最长公共子串(动态规划)

C语言与CPP编程 今天

> 来源：https://www.cnblogs.com/fanguangdexiaoyuer/p/11281179.html

1 描述

有两个字符串（可能包含空格）,请找出其中最长的公共连续子串,输出其长度。(长度在1000以内)

例如：

输入：abcde bcd

输出：3

2 解析

1、把两个字符串分别以行和列组成一个二维矩阵。

2、比较二维矩阵中每个点对应行列字符中否相等，相等的话值设置为1，否则设置为0。

3、通过查找出值为1的最长对角线就能找到最长公共子串。

比如：str=acbcbcef，str2=abcbced，则str和str2的最长公共子串为bcbce，最长公共子串长度为

5。

针对于上面的两个字符串我们可以得到的二维矩阵如下：

	a	b	c	b	c	e	d
a	1	0	0	0	0	0	0
c	0	0	1	0	1	0	0
b	0	1	0	1	0	0	0
c	0	0	1	0	1	0	0
b	0	1	0	1	0	0	0
c	0	0	1	0	1	0	0
e	0	0	0	0	0	1	0
f	0	0	0	0	0	0	0

从上图可以看到，str1和str2共有5个公共子串，但最长的公共子串长度为5。

为了进一步优化算法的效率，我们可以再计算某个二维矩阵的值的时候顺便计算出来当前最长的公共子串的长度，即某个二维矩阵元素的值由record[i][j]=1演变为record[i][j]=1 +record[i-1][j-1]，这样就避免了后续查找对角线长度的操作了。修改后的二维矩阵如下：

	a	b	c	b	c	e	d
a	1	0	0	0	0	0	0
c	0	0	1	0	1	0	0
b	0	1	0	2	0	0	0
c	0	0	2	0	3	0	0
b	0	1	0	3	0	0	0
c	0	0	2	0	4	0	0
e	0	0	0	0	0	5	0
f	0	0	0	0	0	0	0

递推公式为：

当A[i] != B[j]， dp[i][j] = 0

当A[i] == B[j]，

若i = 0 || j == 0， dp[i][j] = 1

否则 dp[i][j] = dp[i - 1][j - 1] + 1

3 代码

暴力法

```
public int getLCS(String s, String s2)
{
        if (s == null || t == null)
        {
            return 0;
        }
        int l1 = s.length();
        int l2 = t.length();
        int res = 0;
        for (int i = 0; i < l1; i++)
        {
            for (int j = 0; j < l2; j++)
            {
                int m = i;
                int k = j;
                int len = 0;
                while (m < l1 && k < l2 && s.charAt(m) == t.charAt(k)) {
                    len++;
                    m++;
                    k++;
                }
                res = Math.max(res, len);
            }
        }
        return res;
}
```

动态规划

```
public int getLCS(String s, String t)
{
        if (s == null || t == null)
        {
            return 0;
        }
        int result = 0;
        int sLength = s.length();
        int tLength = t.length();
        int[][] dp = new int[sLength][tLength];
        for (int i = 0; i < sLength; i++) {
            for (int k = 0; k < tLength; k++) {
                if (s.charAt(i) == t.charAt(k)) {
                    if (i == 0 || k == 0) {
                        dp[i][k] = 1;
                    } else {
                        dp[i][k] = dp[i - 1][k - 1] + 1;
                    }
                    result = Math.max(dp[i][k], result);
                } else {
                    dp[i][k] = 0;
                }
            }
        }
        return result;
}
```

简化一下递推公式：

当A[i] != B[j]，dp[i][j] = 0

否则 dp[i][j] = dp[i - 1][j - 1] + 1

全部都归结为一个公式即可，二维数组默认值为0

```
public int getLCS(String s, String t)
{
        if (s == null || t == null)
        {
            return 0;
        }
        int result = 0;
        int sLength = s.length();
        int tLength = t.length();
        int[][] dp = new int[sLength + 1][tLength + 1];
        for (int i = 1; i <= sLength; i++) {
            for (int k = 1; k <= tLength; k++) {
                if (s.charAt(i - 1) == t.charAt(k - 1)) {
                    dp[i][k] = dp[i - 1][k - 1] + 1;
                    result = Math.max(dp[i][k], result);
                }
            }
        }
        return result;
}
```

点【在看】是最大的支持