

1. Given a sorted array of distinct integers and a target value, return the index if the target is found. If not, return the index where it would be if it were inserted in order.

You must write an algorithm with $O(\log n)$ runtime complexity.

Example 1:

Input: `nums = [1,3,5,6]`, `target = 5`

Output: 2

Example 2:

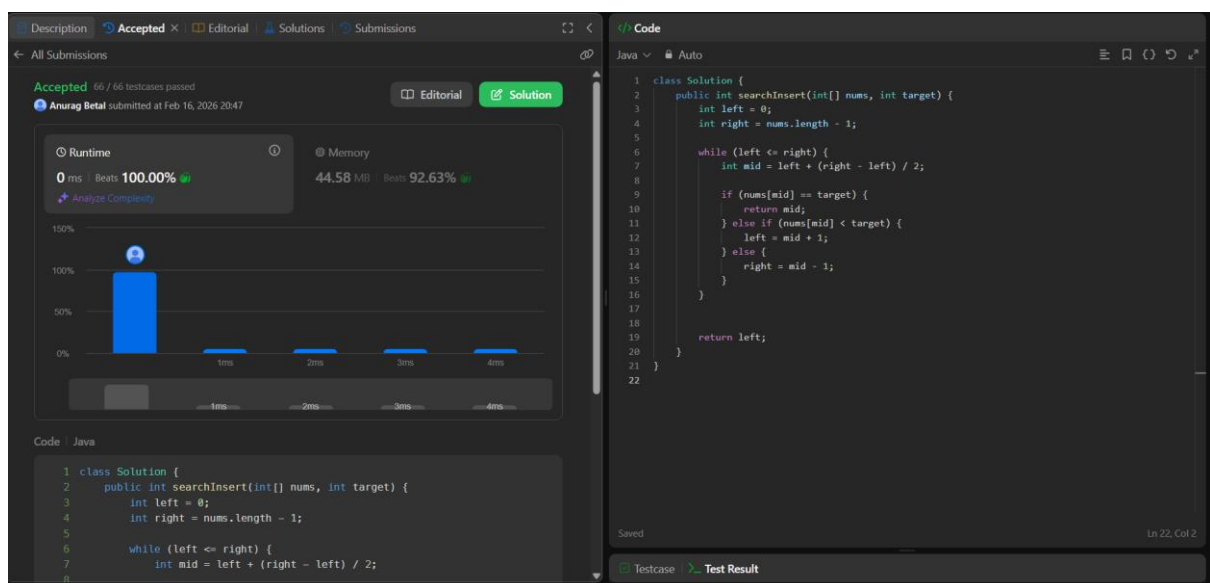
Input: `nums = [1,3,5,6]`, `target = 2`

Output: 1

Example 3:

Input: `nums = [1,3,5,6]`, `target = 7`

Output: 4



The screenshot displays a LeetCode submission for the "Search Insert Position" problem. The interface includes a top navigation bar with tabs for "Description", "Accepted", "Editorial", "Solutions", and "Submissions". The "Accepted" tab is active, showing a submission by "Anurag Botal" submitted on Feb 16, 2026, at 20:47. The submission is marked as "Accepted" with 66/66 testcases passed. The runtime is 0 ms, and the memory is 44.58 MB. The code is written in Java and implements a binary search algorithm. The code is as follows:

```
class Solution {
    public int searchInsert(int[] nums, int target) {
        int left = 0;
        int right = nums.length - 1;
        while (left <= right) {
            int mid = left + (right - left) / 2;
            if (nums[mid] == target) {
                return mid;
            } else if (nums[mid] < target) {
                left = mid + 1;
            } else {
                right = mid - 1;
            }
        }
        return left;
    }
}
```

2. Given an array of distinct integers candidates and a target integer target, return a list of

all unique combinations of candidates where the chosen numbers sum to target. You

may return the combinations in any order.

The same number may be chosen from candidates an unlimited number of times.

Two combinations are unique if the frequency of at least one of the chosen numbers is different.

The test cases are generated such that the number of unique combinations that sum up to target is less than 150 combinations for the given input.

Example 1:

Input: candidates = [2,3,6,7], target = 7

Output: [[2,2,3],[7]]

Explanation:

2 and 3 are candidates, and $2 + 2 + 3 = 7$. Note that 2 can be used multiple times.

7 is a candidate, and $7 = 7$.

These are the only two combinations.

Example 2:

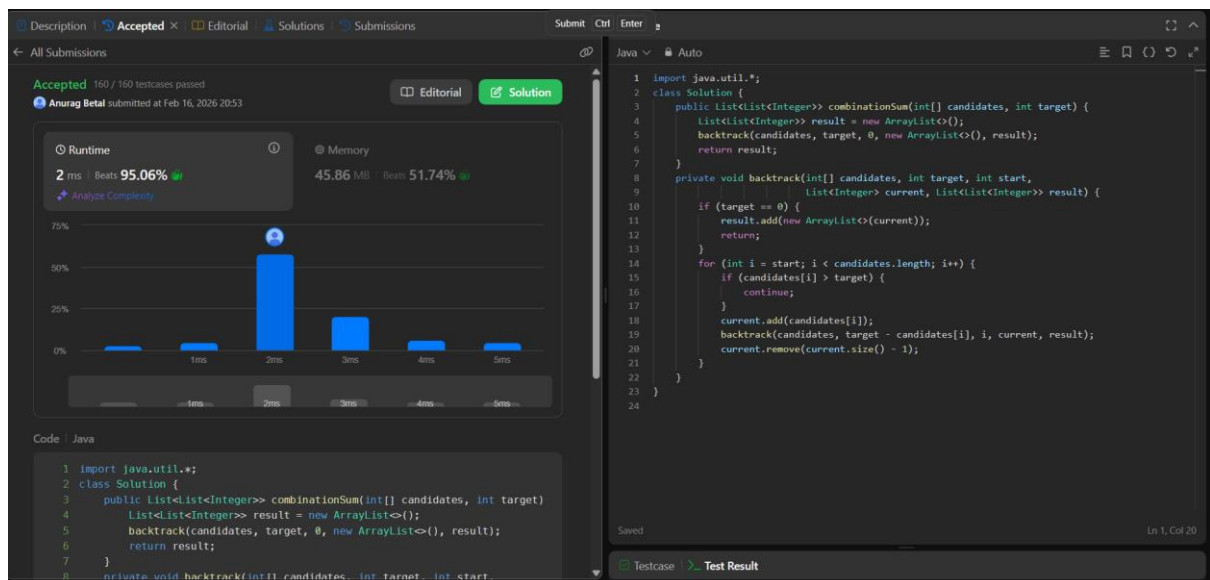
Input: candidates = [2,3,5], target = 8

Output: [[2,2,2,2],[2,3,3],[3,5]]

Example 3:

Input: candidates = [2], target = 1

Output: []



3. Given a collection of candidate numbers (candidates) and a target number (target), find all unique combinations in candidates where the candidate numbers sum to target.

Each number in candidates may only be used once in the combination.

Note: The solution set must not contain duplicate combinations.

Example 1:

Input: candidates = [10,1,2,7,6,1,5], target = 8

Output:

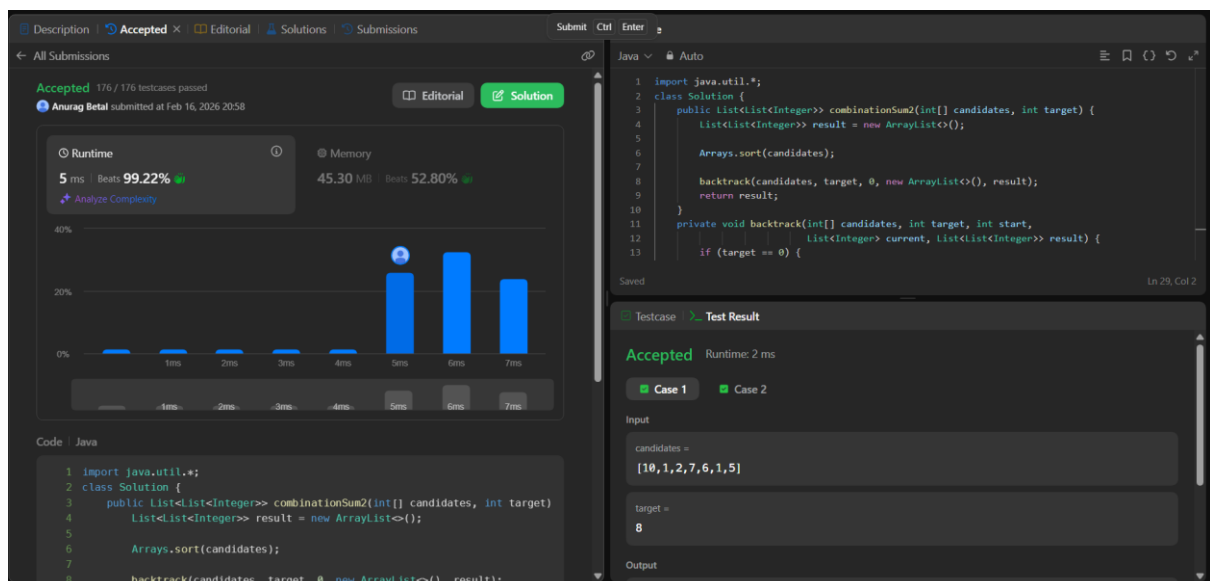
```
[  
  [1,1,6],  
  [1,2,5],  
  [1,7],  
  [2,6]  
]
```

Example 2:

Input: candidates = [2,5,2,1,2], target = 5

Output:

```
[  
  [1,2,2],  
  [5]  
]
```



4. You are given a 0-indexed array of integers nums of length n. You are initially

positioned at index 0.

Each element `nums[i]` represents the maximum length of a forward jump from index `i`.

In other words, if you are at index `i`, you can jump to any index `(i + j)` where:

- $0 \leq j \leq \text{nums}[i]$ and
- $i + j < n$

Return the minimum number of jumps to reach index `n - 1`. The test cases are generated such that you can reach index `n - 1`.

Example 1:

Input: `nums = [2,3,1,1,4]`

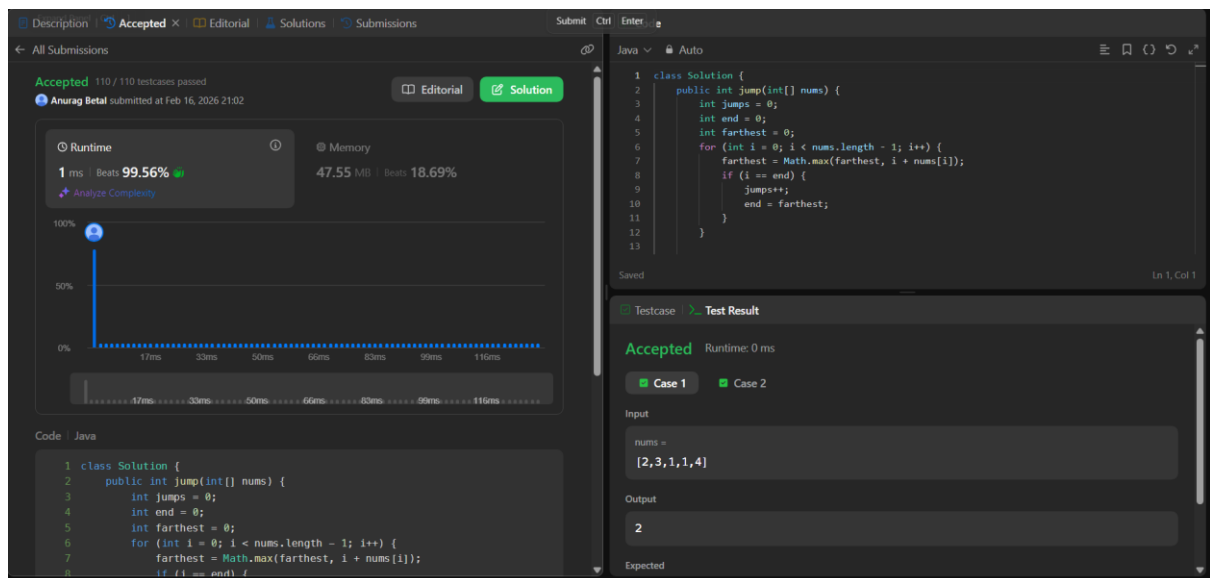
Output: 2

Explanation: The minimum number of jumps to reach the last index is 2. Jump 1 step from index 0 to 1, then 3 steps to the last index.

Example 2:

Input: `nums = [2,3,0,1,4]`

Output: 2



5. Given an array of strings `strs`, group the anagrams together. You can return the answer in any order.

Example 1:

Input: strs = ["eat","tea","tan","ate","nat","bat"]

Output: [["bat"],["nat","tan"],["ate","eat","tea"]]

Explanation:

- There is no string in strs that can be rearranged to form "bat".
- The strings "nat" and "tan" are anagrams as they can be rearranged to form each other.
- The strings "ate", "eat", and "tea" are anagrams as they can be rearranged to form each other.

Example 2:

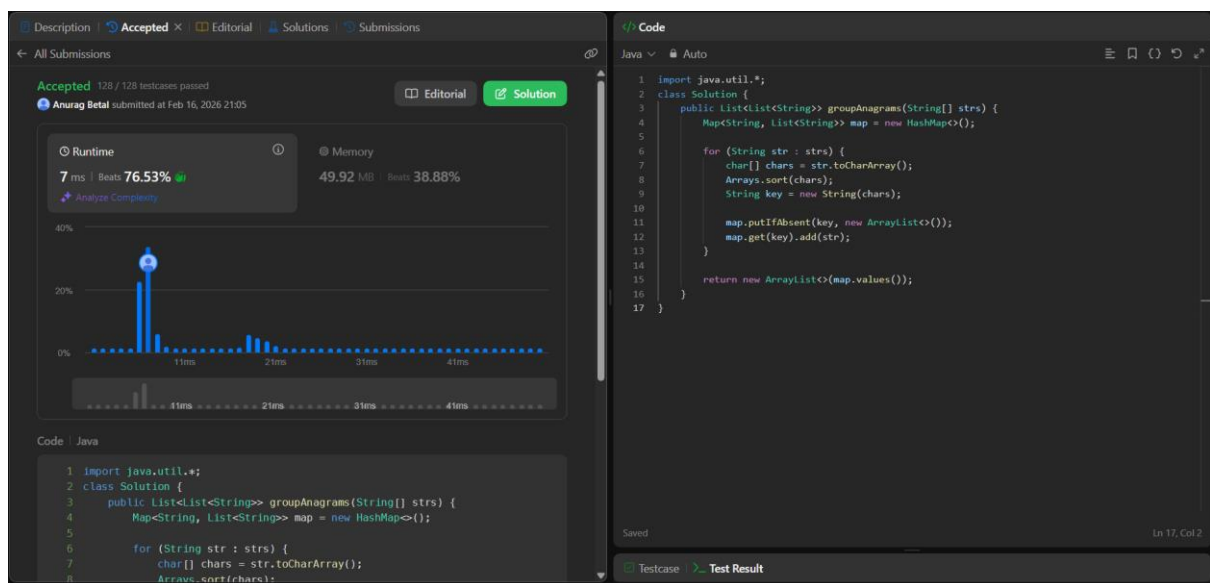
Input: strs = [""]

Output: [[""]]

Example 3:

Input: strs = ["a"]

Output: [["a"]]



6. You are given a large integer represented as an integer array digits, where each digits[i] is the ith digit of the integer. The digits are ordered from most significant to least significant in left-to-right order. The large integer does not contain any

leading 0's.

Increment the large integer by one and return the resulting array of digits.

Example 1:

Input: digits = [1,2,3]

Output: [1,2,4]

Explanation: The array represents the integer 123.

Incrementing by one gives $123 + 1 = 124$.

Thus, the result should be [1,2,4].

Example 2:

Input: digits = [4,3,2,1]

Output: [4,3,2,2]

Explanation: The array represents the integer 4321.

Incrementing by one gives $4321 + 1 = 4322$.

Thus, the result should be [4,3,2,2].

Example 3:

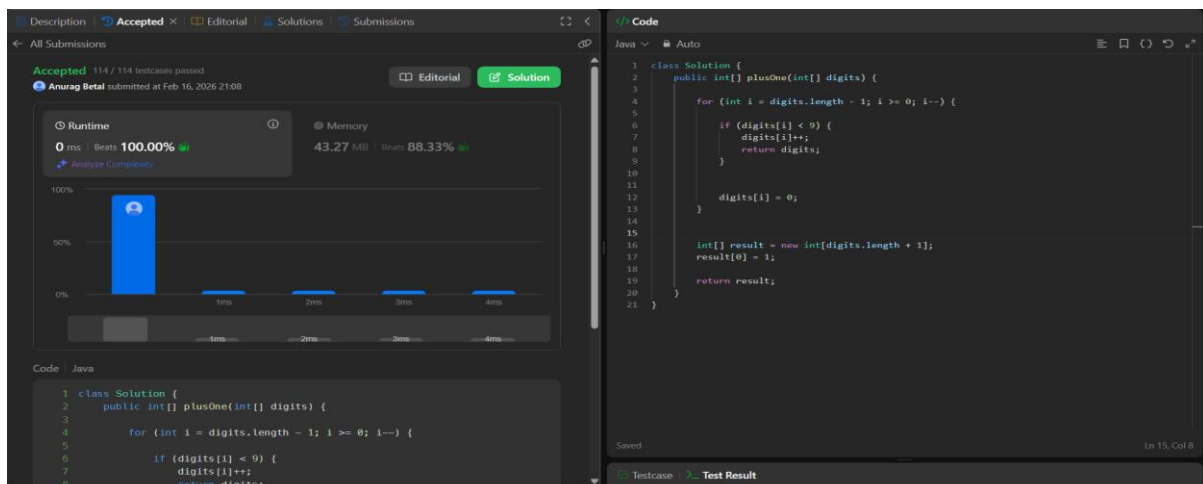
Input: digits = [9]

Output: [1,0]

Explanation: The array represents the integer 9.

Incrementing by one gives $9 + 1 = 10$.

Thus, the result should be [1,0]



```
class Solution {
    public int[] plusOne(int[] digits) {
        for (int i = digits.length - 1; i >= 0; i--) {
            if (digits[i] < 9) {
                digits[i]++;
                return digits;
            }
            digits[i] = 0;
        }
        int[] result = new int[digits.length + 1];
        result[0] = 1;
        return result;
    }
}
```

7. Given an $m \times n$ integer matrix `matrix`, if an element is 0, set its entire row and column to 0's.

You must do it in place.

Example 1:

| | | | | | | |
|---|---|---|---|---|---|---|
| 1 | 1 | 1 | | 1 | 0 | 1 |
| 1 | 0 | 1 | → | 0 | 0 | 0 |
| 1 | 1 | 1 | | 1 | 0 | 1 |

Input: `matrix = [[1,1,1],[1,0,1],[1,1,1]]`

Output: `[[1,0,1],[0,0,0],[1,0,1]]`

Example 2:

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 0 | | 0 | 0 | 0 |
| 3 | 4 | 5 | 2 | → | 0 | 4 | 5 |
| 1 | 3 | 1 | 5 | | 0 | 3 | 1 |

Input: `matrix = [[0,1,2,0],[3,4,5,2],[1,3,1,5]]`

Output: `[[0,0,0,0],[0,4,5,0],[0,3,1,0]]`

Description

Accepted

Editorial

Solutions

Submissions

← All Submissions

Accepted 202 / 202 testcases passed

Anurag Betal submitted at Feb 16, 2026 21:14

Editorial

Solution

Runtime

1 ms Beats 60.16%

Memory

47.40 MB Beats 70.68%

75%

50%

25%

0%

1ms

2ms

3ms

4ms

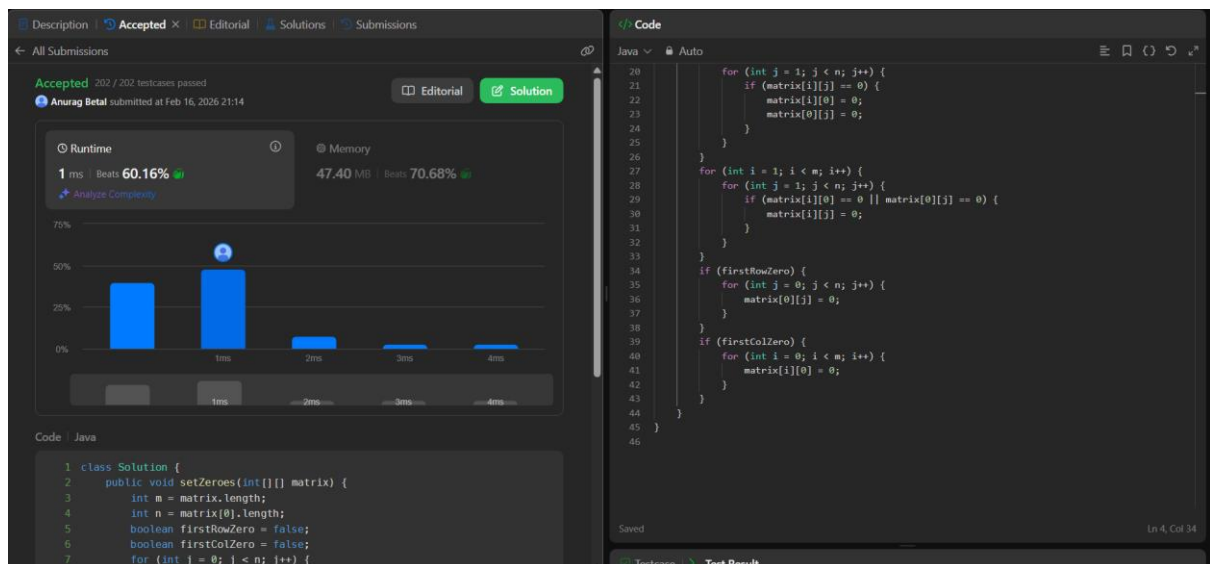
Code

Java

```
1 class Solution {
2     public void setZeroes(int[][] matrix) {
3         int m = matrix.length;
4         int n = matrix[0].length;
5         boolean firstRowZero = false;
6         boolean firstColZero = false;
7         for (int j = 0; j < n; j++) {
8             if (matrix[0][j] == 0) {
9                 firstRowZero = true;
10                break;
11            }
12        }
13        for (int i = 0; i < m; i++) {
14            if (matrix[i][0] == 0) {
15                firstColZero = true;
16                break;
17            }
18        }
19        for (int i = 1; i < m; i++) {
20            for (int j = 1; j < n; j++) {
21                if (matrix[i][j] == 0) {
22                    matrix[i][0] = 0;
23                    matrix[0][j] = 0;
24                }
25            }
26        }
27        for (int i = 1; i < m; i++) {
28            for (int j = 1; j < n; j++) {
29                if (matrix[i][0] == 0 || matrix[0][j] == 0) {
30                    matrix[i][j] = 0;
31                }
32            }
33        }
34    }
35}
```

Testcase

Test Result



8. Given an array `nums` with `n` objects colored red, white, or blue, sort them in-place so that objects of the same color are adjacent, with the colors in the order red, white, and blue.

We will use the integers 0, 1, and 2 to represent the color red, white, and blue, respectively.

You must solve this problem without using the library's sort function.

Example 1:

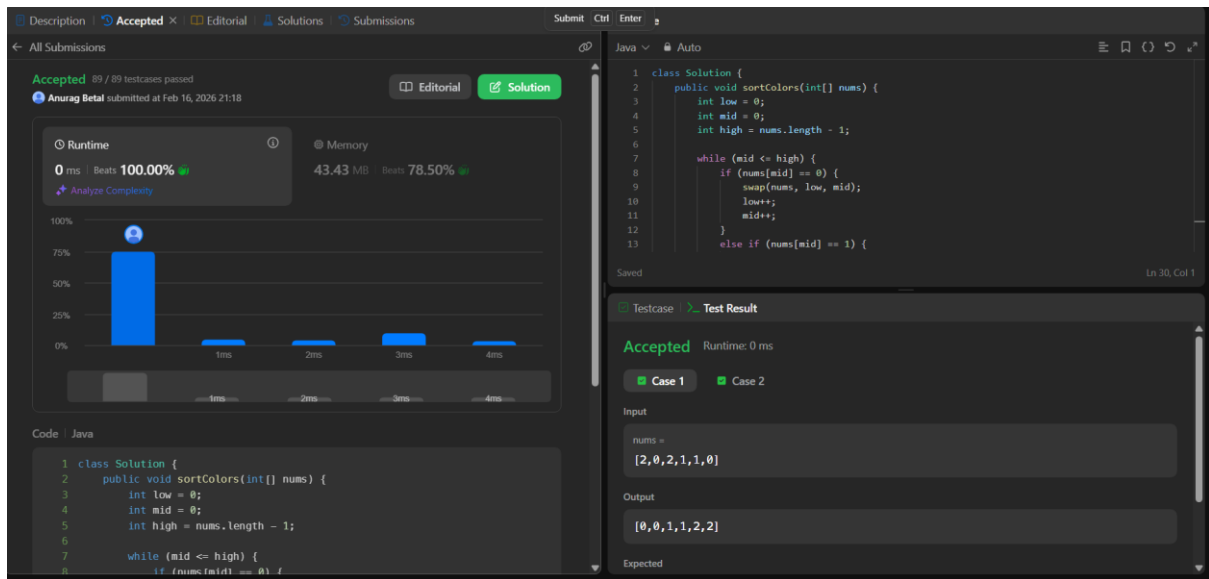
Input: `nums = [2,0,2,1,1,0]`

Output: `[0,0,1,1,2,2]`

Example 2:

Input: `nums = [2,0,1]`

Output: `[0,1,2]`



9. Given an integer array `nums` of unique elements, return all possible subsets (the power set).

The solution set must not contain duplicate subsets. Return the solution in any order.

Example 1:

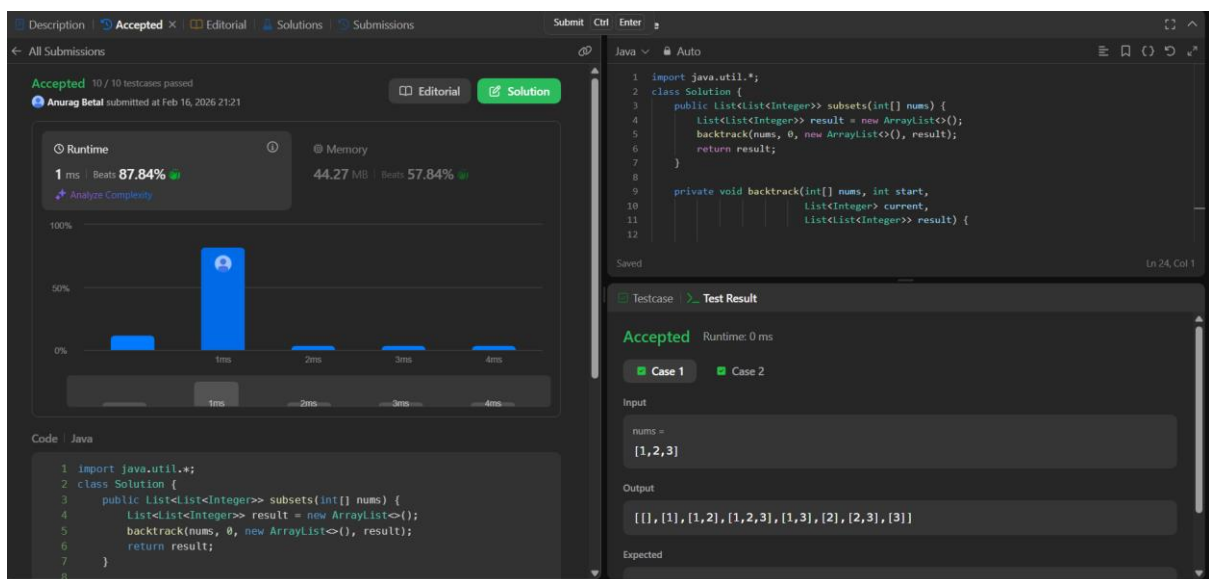
Input: `nums = [1,2,3]`

Output: `[[],[1],[2],[1,2],[3],[1,3],[2,3],[1,2,3]]`

Example 2:

Input: `nums = [0]`

Output: `[[],[0]]`



10. Given an `m x n` grid of characters `board` and a string `word`, return true if `word` exists in

the grid.

The word can be constructed from letters of sequentially adjacent cells, where adjacent cells are horizontally or vertically neighboring. The same letter cell may not be used more than once.

Example 1:

| | | | |
|---|---|---|---|
| A | B | C | E |
| S | F | C | S |
| A | D | E | E |

Input: board = `[["A","B","C","E"],["S","F","C","S"],["A","D","E","E"]]`, word = `"ABCCED"`

Output: true

Example 2:

| | | | |
|---|---|---|---|
| A | B | C | E |
| S | F | C | S |
| A | D | E | E |

Input: board = `[["A","B","C","E"],["S","F","C","S"],["A","D","E","E"]]`, word = `"SEE"`

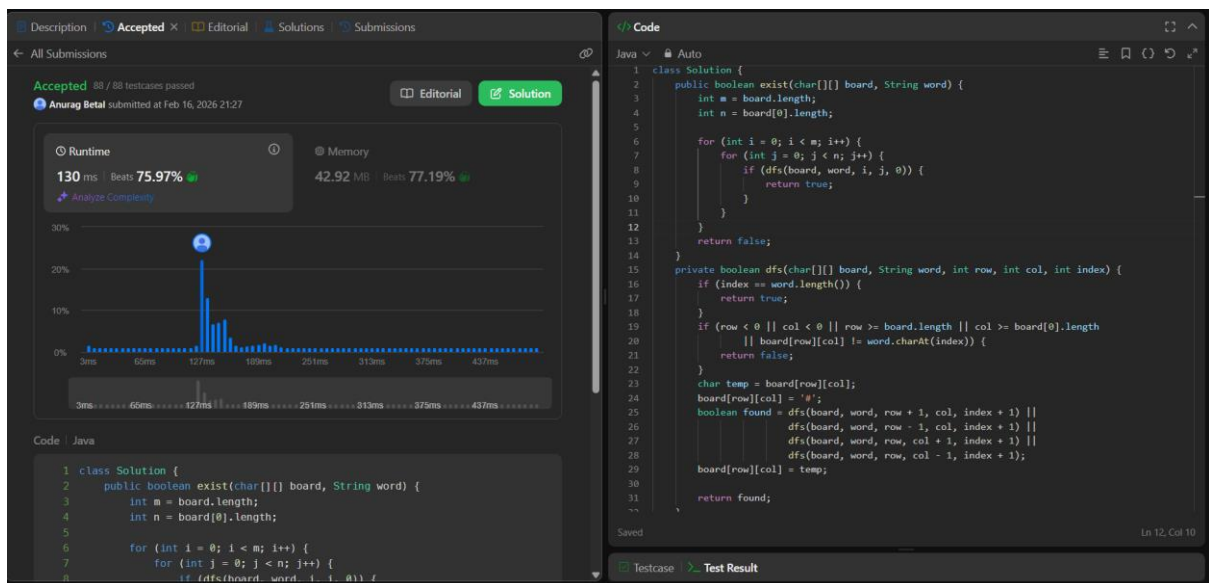
Output: true

Example 3:

| | | | |
|---|---|---|---|
| A | B | C | E |
| S | F | C | S |
| A | D | E | E |

Input: board = [["A","B","C","E"],["S","F","C","S"],["A","D","E","E"]], word = "ABCB"

Output: false



11. Given an array nums of n integers, return an array of all

the unique quadruplets [nums[a], nums[b], nums[c], nums[d]] such that:

- $0 \leq a, b, c, d < n$
- a, b, c, and d are distinct.
- $\text{nums}[a] + \text{nums}[b] + \text{nums}[c] + \text{nums}[d] == \text{target}$

You may return the answer in any order.

Example 1:

Input: nums = [1,0,-1,0,-2,2], target = 0

Output: [[-2,-1,1,2],[-2,0,0,2],[-1,0,0,1]]

Example 2:

Input: nums = [2,2,2,2,2], target = 8

Output: [[2,2,2,2]]

The screenshot shows a LeetCode submission for the 'Four Sum' problem. The submission is accepted, showing a runtime of 19ms (80.52% beats) and memory of 45.90 MB (62.91% beats). The code is in Java and implements a four-sum solution using sorting and a nested loop approach.

```
1 import java.util.*;
2 class Solution {
3     public List<List<Integer>> fourSum(int[] nums, int target) {
4         List<List<Integer>> result = new ArrayList<>();
5         int n = nums.length;
6
7         Arrays.sort(nums);
8
9         for (int i = 0; i < n - 3; i++) {
10
11             // Skip duplicate first elements
12             if (i > 0 && nums[i] == nums[i - 1]) continue;
13
14             for (int j = i + 1; j < n - 2; j++) {
15
16                 // Skip duplicate second elements
17                 if (j > i + 1 && nums[j] == nums[j - 1]) continue;
18
19                 int left = j + 1;
20                 int right = n - 1;
21
22                 while (left < right) {
23                     long sum = (long) nums[i] + nums[j] + nums[left] + nums[right];
24
25                     if (sum == target) {
26                         result.add(Arrays.asList(
27                             nums[i], nums[j], nums[left], nums[right]
28                         ));
29                     }
30
31                     left++;
32                     right--;
33                 }
34             }
35         }
36
37         return result;
38     }
39 }
```

The screenshot shows a LeetCode submission for the 'Four Sum' problem, showing a different implementation of the four-sum solution using a two-pointer approach.

```
1 import java.util.*;
2 class Solution {
3     public List<List<Integer>> fourSum(int[] nums, int target) {
4         List<List<Integer>> result = new ArrayList<>();
5         int n = nums.length;
6
7         Arrays.sort(nums);
8
9         for (int i = 0; i < n - 3; i++) {
10
11             // Skip duplicate first elements
12             if (i > 0 && nums[i] == nums[i - 1]) continue;
13
14             for (int j = i + 1; j < n - 2; j++) {
15
16                 // Skip duplicate second elements
17                 if (j > i + 1 && nums[j] == nums[j - 1]) continue;
18
19                 int left = j + 1;
20                 int right = n - 1;
21
22                 while (left < right) {
23                     long sum = (long) nums[i] + nums[j] + nums[left] + nums[right];
24
25                     if (sum == target) {
26                         result.add(Arrays.asList(
27                             nums[i], nums[j], nums[left], nums[right]
28                         ));
29                     }
30
31                     left++;
32                     right--;
33
34                     while (left < right && nums[left] == nums[left - 1]) left++;
35                     while (left < right && nums[right] == nums[right + 1]) right--;
36                 }
37             }
38         }
39
40         return result;
41     }
42 }
```

12. There is an integer array nums sorted in ascending order (with distinct values).

Prior to being passed to your function, nums is possibly left rotated at an unknown

index k ($1 \leq k < \text{nums.length}$) such that the resulting array is [nums[k], nums[k+1], ...,

nums[n-1], nums[0], nums[1], ..., nums[k-1]] (0-indexed). For

example, [0,1,2,4,5,6,7] might be left rotated by 3 indices and become [4,5,6,7,0,1,2].

Given the array nums after the possible rotation and an integer target, return the index of target if it is in nums, or -1 if it is not in nums.

You must write an algorithm with $O(\log n)$ runtime complexity.

Example 1:

Input: nums = [4,5,6,7,0,1,2], target = 0

Output: 4

Example 2:

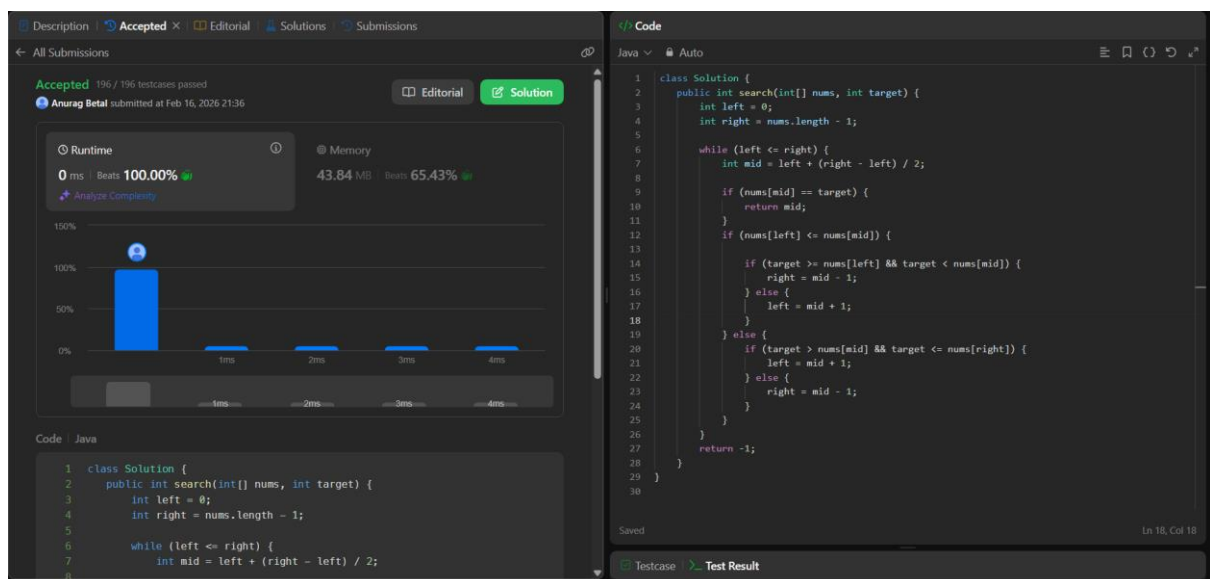
Input: nums = [4,5,6,7,0,1,2], target = 3

Output: -1

Example 3:

Input: nums = [1], target = 0

Output: -1



The screenshot displays a LeetCode submission page for a problem involving a rotated sorted array. The submission is marked as 'Accepted' with 196/196 test cases passed. The user 'Anurag Beral' submitted it on Feb 16, 2026, at 21:36. The runtime is 0ms, beating 100.00% of solutions, and the memory usage is 43.84MB, beating 65.43%. A bar chart shows the runtime performance across different test cases. The code is written in Java and implements a binary search algorithm to find the target in the array.

```
1 class Solution {
2     public int search(int[] nums, int target) {
3         int left = 0;
4         int right = nums.length - 1;
5
6         while (left <= right) {
7             int mid = left + (right - left) / 2;
8
9             if (nums[mid] == target) {
10                 return mid;
11             }
12             if (nums[left] <= nums[mid]) {
13                 if (target >= nums[left] && target < nums[mid]) {
14                     right = mid - 1;
15                 } else {
16                     left = mid + 1;
17                 }
18             } else {
19                 if (target > nums[mid] && target <= nums[right]) {
20                     left = mid + 1;
21                 } else {
22                     right = mid - 1;
23                 }
24             }
25         }
26         return -1;
27     }
28 }
29
30
```

13. Given an array of integers nums sorted in non-decreasing order, find the starting and ending position of a given target value.

If target is not found in the array, return [-1, -1].

You must write an algorithm with $O(\log n)$ runtime complexity.

Example 1:

Input: nums = [5,7,7,8,8,10], target = 8

Output: [3,4]

Example 2:

Input: nums = [5,7,7,8,8,10], target = 6

Output: [-1,-1]

Example 3:

Input: nums = [], target = 0

Output: [-1,-1]

Accepted 88 / 88 testcases passed
Anurag Betal submitted at Feb 16, 2026 21:39

Runtime: 0 ms, Beats 100.00%
Memory: 48.29 MB, Beats 49.95%

Code: Java

```
1 class Solution {
2     public int[] searchRange(int[] nums, int target) {
3         int[] result = new int[2];
4         result[0] = findFirst(nums, target);
5         result[1] = findLast(nums, target);
6         return result;
7     }
8     private int findFirst(int[] nums, int target) {
9         int left = 0, right = nums.length - 1;
10        int index = -1;
11        while (left <= right) {
12            int mid = left + (right - left) / 2;
13            if (nums[mid] == target) {
14                index = mid;
15                right = mid - 1;
16            }
17            else if (nums[mid] < target) {
18                left = mid + 1;
19            }
20            else {
21                right = mid - 1;
22            }
23        }
24        return index;
25    }
26    private int findLast(int[] nums, int target) {
27        int left = 0, right = nums.length - 1;
28        int index = -1;
29        while (left <= right) {
30            int mid = left + (right - left) / 2;
31            if (nums[mid] == target) {
32                left = mid + 1;
33            }
34            else if (nums[mid] < target) {
35                left = mid + 1;
36            }
37            else {
38                right = mid - 1;
39            }
40        }
41        return index;
42    }
43 }
```

Accepted 88 / 88 testcases passed
Anurag Betal submitted at Feb 16, 2026 21:39

Runtime: 0 ms, Beats 100.00%
Memory: 48.29 MB, Beats 49.95%

Code: Java

```
1 class Solution {
2     public int[] searchRange(int[] nums, int target) {
3         int[] result = new int[2];
4         result[0] = findFirst(nums, target);
5         result[1] = findLast(nums, target);
6         return result;
7     }
8     private int findFirst(int[] nums, int target) {
9         int left = 0, right = nums.length - 1;
10        int index = -1;
11        while (left <= right) {
12            int mid = left + (right - left) / 2;
13            if (nums[mid] == target) {
14                index = mid;
15                right = mid - 1;
16            }
17            else if (nums[mid] < target) {
18                left = mid + 1;
19            }
20            else {
21                right = mid - 1;
22            }
23        }
24        return index;
25    }
26    private int findLast(int[] nums, int target) {
27        int left = 0, right = nums.length - 1;
28        int index = -1;
29        while (left <= right) {
30            int mid = left + (right - left) / 2;
31            if (nums[mid] == target) {
32                left = mid + 1;
33            }
34            else if (nums[mid] < target) {
35                left = mid + 1;
36            }
37            else {
38                right = mid - 1;
39            }
40        }
41        return index;
42    }
43 }
```