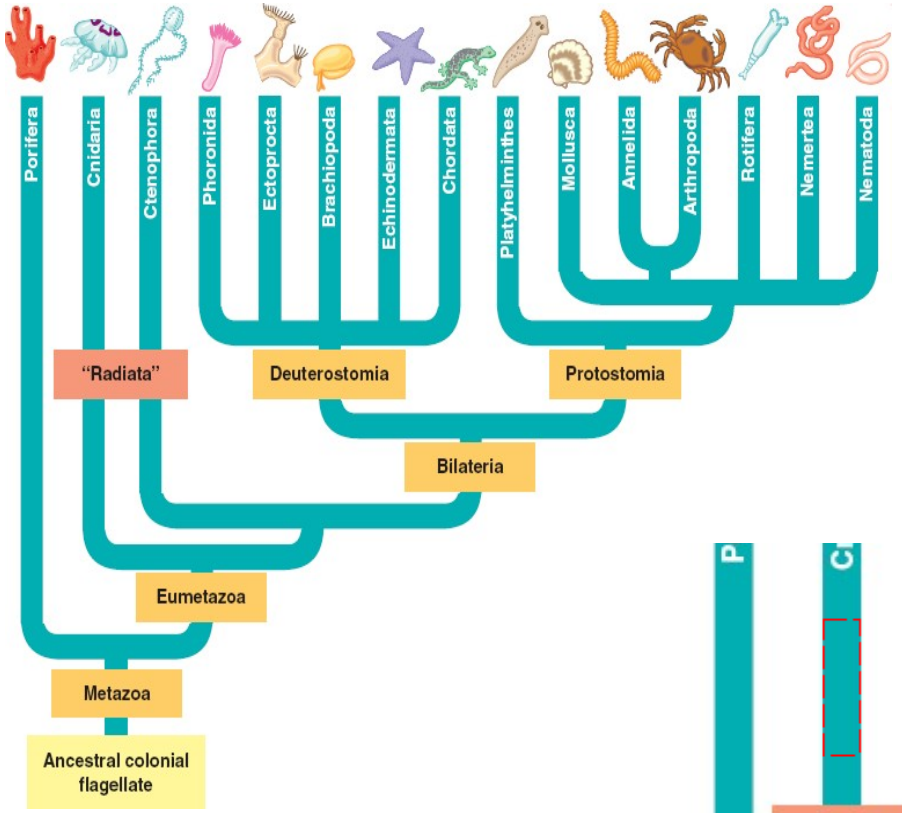
TreeSnatcher Plus

Tutorial 5 Topology issues

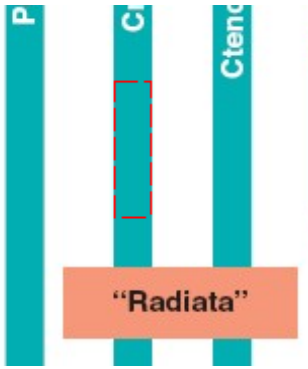
What we want

Please load the image for tutorial 5.
We will look at a tree with multifurcation.
You will learn how to treat the root correctly.



Obstacles:
Text boxes overlap the tree.
The species names are within the branches.

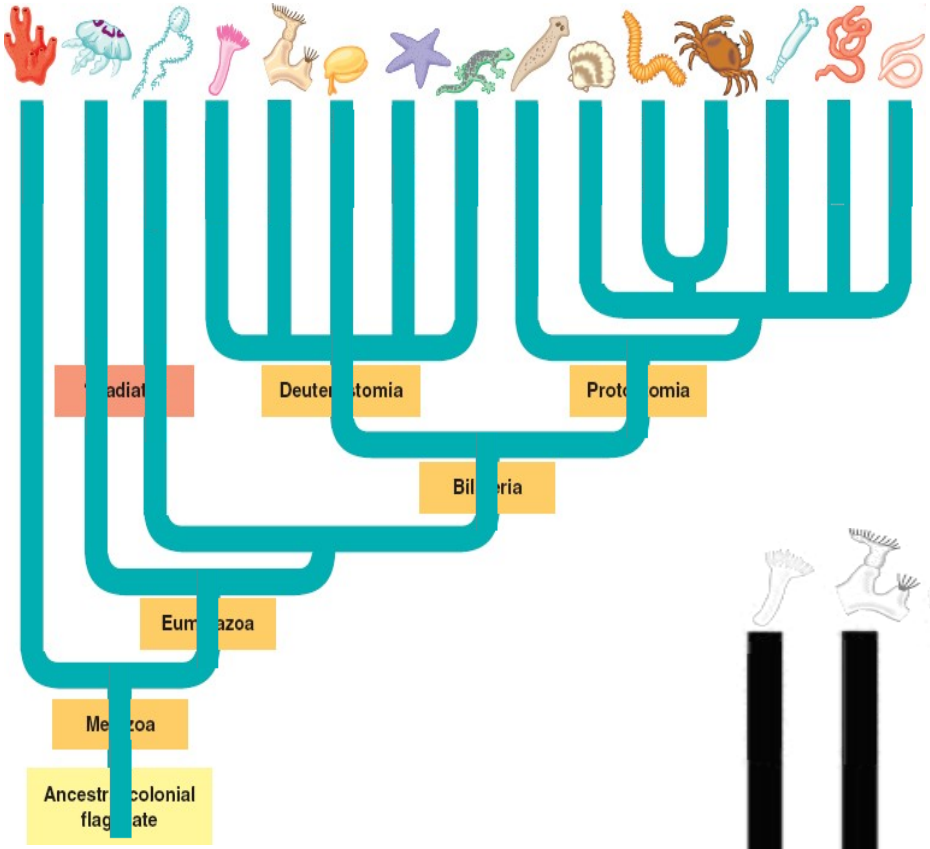
Mark an area of a branch with the **Box Selection**.
It need not be the same as below.



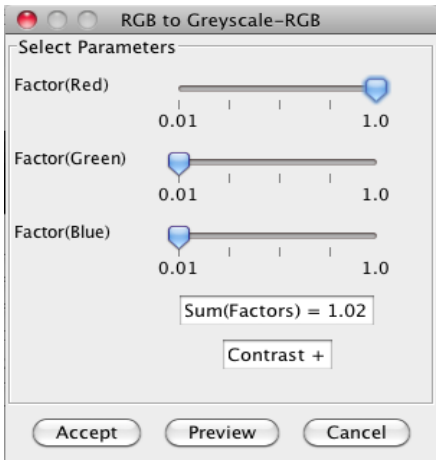
Click **Grab Motif**. TreeSnatcher Plus will now use the image portion as a motif.

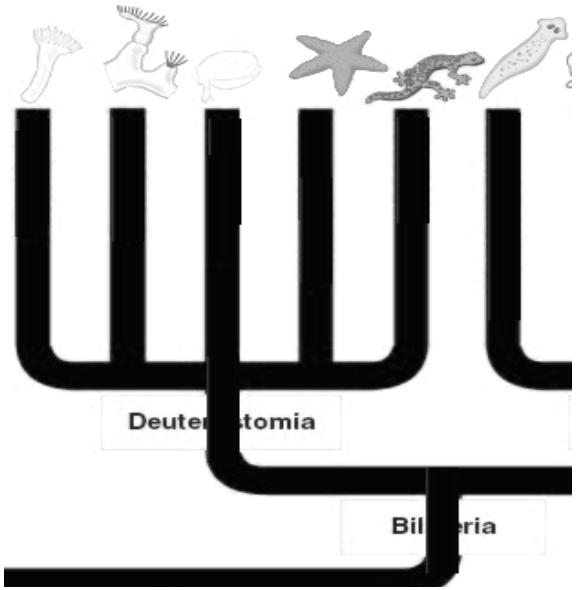
Select the **Stencil** tool.
Now you can paste the motif into the image. Paste it over the species names and over the text boxes.

Try to achieve something similar:



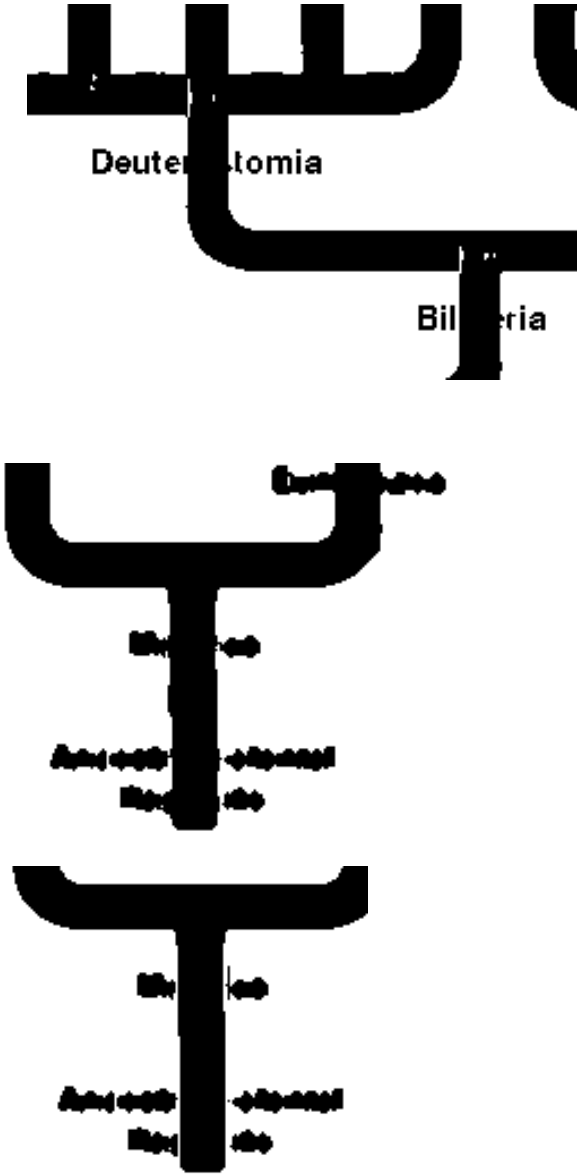
Select **Whole Image**.
We want to get rid of the text boxes.
Therefore, click **Grayscale Conversion.**
In the dialog box, select the values as shown:



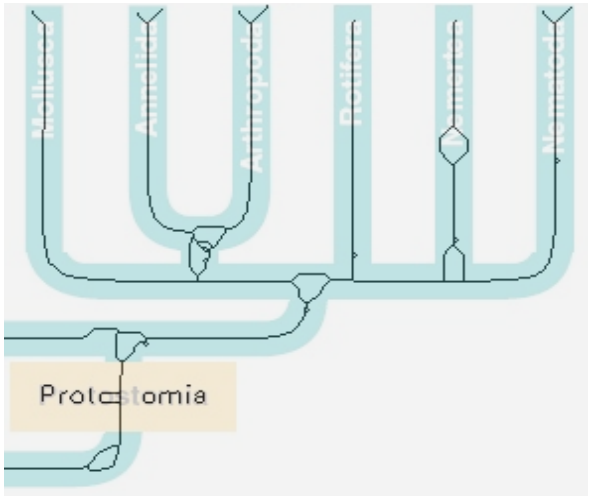


Click **Local Threshold**, or threshold the image manually using **Global Threshold**.

The branches have small gaps here and there that will seriously hamper the thinning.



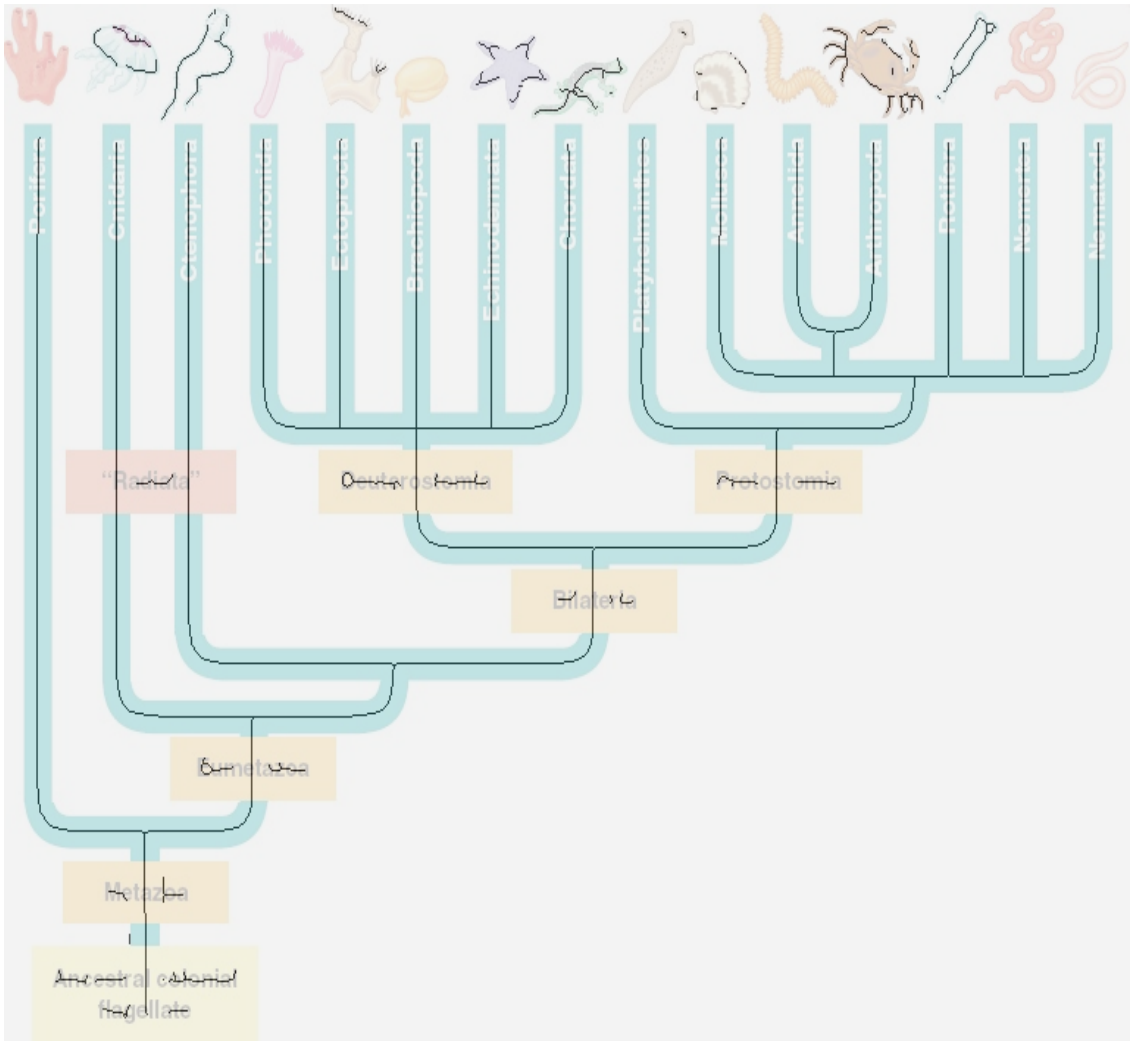
Perform **Thin** now, inspect the resulting tree skeleton.



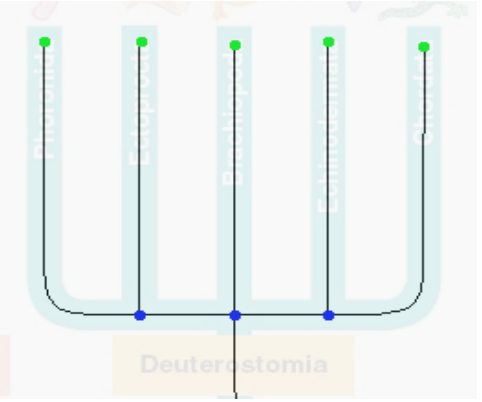
This is completely unusable. You could correct it with the drawing tools, but this would be tedious.
For your amusement, flood the tree and place the nodes.
Afterwards click **Undo** often enough.

Instead, click **Despeckle,** choose box width 4, preview, then accept. The gaps in the branches are gone, but Some artefacts remain that you need to remove.
Use the **White Line** tool to separate the artifacts from the tree. Try to preserve straight contours.

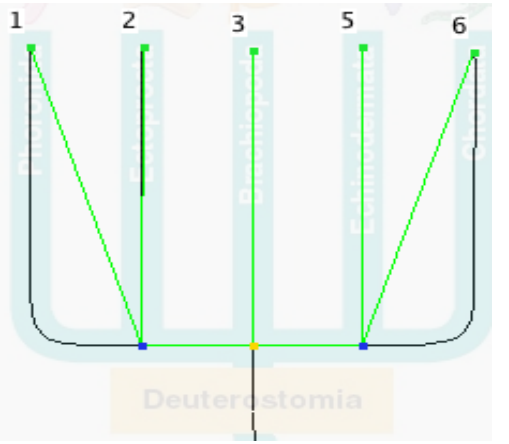
Select **Whole Image**, then **Thin** the image. Does your image look similar?



Issue **Flood Foreground**, then click **Locate Nodes**.

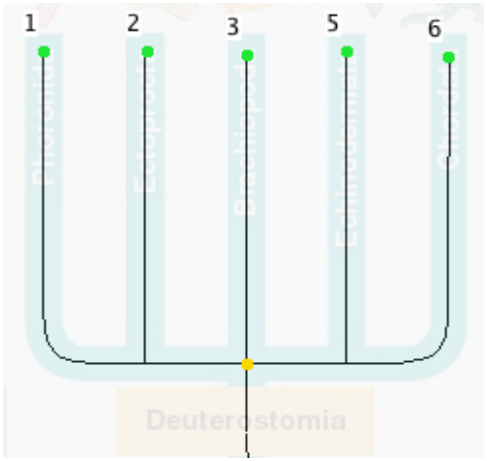


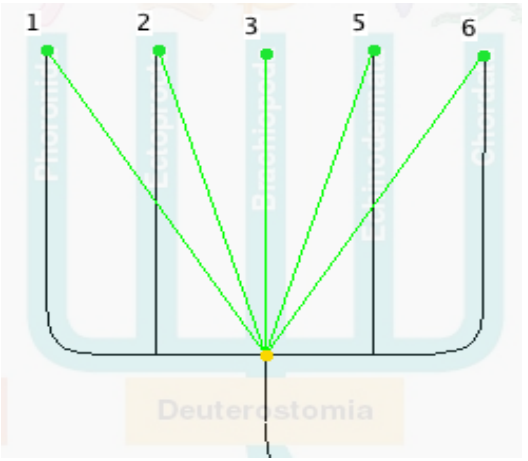
If we go with these inner nodes, the subtree will be bifurcating.



(3, (1, 2), (5, 6));

You might as well want this: (1, 2, 3, 4, 5, 6);

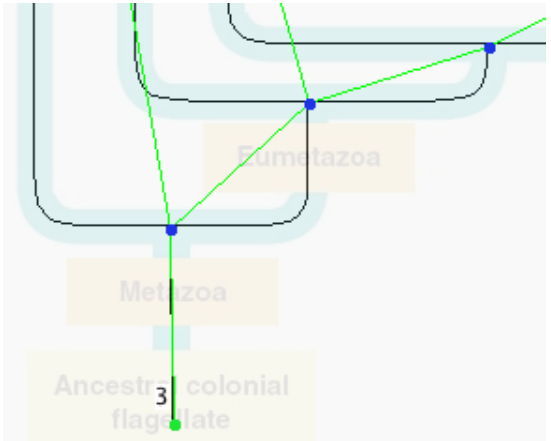




If you also need branch lengths you should inspect the paths using **Show Branch Length Composition**.

Whether a subtree becomes bifurcating or multifurcating is a matter of how many inner nodes you place.

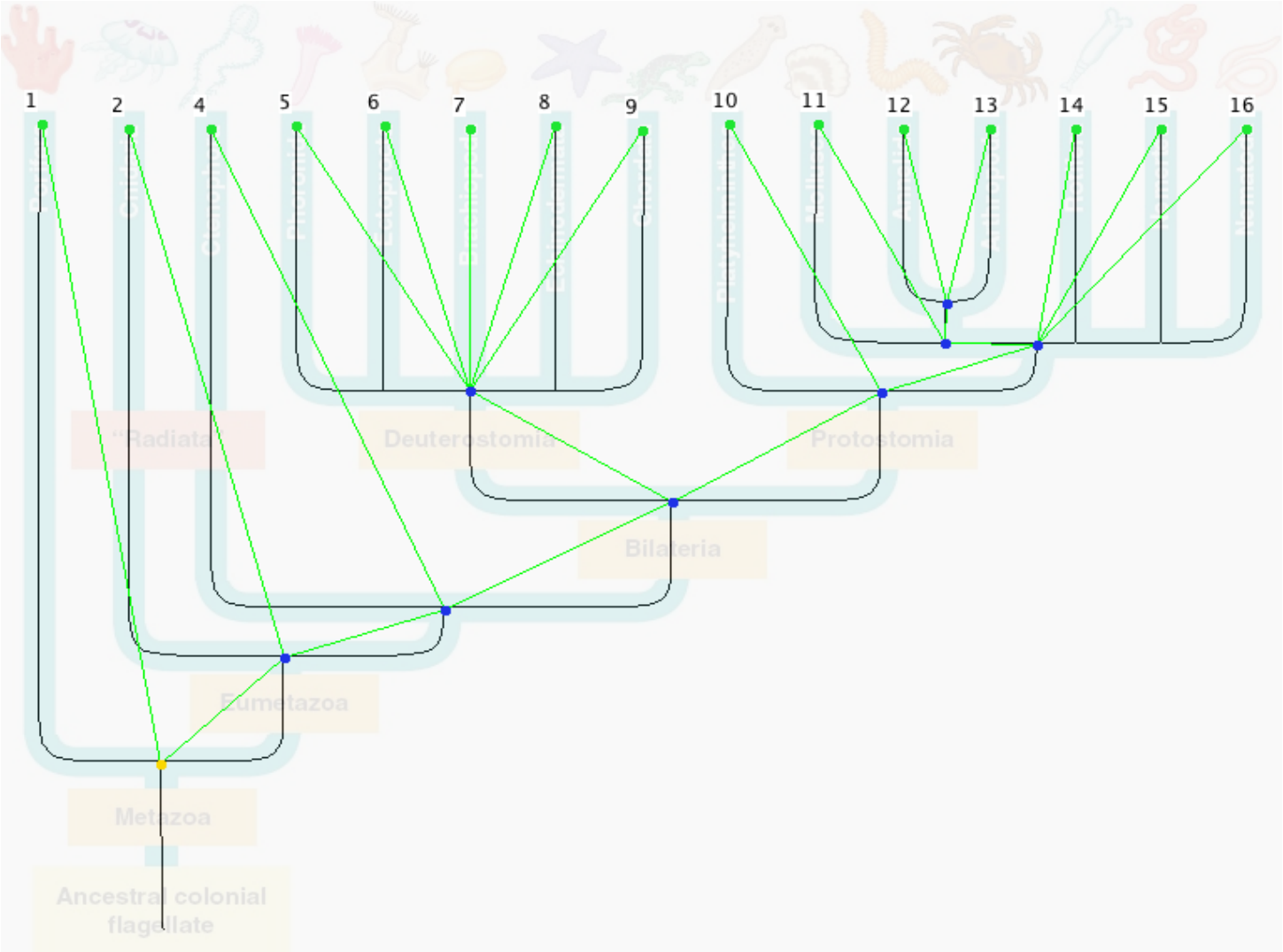
Now let us deal with the root issue.
We know that the tree is rooted. TreeSnatcher Plus cannot distuingish between a tip and the root and, therefore, cannot decide whether a tree is rooted. The Newick format also cannot distinguish between rooted and unrooted trees.



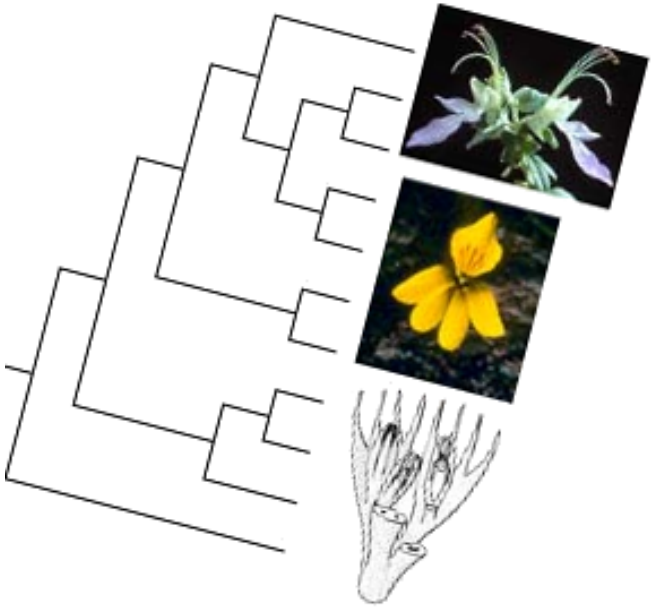
Delete tip 3.
TreeSnatcher Plus will delete the branch between tip 3 and the inner node, then make the inner node the tree origin.

Repeat **Calculate Branches**. If nothing happens, or the program complains that a certain tip is isolated, decrease the look-ahead distance.

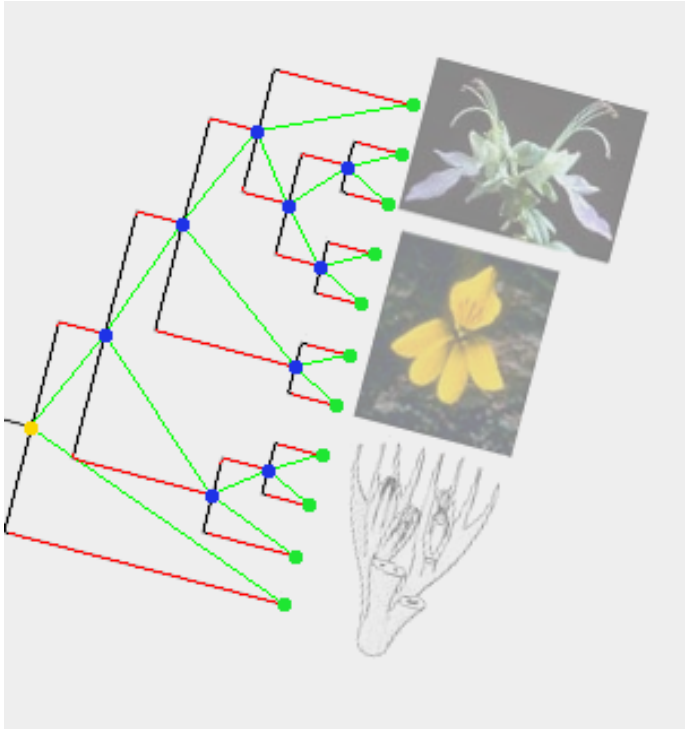
This might be the tree topology we want.

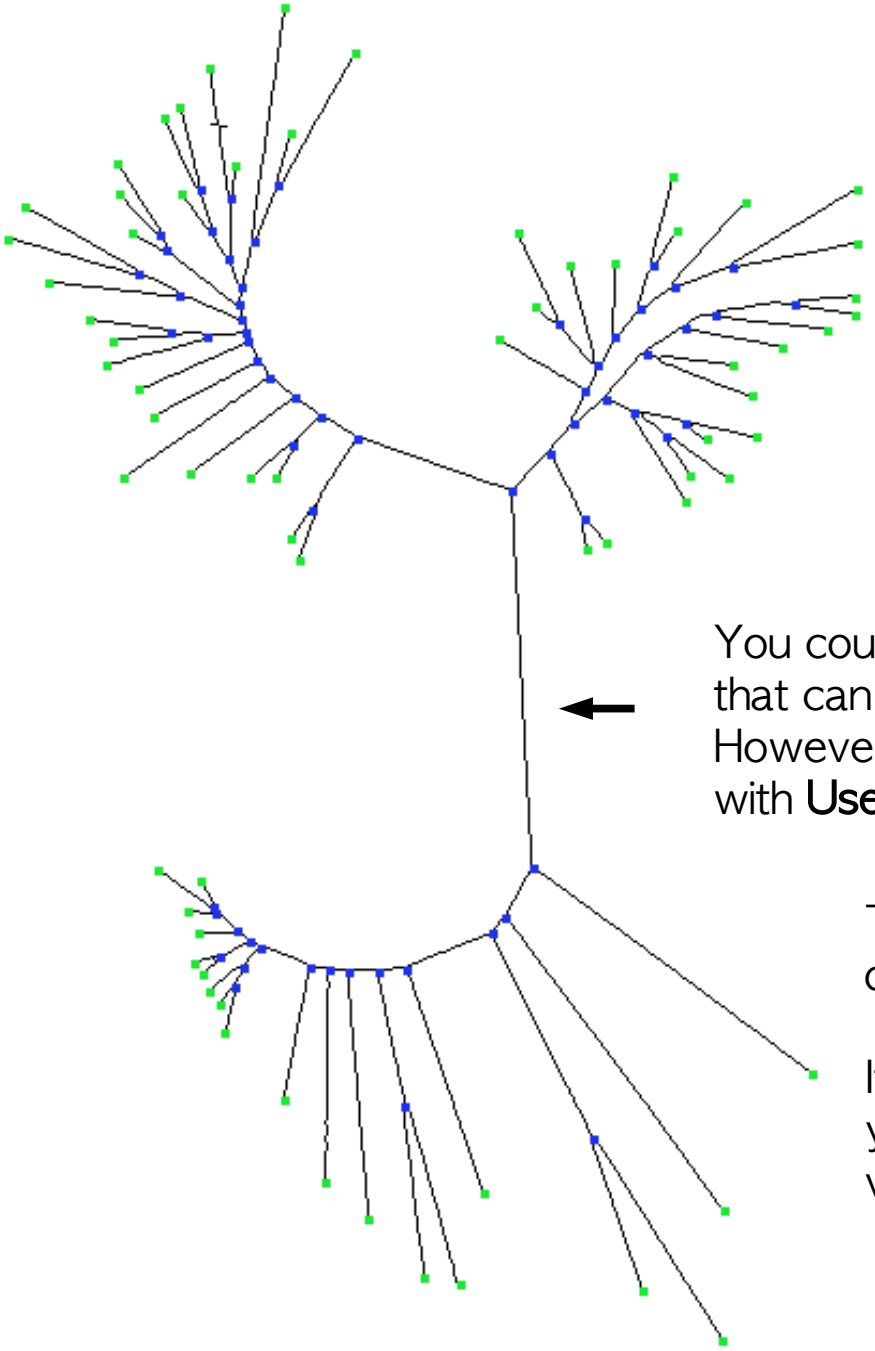


(1, (2, (4, ((5, 6, 7, 8, 9), (10, (14, 15, 16, (11, (12, 13)))))));



TreeSnatcher Plus will also find horizontal and vertical branch segments in images that were photocopied in a hurry…





TreeSnatcher Plus can also handle complicated **Freeform** topologies like this.

After you issued **Locate Nodes** it is likely that you have to move some nodes in the middle of line intersections and to remove nodes.

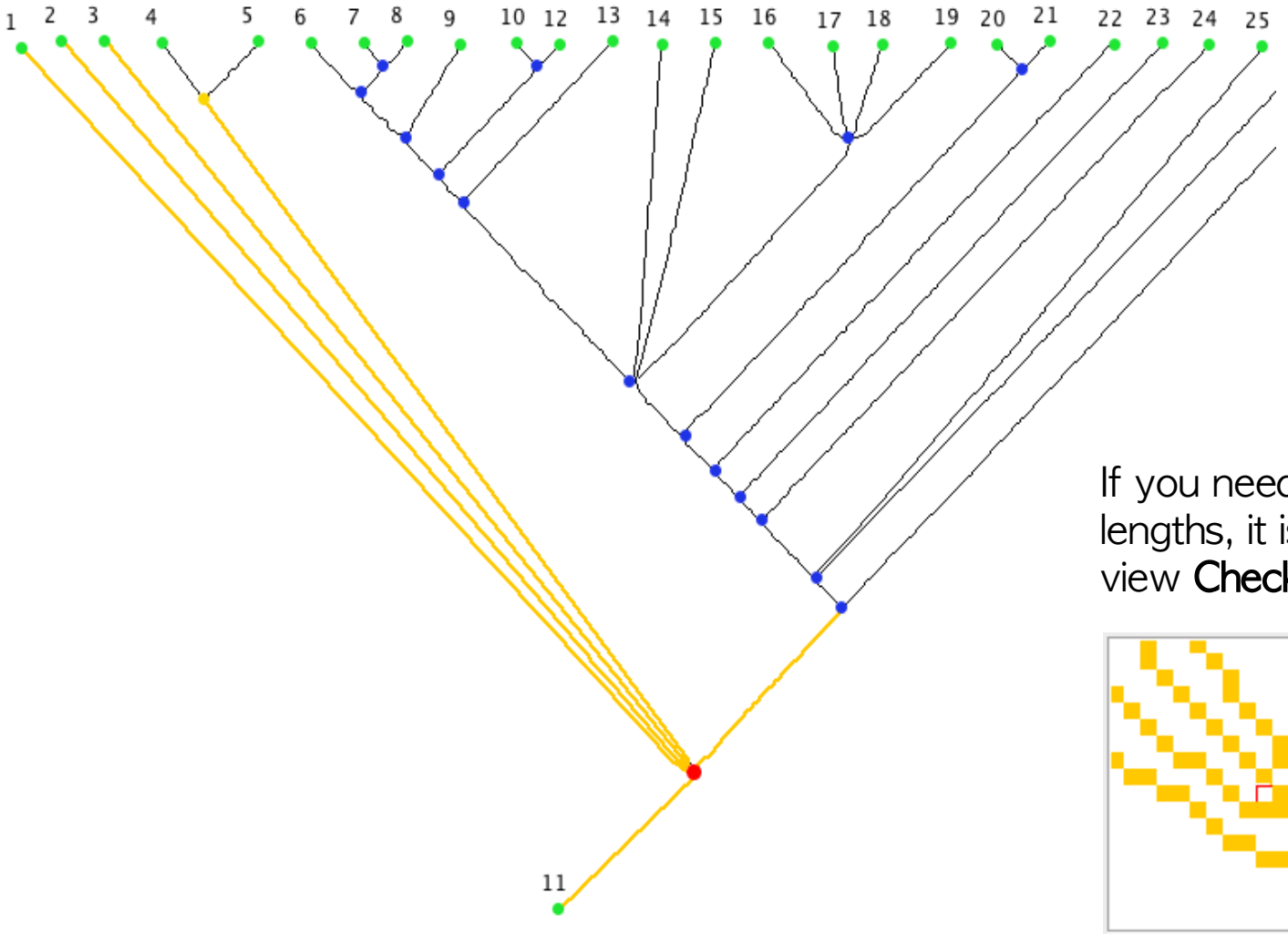
You could insert another inner node here that can serve as a kind of root for the tree. However, you have to designate it as origin with **Use Node as Origin**.

The **look-ahead distance** is two times the smallest distance between a pair of nodes in the tree.

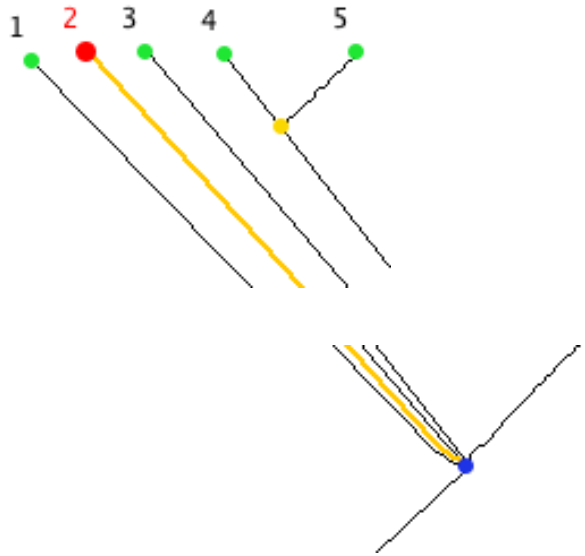
If the program cannot finish the branch calculations you should set the look-ahead distance to a lower value or turn off **Elaborate Path Finding**.

If **Elaborated Path Finding** is turned off, TreeSnatcher Plus uses a simpler algorithm to find the paths. This approach requires you to place all nodes in the center of line intersections.

It is also important to ensure that all paths have a minimum length of ca. 6 pixels incl. node positions. Otherwise they will not be recognized as branches.

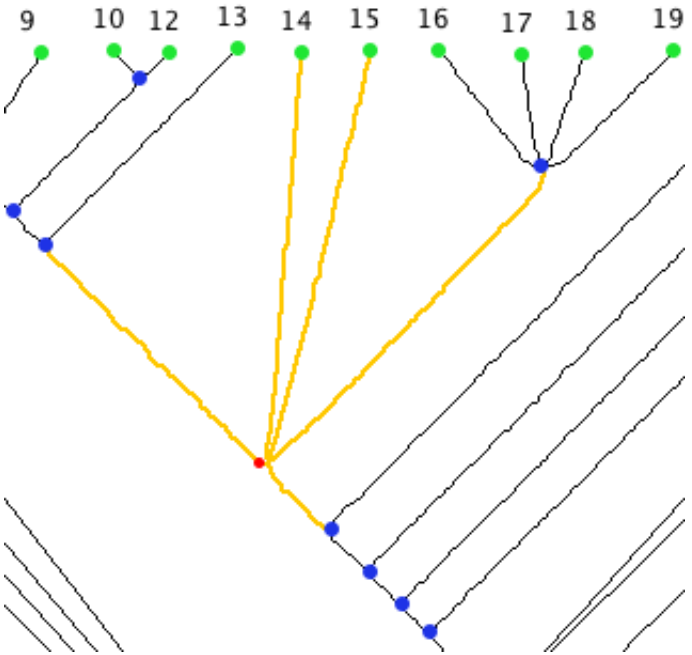


If you need to rely on branch lengths, it is advisable to use the view **Check Linked Branches**.



The path between tip 2 and the inner node cannot be shorter. It exhibits no loops, no detours and connects only these two nodes.

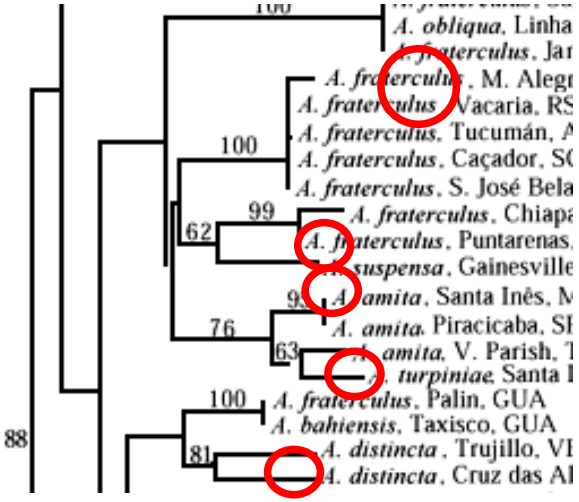
If you encounter a branch length in your tree that arouses your suspicion, i.e. is way too long, you should control the path in the magnifier.

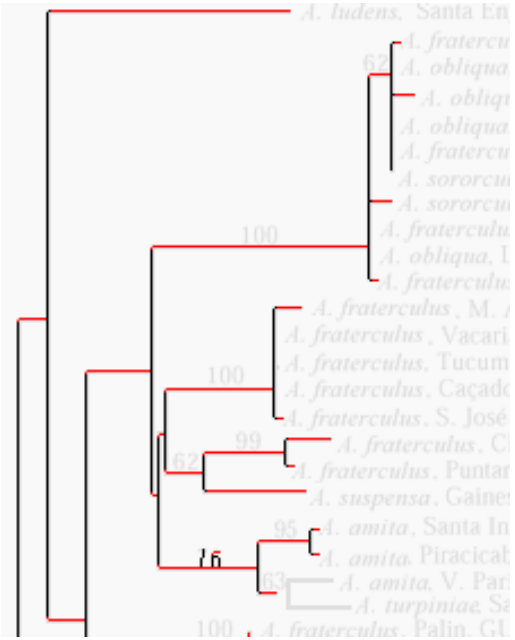


Although you could place the selected inner node more in the center of the intersection, the paths that emerge from this node are perfectly acceptable.

Seeing this, it becomes clear that the quality of branch length measurement in TreeSnatcher Plus depends also on the placement of nodes.

It should be obvious that a graphical method can never exactly reproduce the lengths featured in a numerical data source for the phylogenetic tree.





This is a rectangular tree.
The program expects that the majority of the branches with tips have one direction. From this it determines which direction in the tree is 'horizontal'.

In this illustration the outer branches have multiple directions. There are two interpretations: Either there are some tiny horizontal branches omitted, or the length of those branches is meant to be exactly zero.

Without modifications to the image, the program cannot interpret the topology correctly and/or there will be branches with length zero.

You could modify the image as shown here:
Add at least four pixel long horizontal segments. Then assign a custom length to them. You might want to make them very short. For this to work, you need to activate **Use Mixed Lengths**. Check the result using **Show Branch Length Composition**.

You should conduct all image manipulations in the normal view, **Show All Branches**.