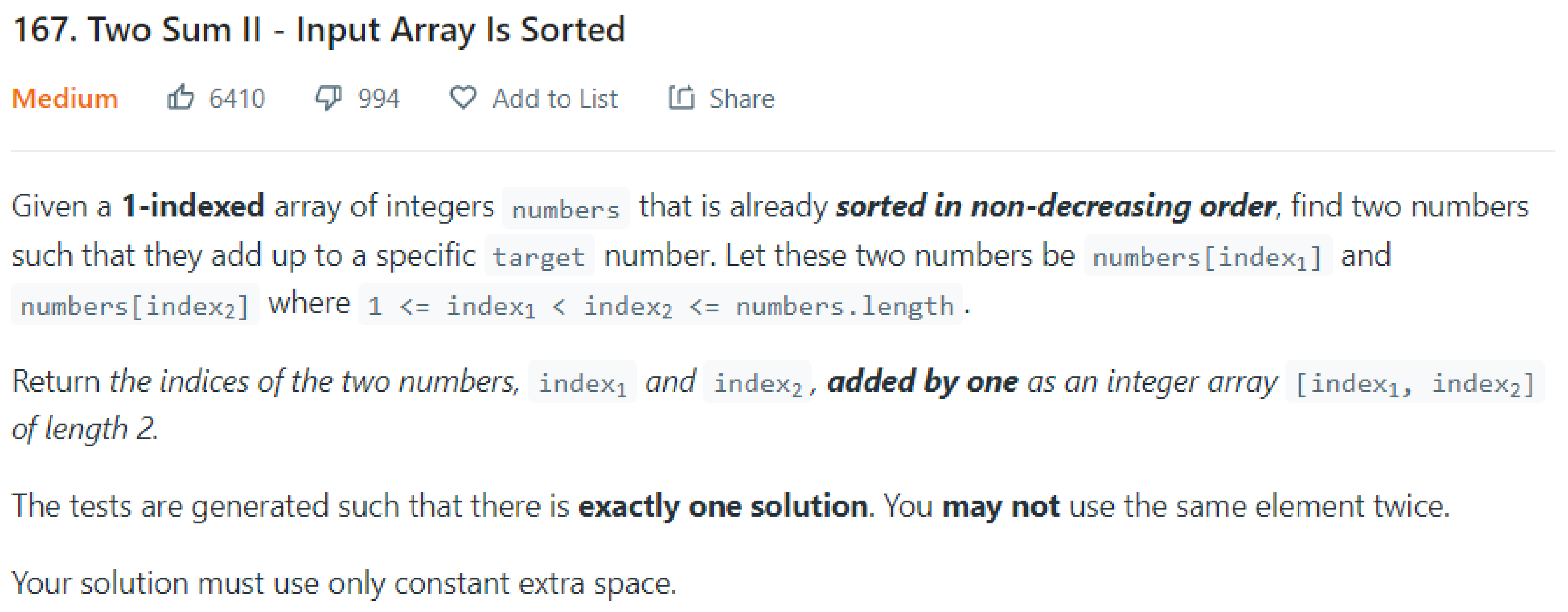


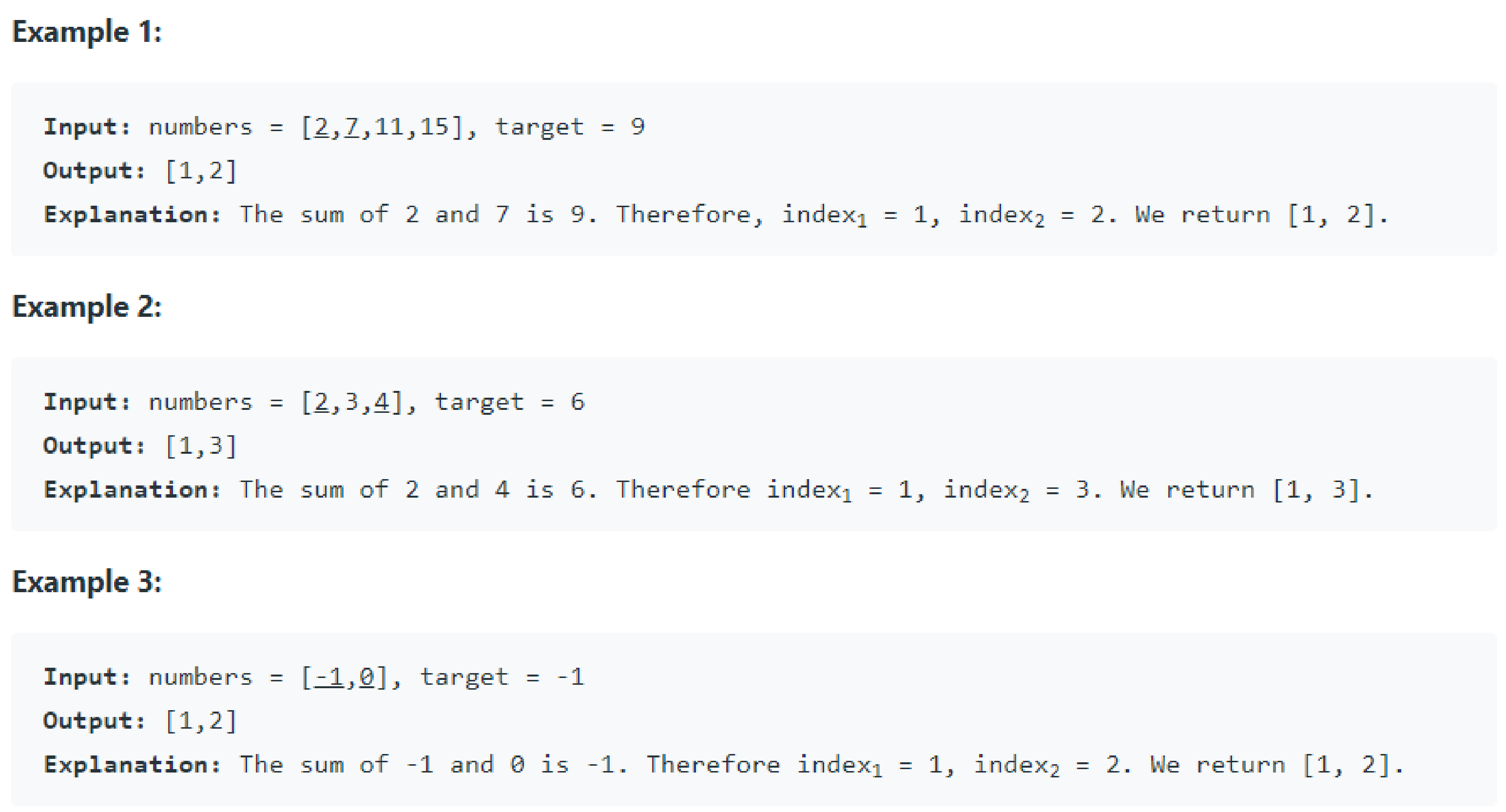
167. Two Sum II - Input Array Is Sorted

Given a **1-indexed** array of integers `numbers` that is already ***sorted in non-decreasing order***, find two numbers such that they add up to a specific `target` number. Let these two numbers be $\texttt{numbers[index}_1\texttt{]}$ and $\texttt{numbers[index}_2\texttt{]}$ where $\texttt{1 <= index}_1\texttt{ < index}_2\texttt{ <= numbers.length}$.

Return *the indices of the two numbers,* $\texttt{index}_1$ *and* $\texttt{index}_2$*,* ***added by one*** *as an integer array* $\texttt{[index}_1\texttt{, index}_2\texttt{]}$ *of length 2.*

The tests are generated such that there is **exactly one solution**. You **may not** use the same element twice.

Your solution must use only constant extra space.



Example 1:

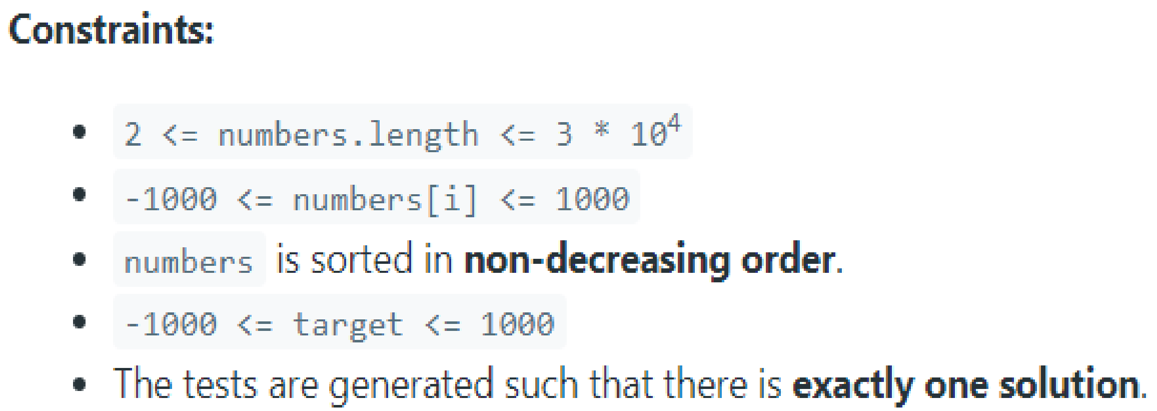
Input: numbers = [2,7,11,15], target = 9
Output: [1,2]
Explanation: The sum of 2 and 7 is 9. Therefore, $index_1$ = 1, $index_2$ = 2. We return [1, 2].

Example 2:

Input: numbers = [2,3,4], target = 6
Output: [1,3]
Explanation: The sum of 2 and 4 is 6. Therefore $index_1$ = 1, $index_2$ = 3. We return [1, 3].

Example 3:

Input: numbers = [-1,0], target = -1
Output: [1,2]
Explanation: The sum of -1 and 0 is -1. Therefore $index_1$ = 1, $index_2$ = 2. We return [1, 2].



Constraints:

- `2 <= numbers.length <= 3 *` 10^4
- `-1000 <= numbers[i] <= 1000`
- `numbers` is sorted in **non-decreasing order**.
- `-1000 <= target <= 1000`
- The tests are generated such that there is **exactly one solution**.

```
vector<int> twoSum(vector<int>& num, int target) {
    int n = num.size(), l = 0, r = n-1;
    vector<int> res(2, 0);
    while(l<r) {
        if((num[l] + num[r]) < target) {
            l++;
        } else if ((num[l] + num[r]) > target) {
            r--;
        } else {
            res[0] = l+1;
            res[1] = r+1;
            break;
        }
    }
    return res;
}
```

#100daysofDSA

/rvislive

Rakesh Vishwakarma