

# 001. Two Sum

## 001 Two Sum

- Array
- Hash Table + two pointers

### Description

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].

### 1. Thought Line

(1) map structure **mapHash** is for storing the occurrence numbers of each number.

### 2. Hash Table + Two Pointers

```
1 class Solution {
2 public:
3     vector<int> twoSum(vector<int>& nums, int target) {
4         vector<int> result;
5         map<int,int> mapHash;
6         if (nums.size()<2) return result;
7
8         for (vector<int>::size_type i=0; i<=nums.size()-1; ++i)
9             ++mapHash[nums[i]];
10
11        for (vector<int>::size_type i=0; i<=nums.size()-2; ++i){
12            int currentValue = nums[i];
13            int biasValue = target-nums[i];
14
15            // cannot use the same element twice.
16            --mapHash[currentValue];
17            if (mapHash[biasValue]>0){
18                for (vector<int>::size_type j=i+1; j<=nums.size()-1; ++j){
19                    if (nums[j]==biasValue){
20                        result.push_back(i);
21                        result.push_back(j);
22                        return result;
23                    }
24                }
25            }
26        }
27        return result;
28    }
29 };
```