**Solve two sum problem (Javascript, Java, C#, Swift, Kotlin, Python, C++, Golang)**

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].

**Solution**:

We define a map to store the element with its index as a map;

We are going to check element one by one;

 Let's say the current element is a;

 We store the element with its index into map;

 and use the target to minus the current element to get the difference as b;

 We check if the map can get anything with b as the key, if the value is not empty and is also not the same with its index, we get the answer.

**Javascript**:

twoSum =(nums, target)=> {

    let map = new Map;

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

        let a = nums[i];

        let b = target - a;

        let j = map.get(b);

        if(j !== undefined) {

          if(j !== i){

             return [i, j ];

          }

        }

        map.set(a, i);

    }

    return [];

}

console.log(twoSum([2,7,11,15],9))

console.log(twoSum([3,2,4],6))

console.log(twoSum([3,3],6))

**Java**:

importjava.util.\*;

public classHelloWorld{

     public static void main(String []args){

       System.out.println(Arrays.toString(twoSum(new int[]{2,7,11,15}, 9)));

       System.out.println(Arrays.toString(twoSum(new int[]{3,2,4}, 6)));

                        System.out.println(Arrays.toString(twoSum(newint[]{3,3}, 6)));

     }

             public static int[] twoSum(int[] nums, inttarget) {

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

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

                                                 int a = nums[i];

                                                 int b = target - a;

                                                 Integer j = map.get(b);

                                                 map.put(a, i);

                                                 if(j != null && i != j) {

                                                             return new int[]{i,j};

                                                 }

                        }

                        return new int[]{};

    }

}

**c#**:

usingSystem;

usingSystem.Collections.Generic;

classHelloWorld {

    public static void Main(string[] args)

    {

       Console.WriteLine(string.Join(",", twoSum(newint[]{2,7,11,15}, 9)));

       Console.WriteLine(string.Join(",", twoSum(new int[]{3,2,4},6)));

       Console.WriteLine(string.Join(",", twoSum(new int[]{3,3},6)));

    }

    public static int[] twoSum(int[] nums, inttarget) {

             Dictionary<int,int> map = new Dictionary<int, int>();

             for(inti = 0 ; i< nums.Length; i++) {

                          int a = nums[i];

                          int b = target - a;

             if(map.ContainsKey(b)) {

                 int j = map[b];

                 if(i != j) {

                     return new int[]{i,j};

                 }

             }

             map[a] = i;

             }

             returnnew int[]{};

    }

}

**Swift**:

importFoundation

functwoSum(\_ nums: [Int], \_ target: Int) -> [Int] {

    var map = [Int : Int]()

    for (i, a) in nums.enumerated() {

        var b = target - a;

        var j = map[b]

        if(j != nil) {

            var jj = j!

            return [i,jj]

        }

        map[a] = i

    }

    return []

}

print(twoSum([2,7,11,15],9))

print(twoSum([3,2,4],6))

print(twoSum([3,3],6))

**Kotlin**:

funtwoSum(nums: IntArray, target: Int): IntArray {

    val map:HashMap<Int,Int> =HashMap<Int,Int>()

    for ((i, a) in nums.withIndex()) {

             varb = target - a

        var j = map.get(b)

        if(j != null) {

            var jj = j!!

            if(jj != i){

                return intArrayOf(i, j)

            }

        }

        map.set(a, i)

            }

    return intArrayOf()

}

fun main() {

    println(twoSum(intArrayOf(2,7,11,15),9).joinToString(","))

    println(twoSum(intArrayOf(3,2,4), 6).joinToString(","))

    println(twoSum(intArrayOf(3,3),6).joinToString(","))

}

**Python**:

deftwoSum(nums, target):

    map = {}

    i = 0

    l = len(nums)

    while i < l:

        a = nums[i]

        b = target - a

        j = map.get(b)

        if j is not None:

            if j != i:

                return [i,j]

        map[a] = i

        i += 1

    return []

print(twoSum([2,7,11,15],9))

print(twoSum([3,2,4],6))

print(twoSum([3,3],6))

**C++**:

#include<iostream>

#include<vector>

#include<map>

usingnamespace std;

vector<int>twoSum(vector<int>& nums, int target) {

    std:map<int, int> map;

    vector<int> ret;

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

        int a = nums[i];

        int b = target - a;

        std::map<int,int>::iterator it =map.find(b);

        if(it != map.end()) {

            int j = it->second;

            if(j != i) {

                ret.push\_back(i);

                ret.push\_back(j);

               return ret;

            }

        }

        map.insert(std::pair<int, int>(a,i));

    }

    return ret;

}

int main()

{

    {

        std::vector<int> nums;

        nums.push\_back(2);

        nums.push\_back(7);

        nums.push\_back(11);

        nums.push\_back(15);

        std::vector<int> ret =twoSum(nums, 9);

        for (std::vector<int>::iteratorit = ret.begin() ; it != ret.end(); ++it) {

            std::cout << \*it <<',';

        }

        std::cout << '\n';

    }

    {

        std::vector<int> nums;

        nums.push\_back(3);

        nums.push\_back(2);

        nums.push\_back(4);

        std::vector<int> ret =twoSum(nums, 6);

        for (std::vector<int>::iteratorit = ret.begin() ; it != ret.end(); ++it) {

            std::cout << \*it <<',';

        }

        std::cout << '\n';

    }

    {

        std::vector<int> nums;

        nums.push\_back(3);

        nums.push\_back(3);

        std::vector<int> ret =twoSum(nums, 6);

        for (std::vector<int>::iteratorit = ret.begin() ; it != ret.end(); ++it) {

            std::cout << \*it <<',';

        }

        std::cout << '\n';

    }

    return 0;

}

**Golang**:

package main

import (

            "fmt"

)

functwoSum(nums []int, target int) []int {

        var mapCheck = make(map[int]int)

        var a int

        var b int

            for i:=0; i < len(nums) ; i ++ {

                        a = nums[i]

                        b = target - a

                        if j, ok := mapCheck[b]; ok {

                                    if( j != i){

                                                return[]int{i,j}

                                    }

                        }

                        mapCheck [a] = i

            }

            return []int{}

}

func main(){

            fmt.Println(twoSum([]int{2,7,11,15},9))

            fmt.Println(twoSum([]int{3,2,4}, 6))

            fmt.Println(twoSum([]int{3,3}, 6))

}

