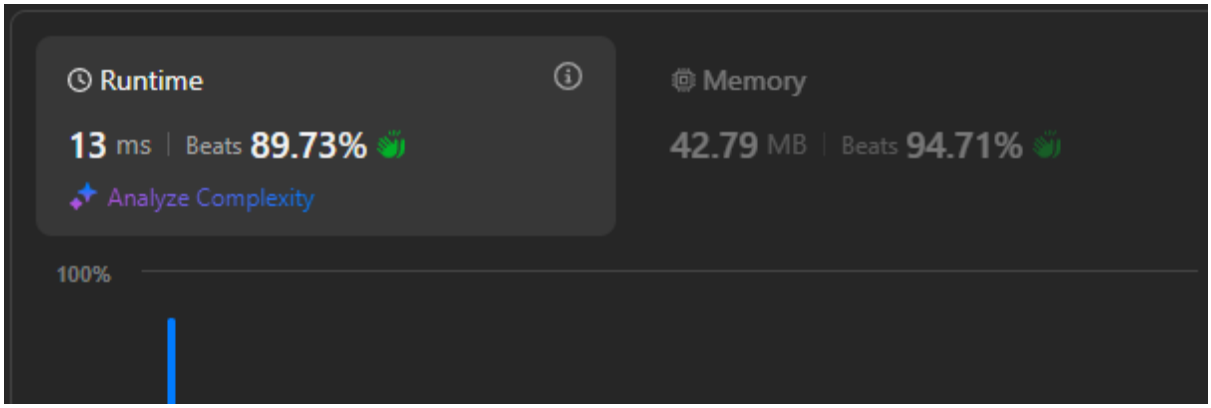
3SUM CLOSEST

CODE :

```
class Solution {
   public int threeSumClosest(int[] nums, int target) {
      Arrays.sort(nums);
      int ans = nums[0] + nums[1] + nums[2];
      int dif = Integer.MAX_VALUE;
      for(int i = 0 ; i<nums.length-2 ; i++){
         int j = i +1;
         int k = nums.length - 1;
         while(j<k){
            int s = nums[i] + nums[j] + nums[k];
            if(s==target){
               return s;
            }
            if(s<target){
               j++;
            }else{
               k--;
            }
            int d = Math.abs(s-target);
            if(dif>d){
               dif = d;
               ans = s;
            }
         }
      }
      return ans;
   }
}
```

OUTPUT :

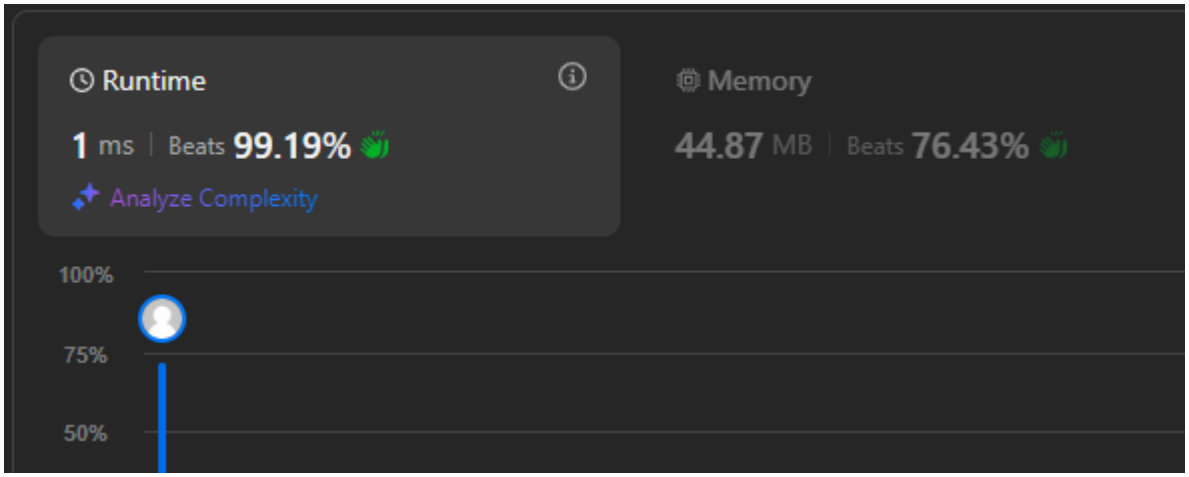


JUMP GAME II

CODE :

```
class Solution {
   public int jump(int[] nums) {
      int j = 0 , l = 0 , r = 0;
      while(r<nums.length - 1){
         int f = Integer.MIN_VALUE;
         for(int i = l ; i<=r ; i++){
            f = Math.max(i+nums[i] , f);
         }
         l= r+1;
         r = f;
         j++;
      }
      return j;
   }
}
```

OUTPUT :

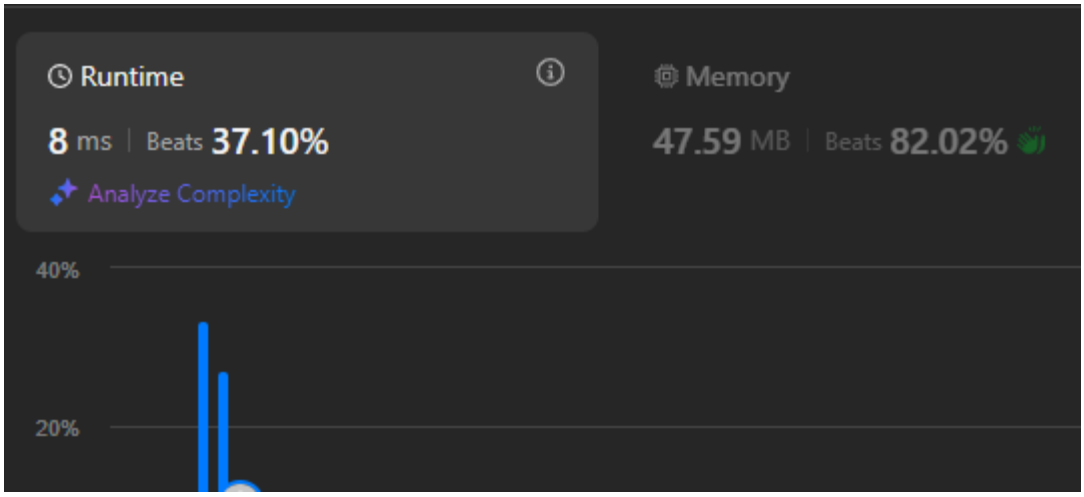


GROUP ANAGARAM'S

CODE :

```
class Solution {
   public List<List<String>> groupAnagrams(String[] strs) {
      HashMap<String , List<String>> hm =  new HashMap<>();
      for(int i = 0 ; i<strs.length ; i++){
         char c[] = strs[i].toCharArray();
         Arrays.sort(c);
         String s= new String(c);
         if(hm.containsKey(s)){
            List<String> l = hm.get(s);
            l.add(strs[i]);
            hm.put(s,l);
         }else{
            List<String> l  = new ArrayList<>();
            l.add(strs[i]);
            hm.put(s,l);
         }
      }
      List<List<String>> ans = new ArrayList<>();
      for(Map.Entry<String,List<String>> e : hm.entrySet()){
         ans.add(e.getValue());
      }
      return ans;
   }
}
```

OUTPUT :



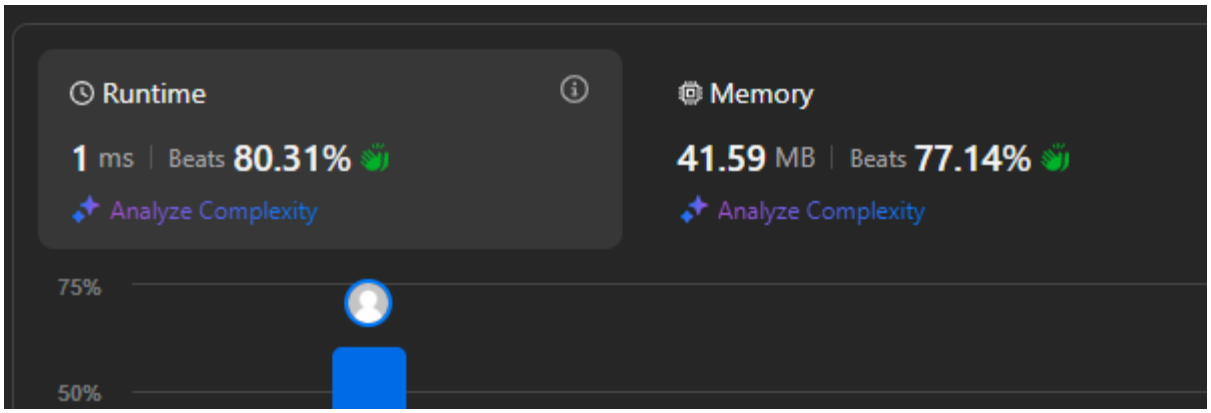
DECODE WAYS

CODE :

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

OUTPUT :

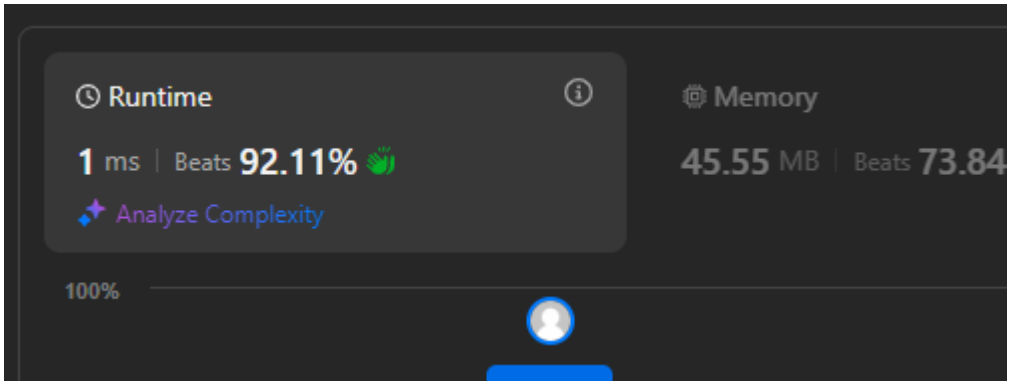


BEST TIME TO BUY AND SELL STOCKS II

CODE :

```
class Solution {
   public int maxProfit(int[] a) {
      int curr=a[0];
      int profit=0;
      for(int i=1;i<a.length;i++){
         if(a[i]<curr) curr=a[i];
         else if(a[i]>curr) {
            profit+=a[i]-curr;
            curr=a[i];
         }
      }
      return profit;
   }
}
```

OUTPUT :

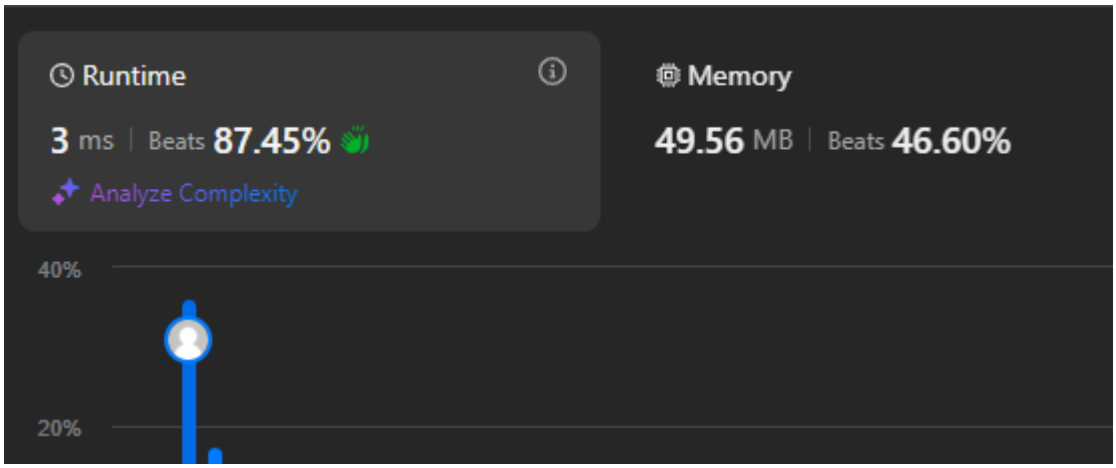


NUMBER OF ISLANDS

CODE :

```
class Solution {
   public void dfs(char[][]grid,int i,int j){
      int m=grid.length,n=grid[0].length;
      if(i<0||j<0||i>=m||j>=n||grid[i][j]=='0')return;
      grid[i][j]='0';
      dfs(grid,i+1,j);
      dfs(grid,i-1,j);
      dfs(grid,i,j+1);
      dfs(grid,i,j-1);
   }
   public int numIslands(char[][] grid) {
      int m=grid.length,n=grid[0].length,count=0;
      for(int i=0;i<m;i++){
         for(int j=0;j<n;j++){
            if(grid[i][j]=='1'){
               count++;
               dfs(grid,i,j);
            }
         }
      }
      return count;
   }
}
```

OUTPUT :

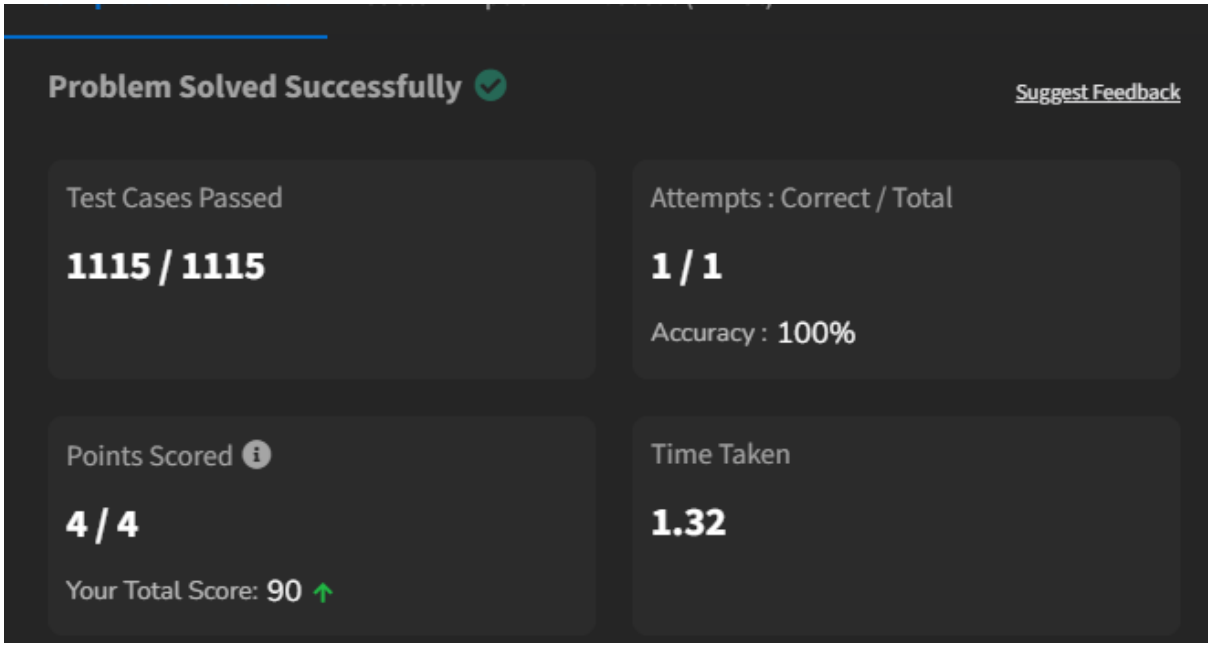


MERGE SORT

CODE :

```
class Solution {
    void mergeSort(int arr[], int l, int r) {
        if (l < r) {
            int m = l + (r - l) / 2;
            mergeSort(arr, l, m);
            mergeSort(arr, m + 1, r);
            merge(arr, l, m, r);
        }
    }
    static void merge(int arr[], int l, int m, int r){
        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++;
        }
    }
}
```

OUTPUT :


Problem Solved Successfully
Suggest Feedback
Test Cases Passed
1115 / 1115
Attempts : Correct / Total
1 / 1
Accuracy : 100%
Points Scored
4 / 4
Your Total Score: 90
Time Taken
1.32

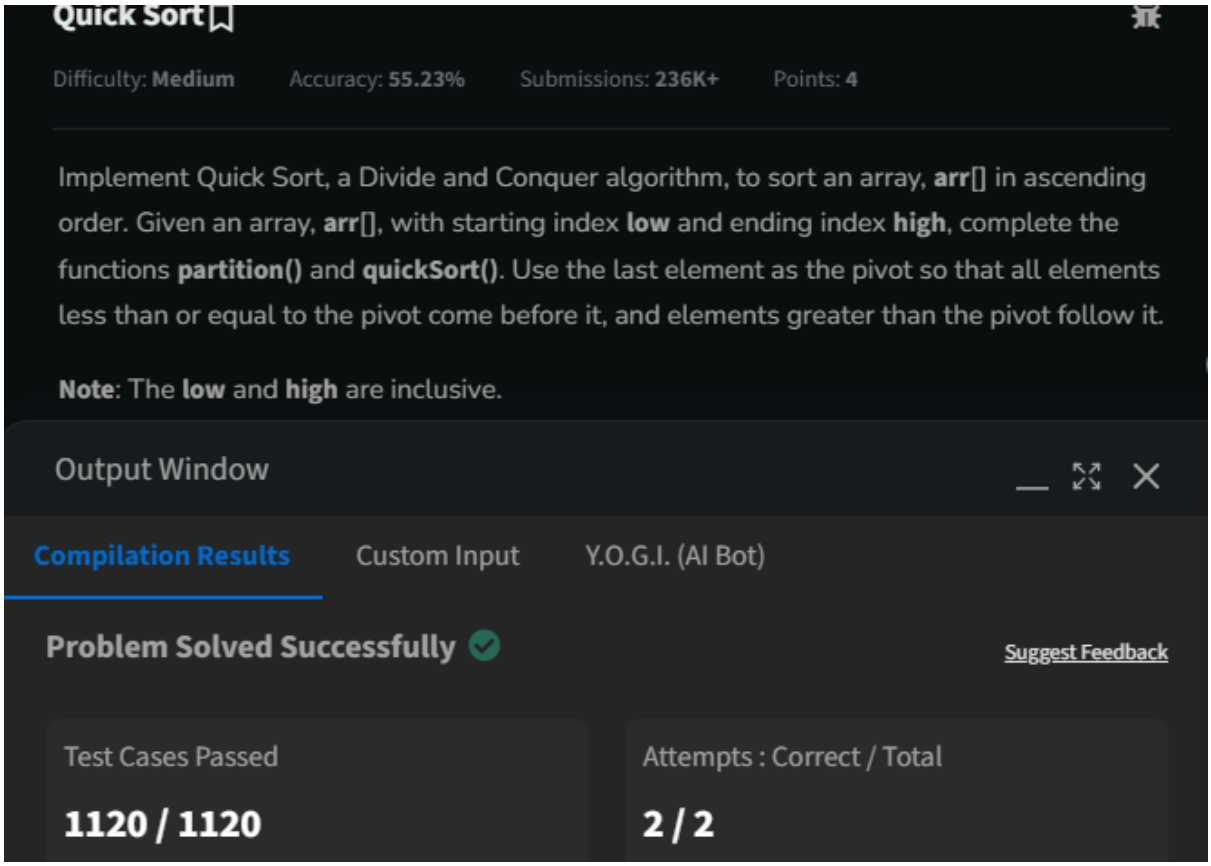
QUICK SORT

CODE :

```
class Solution {
   static void quickSort(int arr[], int low, int high) {
      if(low<high){
         int pi = partition(arr,low,high);
         quickSort(arr,low,pi-1);
         quickSort(arr,pi+1,high);
      }
   }

   static int partition(int arr[], int low, int high) {
      int pivot = arr[high];
      int 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;
   }
   static void swap(int[] arr, int i, int j) {
      int temp = arr[i];
      arr[i] = arr[j];
      arr[j] = temp;
   }
}
```

OUTPUT :


Quick Sort
Difficulty: Medium
Accuracy: 55.23%
Submissions: 236K+
Points: 4
Implement Quick Sort, a Divide and Conquer algorithm, to sort an array, arr[] in ascending order. Given an array, arr[], with starting index low and ending index high, complete the functions partition() and quickSort(). Use the last element as the pivot so that all elements less than or equal to the pivot come before it, and elements greater than the pivot follow it.
Note: The low and high are inclusive.
Output Window
Compilation Results
Custom Input
Y.O.G.I. (AI Bot)
Problem Solved Successfully
Suggest Feedback
Test Cases Passed
1120 / 1120
Attempts : Correct / Total
2 / 2