

01-F-8 理解LCS & 01-F-9 LCS动态规划

#数据结构邓神

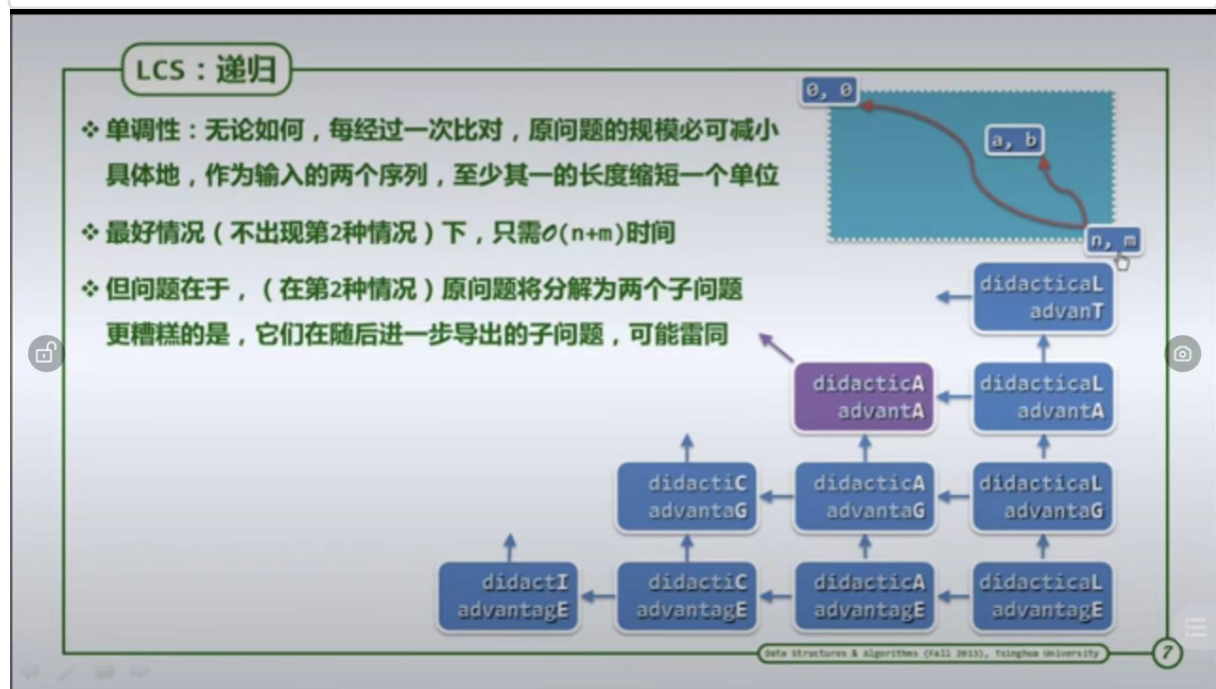
Q: 为什么算法正确:

1: 单调性: 无论如何, 每经过一次比对, 原来的问题规模必然减少
具体的, 作为输入的两个序列, 至少其一的长度缩短一个单位

最好情况下 (也就是只有 0 和 1 两种情况) 只需要 $O(n+m)$ 的时间 也就是每次两个序列都减少一个单位

但问题在于 (第二个情况) 原问题将会分解为两个问题, 更糟糕的是在子问题的子问题中也会出现这问题, 而且可能雷同 (这与 Fib 数列的效率低下问题一致)

在最坏的情况下 复杂度为 $T(n) = O(2^n)$



我们会有大量的重复实例的计算, 我们是否也可以采用记忆法?

最坏的情况下先后共出现 $O(2^n)$ 个

但是各个子问题分别对应于 A 与 B 某个前缀的总和

因此总共不超过 $n*m$ 种

采用 动态规划的方法, 只需要 $O(n*m)$ 时间就可以计算出所有问题

算法：

1. 将所有的子问题思考为一张表

颠倒计算方向，从 $LCS(A[0], B[0])$ 出发，依次计算所有问项

LCS : 迭代

- ❖ 与 `fib()` 类似
这里也有大量重复的递归实例（子问题）
（最坏情况下）先后共计出现 $O(2^n)$ 个
- ❖ 各子问题，分别对应于A和B的某个前缀组合
因此总共不过 $O(n*m)$ 种
- ❖ 采用动态规划的策略
只需 $O(n*m)$ 时间即可计算出所有子问题
- ❖ 为此，只需
 - 0) 将所有子问题（假想地）列成一张表
 - 1) 颠倒计算方向，从 $LCS(A[0], B[0])$ 出发
依次计算出所有项

		d	i	d	a	c	t	i	c	a	l
		0	0	0	0	0	0	0	0	0	0
a	0	0	0	0	1	1	1	1	1	1	1
d	0	1	1	1	1	1	1	1	1	1	1
v	0	1	1	1	1	1	1	1	1	1	1
a	0	1	1	1	2	2	2	2	2	2	2
n	0	1	1	1	2	2	2	2	2	2	2
t	0	1	1	1	2	2	3	3	3	3	3
a	0	1	1	1	2	2	3	3	3	4	4
g	0	1	1	1	2	2	3	3	3	4	4
e	0	1	1	1	2	2	3	3	3	4	4

beta Structures & Algorithms (Fall 2010), Tsinghua University