2.Medium\01.2_sum_problem.cpp

```cpp
/*
QUESITON:-
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].
Example 2:

Input: nums = [3,2,4], target = 6
Output: [1,2]
*/

/*
Approach:

-> Create an empty map to store the elements and their corresponding indices.
>  Iterate through the input array, nums, and for each element:
    Calculate the complement by subtracting the current element from the target value.
    Check if the complement exists in the map.
    If the complement exists, return the indices of the current element and the complement.
    If the complement does not exist, add the current element and its index to the map.
-> If no solution is found, return an empty vector or a message indicating no solution exists.
*/

// CODE:-
vector<int> twoSum(vector<int> &nums, int target)
{
    unordered_map<int, int> mp;
    for (int i = 0; i < nums.size(); i++)
    {
        int remain = target - nums[i];
        if (mp.find(remain) != mp.end() && mp[remain] != i)
            return {i, mp[remain]};
        mp[nums[i]] = i;
    }
    return {-1, -1};
    // If the question asks to just return whether pair exists or not, not the indexes in that case we can sort and easily find the pair sum without extra space
}

// TIME COMPLEXITY = O(N)
// SPACE COMPLEXITY = O(N)
```