**PRACTICE DAY-4**

**Alagu Varshini T**

**CSE**

**1. Binary Search**

import java.io.\*;

class BinarySearch {

int binarysearch(int arr[], int x) {

int low = 0, high = arr.length - 1;

while (low <= high) {

int mid = low + (high - low) / 2;

if (arr[mid] == x) {

return mid;

}

if (arr[mid] < x) {

low = mid + 1;

} else {

high = mid - 1;

}

}

return -1;

}

public static void main(String args[]) {

BinarySearch ob = new BinarySearch();

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

int x = 10;

int result = ob.binarysearch(arr, x);

if (result == -1)

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

else

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

}

}

**Input:**

arr[] = { 2, 3, 4, 10, 40 }

x = 10

**Output:**

Element is present at index 3

**2. Next Greater Element (NGE)**

import java.io.\*;

class NGE {

static void nge(int arr[], int n) {

int i, j;

int next = -1;

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

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

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

next = arr[j];

break;

}

}

System.out.println(arr[i] + " -- " + next);

}

}

public static void main(String[] args) {

int arr[] = { 11, 22, 33, 3 };

int n = arr.length;

nge(arr, n);

}

}

**Input:**

arr[] = { 11, 22, 33, 3 }

**Output:**

11 -- 22

22 -- 33

33 -- -1

3 -- -1

**3. Union of Two Arrays**

import java.util.HashSet;

import java.util.ArrayList;

class Union {

static ArrayList<Integer> findUnion(int[] a, int[] b) {

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

for (int num : a) {

set.add(num);

}

for (int num : b) {

set.add(num);

}

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

for (int i : set) {

result.add(i);

}

return result;

}

public static void main(String[] args) {

int[] a = { 1, 2, 3, 2, 1 };

int[] b = { 3, 2, 2, 3, 3, 2 };

ArrayList<Integer> result = findUnion(a, b);

for (int num : result) {

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

}

}

}

**Input:**

a[] = { 1, 2, 3, 2, 1 }

b[] = { 3, 2, 2, 3, 3, 2 }

**Output:**

1 2 3

**4. Valid Parentheses**

import java.util.Stack;

class Main {

public boolean isValid(String s) {

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

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

if (c == '[') {

st.push(']');

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

st.push('}');

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

st.push(')');

} else if (st.isEmpty() || st.pop() != c) {

return false;

}

}

return st.isEmpty();

}

public static void main(String[] args) {

Main sol = new Main();

System.out.println(sol.isValid("()"));

System.out.println(sol.isValid("()[]{}"));

System.out.println(sol.isValid("(]"));

System.out.println(sol.isValid("([)]"));

System.out.println(sol.isValid("{[]}"));

}

}

**Input:**

s = "()"

**Output:**

true

true

false

false

true

**5. K'th Smallest Element in Unsorted Array**

import java.util.Arrays;

import java.util.Collections;

class Small {

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

Arrays.sort(arr);

return arr[k - 1];

}

public static void main(String[] args) {

Integer arr[] = new Integer[] { 12, 3, 5, 7, 19 };

int k = 2;

System.out.print("K'th smallest element is " + kthSmallest(arr, k));

}

}

**Input:**

arr[] = { 12, 3, 5, 7, 19 }

k = 2

**Output:**

K'th smallest element is 5

**6. Minimize the Maximum Difference Between the Heights**

class minidiffheight {

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

int n = arr.length;

Arrays.sort(arr);

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

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

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

}

return result;

}

public static void main(String[] args) {

int k = 6;

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

int ans = getMinDiff(arr, k);

System.out.println(ans);

}

}

**Input:**

k = 6

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

**Output:**

11

**7. Equilibrium Index of an Array**

import java.util.\*;

public class Main {

public static int equilibriumPoint(long[] arr) {

int n = arr.length;

long leftsum, rightsum;

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

leftsum = 0;

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

leftsum += arr[j];

}

rightsum = 0;

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

rightsum += arr[j];

}

if (leftsum == rightsum)

return i + 1;

}

return -1;

}

public static void main(String[] args) {

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

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

}

}

**Input:**

arr[] = { -7, 1, 5, 2, -4, 3, 0 }

**Output:**

4