Difficulty: **Category:** Searching **Successful Submissions:** 44,792+

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
```

Hints

Hint 1

The Binary Search algorithm works by finding the number in the middle of the input array and comparing it to the target number. Given that the array is sorted, if this middle number is smaller than the target number, then the entire left part of the array is no longer worth exploring since the target number can no longer be in it; similarly, if the middle number is greater than the target number, then the entire right part of the array is no longer worth exploring. Applying this logic recursively eliminates half of the array until the number is found or until the array runs out of numbers.

Hint 2

Write a helper function that takes in two additional arguments: a left pointer and a right pointer representing the indices at the extremities of the array (or subarray) that you are applying Binary Search on. The first time this helper function is called, the left pointer should be zero and the right pointer should be the final index of the input array. To find the index of the middle number mentioned in Hint #1, simply round down the number obtained from: (left + right) / 2. Apply this logic recursively until you find the target number or until the left pointer becomes greater than the right pointer.

Hint 3

Can you implement this algorithm iteratively? Are there any advantages to doing so?

Optimal Space & Time Complexity

O(log(n)) time | O(1) space - where n is the length of the input array