DSA Practice 6 (18-11-24)

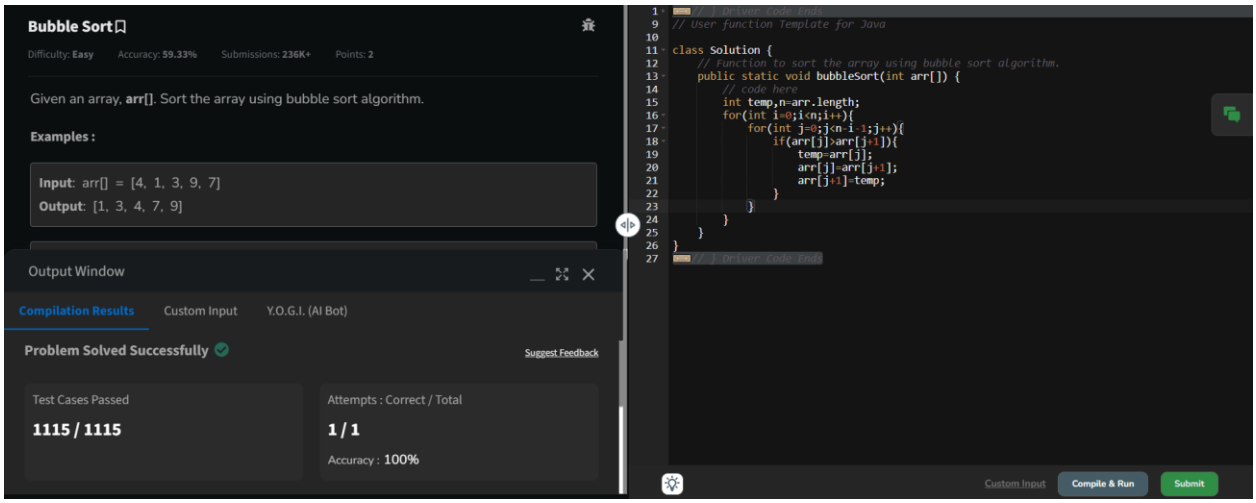
By Jashwanth SA

CSE-C, 22CS066

1. Bubble sort:

```
class Solution {
    // Function to sort the array using bubble sort algorithm.
    public static void bubbleSort(int arr[]) {
        // code here
        int temp,n=arr.length;
        for(int i=0;i<n;i++){
            for(int j=0;j<n-i-1;j++){
                if(arr[j]>arr[j+1]){
                    temp=arr[j];
                    arr[j]=arr[j+1];
                    arr[j+1]=temp;
                }
            }
        }
    }
}
```

Output:



Time complexity: O(n)

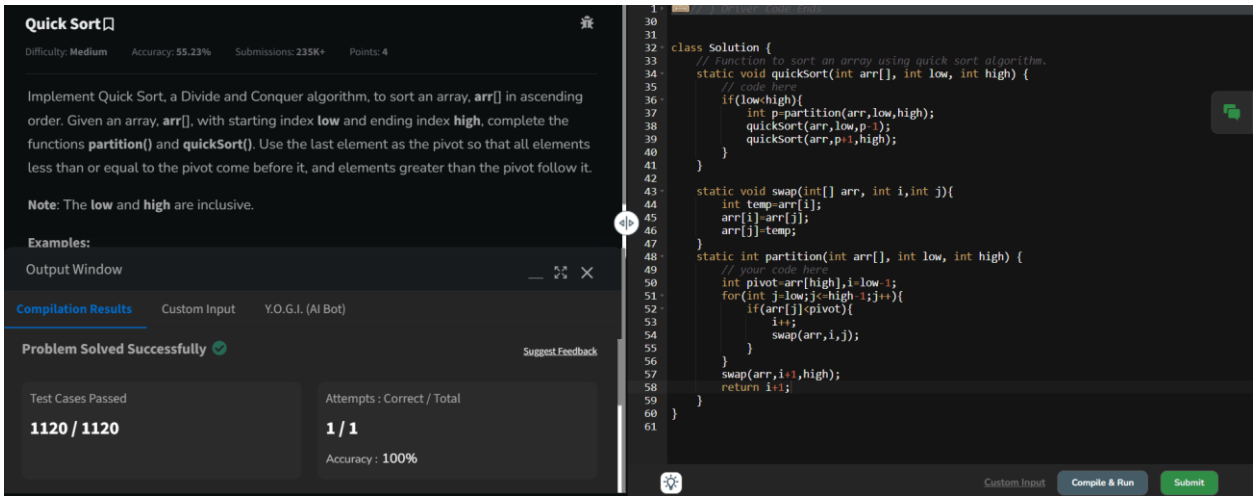
2. Quich sort:

```
class Solution {
    // Function to sort an array using quick sort algorithm.
    static void quickSort(int arr[], int low, int high) {
        // code here
```

```
        if(low<high){
            int p=partition(arr,low,high);
            quickSort(arr,low,p-1);
            quickSort(arr,p+1,high);
        }
    }

    static void swap(int[] arr, int i,int j){
        int temp=arr[i];
        arr[i]=arr[j];
        arr[j]=temp;
    }
    static int partition(int arr[], int low, int high) {
        // your code here
        int pivot=arr[high],i=low-1;
        for(int j=low;j<=high-1;j++){
            if(arr[j]<pivot){
                i++;
                swap(arr,i,j);
            }
        }
        swap(arr,i+1,high);
        return i+1;
    }
}
```

Output:



Time complexity: O(n logn)

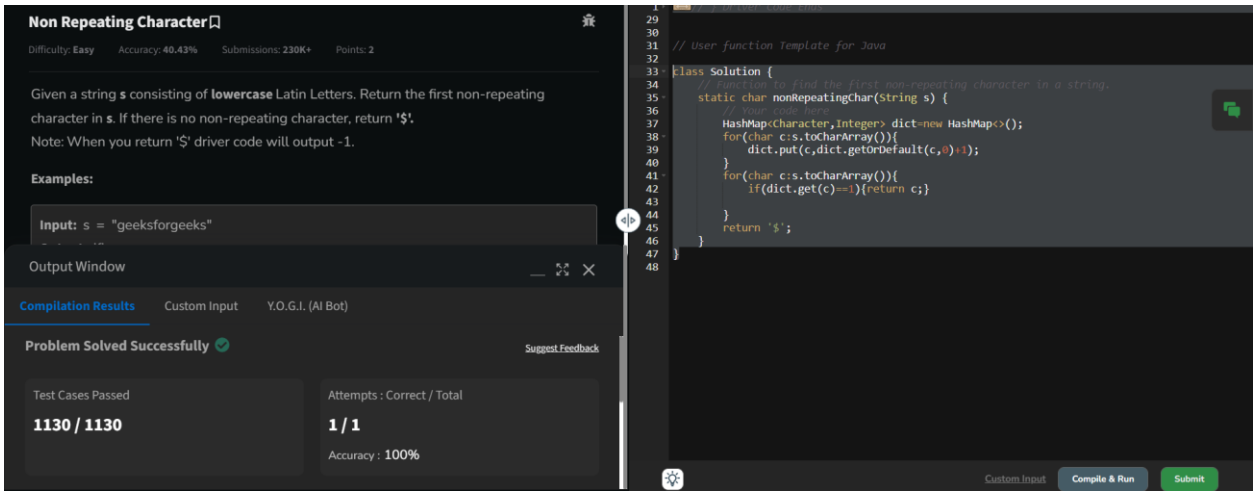
3. Non repeating character:

```
class Solution {
    // Function to find the first non-repeating character in a string.
```

```
    static char nonRepeatingChar(String s) {
        // Your code here
        HashMap<Character,Integer> dict=new HashMap<>();
        for(char c:s.toCharArray()){
            dict.put(c,dict.getOrDefault(c,0)+1);
        }
        for(char c:s.toCharArray()){
            if(dict.get(c)==1){return c;}

        }
        return '$';
    }
}
```

Output:



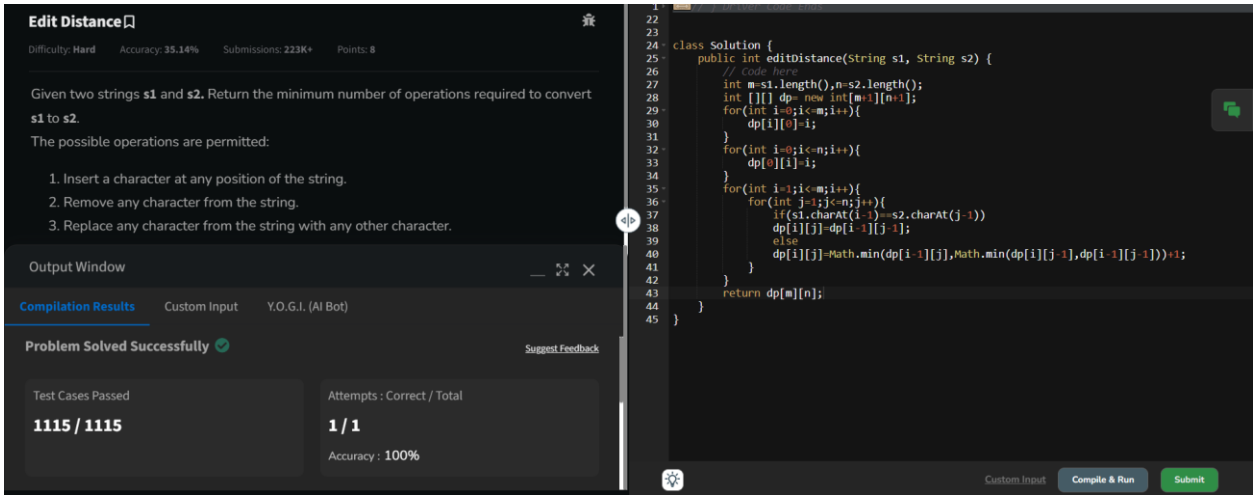
Time complexity: O(n)

4. Edit Distance:

```
class Solution {
    public int editDistance(String s1, String s2) {
        // Code here
        int m=s1.length(),n=s2.length();
        int [][] dp= new int[m+1][n+1];
        for(int i=0;i<=m;i++){
            dp[i][0]=i;
        }
        for(int i=0;i<=n;i++){
            dp[0][i]=i;
        }
        for(int i=1;i<=m;i++){
            for(int j=1;j<=n;j++){
                if(s1.charAt(i-1)==s2.charAt(j-1))
                dp[i][j]=dp[i-1][j-1];
```

```
            else
            dp[i][j]=Math.min(dp[i-1][j],Math.min(dp[i][j-1],dp[i-1][j-1]))+1;
          }
        }
        return dp[m][n];
    }
}
```

Output:



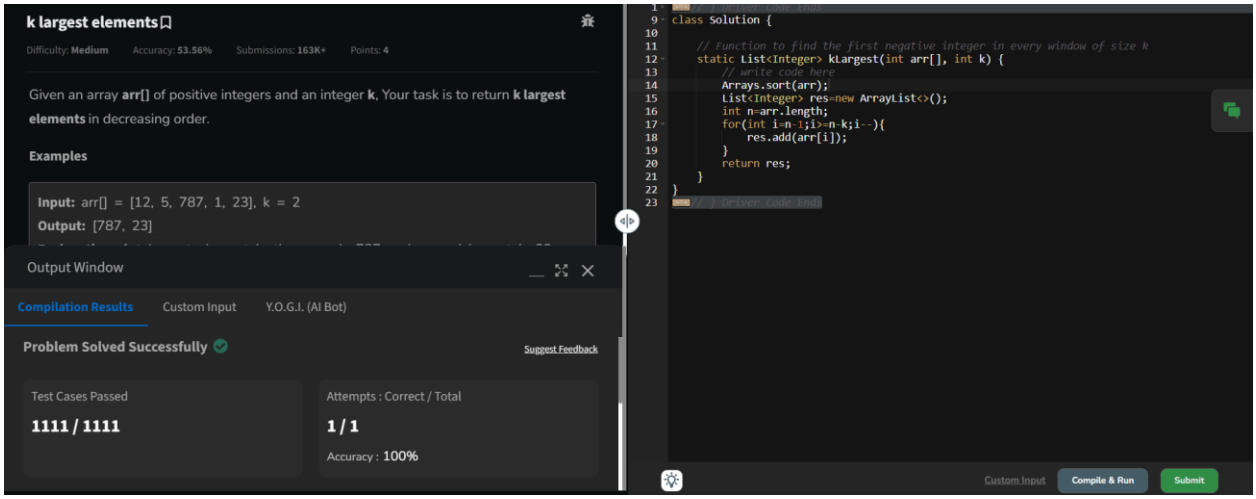
Time complexity: O(n^2)

5. K largest elements:

```
class Solution {

    // Function to find the first negative integer in every window of size k
    static List<Integer> kLargest(int arr[], int k) {
        // write code here
        Arrays.sort(arr);
        List<Integer> res=new ArrayList<>();
        int n=arr.length;
        for(int i=n-1;i>=n-k;i--){
            res.add(arr[i]);
        }
        return res;
    }
}
```

Output:

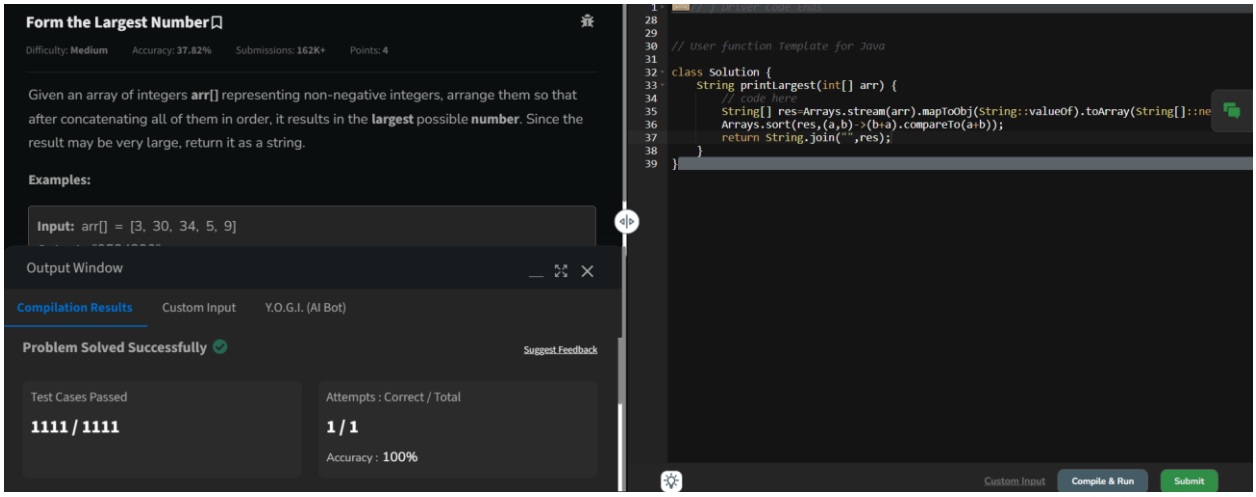


Time complexity: O(n)

6. Form the largest number:

```
class Solution {
   String printLargest(int[] arr) {
      // code here
      String[] res=Arrays.stream(arr).mapToObj(String::valueOf).toArray(String[]::new);
      Arrays.sort(res,(a,b)->(b+a).compareTo(a+b));
      return String.join("",res);
   }
}
```

Output:



Time complexity: O(n)