DAY-4-13/11/24

GEEKSFORGEEKS

1.Kth Smallest

Code:

class Solution {

public static int kthSmallest(int[] arr, int k) {

PriorityQueue <Integer> heap=new PriorityQueue<>(Collections.reverseOrder());

for(int i:arr){

heap.add(i);

if(heap.size()>k){

heap.poll();

}

}

return heap.peek();

}

}

Output:

A screenshot of a phone

Description automatically generated

Time complexity: O(nlogk)

2.Minimize the Heights II

Code:

class Solution {

int getMinDiff(int[] arr, int k) {

int n = arr.length;

if (n == 1) {

return 0;

}

Arrays.sort(arr);

int initialDifference = arr[n - 1] - arr[0];

int res = initialDifference;

for (int i = 1; i < n; i++) {

int newMax = Math.max(arr[i - 1] + k, arr[n - 1] - k);

int newMin = Math.min(arr[0] + k, arr[i] - k);

res = Math.min(result, newMax - newMin);

}

return res;

}

}

A screenshot of a phone

Description automatically generated

Time complexity: O(nlogn)

3.Parenthesis Checker

Code:

class Solution {

// Function to check if brackets are balanced or not.

static boolean isParenthesisBalanced(String s) {

Stack<character>stack=new Stack<>();

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

if(c=='['||c=='{'||c=='('){

stack.push(c);

}

else if(c==']'||c=='}'||c==')'){

if(stack.isEmpty()){

return false;

}

char top=stack.pop();

if ((ch == '}' && top != '{') ||

(ch == ')' && top != '(') ||

(ch == ']' && top != '[')) {

return false;

}

// code here

}

}

4.Equilibrium Point

Code:

class Solution {

public static int equilibriumPoint(int arr[]) {

int n=arr.length;

if(n==1){

return 1;

}

long tot=0;

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

tot+=arr[i];

}

long lef=0;

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

lef-=arr[i];

if(lef==tot){

return i+1;

}

lef+=arr[i];

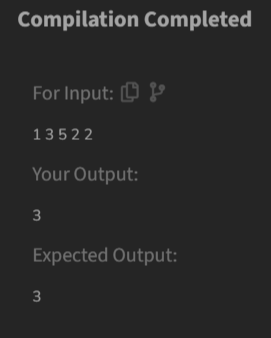
}

return -1;

// code here

}

}



5.Binary search

Code:

class BinarySearch {

public static int binarySearch(int[] arr, int target) {

int left = 0;

int right = arr.length - 1;

while (left <= right) {

int mid = left + (right - left) / 2;

if (arr[mid] == target) {

return mid;

}

if (arr[mid] < target) {

left = mid + 1;

}

else {

right = mid - 1;

}

}

return -1;

}

6. Next greater element

Code:

class Solution {

public ArrayList<Integer> nextLargerElement(int[] arr) {

int n = arr.length;

int[] result = new int[n];

Stack<Integer> stack = new Stack<>();

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

result[i] = -1;

}

for (int i = n - 1; i >= 0; i--) {

while (!stack.isEmpty() && stack.peek() <= arr[i]) {

stack.pop();

}

if (!stack.isEmpty()) {

result[i] = stack.peek();

}

stack.push(arr[i]);

}

ArrayList<Integer> resultList = new ArrayList<>();

for (int num : result) {

resultList.add(num);

}

return resultList;

}

}

A screenshot of a phone

Description automatically generated

Time complexity: O(n)

7.Union of Two Arrays with Duplicate Elements

Code:

class Solution {

public static int findUnion(int a[], int b[]) {

HashSet<Integer> s = new HashSet<>();

for(int i:a){

s.add(i);

}

for(int j:b){

s.add(j);

}

return s.size();

// code here

}

**}**

**A screenshot of a phone

Description automatically generated**

Time complexity: O(n)