# Algorithm and Data Structures

## Problem Statement

Given an array of integers, return the indices of the two numbers such that they add up to a specific target.

## Thought Process

1. Understanding the Problem:   
 - We need to find two numbers in an array that sum up to a given target value.  
 - The output should be the indices of these two numbers.  
  
2. Approach:  
 - We can use a hash map (or object in JavaScript) to store the numbers we have seen so far and their respective indices.  
 - As we iterate through the array, for each number, we check if the difference between the target and the current number exists in the hash map.  
 - If it does, we return the indices of the current number and the number found in the hash map.  
 - If it does not exist, we add the current number and its index to the hash map.  
  
3. Complexity Analysis:  
 - Time Complexity: O(n) - We traverse the list of numbers only once.  
 - Space Complexity: O(n) - We use additional space for the hash map to store the numbers.

## Implementation

Here is the implementation in JavaScript:  
  
```javascript  
/\*\*  
 \* Function to find two indices of numbers in an array that sum to a specific target.  
 \* @param {number[]} nums - The array of integers.  
 \* @param {number} target - The target sum.  
 \* @return {number[]} - Indices of the two numbers that add up to the target.  
 \*/  
function twoSum(nums, target) {  
 const numMap = new Map(); // Create a map to store numbers and their indices  
  
 for (let i = 0; i < nums.length; i++) {  
 const complement = target - nums[i]; // Calculate the required number to reach the target  
   
 // Check if the complement exists in the map  
 if (numMap.has(complement)) {  
 return [numMap.get(complement), i]; // Return the indices of the two numbers  
 }  
   
 // Store the current number and its index in the map  
 numMap.set(nums[i], i);  
 }  
  
 throw new Error("No two sum solution"); // Throw error if no solution found  
}  
  
// Example usage  
const numbers = [2, 7, 11, 15];  
const target = 9;  
console.log(twoSum(numbers, target)); // Output: [0, 1]  
```

## README File

# Two Sum Problem  
  
## Description  
This project implements a solution to the 'Two Sum' problem, which involves finding two numbers in an array that add up to a specific target. The solution utilizes a hash map to achieve an efficient O(n) time complexity.  
  
## Installation  
To run this application, ensure you have Node.js installed. You can clone this repository and run the following command:  
  
```bash  
npm install  
```  
  
## Usage  
You can call the `twoSum` function with an array of integers and a target value. The function will return the indices of the two numbers that add up to the target.  
  
```javascript  
const numbers = [2, 7, 11, 15];  
const target = 9;  
const result = twoSum(numbers, target);  
console.log(result); // Output: [0, 1]  
```  
  
## Error Handling  
If no two numbers sum to the target, the function will throw an error:  
  
```javascript  
throw new Error("No two sum solution");  
```  
  
## Complexity Analysis  
- \*\*Time Complexity\*\*: O(n) - The function iterates through the array once.  
- \*\*Space Complexity\*\*: O(n) - The hash map stores numbers and their indices.  
  
## Third-Party Libraries  
No third-party libraries were used in this implementation.

## Git Commands for Version Control

To create a new branch, commit changes, and merge back into the main branch, you can use the following Git commands:  
  
```bash  
# Create a new branch  
git checkout -b feature/two-sum  
  
# Add changes  
git add .  
  
# Commit changes  
git commit -m "Implement two sum solution"  
  
# Switch back to the main branch  
git checkout main  
  
# Merge the feature branch into main  
git merge feature/two-sum  
  
# Push changes to the remote repository  
git push origin main  
```

## Final Steps

Make sure to review your code and README for clarity and completeness.  
After completing these steps, submit your Git repository link as required.