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 `numbers[index<sub>1</sub>]` and `numbers[index<sub>2</sub>]` where `1 <= first < second <= numbers.length`.

Return *the indices of the two numbers,* `index<sub>1</sub>` *and* `index<sub>2</sub>`*, as an integer array* `[index<sub>1</sub>, index<sub>2</sub>]` *of length 2.*

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

Example 1:

```
Input: numbers = [2,7,11,15], target = 9
Output: [1,2]
Explanation: The sum of 2 and 7 is 9. Therefore index<sub>1</sub> = 1, index<sub>2</sub> = 2.
```

Example 2:

```
Input: numbers = [2,3,4], target = 6
Output: [1,3]
Explanation: The sum of 2 and 4 is 6. Therefore index<sub>1</sub> = 1, index<sub>2</sub> = 3.
```

Example 3:

```
Input: numbers = [-1,0], target = -1
Output: [1,2]
Explanation: The sum of -1 and 0 is -1. Therefore index<sub>1</sub> = 1, index<sub>2</sub> = 2.
```

Constraints:

- `2 <= numbers.length <= 3 * 10<sup>4</sup>`
- `-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**.

```python
class Solution:
    def twoSum(self, numbers: List[int], target: int) -> List[int]:
        i = 0
        j = len(numbers)-1

        while i<j:
            temp = numbers[i]+numbers[j]
            if temp==target:
                return (i+1,j+1)
            elif temp>target:
                j = j-1
            else:
                i = i+1
```