研究生算法课课堂笔记

上课日期： 10 月 24 日 **第(2)节课**

组长学号及姓名： 1601214455 陈震鹏

组员学号及姓名： 1601214451 常媛

1601214468 刘雨轩

1.k 段和代码：

```
F(j,0) = F(j,0) = 0, K = 0
for k = 1 to K:
{
    for j = 1 to n:
    {
        f(j,k) = a_j +max {F(j − 1, k − 1) ; f(j − 1, k)}
    }
}
F(j,k)
```

$$f(j,k) = a_j + \max \begin{cases} F(j-1,k-1) \\ f(j-1,k) \end{cases};$$

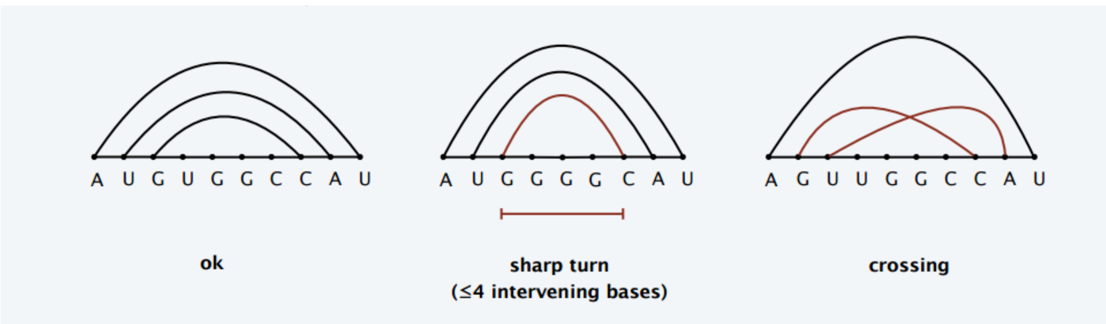
2.RNA

2.1 问题描述：

一条长长的 RNA 序列，求上面的基因，能匹配上的最多对数。匹配条件有三个：1）A 和 U 匹配，G 和 C 匹配，U 和 A 匹配，C 和G 匹配。2）两个匹配的节点间距离至少是 5。3）匹配的关系不能交叉。即如果将匹配的两个点连线，那么所有连线不能交叉。

2.2 问题本质：

矩阵乘法加括号



如图，图一为规范匹配实例，图二两个匹配的点之间距离小于 5，不合法。图三两对匹配的点连线交叉

注：一维动规和二维动规区别：是从一端遍历还是两端遍历

子问题划分：

划分成从第 i 个点，到第 j 个点之间，最多能有多少对匹配。记为 OPT(i,j).

有三种情况：

1）两个点之间距离小于 5，那么 OPT(i,j) = 0.

2 ） 节点 j 无法匹配，那么 OPT(i,j) = OPT(i,j-1)
3 ） 节点 j 可以匹配多个点。那么 OPT(i,j) = 1+max_t{OPT(i,t-1) , OPT(t+1,j-1)}

其中 t 为可以和节点 j 匹配的节点，且 i≤t<j-4.

2.3 递推公式：
其实就是上述情况三

$$f(i,j) = \max\begin{cases} f(i,j-1) \\ max_{i\le k<j} f(i,k-1)+f(k+1,j-1)+1 \end{cases}$$

f(i, j) 为从第 i 个点，到第 j 个点之间，最多能有多少对匹配。
因此第一种情况是不要 j 节点。第二种情况是 j 节点和 k 匹配。
如果是长为 n 的 RNA 序列，OPT(1,n)即为最后所求。

2.4 复杂度分析
空间复杂度就是开一个二维数组，为 n^2。

时间复杂度其实是有技巧的，先算长度为 2 的子段，再算长度为 3 的子段，依次类推，最后算长度为 n 的整体 RNA。这样每次拆分的时候，子段的结果都已经知道了。
但是每次查找哪个节点和 j 节点匹配的时候，还是要搜一遍，因此时间复杂度还是 n^3.

3.矩阵乘法（参照屈婉玲所著《算法设计与分析》）

问题描述：

设 $A_1, A_2, \ldots, A_n$ 为 n 个矩阵的序列，其中 A_i 为 $P_{i-1}\times P_i$ 阶矩阵，i=1, 2, …, n. 这个矩阵链的输入向量 P=<$P_0, P_1, \ldots, P_n$>给出，其中 P_0 是矩阵 A_1 的行数，P_i(i=1, 2, …, n-1)既是矩阵 A_i 的列数也是矩阵 A_{I+1} 的行数，最后的 P_n 是矩阵 A_n 的列数. 计算这个矩阵链需要做 n-1 次两个矩阵的相乘运算，可以用 n-1 对括号表示运算的顺序，因为矩阵乘法满足结合律，无论采用哪种顺序，最后的结果都是一样的. 但是，采用不同的顺序计算的工作量却不一样.

两个矩阵相乘的工作量定义如下：假设矩阵 A_1 与 A_2 相乘，其中 A_1 是 i 行 j 列的矩阵，A_2 是 j 行 k 列的矩阵，那么它们的乘积 A_1A_2 是 i 行 k 列的矩阵，含有 i*k 个元素. 以元素相乘作为基本运算，乘积中每个元素的计算都需要做 j 次乘法，于是计算 A_1A_2 总共需要 i*j*k 次乘法. 举例说明：

假设输入的是 P=<10, 100, 5, 50>，这说明有 3 个矩阵相乘，其中

A_1:10*100, A_2=100*5, A_3=5*50

有两种乘法次序，即(A_1A_2)A_3 和 A_1(A_2A_3).
如果采用第一种次序，执行的基本运算次数是：10*100*5+10*5*50=7500；
如果采用第二种次序，执行的基本运算次数是：10*100*50+100*5*50=75000.

目标：
给定向量 P，确定一种乘法次序，使得基本运算的总次数达到最少.

子问题的划分：

用 $A_{i\ldots j}$ 定义矩阵链 $A_iA_{i+1}\ldots A_j$ 相乘的子问题，m[i, j]表示得到乘积 $A_{i\ldots j}$ 所用的最少基本运算

次数. 假设其最后一次相乘发生在矩阵链 $A_{i…k}$ 和 $A_{k+1…j}$ 之间，即

$$A_iA_{i+1}…A_j = (A_iA_{i+1}… A_k)(A_{k+1}A_{k+2}… A_j) \qquad k=i, i+1, …, j-1$$

那么子问题 $A_{i…j}$ 依赖于子问题 $A_{i…k}$ 和 $A_{k+1…j}$ 的计算结果. 即 m[i, j]的值依赖于 m[i, k]和 m[k+1, j]的值，具体的依赖关系可以表示为

$$\mathrm{m[i, j]} = \begin{cases} 0\ (if\ i == j) \\ \min_{i \le k < j}\{m[i,k] + m[k+1,j] + P_{i-1}P_kP_j\}\ (if\ i < j) \end{cases}$$

上式中的 k 代表问题的划分位置，应该考虑所有可能的划分，即$i \le k < j$从中比较出较小的值；$P_{i-1}P_kP_j$ 是最后把两个子矩阵链 $A_{i…k}$ 和 $A_{k+!…j}$ 的结果矩阵相乘所需要的基本运算次数. 当 i=j 时，矩阵链只有一个矩阵 A_i，这时乘法次数为 0，对应了递推式的初值。从公式中可以得出当 m[i, j]达到最小值时，子问题的优化函数值 m[i, k]和 m[k+1, j]也是最小的。如若不然，假设存在 m'[i, k]小于 m[i, k]， 那么必然存在 m'[i, j]= m'[i, k]+m[k+1, j]比 m[i, j]小。

伪码实现：

*目标：*计算矩阵连 $A_{i…j}$ 所需要的最少乘法运算次数的函数

*输入：*P=<p_{i-1}, p_i, …, p_j>

*输出：*最少乘法运算次数 m[i,j]

伪码：

```
令所有 m[i,j]初值为 0
for r = 2 to n:                                  //r 为计算的矩阵链的长度
    for i = 1 to n-r+1:                          //n-r+1 为最后一个长 r 的链的起始位置
        j=i+r-1                                  //计算链 i-j
        m[i,j]=m[i+1,j]+pi-1*pi*pj               //Ai(Ai+1...Aj)
        for k = i+1 to j-1:                      //k 为分割位置
            cost = m[i,k] + m[k+1,j] + pi-1*pk*pj    //(Ai...Ak)(Ak+1...Aj)
            if cost < m(i,j):
                m[i,j] = cost
```

复杂度：算法时间复杂度 $O(n^3)$, 空间复杂度是 $O(n^2)$