33. Search in Rotated Sorted Array

Medium

Suppose an array sorted in ascending order is rotated at some pivot unknown to you beforehand.

(i.e., [0,1,2,4,5,6,7] might become [4,5,6,7,0,1,2]).

You are given a target value to search. If found in the array return its index, otherwise return -1.

You may assume no duplicate exists in the array.

Your algorithm's runtime complexity must be in the order of *O*(log *n*).

**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

int search(int\* nums, int numsSize, int target) {

int low=0,high=numsSize-1,mid,x,y,a,b;

while(low<=high){

mid=(low+high)/2;

if(nums[mid]==target) return mid;

if(nums[low]<=nums[mid]){

x=low; y=mid;

a=mid+1; b=high;

}else{

x=mid+1; y=high;

a=low; b=mid-1;

}

if(nums[x]<=target&&target<=nums[y]){

low=x; high=y;

}else{

low=a; high=b;

}

}

return -1;

}

Success

[Details](https://leetcode.com/submissions/detail/205591751/)

Runtime: 0 ms, faster than 100.00% of C online submissions for Search in Rotated Sorted Array.

Memory Usage: 815.1 KB, less than 100.00% of C online submissions for Search in Rotated Sorted Array.