① → Two sum –

1. Question –

Given an array of integers nums and an target, return indices of the two numbers such that they add up to target.

Assumptions –

Each input would have exactly one solutions, and you may not use the same element twice.

2. Examples –

nums = [2, 7, 11, 15], target = 9 → [2, 7]

2 + 7 = 9
0 1

[0, 1]

4 → [1, 2, 3] → [0, 2]

3. Intuition / concept

4. Brute force –

7 in the rest of the array

2 ...

11 ...

2, 7, 11, 15

11, ... → -2 in the rest of the array

T(n) = O(n²) ?

S(n) = O(1) ?

2, 7, 11, 15
i
j

5. Optimized Solution –

Map {

2, 6, 7, 11 | t = 9 ⇒ (key, value) pairs

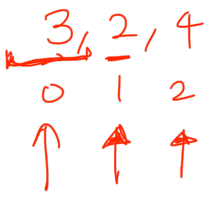
x + y = 9

} x = 9 - 7 = 2

2,

2, 7, 11, 15
0 1 2 3

target = 9 // -6



4?

2

target = 6

{
3 : 0,
2 : 1
}

t = 9

2, 6, 7, 11
0 1 2 3

[0, 2]

{
2 : 0,
6 : 1
}

[2, 1]

O(n)

6. Time complexity / Space complexity-

[2, 7, 11, 15] target = 17

Optimised solution T = O(n)
S = O(n)

{ n : i,
}

7. Code-

```
int[] twoSum(int[] nums, int target) {
    Map<Integer, Integer> map ...
    for(int i=0; i< nums.length; i++) {
        int rem = target - nums[i];
        if (map has rem) {
            return { index of rem,
                     i
            }
        }
        map.insert(nums[i], i)
    }
    return {
        -1,
        -1
    }
}
```