

总结人: bayer star

总结思路: 根据老师重点和前人总结

1. N-W 算法
2. S-W 算法
3. The Neighbor Joining Algorithm
4. fitch 算法
5. Sankoff 算法

1.N-W 算法

、Needleman-Wunsch 算法, 简称 N-W 算法教材 P62

——引入 gap 罚分, 这个才是最重要的, 前面都是引子

引入了空位罚分。以 15 届的习题为例讲解此方法。

习题: Using a scoring system of +3 for a match, -3 for a mismatch, and -1 for a gap, use the Needleman-Wunsch algorithm to fill in a dynamic programming table and find the best alignment between the sequences AGTGTA and AGACATA. If there are ties, list all of the best alignments.

Needleman - Wunsch

$$D_{ij} = \max \begin{cases} D(i-1, j-1) + \delta(A_i, B_j) \\ D(i-1, j) - d \\ D(i, j-1) - d \end{cases}$$

习题中, $\delta(A_i, B_j) = +3$ for match, $\delta(A_i, B_j) = -3$ for mismatch
 $d = 1$ 即 -1 for gap

B \ A	A	G	T	G	T	A	A
A	-7	-3	1	5	4	3	7
T	-6	-2	2	6	5	4	3
A	-5	-1	3	2	1	0	4
C	-4	0	4	3	2	1	5
A	-3	1	5	4	3	2	8
G	-2	2	6	5	4	3	2
A	-1	3	2	1	0	-1	-2
	0	-1	-2	-3	-4	-5	-6

右上角

举例说明计算

$$D_{ij} = \max \begin{cases} 3-1 \\ 4+3 \\ 8-1 \end{cases} = 7$$

sequences AGT-GT-A-A
best alignments AG-A--CATA

最优比对结果有多种(略), 原因是空位罚分低于错配罚分, 由此会产生更多的 gap 而不是 mismatch.

基本要领和上一个算法一样, 不一样的是,

①最下行和最左列不再为零, 而是 gap 罚分的叠加

②计算 $D(i,j)$ 要引入 gap 罚分。

③回溯方法: 从最右上一格开始回溯, 回溯即为找到这个数得到的路径, 如我举例说明计算的这四个数, $4+3=7$ 且 $8-1=7$, 所以既可以向 4 回溯也可以向 8 回溯。

注: 匹配的 3 个, 得 12 分, 空位的 6 个, 扣 6 分, 总分为 6

NW 算法，为了求一个问题的解，可以求小规模子问题的解。Prefix (s, i) 表示序列 s 的长度为 i 的前缀子串。先构造一个 (m+1)*(n+1) 的二阶矩阵 D，D_{i,j} 表示 prefix (s,i) 和 prefix (t, j) 的最佳比对得分。当 j=0，D_{i,0} 表示 prefix (s,i) 与长度为 i 的空串之间的最优比对得分。

第一步初始化，先规定 D_{0,0}=0，由上得：

$$H_{i,0} = g \times i$$

$$H_{0,j} = g \times j$$

第二步递推计算 D_{i,j}

匹配或替换

prefix(s,i-1)	s _i
prefix(t,j-1)	t _j

删除

prefix(s,i-1)	s _i
prefix(t,j)	-

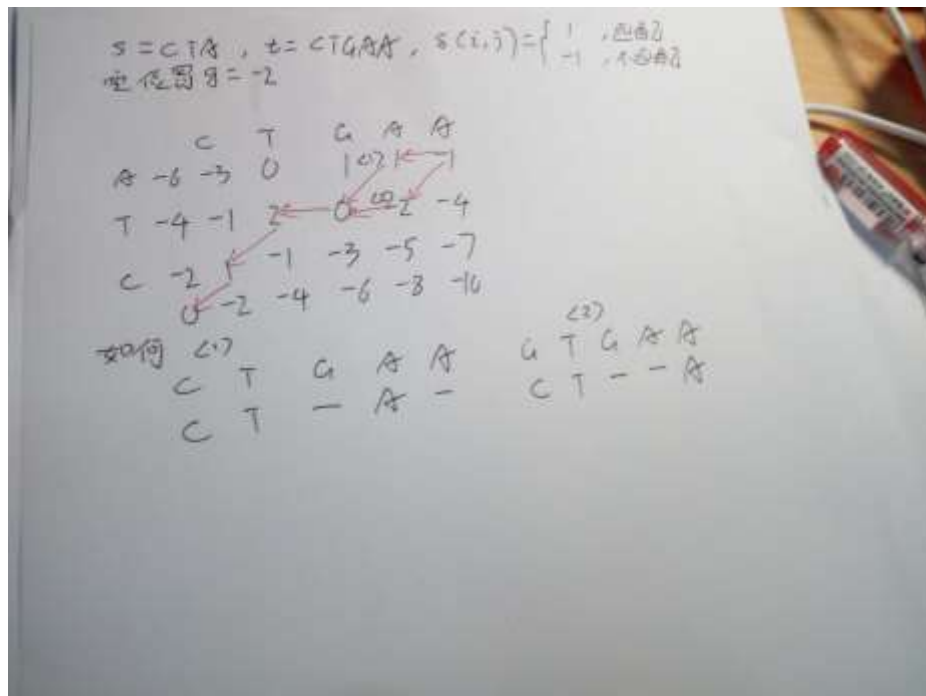
插入

prefix(s,i)	-
prefix(t,j-1)	t _j

这四种情况，替换、删除、插入是有惩罚值的，因此

$$D(i,j) = \max \begin{cases} D(i-1,j-1) + \sigma(A_i, B_j) \\ D(i-1,j) + c_{\text{insert}} \\ D(i,j-1) + c_{\text{delete}} \end{cases}$$

字比较丑



在箭头的方向上是没有空位的。

2. S-W 算法

局部比对 (Local alignment) ppt34

——Smith-Waterman 算法

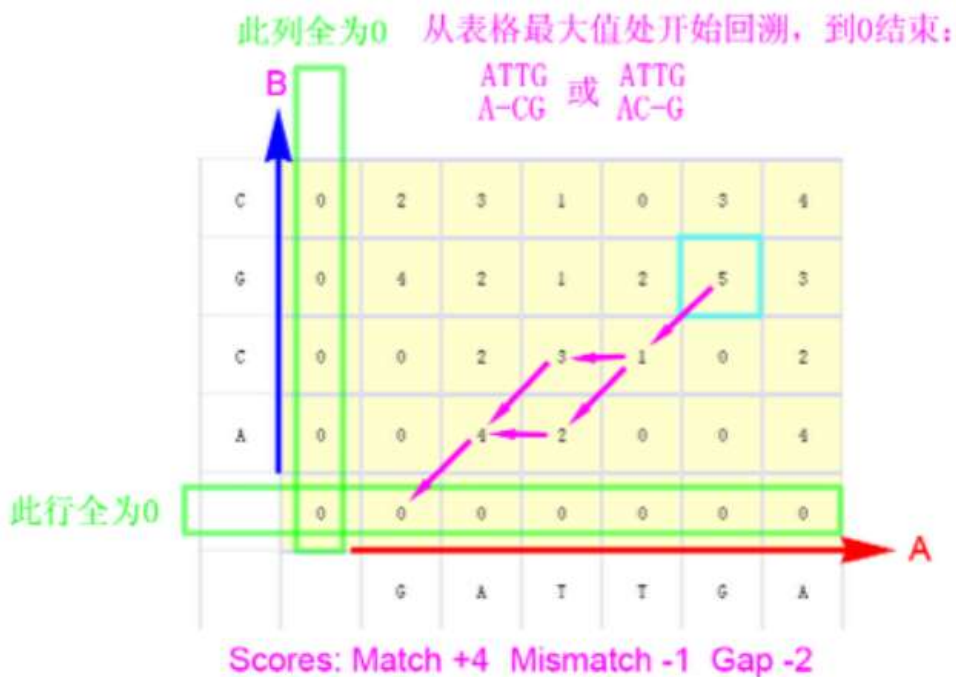
$$D(i, j) = \max \begin{cases} D(i-1, j-1) + \delta(A_i, B_j) \\ D(i-1, j) + g_s \\ D(i, j-1) + g_d \\ 0 \end{cases}$$

相比于 Needleman-Wunsch 算法，此算法要做三点调整：

- ① 序列对比过程中不存在负分，如出现负值，调整为 0；
- ② 和半全局比对类似，在序列 A 和 B 的开头加入 gap 均不需要罚分，因此表格最下行和最左列设置为 0（不再是 gap 罚分的叠加）。
- ③ 和半全局比对类似，不需要序列 A 或序列 B 中任何一个对比完全，因此从表格中选择所有数值中最大的格子开始回溯（既不是最右上格也不是最上行最大格），到 0 回溯结束。

找到 A 的子串和 B 的子串的 common 部分

G A T T G
 A C - G C
 common motif.
 G A T T G
 A - C G C



3.The Neighbor Joining Algorithm

- 4-point condition: 对于每四个 taxa i, j, m, n , 有
$$d(i, j) + d(m, n) \leq \max\{d(i, m) + d(j, n), d(j, m) + d(i, n)\}$$
 py133

The Neighbor Joining Algorithm

- 为了简化构建方法, 先引入几个概念
 - a) X-trees——X 树是叶的集合, 没有非叶的部分, 比如根 (度为 2)、内节点
 - b) 由 X-tree 可以定义 X 的成对距离 (pairwise distance) ——两个叶 x_1 和 x_2 之间的距离是 x_1 到 x_2 的路径的长度。
- 那么, 基于距离构建系统发生树的问题就转换为——给定 X 的成对距离, 找到 X-tree T, 使得由 T 定义的距离尽可能地类似于输入距离
- 于是问题又回来了, 如何判断一个从数据中得到的距离矩阵是否有可能用 X-tree 表示?
- 答案是, 如果对于所有的叶 a, b, c, d 在 X 中, 三个量 $d(a, b) + d(c, d)$, $d(a, c) + d(b, d)$ 和 $d(a, d) + d(b, c)$ 严格大于其他两个, 那么距离 d 是树的距离。
- 进一步具体化构建方法, 需要再引入一个概念——Agglomeration (聚集)
如果相应的叶子邻近相同的内节点, 则在系统发生树中的两个叶子被称为邻居 (neighbors)。
于是, 系统发生树可以迭代地在大步骤中构建——即聚集
 - a) 选择一对 neighbors x, y
 - b) 用新的 (伪) taxa z 代替 x 和 y , 直到只剩下三个 taxa。
- 那么, 该如何聚集, 即选择哪一对 taxa 呢?



简单的选择, 就是直接选成对距离最小的一对叶子, 这就可以通过 UPGMA 算法来完成。

进一步优化, 就需要不止考虑一对 taxa 的 distance, 还要考虑他们与其他 taxa 的距离, 这就需要 The Neighbor Joining Algorithm 来完成!

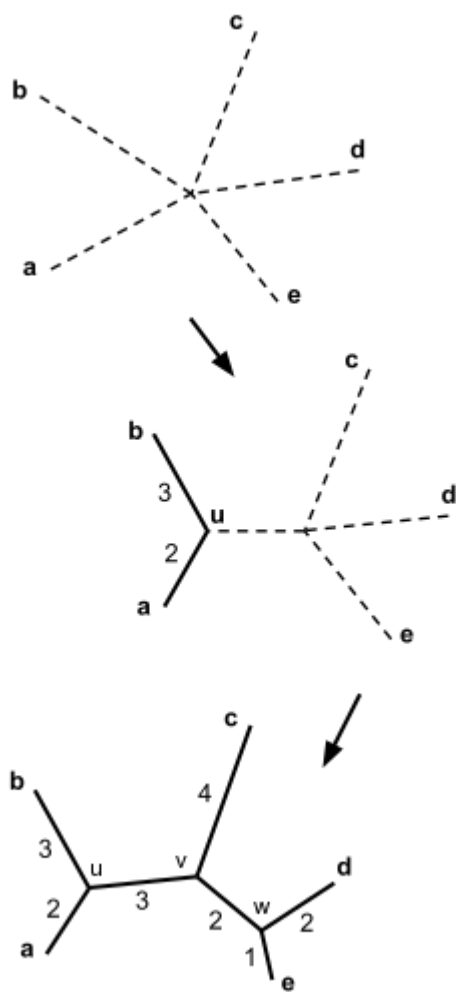
① The Neighbor Joining Algorithm 的运行步骤

The Neighbor Joining Algorithm

的运行步骤:

1. 计算出距离矩阵, 集合 $X = \{a, b, c, d, e\}$
2. 根据 Q 标准, 求最小值
3. 具有 Q 最小值的 x, y , 比如 a, b , 认为它们是邻居, 插入一个内结点 u
4. 计算 $d(a, u)$, $d(b, u)$;
5. 计算 $d(u, c)$, $d(u, d)$, $d(u, e)$
6. 重新记 $X = \{u, c, d, e\}$, 写出距离矩阵
7. 重复 1-6
8. 直到 $|X| = 3$

图解:

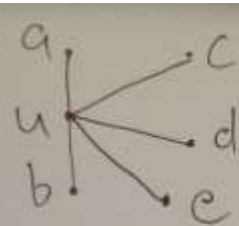


1.Q 标准的计算

$$Q(x, y) = d(x, y) - \frac{1}{n-2} \sum_{z \in X-x} d(x, z) - \frac{1}{n-2} \sum_{z \in X-y} d(y, z).$$

$$Q(x, y) = d(x, y) - \frac{1}{n-2}(r(x) + r(y)).$$

2..d(u,c)



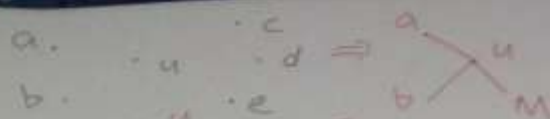
$$d_{uc} = \left[(d_{au} + d_{uc}) + (d_{bu} + d_{uc}) - (d_{au} + d_{bu}) \right] \times \frac{1}{2}$$

$$d_{uc} = \frac{1}{2}(d_{ac} + d_{bc} - d_{ab})$$

一般地

$$d_{uz} = \frac{1}{2}(d_{xz} + d_{yz} - d_{xy})$$

3.d(u,a)的计算



b. $\cdot u \cdot e$
 $x' = \{a, c, d, e\}$
 $M = \{c, d, e\} \quad M = x' - u$
 $|M| = |x'| - 1$

c, d, e 可以看作一个整体, 记为 M

那么 $d_{ua} = \frac{1}{2}(d_{am} + d_{ab} - d_{bm})$, 理由如下:

对于任意一个和左图一样的树

$d_{vx1} = \frac{1}{2}(d_{x1x2} + d_{x1x3} - d_{x2x3})$

$$d_{am} = \frac{1}{|M|} \sum_{z \in M} d(a, z)$$

$$d_{bm} = \frac{1}{|M|} \sum_{z \in M} d(b, z), \text{ 代入得:}$$

$$d_{ua} = \frac{1}{2} \left(\frac{1}{|M|} \sum_{z \in M} d(a, z) + d_{ab} - \frac{1}{|M|} \sum_{z \in M} d(b, z) \right)$$

$$d_{ua} = \frac{1}{2|M|} \left(\sum_{z \in M} (d(a, z) + d(a, b) - d(b, z)) \right)$$

$$d_{ua} = \frac{1}{2(|x'| - 1)} \sum_{z \in x' - u} (d(a, z) + d(a, b) - d(b, z))$$

不失一般性

$$d_{ux} = \frac{1}{2(|x'| - 1)} \sum_{z \in x' - u} (d(x, z) + d(x, y) - d(y, z))$$

	a	b	c	d	e	
a	0	5	9	9	8	$r_a = 31$
b	5	0	10	10	9	$r_b = 34$
c	9	10	0	8	7	$r_c = 34$
d	9	10	8	0	3	$r_d = 30$
e	8	9	7	3	0	$r_e = 27$

$$Q(x, y) = d(x, y) - \frac{1}{n-2} \sum_{z \in X-X} d(x, z) - \frac{1}{n-2} \sum_{z \in X-Y} d(y, z)$$

$$Q(a, b) = d(a, b) - \frac{1}{n-2} (r_a + r_b)$$

	a	b	c	d	e
a		$-\frac{10}{3}$	$-\frac{28}{3}$	$-\frac{29}{3}$	$-\frac{29}{3}$
b			$-\frac{28}{3}$	$-\frac{29}{3}$	$-\frac{29}{3}$
c				$-\frac{29}{3}$	$-\frac{29}{3}$
d					$-\frac{29}{3}$
e					

$$5 - \frac{1}{3} \times 31 - \frac{1}{3} \times 34$$

$Q(a, b)$ 最小，故添加 a, b

~~添加 a, c, d, e~~

$$X' = \{u, c, d, e\}$$

$$d_{ua} = \frac{1}{2 \times (n-1)} \times [9+5+10+9+5+10+8+5+9] = 2$$

$$d_{ub} = \frac{1}{2 \times (n-1)} \times [10+5+9+10+5+9+9+5+8] = 3$$

$$d_{uc} = \frac{1}{2} (d_{ac} + d_{bc} - d_{ab}) = \frac{1}{2} (9+10-5) = 7$$

$$d_{ud} = \frac{1}{2} (d_{ad} + d_{bd} - d_{ab}) = \frac{1}{2} (9+10-5) = 7$$

$$d_{ue} = \frac{1}{2} (d_{ae} + d_{be} - d_{ab}) = \frac{1}{2} \times (8+9-5) = 6$$

	u	c	d	e	
u	0	7	7	6	$r_u = 20$
c	7	0	8	7	$r_c = 22$
d	7	8	0	3	$r_d = 18$
e	6	7	3	0	$r_e = 16$

$$Q(x, y) = d(x, y) - \frac{1}{n-2} (r_x + r_y)$$

	u	c	d	e
u		-14	-12	-12
c			-12	-12
d				-14
e				

$$7 - \frac{1}{2} \times [20 + 22]$$

$$7 - \frac{1}{2} \times [20 + 18]$$

$$Q_{ue} = 6 - \frac{1}{2} \times [20 + 16] = -12$$

$$Q_{dc} = 8 - \frac{1}{2} \times [22 + 18] = -12$$

$$Q_{ec} = 7 - \frac{1}{2} \times [22 + 16] = -12$$

$$Q_{de} = 3 - \frac{1}{2} \times [18 + 16] = -14$$

在 u, c 之间加入 v

$$x' = \{v, d, e\}$$

$$d_{v,x} = \frac{1}{2 \times (3+1)} \sum_{x \in x'} [d(u, z) + d(u, c) - d(\bar{z}, z)]$$

$$= \frac{1}{4} [d(u, d) + d(u, c) - d(\bar{v}, d) + d(u, e) + d(u, c) - d(\bar{e}, e)]$$

$$= \frac{1}{4} [1+7-8+6+7-7] = 3$$

$$d_{v,c} = \frac{1}{4} [3+7-7+7+7-6] = 4$$

$$d_{v,d} = \frac{1}{2} [d_{vd} + d_{cd} - d_{vc}] = \frac{1}{2} [7+8-7] = 4$$

$$d_{v,e} = \frac{1}{2} [d_{ve} + d_{ce} - d_{vd}] = \frac{1}{2} [6+7-7] = 3$$

⑤

	v	d	e	
v	0	4	3	$r_v = 7$
d	4	0	3	$r_d = 7$
e	3	3	0	$r_e = 6$

Q	v	d	e
v	0	-10	-10
d		0	-10
e			0

$$Q_{vd} = 4 - \frac{1}{10} \times (7+7) = -10$$

$$Q_{ve} = 3 - \frac{1}{10} \times (7+6) = -10$$

$$Q_{de} = 3 - \frac{1}{10} \times (7+6) = -10$$

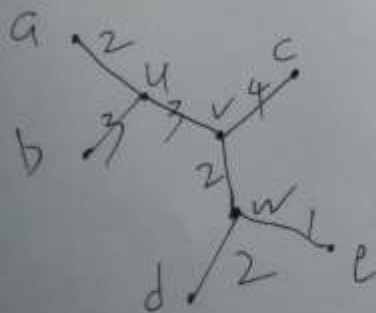
v 和 d 之间加入 w

$$x' = \{w, e\}$$

$$d_{v,w} = \frac{1}{2 \times (2+1)} \times [d_{ve} + d_{vd} - d_{de}] = \frac{1}{3} \times [3+4-3] = 2$$

$$d_{d,w} = \frac{1}{2 \times (2+1)} \times [d_{de} + d_{vd} - d_{ve}] = \frac{1}{3} \times [3+4-3] = 2$$

$$d_{w,e} = \frac{1}{2} \times [d_{ve} + d_{de} - d_{vd}] = \frac{1}{2} \times [3+3-4] = 1$$



$$d_{ab} = 5 \quad d_{cd} = 8$$

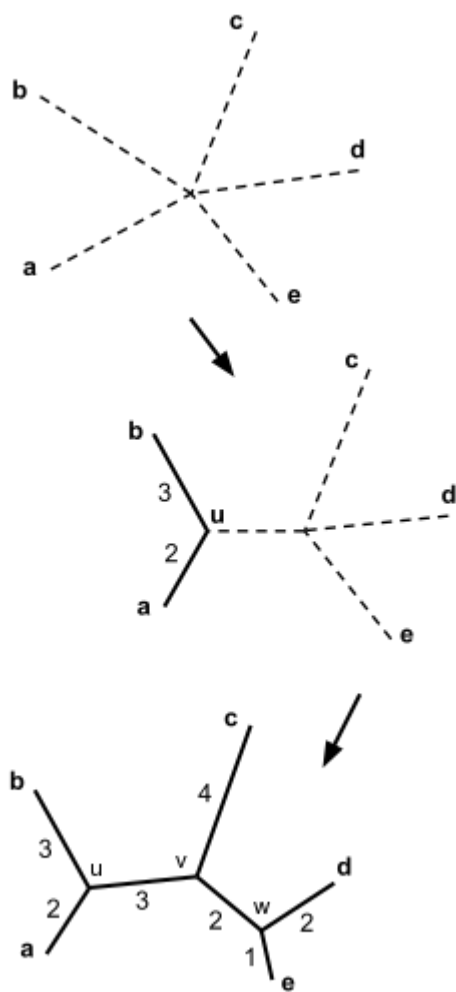
$$d_{ac} = 9 \quad d_{ed} = 3$$

$$d_{ae} = 8$$

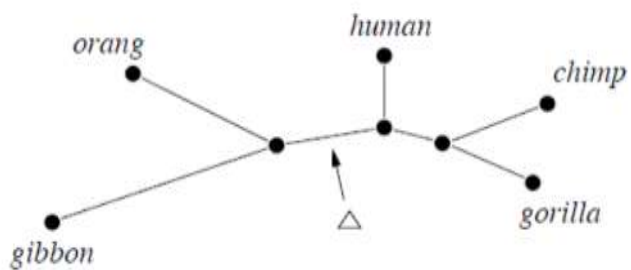
$$d_{bc} = 10$$

$$d_{bd} = 10$$

$$d_{be} = 9$$



4.fitch 算法

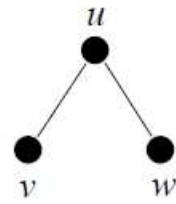


将根节点添加到任何边

自下至上

从叶节点到根节点

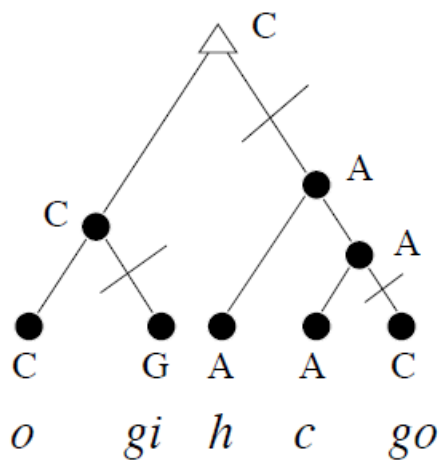
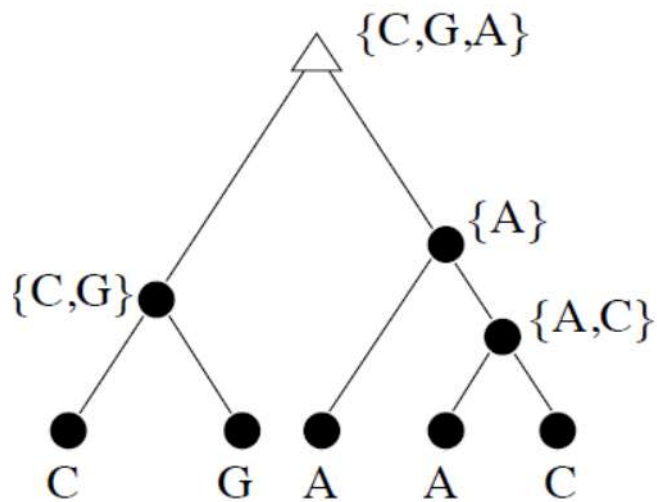
$$u = \begin{cases} \mathcal{V} \cup \mathcal{W}, & \text{if } \mathcal{V} \cap \mathcal{W} = \emptyset \\ \mathcal{V} \cap \mathcal{W}, & \text{else} \end{cases}$$



自上至下

从根节点开始至叶节点，内结点被赋予一个碱基的规则是在根节点上，任取 u_{root} 中任何一个元素 x 作为碱基将父节点的核苷酸 u 赋予子节点 v 。

$$\begin{cases} x, & \text{if } x \in \mathcal{U} \\ \text{any nucleotide,} & \text{else} \end{cases}$$



替换数目为 $S_{ij}=3$

5.Sankoff 算法

本算法采用动态规划计算 S_{ij}

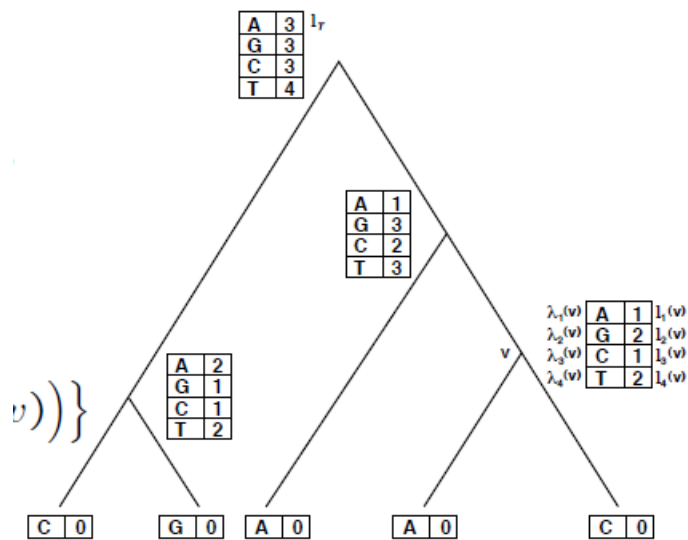
初始：把任何内部边缘当做树的根，然后将叶节点的状态赋值为 0.

递归：使 v 是 T 一个的内结点，令 $\lambda_i(v)$ 是节点 v 的第 i 个状态， $l_i(v)$ 是当将 λ_i 分配给 v 时，以 v 为根的子树的长度。

foreach $\lambda_i(v)$, **do**

$$l_i(v) = \sum_{(w \text{ child of } v)} \min_j \{l_j(w) + c(\lambda_j(w), \lambda_i(v))\}$$

$c(\lambda_j(w), \lambda_i(v))$ 是转换的成本函数。



根映射中的最小条目是简约树长度 l_t 。sankoff 算法从叶节点到根节点依次计算成本向量，当得到的根节点成本向量后，取最小的成本状态，当两个或多个状态在根节点上同时有最小成本时，从这些状态里随机选取一个作为根节点的状态。

给出 n 个 taxa，有多少 binary trees

Unrooted

rooted

$$U_n = \prod_{i=3}^n (2i-5),$$

n = number of taxa

$$R_n = \prod_{i=3}^{n+1} (2i-5)$$

n	U_n	R_n
2	1	1
3	1	3
4	3	15
5	15	105
6	105	945
7	945	10395
8	10395	135135
9	135135	2027025
10	2027025	34459425

$$R_{20} = 8\ 200\ 794\ 532\ 637\ 891\ 559\ 375$$