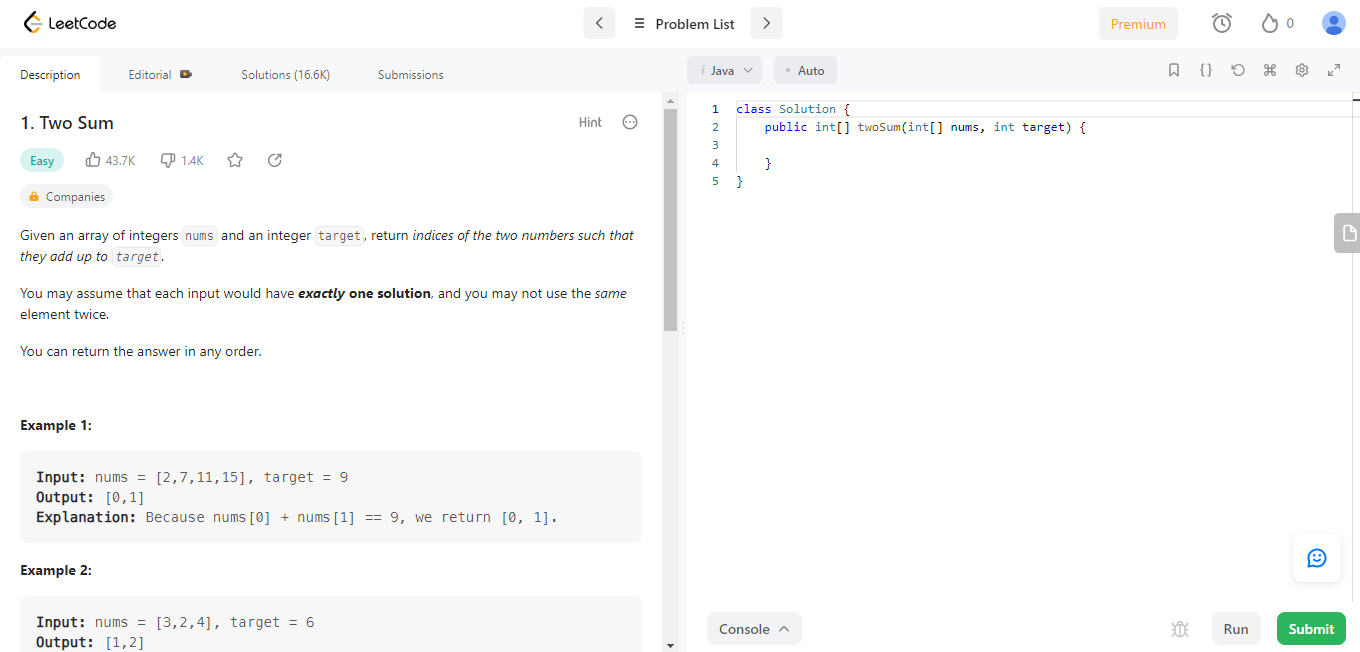
**TWO SUM (EASY)**

****

* **Approaches to solve problem**

1. **First over the brute force approach.**

* We check array if equal null Or length < 2

Return null.

* We will perform a nested ‘for’ loop

For each of element ‘x’ in array.

* Initialize variable called ‘diff’ equal target number minus element of array ‘x’.
* We will want to perform another ‘for’ loop to find ‘x + 1’, check if ‘x + 1’ equal ‘diff’.
* If true return array [x, x + 1].
* If false return null.

**Analysis**

**time complexity:** O(n^2)

1. **Optimized Approach**

* We can further optimize the above solution approach by using a set.
* The algorithm for this optimal approach is as follows:

1. Initialize an empty HashSet.
2. For each integer in the array:
   1. Calculate the difference between the current integer and the targetSum.
   2. Check whether the difference calculated above is present in the set.
      1. If the difference already exists in the set, return the current element and difference as the result.
      2. Otherwise, insert the current integer into the set.

**Analysis**

**The time complexity :** O(n)

1. **Use Sorting along with the two-pointer approach**

There is another approach which works when you need to return the numbers instead of their indexes. Here is how it works:

1. Sort the array.
2. Initialize two variables, one pointing to the beginning of the array (left) and another pointing to the end of the array (right).
3. Loop until left < right, and for each iteration.

3.1- if arr[left] + arr[right] == target, then return the indices.

3.2- if arr[left] + arr[right] < target, increment the left index.

3.3- else, decrement the right index.

This approach is called the two-pointer approach. It is a very common pattern for solving array related problems.

**The time complexity :** O(nlogn)