정렬된 배열을 받아 덧셈하여 타겟을 만들 수 있는 배열의 두 숫자 인덱스를 리턴하라.
#주의: 이 문제에서 배열은 0이 아닌 1부터 시작하는 것으로 한다.

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.

1.투 포인터

```
class Solution:
    def twoSum(self, numbers: List[int], target: int) -> List[int]:
        left, right = 0, len(numbers) - 1
        while not left == right:
            temp = numbers[left] + numbers[right]
            if temp < target:
                left += 1
            elif temp > target:
                right -= 1
            else:
                return [left + 1, right + 1]
```

```
import bisect

class Solution:
    def twoSum(self, numbers: List[int], target: int) -> List[int]:
        for k, v in enumerate(numbers):
            left, right = k + 1, len(numbers) - 1
            expected = target - v
            i = bisect.bisect_left(numbers[k + 1:], expected)
            if i < len(numbers[k + 1:]) and numbers[k + i + 1] == expected:
                return k + 1, k + i + 2
```

```
import bisect

class Solution:
    def twoSum(self, numbers: List[int], target: int) -> List[int]:
        for k, v in enumerate(numbers):
            left, right = k + 1, len(numbers) - 1
            expected = target - v
            nums = numbers[k + 1:]
            i = bisect.bisect_left(nums, expected)
            if i < len(nums) and numbers[k + i + 1] == expected:
                return k + 1, k + i + 2
```

```
import bisect

class Solution:
    def twoSum(self, numbers: List[int], target: int) -> List[int]:
        for k, v in enumerate(numbers):
            left, right = k + 1, len(numbers) - 1
            expected = target - v
```

```
            i = bisect.bisect_left(numbers, expected, k + 1)
            if i < len(numbers) and numbers[i] == expected:
                return k + 1, i + 1
```

2. 이진 검색

```
class Solution:
    def twoSum(self, numbers: List[int], target: int) -> List[int]:
        for k, v in enumerate(numbers):
            left, right = k + 1, len(numbers) - 1
            expected = target - v
            while left <= right:
                mid = left + (right - left) // 2
                if numbers[mid] < expected:
                    left = mid + 1
                elif numbers[mid] > expected:
                    right = mid - 1
                else:
                    return k + 1, mid + 1
```