

第21题——旋转数组的最小数字

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。输入一个递增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。

示例 1:

输入: [3,4,5,1,2]

输出: 1

示例 2:

输入: [2,2,2,0,1]

输出: 0

二分法 (1)

算法流程：初始化：声明 i, j 指针分别指向 `nums` 数组左右两端；循环二分：设 $m = (i + j) / 2$ 为每次二分的中点

($"/$ 代表向下取整除法，因此恒有 $i \leq m < j$)，可分为以下三种情况：

1. 当 $nums[m] > nums[j]$ 时, m 一定在 左排序数组 中，即旋转点 x 一定在 $[m + 1, j]$ 闭区间内，因此执行 $i = m + 1$;
2. 当 $nums[m] < nums[j]$ 时, m 一定在 右排序数组 中，即旋转点 x 一定在 $[i, m]$ 闭区间内，因此执行 $j = m$;
3. 当 $nums[m] = nums[j]$ ，无法判断旋转点 x 在 $[i, m]$ 还是 $[m + 1, j]$ 区间中。解决方案： 执行 $j = j - 1$ 缩小判断范围。

返回值：当 $i = j$ 时跳出二分循环，并返回 旋转点的值 `nums[i]` 即可。

本题参考LeetCode官方题解.

```
class Solution {
public:
    int minArray(vector<int>& numbers) {
        int left=0;//左边界
        int right=numbers.size()-1;//右边界
```

```
while(left<right)
{
    int mid=(right+left)/2;//中间下标
    if(numbers[mid]<numbers[right])
    {
        /*如果 right>mid 则说明该段一定是有序的*/
        right=mid;//边界收缩
    }
    else if(numbers[mid]>numbers[right])
    {
        /*如果 mid>right 则分界点一定在右侧，
        且最小元素一定不包括mid,因此，左边界收缩至mid+1*/
        left=mid+1;
    }
    else
    {
        /*如果 mid==right 则使用暴力方法逐个收缩右边界即可*/
        right--;
    }
}
return numbers[left];
}
};
```