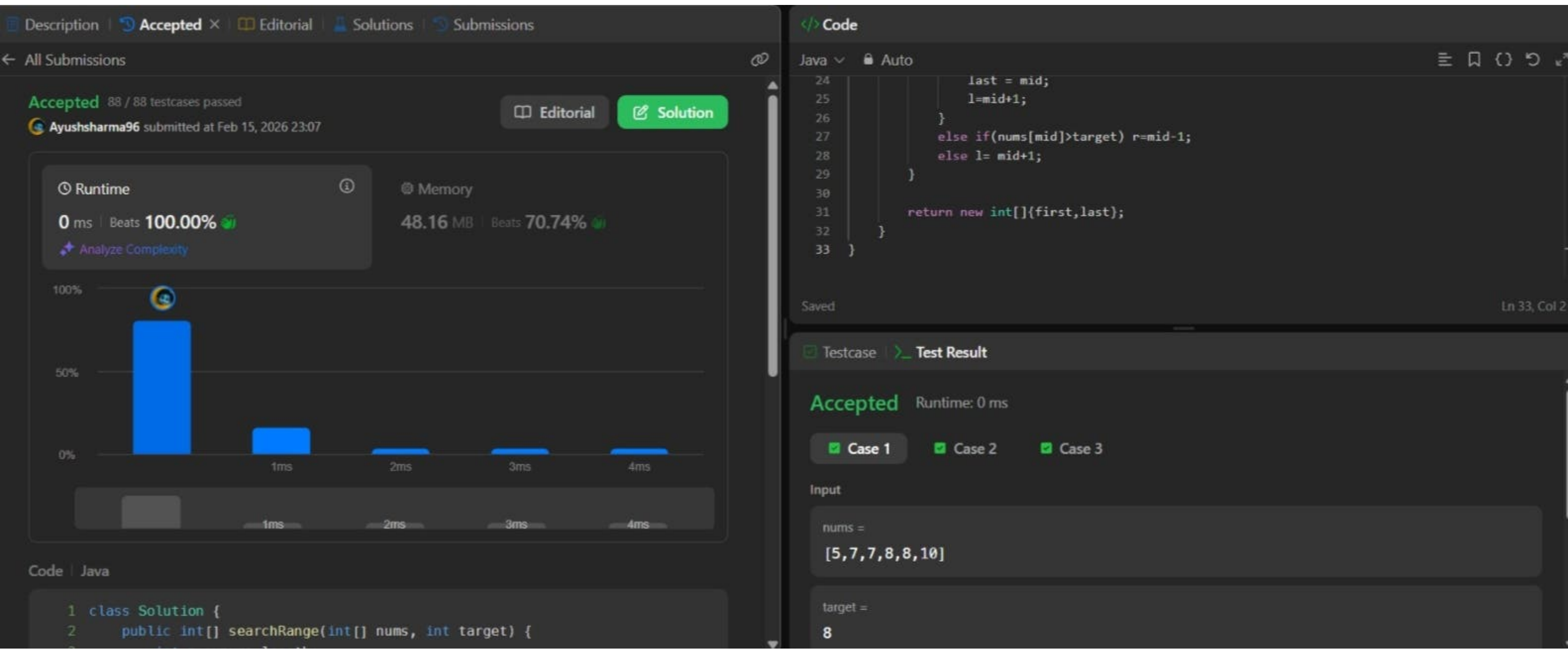


Accepted 88 / 88 testcases passed

Ayushsharma96 submitted at Feb 15, 2026 23:07

Runtime

0 ms | Beats 100.00%

Memory

48.16 MB | Beats 70.74%

Code | Java

```java
class Solution {
    public int[] searchRange(int[] nums, int target) {
```

```java
                last = mid;
                l=mid+1;
            }
            else if(nums[mid]>target) r=mid-1;
            else l= mid+1;
        }

        return new int[]{first,last};
    }
}
```

Saved

Ln 33, Col 2

Accepted Runtime: 0 ms

✓ Case 1 ✓ Case 2 ✓ Case 3

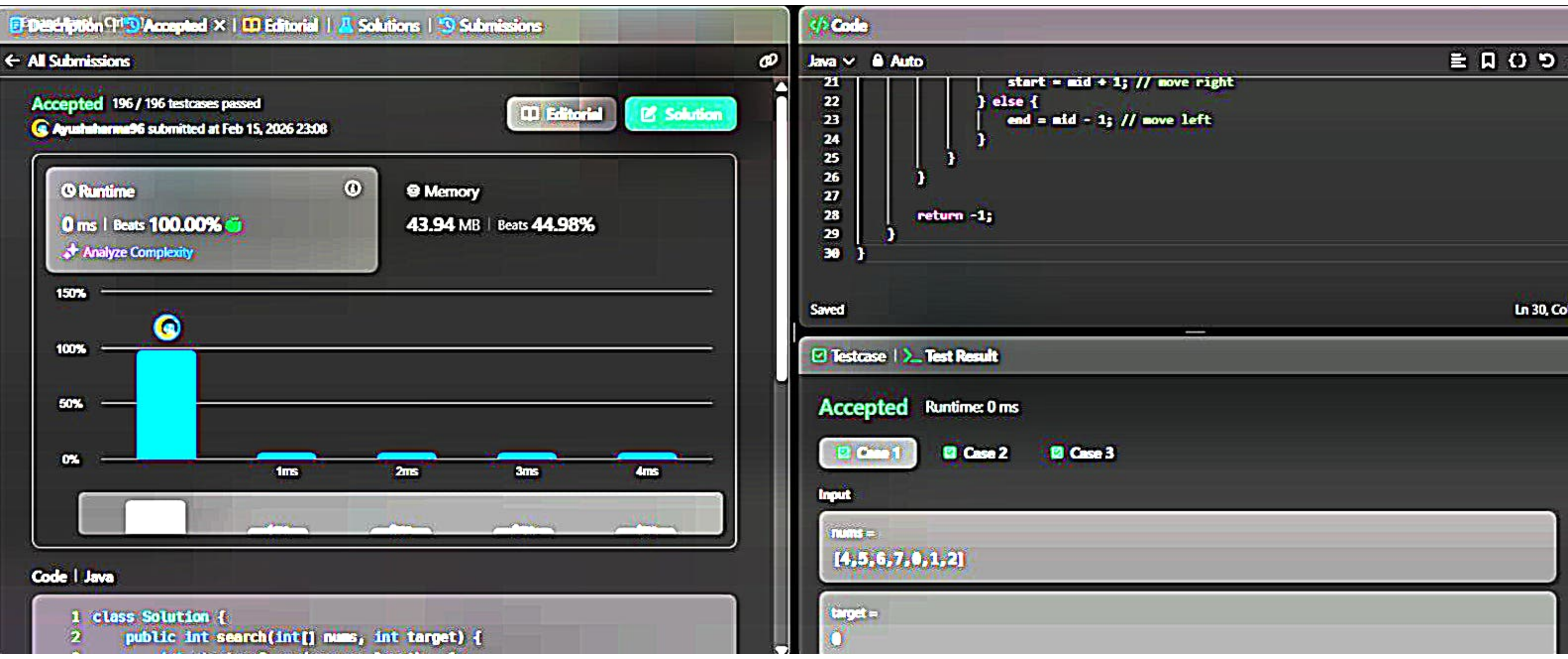
Input

nums =

[5,7,7,8,8,10]

target =

8



Accepted 196 / 196 testcases passed
Ayushsharma96 submitted at Feb 15, 2026 23:08

Runtime
0 ms | Beats 100.00%

Memory
43.94 MB | Beats 44.98%

Code | Java

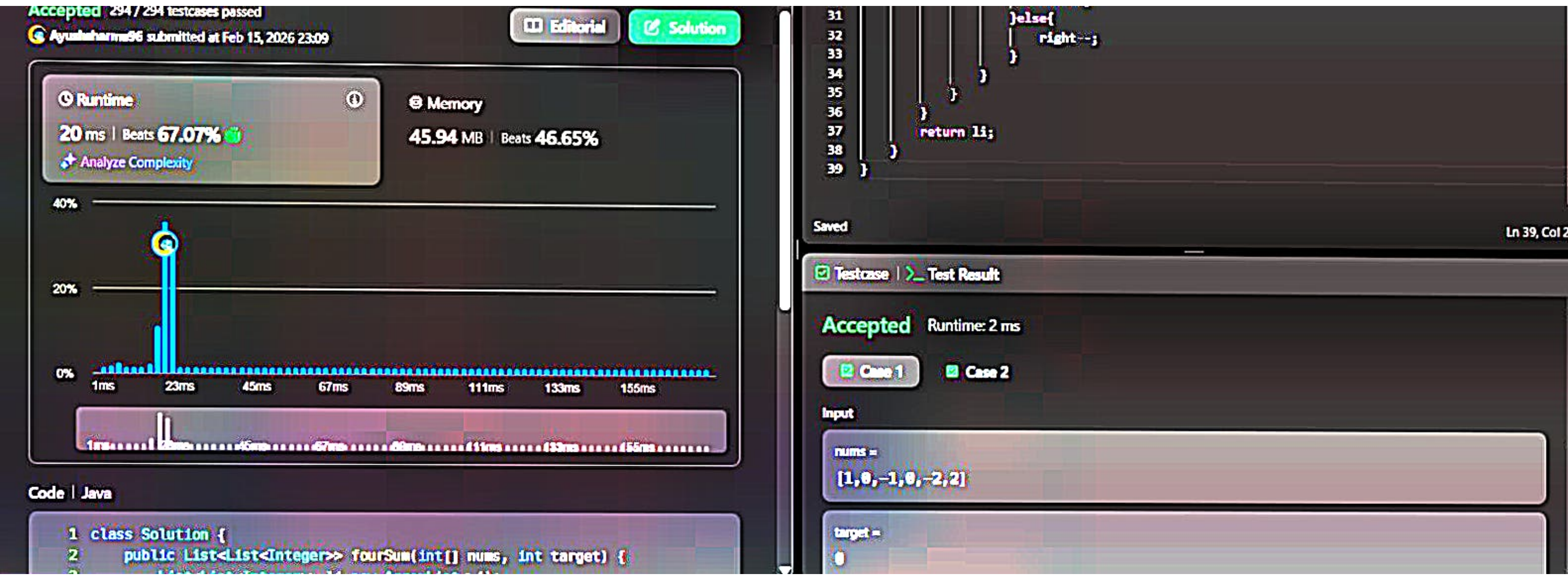
```
class Solution {
    public int search(int[] nums, int target) {
```

```
                    start = mid + 1; // move right
                } else {
                    end = mid - 1; // move left
                }
            }
        }

        return -1;
    }
}
```

Accepted Runtime: 0 ms

Input

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



Accepted 294 / 294 testcases passed

Ayushsharma96 submitted at Feb 15, 2026 23:09

Runtime

20 ms | Beats 67.07%

Memory

45.94 MB | Beats 46.65%

Code | Java

```java
class Solution {
    public List<List<Integer>> fourSum(int[] nums, int target) {
```

```java
                    }else{
                        right--;
                    }
                }
            }
        }
        return li;
    }
}
```

Saved

Ln 39, Col 2

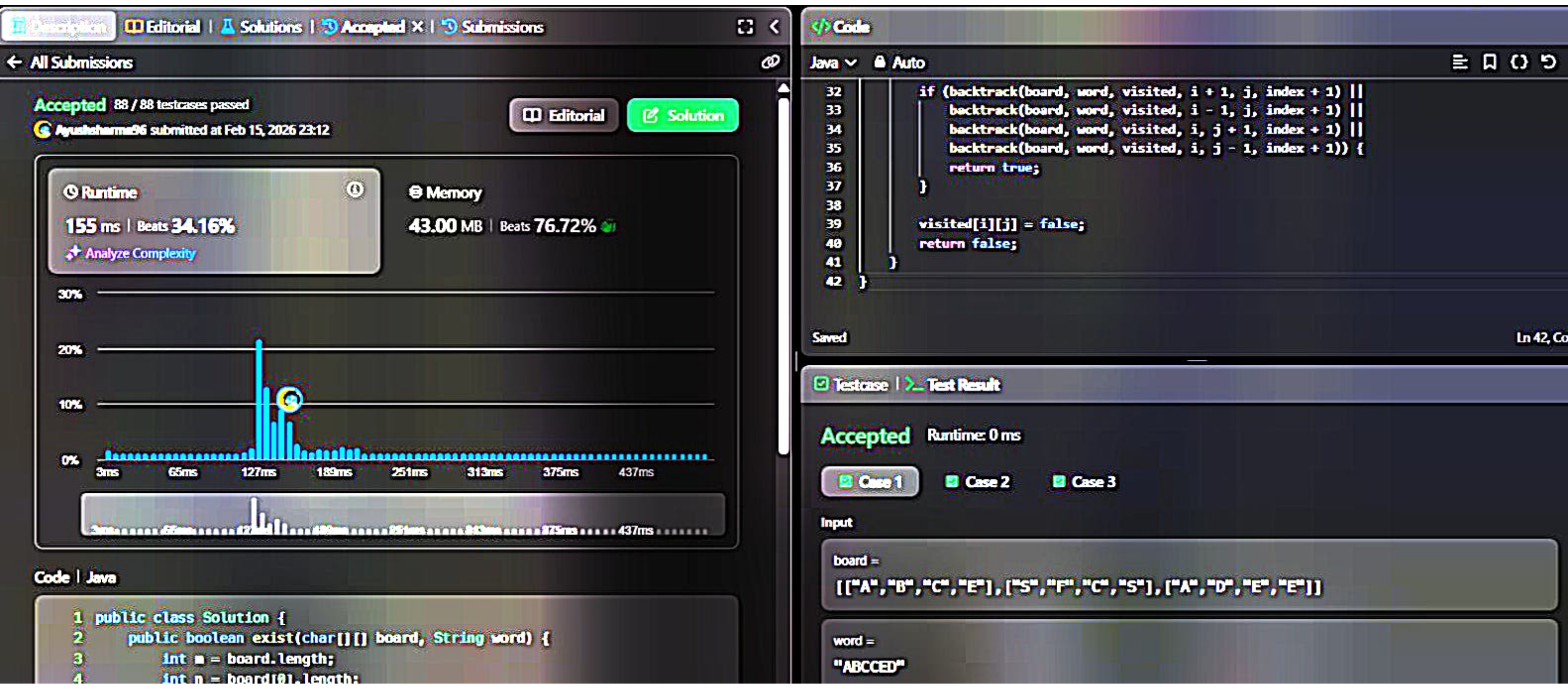
Accepted Runtime: 2 ms

Case 1 Case 2

Input

nums =
[1,0,-1,0,-2,2]

target =
0



Accepted 88 / 88 testcases passed

Ayushsharma96 submitted at Feb 15, 2026 23:12

Runtime

155 ms | Beats 34.16%

Memory

43.00 MB | Beats 76.72%

Code | Java

```java
public class Solution {
    public boolean exist(char[][] board, String word) {
        int m = board.length;
        int n = board[0].length;
```

```java
        if (backtrack(board, word, visited, i + 1, j, index + 1) ||
            backtrack(board, word, visited, i - 1, j, index + 1) ||
            backtrack(board, word, visited, i, j + 1, index + 1) ||
            backtrack(board, word, visited, i, j - 1, index + 1)) {
            return true;
        }

        visited[i][j] = false;
        return false;
    }
}
```

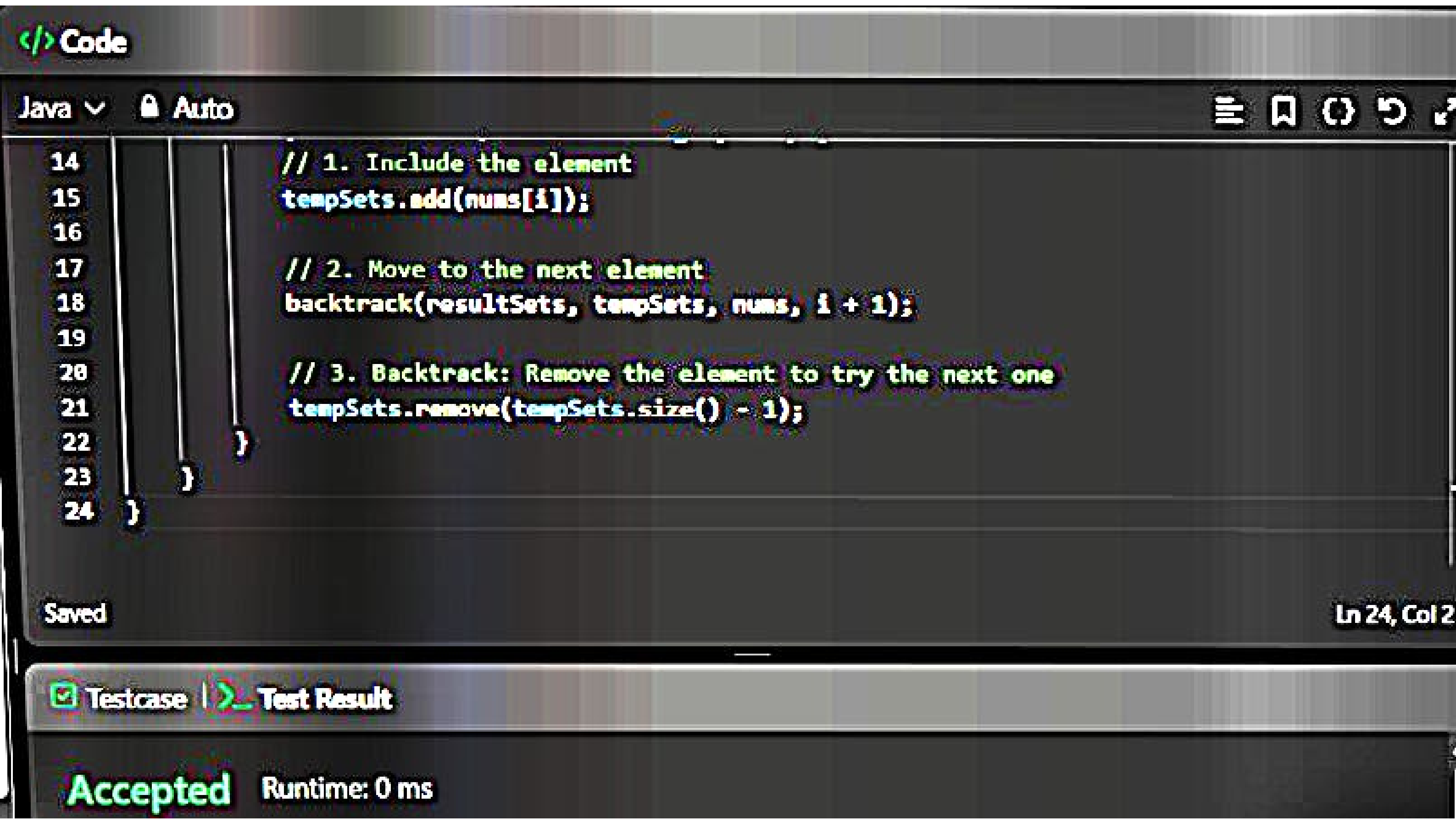
Accepted Runtime: 0 ms

Case 1 Case 2 Case 3

Input

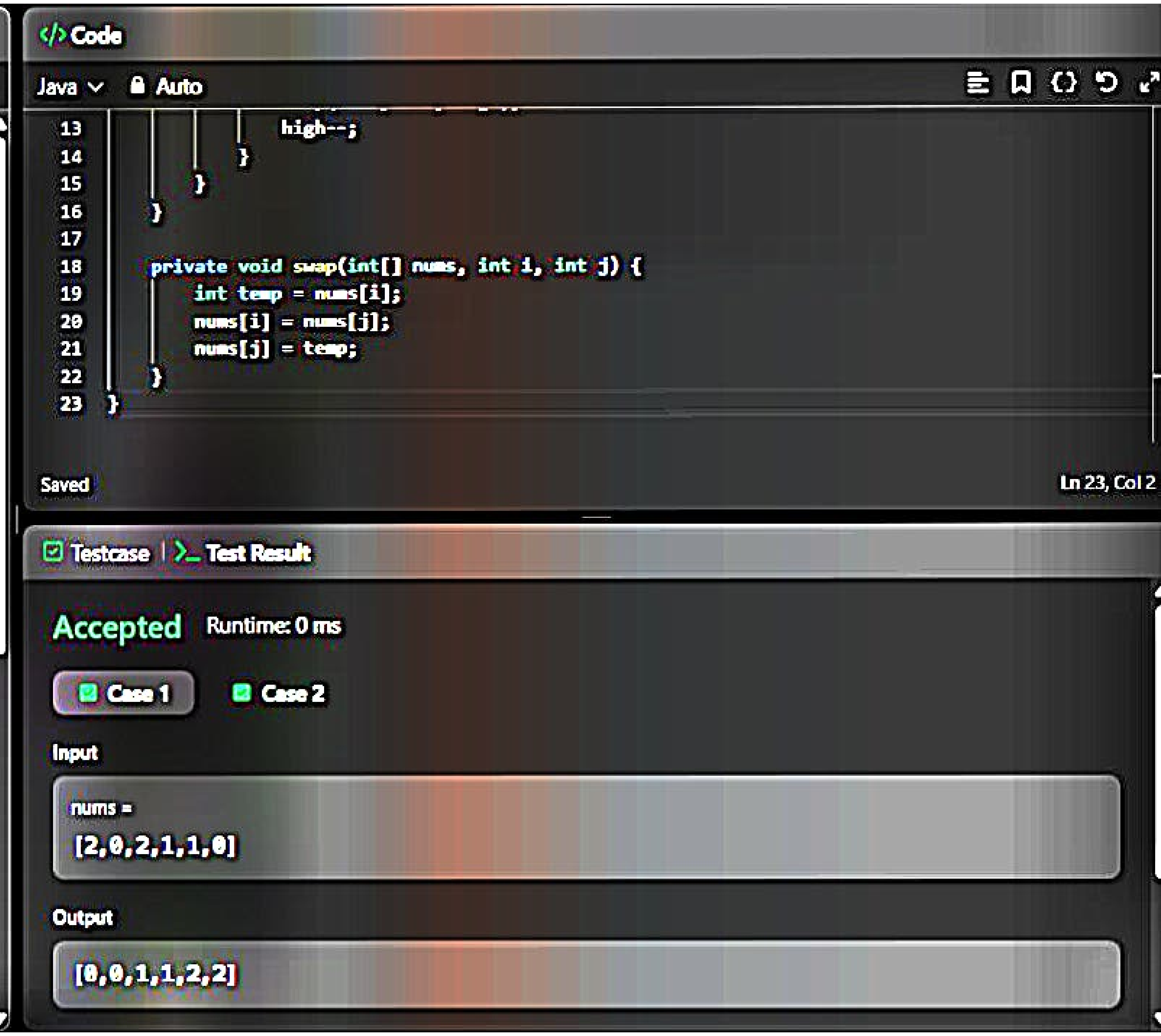
board =
[["A","B","C","E"],["S","F","C","S"],["A","D","E","E"]]

word =
"ABCCED"



```java
            // 1. Include the element
            tempSets.add(nums[i]);

            // 2. Move to the next element
            backtrack(resultSets, tempSets, nums, i + 1);

            // 3. Backtrack: Remove the element to try the next one
            tempSets.remove(tempSets.size() - 1);
        }
    }
}
```

Accepted Runtime: 0 ms



```java
                high--;
            }
        }
    }

    private void swap(int[] nums, int i, int j) {
        int temp = nums[i];
        nums[i] = nums[j];
        nums[j] = temp;
    }
}
```

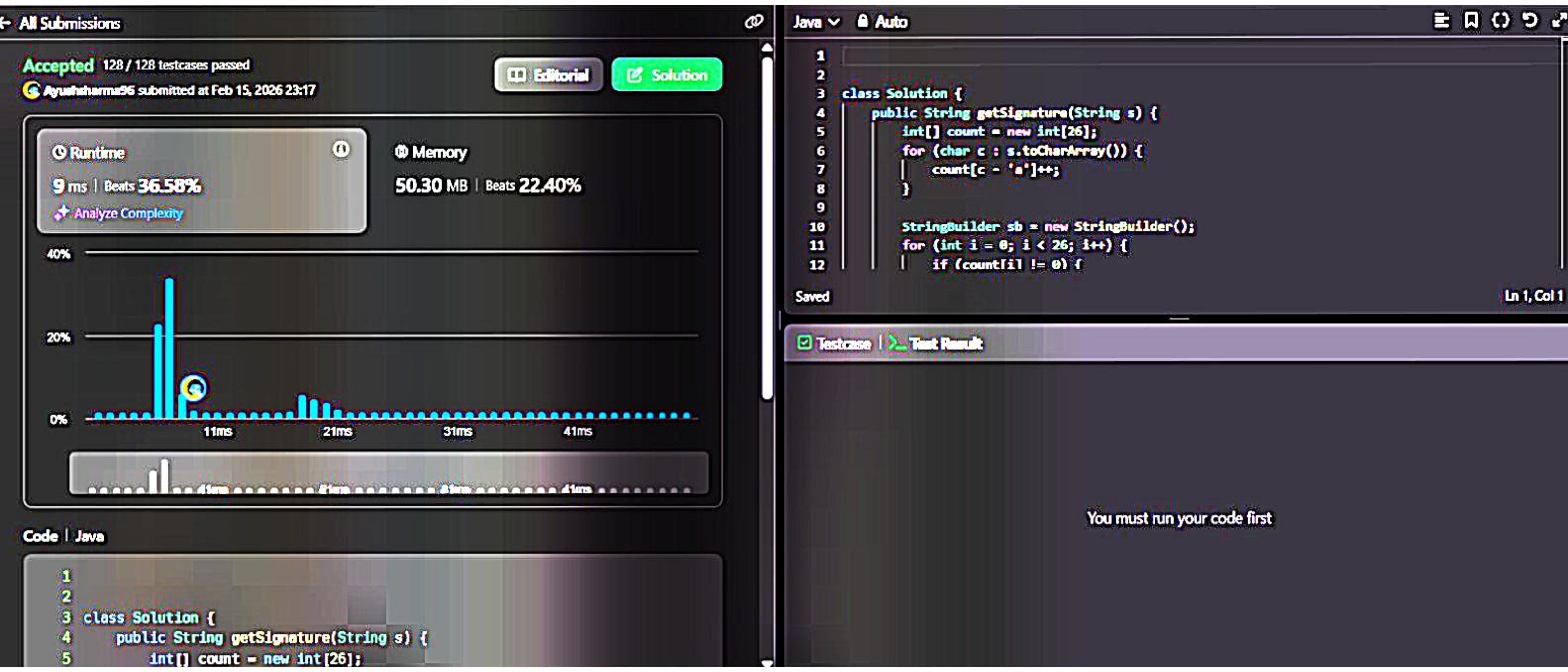
Accepted Runtime: 0 ms

Input

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

Output

```
[0,0,1,1,2,2]
```



Accepted 128 / 128 testcases passed

Ayushsharma96 submitted at Feb 15, 2026 23:17

Runtime

9 ms | Beats 36.58%

Memory

50.30 MB | Beats 22.40%

Code | Java

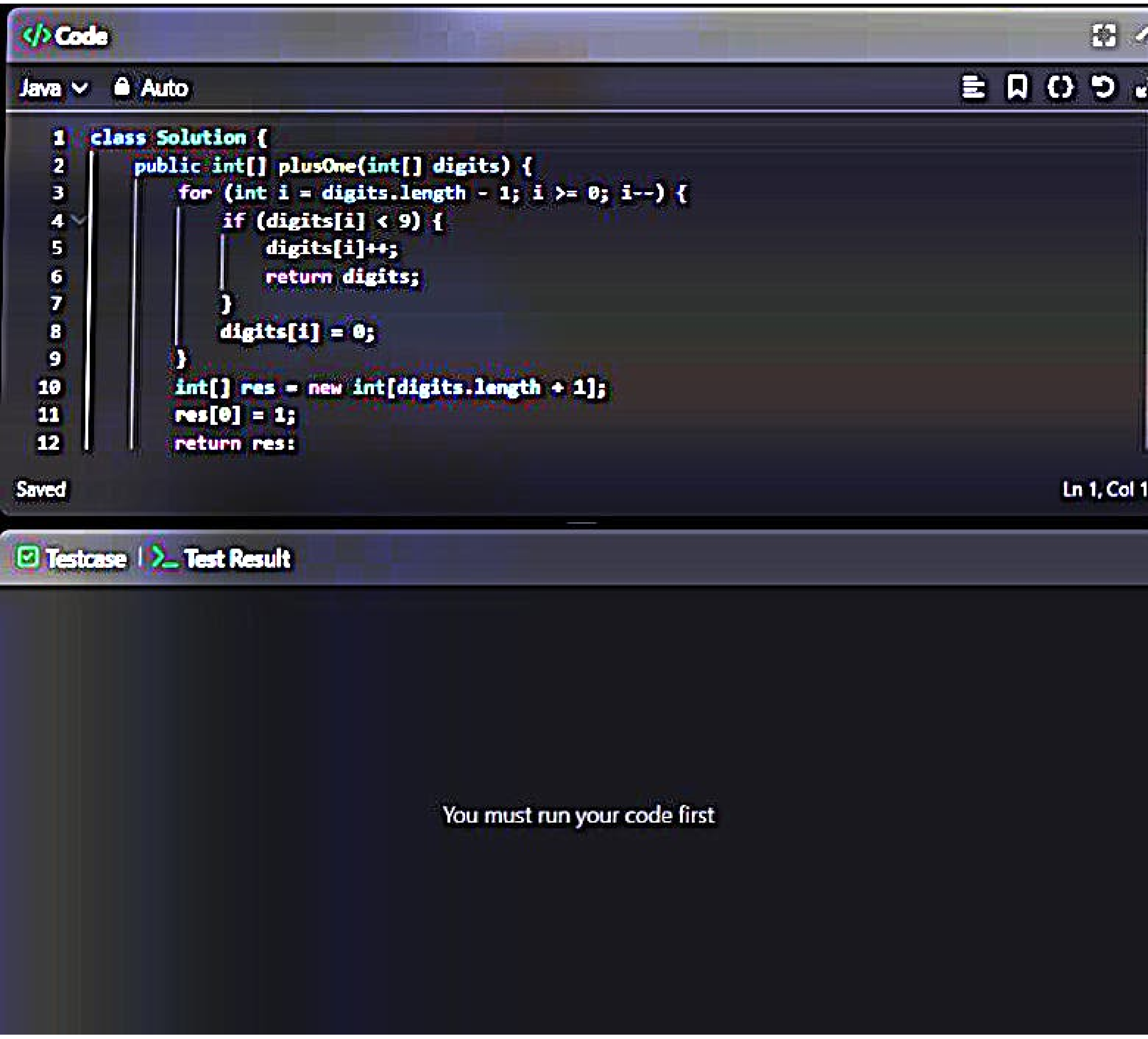
```
class Solution {
    public String getSignature(String s) {
        int[] count = new int[26];
```

```
class Solution {
    public String getSignature(String s) {
        int[] count = new int[26];
        for (char c : s.toCharArray()) {
            count[c - 'a']++;
        }

        StringBuilder sb = new StringBuilder();
        for (int i = 0; i < 26; i++) {
            if (count[i] != 0) {
```

Saved

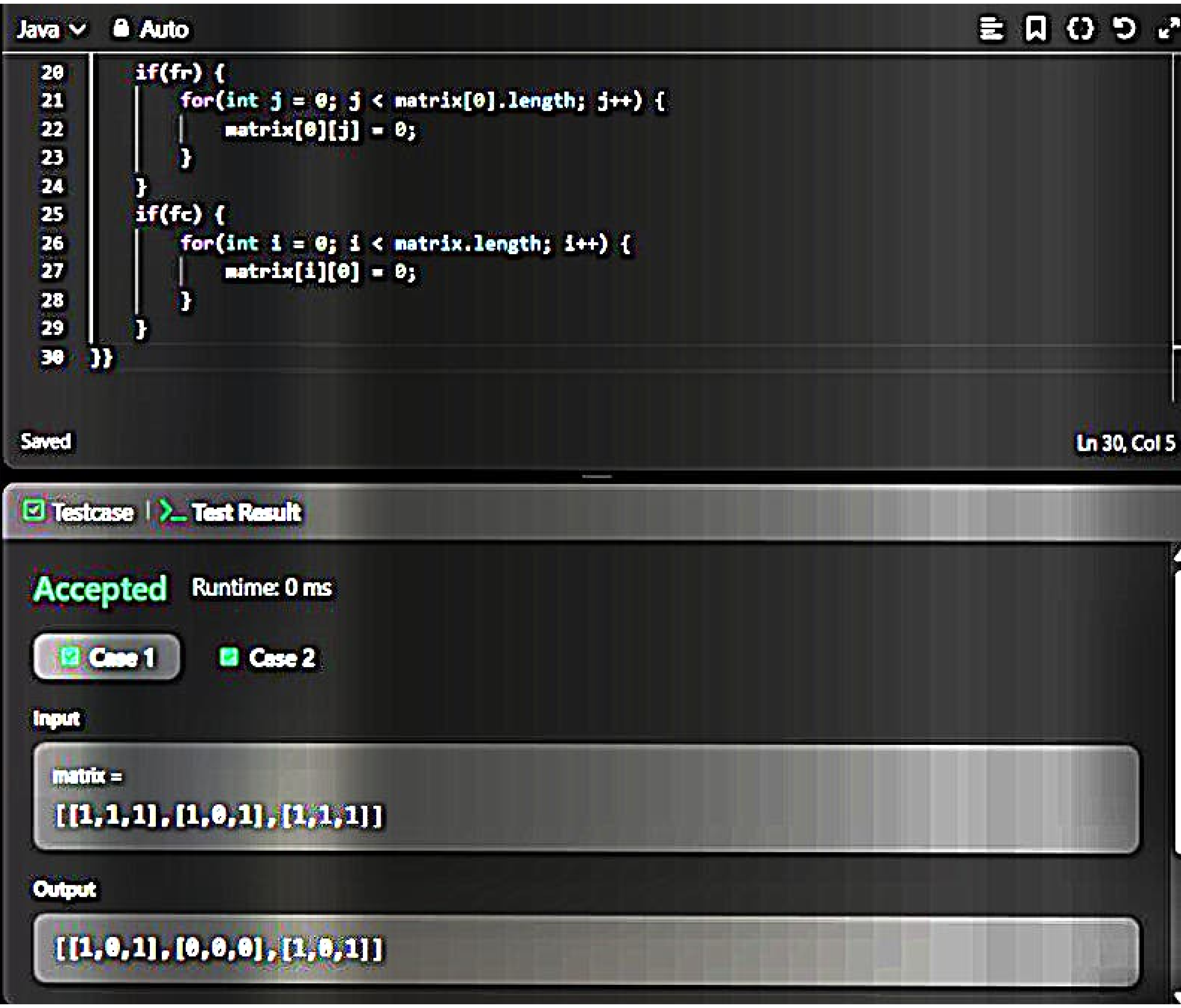
Ln 1, Col 1

You must run your code first



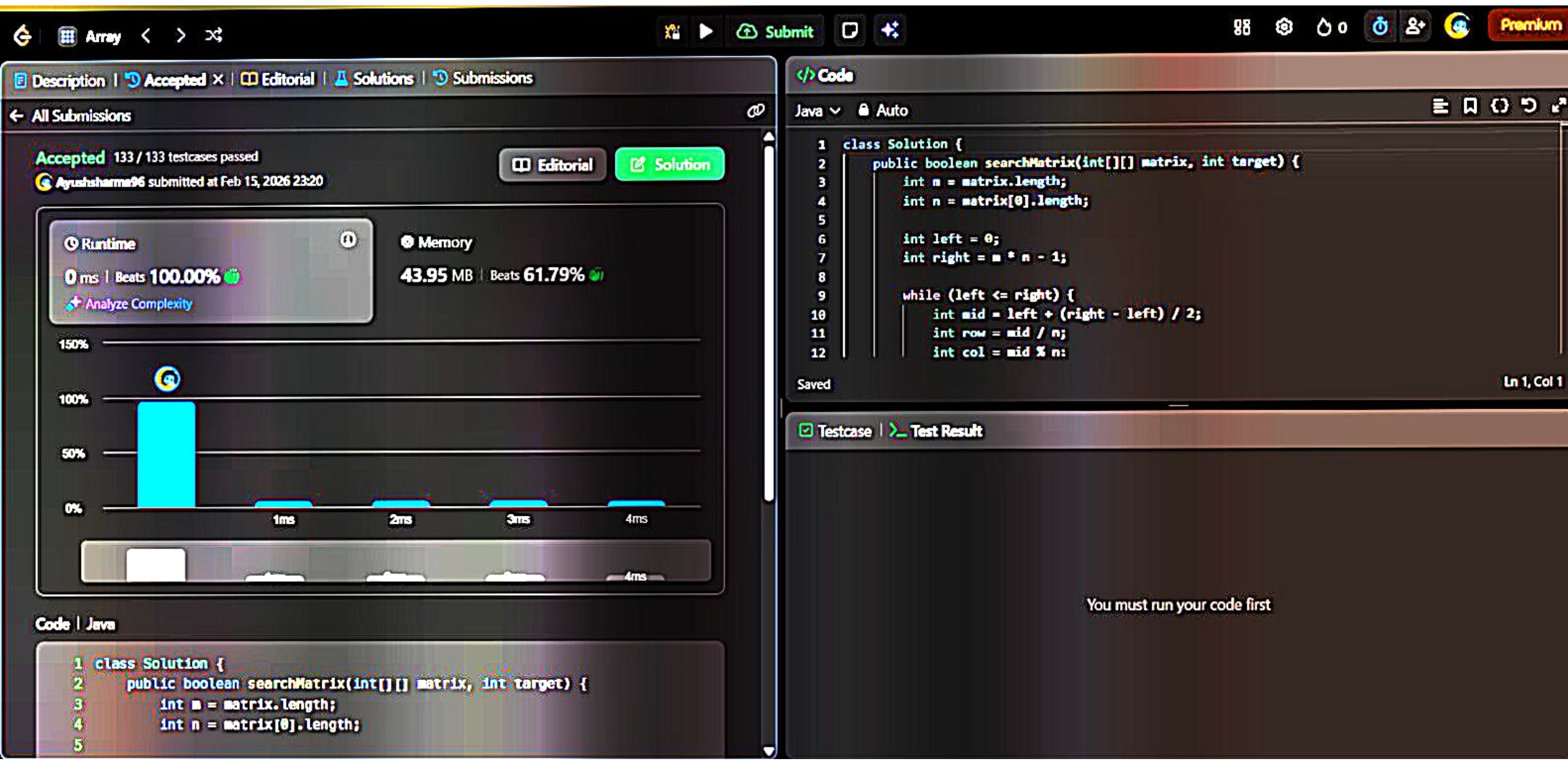
```java
class Solution {
    public int[] plusOne(int[] digits) {
        for (int i = digits.length - 1; i >= 0; i--) {
            if (digits[i] < 9) {
                digits[i]++;
                return digits;
            }
            digits[i] = 0;
        }
        int[] res = new int[digits.length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

You must run your code first



```java
    if(fr) {
        for(int j = 0; j < matrix[0].length; j++) {
            matrix[0][j] = 0;
        }
    }
    if(fc) {
        for(int i = 0; i < matrix.length; i++) {
            matrix[i][0] = 0;
        }
    }
}}
```

```
Accepted   Runtime: 0 ms

Case 1   Case 2

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

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



Accepted 133 / 133 testcases passed

Ayushsharma96 submitted at Feb 15, 2026 23:20

Runtime: 0 ms | Beats 100.00%

Memory: 43.95 MB | Beats 61.79%

Code | Java

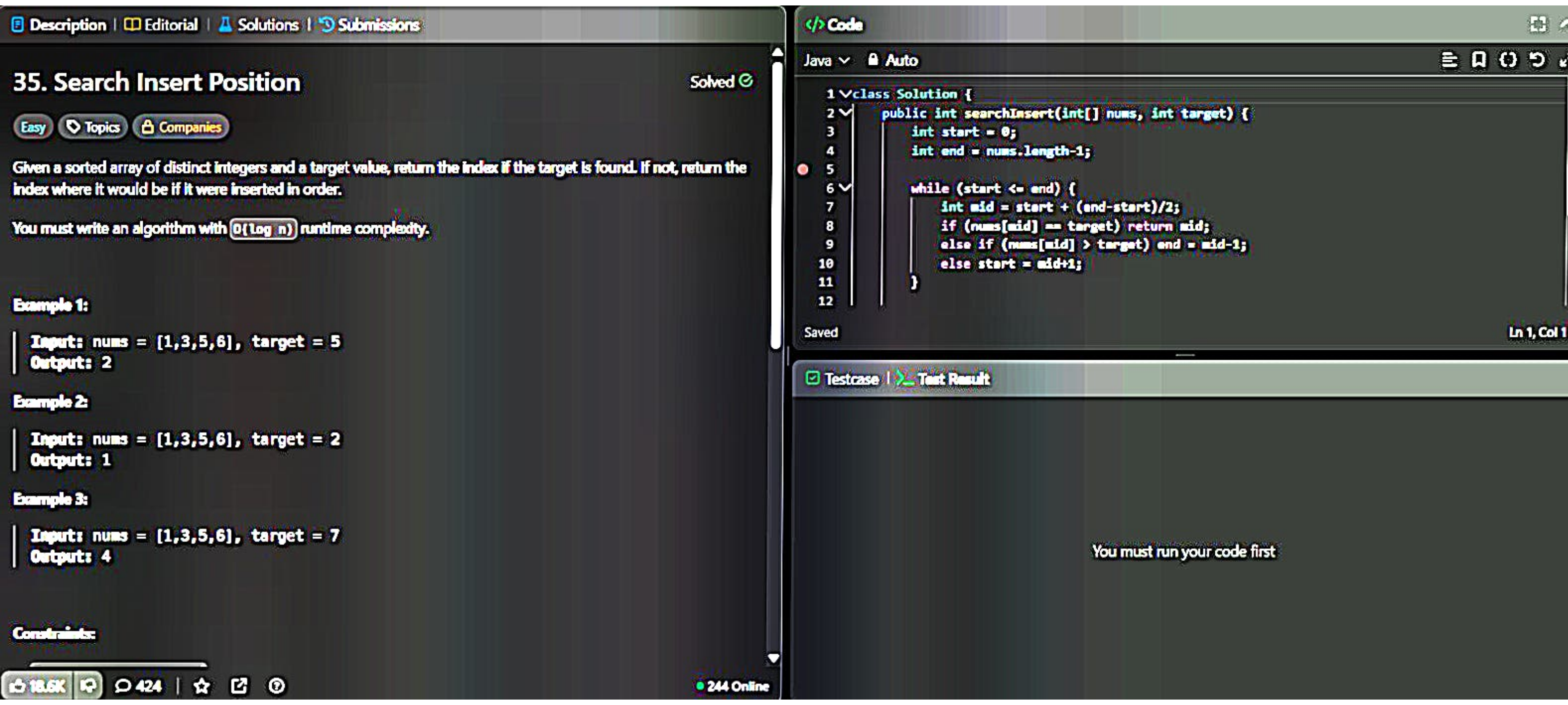
```java
class Solution {
    public boolean searchMatrix(int[][] matrix, int target) {
        int m = matrix.length;
        int n = matrix[0].length;

```

```java
class Solution {
    public boolean searchMatrix(int[][] matrix, int target) {
        int m = matrix.length;
        int n = matrix[0].length;

        int left = 0;
        int right = m * n - 1;

        while (left <= right) {
            int mid = left + (right - left) / 2;
            int row = mid / n;
            int col = mid % n:
```

Saved

Ln 1, Col 1

You must run your code first



35. Search Insert Position

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
```

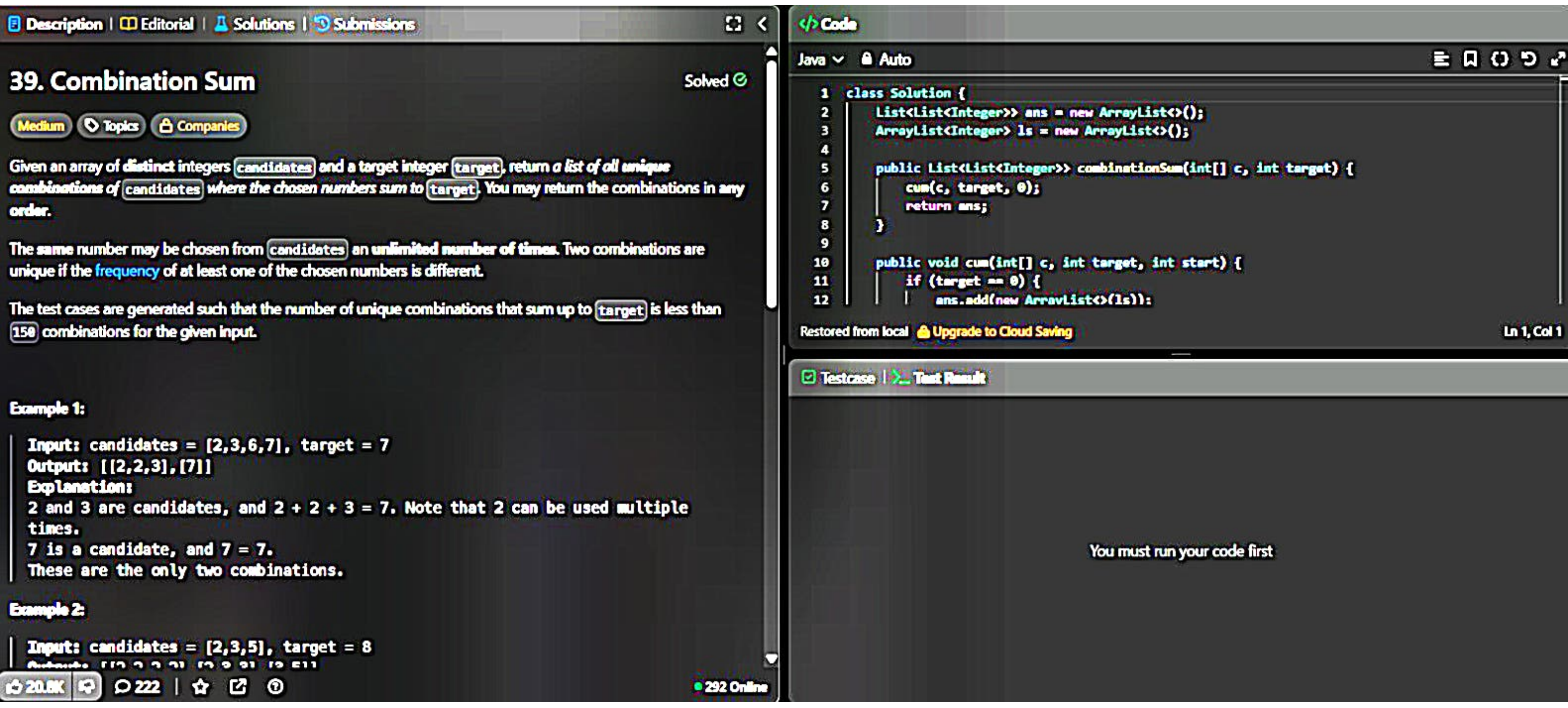
Example 3:

```
Input: nums = [1,3,5,6], target = 7
Output: 4
```

Constraints:

```java
class Solution {
    public int searchInsert(int[] nums, int target) {
        int start = 0;
        int end = nums.length-1;

        while (start <= end) {
            int mid = start + (end-start)/2;
            if (nums[mid] == target) return mid;
            else if (nums[mid] > target) end = mid-1;
            else start = mid+1;
        }

```

You must run your code first



39. Combination Sum

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.

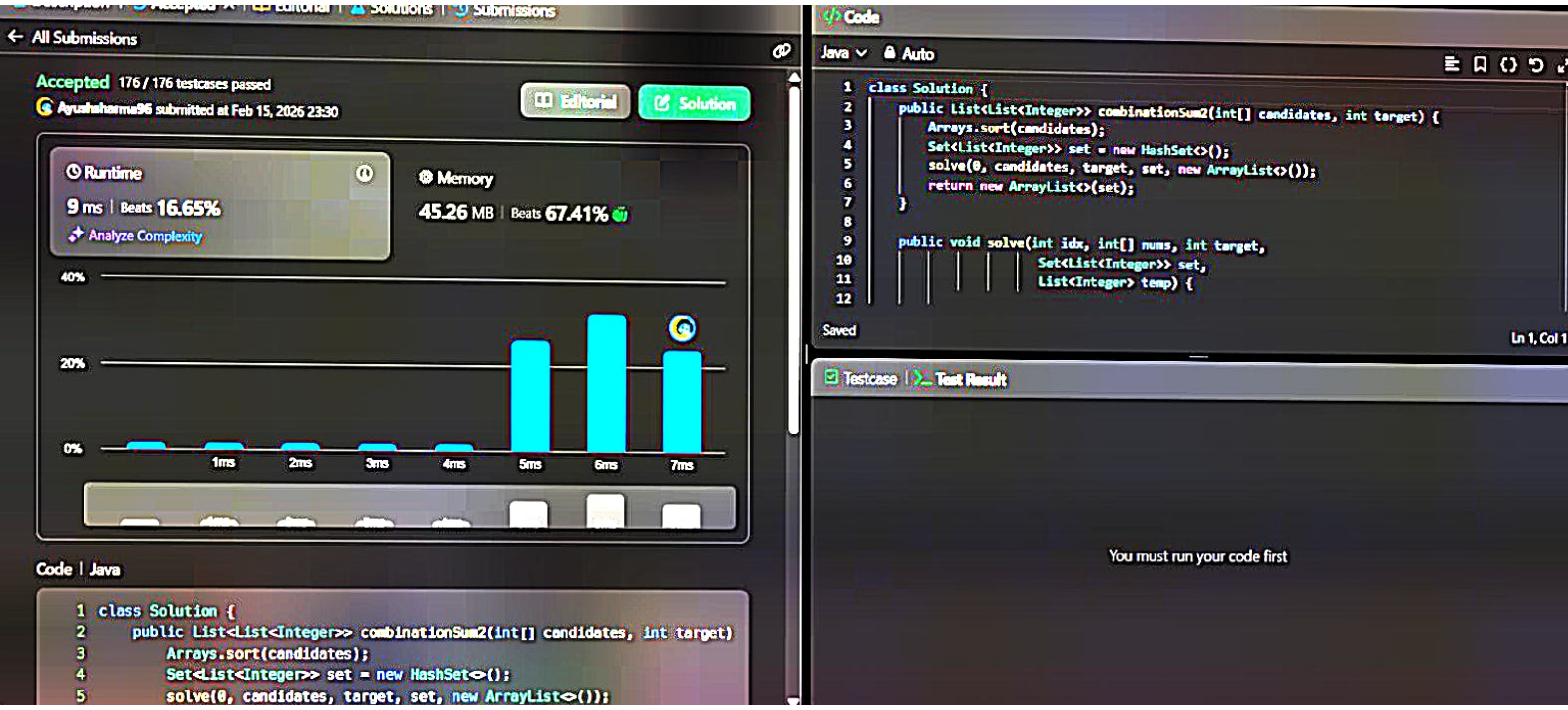
Example 1:

```
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
```

```java
class Solution {
    List<List<Integer>> ans = new ArrayList<>();
    ArrayList<Integer> ls = new ArrayList<>();

    public List<List<Integer>> combinationSum(int[] c, int target) {
        cum(c, target, 0);
        return ans;
    }

    public void cum(int[] c, int target, int start) {
        if (target == 0) {
            ans.add(new ArrayList<>(ls));
```



Accepted 176 / 176 testcases passed

Ayushsharma96 submitted at Feb 15, 2026 23:30

Runtime

9 ms | Beats 16.65%

Memory

45.26 MB | Beats 67.41%

Code | Java

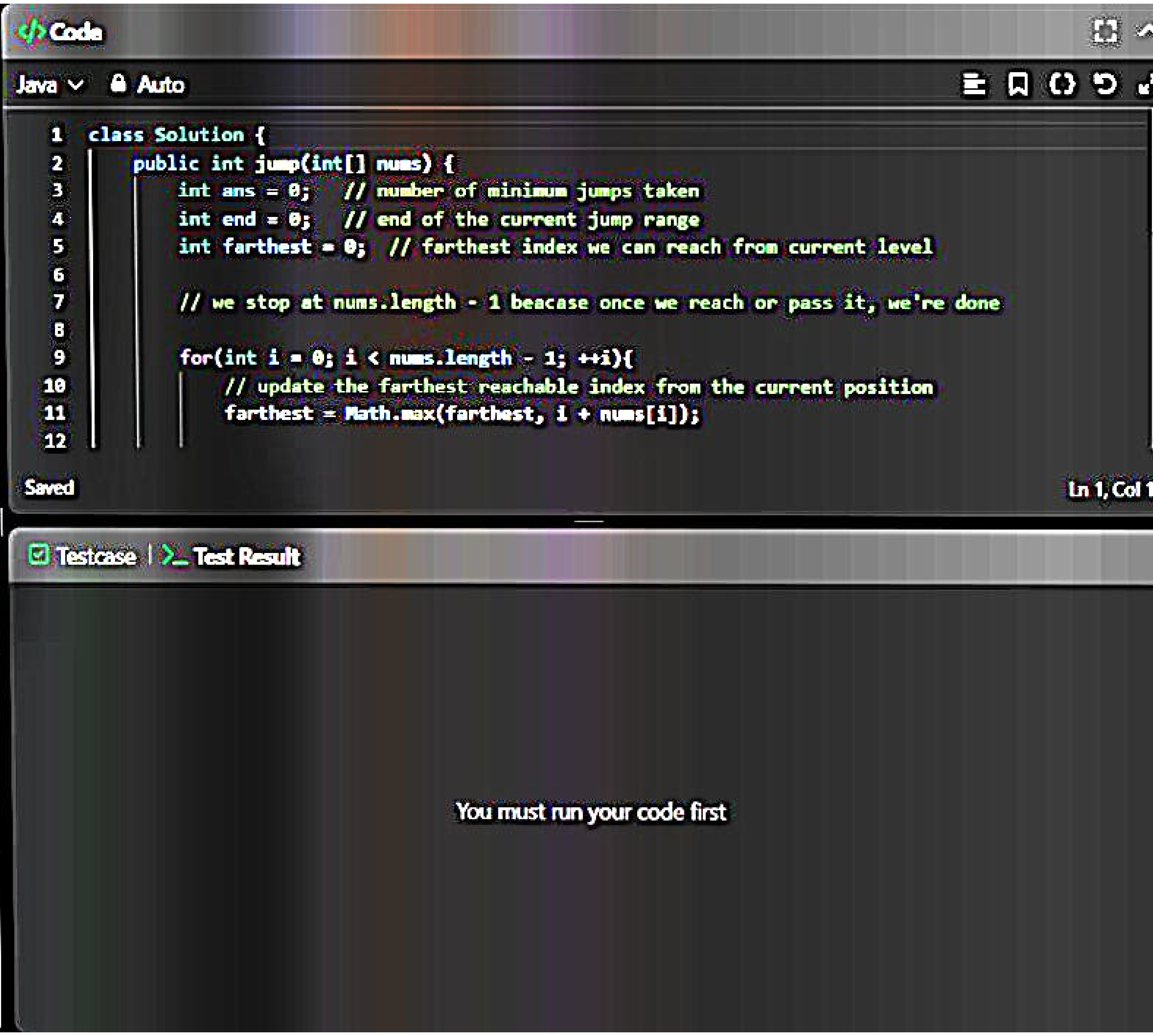
```
class Solution {
    public List<List<Integer>> combinationSum2(int[] candidates, int target)
        Arrays.sort(candidates);
        Set<List<Integer>> set = new HashSet<>();
        solve(0, candidates, target, set, new ArrayList<>());
```

```
class Solution {
    public List<List<Integer>> combinationSum2(int[] candidates, int target) {
        Arrays.sort(candidates);
        Set<List<Integer>> set = new HashSet<>();
        solve(0, candidates, target, set, new ArrayList<>());
        return new ArrayList<>(set);
    }

    public void solve(int idx, int[] nums, int target,
                      Set<List<Integer>> set,
                      List<Integer> temp) {

```

Saved

Ln 1, Col 1

You must run your code first



```java
class Solution {
    public int jump(int[] nums) {
        int ans = 0;   // number of minimum jumps taken
        int end = 0;   // end of the current jump range
        int farthest = 0;  // farthest index we can reach from current level

        // we stop at nums.length - 1 beacase once we reach or pass it, we're done

        for(int i = 0; i < nums.length - 1; ++i){
            // update the farthest reachable index from the current position
            farthest = Math.max(farthest, i + nums[i]);

```

You must run your code first