下来，比如遍历到字符'a'，其编码值为 97，则令 map[97]--，如果减少之后的值小于 0，直接返回 false。如果遍历完 str2，map 中的值也没出现负值，则返回 true。

具体请参看如下代码中的 isDeformation 方法。

```
public boolean isDeformation(String str1, String str2) {
        if (str1 == null || str2 == null || str1.length() != str2.length()) {
                return false;
        }
        char[] chas1 = str1.toCharArray();
        char[] chas2 = str2.toCharArray();
        int[] map = new int[256];
        for (int i = 0; i < chas1.length; i++) {
                map[chas1[i]]++;
        }
        for (int i = 0; i < chas2.length; i++) {
                if (map[chas2[i]]-- == 0) {
                        return false;
                }
        }
        return true;
}
```

如果字符的类型很多，可以用哈希表代替长度为 256 的整型数组，但整体过程不变。如果字符的种类为 M，str1 和 str2 的长度为 N，那么该方法的时间复杂度为 $O(N)$，额外空间复杂度为 $O(M)$。

字符串中数字子串的求和

【题目】

给定一个字符串 str，求其中全部数字串所代表的数字之和。

【要求】

1．忽略小数点字符，例如"A1.3"，其中包含两个数字 1 和 3。

2．如果紧贴数字子串的左侧出现字符"-"，当连续出现的数量为奇数时，则数字视为负，连续出现的数量为偶数时，则数字视为正。例如，"A-1BC--12"，其中包含数字为-1 和 12。

【举例】

str="A1CD2E33"，返回 36。

str="A-1B--2C--D6E"，返回 7。

【难度】

士 ★☆☆☆

【解答】

解决本题能做到时间复杂度为 $O(N)$、额外空间复杂度为 $O(1)$的方法有很多。本书仅提供一种供读者参考。解法的关键是如何在从左到右遍历 str 时，准确收集每个数字并累加起来。具体过程如下：

1．生成三个变量。整型变量 res，表示目前的累加和；整型变量 num，表示当前收集到的数字；布尔型变量 posi，表示如果把 num 累加到 res 里，num 是正还是负。初始时，res=0，num=0，posi=true。

2．从左到右遍历 str，假设遍历到字符 cha，根据具体的 cha 有不同的处理。

3．如果 cha 是'0'~'9'，cha-'0'的值记为 cur，假设之前收集的数字为 num，此时举例说明。比如 str="123"，初始时 num=0，posi=true。当 cha=='1'时，num 变成 1；cha=='2'时，num 变成 12；cha=='3'时，num 变成 123。再如 str="-123"，初始时 num=0，posi=true。当 cha=='-'时，posi 变成 false，cha 不是'0'~'9'的情况接下来会说明，读者可以先认为在收集数字时 posi 的符号一定是正确的。cha=='1'时，num 变成-1，cha=='2'时，num 变成-12。cha=='3'时，num 变成-123。总之， num = num * 10 + (posi ? cur : -cur)。

4．如果 cha 不是'0'~'9'，此时不管 cha 具体是什么，都是累加时，令 res+=num，然后令 num=0，累加完 num 当然要清零。累加完成后，再看 cha 具体的情况。如果 cha 不是字符'-'，令 posi=true，即如果 cha 既不是数字字符，也不是'-'字符，posi 都变为 true。如果 cha 是字符'-'，此时看 cha 的前一个字符，如果前一个字符也是'-'字符，则 posi 改变符号，即 posi=!posi；否则令 posi=false。

5．既然我们把累加的时机放在了 cha 不是数字字符的时候，那么如果 str 是以数字字符结尾的，会出现最后一个数字没有累加的情况。所以遍历完成后，令 res+=num，防止最后的数字累加不上的情况发生。

6．最后返回 res。

具体实现请参看如下代码中的 numSum 方法。

```
public int numSum(String str) {
        if (str == null) {
```

```
                return 0;
        }
        char[] charArr = str.toCharArray();
        int res = 0;
        int num = 0;
        boolean posi = true;
        int cur = 0;
        for (int i = 0; i < charArr.length; i++) {
                cur = charArr[i] - '0';
                if (cur < 0 || cur > 9) {
                        res += num;
                        num = 0;
                        if (charArr[i] == '-') {
                                if (i - 1 > -1 && charArr[i - 1] == '-') {
                                        posi = !posi;
                                } else {
                                        posi = false;
                                }
                        } else {
                                posi = true;
                        }
                } else {
                        num = num * 10 + (posi ? cur : -cur);
                }
        }
        res += num;
        return res;
}
```

去掉字符串中连续出现 k 个 0 的子串

【题目】

给定一个字符串 str 和一个整数 k，如果 str 中正好有连续的 k 个'0'字符出现时，把 k 个连续的'0'字符去除，返回处理后的字符串。

【举例】

str="A00B"，k=2，返回"A00B"。

str="A0000B000"，k=3，返回"A0000B"。

【难度】

士 ★☆☆☆