hw7

> 15.2-1，15.2-5；15.3-2，15.3-4；15.4-1，15.4-4；

15.2-1

15.2-1 对矩阵规模序列⟨5，10，3，12，5，50，6⟩，求矩阵链最优括号化方案。

该序列的一个最优括号化是 $(A_1A_2)((A_3A_4)(A_5A_6))$，这需要 $5*50*6+3*12*5+5*10*3+3*5*6+5*3*6=1500+180+150+90+90=2010$。

15.2-5

15.2-5 令 $R(i, j)$表示在一次调用 MATRIX-CHAIN-ORDER 过程中，计算其他表项时访问表项 $m[i, j]$的次数。证明：

$$\sum_{i=1}^{n}\sum_{j=i}^{n}R(i,j)=\frac{n^3-n}{3}$$

（提示：证明中可用到公式(A.3)。）

计算的是在m中引用与正在计算的不同条目的次数，也就是第10行执行的次数的2倍。

$$\begin{aligned}
\sum_{l=2}^{n}\sum_{i=1}^{n-l+1}\sum_{k=i}^{i+l-2}2 &= \sum_{l=2}^{n}\sum_{i=1}^{n-l+1}(l-1)2 \\
&= \sum_{l=2}^{n}2(l-1)(n-l+1) \\
&= \sum_{l=1}^{n-1}2l(n-l) \\
&= 2n\sum_{l=1}^{n-1}l-2\sum_{l=1}^{n-1}l^2 \\
&= n^2(n-1)-\frac{(n-1)n(2n-1)}{3} \\
&= n^3-n^2-\frac{2n^3-3n^2+n}{3} \\
&= \frac{n^3-n}{3}
\end{aligned}$$

15.3-2

15.3-2 对一个 16 个元素的数组，画出 2.3.1 节中 MERGE-SORT 过程运行的递归调用树。解释备忘技术为什么对 MERGE-SORT 这种分治算法无效。

让 $[i\,..\,j]$ 表示对原始数组中位置 i 到 j 的元素进行排序的归并排序调用。递归树的根将是 [1..n]，并且在任何节点 $[i\,..\,j]$，它将分别有 $[i\,..\,(j-i)/2]$ 和 $[(j-i)/2+1..\,j]$ 作为它的左右子节点。如果 $j-i=1$，那么将没有子节点。记忆化方法无法加快归并排序，因为子问题并不重叠。排序一个大小为 n 的列表并不等同于排序另一个大小为 n 的列表，所以存储子问题的解决方案没有节省，因为每个解决方案最多被使用一次。

15.3-4

15.3-4 如前所述，使用动态规划方法，我们首先求解子问题，然后选择哪些子问题用来构造原问题的最优解。Capulet 教授认为，我们不必为了求原问题的最优解而总是求解出所有子问题。她建议，在求矩阵链乘法问题的最优解时，我们总是可以在求解子问题之前选定 $A_iA_{i+1}\cdots A_j$ 的划分位置 A_k（选定的 k 使得 $p_{i-1}p_kp_j$ 最小）。请找出一个反例，证明这个贪心方法可能生成次优解。

给定矩阵 A_1, A_2, A_3 和 A_4，其维度为 $p_0, p_1, p_2, p_3, p_4 = 1000, 100, 20, 10, 1000$。为了最小化 $p_0p_kp_4$，选择 $k = 3$ 是必要的，因此需求解矩阵 $A_1A_2A_3$ 的乘积问题，同时矩阵 A_4 的乘积问题可自动得到解决。依照算法，这通过在 $k = 2$ 处分割来完成。完整的加括号方法是 $(((A_1A_2)A_3)A_4)$。该方法需要 $1000 \cdot 100 \cdot 20 + 1000 \cdot 20 \cdot 10 + 1000 \cdot 10 \cdot 1000 = 12,200,000$ 次标量乘法。而若采用不同的括号化顺序 $((A_1\,(A_2A_3))A_4)$，则仅需 $100 \cdot 20 \cdot 10 + 1000 \cdot 100 \cdot 10 + 1000 \cdot 10 \cdot 1000 = 11,020,000$ 次标量乘法，较之前的方法要少。因此该贪婪算法得出的解并非最优。

15.4-1

15.4-1 求⟨1，0，0，1，0，1，0，1⟩和⟨0，1，0，1，1，0，1，1，0⟩的一个 LCS。

最长公共子序列（LCS）是 ⟨1, 0, 1, 0, 1, 0⟩。可以观察到这一点，第一个列表包含“00”，而第二个列表则不含“00”。同样，第二个列表包含两个“11”的副本，而第一个列表则不含。为了调和这一差异，任何LCS都必须至少跳过三个元素。由于满足这一点，就可以知道找到的公共子序列是最大的。

15.4-4

15.4-4 说明如何只使用表 c 中 $2\times\min(m, n)$ 个表项及 $O(1)$ 的额外空间来计算 LCS 的长度。然后说明如何只用 $\min(m, n)$ 个表项及 $O(1)$ 的额外空间完成相同的工作。

在计算最长公共子序列（LCS）问题的过程中，由于当前行的计算仅依赖于c表格的前一行，所以在计算第k行时，第$k-2$行可以被释放，因为在后续计算长度过程中不再需要该行数据。

为了进一步节省空间，注意到要计算$c[i, j]$，仅需使用到$c[i-1, j], c[i-1, j-1]$ 和 $c[i, j-1]$ 这三个值。因此，可以在计算的同时逐个释放那些不再需要的前一行条目，将空间需求降至 $\min(m, n)$。

从依赖的三个条目计算出下一个条目的时间和空间成本均为$O(1)$，这样的处理既保证了计算效率，又显著减少了所需的存储空间。