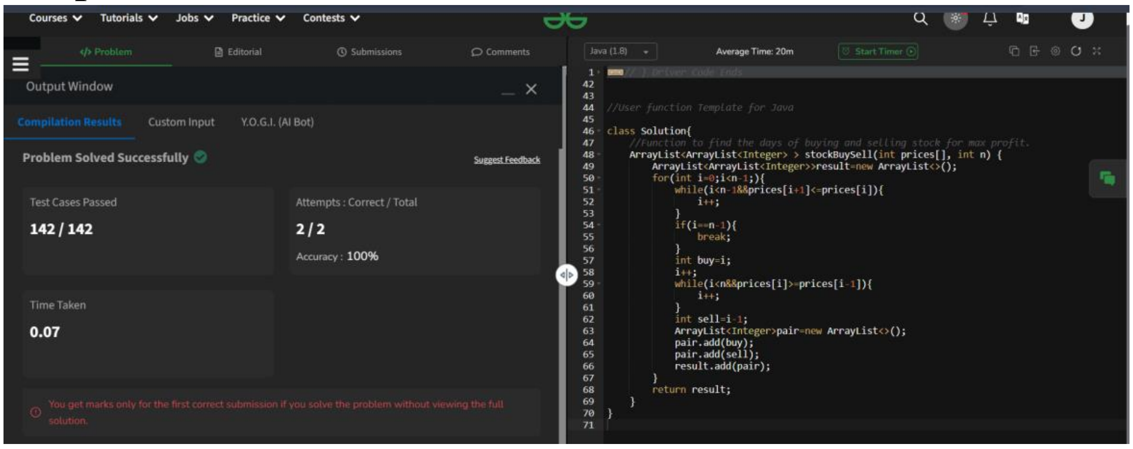
DSA PRACTICE QUESTIONS- DAY 5

Name: Jeevitha R
Reg No: 22IT040
Date: 14/11/2024

1. Stock buy and sell

```
class Solution{
    ArrayList<ArrayList<Integer> > stockBuySell(int prices[], int n) {
        ArrayList<ArrayList<Integer>>result=new ArrayList<>();
        for(int i=0;i<n-1;){
            while(i<n-1&&prices[i+1]<=prices[i]){
                i++;
            }
            if(i==n-1){
                break;
            }
            int buy=i;
            i++;
            while(i<n&&prices[i]>=prices[i-1]){
                i++;
            }
            int sell=i-1;
            ArrayList<Integer>pair=new ArrayList<>();
            pair.add(buy);
            pair.add(sell);
            result.add(pair);
        }
        return result;
    }
}
```

Output



Output Window

Compilation Results   Custom Input   Y.O.G.I. (AI Bot)

Problem Solved Successfully

Suggest Feedback

Test Cases Passed
142 / 142

Attempts : Correct / Total
2 / 2
Accuracy : 100%

Time Taken
0.07

You get marks only for the first correct submission if you solve the problem without viewing the full solution.

Average Time: 20m

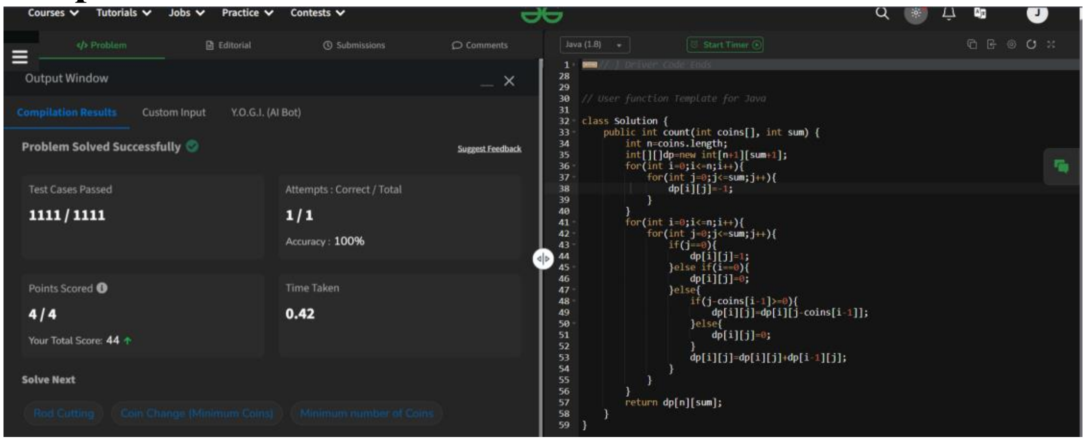
```
// } Driver Code Ends


//User function Template for Java

class Solution{
    //Function to find the days of buying and selling stock for max profit.
    ArrayList<ArrayList<Integer> > stockBuySell(int prices[], int n) {
        ArrayList<ArrayList<Integer>>result=new ArrayList<>();
        for(int i=0;i<n-1;){
            while(i<n-1&&prices[i+1]<=prices[i]){
                i++;
            }
            if(i==n-1){
                break;
            }
            int buy=i;
            i++;
            while(i<n&&prices[i]>=prices[i-1]){
                i++;
            }
            int sell=i-1;
            ArrayList<Integer>pair=new ArrayList<>();
            pair.add(buy);
            pair.add(sell);
            result.add(pair);
        }
        return result;
    }
}
```

Time complexity: O(n)
Space complexity: O(n)

2.Coin change (Count ways)

```
class Solution {
    public int count(int coins[], int sum) {
        int n=coins.length;
        int[][]dp=new int[n+1][sum+1];
        for(int i=0;i<=n;i++){
            for(int j=0;j<=sum;j++){
                dp[i][j]=-1;
            }
        }
        for(int i=0;i<=n;i++){
            for(int j=0;j<=sum;j++){
                if(j==0){
                    dp[i][j]=1;
                }else if(i==0){
                    dp[i][j]=0;
                }else{
                    if(j-coins[i-1]>=0){
                        dp[i][j]=dp[i][j-coins[i-1]];
                    }else{
                        dp[i][j]=0;
                    }
                    dp[i][j]=dp[i][j]+dp[i-1][j];
                }
            }
        }
        return dp[n][sum];
    }
}
```

Output



Output Window

Problem Solved Successfully

Test Cases Passed
1111 / 1111

Attempts : Correct / Total
1 / 1
Accuracy : 100%

Points Scored
4 / 4
Your Total Score: 44

Time Taken
0.42

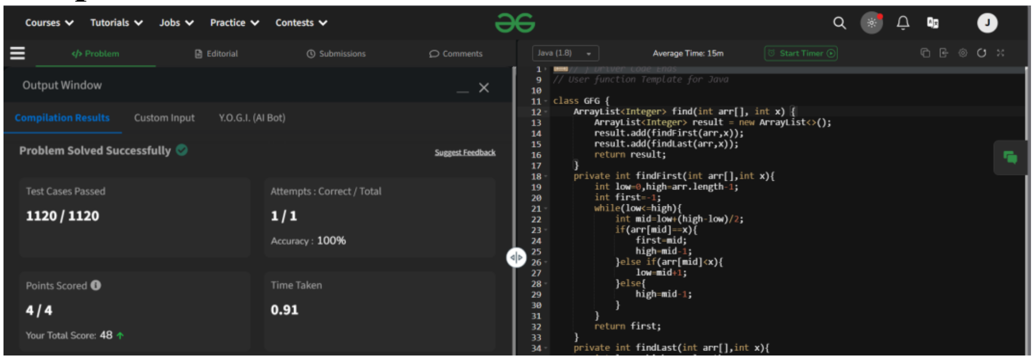
```
// } Driver Code Ends


// User function Template for Java

class Solution {
    public int count(int coins[], int sum) {
        int n=coins.length;
        int[][]dp=new int[n+1][sum+1];
        for(int i=0;i<=n;i++){
            for(int j=0;j<=sum;j++){
                dp[i][j]=-1;
            }
        }
        for(int i=0;i<=n;i++){
            for(int j=0;j<=sum;j++){
                if(j==0){
                    dp[i][j]=1;
                }else if(i==0){
                    dp[i][j]=0;
                }else{
                    if(j-coins[i-1]>=0){
                        dp[i][j]=dp[i][j-coins[i-1]];
                    }else{
                        dp[i][j]=0;
                    }
                    dp[i][j]=dp[i][j]+dp[i-1][j];
                }
            }
        }
        return dp[n][sum];
    }
}
```

Time complexity: O(n *sum)
Space complexity: O(n*sum)

3.First and last Occurrences

```
class GFG {
   ArrayList<Integer> find(int arr[], int x) {
      ArrayList<Integer> result = new ArrayList<>();
      result.add(findFirst(arr,x));
      result.add(findLast(arr,x));
      return result;
   }
   private int findFirst(int arr[],int x){
      int low=0,high=arr.length-1;
      int first=-1;
      while(low<=high){
         int mid=low+(high-low)/2;
         if(arr[mid]==x){
            first=mid;
            high=mid-1;
         }else if(arr[mid]<x){
            low=mid+1;
         }else{
            high=mid-1;
         }
      }
      return first;
   }
   private int findLast(int arr[],int x){
      int low=0,high=arr.length-1;
      int last=-1;
      while(low<=high){
         int mid=low+(high-low)/2;
         if(arr[mid]==x){
            last=mid;
            low=mid+1;
         }else if(arr[mid]<x){
            low=mid+1;
         }else{
            high=mid-1;
         }
      }
      return last;
   }
}
```

Output



Problem Solved Successfully

Test Cases Passed
1120 / 1120

Attempts : Correct / Total
1 / 1
Accuracy : 100%

Points Scored
4 / 4
Your Total Score: 48

Time Taken
0.91

Average Time: 15m

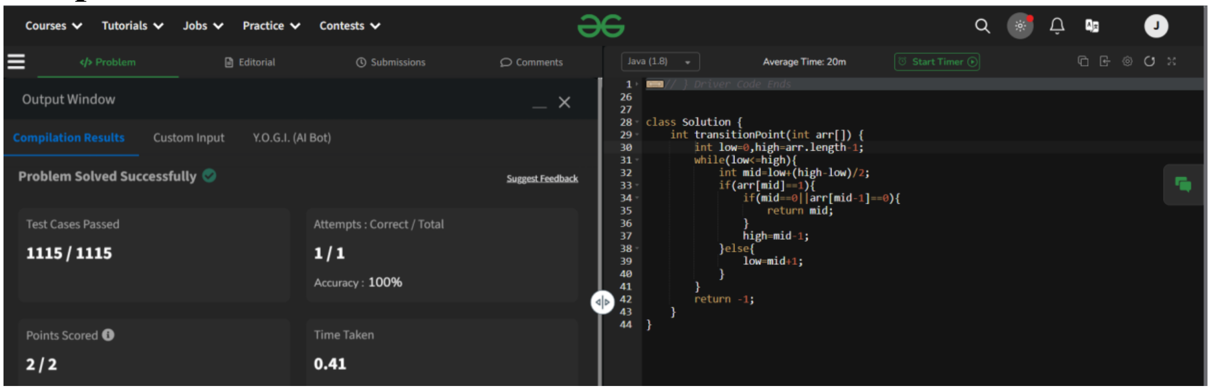
```
// } Driver Code Ends
// User function Template for Java

class GFG {
    ArrayList<Integer> find(int arr[], int x) {
        ArrayList<Integer> result = new ArrayList<>();
        result.add(findFirst(arr,x));
        result.add(findLast(arr,x));
        return result;
    }
    private int findFirst(int arr[],int x){
        int low=0,high=arr.length-1;
        int first=-1;
        while(low<=high){
            int mid=low+(high-low)/2;
            if(arr[mid]==x){
                first=mid;
                high=mid-1;
            }else if(arr[mid]<x){
                low=mid+1;
            }else{
                high=mid-1;
            }
        }
        return first;
    }
    private int findLast(int arr[],int x){
```

Time complexity: O(log N)
Space complexity: O(N)

4.Find transition point

```
class Solution {
    int transitionPoint(int arr[]) {
        int low=0,high=arr.length-1;
        while(low<=high){
            int mid=low+(high-low)/2;
            if(arr[mid]==1){
                if(mid==0||arr[mid-1]==0){
                    return mid;
                }
                high=mid-1;
            }else{
                low=mid+1;
            }
        }
        return -1;
    }
}
```

Output



Output Window

Compilation Results   Custom Input   Y.O.G.I. (AI Bot)

Problem Solved Successfully   Suggest Feedback

Test Cases Passed
1115 / 1115

Attempts : Correct / Total
1 / 1
Accuracy : 100%

Points Scored
2 / 2

Time Taken
0.41

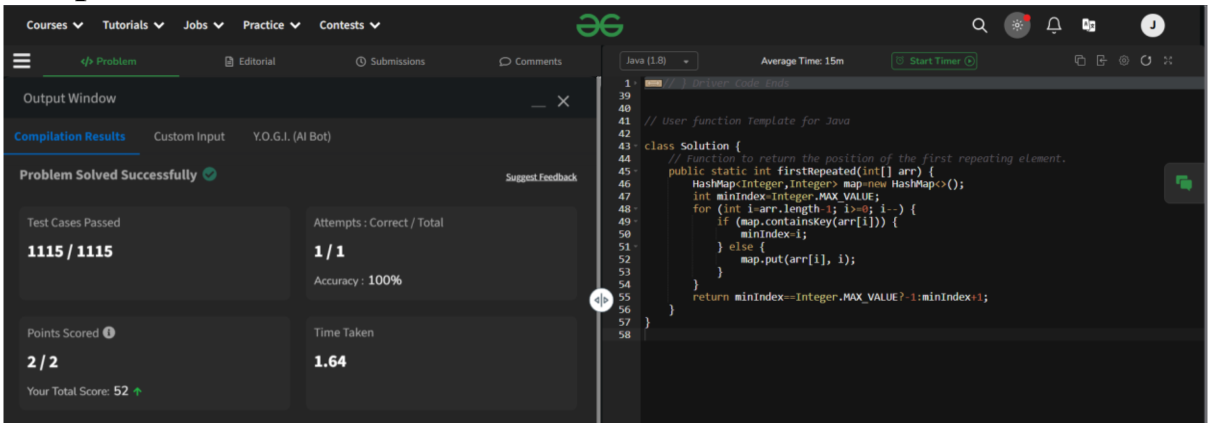
```
// } Driver Code Ends


class Solution {
    int transitionPoint(int arr[]) {
        int low=0,high=arr.length-1;
        while(low<=high){
            int mid=low+(high-low)/2;
            if(arr[mid]==1){
                if(mid==0||arr[mid-1]==0){
                    return mid;
                }
                high=mid-1;
            }else{
                low=mid+1;
            }
        }
        return -1;
    }
}
```

Time complexity: O(log N)
Space complexity: O(N)

5.First repeating element

```
class Solution {
    // Function to return the position of the first repeating element.
    public static int firstRepeated(int[] arr) {
        HashMap<Integer,Integer> map=new HashMap<>();
        int minIndex=Integer.MAX_VALUE;
        for (int i=arr.length-1; i>=0; i--) {
            if (map.containsKey(arr[i])) {
                minIndex=i;
            } else {
                map.put(arr[i], i);
            }
        }
        return minIndex==Integer.MAX_VALUE?-1:minIndex+1;
    }
}
```

Output



Average Time: 15m

Output Window

Compilation Results | Custom Input | Y.O.G.I. (AI Bot)

Problem Solved Successfully

Suggest Feedback

Test Cases Passed
1115 / 1115

Attempts : Correct / Total
1 / 1
Accuracy : 100%

Points Scored
2 / 2
Your Total Score: 52

Time Taken
1.64

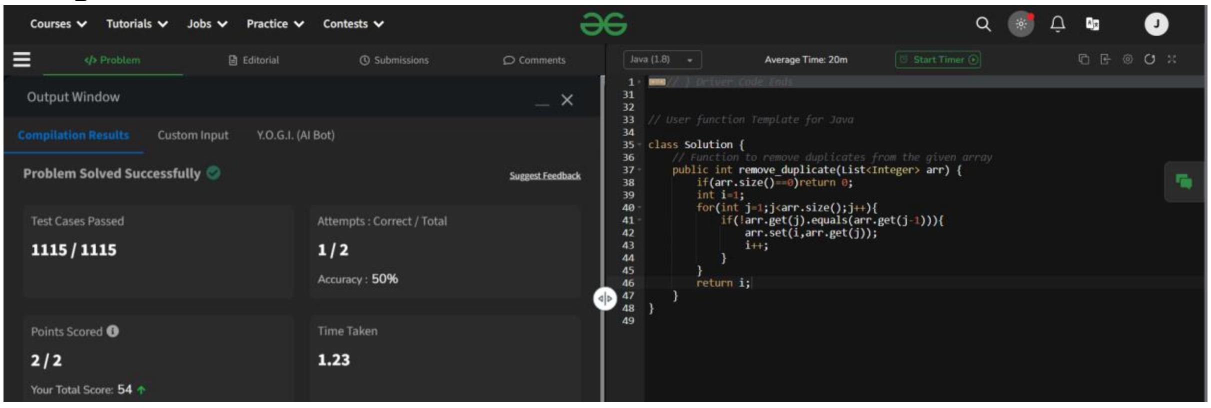
```
// } Driver Code Ends


// User function Template for Java

class Solution {
    // Function to return the position of the first repeating element.
    public static int firstRepeated(int[] arr) {
        HashMap<Integer,Integer> map=new HashMap<>();
        int minIndex=Integer.MAX_VALUE;
        for (int i=arr.length-1; i>=0; i--) {
            if (map.containsKey(arr[i])) {
                minIndex=i;
            } else {
                map.put(arr[i], i);
            }
        }
        return minIndex==Integer.MAX_VALUE?-1:minIndex+1;
    }
}
```

Time complexity: O(N)
Space complexity: O(N)

6.Remove duplicate sorted array

```
class Solution {
    // Function to remove duplicates from the given array
    public int remove_duplicate(List<Integer> arr) {
        if(arr.size()==0) return 0;
        int i=1;
        for(int j=1;j<arr.size();j++){
            if(!arr.get(j).equals(arr.get(j-1))){
                arr.set(i, arr.get(j));
                i++;
            }
        }
        return i;
    }
    }
```

Output:



Problem Solved Successfully

Test Cases Passed
1115 / 1115

Attempts : Correct / Total
1 / 2
Accuracy : 50%

Points Scored
2 / 2
Your Total Score: 54

Time Taken
1.23

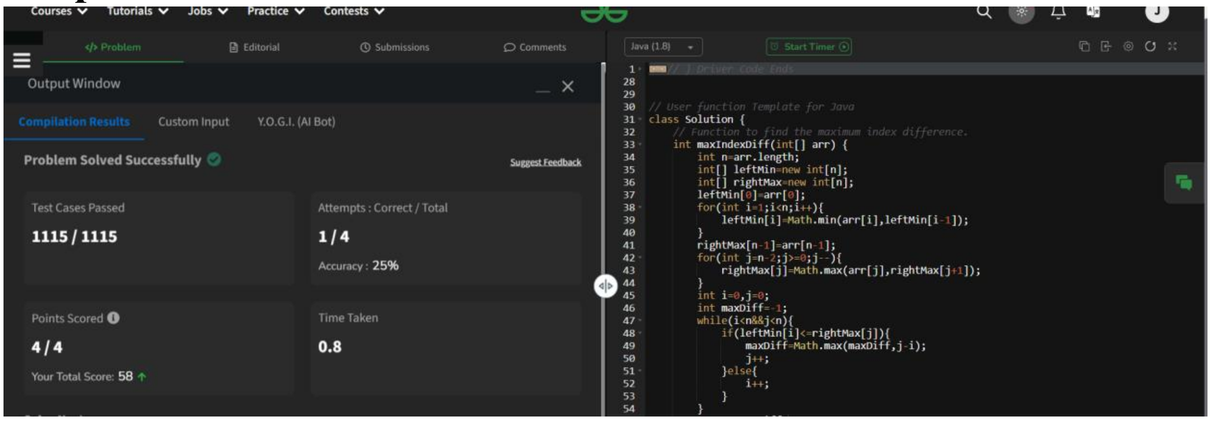
```
// } Driver Code Ends


// User function Template for Java

class Solution {
    // Function to remove duplicates from the given array
    public int remove_duplicate(List<Integer> arr) {
        if(arr.size()==0)return 0;
        int i=1;
        for(int j=1;j<arr.size();j++){
            if(!arr.get(j).equals(arr.get(j-1))){
                arr.set(i,arr.get(j));
                i++;
            }
        }
        return i;
    }
}
```

Time complexity: O(N)
Space complexity: O(1)

7. Maximum index

```
class Solution {
    // Function to find the maximum index difference.
    int maxIndexDiff(int[] arr) {
        int n=arr.length;
        int[] leftMin=new int[n];
        int[] rightMax=new int[n];
        leftMin[0]=arr[0];
        for(int i=1;i<n;i++){
            leftMin[i]=Math.min(arr[i],leftMin[i-1]);
        }
        rightMax[n-1]=arr[n-1];
        for(int j=n-2;j>=0;j--){
            rightMax[j]=Math.max(arr[j],rightMax[j+1]);
        }
        int i=0,j=0;
        int maxDiff=-1;
        while(i<n&&j<n){
            if(leftMin[i]<=rightMax[j]){
                maxDiff=Math.max(maxDiff,j-i);
                j++;
            }else{
                i++;
            }
        }
        return maxDiff;
    }
}
```

Output



Problem Solved Successfully

Test Cases Passed
1115 / 1115

Attempts : Correct / Total
1 / 4
Accuracy : 25%

Points Scored
4 / 4
Your Total Score: 58

Time Taken
0.8

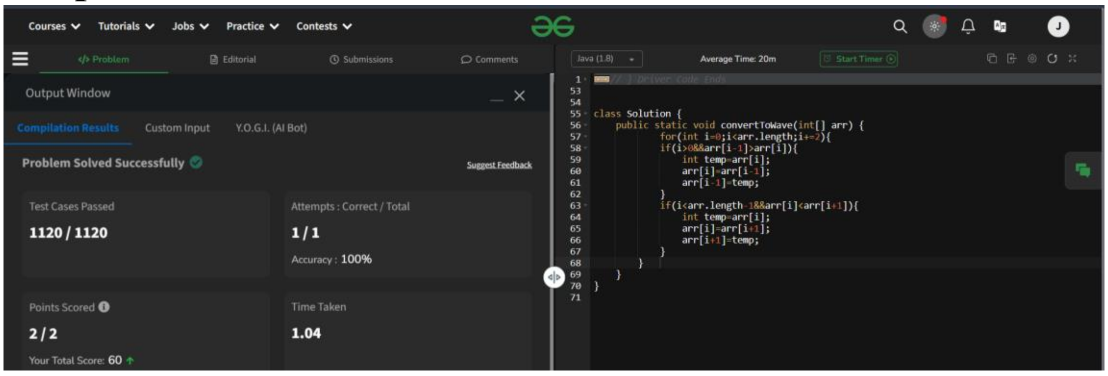
```
// } Driver Code Ends


// User function Template for Java
class Solution {
    // Function to find the maximum index difference.
    int maxIndexDiff(int[] arr) {
        int n=arr.length;
        int[] leftMin=new int[n];
        int[] rightMax=new int[n];
        leftMin[0]=arr[0];
        for(int i=1;i<n;i++){
            leftMin[i]=Math.min(arr[i],leftMin[i-1]);
        }
        rightMax[n-1]=arr[n-1];
        for(int j=n-2;j>=0;j--){
            rightMax[j]=Math.max(arr[j],rightMax[j+1]);
        }
        int i=0,j=0;
        int maxDiff=-1;
        while(i<n&&j<n){
            if(leftMin[i]<=rightMax[j]){
                maxDiff=Math.max(maxDiff,j-i);
                j++;
            }else{
                i++;
            }
        }
```

Time complexity: O(n)
Space complexity: O(n)

8. Wave Array

```
class Solution {
    public static void convertToWave(int[] arr) {
            for(int i=0;i<arr.length;i+=2){
            if(i>0&&arr[i-1]>arr[i]){
                int temp=arr[i];
                arr[i]=arr[i-1];
                arr[i-1]=temp;
            }
            if(i<arr.length-1&&arr[i]<arr[i+1]){
                int temp=arr[i];
                arr[i]=arr[i+1];
                arr[i+1]=temp;
            }
        }
    }
}
```

Output



Problem Solved Successfully

Test Cases Passed
1120 / 1120

Attempts : Correct / Total
1 / 1
Accuracy : 100%

Points Scored
2 / 2
Your Total Score: 60

Time Taken
1.04

```
// } Driver Code Ends


class Solution {
    public static void convertToWave(int[] arr) {
            for(int i=0;i<arr.length;i+=2){
            if(i>0&&arr[i-1]>arr[i]){
                int temp=arr[i];
                arr[i]=arr[i-1];
                arr[i-1]=temp;
            }
            if(i<arr.length-1&&arr[i]<arr[i+1]){
                int temp=arr[i];
                arr[i]=arr[i+1];
                arr[i+1]=temp;
            }
        }
    }
}
```

Time complexity: O(n)
Space complexity: O(1)