# Problem 1 : Two Sum

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.

Before coding :

- Having an approch
- will perform a for loop
  - For each of element “x”
- will perform  for loop to find “y” where x + y = target
- we want to check if (target - x) exist !!!!
- This is because target - x + x = target
- we will save the previous element
- we should also that return indices of two numbers : that means to track elements and index
- we can create a Hash map to map element → index
- we can check a Hash Map for “target -x”

Answers :

create a Hash Map to check and track of elements and index

- if map contrains (target - nums[i])
  - return (i , maps.get(target - nums[i])
  - put nums[i] and “i” into “map”
  - return an empty array

what’s is the time complexity !

- Time complexity = O(n) is the length if the input array

Solution Code :

```
class Solution {
    public int[] twoSum(int[] nums, int target) {
        Map<Integer,Integer> map = new HashMap<>();
        for(int i =0 ; i < nums.length ; i++){
            if (map.containsKey(target - nums[i])){
                return new int[]{i,map.get(target - nums[i])}
            }
            map.put(nums[i],i);
        }
        return new int[0];

    }
}
```