SARAVANAN BS-AI&DS-DSA-PRACTICE-5

1. Stock buy and sell:

class Solution{

//Function to find the days of buying and selling stock for max profit.

ArrayList<ArrayList<Integer> > stockBuySell(int A[], int n) {

// code here

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

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

if(A[i+1] > A[i]){

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

a.add(i);

a.add(i+1);

arr.add(a);

}

}

return arr;

}

}

Result:

Since there can be multiple solutions, the driver code will print 1 if your answer is correct, otherwise, it will return 0. In case there's no profit the driver code will print the string "No Profit" for a correct solution.

N = 7  
A[] = {100,180,260,310,40,535,695}  
Output:  
1

N = 5  
A[] = {4,2,2,2,4}  
Output:  
1

Time complexity: O(N)

2. First and Last Occurrences:

class GFG {

ArrayList<Integer> find(int arr[], int x) {

// code here

ArrayList<Integer> ind = new ArrayList();

ind.add(-1);

ind.add(-1);

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

if(arr[i] == x){

ind.set(0, i);break;}

}

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

if(arr[i] == x){

ind.set(1,i);

break;

}

}

return ind;

}}

Result:

**Input:** arr[] = [1, 3, 5, 5, 5, 5, 67, 123, 125], x = 5  
**Output:** [2, 5]

**Input:** arr[] = [1, 3, 5, 5, 5, 5, 7, 123, 125], x = 7  
**Output:** [6, 6]

**Input:** arr[] = [1, 2, 3], x = 4  
**Output:** [-1, -1]

Time complexity: O(n)

3.Find Transition point

class Solution {

int transitionPoint(int arr[]) {

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

if(arr[i] == 1){

return i;

}

}

return -1;

}

}

Result:

**Input:** arr[] = [0, 0, 0, 1, 1]  
**Output:** 3

**Input:** arr[] = [0, 0, 0, 0]  
**Output:** -1

**Input:** arr[] = [1, 1, 1]  
**Output:** 0

Time Complexity: O(n)

4.First Repeating element

class Solution {

// Function to return the position of the first repeating element.

public static int firstRepeated(int[] arr) {

HashMap<Integer, Integer> hash = new HashMap<>();

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

hash.put(arr[i], hash.getOrDefault(arr[i], 0)+1);

}

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

if(hash.get(arr[i]) > 1){

return i+1;

}

}

return -1;

}

}

Result:

**Input:** arr[] = [0, 0, 0, 1, 1]  
**Output:** 3

**Input:** arr[] = [0, 0, 0, 0]  
**Output:** -1

**Input:** arr[] = [1, 1, 1]  
**Output:** 0

**Input:** arr[] = [0, 1, 1]  
**Output:** 1

Time complexity: O(N)

5.Remove duplicates in sorted array

class Solution {

// Function to remove duplicates from the given array

public int remove\_duplicate(List<Integer> arr) {

// Code Here

int n = arr.size();

int m=0;

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

if(!arr.get(m).equals(arr.get(i))) {

m++;

arr.set(m, arr.get(i));

}

}

return m+1;

}

}

Result:

**Input:** arr = [2, 2, 2, 2, 2]  
**Output:** [2]

**Input:** arr = [1, 2, 4]  
**Output:** [1, 2, 4]

Time complexity: O(N)

6.Maximum Index

class Solution {

// Function to find the maximum index difference.

int maxIndexDiff(int[] a) {

int n = a.length;

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

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

if(st.isEmpty() || a[st.peek()]<a[i]){

st.push(i);

}

}

int i=0;

int max=0;

while(i<n && st.size()>0){

if(a[i]>a[st.peek()]){

i++;

}else{

max= Math.max(st.peek()-i,max);

st.pop();

}

}

return max;

}

}

Result:

**Input:** arr[] = [1, 10]  
**Output:** 1

**Input:** arr[] = [34, 8, 10, 3, 2, 80, 30, 33, 1]  
**Output:** 6

Time complexity: O(N)

7.Wave Array:

class Solution {

public static void convertToWave(int[] arr) {

int n = arr.length;

if(n % 2 == 1){

for(int i = 0; i < arr.length-1; i+=2){

int temp = arr[i];

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

arr[i+1] = temp;

}

}

else{

for(int i = 0; i < arr.length; i+=2){

int temp = arr[i];

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

arr[i+1] = temp;

}

}

}

}

Result:

**Input:** arr[] = [1, 2, 3, 4, 5]  
**Output: [**2, 1, 4, 3, 5]

**Input:** arr[] = [2, 4, 7, 8, 9, 10]  
**Output: [**4, 2, 8, 7, 10, 9]

Time comlexity: O(N)