# 1、两数之和（Two Sum）

## 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]

## 思路：

* 两遍循环

static void Main(string[] args)  
{  
 int[] nums = {2, 5, 11, 7};  
 int target = 9;  
 int[] result = TwoSum(nums, target);  
 //int[] result1 = TwoSum1(nums, target);  
 foreach (int i in result)  
 {  
 Console.WriteLine(i);  
 }  
  
 Console.ReadKey();  
}  
//法一  
public static int[] TwoSum(int[] nums, int target)  
{  
 int[] result = new int[] { };  
 for (int i = 0; i < nums.Length; i++)  
 {  
 for (int j = i + 1; j < nums.Length; j++)  
 {  
 if (nums[i] + nums[j] == target)  
 {  
 result = new[] { i, j };  
 }  
 }  
 }  
 return result;  
}

* 利用字典
* //法二   
  public static int[] TwoSum1(int[] nums, int target)   
  {   
   //用数组中的值做key，索引做value存下所有值   
   Dictionary<int, int> map = new Dictionary<int, int>();   
   for (int i = 0; i < nums.Length; i++)   
   {   
   int num = nums[i];   
   //ContainsKey是否包含特定键   
   //若字典中已经存在这个差值，说明匹配成功   
   if (map.ContainsKey(target - num))   
   {   
   //map[Key,Value] 序号 值   
   //map[key]这和数组是一样的   
   return new int[] { map[target - num], i };   
   }   
   else   
   {   
   //不等于把索引给字典，放在value中   
   //体会把数字放在key中的原因   
   map[num] = i;   
   }   
   }   
   return null;   
  }