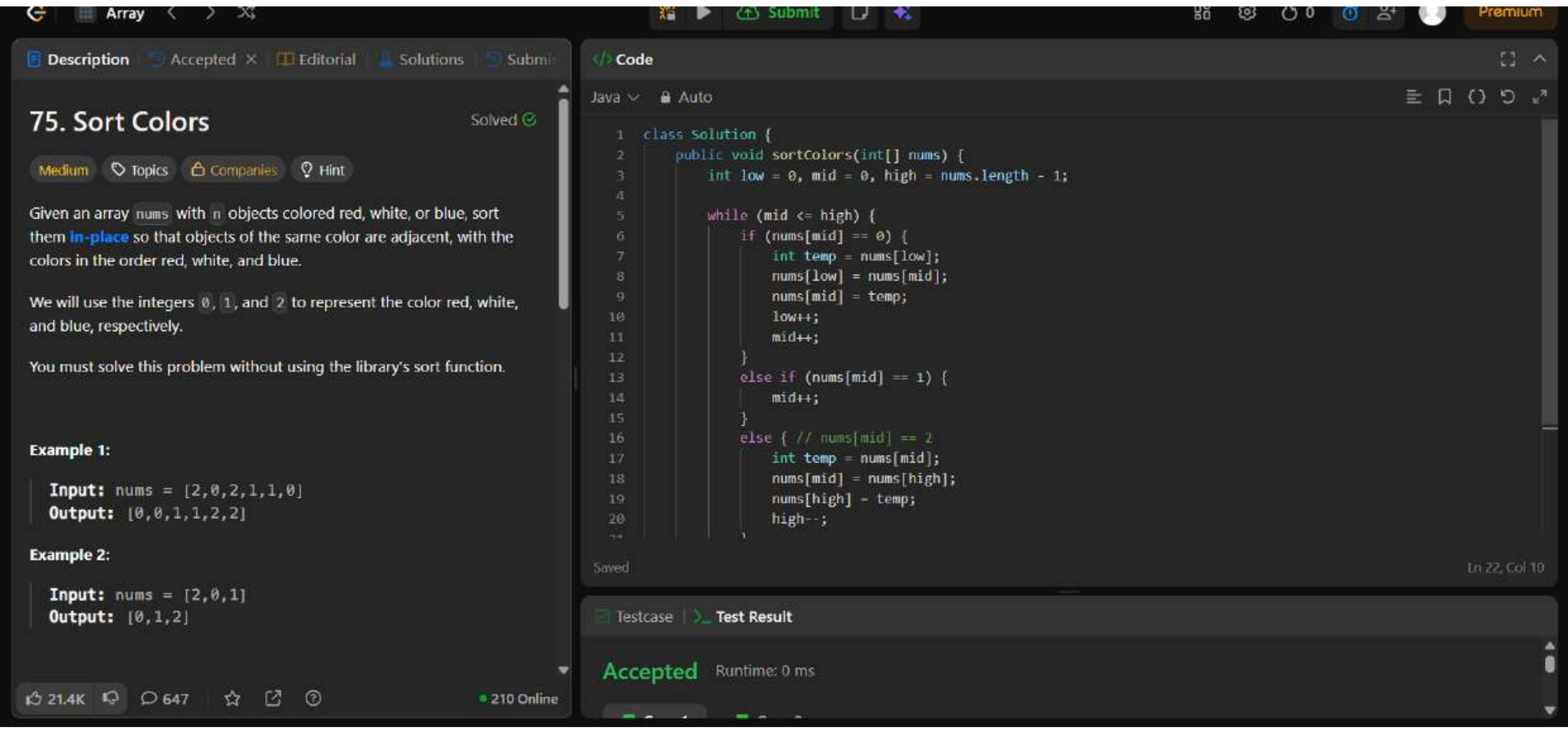


75. Sort Colors

Solved

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:

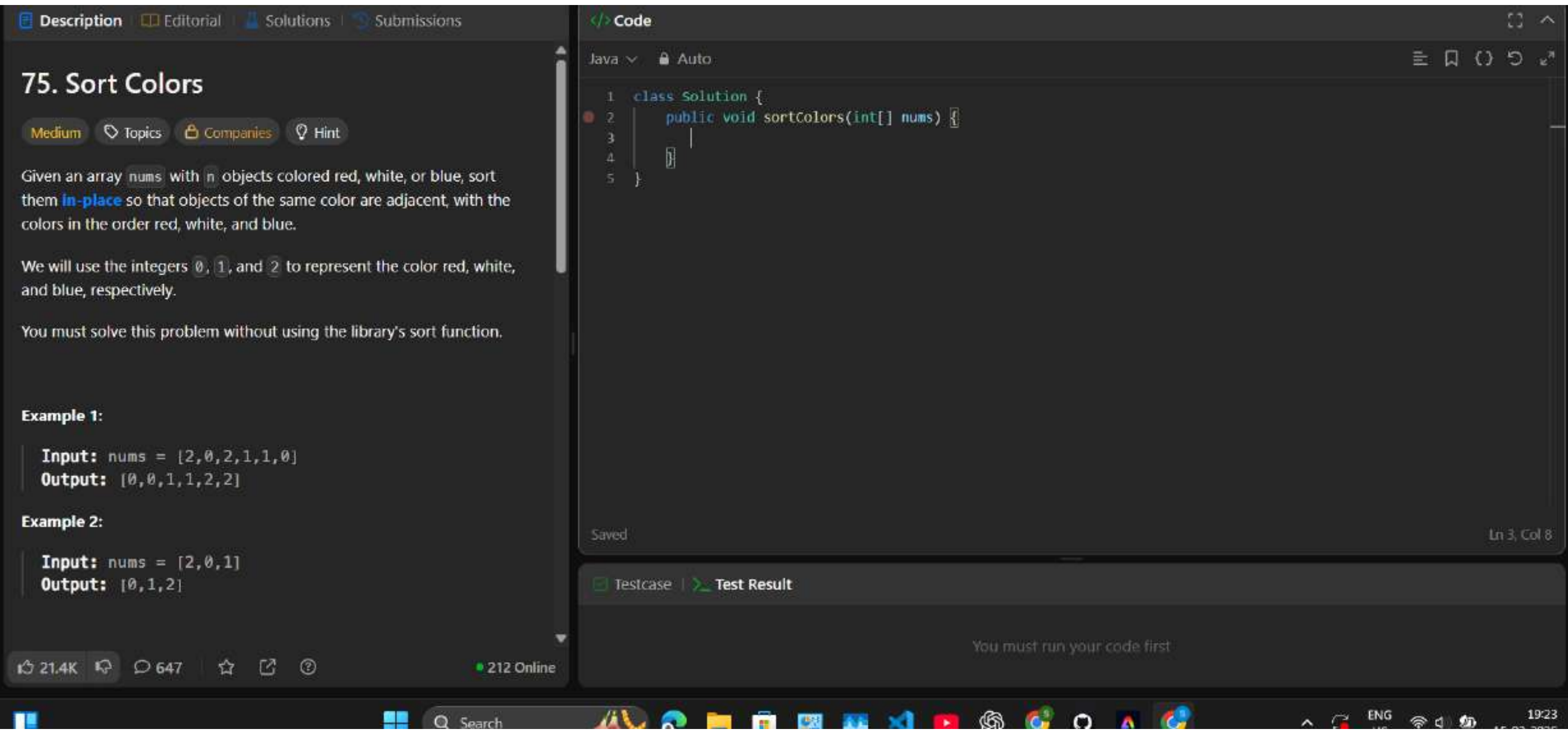
```
Input: nums = [2,0,1]
Output: [0,1,2]
```

```java
class Solution {
    public void sortColors(int[] nums) {
        int low = 0, mid = 0, high = nums.length - 1;

        while (mid <= high) {
            if (nums[mid] == 0) {
                int temp = nums[low];
                nums[low] = nums[mid];
                nums[mid] = temp;
                low++;
                mid++;
            }
            else if (nums[mid] == 1) {
                mid++;
            }
            else { // nums[mid] == 2
                int temp = nums[mid];
                nums[mid] = nums[high];
                nums[high] = temp;
                high--;
```

Saved

Ln 22, Col 10

Accepted Runtime: 0 ms



Description | Editorial | Solutions | Submissions

75. Sort Colors

Medium | Topics | Companies | Hint

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]
```

21.4K | 647 | 212 Online

Code

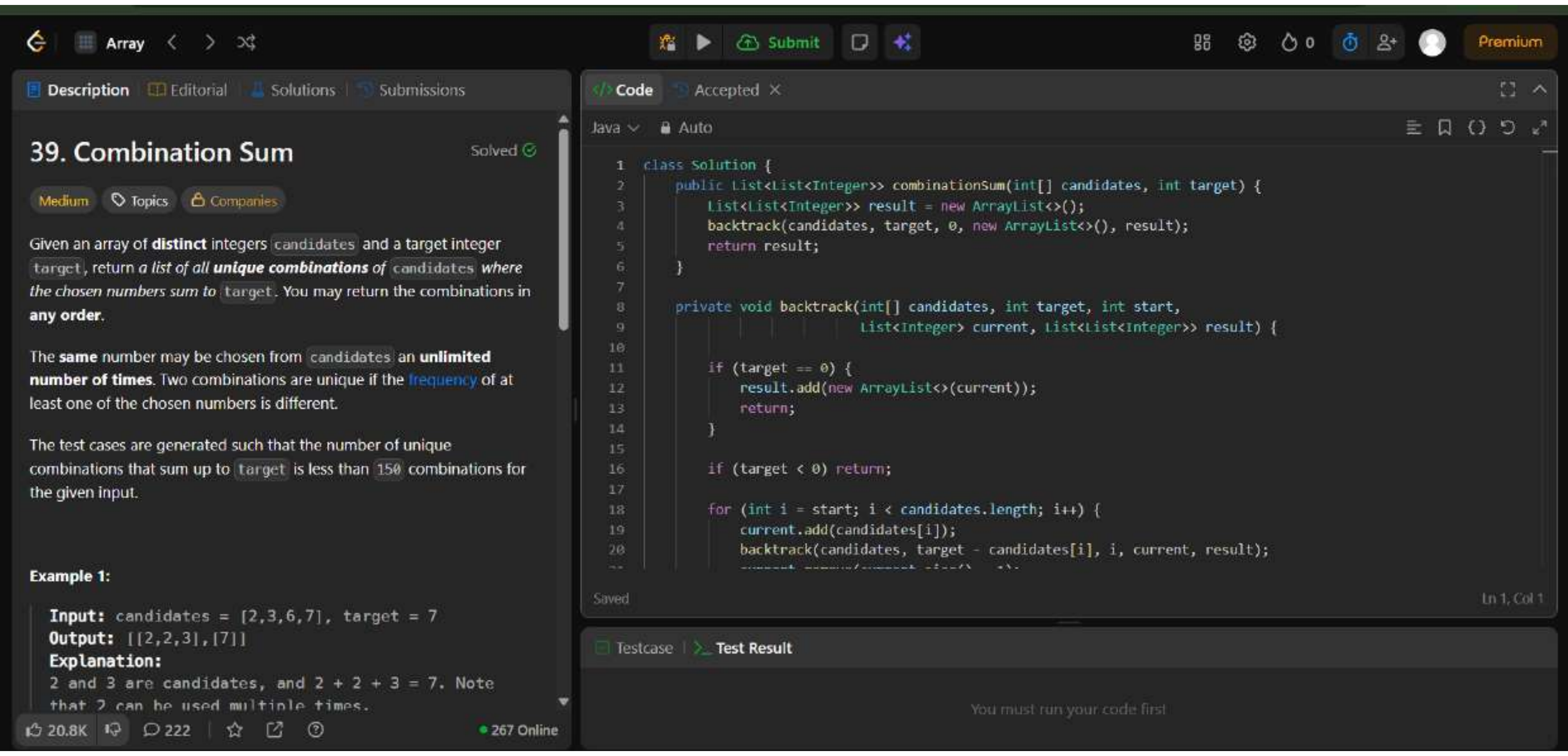
Java | Auto

```java
class Solution {
    public void sortColors(int[] nums) {
        
    }
}
```

Saved | Ln 3, Col 8

Testcase | Test Result

You must run your code first



39. Combination Sum

Solved ✓

Medium · Topics · Companies

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.
```

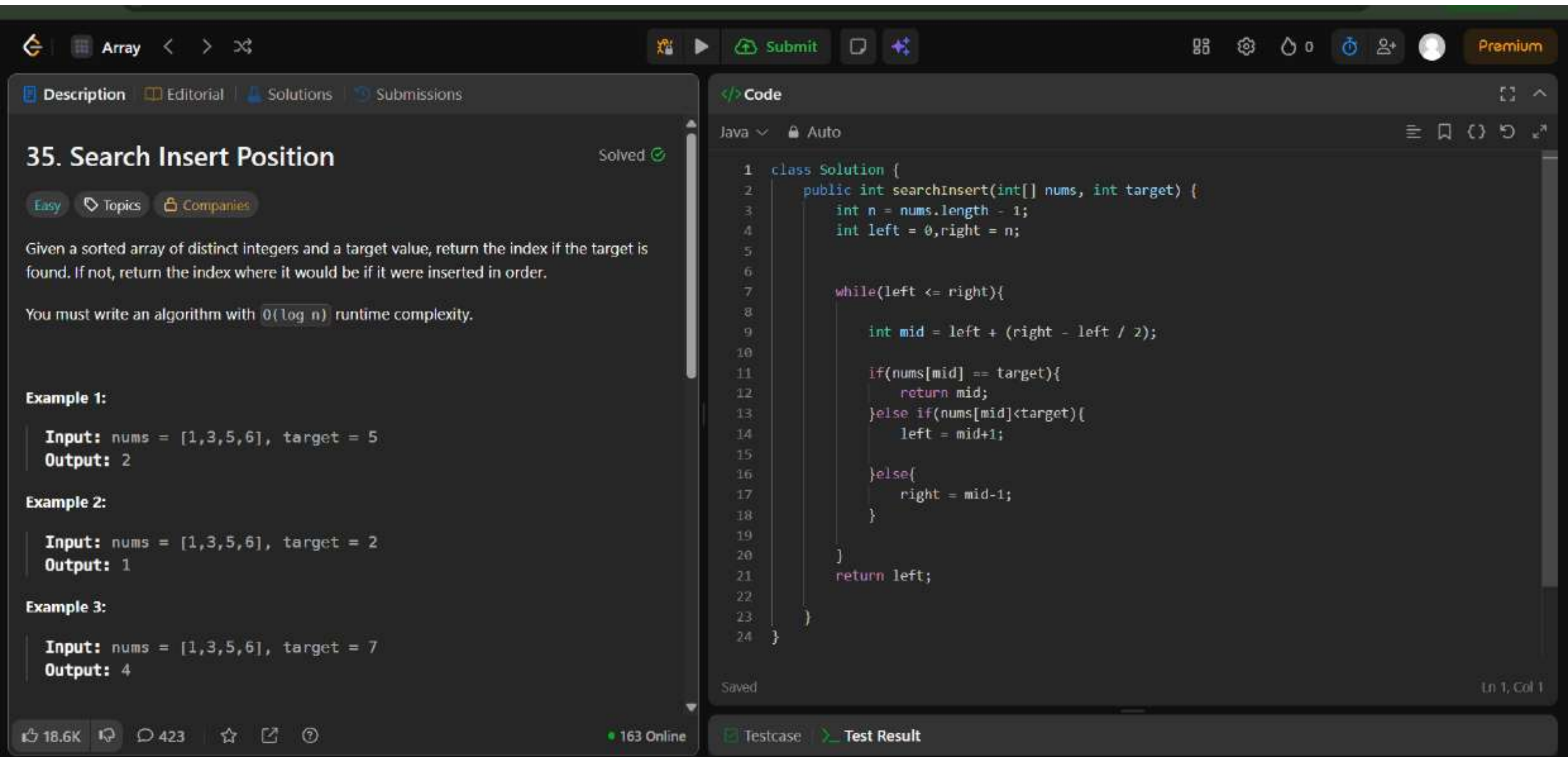
20.8K · 222 · 267 Online

```java
class Solution {
    public List<List<Integer>> combinationSum(int[] candidates, int target) {
        List<List<Integer>> result = new ArrayList<>();
        backtrack(candidates, target, 0, new ArrayList<>(), result);
        return result;
    }

    private void backtrack(int[] candidates, int target, int start,
                           List<Integer> current, List<List<Integer>> result) {

        if (target == 0) {
            result.add(new ArrayList<>(current));
            return;
        }

        if (target < 0) return;

        for (int i = start; i < candidates.length; i++) {
            current.add(candidates[i]);
            backtrack(candidates, target - candidates[i], i, current, result);
```

Saved

Ln 1, Col 1

You must run your code first



35. Search Insert Position

Solved

Easy Topics Companies

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
```

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


        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;

    }
}
```

Saved

Ln 1, Col 1