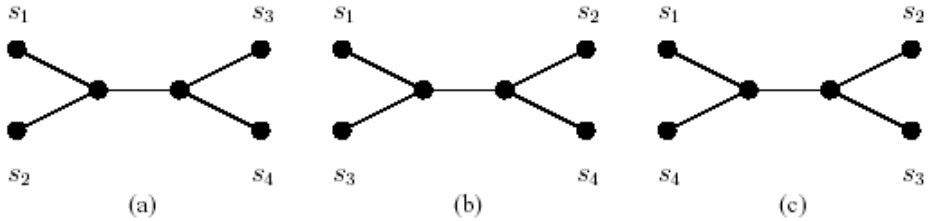
Summary of the beginning paragraph:

- There will be internal nodes which represent unknown species and the length of each edge (*a*, *b*) represents the time needed to evolve from *a* to *b*.

5.1

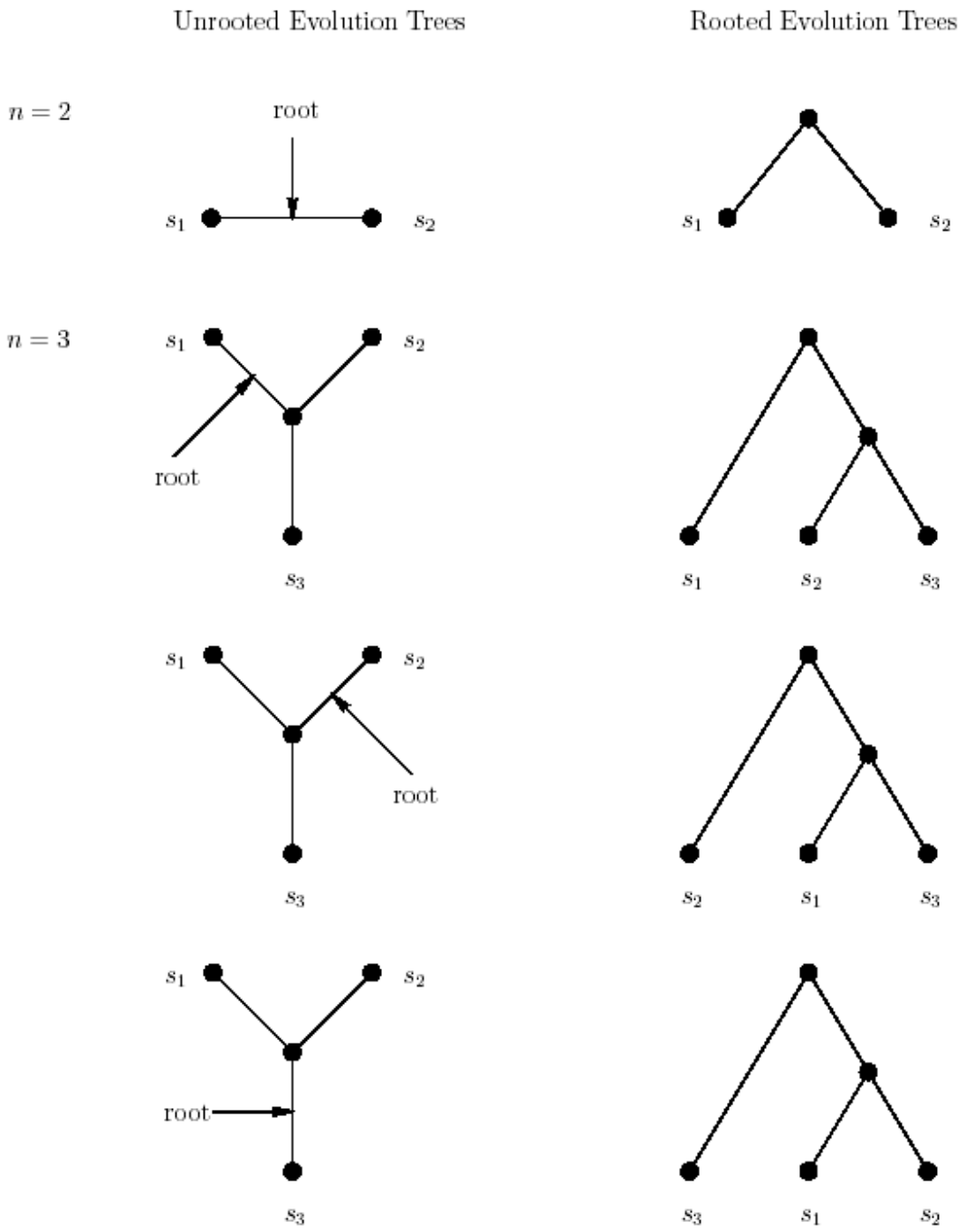
- **Key points about evolution trees:**
  - **I.** ***Only leaf nodes*** denote *species*.
  - **II.** There are two kinds evolution trees: unrooted and rooted
  - **III.** In a rooted evolution tree, the degree of each internal node is **3**. (Except for the root node)
  - **IV.** In an unrooted evolution tree, the degree of each internal node is also **3**. (It can have four species.) See the figure below.



<Note! The *marked* nodes are *species* (leaf nodes)!! >

  - V. The input is ***always a distance matrix*** all among the species. ***Besides, we always assume that the distances satisfy the triangular inequality relationship.***
  - VI. If the evolution tree is ***rooted***, then the ***distances from the root to all leaf nodes are the same.***
  - **VII.** Let $dt(s_i, s_j)$ denote the distance between species s_i and s_j . Let $d(s_i, s_j)$ denote the distance between s_i and s_j in the ***distance matrix***. Then $dt(s_i, s_j) \geq d(s_i, s_j)$.

- *NE*(*n*) denotes the number of ***edges*** of an unrooted tree. Then ***NE*(*n*) = 2*n* - 3**
- *TU*(*n*) denotes the number of ***unrooted evolution trees*** for *n* species.
- $TU(n+1) = NE(n) * TU(n) = (2n-3)TU(n) \text{ or } TU(n) = (2n-5)TU(n-1)$
  $\Rightarrow TU(n) = (2n-5)(2n-7)\cdots 1$
- Key to convert an unrooted tree to a rooted tree:
  $\Rightarrow$ By ***splitting*** *any edge* of the tree and ***adding*** *a root node*:
  View the graph below:



- Let *TR*(*n*) denote the number of ***rooted trees*** for n species. Then:
  ***TR*(*n*) = (2*n*-3)*TU*(n) = (2*n*-3)(2*n*-5)(2*n*-7)⋯1 = *TU*(*n*+1)**
  (Note: ***TR*(*n*) = *TU*(*n*+1)**)

∴ Num(rooted evolution tree) >> Num(unrooted evolution tree)

So we consider the unrooted evolution trees. But we don't have a "root" to explain evolution. ⇒ ***Adding a species*** which is *exceedingly different* from the species which we are analyzing.

- This *outlier species* will ***cause a long link*** $\Rightarrow$ So we can use that to ***identify a root!!***

5.2

- We are asked to construct an evolution tree to properly reflect these distance (In the distance matrix)
- Here are some different evolution trees:

  I. Minimax Evolution trees:

  Max($dt(s_i, s_j) - d(s_i, s_j)$ *) is minimized.*

  II. Minisum Evolution trees:
  The total sum *of all pairs of distances among* ***leaf*** *nodes is minimized.*

  III. Minisize Evolution trees:
  The total length *of the tree is minimized.*

- **Minimal spanning tree approach** $\Rightarrow$ *A new approach to construct rooted evolution trees.*
-

	Minimax	Minisum	Minisize
Unrooted	NPC	NPC	Unknown
Rooted	$O(n^2)$	NPC	NPC

5.3

- In this section we'll introduce a *minimax evolution tree algorithm* for *rooted evolution trees*. This algorithm is recursive. **Basic principle is as follows:**

  I. Let s_i and s_j be the two species having the longest distance in the distance matrix.

  II. Our rooted evolution tree will have two subtrees (Use algorithm 5.1 to obtain), which we name them T_i and T_j, and $s_i \in T_i$, $s_j \in T_j$

  III. $dt(\text{root}, s_i) = dt(\text{root}, s_j) = \frac{1}{2} d(s_i, s_j)$

  $\Rightarrow$ ***The longest distance is exactly preserved.***

IV. $dt(s_i, s_j) = d(s_i, s_j)$.

Since $dt(s_i, s_j) \geq d(s_i, s_j)$, $Max(dt(s_i, s_j) - d(s_i, s_j))$ *is* minimized

-

Algorithm 5.1 A Rooted Minimax Evolution Tree Algorithm.
Input: A Distance Matrix of a Set *S* of *n* species $s_1, s_2, \cdots, s_n$.
Output: A Rooted Minimax Evolution Tree for *S*.
Step 1: If *S* contains only one species *x*, return node *x* as the tree.
Step 2: Find the longest $d(s_i, s_j)$ in the distance matrix. **Find a minimal spanning tree of *S*.**
Step 3: Find the longest edge *e* in the path linking s_i and s_j in the minimal spanning tree.

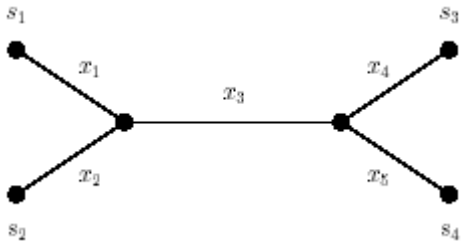
Let S_i and S_j be the two sets of species obtained by breaking edge *e*.

Step 4: Use this algorithm *recursively* to find subtrees T_i and T_j for S_i and S_j respectively.
Step 5: Construct a rooted tree with T_i and T_j as subtrees. Let the distance from the root *r* of this tree to the root of $T_i(T_j)$ be $h_i(h_j)$. Set $h_i(h_j)$ so that

$$dt(r, s_i) = dt(r, s_j) = \frac{1}{2} d(s_i, s_j)$$

5.4

- d_{ij} denotes the distance between s_i and s_j. x_i's
- Suppose that the graph below is the best one for being a minisize evolution tree of our original unrooted evolution tree.



Then, how to determine $x_i's$ for i = to 5 ?

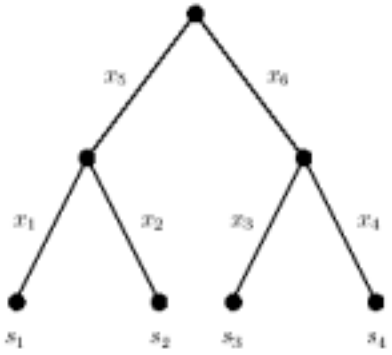
⇒ *By linear programming approach !*

⇒ Minimize $x_1 + x_2 + x_3 + x_4 + x_5$

⇒

$$\begin{aligned} & & x_1 + x_2 & \geq d_{12} \\ & & x_1 + x_3 + x_4 & \geq d_{13} \\ & & x_1 + x_3 + x_5 & \geq d_{14} \\ & \text{Subject to} & x_2 + x_3 + x_4 & \geq d_{23} \\ & & x_2 + x_3 + x_5 & \geq d_{24} \\ & & x_4 + x_5 & \geq d_{34} \end{aligned}$$

- And suppose that our evolution tree is a rooted one as the graph below:



Minimize $x_1 + x_2 + x_3 + x_4 + x_5 + x_6$

$$\begin{aligned} & & x_1 + x_2 & \geq d_{12} \\ & & x_1 + x_5 + x_6 + x_3 & \geq d_{13} \\ & & x_1 + x_5 + x_6 + x_4 & \geq d_{14} \\ & \text{Subject to} & x_2 + x_5 + x_6 + x_3 & \geq d_{23} \\ & & x_2 + x_5 + x_6 + x_4 & \geq d_{24} \\ & & x_3 + x_4 & \geq d_{34} \\ & & x_5 + x_1 = x_5 + x_2 & = x_6 + x_3 = x_6 + x_4 \end{aligned}$$

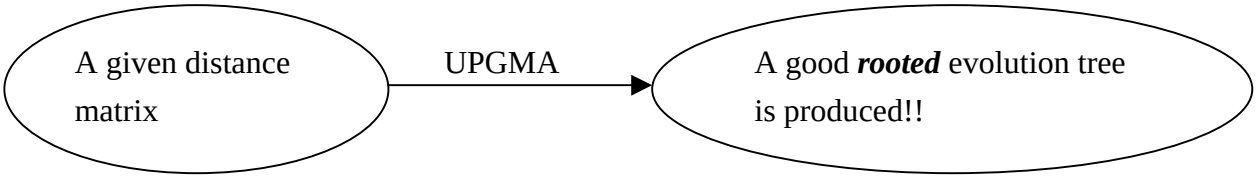
This approach *cannot be used for minimax evolution tree for unrooted evolution trees* because *it is unknown how to formulate this problem as a linear programming problem.*

- *The number of evolution trees is **exponential with respect to n**.*

5.5

- ***The Unweighted Pair Group Method with Arithmetic Mean = UPGMA***

(The spirit of greedy method)



- **Algorithm for UPGMA:**

Algorithm 5.2 The Unweighted Pair Group Method with Arithmetic Mean Algorithm.

Input: A set *S* of *n* species and its distance matrix.

Output: A rooted evolutionary tree structure for *S*.

Step 1: Find two species *x* and *y* such that ***d*(*x*,*y*) is the shortest**.

Step 2: Create a new species, denoted as (*x*,*y*).

Construct a tree using (*x*,*y*) as the root and subtrees rooted at *x* and *y* respectively as the descendants of the root (*x*,*y*).

Delete *x* **and** *y* **from the distance matrix**.

Step 3: If all species have been deleted,

return the tree rooted at (*x*,*y*) and exit.

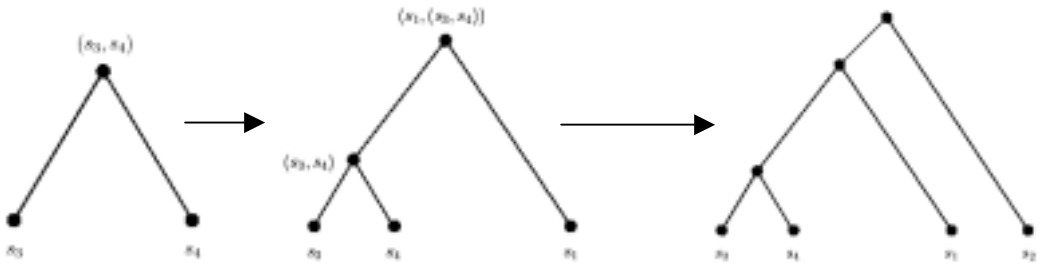
Otherwise update the distance to a new distance matrix.

The distance $d(z,(x,y))$ is calculated as:

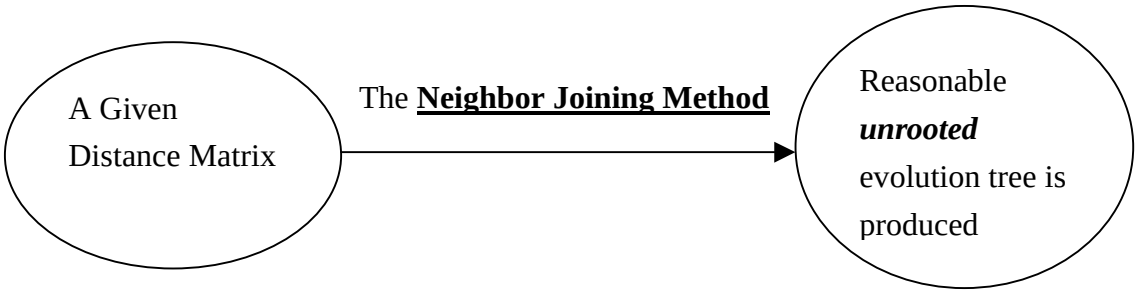
$$d(z,(x,y)) = \frac{1}{2}(d(z,x)) + d(z,y))$$

Step 4: Go to Step 1.

- **Just see the graph below we can know the spirit of UPGMA :**



5.6

- 

- Now we show the algorithm below:

Algorithm 5.3 Neighbor joining method.
Input: A set *S* of *n* species and its distance matrix.
Output: An unrooted evolutionary tree structure for *S*.
Step 1: Construct a **1-star tree *T*** with ***x* as center node** and **species as leaf nodes**.

Calculate average(s_i) = $\frac{1}{n-1}\sum_{j\neq i} d(s_i, s_j)$.

k = 1.

Step 2: If the degree of x is greater than 3, find two species s_i and s_j adjacent to x such that (average(s_i) + average(s_j) - $d(s_i, s_j)$) is maximized.

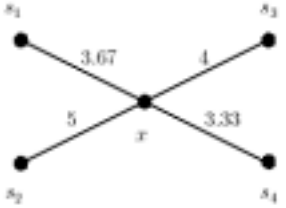
Step 3: Insert an interval node x_k with degree 3 into T such that x_k is connected to x, s_i and s_j.

Step 4: If the degree of x is equal to 3, return T and exit; otherwise $k = k + 1$ and go to Step 2.

- The distance from the unique internal node (that is "*x*") = the mean of the distances from this specie to all other species.

For example: $W(x, s_1) = \frac{1}{3}(d(s_1, s_2) + d(s_1, s_3) + d(s_1, s_4)) = 3.67$

(也就是說, s_1 連到中心點 x 的距離等於 s_1 連到其他點距離的平均值)

- **Note!** 

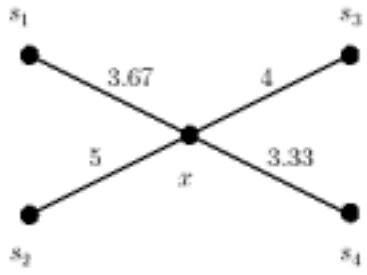
is not an unrooted evolution tree.

Note! In an unrooted evolution tree, the degree of each internal node is also **3**.

(From section 5.1)

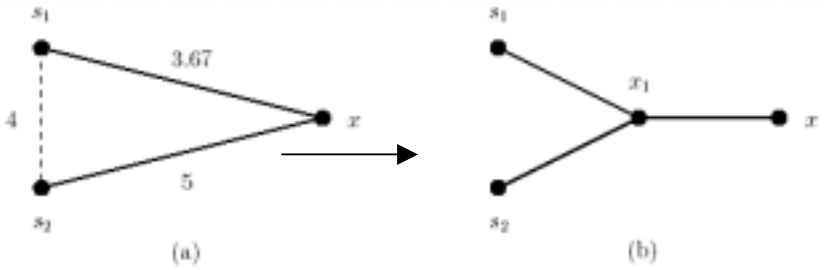
- Let average(s_i) = $\frac{1}{n-1}\sum_{i\neq j} d(s_i, s_j)$
- For example:

	s_1	s_2	s_3	s_4
s_1	0	2	3	3.1
s_2		0	3.6	5
s_3			0	1
s_4				0



We have to determine which two species are to be paired.

Now imagine that s_1 and s_2 are to be paired. Connect s_1 and s_2.



$$NC = \frac{1}{2}(average(s_1) + average(s_2) + d(s_1, s_2))$$

$$= \frac{1}{2}(3.67 + 5 + 4) = 6.33$$

Note: *NC means " **N**ew Connection **C**ost".*

Question: Why does it use the coefficient "$\frac{1}{2}$" **?**

-

$$W(s_1, x_1) = NC - \texttt{average}(s_1)$$

$$W(s_2, x_1) = NC - \texttt{average}(s_2)$$

$$W(x_1, x) = NC - d(s_1, s_2)$$

And $W(s_1, x_1) + W(s_2, x_1) = d(s_1, s_2)$, *so **the distance between** s_1 **and** s_2 **is preserved.***

Then here comes with the new structure: