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 ★★☆☆

【解答】

解决本题的方法有很多，本书仅提供一种供读者参考。首先检查 str 是否符合日常书写的整数形式，具体判断如下：

1．如果 str 不以“-”开头，也不以数字字符开头，例如，str=="A12"，返回 false。

2．如果 str 以“-”开头。但是 str 的长度为 1，即 str=="-"，返回 false。如果 str 的长度大于 1，但是“-”的后面紧跟着“0”，例如，str=="-0"或"-012"，返回 false。

3．如果 str 以“0”开头，但是 str 的长度大于 1，例如，str=="023"，返回 false。

4．如果经过步骤 1~步骤 3 都没有返回，接下来检查 str[1..N-1]是否都是数字字符，如果有一个不是数字字符，返回 false。如果都是数字字符，说明 str 符合日常书写，返回 true。

具体检查过程请参看如下代码中的 isValid 方法。

```
public boolean isValid(char[] chas) {
        if (chas[0] != '-' && (chas[0] < '0' || chas[0] > '9')) {
                return false;
        }
        if (chas[0] == '-' && (chas.length == 1 || chas[1] == '0')) {
                return false;
        }
        if (chas[0] == '0' && chas.length > 1) {
                return false;
        }
        for (int i = 1; i < chas.length; i++) {
                if (chas[i] < '0' || chas[i] > '9') {
                        return false;
                }
        }
        return true;
}
```

如果 str 不符合日常书写的整数形式，根据题目要求，直接返回 0 即可。如果符合，则进行如下转换过程：

1．生成 4 个变量。布尔型常量 posi，表示转换的结果是负数还是非负数，这完全由 str 开头的字符决定，如果以“-”开头，那么转换的结果一定是负数，则 posi 为 false，否则 posi 为 true。整型常量 minq，minq 等于 Integer.MIN_VALUE/10，即 32 位整数最小值除以 10 得到的商，其意义稍后说明。整型常量 minr，minr 等于 Integer.MIN_VALUE%10，即 32 位整数最小值除以 10 得到的余数，其意义稍后说明。整型变量 res，转换的结果，初始时 res=0。

2．32 位整数的最小值为-2147483648，32 位整数的最大值为 2147483647。可以看出，

最小值的绝对值比最大值的绝对值大 1，所以转换过程中的绝对值一律以负数的形式出现，然后根据 posi 决定最后返回什么。比如 str="123"，转换完成后的结果是-123，posi=true，所以最后返回 123。再如 str="-123"，转换完成后的结果是-123，posi=false，所以最后返回 -123。比如 str="-2147483648"，转换完成后的结果是-2147483648，posi=false，所以最后返回-2147483648。比如 str="2147483648"，转换完成后的结果是-2147483648，posi=true，此时发现-2147483648 变成 2147483648 会产生溢出，所以返回 0。也就是说，既然负数比正数拥有更大的绝对值范围，那么转换过程中一律以负数的形式记录绝对值，最后再决定返回的数到底是什么。

3．如果 str 以'-'开头，从 str[1]开始从左往右遍历 str，否则从 str[0]开始从左往右遍历 str。举例说明转换过程，比如 str="123"，遍历到'1'时，res=res*10+(-1)==-1，遍历到'2'时，res=res*10+(-2)==-12，遍历到'3'时，res=res*10+(-3)==-123。比如 str="-123"，字符'-'跳过，从字符'1'开始遍历，res=res*10+(-1)==-1，遍历到'2'时，res=res*10+(-2)==-12，遍历到'3'时，res=res*10+(-3)==-123。遍历的过程中如何判断 res 已经溢出了？假设当前字符为 a，那么'0'-a 就是当前字符所代表的数字的负数形式，记为 cur。如果在 res 加上 cur 之前，发现 res 已经小于 minq，那么当 res 加上 cur 之后一定会溢出，比如 str="3333333333"，遍历完倒数第二个字符后，res==-333333333 < minq==-214748364，所以当遍历到最后一个字符时，res*10 肯定会产生溢出。如果在 res 加上 cur 之前，发现 res 等于 minq，但又发现 cur 小于 minr，那么当 res 加上 cur 之后一定会溢出，比如 str="2147483649"，遍历完倒数第二个字符后，res==-214748364 == minq，当遍历到最后一个字符时发现有 res==minq，同时也发现 cur==-9 < minr==-8，那么当 res 加上 cur 之后一定会溢出。出现任何一种溢出情况时，直接返回 0。

4．遍历后得到的 res 根据 posi 的符号决定返回值。如果 posi 为 true，说明结果应该返回正，否则说明应该返回负。如果 res 正好是 32 位整数的最小值，同时又有 posi 为 true，说明溢出，直接返回 0。

全部过程请参看如下代码中的 convert 方法。

```
public int convert(String str) {
        if (str == null || str.equals("")) {
                return 0; // 不能转
        }
        char[] chas = str.toCharArray();
        if (!isValid(chas)) {
                return 0; // 不能转
        }
        boolean posi = chas[0] == '-' ? false : true;
        int minq = Integer.MIN_VALUE / 10;
        int minr = Integer.MIN_VALUE % 10;
```

```
        int res = 0;
        int cur = 0;
        for (int i = posi ? 0 : 1; i < chas.length; i++) {
                cur = '0' - chas[i];
                if ((res < minq) || (res == minq && cur < minr)) {
                        return 0; // 不能转
                }
                res = res * 10 + cur;
        }
        if (posi && res == Integer.MIN_VALUE) {
                return 0; // 不能转
        }
        return posi ? -res : res;
}
```

替换字符串中连续出现的指定字符串

【题目】

给定三个字符串 str、from 和 to，把 str 中所有 from 的子串全部替换成 to 字符串，对连续出现 from 的部分要求只替换成一个 to 字符串，返回最终的结果字符串。

【举例】

str="123abc"，from="abc"，to="4567"，返回"1234567"。

str="123"，from="abc"，to="456"，返回"123"。

str="123abcabc"，from="abc"，to="X"，返回"123X"。

【难度】

士 ★☆☆☆

【解答】

解决本题的方法有很多。本书仅提供一种供读者参考。如果把 str 看作字符类型的数组，首先把 str 中 from 部分所有位置的字符编码设为 0（即空字符），比如，str="12abcabca4"，from="abc"，处理后 str 为['1','2',0,0,0,0,0,0,'a','4']。具体过程如下：

1. 生成整型变量 match，表示目前匹配到 from 字符串的什么位置，初始时，match=0。

2. 从左到右遍历 str 中的每个字符，假设当前遍历到 str[i]。