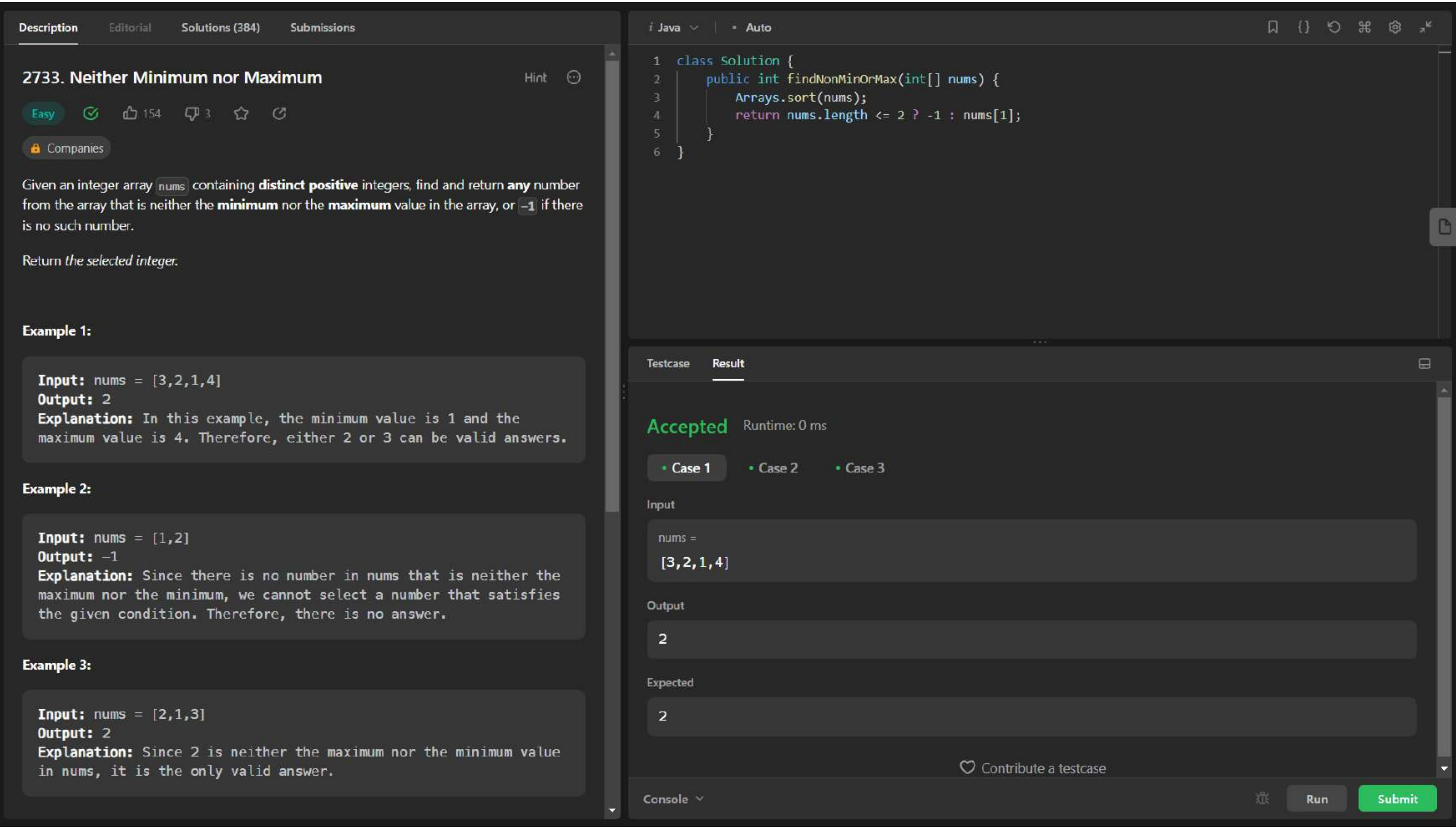


## 2733. Neither Minimum nor Maximum

Given an integer array `nums` containing **distinct positive** integers, find and return **any** number from the array that is neither the **minimum** nor the **maximum** value in the array, or `-1` if there is no such number.

Return *the selected integer.*

**Example 1:**

```
Input: nums = [3,2,1,4]
Output: 2
Explanation: In this example, the minimum value is 1 and the
maximum value is 4. Therefore, either 2 or 3 can be valid answers.
```

**Example 2:**

```
Input: nums = [1,2]
Output: -1
Explanation: Since there is no number in nums that is neither the
maximum nor the minimum, we cannot select a number that satisfies
the given condition. Therefore, there is no answer.
```

**Example 3:**

```
Input: nums = [2,1,3]
Output: 2
Explanation: Since 2 is neither the maximum nor the minimum value
in nums, it is the only valid answer.
```

```java
class Solution {
    public int findNonMinOrMax(int[] nums) {
        Arrays.sort(nums);
        return nums.length <= 2 ? -1 : nums[1];
    }
}
```

Accepted Runtime: 0 ms

Case 1 Case 2 Case 3

Input

nums =
[3,2,1,4]

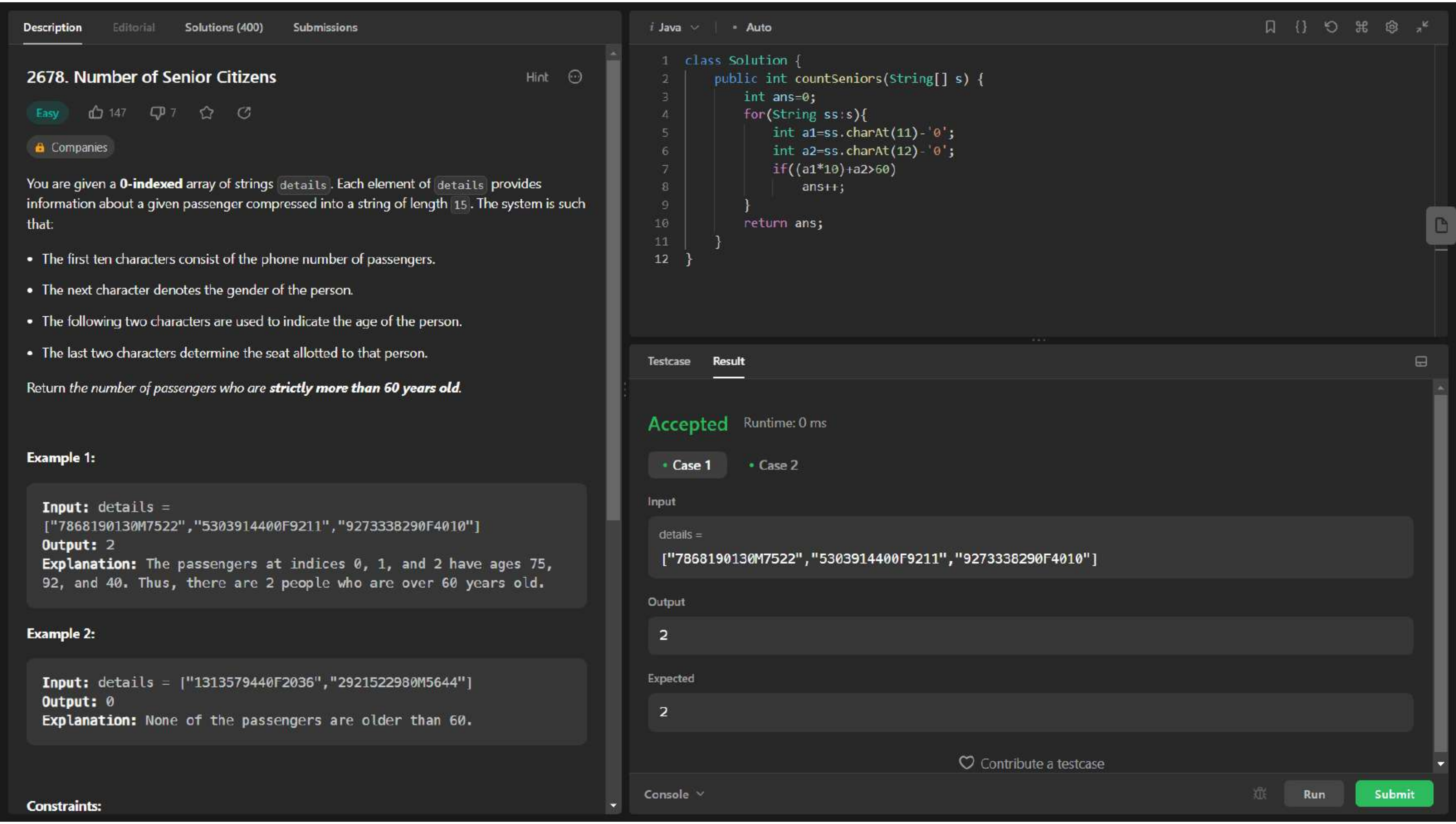
Output

2

Expected

2

Contribute a testcase



## 2678. Number of Senior Citizens

You are given a **0-indexed** array of strings `details`. Each element of `details` provides information about a given passenger compressed into a string of length `15`. The system is such that:

- The first ten characters consist of the phone number of passengers.
- The next character denotes the gender of the person.
- The following two characters are used to indicate the age of the person.
- The last two characters determine the seat allotted to that person.

Return *the number of passengers who are* ***strictly more than 60 years old.***

**Example 1:**

```
Input: details =
["7868190130M7522","5303914400F9211","9273338290F4010"]
Output: 2
Explanation: The passengers at indices 0, 1, and 2 have ages 75,
92, and 40. Thus, there are 2 people who are over 60 years old.
```

**Example 2:**

```
Input: details = ["1313579440F2036","2921522980M5644"]
Output: 0
Explanation: None of the passengers are older than 60.
```

**Constraints:**

```java
class Solution {
    public int countSeniors(String[] s) {
        int ans=0;
        for(String ss:s){
            int a1=ss.charAt(11)-'0';
            int a2=ss.charAt(12)-'0';
            if((a1*10)+a2>60)
                ans++;
        }
        return ans;
    }
}
```

Accepted Runtime: 0 ms

Case 1 Case 2

Input

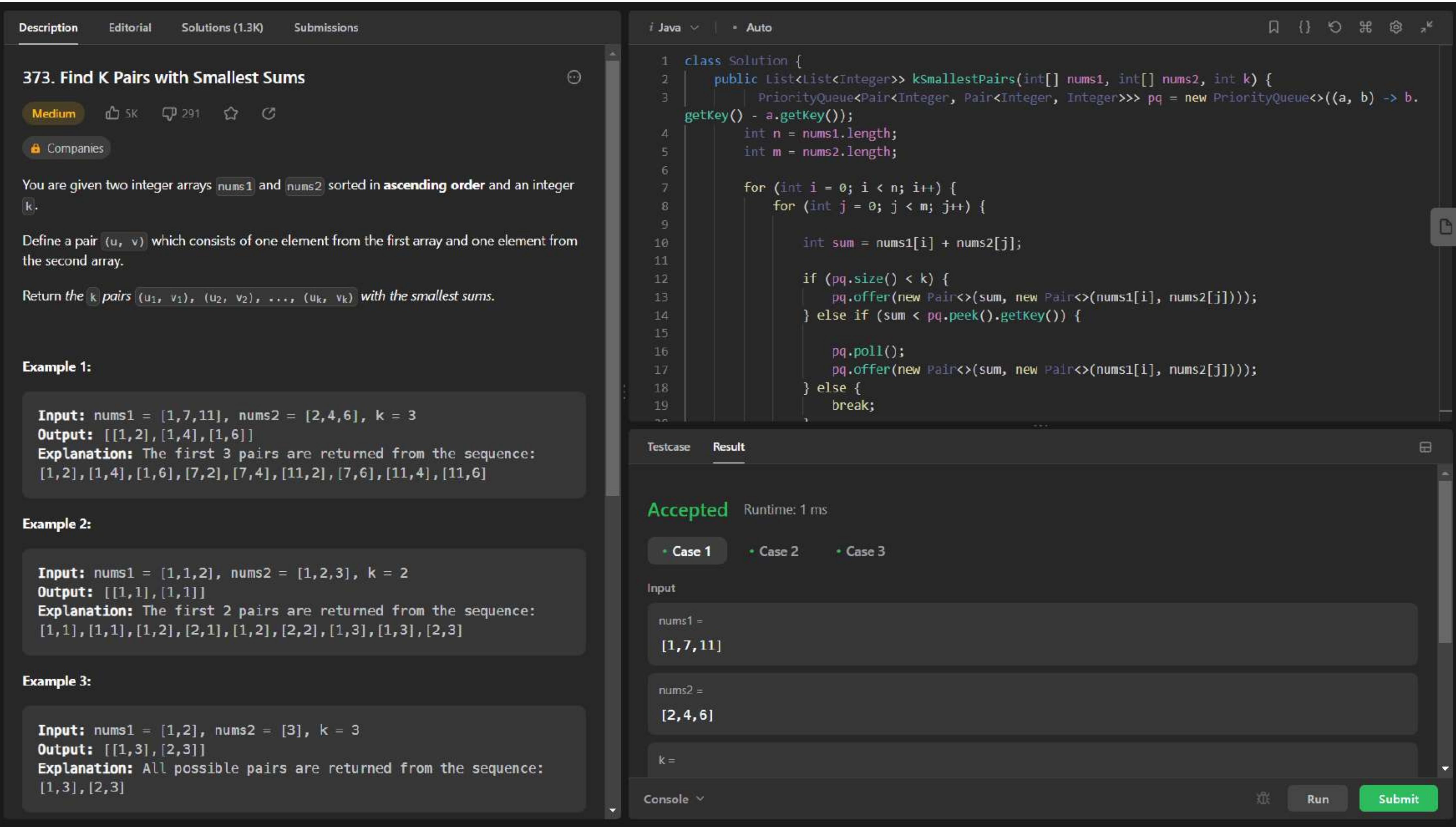
details =
["7868190130M7522","5303914400F9211","9273338290F4010"]

Output

2

Expected

2



## 373. Find K Pairs with Smallest Sums

You are given two integer arrays `nums1` and `nums2` sorted in **ascending order** and an integer `k`.

Define a pair `(u, v)` which consists of one element from the first array and one element from the second array.

Return *the* `k` *pairs* `(u1, v1), (u2, v2), ..., (uk, vk)` *with the smallest sums.*

**Example 1:**

```
Input: nums1 = [1,7,11], nums2 = [2,4,6], k = 3
Output: [[1,2],[1,4],[1,6]]
Explanation: The first 3 pairs are returned from the sequence:
[1,2],[1,4],[1,6],[7,2],[7,4],[11,2],[7,6],[11,4],[11,6]
```

**Example 2:**

```
Input: nums1 = [1,1,2], nums2 = [1,2,3], k = 2
Output: [[1,1],[1,1]]
Explanation: The first 2 pairs are returned from the sequence:
[1,1],[1,1],[1,2],[2,1],[1,2],[2,2],[1,3],[1,3],[2,3]
```

**Example 3:**

```
Input: nums1 = [1,2], nums2 = [3], k = 3
Output: [[1,3],[2,3]]
Explanation: All possible pairs are returned from the sequence:
[1,3],[2,3]
```

```java
class Solution {
    public List<List<Integer>> kSmallestPairs(int[] nums1, int[] nums2, int k) {
        PriorityQueue<Pair<Integer, Pair<Integer, Integer>>> pq = new PriorityQueue<>((a, b) -> b.
getKey() - a.getKey());
        int n = nums1.length;
        int m = nums2.length;

        for (int i = 0; i < n; i++) {
            for (int j = 0; j < m; j++) {

                int sum = nums1[i] + nums2[j];

                if (pq.size() < k) {
                    pq.offer(new Pair<>(sum, new Pair<>(nums1[i], nums2[j])));
                } else if (sum < pq.peek().getKey()) {

                    pq.poll();
                    pq.offer(new Pair<>(sum, new Pair<>(nums1[i], nums2[j])));
                } else {
                    break;
```

Accepted Runtime: 1 ms

Input

nums1 =
[1,7,11]

nums2 =
[2,4,6]

k =