DAY 6-GEEKSFORGEEKS

1.Quicksort

class Solution {

static void quickSort(int arr[], int low, int high) {

if (low < high) {

int pivotIndex = partition(arr, low, high);

quickSort(arr, low, pivotIndex - 1);

quickSort(arr, pivotIndex + 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];

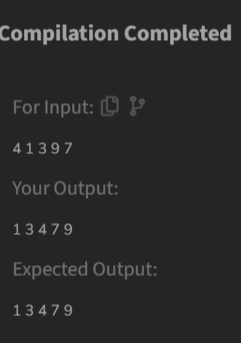
arr[i + 1] = arr[high];

arr[high] = temp;

return i + 1;

}

}



Time complexity: O(nlogn)

2.bubble sort

class Solution {

// Function to sort the array using bubble sort algorithm.

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;

}

}

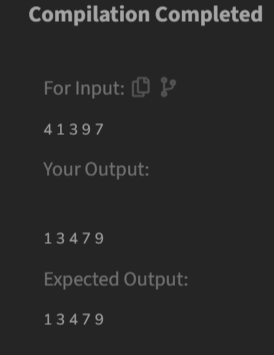
}

System.out.println();

// code here

}

}



3.Non-repeating character

class Solution {

static char nonRepeatingChar(String s) {

HashMap<Character, Integer> charCount = new HashMap<>();

for (char c : s.toCharArray()) {

charCount.put(c, charCount.getOrDefault(c, 0) + 1);

}

for (char c : s.toCharArray()) {

if (charCount.get(c) == 1) {

return c;

}

}

return '$';

}

}



4.Edit Distance

class Solution {

public int editDistance(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++) {

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

if (i == 0) {

dp[i][j] = j;

}

else if (j == 0) {

dp[i][j] = i;

}

else if (s1.charAt(i - 1) == s2.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];

}

}

A screenshot of a phone

Description automatically generated

Time complexity: O(n)

5.Kth largest element

class Solution {

// Function to find the first negative integer in every window of size k

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

int a=new int[k];

Arrays.sort(arr);

int index = 0;

for (int i = arr.length - 1; i >= arr.length - k; i--) {

a[index++] = arr[i];

}

return a;

}

}

6.Form the largest number

class Solution {

String printLargest(int[] arr) {

String[] strArr = new String[arr.length];

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

strArr[i] = Integer.toString(arr[i]);

}

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

@Override

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) {

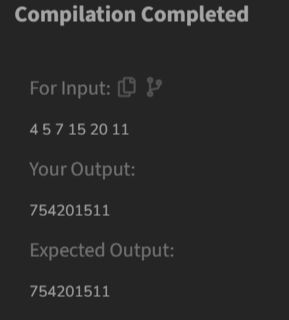
result.append(num);

}

return result.toString();

}

}



Time complexity: O(logn)