# Dry Run:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **i** | **j** | **nums[i]** | **nums[j]** | **Sum** | **Match?** |
| 0 | 1 | 3 | 1 | 4 | ❌ |
| 0 | 2 | 3 | 4 | 7 | ❌ |
| 0 | 3 | 3 | 6 | 9 | ❌ |
| 0 | 4 | 3 | 5 | 8 | ❌ |
| 0 | 5 | 3 | 9 | 12 | ❌ |
| 1 | 2 | 1 | 4 | 5 | ❌ |
| 1 | 3 | 1 | 6 | 7 | ❌ |
| 1 | 4 | 1 | 5 | 6 | ❌ |
| 1 | 5 | 1 | 9 | **10** | ✅ → return [1, 5] |

# Solution 1

class Solution {

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

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

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

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

return new int[]{i, j};

}

}

return new int[]{};

}

}

# Dry Run:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **i** | **nums[i]** | **Complement** | **Map (value → index)** | **Found Complement?** | **Action** |
| 0 | 3 | 7 | {} | ❌ | Add 3 → map[3]=0 |
| 1 | 1 | 9 | {3=0} | ❌ | Add 1 → map[1]=1 |
| 2 | 4 | 6 | {3=0, 1=1} | ❌ | Add 4 → map[4]=2 |
| 3 | 6 | 4 | {3=0, 1=1, 4=2} | ✅ → found 4 | Return [2, 3] ✅ |

# Solution 2

class Solution {

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

        //{7:1}

        Map<Integer, Integer> a = new HashMap<>();

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

            if(a.containsKey(target - nums[i]))

                return new int[]{a.get(target-nums[i]), i};

            a.put(nums[i], i);

        }

        return new int[]{};

    }

}