Solution 1

```
class Solution {

    public int[] twoSum(int[] nums, int target) {

        int[] res = new int[2];

        for(int i = 0; i < nums.length; i++) {

            for (int j = i + 1; j < nums.length; j++) {

                if (nums[i] + nums[j] == target) {

                    res[0] = i;

                    res[1] = j;

                    return res; // Return as soon as we find the answer

                }

            }

        }

        return res; // In case no valid pair is found (optional - depends on problem constraints)

    }

}
```

Solution 2(By chatGPT)

Your code works logically for finding two indices whose values in the array sum to the target, but there are **a couple of things to improve**:

✅ 1. Immediate return on finding the answer

Currently, your loop continues checking even after finding the correct pair. You should **return immediately** after finding the solution:

✅ Updated Code:

```
class Solution {
    public int[] twoSum(int[] nums, int target) {
        int[] res = new int[2];
        for(int i = 0; i < nums.length; i++) {
            for (int j = i + 1; j < nums.length; j++) {
                if (nums[i] + nums[j] == target) {
                    res[0] = i;
                    res[1] = j;
                    return res; // Return as soon as we find the answer
                }
            }
        }
        return res; // In case no valid pair is found (optional - depends on problem constraints)
    }
}
```

🧠 Explanation:

- **Time Complexity:** O(n²)
- **Space Complexity:** O(1)
- It returns the **first pair** of indices that sum to the target.

⚡ Bonus: Optimized Version using HashMap (O(n) time)

```
import java.util.HashMap;

class Solution {
    public int[] twoSum(int[] nums, int target) {
        HashMap<Integer, Integer> map = new HashMap<>(); // value -> index
        for (int i = 0; i < nums.length; i++) {
            int complement = target - nums[i];
            if (map.containsKey(complement)) {
                return new int[]{ map.get(complement), i };
            }
            map.put(nums[i], i);
        }
        return new int[]{}; // if no result is found
    }
```

```
}
```

Let me know if you want a breakdown of how this optimized version works.

```
        }
        return new int[]{}; // if no result is found
    }
}
```