Name:Sudheep S Dept:CSE Reg No:22CS!71 Date:13.11.24

1.Kth Smallest Element

import java.util.PriorityQueue;

public class Main{

public static void main(String []args){

int[] arr={10, 5, 4, 3, 48, 6, 2, 33, 53, 10 };

int k=4;

int n=arr.length;

PriorityQueue<Integer> q=new PriorityQueue<>((a, b) -> b - a);

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

q.offer(arr[i]);

if ( q.size() > k ){

q.poll();

}

}

System.out.println(q.peek());

}

}  
  


Time Complexity:O(N \* log(k))

Space Complexity: O(k)

2.Minimize the height II

import java.util.Arrays;

public class Main{

public static void main(String[] args) {

int k = 6;

int[] arr = {12, 6, 4, 15, 17, 10};

int n = arr.length;

Arrays.sort(arr);

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

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

if (arr[i] - k < 0)

continue;

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

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

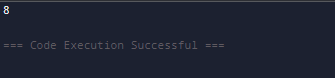
res = Math.min(res, maxH - minH);

}

System.out.println(res);

}

}



3. Parentheses Checker

import java.util.Stack;

public class Main {

public static void main(String[] args) {

String s = "{()}[]";

Stack<Character> st = new Stack<>();

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

if (s.charAt(i) == '{' || s.charAt(i) == '(' || s.charAt(i) == '[') {

st.push(s.charAt(i));

} else if (s.charAt(i) == '}' || s.charAt(i) == ')' || s.charAt(i) == ']') {

if (st.isEmpty()) {

System.out.println("false");

return;

}

if ((st.peek() == '(' && s.charAt(i) == ')') ||

(st.peek() == '{' && s.charAt(i) == '}') ||

(st.peek() == '[' && s.charAt(i) == ']')) {

st.pop();

} else {

System.out.println("false");

return;

}

}

}

if (st.isEmpty()) {

System.out.println("true");

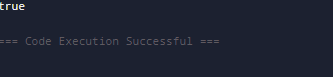
} else {

System.out.println("false");

}

}

}



Time Complexity:O(n)

Space Complexity: O(n)

4. Equilibrium point

import java.util.\*;

public class Main {

public static void main(String[] args) {

long[] a = { -7, 1, 5, 2, -4, 3, 0 };

int n = a.length;

long l, r;

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

l = 0;

for (int j = 0; j < i; j++)

l += a[j];

r = 0;

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

r += a[j];

if (l == r) {

System.out.println(i + 1);

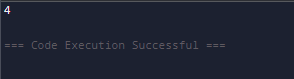
return;

}

}

System.out.println(-1);

}

}  
  


Time complexity:O(n^2) and Space complexity:O(1)

5.Binary Search

class Main {

public static void main(String args[]) {

int a[] = { 2, 3, 4, 10, 40 };

int x = 10;

int l = 0, h = a.length - 1;

int r = -1;

while (l <= h) {

int m = l + (h - l) / 2;

if (a[m] == x) {

r = m;

break;

}

if (a[m] < x)

l = m + 1;

else

h = m - 1;

}

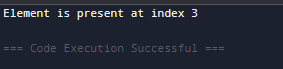
if (r == -1)

System.out.println("Element is not present in array");

else

System.out.println("Element is present at index " + r);

}

}  


Time complexity:O(log n) and Space complexity:O(1)

6.Next Greater Element  
class Main {

public static void main(String args[]) {

int a[] = { 11, 13, 21, 3 };

int n = a.length;

int n1, i, j;

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

n1 = -1;

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

if (a[i] < a[j]) {

n1 = a[j];

break;

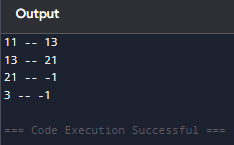
}

}

System.out.println(a[i] + " -- " + n1);

}

}

}  


Time complexity:O(n^2) and Space complexity:O(1)

7. Union of Two Arrays with Duplicate Elements  
  
import java.util.HashSet;

import java.util.Scanner;

class Main{

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

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

        for (int num : a) {

            unionSet.add(num);

        }

        for (int num : b) {

            unionSet.add(num);

        }

        return unionSet.size();

    }

    public static void main(String[] sasta) {

        Scanner scanner = new Scanner(System.in);

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

        int n = scanner.nextInt();

        int[] a = new int[n];

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

        for (int i = 0; i < n; i++) {

            a[i] = scanner.nextInt();

        }

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

        int m = scanner.nextInt();

        int[] b = new int[m];

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

        for (int i = 0; i < m; i++) {

            b[i] = scanner.nextInt();

        }

        Main solution = new Main();

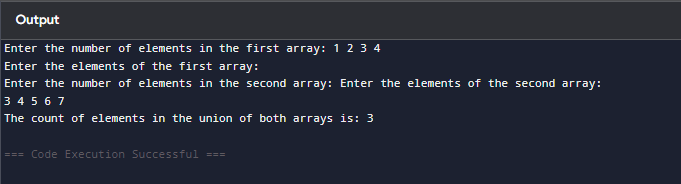
        int unionCount = solution.findUnion(a, b);

        System.out.println("The count of elements in the union of both arrays is: " + unionCount);

        scanner.close();

    }

}

  
time complexity and space complexity : O(n)