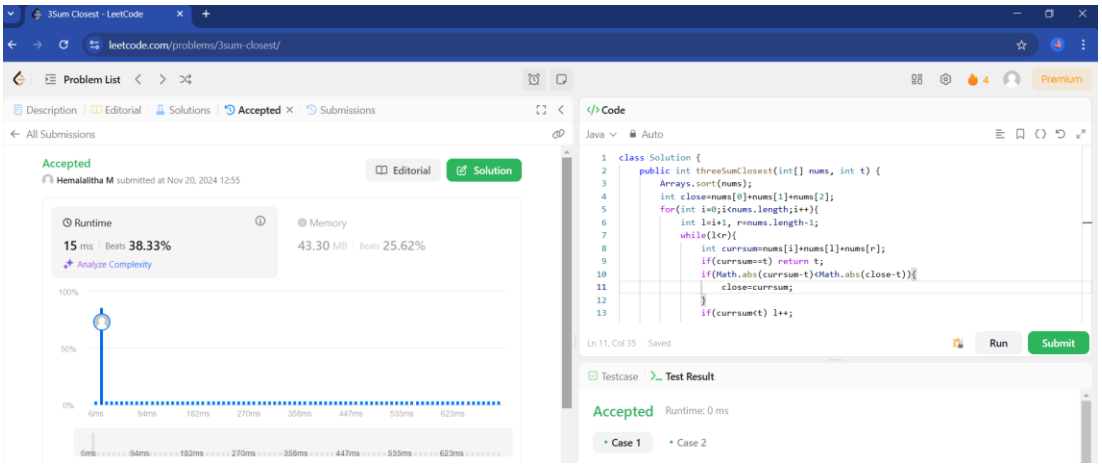
**1. Triplet sum in array:**

**CODE:**

```
class Solution {
    public int threeSumClosest(int[] nums, int t) {
        Arrays.sort(nums);
        int close=nums[0]+nums[1]+nums[2];
        for(int i=0;i<nums.length;i++){
            int l=i+1, r=nums.length-1;
            while(l<r){
                int currsum=nums[i]+nums[l]+nums[r];
                if(currsum==t) return t;
                if(Math.abs(currsum-t)<Math.abs(close-t)){
                    close=currsum;
                }
                if(currsum<t) l++;
                else r--;
            }
        }
        return close;
    }
}
```

**OUTPUT:**




Time Complexity: O(n^2)

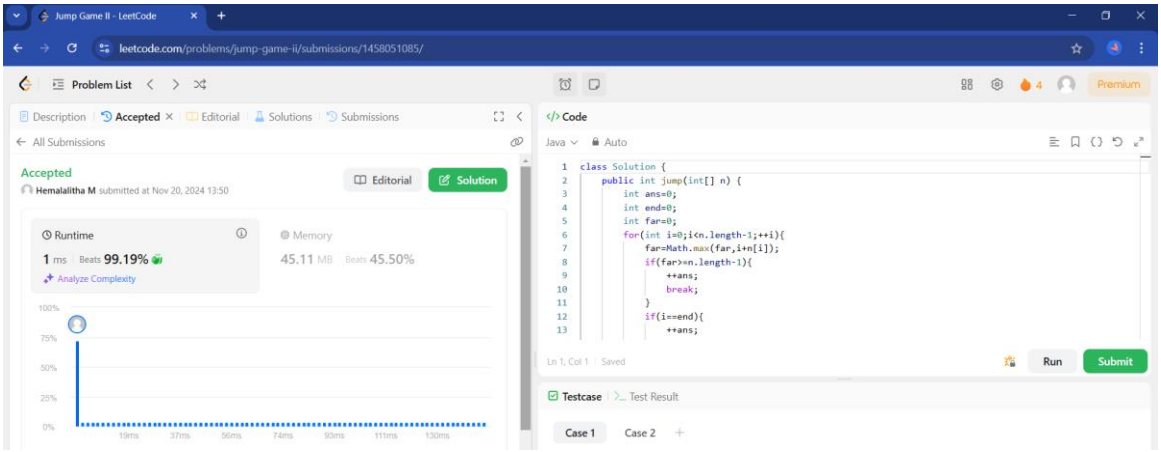
**2. Jump game 2:**

**CODE:**

```
class Solution {
    public int jump(int[] n) {
        int ans=0;
        int end=0;
        int far=0;
        for(int i=0;i<n.length-1;++i){
            far=Math.max(far,i+n[i]);
            if(far>=n.length-1){
                ++ans;
                break;
            }
```

```
            if(i==end){
                ++ans;
                end=far;
            }
        }
        return ans;
    }
}
```

**OUTPUT:**



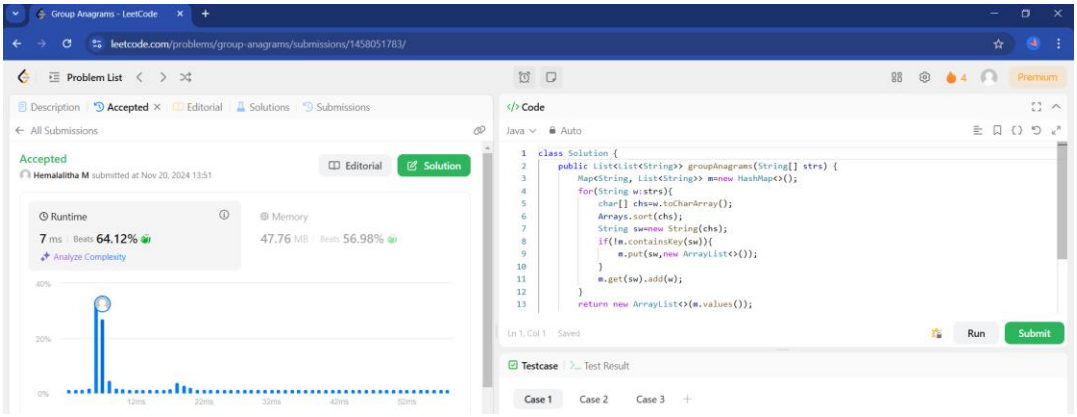

Time Complexity: O(n)

**3. Group Anagrams:**

**CODE:**

```
class Solution {
    public List<List<String>> groupAnagrams(String[] strs) {
        Map<String, List<String>> m=new HashMap<>();
        for(String w:strs){
            char[] chs=w.toCharArray();
            Arrays.sort(chs);
            String sw=new String(chs);
            if(!m.containsKey(sw)){
                m.put(sw,new ArrayList<>());
            }
            m.get(sw).add(w);
        }
        return new ArrayList<>(m.values());
    }
}
```

**OUTPUT:**



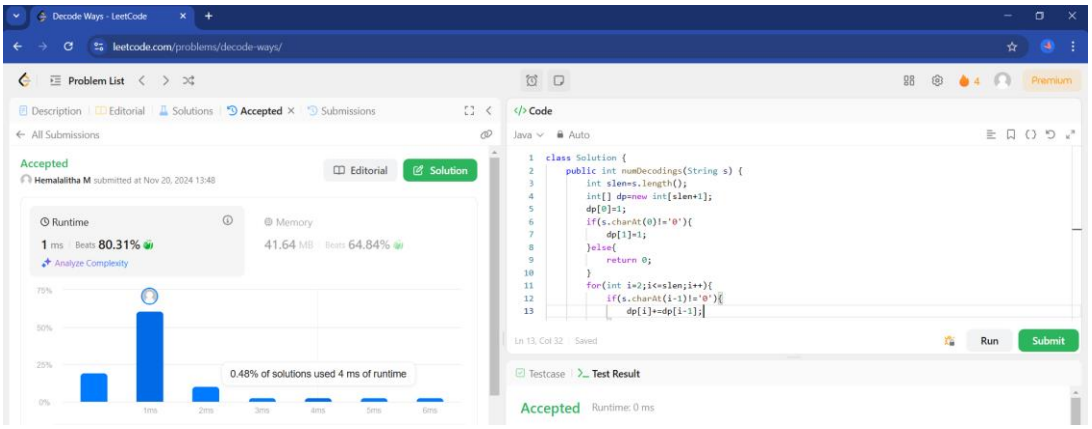

Time Complexity: O(n * m log m)

**4. Decode ways:**

**CODE:**

```
class Solution {
    public int numDecodings(String s) {
        int slen=s.length();
        int[] dp=new int[slen+1];
        dp[0]=1;
        if(s.charAt(0)!='0'){
            dp[1]=1;
        }else{
            return 0;
        }
        for(int i=2;i<=slen;i++){
            if(s.charAt(i-1)!='0'){
                dp[i]+=dp[i-1];
            }
            if(s.charAt(i-2)=='1' || (s.charAt(i-2)=='2' && s.charAt(i-1)<='6')){
                dp[i]+=dp[i-2];
            }
        }
        return dp[slen];
    }
}
```

**OUTPUT:**



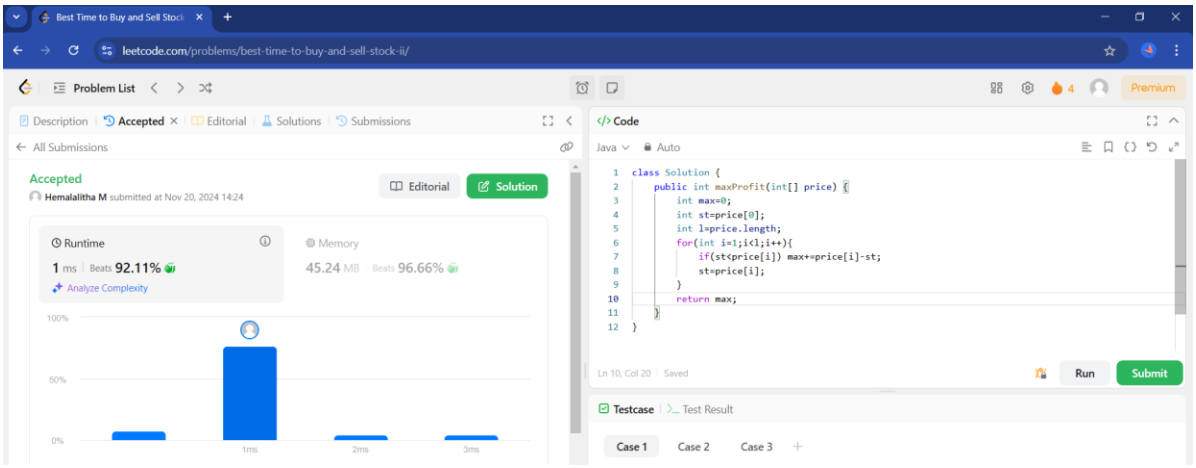

Time complexity: O(n)

**5. Best time to buy and sell stock 2:**

**CODE:**

```
class Solution {
    public int maxProfit(int[] price) {
        int max=0;
        int st=price[0];
        int l=price.length;
        for(int i=1;i<l;i++){
            if(st<price[i]) max+=price[i]-st;
            st=price[i];
        }
        return max;
    }
}
```

**OUTPUT:**




Time complexity: O(n)

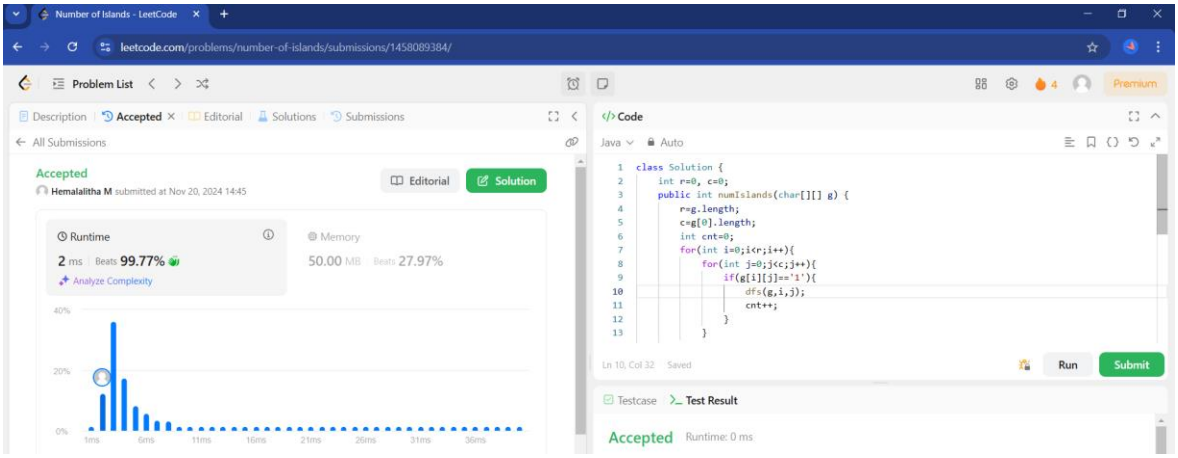
**6. Number of islands:**

**CODE:**

```
class Solution {
    int r=0, c=0;
    public int numIslands(char[][] g) {
        r=g.length;
        c=g[0].length;
        int cnt=0;
        for(int i=0;i<r;i++){
            for(int j=0;j<c;j++){
                if(g[i][j]=='1'){
                    dfs(g,i,j);
                    cnt++;
                }
            }
        }
        return cnt;
    }
    public void dfs(char[][] g, int i, int j){
        if(i<0 || i>=r || j<0 || j>=c || g[i][j]!='1'){
```

```
            return;
        }
        g[i][j]='2';
        dfs(g,i,j-1);
        dfs(g,i-1,j);
        dfs(g,i,j+1);
        dfs(g,i+1,j);
    }
}
```

**OUTPUT:**




Time complexity: O(m*n)

**7. Quick sort:**

**CODE:**

```
import java.util.*;
class quick_sort{
   static int parts(int[] a, int l, int h){
      int piv=a[h];
      int i=l-1;
      for(int j=l;j<=h-1;j++){
         if(a[j]<piv){
            i++;
            swap(a,i,j);
         }
      }
      swap(a,i+1,h);
      return i+1;
   }
   static void swap(int[] a, int i, int j){
      int t=a[i];
      a[i]=a[j];
      a[j]=t;
   }
   static void quicksort(int[] a, int l, int h){
      if(l<h){
         int pi=parts(a,l,h);
         quicksort(a,l,pi-1);
```

```
        quicksort(a,pi+1,h);
      }
   }
   public static void main(String[] args){
      int[] a={10,7,8,9,1,5};
      int n=a.length;
      quicksort(a,0,n-1);
      for(int v:a){
         System.out.print(v+" ");
      }
   }
}
```

**OUTPUT:**

```
[Running] cd "c:\Data\Knowledge\DSA\Programs\Day_6\" && javac quick_sort.java && java quick_sort
1 5 7 8 9 10
```

Time Complexity: O(n log n)

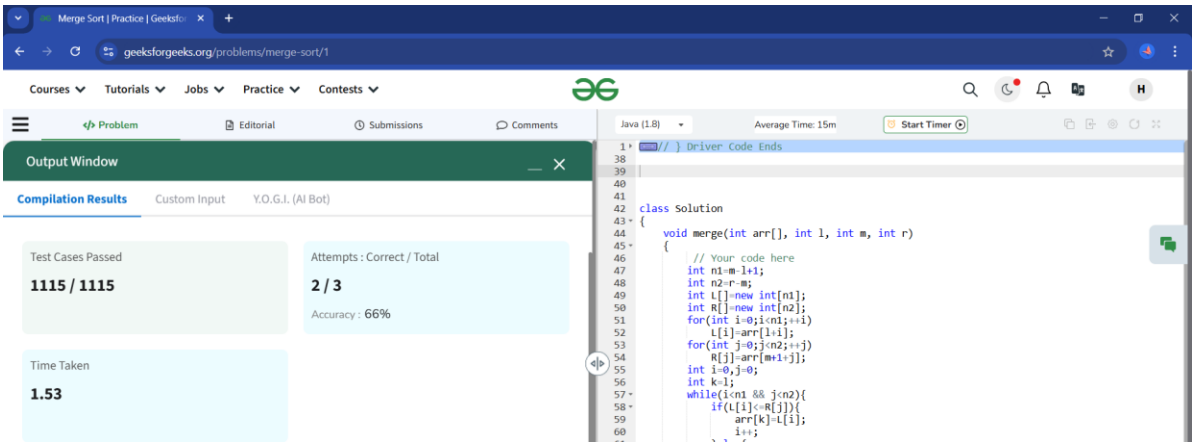
**8. Merge sort:**

**CODE:**

```
class Solution
{
   void merge(int arr[], int l, int m, int r)
   {
       // Your code here
      int n1=m-l+1;
      int n2=r-m;
      int L[]=new int[n1];
      int R[]=new int[n2];
      for(int i=0;i<n1;++i)
         L[i]=arr[l+i];
      for(int j=0;j<n2;++j)
         R[j]=arr[m+1+j];
      int i=0,j=0;
      int k=l;
      while(i<n1 && j<n2){
         if(L[i]<=R[j]){
            arr[k]=L[i];
            i++;
         }else{
            arr[k]=R[j];
            j++;
         }
         k++;
      }
      while(i<n1){
         arr[k]=L[i];
         i++;
         k++;
```

```
    }
    while(j<n2){
      arr[k]=R[j];
      j++;
      k++;
    }
  }
  void mergeSort(int arr[], int l, int r)
  {
    //code here
    if(l<r){
      int m=l+(r-l)/2;
      mergeSort(arr,l,m);
      mergeSort(arr,m+1,r);
      merge(arr,l,m,r);
    }
  }
}
```

**OUTPUT:**




Time complexity: O(n log n)

**9. Ternary search:**

**CODE:**

```
class ternary_search{
  static int ternarysearch(int l, int r, int k, int a[]){
    if(r>=l){
      int m1=l+(r-l)/3;
      int m2=r-(r-l)/3;
      if(a[m1]==k){
        return m1;
      }
      if(a[m2]==k){
        return m2;
      }
      if(k<a[m1]){
        return ternarysearch(l,m1-1,k,a);
      }
```

```
        else if(k>a[m2]){
            return ternarysearch(m2+1,r,k,a);
        }
        else{
            return ternarysearch(m1+1,m2-1,k,a);
        }
    }
    return -1;
  }
  public static void main(String[] args){
    int l, r, p, k;
    int a[]={1,2,3,4,5,6,7,8,9,10};
    l=0;
    r=9;
    k=5;
    p=ternarysearch(l,r,k,a);
    System.out.println("Index of "+k+" is "+p);
  }
}
```

**OUTPUT:**

```
[Running] cd "c:\Data\Knowledge\DSA\Programs\Day_8\" && javac ternary_search.java && java ternary_search
Index of 5 is 4
```

Time complexity: O(2*log n)

**10. Interpolation search:**

**CODE:**

```
import java.util.*;
class interpolation_search{
  public static int interpolationsearch(int[] a, int l, int h, int x){
    int pos;
    if(l<=h && x>=a[l] && x<=a[h]){
      pos=l+(((h-l)/(a[h]-a[l]))*(x-a[l]));
      if(a[pos]==x) return pos;
      if(a[pos]<x) return interpolationsearch(a,pos+1,h,x);
      if(a[pos]>x) return interpolationsearch(a,l,pos-1,x);
    }
    return -1;
  }
  public static void main(String[] args){
    int a[]={10,12,13,16,18,19,20,21,22,23,24,33,35,42,47};
    int n=a.length;
    int x=18;
    int ind=interpolationsearch(a,0,n-1,x);
    if(ind!=-1) System.out.println("Element found at index "+ind);
    else System.out.println("Element not found");
  }
}
```

**OUTPUT:**

```
[Running] cd "c:\Data\Knowledge\DSA\Programs\Day_8\" && javac interpolation_search.java && java
interpolation_search
Element found at index 4
```

Time complexity: O(log(log n))

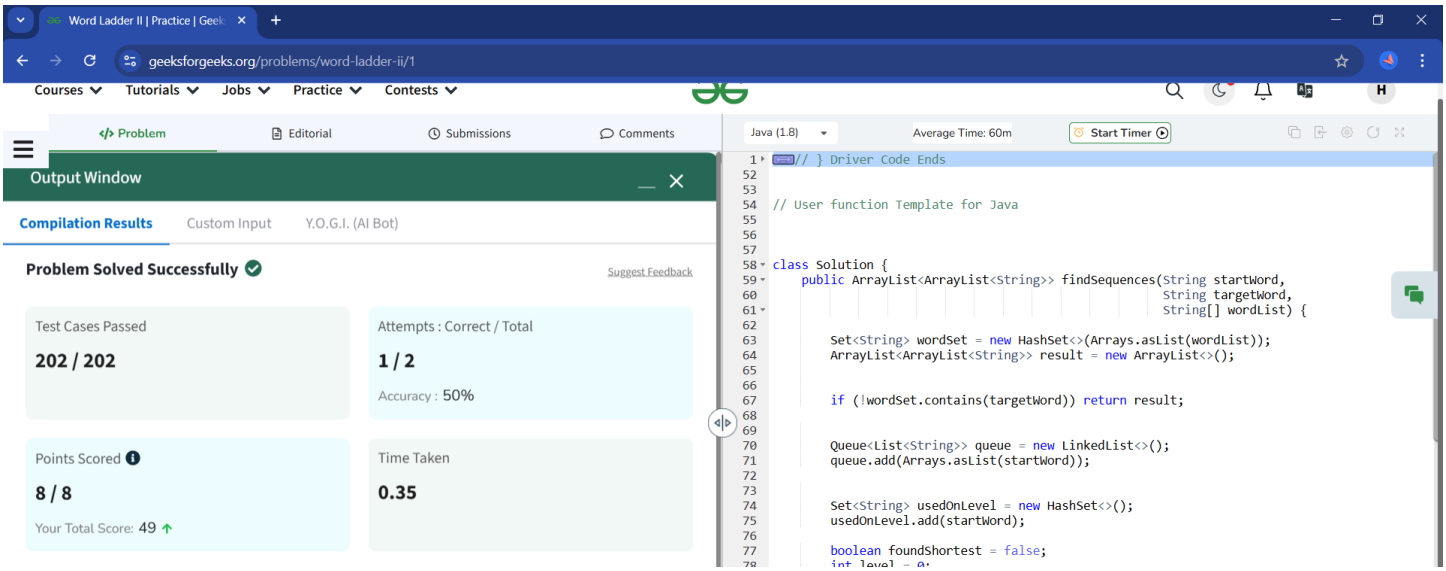
**11. Word ladder 2:**

**CODE:**

```
class Solution {
    public ArrayList<ArrayList<String>> findSequences(String startWord,
                                        String targetWord,
                                        String[] wordList) {
        Set<String> wordSet = new HashSet<>(Arrays.asList(wordList));
        ArrayList<ArrayList<String>> result = new ArrayList<>();
        if (!wordSet.contains(targetWord)) return result;
        Queue<List<String>> queue = new LinkedList<>();
        queue.add(Arrays.asList(startWord));
        Set<String> usedOnLevel = new HashSet<>();
        usedOnLevel.add(startWord);
        boolean foundShortest = false;
        int level = 0;
        while (!queue.isEmpty() && !foundShortest) {
            int qSize = queue.size();
            level++;
            for (int i = 0; i < qSize; i++) {
                List<String> currentPath = queue.poll();
                String lastWord = currentPath.get(currentPath.size() - 1);
                for (int j = 0; j < lastWord.length(); j++) {
                    char[] charArray = lastWord.toCharArray();
                    for (char c = 'a'; c <= 'z'; c++) {
                        charArray[j] = c;
                        String transformedWord = new String(charArray);
                        if (transformedWord.equals(targetWord)) {
                            foundShortest = true;
                            ArrayList<String> newPath = new ArrayList<>(currentPath);
                            newPath.add(transformedWord);
                            result.add(newPath);
                        }
                        if (wordSet.contains(transformedWord)) {
                            usedOnLevel.add(transformedWord);
                            ArrayList<String> newPath = new ArrayList<>(currentPath);
                            newPath.add(transformedWord);
                            queue.add(newPath);
                        }
                    }
                }
            }
            for (String word : usedOnLevel) {
```

```
            wordSet.remove(word);
        }
        usedOnLevel.clear();
    }
    return result;
  }
}
```

**OUTPUT:**




Time complexity: O(n * m^2)

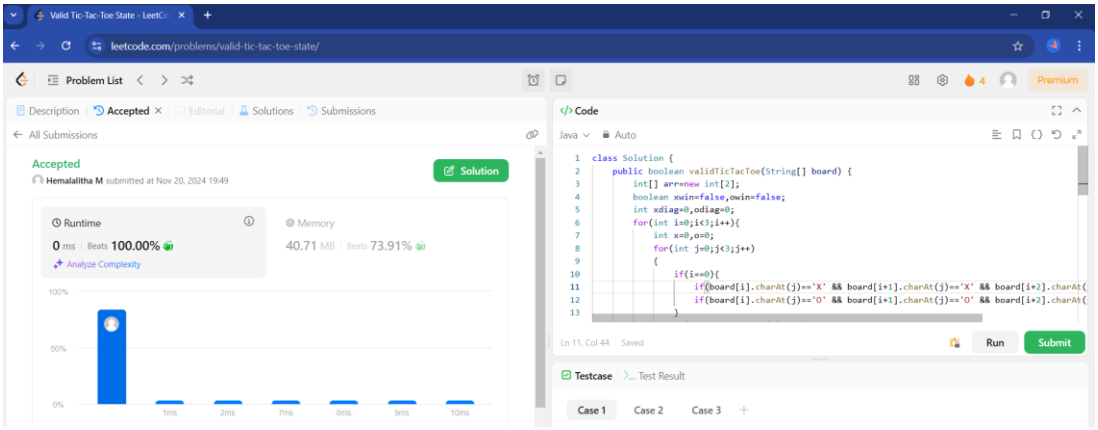
**12. Tic tac toe:**

**CODE:**

```
class Solution {
    public boolean validTicTacToe(String[] board) {
        int[] arr=new int[2];
        boolean xwin=false,owin=false;
        int xdiag=0,odiag=0;
        for(int i=0;i<3;i++){
            int x=0,o=0;
            for(int j=0;j<3;j++)
            {
                if(i==0){
                    if(board[i].charAt(j)=='X' && board[i+1].charAt(j)=='X' &&
board[i+2].charAt(j)=='X')xwin=true;
                    if(board[i].charAt(j)=='O' && board[i+1].charAt(j)=='O' &&
board[i+2].charAt(j)=='O')owin=true;
                }
                if(board[i].charAt(j)=='X'){
                    if(i==j)xdiag++;
                    arr[1]++;
                    x++;
                }
                else if(board[i].charAt(j)=='O'){
                    if(i==j)odiag++;
                    arr[0]++;
                    o++;
```

```
                }
            }
            if(o==3 && owin)return false;
            if(owin && xwin)return false;
            if(x==3)xwin=true;
            else if(o==3)owin=true;
        }

        if(xdiag==3)xwin=true;
        if(odiag==3)owin=true;

        if(board[0].charAt(2)=='X' && board[1].charAt(1)=='X' &&
board[2].charAt(0)=='X')xwin=true;
        if(board[0].charAt(2)=='O' && board[1].charAt(1)=='O' &&
board[2].charAt(0)=='O')owin=true;

        if(arr[0]>=arr[1] && xwin || (arr[1]>arr[0] && owin))return false;
        if(arr[0]>arr[1] || Math.abs(arr[0]-arr[1])>1)return false;

        if(xwin&&owin) return false;
        else return true;
    }
}
```

**OUTPUT:**




Time complexity: O(1)