题目描述：

Given an array of integers that is already ***sorted in ascending order***, find two numbers such that they add up to a specific target number.

The function twoSum should return indices of the two numbers such that they add up to the target, where index1 must be less than index2. Please note that your returned answers (both index1 and index2) are not zero-based.

You may assume that each input would have *exactly* one solution and you may not use the *same* element twice.

Input: numbers={2, 7, 11, 15}, target=9

Output: index1=1, index2=2

自己的解题思路：

1.循环嵌套 时间负责度O(n平方) 结果：时间溢出

2.用target-numers[i]，然后在数组里用折半查找寻找这个相减的差，时间复杂度是O(nlog2n) 结果：时间溢出

3.最终看了题解，主要思想还是**空间换时间**，把数组每个元素与target的差值都存到map里，使用hashmap的get来提升查找速度，时间复杂度是O(n) 结果：accepted

题解代码：

```
class Solution {
    public int[] twoSum(int[] numbers, int target) {
        int[] result = new int[2];
        Map<Integer,Integer> map = new HashMap<>();
        for(int i = 0;i < numbers.length;i++){
            map.put(target-numbers[i],i);
        }
        for(int i = 0;i < numbers.length;i++){
            Integer otherIndex = map.get(numbers[i]);
            if(otherIndex != null && otherIndex != i){
                result[0] = i + 1;
                result[1] = otherIndex + 1;
                if(result[0]>result[1]){
                    int tmp = result[1];
                    result[1] = result[0];
                    result[0] = tmp;
                }
```

```
            }
        }
        return result;
    }
}
```