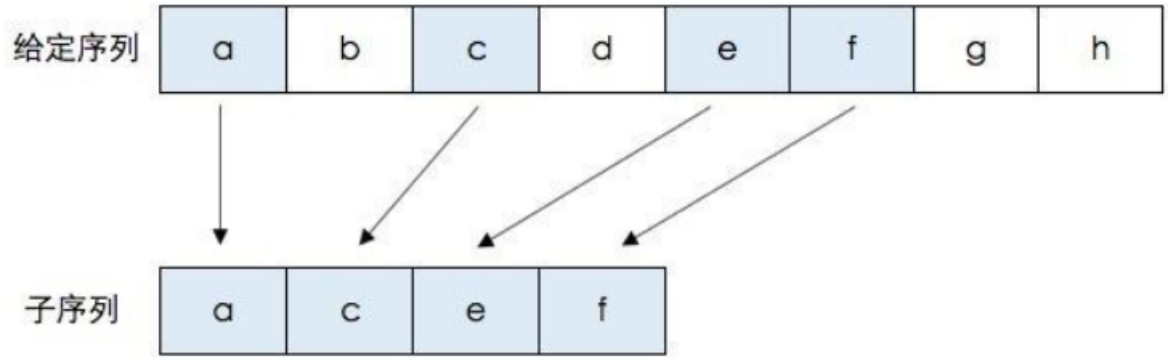
动态规划算法求解最长公共子序列问题

PB19030861 王湘峰

一、问题描述

给定两个字符串 s_1 和 s_2 ，要求给出两者最长的公共字符子序列的长度。即最长公共子序列问题（longest common sequence，LCS）。子序列的定义：如果字符串a中字符出现的先后顺序在字符串b中也存在，则称a是b的子序列。例如ace是abcde的子序列。



二、算法原理

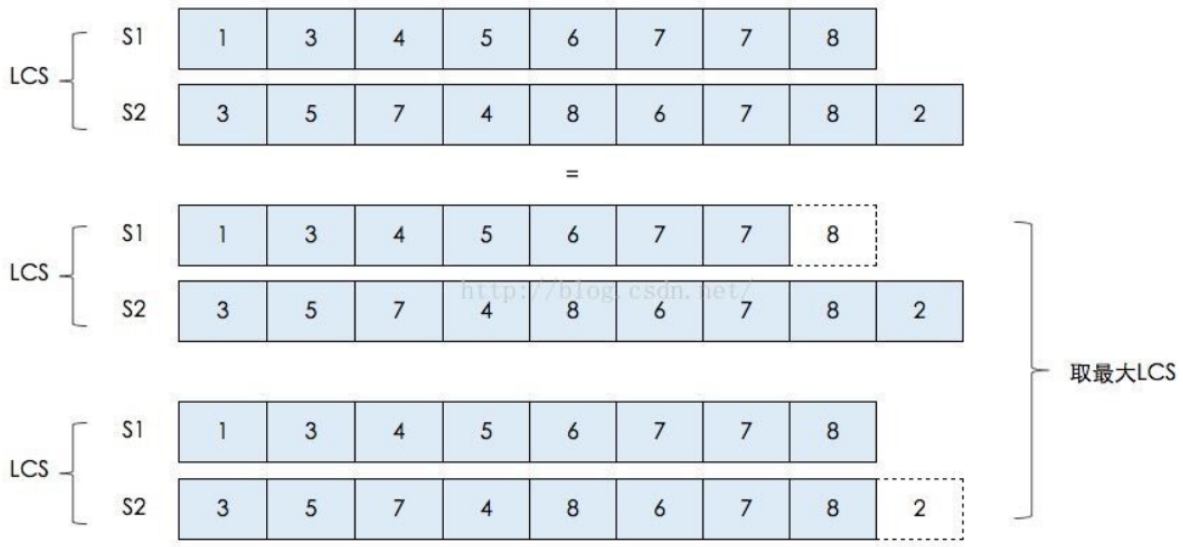
该问题是经典动态规划问题之一，其关键是找到问题的状态转移方程。分别考虑如下情况：

① 若字符串 s_1 和 s_2 的最后一位字符相同，记为 c ，则 c 一定包含在最长公共子序列中。

证明：

反证法，假设 c 不在最长公共子序列中，设公共最长子序列为 s' ，则 s' 是 s_1-c 和 s_2-c 的 LCS，但由于c是 s_1 和 s_2 的最后一位，那么 $s'+c$ 一定是 s_1 和 s_2 的LCS，得证。

② 若字符串 s_1 和 s_2 的最后一位字符不相同，那么最长公共子序列的长度应为 $max\{LCS(s_1[1...n_1-1],s_2),LCS(s_1,s_2[1...n_2-1])\}$，即为$s_1$去掉最后一位字符后与$s_2$的$LCS$、以及$s_2$去掉最后一位字符后与$s_1$的$LCS$中的较大者。（其中 $LCS(s_1,s_2)$表示字符串 s_1 和 s_2 的最长公共子序列长度）



若s_1的长度为 i，s_2 的长度为 j，将 $LCS(s_1,s_2)$ 简记为$LCS(i,j)$.

综上知，函数LCS(s_1,s_2)的转移方程为

$$LCS[i,j]=\begin{cases}0 & \text{if } i=0 \text{ or } j=0\\ LCS[i-1,j-1]+1 & \text{if } i,j>0, s_1[i]=s_2[j]\\ \max\{LCS[i,j-1],LCS[i-1,j]\} & \text{if } i,j>0, s_1[i]\neq s_2[j]\end{cases}$$

有了上述递归式，问题的求解似乎就迎刃而解了，可是仔细分析发现如果直接用递归函数计算的话，不论是空间复杂度还是时间复杂度都差强人意，究其原因，是因为递归算法重复计算了相同的子问题。例如在计算 $LCS[i,j-1]$ 和 $LCS[i-1,j]$ 时都计算了子问题 $LCS[i-1,j-1]$. 不难想到可自底向上地解决问题，即从$LCS[0,0]$ 开始，一直计算到 $LCS[i,j]$ ，并保留中间计算结果。这就是动态规划的迭代算法，具有更高的执行效率。迭代算法的原理可用下图表示：

下标i \ 下标j		0	1	2	3	4	5	6	7	8	9
	$S2_j$	3	5	7	4	8	6	7	8	2	
0	$S1_i$										
1	1										
2	3										
3	4										
4	5										
5	6										
6	7										
7	7										
8	8										

图中第i行第j列的值即为$LCS[i,j]$

复杂度分析：由于填写表格中的每个空花费的时间都是$O(1)$，所以算法的时间复杂度为$O(n_1n_2)$.（n_1和n_2分别是s_1和s_2的长度）

三、数据集说明

本程序输入和处理的数据均为字符串，输出为整数。

四、关键代码展示

```
def LCS(s1, s2):
    start = time()
    n1 = len(s1)
    n2 = len(s2)
    table = [[0] * (n2 + 1) for i in range(n1 + 1)]
```

```
    record = [[0] * n2 for i in range(n1)]
    for i in range(n1):
        for j in range(n2):
            if s1[i] == s2[j]:
                table[i + 1][j + 1] = table[i][j] + 1
                record[i][j] = 1
            elif table[i][j + 1] > table[i + 1][j]:
                table[i + 1][j + 1] = table[i][j + 1]
                record[i][j] = 2
            else:
                table[i + 1][j + 1] = table[i + 1][j]
                record[i][j] = 3
    # print longest common string
    lcs = []
    i, j = n1 - 1, n2 - 1
    while True:
        if i < 0 or j < 0:
            break
        if record[i][j] == 1:
            lcs.insert(0, s1[i])
            i -= 1
            j -= 1
        elif record[i][j] == 2:
            i -= 1
        else:
            j -= 1
    print('\n最长公共子序列为：', end='')
    for s in lcs:
        print(s, end='')
    stop = time()
    print('\n长度为', table[n1][n2], ' 用时：%.16f ms\n' % ((stop - start) * 1000))
```

变量解释

table是计算LCS时储存中间值的表，其规模为$(n_1+1)(n_2+1)$. record是用于记录最长公共子序列的表格，通过record可以回溯公共子序列。

五、用户指南

环境配置

本程序编写语言为Python 3.8，需要安装Python3及以上编译器才可以运行。

输入输出说明:

①输入任意两个字符串（用空格分开），敲击Enter即可输出最长公共子串的长度和处理时间

②输入exit并敲击Enter即可退出程序

测试样例说明

在文件 “LCS样例.txt” 中包含了本人挑选的几个样例,包括《再别康桥》和《雨巷》的LCS、圆周率π和自然对数的底e的LCS等。

使用方式为：直接全部复制，在命令行粘贴即可。

六、程序测试结果

以 “LCS样例.txt” 中的数据为例：

```
D:\干点正事\大三上\运筹学\实验报告\1>D:\干点正事\大三上\运筹学\实验报告\1\LCS.py
---程序名称：最长公共子序列问题---
-----  王湘峰PB19030861 -----
请输入两个字符串，以回车作为分割，输入exit以结束
中国科学技术大学
中国科学院大学

最长公共子序列为：中国科学大学
长度为 6  用时：0.1137256622314453 ms

轻轻的我走了正如我轻轻的来我轻轻的招手作别西天的云彩那河畔的金柳是夕阳中的新娘波光里的艳影在
我的心头荡漾软泥上的青荇油油的在水底招摇在康河的柔波里我甘心做一条水草那榆荫下的一潭不是清泉
是天上虹揉碎在浮藻间沉淀着彩虹似的梦寻梦撑一支长篙向青草更青处漫溯满载一船星辉在星辉斑斓里放
歌但我不能放歌悄悄是别离的笙箫夏虫也为我沉默沉默是今晚的康桥悄悄的我走了正如我悄悄的来我挥一
挥衣袖不带走一片云彩
撑着油纸伞独自彷徨在悠长悠长又寂寥的雨巷我希望逢着一个丁香一样的结着愁怨的姑娘她是有丁香一样
的颜色丁香一样的芬芳丁香一样的忧愁在雨中哀怨哀怨又彷徨她彷徨在这寂寥的雨巷撑着油纸伞像我一样
像我一样地默默彳亍着冷漠凄清又惆怅她静默地走近走近又投出太息一般的眼光她飘过像梦一般的像梦一
般的凄婉迷茫像梦中飘过一枝丁香地我身旁飘过这女郎她静默地远了远了到了颓圮的篱墙走尽这雨巷在雨
的哀曲里消了她的颜色散了她的芬芳消散了甚至她的太息般的眼光丁香般的惆怅撑着油纸伞独自彷徨在悠
长悠长又寂寥的雨巷我希望飘过一个丁香一样的结着愁怨的姑娘

最长公共子序列为：的我的的娘的的的在在的我一一着的梦一一的我默的走了的我一一
长度为 29  用时：11.0032558441162109 ms

3.14159265358979323846264338327950288419716939937510582097494459230781640628620899862803482534
2117067982148086513282306647093844609550582231725359408128481117450284102701938521105559644622
2948954930381964428810975665933446128475648233786783165271201909145648566923460348610454326648
2133936072602491412737245870066063155881748815209209628292540917153643678925903600113305305488
2046652138414695194151160943305727036575959195309218611738193261179310511854807446237996274956
7351885752724891227938183011949129833673362440656643086021394946395224737190702179860943702770
5392171762931767523846748184676694051320005681271452635608277857713427577896091736371787214684
4090122495343014654958537105079227968925892354201995611212902196086403441815981362977477130996
0518707211349999998372978049951059731732816096318595024459455346908302642522308253344685035261
9311881710100031378387528865875332083814206171776691473035982534904287554687311595628638823537
8759375195778185778053217122680661300192787661119590921642019892.7182818284590452353602874713
5266249775724709369995957496696762772407663035354759457138217852516642742746639193200305992181
7413596629043572900334295260595630738132328627943490763233829880753195251019011573834187930702
1540891499348841675092447614606680822648001684774118537423454424371075390777449920695517027618
3860626133138458300075204493382656029760673711320070932870912744374704723069697720931014169283
6819025515108657463772111252389784425056953696770785449969967946864454905987931636889230098793
1277361782154249992295763514822082698951936680

最长公共子序列为：.1159535328426295247939959749966220603534794513282664744631235921817135966294
5903429565933128476233883152101901538341930702491434884150920682240164711534234544237107390744
9929551727183860663134530007209262976737113200702871277707367720930146983689255151086574677211
3897845059369685449696484450598731388230831277617825424922573518220698951980
长度为 340  用时：97.1593856811523438 ms

exit
```

可以看出计算花费的时间基本是毫秒级的，与直接递归相比具有巨大的提升。

七、分析总结

本次实验实现了解决LCS问题的动态规划算法，并且将其递归形式改成了效率更高的迭代形式。算法本身较为简单，程序的执行效率较为满意。其中实验中用到的矩阵存储的思想在多种动态规划中均有应用，若有机会还会尝试用其解决背包问题和找零问题。