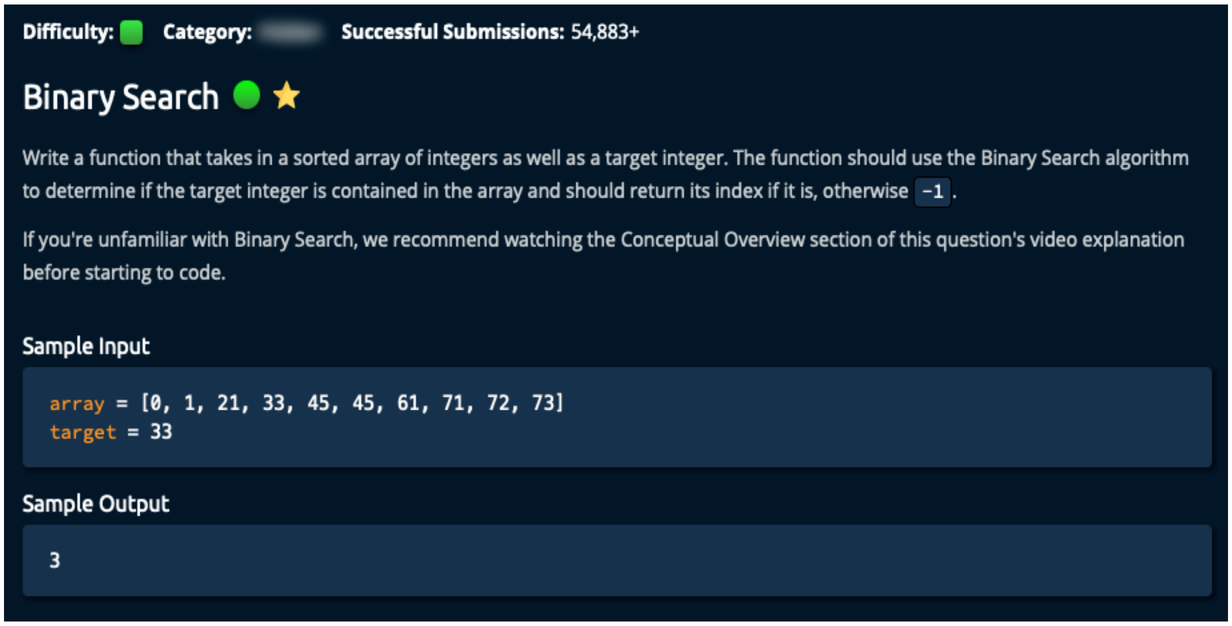
Code Challenge #14 Binary Search (Easy)


Difficulty: 🟩 **Category:** [redacted] **Successful Submissions:** 54,883+

Binary Search ● ★

Write a function that takes in a sorted array of integers as well as a target integer. The function should use the Binary Search algorithm to determine if the target integer is contained in the array and should return its index if it is, otherwise `-1`.

If you're unfamiliar with Binary Search, we recommend watching the Conceptual Overview section of this question's video explanation before starting to code.

Sample Input

```
array = [0, 1, 21, 33, 45, 45, 61, 71, 72, 73]
target = 33
```

Sample Output

```
3
```

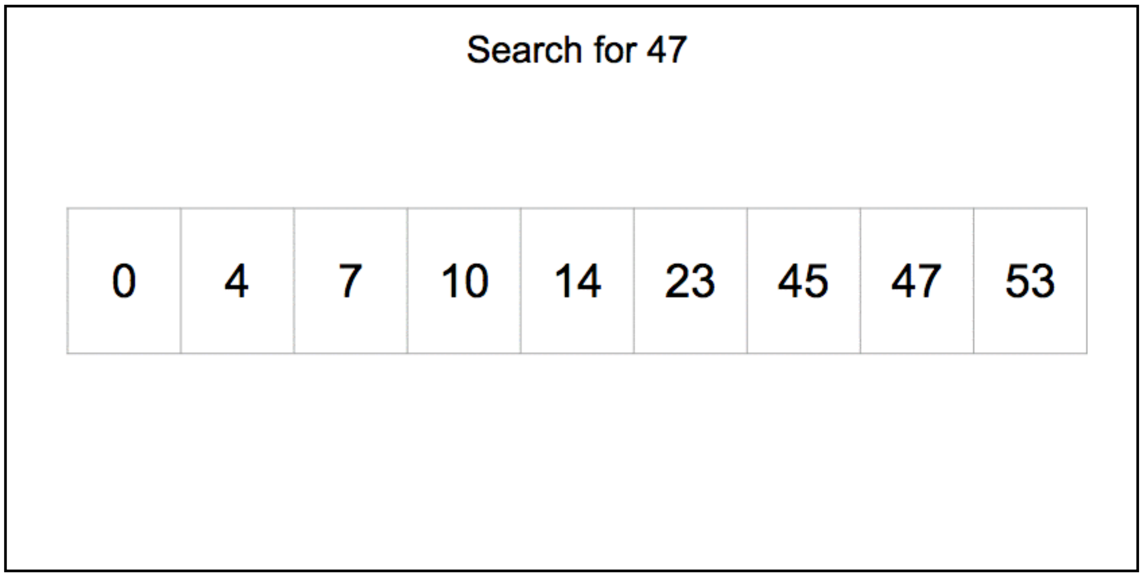
Solution #1

```
function binarySearch(array, target) {
  return binarySearchHelper(array, target, 0, array.length - 1);
}

function binarySearchHelper(array, target, left, right) {
 if (left > right) return -1;
 const middle = Math.floor((left + right) / 2);
 const potentialMatch = array[middle];
 if (target === potentialMatch) {
                return middle;
        } else if (target < potentialMatch) {
                return binarySearchHelper(array, target, left, middle - 1)
        } else {
                return binarySearchHelper(array, target, middle + 1, right);
        }
 }
```

Explanation

The binary search algorithm is a search algorithm based on an ordered list of items. We have a target value for example 47. We look for the middle value in a list of items, in the example below the middle number is 14. We compare the target value with the middle item. If it is larger than it, we know to remove all numbers (half of the list of numbers) below 14. We then look for the next middle value in the remaining list of items aka 45 and compare it to the target value and repeat the process until we find the target value or we show that the value is not located in the array.



The solution for this problem is implemented using a helper method inside the main method called binarySearch. This main method called binarySearch only returns the helper method called binarySearchHelper. The binarySearchHelper helper method takes four initial arguments which are array, target, 0, array.length – 1. The default value for the four arguments of the binarySearchHelper methods are array, target, left and right when initially implemented. The first part of the binarySearchHelper method includes an edge case test to see if the left value is greater than the right value. This would only occur if the target number were a number that did not exist in the array. If this condition is true we will return -1. We need to find the middle value of the array and we do this by creating a const variable called middle which calculates the middle index value using Math.floor((left + right) / 2). We use Math.floor to round down the index value. We then have a const variable called potentialMatch which is the value at the middle of the array using array[middle]. We check to see if the target value === potentialMatch. If it does we return middle (index value) else if the target value is less than the potentialMatch value we get rid of the values to the right of the potential match by moving the right pointer to the left of the middle value by using middle – 1. We then use the helper method again by calling return on it using the new right pointer value (return binarySearchHelper(array, target, left, middle – 1)). If the target value is greater than the potential match we move the left pointer variable to the right of the middle value by using middle + 1. We do this by using another else statement and return binarySearchHelper(array, target, middle + 1, right). This process continues until we find the target value in the array or if the value does not exist in the array we return -1 based on the first edge case test. This code runs in O(n) time complexity.