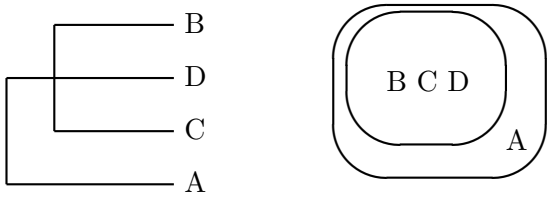
Figure 1: The assumption of A as the outgroup



1 Inferring trees from a data matrix

Last lecture, we considered a character matrix shown in table 1. Taxon A is the outgroup, and the table uses colors to indicate the information content of the characters. Blue for characters with only autapomorphies, green characters for with a synapomorphy that support the assumed separation between the ingroup and outgroup, and red for the character with a synapomorphy that can inform the other parts of the tree.

Table 1: Data with color-coding of character types

	Character #											
Taxon	1	2	3	4	5	6	7	8	9	10	11	12
A	0	0	0	0	0	0	0	0	0	0	0	0
B	1	0	0	0	0	1	1	1	1	1	1	1
C	0	1	1	1	0	1	1	1	1	1	1	0
D	0	0	0	0	1	1	1	1	1	0	0	1

Once again we assume that A is the outgroup (see figure 1).

2 Phylogenetic inference

Figure 2 shows a tree for this dataset with characters mapped onto it. Tick-marks on branches are paired with numbers. These numbers indicate the number of the character(s) in the matrix that support that branch. Each character changes from the plesiomorphic state to the apomorphic state on the branch with the corresponding tick mark.

Note that in table 1, characters #10 and #11 support the tree that puts taxa B and C together. However on this tree character #12 displays homoplasy. Character #12 favors a different tree – the tree with B and D together (shown in figure 3). On *that* tree, character #12 would be inferred to have evolved along the branch that leads to the clade with B+D.

Figure 4 shows a hierarchical characterization in which each character results in a line surrounding the taxa with apomorphic states. The lines are colored according to the charac-

Figure 2: A tree for the data in table 1

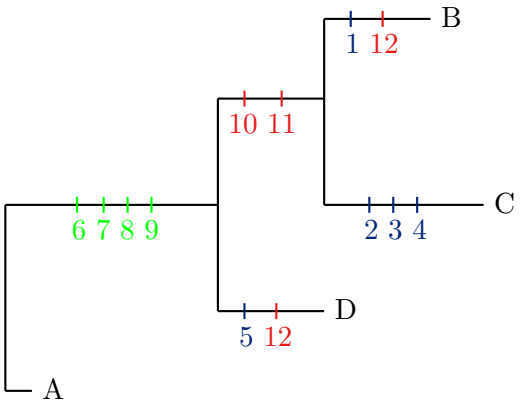
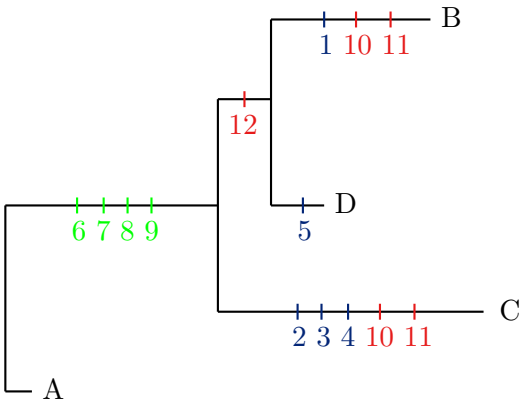


Figure 3: An alternative tree for the data in table 1



ter colors (but not labelled to avoid cluttering the figure). Unlike previous examples (in the previous lectures) the lines demarcating groups in this example must cross each other. Specifically the oval that corresponds to character 12 (one of the red ovals), has to intersect with the lines that represent characters 10 and 11. This crossing of lines is a sign of character conflict.

In Hennigian analyses

1. characters 10 and 11 both "say" that taxa B and C are more closely related to each other than either are to taxon C.
2. character 12 says that B and D are sister taxa (because the share the apomorphic state, the should belong to a monophyletic group that excludes C and A)

But these two statements are in direct conflict – they cannot both be true. One solution is to go back to the organisms and reexamine your assumptions about character and character state homology. Recoding the data could result in a new, homoplasy free character matrix.

This process of going back and forth between character coding and phylogenetic character analysis is the testing of primary homology statements – it is often called "reciprocal illumination."

Figure 4: nesting relationships for data in table 1

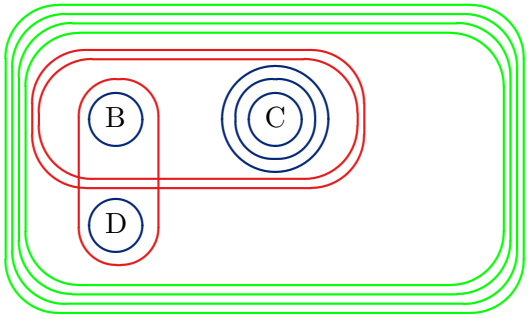


Table 2: The pairwise distance matrix (in number of characters that differ) for the characters shown in Table 1

	Taxon			
Taxon	A	B	C	D
A	-	8	9	6
B	8	-	5	4
C	9	5	-	7
D	6	4	7	-

Note (figure 5 and Table 3) that when there are homoplastic characters in the matrix, Buneman's four point condition will not necessarily hold.

Buneman's four point condition

Four any four taxa (say A, B, C, and D) there are 6 taxon-to-taxon distances that can be estimated from character data:

$$d_{AB}, d_{AC}, d_{AD}, d_{BC}, d_{BD}, \text{ and } d_{CD}$$

There are three ways that you can construct sums of 2 of these distances while making sure that each of the four taxa appear in the formulae:

$$d_{AB} + d_{CD} \quad (1)$$
$$d_{AC} + d_{BC} \quad (2)$$
$$d_{AD} + d_{BD} \quad (3)$$

Buneman's four point condition states that at least two of the sums will be identical to each other, and the third sum will be smaller. The smallest sum corresponds to the sum of distances that pair up sister groups.

Table 3 demonstrates why Buneman's condition should hold (using branch lengths that are

shown in Figure 5), but we can see tha for the data shown in Table 2 the condition does *not* hold. None of the sums shown in the fourth column of the table are identical.

Buneman's condition does not hold in this case becaus the distances in table 2 are not accurate estimates of the evolutionary distance between taxa. This occurs because homoplasy masks evolutionary events. Refer to the mapping of characters in Figure 2. Note that there are six events that occur on the path from B to D (character state changes in characters 1, 5, 10, and 11 and two changes in character 12), but the homoplasy in character 12 results in B and D sharing the same state. Thus, when we calculate the pairwise distance matrix (figure 2) we miss (fail to count) both of these events and estimate a distance of 4. We underestimate the true evolutionary distance by 2 events. So homoplasy confounds distance-based approaches as well as Hennigian analysis.

Figure 5: $AD|BC$ Tree with edge lengths (introducing the notation for Buneman's fourpoint condition

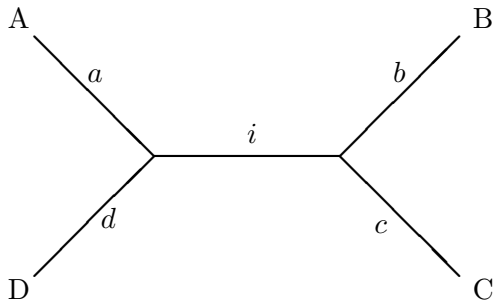
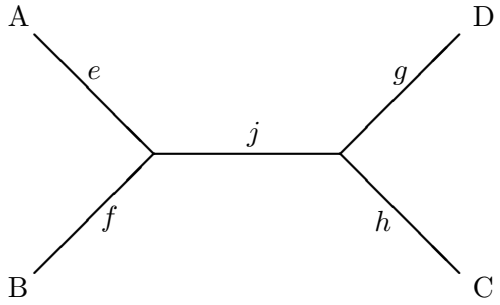


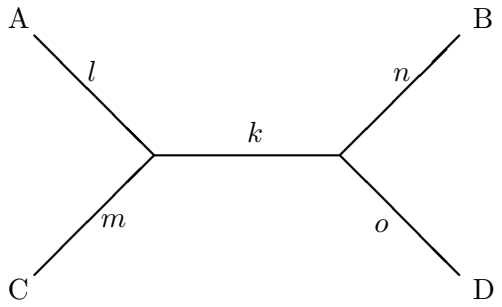
Figure 6: $AB|CD$ Tree with edge lengths (introducing the notation for Buneman's fourpoint condition



Tree	distance sum	sum on Fig 5 tree	sum on Fig 6 tree	sum on Fig 7 tree	from table 2
$AB\|CD$	$d_{AB}+d_{CD}$	$a+b+c+d+2i$	$e+f+g+h$	$l+m+n+o+2k$	$8+7=\mathbf{15}$
$AC\|BD$	$d_{AC}+d_{BD}$	$a+b+c+d+2i$	$e+f+g+h+2j$	$l+m+n+o$	$9+4=\mathbf{13}$
$AD\|BC$	$d_{AD}+d_{BC}$	$a+b+c+d$	$e+f+g+h+2j$	$l+m+n+o+2k$	$6+5=\mathbf{11}$

Table 3: Failure of Buneman four point condition on data set with homoplasy

Figure 7: $AC|BD$ Tree with edge lengths (introducing the notation for Buneman's fourpoint condition



3 Inferring trees from data with character conflict

What if we don't want to return to the character coding and go back to the specimens? We may feel that we have fully examined the characters in an objective way at the beginning stage of the analysis. Returning to the character coding decisions with the intent of eliminating character conflict in our matrix could lead to arbitrary decisions of character rejection. Even if our secondary character-recoding or character-rejection decisions aren't "arbitrary" they could be very subjective and hard to replicate by other workers. Another danger is that the "second iteration" of the data matrix would look cleaner than the real data was – this could lead other scientists to put too much faith in our results.

The predominant phylogenetic techniques today try to avoid recoding the matrix. If the "reciprocal illumination" approach to recoding characters is employed, then it is done at an early stage based on experience from related groups (this leads to knowledge that a certain character is not "good" for a particular group).

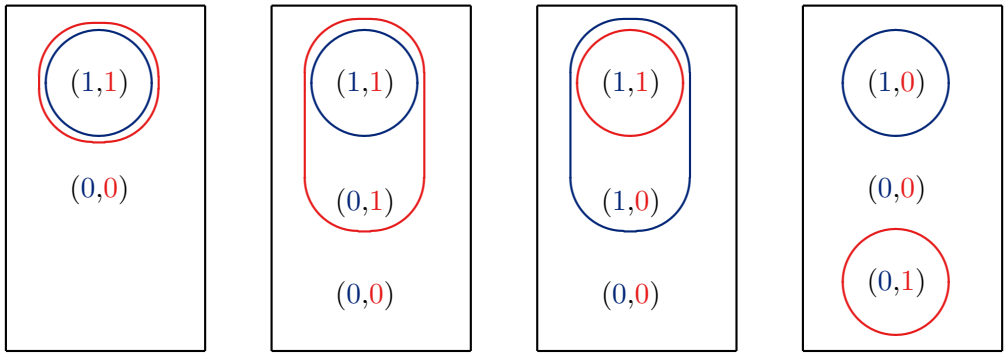
4 Compatibility

One method to infer a tree from a data set that shows incongruent signals of evolutionary relationships is to remove characters from the data matrix that seem to conflict with other characters. To make this practice objective, we could try to remove the *fewest* characters from the matrix required to yield a conflict-free matrix. This is referred to as a maximal compatibility – the determination of the largest number of characters that are all compatible with each other.

Two characters are compatible with each other if there is a tree on which both characters can be mapped without any homoplasy.

Fortunately, there is a easy method to determine if two characters are incompatible with each

Figure 8: The 4 possible nesting relationships two compatible characters



other. If we consider the possible combination that a taxon can display for two characters there are only four possibilities:

(0,0) – a character state of 0 for both characters.

(0,1) – a character state of 0 for the first character and state 1 for the second character.

(1,0) – a character state of 1 for the first character and state 0 for the second character.

(1,1) – a character state of 1 for both characters

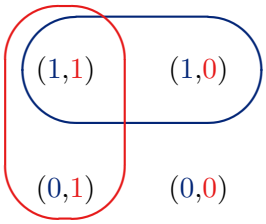
If you consider the states for a pair of characters over all of the taxa, and you see each of these four of these combinations of characters states, then the two characters are **incompatible**.

Figure 8 depicts the combinations of states that could be observed when you look at pairs of character states among compatible characters. In this figure, this first character is shown in blue – taxa with state 1 in this character can be put inside the blue oval. The second character is shown in red. Taxa with state 1 for the second character would fall inside the red ovals. As in other nesting diagrams that we have seen, the ability to draw the diagrams without making lines cross each other indicates the absence of conflict. Note that, this figure only depicts three of the four possible combinations of characters.

Figure 9 shows what happens when all four combinations (0,0), (0,1), (1,0), and (1,1) are seen when two characters are paired up. The red and blue lines must cross – indicating incompatibility.

You can think of compatibility as an attempt to explain the data, by saying that any character conflict (incompatibility between characters) reflects an error in character coding (the decision of how to represent homologous aspects of organisms as a character in a matrix). If we assume that such errors are rare, and that when they occur we should be suspicious of the entire character (column of the matrix), then a logical approach is to figure out how to reject as few of our primary statements of homology as possible to eliminate the conflict. This is exactly what the compatability method does.

Figure 9: Incompatible characters revealed by the presence of all four combinations of character state pairs.



After we have culled the matrix to remove incompatible characters, then we can use Hennigian character analysis to infer the tree.

5 Parsimony

While the compatibility approach is logical, it seems radical to throw out an entire character (column of the matrix) when we detect conflict. It may well be the case that only a few of our character state assignments were in error. As shown in figures 2 and 3, we can map any character matrix onto a tree by allowing some characters to be reconstructed with homoplasy. When we map a character onto a tree and we find that we must posit homoplasy, this means that some of the character states appear to have evolved more than once. This indicates that our homology statements in terms of character state assignments were not perfect.

In the compatibility method in which we throw the entire character out when we need to detect that at least one of our homology statements must be wrong. When we use **parsimony** criterion for phylogenetic inference, we are not this extreme. Instead we express a preference for trees that require the minimum amount of homoplasy.

It turns out that this is equivalent to preferring trees with the fewest number of inferred character state transitions. When we have changes in characters mapped to a tree, then we can simply count the number of tick marks (each mark represents a change of state $0 \leftrightarrow 1$). This number of changes or *steps* is the parsimony score of the tree. We can score each character on a tree and arrive at a parsimony score for each character if we choose. The score for the whole tree is simply the sum of the score for each character.

Table 4 shows the parsimony score for each character in Table 1. Note that only the last three characters have scores that vary between the trees. These characters have scores shown in red. They are the ones that display potential synapomorphies within the ingroup, so they are the potentially informative characters in a Henngian character analysis. They are referred to as *parsimony-informative* because when we are using parsimony to evaluate trees, it is only

characters like these that carry information for inferring the relationships between taxa.

Note that parsimony would prefer the tree that groups B+C together, because it has the smallest score (13) – the minimum number of character state changes required to explain the data. This is not too surprising: the B+C character had only one character (#12) for which we need to invoke homoplasy to explain the characters. In contrast, the B+D tree only requires one change for character #12, but requires two changes for characters #10 and #11. The C+D tree is the worst explanation, according to parsimony – no character shows a synapomorhy uniting C and D, and three characters (#10,#11, and #12) require multiple events to explain the character state distributions.

Table 4: Table 1 with tree scores

	Character #												
Taxon	1	2	3	4	5	6	7	8	9	10	11	12	
A	0	0	0	0	0	0	0	0	0	0	0	0	
B	1	0	0	0	0	1	1	1	1	1	1	1	
C	0	1	1	1	0	1	1	1	1	1	1	0	
D	0	0	0	0	1	1	1	1	1	0	0	1	
Tree	Character Score												Total Score
B+C	1	1	1	1	1	1	1	1	1	1	1	2	13
B+D	1	1	1	1	1	1	1	1	1	2	2	1	14
C+D	1	1	1	1	1	1	1	1	1	2	2	2	15

5.1 Fitch optimization

In the last section, I told you the minimal # of changes required for each character – but how did I know this number?

A brute force method would be to assign character states to all of the internal nodes of the tree. If we have a tree with internal states assignments, then the number of changes required is easy to calculate

A bit about sets

Walter Fitch's (1970 and 1971) algorithms for calculating the parsimony score of a tree use a bit of set notation. A **set** is a collection of objects. The objects are usually referred to as **elements** of the set. The common notation for a set is a pair of curly braces - {}.

The operations on sets used in Fitch's algorithm are the intersection and the union.

The **union** of two sets is a set that contains all of the elements that are in either sets. The union is denote $\cup$. For example:

$$\{2, 5, 6, 9\} \cup \{1, 5, 7, 9, 10\} = \{1, 2, 5, 6, 7, 9\}$$

The **intersection** of two sets is a set that contains only elements that are in *both* sets. The $\cap$ is used to represent an intersection. For example

$$\{2, 5, 6, 9\} \cap \{1, 5, 7, 9, 10\} = \{5, 9\}$$

An empty set, denote with a $\emptyset$, is a set that contains no elements. For example:

$$\{2, 6, 9\} \cap \{1, 5, 7, 10\} = \{\} = \emptyset$$

5.2 Fitch downpass

Walter Fitch showed that you can calculate the best possible parsimony score for a character on a tree (the fewest number of changes-of-state that explain the data), using a single pass down the tree from the leaves to the root.

1. Start with score=0
2. Use the character states observed for each leaf in the tree to create a set of character states for each leaf on the tree.
3. Repeat the following steps as you move from the leaves to the root of tree (if you move down the tree in such a way that you only encounter an ancestral node after you have determined a state-set for the node's descendants, then you can complete the algorithm in a single "traversal" from leaves to the root):
   (a) Let s_A denote the state set of the ancestral node. Since we did not observe this species, s_A is unknown.
   (b) Let s_L denote the state set of the left descendant, and let s_R denote the state set of the right descendant. s_R and s_L will be known because the taxa are observed (if the descendant nodes are leaves), or the state set was inferred from a previous iteration of the algorithm.

   If $s_L \cap s_R = \emptyset$
   then $s_A = s_L \cup s_R$ and add 1 to the current score
   otherwise $s_A = s_L \cap s_R$

You can think of the logic of the algorithm as:

1. "if the two descendants show the same states, then assume that the ancestor had the states that are in common" – this is the part in which you assign $s_A = s_L \cap s_R$ if $s_L \cap s_R \neq \emptyset$.

2. "if the two descendants have different states, then assume that the ancestor could have had *any* state that is in at least one of the children. If this is the case, then we know that we will need to have one change-of-state in this part of the tree, so we add one to the score" – this is the part in which we augment the score, and assign $s_A = s_L \cup s_R$.

Important note: this downpass of the Fitch algorithm only gives us the parsimony score of the character on the tree. It does not (necessarily) give us the most-parsimonious ancestral character state reconstructions for each node. Technically the ancestral state sets are referred to as the *preliminary* state sets for the nodes. We will not cover the algorithm to determine which are the most parsimonious evolutionary scenarios (I am happy to discuss it with you, if are interested).

To evaluate trees using parsimony we do not need to know all of the ancestral character state assignments, we just need the score. So the "downpass" of Fitch's algorithm gives us what we need to score a tree using parsimony.

Figures 10-18 give an example of how the algorithm works.

Table 5: A data matrix consisting of one character in a DNA sequence

Taxon	Character state
A	A
B	C
C	C
D	A
E	A
F	C
G	G
H	A
I	A

Figure 10: Fitch algorithm steps 1 and 2 for that DNA sequence character shown in Table 5

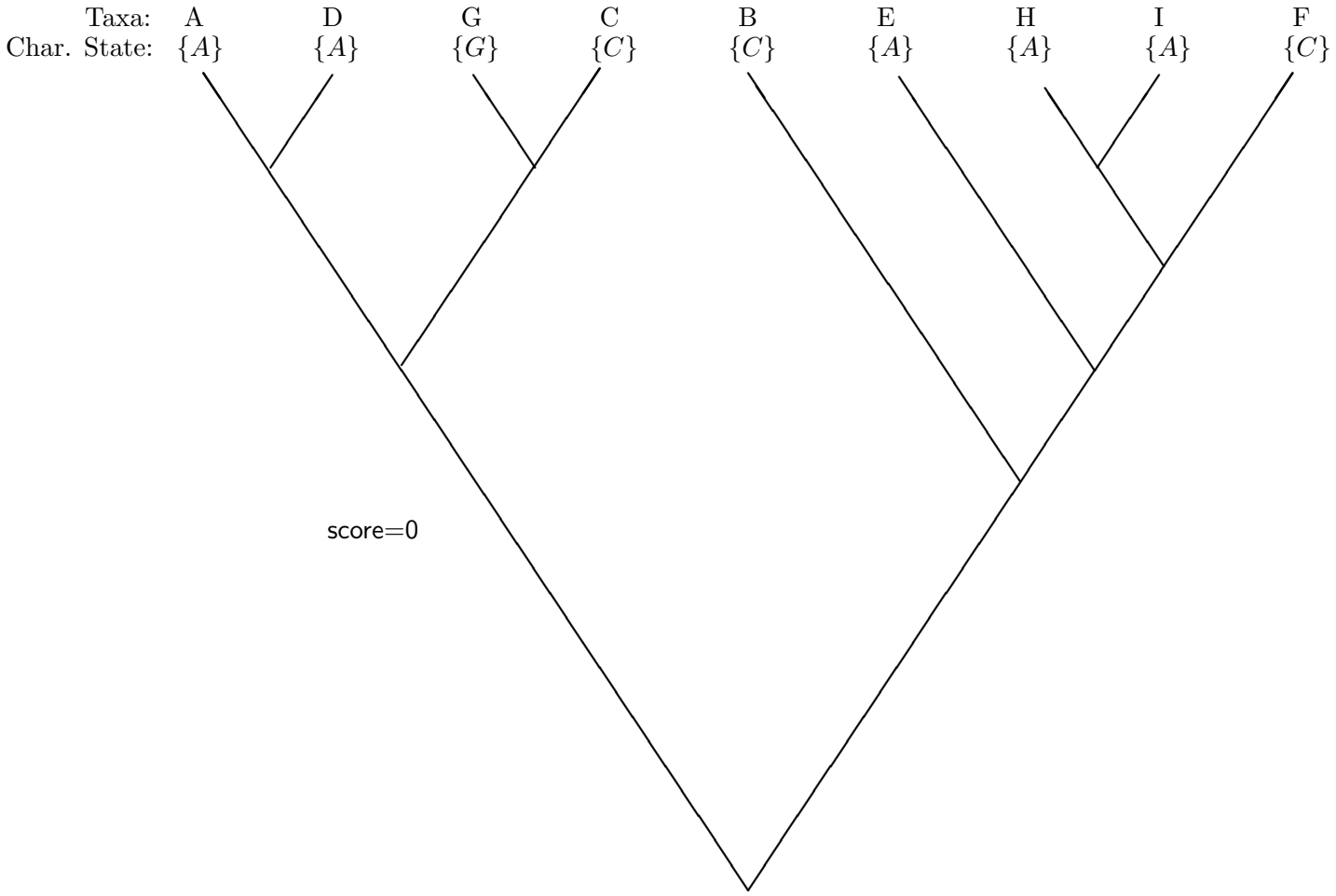


Figure 11: Fitch algorithm step 3 (first ancestral node) for that DNA sequence character shown in Table 5

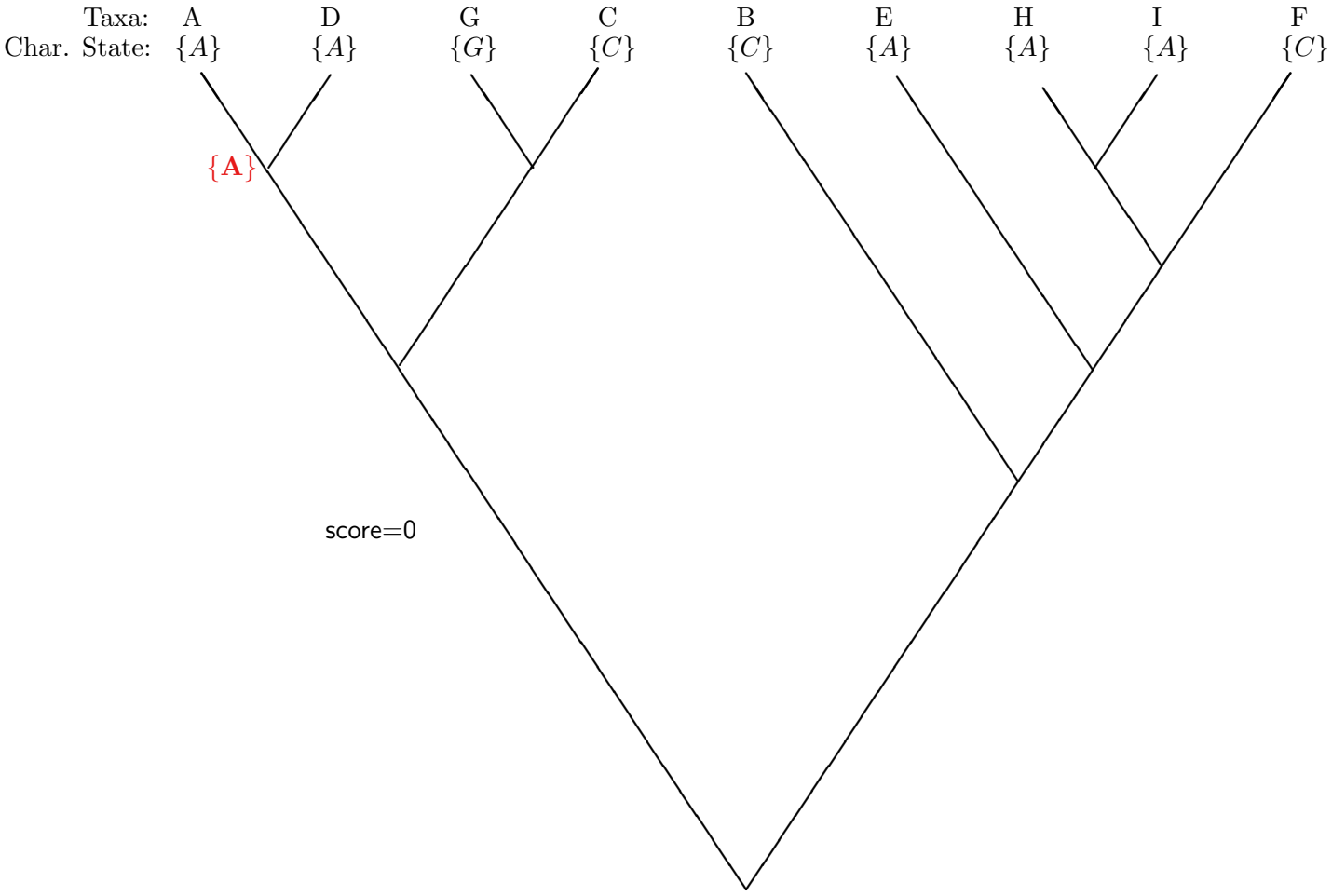


Figure 12: Fitch algorithm step 3 (second ancestral node) for that DNA sequence character shown in Table 5

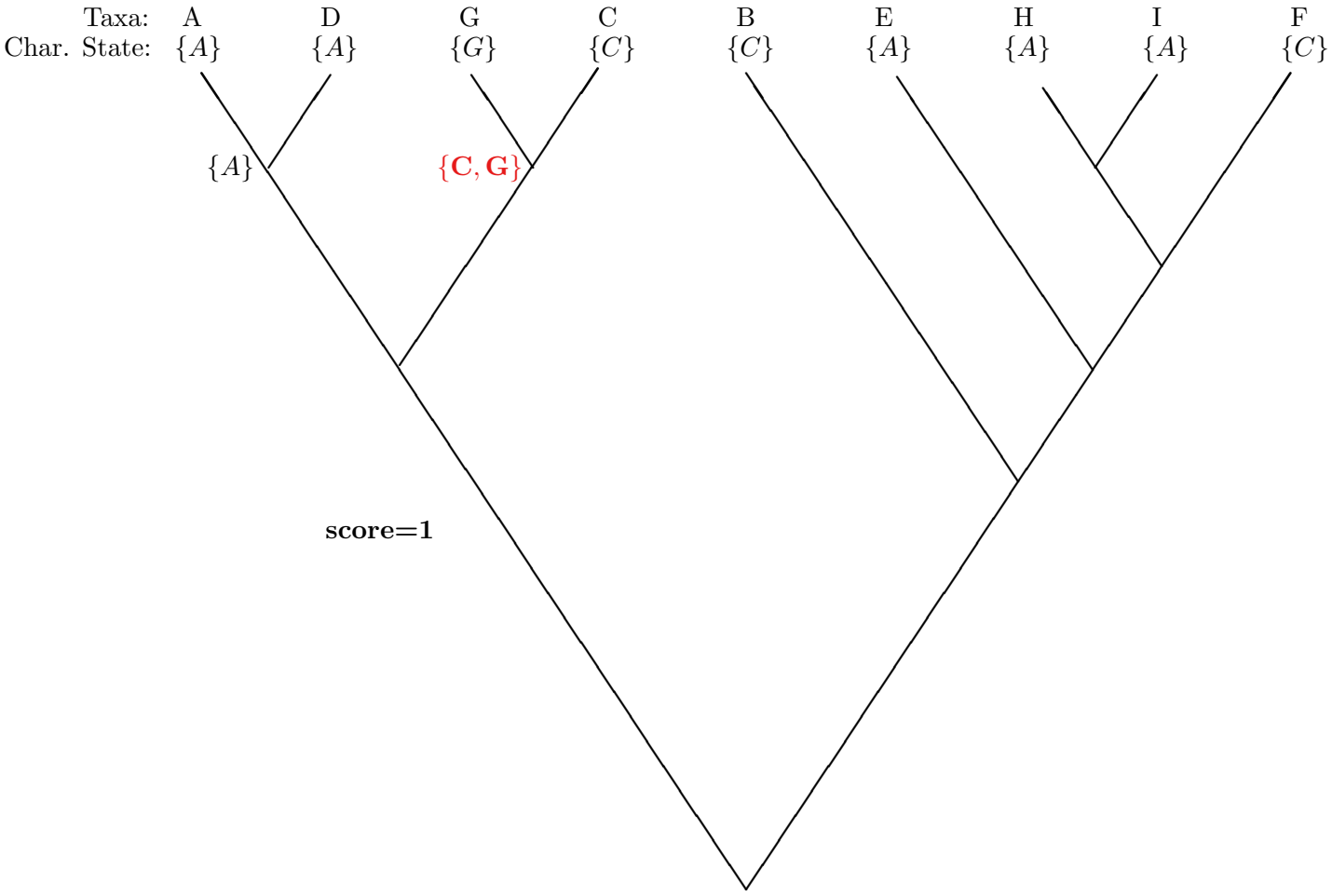


Figure 13: Fitch algorithm step 3 (third ancestral node) for that DNA sequence character shown in Table 5

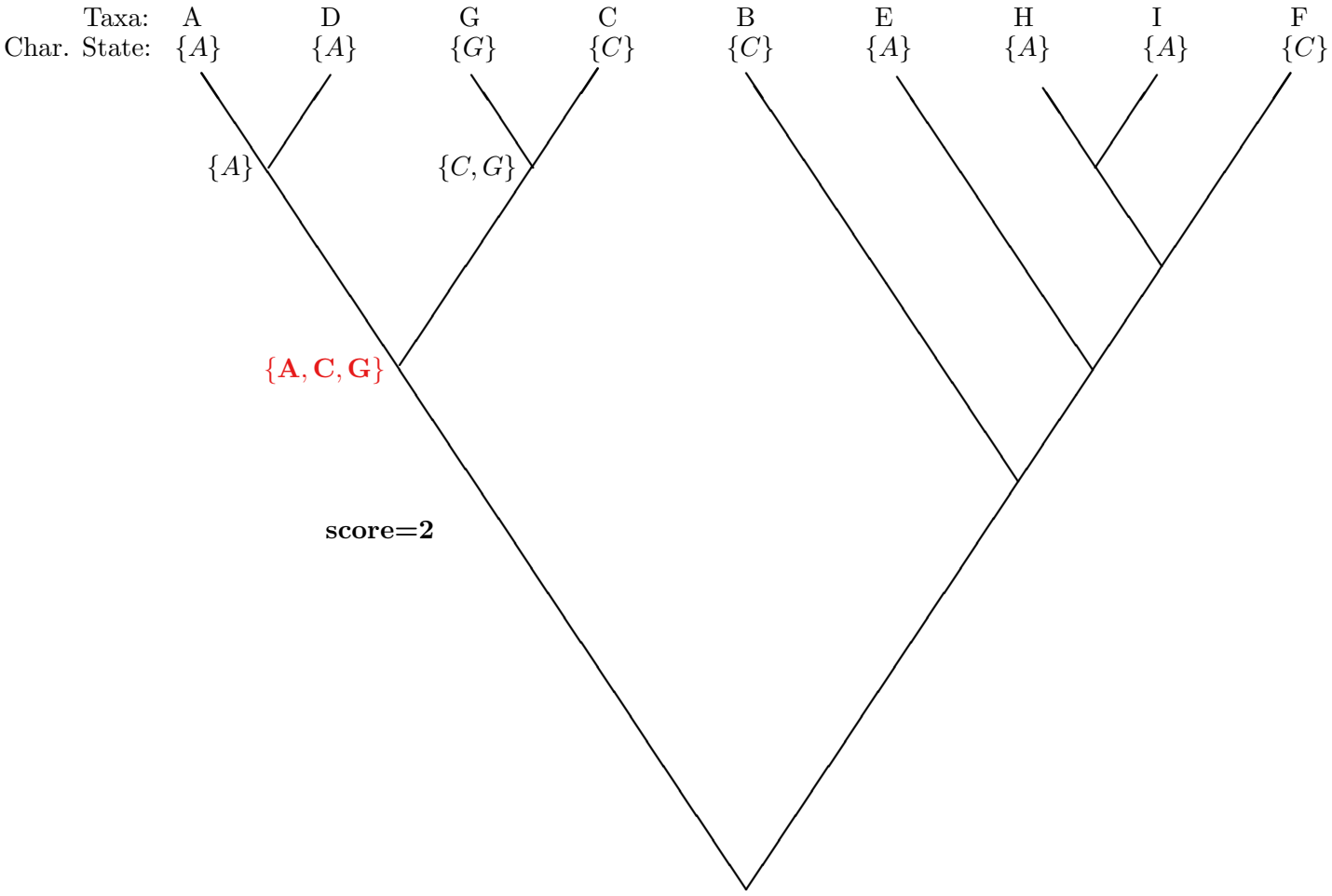


Figure 14: Fitch algorithm step 3 (fourth ancestral node) for that DNA sequence character shown in Table 5

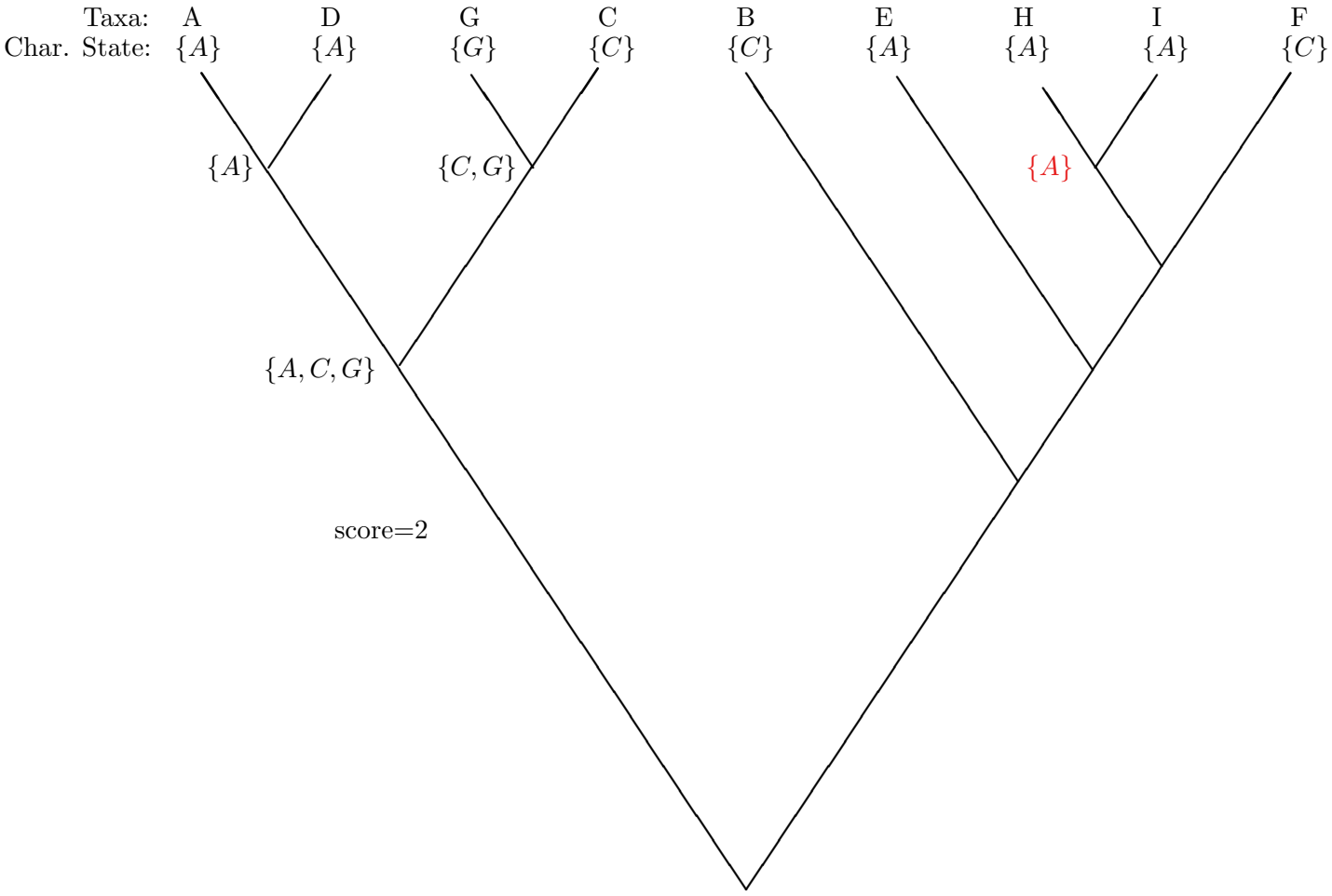


Figure 15: Fitch algorithm step 3 (fifth ancestral node) for that DNA sequence character shown in Table 5

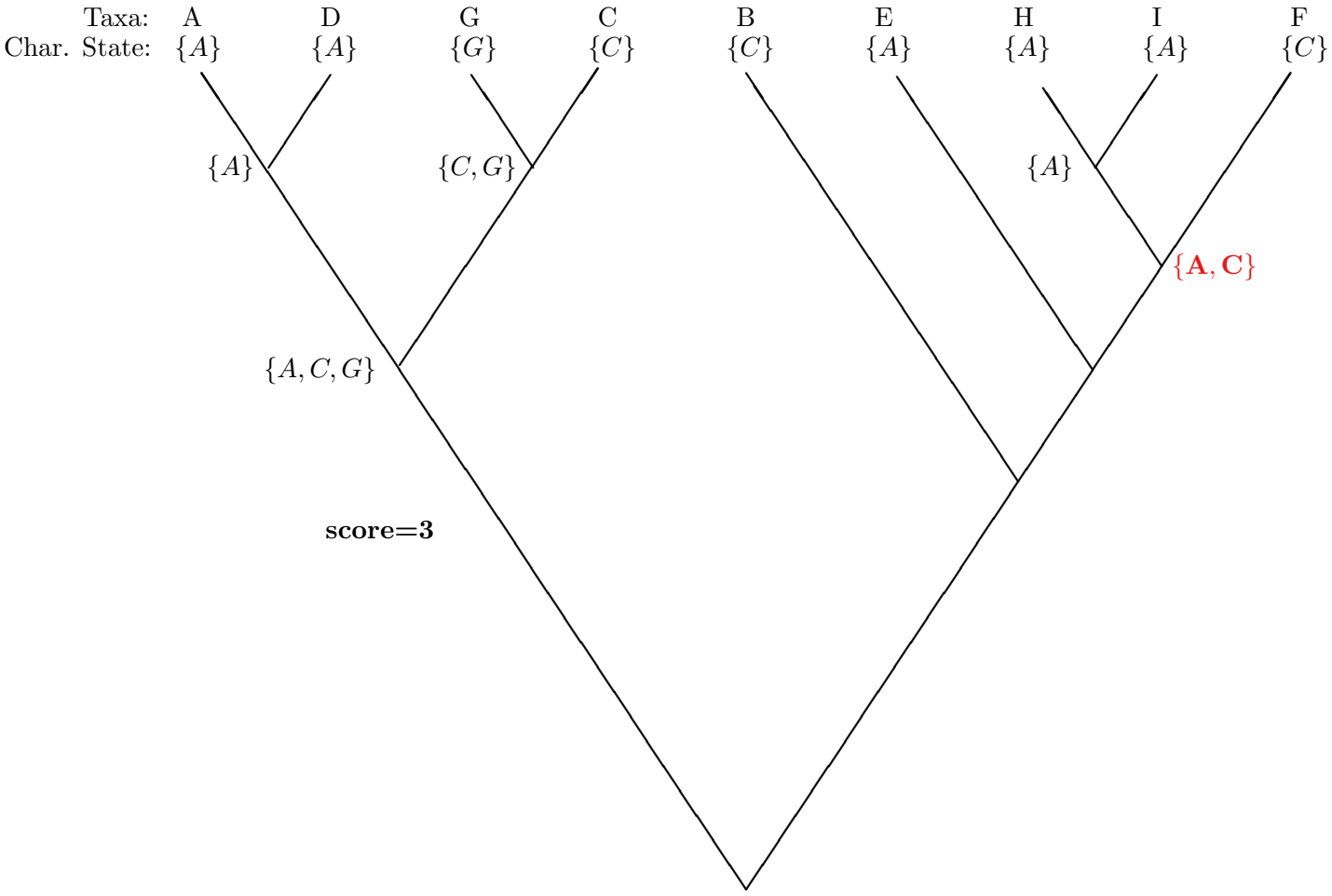


Figure 16: Fitch algorithm step 3 (fifth ancestral node) for that DNA sequence character shown in Table 5

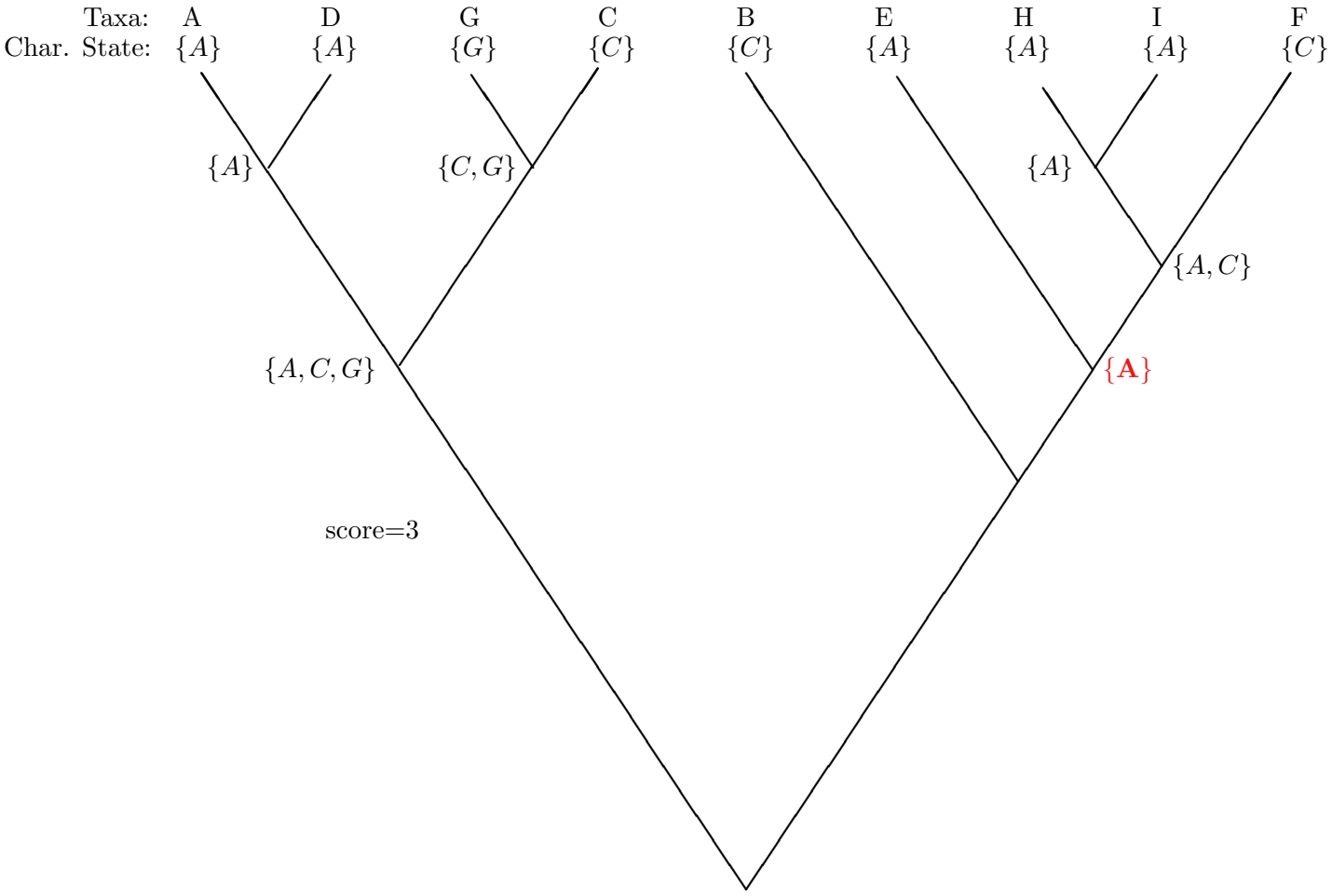


Figure 17: Fitch algorithm step 3 (sixth ancestral node) for that DNA sequence character shown in Table 5

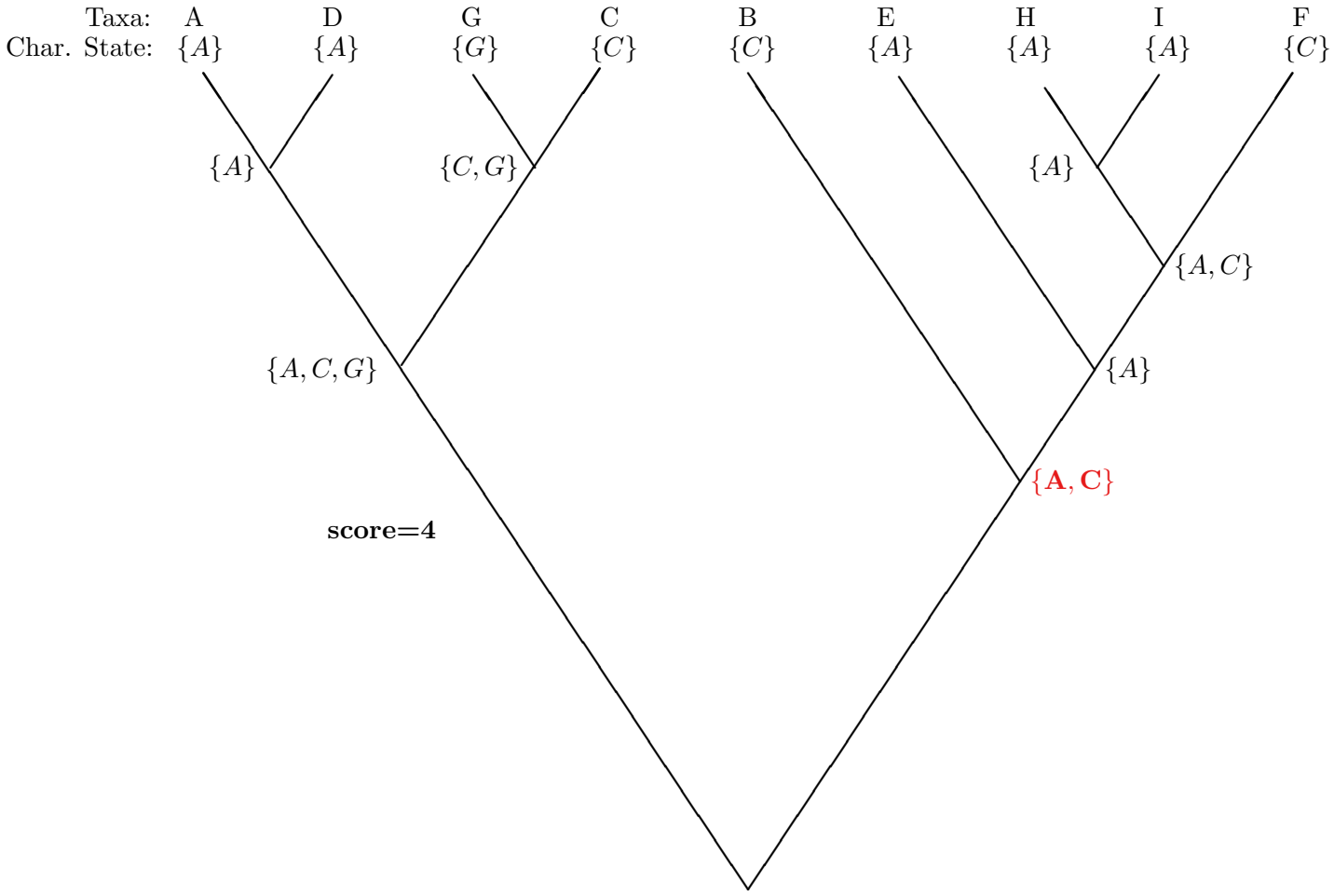


Figure 18: Fitch algorithm step 3 (seventh ancestral node – root) for that DNA sequence character shown in Table 5