url: https://leetcode-cn.com/problems/two-sum/

**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].**

**The minute I saw this problem,  a initial idea that traveses the whole array came up.**
**And it has been proven worked. Though with low efficieny and quite high time complexity.**
**Considering I haven't done such algorithm problems and barely remembered any valid algorithm to solve this, I gave up and refered to the offical answer.**
**The official answer was give by Java and mainly used hash.**
**I know that dic{} in Python adopts hash structure, so I figured maybe I could finish it by employing dic structure in Python.**

**So this is my solution in Python after some comparsion with comments:**

```
class Solution:
    def twoSum(self, nums: List[int], target: int) -> List[int]:
        hashmap = {}
        for i, num in enumerate(nums):
            another_num = target - num
            if another_num in hashmap.keys():
                return [ hashmap[another_num], i ]
            hashmap[num] = i

        return None
```

**In the meantime, i tried to learn more about C++ and C; Therefore I also borrowed some methods in the review.**

**STL in C++ has a lot of stuff needed to be learned.**
**The specific implementation details, as shown here:**
**C++:**

```
class Solution {
public:
    vector<int> twoSum(vector<int>& nums, int target) {
        map<int,int> hashmap;
        vector<int> result(2,-1);
        for (int i =0; i<nums.size(); i++)
        {
            if (hashmap.count(target - nums[i])>0)
            {
                result[0] = hashmap[target - nums[i]];
                result[1] = i;
                break;}
            hashmap[nums[i]] = i;
        }
       return result;
    }
};
```

**C:**

```
int* twoSum(int* nums, int numsSize, int target) {
    int* result = (int*)malloc(sizeof(int)*2);
    int min = INT_MAX, max = INT_MIN;

    for (int i = 0; i < numsSize; i++) {
        if (nums[i] < min)
            min = nums[i];
        if (nums[i] > max)
            max = nums[i];
    }
```

```
    int len = max-min+1;
    int* hash = malloc(sizeof(int)*len);


    for (int i = 0; i < len; i++)
        hash[i] = -1;
    for (int i = 0; i < numsSize; i++) {
        int diff = target-nums[i]-min;

        if (diff >= 0 && diff < len && hash[diff] >= 0 && i != hash[diff]) {
            result[0] = i;
            result[1] = hash[diff];
            return result;
        }
        else
            hash[nums[i] - min] = i;
    }

    return NULL;
}
```