**难度: 简单**

给定一个已按照升序排列 的有序数组，找到两个数使得它们相加之和等于目标数。

函数应该返回这两个下标值 index1 和 index2，其中 index1 必须小于 index2。

说明:

返回的下标值（index1 和 index2）不是从零开始的。

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。

示例:

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 。

来源：力扣（LeetCode）

链接：https://leetcode-cn.com/problems/two-sum-ii-input-array-is-sorted

著作权归领扣网络所有。商业转载请联系官方授权，非商业转载请注明出处。

**思路1(暴力解法):**

1). 每次第一个和后面所有的进行比较,然后第二个和最后的所有比较

2). 直到找到数组下标,或者找不到返回-1,-1

|  |
| --- |
| public int[] twoSum(int[] numbers, int target) {  for (int i = 0; i < numbers.length; i++) {  for (int j = i+1; j < numbers.length; j++) {  if (numbers[i] + numbers[j] == target) {  return new int[]{i+1 , j+1};  }  }  }  return new int[]{-1,-1};  } |

**思路2(双指针法):**

1). 定义左右指针

2). 如果两个数相加大于targer的话,那么就右指针-1

3). 如果两个数相加小于targer的话,那么就左指针+1

|  |
| --- |
| public int[] twoSum2(int[] numbers, int target) {  int left = 0;  int right =numbers.length -1;  while(left < right) {  if (numbers[left]+numbers[right] < target) {  left++;  }else if (numbers[left]+numbers[right] > target) {  right--;  }else {  return new int[]{left+1 , right+1};  }  }  return new int[]{-1,-1};  } |