

• Days 4 - Arsh Goyal Challenge

Two sum

1. Two Sum

Easy 48196 1571 Add to List Share

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]`.

Brute-Force-

`[2, 7, 11, 15]`, $T = 9$

Take first element and add with other element and check if we got our target

`[2, 7, 11, 15]`

$2 + 7 == 9$ return `[i, j]`

If $T = 23$

`[2, 7, 11, 15]`

$i = 0$ $j = 1$ $2 + 7 == 23$ False $j = 2$
 $2 + 11 == 23$ False $j = 3$
 $2 + 15 == 23$ False

$i++$

$i = 1$

$7 + 11 == 23$ False $j = 2$
 $7 + 15 == 23$ True $j = 3$
return `[i, j]`

T.C - $O(n^2)$

S.C - $O(1)$

Better Approach: (Hashing)

let our target is 14 from given array

$[2, 6, 5, 8, 11]$

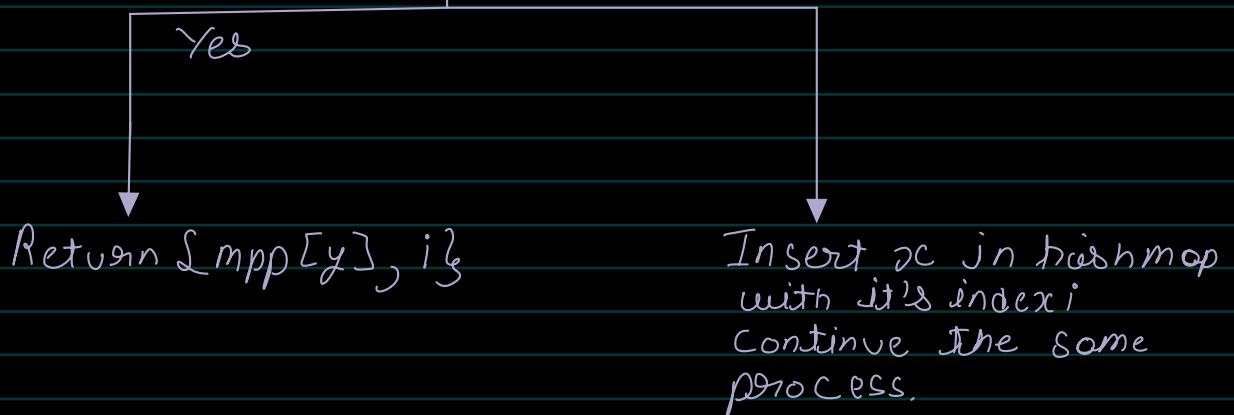
$$x + y = 14$$

↓
I will take from Array

i.e. - 2 $\Rightarrow 2 + y = 14$
 $y = 14 - 2$
 $y = 12$

y is the another number we need to get our target

Now we will check if 12 exist in our hashmap



$[2, 6, 5, 8, 11]$

↑ ↑ ↑ ↑

$i=0$, Find(12)

False

Insert (2: 0)

$i=1$, Find(8)

False

Insert (6: 1)

$i=2$, Find(11)

False

Insert (5: 2)

$i=3$ Find(6)

Found (6) True

return [1, 3]

T.C - $O(n)$
S.C - $O(n)$

element / key	
5	2
6	1
2	0

Index / value

seen element

Optimize Approach: (For Getting True & False only)

First we will sort the array

[2, 5, 6, 8, 11]

and place the pointer left ptr at 0-index and Right at n-1 index

while loop

[2, 5, 6, 8, 11] (Binary Search)

Left Right

$\text{nums}[\text{left}] + \text{nums}[\text{right}] == \text{Target}$ (Return true)

$\text{nums}[\text{left}] + \text{nums}[\text{right}] > \text{Target}$ (Right-)

$\text{nums}[\text{left}] + \text{nums}[\text{right}] < \text{Target}$ (Left++)

Return False

Code

```
class Solution {
public:
    vector<int> twoSum(vector<int>& nums, int target) {
        unordered_map<int,int> mpp;
        for(int i=0;i<nums.size();i++){
            int y=target-nums[i];
            if(mpp.count(y)) return {mpp[y],i};
            mpp[nums[i]]=i;
        }
        return {};
    }
};
```