```
                mid = tmp;
                i--;
            }
            return res;
        }
```

最长公共子序列问题

【题目】

给定两个字符串 str1 和 str2，返回两个字符串的最长公共子序列。

【举例】

str1="1A2C3D4B56"，str2="B1D23CA45B6A"。

"123456"或者"12C4B6"都是最长公共子序列，返回哪一个都行。

【难度】

尉 ★★☆☆

【解答】

本题是非常经典的动态规划问题，先来介绍求解动态规划表的过程。如果 str1 的长度为 M，str2 的长度为 N，生成大小为 $M \times N$ 的矩阵 dp，行数为 M，列数为 N。dp[i][j]的含义是 str1[0..i]与 str2[0..j]的最长公共子序列的长度。从左到右，再从上到下计算矩阵 dp。

1. 矩阵 dp 第一列即 dp[0..M-1][0]，dp[i][0]的含义是 str1[0..i]与 str2[0]的最长公共子序列长度。str2[0]只有一个字符，所以 dp[i][0]最大为 1。如果 str1[i]==str2[0]，令 dp[i][0]=1，一旦 dp[i][0]被设置为 1，之后的 dp[i+1..M-1][0]也都为 1。比如，str1[0..M-1]="ABCDE"，str2[0]="B"。str1[0]为"A"，与 str2[0]不相等，所以 dp[0][0]=0。str1[1]为"B"，与 str2[0]相等，所以 str1[0..1]与 str2[0]的最长公共子序列为"B"，令 dp[1][0]=1。之后的 dp[2..4][0]肯定都是 1，因为 str[0..2]、str[0..3]和 str[0..4]与 str2[0]的最长公共子序列肯定有"B"。

2. 矩阵 dp 第一行即 dp[0][0..N-1]与步骤 1 同理，如果 str1[0]==str2[j]，则令 dp[0][j]=1，一旦 dp[0][j]被设置为 1，之后的 dp[0][j+1..N-1]也都为 1。

3. 对其他位置(i,j)，dp[i][j]的值只可能来自以下三种情况：

- 可能是 dp[i-1][j]，代表 str1[0..i-1]与 str2[0..j]的最长公共子序列长度。比如，

str1="A1BC2", str2="AB34C"。str1[0..3]（即"A1BC"）与 str2[0..4]（即"AB34C"）的最长公共子序列为"ABC", 即 dp[3][4]为 3。str1[0..4]（即"A1BC2"）与 str2[0..4]（即"AB34C"）的最长公共子序列也是"ABC", 所以 dp[4][4]也为 3。

- 可能是 dp[i][j-1], 代表 str1[0..i]与 str2[0..j-1]的最长公共子序列长度。比如, str1="A1B2C", str2="AB3C4"。str1[0..4]（即"A1B2C"）与 str2[0..3]（即"AB3C"）的最长公共子序列为"ABC", 即 dp[4][3]为 3。str1[0..4]（即"A1B2C"）与 str2[0..4]（即"AB3C4"）的最长公共子序列也是"ABC", 所以 dp[4][4]也为 3。
- 如果 str1[i]==str2[j], 还可能是 dp[i-1][j-1]+1。比如 str1="ABCD", str2="ABCD"。str1[0..2]（即"ABC"）与 str2[0..2]（即"ABC"）的最长公共子序列为"ABC", 即 dp[2][2]为 3。因为 str1[3]==str2[3]=="D", 所以 str1[0..3]与 str2[0..3]的最长公共子序列是"ABCD"。

这三个可能的值中，选最大的作为 dp[i][j]的值。具体过程请参看如下代码中的 getdp 方法。

```
public int[][] getdp(char[] str1, char[] str2) {
        int[][] dp = new int[str1.length][str2.length];
        dp[0][0] = str1[0] == str2[0] ? 1 : 0;
        for (int i = 1; i < str1.length; i++) {
            dp[i][0] = Math.max(dp[i - 1][0], str1[i] == str2[0] ? 1 : 0);
        }
        for (int j = 1; j < str2.length; j++) {
            dp[0][j] = Math.max(dp[0][j - 1], str1[0] == str2[j] ? 1 : 0);
        }
        for (int i = 1; i < str1.length; i++) {
            for (int j = 1; j < str2.length; j++) {
                dp[i][j] = Math.max(dp[i - 1][j], dp[i][j - 1]);
                if (str1[i] == str2[j]) {
                    dp[i][j] = Math.max(dp[i][j], dp[i - 1][j - 1] + 1);
                }
            }
        }
        return dp;
}
```

dp 矩阵中最右下角的值代表 str1 整体和 str2 整体的最长公共子序列的长度。通过整个 dp 矩阵的状态，可以得到最长公共子序列。具体方法如下：

1. 从矩阵的右下角开始，有三种移动方式：向上、向左、向左上。假设移动的过程中，i 表示此时的行数，j 表示此时的列数，同时用一个变量 res 来表示最长公共子序列。

2. 如果 dp[i][j]大于 dp[i-1][j]和 dp[i][j-1]，说明之前在计算 dp[i][j]的时候，一定是选

择了决策 dp[i-1][j-1]+1，可以确定 str1[i]等于 str2[j]，并且这个字符一定属于最长公共子序列，把这个字符放进 res，然后向左上方移动。

3．如果 dp[i][j]等于 dp[i-1][j]，说明之前在计算 dp[i][j]的时候，dp[i-1][j-1]+1 这个决策不是必须选择的决策，向上方移动即可。

4．如果 dp[i][j]等于 dp[i][j-1]，与步骤 3 同理，向左方移动。

5．如果 dp[i][j]同时等于 dp[i-1][j]和 dp[i][j-1]，向上还是向下无所谓，选择其中一个即可，反正不会错过必须选择的字符。

也就是说，通过 dp 求解最长公共子序列的过程就是还原出当时如何求解 dp 的过程，来自哪个策略就朝哪个方向移动。全部过程请参看如下代码中的 lcse 方法。

```java
public String lcse(String str1, String str2) {
    if (str1 == null || str2 == null || str1.equals("") || str2.equals("")) {
        return "";
    }
    char[] chs1 = str1.toCharArray();
    char[] chs2 = str2.toCharArray();
    int[][] dp = getdp(chs1, chs2);
    int m = chs1.length - 1;
    int n = chs2.length - 1;
    char[] res = new char[dp[m][n]];
    int index = res.length - 1;
    while (index >= 0) {
        if (n > 0 && dp[m][n] == dp[m][n - 1]) {
            n--;
        } else if (m > 0 && dp[m][n] == dp[m - 1][n]) {
            m--;
        } else {
            res[index--] = chs1[m];
            m--;
            n--;
        }
    }
    return String.valueOf(res);
}
```

计算 dp 矩阵中的某个位置就是简单比较相关的 3 个位置的值而已，所以时间复杂度为 $O(1)$，动态规划表 dp 的大小为 $M\times N$，所以计算 dp 矩阵的时间复杂度为 $O(M\times N)$。通过 dp 得到最长公共子序列的过程为 $O(M+N)$，因为向左最多移动 N 个位置，向上最多移动 M 个位置，所以总的时间复杂度为 $O(M\times N)$，额外空间复杂度为 $O(M\times N)$。如果题目不要求返回最长公共子序列，只想求最长公共子序列的长度，那么可以用空间压缩的方法将额外空间复杂度减小为 $O(\min\{M,N\})$，有兴趣的读者请阅读本书“矩阵的最小路径和”问题，这里

不再详述。

最长公共子串问题

【题目】

给定两个字符串 str1 和 str2，返回两个字符串的最长公共子串。

【举例】

str1="1AB2345CD"，str2="12345EF"，返回"2345"。

【要求】

如果 str1 长度为 M，str2 长度为 N，实现时间复杂度为 $O(M\times N)$，额外空间复杂度为 $O(1)$的方法。

【难度】

校 ★★★☆

【解答】

经典动态规划的方法可以做到时间复杂度为 $O(M\times N)$，额外空间复杂度为 $O(M\times N)$，经过优化之后的实现可以把额外空间复杂度从 $O(M\times N)$降至 $O(1)$，我们先来介绍经典方法。

首先需要生成动态规划表。生成大小为 $M\times N$ 的矩阵 dp，行数为 M，列数为 N。dp[i][j]的含义是，在必须把 str1[i]和 str2[j]当作公共子串最后一个字符的情况下，公共子串最长能有多长。比如，str1="A1234B"，str2="CD1234"，dp[3][4]的含义是在必须把 str1[3]（即'3'）和 str2[4]（即'3'）当作公共子串最后一个字符的情况下，公共子串最长能有多长。这种情况下的最长公共子串为"123"，所以 dp[3][4]为 3。再如，str1="A12E4B"，str2="CD12F4"，dp[3][4]的含义是在必须把 str1[3]（即'E'）和 str2[4]（即'F'）当作公共子串最后一个字符的情况下，公共子串最长能有多长。这种情况下根本不能构成公共子串，所以 dp[3][4]为 0。介绍了 dp[i][j]的意义后，接下来介绍 dp[i][j]怎么求。具体过程如下：

1．矩阵 dp 第一列即 dp[0..M-1][0]。对某一个位置$(i,0)$来说，如果 str1[i]==str2[0]，令 dp[i][0]=1，否则令 dp[i][0]=0。比如 str1="ABAC"，str2[0]="A"。dp 矩阵第一列上的值依次