**问题：**

Given an array of integers, return **indices** of the two numbers such that they add up to a specific target.

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

**Example:**

Given nums = [2, 7, 11, 15], target = 9,

Because nums[**0**] + nums[**1**] = 2 + 7 = 9,

return [**0**, **1**].

**问题描述：**

A+B=C,给出C，给出含有A和B的数组，找出满足A+B=C的A和B输出相应数组下标。

**自己的想法**：

A+B=C，知道C通过循环找A，再通过嵌套循环找B，如果满足A+B=C，输出A,B的下标。嵌套循环时间复杂度为O（n\*n）

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

**答案：**

A+B=C,知道C，确定A遍历，B=C-A。找B，存在B,输出A,B以及下标

寻找B，若遍历数组时间复杂度为O（n）用hashmap找，时间复杂度为O（1）

Map<A,A的下标>这种方式存储相应的数组信息，查找Key值比Value值更快捷。

流程：1.建立map<>

2.循环，计算对应A值下的B值：B=C-A

3.用map.containsKey（B）判断map里是否含有对应的B值

若存在，再次查找B的下标，map的value为数值的下标用map.get（B）取值，返回A&B以及相应的下标：A=nums[i],A下标=i，B=C-A，B下标=map.get（B）。

若不存在，将A值以及下标存入Map。

时间复杂度：O(n\*1)

代码：

**public** **int[]** **twoSum(int[]** nums**,** **int** target**)** **{**

Map**<**Integer**,** Integer**>** map **=** **new** HashMap**<>();**

**for** **(int** i **=** 0**;** i **<** nums**.**length**;** i**++)** **{**

**int** complement **=** target **-** nums**[**i**];**

**if** **(**map**.**containsKey**(**complement**))** **{**

**return** **new** **int[]** **{** map**.**get**(**complement**),** i **};**

**}**

map**.**put**(**nums**[**i**],** i**);**

**}**

**throw** **new** IllegalArgumentException**(**"No two sum solution"**);**

**}**