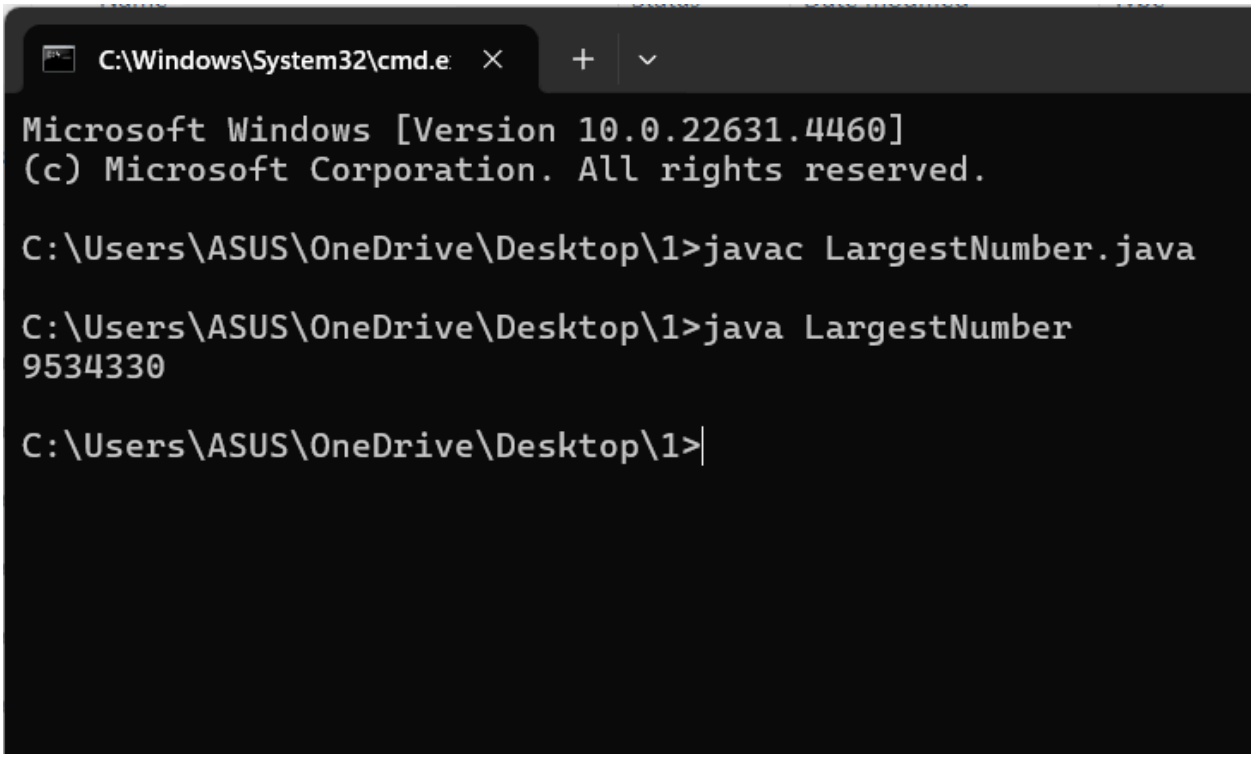


Time complexity:O(nlogk+klogk)

5)Form the largest Number

```
import java.util.*;

public class LargestNumber {
```

```
    public static void main(String[] args) {
        int[] nums = {3, 30, 34, 5, 9};
        System.out.println(formLargestNumber(nums));
    }

    public static String formLargestNumber(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, (a, b) -> (b + a).compareTo(a + b));

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

        StringBuilder result = new StringBuilder();
        for (String num : strNums) {
            result.append(num);
        }

        return result.toString();
    }
}
```



Time complexity:O(n logn)