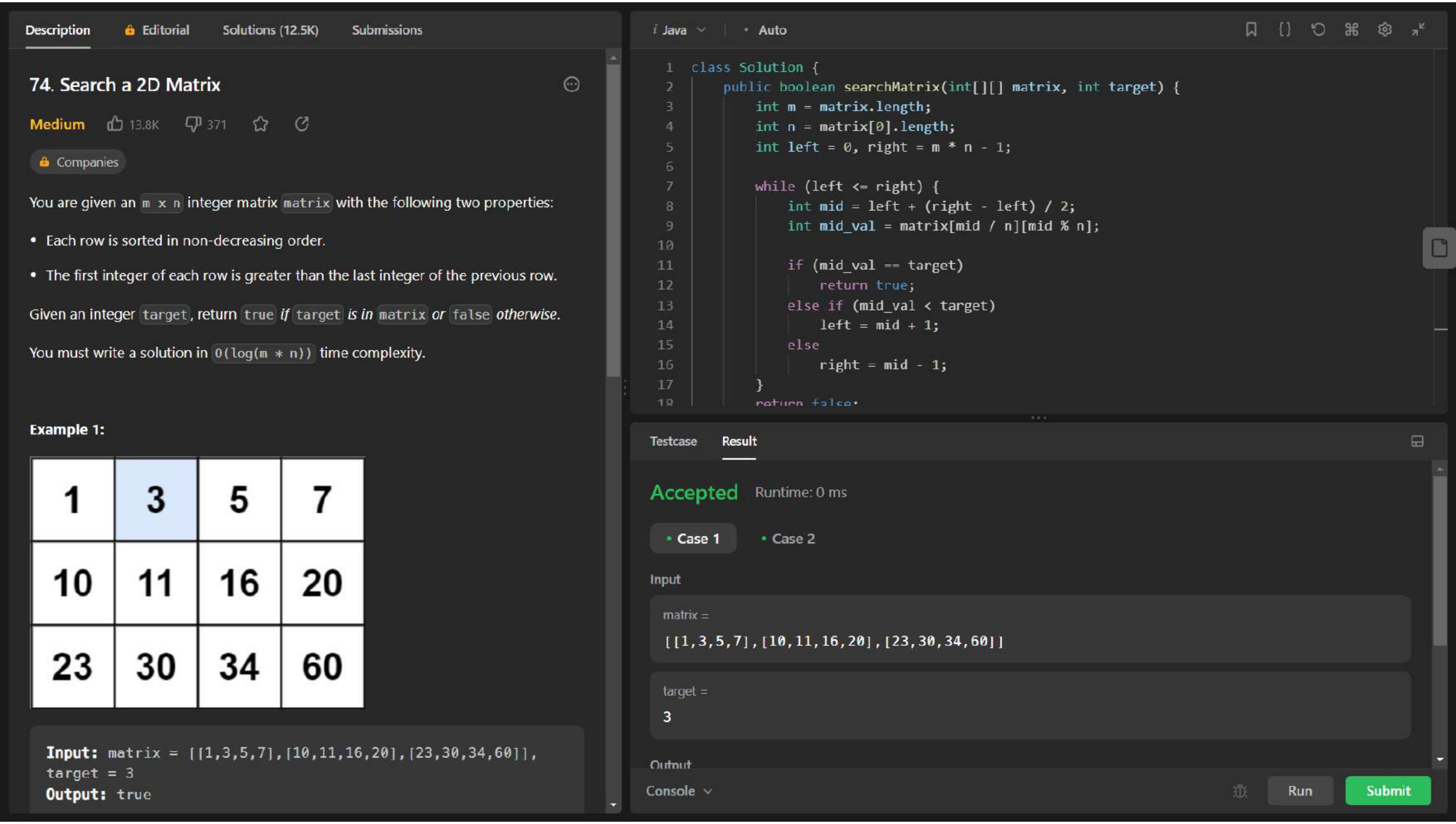


74. Search a 2D Matrix

Medium 13.8K 371

Companies

You are given an `m x n` integer matrix `matrix` with the following two properties:

- Each row is sorted in non-decreasing order.
- The first integer of each row is greater than the last integer of the previous row.

Given an integer `target`, return `true` *if* `target` *is in* `matrix` *or* `false` *otherwise.*

You must write a solution in `O(log(m * n))` time complexity.

Example 1:

1	3	5	7
10	11	16	20
23	30	34	60

```
Input: matrix = [[1,3,5,7],[10,11,16,20],[23,30,34,60]], target = 3
Output: true
```

```java
class Solution {
    public boolean searchMatrix(int[][] matrix, int target) {
        int m = matrix.length;
        int n = matrix[0].length;
        int left = 0, right = m * n - 1;

        while (left <= right) {
            int mid = left + (right - left) / 2;
            int mid_val = matrix[mid / n][mid % n];

            if (mid_val == target)
                return true;
            else if (mid_val < target)
                left = mid + 1;
            else
                right = mid - 1;
        }
        return false;
```

Accepted Runtime: 0 ms

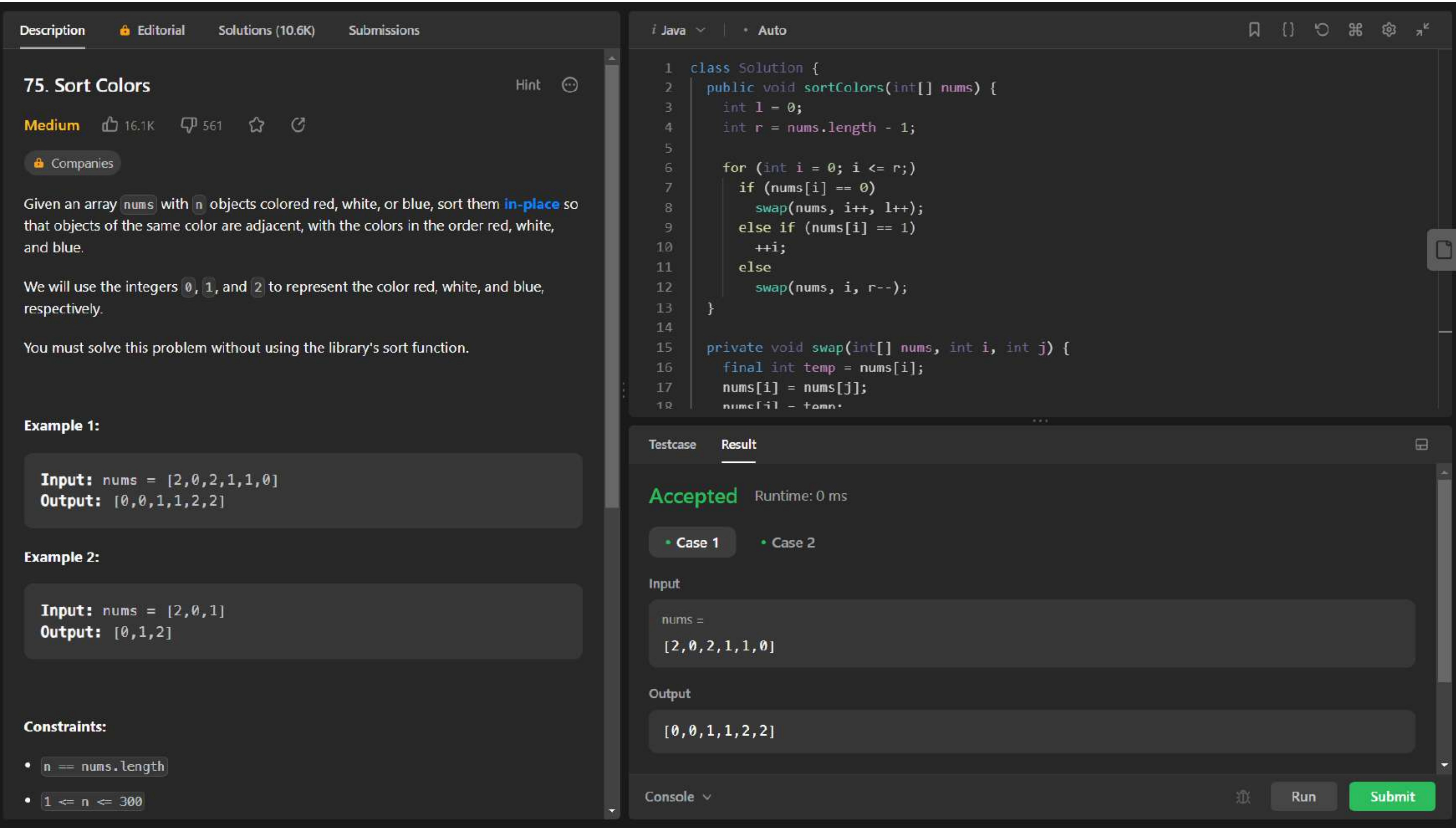
Case 1 Case 2

Input

matrix =
```
[[1,3,5,7],[10,11,16,20],[23,30,34,60]]
```

target =
```
3
```

Output



Description · Editorial · Solutions (10.6K) · Submissions

75. Sort Colors

Medium 16.1K 561

Companies

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]
```

Constraints:

- `n == nums.length`
- `1 <= n <= 300`

Java · Auto

```java
class Solution {
  public void sortColors(int[] nums) {
    int l = 0;
    int r = nums.length - 1;

    for (int i = 0; i <= r;)
      if (nums[i] == 0)
        swap(nums, i++, l++);
      else if (nums[i] == 1)
        ++i;
      else
        swap(nums, i, r--);
  }

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

Testcase · Result

Accepted Runtime: 0 ms

Case 1 · Case 2

Input

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

Output

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