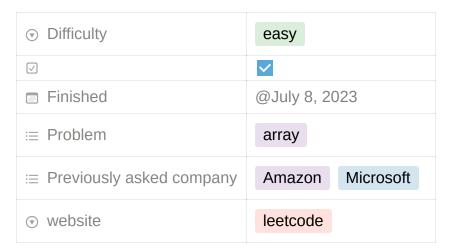
# 1. Two Sum



### **Question:**

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

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

You can return the answer in any order.

# **Example 1:**

```
Input: nums = [2,7,11,15], target = 9
Output: [0,1]
Explanation: Because nums[0] + nums[1] == 9, we return [0, 1].
```

# **Example 2:**

```
Input: nums = [3,2,4], target = 6
Output: [1,2]
```

#### **Example 3:**

```
Input: nums = [3,3], target = 6
Output: [0,1]
```

# My solution:

Time complexity: O(n)

Space complexity: O(n)

1. Two Sum

The only disadvantage is we have to do two traversals/passes in this solution and every time we have to check the current element and the element that is fetched from the hash map must not be from same index.

# **Optimal solution:**

Time complexity: O(n)

Space complexity: O(n)

```
class Solution(object):
    def twoSum(self, nums, target):
        dict = {}
        n = len(nums)
        for i in range(n):
            if target-nums[i] in dict:
                return [i,dict[target-nums[i]]]
        else:
                dict[nums[i]] = i
```

1. Two Sum