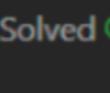


Two Sum

167. Two Sum II - Input Array Is Sorted

Solved 

Medium Topics Companies

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 $1 \leq index1 < index2 \leq numbers.length$.

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]`

return any 2 indices, which results the sum of both indices equals to the given target

2 pointers technique

If you are required to find elements that are satisfying certain constraints or conditions \rightarrow you use 2 pointers technique

Most commonly used problems:

- ① Two Sum
- ② Three Sum, four sum, closest three sum...
- ③ Container with most water
- ④ Sort n Searching
- ⑤ String manipulations
- ⑥ Palindrome problems
- ⑦ LinkedList problems
- ⑧ Sliding window problems

So, as our array is sorted, we can make observations out of it

arr: $(2, 7, 11, 15)$

If you set pointer $\rightarrow i$ at 0 , j at $n-1 \Rightarrow (2, 7, 11, 15)$

\uparrow \uparrow
i *j*

* if you sum up $arr[i] + arr[j]$ \rightarrow you will get some 'x' ans right?

let us assume, $(\text{Sum we got} == X)$

Case - I

If $(\text{sum} < \text{target}) \downarrow$

it means our sum need to be increased to match target value

for that, you have to increment i , as your array is sorted, if you move i forward, the value definitely increase

Case - II

If $(\text{sum} > \text{target}) \downarrow$

it means our sum need to be decreased to match target value

for that, you have to decrement j , as your array is sorted, if you move j backward, the value definitely decrease

Case - III

If $(\text{sum} == \text{target})$

return the index values of (i, j)

<u>i</u>	<u>j</u>	<u>Sum</u>	<u>target</u>	<u>Operation</u>
$\{2, 7, 11, 18\}$ ↑ ↑	0 3	17	< 18	$i++$
$\{2, 7, 11, 18\}$ ↑ ↑	1 3	21	> 18	$j--$
$\{2, 7, 11, 18\}$ ↑ ↑	1 2	18	$= 18$	✓ (return $\{1, 2\}$)