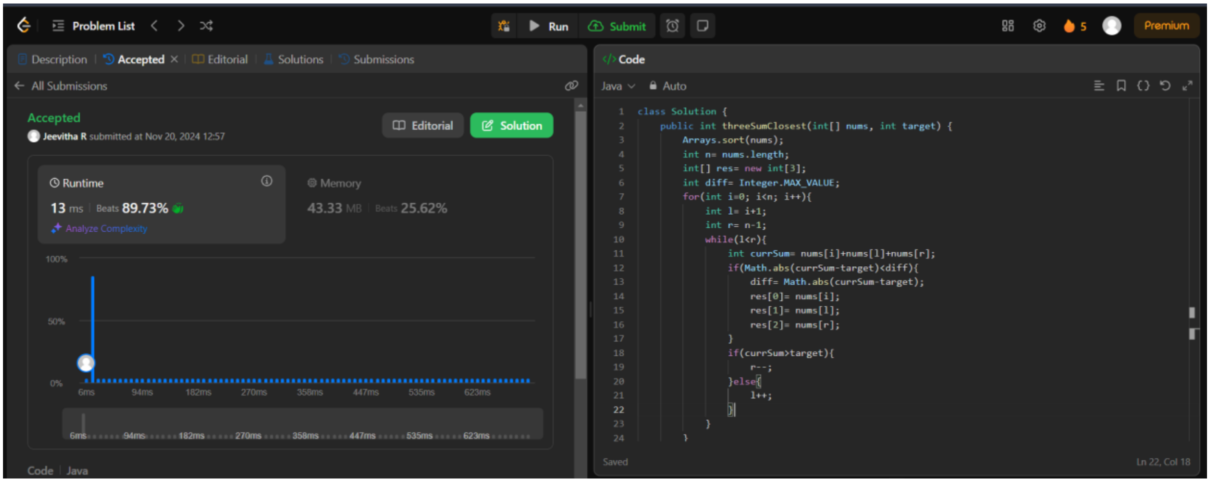
DSA PRACTICE QUESTIONS- DAY 8

NAME: Jeevitha R
REG NO: 22IT040
DATE: 20/11/2024

1. 3Sum Closest

```
class Solution {
    public int threeSumClosest(int[] nums, int target) {
        Arrays.sort(nums);
        int n= nums.length;
        int[] res= new int[3];
        int diff= Integer.MAX_VALUE;
        for(int i=0; i<n; i++){
            int l= i+1;
            int r= n-1;
            while(l<r){
                int currSum= nums[i]+nums[l]+nums[r];
                if(Math.abs(currSum-target)<diff){
                    diff= Math.abs(currSum-target);
                    res[0]= nums[i];
                    res[1]= nums[l];
                    res[2]= nums[r];
                }
                if(currSum>target){
                    r--;
                }else{
                    l++;
                }
            }
        }
        return res[0]+res[1]+res[2];
    }
}
```

Output



Accepted
Jeevitha R submitted at Nov 20, 2024 12:57

Runtime
13 ms | Beats 89.73%

Memory
43.33 MB | Beats 25.62%

Code | Java

```
class Solution {
    public int threeSumClosest(int[] nums, int target) {
        Arrays.sort(nums);
        int n= nums.length;
        int[] res= new int[3];
        int diff= Integer.MAX_VALUE;
        for(int i=0; i<n; i++){
            int l= i+1;
            int r= n-1;
            while(l<r){
                int currSum= nums[i]+nums[l]+nums[r];
                if(Math.abs(currSum-target)<diff){
                    diff= Math.abs(currSum-target);
                    res[0]= nums[i];
                    res[1]= nums[l];
                    res[2]= nums[r];
                }
                if(currSum>target){
                    r--;
                }else{
                    l++;
                }
            }
        }
```

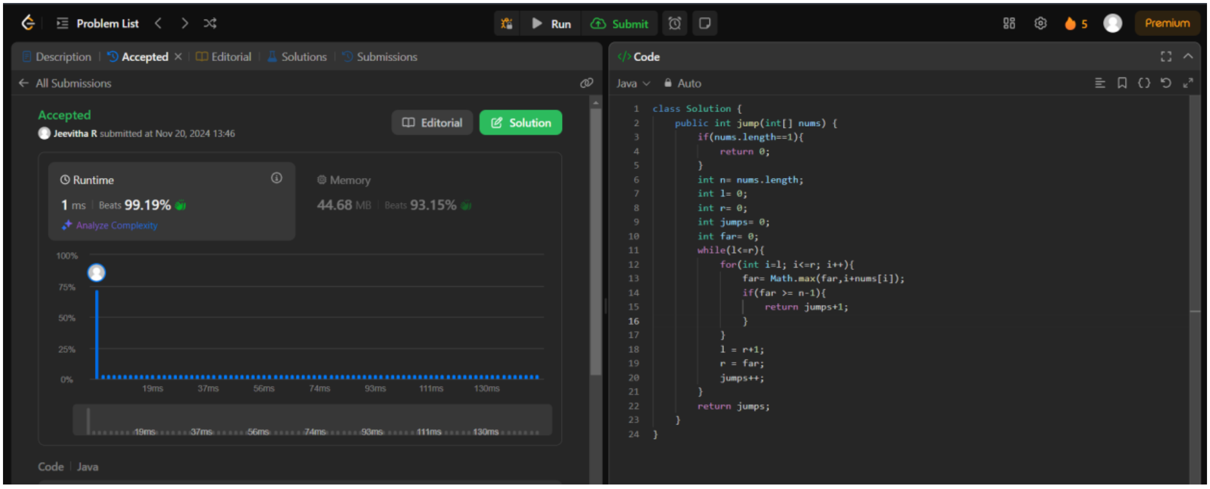
Saved
Ln 22, Col 18

Time complexity: O(n^2)
Space complexity: O(1)

2. Jump Game II

```
class Solution {
    public int jump(int[] nums) {
        if(nums.length==1){
            return 0;
        }
        int n= nums.length;
        int l= 0;
        int r= 0;
        int jumps= 0;
        int far= 0;
        while(l<=r){
            for(int i=l; i<=r; i++){
                far= Math.max(far,i+nums[i]);
                if(far >= n-1){
                    return jumps+1;
                }
            }
            l = r+1;
            r = far;
            jumps++;
        }
        return jumps;
    }
}
```

Output



Accepted
Jeevitha R submitted at Nov 20, 2024 13:46

1 ms | Beats 99.19%

44.68 MB | Beats 93.15%

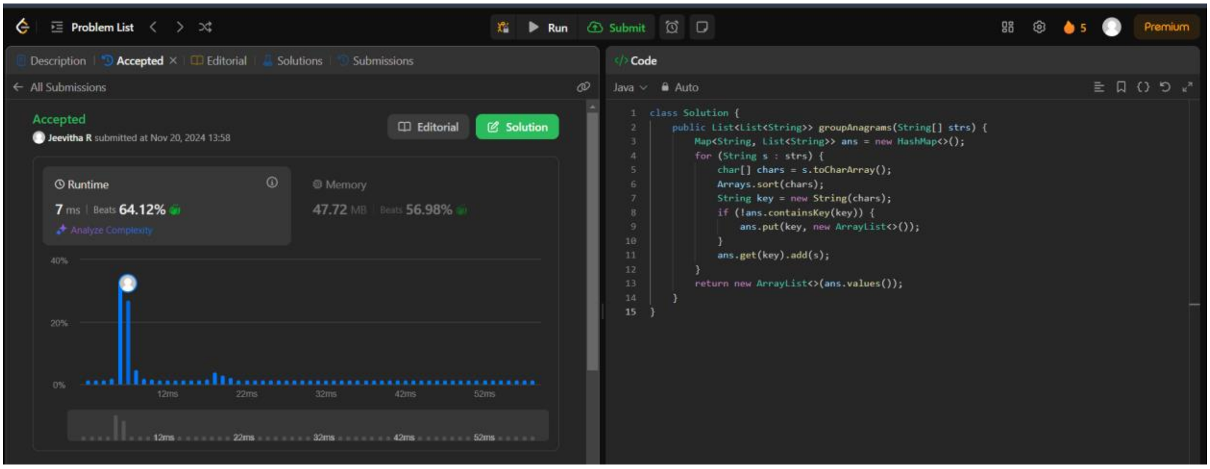
```
class Solution {
    public int jump(int[] nums) {
        if(nums.length==1){
            return 0;
        }
        int n= nums.length;
        int l= 0;
        int r= 0;
        int jumps= 0;
        int far= 0;
        while(l<=r){
            for(int i=l; i<=r; i++){
                far= Math.max(far,i+nums[i]);
                if(far >= n-1){
                    return jumps+1;
                }
            }
            l = r+1;
            r = far;
            jumps++;
        }
        return jumps;
    }
}
```

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

3. Group Anagrams

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

Output



Accepted

Jeevitha R submitted at Nov 20, 2024 13:58

Runtime

7 ms | Beats 64.12%

Memory

47.72 MB | Beats 56.98%

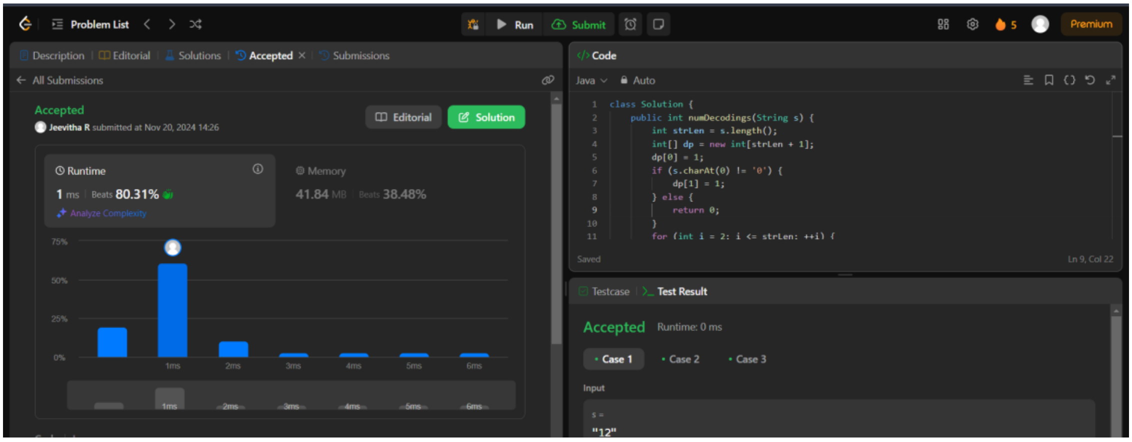
```
class Solution {
    public List<List<String>> groupAnagrams(String[] strs) {
        Map<String, List<String>> ans = new HashMap<>();
        for (String s : strs) {
            char[] chars = s.toCharArray();
            Arrays.sort(chars);
            String key = new String(chars);
         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

Time complexity: O(n·m logm)
Space complexity: O(n·m)

4. Decode ways

```
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



Accepted
Jeevitha R submitted at Nov 20, 2024 14:26

Runtime: 1 ms | Beats 80.31%

Memory: 41.84 MB | Beats 38.48%

```
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) {
```

Accepted Runtime: 0 ms

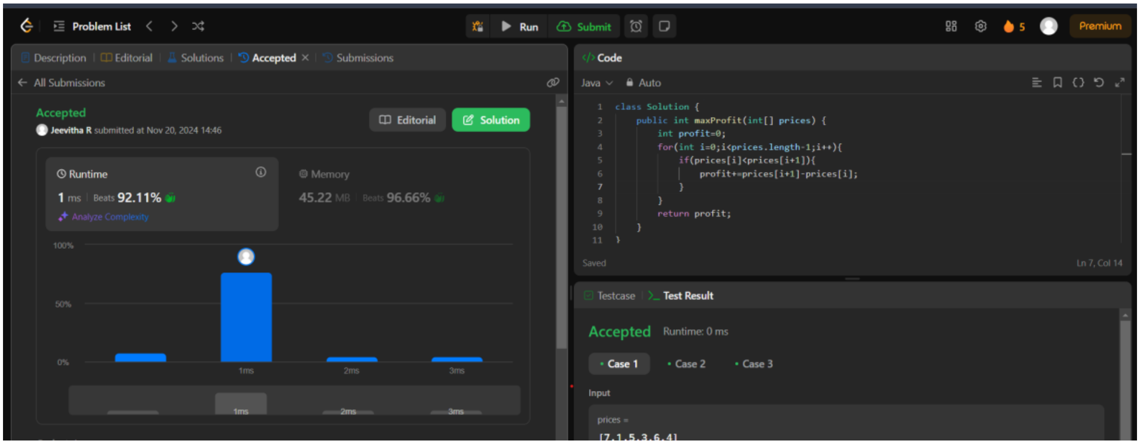
s =
"12"

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

5. Best time to buy and sell stock II

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

Output



Accepted
Jeevitha R submitted at Nov 20, 2024 14:46

Runtime: 1 ms | Beats 92.11%

Memory: 45.22 MB | Beats 96.66%

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

Accepted Runtime: 0 ms

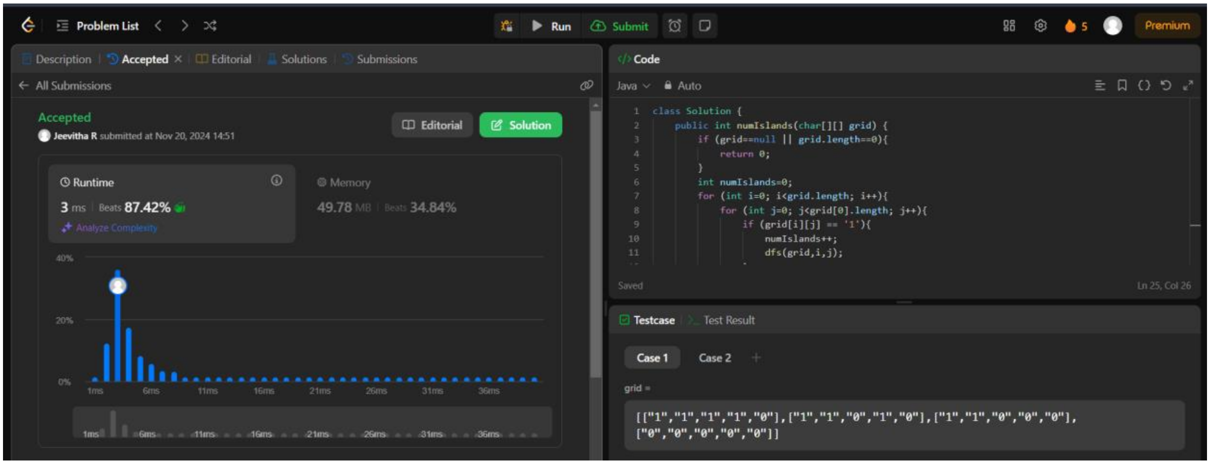
prices =
[7,1,5,3,6,4]

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

6. Number of islands

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

Output



Accepted
Jeevitha R submitted at Nov 20, 2024 14:51

Runtime
3 ms | Beats 87.42%

Memory
49.78 MB | Beats 34.84%

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

grid =

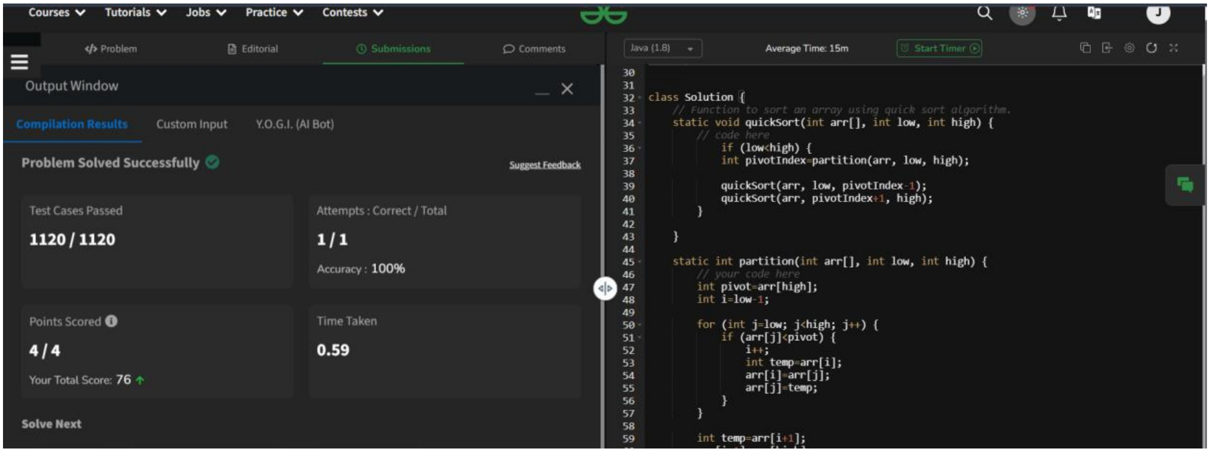
```
[["1","1","1","1","0"],["1","1","0","1","0"],["1","1","0","0","0"],
["0","0","0","0","0"]]
```

Time complexity: O(MXN)
Space complexity: O(MXN)

7. Quick 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 pivotIndex=partition(arr, low, high);

            quickSort(arr, low, pivotIndex-1);
            quickSort(arr, pivotIndex+1, high);
        }

    }

    static int partition(int arr[], int low, int high) {
        // your code here
        int pivot=arr[high];
        int i=low-1;

        for (int j=low; j<high; j++) {
            if (arr[j]<pivot) {
                i++;
                int temp=arr[i];
                arr[i]=arr[j];
                arr[j]=temp;
            }
        }

        int temp=arr[i+1];
        arr[i+1]=arr[high];
        arr[high]=temp;

        return i+1;
    }
}
```

Output



Problem Solved Successfully

Test Cases Passed
1120 / 1120

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

Points Scored
4 / 4
Your Total Score: 76

Time Taken
0.59

Average Time: 15m

```
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 pivotIndex=partition(arr, low, high);

            quickSort(arr, low, pivotIndex-1);
            quickSort(arr, pivotIndex+1, high);
        }

    }

    static int partition(int arr[], int low, int high) {
        // your code here
        int pivot=arr[high];
        int i=low-1;

        for (int j=low; j<high; j++) {
            if (arr[j]<pivot) {
                i++;
                int temp=arr[i];
                arr[i]=arr[j];
                arr[j]=temp;
            }
        }

        int temp=arr[i+1];
```

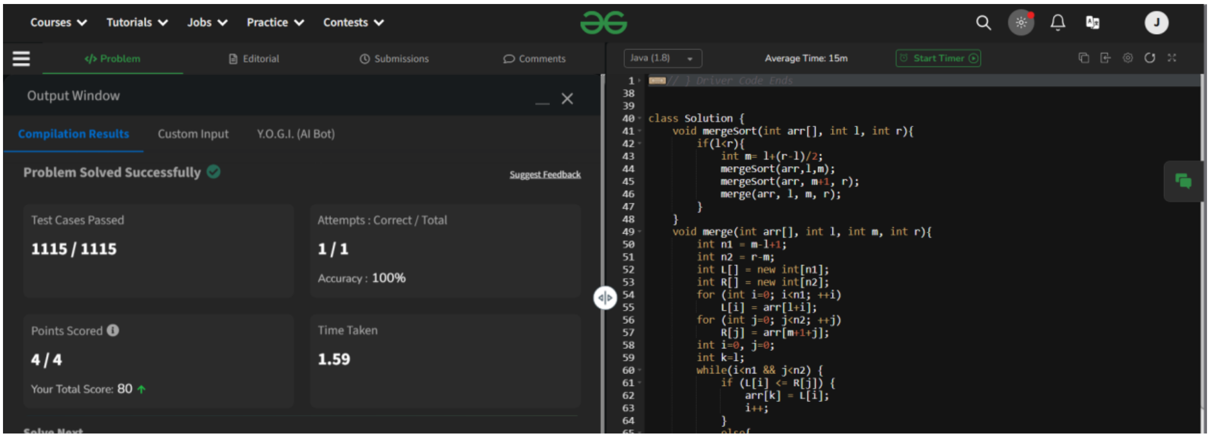
Time complexity: O(n logn)

Space complexity: O(log n)

8. Merge Sort

```
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);
        }
    }
    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++;
        }
    }
    void printArray(int arr[]){
        int n = arr.length;
        for (int i=0; i<n; ++i)
            System.out.print(arr[i] + " ");
        System.out.println();
    }
}
```

Output



Problem Solved Successfully ✓

Test Cases Passed
1115 / 1115

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

Points Scored
4 / 4
Your Total Score: 80 ↑

Time Taken
1.59

Solve Next

```
// } Driver Code Ends


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);
        }
    }
    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{
```

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