```
        }
        return String.valueOf(chas);
    }
```

判断两个字符串是否互为旋转词

【题目】

如果一个字符串 str，把字符串 str 前面任意的部分挪到后面形成的字符串叫作 str 的旋转词。比如 str="12345"，str 的旋转词有"12345"、"23451"、"34512"、"45123"和"51234"。给定两个字符串 a 和 b，请判断 a 和 b 是否互为旋转词。

【举例】

a="cdab"，b="abcd"，返回 true。

a="1ab2"，b="ab12"，返回 false。

a="2ab1"，b="ab12"，返回 true。

【要求】

如果 a 和 b 长度不一样，那么 a 和 b 必然不互为旋转词，可以直接返回 false。当 a 和 b 长度一样，都为 N 时，要求解法的时间复杂度为 $O(N)$。

【难度】

士 ★☆☆☆

【解答】

本题的解法非常简单，如果 a 和 b 的长度不一样，字符串 a 和 b 不可能互为旋转词。如果 a 和 b 长度一样，先生成一个大字符串 b2，b2 是两个字符串 b 拼在一起的结果，即 String b2 = b + b。然后看 b2 中是否包含字符串 a，如果包含，说明字符串 a 和 b 互为旋转词，否则说明两个字符串不互为旋转词。这是为什么呢？举例说明，假设 a="cdab"，b="abcd"。b2="abcdabcd"，b2[0..3]=="abcd"是 b 的旋转词，b2[1..4]=="bcda"是 b 的旋转词……b2[i..i+3]都是 b 的旋转词，b2[4..7]=="abcd"是 b 的旋转词。由此可见，如果一个字符串 b 长度为 N。在通过 b 生成的 b2 中，任意长度为 N 的子串都是 b 的旋转词，并且 b2 中包含字符串 b 的

所有旋转词。所以这种方法是有效的，请参看如下代码中的 isRotation 方法。

```
public boolean isRotation(String a, String b) {
        if (a == null || b == null || a.length() != b.length()) {
                return false;
        }
        String b2 = b + b;
        return getIndexOf(b2, a) != -1; // getIndexOf -> KMP Algorithm
}
```

isRotation 方法中 getIndexOf 函数的功能是如果 b2 中包含 a，则返回 a 在 b2 中的开始位置，如果不包含 a，则返回-1，即 getIndexOf 是解决匹配问题的函数，如果想让整个过程在 $O(N)$的时间复杂度内完成，那么字符串匹配问题也需要在 $O(N)$的时间复杂度内完成。这正是 KMP 算法做的事情，getIndexOf 函数就是 KMP 算法的实现。若要了解 KMP 算法的过程和实现，请参看本书“KMP 算法”的内容。

将整数字符串转成整数值

【题目】

给定一个字符串 str，如果 str 符合日常书写的整数形式，并且属于 32 位整数的范围，返回 str 所代表的整数值，否则返回 0。

【举例】

str="123"，返回 123。

str="023"，因为"023"不符合日常的书写习惯，所以返回 0。

str="A13"，返回 0。

str="0"，返回 0。

str="2147483647"，返回 2147483647。

str="2147483648"，因为溢出了，所以返回 0。

str="-123"，返回-123。

【难度】

尉 ★★☆☆