公式字符串求值

【题目】

给定一个字符串 str，str 表示一个公式，公式里可能有整数、加减乘除符号和左右括号，返回公式的计算结果。

【举例】

str="48*((70-65)-43)+8*1"，返回-1816。

str="3+1*4"，返回 7。

str="3+(1*4)"，返回 7。

【说明】

1．可以认为给定的字符串一定是正确的公式，即不需要对 str 做公式有效性检查。

2．如果是负数，就需要用括号括起来，比如"4*(-3)"。但如果负数作为公式的开头或括号部分的开头，则可以没有括号，比如"-3*4"和"(-3*4)"都是合法的。

3．不用考虑计算过程中会发生溢出的情况。

【难度】

校 ★★★☆

【解答】

本题考查面试者设计程序和代码实现的能力，实现方式有很多，本书提供一种方法供读者参考。假设 value 方法是一个递归过程，具体解释如下。

从左到右遍历 str，开始遍历或者遇到字符'('时，就进行递归过程。当发现 str 遍历完，或者遇到字符')'时，递归过程就结束。比如"3*(4+5)+7"，一开始遍历就进入递归过程 value(str,0)，在递归过程 value(str,0)中继续遍历 str，当遇到字符'('时，递归过程 value(str,0)又重复调用递归过程 value(str,3)。然后在递归过程 value(str,3)中继续遍历 str，当遇到字符')'时，递归过程 value(str,3)结束，并向递归过程 value(str,0)返回两个结果，第一结果是 value(str,3)遍历过的公式字符子串的结果，即"4+5"==9，第二个结果是 value(str,3)遍历到的

位置，即字符")"的位置==6。递归过程 value(str,0)收到这两个结果后，既可知道交给value(str,3)过程处理的字符串结果是多少("(4+5)"的结果是 9)，又可知道自己下一步该从什么位置继续遍历(该从位置 6 的下一个位置（即位置 7）继续遍历)。总之，value 方法的第二个参数代表递归过程是从什么位置开始的，返回的结果是一个长度为 2 的数组，记为 res。res[0]表示这个递归过程计算的结果，res[1]表示这个递归过程遍历到 str 的什么位置。

既然在递归过程中遇到'('就交给下一层的递归过程处理，自己只用接收'('和')'之间的公式字符子串的结果，所以对所有的递归过程来说，可以看作计算的公式都是不含有'('和')'字符的。比如，对递归过程 value(str,0)来说，实际上计算的公式是"3*9+7"，"(4+5)"的部分交给递归过程 value(str,3)处理，拿到结果 9 之后，再从字符'+'继续。所以，只要想清楚如何计算一个不含有'('和')'的公式字符串，整个实现就完成了。

全部过程请参看如下代码中的 getValue 方法。

```
public int getValue(String exp) {
        return value(exp.toCharArray(), 0)[0];
}

public int[] value(char[] chars, int i) {
        Deque<String> deq = new LinkedList<String>();
        int pre = 0;
        int[] bra = null;
        while (i < chars.length && chars[i] != ')') {
                if (chars[i] >= '0' && chars[i] <= '9') {
                        pre = pre * 10 + chars[i++] - '0';
                } else if (chars[i] != '(') {
                        addNum(deq, pre);
                        deq.addLast(String.valueOf(chars[i++]));
                        pre = 0;
                } else {
                        bra = value(chars, i + 1);
                        pre = bra[0];
                        i = bra[1] + 1;
                }
        }
        addNum(deq, pre);
        return new int[] { getNum(deq), i };
}

public void addNum(Deque<String> deq, int num) {
        if (!deq.isEmpty()) {
                int cur = 0;
                String top = deq.pollLast();
                if (top.equals("+") || top.equals("-")) {
                        deq.addLast(top);
                } else {
                        cur = Integer.valueOf(deq.pollLast());
```

(用递归处理

```
                        num = top.equals("*") ? (cur * num) : (cur / num);
                }
        }
        deq.addLast(String.valueOf(num));
}

public int getNum(Deque<String> deq) {
        int res = 0;
        boolean add = true;
        String cur = null;
        int num = 0;
        while (!deq.isEmpty()) {
                cur = deq.pollFirst();
                if (cur.equals("+")) {
                        add = true;
                } else if (cur.equals("-")) {
                        add = false;
                } else {
                        num = Integer.valueOf(cur);
                        res += add ? num : (-num);
                }
        }
        return res;
}
```

0 左边必有 1 的二进制字符串数量

【题目】

给定一个整数 N，求由"0"字符与"1"字符组成的长度为 N 的所有字符串中，满足"0"字符的左边必有"1"字符的字符串数量。

【举例】

N=1。只由"0"与"1"组成，长度为 1 的所有字符串："0"、"1"。只有字符串"1"满足要求，所以返回 1。

N=2。只由"0"与"1"组成，长度为 2 的所有字符串为："00"、"01"、"10"、"11"。只有字符串"10"和"11"满足要求，所以返回 2。

N=3。只由"0"与"1"组成，长度为 3 的所有字符串为："000"、"001"、"010"、"011"、"100"、"101"、"110"、"111"。字符串"101"、"110"、"111"满足要求，所以返回 3。