**DATE : 18.11.2024 - DSA PRACTICE**

1. BUBBLE SORT

import java.util.\*;

class BubbleSort {

public void bubbleSort(int[] arr) {

int n = arr.length;

boolean swapped;

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

swapped = false;

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;

swapped = true;

}

}

if (!swapped) {

break;

}

}

}

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 sorter = new BubbleSort();

sorter.bubbleSort(arr);

System.out.println("Sorted array:");

for (int num : arr) {

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

}

}

}

Output:

Enter the number of elements: 5

Enter the elements: 64 34 25 12 22

Sorted array:

12 22 25 34 64

1. Non Repeating Character

import java.util.\*;

class Solution {

public static char firstNonRepeating(String str) {

Map<Character, Integer> map = new LinkedHashMap<>();

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

map.put(str.charAt(i), map.getOrDefault(str.charAt(i), 0) + 1);

}

for (Map.Entry<Character, Integer> entry : map.entrySet()) {

if (entry.getValue() == 1) {

return entry.getKey();

}

}

return '-1';

}

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.print("Enter a string: ");

String str = sc.nextLine();

char result = firstNonRepeating(str);

System.out.println("First non-repeating character: " + result);

sc.close();

}

}

Output:

Enter a string: swiss

First non-repeating character: w

1. K largest Elements

import java.util.\*;

class Solution {

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

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

for (int num : arr) {

minHeap.add(num);

if (minHeap.size() > k) {

minHeap.poll();

}

}

System.out.print("K largest elements: ");

while (!minHeap.isEmpty()) {

System.out.print(minHeap.poll() + " ");

}

}

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.print("Enter the number of elements in the array: ");

int n = sc.nextInt();

int[] arr = new int[n];

System.out.println("Enter the elements of the array: ");

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

arr[i] = sc.nextInt();

}

System.out.print("Enter the value of k: ");

int k = sc.nextInt();

kLargest(arr, k);

sc.close();

}

}

Output:

Enter the number of elements in the array: 6

Enter the elements of the array:

10 4 3 50 5 20

Enter the value of k: 3

K largest elements: 5 10 20

4. **Form largest Number:**

**CODE:**

import java.util.\*;

class Solution {

String printLargest(int[] arr) {

int n = arr.length;

String[] strs = new String[n];

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

strs[i] = String.valueOf(arr[i]);

}

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

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

return "0";

}

StringBuilder largestNum = new StringBuilder();

for (String str : strs) {

largestNum.append(str);

}

return largestNum.toString();

}

}

public class Main {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.println("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();

}

Solution solution = new Solution();

String result = solution.printLargest(arr);

System.out.println("The largest number is: " + result);

}

}

Output:

Enter the number of elements:

5

Enter the elements:

3 30 34 5 9

The largest number is: 9534330

Time Complexity: O(n logn)

Space Complexity : O(n)

**Quick Sort**

CODE:

import java.util.Scanner;

class Solution {

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);

}

}

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 class Main {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.println("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();

}

Solution.quickSort(arr, 0, n - 1);

System.out.println("Sorted array:");

for (int num : arr) {

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

}

}

}

Output:

Enter the number of elements:

6

Enter the elements:

10 7 8 9 1 5

Sorted array:

1 5 7 8 9 10

Time Complexity : O(n log n)

Space Complexity : O(log n)

**Edit Distance**

CODE :

import java.util.Scanner;

class Main1 {

public int editDistance(String str1, String str2) {

int m = str1.length();

int n = str2.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 (str1.charAt(i-1) == str2.charAt(j-1))

dp[i][j] = dp[i-1][j-1];

else

dp[i][j] = Math.min(dp[i-1][j], Math.min(dp[i][j-1], dp[i-1][j-1])) + 1;

}

}

return dp[m][n];

}

}

public class Main {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.println("Enter first string:");

String str1 = scanner.nextLine();

System.out.println("Enter second string:");

String str2 = scanner.nextLine();

Main1 solution = new Main1();

int result = solution.editDistance(str1, str2);

System.out.println("The minimum edit distance is: " + result);

}

}

Output:

Enter first string:

kitten

Enter second string:

Sitting

The minimum edit distance is: 3

Time Complexity : O(m\*n)

Space Complexity : O(m\*n)