**Bhuvaneswari P AIDS DSA Practice-6 18/11/24**

**1.BubbleSort**

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 - 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 main(String[] args) {

int[] arr = {5, 3, 8, 6, 2};

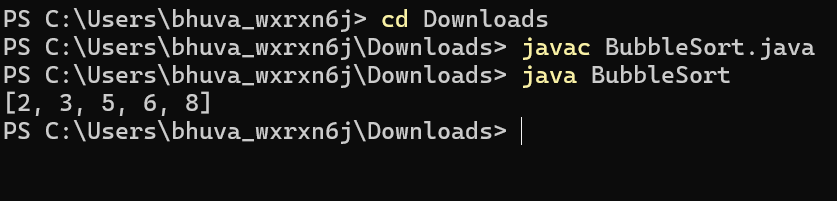
bubbleSort(arr);

System.out.println(Arrays.toString(arr));

}

}

**Output:**

****

**Time Complexity:** O(n^2)

**Space Complexity:** O(1)

**2.QuickSort**

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];

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 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);

}

public static void printArray(int[] arr) {

for (int num : arr) {

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

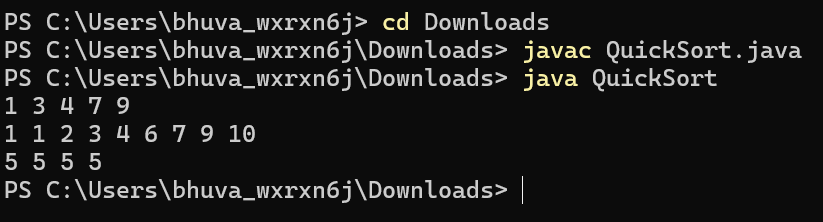
}

System.out.println();

}

}

Output:

****

**Time Complexity:** O(n^2)

**Space Complexity:** O(n)

**3.Non-repeating character**

public class FirstNonRepeating {

public static char firstNonRepeatingCharacter(String s) {

int[] count = new int[26];

for (int i = 0; i < s.length(); i++) {

count[s.charAt(i) - 'a']++;

}

for (int i = 0; i < s.length(); i++) {

if (count[s.charAt(i) - 'a'] == 1) {

return s.charAt(i);

}

}

return '$';

}

public static void main(String[] args) {

System.out.println(firstNonRepeatingCharacter("geeksforgeeks"));

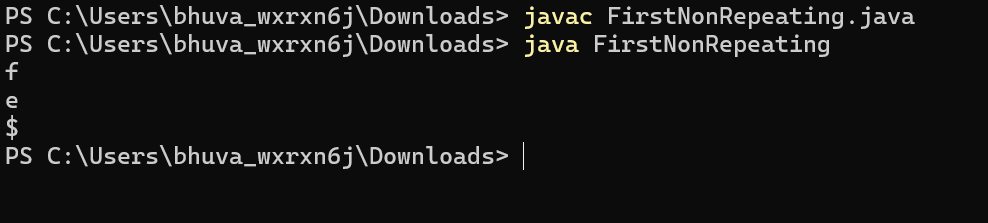
System.out.println(firstNonRepeatingCharacter("racecar"));

System.out.println(firstNonRepeatingCharacter("aabbccc"));

}

}

**Output:**

****

**Time Complexity:** O(n)

**Space Complexity:** O(1)

**4.Edit Distance**

public class EditDistance {

public static int minDistance(String str1, String str2) {

return calculateEditDistance(str1, str2, str1.length(), str2.length());

}

private static int calculateEditDistance(String str1, String str2, int len1, int len2) {

if (len1 == 0) return len2;

if (len2 == 0) return len1;

if (str1.charAt(len1 - 1) == str2.charAt(len2 - 1)) {

return calculateEditDistance(str1, str2, len1 - 1, len2 - 1);

}

int insertion = calculateEditDistance(str1, str2, len1, len2 - 1);

int deletion = calculateEditDistance(str1, str2, len1 - 1, len2);

int replacement = calculateEditDistance(str1, str2, len1 - 1, len2 - 1);

return 1 + Math.min(insertion, Math.min(deletion, replacement));

}

public static void main(String[] args) {

System.out.println(minDistance("geek", "gesek"));

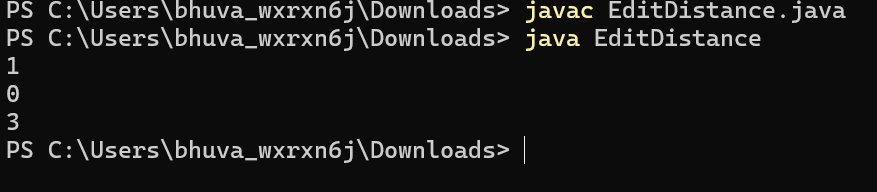
System.out.println(minDistance("gfg", "gfg"));

System.out.println(minDistance("abc", "def"));

}

}

**Output:**

****

**Time Complexity**: O(3^max(m,n))

**Space Complexity:** O(max(m,n))

**5.k-largest element**

import java.util.Arrays;

public class KlargestElements {

public int[] kLargest(int[] arr, int k) {

Arrays.sort(arr);

int[] result = new int[k];

int n = arr.length;

for (int i = 0; i < k; i++) {

result[i] = arr[n - 1 - i];

}

return result;

}

public static void main(String[] args) {

KlargestElements solution = new KlargestElements();

int[] arr = {1, 23, 12, 9, 30, 2, 50};

int k = 3;

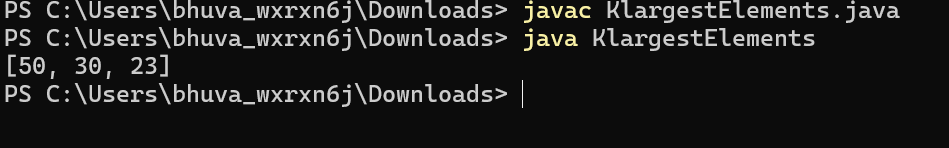
int[] largest = solution.kLargest(arr, k);

System.out.println(Arrays.toString(largest));

}

}

**Output:**

****

**Time Complexity: O(n logn)**

**Space Complexity: O(k)**

**6.Form the largest number**

import java.util.Arrays;

import java.util.Comparator;

public class FormLargestElement {

public String printLargest(int[] arr) {

String[] strArr = Arrays.stream(arr)

.mapToObj(String::valueOf)

.toArray(String[]::new);

Arrays.sort(strArr, new Comparator<String>() {

@Override

public int compare(String a, String b) {

String ab = a + b;

String ba = b + a;

return ba.compareTo(ab);

}

});

StringBuilder result = new StringBuilder();

for (String s : strArr) {

result.append(s);

}

if (result.charAt(0) == '0') {

return "0";

}

return result.toString();

}

public static void main(String[] args) {

FormLargestElement solution = new FormLargestElement();

int[] arr1 = {3, 30, 34, 5, 9};

int[] arr2 = {54, 546, 548, 60};

int[] arr3 = {3, 4, 6, 5, 9};

System.out.println(solution.printLargest(arr1));

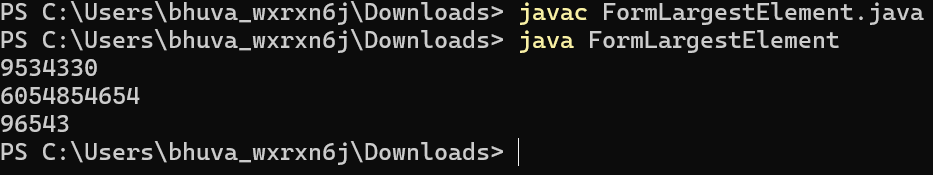
System.out.println(solution.printLargest(arr2));

System.out.println(solution.printLargest(arr3));

}

}

**Output:**

****

**Time Complexity:** O(n⋅k⋅logn)

**Space Complexity:** O(n)