

绪论

动态规划：最长公共子序列

世上一切都无独有偶，为什么你与我却否？

Make it work, make it right, make it fast.

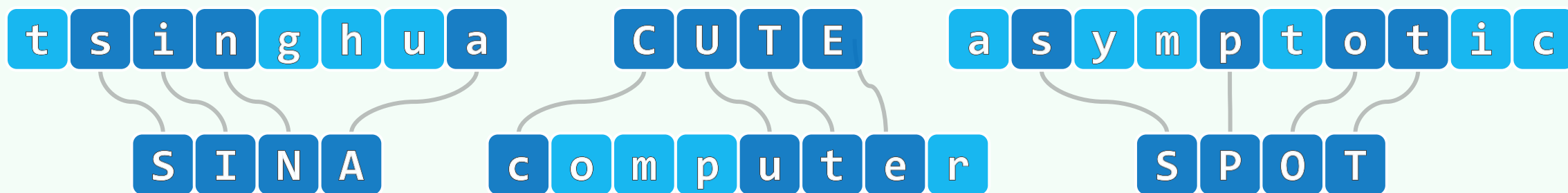
- Kent Beck

邓俊辉

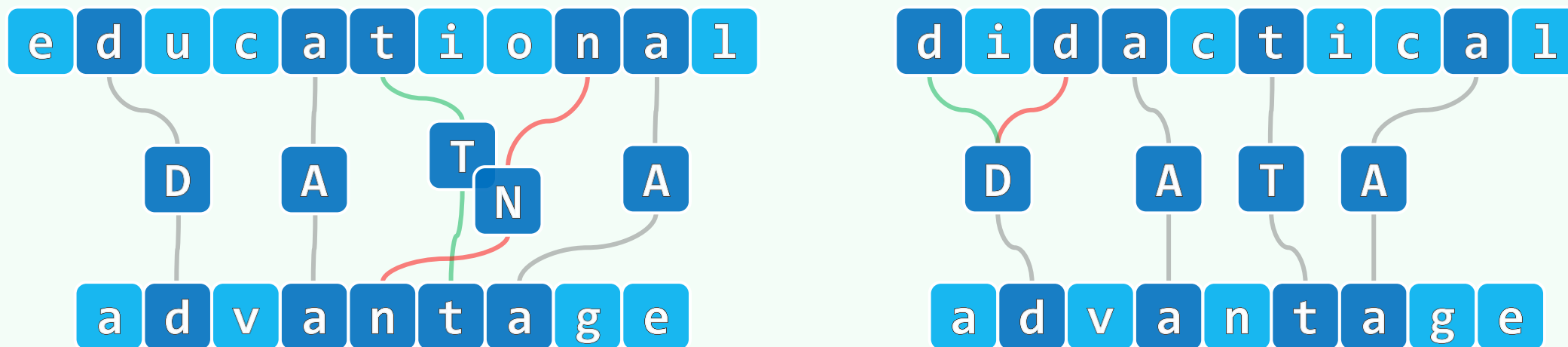
deng@tsinghua.edu.cn

问题

❖ 子序列 (Subsequence) : 由序列中若干字符 , 按原相对次序构成



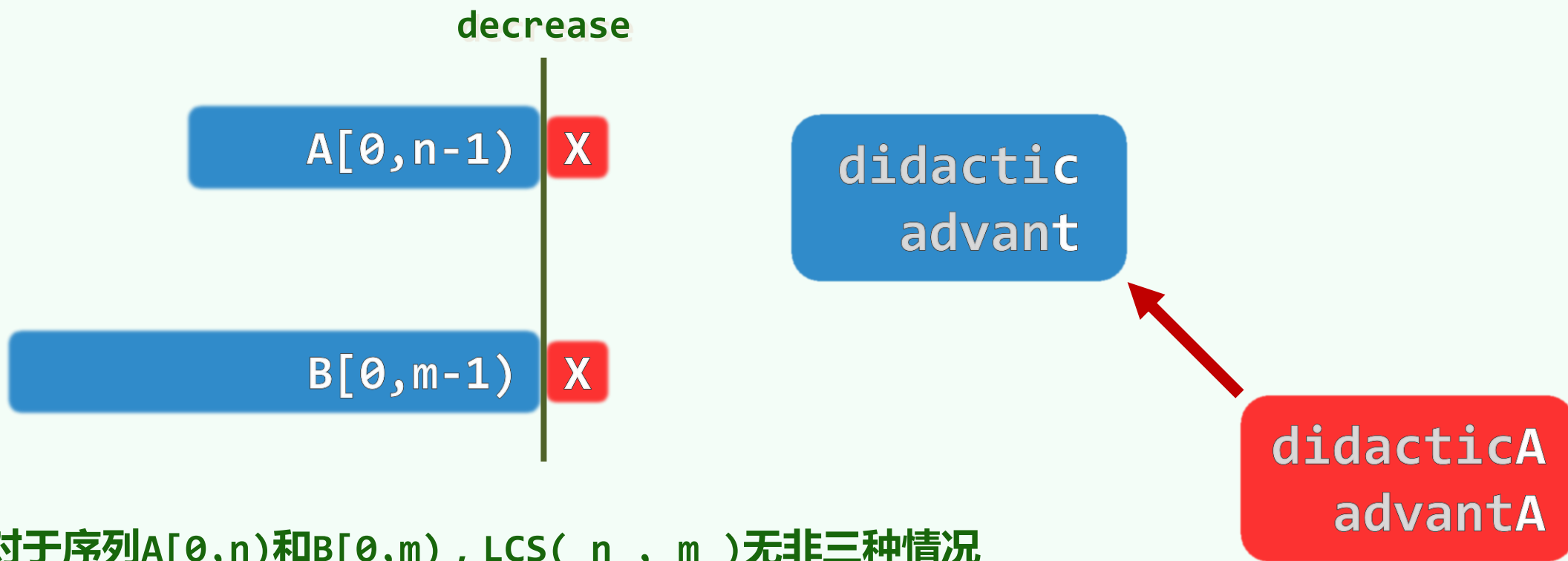
❖ 最长公共子序列 (Longest Common Subsequence) : 两个序列公共子序列中的最长者



❖ 可能有多個

可能有歧义

减治递归

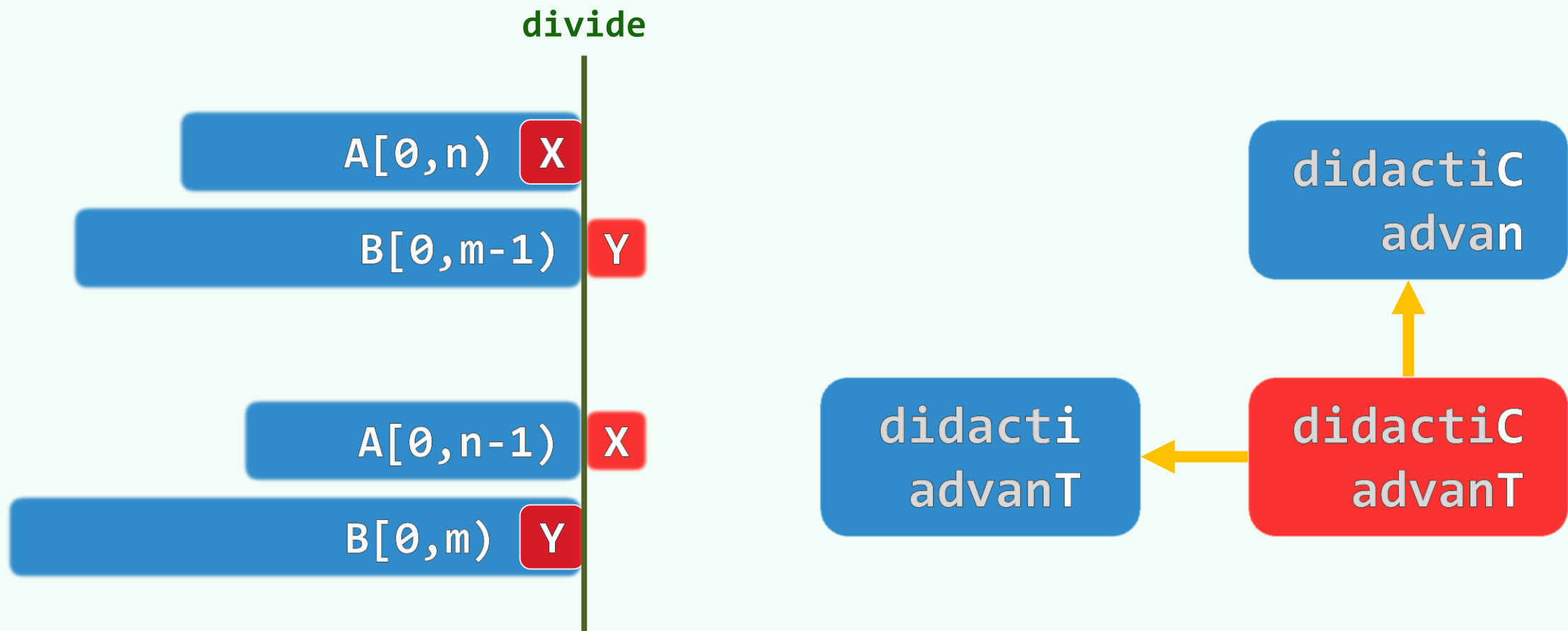


❖ 对于序列 $A[0, n)$ 和 $B[0, m)$ ， $LCS(n, m)$ 无非三种情况

0) 若 $n = -1$ 或 $m = -1$ ，则取作空序列（""） //递归基

1) 若 $A[n] = 'X' = B[m]$ ，则取作： $LCS(n-1, m-1) + 'X'$ //减而治之

分治递归

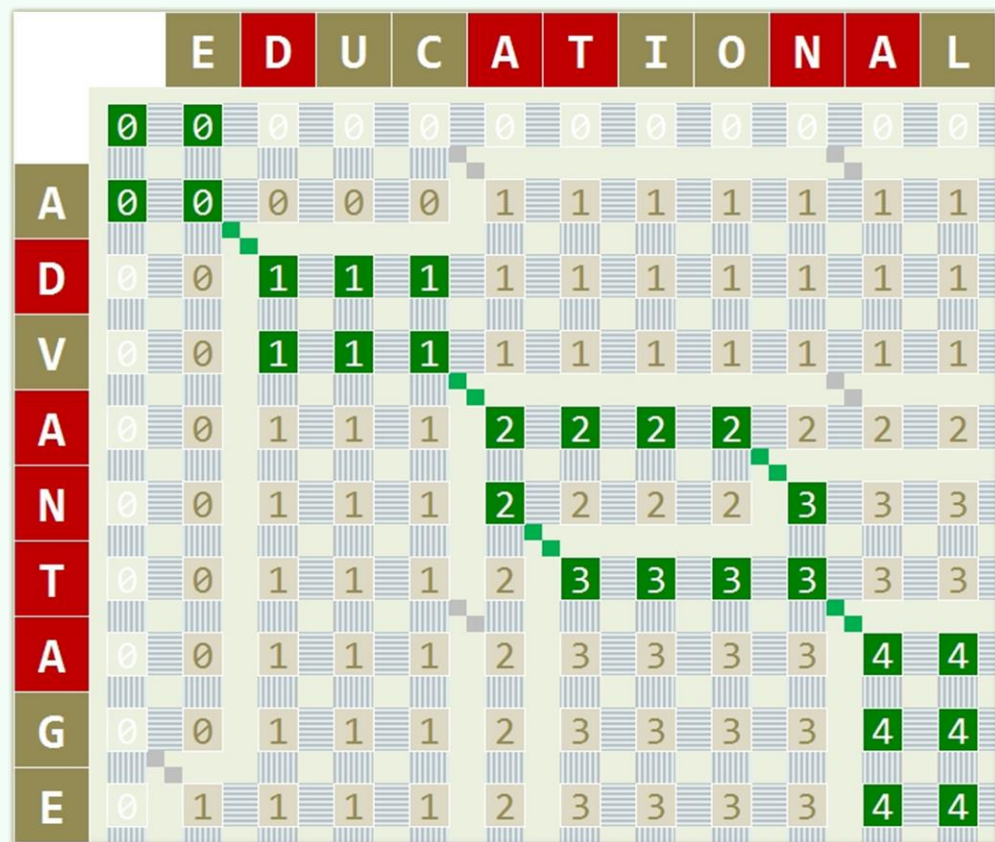


2) $A[n] \neq B[m]$, 则在 $LCS(n, m-1)$ 与 $LCS(n-1, m)$ 中取更长者

//分而治之

理解

❖ LCS的每一个解，对应于 $(0,0)$ 与 (n,m) 之间的一条**单调通路**；反之亦然



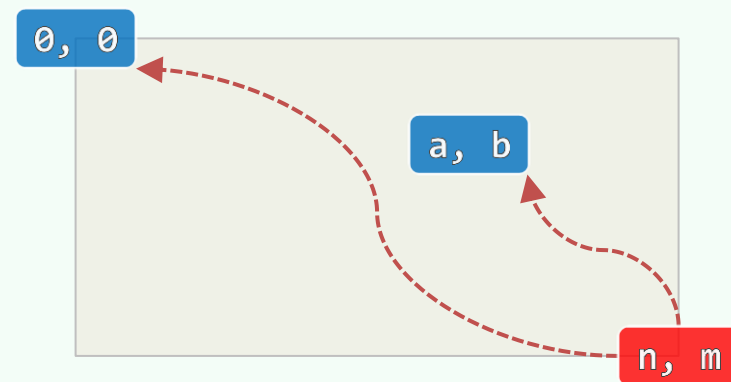
多解



歧义

复杂度

- ❖ 单调性：每经一次比对，至少一个序列的长度缩短一个单位
- ❖ 最好情况，只需 $O(n + m)$ 时间 //比如...
- ❖ 然而最坏情况下，子问题数量不仅会增加，且可能大量雷同



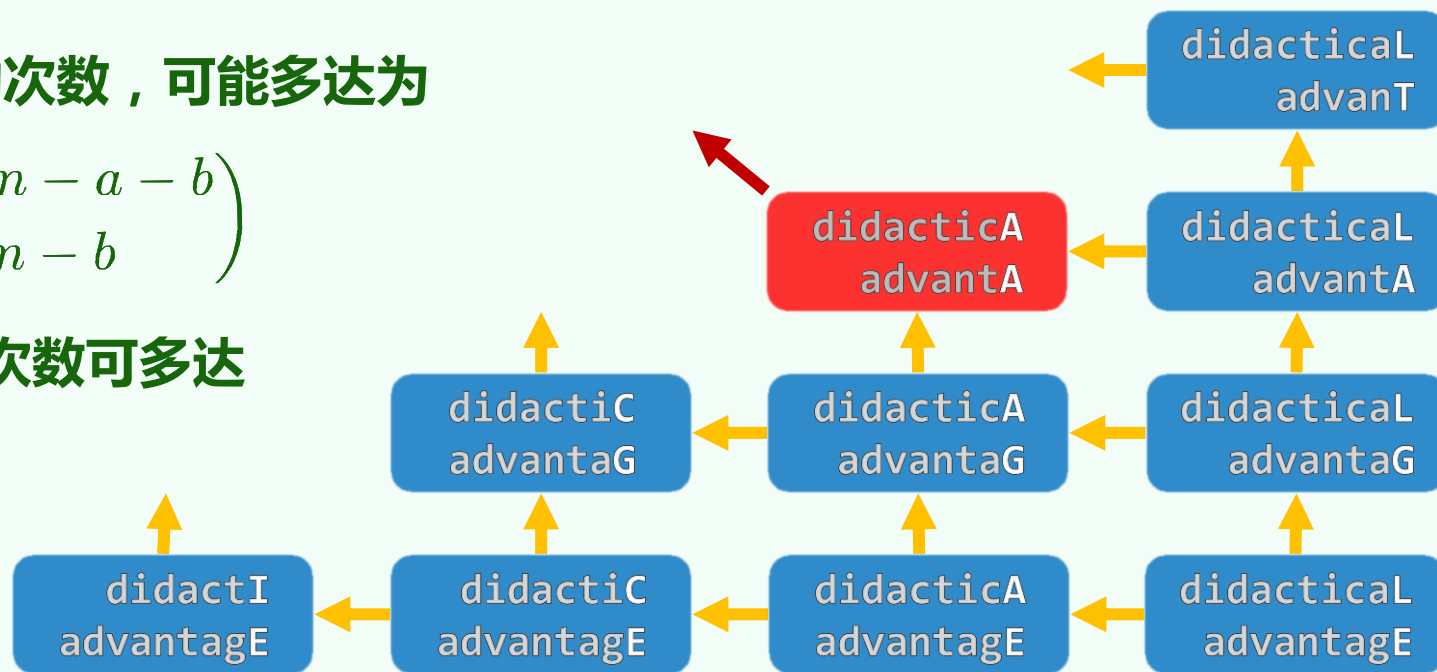
子任务 $LCS(A[a], B[b])$ 重复的次数，可能多达为

$$\binom{n + m - a - b}{n - a} = \binom{n + m - a - b}{m - b}$$

特别地， $LCS(A[0], B[0])$ 的次数可多达

$$\binom{n + m}{n} = \binom{n + m}{m}$$

当 $n = m$ 时，为 $\Omega(2^n)$



动态规划

❖ 与fib()类似，这里也有大量重复的递归实例（子问题）

各子问题，分别对应于A和B的某个前缀组合

因此实际上，总共不过 $\mathcal{O}(n \cdot m)$ 种

❖ 采用动态规划的策略

只需 $\mathcal{O}(n \cdot m)$ 时间即可计算出所有子问题

❖ 为此，只需

- 将所有子问题（假想地）列成一张表

- 颠倒计算方向：从LCS(0,0)出发，依次计算出所有项——直至LCS(n,m)

		d	i	d	a	c	t	i	c	a	l
	0	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

课后

- ❖ 温习：程序设计基础（第3版）之第11章（动态规划）
- ❖ 自学：Introduction to Algorithms, §15.1, §15.3, §15.4
- ❖ 本节所介绍的迭代式LCS算法，似乎需要记录每个子问题的局部解，从而导致空间复杂度激增
试改进该算法，使得每个子问题只需常数空间，即可保证最终得到LCS的组成（而非仅仅长度）
- ❖ 考查序列 $A = \text{"immaculate"}$ 和 $B = \text{"computer"}$
 - 它们的LCS是什么？
 - 这里的解是否唯一？是否有歧义性？按本节所给的算法，找出的是其中哪一个解？
- ❖ 实现LCS算法的递归版和迭代版，并通过实测比较其运行时间
- ❖ 采用Memoization策略，实现fib与LCS算法