Core Java-Assessment10

### ****1. Sliding Window and Two-Pointer Approach****

import java.util.\*;

public class SlidingWindow {

public static void main(String[] args) {

int[] nums = {1, 2, 3, 4, 5, 6, 7};

int k = 3; // Subarray size

// Finding the sum of subarrays of size k

int sum = 0;

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

sum += nums[i];

}

System.out.println("Initial sum: " + sum);

for (int i = k; i < nums.length; i++) {

sum += nums[i] - nums[i - k];

System.out.println("Updated sum: " + sum);

}

}

}

### ****2. Maximum Element in an Array****

public class MaximumElement {

public static void main(String[] args) {

int[] nums = {5, 3, 8, 1, 9, 6};

int max = nums[0];

for (int num : nums) {

if (num > max) {

max = num;

}

}

System.out.println("Maximum Element: " + max);

}

}

### ****3. Superior Elements or Leader Elements in Array****

public class LeaderElements {

public static void main(String[] args) {

int[] arr = {16, 17, 4, 3, 5, 2};

int n = arr.length;

int maxFromRight = arr[n - 1];

System.out.println(maxFromRight);

for (int i = n - 2; i >= 0; i--) {

if (arr[i] > maxFromRight) {

maxFromRight = arr[i];

System.out.println(arr[i]);

}

}

}

}

### ****4. Reverse an Array****

public class ReverseArray {

public static void main(String[] args) {

int[] arr = {1, 2, 3, 4, 5};

int start = 0, end = arr.length - 1;

while (start < end) {

int temp = arr[start];

arr[start] = arr[end];

arr[end] = temp;

start++;

end--;

}

System.out.println("Reversed Array: " + Arrays.toString(arr));

}

}

### ****5. Reverse an Array in Sub-groups of Size k****

public class ReverseInSubGroups {

public static void main(String[] args) {

int[] arr = {1, 2, 3, 4, 5, 6, 7, 8};

int k = 3;

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

int left = i;

int right = Math.min(i + k - 1, arr.length - 1);

while (left < right) {

int temp = arr[left];

arr[left] = arr[right];

arr[right] = temp;

left++;

right--;

}

}

System.out.println("Reversed in subgroups: " + Arrays.toString(arr));

}

}

public class BinarySort {

public static void main(String[] args) {

int[] arr = {0, 1, 1, 0, 1, 0, 1};

int left = 0, right = arr.length - 1;

while (left < right) {

if (arr[left] == 0) {

left++;

} else if (arr[right] == 1) {

right--;

} else {

int temp = arr[left];

arr[left] = arr[right];

arr[right] = temp;

left++;

right--;

}

}

System.out.println("Sorted Array: " + Arrays.toString(arr));

}

}

### ****7. Sort Colors (Dutch National Flag Problem)****

public class SortColors {

public static void main(String[] args) {

int[] arr = {2, 0, 2, 1, 1, 0};

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

while (mid <= high) {

if (arr[mid] == 0) {

int temp = arr[low];

arr[low] = arr[mid];

arr[mid] = temp;

low++;

mid++;

} else if (arr[mid] == 1) {

mid++;

} else {

int temp = arr[mid];

arr[mid] = arr[high];

arr[high] = temp;

high--;

}

}

System.out.println("Sorted Colors: " + Arrays.toString(arr));

}

}

### ****8. Majority Element (Element that appears more than n/2 times)****

**Code Example:**

java

Copy

public class MajorityElement {

public static void main(String[] args) {

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

int candidate = findMajorityElement(arr);

System.out.println("Majority Element: " + candidate);

}

public static int findMajorityElement(int[] arr) {

int count = 0, candidate = -1;

for (int num : arr) {

if (count == 0) {

candidate = num;

}

count += (num == candidate) ? 1 : -1;

}

return candidate;

}

}

### ****9. Game of Thrones - Find the Winner (Majority Element Problem)****

import java.util.HashMap;

public class GameOfThrones {

public static void main(String[] args) {

String[] votes = {"Arya", "Jon", "Arya", "Sansa", "Jon", "Arya"};

String winner = findWinner(votes);

System.out.println("Winner: " + winner);

}

public static String findWinner(String[] votes) {

HashMap<String, Integer> voteCount = new HashMap<>();

for (String vote : votes) {

voteCount.put(vote, voteCount.getOrDefault(vote, 0) + 1);

}

String winner = "";

int maxVotes = 0;

for (String player : voteCount.keySet()) {

if (voteCount.get(player) > maxVotes) {

maxVotes = voteCount.get(player);

winner = player;

}

}

return winner;

}

}

### ****10. Longest Substring Without Repeating Characters****

**Code Example:**

java

Copy

import java.util.HashSet;

public class LongestSubstring {

public static void main(String[] args) {

String s = "abcabcbb";

int length = longestSubstring(s);

System.out.println("Length of longest substring without repeating characters: " + length);

}

public static int longestSubstring(String s) {

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

int maxLength = 0;

int left = 0;

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

while (set.contains(s.charAt(right))) {

set.remove(s.charAt(left));

left++;

}

set.add(s.charAt(right));

maxLength = Math.max(maxLength, right - left + 1);

}

return maxLength;

}

}

### ****11. Pair Sum of Elements or Two Sum****

窗体顶端

import java.util.\*;

public class TwoSum {

public static void main(String[] args) {

int[] arr = {2, 7, 11, 15};

int target = 9;

int[] result = twoSum(arr, target);

System.out.println("Indices: " + Arrays.toString(result));

}

public static int[] twoSum(int[] nums, int target) {

Map<Integer, Integer> map = new HashMap<>();

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

int complement = target - nums[i];

if (map.containsKey(complement)) {

return new int[] { map.get(complement), i };

}

map.put(nums[i], i);

}

return new int[] {}; // return empty array if no solution

}

}