167. Two Sum II - Input Array Is Sorted

Medium 5760 960 Add to List Share

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 `numbers[index1]` and `numbers[index2]` where `1 <= index1 < index2 <= numbers.length`.

Return *the indices of the two numbers,* `index1` *and* `index2`*,* ***added by one*** *as an integer array* `[index1, index2]` *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, index1 = 1, index2 = 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 index1 = 1, index2 = 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 index1 = 1, index2 = 2. We return [1, 2].
```

Constraints:

- `2 <= numbers.length <= 3 * 10^4`
- `-1000 <= numbers[i] <= 1000`
- `numbers` is sorted in **non-decreasing order**.

```java
class Solution {
    public int[] twoSum(int[] numbers, int target) {

        int firstPointer = 0;
        int secondPointer = numbers.length - 1;

        while(firstPointer < secondPointer){

            int sum = numbers[firstPointer] + numbers[secondPointer];

            if(sum == target){
                return new int[]{firstPointer + 1, secondPointer + 1};
            }else if(sum < target){
                firstPointer++;
            }else if(sum > target){
                secondPointer--;
            }
        }


        return null;
    }
}
```

Testcase | Run Code Result | Debugger

Accepted Runtime: 0 ms

Your input
```
[2,7,11,15]
9
```

Output `[1,2]` Diff

Expected `[1,2]`