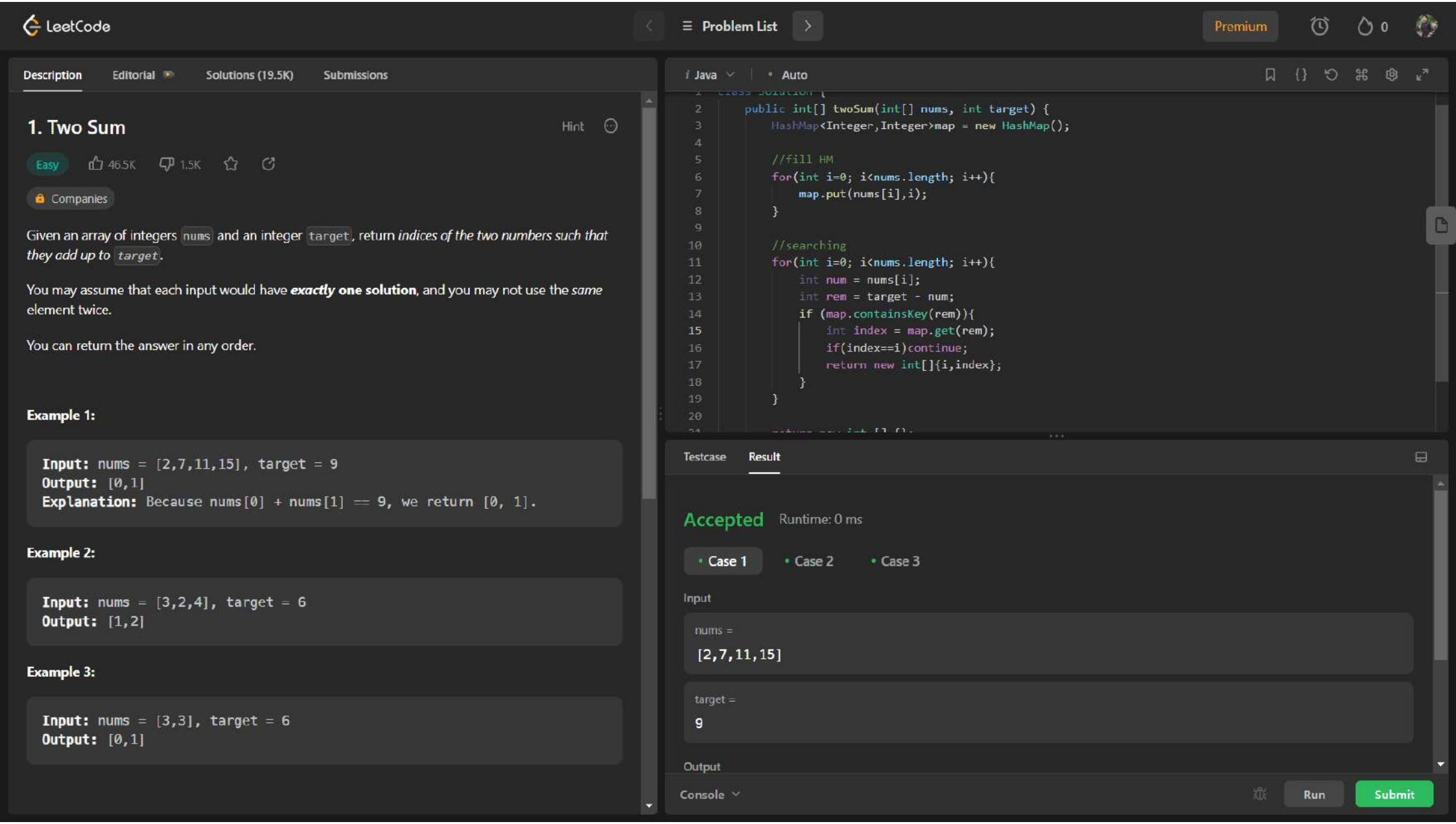

LeetCode
Problem List
Premium
Description
Editorial
Solutions (19.5K)
Submissions
1. Two Sum
Hint
Easy
46.5K
1.5K
Companies
Given an array of integers nums and an integer target, return indices of the two numbers such that they add up to target.
You may assume that each input would have exactly one solution, and you may not use the same element twice.
You can return the answer in any order.
Example 1:
Input: nums = [2,7,11,15], target = 9
Output: [0,1]
Explanation: Because nums[0] + nums[1] == 9, we return [0, 1].
Example 2:
Input: nums = [3,2,4], target = 6
Output: [1,2]
Example 3:
Input: nums = [3,3], target = 6
Output: [0,1]
Java
Auto
public int[] twoSum(int[] nums, int target) {
HashMap<Integer,Integer>map = new HashMap();
//fill HM
for(int i=0; i<nums.length; i++){
map.put(nums[i],i);
}
//searching
for(int i=0; i<nums.length; i++){
int num = nums[i];
int rem = target - num;
if (map.containsKey(rem)){
int index = map.get(rem);
if(index==i)continue;
return new int[]{i,index};
}
}
Testcase
Result
Accepted
Runtime: 0 ms
Case 1
Case 2
Case 3
Input
nums =
[2,7,11,15]
target =
9
Output
Console
Run
Submit

Description | Editorial | Solutions (13.1K) | Submissions

2. Add Two Numbers

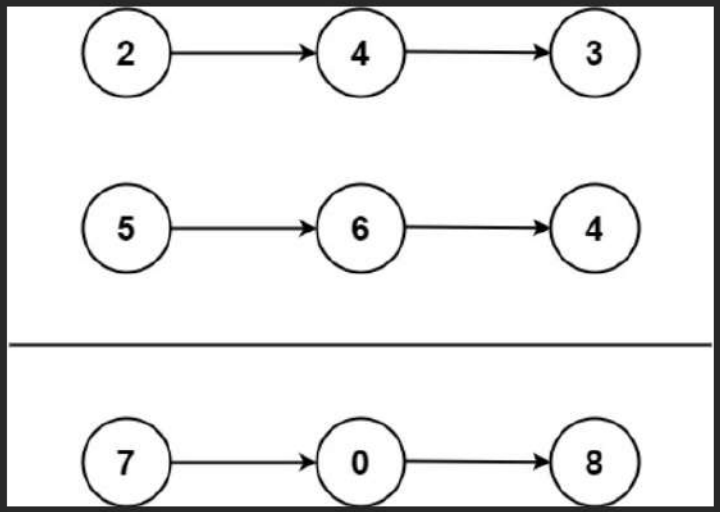
Medium | 26K | 5K

Companies

You are given two **non-empty** linked lists representing two non-negative integers. The digits are stored in **reverse order**, and each of their nodes contains a single digit. Add the two numbers and return the sum as a linked list.

You may assume the two numbers do not contain any leading zero, except the number 0 itself.

Example 1:




```
Input: l1 = [2,4,3], l2 = [5,6,4]
Output: [7,0,8]
Explanation: 342 + 465 = 807.
```

Example 2:

Java | Auto

```java
public ListNode addTwoNumbers(ListNode l1, ListNode l2) {
ListNode result = new ListNode(0);
ListNode ptr = result;

int carry = 0;    // Set default carry

while (l1 != null || l2 != null) {

  int sum = 0 + carry;    // Initialize sum

  if (l1 != null) {    // Use number from first list
    sum += l1.val;
    l1 = l1.next;
  }

  if (l2 != null) {    // Use number from 2nd list
    sum += l2.val;
    l2 = l2.next;
  }

  carry = sum / 10;    // Get sum and carry
  sum = sum % 10;
  ptr.next = new ListNode(sum);
  ptr = ptr.next;
```

Testcase | Result

Accepted Runtime: 0 ms

• Case 1 • Case 2 • Case 3

Input

l1 =
[2,4,3]

l2 =
[5,6,4]

Console | Run | Submit

Description | Editorial | Solutions (15.1K) | Submissions

3. Longest Substring Without Repeating Characters

Medium | 34K | 1.5K

Companies

Given a string `s`, find the length of the **longest substring** without repeating characters.

Example 1:

```
Input: s = "abcabcbb"
Output: 3
Explanation: The answer is "abc", with the length of 3.
```

Example 2:

```
Input: s = "bbbbb"
Output: 1
Explanation: The answer is "b", with the length of 1.
```

Example 3:

```
Input: s = "pwwkew"
Output: 3
Explanation: The answer is "wke", with the length of 3.
Notice that the answer must be a substring, "pwke" is a subsequence and not a substring.
```

Constraints:

- `0 <= s.length <= 5 * 10^4`
- `s` consists of English letters, digits, symbols and spaces.

Java | Auto

```java
class Solution {
    public int lengthOfLongestSubstring(String s) {
        int left =0 , right = 0;
        Set<Character> seen = new HashSet();
        int max = 0;

        while(right<s.length()){
            char c = s.charAt(right);
            if(seen.add(c)){
                max = Math.max(max,right-left+1);
                right++;
            }else{
                while(s.charAt(left)!=c){
                    seen.remove(s.charAt(left));
                    left++;
                }
                seen.remove(c);left++;
            }
        }
        return max;
    }
}
```

Testcase | Result

Accepted Runtime: 0 ms

Case 1 | Case 2 | Case 3

Input

s =
"abcabcbb"

Output

3

Console | Run | Submit