第三章　多序列比对

多序列比对问题是研究如何比对两个以上的序列。一方面，多个序列之间的比对往往是分子生物学和比较基因组学每个研究课题的起点。另一方面，迄今为止我们还没有找到快速地、准确地计算上千个蛋白序列之间比对的算法。作为一个尚未完全解决的问题，多序列比对目前仍然是生物信息学中最重要的研究课题之一。在这一章，我们将重点介绍多序列比对的基本概念、方法和普遍使用的程序。

§3.1　为什么需要比对多个生物序列?

具有相似三维结构及性质的蛋白质组成一个蛋白质簇。目前，依据结构域 (参见 §3.2) 的各种性质，将蛋白质划分成大约六万个互相重叠的簇。例如，同源异型蛋白质簇中每个蛋白质都包含一个由六十一个氨基酸组成的同源结构区域 (homobox domain)。这些蛋白质在动物发育中起着重要作用。一些蛋白质簇包含数十个，甚至上千个蛋白质成员。属于这些簇的一些蛋白质序列往往有较低的相似度。因此，在根据序列的性质判断一个蛋白质是否属于这些簇时，多序列比对要比双序列比对更有效。PSI-BLAST 正是迭代的使用输出序列之间的比对提高了其同源体搜索的准确率。

蛋白质的一些功能元区域仅含几个氨基酸 (图 3.1)。在双序列比对中看到连续几个氨基酸相同的概率相对较大，只有多个序列在这么短的区域上高度相似才在统计上具有显著性。类似地，基因转录因子的结合位点所在的启动子区域也很短，但往往有极强的模式。所以，识别启动子区域也需要多序列的比对或者与多序列比对有关的指标。

进化理论告诉我们所有物种都来自同一祖先。这意味着同一基因在不同生物体中的序列都来自它们的祖先序列。在进化过程中，生物体中的基因序列不断发生各种各样的突变。来自两个物种的同一蛋白质的基因编码序列越相似，这两个物种之间的进化距离就越小。因此，自从发明了蛋白质和 DNA 的测序技术之后，生物学家便迅速地将进化的研究推进到分子水平。多个 DNA 或氨基酸序列之间的比对便成了建立分子系统进化树的起点。所以，多序列比对的方法和工具可以说是比较基因组学和进化基因组学的基石。

§3.2　模体、谱、共识序列

肽链必须折叠成特定的三级结构才能有生物活性，发挥蛋白质分子的功能。蛋白质在局部区域里的肽链往往形成规则的形状。这些最常见的形状有 α 螺旋和 β 片。在一个 α 螺旋结构中，一个肽链的骨架围绕一个中心轴通过由氨基酸残基上的 -C=O 与后四位上的氨基酸残基上的 -NH 之间形成的氢

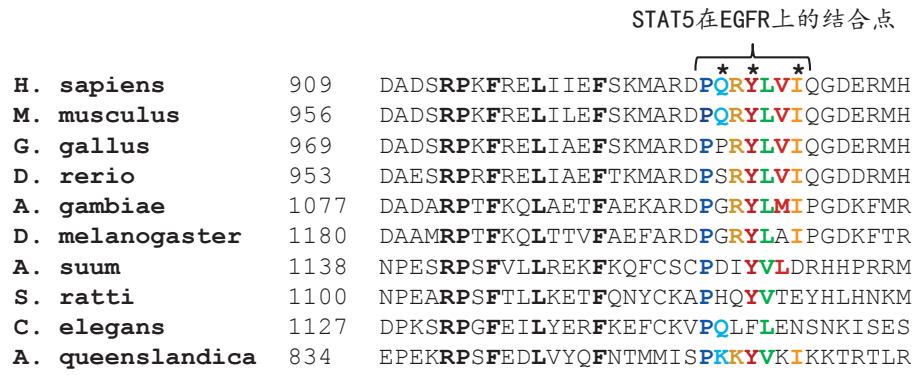


图 3.1: 在磷酸化过程中, 信号转导和转录激活 (STAT) 蛋白质 5B 及其同源蛋白质在表皮生长因子受体 (EGFR) 序列上的结合位点仅有 7 个氨基酸 (方括号内)。图中所示的多序列比对包含 10 个 EGFR 序列片段, 自上而下分别来自人、小鼠、鸡、斑马鱼、疟蚊、果蝇、猪蛔虫、鼠类圆线虫、秀丽隐杆线虫和海绵。

键以螺旋的形式伸展。β 片又称为 β- 折叠结构。它由多个伸展的肽链 (3~10 个氨基酸) 组成。在一个 β- 片上, 肽链按平行或反平行排列, 并且一个肽链主链上的 C=O 与其相邻肽链上的 N-H 之间形成氢键。氢键与肽链的长轴近于垂直, 用于维持折叠片的稳定。

α 螺旋和 β 片中的肽链对应的氨基酸序列段由于其重要的生物功能在进化过程中很少发生突变。这些氨基酸片段称为蛋白质的结构域。一些蛋白质只有一个结构域。但大部分蛋白质有多个结构域组成。

蛋白质的结构域和其他功能元区域的序列高度相似。这些保守区域在同源蛋白序列中的小序列片段集通常用一个模体 (motif) 来表示。模体按下列规则构成:

单个字符: 表示一个位点只允许这个氨基酸出现, 除 B、J、O、U、X 以外, 每个 A 到 Z 的字符代表一个氨基酸。

x: 表示一个位点允许任何一个氨基酸出现。

{…}: 表示一个位点允许除大括号里氨基酸以外的任何一个出现。

[…]: 表示一个位点只允许方括号里的氨基酸出现。

(n): 表示连续 n 个位点。

-: 氨基酸位点之间的间隔符。在一些文献里, 这个间隔符省略不写。

注意当 {} 不写任何氨基酸时, 它表示在某个位点上允许任何氨基酸, 和 x 的功能一样。当 (n) 被放在前四个符号之后表示, 它表示每个位点都符合同一

规律。例如，A(3) 表示 A 连续出现在三个位点上，和 A-A-A 等价。可以看出，这些规则非常灵活。不同的模体可以代表同一个序列集合。

例如锌指 (Zinc-finger) 结构域的模体是

```
C-x-H-x-[LIVMFY]-C-x(2)-C-[LIVMYA]
```

它表明锌指结构域是一个由 10 个氨基酸组成的保守区域。第一和九个氨基酸是半胱氨酸 (C)；第二、四、七、八位点上可以是任意氨基酸；第三个氨基酸是组氨酸 (H)。第五个氨基酸有多种选择。它可以是亮氨酸 (L)、异亮氨酸 (I)、缬氨酸 (V)、甲硫氨酸 (M)、苯丙氨酸 (F)、或酪氨酸 (Y)。同样，第十个氨基酸可以是亮氨酸 (L)、异亮氨酸 (I)、缬氨酸 (V)、甲硫氨酸 (M)、酪氨酸 (Y)、或丙氨酸 (A)。

再例如，根据图 3.1，脊椎动物信号转导和转录激活蛋白 5B 在表皮生长因子受体上的结合位点的模体是`P-[QPS]-Y-L-V-I`。

使用模体表示保守序列的好处是直观、简洁。但它并不适用于表示保守性较低或者较长的结构域。这些结构域序列特征常常使用一个谱 (profile) 来表示。谱概括一组蛋白质序列在若干连续位点上氨基酸分布的具体信息。它通常以数字矩阵的形成给出，其中每列对应一个氨基酸或者空字符 '-'。例如，4 个氨基酸组成的保守区域的谱有下列形式：

	A	C	D	E	F	G	H	I	L	K	M	N	P	Q	R	S	T	V	W	Y	-
P1	1	0	7	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1
P2	0	6	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2
P3	0	0	0	0	0	7	0	0	0	0	2	0	0	0	0	0	0	0	0	0	1
P3	0	0	0	0	0	0	9	0	0	0	0	0	0	0	0	0	0	0	0	0	1

其中每一行对应一个位点。每个数表示对应氨基酸或 '-' 在对应位点上出现的频率。例如，第一行的数据表明在第一个位点上，丙氨酸、亮氨酸和 '-' 各出现在一成的序列上；天冬氨酸则出现在七成的序列上。

蛋白质结构域上的谱往往总结了一个蛋白质家族的序列特征。它一般从多序列比对得出。谱不但可以提高同源序列查询的精确度，还可以用于多个序列的比对。

数据库 PROSITE(prosite.expasy.org) 收录了上千个蛋白结构区域的模体和谱。它还提供了三种不同的分析工具：(1) 它允许利用数据库分析蛋白序列中的结构域和模体；(2) 它也可以查询包含一个模体的蛋白质序列；(3) 它还可以检查是否给定的模体出现在给定的蛋白序列中。

最后，蛋白质结构区域的共识序列 (consensus sequence) 由在每个位点上出现次数最高的字符组成。不难看出，作为谱的一个简化形式，一个蛋白质结构区域的共识序列也是对序列保守性的另一种刻画。

上面给出的这些概念也广泛用于概括 DNA 序列中的保守区域，例如基因转录因子的结合区域和基因序列里内显子与外显子的分界点附近的保守区

域。

§3.3 Logo：一个序列保守区域的可视化方法

徽标是将几种不相干的元素混搭在一起形成的一种具有象征意义的东西。最为我们熟知的莫过于 2008 年北京奥运会会徽。它有三部分构成：1. 象证一个人的 “京” 字中国印；2. “Beijing 2008” 的字样，记录了该届奥运会的地点和时间；3. 奥运五环，象征奥运会是五大洲共同参与的体育盛会。

徽标又叫标识。序列标识图 (图 3.2) 是显示序列保守区域的共识序列、每个位置上各个氨基酸或核苷酸出现的频率以及各个位点上的序列信息量的一种可视化方法。

Logo 程序 (Schneider & Stephens, 1990) 根据序列保守区域的多序列比对来绘制它的序列标识图。在一个标识图像里，由大小不一的字符形成的一个堆栈代表序列保守区域的一个位点。每个核苷酸或氨基酸的高度和它在对应位点上出现的频率成比例。堆栈的总高度代表对应位点上的序列信息，以比特（bit）为单位。在每个堆栈里，字符按其出现的频率大小自上而下排列。所以，位于各个堆栈最上方的字符组成保守区域的共识序列。

在 WebLogo 程序 (Crooks et al., 2004) 绘制的序列标识图里，每个字符按其化学性质着色。对于 DNA/RNA 序列，核苷酸 A、G、C、T(U) 的默认色分别是绿、橘、蓝、红。对于蛋白质序列，带极性侧链的氨基酸 (G、S、T、Y、C、Q、N) 着绿色； 碱性氨基酸 (K、R、H) 着蓝色； 酸性氨基酸 (D、E) 着红色；疏水氨基酸 (A、, V、L、I、P、W、F、M) 着黑色。此外，用户也可以根据自己的需求配色。

Schneider 和 Stephens 将多序列比对中一个位点上的信息，B, 定义为熵的最大值减去该列上字符分布的熵：

$$B = \log_2 N - \left(- \sum_{j=1}^{N} p_j \log_2 p_j \right)。 \tag{3.1}$$

这里，p_j 是字符 j 在该位置上出现的频率；N 等于 4(DNA/RNA 序列)，或者等于 20(蛋白质序列)。在字符符合一致分布的假设下，仅含一个字符的比对列具有最大的信息，分别是 $\log_2 4$ (=2) (DNA/RNA 序列) 和 $\log_2 20 (\approx 4.32)$ (蛋白质序列) 比特。当序列比对包含少于 20 个 DNA/RNA 序列或者 40 个蛋白质序列时，$\left(- \sum_{j=1}^{N} p_j \log_2 p_j \right)$ 会系统性地低估一个位点上的熵值。因此，在计算一列的信息量时，Logo 程序在公式 (3.1) 的右边还添加了一个校正项。在此，我们不加以详细讨论。

目前，序列标识图已广泛用于发现和显示各种序列保守性。它可以用于显示 BLOCKS 蛋白序列数据库里的比对和 DNA- 结合点的模体。它还可以用于分析基因转录的剪接位点。

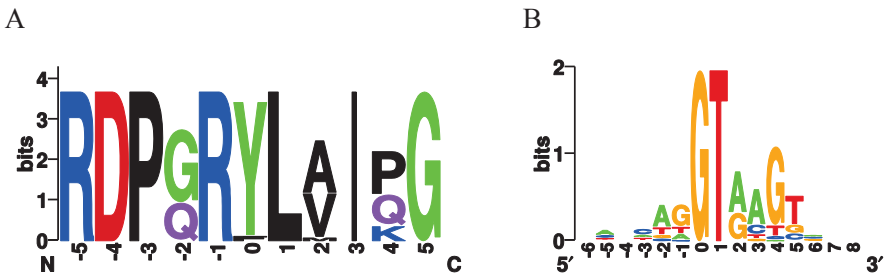


图 3.2: 序列标识图的例子。(A) 在磷酸化过程中, 信号转导和转录激活 (STAT) 蛋白质 5B 及其同源蛋白质在表皮生长因子受体 (EGFR) 序列上的结合区域 (图 3.1) 的标识图。(B) 外显子和内显子分界保守区域的标识图。N-C (A) 和 5′-3′ (B) 均表示序列的方向。

§3.4 多序列比对的 SP 分数

在基本空位罚分法则下，一个双序列比对的得分是所有列分数之和。同样地，我们可以考虑把一个多序列比对的得分定义为列分数的和。但是，这种方法是不切实际的。因为所有可能的比对列的数目随比对所涉及序列的数目成指数倍增长。如果要比对 1000 个 DNA 序列，单单给出所有的 $(5^{1000}-1)$ 个可能列的分数就是一项不可能完成的任务。

多序列比对的每列本身可以看成一个序列。另一种尝试是使用每列的熵定义该列的分数。考虑 k 条蛋白质序列之间的比对中的一列。假设氨基酸 a 在该列中出现 n_a 次，我们可以定义这个列的分数为 $\sum_a \frac{n_a}{k}\log_2\left(\frac{n_a}{k}\right)$，这里我们约定 $x\log x = 0$。如此定义的列分数很容易计算。但是，这种定义的问题是不知道如何快速的计算一组序列的最优比对。

当前最常用的计分法则是考虑所有列上的字符对。设 $(a_1, a_2, \cdots, a_m)$ 是 m 条序列之间的某个比对，$\mathcal{A}$，中的一列。这个列一共包含了 $\binom{m}{2}$ 字符对 $\binom{a_k}{a_j}$，$k<j$。这些字符对的分数 $s(a_k, a_j)$ 之和

$$s(a_1,a_2)+s(a_1,a_3)+\cdots+s(a_k,a_j)+\cdots+s(a_{m-1},a_m)$$

是该列的得分，其中 $s(-,-)=0$。$\mathcal{A}$ 的分数 $s(\mathcal{A})$ 定义为其中所有列分数的总和。考虑下列比对，M：

```
G C - T C
A - A - C
G - A T C
```

假设匹配列、非配列和增减列的分数分别是 3、-2 和 -1. 在第一列里有 3 个碱

基对:

$$\binom{\mathtt{G}}{\mathtt{A}}, \binom{\mathtt{G}}{\mathtt{G}}, \binom{\mathtt{A}}{\mathtt{G}}。$$

它们的分数分别是 -2、3 和 -2。因此，第一列的分数是 $-2+3-2=-1$。类似地，第二列至第五列的分数分别为 -2，1，1，9. 把这五个列分数加起来，我们得出 M 的得分是 8.

对于两条序列之间的比对，上述计分法则和前面讨论双序列比对时使用的基本空位计分法则一致。对于 m 条序列 $S_1, S_2, \cdots, S_m$ $(m \geq 3)$ 之间的一个比对 $\mathcal{A}$：

$$\begin{array}{cccccc} a_{11} & a_{12} & a_{13} & \cdots & a_{1(n-1)} & a_{1n} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2(n-1)} & a_{2n} \\ \vdots & \vdots & \vdots & \cdots & \vdots & \vdots \\ a_{m1} & a_{m2} & a_{m3} & \cdots & a_{m(n-1)} & a_{mn} \end{array} \tag{3.2}$$

仅含第 k 行和第 j 行的 $2\times n$ 子矩阵 $\mathcal{A}(k,j)$ 叫 $\mathcal{A}$ 在第 k 条和第 j 条序列上的投射。把 $\mathcal{A}(k,j)$ 里仅含空字符的列删除便得到第 k 条和第 j 条序列之间的一个比对 $\mathcal{A}'(k,j)$，我们称其为一个诱导比对。因为 $s(-,-)=0$，

$$\mathrm{SP}(\mathcal{A}) = \sum_{1\leq l\leq n}\sum_{1\leq k<j\leq m} s(a_{kl}, a_{jl}) = \sum_{1\leq k<j\leq m}\sum_{1\leq l\leq n} s(a_{kl}, a_{jl}) = \sum_{1\leq k<j\leq m} s\big(\mathcal{A}'(k,j)\big)。 \tag{3.3}$$

因此，这种计分方法被称为 Sum of Pairs (SP) 计分法则。回到上述比对 M。它在前两个序列上的投射是：

```
G C - T C
A - A - C
```

在这个投射里，没有具有两个空字符的列。所以，它本身就是前两个序列之间的一个比对，其分数是 -2。M 在第一序列和第三序列上的投射也是这两个序列之间的一个比对，其分数是 7。M 在后两个序列上的投射是：

```
A - A - C
G - A T C
```

其中第二列含两个空字符。去掉第二列后得到下列比对:

```
A A - C
G A T C
```

其分数为 3。不难看出，这三个诱导出的双序列比对的分数之和也应是 8。

SP 计分方法的最大好处是我们可以很容易地将双序列比对算法推广到多序列比对。在这章的其余部分，我们的讨论都将围绕着 SP 计分法则进行。

§3.5 多序列比对的复杂性

§3.5.1 动态规划算法

在第一章里我们已经给出了计算两序列之间最优比对的动态规划算法。现在，我们将这个算法推广到多序列的情形。考虑字母表 Σ 上的三条序列：

$$\begin{aligned} X:&\quad x_1x_2\cdots x_l,\\ Y:&\quad y_1y_2\cdots y_m,\\ Z:&\quad z_1z_2\cdots z_n。\end{aligned}$$

为了在 SP 打分法则下求这三条序列之间的最优比对，我们假设 $\mathrm{SP}(i,j,k)$ 是 $X[1,i]$、$Y[1,j]$ 和 $Z[1,k]$ 之间的最优比对的得分。再令 $\mathcal{A}(i,j,k)$ 是 $X[1,i]$、$Y[1,j]$ 和 $Z[1,k]$ 之间的一个最优比对，即它的分数是 $\mathrm{SP}(i,j,k)$。注意到 $\mathcal{A}(i,j,k)$ 的最后一列

$$\begin{pmatrix} a\\ b\\ c\end{pmatrix}$$

只有下面 7 中可能：

$$\begin{pmatrix} x_i\\ -\\ -\end{pmatrix},\ \begin{pmatrix} -\\ y_j\\ -\end{pmatrix},\ \begin{pmatrix} -\\ -\\ z_k\end{pmatrix},\ \begin{pmatrix} x_i\\ y_j\\ -\end{pmatrix},\ \begin{pmatrix} x_i\\ -\\ z_k\end{pmatrix},\ \begin{pmatrix} -\\ y_j\\ z_k\end{pmatrix},\ \begin{pmatrix} x_i\\ y_j\\ z_k\end{pmatrix}。$$

定于 0-1 函数 f 如下:

$$f(a)=\begin{cases} 1, & a\in\Sigma,\\ 0 & a=\text{‘–’}。\end{cases}$$

不妨假设 $\mathcal{A}(i,j,k)$ 有 t 列。因为 $\mathcal{A}(i,j,k)$ 是 $X[1,i]$、$Y[1,j]$ 和 $Z[1,k]$ 之间的最优比对，$\mathcal{A}(i,j,k)$ 的前 $(t-1)$ 列一定是 $X[1,i-f(a)]$、$Y[1,j-f(b)]$ 和 $Z[1,k-f(c)]$ 之间的最优比对。于是，对于任何 i、j 和 k, $\mathrm{SP}(i,j,k)$ 满足下列递推关系：

$$\mathrm{SP}(0,0,0)=0; \tag{3.4}$$

$$\mathrm{SP}(i,j,k)=\max\begin{cases} \mathrm{SP}(i-1,j,k)+\mathrm{SP}(x_i,-,-),\\ \mathrm{SP}(i,j-1,k)+\mathrm{SP}(-,y_j,-),\\ \mathrm{SP}(i,j,k-1)+\mathrm{SP}(-,-,z_k),\\ \mathrm{SP}(i-1,j-1,k)+\mathrm{SP}(x_i,y_j,-),\\ \mathrm{SP}(i-1,j,k-1)+\mathrm{SP}(x_i,-,z_k),\\ \mathrm{SP}(i,j-1,k-1)+\mathrm{SP}(-,y_j,z_k),\\ \mathrm{SP}(i-1,j-1,k-1)+\mathrm{SP}(x_i,y_j,z_k)\end{cases} \tag{3.5}$$

其中我们约定当 i、j 或 k 小于零时 $\mathrm{SP}(i,j,k) = -\infty$。在 (3.4) 的右边，每行中的最后一项是对应列的 SP 分数。例如，$\mathrm{SP}(x_i, y_j, -) = s(x_i, y_j) + s(x_i, -) + s(y_i, -)$。

和双序列情形类似，我们利用 (3.4) 可以求出 $\mathrm{SP}(l,m,n)$。追溯计算 $\mathrm{SP}(l,m,n)$ 的流程，我们可以列出至少一个 X、Y 和 Z 之间的最优比对。

类似地，如果我们考虑 Σ 上 s 条序列上的比对，公式 (3.4) 可以推广为：

$$\begin{aligned}&\mathrm{SP}(l_1, l_2, \cdots, l_s)\\ =\ &\max_{(\sigma_1,\sigma_2,\cdots,\sigma_s)\in\Sigma_+^t} \mathrm{SP}(l_1 - f(\sigma_1), l_2 - f(\sigma_2), \cdots, l_s - f(\sigma_s)) + \mathrm{SP}(\sigma_1, \sigma_2, \cdots, \sigma_s),\end{aligned}$$

其中 $\Sigma_+^t = (\Sigma + \{-\})^s / \{(-,-,\cdots,-)\}$，即所有可能的比对列的集合。虽然这个算法不复杂，但它并不实际。这是因为当 $s \geq 100$ 时它的时间复杂性是 $O(l_1 l_2 \cdots l_s) \geq 2^{100}$。换句话讲，这个算法在比对 100 条序列时所需要的时间要用亿年来计时。

§3.5.2 NP- 难解性

最大公约数问题是求任意给定的两个正整数的最大公约数。我们在小学就知道辗转相除法是解决这个问题的一个算法：

```
int LCD(int a, int b){
    int c;

    if (a < b) LCD(b, a);
    c= a%b;
    if (c==0) return b; else LCD(b, c);
}
```

在这个算法里，指令行 $c = a\%b$ 是将 a 除以 b 所得到的余数赋给变量 c. 当输入整数是 14 和 8 时，第一个条件指令行被跳过；第二个指令行执行完后，c 的值是 6；然后，通过调用 LCD（8，6）执行第三行。在运行 LCD（8，6）时，LCD（6，2）被调用，最后输出 2. 不难看出，这个算法所需要的算术运算的次数和两个输入数的大小有关。也就是说，它所需要的运算次数是两个输入数的规模的函数。

上述事实对其它问题及其任意一个算法也成立。在研究算法分析和设计时，我们把存储具体问题所需要的空间定义为这个问题的规模。例如，一个最大公约数问题的输入是两个整数 a 和 b。一个整数在计算机里是以二进位的方式保存的。因此，具体最大公约数问题 (a,b) 的输入的规模应是 $\log_2 a + \log_2 b$ 比特 (bits)，而不是 $a + b$! 同样地，在讨论数论问题的算法时，我们将比特运算 (1+1=10, 1+0=1 等）视为基本运算。

数独游戏也可以看作是一个离散算法问题。每一个具体的数独问题有一些从 1 至 9 的整数出现在一个 9×9 的方阵里 (图 3.3)。这个游戏要求在每个

8		9		4			2	5
		5		7				6
		7		5	8			
6			4			8	7	
				3				
	7	1			5			3
			2	8				
7			5	6		1		
2	9			1		5	8	

图 3.3: 数独问题的具体例子。

空白方格上填上一个数使得每一行、每一列、以及每个 3×3 的方盒里恰巧包含 9 个不同的数字。这个问题的输入规模是多少呢？每个具体问题包括最多 81 个数，以及每个数需要 4 个比特。所以，如果我们约定从左到右，从上到下依次将每个数保存在计算机的内存里，每个具体数独问题的规模是 324 个比特。

考虑某个问题的一个算法。设 $f(X)$ 是该算法在规模为 n 的具体实例 X 上运行所需要的基本运算的次数。我们称

$$F(n) = \max_{X\in S_n} f(X)$$

为这个算法的时间复杂性函数，其中 S_n 是所有规模小于 n 的具体实例的子集。如果存在和 n 无关的常数 c 使得，对于任意大的 n, $F(n)\le cn$，我们称该算法是线性时间算法。更一般地，如果存在和 n 无关的 c 和 k 使得，$F(n)\le cn^k$，我们称该算法是 (k 次) 多项式时间算法。

回到辗转相除法算法。每次调用辗转相除法程序 `LCD(,)` 时，计算 `c` 的指令行最多需要 $\log_2 a$ 次比特运算；并且每调用一次，参数的最大值降低至少一倍，即 $a_2\le a_1/2$。概括起来，输入 a,b 后，辗转相除算法在停止之前最多执行 $(\log_2 a)^2$ 次运算。因为 (a,b) 的规模是 $\log_2 a+\log_2 b$, 辗转相除算法是一个二次多项式时间算法。同样道理，我们在日常生活中使用的整数加法算法是线性时间算法。

有多项式时间算法的问题称为多项式时间可解。多项式时间可解的问题简称为 P- 问题。上面讨论说明最大公约数问题是 P- 问题。不难看出，双序列比的动态规划算法是二次多项式时间算法。从而双序列比对问题也是 P- 问题。

许多实际计算问题是 P- 问题。也有非常多的棘手问题不是 P- 问题。还有许多问题冒似 P- 问题，但不能证明它们是 P- 问题。在计算机模型和理论研究中，理论计算机学家利用不确定图灵机模型定义了一类更广泛问题，称

之为 NP- 问题。如果一个问题的解的规模不大于问题本身的规模的一个多项式函数，那么这个问题是 NP- 问题。哈密尔顿回路问题是图论中的一个古老问题。这个问题是判断一个图里是否存在一个圈满足：它通过每个顶点并且每个顶点在这个圈中仅出现一次。因为存储一个圈所需要的空间资源比保存原问题的空间资源小，哈密尔顿回路问题是一个 NP- 问题。素数分解问题是另一个著名 NP- 问题。

如果把数独问题推广为在 $n^2 \times n^2$ 的一个方阵中不含数的方格里填上 1 到 n 的一个数使得每一行、每一列、以及每个 $n \times n$ 的方盒里恰巧包含 n 个不同的数字。这个广义数独问题也是 NP- 问题。其原因是解的规模和问题的规模一样。

证明是否每个 NP- 问题都有多项式时间算法是计算机理论乃至数学领域里的一个著名的公开问题，简称为 P =?NP。在做几何证明题时，我们常常添加几个辅助线，便把一个问题转化成另一个已知的命题。在设计算法时，我们也有类似地经验。有时候，我们可以将一个问题的实例转化成另一个问题的一个具有同样规模同样解的实例。在研究 P 是否等于 NP 时，计算机学家发现了一类难解的 NP- 问题，叫作 NP- 完全 (complete) 问题。(complete) 问题根据定义，如果一个问题本身是 NP- 问题并且它的一个多项式时间算法（如果存在的话）可以转换为任何其它 NP- 问题的多项式时间算法，那么这个问题是 NP- 完全的。虽然这个定义令人费解，成千上万个来自不同领域的计算问题都被证明是 NP- 完全问题。例如，哈密尔顿回路问题是 NP- 完全问题。广义数独问题也是 NP- 完全问题。

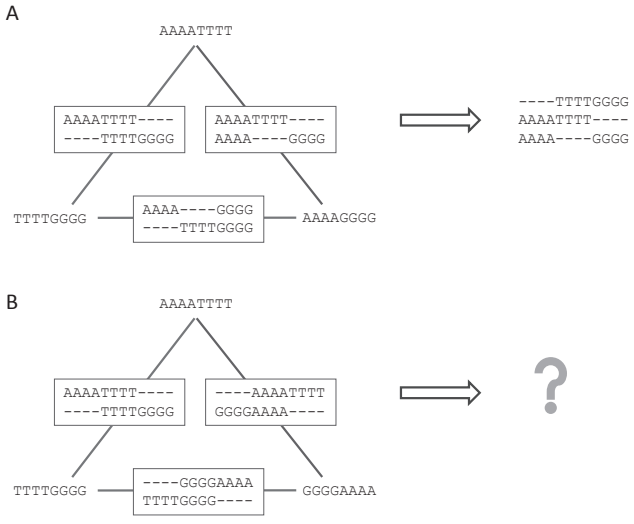


图 3.4: 双序列比对的相容性。(A) 双序列最优比对相容的一个例子。合并这些比对便得到这三条序列之间的最优比对。(B) 双序列最优比对不相容的一个例子。

回到多序列比对问题。如果我们考虑有 c 个字母的字母表，每个字母在计算机里可以用 $\log_2 c$ 比特的空间存储，和序列比对问题所涉及的序列长度无关。所以，一个序列的规模可以简单地假定为它的长度。一个具体多序列比对问题的规模就是各个输入序列 X_i 的长度 $|X_i|$ 之和。不妨设输入序列是 $X_1, X_2, \cdots, X_m$。因为比对矩阵中不含仅有空字符的列，这些序列之间的任何一个比对最多有 $\sum_i |X_i|$ 列。所以，比对问题的解的规模最多是 $m \times \sum_i |X_i|$，小于 $(\sum_i |X_i|)^2$。这证明了多序列比对是 NP- 问题。

考虑序列 $X_1, X_2, \cdots, X_m$ 之间的一个具体的比对 $\mathcal{A}$。对于任何的 i 和 j，$i < j$，$\mathcal{A}$ 诱导出的双序列比对 $\mathcal{A}(i,j)$ 可能是，也可能不是 X_i 和 X_j 之间的最优比对。反过来，如果所有 $\binom{m}{2}$ 个诱导出的双序列比对 $\mathcal{A}(i,j)$ 都是对应序列的最优比对，那么 $\mathcal{A}$ 一定是一个最优比对。在解决实际比对问题时，所考虑的序列常常不相容 (图 3.4)。也就是说，并不是任意给一些双序列最优比对，都存在一个给定序列之间的一个比对满足：它的诱导比对和给出的双序列比对相同。这正是多序列比对的难点所在。事实上，判断多序列上是否存在得分为 c 的比对也是一个 NP- 完全问题 (Wang & Jiang, 1994)。如果多序列比对问题是多项式时间可解的，那么，所有的 NP- 问题也都有多项式时间算法。换句话讲，P=NP! 这隐含着多序列比对问题不太可能有多项式时间算法。

对于一个 NP- 完全问题，研究者往往探索它是否有精确的多项式时间启发式算法或者是否有近似性好的近似算法。在过去的二十年内，多序列比对问题始终是生物信息学的热点。因为新一代测序技术使测序成本大大降低，以致于它已经成生物医学研究的主要手段，作者相信它还是未来十几年的热点。来自概率统计、机器学习和信号处理等不同学科的多种方法已经都被运用到多序列比对上。对序列比对当前研究的各种课题有兴趣的读者，可参见在本章结尾给出的有关参考文献。

§3.6 渐进式比对

§3.6.1 渐进式的基本策略

如果 X 和 Y 是字母表 Σ 上的两条序列，那么 X 和 Y 之间的比对其实是

$$\left\{\binom{\sigma_1}{\sigma_2}, \binom{\sigma}{-}, \binom{-}{\sigma} \mid \sigma_1, \sigma_2, \sigma \in \Sigma\right\}$$

上的一条序列。例如，

```
A A - C
G A T C
```

可以看作是含有下列 4 个字符一个序列：$\binom{A}{G}, \binom{A}{A}, \binom{-}{T}, \binom{C}{C}$。类似地，$k$ 条序列之间的比对是

$$(\Sigma \cup \{-\})^k = \{(\sigma_1, \sigma_2, \cdots, \sigma_n)^T \mid \text{存在 } i \text{ 使得 } \sigma_i \neq \text{`-'}\}$$

上的 “序列”，其中 T 是转置运算。基于此，m 条序列之间的一个比对可以看作是任何 k 行所形成的子比对和其它剩下的 $m-k$ 行所形成的子比对之间的比对 (图 3.5)。

由于多序列比对问题的 NP- 难解性，上述事实 “比对也是序列” 以及 “多序列比对是两个子比对的比对” 导出下列多序列比对的渐进式策略 (progressive strategy)：

- 从输入的序列集 $\mathcal{S}$ 里适当地选取两条序列 S_i 和 S_j，计算它们之间的一个最优比对 A_{ij}。
- 递归地从 $\mathcal{S} = \mathcal{S} - \{S_i, S_j\} + \{A_{ij}\}$ 计算出的输入序列之间的一个比对。

注意在第 k 递归步开始时我们有一些序列和子比对。因此，使用渐进式策略比对多个序列的基本步骤包括：

1. 在一个序列/比对组成的集合中选取两序列/比对进行比对。
2. 计算一条输入序列和一个 (多序列) 比对之间的比对。
3. 计算两个 (多序列) 比对之间的比对。

如何设计渐进式比对算法依赖于如何实现上述三个基本步骤。

我们首先考察如何在一系列序列/比对中选取两个进行比对。毫无疑问，我们会选取最相似的两个序列/比对。这是因为高相似度意味着在进化过程中它们之间所发生的突变少从而容易比对。我们可以用比对来衡量两条序列之间的相似度。但是，我们还不知道如何衡量两个比对之间的相似度。后者是设计渐进式比对算法的一个关键问题。

如何计算两个比对或者一个序列和一个比对之间的比对呢？如果我们把两个输入比对看作是两条 “序列”，每个输入比对中的一个列变成了一个 “字符”。如果我们知道如何给这些超级 “字符” 的匹配打分，我们至少可以使用动态规划算法计算比对的比对。由于输入比对可以包含数目非常大的、不一样的列，直接制定这样的打分矩阵不切实际 (见 §3.4)。所以，比对两个比对的关键还是怎样快速的对列与列的匹配打分以及如何快速的寻找比对和比对之间的最优比对。

§3.6.2 Feng-Doolittle 比对算法

在一九八七年，Feng 和 Doolittle 率先提出了第一个渐进式比对的程序。该程序使用两个比对中出现的序列之间的平均比对分数来衡量它们的相似性。具体地讲，假设 $\mathcal{A}$ 是 $S_1, S_2, \cdots, S_m$ 之间的比对以及 $\mathcal{B}$ 是 $S'_1, S'_2, \cdots, S'_n$ 之间的比对，$\mathcal{A}$ 和 $\mathcal{B}$ 的之间的相似性被量化为

$$\frac{1}{mn} \sum_{1 \le i \le m} \sum_{1 \le j \le n} s(S_i, S'_j),$$

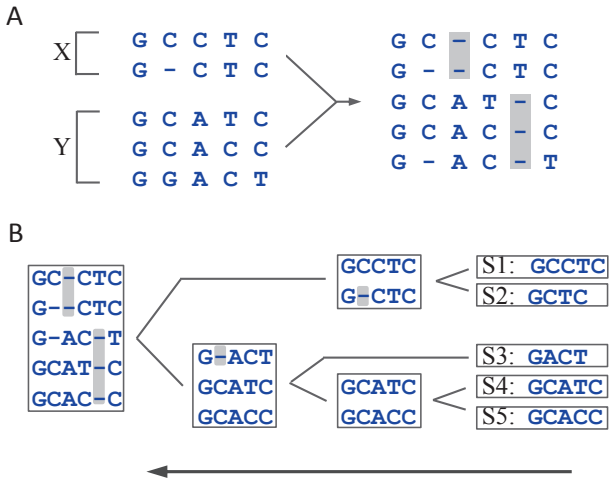


图 3.5: 渐进式比对策略的示意图。(A) 通过将空字符组成的空列适当地插入 X 及 Y 内的相邻列之间将 X 和 Y 加以比对。(B) 五条 DNA 序列上的一个渐进式比对过程。箭头表示这个过程的方向。

其中 $s(,)$ 是两序列之间的最优比对的分数。

更重要地，Feng 和 Doolittle 解决了如何将不同序列集上的两个比对合并起来，形成一个新的比对。考虑上面定义的比对 $\mathcal{A}$ 和 $\mathcal{B}$。首先确定 S_t $(1 \le t \le m)$ 和 $S'_{t'}$ $(1 \le t' \le n)$ 使得 S_t 和 $S'_{t'}$ 之间的最优比对的得分，$s(S_t, S'_{t'})$，最高，即 $s(S_t, S'_{t'}) = \max_{1 \le i \le m, 1 \le j \le n} s(S_i, S'_j)$。然后使用 S_t 和 $S'_{t'}$ 之间的一个最优比对，$\mathcal{R}$，合并 $\mathcal{A}$ 和 $\mathcal{B}$。我们知道，S_t 中的任何一个字符 a 出现在 $\mathcal{A}$ 中唯一的一列，记为 C_a；$S'_{t'}$ 中的任何一个字符 a' 出现在 $\mathcal{B}$ 中唯一的一列，记为 $C_{a'}$。基于这个事实，我们将 $\mathcal{R}$ 作为参照按照下列方法将 $\mathcal{A}$ 和 $\mathcal{B}$ 合并在一起得到比对 $\mathcal{C}$。

> 从左到右考虑 R 中的每一列 $\binom{a}{a'}$。如果 $a \neq$ '-' 且 $a' \neq$ '-'，我们将 C_a 放置在 $C_{a'}$ 之上得到 $\mathcal{C}$ 的一列。如果 $a =$ '-'，那么，$a' \neq$ '-'。我们把由 m 个空字符所组成 "空列" 放置在 $C_{a'}$ 之上得到 $\mathcal{C}$ 的一列。如果 $a' =$ '-'，那么，$a \neq$ '-'。我们将 C_a 放置在由 n 个空字符所组成 "空列" 上得到 $\mathcal{C}$ 的一列。最后，将 $\mathcal{A}$ 中不含 S_t 中任何字符的列放置在由 n 个空字符所组成 "空列" 之上形成 $\mathcal{C}$ 的一列。将 $\mathcal{B}$ 中不含 $S'_{t'}$ 中任何字符的列放置在由 m 个空字符所组成 "空列" 之下形成 $\mathcal{C}$ 的一列。

这种处理 $\mathcal{R}$ 中的增删列的方法称为 " 一旦空位，永远空位 " 法则。

假如我们要合并 S_1 和 S_2 之间的比对:

S_1 A T T G C C A T T - -
S_2 A T C - C A A T T T T

及 S_3 和 S_4 之间的比对:

S_3 A G T G C C A T T
S_4 A G C T T C - T T

我们令这两个矩阵分别为 U 和 V。首先，分别计算 S_1 和 S_3、S_1 和 S_4、S_2 和 S_3、S_2 和 S_4 的最优比对的得分。设下列 S_1 和 S_3 之间的最优比对，R，的得分最高：

S_1 A T - T G C C A T T
S_3 A - G T G C C A T T

我们将以 R 作为参照比对合并 U 和 V。在 R 中，S_1 的第一个字符 A 和 S_3 的第一个字符 A 匹配。因为这两个 A 分别出现在 U 和 V 的第一列 $\binom{A}{A}$，合并后所得到比对的第一列有 4 个 A 组成 (图 3.6)。比对 R 的第二列是一个增减列 $\binom{T}{-}$，我们使用“一旦空位，永远空位”法则得到合并后的第二列 (图 3.6)。依次类推，我们可以计算出合并后所得到的比对的第三到九列。比对 U 中的最后两列 $\binom{-}{T}$ 不含 S_1 中任何字符。所以，它在 V 中没有对应的列。添加空列后我们得到输出比对的最后两列 (图 3.6)。

Feng-Doolittle 比对是一个二次多项式时间的算法。在含 m 条序列的集合 S 上，它首先需要计算输入序列两两之间的最优比对。这需要

$$\sum_{X,Y\in S} O(|X|\times|Y|) = O\left(\sum_{X,Y\in S}|X|\times|Y|\right)$$

次运算，其中 $|\cdot|$ 是对应序列的长度。其次，它需要递归运行 $m-1$ 次。在每一次递归里，我们需要计算前一步得出的比对和其它比对的相似度。这需要最多 $O(m)$ 次运算因为我们已经计算了输入序列两两之间的最优比对的得分。然后，利用各个比对之间的相似得分选取两个比对进行合并。合并两个比对所需要的运算次数大约是它们的列数之和 ($\leq \sum_{X\in S}|X|$)。概括起来，Feng-Doolittle 比对的时间复杂性是 $O\left(\sum_{X,Y\in S}|X||Y| + (m-1)(m+\sum_{X\in S}|X|)\right)$。

§3.7 近似算法

§3.7.1 序列编辑距离

生物学家使用比对的得分衡量两个生物序列的相似性。两个序列越相似，它们的最优比对的分数越高。然而，在信息论和计算机科学里，科学家通常选择使用距离来比较序列。在研究纠错密码时，两个等长的密码字 (即 0-1 序列) 中同时拥有不同比特 (即 0 和 1) 的对应位的数目定义为这两个字的海明

图 3.6: Feng-Doolittle 算法之比对的合并。使用 S_1 和 S_3 之间的比对 R 合并比对 U 和 V。S_1 和 S_3 之间的竖线表示 R 中含两个非空字符的列。方框里的空列是使用 “ 一旦空位，永远空位” 法则的结果。

(Hamming) 距离。海明距离这个概念虽然在信息学和计算机科学中很重要，但仅适用于比较高度保守的、长度相同的基因或蛋白质序列片段。基因序列复制过程中的错误不仅导致核苷酸的替换，也导致核苷酸的删除和插入。因此，一个基因和其对应的蛋白质在不同物种里的序列长度往往不同。也就是说，一个基因或蛋白质序列中的第 k 个字符可以和其同源序列中任何一个位置上的字符对应。而海明距离则假设比较序列中同一位置上的字符相互对应。

在一九六六年，俄国数学家 Levenshtein 引入了序列的编辑距离 (edit distance)。他把删去一个字符、增加一个字符、 和单字符替换定义为基本编辑运算。两个序列之间的编辑距离等于把一个序列转换成另一个所需要的最少基本编辑运算的数目。例如，使用一个替换、一个删除和一个添加就可将 TATCCGT 转变成 TACCCTA(图 3.7)。这意味着这两条序列之间的编辑距离至多是 3。与海明距离不一样，编辑距离可以被用于比较任何两条序列。

编辑距离之所以得到广泛应用是因为它不仅直观而且容易计算。计算编辑距离其实是全序列比对的一个特例。对于任何两条序列 X 和 Y, 它们之间的编辑距离等于在匹配分为 0、非匹配和增删列的分均为 -1 的计分法则下它们的最优比对的得分的绝对值。

由于基本编辑运算对应序列比对所考虑的三种序列突变事件，编辑距离比海明距离更适合比较生物序列。但是，我们知道一个序列删除事件往往将一段而不是一个核苷酸去掉。我们可以使用广义的编辑运算定义序列的距离吗？其实，由广义的替换、删除和增加组成的序列转换是可计算理论中的一个研究对象，叫重写系统 (rewriting system)（Book & Otto, 1993)。可计算理论的研究告诉我们采用广义的编辑运算定义的距离是不可计算的，即不存在适用于计算任何两个序列之间距离的计算机程序。

§3.7.2 星型比对算法

假设我们要计算字母表 Σ 上的 m 条序列 $S_i(1 \le i \le m)$ 之间的最优比对。

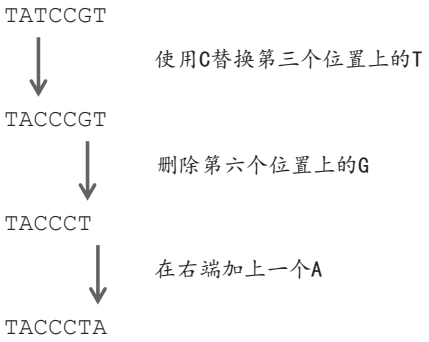


图 3.7: 三个基本运算可将 TATCCGT 转变成 TACCCTA。

定义函数 $g : (\Sigma \cup \{-\}) \times (\Sigma \cup \{-\}) \to \{0, 1\}$ 如下：

$$g(a,b) = \begin{cases} 0 & \text{if } a = b, \\ 1 & \text{if } a \neq b。 \end{cases}$$

对于 $\{S_i \mid 1 \le i \le m\}$ 上的任何一个比对 $\mathcal{A} = (a_{ij})_{m\times n}$，我们定义

$$E(\mathcal{A}) = \sum_{1\le i\le m} \sum_{i<i'\le m} \sum_{1\le j\le n} g(a_{ij}, a_{i'j})。$$

不难看出，$E(\mathcal{A})$ 也满足 (3.3) 给出的性质：

$$E(\mathcal{A}) = \sum_{1\le i<i'\le m} E\left(\mathcal{A}'(i,i')\right), \tag{3.6}$$

其中 $\mathcal{A}'(i,i')$ 是 $\mathcal{A}$ 诱导出的序列 i 和 i' 之间的一个比对。

不难看出，$E(\mathcal{A})$ 是编辑距离的推广。$\mathcal{A}$ 的比对分数越高，其对应的序列 S_i 越相似，从而 $E(\mathcal{A})$ 的值也越小。于是，从编辑距离的角度研究多序列比对时，比对问题则变为最小值问题，即寻找使 $E(\mathcal{A})$ 最小的比对 $\mathcal{A}$。

星型比对也是一种渐进比对算法。它首先计算所有两序列之间的编辑距离 $E(S_i, S_j)$ 并且选取一个序列 S_t 使得它到其它各序列的编辑距离的和最小，即：

$$\sum_{1\le j\le m} E(S_t, S_j) = \min_i \sum_{1\le j\le m} E(S_i, S_j)。 \tag{3.7}$$

如果有多个序列都具有上述的性质。我们在它们中间任意选取一个。然后，以 S_t 作为参考，将 $m-1$ 个和编辑距离 $E(S_t, S_j)$ 对应的比对 $\mathcal{A}(S_t, S_j)$ $(j \neq m)$ 按下列方法合并起来得到 $\{S_i\}$ 上的一个比对 $\mathcal{B}$ (图 3.8)。

对于 S_t 中的任何一个字符 a，如果 $\binom{a}{a_j}$ 是 $\mathcal{A}(S_t, S_j)$ 中含 a 的列，那么 $C_a = (a_1, a_2, \cdots, a_{t-1}, a, a_{t+1}, \cdots, a_m)$ 形成 $\mathcal{B}$ 的一列。对于任何

$\mathcal{A}(S_t, S_j)$ 中不含 S_t 中任何字符的插入列 $\binom{-}{a_j}$, 使用 “一旦空位，永远空位” 法则将它扩张成 $\mathcal{B}$ 的一列。在合并的过程中，对应列的次序保持不变。也就是说，对于 S_t 的第 i 和 $i+1$ 个字符 a 和 b, 在 $\mathcal{B}$ 中 a 所在的列，C_a, 一定出现在 b 所在的列, C_b, 的左边。而且由在 $\mathcal{A}(S_t, S_j)$ 中介于 a 和 b 所在列之间的添加列 $\binom{-}{c}$ 扩张得到的列在 $\mathcal{B}$ 中也以相同的次序出现在 C_a 和 C_b 之间。

我们考虑下列四条 DNA 序列：

S_1: AGTATGC,
S_2: ATCAGTGC,
S_3: ATAAGGC,
S_4: ACATCG.

使用全序列比对算法，我们可以得到它们两两之间的编辑距离 $E(S_i, S_j)$, 其矩阵假设为：

	S_1	S_2	S_3	S_4	Row sum
S_1	0	3	3	4	10
S_2	3	0	2	4	9
S_3	3	2	0	4	9
S_4	4	4	4	0	12

因为 S_2 到其它序列的编辑距离的和最小，我们通过选取 S_2 作为参考，将 S_2 和其它三条序列之间的最优比对：

```
S1   A G T - A - T G C
S2   A - T C A G T G C

S2   A T C A G T G C
S3   A T A A G - G C

S2   A T C A G T G C -
S4   A - C A - T - C G
```

合并起来 (图 3.8)。S_2 的第一个字符 A 在所要合并的比对中分别和 S_1、S_3 和 S_4 中的第一个字符 A 匹配，所以，输出比对中的第一列由四个 A 组成。S_2 的第二个字符 T 在所要合并的比对中分别和 S_1 中的第三个字符 T 以及 S_3 中的第二个字符 T 匹配。但在 S_2 和 S_4 的比对中，这个 T 和空字符形成一个增减列。所以，我们得到结果中的第三列由三个 T 和一个空字符组成。我们之所以把它列为第三列是因为, 在 S_1 和 S_2 的比对中, 这个 T 位于第三列。也就是说，在 S_1 和 S_2 的比对中，第二列是增减列，不含 S_2 中的任何字符。使用 “一旦空位，永远空位” 法则将这个增减列扩张成结果中的第二列。星型比对算法输出的比对如图 3.8 所示。

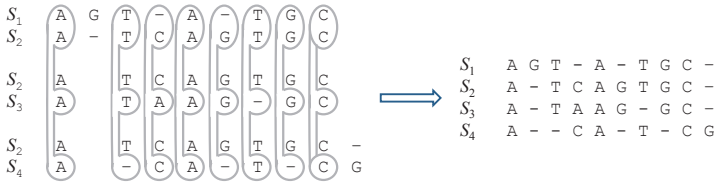


图 3.8: 以 S_2 为参考合并三个包含 S_2 的双序列比对。图中显示了如何利用参考序列 S_2 中的字符得出输出比对的列。在两序列比对中没被圈起来的增删列需要使用“一旦空位，永远空位”法则扩张成输出比对中的列。

定理 3.7.1 假设 $\mathcal{O}$ 是一组序列在编辑距离下的最优比对，即 $E(\mathcal{O})$ 的值最小。星型算法输出的矩阵 $\mathcal{M}$ 满足

$$E(\mathcal{O}) \le E(\mathcal{M}) \le 2\left(1 - \frac{1}{m}\right) E(\mathcal{O})。 \tag{3.8}$$

证明 (3.8) 中的第一个不等式可以从 $\mathcal{O}$ 是最优比对得出。为了证明第二个不等式，我们令 $E(i,j)$ 是 S_i 和 S_j 之间的编辑距离。最优比对 $\mathcal{O}$ 在序列 i 和 j 上的诱导出的比对 $\mathcal{O}'(i,j)$ 满足

$$E(\mathcal{O}'(i,j)) \ge E(i,j)。 \tag{3.9}$$

因为 $E(i,j)$ 是从最优比对得出的距离。不失一般性，假设在执行星型算法时我们选取 S_1 作为参考序列。于是，对于任何的 i,

$$\sum_{1\le j\le m} E(i,j) \ge \sum_{1\le j\le m} E(1,j)。 \tag{3.10}$$

于是，利用 (3.9) 和 (3.10)，我们有：

$$\begin{aligned} 2E(\mathcal{O}) &= \textstyle\sum_{1\le i\le m}\sum_{1\le j\le m} E(\mathcal{O}'(i,j)) \\ &\ge \textstyle\sum_{1\le i\le m}\left[\sum_{1\le j\le m} E(i,j)\right] \\ &\ge \textstyle\sum_{1\le i\le m}\left[\sum_{1\le j\le m} E(1,j)\right] \\ &= m\textstyle\sum_{1\le j\le m} E(1,j)。 \end{aligned} \tag{3.11}$$

因为序列 S_1 被选为参照序列，根据星型算法合并比对的方法，$\mathcal{A}$ 在 S_1 和 S_j 上的诱导比对 $\mathcal{A}'(i,j)$ 是合并时使用的它们之间的最优比对。于是，

$$E(\mathcal{A}'(1,j)) = E(1,j),$$

利用距离的三角不等式性质，我们进一步有

$$E(\mathcal{A}'(i,j)) \le E(\mathcal{A}'(1,i) + E(\mathcal{A}'(1,j)) = E(1,i) + E(1,j)。 \tag{3.12}$$

使用 (3.12),

$$\begin{aligned} 2E(\mathcal{M}) &= \sum_{1\le i\le m}\sum_{1\le j\le m} E(\mathcal{M}'(i,j)) \\ &\le \sum_{1\le i\le m}\sum_{1\le j\le m}(E(1,i)+E(1,j)) \\ &= 2(m-1)\sum_{1\le j\le m} E(1,j)。 \end{aligned} \tag{3.13}$$

结合 (3.11) 和 (3.13)，我们得到 (3.8) 中的第二个不等式：

$$E(\mathcal{M}) \le (m-1)\sum_{1\le j\le m} E(1,j) \le 2(m-1)E(\mathcal{O})/m = 2(1-1/m)E(\mathcal{O})。$$

□

上述定理表明星型算法虽然简单但它的近似性比较好。

§3.8 多序列比对实用程序

§3.8.1 ClustalW

最常用的多序列比对程序要属 ClustalW。读者可以在欧洲生物信息研究所的网页 (www.ebi.ac.uk/clustalw) 上运行这个程序; 也可以通过从 Clustal 程序的主页 (www.clustal.org) 上下载安装 ClustalW(命令行版本) 或 ClustalX(GUI 版本) 在自己的电脑上运行这个程序。ClustalX 还提供了序列比对的编辑器。

这里，我们通过比对 STAT5B 分别在人、小鼠、鸡、斑马鱼以及它们在果蝇和海绵中的同源序列我们来说明如何使用 ClustalW (图 3.9)。首先，我们将这六个蛋白序列以 FASTA 的格式拷贝到输入序列栏里。ClustalW 允许读者设置一些双序列和多序列比对的参数以便得到好的结果。双序列比对的参数包括打分矩阵，空位基本罚分以及每个增删列的平均罚分。多序列比对的参数除打分矩阵和空位罚分参数以外还有程序本身迭代次数、建构进化树的方法 (参见第五章) 和比对输出的格式。ClustalW 的算法有三个步骤 (图 3.10)。

ClustalW 首先使用动态规划算法计算输入序列两两之间的最优序列比对。在我们这个例子里，一共要考虑十五对序列之间的比对。假设有 k 个输入序列，ClustalW 要计算 $\binom{k}{2}$ 对序列的比对。

其次，ClustalW 将所得到的双序列比对再转化为对应序列之间的距离。在最初的版本里，该程序利用相当简单的方式计算两个序列，X 和 Y，之间的距离，$d(X,Y)$。假设在第一步里得到的 X 和 Y 之间的序列比对是 A。如果 A 包含 n_1 个匹配列和 n_2 个非匹配列，那么

$$d(X,Y) = \frac{n_1}{n_1+n_2}。$$

另外一种方式是把 A 的得分 s_A 转换成 X 和 Y 之间的距离。令 s_X 和 s_Y 分别是 X 和 Y 和它们自身的比对得分。再令 $s_{\text{average}} = (s_X + s_Y)/2$，以及

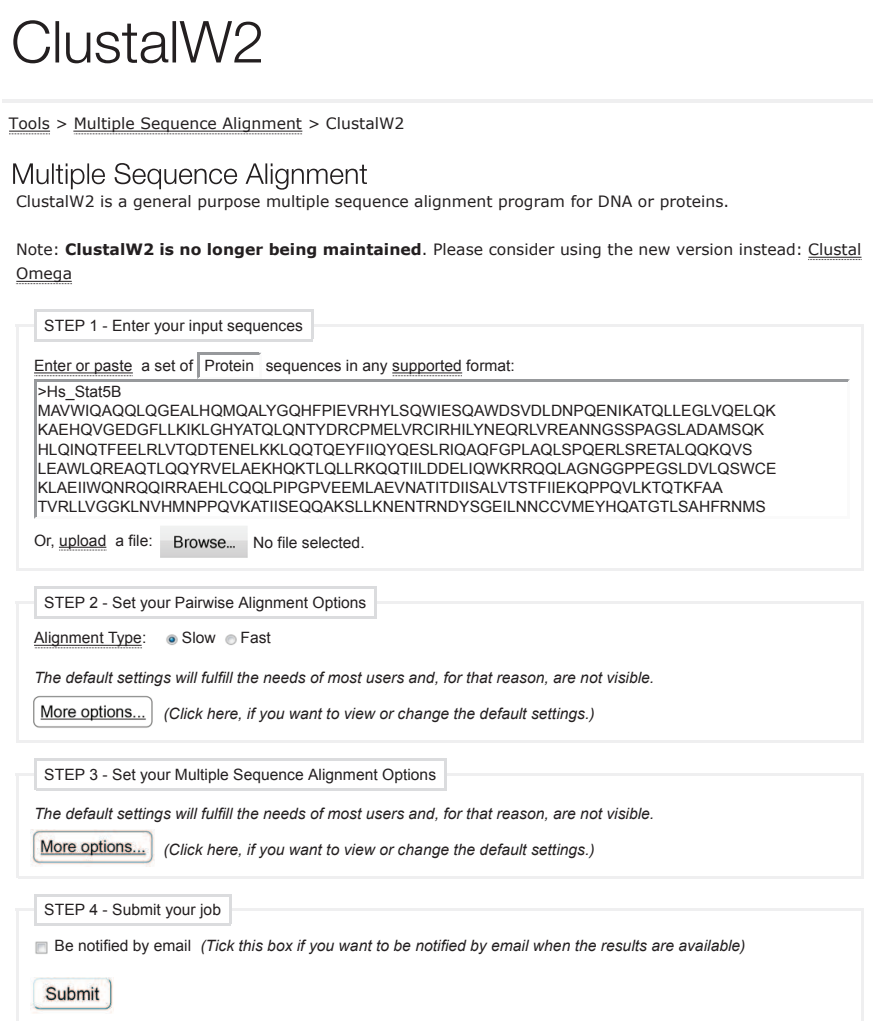


图 3.9: 在 http://www.ebi.ac.uk/Tools/msa/clustalw2/上使用 ClustalW 比对六个 STAT 蛋白序列。所使用的六个 STAT 蛋白序列分别是人 STAT5B(NP_036580.2)、小鼠 STAT5B(NP_033310.2)、鸡 STAT5B(NP_990110.1)、斑马鱼 STAT5.2(NP_001003984.1)、果蝇 STAT(AAS65180.1) 和海绵 STAT(XP_003385278.1)

s_{rand} 是分别和 X 以及 Y 等长的两条随机序列之间的比对分数。X 和 Y 之间的距离定义为

$$d(X,Y) = -\ln \frac{s_A - s_{\text{rand}}}{s_{\text{average}} - s_{\text{rand}}} + C,$$

其中 C 是一个可调整常数。计算序列之间距离的目的是利用 Neighbor-Joining 算法 (§5.4.3) 构造输入序列上的进化树，T。

T 定义了一个执行渐进式比对所需要的次序。所以，我们称 T 为比对过程的向导树。在最后一步，ClustalW 按照 T 所定义的次序计算输入序列上的

```
 CLUSTAL 2.1 Multiple Sequence Alignments


Sequence type explicitly set to Protein
Sequence format is Pearson
Sequence 1: Hs_Stat5B       787 aa
Sequence 2: Mm_STAT5B       786 aa
Sequence 3: Gg_STAT5b       787 aa
Sequence 4: Dr_STAT5.2      787 aa
Sequence 5: Aq_STAT         788 aa
Sequence 6: Dm_STAT         761 aa
Start of Pairwise alignments
Aligning...

Sequences (1:2) Aligned. Score: 96.95
Sequences (1:3) Aligned. Score: 91.23
Sequences (1:4) Aligned. Score: 72.43
Sequences (1:5) Aligned. Score: 33.93
Sequences (1:6) Aligned. Score: 28.12
Sequences (2:3) Aligned. Score: 90.20
Sequences (2:4) Aligned. Score: 73.28
Sequences (2:5) Aligned. Score: 34.86
Sequences (2:6) Aligned. Score: 28.52
Sequences (3:4) Aligned. Score: 72.55
Sequences (3:5) Aligned. Score: 34.94
Sequences (3:6) Aligned. Score: 27.46
Sequences (4:5) Aligned. Score: 36.34
Sequences (4:6) Aligned. Score: 29.96
Sequences (5:6) Aligned. Score: 25.76
Guide tree file created:   [clustalw2-I20130727-052302-0093-93318054-pg.dnd]

There are 5 groups
Start of Multiple Alignment

Aligning...
Group 1: Sequences:   2      Score:12782
Group 2: Sequences:   3      Score:12443
Group 3: Sequences:   4      Score:12220
Group 4: Sequences:   5      Score:5324
Group 5:                     Delayed
Alignment Score 35807

CLUSTAL-Alignment file created  [clustalw2-I20130727-052302-0093-93318054-pg.aln]
```

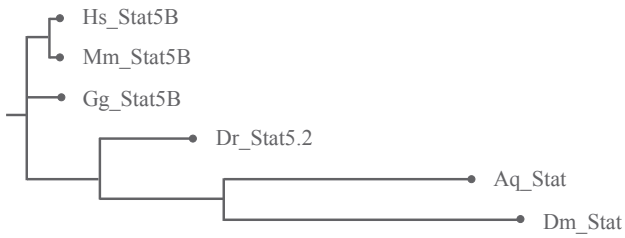
图 3.10: ClustalW 运行过程在输出文件里给出。该过程的第一部分列出输入序列的名称和所含碱基或氨基酸的个数。第二部分给出所有双序列的最优比对的分数以及存放生成进化树的文件名称。第三部分给出如何按照进化树逐步合并比对得到输出的比对。

多序列比对。在我们这个例子中，构造的向导树如图 3.11 所示。ClustalW 首先比对人和小鼠的 STAT5B 序列；再将鸡的 STAT5B 序列与人和小鼠 STAT5B 序列的比对合并起来，得到一个三序列的比对。另一方面，比对海绵和果蝇的 STAT 序列；再加入斑马鱼的 STAT5.2 序列，得到另一个三序列比对。最后，两个三序列比对合并在一起得到最终的结果 (图 3.12)。

ClustalW 在以后的不同版本里采用了一系列措施来改进它的比对性能。在最初的版本里，ClustalW 对输入序列不加区别。现在用户可以将不同的权值加在输入序列上。例如，当输入序列中包含多个非常相似的序列时，我们可以给这些序列赋上小的权值。这样做的目的是这些相似序列之间的比对不会误导最后的比对结果。另一个改进是用户可以选择更多的从双序列比对或其分数计算序列距离的方法 (§5.4.2)。

ClustalW 的优点是速度快。它的缺点是在相似度低的序列上的比对的精

A



B

((Hs_Stat5B:0.01495, Mm_Stat5B:0.01594):0.03001,
Gg_Stat5B:0.04754, (Dr_Stat5.2:0.12700,
(Aq_Stat:0.33871, Dm_Stat:0.040373):0.17027):0.09942);

图 3.11: ClustalW 在比对 STAT 序列时所构造和使用的向导树。(A) 该树的图形表示。(B) ClustalW 按 Newick 文本格式输出该树的表达式。

确度较低。这是因为在比对过程中，一旦一对字符匹配起来，在后续的比对过程中这个匹配永远存在。所以，在比对开始阶段所发生的错误匹配在后续过程被继续的放大。

§3.8.2 MUSCLE

自从二零零四年引入以来，MUSCLE 便得到分子生物学家和生物信息工作者的青睐。这主要是因为它的性能好、速度快等特点。在个人电脑上它可以在短短 21 秒内计算出 1000 个平均长度为 280 个氨基酸的蛋白序列之间的比对。它的高性能主要是通过多次迭代将几种不同的渐进式比对方法揉和在一起达到的。

MUSCLE 比对过程分为三个阶段。在第一阶段，MUSCLE 利用所有 k-字 (即长度是 k 的序列) 在输入序列中出现的次数简单地计算两两之间的距离。这是基于两条序列越相似，它们中同时出现的 k- 字的次数就越多。

考虑两个序列，X 和 Y。对于任何一个具体的 k- 字，ω，我们分别用 $n_X(\omega)$ 和 $n_Y(\omega)$ 表示 ω 在 X 和 Y 中出现的次数。如果 ω 在一个序列里出现的次数比在另一个里出现的多，那么多余的 ω 不可能匹配起来。所以，$\min\{n_X(\omega), n_Y(\omega)\}$ 是在 X 和 Y 的任何一个比对列中匹配起来的 ω 的最高次数。MUSCLE 使用下列定义计算 X 和 Y 之间的距离：

$$m_k(X,Y) = 1 - \frac{\sum_\omega \min\{n_X(\omega), n_Y(\omega)\}}{\min\{l(X), l(Y)\} - k + 1},$$

图 3.12: 在 STAT 序列上 ClustalW 计算的比对结果。该比对共有 825 列，其中的前 120 列和后 105 列在图中给出。每列下面的符号表示一列的保守程度。星号 (*) 表示该列是完全匹配列。冒号 (:) 表示该列中出现的氨基酸不同，但它们的大小和亲水性都基本一致。句号 (.) 表示该列中出现的氨基酸的大小或亲水性基本一样。

其中 $l(X)$ 和 $l(Y)$ 分别是 X 和 Y 的长度。上述公式的右边第二项的分子上是在两序列中所有匹配起来的 k- 字数目的一个估计；而分母是所有 k- 字在较短的一个序列中出现的总次数。$m_k(X,Y)$ 可以看作是两序列里不能匹配起来的所有 k 字所占比例的一个近似值。

例如，在 S_1: ATTGCCATTA 和 S_2：ATCCAATTTT 中，AT在每个序列里出现两次，TT分别出现 2 次和 3 次，CC及CA在每个序列里各出现 1 次。其它的 2- 字：

AA, AG, AC, GA, GG, GC, GT, CG, CT, TA, TG, TC

至少在一个序列没有出现。所以，$m_2(S_1,S_2) = 1-(2+2+1+1)/(10-2+1) = 1/3$。

得到输入序列两两之间的距离之后，MUSCLE 使用 UPGMA 方法 (见 §5.4.4) 构建第一个向导树 T_1，并利用它计算出输入序列之间的第一个比对 A_1。

在第二阶段，MUSCLE 利用比对 A_1 所诱导的双序列比对计算序列两两之间的 Kimura 距离：

$$d_k(X,Y) = -\ln\left(1 - n_{\mathtt{id}} - \frac{1}{5}n_{\mathtt{id}}^2\right),$$

其中 $n_{\mathtt{id}}$ 是 X 和 Y 的比对中完全匹配列所占的比例。再次使用 UPGMA 方法从所得到的 Kimura 距离构建第二个向导树 T_2 以及利用它计算出输入序列之间的第二个比对 A_2。

向导树 T_2 中每个叶节点代表一个输入序列。在 T_2 上删除一个分枝得到两个子树，从而将输入序列集划分成不相交的两个子集。在第三阶段，MUSCLE 首先将 T_2 上的分枝按其到树根的距离从大到小排成一列，记为 L。然后，从 L 中的第一个分枝开始考虑。利用它将输入序列分成两个不相交的子集 S_1 和 S_2。利用 A_2 在 S_1 和 S_2 上诱导的子比对的谱将它们重新合并成一个比对 A_3(参见习题 15)。

如果 A_3 的 SP- 分数比 A_2 小，在 L 上选取下一个分枝重复上述过程。如果 A_3 的 SP- 分数比 A_2 高，用 A_3 替代 A_2 重复上述过程。当这个迭代过程不能产生更好的比对或者迭代次数已经达到用户所设置的次数时，比对过程将终止。

§3.8.3 其它多序列比对程序

表 3.1 列出了除 ClustalW 和 MUSCLE 以外其它一些常用的多序列比对程序。

如果说渐进式比对策略是提高比对速度的重要方法，那么，采用一致打分方法是各种比对程序提高其性能的主要手段。一致打分方法依据参与比对序列的各种性质给比对输入序列量身打造一个打分矩阵。自制特定的打分矩阵具有下列几方面优点。

第一，通用的打分矩阵 (BLOSUM 和 PAM 打分矩阵) 是考虑所有可能序列比对得到的打分矩阵。当比对特定一组序列时，现有的 BLOSUM 或者 PAM 打分矩阵并不一定是最好的打分矩阵。制定一个特除的矩阵可以考虑到输入序列的相似性。

第二，考虑两个蛋白质序列，X 和 Y。如果在 X、Y 和另一条序列 Z 的最优比对中，X 的字符 x_i 及 Y 的字符 y_j 和 Z 中的 z_k 构成一列，那么，x_i 和 y_j 就可能是从同一氨基酸进化而来。所以，若干个输入序列上的比对含有对改进多序列比对的有用信息。自定义打分方法允许我们通过和其它字符的关系调整 x_i 和 y_j 匹配的得分。

第三，设计空位罚分法则是比对研究中的一个相当复杂的问题。至今也没有建立起它的理论基础。在比对中引进空字符的唯一目的是为了带来后面列上的匹配。自定义打分矩阵已经包含了空位位置的信息。所以，利用局部比对的信息制定打分矩阵可以让我们避免考虑如何给空位打分。

T-Coffee 是第一个使用一致打分方法的比对程序。通过自定义打分矩阵，T-Coffee 利用已有的双序列或者三序列上的 全序列或局部序列段上的比对来提高比对最终结果的精确性。它还利用其它程序输出的比对结果来比对一组序列。可以说，T-Coffee 是一个泛比对程序。MAFFT 的原始版本仅是一个重复使用渐进式方法的快速比对程序。为了提高其性能，后来的版本也相继采用了一致打分技巧。ProbCons 从概率的角度使用了自定义打分方法。

PRANK 是另一个据有特色的多序列比对程序。它最大的特点是在渐进比对的过程中将插入和删除区别对待。

DIALIGN 是通过合并高相似区域的局部比对计算一组序列的比对。FSA 是 DIALIGN 的概率版本。

最后，提醒读者虽然上面列举的比对程序性能比较好，但是在相似度较低的序列上它们输出的比对结果往往也不可靠 (见习题 14)。

§3.9 *基因组的比对

由于高通量测序技术的发展，全基因组的测序正在以前所未有的速度进行着。因此，比对基因组序列变成了比较基因组学 (comparative genomics) 和综合基因组学 (metagenomics) 的一个根本问题。比对基因组区域序列的主要动机是寻找哪些区域受正向选择的影响在一个谱系上快速的突变以及哪些区

表 3.1: 常用的多序列比对程序。

程序	特点	网址
MAFFT	重复的使用渐进式比对。采用自定义打分矩阵	mafft.cbrc.jp/alignment/software/
PRANK	利用系统进化树，将删除和插入区别对待	www.ebi.ac.uk/goldman-srv/prank/
T-Coffee	使用自定义打分矩阵，整合两个输入序列之间的各种比对信息	www.tcoffee.org/
ProbCons	使用自定义打分矩阵的概率版本，仅用于氨基酸序列的比对	probcons.stanford.edu/
DIALIGN	通过连接高相似区域的局部比对得到输入序列之间的比对	dialign.gobics.de/
FSA	合并局部比对的概率版本	http://fsa.sourceforge.net/

域受负向选择的影响处于高度保守状态。比对全基因组的动机主要是推断基因组结构组织的进化。比对基因组序列和比对蛋白序列相比有下列几个特点。

首先，蛋白质的比对涉及上百个甚至上千个长度很少超过几千个氨基酸的序列。而比对基因组序列所涉及的序列常常数目少，但每个区域的长度经常是几百万甚至上千万个碱基对。

其次，一个蛋白质家族的序列在负向选择的影响下发生突变的速度小，从而比较保守。相反地，非基因编码区的 DNA 序列保守性差。所以，比对来自分化时间久远的基因组序列是一件非常困难的任务。另外，真核生物的基因组序列往往包含有大量重复的片段，这也增加了比对的难度。

最后，基因组在进化过程中历经数次重组。重组事件包括染色体甚至整个基因组的复制，长序列片段的插入 (insertion)、删除 (deletion)、反转 (reversal) 和移位 (transposition)。虽然我们还不清楚染色体复制和重组的机制，这些事件的发生是没有疑问的。其证据包括现存物种的基因组所含染色体的数目相差极大。人类有 46 条染色体；黑猩猩有 48 条染色体；猴子则有多达 62 条染色体。

序列片段的反转是指一个双链片段反转后重新插入原来的位置，这里序列反转的规则是每条链上的序列颠倒之后再将A(T) 替换成T(A) 以及C(G) 替换成G(C)：

$$\begin{array}{l} \texttt{...ATTT}\mathbf{AGCC}\texttt{AGGG...} \\ \texttt{...TAAA}\mathbf{TCGG}\texttt{TCCC...} \end{array} \Longrightarrow \begin{array}{l} \texttt{...ATTT}\mathbf{GGCT}\texttt{AGGG...} \\ \texttt{...TAAA}\mathbf{CCGA}\texttt{TCCC...} \end{array}$$

片段的移位事件是一个序列段从一个位置平移到另一个位置：

$$\begin{array}{l} \texttt{...ATTT}\mathbf{CCCCC}\texttt{AGGG...} \\ \texttt{...TAAA}\mathbf{TTTTT}\texttt{TCCC...} \end{array} \Longrightarrow \begin{array}{l} \texttt{...ATTTAG}\mathbf{CCCCC}\texttt{GG...} \\ \texttt{...TAAATC}\mathbf{TTTTT}\texttt{CC...} \end{array}$$

由于反转和移位事件，在基因组序列中，一个序列上的核苷酸碱基不一定按其祖先序列上的次序排列。考虑反转和移位突变事件大大增加了比对的难度。

UCSC(Univ. of California at Santa Cruz) 基因组服务器 (genome.ucsc.edu) 提供了数十个物种的基因组序列的查询和比对。当用户输入一段序列时，服务器显示它在所有物种上的同源序列 (包括它自己) 之间的比对。这个多序列比对是一个渐进式比对的结果。首先，得到的同源序列两两之间的双序列比对先由 BLASTZ(http://www.bx.psu.edu/miller_lab/) 产生。再使用 MULTIZ 计算一个渐进式比对。

另一个常用的比对多个基因组序列的程序是 MAUVE (http://asap.ahabs.wisc.edu/software/mauve/overview.html)。MAUVE 考虑了各种基因组突变事件以及碱基的替换 (图 3.13)。这个程序速度快，占用空间小。它可以在 4 个小时内计算出 9 个长度介于 4~5 百万个碱基对的细菌序列的比对。

§3.10 参考文献

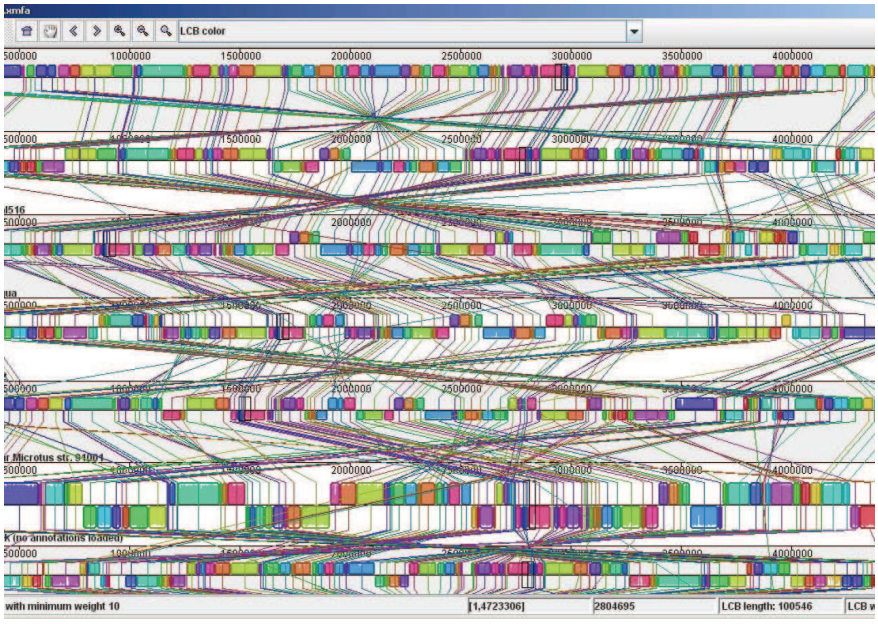


图 3.13: 使用 MAUVE 得出的 9 个细菌基因组的比对。经作者允许取自 MAUVE 网站。

1. Book RV, Otto F. (1993) String-rewriting Systems, Springer, New York, USA.

2. Crooks GE, Hon G, Chandonia JM, Brenner SE. (2004) WebLogo: a sequence logo generator. *Genome Res.* 14: 1188–1190.

3. Edgar RC. (2004) MUSCLE: A multiple sequence alignment method with reduced time and space complexity. *BMC Bioinformatics* 5: no. 113.

4. Feng DF, Doolittle RF. (1987) Progressive sequence alignment as a prerequisite to correct phylogenetic trees. *J. Mol. Evol.* 25: 351–360.

5. Löytynoja A. (2012) Alignment methods: strategies, challenges, benchmarking, and comparative overview. In *Evolutionary Genomics*, pp.203–235, Humana Press.

6. Katoh K, Misawa K, Kuma K, Miyata, T. (2002) MAFFT: a novel method for rapid multiple sequence alignment based on fast Fourier transform. *Nucl Acids Res.* 30: 3059–3066.

7. Schneider TD, Stephens RM. (1990) Sequence logos: A new way to display consensus sequences. *Nucleic Acids Res.* 18: 6097–6100.

8. Thompson JD, Higgins DG, Gibson TJ. (1994) CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. *Nucleic Acids Res.* 22: 4673–4680.

9. Wang LS, Jiang T. (1994) On the complexity of multiple sequence alignment. *J. Comput. Biol.* 1: 337–348.

§3.11 练习题

1. [S]-x-[RK] 是激酶蛋白 C 的磷酸化位点的模体。它是下列哪方面的一个例子：

(a) 表示同源蛋白序列特征的模体。
(b) 表示一些不必同源的蛋白序列特征的模体。
(c) 表示同源蛋白序列特征的结构区域。
(d) 表示一些不必同源的蛋白序列特征的结构区域。

2. 使用 PROSITE 里的工具分析珠蛋白序列的结构区域。哪一个结构区域最能刻画珠蛋白家族的序列特征？

3. 解释下列每个模体并指出它表示多少个可能的序列：

(a) `T-T-T-x(4)-G-A-A`
(b) `G-{C}-W`
(c) `N-{P}-[ST]-{P}`

4. 推导出计算四条序列之间的最优比对的 SP 得分的递推公式。

5. 设计一个多序列局部比对的快速算法。

6. 在 Feng-Doolittle 多序列比对算法中使用的 “一旦空位，永远空位” 的目的是：

(a) 确保出现在不太相似的序列之间的空位在比对结果中保留下来。
(b) 确保出现在高度相似的序列之间的空位在比对结果中保留下来。
(c) 确保在进化早期分化的序列在渐进比对的过程中先进行比对。
(d) 确保空位不会在插入的序列中放错位置。

7. （Wang 和 Jiang，1994）证明多序列比对问题是 NP- 难解问题。

8. 在使用渐进式策略进行多序列比对时，我们需要简单的合并两个比对。假设 X 是 k 条序列上的一个比对，Y 是 l 条序列上的一个比对。X 和 Y 的一个比对 Z 是一个有 $k+l$ 行的多序列比对，其中前 k 行是在 X 中插入若干空列得到，而后 l 行是在 Y 中插入若干空列得到。令 Z 的 $k+l$ 行分别是 $Z_1, Z_2, \cdots, Z_{k+l}$. 对于任何 i, j 满足 $1 \le i \le k$ 及 $k+1 \le j \le k+l$，从 Z_i 和 Z_j 中去掉空列得到一个双序列比对，称其为 Z 在对应序列上的诱导比对，记为 $Z(i, j)$。例如，如果 Z_i 和 Z_j 分别是

```
A - - - - - G G
A A - T - C G G
```

那么，对应的诱导比对是

```
A - - - G G
A A T C G G
```

Z 的 SP- 分数定义为

$$\sum_{1\le i\le k}\ \sum_{k+1\le j\le k+l} s(Z(i,k+j)),$$

其中 $s(Z(i,j))$ 是双序列比对 $Z(i,j)$ 在某个基本空位罚分模型下的分数。设计一个动态规划算法计算任何两个比对上的分数最大的比对。

9. 证明在仿射空位罚分的模型下，前题中定义的将比对进行比对的问题是 NP- 难解问题。

10. 为什么 ClustalW 没有像 BLAST 程序一样输出比对结果的期望 (E-) 值？

(a) 多序列比对的比分统计理论还没有建立起来。

(b) ClustalW 报告了和 BLAST 程序不一样的期望值。

11. 使用匹配分数、非匹配分数和增减列的分数分别是 2、-3、-4 的基本空位法则计算下列四条序列上的星型比对：

```
AGTATGC
ATCAGTGC
ATAAGGC
ACATCG
```

12. 如何改进一个多序列比对方法的性能？

(a) 利用 PSI-BLAST 搜索添加更多的同源序列。

(b) 考虑蛋白序列的二级结构。

(c) 考虑蛋白序列的三级结构。

(d) 上面所有的策略。

13. 使用 ClustalW 比对人 β 珠蛋白 (NP_000509) 及其在黑猩猩、狗和小鼠中的同源序列 (XP_508242, XP_537902, NP_058653)。H64(即第 64 个位置上的组氨酸) 和 H93 在人 β- 珠蛋白和血红素结合上起着重要作用。试问 H64 和

H93 出现在所得到的比对里的匹配列上吗？各珠蛋白的序列可以利用其编号通过使用 BLAST(http://blast.ncbi.nlm.nih.gov/Blast.cgi) 搜索 NCBI 数据库得到。

14.使用 ClustalW、ProbCons、T-Coffee、MUSCLE 比对人 β- 珠蛋白 (globin)(PDB 索引号：2hhb)、人肌红珠蛋白 (2MM1), 人神经珠蛋白 (1OJ6_A)、黄豆血红珠蛋白 (1FSL) 和大米血红珠蛋白 (1D8U) 的蛋白序列。这些程序输出的比对在结构区域上相同吗？如果必要，可以调整每个比对程序的运行参数。

15. 在比对两序列时，打分矩阵给出含两个氨基酸 i 和 j 的列的分数 S_{ij}; 空位打分法则给出增删列的分数。推广到比对两个谱 P 和 Q 时，由 P 中的列 x 和 Q 中的列 y 组成的列的分数定义为:

$$\sum_i \sum_j f_i^x f_j^y S_{ij}, \tag{3.14}$$

其中 i 和 j 是一个氨基酸或者空字符 '-'，$f_i^x(f_i^y)$ 是 i 在 $x(y)$ 上的频率，如果 i 或者 j 是空字符，$S_{ij}=0$。设计比对两个谱的算法。

16. 反转是基因组重组最普遍的突变事件。人基因组大约是小鼠基因组中三百个保守区域序列的重组。于是，生物学家希望知道如何使用最少的反转突变事件将一个基因组转换成另一个基因组。在比较两个线粒体基因组时，如果将保守区域序列段抽象成一个元素，那么每个基因组可以表示成这些元素集上的一个置换。与此同时，反转突变可以看作将一个置换变成另一个的变换。例如，将从第 i 个元素到第 j 个元素的区域反转把置换

$$\pi_1\pi_2\cdots\pi_{i-1}\overrightarrow{\pi_i\cdots\pi_{j-1}\pi_j}\pi_{j+1}\cdots\pi_n$$

变成

$$\pi_1\pi_2\cdots\pi_{i-1}\overleftarrow{\pi_j\cdots\pi_{i+1}\pi_i}\pi_{j+1}\cdots\pi_n\text{。}$$

从一个置换到另一个置换所需要的反转变换的最少次数定义为这两个置换之间的反转距离。计算两个置换之间的反转距离叫反转距离问题。计算 34658172 和 12345678 之间的距离。

17. 考虑 $\{1,2,\cdots,n\}$ 上的置换。不难看出任何一个置换都可以变成恒等置换。将从一个置换 π 变换到恒等置换 $I=12\cdots n$ 所需要的最少反转的次数定义为它的反转距离 $d(\pi)$。令 $\pi=\pi_1\pi_2\cdots\pi_n$。对应任何一个位置 j, 如果 $|\pi_j-\pi_{j+1}|>1$, 我们称 j 是一个断点。证明 $d(\pi)$ 总是大于断点数的一半。

18. 考虑 $N=\{1,2,\cdots,n\}$ 上的置换。将从第一个位置到另外一个位置的区域反转叫前端反转。不难证明，每个 N 上的置换最多需要经 $2n-2$ 次前端反转就可以转换成恒等置换。在 1970 年代，比尔 · 盖茨 (著名企业家，微软公司创始人) 在哈佛读书时和他的老师一起证明了 N 上任何一个置换至多需要