14-11-2024 DSA PROBLEMS PRACTICE - 5

1.BUY AND SELL STOCK - II

package Problems;

public class BuyandSell2 {

public int maxProfit(int a[]) {

int profit = 0;

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

if(a[i] > a[i-1]) {

profit += a[i]-a[i-1];

}

} return profit;

}

public static void main(String[] args) {

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

// ClassName object Name = new ClassName();

BuyandSell2 solution = new BuyandSell2();

// Creating instance

int result = solution.maxProfit(prices);

System.***out***.println("The max profit of stocks II: " + result);

}

}

O/P: The Max profit of stocks II: 13

Time Complexity: O(n)

Space Complexity: O(1)

2.FirstLastOccurence

package Problems;

public class FirstLastOccurence {

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

int first = *binarySearchFirst*(arr,target);

int last = *binarySearchLast*(arr,target);

return new int[]{first,last};

}

private static int binarySearchFirst(int[] arr, int target) {

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

int firstOccurence = -1;

while(left<=right) {

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

if (arr[mid]== target){

firstOccurence = mid;

right = mid - 1;

}

else if(arr[mid]<target) {

left = mid +1;

}

else {

right = mid - 1;

}

}

return firstOccurence;

}

private static int binarySearchLast(int[] arr, int target) {

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

int lastOccurence = -1;

while(left<=right) {

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

if (arr[mid]== target){

lastOccurence = mid;

left = mid + 1;

}

else if(arr[mid]<target) {

right = mid - 1;

}

else {

left = mid + 1;

}

}

return lastOccurence;

}

public static void main(String[] args) {

int[] arr = {2, 4, 10, 10, 10, 18, 20};

int target = 10;

int[] result = *findFirstAndLast*(arr, target);

System.***out***.println("First Occurrence: " + result[0]);

System.***out***.println("Last Occurrence: " + result[1]);

}

}

Op:

First Occurrence: 2

Last Occurrence: 3

Time Complexity: O(logn)

Space Complexity: O(1)

3.Return TransistionPoint  
  
package Problems;

public class TransistionPoint {

public static int tp(int arr[],int n) {

int beg = 0;

int end = n-1;

int flag = 0;//to denote that if the array contain no transition point

if (n ==1 && arr[0] ==0) {

return -1 ;

}

while(beg<=end)

{

int mid = (beg+end)/2;

if (arr[mid] == 1 && arr[mid-1] == 0)

{

return mid;

}

else if(arr[mid] == 1)

{

flag = 1;

end = mid -1 ;

}

else

{

beg = mid + 1;

}

}

if(flag == 0) {

return -1;

}return flag;

}

public static void main(String[] args) {

int n = 5;

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

int result = *tp*(arr,n);

System.***out***.println("The transition point is: "+ result);

}

}

OP: The transition point is: 3

Time Complexity: O(logn)

Space Complexity: O(1)

4.Maximum Index  
  
package Problems;

public class MaximumIndex {

public int MaxiIndex(int[] arr) {

int curr\_max = arr[0];

int maxi\_ind = 0;

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

if(curr\_max < arr[i]) {

curr\_max = arr[i];

maxi\_ind = i;

}

}return maxi\_ind;

}

public static void main(String[] args) {

int[] arr = {9,3,3,98,6,6,7,9};

MaximumIndex obj = new MaximumIndex();

int result = obj.MaxiIndex(arr);

System.***out***.println("The Maximum Index is : "+ result);

}

}

Time Complexity: O(n)

Space Complexity: O(1)

5.WAVEFORM

package Problems;

import java.util.Arrays;

public class WaveForm {

public static int[] Waves(int[] arr) {

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

if( i > 0 && arr[i]<arr[i-1]) {

*swap*(arr,i,i-1);

}

if(i<arr.length-1&& arr[i]<arr[i+1]) {

*swap*(arr,i,i+1);

}

}return arr;

}

private static void swap(int[] arr,int i,int j) {

int temp = arr[i];

arr[i] = arr[j];

arr[j] = temp;

}

public static void main(String[] args) {

int[] arr= {0,9,8,10,12,5,4,3};

int[] result = WaveForm.*Waves*(arr);

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

}

}

OP:  
Waveform : [9, 0, 10, 8, 12, 4, 5, 3]

Time Complexity: O(logn)

Space Complexity: O(1)