Date: 14/11/2024

Practice set 5:

1. Wave Array:

Java code:

import java.util.Arrays;

public class WaveArray {

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

        Arrays.sort(arr);

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

            if (i!=arr.length-1){

                int temp=arr[i];

                arr[i]=arr[i+1];

                arr[i+1]=temp;

            }

        }

        return arr;

    }

    public static void main(String[] args) {

        int[] arr1={1, 2, 3, 4, 5};

        int[] arr2={2, 4, 7, 8, 9, 10};

        System.out.println(Arrays.toString(sort(arr1)));

        System.out.println(Arrays.toString(sort(arr2)));

    }

}

A black screen with blue and white text

Description automatically generatedOutput:

Time complexity: O(n log n)

Space complexity: O(1)

1. First and Last Occurrences:

Java code:

import java.util.ArrayList;;

public class FirstAndLastOccurance {

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

        int a=-1,b=-1;

        int low=0,high=arr.length-1;

        while (low<=high){

            if (arr[low]==x){

                a=low;

            }else{

                low++;

            }

            if (arr[high]==x){

                b=high;

            }else{

                high--;

            }

            if (a!=-1 && b!=-1){

                break;

            }

        }

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

        r.add(a);

        r.add(b);

        return r;

    }

    public static void main(String[] args) {

        int[] arr1={1, 3, 5, 5, 5, 5, 67, 123, 125};

        int[] arr3={1, 3, 5, 5, 5, 5, 7, 123, 125};

        int[] arr2={1, 2, 3};

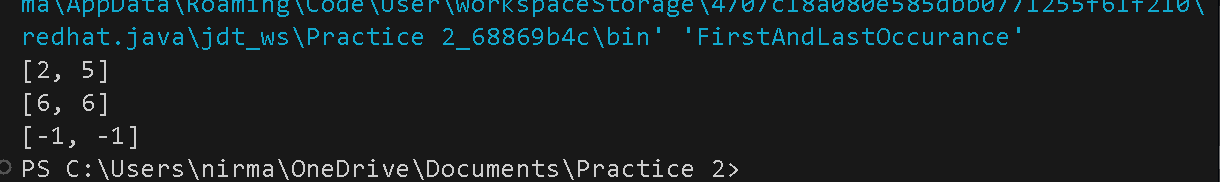
        System.out.println(find(arr1, 5));

        System.out.println(find(arr3, 7));

        System.out.println(find(arr2, 4));

    }

}

Output:

Time complexity: O(n)

Space complexity: O(1)

1. Find Transition Point:

Java code:

public class FindTransitionPoint {

    public static int transitionPoint(int arr[]) {

        int low=0,high=arr.length-1;

        int r=-1;

        while (low<=high){

            int mid=low+(high-low)/2;

            if (arr[mid]==1){

                r=mid;

                high=mid-1;

            }else{

                low=mid+1;

            }

        }

        return r;

    }

    public static void main(String[] args) {

        int[] arr1={0, 0, 0, 1, 1};

        int[] arr2={0, 0, 0, 0};

        int [] arr3={1, 1, 1};

        int[] arr4={0, 1, 1};

        System.out.println(transitionPoint(arr1));

        System.out.println(transitionPoint(arr2));

        System.out.println(transitionPoint(arr3));

        System.out.println(transitionPoint(arr4));

    }

}

A screen shot of a computer

Description automatically generatedOutput:

Time complexity: O(log n)

Space complexity O(1)

1. Find Repeating Element:

Java code:

import java.util.HashMap;

public class FindRepeatingElement {

    public static int find(int[] arr){

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

        for (int i:arr){

            h.put(i,h.getOrDefault(i,0)+1);

        }

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

            if (h.get(arr[j])>1){

                return j+1;

            }

        }

        return -1;

    }

    public static void main(String[] args) {

        int[] arr1={1, 5, 3, 4, 3, 5, 6};

        int[] arr2={1, 2, 3, 4};

        System.out.println(find(arr1));

        System.out.println(find(arr2));

    }

}

A screen shot of a computer

Description automatically generatedOutput:

Time complexity: O(n)

Space complexity: O(n)

1. Remove Duplicates Sorted Array:

Java code:

import java.util.ArrayList;

import java.util.Arrays;

import java.util.List;

public class RemoveFromSortedArray {

    public static int remove(List<Integer> arr){

        int n=arr.size();

        int r=1;

        if (n<=1){

            return n;

        }

        for (int i=1;i<n;i++){

            if (!arr.get(i).equals(arr.get(i-1))){

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

                r++;

            }

        }

        return r;

    }

    public static void main(String[] args) {

        List<Integer> arr1=new ArrayList<>(Arrays.asList(2, 2, 2, 2, 2));

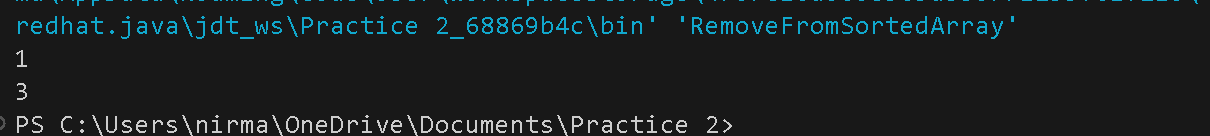
        List<Integer> arr2=new ArrayList<>(Arrays.asList(1, 2, 4));

        System.out.println(remove(arr1));

        System.out.println(remove(arr2));

    }

}

Output:

Time complexity: O(n)

Space complexity: O(1)

1. Maximum Index:

Java code:

public class MaximumIndex {

    public static int res(int[] arr){

        int n=arr.length;

        int[] a=new int[n];

        int[] b=new int[n];

        a[0]=arr[0];

        for (int i=1;i<n;i++){

            a[i]=Math.min(arr[i],a[i-1]);

        }

        b[n-1]=arr[n-1];

        for (int j=n-2;j>=0;j--){

            b[j]=Math.max(arr[j],b[j+1]);

        }

        int i=0,j=0;

        int r=-1;

        while (i<n && j<n){

            if (a[i]<=b[j]){

                r=Math.max(r,j-i);

                j++;

            }else{

                i++;

            }

        }

        return r;

    }

    public static void main(String[] args) {

        int[] arr1={1, 10};

        int[] arr2={34, 8, 10, 3, 2, 80, 30, 33, 1};

        System.out.println(res(arr1));

        System.out.println(res(arr2));

    }

}

A screen shot of a computer

Description automatically generatedOutput:

Time complexity: O( n)

Space complexity: O(n)

1. Coin Change (Count Ways):

Java code:

public class CountCoins {

    public static int count(int coins[], int sum) {

        int[] dp=new int[sum+1];

        dp[0]=1;

        for (int i:coins){

            for (int j=i;j<=sum;j++){

                dp[j]+=dp[j-i];

            }

        }

        return dp[sum];

    }

    public static void main(String[] args) {

        int[] arr1={1, 2, 3};

        int[] arr2={2, 5, 3, 6};

        int[] arr3={5, 10};

        int s1=4,s2=10,s3=3;

        System.out.println(count(arr1, s1));

        System.out.println(count(arr2, s2));

        System.out.println(count(arr3, s3));

    }

}

A screen shot of a computer

Description automatically generatedOutput:

Time complexity: O(mxsum)

Space complexity: O(sum)

1. Stock buy and sell:

Java code:

import java.util.ArrayList;

public class StockBuyAndSell {

    public static ArrayList<ArrayList<Integer>> stock(int A[], int n) {

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

        int i=0;

        while (i<n-1){

            while (i<n-1 && A[i]>=A[i+1]){

                i++;

            }

            if (i==n-1){

                break;

            }

            int b=i;

            i++;

            while (i<n && A[i]>=A[i-1]){

                i++;

            }

            int s=i-1;

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

            p.add(b);

            p.add(s);

            r.add(p);

        }

        return r;

    }

    public static void main(String[] args) {

        int[] arr1={100,180,260,310,40,535,695};

        int[] arr2={4,2,2,2,4};

        int n1=7,n2=5;

        System.out.println(stock(arr1, n1));

        System.out.println(stock(arr2, n2));

    }

}

A screen shot of a computer

Description automatically generatedOutput:

Time complexity: O(n)

Space complexity: O(n)