```
    }

    public void swap(int[] arr, int index1, int index2) {
            int tmp = arr[index1];
            arr[index1] = arr[index2];
            arr[index2] = tmp;
    }
```

判断一个数是否是回文数

【题目】

定义回文数的概念如下：

- 如果一个非负数左右完全对应，则该数是回文数，例如：121，22 等。
- 如果一个负数的绝对值左右完全对应，也是回文数，例如：-121，-22 等。

给定一个 32 位整数 num，判断 num 是否是回文数。

【难度】

士 ★☆☆☆

【解答】

本题的实现方法当然有很多种，本书介绍一种仅用一个整型变量就可以实现的方法，步骤如下：

1．假设判断的数字为非负数 *n*，先生成变量 help，开始时 help=1。

2．用 help 不停地乘以 10，直到变得与 num 的位数一样。例如：num 等于 123321 时，help 就是 100000。num 如果是 131，help 就是 100，总之，让 help 与 num 的位数一样。

3．那么 num/help 的结果就是最高位的数字，num%10 就是最低位的数字，比较这两个数字，不相同则直接返回 false。相同则令 num=(num%help)/10，即 num 变成除去最高位和最低位两个数字之后的值。令 help/=100，即让 help 变得继续和新的 num 位数一样。

4．如果 num==0，表示所有的数字都已经对应判断完，返回 true，否则重复步骤 3。

上述方法就是让 num 每次剥掉最左和最右两个数，然后逐渐完成所有对应的判断。需要注意的是，如上方法只适用于非负数的判断，如果 *n* 为负数，则先把 *n* 变成其绝对值，然后用上面的方法进行判断。同时还需注意，32 位整数中的最小值为-2147483648，它是转不成相应的绝对值的，可这个数也很明显不是回文数。所以，如果 *n* 为-2147483648，直接

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：