1. Two Sum

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

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]
Example 3:

Input: nums = [3,3], target = 6
Output: [0,1]

Constraints:

2 <= nums.length <= 104
-109 <= nums[i] <= 109
-109 <= target <= 109
Only one valid answer exists.

Follow-up: Can you come up with an algorithm that is less than O(n2) time complexity?

Solutions

Brute-Force

```
var twoSum = function(nums, target) {
    for (let i=0;i<nums.length;i++){
        for(let j=i+1;j<nums.length;j++){
            if(nums[i]+nums[j]==target)
                return [i,j]
        }
    }
};
```

Sorting

```
var twoSum = function(nums, target) {
    const sortedNums = nums.slice().sort((a, b) => a - b);
    let left = 0, right = sortedNums.length - 1;
    while (left < right) {
        const sum = sortedNums[left] + sortedNums[right];
        if (sum === target) {
            return [sortedNums[left], sortedNums[right]];
        } else if (sum < target) {
            left++;
        } else {
            right--;
        }
    }
};
```

HashMap

```
var twoSum = function(nums, target) {
    var hashMap =  new Map()
    for(let i=0;i<nums.length;i++){
        let complement = target-nums[i]

        if (hashMap.has(complement)) {
            return [hashMap.get(complement), i];
        }
        hashMap.set(nums[i], i);
    }
};
```