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

**Search in Rotated Sorted Array**

**There is an integer array nums sorted in ascending order (with distinct values).**

**Prior to being passed to your function, nums is possibly rotated at an unknown pivot index k (1 <= k < nums.length) such that the resulting array is [nums[k], nums[k+1], ..., nums[n-1], nums[0], nums[1], ..., nums[k-1]] (0-indexed). For example, [0,1,2,4,5,6,7] might be rotated at pivot index 3 and become [4,5,6,7,0,1,2].**

**Given the array nums after the possible rotation and an integer target, return the index of target if it is in nums, or -1 if it is not in nums.**

**You must write an algorithm with O(log n) runtime complexity.**

Example 1:

Input: nums = [4,5,6,7,0,1,2], target = 0

Output: 4

Example 2:

Input: nums = [4,5,6,7,0,1,2], target = 3

Output: -1

Example 3:

Input: nums = [1], target = 0

Output: -1

Constraints:

1 <= nums.length <= 5000

-104 <= nums[i] <= 104

All values of nums are unique.

nums is an ascending array that is possibly rotated.

-104 <= target <= 104

**Method 1: ()**

**Intuition**

Let's go with an example array[0,1,2,4,5,6,7]

We can rotate it to get 7 different arrays:(which can be divided into three groups)

0, 1, 2, 4, 5, 6, 7 //Group 1 -- no rotation

7, 0, 1, 2, 4, 5, 6 //Group 2 -- num[l] > num[mid]

6, 7, 0, 1, 2, 4, 5

5, 6, 7, 0, 1, 2, 4

4, 5, 6, 7, 0, 1, 2 //Group 3 -- num[mid] > num[r]

2, 4, 5, 6, 7, 0, 1

1, 2, 4, 5, 6, 7, 0

**Approach**

Divide input array into parts :

[left of mid], [mid element] , [right of mid]

If mid element is the target then return mid.

Now we check, which Group the given input vector nums belongs to.

For Group 2 : right of mid is sorted.

So check if target belongs to [right of mid].

If it does, select [right of mid] as new vector where search needs to be conducted, otherwise select [left of mid]

For Group 3 : left of mid is sorted.

So check if target belongs to [left of mid].

If it does, select [left of mid] as new vector where search needs to be conducted, otherwise select [right of mid]

Note : No need to write separate condition for Group 1 since Code for Group 3 also satisfies condition for Group 1.

Time Complexity: O(log n) *[Binary Search]*

Space Complexity: O(1) *[]*

int search(vector<int>& nums, int target) {

        int l=0, r= nums.size()-1, mid;

        while(l<=r){

            mid = l + (r-l)/2;

            if(target==nums[mid])

                return mid;

            else if(nums[l]>nums[mid])

                    if(target>nums[mid] && target<=nums[r])

                        l = mid + 1;

                    else r = mid - 1;

            else {

                    if(target>=nums[l] && target<nums[mid])

                        r = mid - 1;

                    else l = mid + 1;

            }

        }

        return -1;

    }