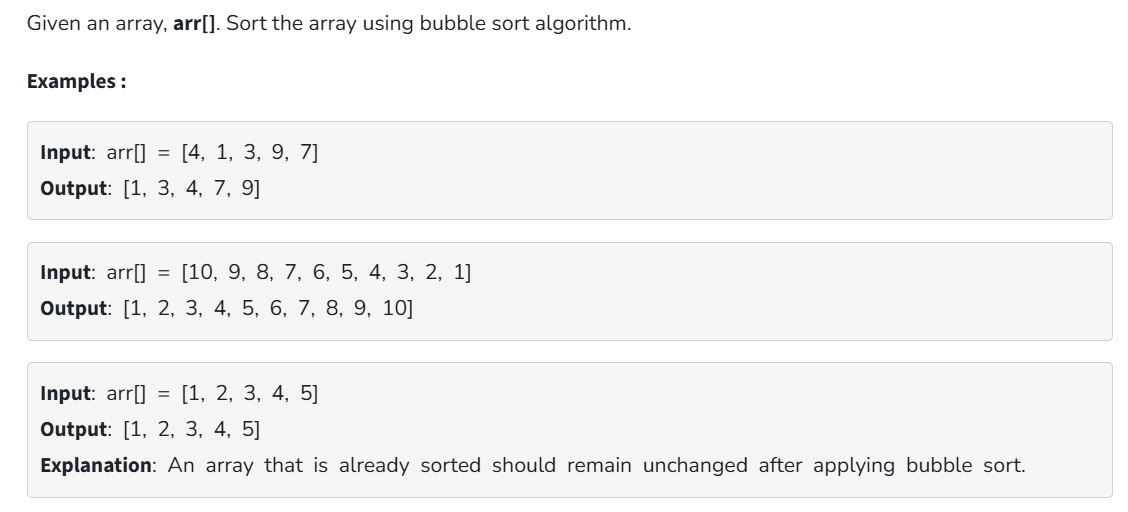
**DSA Practice Test – 6** 18th Nov 2024

**1. Bubble Sort**

****

**Code:**

class Solution {

public static void bubbleSort(int arr[]) {

for(int i=0;i<arr.length-1;i++){

boolean s=false;

for(int j=0;j<arr.length-i-1;j++){

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

int t=arr[j];

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

arr[j+1]=t;

s=true;

}

}

if(s==false){

break;

}

}

}

}

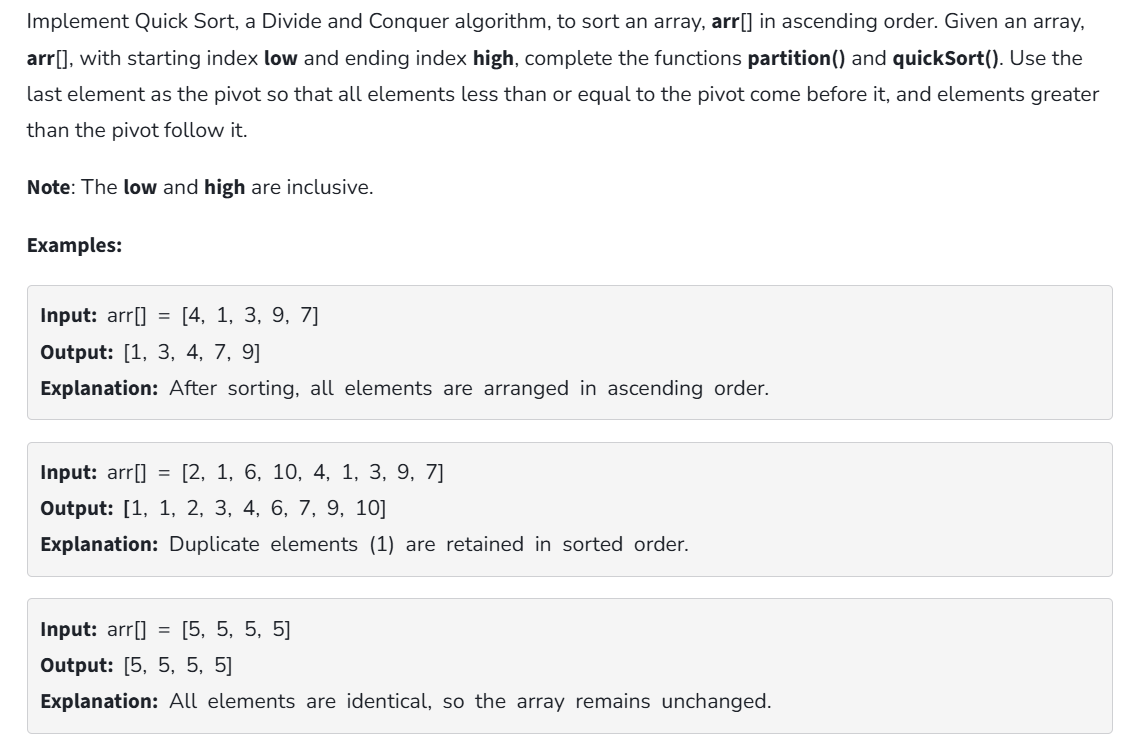
**Output:**



**Time Complexity:** O (n2)

**Space Complexity:** O (1)

**2. Quick Sort**



**Code:**

class QuickSort {

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

int pivot = arr[high];

int i = low - 1;

for (int j = low; j <= high - 1; j++) {

if (arr[j] < pivot) {

i++;

swap(arr, i, j);

}

}

swap(arr, i + 1, high);

return i + 1;

}

static void swap(int[] arr, int i, int j) {

int temp = arr[i];

arr[i] = arr[j];

arr[j] = temp;

}

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 main(String[] args) {

int[] arr = {10, 7, 8, 9, 1, 5};

int n = arr.length;

quickSort(arr, 0, n - 1);

for (int val : arr) {

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

}

}

}

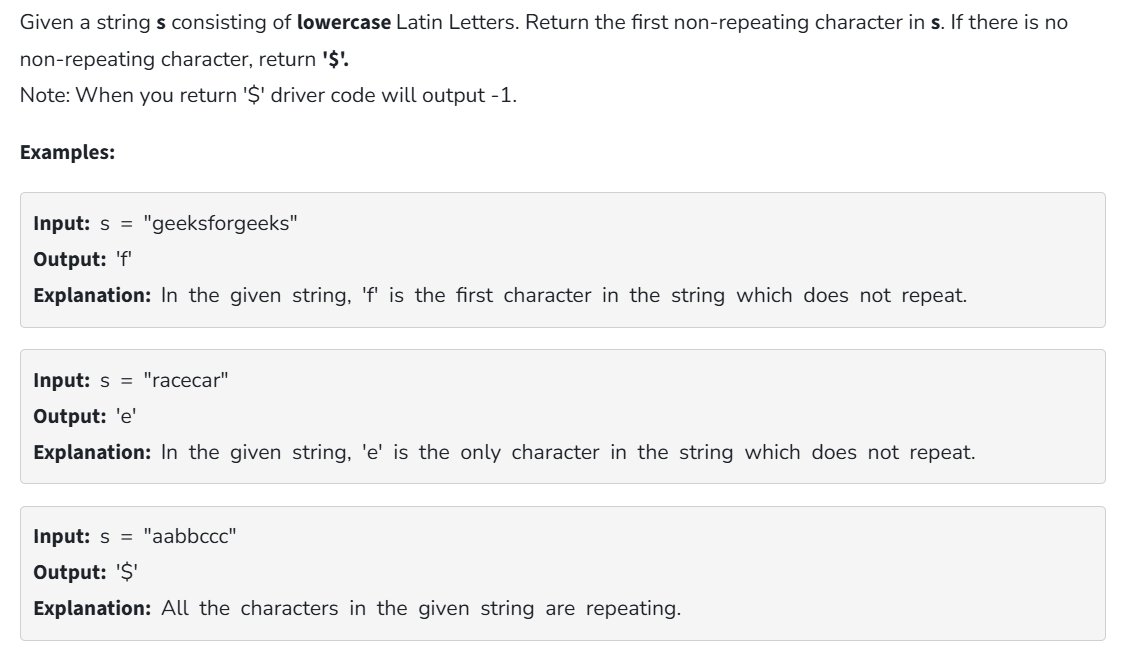
**Output:**



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

**Space Complexity:** O (1)

**3. Non – Repeating Character**



**Code:**

class NonRepeatingCharacter {

static final int MAX\_CHAR = 26;

static char nonRepeatingChar(String s) {

int[] freq = new int[MAX\_CHAR];

for (char c : s.toCharArray())

freq[c - 'a']++;

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

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

return s.charAt(i);

}

return '$';

}

public static void main(String[] args) {

String s = "racecar";

System.out.println(nonRepeatingChar(s));

}

}

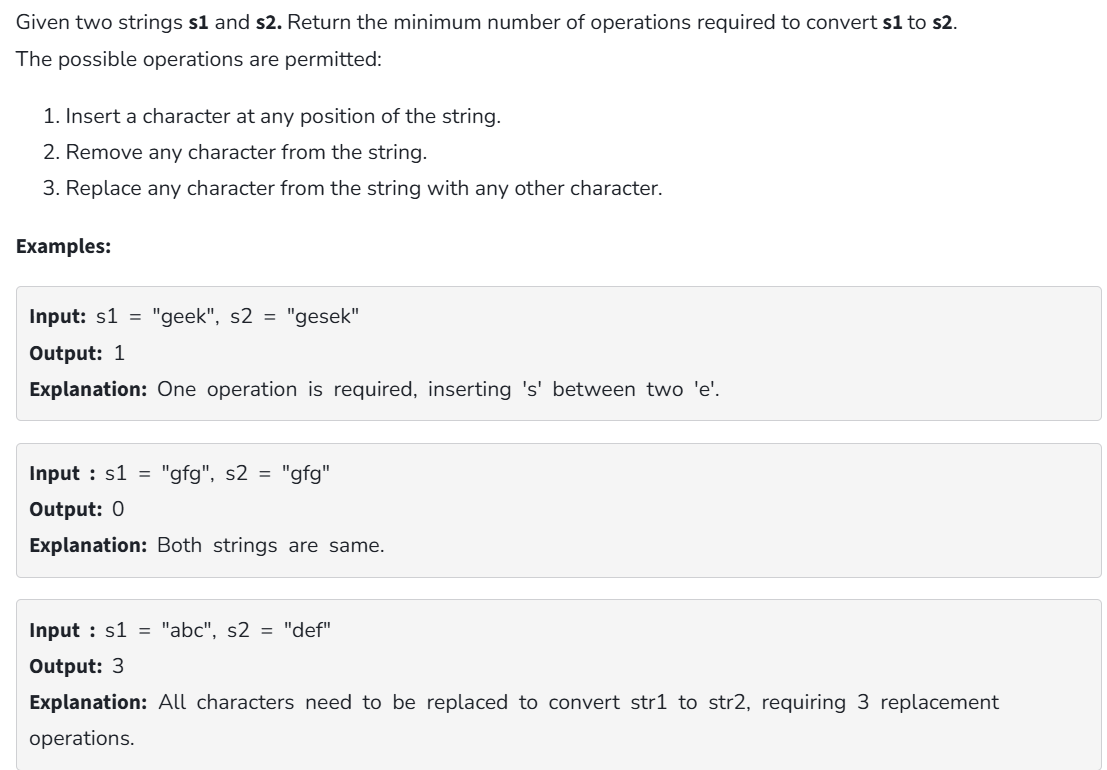
**Output:**



**Time Complexity:** O (n)

**Space Complexity:** O (1)

**4. Edit Distance**

****

**Code:**

public class Solution {

public static int editDistRec(String s1, String s2, int m, int n) {

if (m == 0) return n;

if (n == 0) return m;

if (s1.charAt(m - 1) == s2.charAt(n - 1))

return editDistRec(s1, s2, m - 1, n - 1);

return 1 + Math.min(Math.min(editDistRec(s1, s2, m, n - 1),

editDistRec(s1, s2, m - 1, n)),

editDistRec(s1, s2, m - 1, n - 1));

}

public static int editDist(String s1, String s2) {

return editDistRec(s1, s2, s1.length(), s2.length());

}

public static void main(String[] args) {

String s1 = "GEEXSFRGEEKKS";

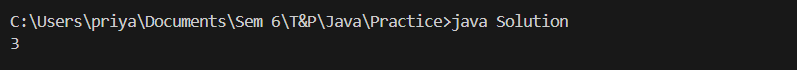
String s2 = "GEEKSFORGEEKS";

System.out.println(editDist(s1, s2));

}

}

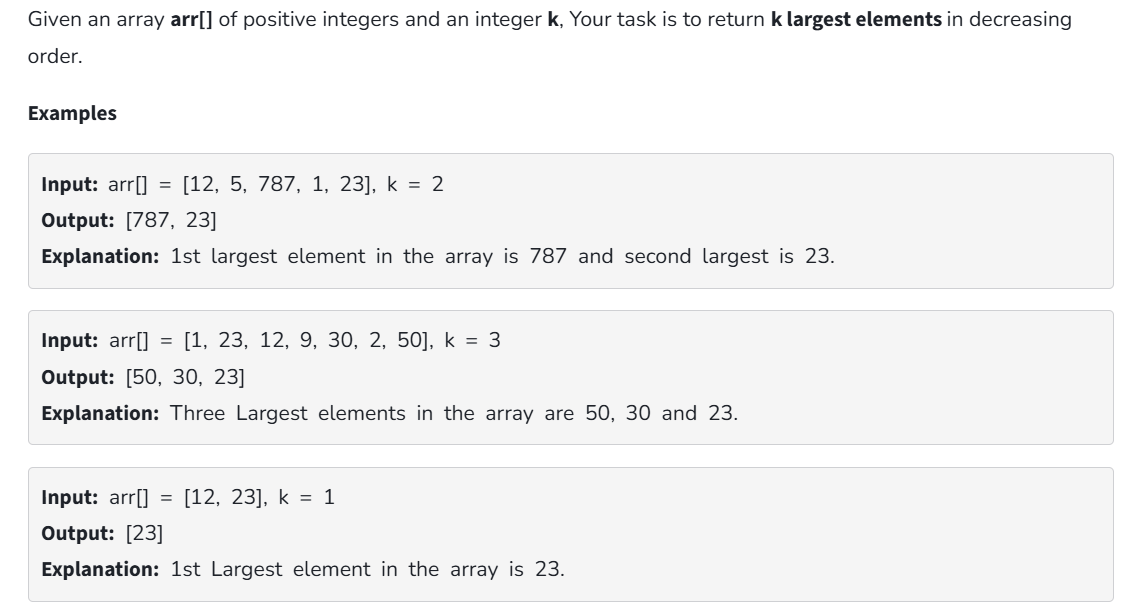
**Output:**



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

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

**5. K – Largest Element**



**Code:**

import java.util.\*;

public class Solution {

public static List<Integer> kLargest(int[] arr, int k) {

PriorityQueue<Integer> minHeap = new PriorityQueue<>(k);

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

minHeap.add(arr[i]);

}

for (int i = k; i < arr.length; i++) {

if (arr[i] > minHeap.peek()) {

minHeap.poll();

minHeap.add(arr[i]);

}

}

List<Integer> res = new ArrayList<>();

while (!minHeap.isEmpty()) {

res.add(minHeap.poll());

}

Collections.reverse(res);

return res;

}

public static void main(String[] args) {

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

int k = 3;

List<Integer> res = kLargest(arr, k);

for (int ele : res) {

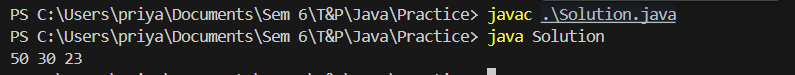
System.out.print(ele + " ");

}

}

}

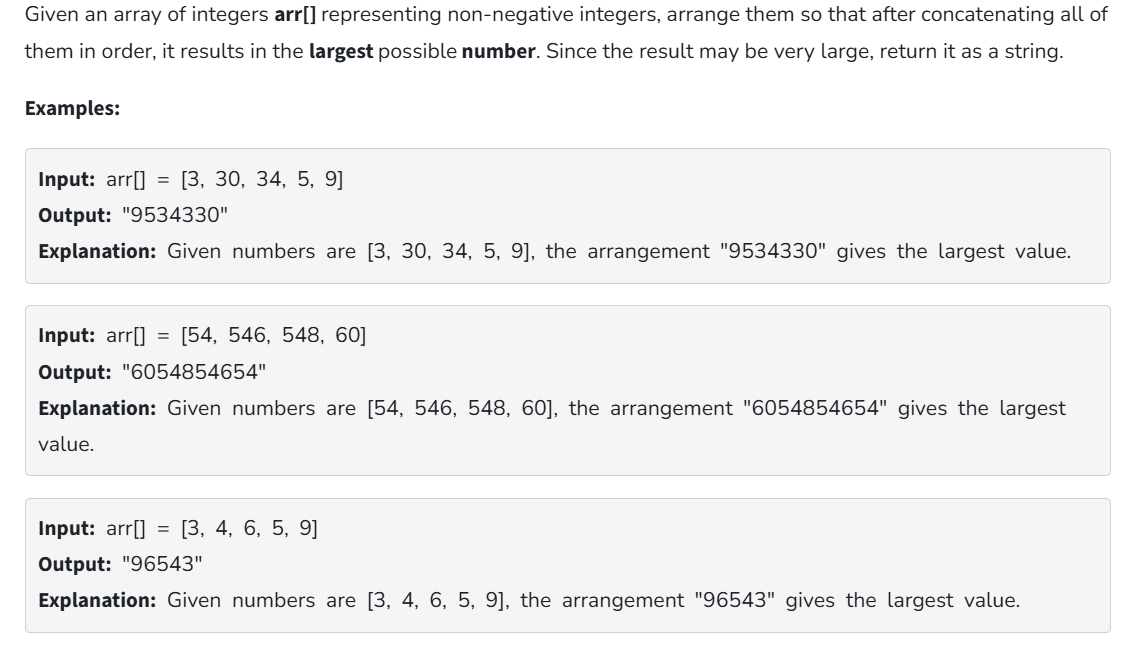
**Output:**



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

**Space Complexity:** O (k)

**6. Form the Largest number**

****

**Code:**

import java.util.Arrays;

public class Solution {

public static String largestNumber(String[] arr) {

Arrays.sort(arr, (a, b) -> (b + a).compareTo(a + b));

if (arr[0].equals("0")) {

return "0";

}

return String.join("", arr);

}

public static void main(String[] args) {

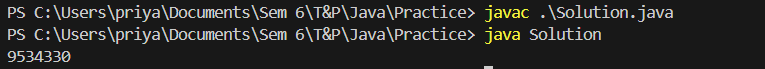
String[] arr = {"3", "30", "34", "5", "9"};

System.out.println(largestNumber(arr));

}

}

**Output:**



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

**Space Complexity:** O (1)