返回 false。具体过程请参看如下代码中的 isPalindrome 方法。

```
public boolean isPalindrome(int n) {
    if (n == Integer.MIN_VALUE) {
        return false;
    }
    n = Math.abs(n);
    int help = 1;
    while (n / help >= 10) { // 防止 help 溢出
        help *= 10;
    }
    while (n != 0) {
        if (n / help != n % 10) {
            return false;
        }
        n = (n % help) / 10;
        help /= 100;
    }
    return true;
}
```

在有序旋转数组中找到最小值

【题目】

有序数组 arr 可能经过一次旋转处理,也可能没有,且 arr 可能存在重复的数。例如,有序数组[1,2,3,4,5,6,7],可以旋转处理成[4,5,6,7,1,2,3]等。给定一个可能旋转过的有序数组 arr, 返回 arr 中的最小值。

【难度】

尉★★☆☆

【解答】

为了方便描述,我们把没经过旋转前,有序数组 arr 最左边的数,在经过旋转之后所处的位置叫作"断点"。例如,题目例子里的数组,旋转后断点在1所处的位置,也就是位置4。如果没有经过旋转处理,断点在位置0。那么只要找到断点,就找到了最小值。

本书提供的方式做到了尽可能多地利用二分查找,但是最差情况下仍无法避免 O(N)的时间复杂度。我们假设目前想在 arr[low..high]这个范围上找到这个范围的最小值(那么初始时 low=0,high==arr.length-1),以下是具体过程:

- 1. 如果 arr[low] <arr[high], 说明 arr[low..high] 上没有旋转, 断点就是 arr[low], 返回 arr[low] 即可。
 - 2. 令 mid=(low+high)/2, mid 即 arr[low..high]中间的位置。
- 1) 如果 arr[low]>arr[mid], 说明断点一定在 arr[low..mid]上, 则令 high=mid, 然后回到步骤 1。
- 2) 如果 arr[mid]>arr[high], 说明断点一定在 arr[mid..high]上, 令 low=mid, 然后回到 步骤 1。
- 3. 如果步骤 1 和步骤 2 的逻辑都没有命中,说明什么呢?步骤 1 没有命中说明 arr[low]>=arr[high],步骤 2 的 1)没有命中说明 arr[low]<=arr[mid],步骤 2 的 2)没有命中说明, arr[mid]<=arr[high]。此时只有一种情况,也就是 arr[low]==arr[mid]==arr[high]。面对这种情况根本无法判断断点的位置在哪里,很多书籍在面对这种情况时都选择直接遍历 arr[low..high]的方法找出断点。但其实还是可以继续为二分创造条件,生成变量 i,初始时令 i=low,开始向右遍历 arr(i++),那么会有以下三种情况:
 - 情况 1: 遍历到某个位置时发现 arr[low]>arr[i], 那么 arr[i]就是断点处的值, 因为 在 arr 中发现的降序必然是断点, 所以直接返回 arr[i]。
 - 情况 2: 遍历到某个位置时发现 arr[low]<arr[i], 说明 arr[i]>arr[mid], 那么说明断点在 arr[i..mid]上。此时又可以开始二分,令 high=mid,重新回到步骤 1。
 - 情况 3:如果 i==mid 都没有出现情况 1 和情况 2,说明从 arr 的 low 位置到 mid 位置,值全部都一样。那么断点只可能在 arr[mid..high]上,所以令 low=mid,进行后续的二分过程,重新回到步骤 1。

全部过程请参看如下代码中的 getMin 方法。

```
public int getMin(int[] arr) {
    int low = 0;
    int high = arr.length - 1;
    int mid = 0;
    while (low < high) {
        if (low == high - 1) {
            break;
        }
        if (arr[low] < arr[high]) {
            return arr[low];
        }
        mid = (low + high) / 2;
        if (arr[low] > arr[mid]) {
            high = mid;
            continue;
        }
}
```

```
if (arr[mid] > arr[high]) {
        low = mid;
        continue;
}
while (low < mid) {
        if (arr[low] == arr[mid]) {
            low++;
        } else if (arr[low] < arr[mid]) {
            return arr[low];
        } else {
            high = mid;
            break;
        }
}
return Math.min(arr[low], arr[high]);</pre>
```

在有序旋转数组中找到一个数

【题目】

有序数组 arr 可能经过一次旋转处理,也可能没有,且 arr 可能存在重复的数。例如,有序数组[1,2,3,4,5,6,7],可以旋转处理成[4,5,6,7,1,2,3]等。给定一个可能旋转过的有序数组 arr,再给定一个数 num,返回 arr 中是否含有 num。

【难度】

尉★★☆☆

【解答】

为了方便描述,我们把没经过旋转前,有序数组 arr 最左边的数,在经过旋转之后所处的位置叫作断点。例如,题目例子里的数组,在旋转后断点在 1 所处的位置,也就是位置 4。如果一个数组没有经过旋转处理,断点在位置 0。

本书提供的方式做到了尽可能多地利用二分查找,但是最差情况下仍无法避免 O(N)的时间复杂度,以下是具体过程:

- 1. 用 low 和 high 变量表示 arr 上的一个范围,每次判断 num 是否在 arr[low..high]上,初始时,low=0, high=arr.length-1,然后进入步骤 2。
 - 2. 如果 low>high, 直接进入步骤 5, 否则令变量 mid=(low+high)/2, 也就是二分的位