# 11. 旋转数组的最小数字

## 题目链接

[牛客网](https://www.nowcoder.com/practice/9f3231a991af4f55b95579b44b7a01ba?tpId=13&tqId=11159&tPage=1&rp=1&ru=/ta/coding-interviews&qru=/ta/coding-interviews/question-ranking&from=cyc_github)

## 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。输入一个非递减排序的数组的一个旋转，输出旋转数组的最小元素。

## 解题思路

将旋转数组对半分可以得到一个包含最小元素的新旋转数组，以及一个非递减排序的数组。新的旋转数组的长度是原数组的一半，从而将问题规模减少了一半，这种折半性质的算法的时间复杂度为 O(log2N)。

此时问题的关键在于确定对半分得到的两个数组哪一个是旋转数组，哪一个是非递减数组。我们很容易知道非递减数组的第一个元素一定小于等于最后一个元素。

通过修改二分查找算法进行求解（l 代表 low，m 代表 mid，h 代表 high）：

* 当 nums[m] <= nums[h] 时，表示 [m, h] 区间内的数组是非递减数组，[l, m] 区间内的数组是旋转数组，此时令 h = m；
* 否则 [m + 1, h] 区间内的数组是旋转数组，令 l = m + 1。

public int minNumberInRotateArray(int[] nums) {
 if (nums.length == 0)
 return 0;
 int l = 0, h = nums.length - 1;
 while (l < h) {
 int m = l + (h - l) / 2;
 if (nums[m] <= nums[h])
 h = m;
 else
 l = m + 1;
 }
 return nums[l];
}

如果数组元素允许重复，会出现一个特殊的情况：nums[l] == nums[m] == nums[h]，此时无法确定解在哪个区间，需要切换到顺序查找。例如对于数组 {1,1,1,0,1}，l、m 和 h 指向的数都为 1，此时无法知道最小数字 0 在哪个区间。

public int minNumberInRotateArray(int[] nums) {
 if (nums.length == 0)
 return 0;
 int l = 0, h = nums.length - 1;
 while (l < h) {
 int m = l + (h - l) / 2;
 if (nums[l] == nums[m] && nums[m] == nums[h])
 return minNumber(nums, l, h);
 else if (nums[m] <= nums[h])
 h = m;
 else
 l = m + 1;
 }
 return nums[l];
}

private int minNumber(int[] nums, int l, int h) {
 for (int i = l; i < h; i++)
 if (nums[i] > nums[i + 1])
 return nums[i + 1];
 return nums[l];
}