# 题目

已知一个长度为n的数组，预先按照升序排列，经由1到n次旋转后，得到输入数组。例如，原数组nums = [0,1,2,4,5,6,7] 在变化后可能得到：

若旋转 4 次，则可以得到 [4,5,6,7,0,1,2]

若旋转 7 次，则可以得到 [0,1,2,4,5,6,7]

注意，数组 [a[0], a[1], a[2], ..., a[n-1]] 旋转一次 的结果为数组 [a[n-1], a[0], a[1], a[2], ..., a[n-2]] 。

给你一个元素值 互不相同 的数组 nums ，它原来是一个升序排列的数组，并按上述情形进行了多次旋转。请你找出并返回数组中的 最小元素 。

你必须设计一个时间复杂度为 O(log n) 的算法解决此问题。

示例 1：

输入：nums = [3,4,5,1,2]

输出：1

解释：原数组为 [1,2,3,4,5] ，旋转 3 次得到输入数组。

示例 2：

输入：nums = [4,5,6,7,0,1,2]

输出：0

解释：原数组为 [0,1,2,4,5,6,7] ，旋转 3 次得到输入数组。

示例 3：

输入：nums = [11,13,15,17]

输出：11

解释：原数组为 [11,13,15,17] ，旋转 4 次得到输入数组。

提示：

n == nums.length

1 <= n <= 5000

-5000 <= nums[i] <= 5000

nums 中的所有整数 互不相同

nums 原来是一个升序排序的数组，并进行了 1 至 n 次旋转

# 分析

## 方法一：二分查找

思路：

代码：

class Solution {

public:

int findMin(vector<int>& nums) {

int low = 0, high = nums.size() - 1;

while (low < high) {

int mid = (high - low) /2 + low;

if (nums[mid] < nums[high]) {

high = mid;

} else {

low = mid + 1;

}

}

return nums[low];

}

};

复杂度分析：

时间复杂度：时间复杂度为O(logn)，其中n是数组nums的长度。在二分查找的过程中，每一步会忽略一半的区间，因此时间复杂度为O(logn)。

空间复杂度：O(1)。