旋转数组查找LeetCode\_33\_SearchInRotatedSortedArray\_剑指Offer\_11\_旋转数组的最小数字

# LeetCode\_33\_SearchInRotatedSortedArray

## 题目介绍

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

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

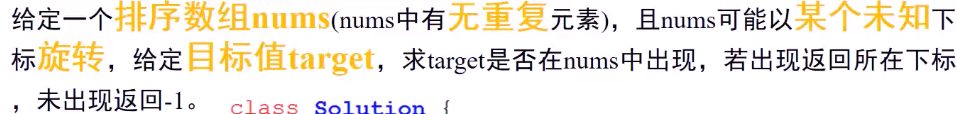
Output: 4

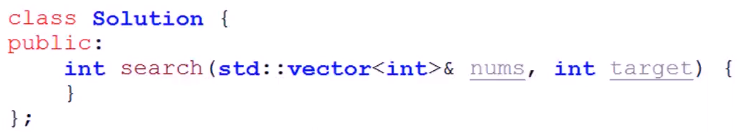
Example 2:

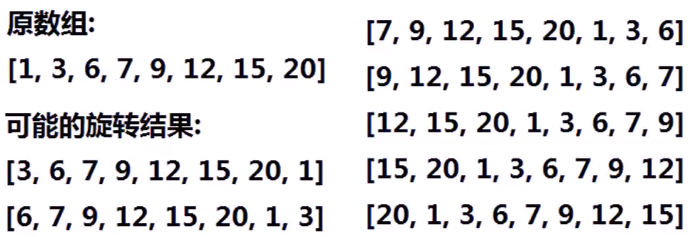
Input: nums = [4,5,6,7,0,1,2], target = 3

Output: -1

难度：Medium

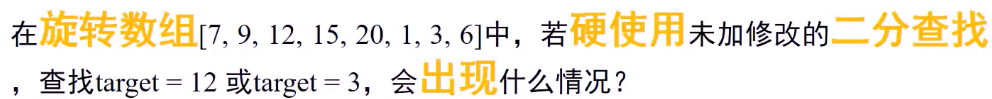


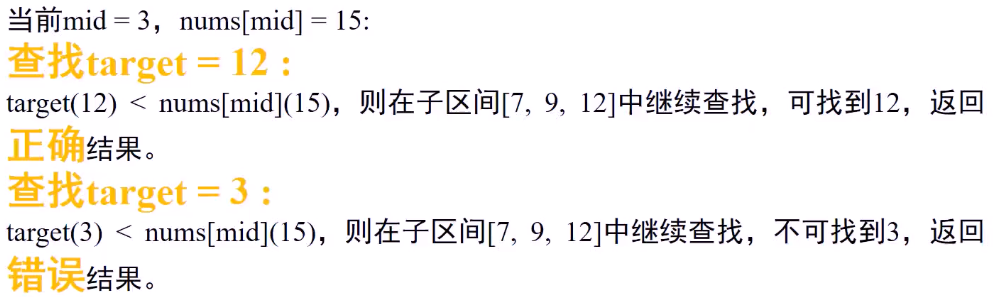


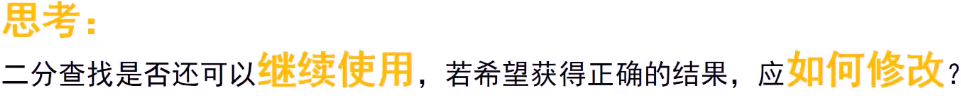


## 分析思路：

### 思考：







### 解决思路

分情况讨论：

首先根据**target与nums[mid]**进行比较，分成三种情况:**等于、大于、小于**。

这样仅仅是进行分类，不能得到什么有效信息。

第二个判断条件nums[begin]与nums[mid]比较重要，可以获知**递增区间与旋转区间的前后**。

* nums[begin]<nums[mid]：可以得出[begin,mid-1]肯定为递增区间，[mid+1,end]为旋转区间。
* nums[begin]<nums[mid]：可以得出[begin,mid-1]肯定为旋转区间，[mid+1,end]为递增区间。

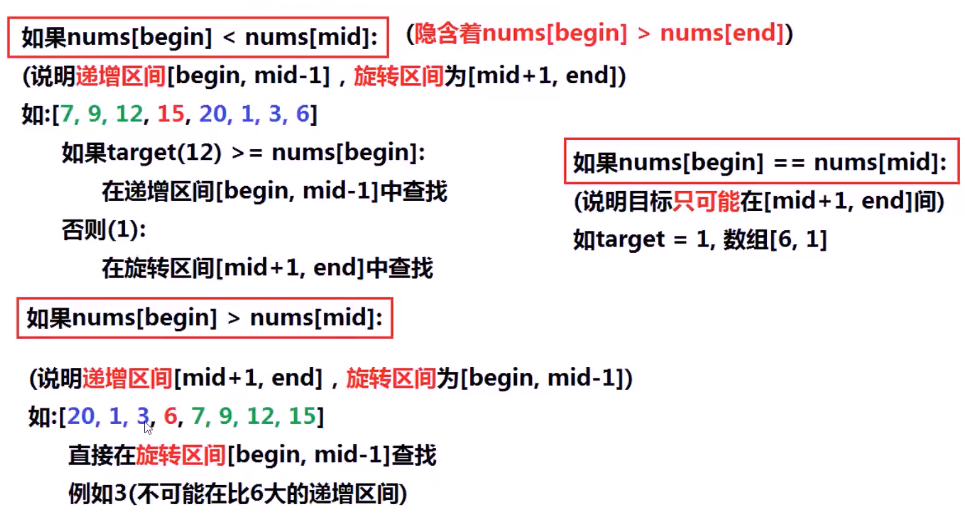
两个判断条件结合在一起就可以知道下一步应该搜索的区间，从而while循环实现。

下面分成7种情况讨论:

* **target==nums[mid];直接返回mid；**
* **target<nums[mid]且nums[begin]<nums[mid];前一段为递增区间，下次搜索哪一段需要由target与nums[begin]决定，若target<=nums[begin],返回后一段区间，low=mid+1; target>nums[begin]返回前一段区间，high=mid-1;**
* **target<nums[mid]且nums[begin]>nums[mid];后一段为递增区间，下次搜索后一段区间。**
* **target>nums[mid]且nums[begin]<nums[mid];前一段为递增区间，下次搜索后一段区间,low = mid+1。**
* **target>nums[mid]且nums[begin]>nums[mid];后一段为递增区间，下次搜索哪一段需要由target与nums[begin]决定，若target<=nums[begin],返回后一段区间，low=mid+1; target>nums[begin]返回前一段区间，high=mid-1;**

### **target=nums[mid]:** 直接返回**mid**。

### **target<nums[mid]**



### **target>nums[mid]**



## 代码实现

### 示例代码





### Java代码实现：

public class **LeetCode\_33\_SearchInRotatedSortedArray** {

@Test

public void test(){

int[] nums = {23,25,34,56,199,2,5,6,10};

for(int i = 0;i < nums.length;i++){

int target = nums[i];

int index = binarySearchInRotatedSortedArray(nums,0,nums.length-1,target);

System.out.println("target+\";index = \" + index = " + target+";index = " + index);

}

int target = 66;

int index = binarySearchInRotatedSortedArray(nums,0,nums.length-1,target);

System.out.println("target+\";index = \" + index = " + target+";index = " + index);

}

public int **binarySearchInRotatedSortedArray**(int[] nums,int low,int high,int target){

while(low <= high){

int midIndex = (low + high)/2;

if(target < nums[midIndex]) {

if(nums[low] <= nums[midIndex]){//前一段为递增区间

if(target > nums[low]){

high = midIndex -1;

}else if (target < nums[low]){

low = midIndex+1;

}else{

return low;

}

}else{//nums[low] > nums[midIndex]后一段为递增区间

high = midIndex -1;

}

// else{

// low = midIndex +1;

// }

}else if(target > nums[midIndex]){

if(nums[low] <= nums[midIndex]){//前一段为递增区间(可以归在一起)

low = midIndex+1;

}else{ //(nums[low] > nums[midIndex])

if(target > nums[low]){

high = midIndex - 1;

}else if (target < nums[low]){

low = midIndex + 1;

}else{

return low;

}

}

// else{//容易忽略

// low = midIndex + 1;

// }

}else{//target == nums[midIndex]

return midIndex;

}

}

return -1;

}

}

# 剑指Offer\_11\_旋转数组的最小数字

这个题其实要比第一题更难些。

## 题目介绍

/\*\*\*\*\*剑指Offer题目：旋转数组的最小数字\*\*\*\*\*\*\*\*/

/\*\*

\* 给定一个旋转数组，返回最小数字。

\* 题目描述：

\* 把一个数组最开始的若干个元素搬到数组的末尾，我们称之为数组的旋转。

\* 输入一个非减排序的数组的一个旋转，输出旋转数组的最小元素。

\* 例如数组{3,4,5,1,2}为{1,2,3,4,5}的一个旋转，该数组的最小值为1。

\* NOTE：给出的所有元素都大于0，若数组大小为0，请返回0。

\*/

## 思路介绍

这里也是基于二分查找，但是改动的比较多。如果是给定一个值target，去寻找位置，只需要比较target与nums[mid]即可，但是这里没有给定target，自己寻找最小值。

策略：利用两个指针，一个low，一个high，分别指向前一段和后一段，等到high-low==1即high指向前一段的最后一个也就是最大值，low指向后一段的第一个即最小值时，返回nums[high]即代表寻找到。

那么如何根据情况更改low和high呢？

如果nums[mid]大于等于nums[low],说明mid还在前一段，就让low=mid；

否则，也就是mid在后一段，那么让high=mid。

**注意：这里一定不要用mid+1或mid-1,以免low指向第二段或者high指向第一段。**

## Java代码

@Test

public void test1(){

int[] nums = {6501,6828,6963,7036,7422,7674,8146,8468,8704,8717,9170,9359,

9719,9895,9896,9913,**9962**,**154**,293,334,492,1323,1479,1539,1727,

1870,1943,2383,2392,2996,3282,3812,3903,4465,4605,4665,4772,

4828,5142,5437,5448,5668,5706,5725,6300,6335};

int min = minNumberInRotateArray(nums);

System.out.println("min = " + min);

}

public int minNumberInRotateArray(int [] nums) {

if(nums == null||nums.length == 0) return 0;

int low = 0,high = nums.length-1;

**if(nums[low] < nums[high]) return nums[low];** //没有旋转的情况

while(nums[low] >= nums[high]){//有旋转

**if(high - low == 1) return nums[high];//终止条件**

int mid = (low + high)>>1;

if(nums[mid]==nums[high]&&nums[mid]==nums[low]){//三者相同的时候，只能顺序查找如[1,0,1,1,1,1]

return searchMinNum(nums,low,high);

}

//说明：传递mid，不是mid+1或mid-1目的就是让low始终指向前一段，而high指向后一段；

// 否则low会指向后一段，high会指向前一段

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

**low = mid;//**一定是mid，不要习惯性mid+1,否则错误

}else{//nums[mid] < nums[low]

**high = mid**;//一定是mid，不要习惯性mid-1,否则错误

}

}

return -1;

}

public int **searchMinNum**(int[] nums,int low,int high){

int min = nums[low];

for(int i = low+1;i <= high;i++){

if(min > nums[i]) min = nums[i];

}

return min;

}