稿子：

P1：我们是第13组，我们找的论文是关于LCSk问题的新算法。

P2：首先是LCS问题的原始定义——给定两个序列A、B，找出它们的最长公共子序列。备注子序列不用连续

P3：介绍下为什么要研究LCS问题。LCS问题是很经典的计算机科学领域的问题，它是数据比较程序的基础，应用方面有计算语言学、生物信息学领域，也被广泛地用在版本控制系统比如大家熟悉的Git，越是基础的东西往往越是重要。

P4：我们的proj所做的论文是提出了解决LCSk和LCS>=k问题的空间效率更高的新算法，这两个问题是LCS问题的衍生，定义如下。

LCSk这个定义简单来说就是找到长度为k的共同子序列的个数。

我们的报告里主要只研究了这个算法，LCS>=k的算法和它很相似因此略过了。

P5：首先根据这个定义，最原始的算法就是标准的动态规划。这个f(i,j)就是考虑A的前i个元素和B的前j个元素，有多少个k长度共同子序列，状态转移方程如图所示。这里的d(i,j)存的是从ai和bj倒着往前数的连续相同元素的个数，然后lambda(d(i,j))就是到i,j处够不够新凑一个k长度的子序列。

P6：f(I,j)具体是这么更新的，实际要填的是右下的(n-k)(m-k)个格子，其他的因为考虑的A或B自身不满k个所以显然是0，时间复杂度和空间复杂度都是O(mn)。

空间上还可以优化，注意到从上往下一行一行填的话，算某一行的时候只需要它本身那行、它上一行、和它往前数k行的那一行，所以我们往下填的过程中只需要保留最近的k行就够了，所以空间复杂度可以降到O(km)。但是这样的话，如果我们不止需要共同子序列个数、还需要给出这些序列具体是什么的话，这k行的信息量就不够了。

P7：要找出具体序列，也就是需要一个traceback的操作，论文里提出的算法就是结合divide and conquer。这个算法和课上之前讲过的Sequence Alignment有些相似。这个就是分治的伪代码，后面会具体展开。

P8：首先需要衍生一下LCSk问题的定义——把原来f(i,j)表示考虑A的前i个和B的前j个升级成考虑序列A的i0到i1区间和B的j0到j1区间的k长度公共子序列个数，类似的，LCSk Reverse定义g函数考虑A的i1到i0和B的j1到j0，也就是把A、B反向后的公共子序列个数。注意到，当i0=j0=1，升级版f就等于原来的f(i,j)，同样，当i1=n,j1=m，升级版g也可以退化到g(i,j)。于是当i0=j0=1,i1=n,j1=m，f(n,m)=g(1,1)。

P9：接下来就是正式的算法提出。首先为了保持O(km)的空间复杂度，我们依然只维护 (k+1)行，我们用两张(k+1)\*m的表来存“子序列个数”这个数字结果，右边这个伪代码里第三行的xi函数计算的L1对应f，eta函数计算的L2对应g，根据第二行的l的设置，每次L1和L2都会有中间k-1行重合的部分。

P10：然后插播一个重要的定理，就是任意k行里，挨着的斜对角的f(i,j)+g(i+1,j+1)的最大值，就是我们要找的整个A、B序列的子序列个数。具体证明很长，要分很多种情况讨论，这里略过，感兴趣可以看原论文。

P11：有了上述定理后，就可以用split函数找出等于最终结果的那个i，j分割点，后面称为k1，k2。这个函数是利用存的表大L1和L2，找出满足右下条件的小l1，l2。

P12：如果小l1，l2大于1，说明现在考虑的区间里有多个相同子序列，依然难以追溯，于是根据分割点k1、k2继续divide，直到l1，l2等于1，这意味着当前区间里只有一个相同子序列，于是找到这个区间里最靠前的还是1的位置，把它记录下来就可以了。

P13：然后是复杂度分析，时间上依然是O(mn)。上面这个T(n,m)是递归表达式，我们先假设它确实是O(mn)并且被c1mn bound住了，也记加的那项O(mn)被c2mn bound。

哦这个算法是只有n>2k的时候才用，n<=2k的时候直接用原始的标准dp就行，也是O(km)的空间复杂度。当n>2k的时候，k1有个性质时它一定在n/4和四分之三n之间，为了证明假设成立，需要证明第一行这个放缩完也能被一些c1mn bound住。第二行这个≤是放缩k1，k1乘k2项用四分之三n放缩，后面这项用四分之n放缩，然后就等于号计算，最后化简出来是≤四分之三c1加c2的mn，要让这个小于等于c1，显然是存在这样的c1的，只要让c1≥4c2就行。所以假设成立，时间复杂度O(mn)

P14：然后是空间复杂度，我们分治过程中最大的临时存储就L1和L2两张表，各自(k+1)m个格子，其他的k1，k2都是O(1)复杂度,所以总的空间复杂度是是O(km)。

P15：这是我们根据它伪代码写的程序跑出来的结果，STD Answer是原始dp的结果，用来验证后面分治算法对不对的，至少我们测的看起来挺对的。

P16：further improvement没有，略微一点点改动大概是我们L1和L2开的全局变量省去了反复创建和销毁数组……最后是分工，代码是李欣然负责，report的撰写是陈雨瑶和我。

谢谢大家，我们的pre结束了。