Two Sum -Practice Que

02 June 2025 08:35

We are given list of integer
nums = [3,4,5,6]
we need to find out the indices of numbers which has target = 7

[3, 4, 5, 6]
0 1 2 3
= 7 = [0, 1]

3	4	5	6

↳ In Brute force approach we just iterate each pair of the elements of the array and check the sum of them is == target or not.

3+4=7 4+3=7 5+6=11 6+3=9
3+5=8 4+5=9 5+4=9 6+4=10
3+6=9 4+6=10 5+3=8 6+5=11

Note: This is not efficient way to find out the solution for this

Better Approach:
Hash Map Indexing

3	4	5	6

If you see in above problem we need to consider one value as assumption from above array & check can only another value from array whether after adding it meets the target in the form of sum
Eg. if we consider 3, we need to check for rest of the elements we will have 4 here meets 3+4=7

HashMap
Val: Index } we will consider empty hash map over here and just update those indices which has sum = target

Here we are using the technique diff = target - element of array
If diff is present in the rest array then second value as reference is our second value.

Eg

0	1	2	3
3	4	5	6

target = 7

7-3 7-4 7-5 7-6
diff. = 4 3 = 2 = 1

4 is present at index 1

3 is present at index 0

These diff are not present elsewhere in the array so that we don't consider them.

So these are the two indices which have sum = 7
= [0, 1]

* Fundamental thing need to be keep in mind.

```
prev_map = {}    val - index
for i, n in enumerate(nums):
    diff = target - n
    if diff in prev_map:
        return [prev_map[diff], i]
    prev_map[n] = i
```

↳ Here if prev_map dict is empty it will update the numbers at the ith index first then execute rest of the loop.