• 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 if == 9 return {i, j}

if T = 23

[2, 7, 11, 15]

i = 0

2 + 7 == 23 False j = 1

2 + 11 == 23 False j = 2

2 + 15 == 23 False j = 3

i++

i = 1

7 + 11 == 23 False j = 2

7 + 15 = 23 True j = 3

return {i, j}

T.C - O(n²)

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 ⇒ $2 + y = 14$

$y = 14 - 2$

$\boxed{y = 12}$

y is the another number we need to get our target

Now we will check if 12 exist in our hashmap

Yes

Return {mpp[y], i}

Insert x in hashmap with it's index i continue the same process.

[2, 6, 5, 8, 11]

T.C - O(n)

S.C - O(n)

i=0, Find(12)	i=1, Find(8)
False	False
Insert (2:0)	Insert (6:1)

i=2, Find(11)	i=3 Find(6)
False	Found (6) True
Insert (5:2)	return {1, 3}

element / key	
5 : 2	→ Index / value
6 : 1	
2 : 0	

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

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

left → Right

while loop

nums[left] + nums[right] == Target (Return true)

nums[left] + nums[right] > Target (Right--)

nums[left] + nums[right] < Target (Left++)

Return False

Code

```cpp
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 {};

    }
};
```