最长公共子串(Longest Common Substring)

与

最长公共子序列(Longest Common Subsequence)

在了解最长公共子序列之前，我们先了解一下最长公共子串：

给定两个字符串，求出它们之间最长的相同子字符串的长度。

举个例子，假如有两个字符串：ABCD 和 ABDBCDF

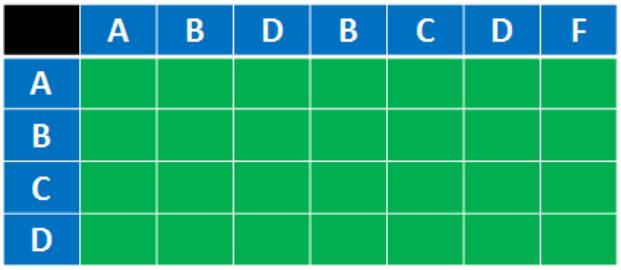
他们的公共子串分别有：A, B, C, D, AB, BC, CD, BCD

其中最长的就是 BCD 了，所以 BCD 就是它们两个的最长公共子串，注意最长公共子串可能不止一个哦...

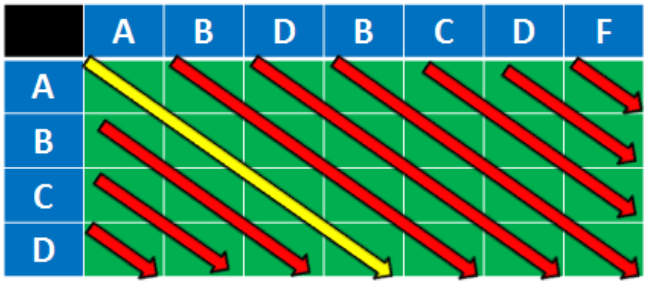
接下来先讲一种很简单很好理解的方法：

用例子来说明一切，还是上面的 ABCD 和 ABDBCDF

这两个字符串的长度分别是 4 和 7，我们建立一个 4*7 的二维数组，我们这里用行表示前面的 ABCD，列表示后面的 ABDBCDF：



然后，我们看每一条对角线：



我们先挑上面这条黄色的来演示一下：

先判断最上面那个格子，因为行列都是 A，所以格子应该是 1：

接下来，看对角线上的第二个格子也就是上面标红的格子：

因为行列都是 B，所以这个格子等于它的左上方那个格子的数字加 1，所以是 2：

	A	B	D	B	C	D	F
A	1						
B		2					
C							
D							

接下来继续看下一个格子，因为行是 C，列是 D，字符不相同，所以这个格子置零，看到这里我相信你应该知道怎么求解这个表格了，就不赘述了，直接看结果：

	A	B	D	B	C	D	F
A	1	0	0	0	0	0	0
B	0	2	0	1	0	0	0
C	0	0	0	0	2	0	0
D	0	0	1	0	0	3	0

这个结果有什么用？

如果你在刚刚求解表格的时候有稍微想那么一下，那么或许你会发现，求解的过程就是在求它的所有公共子串...

你看这里出来的非零数字，出那几个连续的就知道它在每个对角线上面的公共子串了：AB, D, BCD，咦？怎么只有三个？前面不是说有好多个么？

前面说有：A, B, C, D, AB, BC, CD, BCD

这是不是都是上面三个子串的子集？

那么最长的公共子串在哪里？你找表格里面最大的数字不就好了么...找到它是不是就可以找到完整的子串了？

额，有些人可能要说师兄这也太简单了吧...是啊...好像是很简单的样子...可是你不要先把表格求出来再来找最大值...你一边求就一边记录表格的最大值在哪里...这样就不用求完表格再回头来扫描整个表格找最大值了...

其实还有可以优化的...你可以想想，有没有必要把整个表格都存储下来？

接下来我们来了解一下什么是最长公共子序列(Longest Common Subsequence)，我们常说的 LCS 就是这个最长公共子序列：

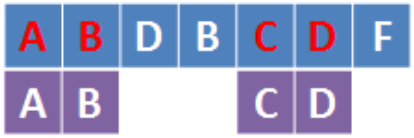
先看看维基百科是怎么定义的：

一个数列 S，如果分别是两个或多个已知数列的子序列，且是所有符合此条件序列中最长的，则 S 称为已知序列的最长公共子序列。

好像跟前面的最长公共子串差不多哦？

这两个还是有很大区别的，子序列是可以不连续的，而子串必须是连续的，还是刚刚那个例子：ABCD 和 ABDBCDF

它的最长公共子序列其实是 ABCD，怎么回事呢？把它画出来就清楚了：



子序列并不要求这几个字符是连续出现的，只要相对顺序不变就可以了。也就是说，A 必须在 B 前面，B 在 C 前面，C 在 D 前面就好了，并不要求 B 后面紧跟着就是 C。

那么这个问题怎么求解呢？

我们还是需要刚刚那张表，不过这张表的意义跟刚刚不一样：

第 i 行第 j 列的格子里面的数字表示 ABCD 的前 i 个字符与 ABDBCDF 的前 j 个字符之间的最长公共子序列长度。

	A	B	D	B	C	D	F
A	1	1	1				
B	1						
C							
D							

这里我们对比一下可以知道，红色框跟紫色框的数值都跟前面不一样了。

紫色框是第 2 行第 1 列，表示 ABCD 的前 2 个字符跟 ABDBCDF 的前 1 个字符之间的最长公共子序列长度，也就是 AB 与 A 之间的最长公共子序列长度，自然就是 1 了...

同理，红色框的 1 代表 A 与 ABD 之间的最长公共子序列长度。

接下来还是直接讲怎么求解这张表，边讲边解释，最后再做总结。

求解这张表有两种方式，本质上都是一样的：

1、 从左到右，从上往下，一行一行求解

先从左到右求出第一行的所有元素的值，接着从左到右求第二行的值，一直到最后一行

2、 从上往下，从左到右，一列一列求解

先从上往下求出第一列的所有元素的值，接着从上往下求第二列的值，一直到最后一列

这里讲解第一种按行求解的方式：

首先是最左上角这个元素的值，这是要初始化的，由于行列都是 A，所以这个值是 1：

	A	B	D	B	C	D	F
A	1						
B							
C							
D							

然后接下来从左往右求第一行的所有值，这里我先直接把结果贴出来，先不要想怎么求，先看懂后面那几行怎么求就知道这个要怎么求了，当然第二行第一列的也先不管怎么求，后面会解释：

	A	B	D	B	C	D	F
A	1	1	1	1	1	1	1
B	1						
C							
D							

接下来就看红色格子要怎么求了：

这个格子表示的是 AB 与 AB 的最长公共子序列长度，那自然是 2 了，关键是，怎么来的？

我们先看看它左边、上边、左上那三个格子的意义：

1、 左边：AB 与 A 的最长公共子序列长度

2、 上边：A 与 AB 的最长公共子序列长度

3、 左上：A 与 A 的最长公共子序列长度

而红色格子是 AB 与 AB，是在它左上的基础上，两个字符串都增加一个字符，并且是相同的字符，所以它的最长公共子序列是不是就应该是它左上的格子数值加 1？

给两个字符串都增加一个相同的字符，那么是不是它的最长公共子序列长度就是原来的加 1？比如 ABCD 与 ATC 的最长公共子序列是 AC，那我给它们两个后面都加个 E，是不是它们（ABCDE 与 ATCE）的最长公共子序列就变成了 ACE，长度就增加了 1？

所以这一步求解的结果应该是这样的，我们把它左上的数字也标红了，表示它是有它左上的格子加 1 得到的：

	A	B	D	B	C	D	F
A	1	1	1	1	1	1	1
B	1	2					
C							
D							

接下来继续求解下一个红色格子：

首先我们还是来看这个格子的意义：AB 与 ABD 的最长公共子序列长度，所以结果依然是 2，这个 2 应该怎么来？

依然是看它左边、上边、左上那三个格子的意义：

1、 左边：AB 与 AB 的最长公共子序列长度

2、 上边：A 与 ABD 的最长公共子序列长度

3、 左上：A 与 AB 的最长公共子序列长度

我们先假设跟前面一样用左上这个格子，也就是说，从 A 与 AB 增加到 AB 与 ABD，我只知道新增的两个字符不一样，哪里知道你会不会跟我原来前面的一样，所以这时候就不能用它来看了...

其实这是给两个字符串分别增加了一个不同的字符，拆开来看，其实就相当于先给其中一个增加一个字符，再给另一个增加一个字符，也就是有下面这两种情况：

1、 从 A 与 AB，先给前面的 A 增加一个 B，变成 AB 与 AB（这时候最长公共子序列长度就变成 2 了），再给后面的 AB 增加一个 D，变成 AB 与 ABD

2、 从 A 与 AB，先给后面的 AB 增加一个 D，变成 A 与 ABD（这时候最长公共子序列长度还是 1），再给前面的 A 增加一个 B，变成 AB 与 ABD

因为我们知道新增的这两个字符是不一样的（如果是一样的，那么我们应该是按照上面描述的那种方式去计算，而不是这种方式），所以关键在于，新增的字符跟我原先的字符是否一样，就像这里，增加一个 B 和增加一个 D，其实 B 是跟我前面一样的，所以最长公共子序列长度可以加 1，这是在用第 1 种方式增加的时候就可以检测出来的，同理，有些时候你也可以从第二种方式检测出来，比如最简单的你把两个字符串调换过来就好了，从 AB 与 A 增加到 ABD 与 AB，当然我们等下有个从 AB 与 A 增加到 ABC 与 AB 的例子...

既然两种方式都有可能检测出最长公共子序列长度加 1，那么这里的最长公共子序列长度就应该是这两种方式里面较大的那个咯，所以就应该是它左边或者上边这两个格子之间较大的那个：

	A	B	D	B	C	D	F
A	1	1	1	1	1	1	1
B	1	2	2				
C							
D							

换句话讲，你可以有两种方式来获得这个值：

	A	B	D	B	C	D	F
A	1	1	1	1	1	1	1
B	1	2	2	1			
C							
D							

	A	B	D	B	C	D	F
A	1	1	1	1	1	1	1
B	1	2	2	2			
C							
D							

这种时候很明显就是应该取这个较大的了。

接下来继续往后求，中间就 pass 了，直接到刚刚说的 ABC 与 AB 的情况：

	A	B	D	B	C	D	F
A	1	1	1	1	1	1	1
B	1	2	2	2	2	2	2
C	1						
D							

这时候，因为 C != B，所以我们还是有两种方式，最终我们采取了上面的 2：

	A	B	D	B	C	D	F
A	1	1	1	1	1	1	1
B	1	2	2	2	2	2	2
C	1	2					
D							

所以除了第一行和第一列之外的所有格子，我们都会求解了，现在问题是，怎么求解第一行和第一列？

有两种方式：

1、 对第一行和第一列先做特殊处理：

对于第一行，因为它没有上一格和左上，所以如果行列字符相同，则最长公共子序列长度是 1，如果不相同，那么就直接让它等于它的左边格子的值（如果是第 0 个格子就让它等于 0）

对于第一列，同理了，不废话了...

2、 做一步预处理，让第一行和第一列变成中间行中间列，无需特殊对待

我觉得你看看这个表格大概就能顿悟了：

		A	B	D	B	C	D	F
	0	0	0	0	0	0	0	0
A	0	1	1	1	1	1	1	1
B	0	1						
C	0	1						
D	0	1						

在最上面跟最左边增加一行 0 和一列 0 就好了嘛，这些 0 的意义就是，字符串跟字符空串的最长公共子序列长度是 0 嘛

好的，接下来总结：

对于一个格子，如果行列字符相同，那么它的数值就是它左上格子的数值加 1

如果行列字符不相同，那么它的数值就是它左边或者上边这两个格子的数值中较大的那个。

就这么简单两句话，具体代码自己实现去~

当然，这里给你算出了最长公共子序列的长度，也许你还需要输出一个或者所有最长公共子序列，这个你就自己想想怎么通过这个表格来找最长公共子序列吧~