```
                chasl++;
            }
            while (chas[chasr] != lps[lpsr]) {
                chasr--;
            }
            set(res, resl, resr, chas, tmpl, chasl, chasr, tmpr);
            resl += chasl - tmpl + tmpr - chasr;
            resr -= chasl - tmpl + tmpr - chasr;
            res[resl++] = chas[chasl++];
            res[resr--] = chas[chasr--];
            lpsl++;
            lpsr--;
        }
        return String.valueOf(res);
    }

    public void set(char[] res, int resl, int resr, char[] chas, int ls,
            int le, int rs, int re) {
        for (int i = ls; i < le; i++) {
            res[resl++] = chas[i];
            res[resr--] = chas[i];
        }
        for (int i = re; i > rs; i--) {
            res[resl++] = chas[i];
            res[resr--] = chas[i];
        }
    }
```

括号字符串的有效性和最长有效长度

【题目】

给定一个字符串 str，判断是不是整体有效的括号字符串。

【举例】

str="()"，返回 true；str="(()())"，返回 true；str="(())"，返回 true。

str="())"。返回 false；str="()("，返回 false；str="()a()"，返回 false。

【补充题目】

给定一个括号字符串 str，返回最长的有效括号子串。

【举例】

str="(()())"，返回 6；str="())"，返回 2；str=")()()()("，返回 4。

【难度】

原问题 士 ★☆☆☆

补充问题 尉 ★★☆☆

【解答】

原问题。判断过程如下：

1．从左到右遍历字符串 str，判断每一个字符是不是'('或')'，如果不是，就直接返回 false。

2．遍历到每一个字符时，都检查到目前为止'('和')'的数量，如果')'更多，则直接返回 false。

3．遍历后检查'('和')'的数量，如果一样多，则返回 true，否则返回 false。

具体过程参看如下代码中的 isValid 方法。

```
public boolean isValid(String str) {
        if (str == null || str.equals("")) {
                return false;
        }
        char[] chas = str.toCharArray();
        int status = 0;
        for (int i = 0; i < chas.length; i++) {
                if (chas[i] != ')' && chas[i] != '(') {
                        return false;
                }
                if (chas[i] == ')' && --status < 0) {
                        return false;
                }
                if (chas[i] == '(') {
                        status++;
                }
        }
        return status == 0;
}
```

补充问题。用动态规划求解，可以做到时间复杂度为 $O(N)$，额外空间复杂度为 $O(N)$。首先生成长度和 str 字符串一样的数组 dp[]，dp[i]值的含义为 str[0..i]中必须以字符 str[i]结尾的最长的有效括号子串长度。那么 dp[i]值可以按如下方式求解：

1．dp[0]=0。只含有一个字符肯定不是有效括号字符串，长度自然是 0。

2．从左到右依次遍历 str[1..N-1]的每个字符，假设遍历到 str[i]。

3．如果 str[i]=='('，有效括号字符串必然是以')'结尾，而不是以'('结尾，所以 dp[i] = 0。

4．如果 str[i]==')'，那么以 str[i]结尾的最长有效括号子串可能存在。dp[i-1]的值代表必须以 str[i-1]结尾的最长有效括号子串的长度，所以如果 i-dp[i-1]-1 位置上的字符是'('，就能与当前位置的 str[i]字符再配出一对有效括号。比如"(()())"，假设遍历到最后一个字符')'，必须以倒数第二个字符结尾的最长有效括号子串是"()()"，找到这个子串之前的字符，即 i-dp[i-1]-1 位置的字符，发现是'('，所以它可以和最后一个字符再配出一对有效括号。如果该情况发生，dp[i]的值起码是 dp[i-1]+2，但还有一部分长度容易被人忽略。比如，"()(())"，假设遍历到最后一个字符')'，通过上面的过程找到的必须以最后字符结尾的最长有效括号子串起码是"(())"，但是前面还有一段"()"，可以和"(())"结合在一起构成更大的有效括号子串。也就是说，str[i-dp[i-1]-1]和 str[i]配成了一对，这时还应该把 dp[i-dp[i-1]-2]的值加到 dp[i]中，这么做表示把 str[i-dp[i-1]-2]结尾的最长有效括号子串接到前面，才能得到以当前字符结尾的最长有效括号子串。

5．dp[0..N-1]中的最大值就是最终的结果。

具体过程请参看如下代码中的 maxLength 方法。

```
public int maxLength(String str) {
        if (str == null || str.equals("")) {
                return 0;
        }
        char[] chas = str.toCharArray();
        int[] dp = new int[chas.length];
        int pre = 0;
        int res = 0;
        for (int i = 1; i < chas.length; i++) {
                if (chas[i] == ')') {
                    pre = i - dp[i - 1] - 1;
                    if (pre >= 0 && chas[pre] == '(') {
                        dp[i] = dp[i - 1] + 2 + (pre > 0 ? dp[pre - 1] : 0);
                    }
                }
                res = Math.max(res, dp[i]);
        }
        return res;
}
```