Sorted & Rotated Array Search

Given a rotated, sorted array and a target value, return the index of the target value. If the target value does not exist inside of the collection, return -1.

A rotated, sorted array means that some number of elements have been taken from one end of the array and moved to the other end while maintaining the sorted status of the moved elements.

[**1,2**,3,4,5,6,7] → [3,4,5,6,7,**1,2**]  
  
Input: Array of integers, target value  
Output: Integer index of target value (-1 if not found)

# Example

Input: [6,8,11,15,17,3,5], 3 => Output: 5

Input: [6,8,11,15,17,3,5], 10 => Output: -1

# Constraints

Time Complexity: O(log(N))  
Auxiliary Space Complexity: O(log(N))  
  
All elements of the array are unique

# Solution

Perform a modified binary search:

1. Initialize a ‘start index of 0 and an ‘end’ of index length - 1
2. Perform a while loop until ‘end’ is less than ‘start’
   1. Compute the ‘mid’ index
   2. If the ‘target’ is found at ‘mid’, return ‘mid’
   3. Otherwise, check which side is sorted by comparing the ‘start’ and ‘end’ values to the ‘mid’. The target value should either exist in-between the sorted region or outside.
   4. Discard the unused half by updating the ‘end’ to mid - 1 or ‘start’ to mid+1
3. If the while loop breaks without a matching value found, return -1

# Resources

https://leetcode.com/problems/search-in-rotated-sorted-array/