

# 1. Binary Search

Time Complexity :  $O(\log n)$

**Binary Search**

Difficulty: Easy Accuracy: 44.32% Submissions: 530K+ Points: 2

Given a sorted array `arr` and an integer `k`, find the position(0-based indexing) at which `k` is present in the array using binary search.

Note: If multiple occurrences are there, please return the smallest index.

**Examples:**

Input: `arr[] = [1, 2, 3, 4, 5]`, `k = 4`  
 Output: 3  
 Explanation: 4 appears at index 3.

Input: `arr[] = [11, 22, 33, 44, 55]`, `k = 445`  
 Output: -1  
 Explanation: 445 is not present.

Note: Try to solve this problem in constant space i.e  $O(1)$

**Constraints:**

- $1 \leq \text{arr.size}() \leq 10^5$
- $1 \leq \text{arr}[i] \leq 10^6$
- $1 \leq k \leq 10^6$

```

1 // Driver Code Starts
2
3
4 // User function Template for Java
5
6 class Solution {
7     // Your code here
8     public int binarysearch(int[] arr, int target) {
9         int left = 0;
10        int right = arr.length - 1;
11        int mid = 0;
12        while (left <= right) {
13            mid = (left+right)/2;
14
15            if(arr[mid] == target){
16                return mid;
17            }
18            else if(target > arr[mid]){
19                left = mid + 1;
20            }
21            else{
22                right = mid - 1;
23            }
24        }
25        return -1;
26    }
27 }
  
```

# 2. Kth smallest Element

Time Complexity :  $O(n)$

**Output Window**

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

Problem Solved Successfully [Suggest Feedback](#)

Test Cases Passed	Attempts : Correct / Total
1110 / 1110	1 / 1
	Accuracy : 100%
Points Scored	Time Taken
4 / 4	0.26
Your Total Score: 6 <span style="color: green;">↑</span>	

**Solve Next**

[Smallest Positive Missing Number](#) [Valid Pair Sum](#) [Optimal Array](#)

```

1 // Driver Code Starts
2
3 // User function Template for Java
4
5 class Solution {
6     // Your code here
7     public static int kthSmallest(int[] arr, int k) {
8         PriorityQueue<Integer> q = new PriorityQueue<>(collections.reverseorder());
9
10        for(int i = 0; i<arr.length;i++){
11            q.offer(arr[i]);
12            if(q.size() > k){
13                q.poll();
14            }
15        }
16        return q.peek();
17    }
18 }
  
```

### 3. Parenthesis Checker

Time Complexity :  $O(n)$

Output Window

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

Problem Solved Successfully [Suggest Feedback](#)

Test Cases Passed	Attempts : Correct / Total
1111 / 1111	1 / 2
	Accuracy : 50%

Points Scored	Time Taken
2 / 2	0.41
Your Total Score: 8 <span>↑</span>	

Solve Next

[Get min at pop](#) [Equal point in a string of brackets](#) [Easy string](#)

```
1 // Driver Code Starts
2
30
31
32
33
34
35
36 class Solution {
37     // Function to check if brackets are balanced or not.
38     static boolean isParenthesisBalanced(String s) {
39         // code here
40         Stack<Character> stack = new Stack<>();
41         for(char i: s.toCharArray()){
42             if(i=='('){
43                 stack.add('(');
44             }
45             else if (i==')'){
46                 stack.add(')');
47             }
48             else if(i=='('){
49                 stack.add(')');
50             }
51             else if(i==')'){
52                 if(!stack.isEmpty() && stack.peek() == i){
53                     stack.pop();
54                 }
55             }
56             else {return false;}
57         }
58         return stack.isEmpty();
59     }
60 }
```

### 4. Minimize the heights - 2

Output Window

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

Problem Solved Successfully [Suggest Feedback](#)

Test Cases Passed	Attempts : Correct / Total
1115 / 1115	1 / 8
	Accuracy : 12%

Points Scored	Time Taken
0 / 4	0.64
Your Total Score: 8	

Solve Next

[Minimum Jumps](#) [A difference of values and indexes](#) [Minimize the Heights I](#)

```
1 // Driver Code Starts
2
34
35
36
37
38
39 // User function Template for
40 int getMinDiff(int[] arr, int k) {
41     int n = arr.length;
42     Arrays.sort(arr);
43
44     int ans = arr[n - 1] - arr[0];
45
46     int tempmin, tempmax;
47     tempmin = arr[0];
48     tempmax = arr[n - 1];
49
50     for (int i = 1; i < n; i++) {
51         if (arr[i] - k < 0)
52             continue;
53
54         tempmin = Math.min(arr[0] + k, arr[i] - k);
55
56         tempmax = Math.max(arr[i - 1] + k, arr[n - 1] - k);
57         ans = Math.min(ans, tempmax - tempmin);
58     }
59     return ans;
60 }
61
62
63 }
```

Time Complexity :  $O(n \log n)$

## 5. Equilibrium point

The screenshot shows a coding platform interface with a dark theme. On the left, a sidebar contains navigation links: Courses, Tutorials, Jobs, Practice, and Contests. The main area is divided into two panels. The left panel displays the problem status: 'Problem Solved Successfully' with a green checkmark. It shows 'Test Cases Passed' as 1111/1111, 'Attempts: Correct / Total' as 1/3, 'Accuracy: 33%', 'Points Scored' as 2/2, and 'Your Total Score: 10'. The right panel shows the code editor with a Java solution for the 'Equilibrium point' problem. The code defines a class 'Solution' with a method 'equilibriumPoint' that finds the equilibrium point in an array. The average time for the problem is 15m.

```
1  // Driver Code Starts
2
39
40
41
42 class Solution {
43     // Function to find equilibrium point in the array.
44     // code here
45     int left_sum = 0;
46     if(arr.length == 1){
47         return 1;
48     }
49     int total_sum = 0;
50     for(int i : arr){
51         total_sum += i;
52     }
53     for(int i = 0; i < arr.length; i++){
54         total_sum -= arr[i];
55         if(left_sum == total_sum){
56             return i+1;
57         }
58         left_sum += arr[i];
59     }
60     return -1;
61 }
62
63
64
```

Time Complexity :  $O(n)$

## 6. Next Greater Element

The screenshot shows a coding platform interface with a dark theme. On the left, a sidebar contains navigation links: Courses, Tutorials, Jobs, Practice, and Contests. The main area is divided into two panels. The left panel displays the problem status: 'Problem Solved Successfully' with a green checkmark. It shows 'Test Cases Passed' as 1110/1110, 'Attempts: Correct / Total' as 1/2, 'Accuracy: 50%', 'Points Scored' as 4/4, and 'Your Total Score: 14'. The right panel shows the code editor with a Java solution for the 'Next Greater Element' problem. The code defines a class 'Solution' with a method 'nextLargerElement' that finds the next greater element for each element in the array using a stack. The average time for the problem is 20m.

```
1  // Driver Code Starts
2
36
37
38
39 class Solution {
40     // Function to find the next greater element for each element of the array.
41     public ArrayList<Integer> nextLargerElement(int[] arr) {
42         ArrayList<Integer> ans = new ArrayList<>();
43         Stack<Integer> stack = new Stack<>();
44
45         for (int i = arr.length - 1; i >= 0; i--) {
46             while (!stack.isEmpty() && stack.peek() <= arr[i]) {
47                 stack.pop();
48             }
49             if (stack.isEmpty()) {
50                 ans.add(-1);
51             } else {
52                 ans.add(stack.peek());
53             }
54             stack.push(arr[i]);
55         }
56         collections.reverse(ans);
57         return ans;
58     }
59 }
60
61
62
```

Time Complexity :  $O(n)$

## 7. Union of two arrays with duplicate elements

The screenshot shows a coding platform interface with the following details:

- Problem:** Union of two arrays with duplicate elements
- Compilation Results:** Problem Solved Successfully
- Test Cases Passed:** 1111 / 1111
- Attempts:** Correct / Total: 1 / 1
- Accuracy:** 100%
- Points Scored:** 2 / 2
- Time Taken:** 0.87
- Your Total Score:** 16
- Solve Next:** Intersection of Two arrays with Distinct Elements, LCM of given array elements, Perfect Squares in a Range

The code on the right is a Java solution using a HashSet to find the union of two arrays:

```
1 // Driver Code Starts
2
3 // User function Template for Java
4
5 class Solution {
6     public static int findUnion(int a[], int b[]) {
7         // code here
8         Set<Integer> set = new HashSet<>();
9         for(int i: a){
10             set.add(i);
11         }
12         for(int j: b){
13             set.add(j);
14         }
15         return set.size();
16     }
17 }
```

Time Complexity :  $O(n+m)$