题目

给你一个下标从 1 开始的整数数组 numbers， 该数组已按 非递减顺序排列， 请你从数组中找出满足相加之和等于目标数 target 的两个数。如果设这两个数分别是 numbers[index1] 和 numbers[index2]， 则 1 <= index1 < index2 <= numbers.length 。

以长度为 2 的整数数组 [index1, index2] 的形式返回这两个整数的下标 index1 和 index2。

你可以假设每个输入 只对应唯一的答案， 而且你 不可以 重复使用相同的元素。

你所设计的解决方案必须只使用常量级的额外空间。

举例：

```
示例 1：

输入：numbers = [2,7,11,15], target = 9
输出：[1,2]
解释：2 与 7 之和等于目标数 9 。因此 index1 = 1, index2 = 2 。返回 [1, 2] 。
示例 2：

输入：numbers = [2,3,4], target = 6
输出：[1,3]
解释：2 与 4 之和等于目标数 6 。因此 index1 = 1, index2 = 3 。返回 [1, 3] 。
示例 3：

输入：numbers = [-1,0], target = -1
输出：[1,2]
解释：-1 与 0 之和等于目标数 -1 。因此 index1 = 1, index2 = 2 。返回 [1, 2] 。
```

提示：

2 <= numbers.length <= 3 * 104
-1000 <= numbers[i] <= 1000
numbers 按 非递减顺序 排列
-1000 <= target <= 1000
仅存在一个有效答案

代码

C++:

```cpp
class Solution {
public:
    vector<int> twoSum(vector<int> &numbers, int target) {
        int left = 0, right = numbers.size() - 1;
        while (true) {
            int s = numbers[left] + numbers[right];
            if (s == target) return {left + 1, right + 1}; // 题目要求下标从 1 开始
            s > target ? --right : ++left;
        }
    }
};
```