class Solution:

def search(self, nums, target):

inp=nums

t=target

high = len(inp)-1

low=0

global res

res=-1

def helper(A,high,low,x):

global res

*#print("")*

*if* low>high:

*return*

mid=(high-low)/2+low

mid=int(mid)

*#print("L,M,H", A[low], A[mid], A[high], end=",")*

*if* A[mid]==x:

*#print("case 1")*

res=mid

*#print("this is result",res)*

*return* mid

*# to Optimize use the below code*

*# if A[low]<=A[high]:*

*# #print("case 2", end='')*

*# if A[mid]<x:*

*# #print("case 2A mid<x")*

*# low=mid+1*

*# else:*

*# #print("case 2b mid>x")*

*# high=mid-1*

helper(A, high, low, x)

*else*:

*if* A[low]<=A[mid]:

*if* x>=A[low] and x<A[mid]:

high = mid - 1

*else*:low=mid+1

*if* A[mid]<=A[high]:

*if* x>A[mid] and x<=A[high]:

low = mid + 1

*else*:high=mid-1

helper(A, high, low, x)

helper(inp, high, low, t)

*return* res

def search(self, nums, target):

if not nums:

return -1

return self.binarySearch(nums, target, 0, len(nums)-1)

def binarySearch(self, nums, target, start, end):

if end < start:

return -1

mid = (start+end)/2

if nums[mid] == target:

return mid

if nums[start] <= target < nums[mid]: # left side is sorted and has target

return self.binarySearch(nums, target, start, mid-1)

elif nums[mid] < target <= nums[end]: # right side is sorted and has target

return self.binarySearch(nums, target, mid+1, end)

elif nums[mid] > nums[end]: # right side is pivoted

return self.binarySearch(nums, target, mid+1, end)

else: # left side is pivoted

return self.binarySearch(nums, target, start, mid-1)

def search(self, nums, target):

i = 0

j = len(nums) - 1

while i <= j:

mid = (i + j) / 2

if nums[mid] == target:

return mid

elif nums[i] == target:

return i

elif nums[j] == target:

return j

elif nums[mid] > nums[i]:

if target > nums[mid]:

i = mid + 1

elif target > nums[i]:

j = mid -1

else:

i = mid + 1

else:

if target < nums[mid]:

j = mid -1

elif target > nums[i]:

j = mid -1

else:

i = mid + 1

return -1

