题目：给定一个整数数组 nums 和一个目标值 target，请你在该数组中找出和为目标值的那 两个 整数，并返回他们的数组下标。

（你可以假设每种输入只会对应一个答案。但是，数组中同一个元素不能使用两遍。）

示例:

给定 nums = [2, 7, 11, 15], target = 9

因为 nums[0] + nums[1] = 2 + 7 = 9

所以返回 [0, 1]

方法一：暴力法

```
public int[] twoSum(int[] nums, int target) {
 int[] arr = new int[2];
 for (int i = 0; i < nums.length-1; i++) {
 for (int j = i+1; j<nums.length; j++) {
 int sum = 0;
 sum = nums[i] + nums[j];
 if (sum == target) {
 arr[0]=i;
 arr[1]=j;
  return arr;
  }
  }
  }
  return null;
}
```

方法二：两遍哈希表

```
public int[] twoSum(int[] nums, int target) {
 HashMap<Integer, Integer> map = new HashMap<>();
 int[] arr = new int[2];
 for (int i = 0; i < nums.length; i++) {
```

```
map.put(nums[i], i);
}
for (int i = 0; i < nums.length; i++) {
Integer complete = target - nums[i];
if (map.containsKey(complete) && map.get(complete) != i) {
 arr[0] = map.get(complete);
 arr[1] = i;
 Arrays.sort(arr);
 return arr;
 }
 }
 return arr;
}
```

在第一次迭代中，我们将每个元素的值和它的索引添加到表中。然后，在第二次迭代中，我们将检查每个元素所对应的目标元素（*target - nums[i]*）是否存在于表中。注意，该目标元素不能是 *nums[i]* 本身！

方法三：一遍哈希表

```
public int[] twoSum(int[] nums, int target) {
 HashMap<Integer, Integer> map = new HashMap<>();
 int[] arr = new int[2];
 for (int i = 0; i < nums.length; i++) {
 Integer complete = target - nums[i];
 if (map.containsKey(complete) && map.get(complete) != i) {
 arr[0]=map.get(complete);
 arr[1]=i;
 Arrays.sort(arr);
 return arr;
 }
 map.put(nums[i], i);
 }
 return null;
}
```

在进行迭代并将元素插入到表中的同时，我们还会回过头来检查表中是否已经存在当前元素所对应的目标元素。如果它存在，那我们已经找到了对应解，并立即将其返回。