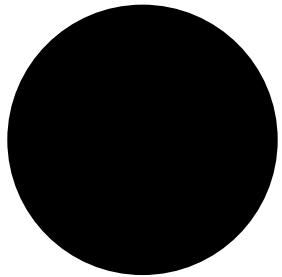
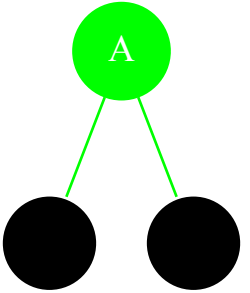
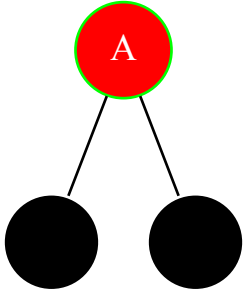
We begin with an empty tree.



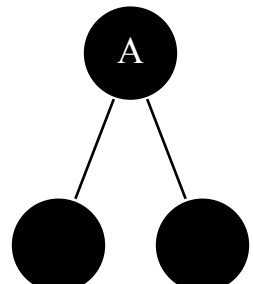
We are adding a new node with key 'A' to the tree.
Since there were no elements in the tree prior to this, the new element becomes the root.



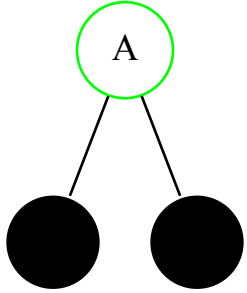
Now that we have positioned the new node ('A'), we need to ensure the correctness of the red-black tree.
Since we assumed the new node to be coloured red, we can only violate the 2. and 4. rules of the tree.
Now we set out to correct any discrepancies, by evaluating the colour of the newly added node's parent as well as its uncle, and acting accordingly.



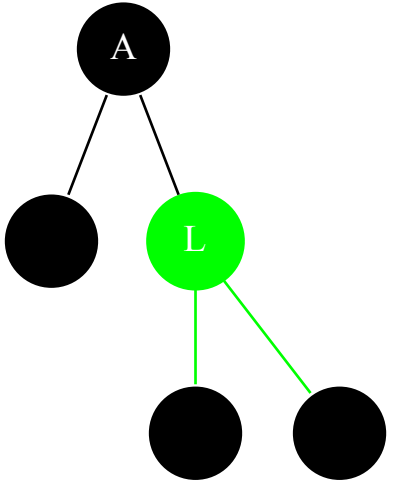
Since all of the required red-black properties are satisfied, we have arrived to a correctly formed red-black tree.



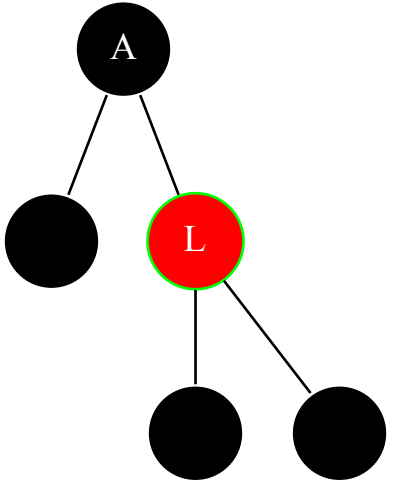
We are adding a new node with key 'L' to the tree.
The first step is to find a leaf (Nill), which the new node will replace.
We begin by examining the root (key 'A').



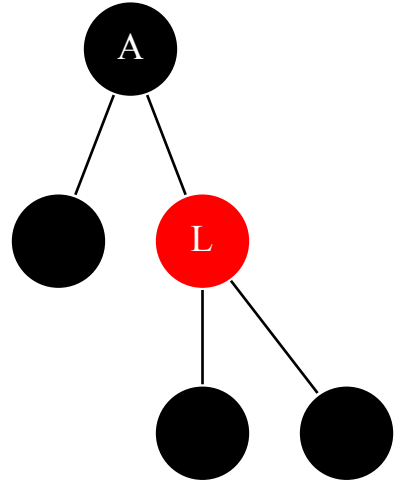
Since the key of the new node ('L') is greater than the one of its new parent ('A'), we add it to the right of the parent.



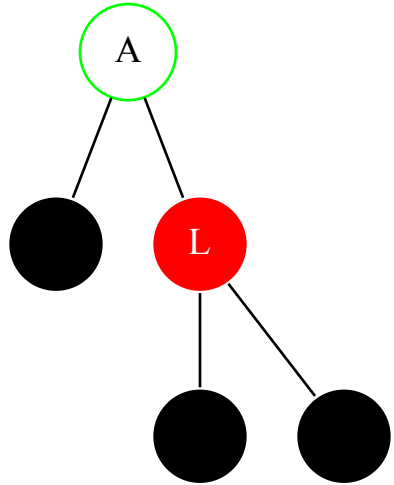
Now that we have positioned the new node ('L'), we need to ensure the correctness of the red-black tree.
Since we assumed the new node to be coloured red, we can only violate the 2. and 4. rules of the tree.
Now we set out to correct any discrepancies, by evaluating the colour of the newly added node's parent as well as its uncle, and acting accordingly.



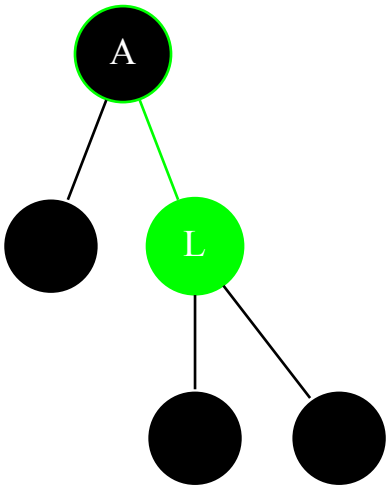
Since all of the required red-black properties are satisfied, we have arrived to a correctly formed red-black tree.



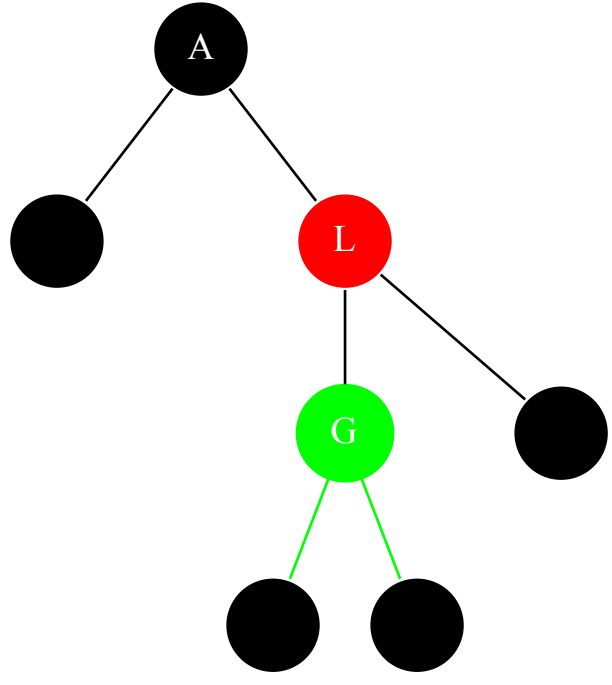
We are adding a new node with key 'G' to the tree.
The first step is to find a leaf (Nill), which the new node will replace.
We begin by examining the root (key 'A').



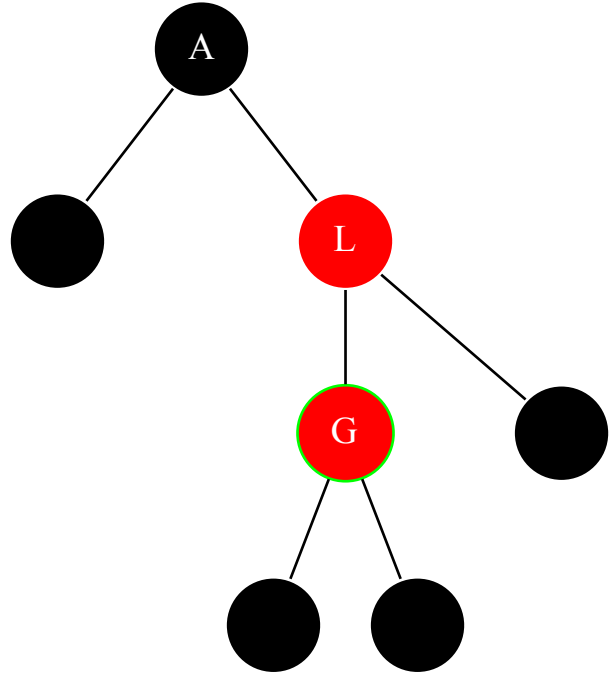
We are adding a new node with key 'G' to the tree.
By examining the node with key 'A' we have arrived at the node with key 'L'.



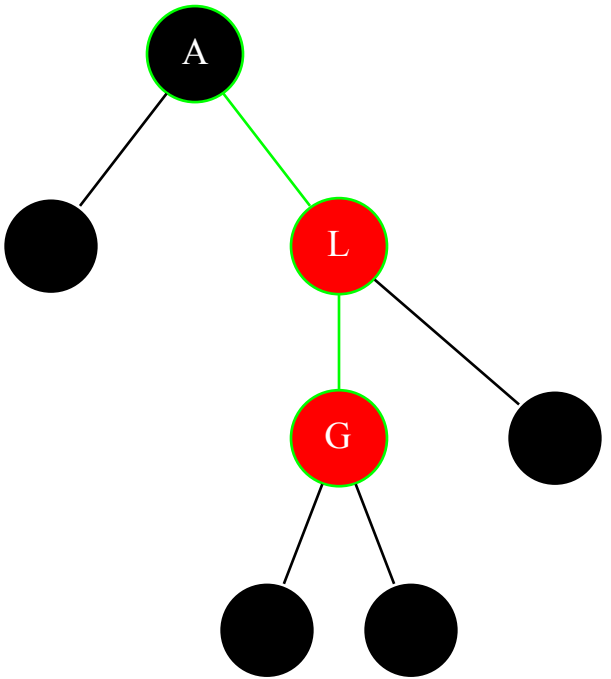
Since the key of the new node ('G') is smaller than the one of its new parent ('L'), we add it to the left of the parent.



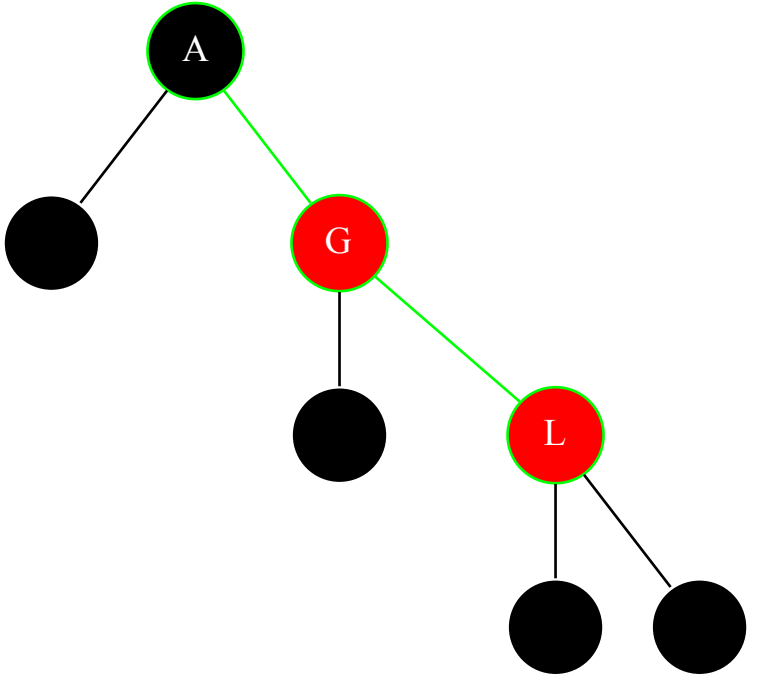
Now that we have positioned the new node ('G'), we need to ensure the correctness of the red-black tree.
Since we assumed the new node to be coloured red, we can only violate the 2. and 4. rules of the tree.
Now we set out to correct any discrepancies, by evaluating the colour of the newly added node's parent as well as its uncle, and acting accordingly.



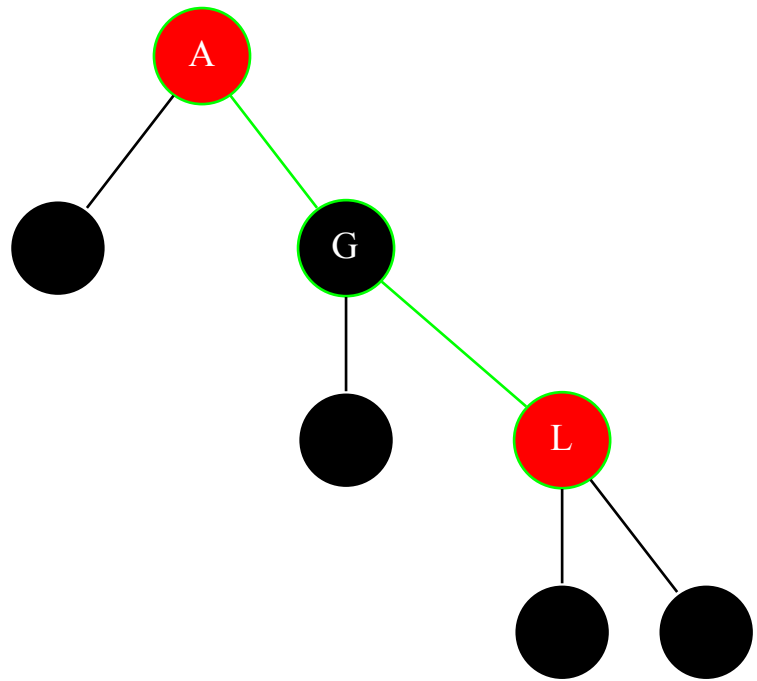
Since the uncle ('Nill') is black, and the parent ('L') is red, we cannot just perform a recolouring, but have to perform a rotation.
This rotation is needed because the currently examined node ('G'), its father ('L') and grandfather ('A') are not 'in-line'.



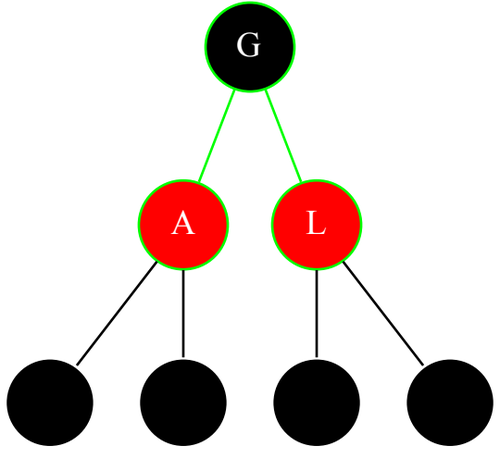
After we performed the before-mentioned rotation, the currently examined node ('L'), its parent ('G') and grandparent ('A') are now 'in-line'.



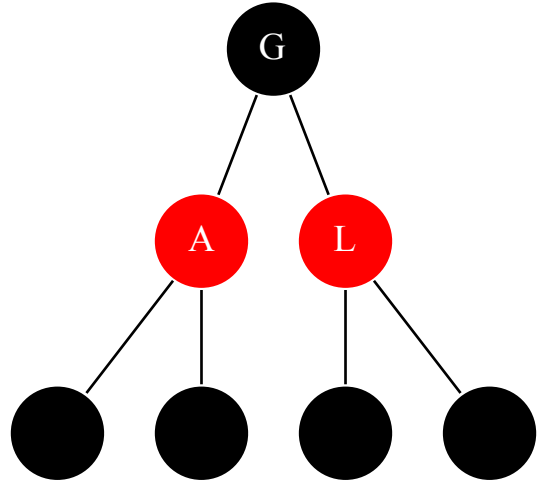
We have now made the currently examined node ('L'), its parent ('G') and grandparent ('A') align.
We now had to recolour them, so that when we rotate them in the next step, we will get a correct tree.



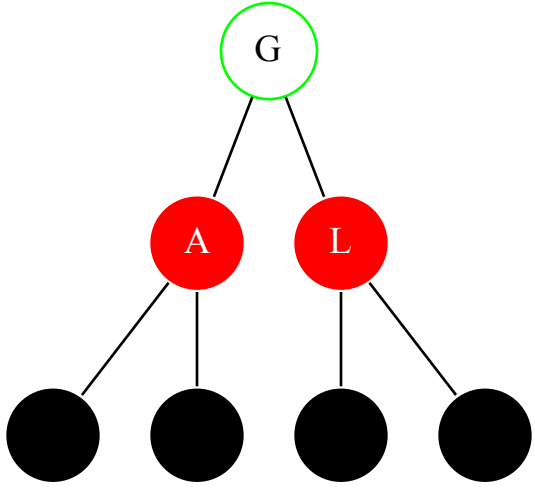
Since the currently examined node ('L'), its parent ('G') and grandparent ('A') are 'in-line' and properly coloured now, just one more rotation gives a correct red-black tree.



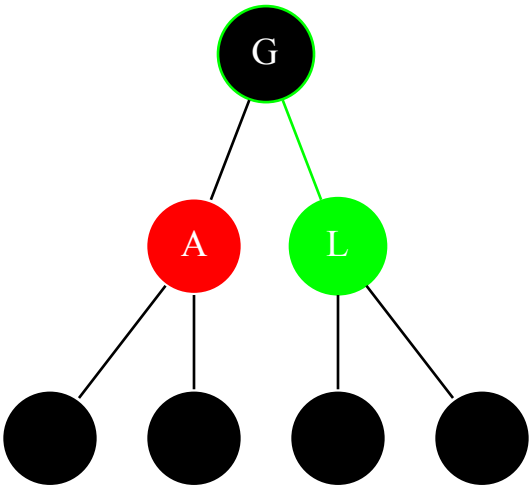
Since all of the required red-black properties are satisfied, we have arrived to a correctly formed red-black tree.



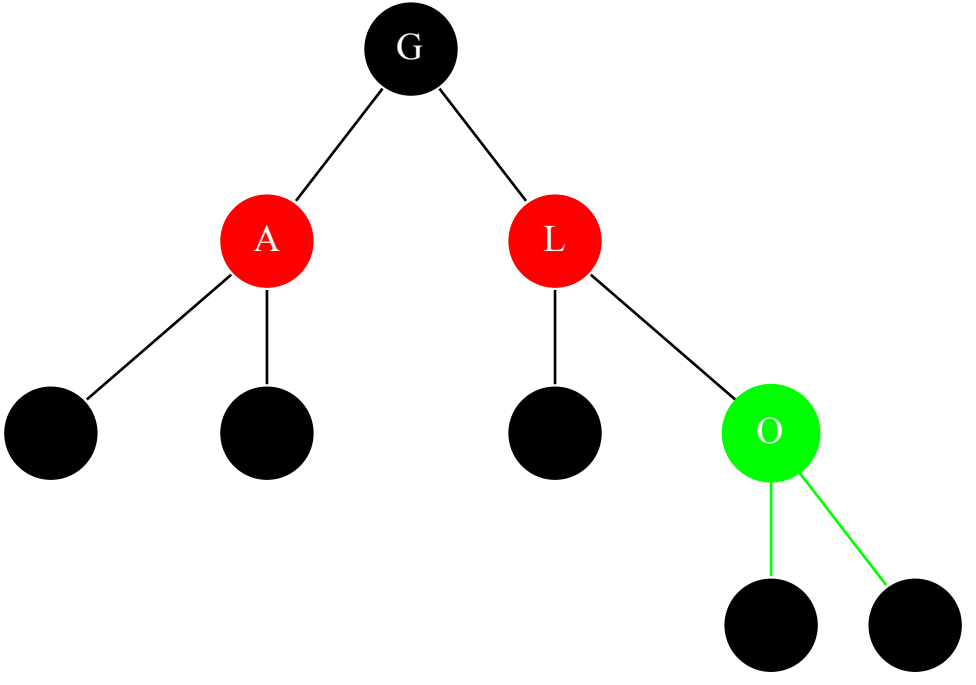
We are adding a new node with key 'O' to the tree.
The first step is to find a leaf (Nill), which the new node will replace.
We begin by examining the root (key 'G').



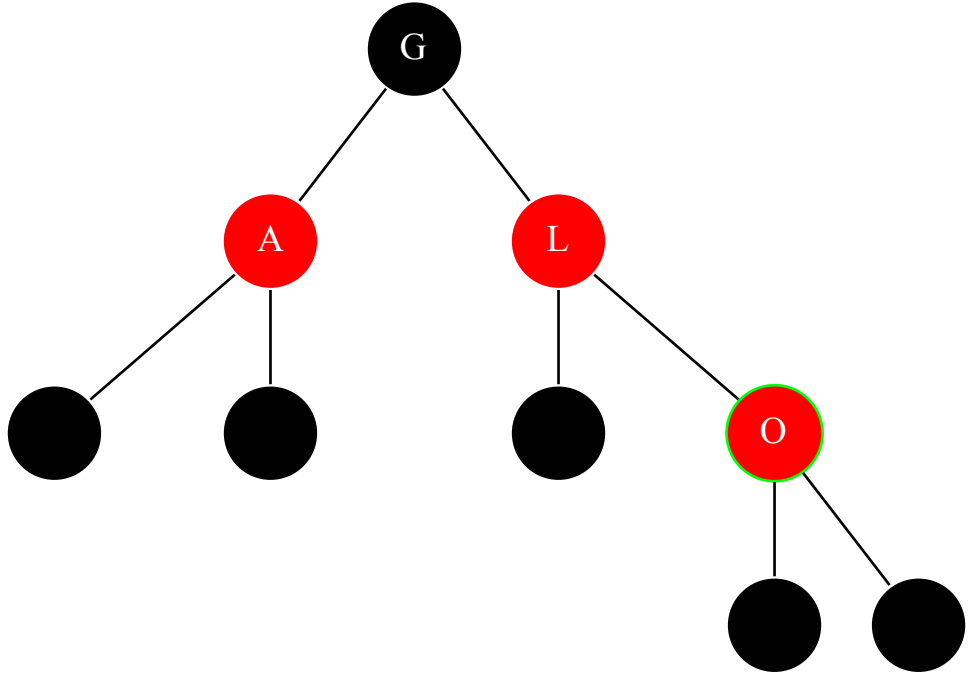
We are adding a new node with key 'O' to the tree.
By examining the node with key 'G' we have arrived at the node with key 'L'.



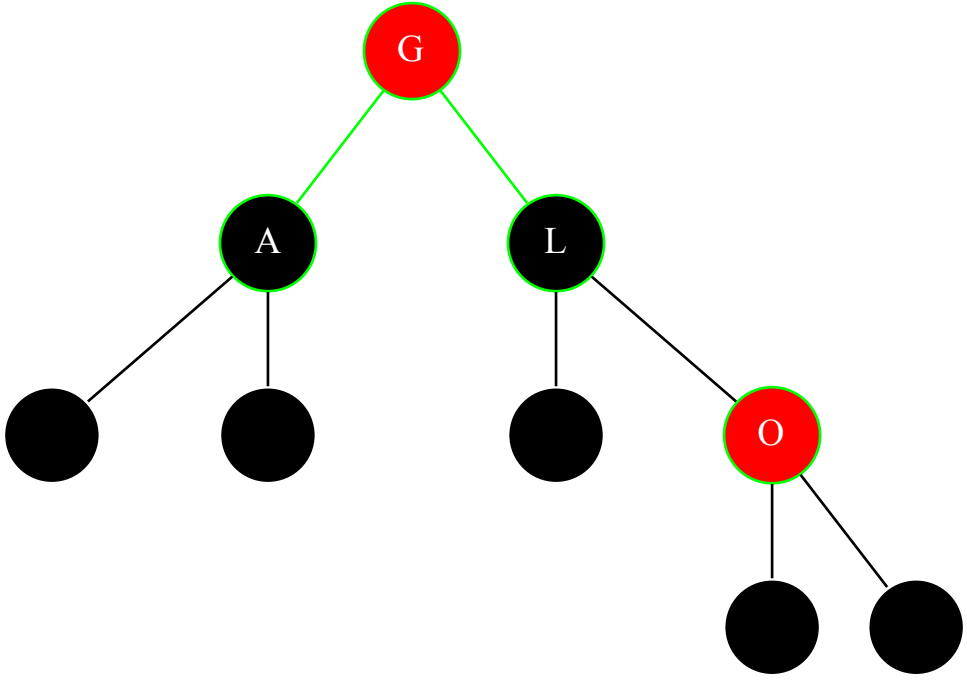
Since the key of the new node ('O') is greater than the one of its new parent ('L'), we add it to the right of the parent.



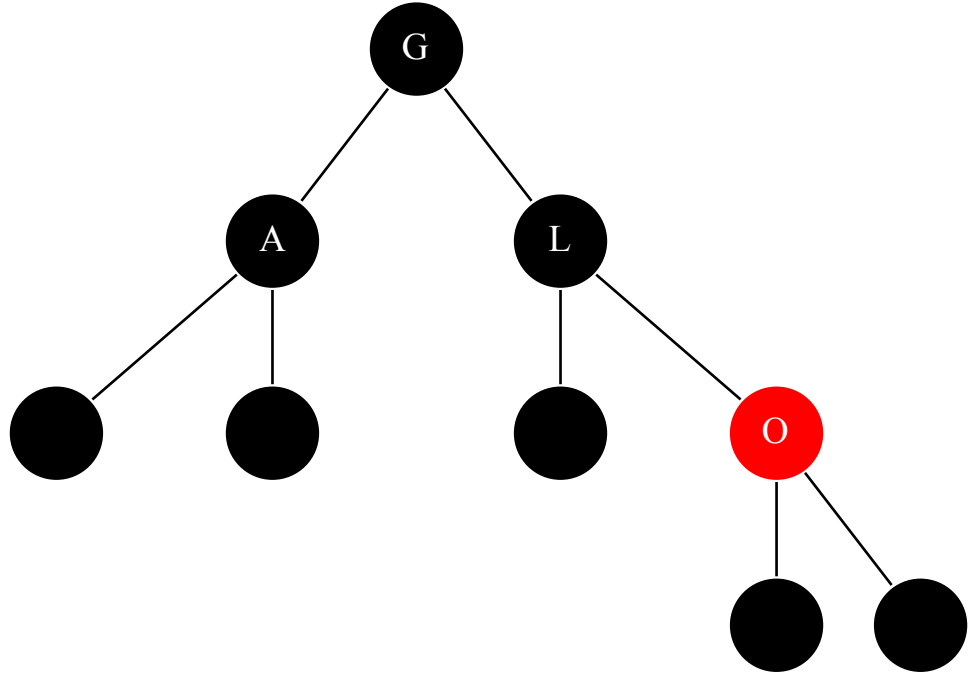
Now that we have positioned the new node ('O'), we need to ensure the correctness of the red-black tree.
Since we assumed the new node to be coloured red, we can only violate the 2. and 4. rules of the tree.
Now we set out to correct any discrepancies, by evaluating the colour of the newly added node's parent as well as its uncle, and acting accordingly.



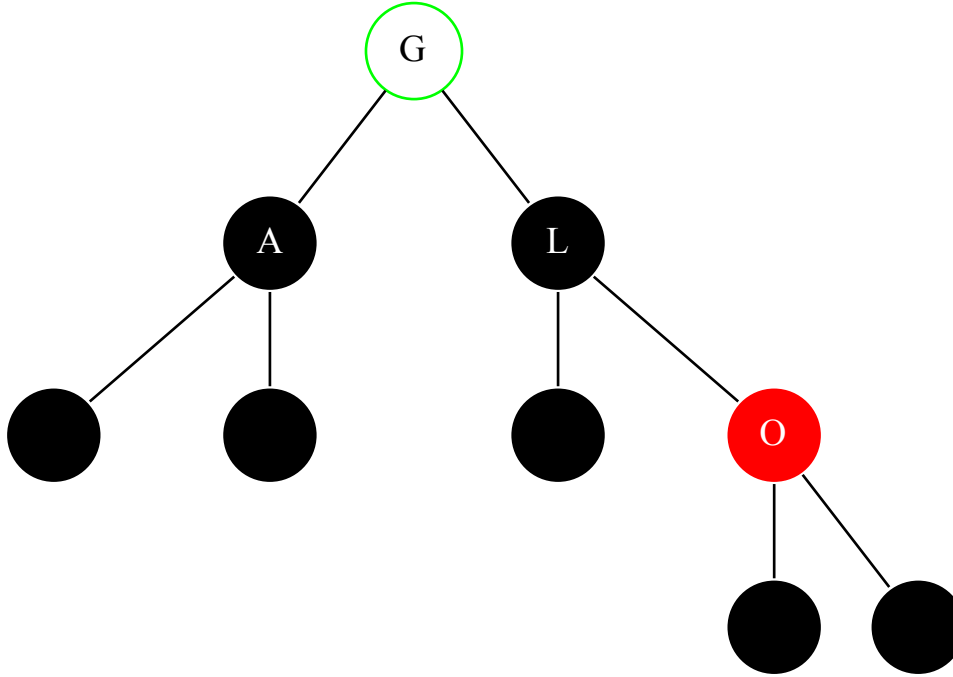
Since the uncle ('A') was red and we know that the parent ('L') was also red, we just had to perform a colour rebalancing.
This means that we coloured the currently examined nodes parent ('L') and uncle black (A), and coloured its grandfather ('G') red.
By doing this we correct the fourth rule of the red-black trees.



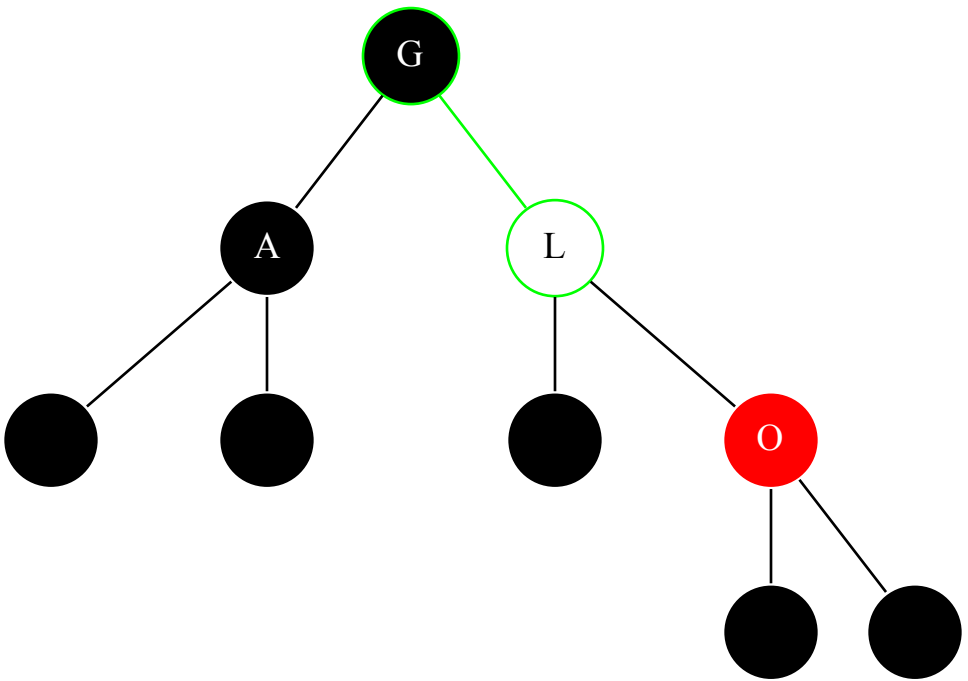
Since all of the required red-black properties are satisfied, we have arrived to a correctly formed red-black tree.



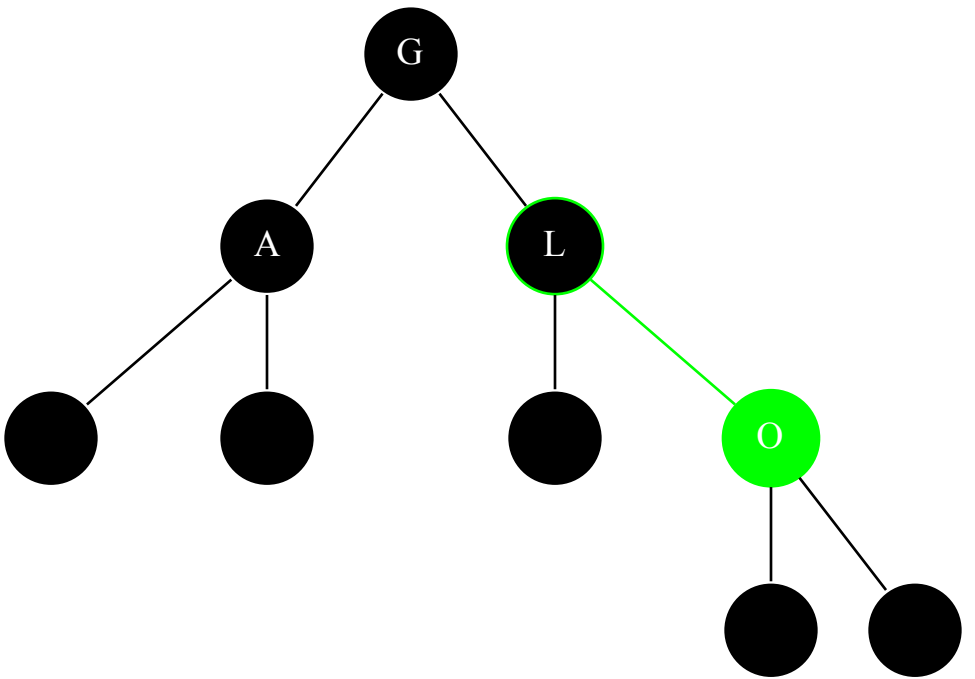
We are adding a new node with key 'R' to the tree.
The first step is to find a leaf (Nill), which the new node will replace.
We begin by examining the root (key 'G').



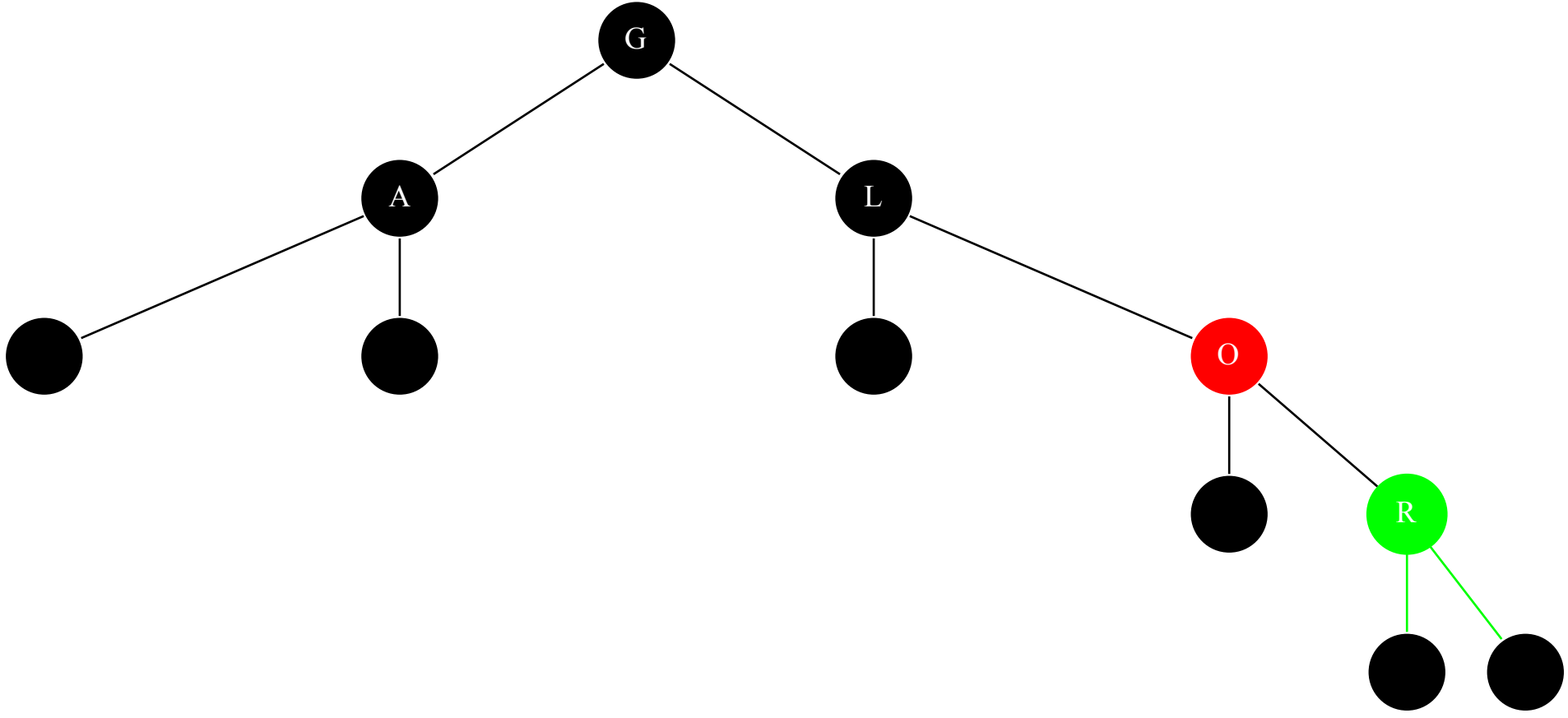
We are adding a new node with key 'R' to the tree.
By examining the node with key 'G' we have arrived at the node with key 'L'.



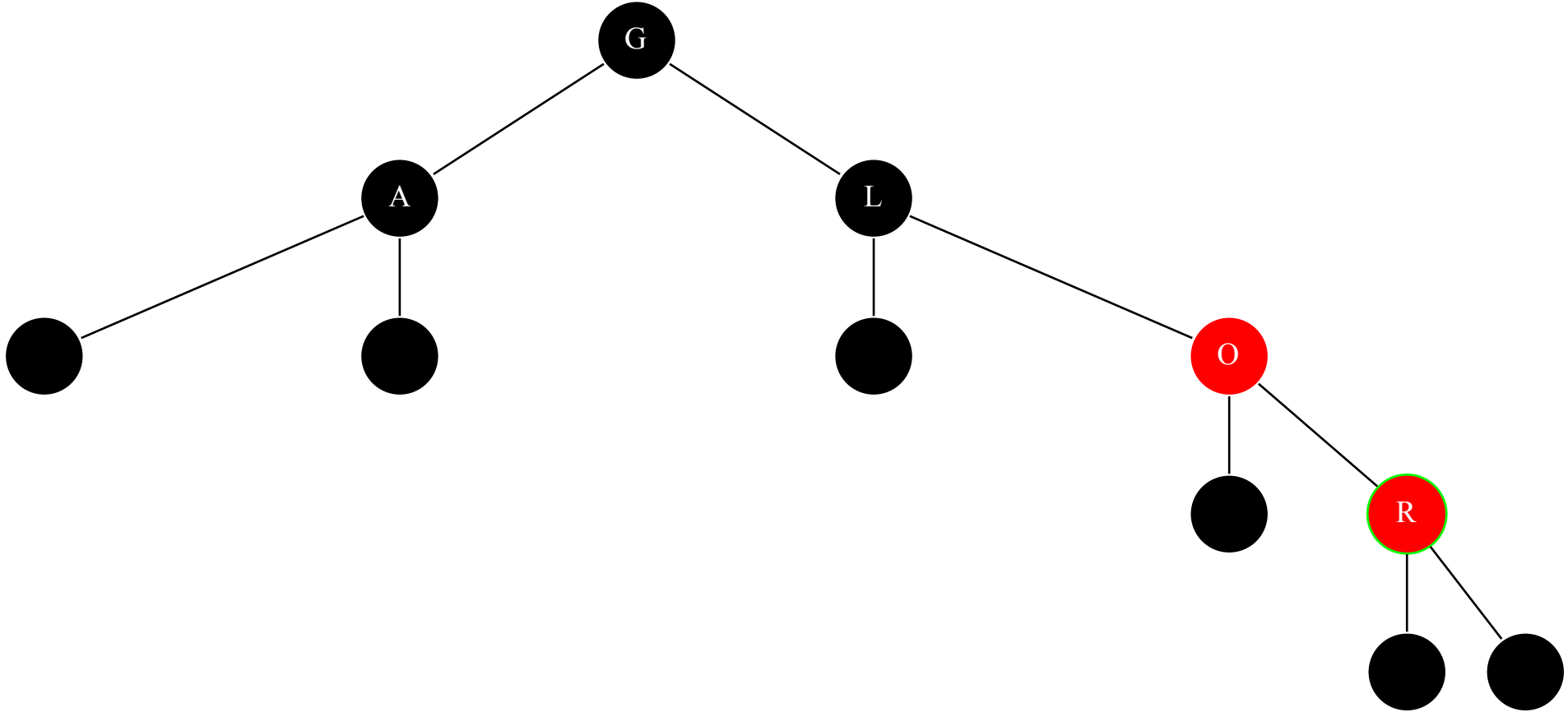
We are adding a new node with key 'R' to the tree.
By examining the node with key 'L' we have arrived at the node with key 'O'.



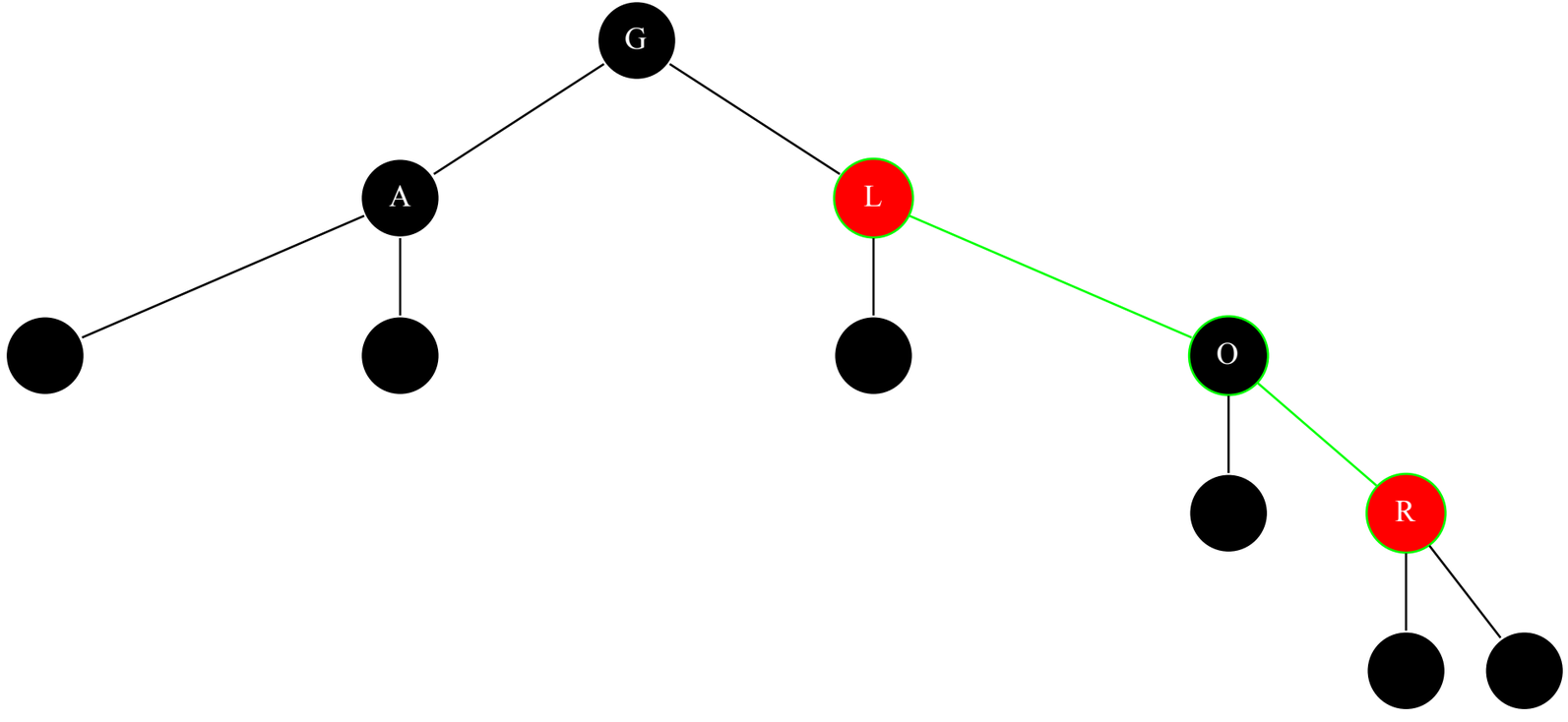
Since the key of the new node ('R') is greater than the one of its new parent ('O'), we add it to the right of the parent.



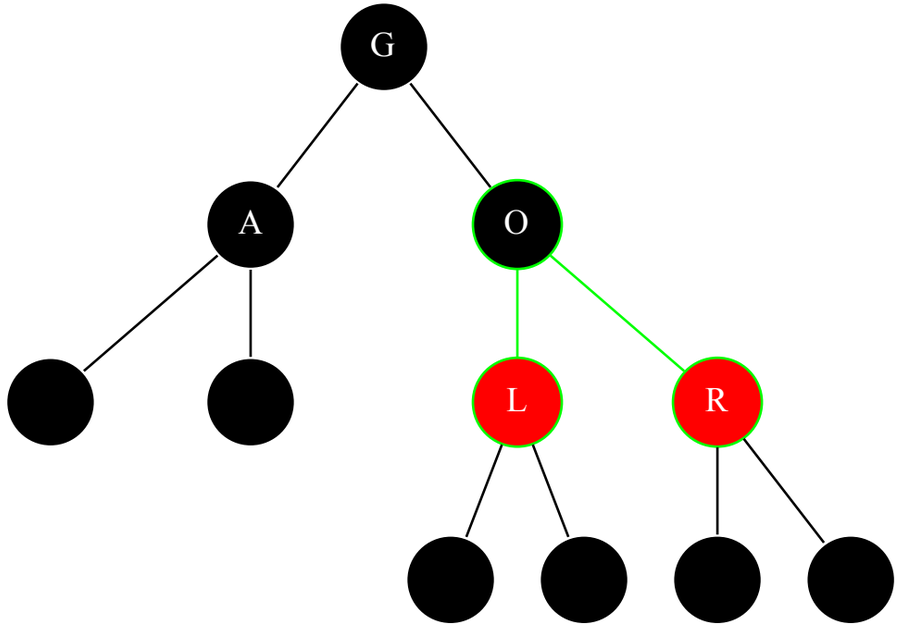
Now that we have positioned the new node ('R'), we need to ensure the correctness of the red-black tree.
Since we assumed the new node to be coloured red, we can only violate the 2. and 4. rules of the tree.
Now we set out to correct any discrepancies, by evaluating the colour of the newly added node's parent as well as its uncle, and acting accordingly.



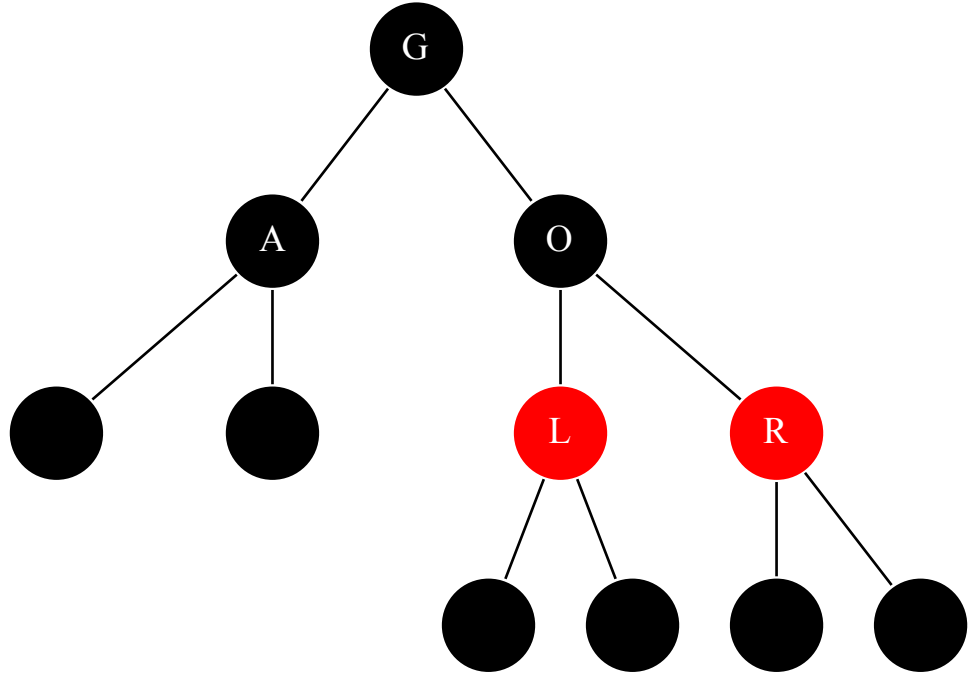
Since the currently examined node ('R'), its parent ('O') and grandparent ('L') are 'in-line', we now had to recolour them, so that when we rotate them in the next step, we will get a correct tree.



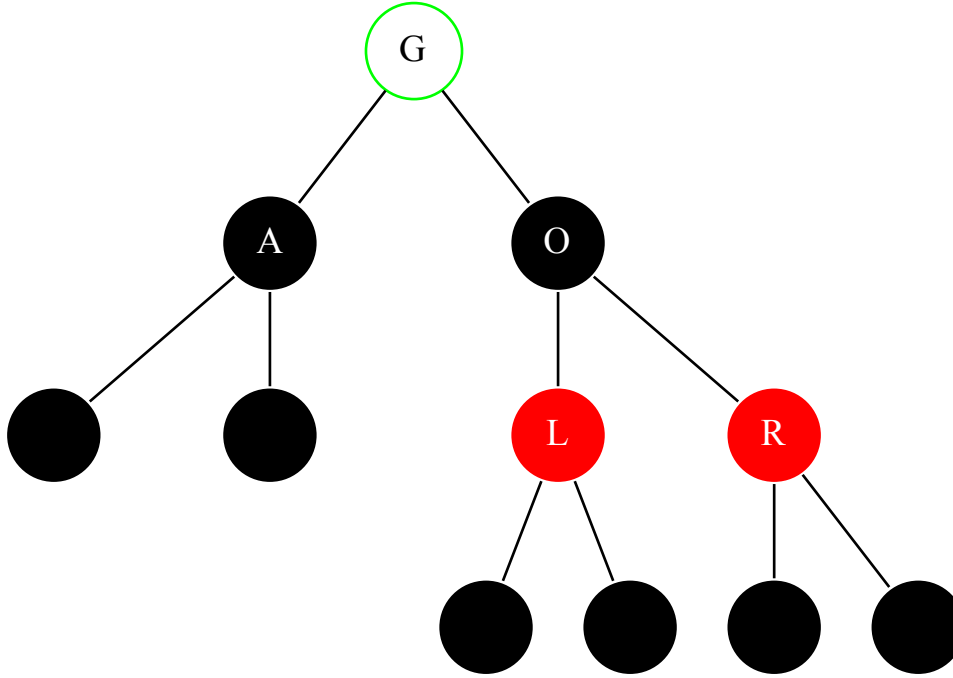
Since the currently examined node ('R'), its parent ('O') and grandparent ('L') are 'in-line' and properly coloured now, just one more rotation gives a correct red-black tree.



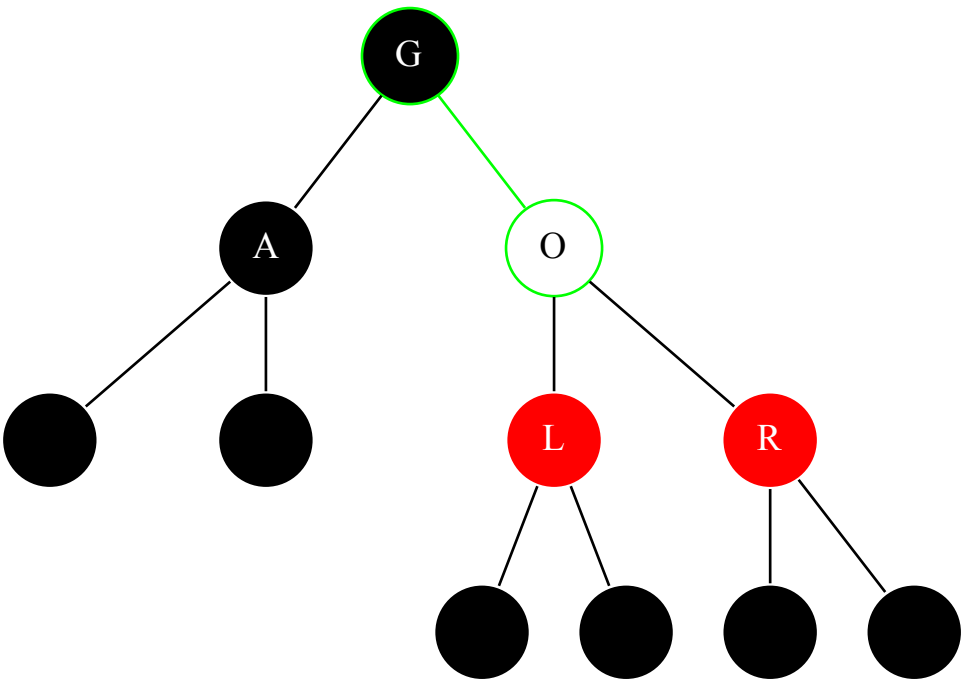
Since all of the required red-black properties are satisfied, we have arrived to a correctly formed red-black tree.



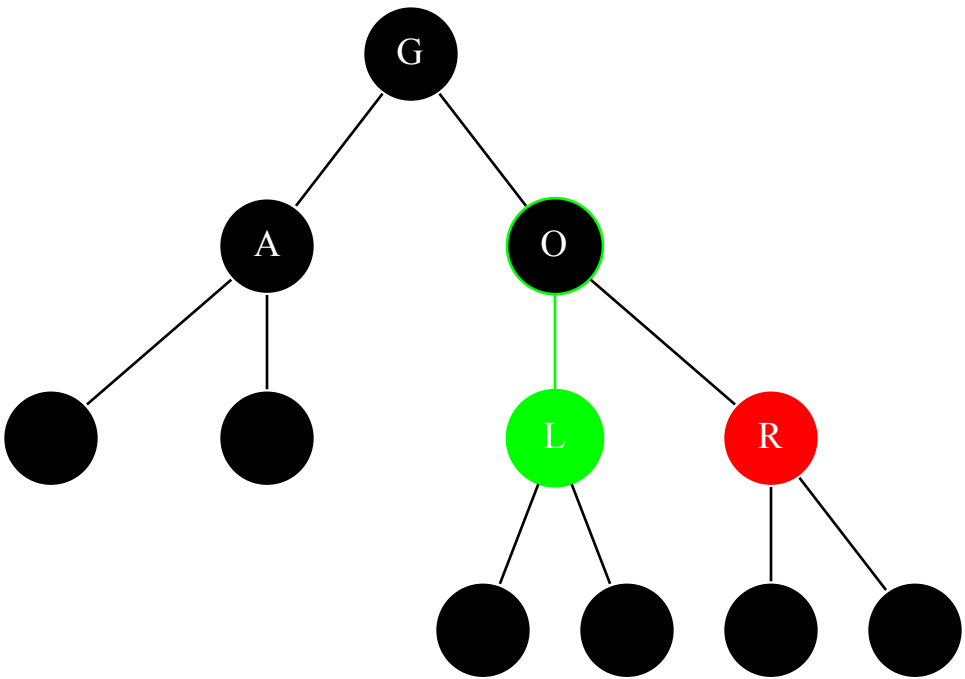
We are adding a new node with key 'I' to the tree.
The first step is to find a leaf (Nill), which the new node will replace.
We begin by examining the root (key 'G').



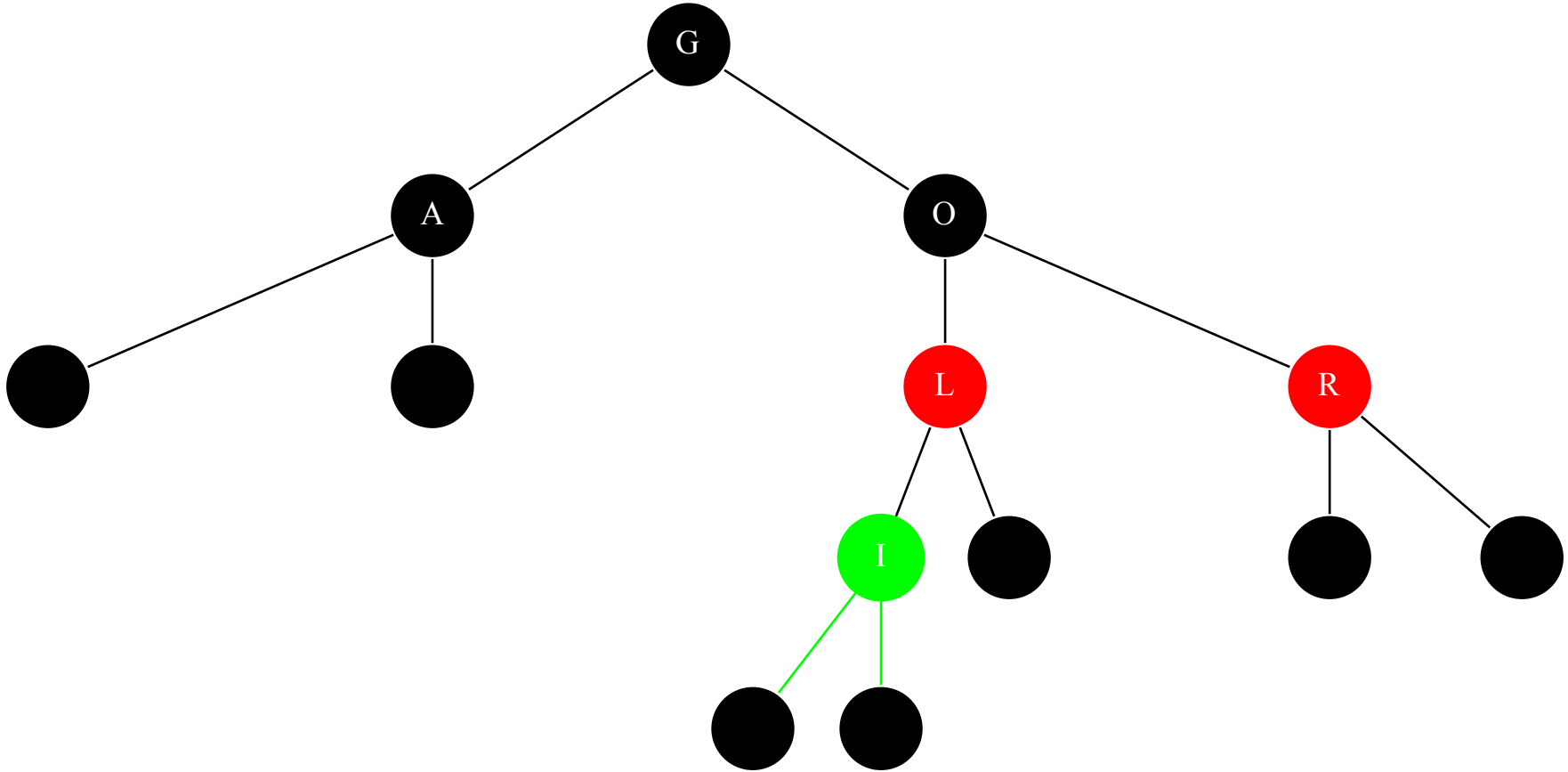
We are adding a new node with key 'I' to the tree.
By examining the node with key 'G' we have arrived at the node with key 'O'.



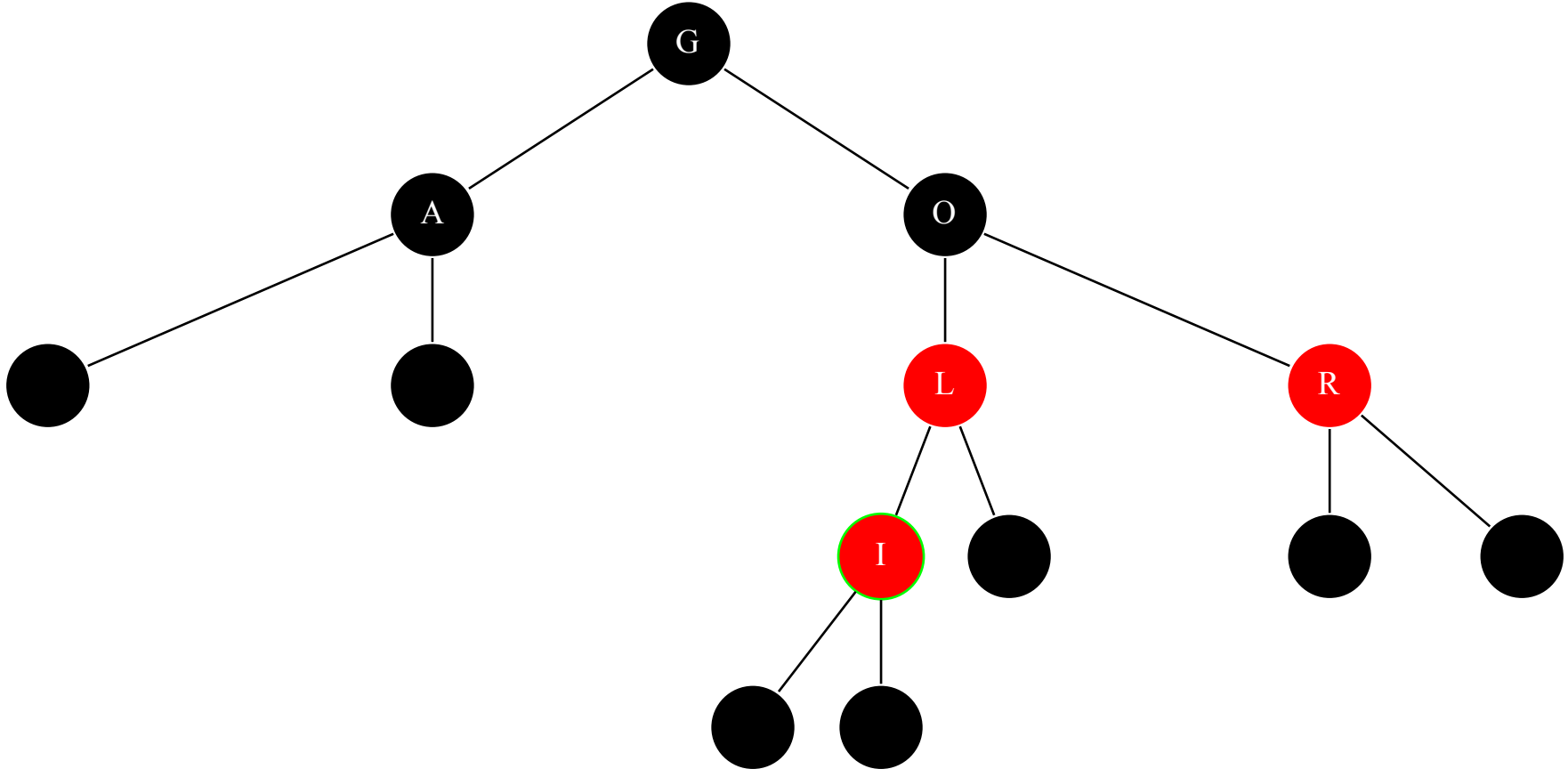
We are adding a new node with key 'I' to the tree.
By examining the node with key 'O' we have arrived at the node with key 'L'.



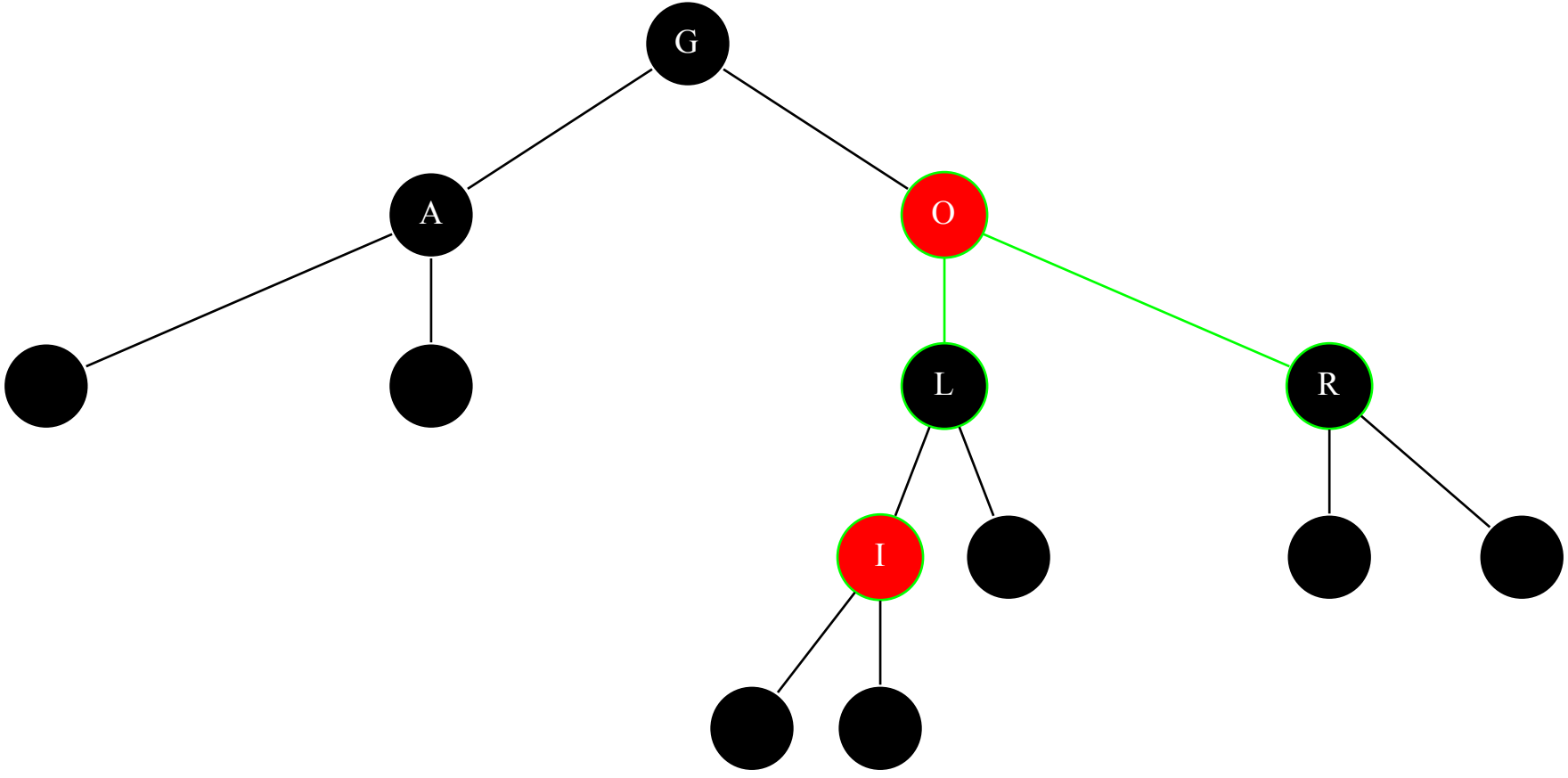
Since the key of the new node ('I') is smaller than the one of its new parent ('L'), we add it to the left of the parent.



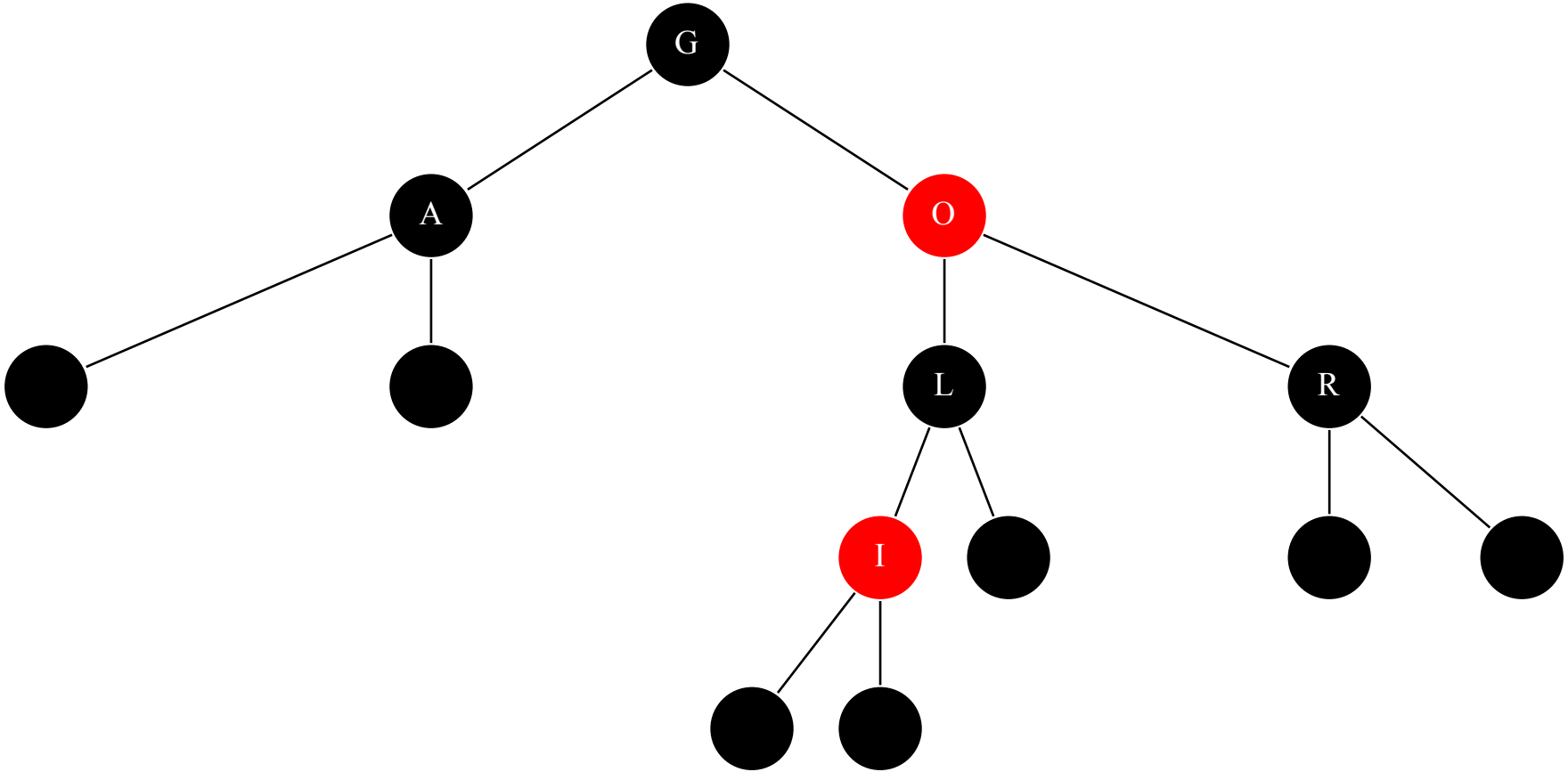
Now that we have positioned the new node ('I'), we need to ensure the correctness of the red-black tree.
Since we assumed the new node to be coloured red, we can only violate the 2. and 4. rules of the tree.
Now we set out to correct any discrepancies, by evaluating the colour of the newly added node's parent as well as its uncle, and acting accordingly.



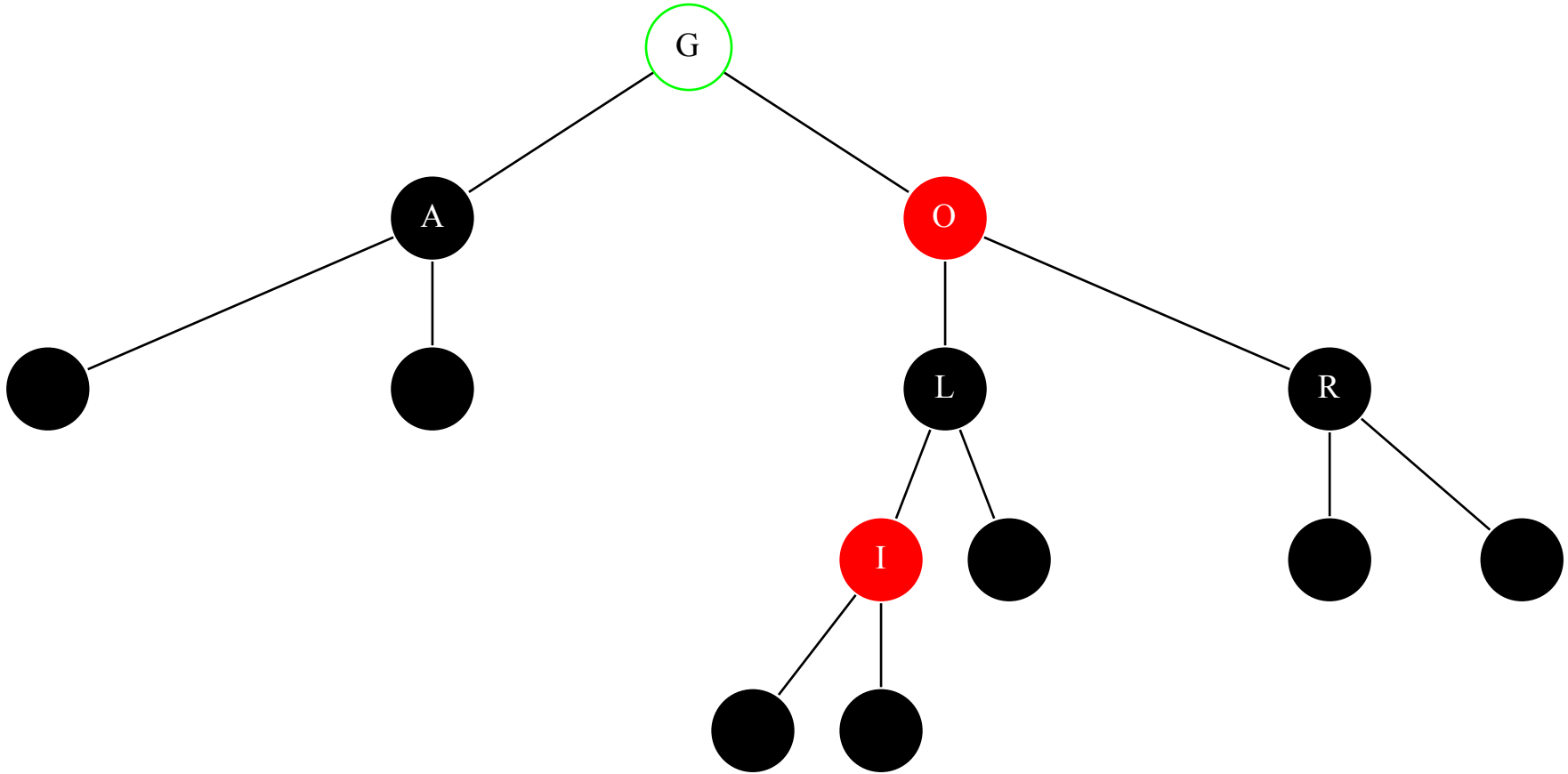
Since the uncle ('R') was red and we know that the parent ('L') was also red, we just had to perform a colour rebalancing.
This means that we coloured the currently examined nodes parent ('L') and uncle black (R), and coloured its grandfather ('O') red.
By doing this we correct the fourth rule of the red-black trees.



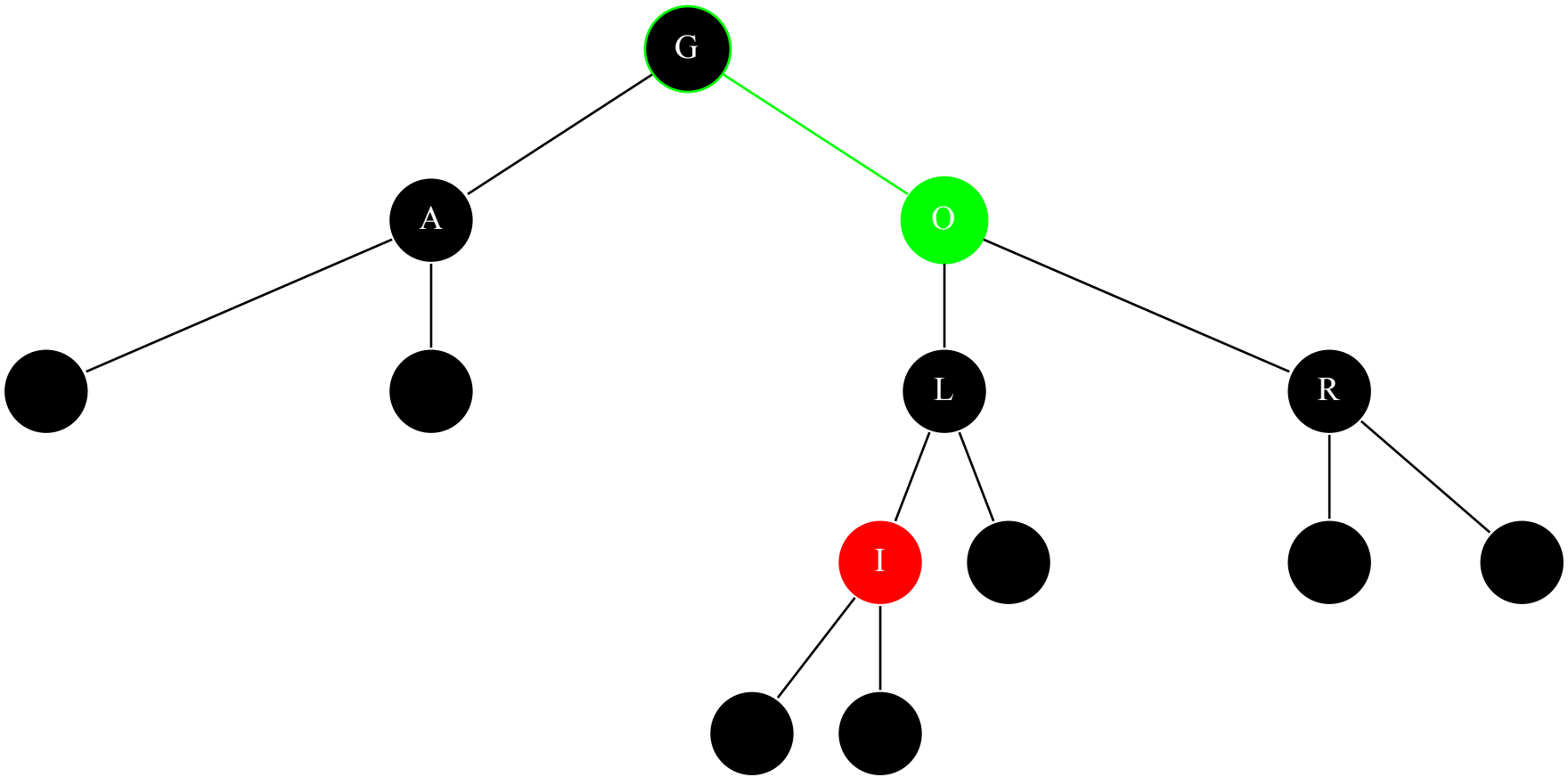
Since all of the required red-black properties are satisfied, we have arrived to a correctly formed red-black tree.



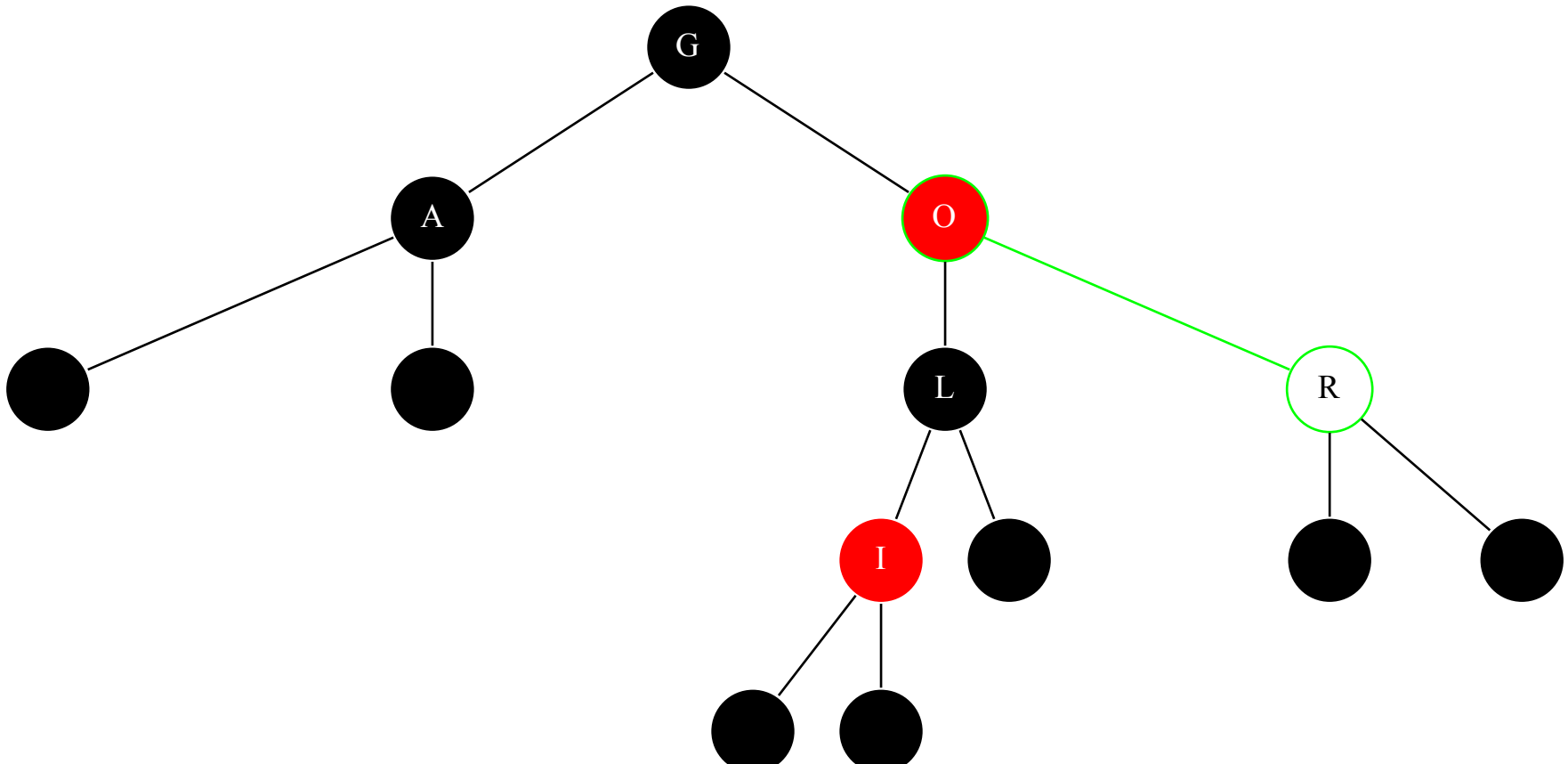
We are adding a new node with key 'T' to the tree.
The first step is to find a leaf (Nill), which the new node will replace.
We begin by examining the root (key 'G').



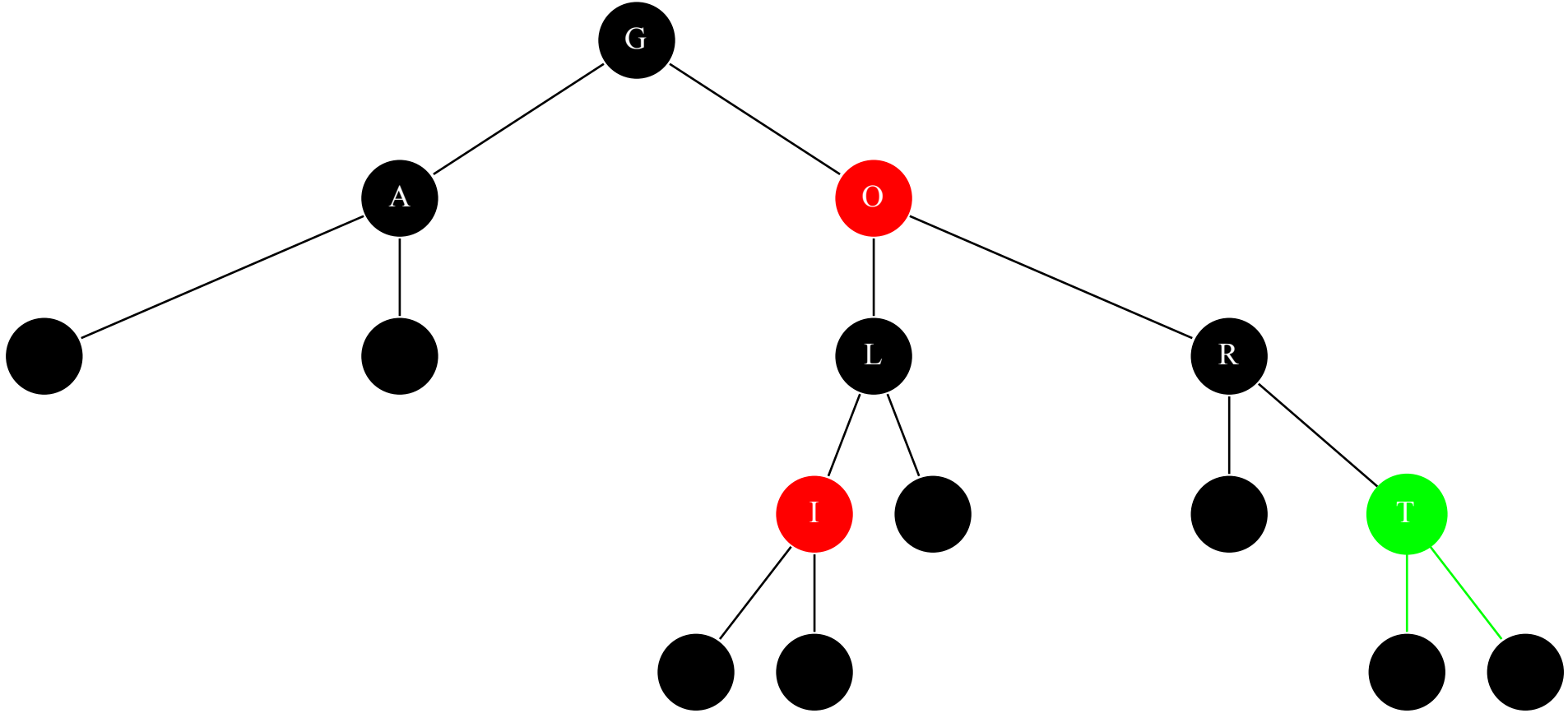
We are adding a new node with key 'T' to the tree.
By examining the node with key 'G' we have arrived at the node with key 'O'.



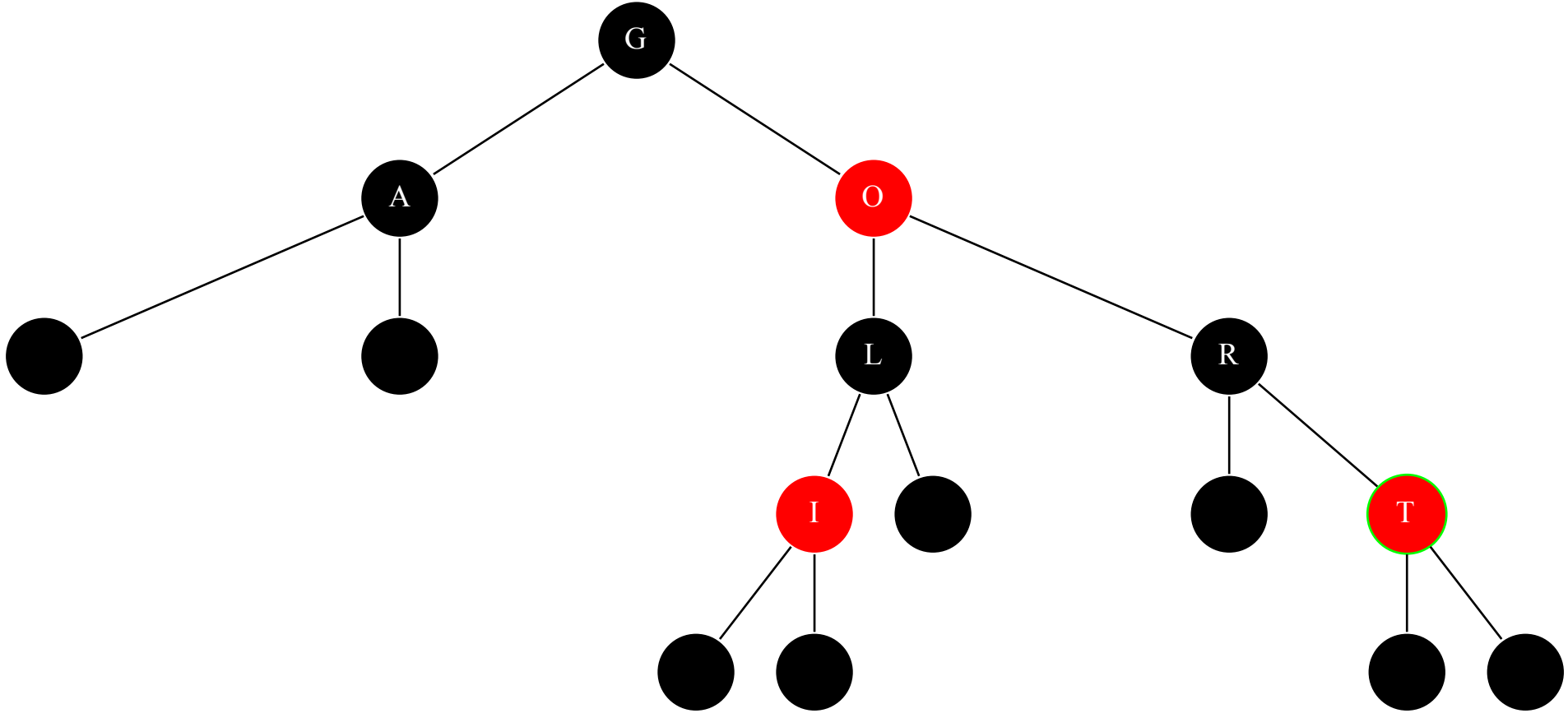
We are adding a new node with key 'T' to the tree.
By examining the node with key 'O' we have arrived at the node with key 'R'.



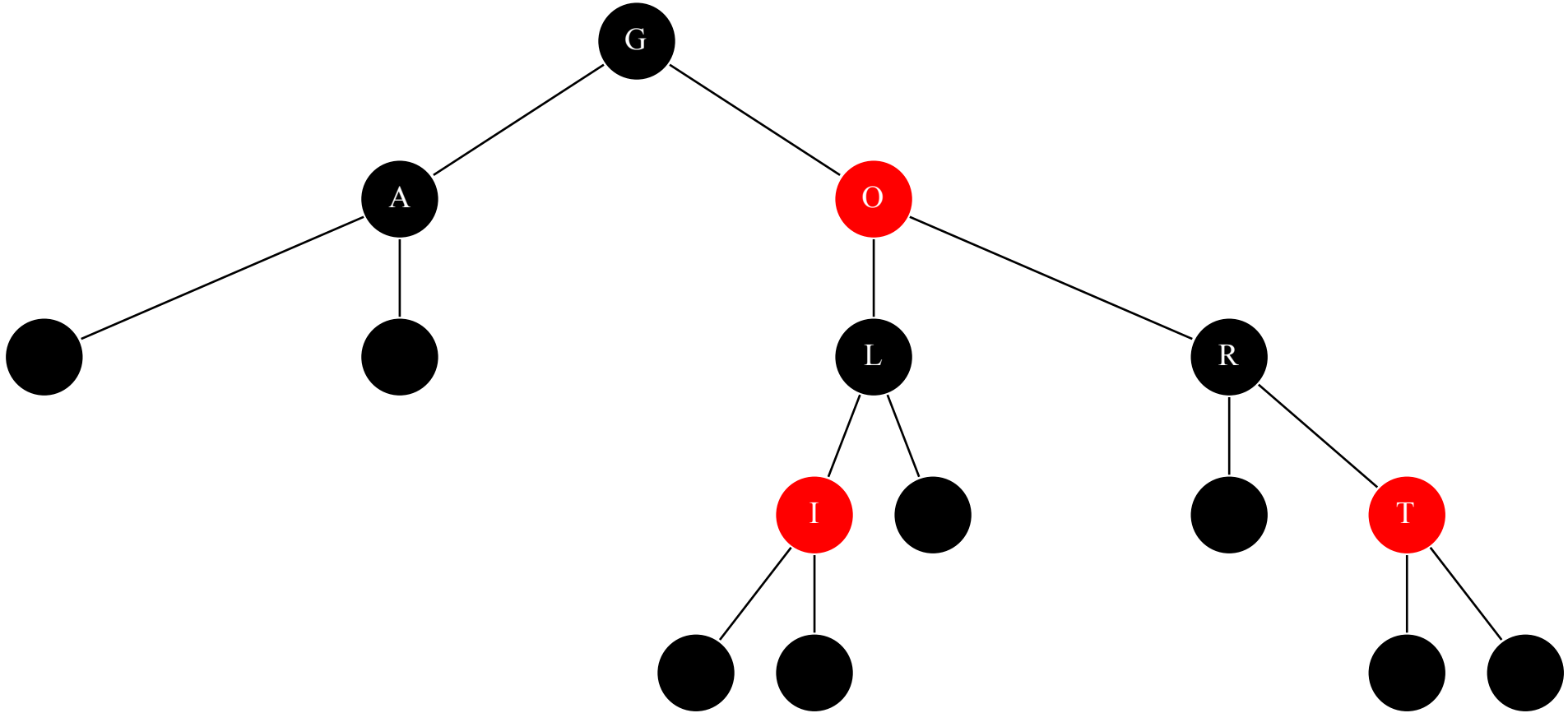
Since the key of the new node ('T') is greater than the one of its new parent ('R'), we add it to the right of the parent.



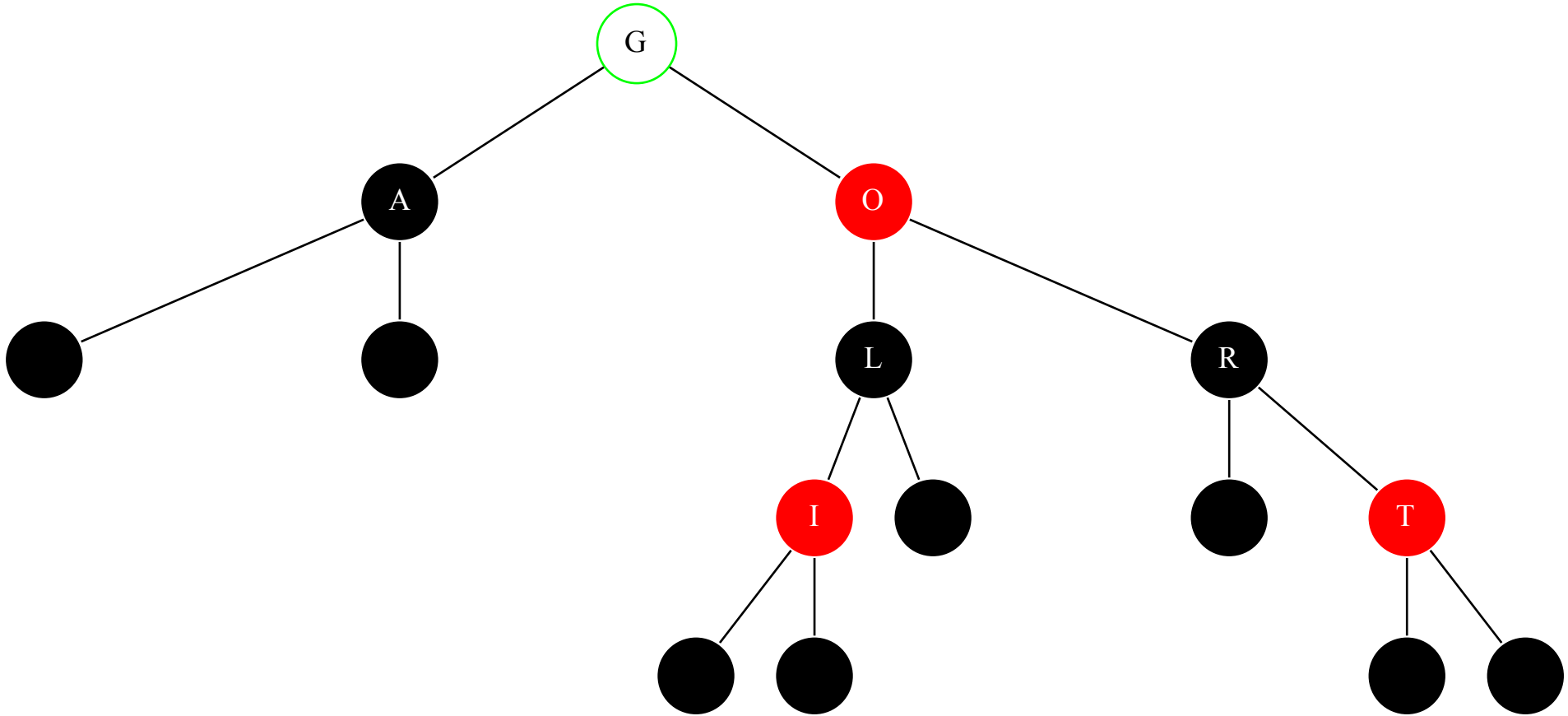
Now that we have positioned the new node ('T'), we need to ensure the correctness of the red-black tree.
Since we assumed the new node to be coloured red, we can only violate the 2. and 4. rules of the tree.
Now we set out to correct any discrepancies, by evaluating the colour of the newly added node's parent as well as its uncle, and acting accordingly.



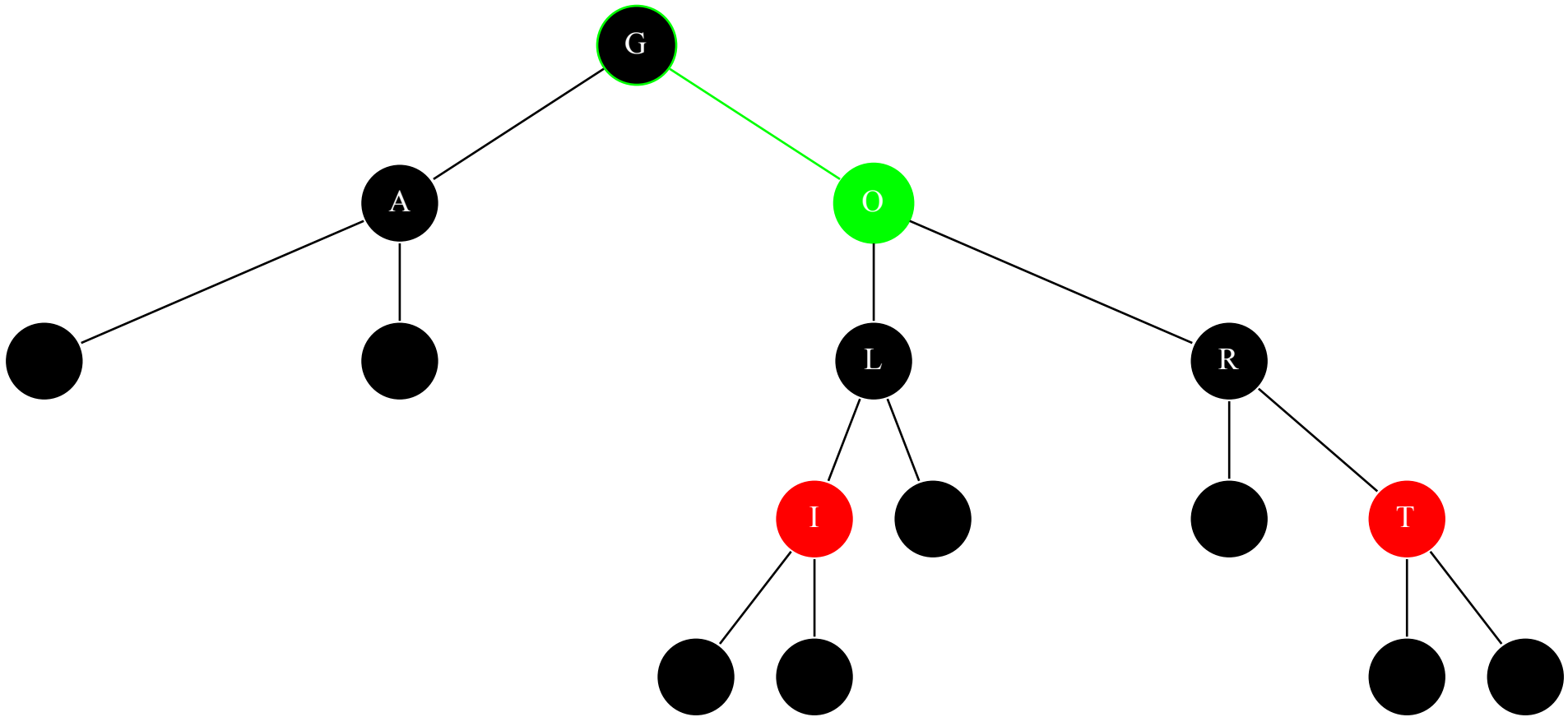
Since all of the required red-black properties are satisfied, we have arrived to a correctly formed red-black tree.



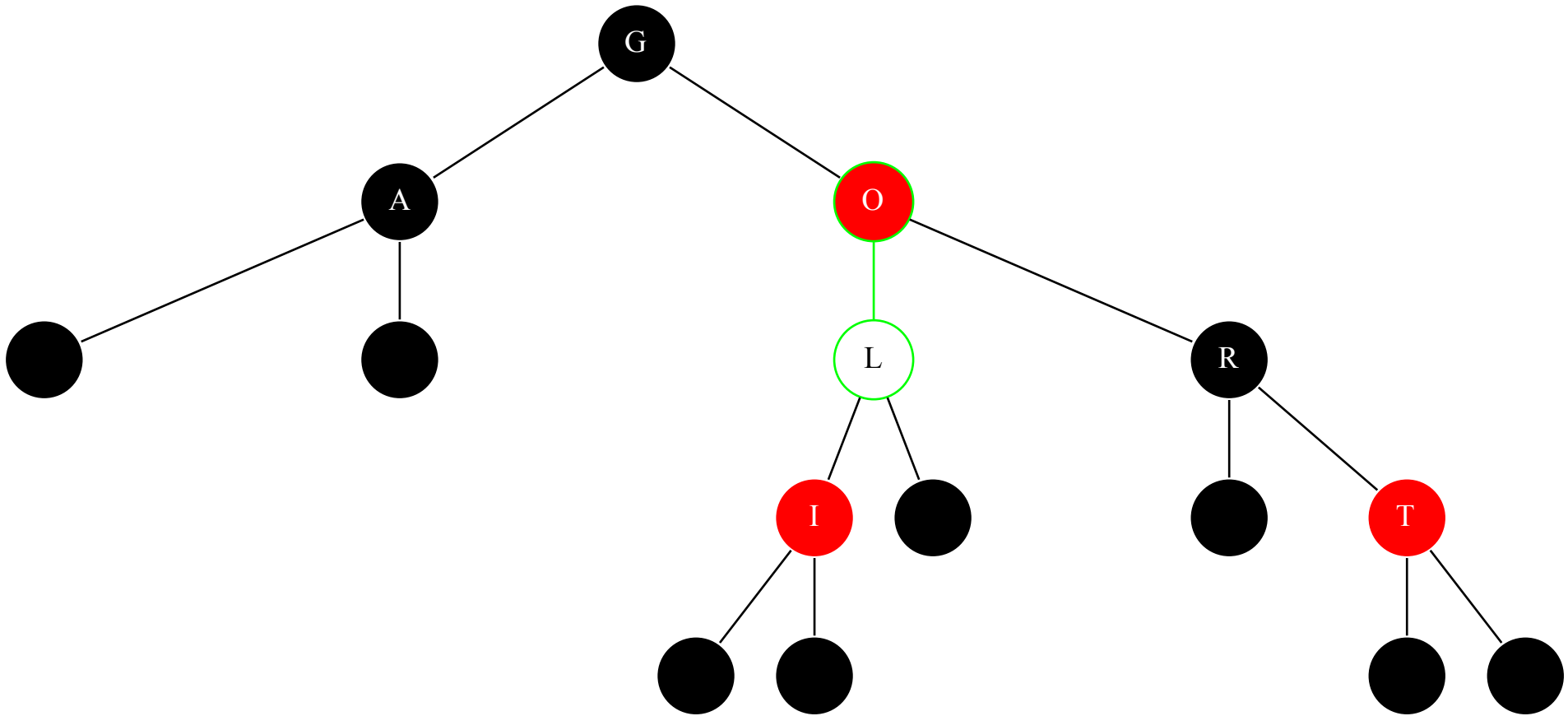
We are adding a new node with key 'H' to the tree.
The first step is to find a leaf (Nill), which the new node will replace.
We begin by examining the root (key 'G').



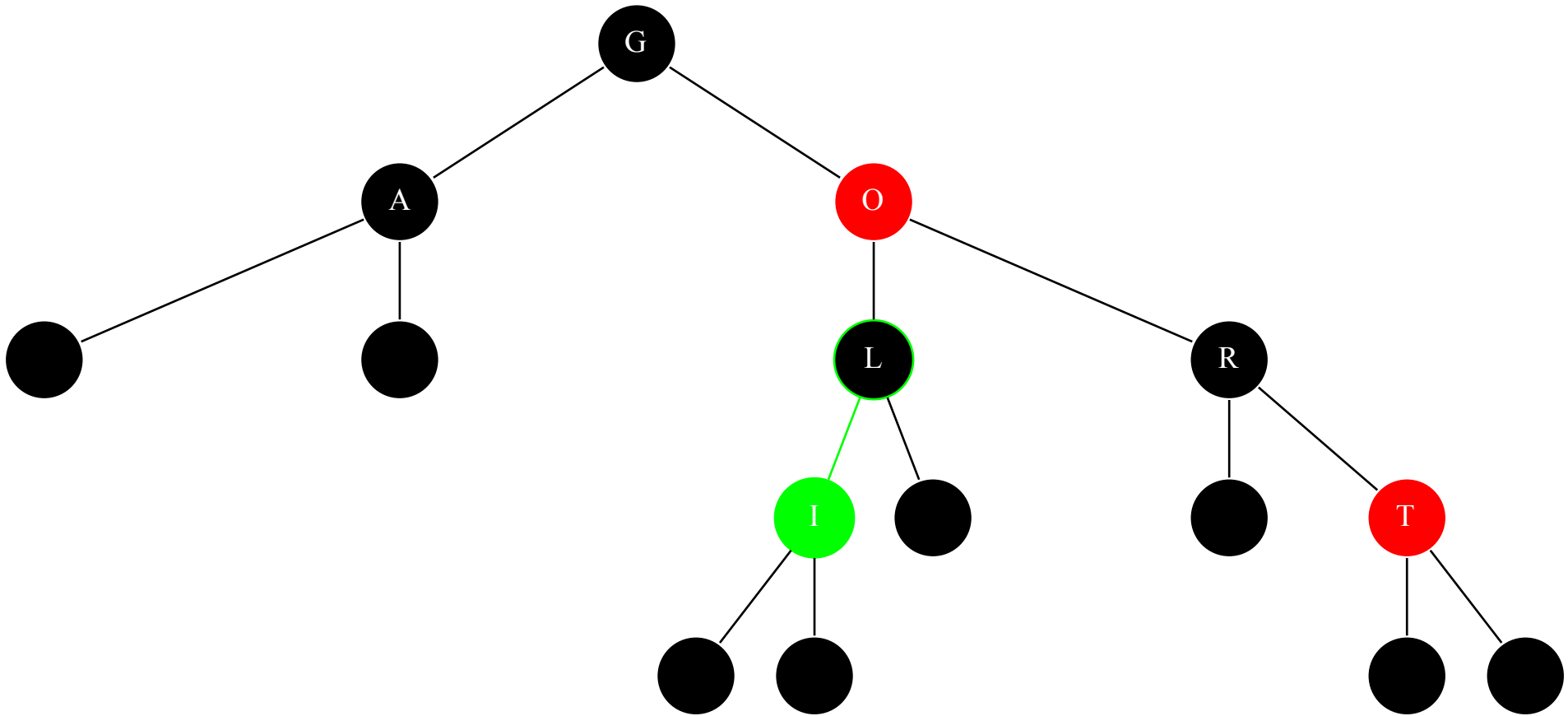
We are adding a new node with key 'H' to the tree.
By examining the node with key 'G' we have arrived at the node with key 'O'.



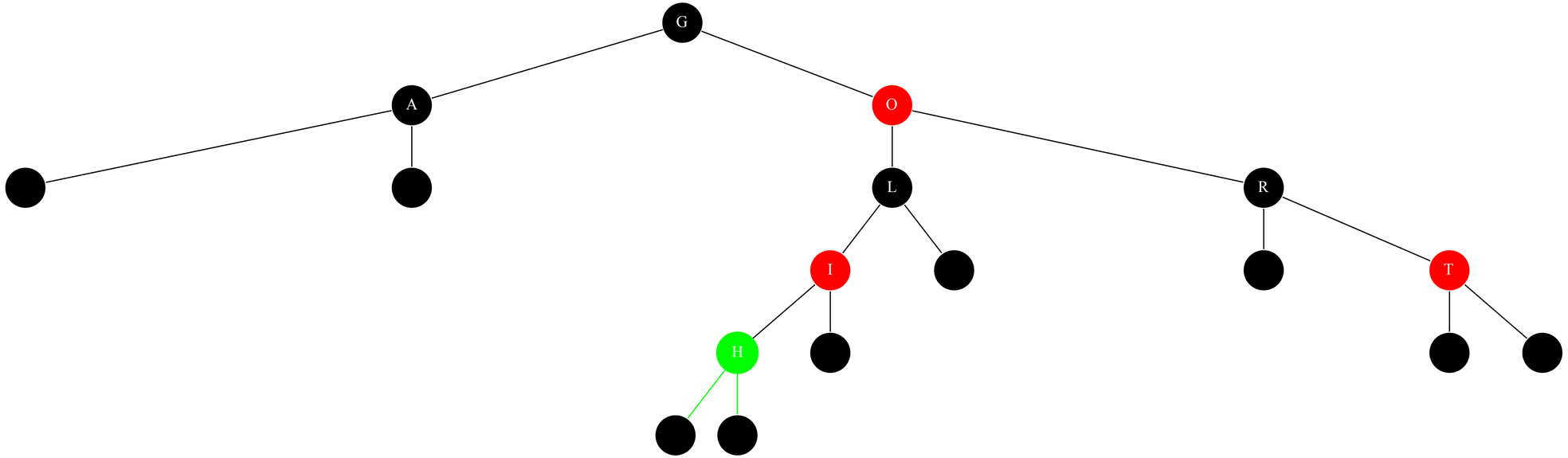
We are adding a new node with key 'H' to the tree.
By examining the node with key 'O' we have arrived at the node with key 'L'.



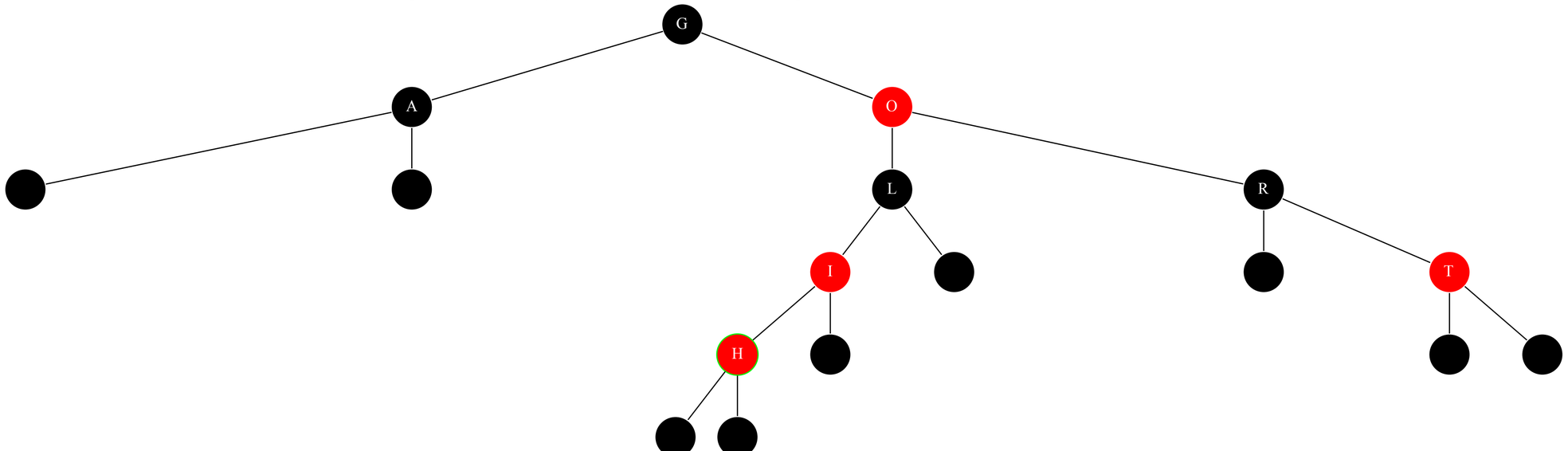
We are adding a new node with key 'H' to the tree.
By examining the node with key 'L' we have arrived at the node with key 'I'.



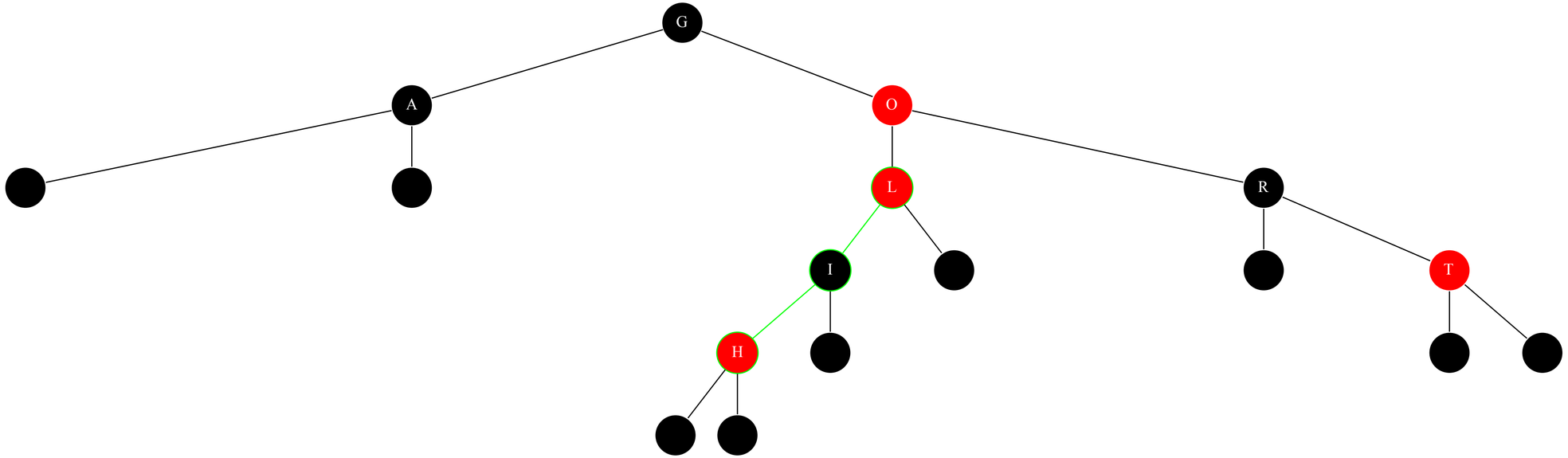
Since the key of the new node ('H') is smaller than the one of its new parent ('I'), we add it to the left of the parent.



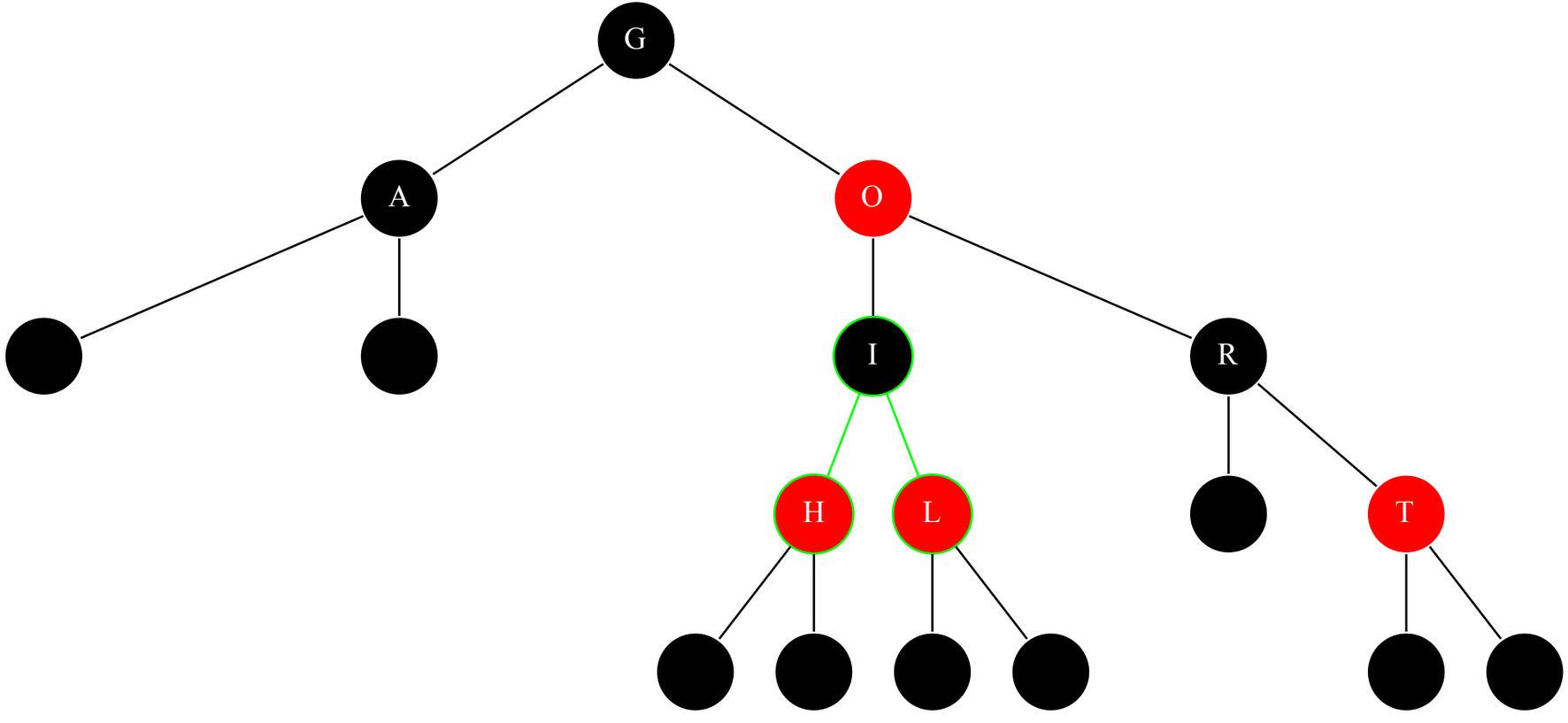
Now that we have positioned the new node ('H'), we need to ensure the correctness of the red-black tree.
Since we assumed the new node to be coloured red, we can only violate the 2. and 4. rules of the tree.
Now we set out to correct any discrepancies, by evaluating the colour of the newly added node's parent as well as its uncle, and acting accordingly.



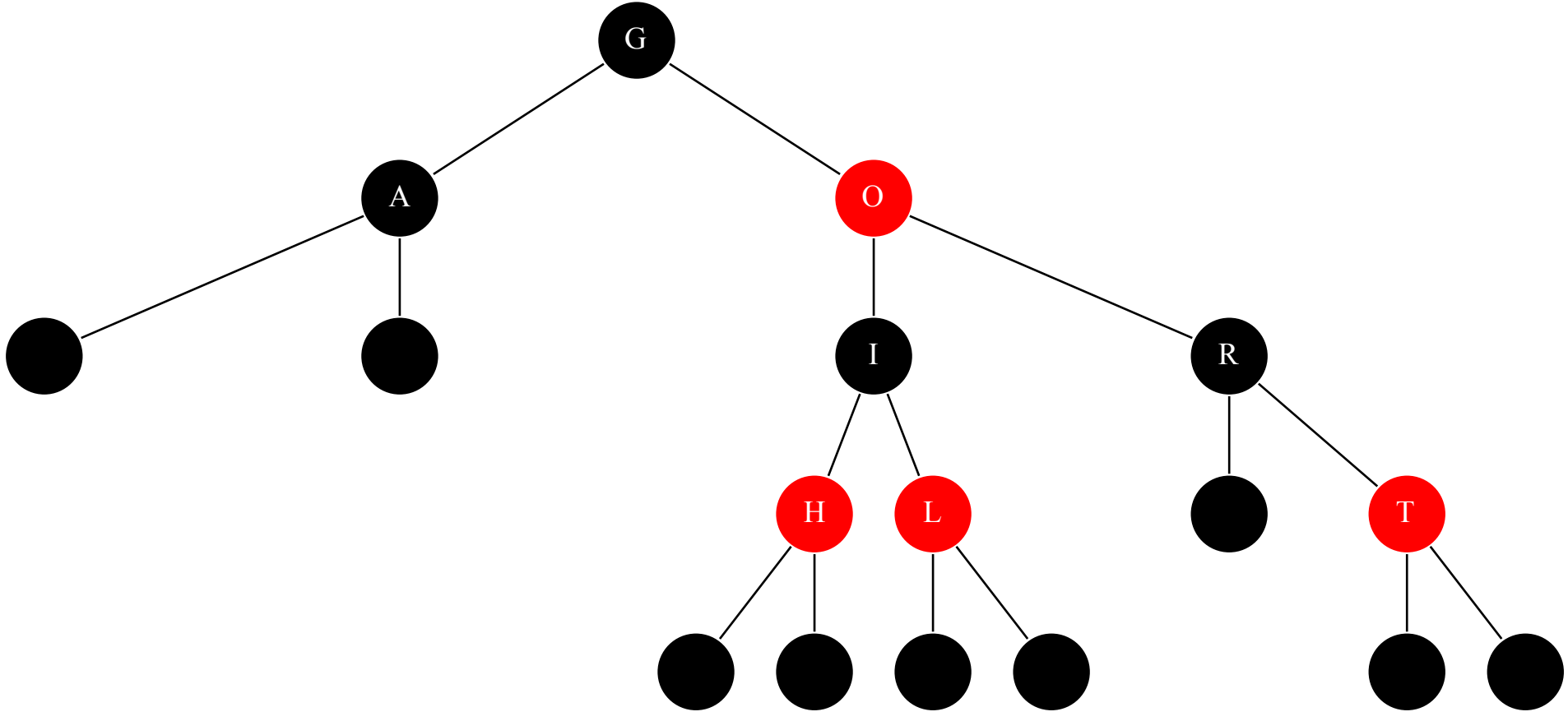
Since the currently examined node ('H'), its parent ('I') and grandparent ('L') are 'in-line', we now had to recolour them, so that when we rotate them in the next step, we will get a correct tree.



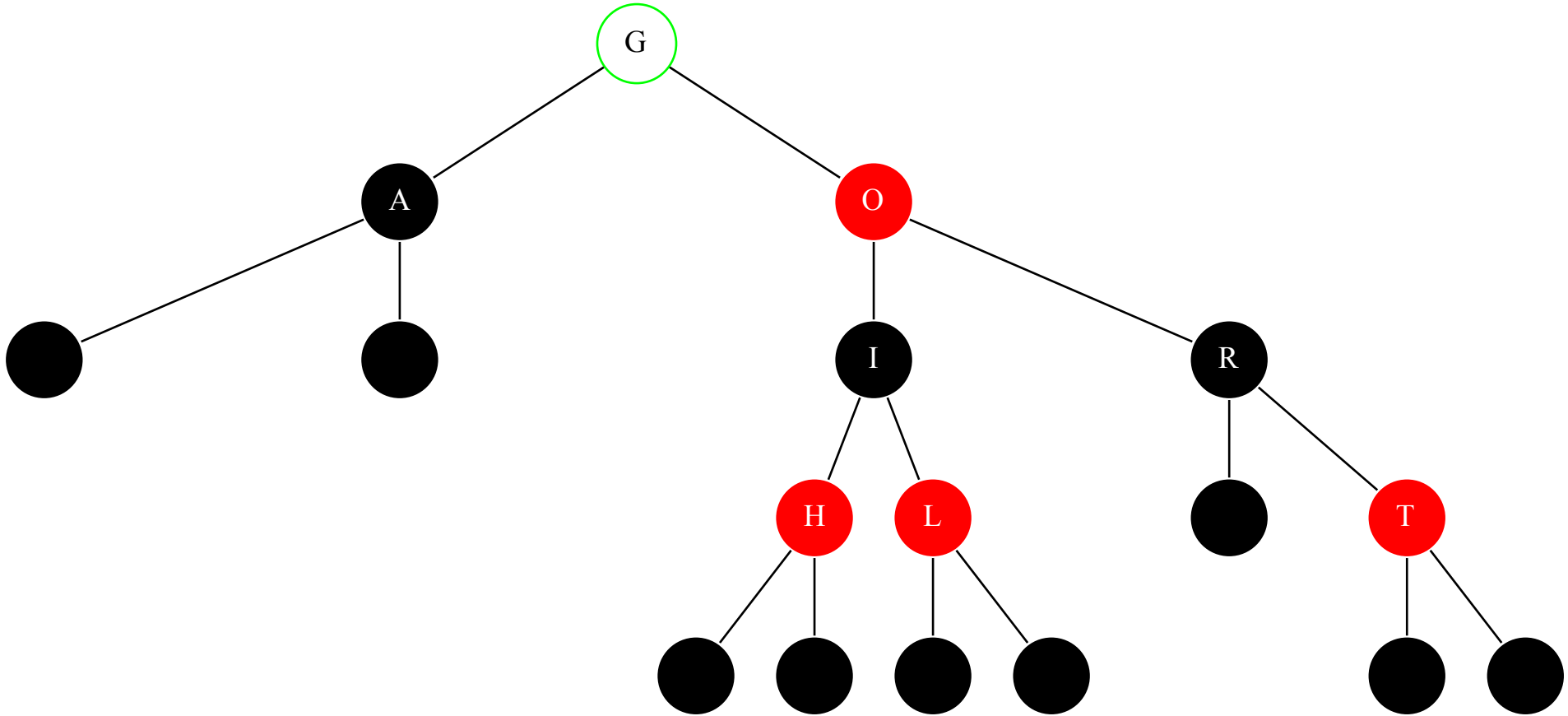
Since the currently examined node ('H'), its parent ('I') and grandparent ('H') are 'in-line' and properly coloured now, just one more rotation gives a correct red-black tree.



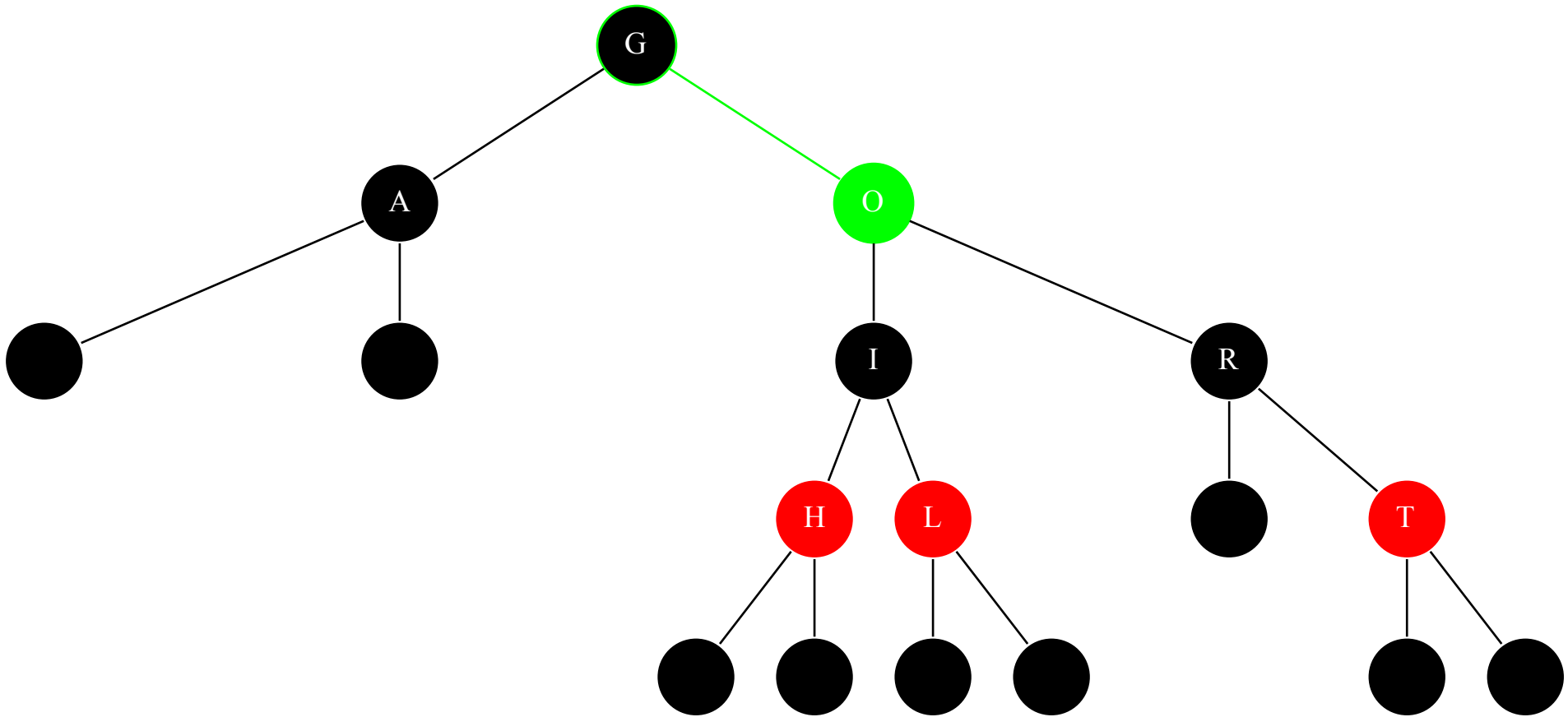
Since all of the required red-black properties are satisfied, we have arrived to a correctly formed red-black tree.



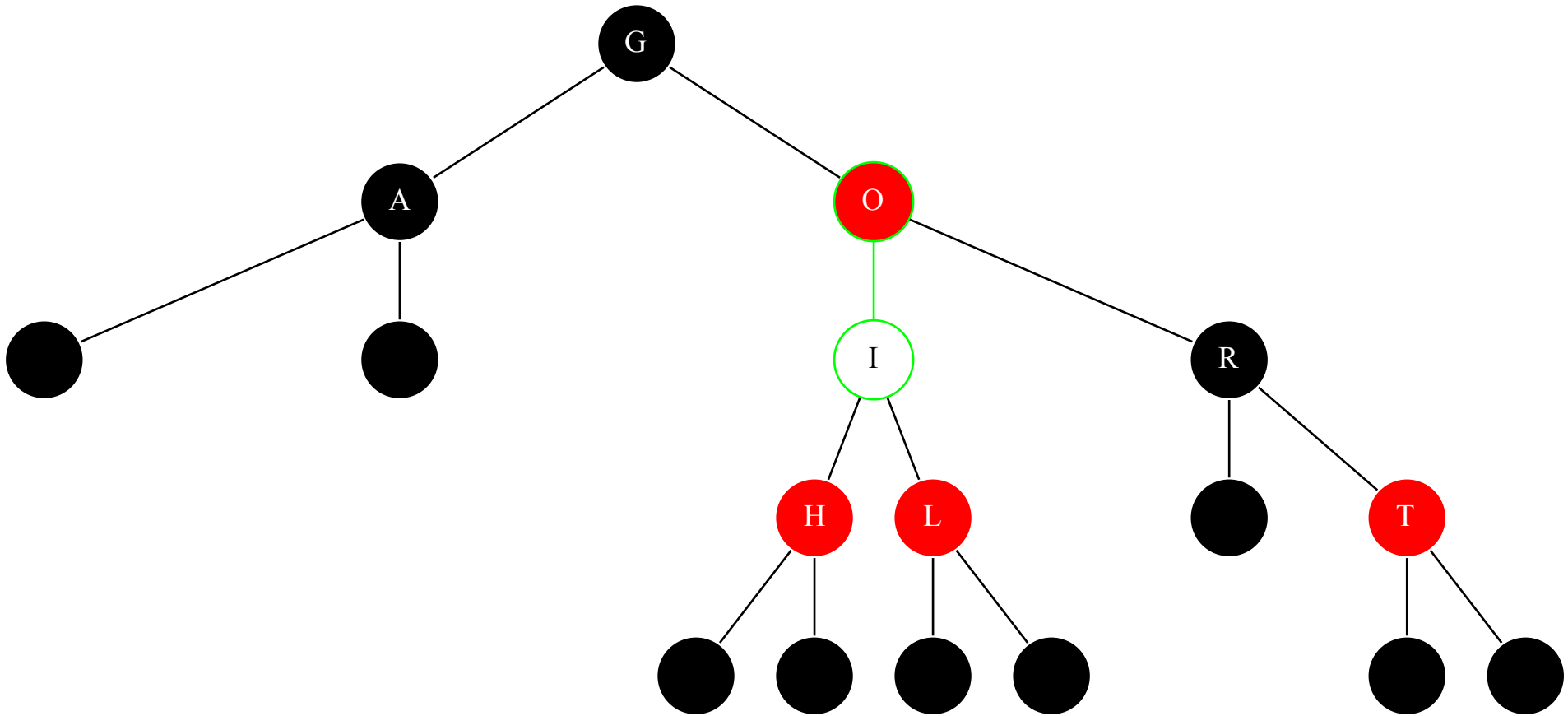
We are adding a new node with key 'M' to the tree.
The first step is to find a leaf (Nill), which the new node will replace.
We begin by examining the root (key 'G').



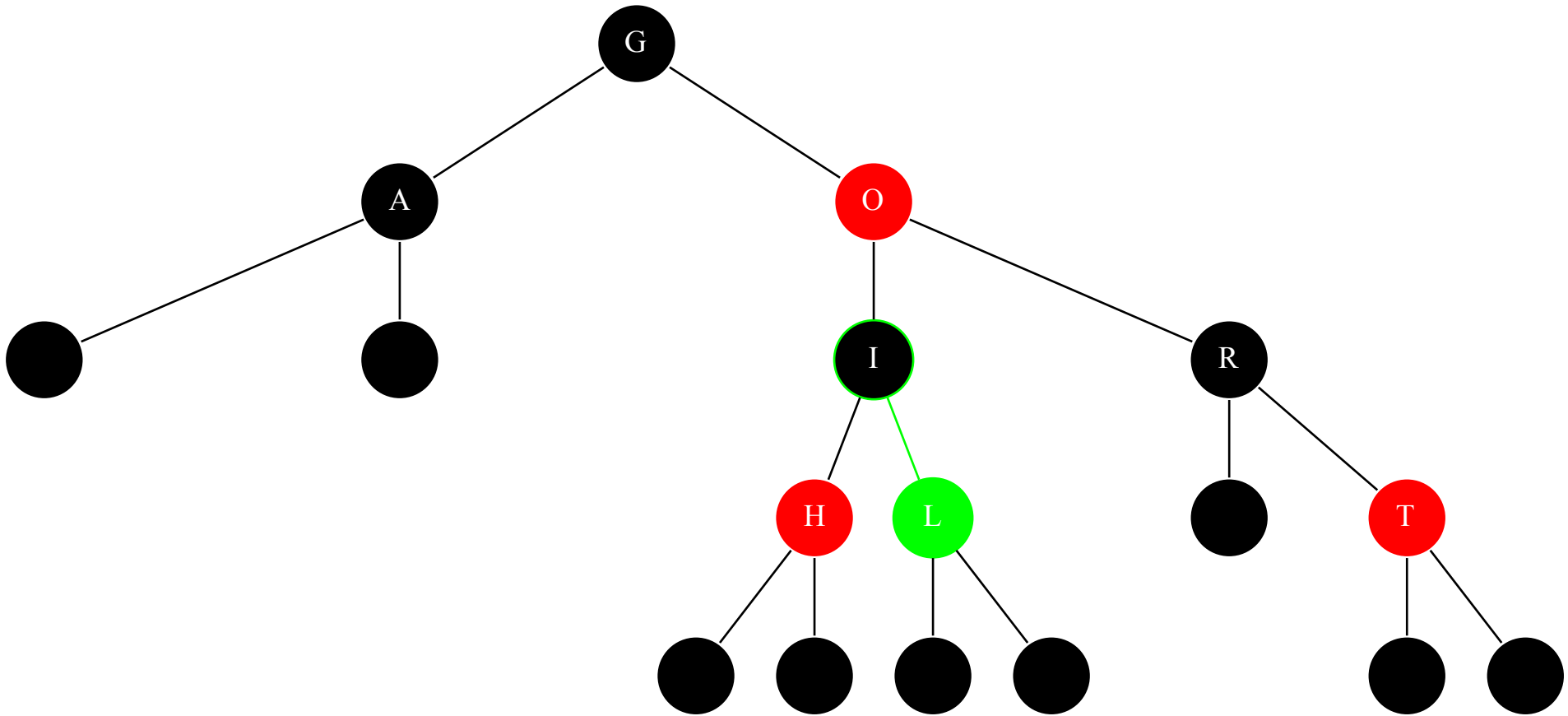
We are adding a new node with key 'M' to the tree.
By examining the node with key 'G' we have arrived at the node with key 'O'.



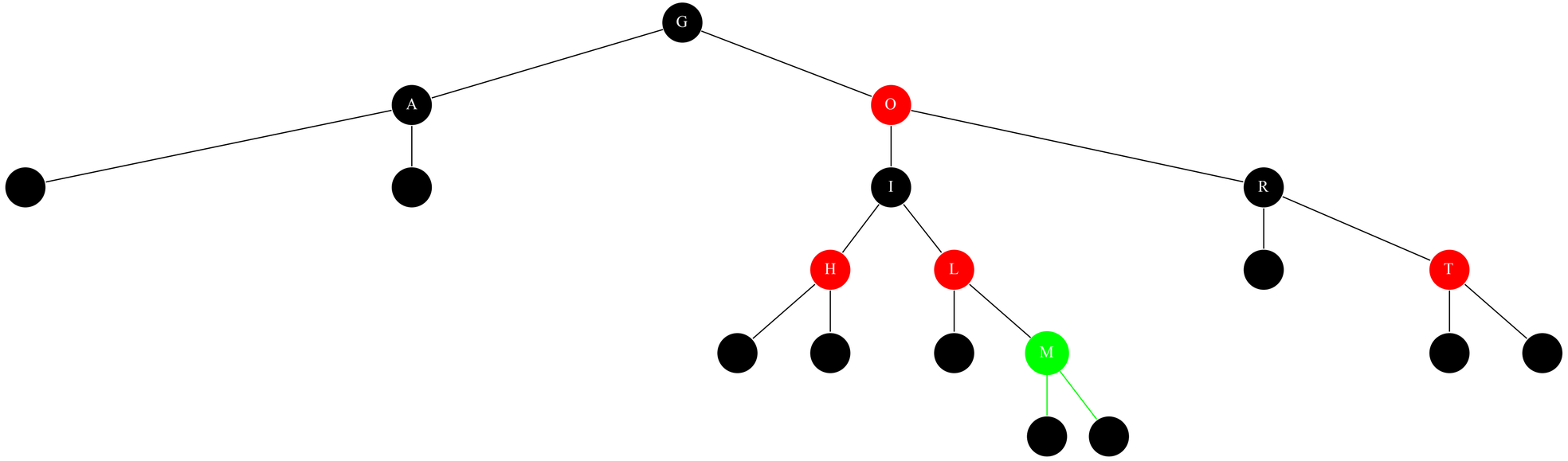
We are adding a new node with key 'M' to the tree.
By examining the node with key 'O' we have arrived at the node with key 'I'.



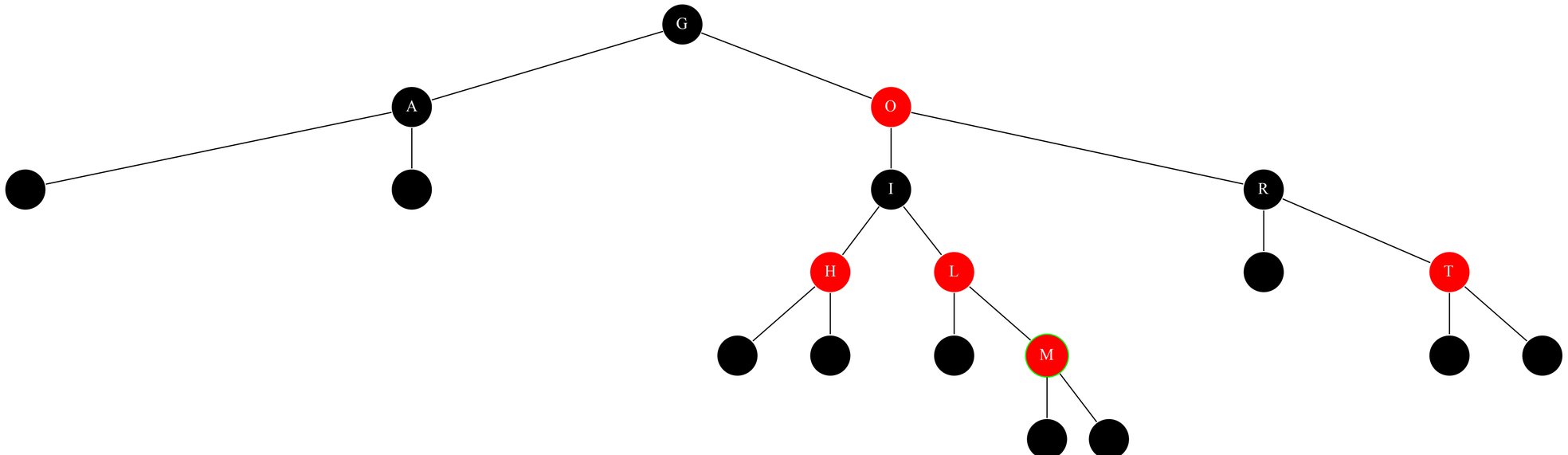
We are adding a new node with key 'M' to the tree.
By examining the node with key 'I' we have arrived at the node with key 'L'.



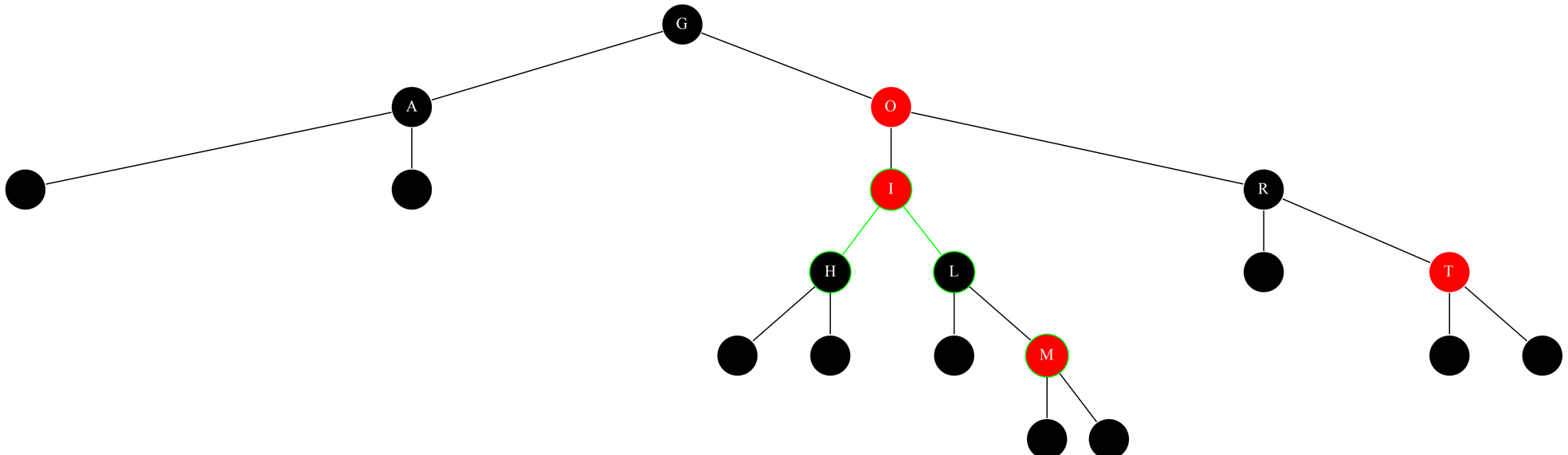
Since the key of the new node ('M') is greater than the one of its new parent ('L'), we add it to the right of the parent.



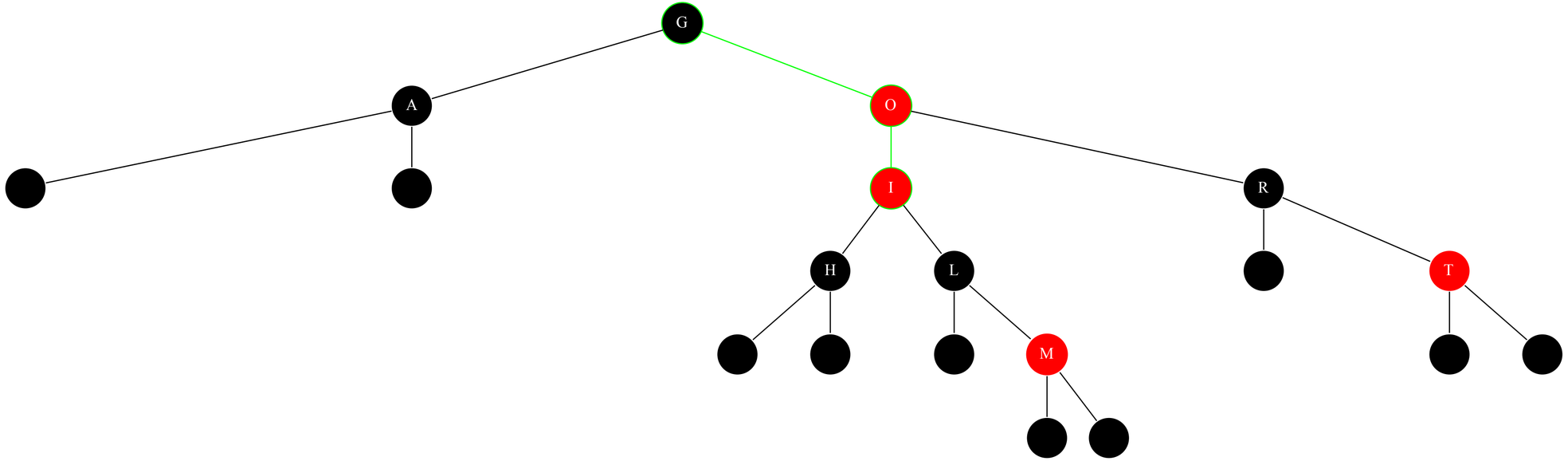
Now that we have positioned the new node ('M'), we need to ensure the correctness of the red-black tree.
Since we assumed the new node to be coloured red, we can only violate the 2. and 4. rules of the tree.
Now we set out to correct any discrepancies, by evaluating the colour of the newly added node's parent as well as its uncle, and acting accordingly.



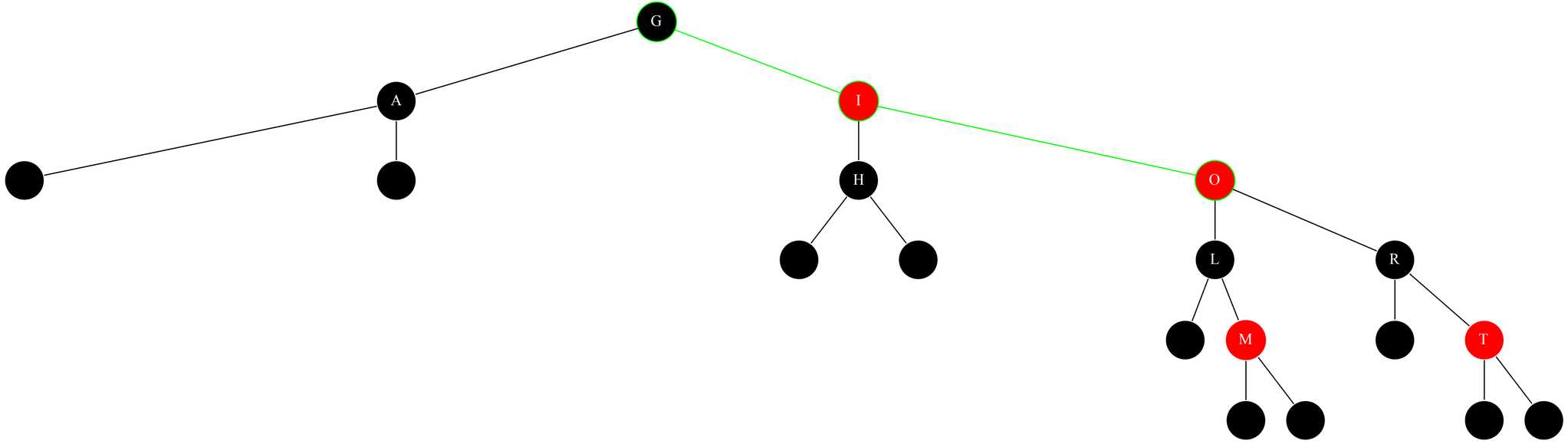
Since the uncle ('H') was red and we know that the parent ('L') was also red, we just had to perform a colour rebalancing.
This means that we coloured the currently examined nodes parent ('L') and uncle black (H), and coloured its grandfather ('I') red.
By doing this we correct the fourth rule of the red-black trees.



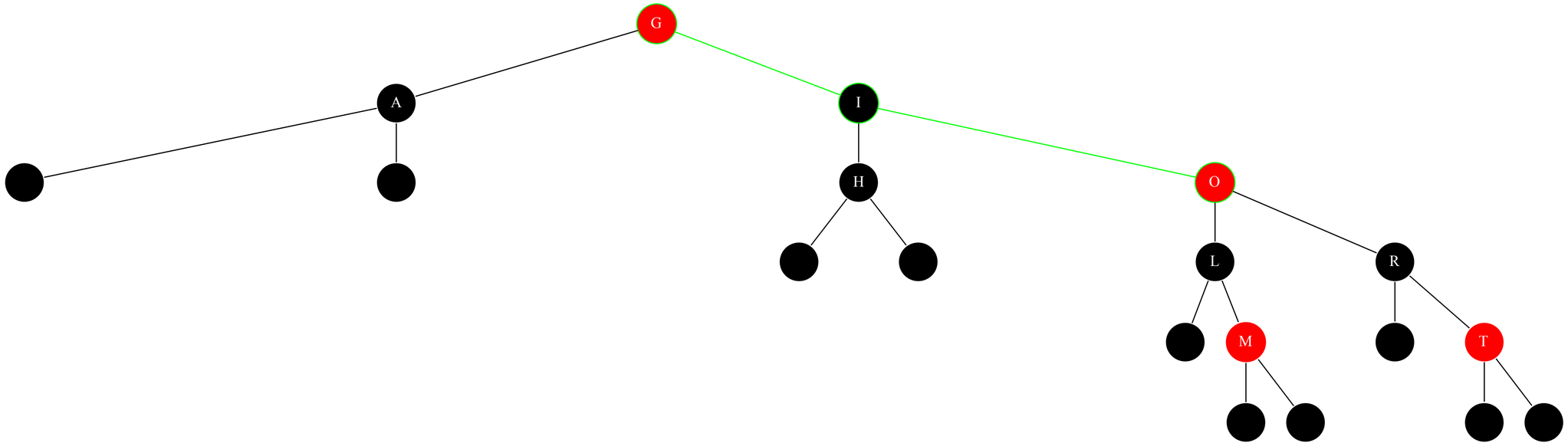
Since the uncle ('A') is black, and the parent ('O') is red, we cannot just perform a recolouring, but have to perform a rotation.
This rotation is needed because the currently examined node ('I'), its father ('O') and grandfather ('G') are not 'in-line'.



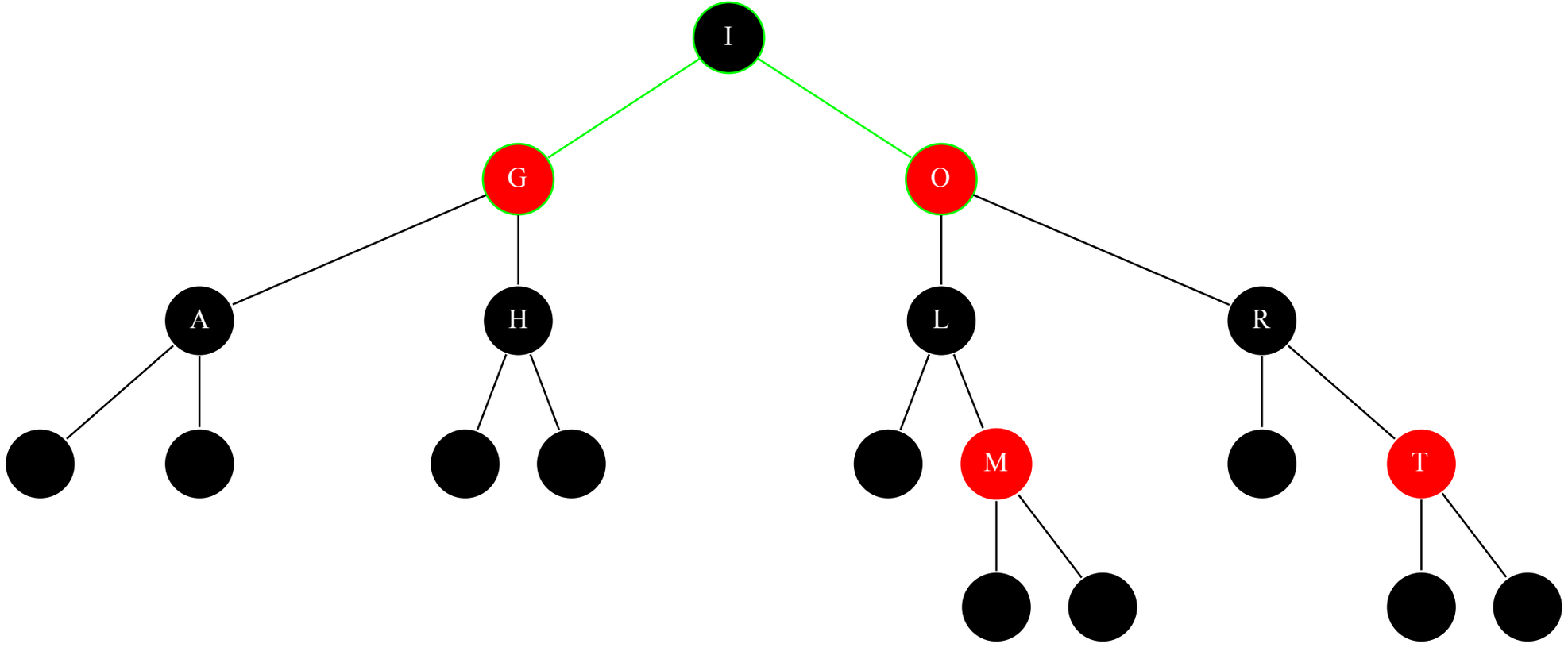
After we performed the before-mentioned rotation, the currently examined node ('O'), its parent ('I') and grandparent ('G') are now 'in-line'.



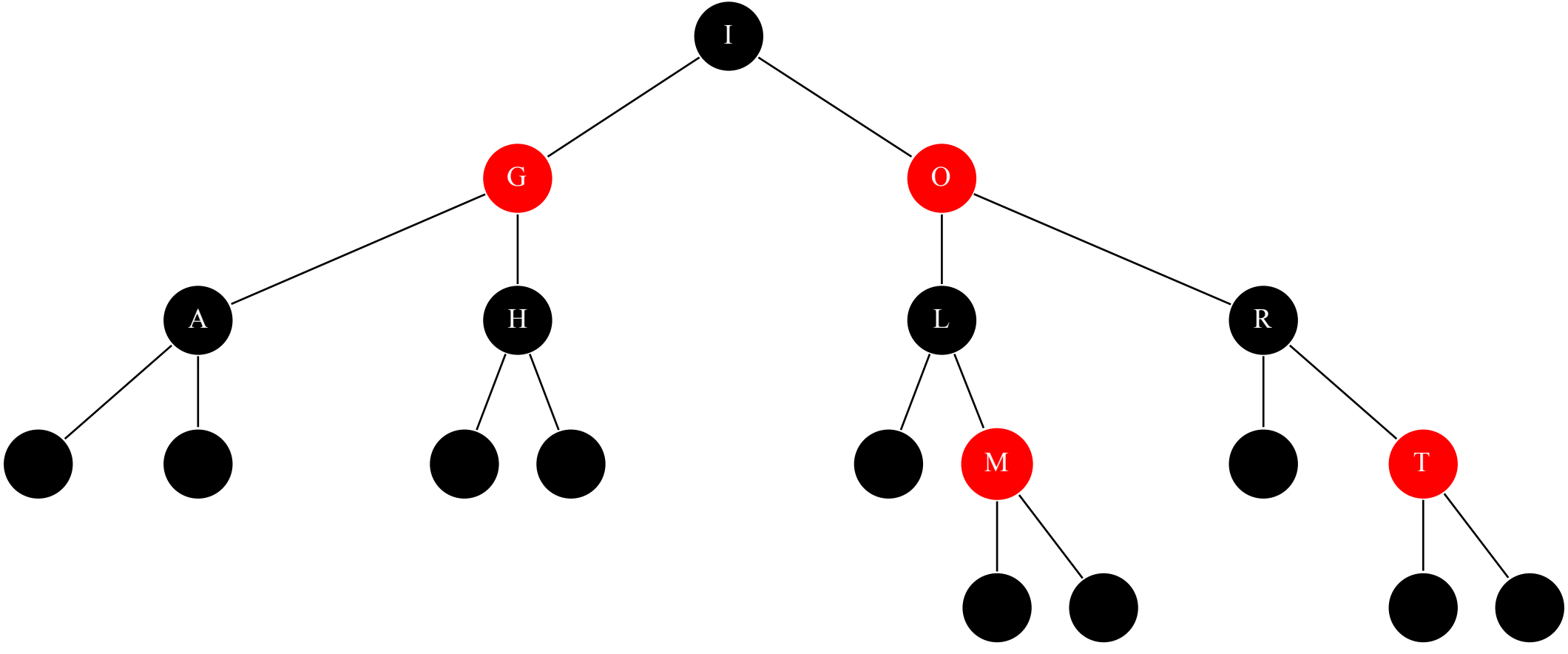
We have now made the currently examined node ('O'), its parent ('I') and grandparent ('G') align.
We now had to recolour them, so that when we rotate them in the next step, we will get a correct tree.



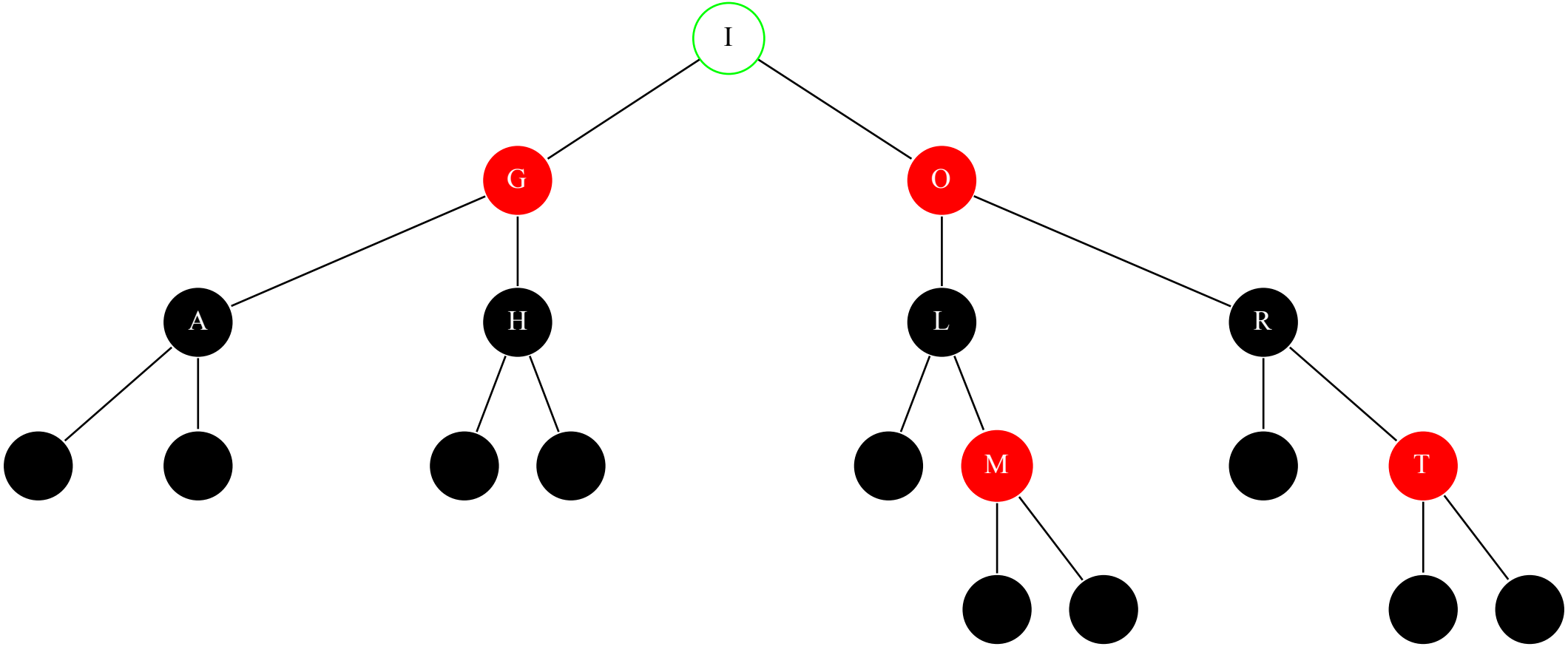
Since the currently examined node ('O'), its parent ('I') and grandparent ('G') are 'in-line' and properly coloured now, just one more rotation gives a correct red-black tree.



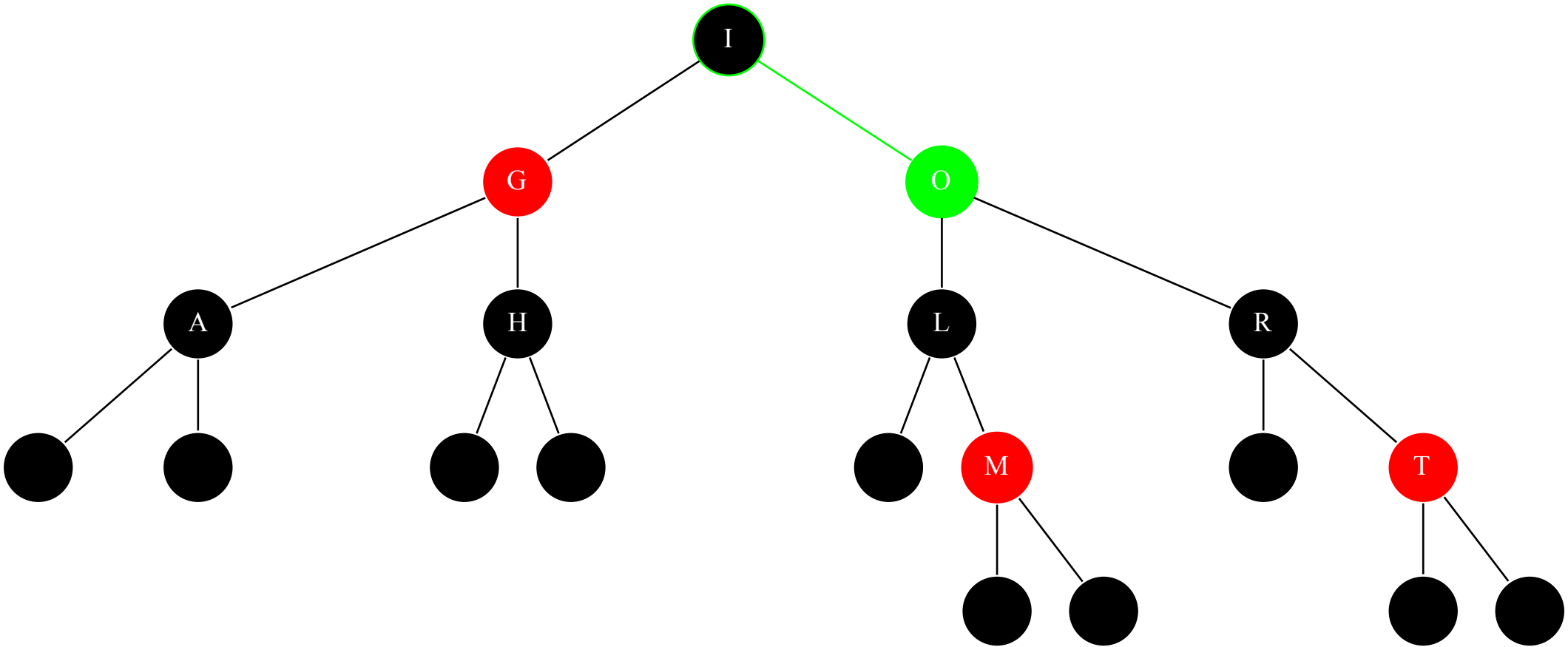
Since all of the required red-black properties are satisfied, we have arrived to a correctly formed red-black tree.



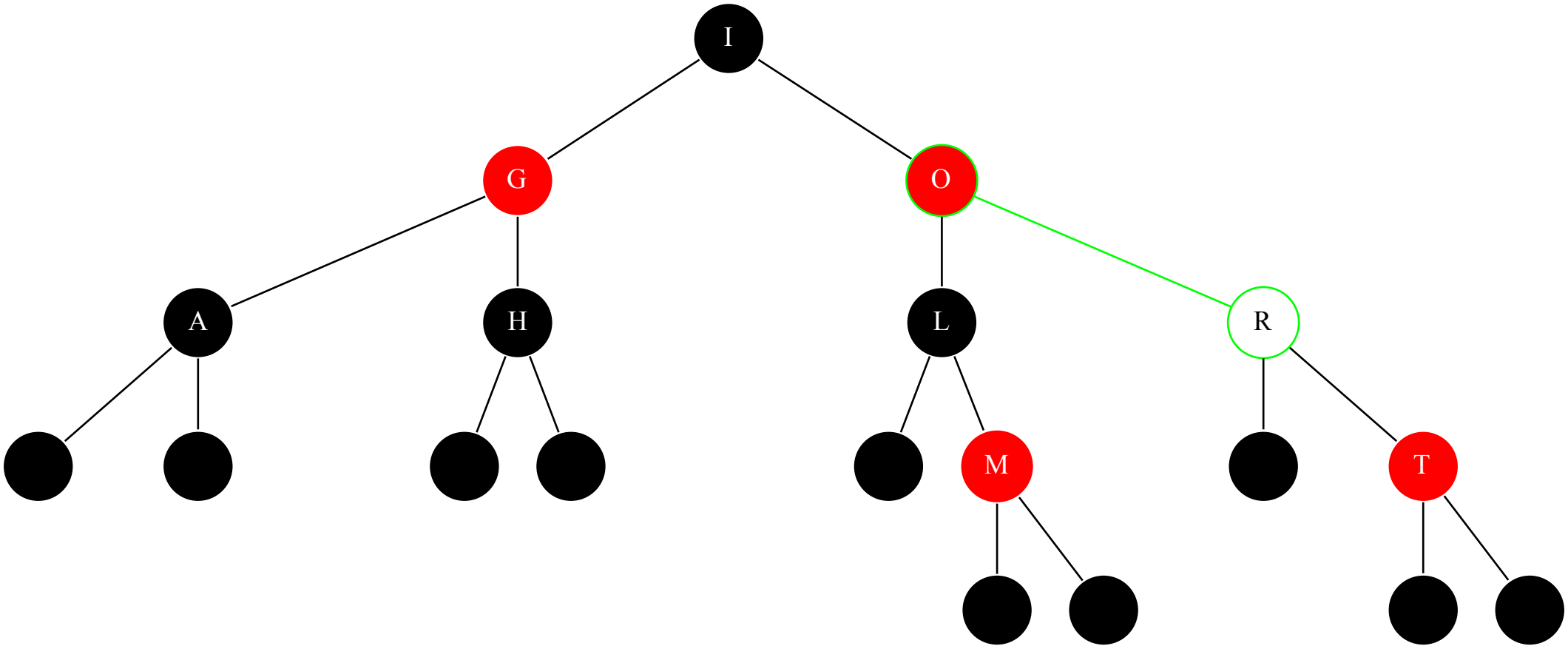
We are adding a new node with key 'S' to the tree.
The first step is to find a leaf (Nill), which the new node will replace.
We begin by examining the root (key 'I').



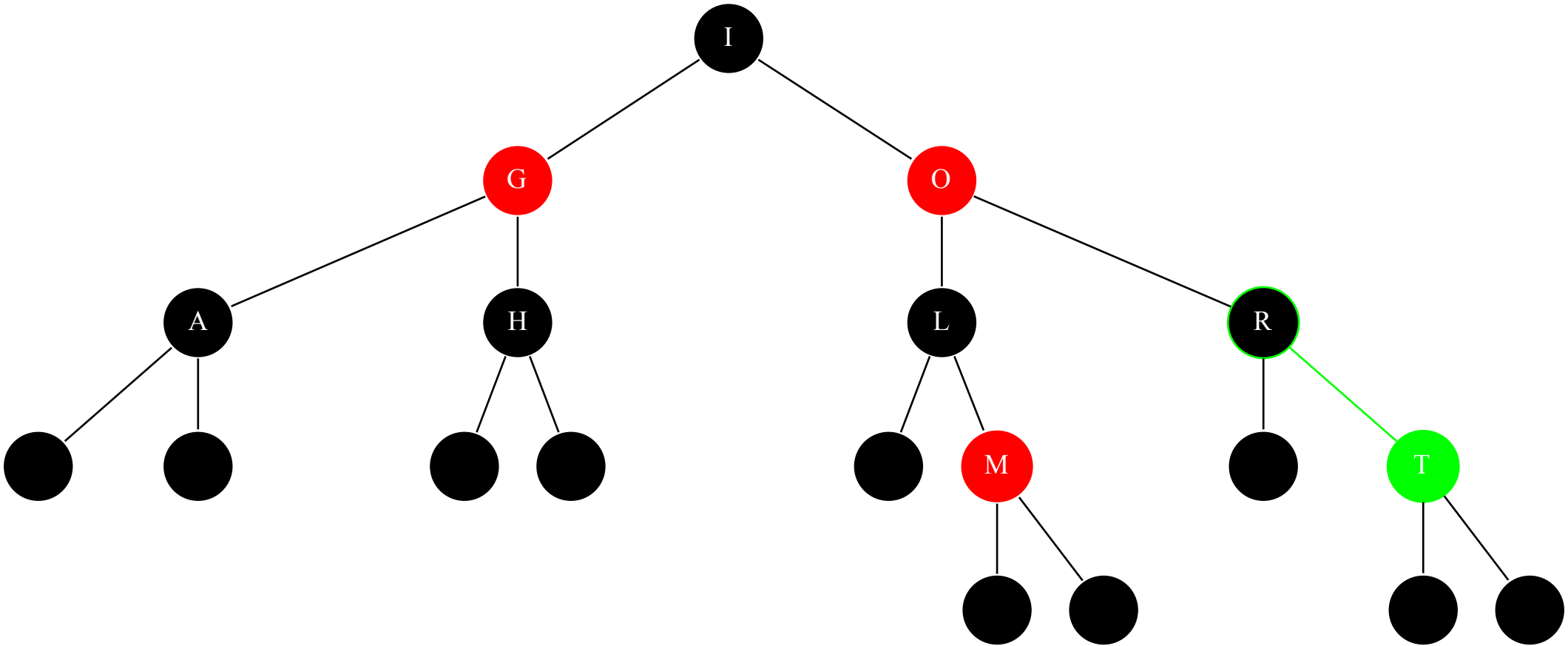
We are adding a new node with key 'S' to the tree.
By examining the node with key 'I' we have arrived at the node with key 'O'.



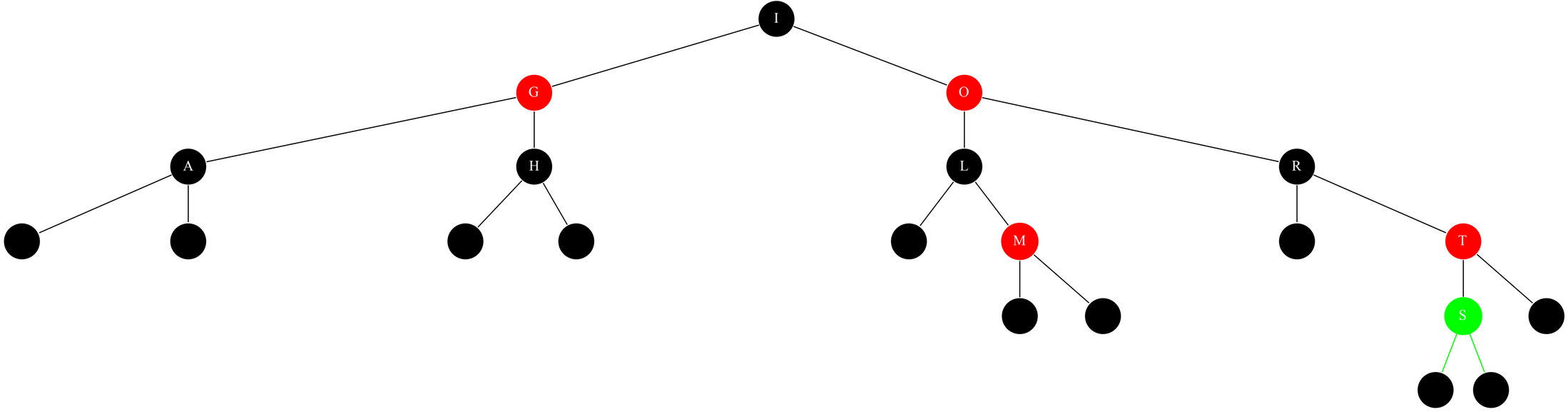
We are adding a new node with key 'S' to the tree.
By examining the node with key 'O' we have arrived at the node with key 'R'.



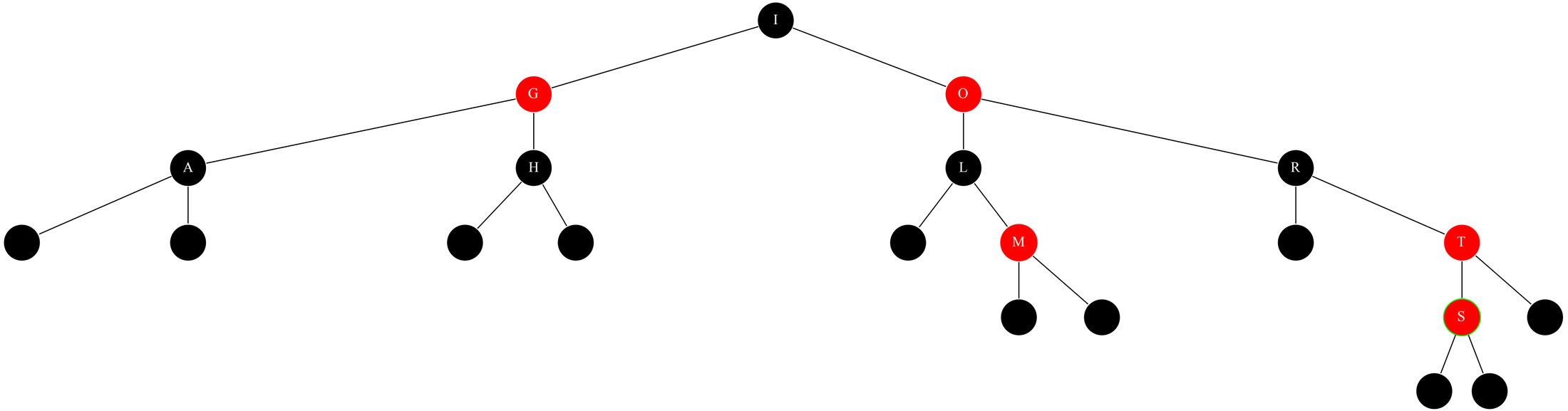
We are adding a new node with key 'S' to the tree.
By examining the node with key 'R' we have arrived at the node with key 'T'.



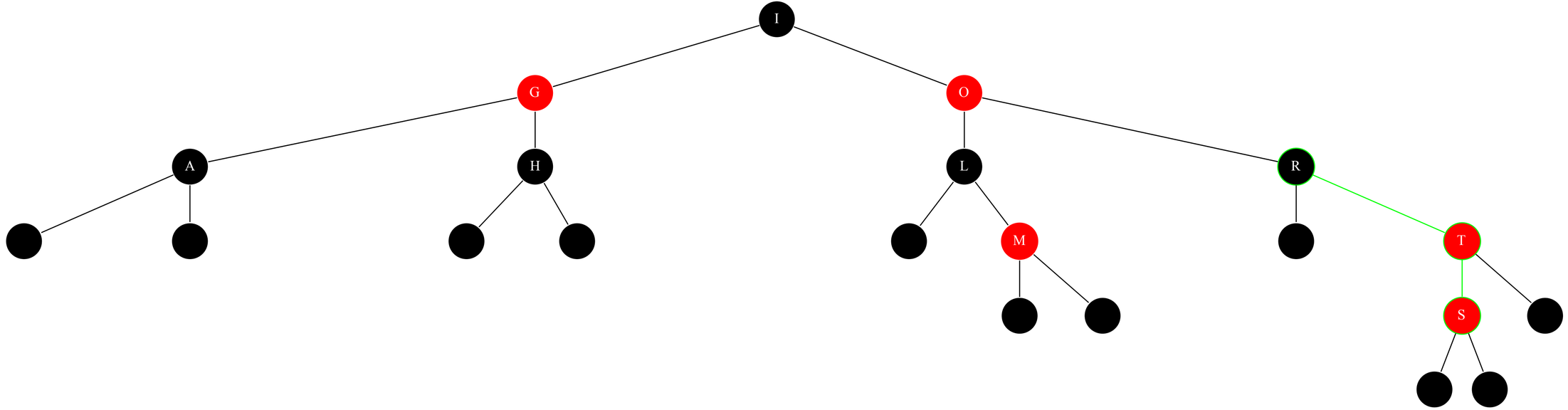
Since the key of the new node ('S') is smaller than the one of its new parent ('T'), we add it to the left of the parent.



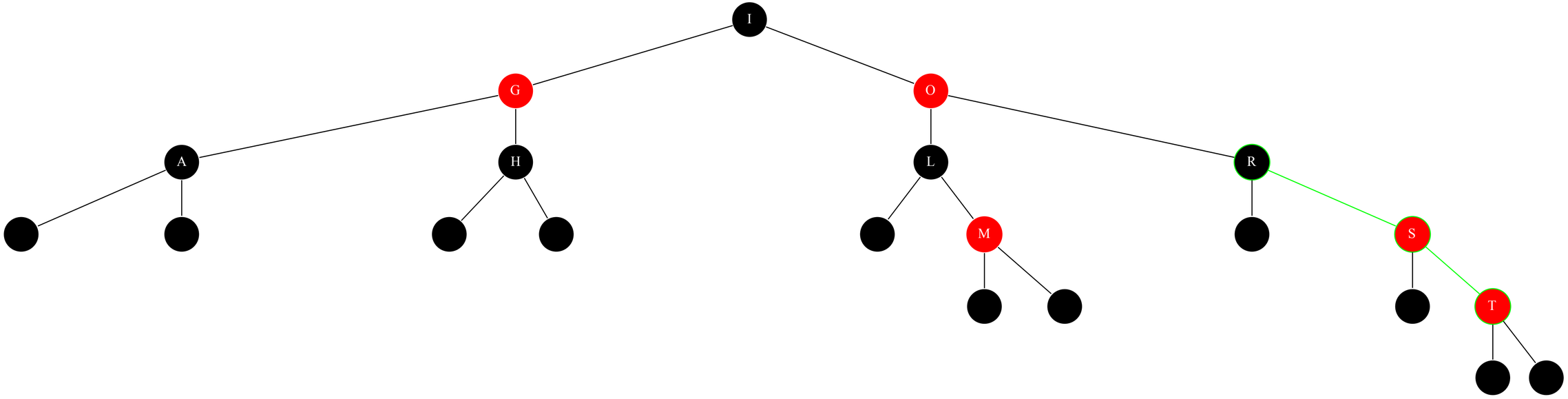
Now that we have positioned the new node ('S'), we need to ensure the correctness of the red-black tree.
Since we assumed the new node to be coloured red, we can only violate the 2. and 4. rules of the tree.
Now we set out to correct any discrepancies, by evaluating the colour of the newly added node's parent as well as its uncle, and acting accordingly.



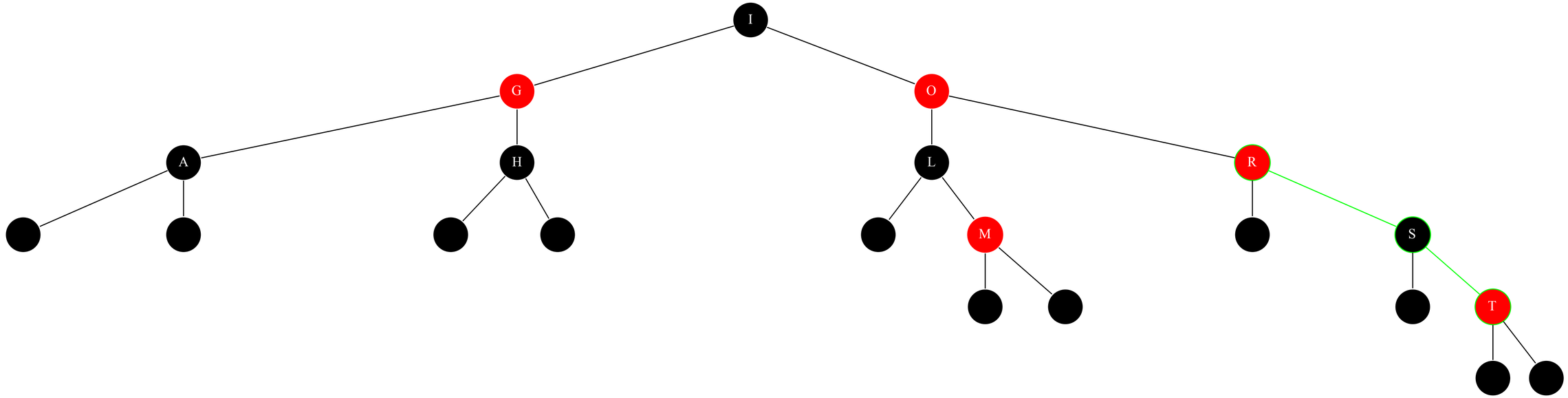
Since the uncle ('Nill') is black, and the parent ('T') is red, we cannot just perform a recolouring, but have to perform a rotation.
This rotation is needed because the currently examined node ('S'), its father ('T') and grandfather ('R') are not 'in-line'.



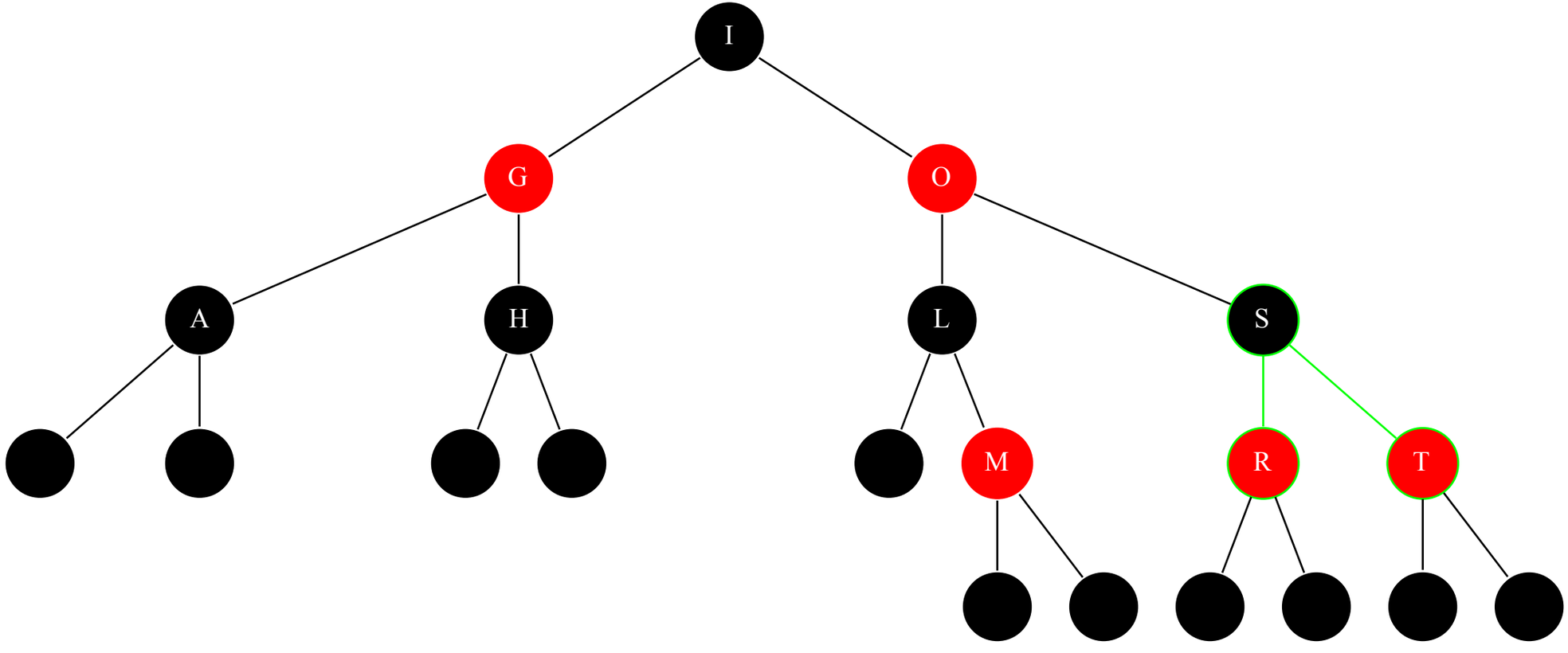
After we performed the before-mentioned rotation, the currently examined node ('T'), its parent ('S') and grandparent ('R') are now 'in-line'.



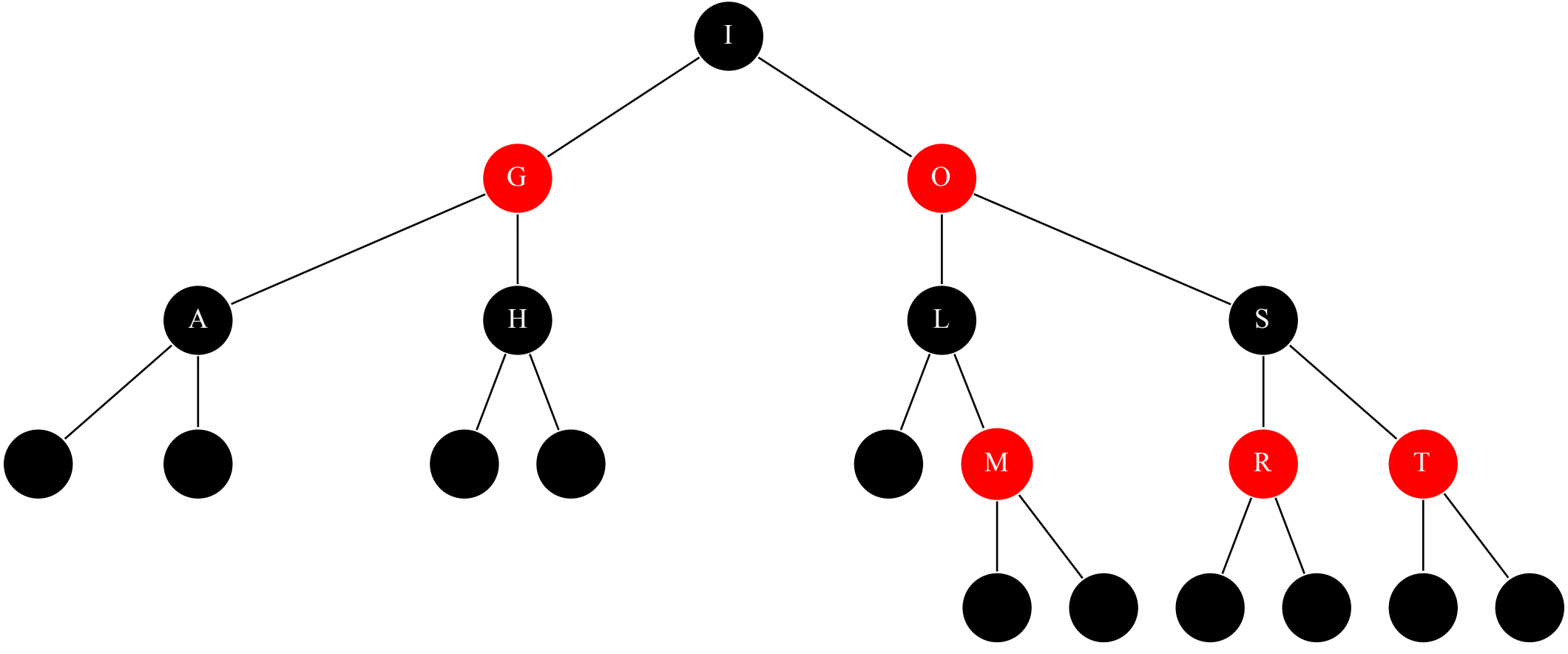
We have now made the currently examined node ('T'), its parent ('S') and grandparent ('R') align.
We now had to recolour them, so that when we rotate them in the next step, we will get a correct tree.



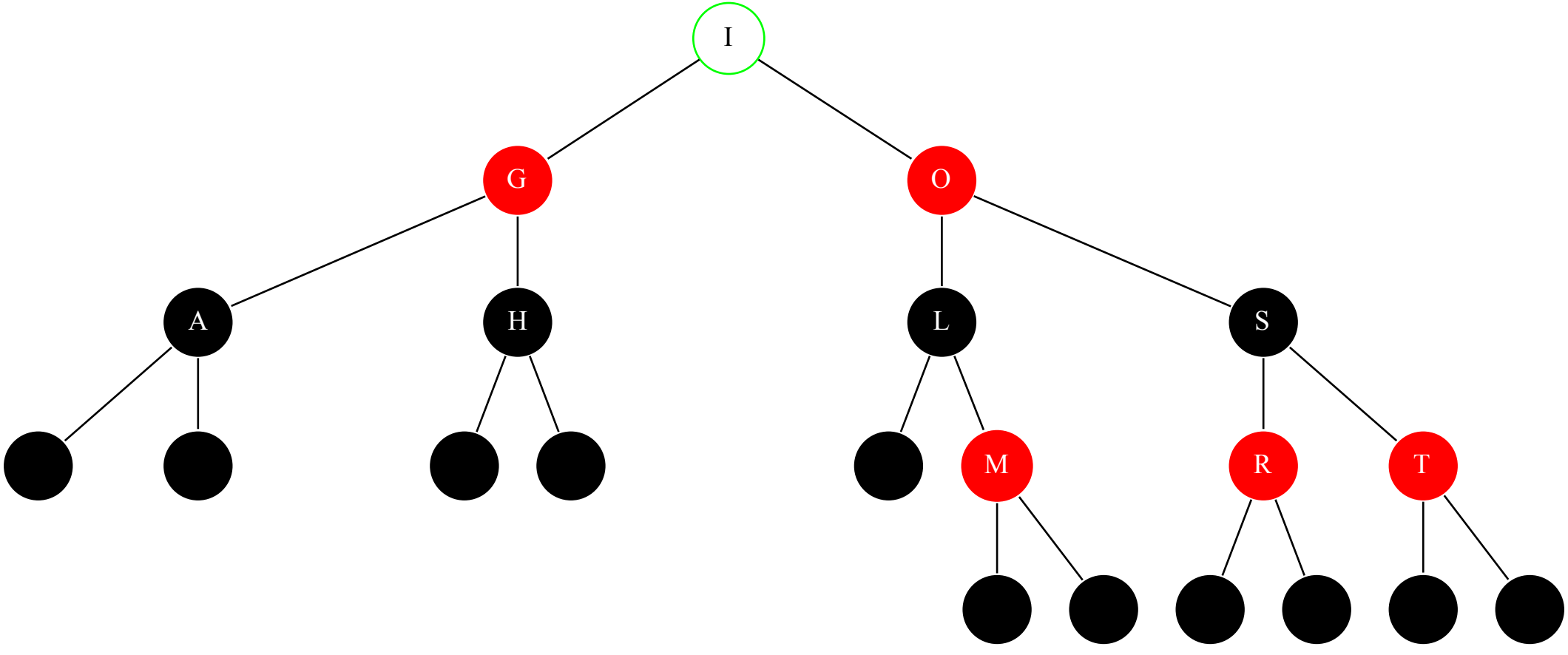
Since the currently examined node ('T'), its parent ('S') and grandparent ('R') are 'in-line' and properly coloured now, just one more rotation gives a correct red-black tree.



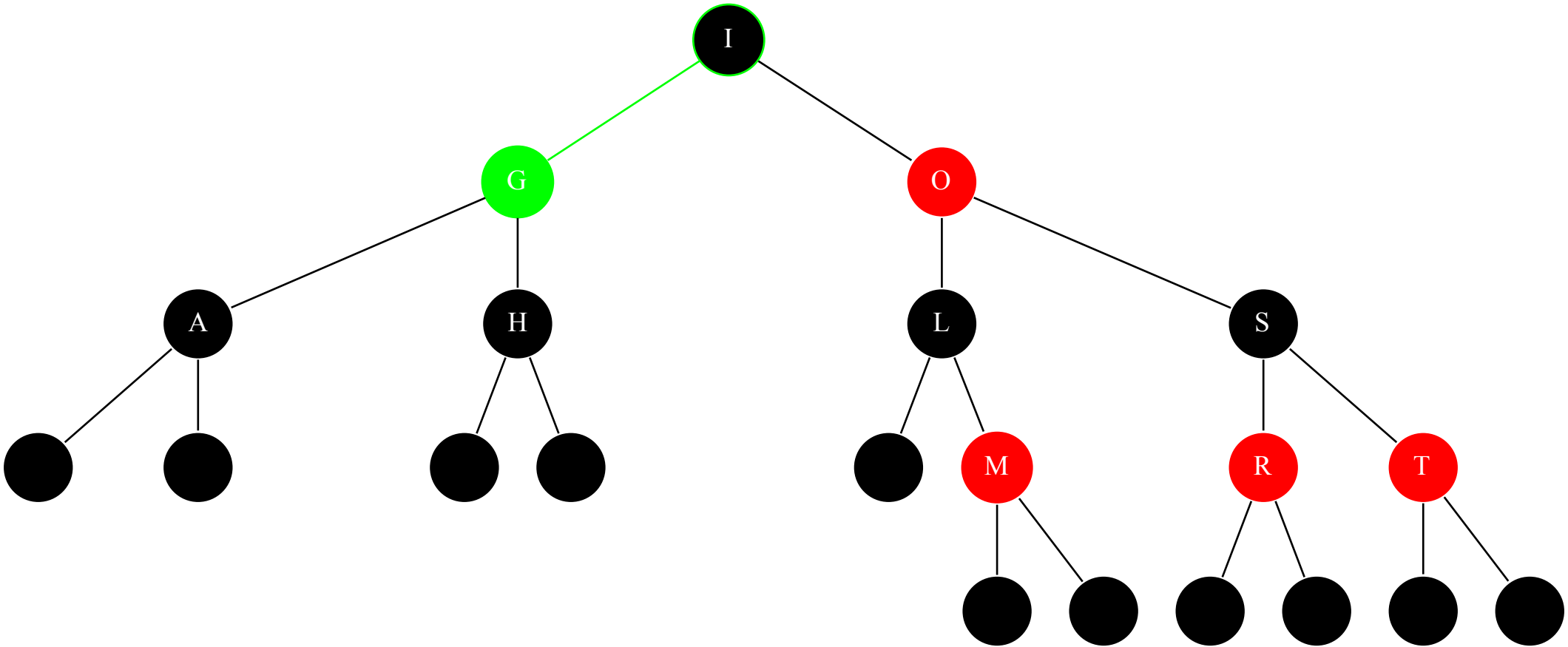
Since all of the required red-black properties are satisfied, we have arrived to a correctly formed red-black tree.



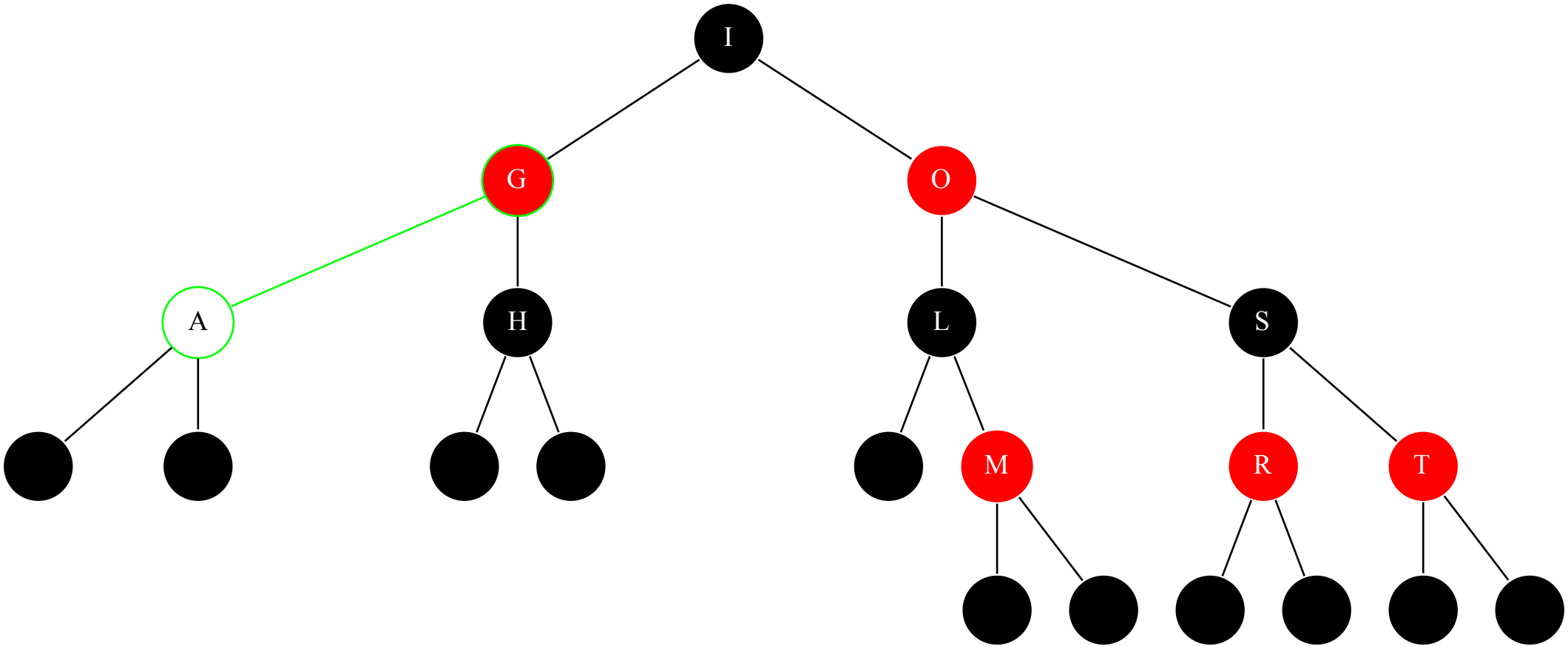
We are adding a new node with key 'C' to the tree.
The first step is to find a leaf (Nill), which the new node will replace.
We begin by examining the root (key 'I').



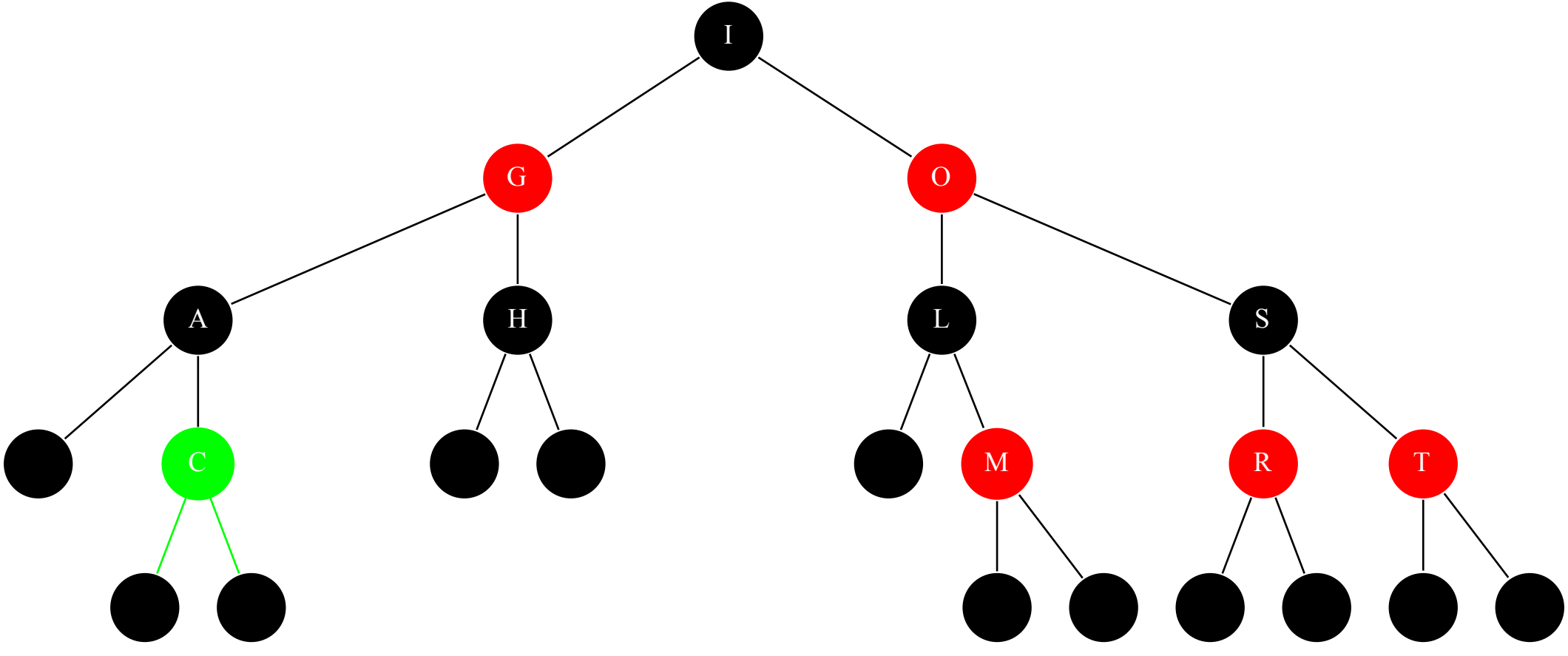
We are adding a new node with key 'C' to the tree.
By examining the node with key 'I' we have arrived at the node with key 'G'.



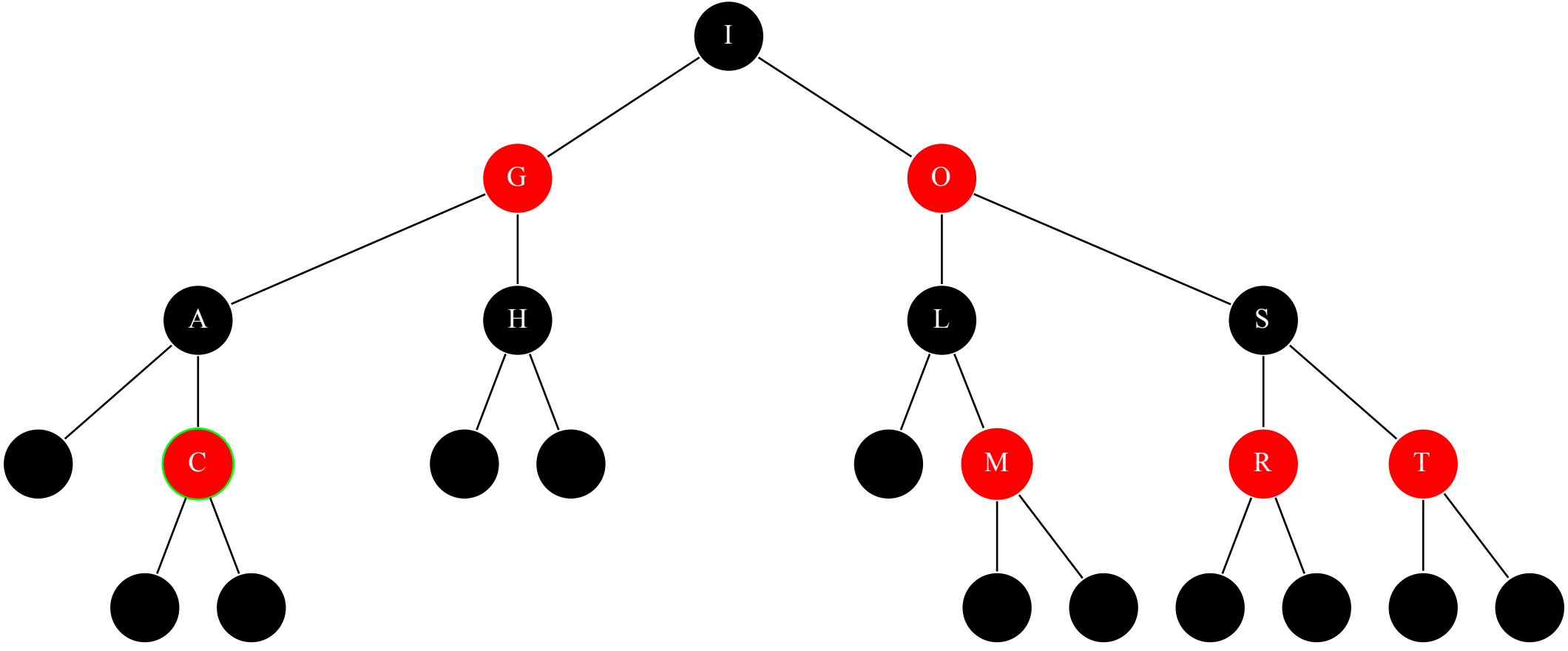
We are adding a new node with key 'C' to the tree.
By examining the node with key 'G' we have arrived at the node with key 'A'.



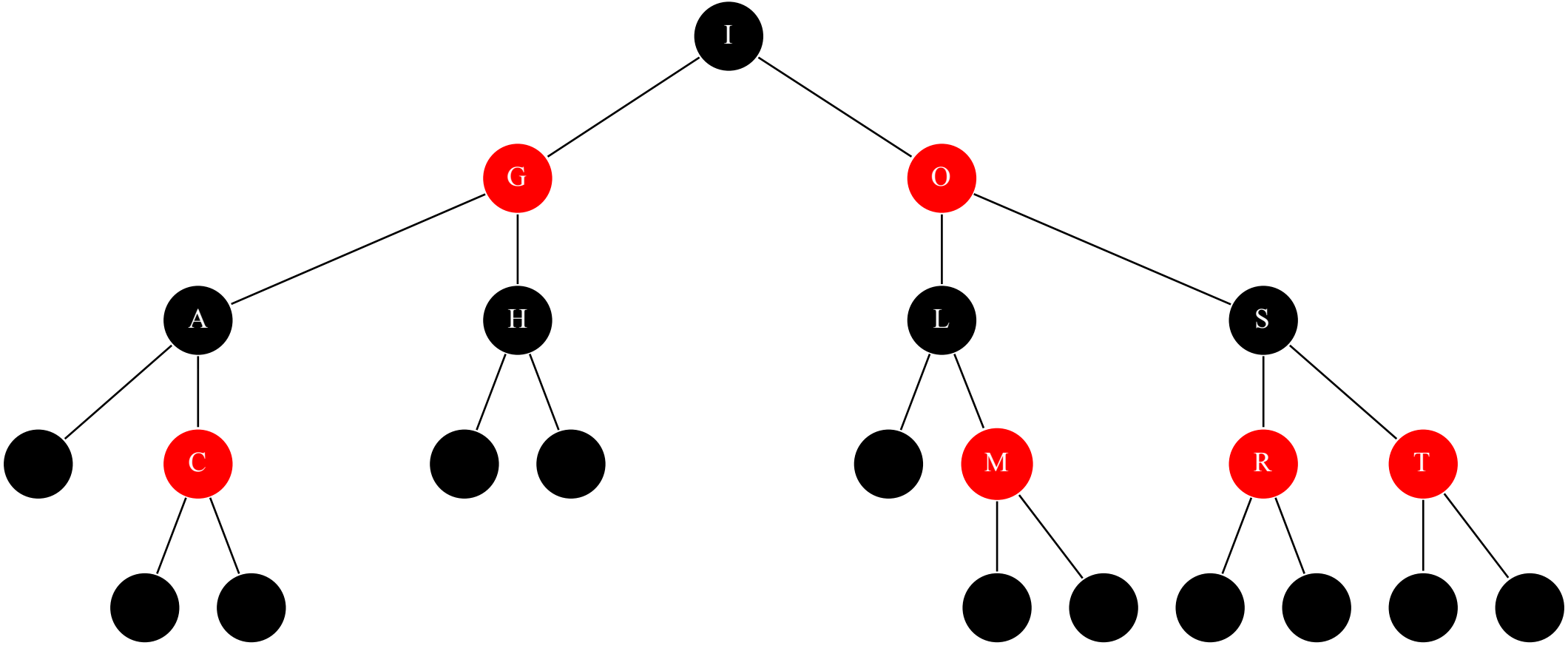
Since the key of the new node ('C') is greater than the one of its new parent ('A'), we add it to the right of the parent.



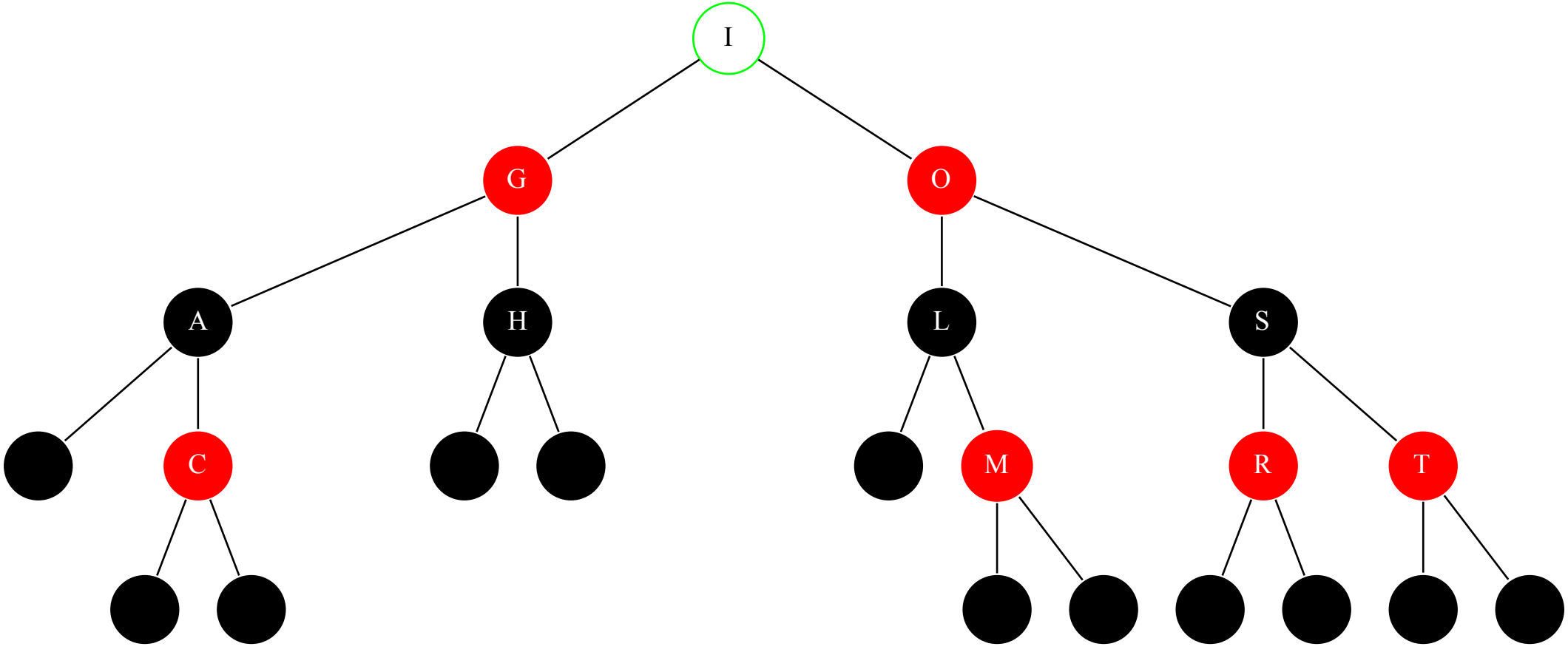
Now that we have positioned the new node ('C'), we need to ensure the correctness of the red-black tree.
Since we assumed the new node to be coloured red, we can only violate the 2. and 4. rules of the tree.
Now we set out to correct any discrepancies, by evaluating the colour of the newly added node's parent as well as its uncle, and acting accordingly.



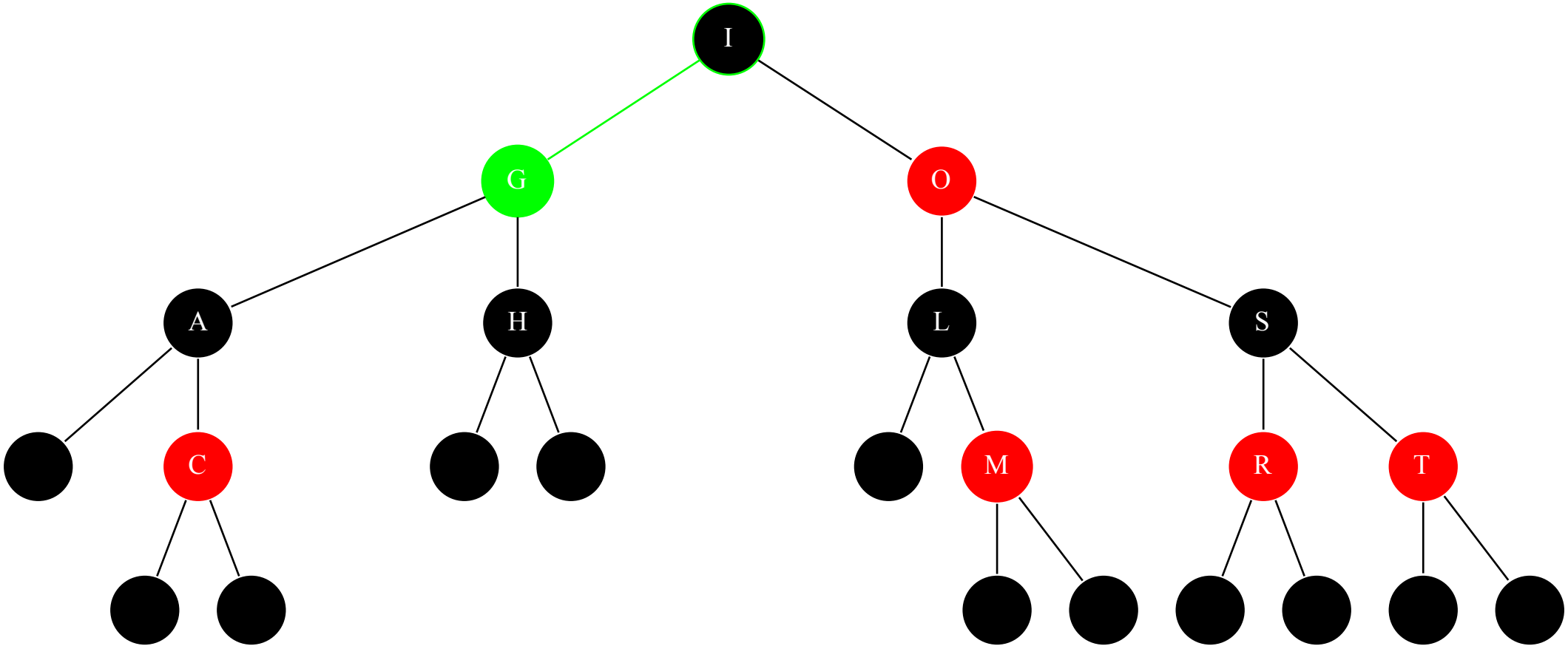
Since all of the required red-black properties are satisfied, we have arrived to a correctly formed red-black tree.



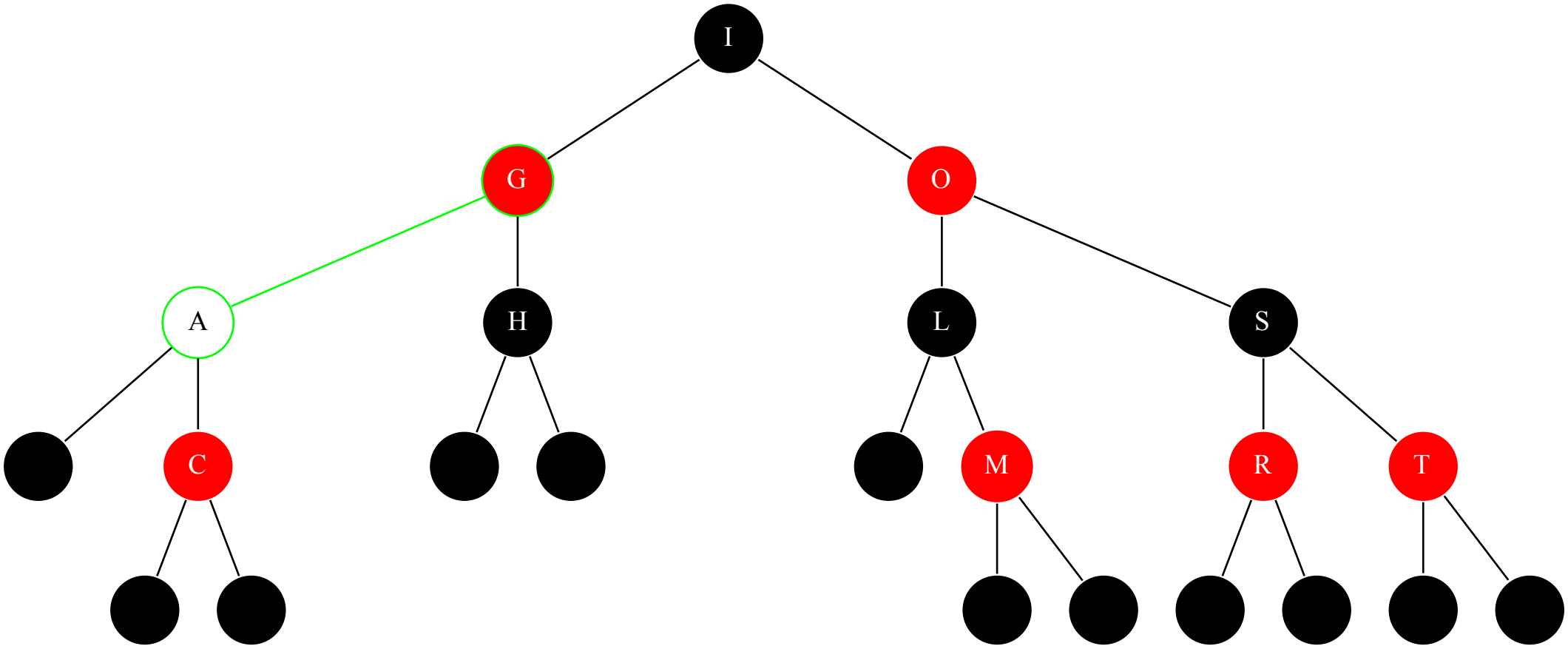
We are adding a new node with key 'A' to the tree.
The first step is to find a leaf (Nill), which the new node will replace.
We begin by examining the root (key 'I').



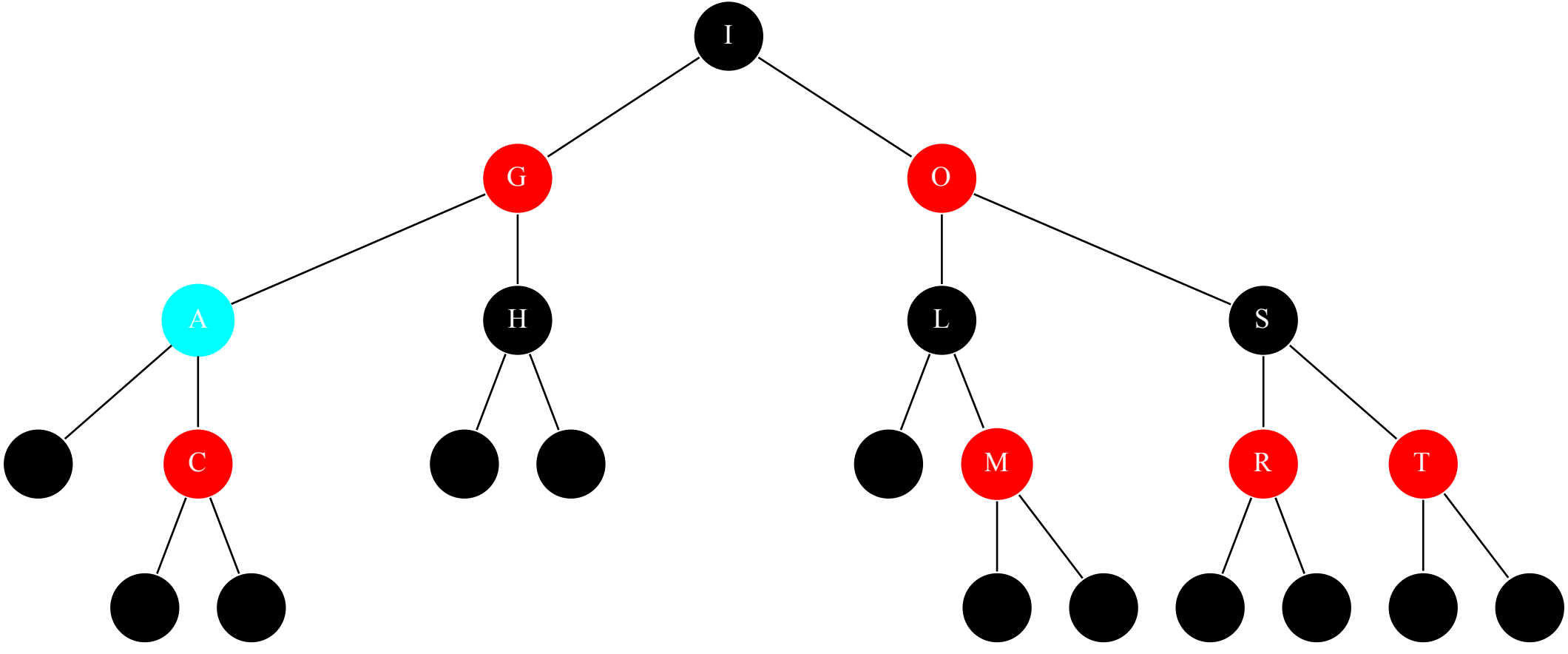
We are adding a new node with key 'A' to the tree.
By examining the node with key 'I' we have arrived at the node with key 'G'.



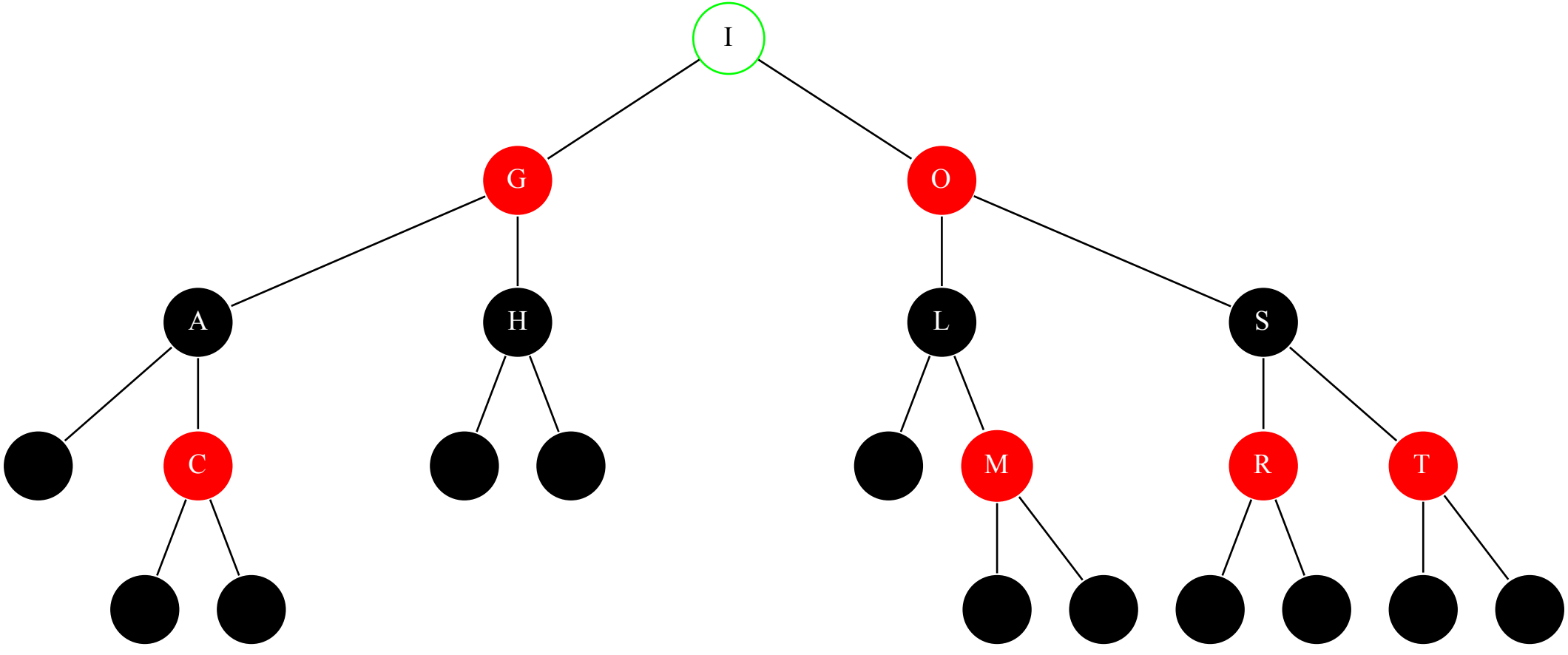
We are adding a new node with key 'A' to the tree.
By examining the node with key 'G' we have arrived at the node with key 'A'.



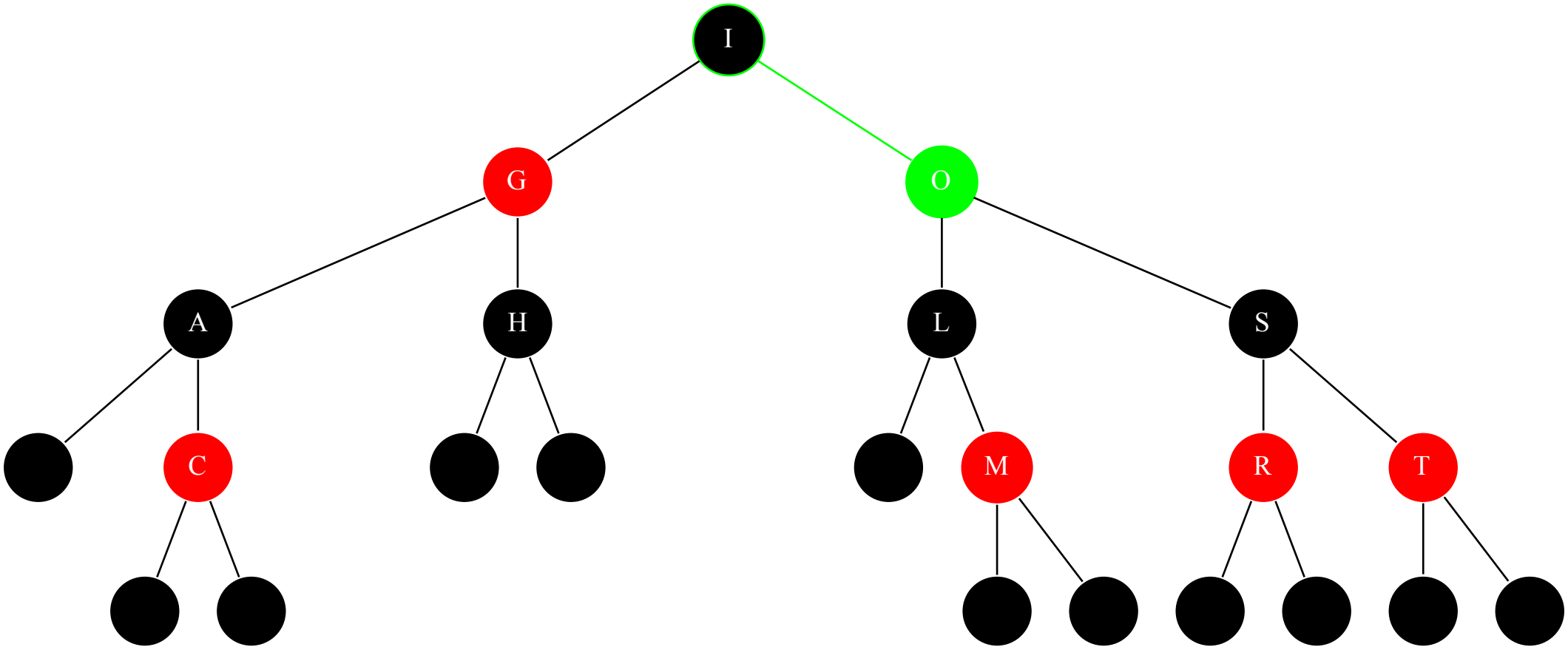
The node with key 'A' is already in the tree, so we terminate the insertion.



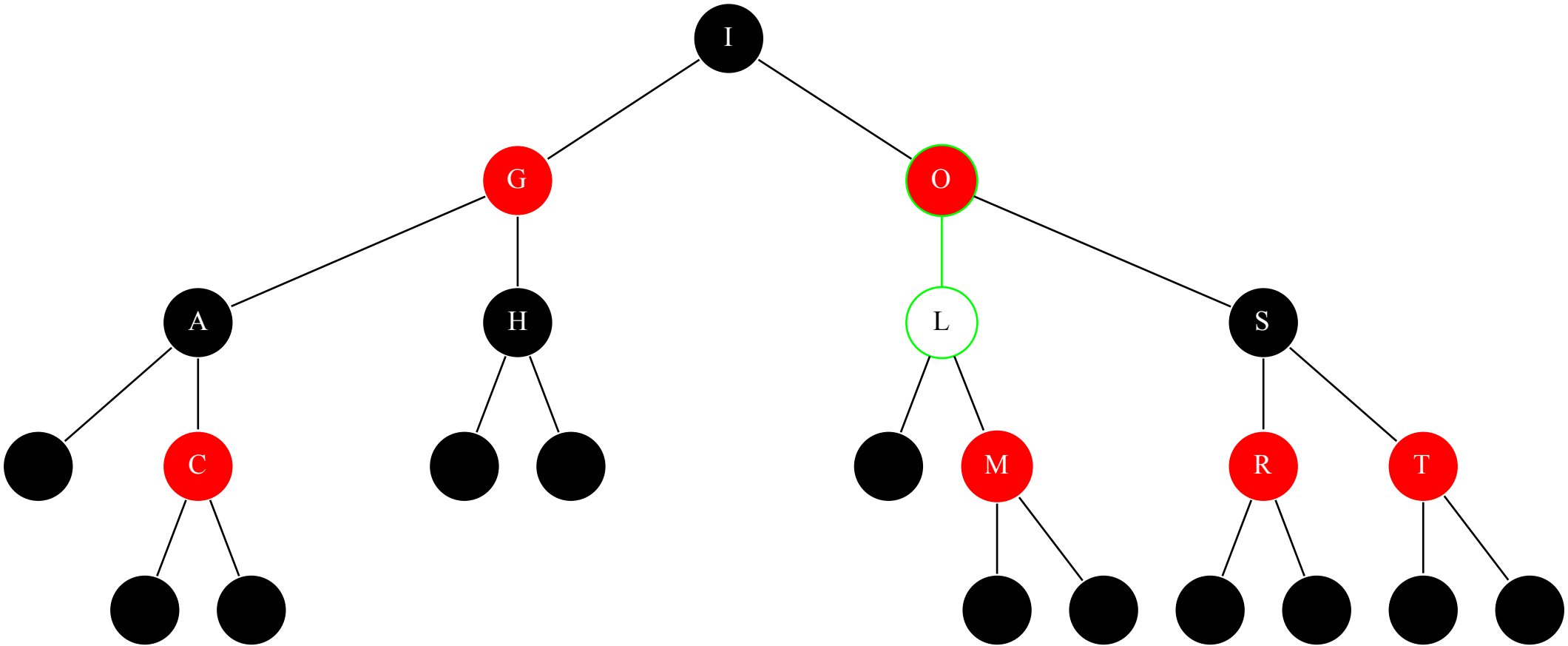
We are adding a new node with key 'M' to the tree.
The first step is to find a leaf (Nill), which the new node will replace.
We begin by examining the root (key 'I').



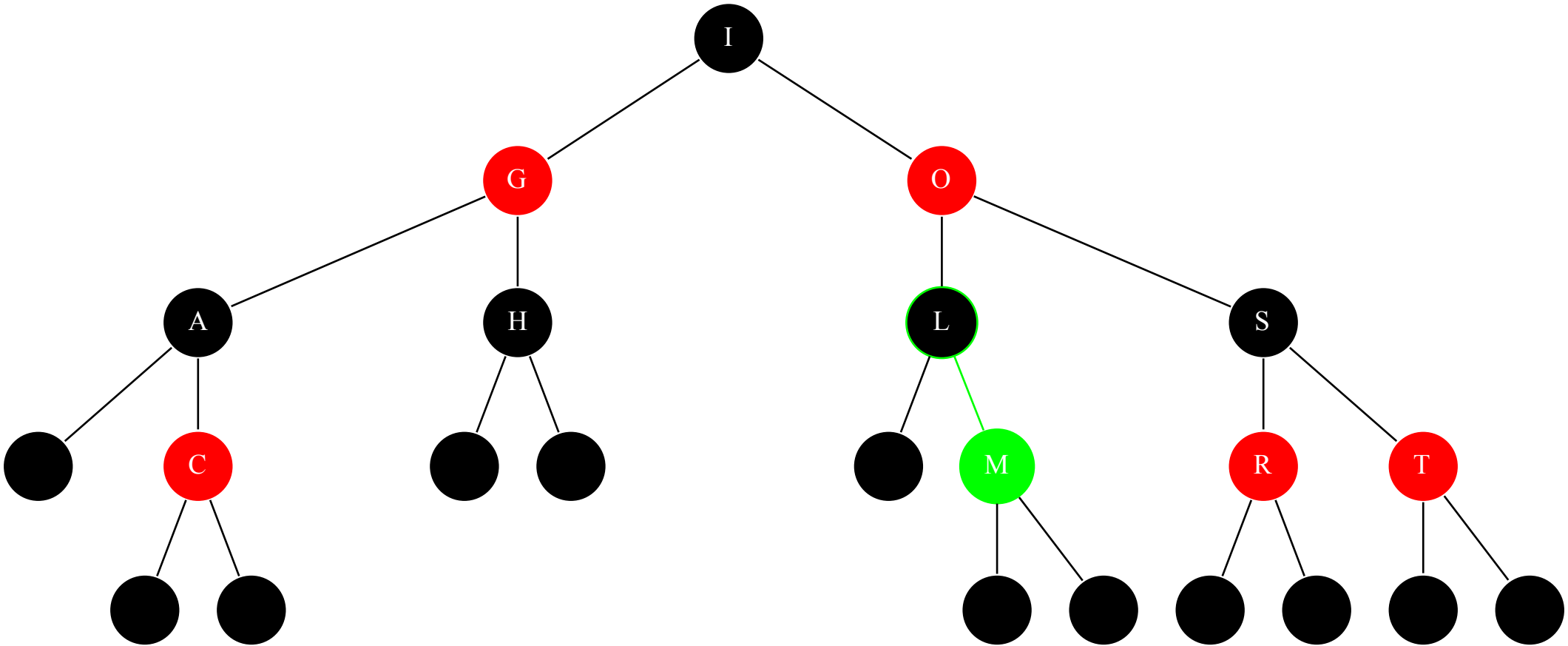
We are adding a new node with key 'M' to the tree.
By examining the node with key 'I' we have arrived at the node with key 'O'.



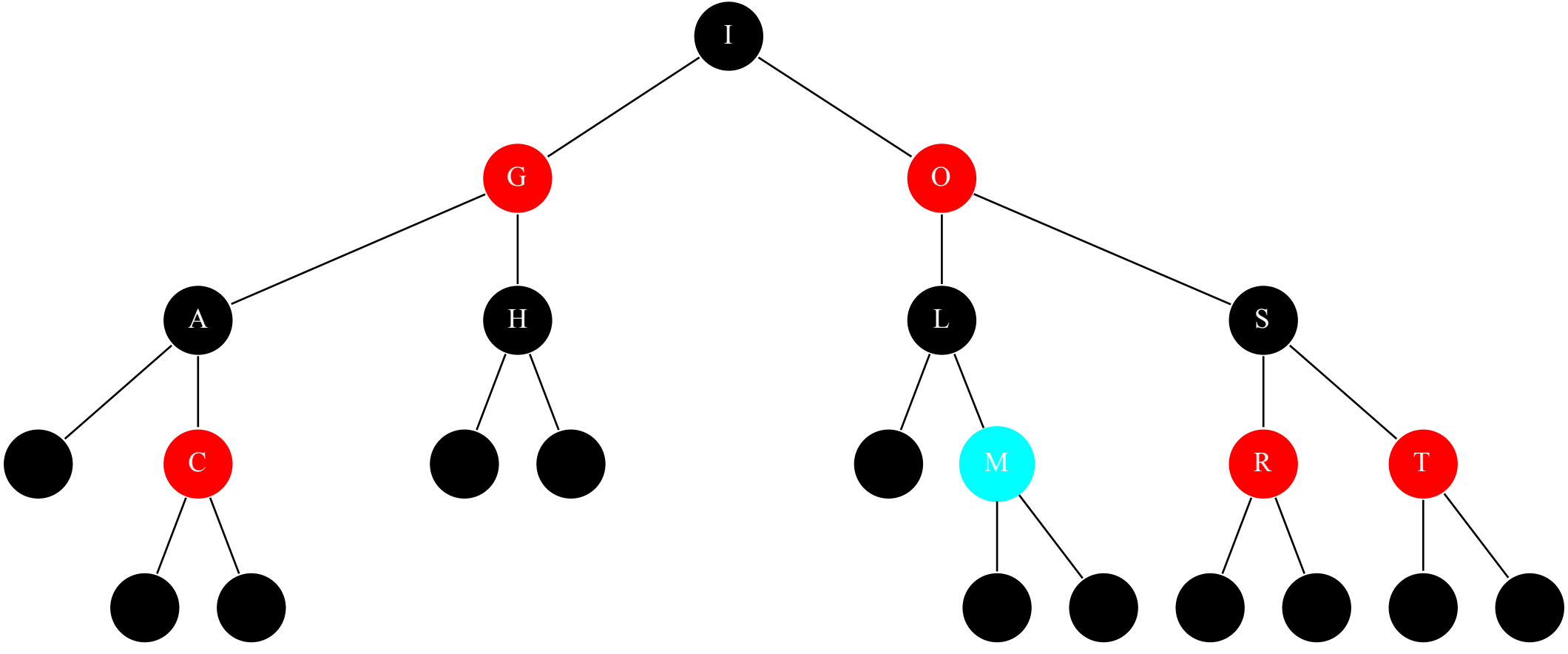
We are adding a new node with key 'M' to the tree.
By examining the node with key 'O' we have arrived at the node with key 'L'.



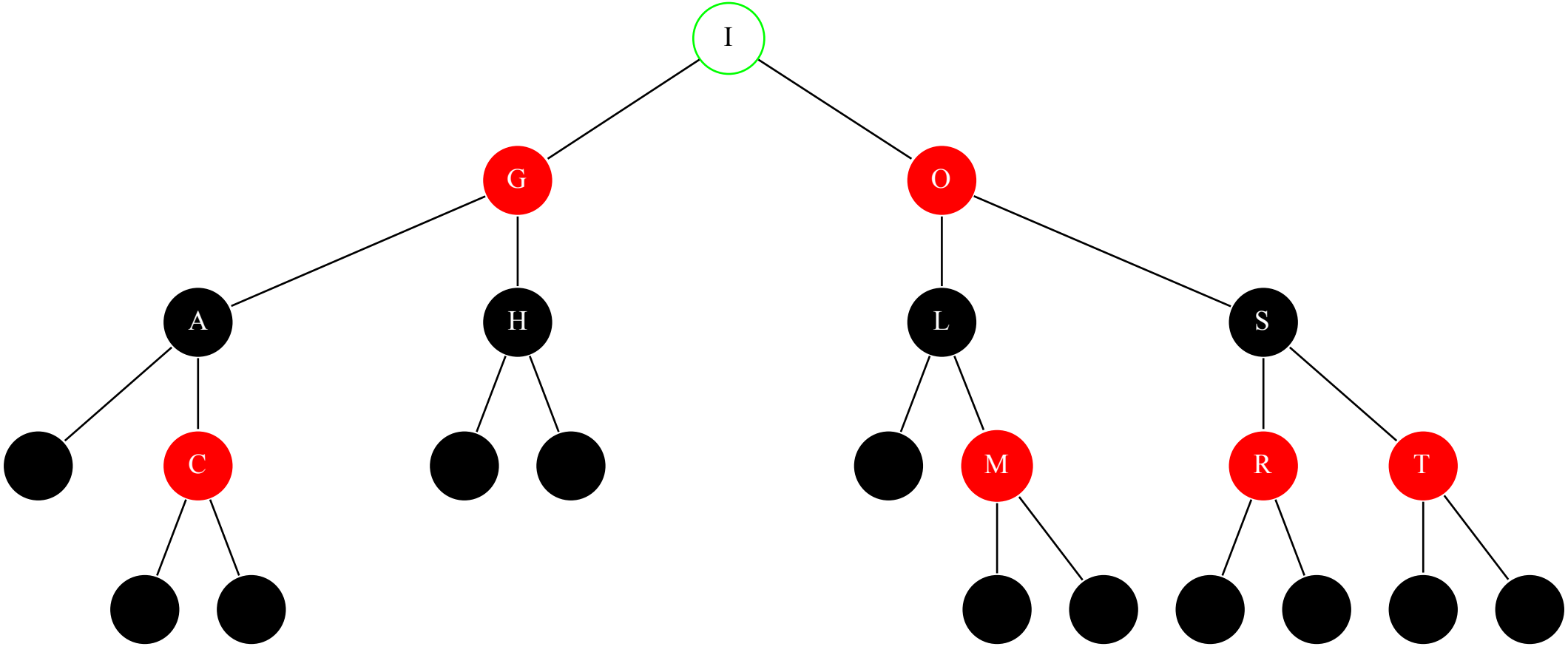
We are adding a new node with key 'M' to the tree.
By examining the node with key 'L' we have arrived at the node with key 'M'.



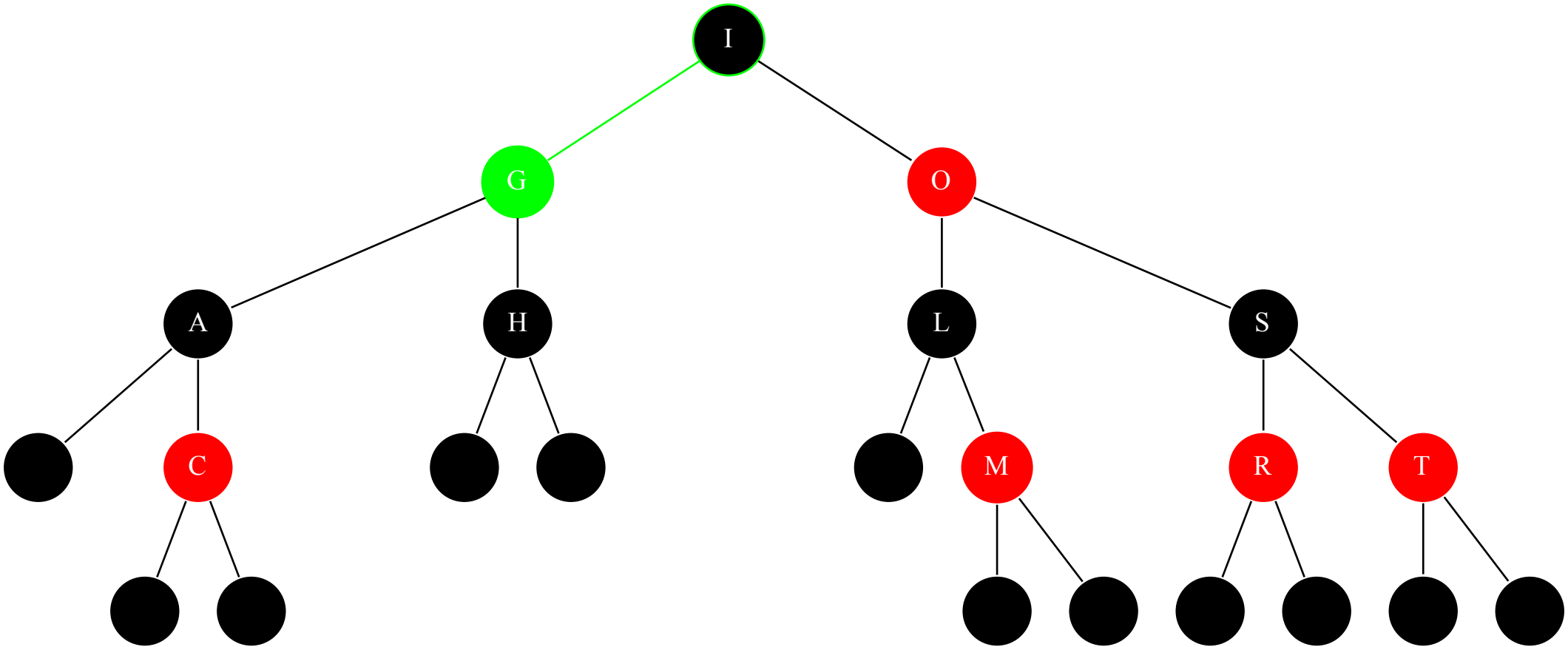
The node with key 'M' is already in the tree, so we terminate the insertion.



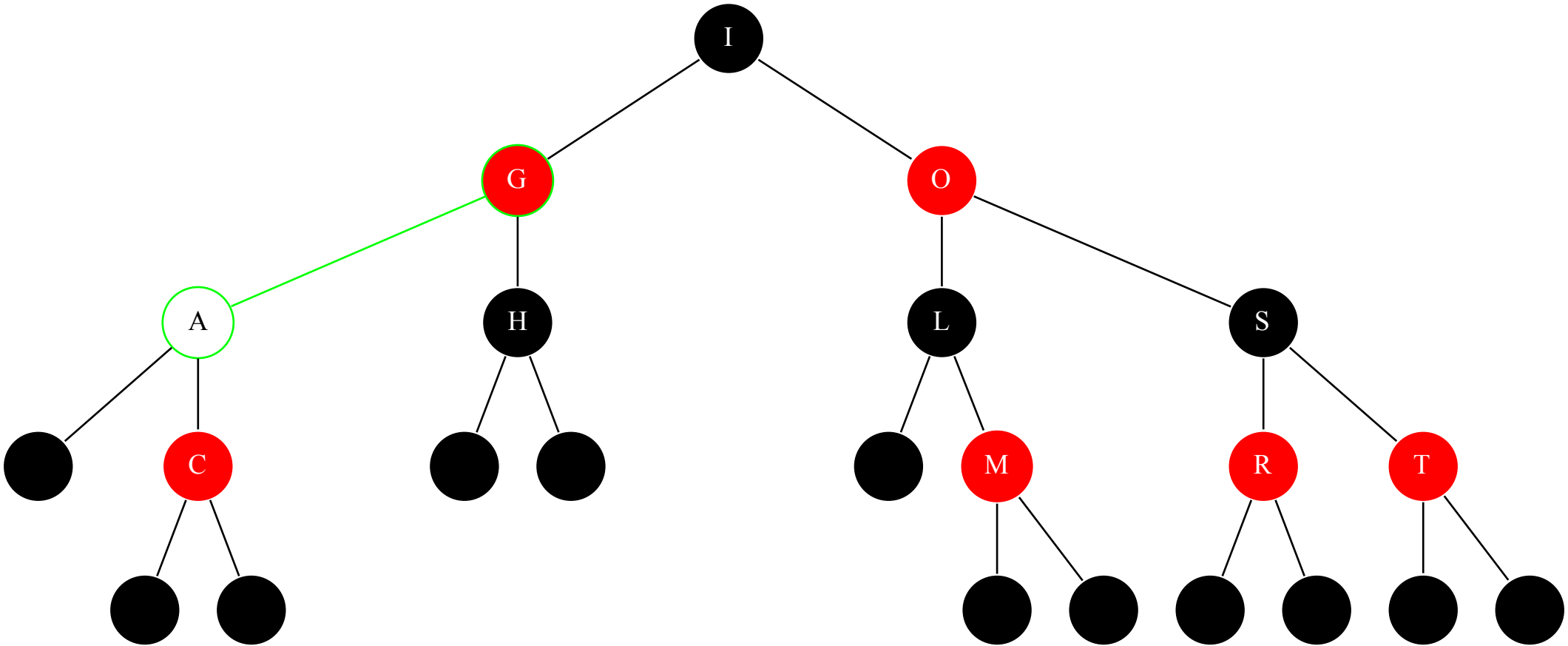
We are adding a new node with key 'B' to the tree.
The first step is to find a leaf (Nill), which the new node will replace.
We begin by examining the root (key 'I').



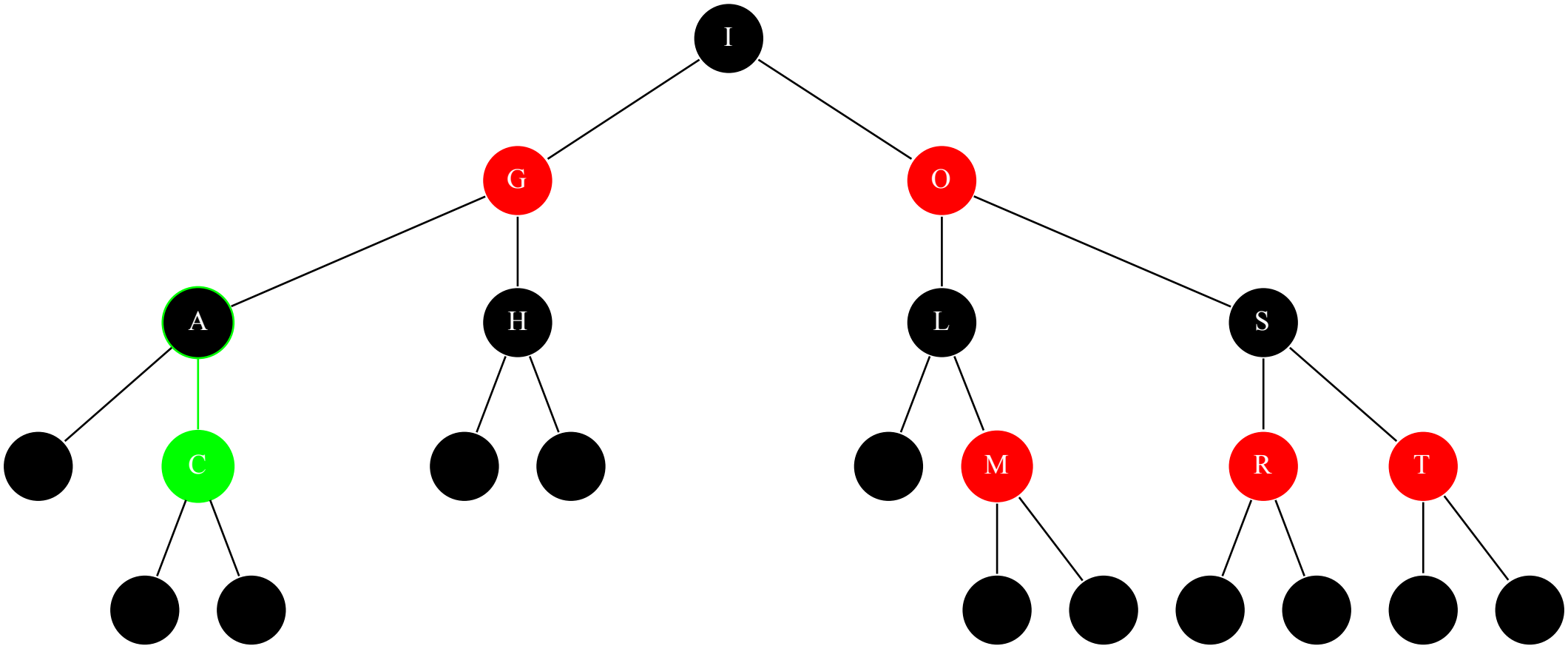
We are adding a new node with key 'B' to the tree.
By examining the node with key 'I' we have arrived at the node with key 'G'.



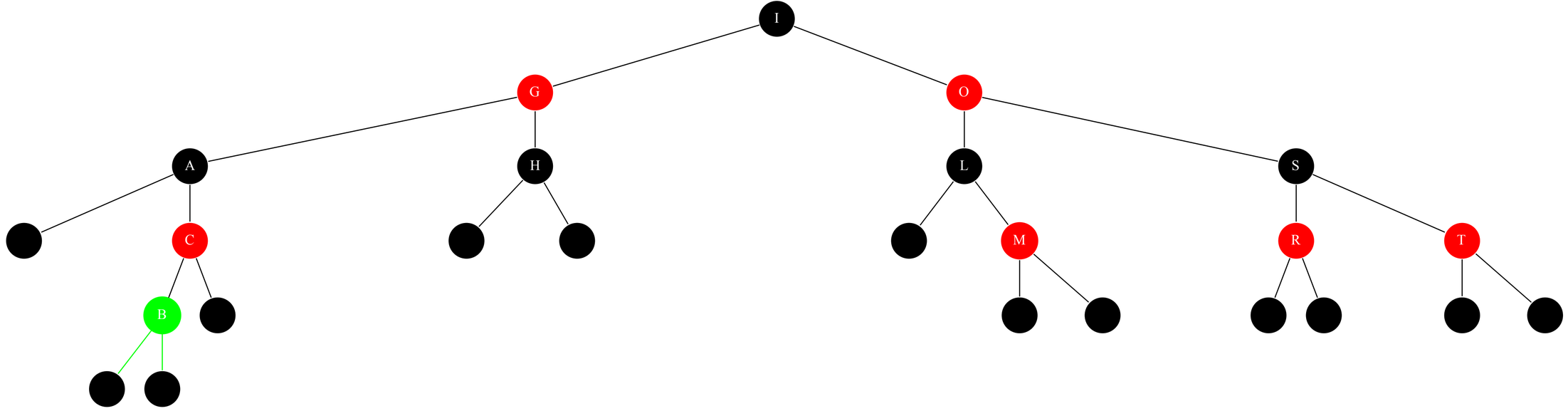
We are adding a new node with key 'B' to the tree.
By examining the node with key 'G' we have arrived at the node with key 'A'.



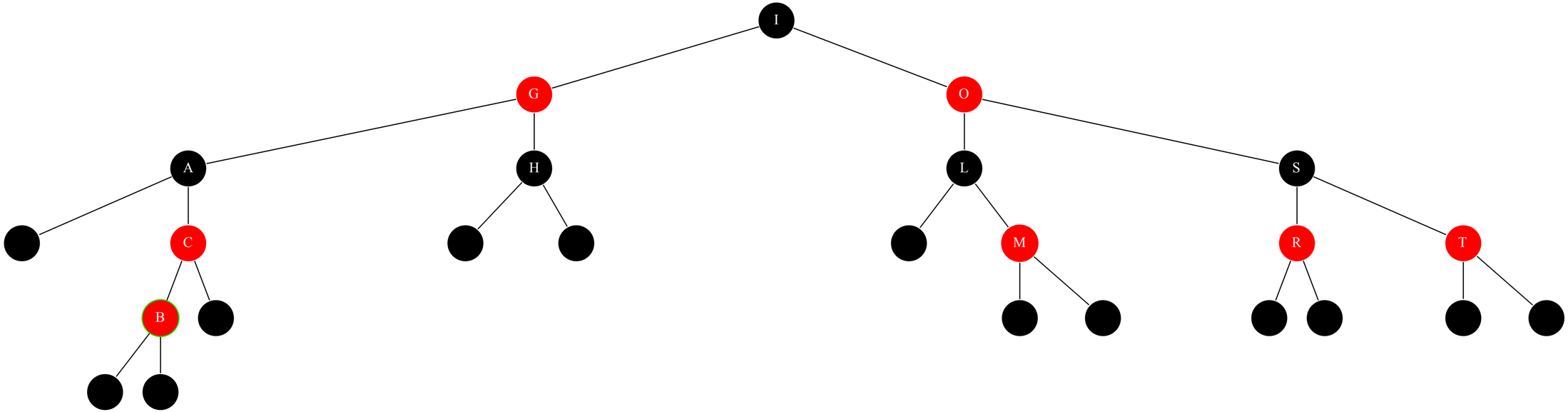
We are adding a new node with key 'B' to the tree.
By examining the node with key 'A' we have arrived at the node with key 'C'.



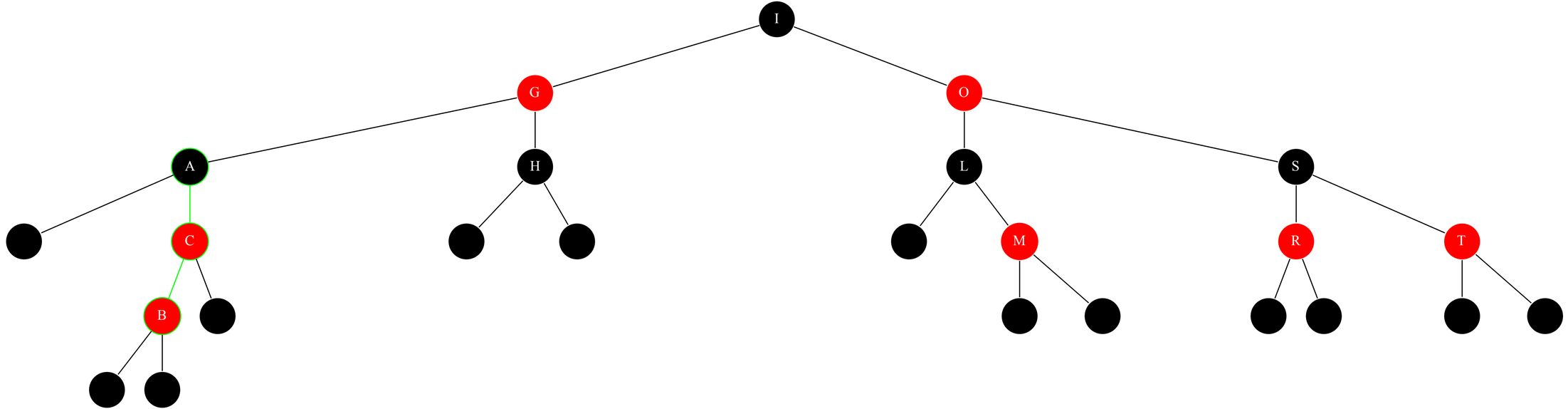
Since the key of the new node ('B') is smaller than the one of its new parent ('C'), we add it to the left of the parent.



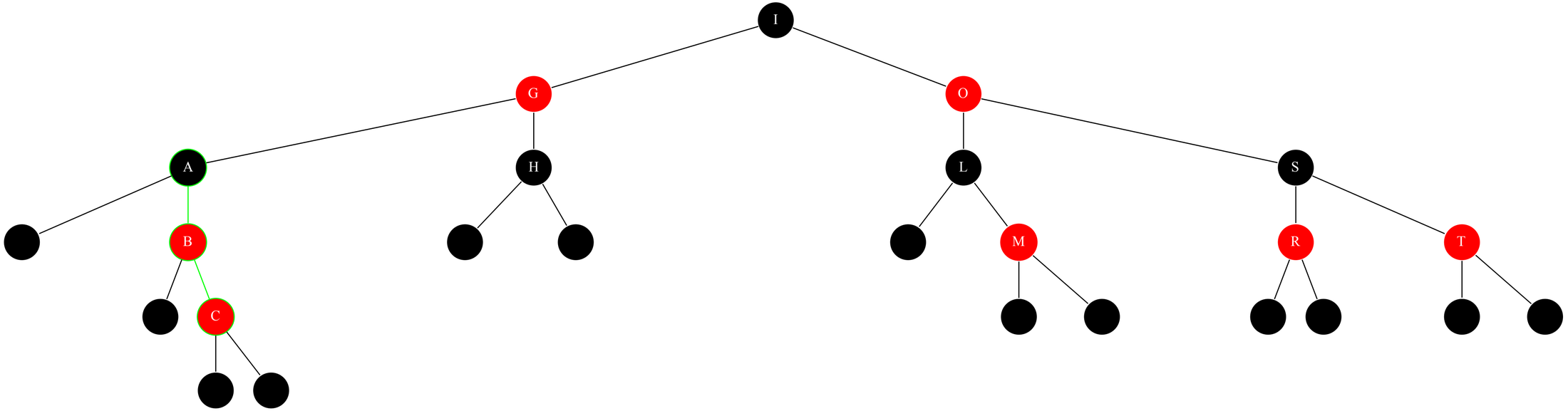
Now that we have positioned the new node ('B'), we need to ensure the correctness of the red-black tree.
Since we assumed the new node to be coloured red, we can only violate the 2. and 4. rules of the tree.
Now we set out to correct any discrepancies, by evaluating the colour of the newly added node's parent as well as its uncle, and acting accordingly.



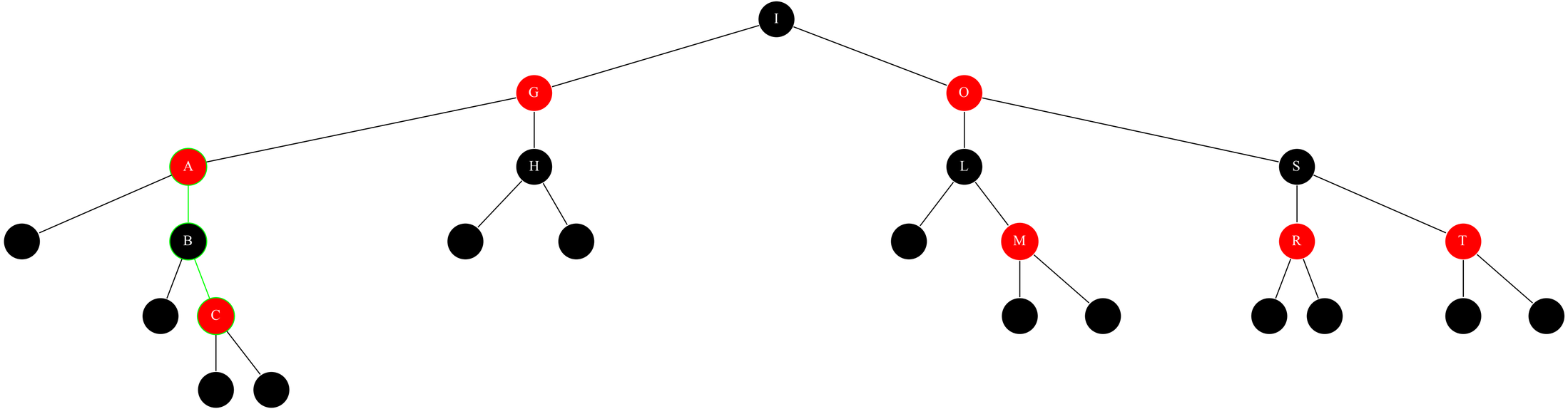
Since the uncle ('Nill') is black, and the parent ('C') is red, we cannot just perform a recolouring, but have to perform a rotation.
This rotation is needed because the currently examined node ('B'), its father ('C') and grandfather ('A') are not 'in-line'.



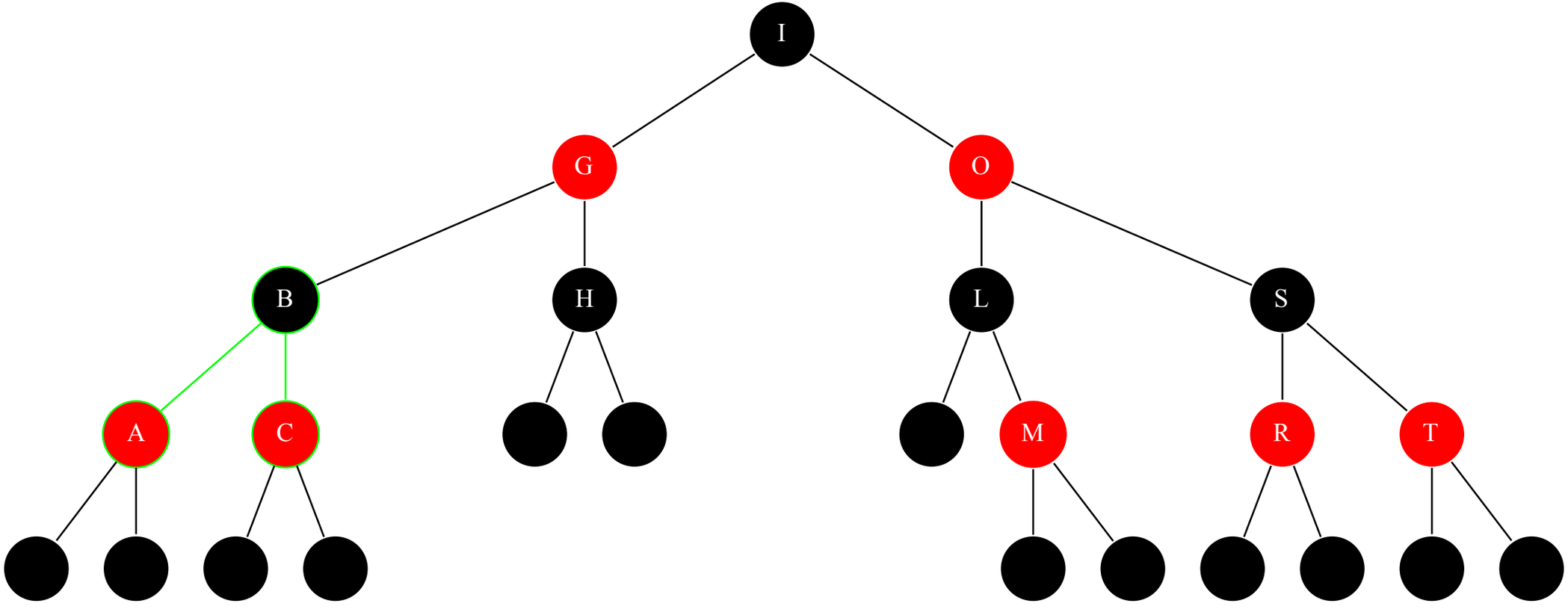
After we performed the before-mentioned rotation, the currently examined node ('C'), its parent ('B') and grandparent ('A') are now 'in-line'.



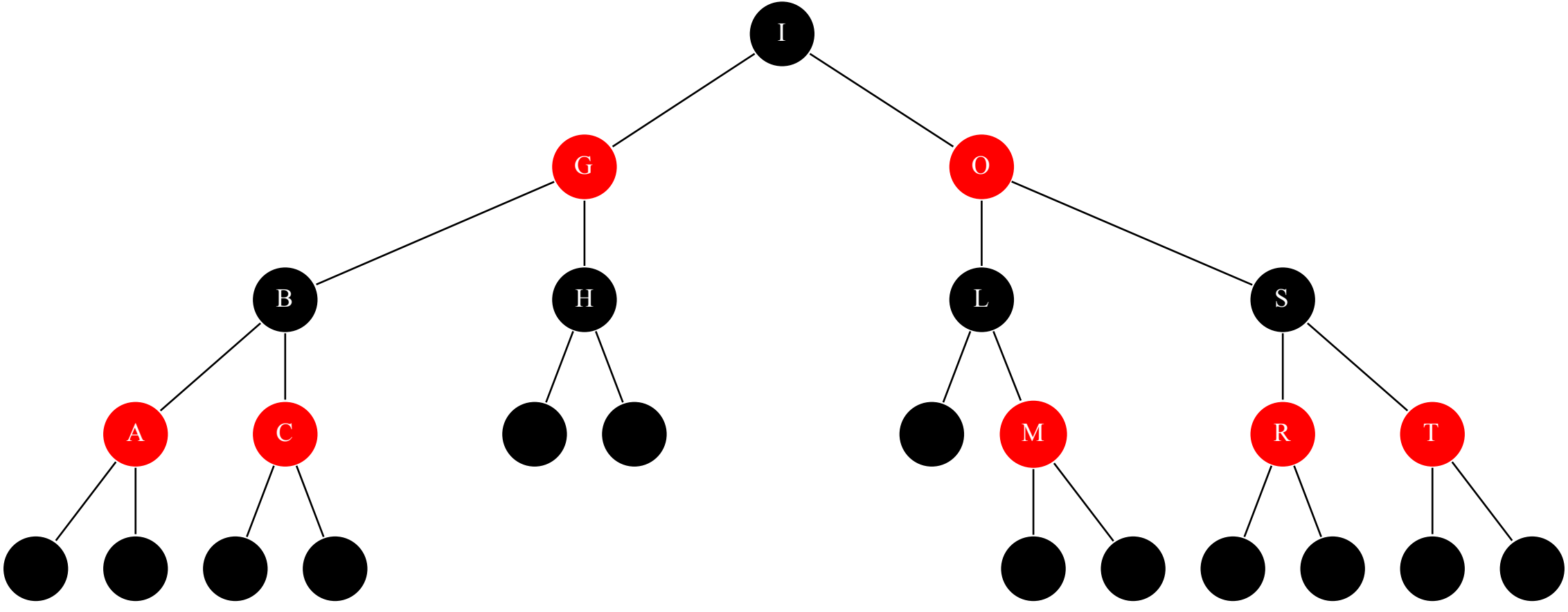
We have now made the currently examined node ('C'), its parent ('B') and grandparent ('A') align.
We now had to recolour them, so that when we rotate them in the next step, we will get a correct tree.



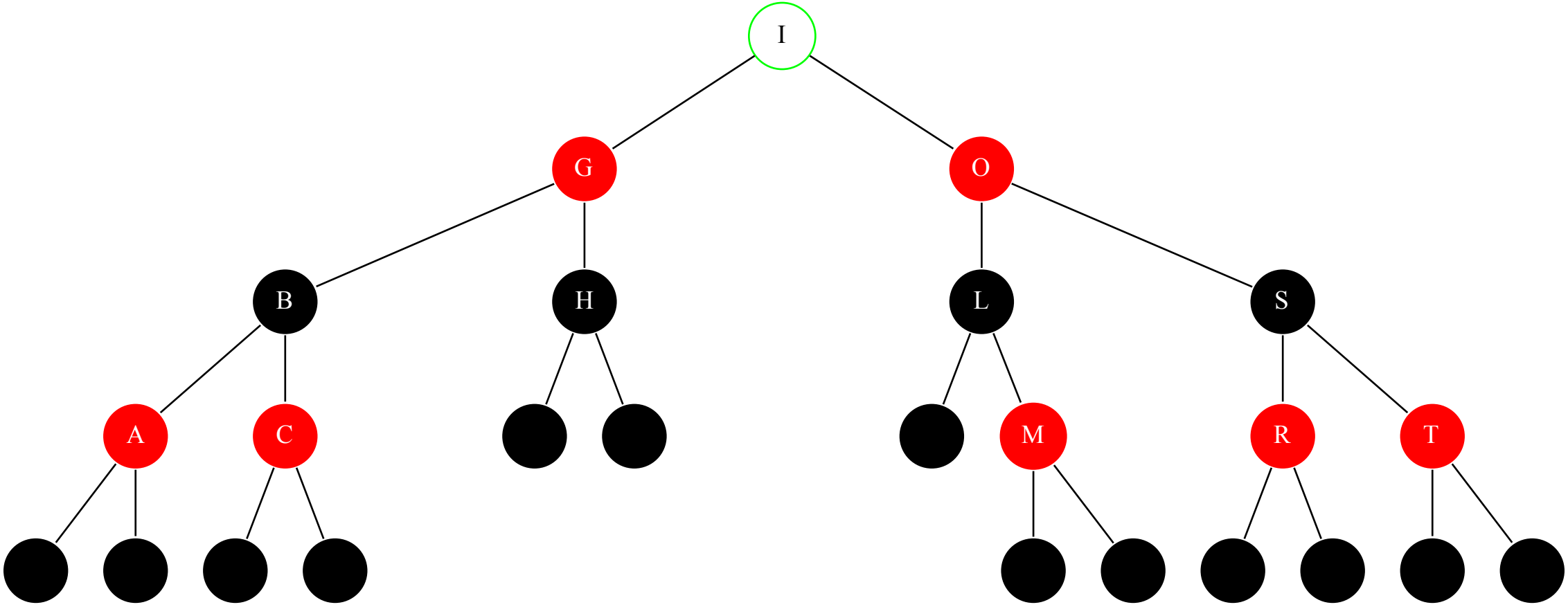
Since the currently examined node ('C'), its parent ('B') and grandparent ('A') are 'in-line' and properly coloured now, just one more rotation gives a correct red-black tree.



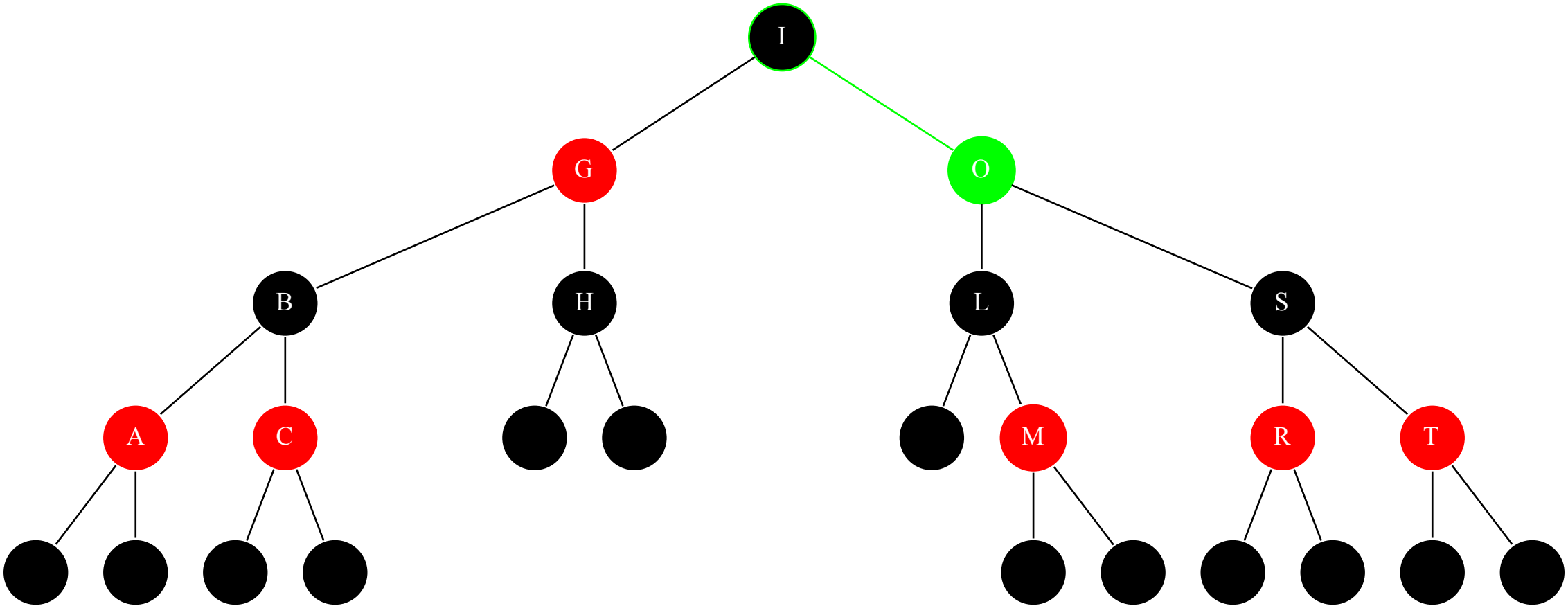
Since all of the required red-black properties are satisfied, we have arrived to a correctly formed red-black tree.



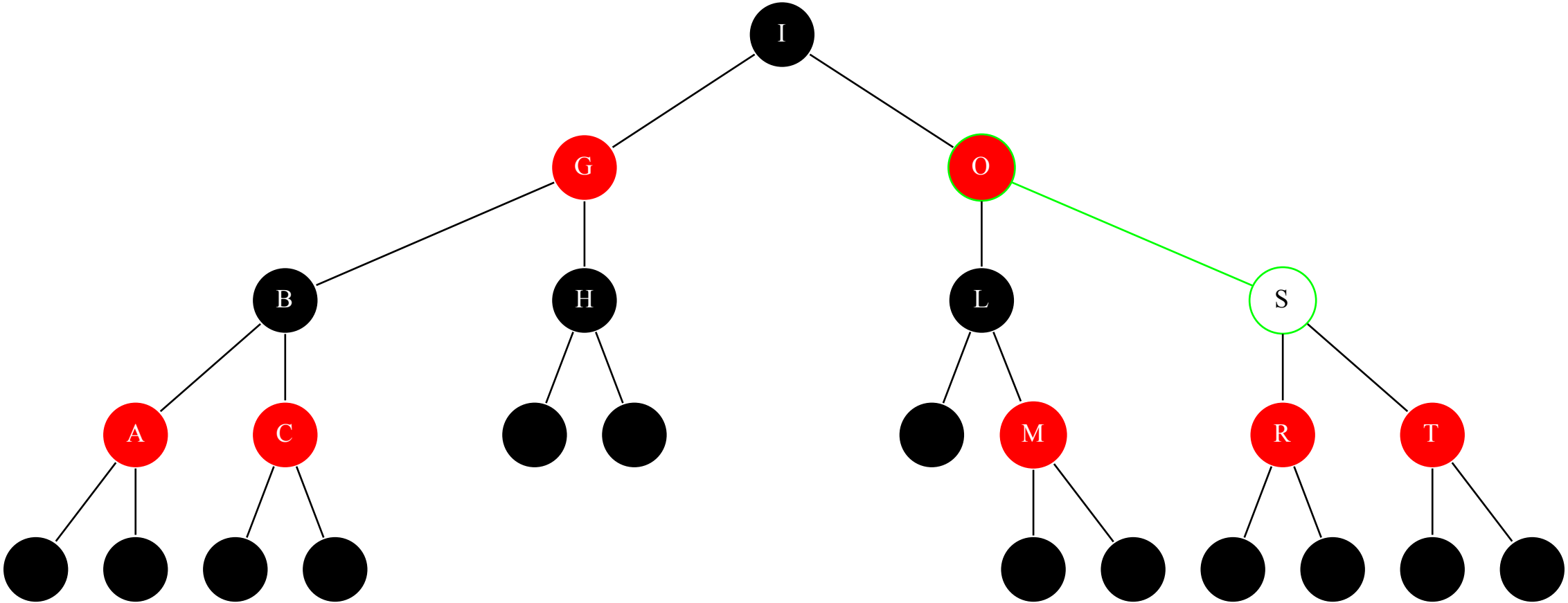
We are adding a new node with key 'R' to the tree.
The first step is to find a leaf (Nill), which the new node will replace.
We begin by examining the root (key 'I').



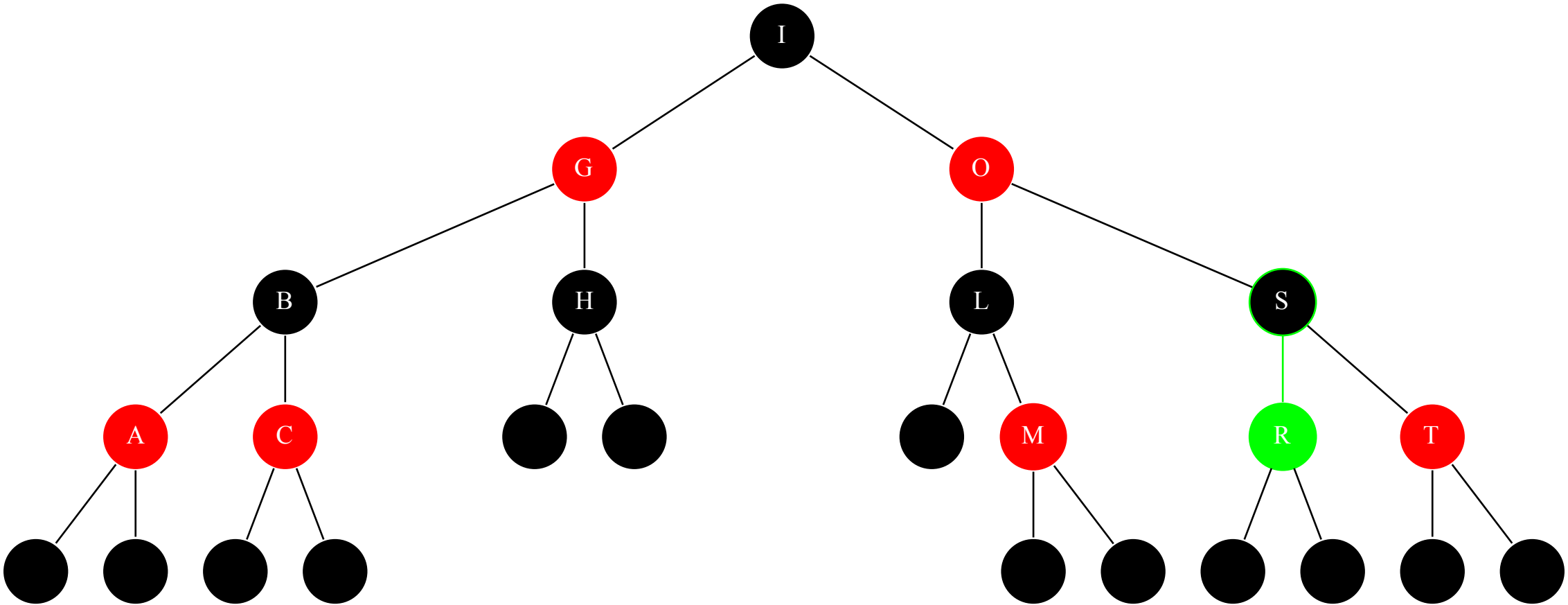
We are adding a new node with key 'R' to the tree.
By examining the node with key 'I' we have arrived at the node with key 'O'.



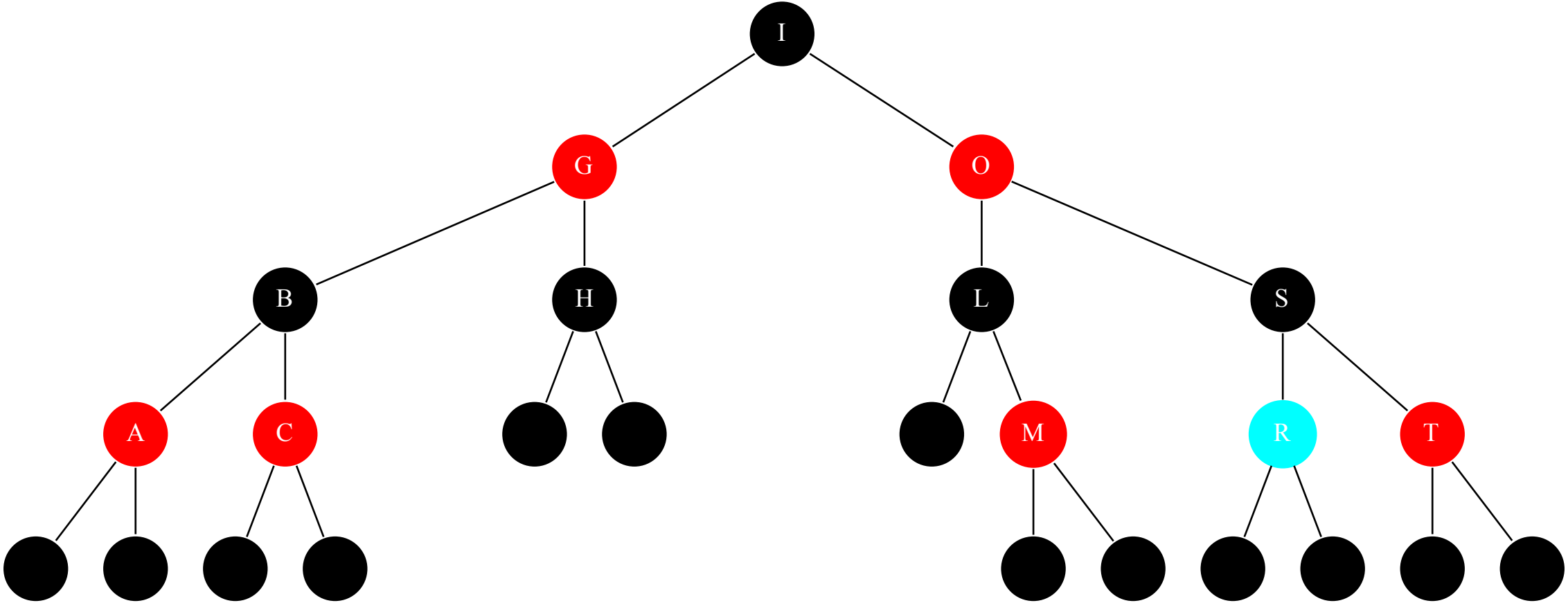
We are adding a new node with key 'R' to the tree.
By examining the node with key 'O' we have arrived at the node with key 'S'.



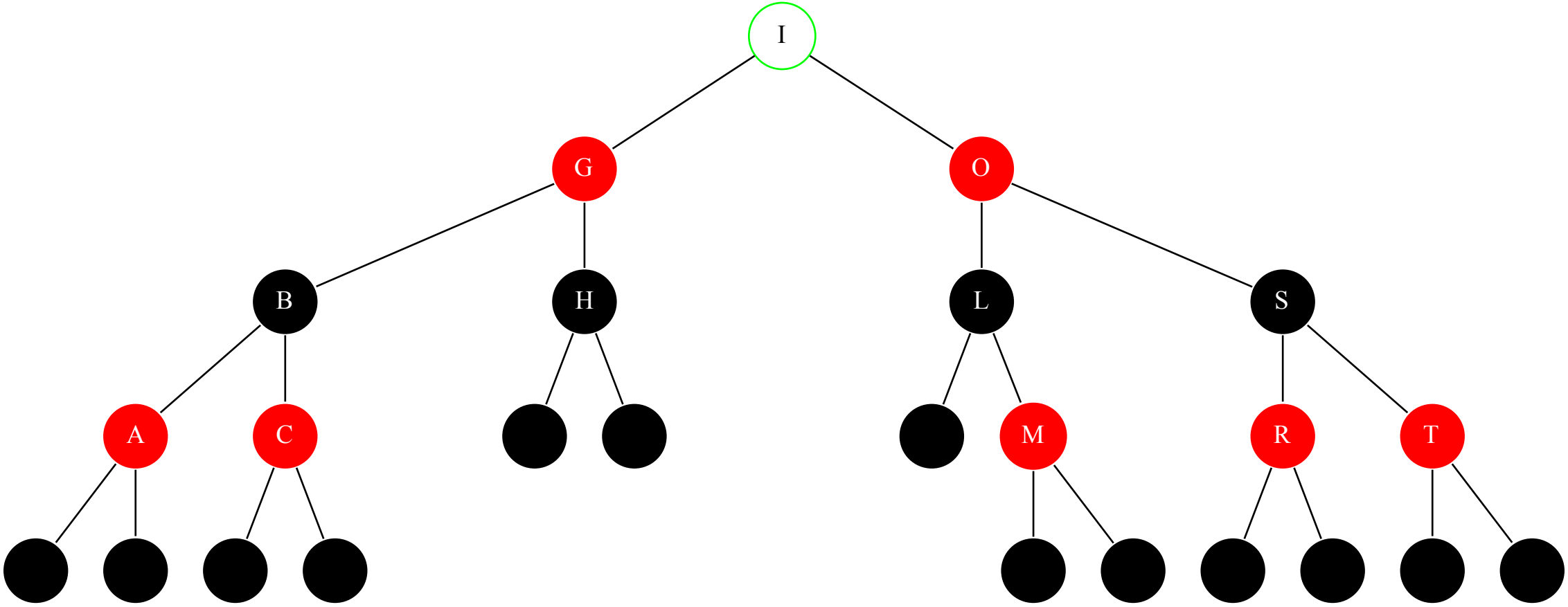
We are adding a new node with key 'R' to the tree.
By examining the node with key 'S' we have arrived at the node with key 'R'.



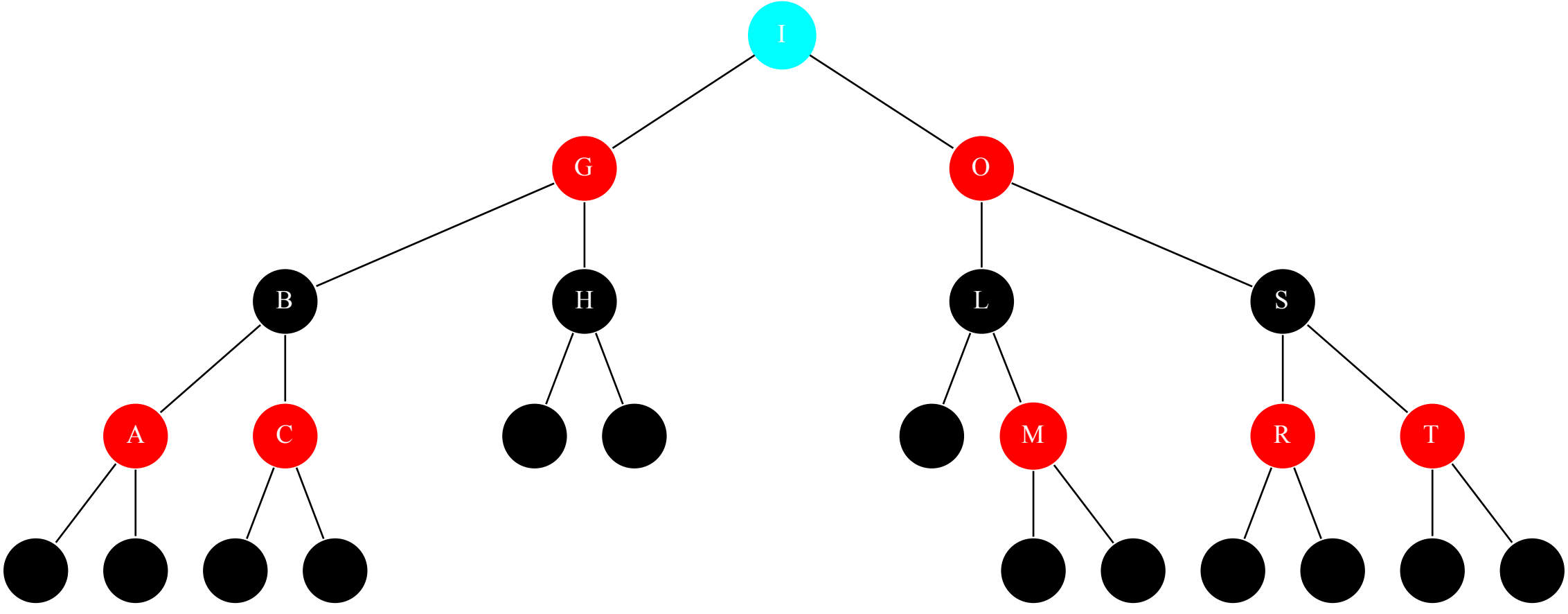
The node with key 'R' is already in the tree, so we terminate the insertion.



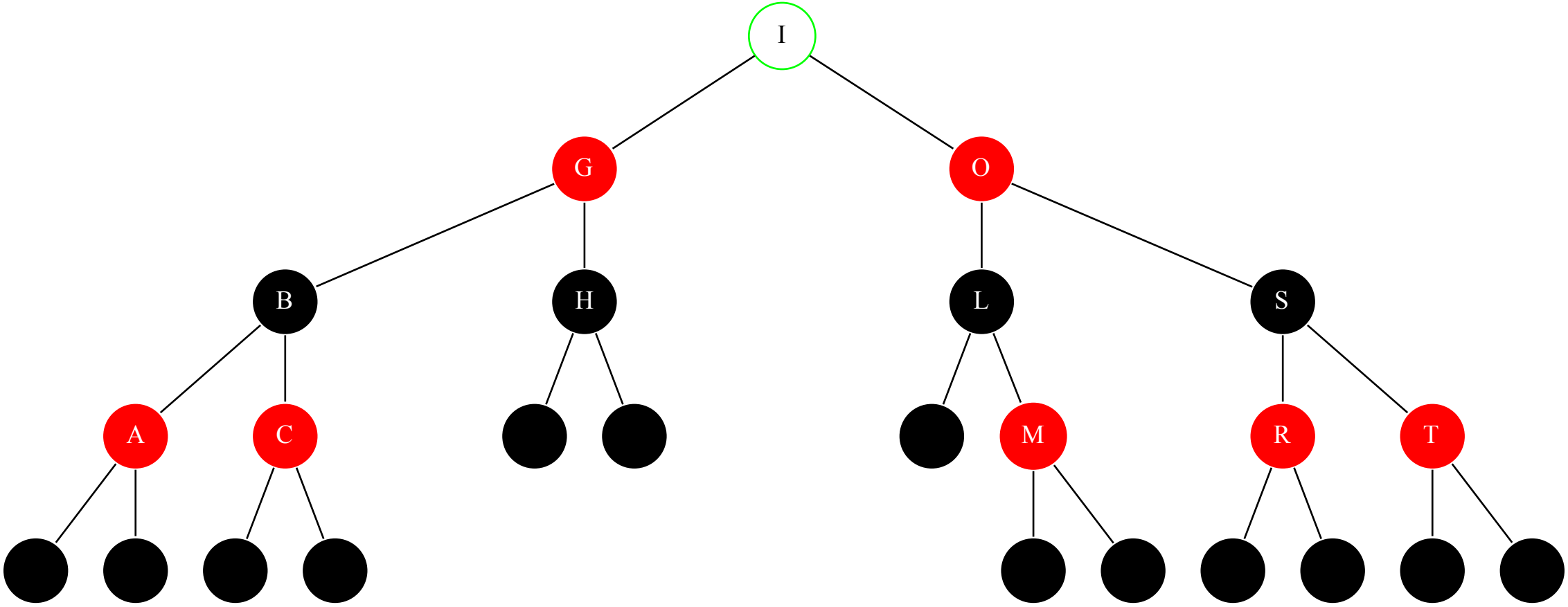
We are adding a new node with key 'I' to the tree.
The first step is to find a leaf (Nill), which the new node will replace.
We begin by examining the root (key 'I').



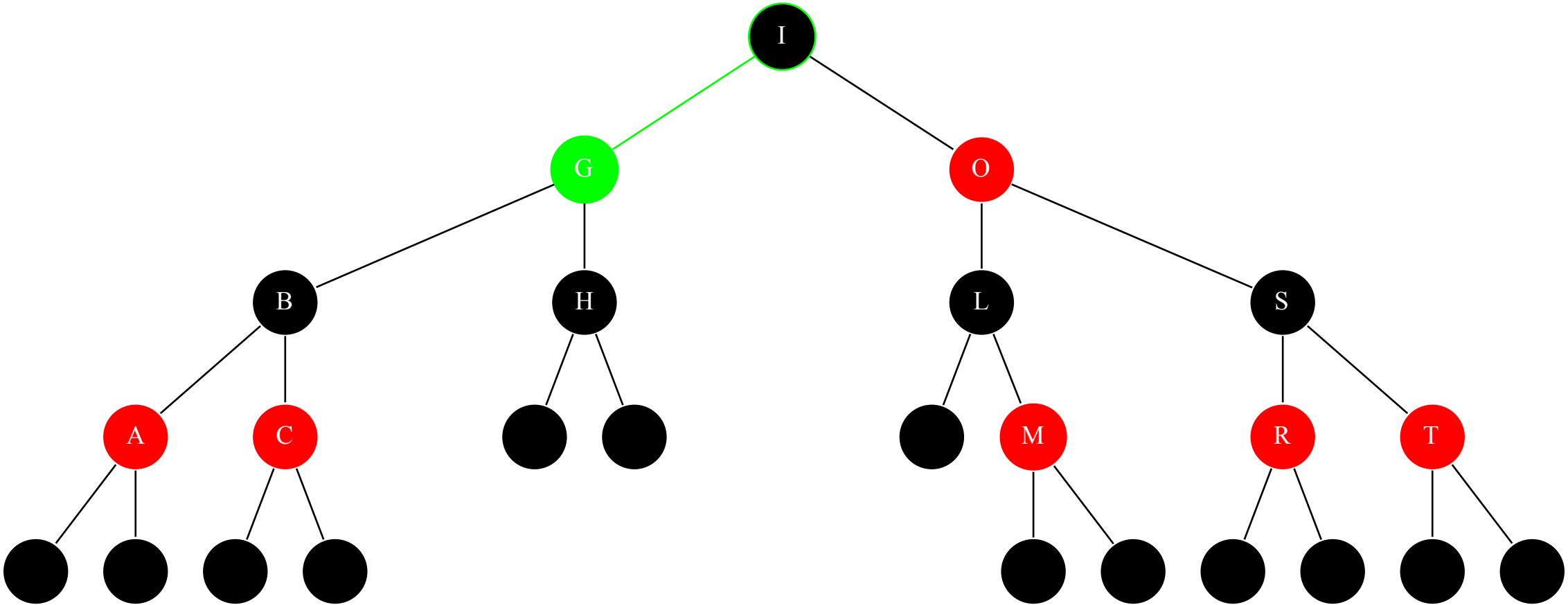
The node with key 'I' is already in the tree, so we terminate the insertion.



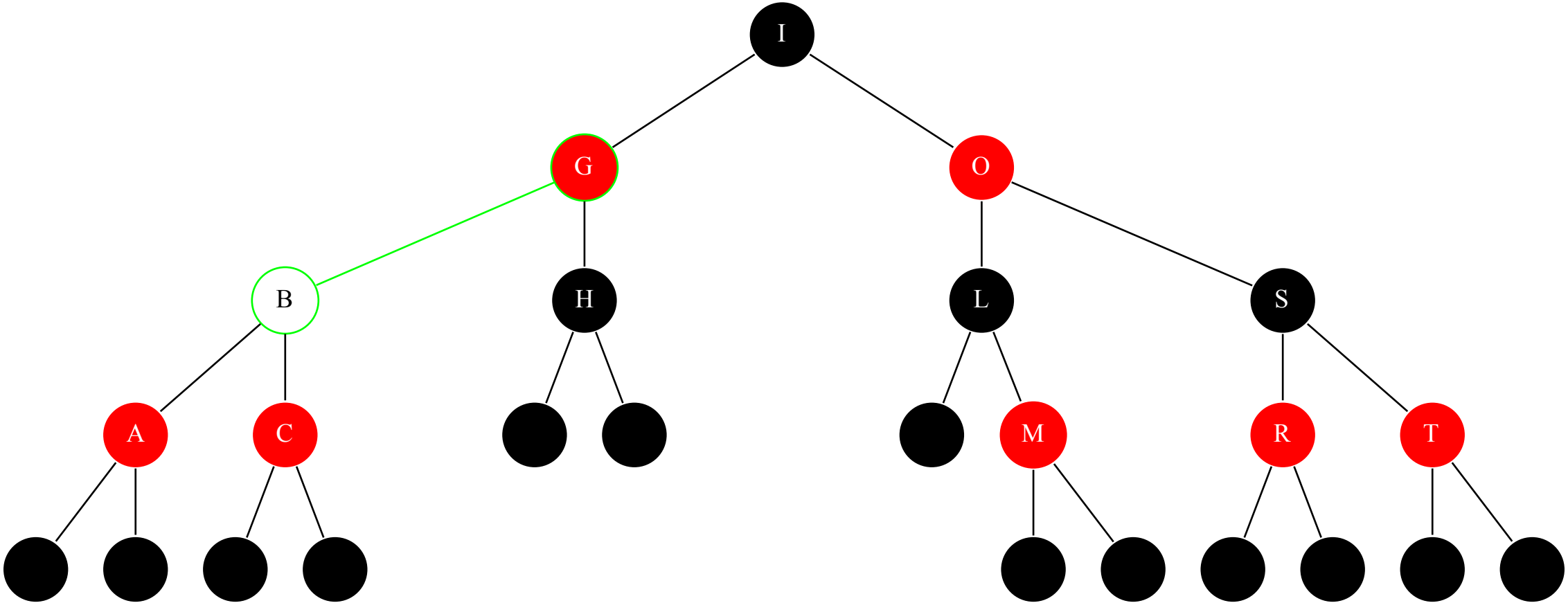
We are adding a new node with key 'D' to the tree.
The first step is to find a leaf (Nill), which the new node will replace.
We begin by examining the root (key 'I').



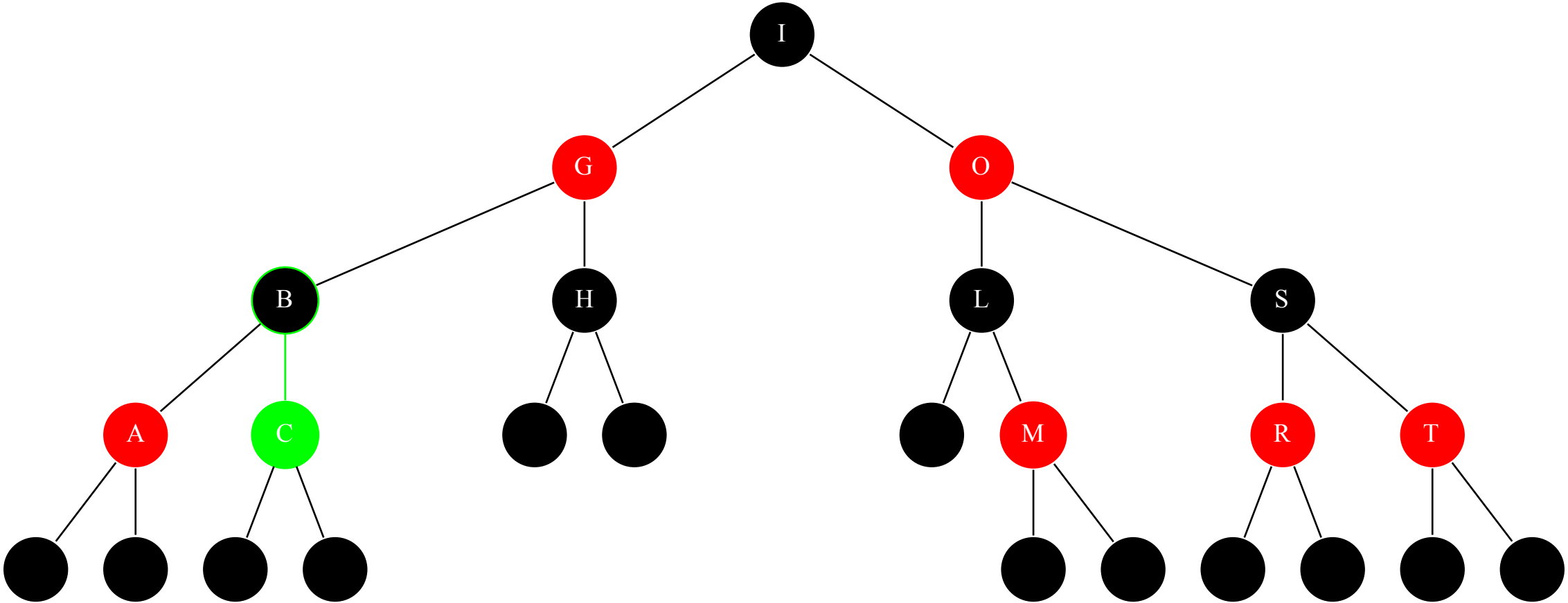
We are adding a new node with key 'D' to the tree.
By examining the node with key 'I' we have arrived at the node with key 'G'.



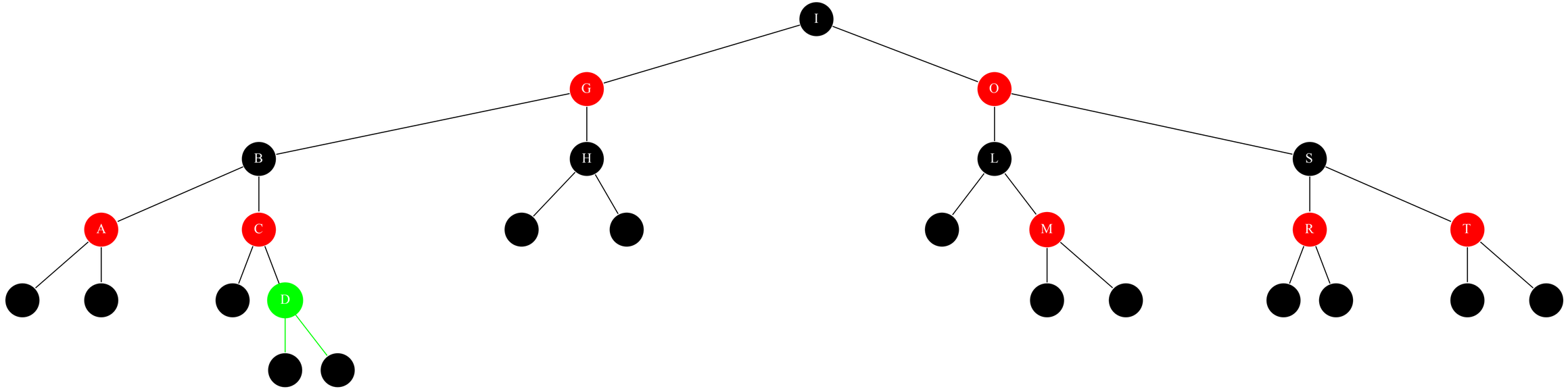
We are adding a new node with key 'D' to the tree.
By examining the node with key 'G' we have arrived at the node with key 'B'.



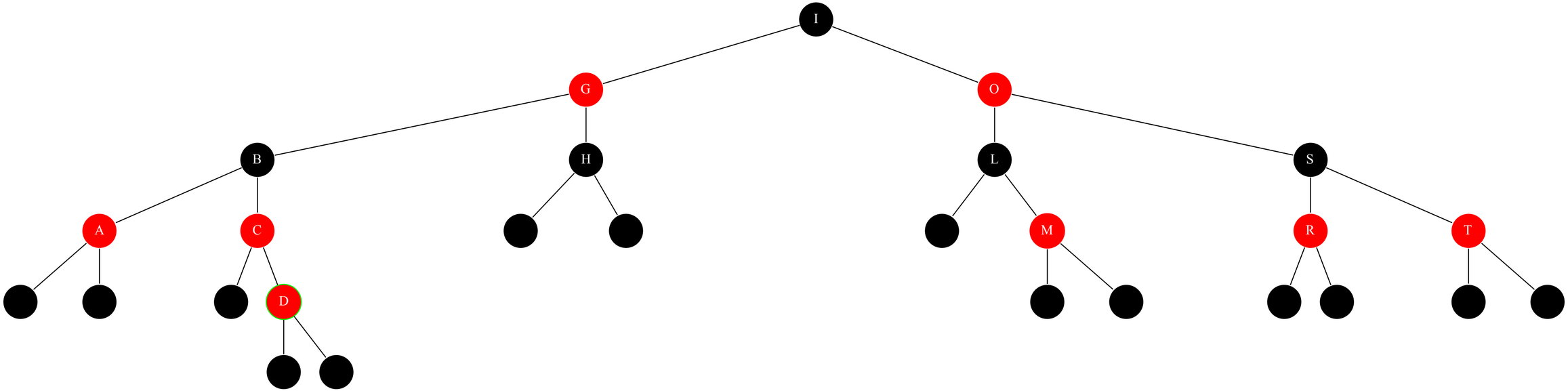
We are adding a new node with key 'D' to the tree.
By examining the node with key 'B' we have arrived at the node with key 'C'.



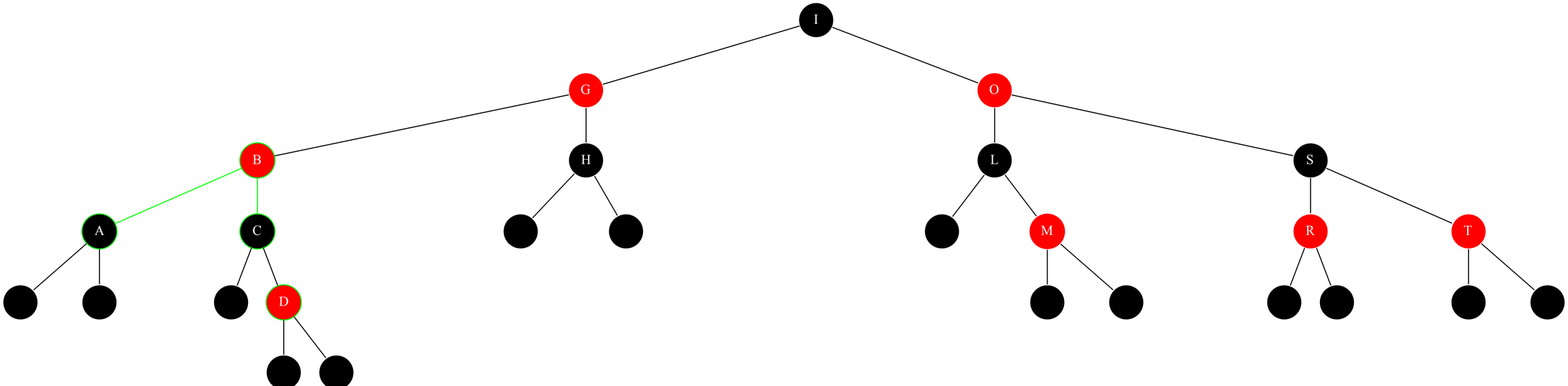
Since the key of the new node ('D') is greater than the one of its new parent ('C'), we add it to the right of the parent.



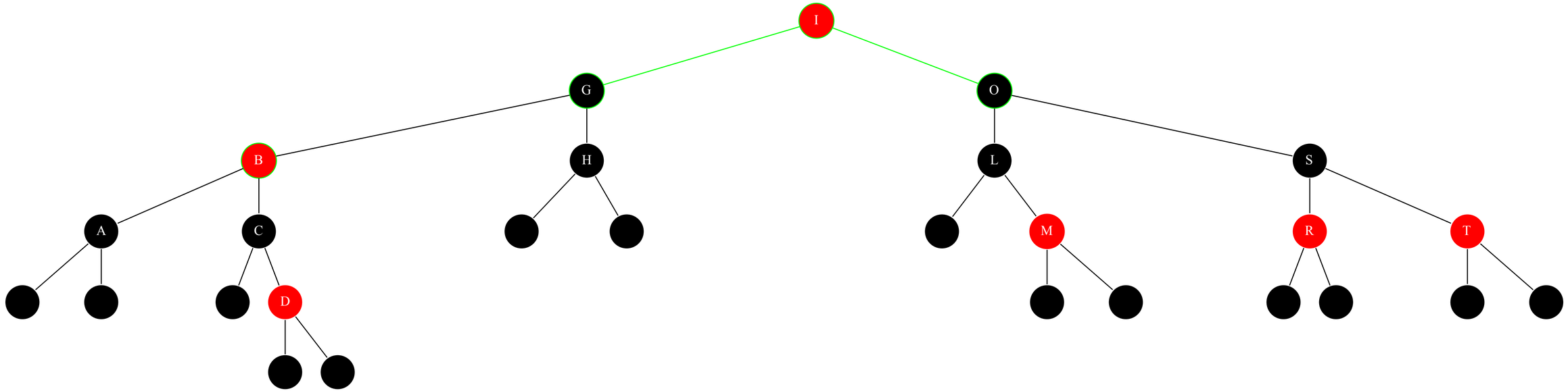
Now that we have positioned the new node ('D'), we need to ensure the correctness of the red-black tree.
Since we assumed the new node to be coloured red, we can only violate the 2. and 4. rules of the tree.
Now we set out to correct any discrepancies, by evaluating the colour of the newly added node's parent as well as its uncle, and acting accordingly.



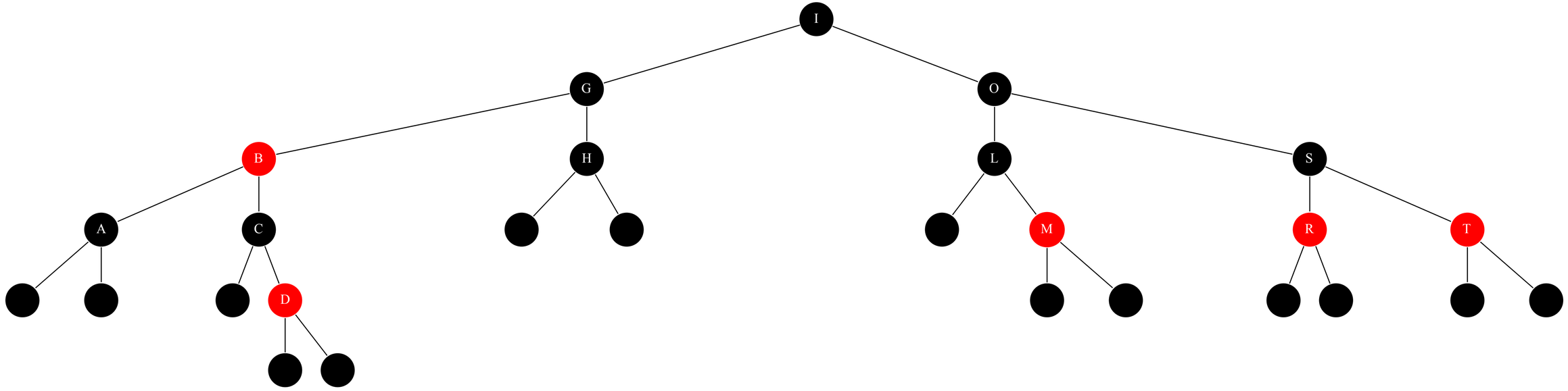
Since the uncle ('A') was red and we know that the parent ('C') was also red, we just had to perform a colour rebalancing.
This means that we coloured the currently examined nodes parent ('C') and uncle black (A), and coloured its grandfather ('B') red.
By doing this we correct the fourth rule of the red-black trees.

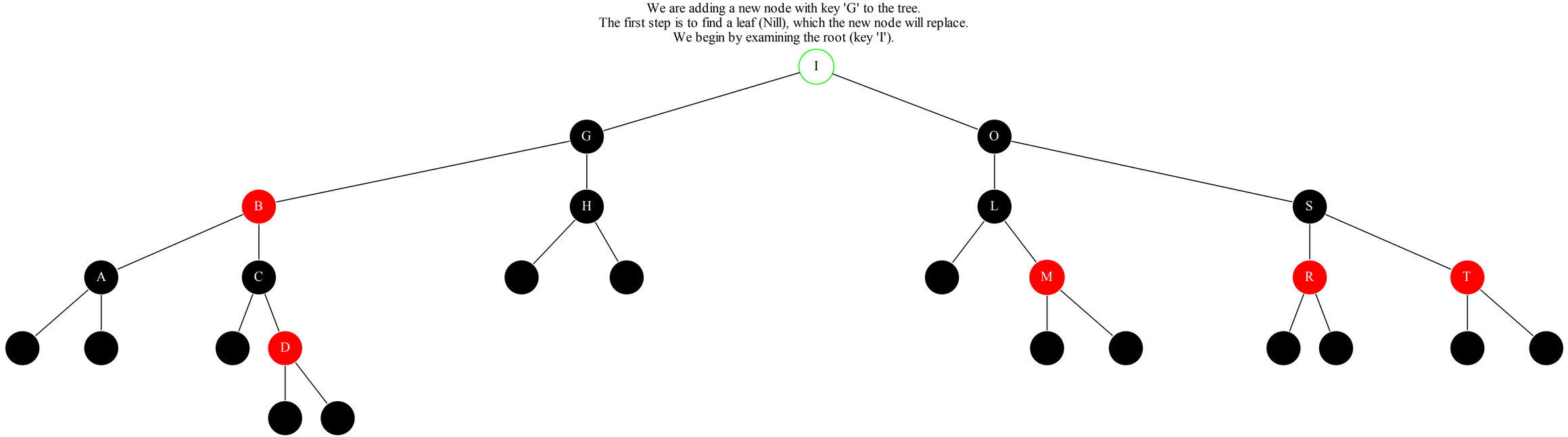


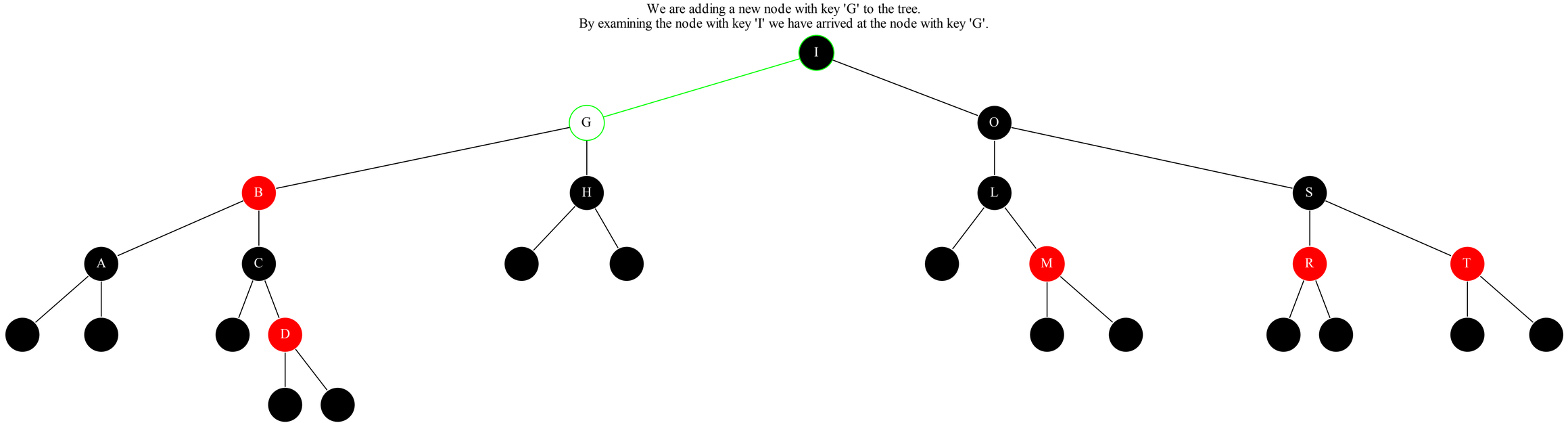
Since the uncle ('O') was red and we know that the parent ('G') was also red, we just had to perform a colour rebalancing.
This means that we coloured the currently examined nodes parent ('G') and uncle black (O), and coloured its grandfather ('I') red.
By doing this we correct the fourth rule of the red-black trees.



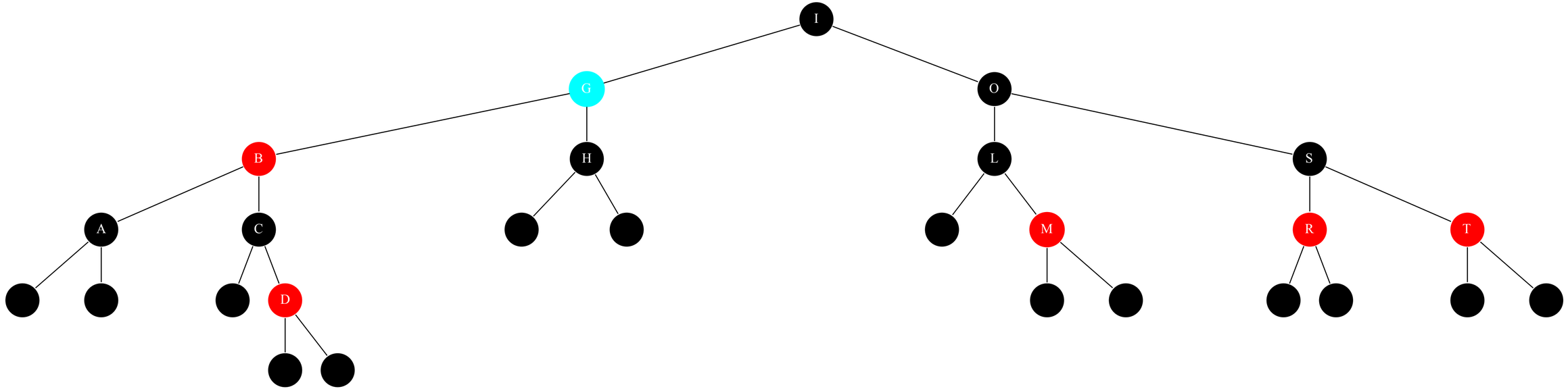
Since all of the required red-black properties are satisfied, we have arrived to a correctly formed red-black tree.

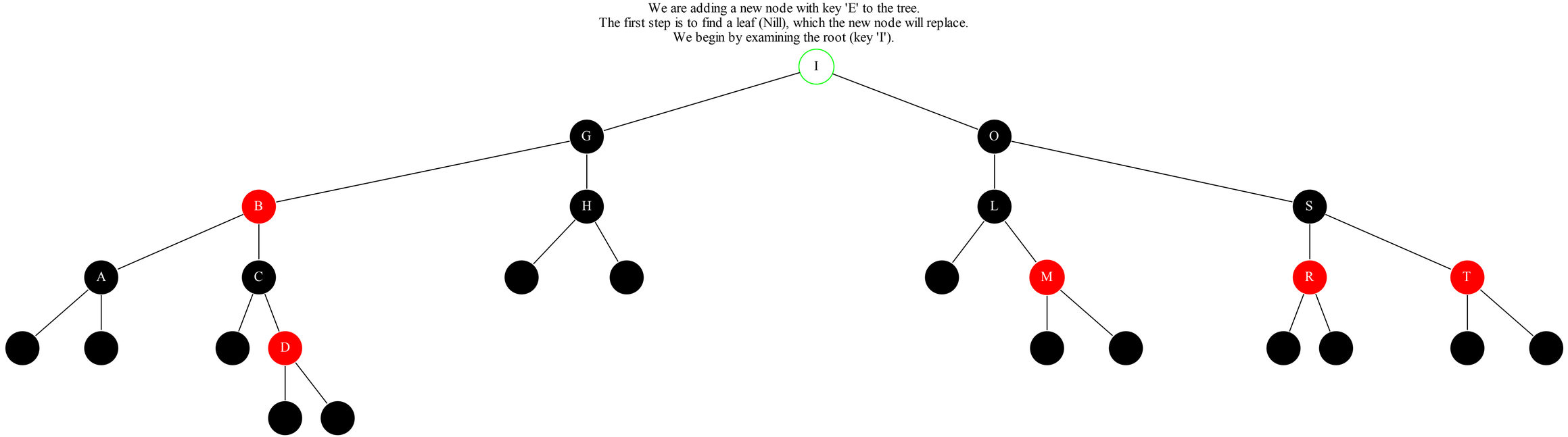


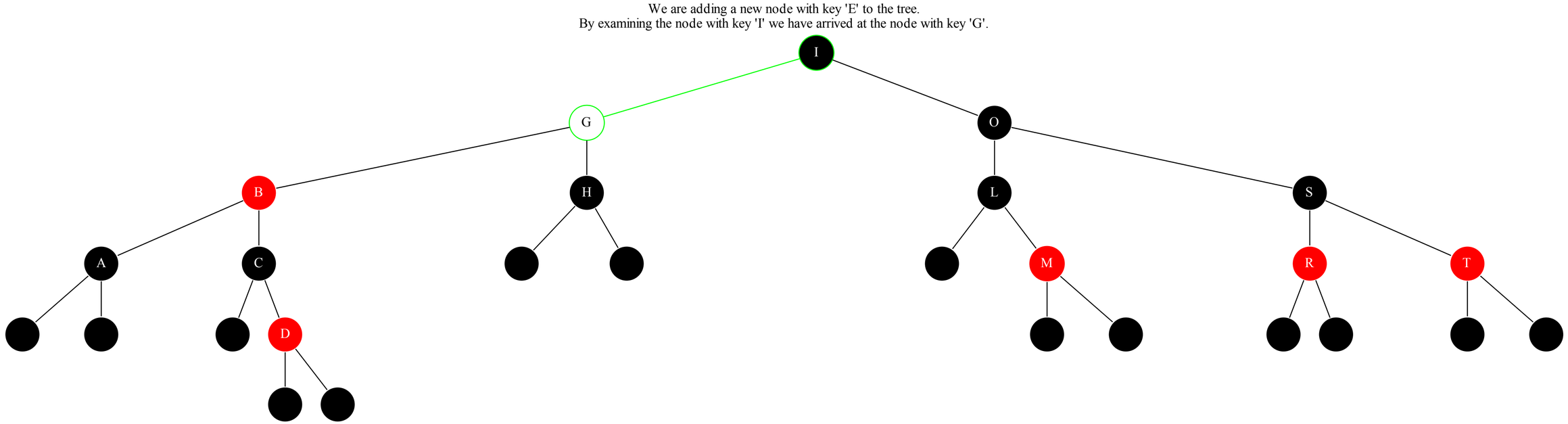

We are adding a new node with key 'G' to the tree.
The first step is to find a leaf (Nill), which the new node will replace.
We begin by examining the root (key 'I').
I
G
O
B
H
L
S
A
C
M
R
T
D

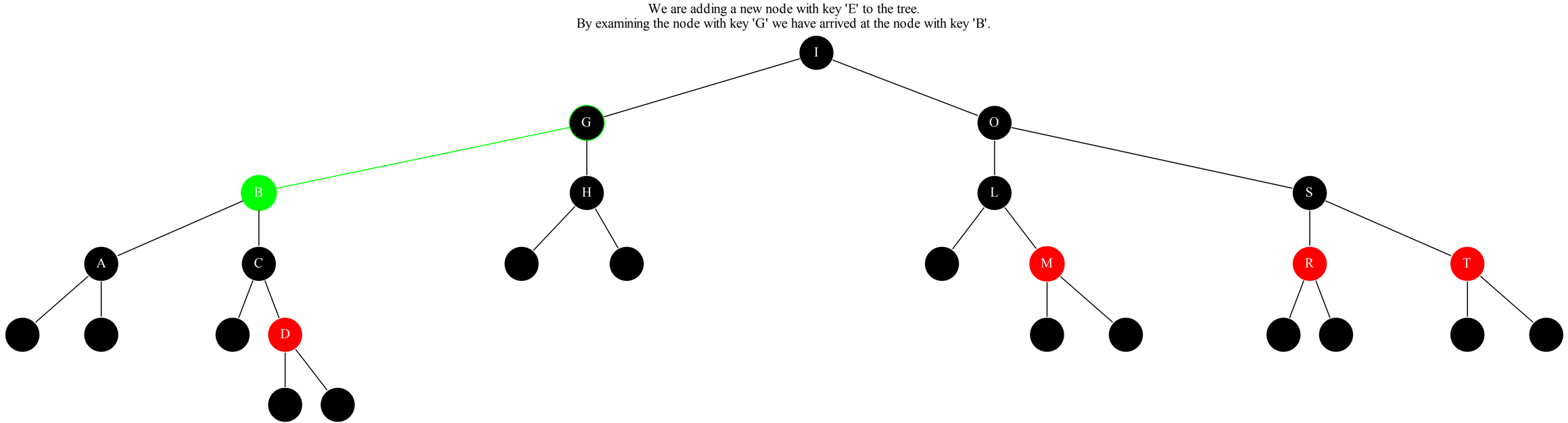

We are adding a new node with key 'G' to the tree.
By examining the node with key 'I' we have arrived at the node with key 'G'.
I
G
O
B
H
L
S
A
C
M
R
T
D

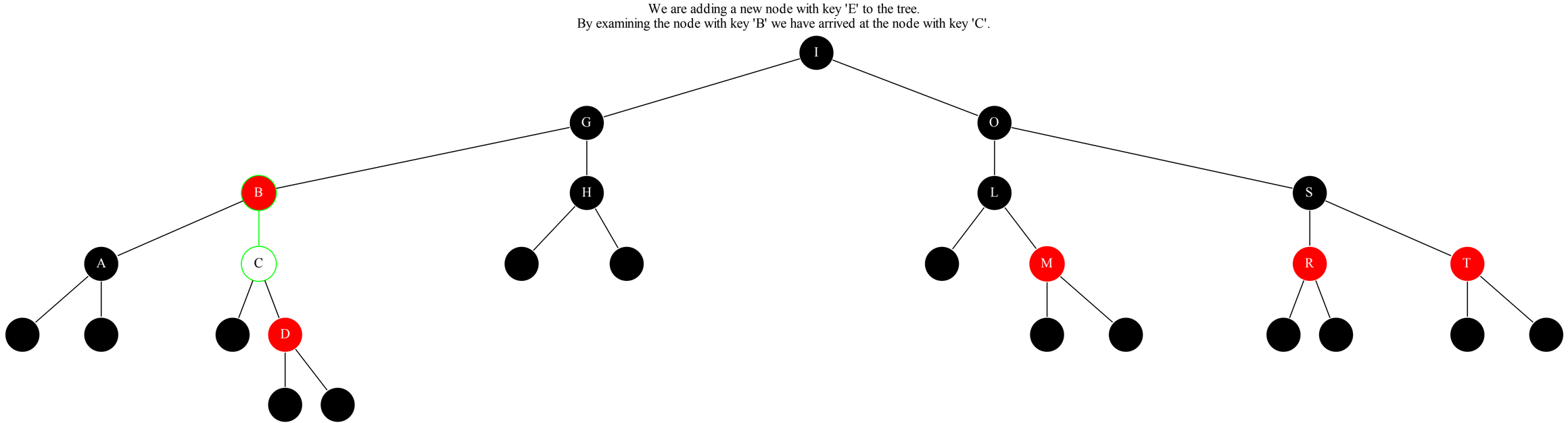
The node with key 'G' is already in the tree, so we terminate the insertion.

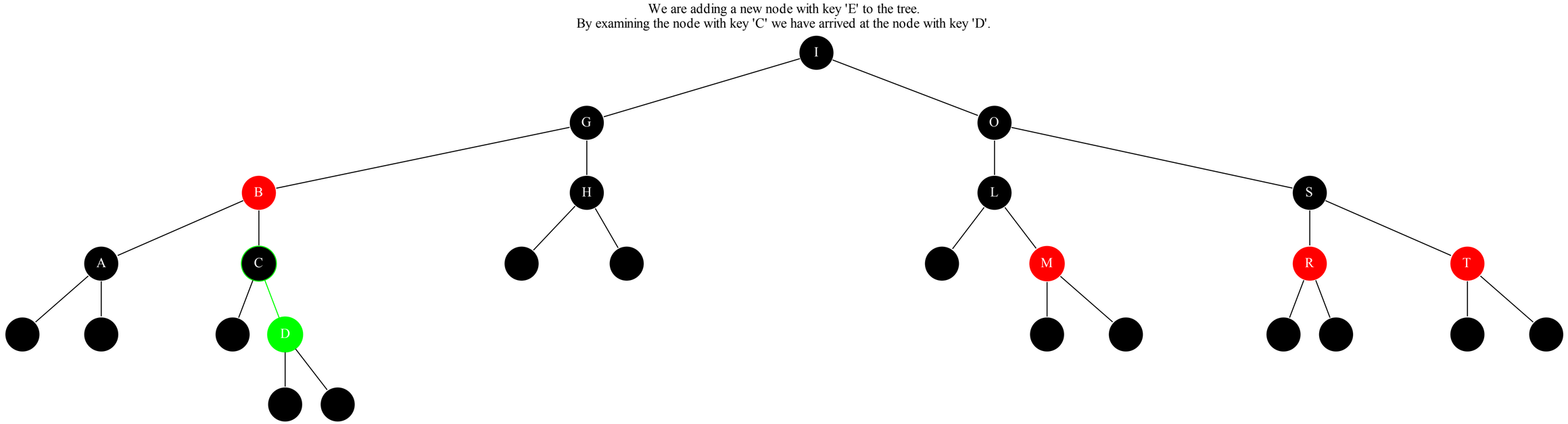



We are adding a new node with key 'E' to the tree.
The first step is to find a leaf (Nill), which the new node will replace.
We begin by examining the root (key 'I').
I
G
O
B
H
L
S
A
C
M
R
T
D

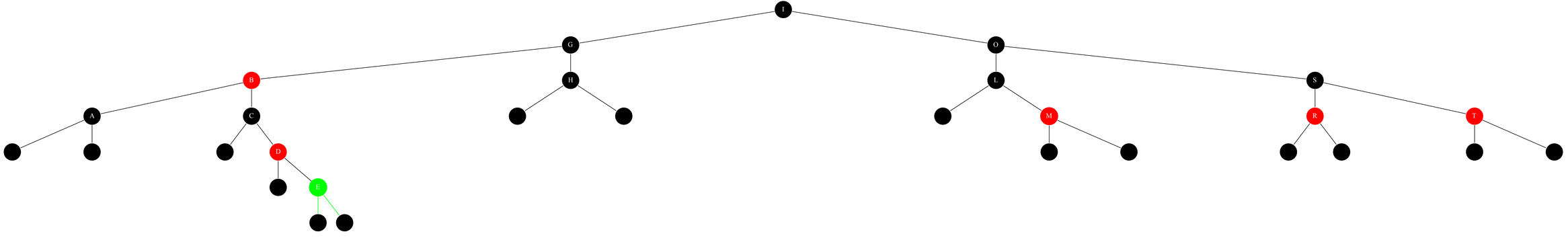

We are adding a new node with key 'E' to the tree.
By examining the node with key 'I' we have arrived at the node with key 'G'.
I
G
O
B
H
L
S
A
C
M
R
T
D

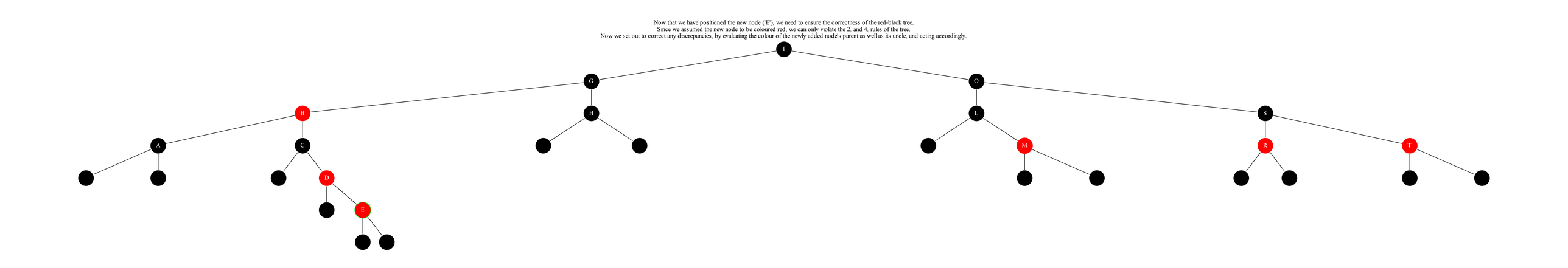

We are adding a new node with key 'E' to the tree.
By examining the node with key 'G' we have arrived at the node with key 'B'.
I
G
O
B
H
L
S
A
C
M
R
T
D


We are adding a new node with key 'E' to the tree.
By examining the node with key 'B' we have arrived at the node with key 'C'.
I
G
O
B
H
L
S
A
C
M
R
T
D

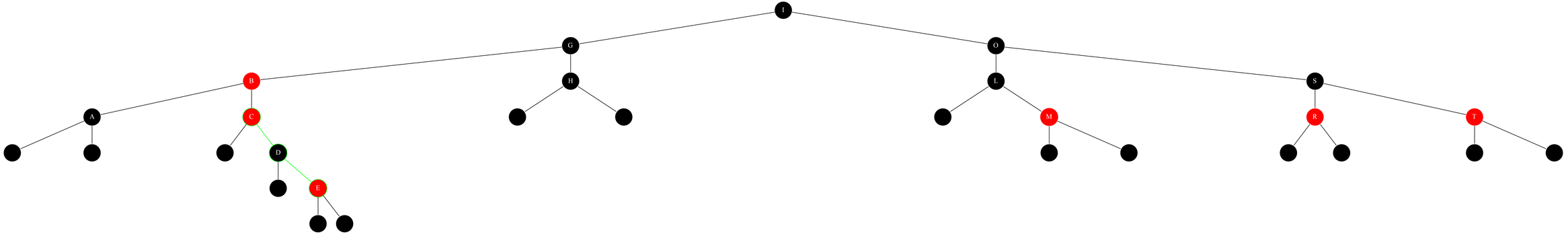

We are adding a new node with key 'E' to the tree.
By examining the node with key 'C' we have arrived at the node with key 'D'.
I
G
O
B
H
L
S
A
C
M
R
T
D

Since the key of the new node ('E') is greater than the one of its new parent ('D'), we add it to the right of the parent.

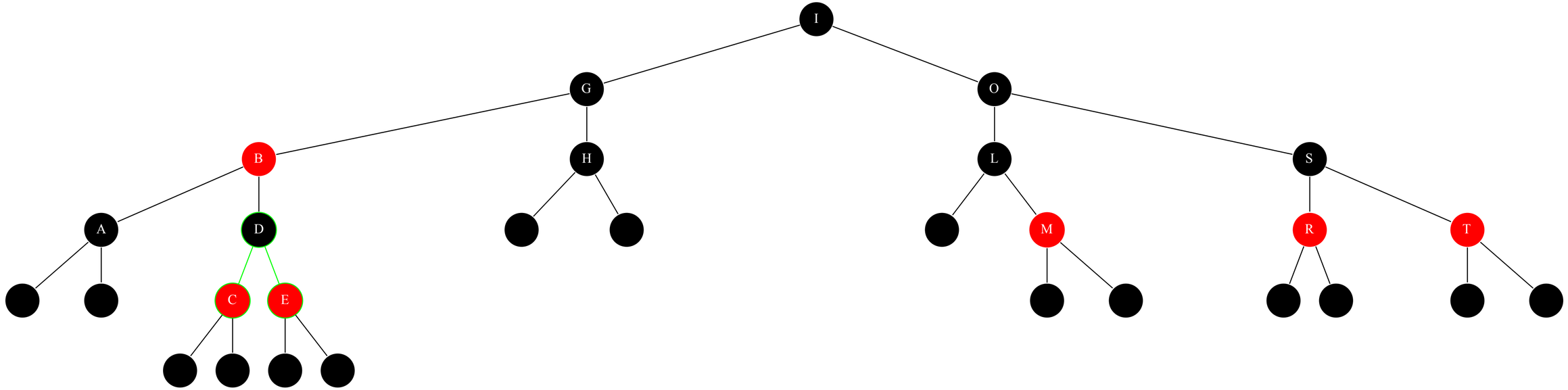



Now that we have positioned the new node ('E'), we need to ensure the correctness of the red-black tree.
Since we assumed the new node to be coloured red, we can only violate the 2. and 4. rules of the tree.
Now we set out to correct any discrepancies, by evaluating the colour of the newly added node's parent as well as its uncle, and acting accordingly.
I
G
O
B
H
L
S
A
C
M
R
T
D
E

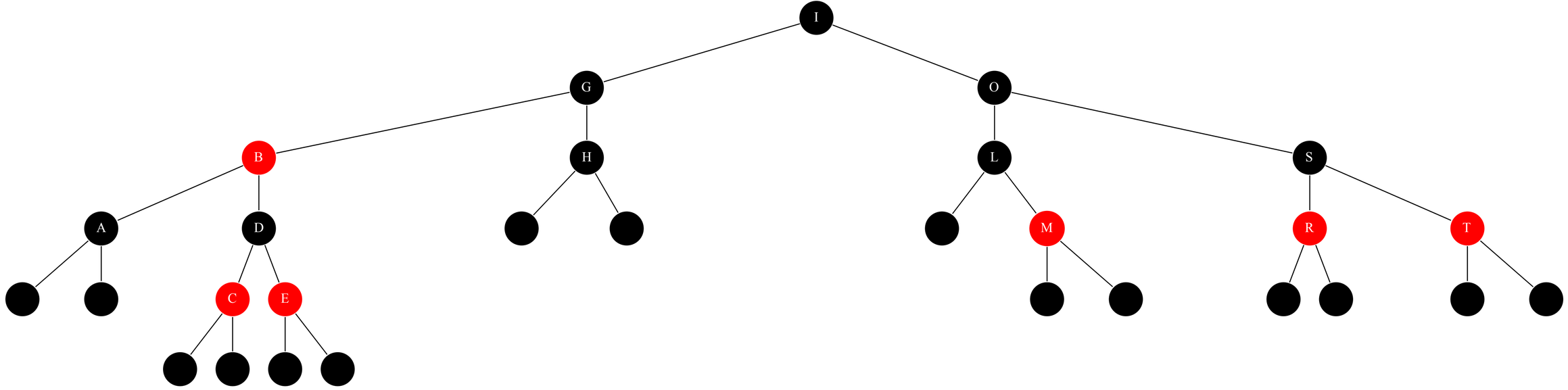
Since the currently examined node ('E'), its parent ('D') and grandparent ('C') are 'in-line', we now had to recolour them, so that when we rotate them in the next step, we will get a correct tree.

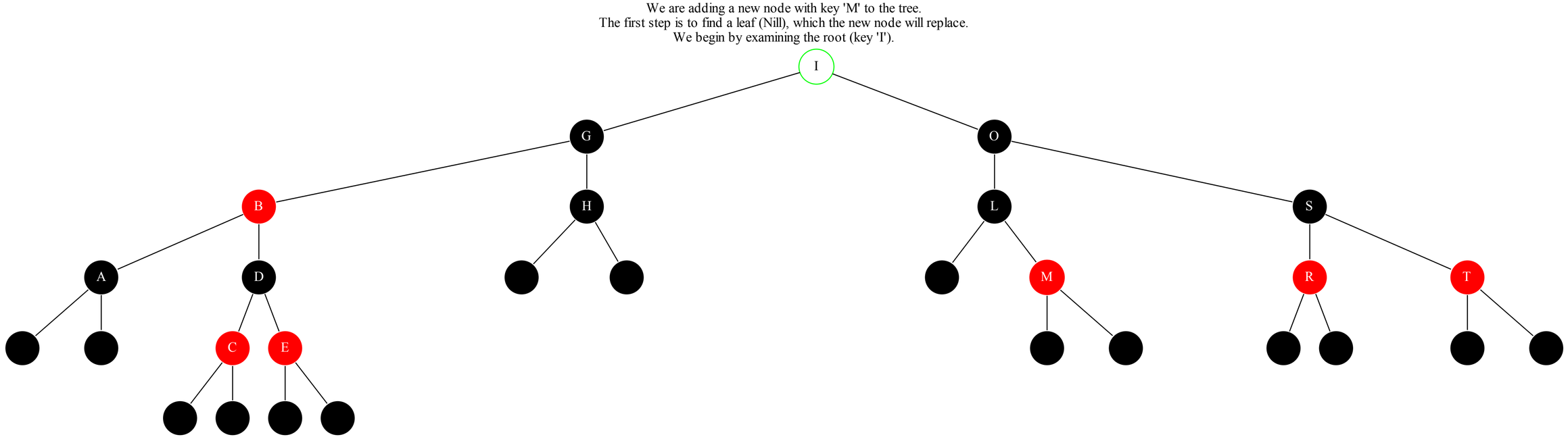


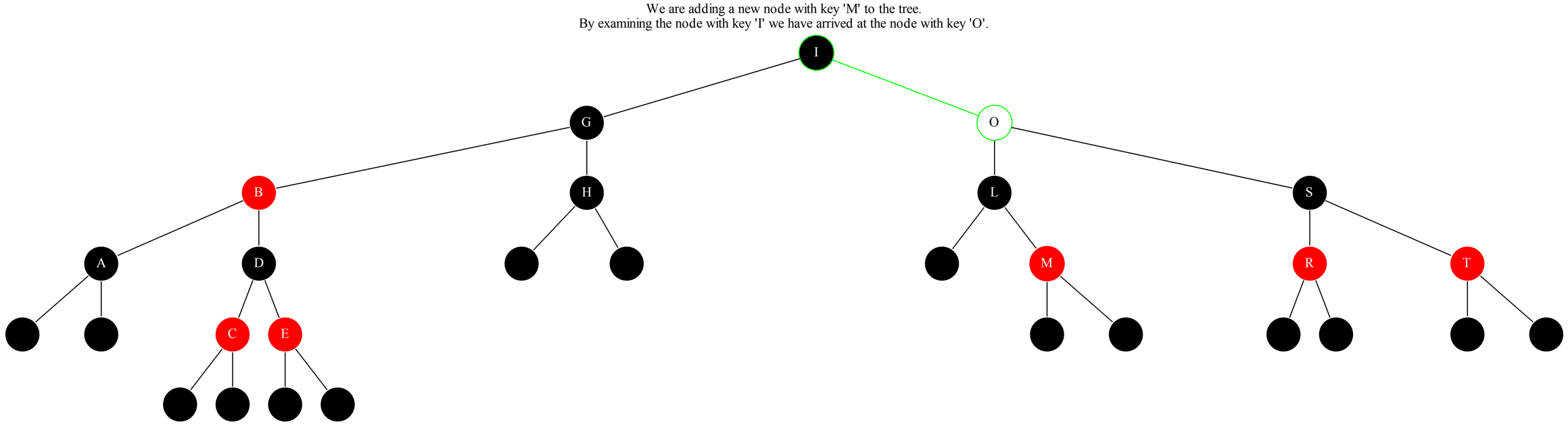
Since the currently examined node ('E'), its parent ('D') and grandparent ('C') are 'in-line' and properly coloured now, just one more rotation gives a correct red-black tree.

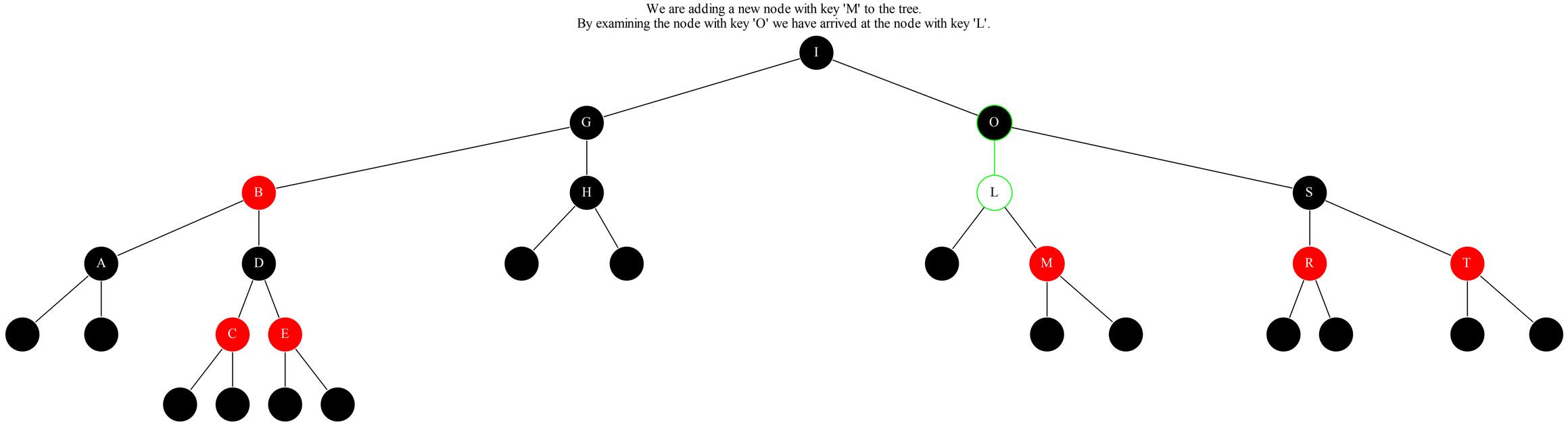


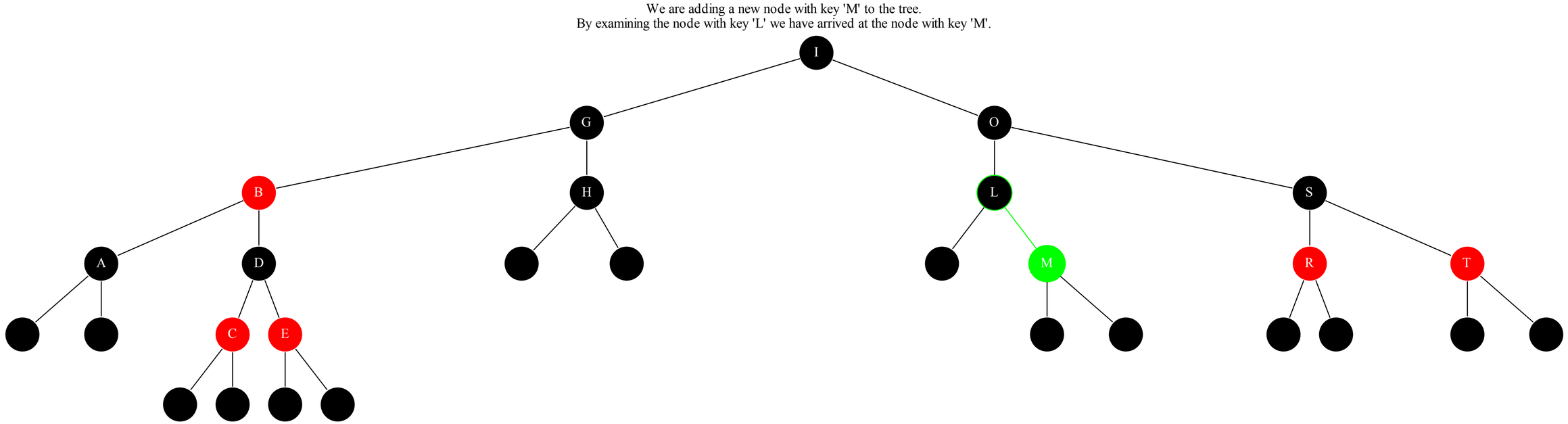
Since all of the required red-black properties are satisfied, we have arrived to a correctly formed red-black tree.

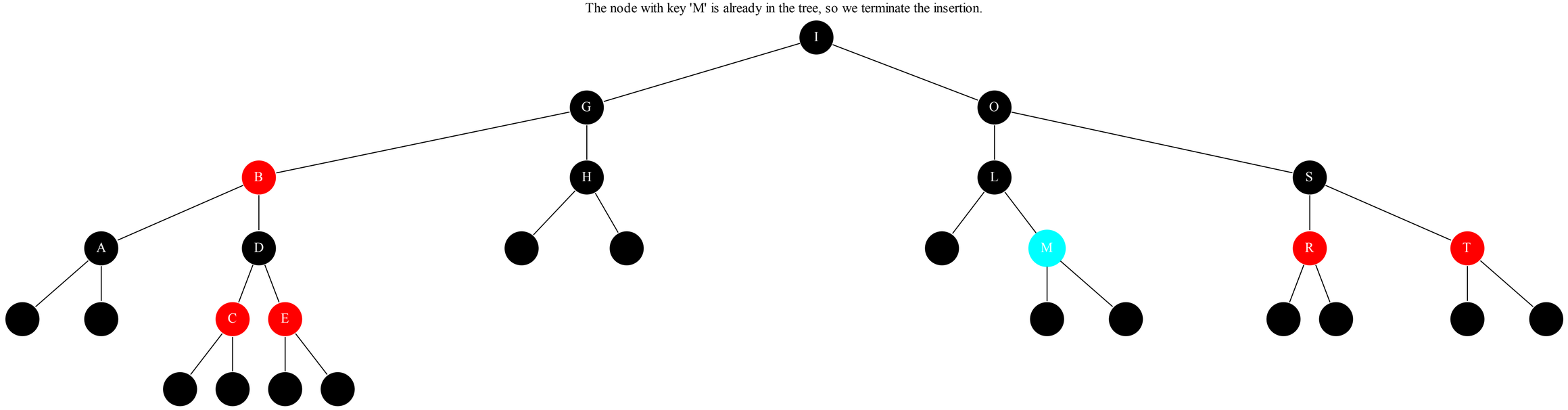


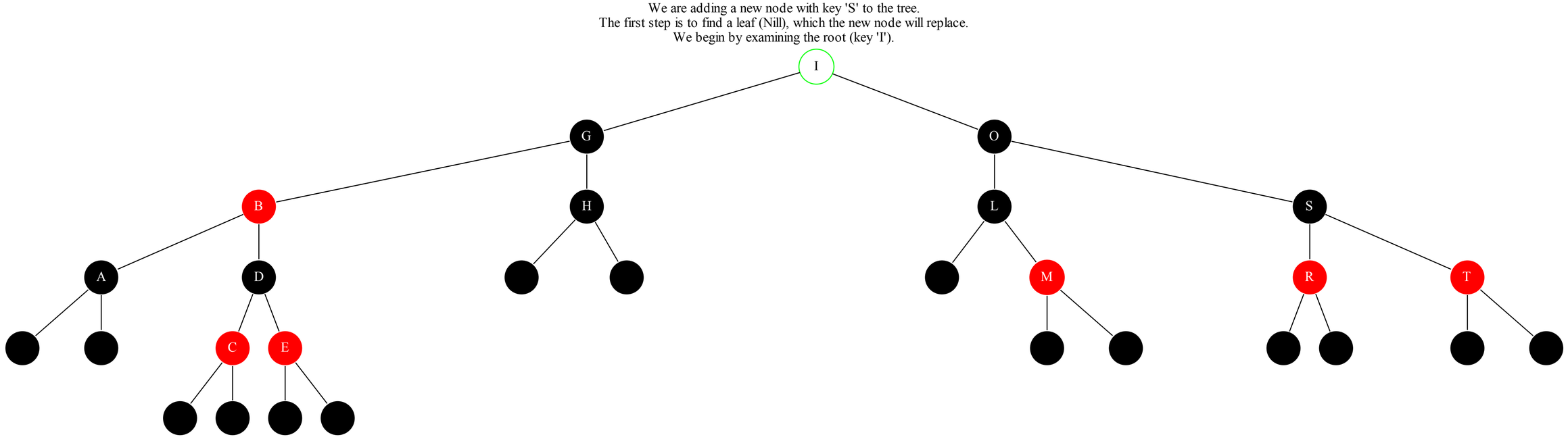

We are adding a new node with key 'M' to the tree.
The first step is to find a leaf (Nill), which the new node will replace.
We begin by examining the root (key 'I').
I
G
O
B
H
L
S
A
D
M
R
T
C
E

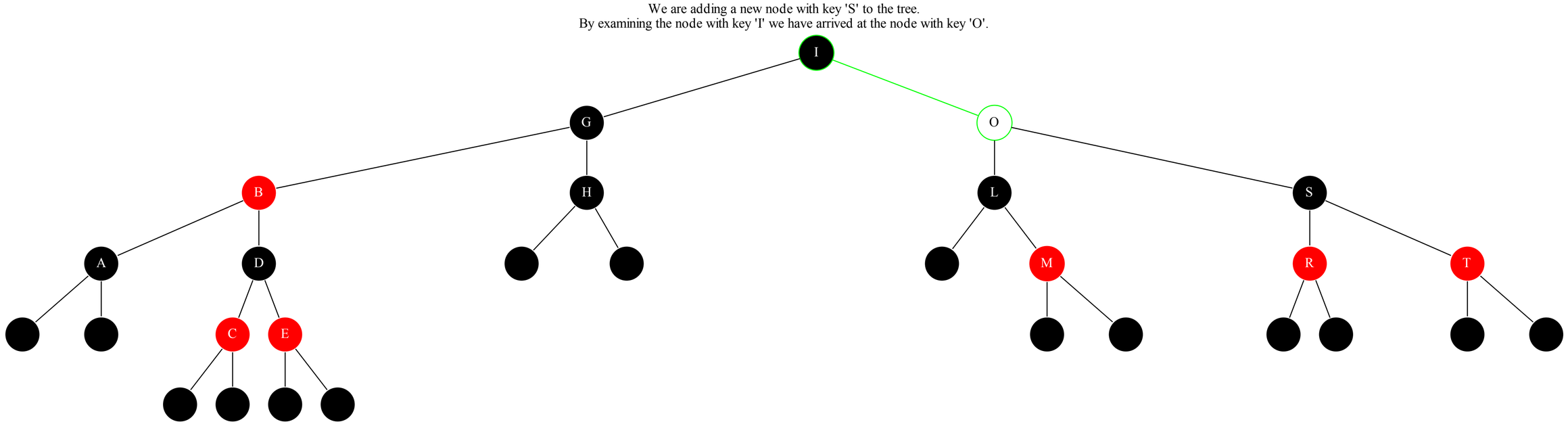

We are adding a new node with key 'M' to the tree.
By examining the node with key 'I' we have arrived at the node with key 'O'.
I
G
O
B
H
L
S
A
D
M
R
T
C
E

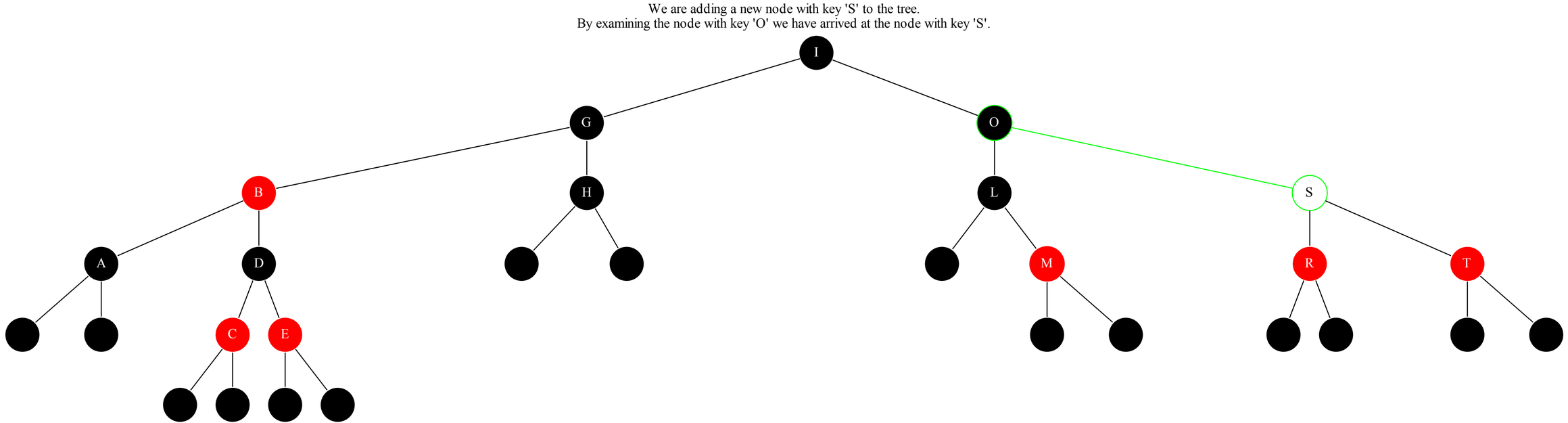

We are adding a new node with key 'M' to the tree.
By examining the node with key 'O' we have arrived at the node with key 'L'.
I
G
O
B
H
L
S
A
D
M
R
T
C
E


We are adding a new node with key 'M' to the tree.
By examining the node with key 'L' we have arrived at the node with key 'M'.
I
G
O
B
H
L
S
A
D
M
R
T
C
E

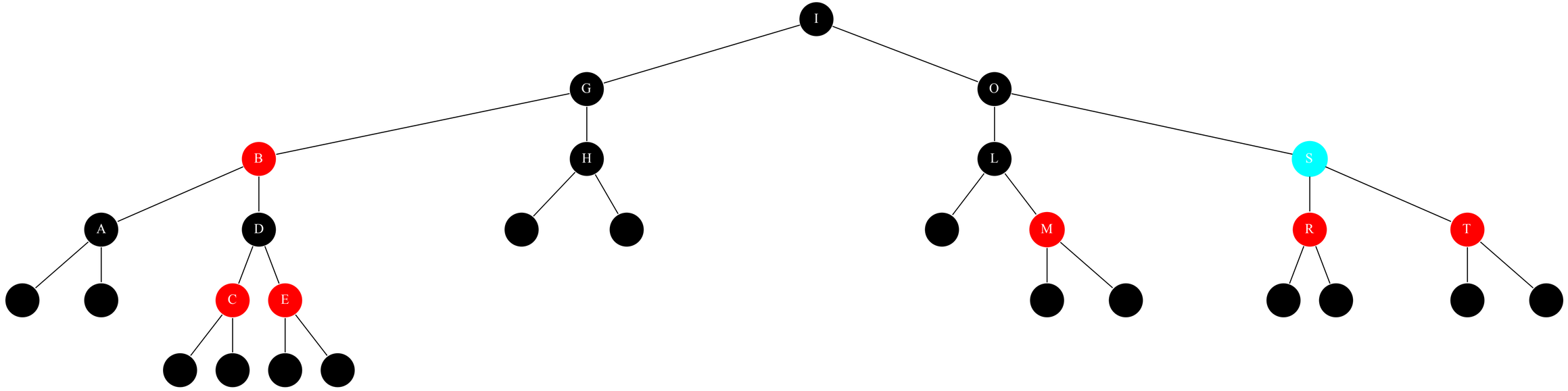

The node with key 'M' is already in the tree, so we terminate the insertion.
I
G
O
B
H
L
S
A
D
M
R
T
C
E

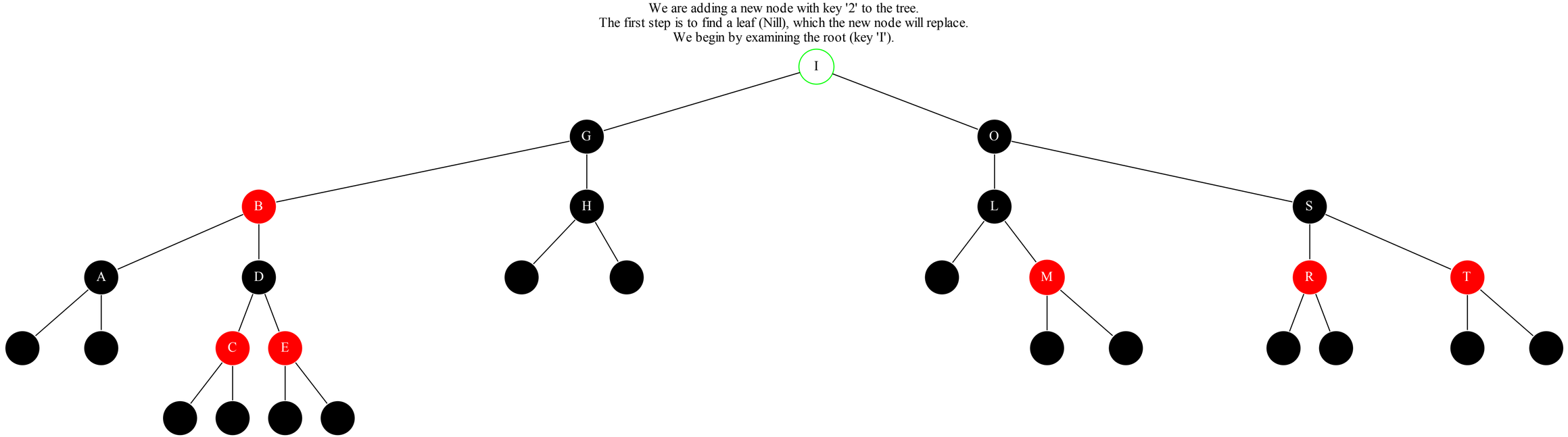

We are adding a new node with key 'S' to the tree.
The first step is to find a leaf (Nill), which the new node will replace.
We begin by examining the root (key 'I').
I
G
O
B
H
L
S
A
D
M
R
T
C
E

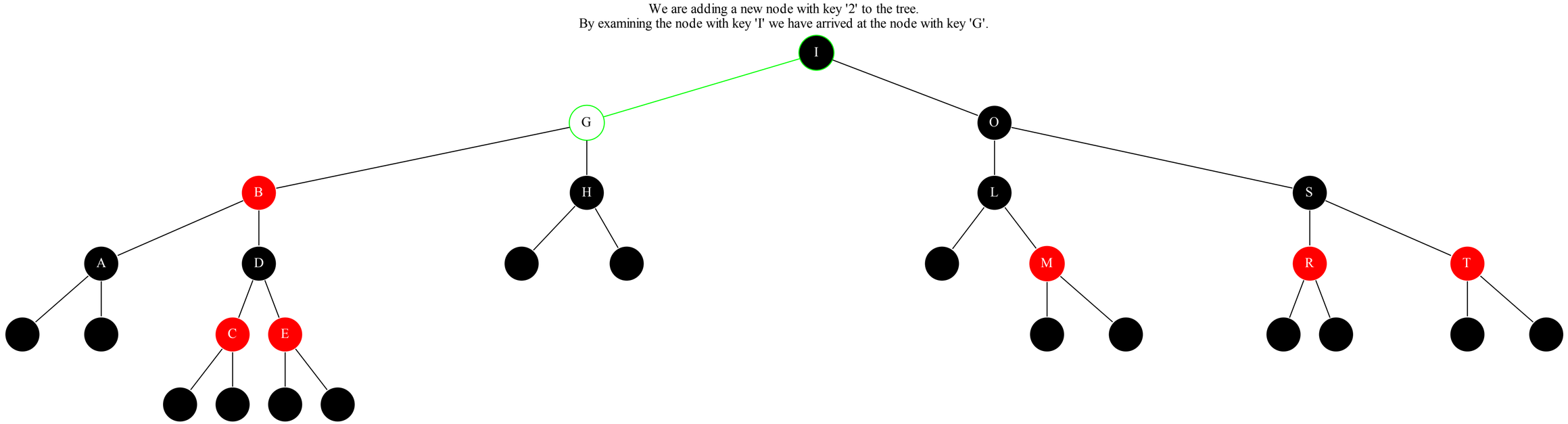

We are adding a new node with key 'S' to the tree.
By examining the node with key 'I' we have arrived at the node with key 'O'.
I
G
O
B
H
L
S
A
D
M
R
T
C
E

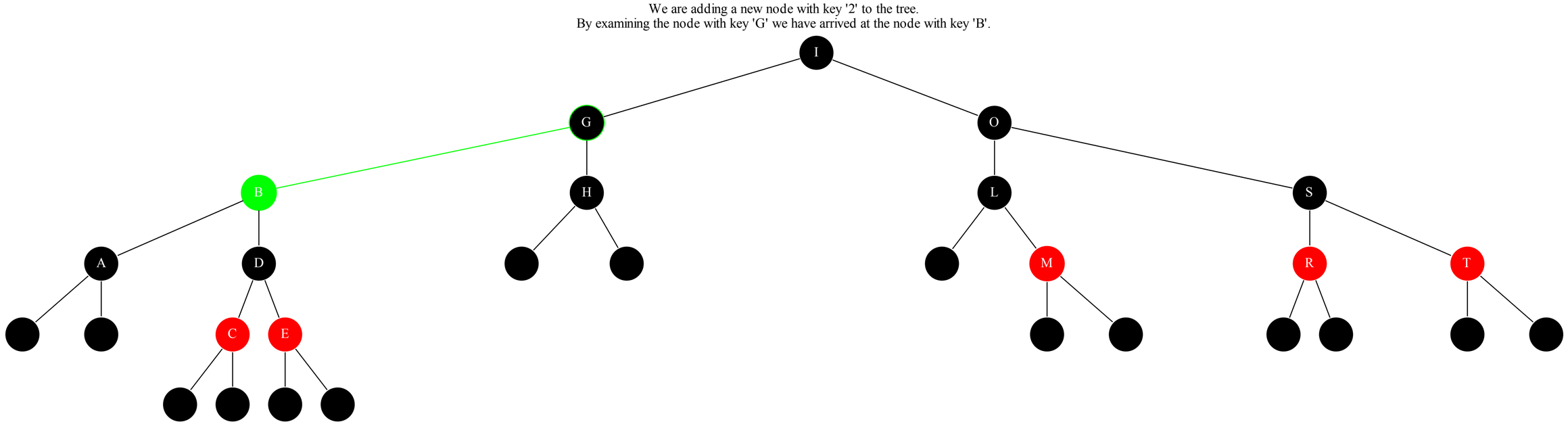

We are adding a new node with key 'S' to the tree.
By examining the node with key 'O' we have arrived at the node with key 'S'.
I
G
O
B
H
L
S
A
D
M
R
T
C
E

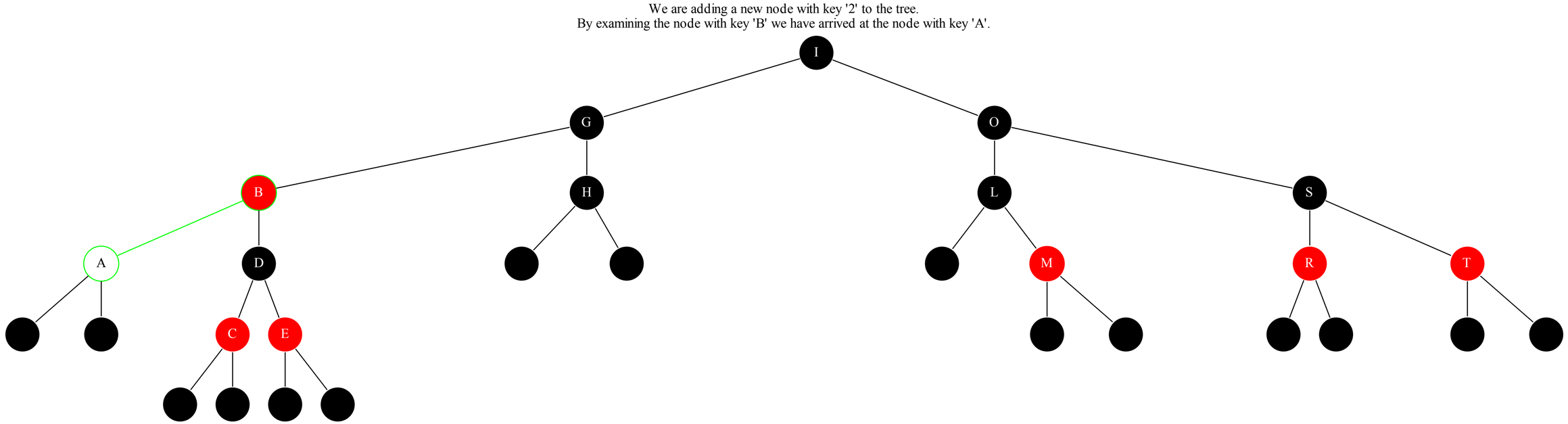
The node with key 'S' is already in the tree, so we terminate the insertion.



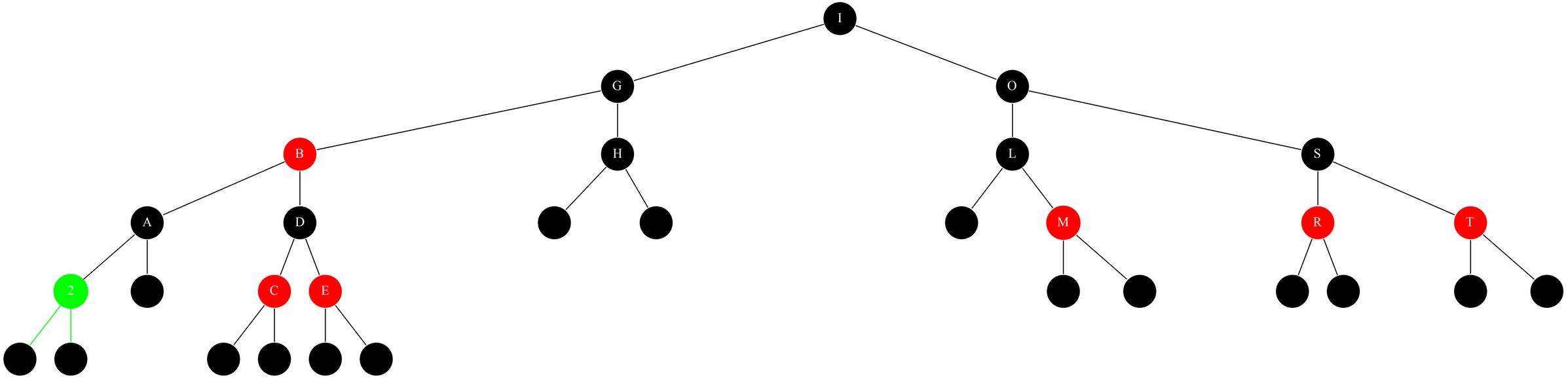

We are adding a new node with key '2' to the tree.
The first step is to find a leaf (Nill), which the new node will replace.
We begin by examining the root (key 'I').
I
G
O
B
H
L
S
A
D
M
R
T
C
E


We are adding a new node with key '2' to the tree.
By examining the node with key 'I' we have arrived at the node with key 'G'.
I
G
O
B
H
L
S
A
D
M
R
T
C
E

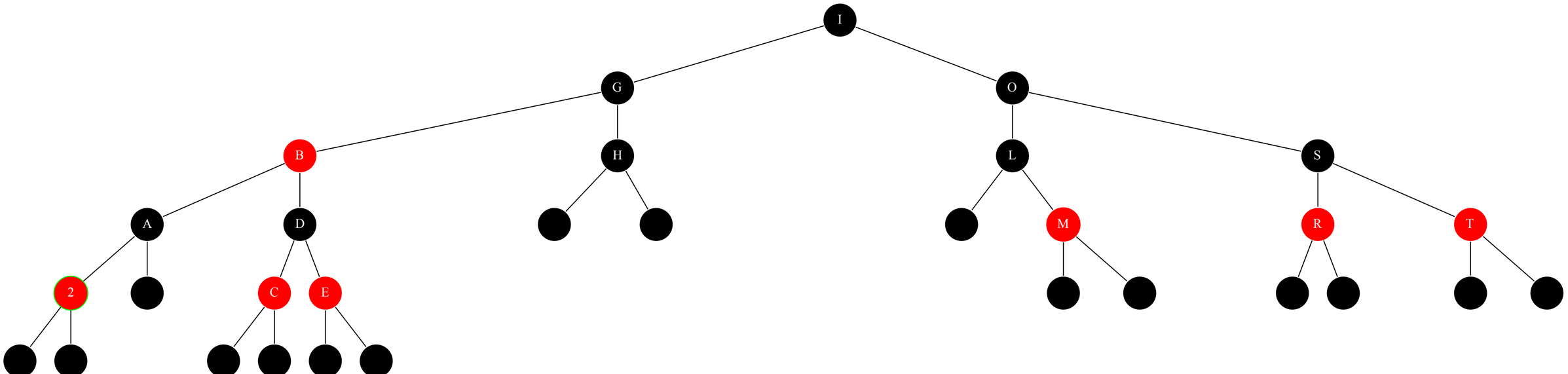

We are adding a new node with key '2' to the tree.
By examining the node with key 'G' we have arrived at the node with key 'B'.
I
G
O
B
H
L
S
A
D
M
R
T
C
E


We are adding a new node with key '2' to the tree.
By examining the node with key 'B' we have arrived at the node with key 'A'.
I
G
O
B
H
L
S
A
D
M
R
T
C
E

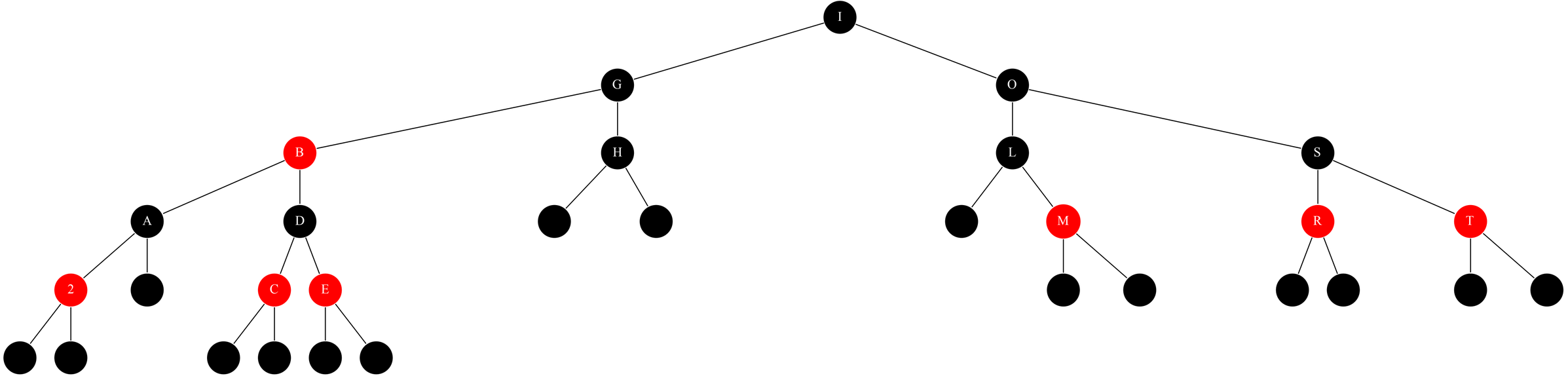
Since the key of the new node ('2') is smaller than the one of its new parent ('A'), we add it to the left of the parent.

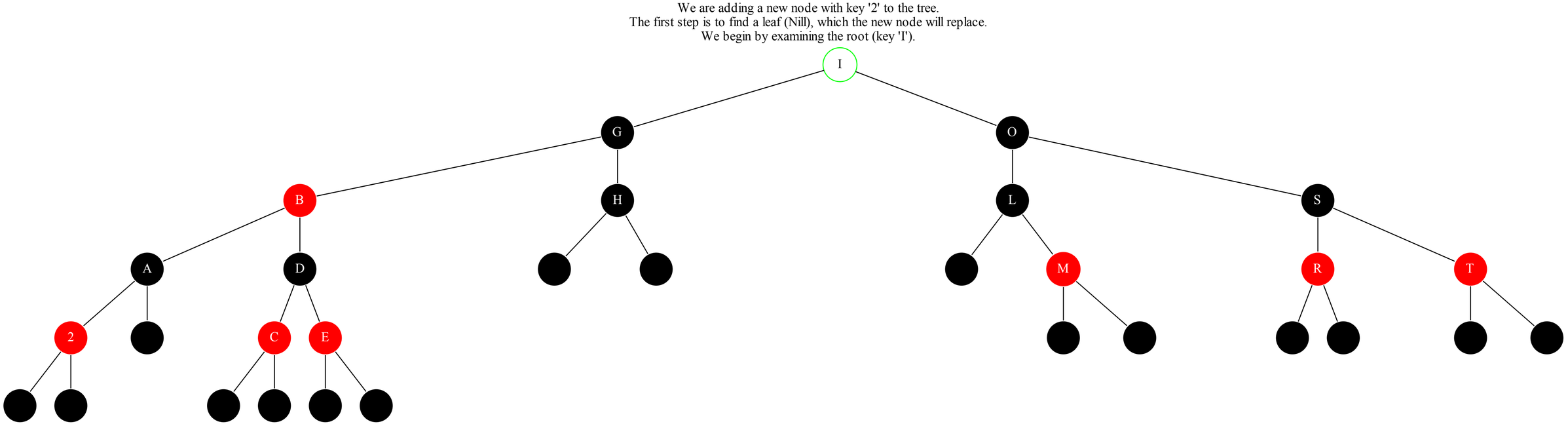


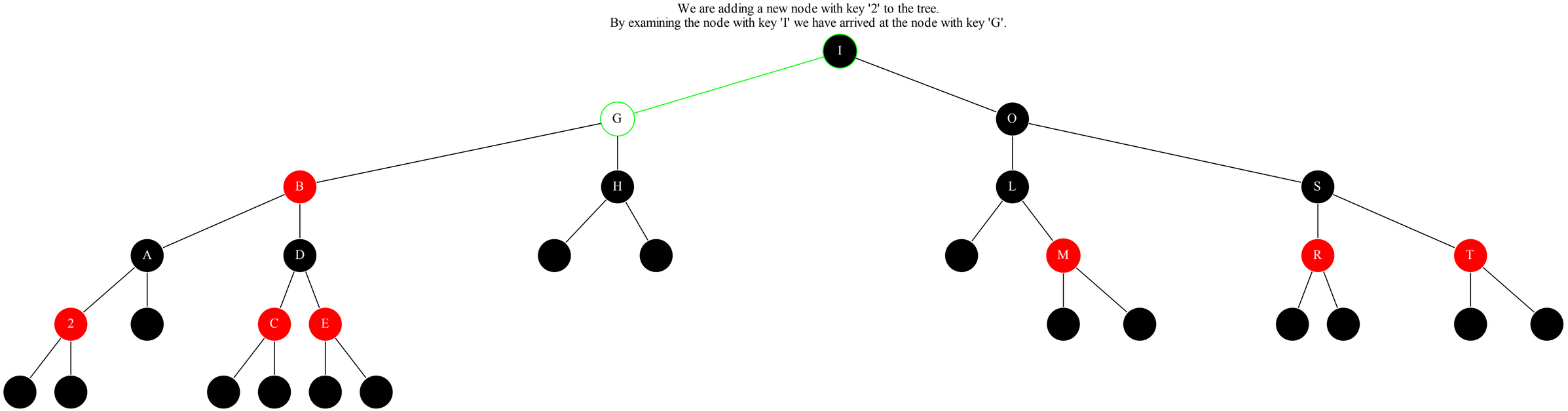
Now that we have positioned the new node ('2'), we need to ensure the correctness of the red-black tree.
Since we assumed the new node to be coloured red, we can only violate the 2. and 4. rules of the tree.
Now we set out to correct any discrepancies, by evaluating the colour of the newly added node's parent as well as its uncle, and acting accordingly.

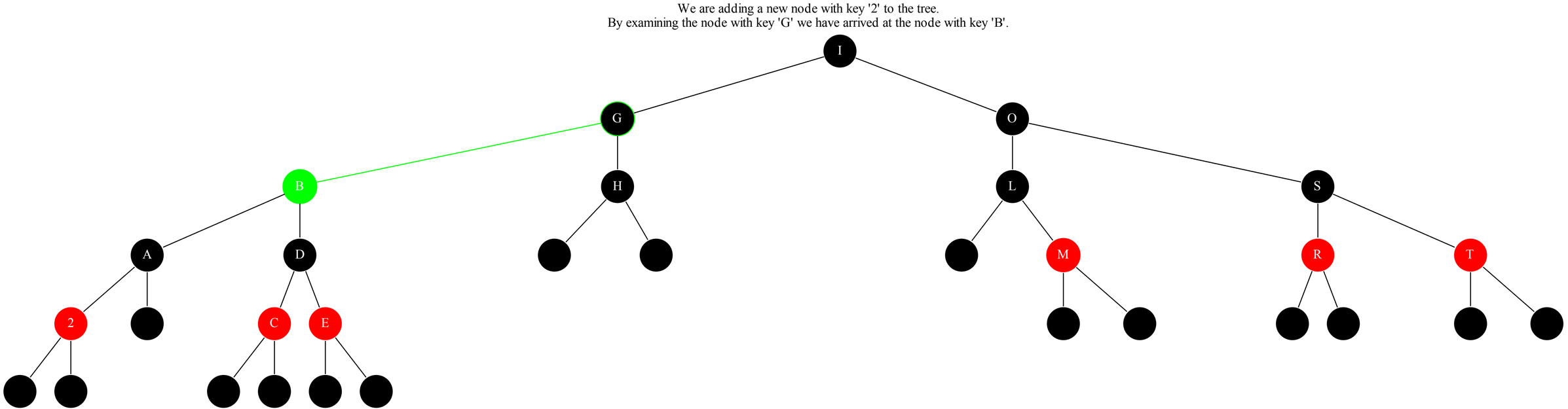


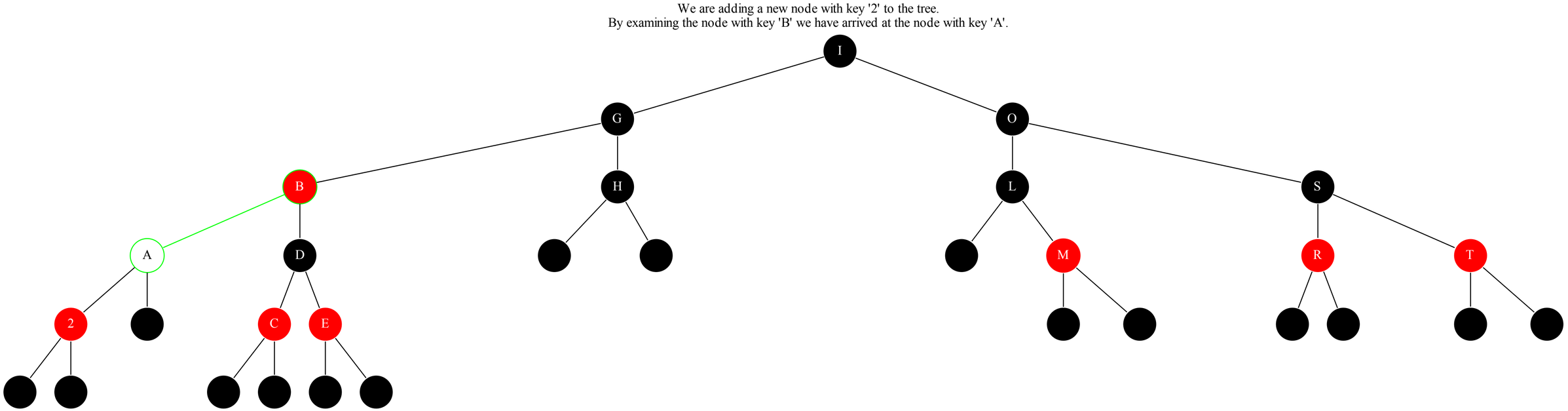
Since all of the required red-black properties are satisfied, we have arrived to a correctly formed red-black tree.

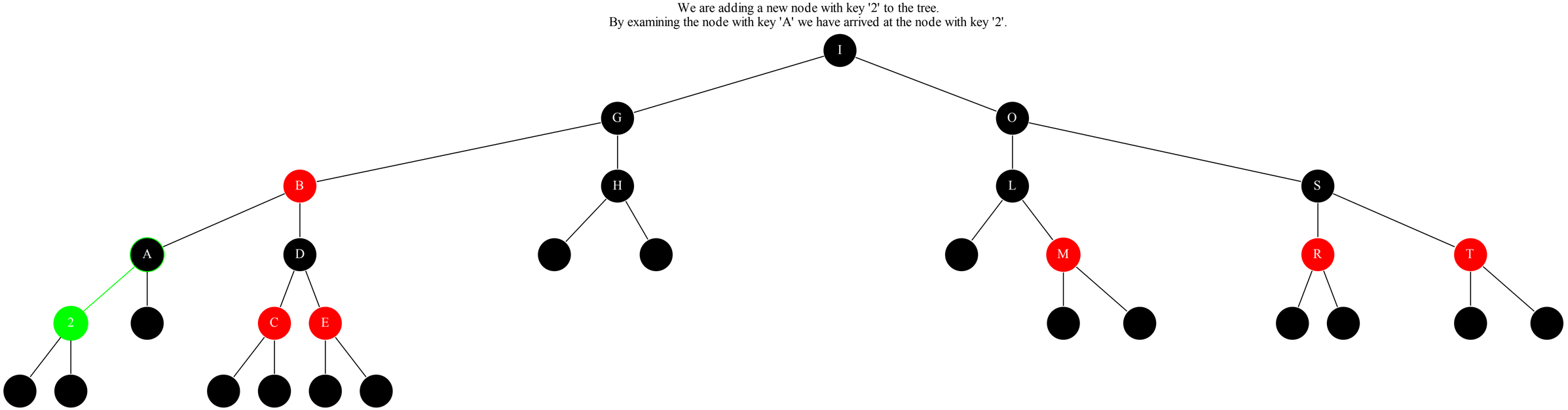


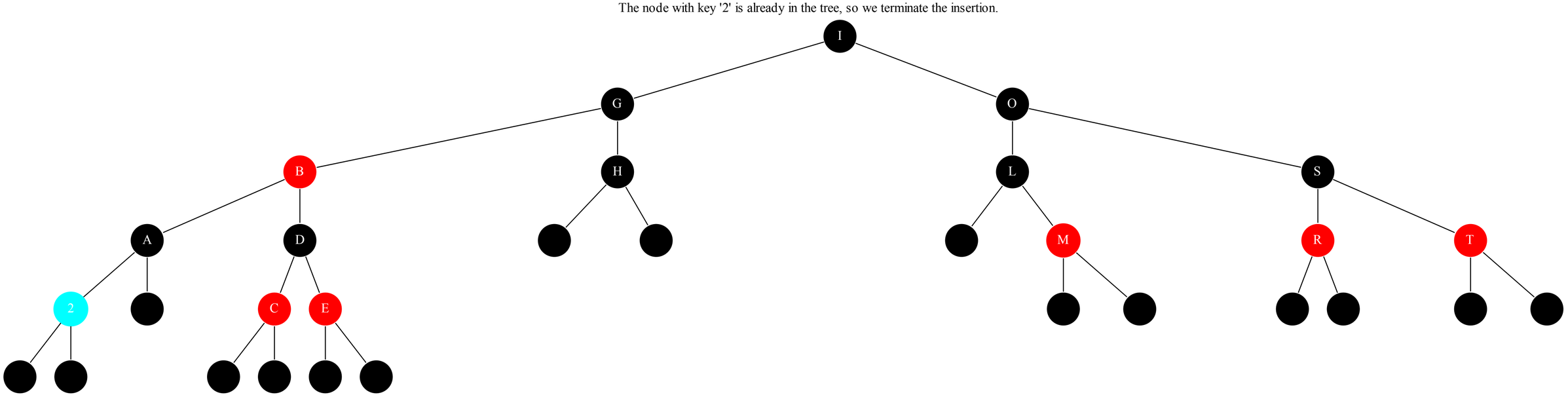

We are adding a new node with key '2' to the tree.
The first step is to find a leaf (Nill), which the new node will replace.
We begin by examining the root (key 'I').
I
G
O
B
H
L
S
A
D
M
R
T
2
C
E

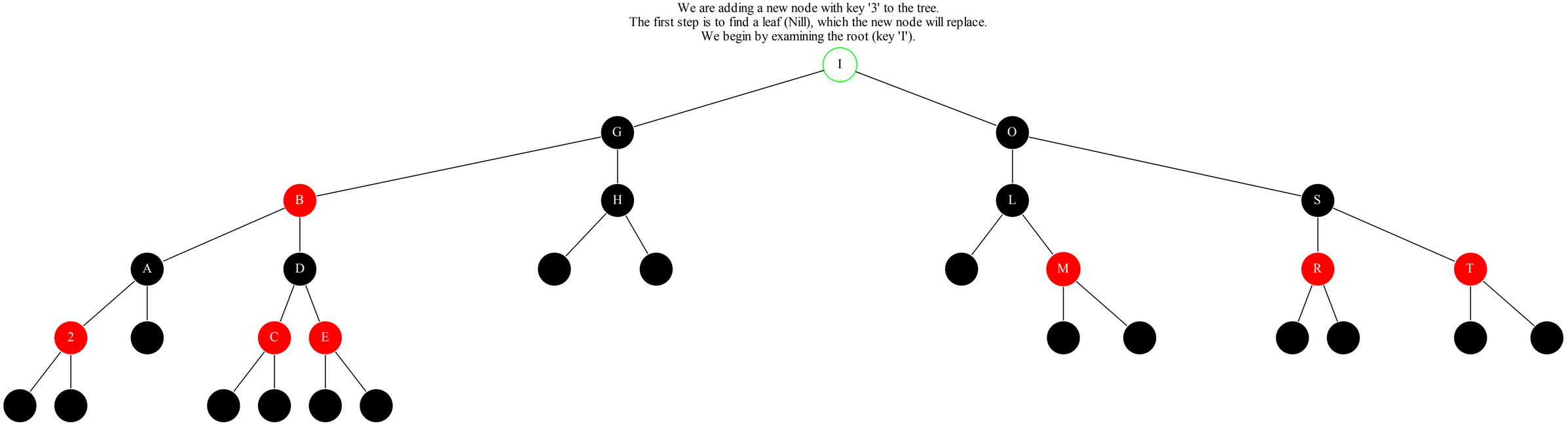

We are adding a new node with key '2' to the tree.
By examining the node with key 'I' we have arrived at the node with key 'G'.
I
G
O
B
H
L
S
A
D
M
R
T
2
C
E

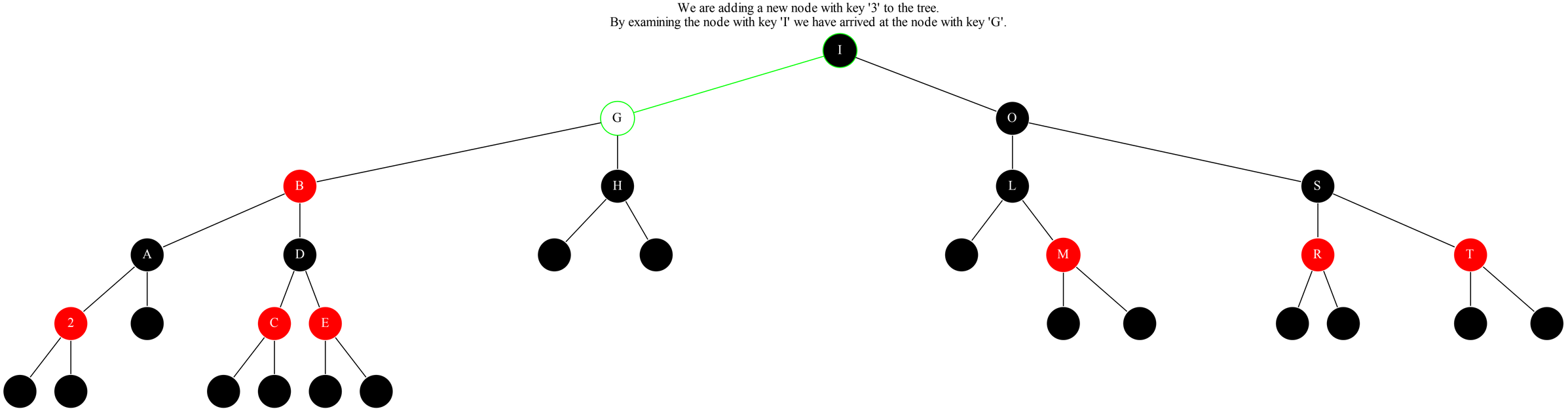

We are adding a new node with key '2' to the tree.
By examining the node with key 'G' we have arrived at the node with key 'B'.
I
G
O
B
H
L
S
A
D
M
R
T
2
C
E

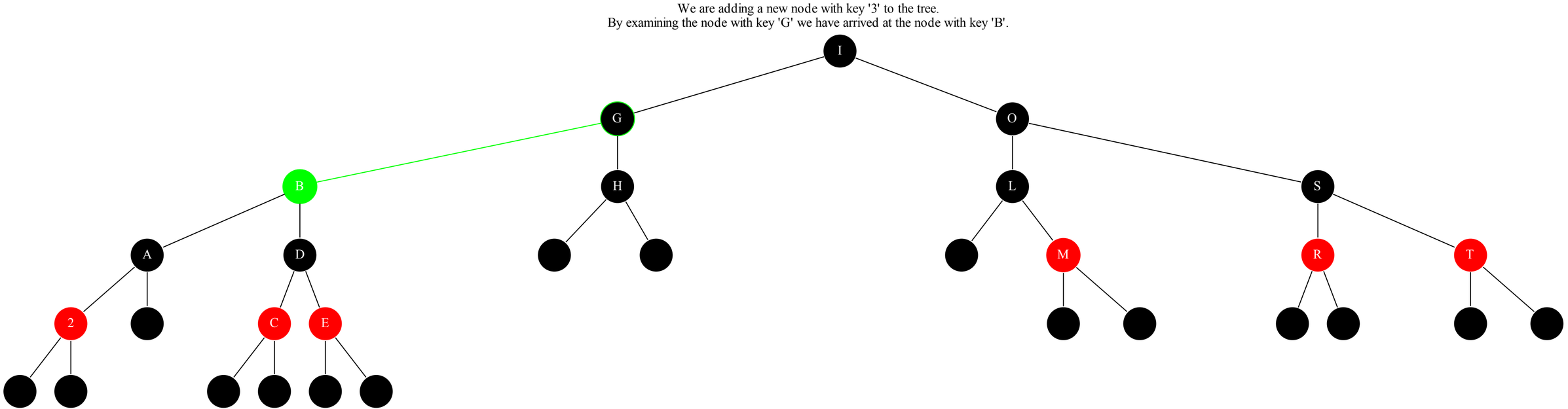

We are adding a new node with key '2' to the tree.
By examining the node with key 'B' we have arrived at the node with key 'A'.
I
G
O
B
H
L
S
A
D
M
R
T
2
C
E

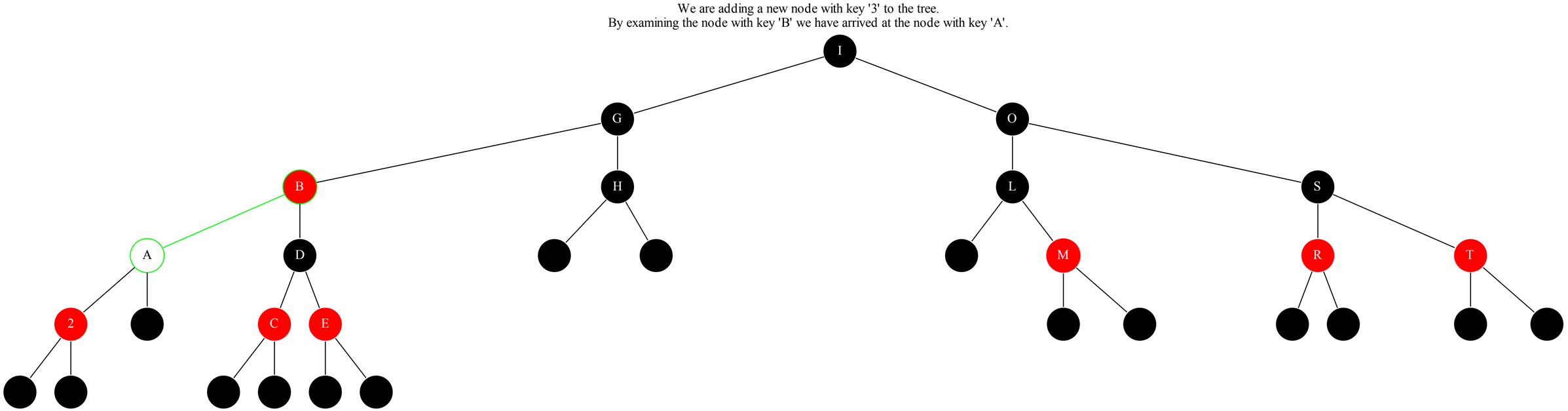

We are adding a new node with key '2' to the tree.
By examining the node with key 'A' we have arrived at the node with key '2'.
I
G
O
B
H
L
S
A
D
M
R
T
2
C
E

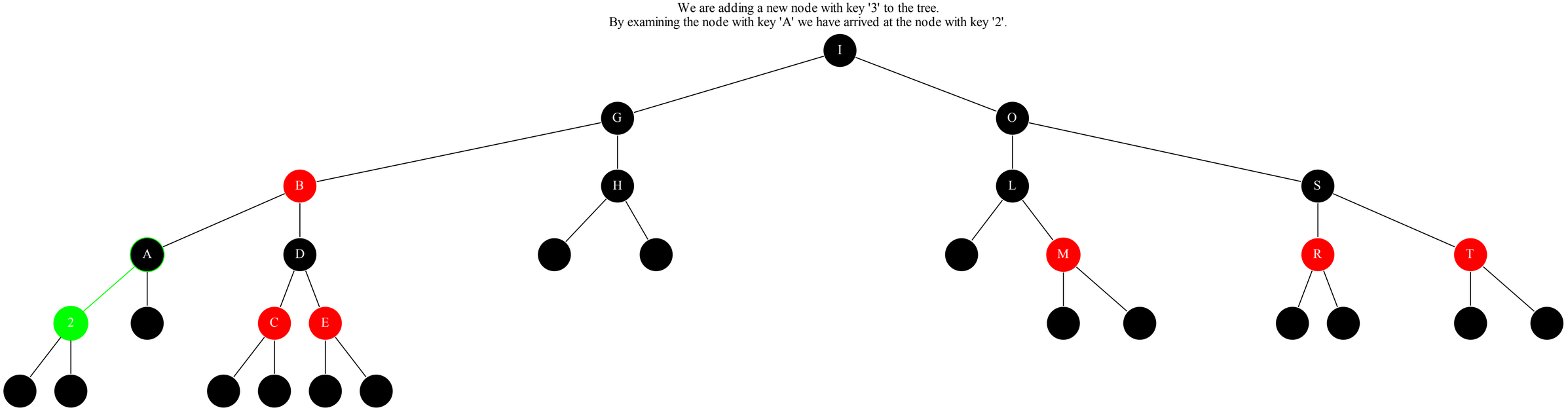

The node with key '2' is already in the tree, so we terminate the insertion.
I
G
O
B
H
L
S
A
D
M
R
T
2
C
E


We are adding a new node with key '3' to the tree.
The first step is to find a leaf (Nill), which the new node will replace.
We begin by examining the root (key 'I').
I
G
O
B
H
L
S
A
D
M
R
T
2
C
E

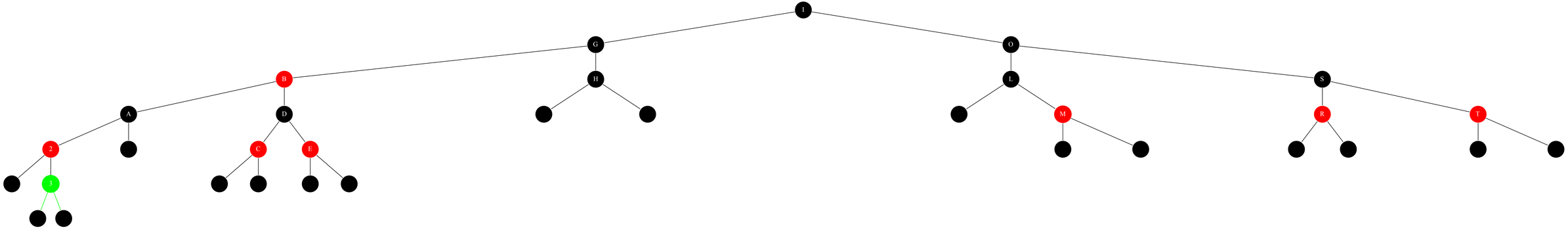

We are adding a new node with key '3' to the tree.
By examining the node with key 'I' we have arrived at the node with key 'G'.
I
G
O
B
H
L
S
A
D
M
R
T
2
C
E

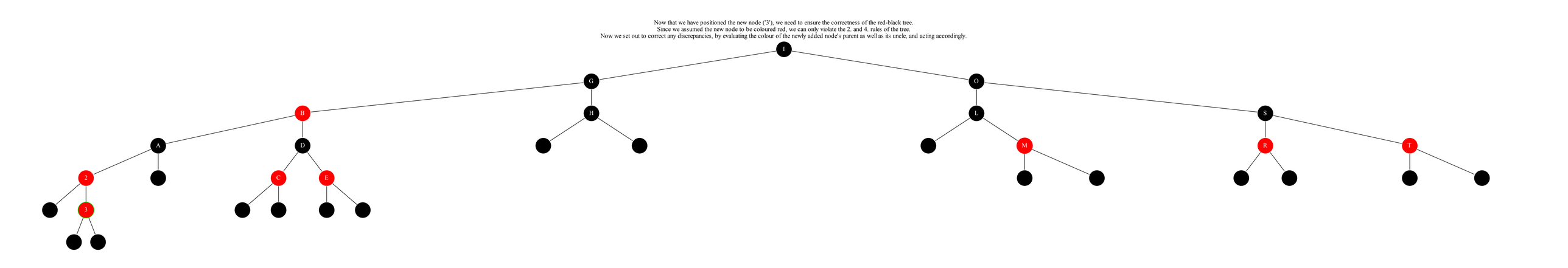

We are adding a new node with key '3' to the tree.
By examining the node with key 'G' we have arrived at the node with key 'B'.
I
G
O
B
H
L
S
A
D
M
R
T
2
C
E

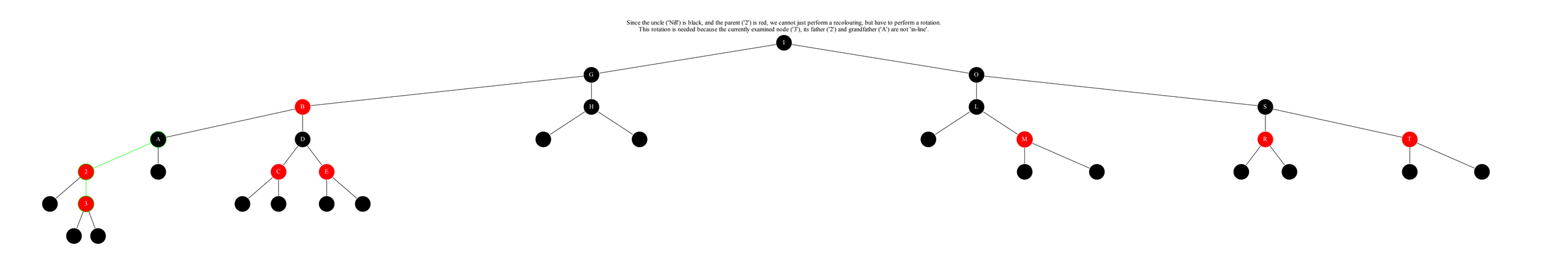

We are adding a new node with key '3' to the tree.
By examining the node with key 'B' we have arrived at the node with key 'A'.
I
G
O
B
H
L
S
A
D
M
R
T
2
C
E


We are adding a new node with key '3' to the tree.
By examining the node with key 'A' we have arrived at the node with key '2'.
I
G
O
B
H
L
S
A
D
M
R
T
2
C
E

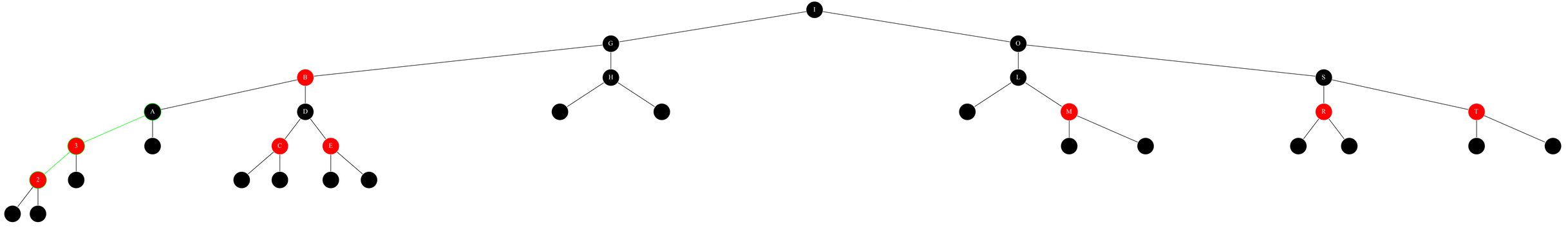
Since the key of the new node ('3') is greater than the one of its new parent ('2'), we add it to the right of the parent.

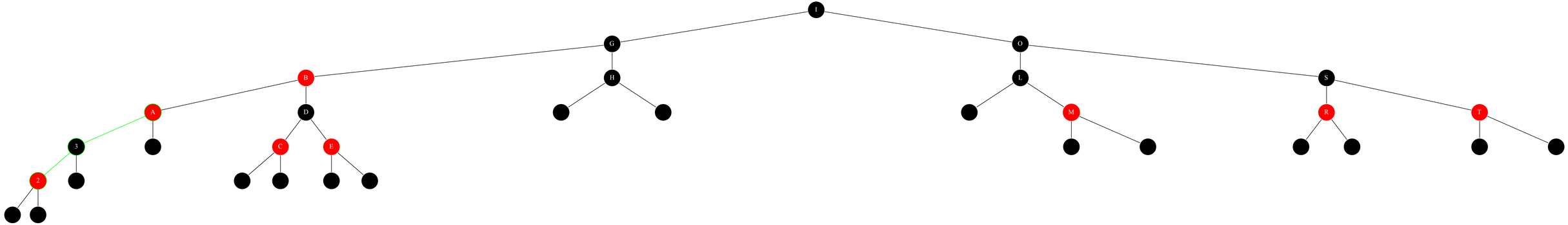



Now that we have positioned the new node ('3'), we need to ensure the correctness of the red-black tree.
Since we assumed the new node to be coloured red, we can only violate the 2. and 4. rules of the tree.
Now we set out to correct any discrepancies, by evaluating the colour of the newly added node's parent as well as its uncle, and acting accordingly.
I
G
O
B
H
L
S
A
D
M
R
T
2
C
E
3

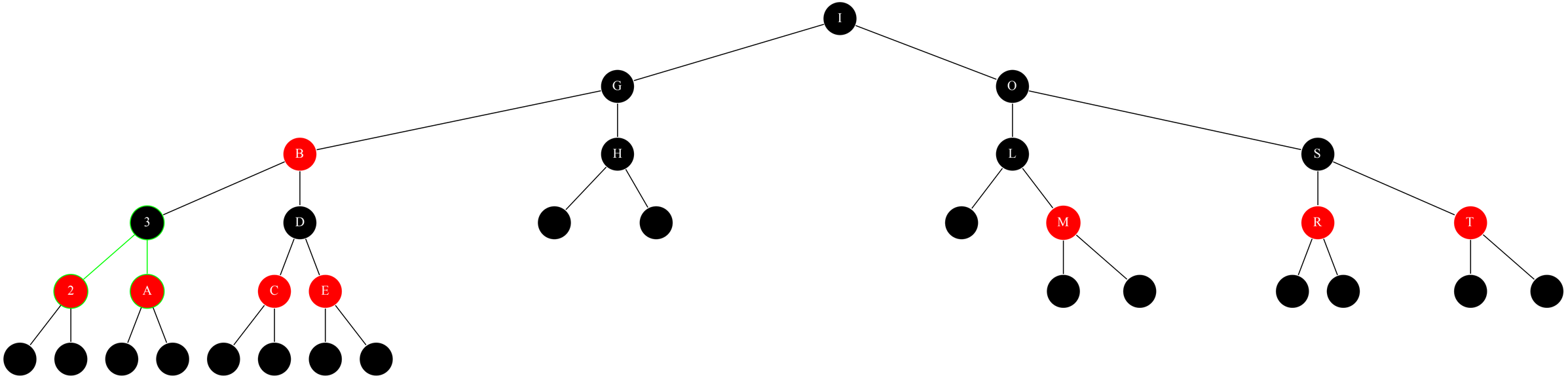

Since the uncle ('Nill') is black, and the parent ('2') is red, we cannot just perform a recolouring, but have to perform a rotation.
This rotation is needed because the currently examined node ('3'), its father ('2') and grandfather ('A') are not 'in-line'.
I
G
O
B
H
L
S
A
D
M
R
T
2
C
E
3

After we performed the before-mentioned rotation, the currently examined node ('2'), its parent ('3') and grandparent ('A') are now 'in-line'.

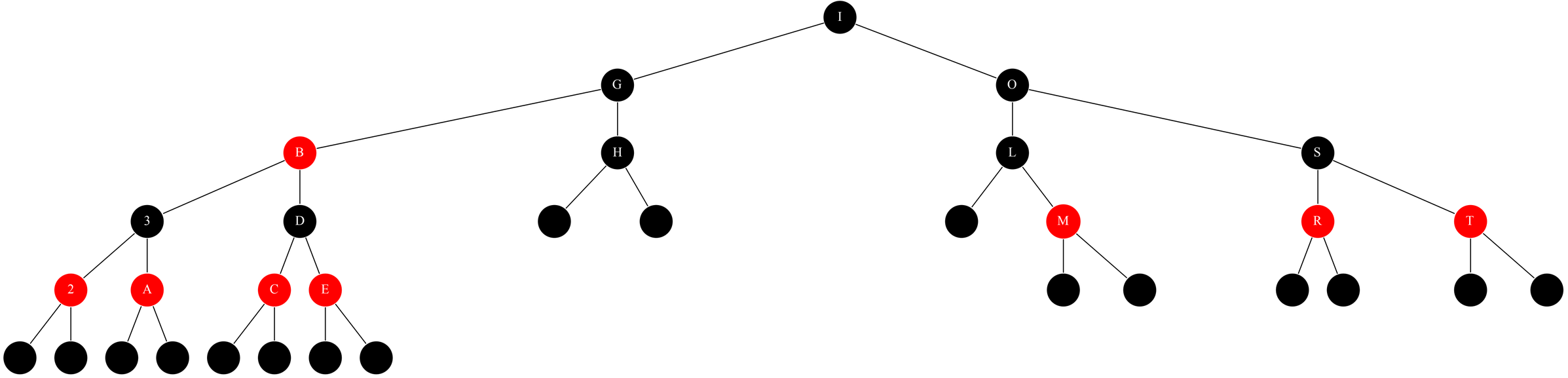


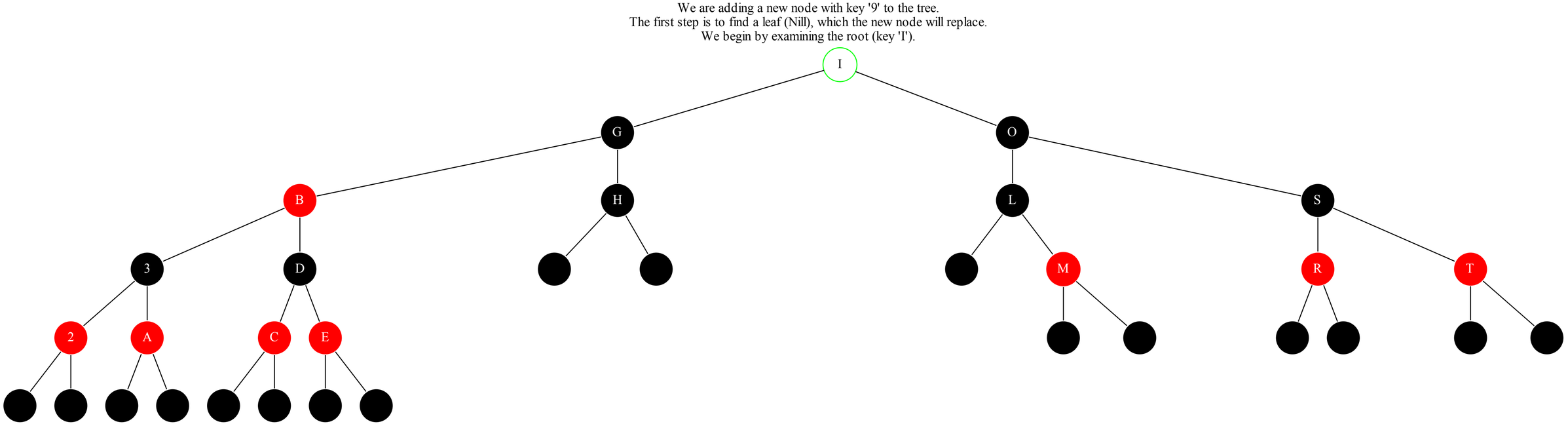

We have now made the currently examined node ('2'), its parent ('3') and grandparent ('A') align.
We now had to recolour them, so that when we rotate them in the next step, we will get a correct tree.
I
G
O
B
H
L
S
A
D
M
R
T
3
C
E
2

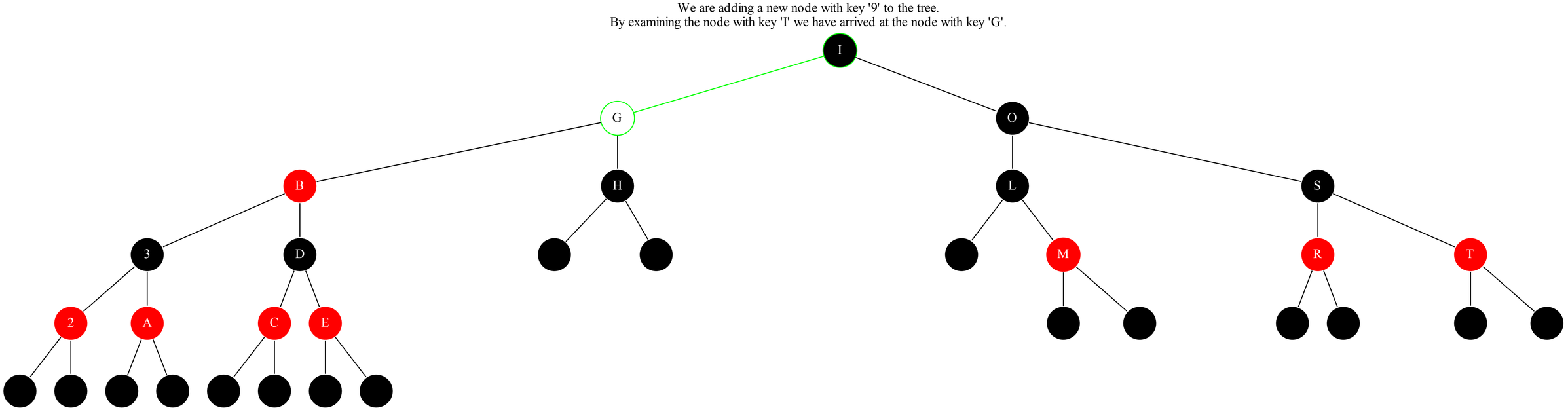
Since the currently examined node ('2'), its parent ('3') and grandparent ('2') are 'in-line' and properly coloured now, just one more rotation gives a correct red-black tree.

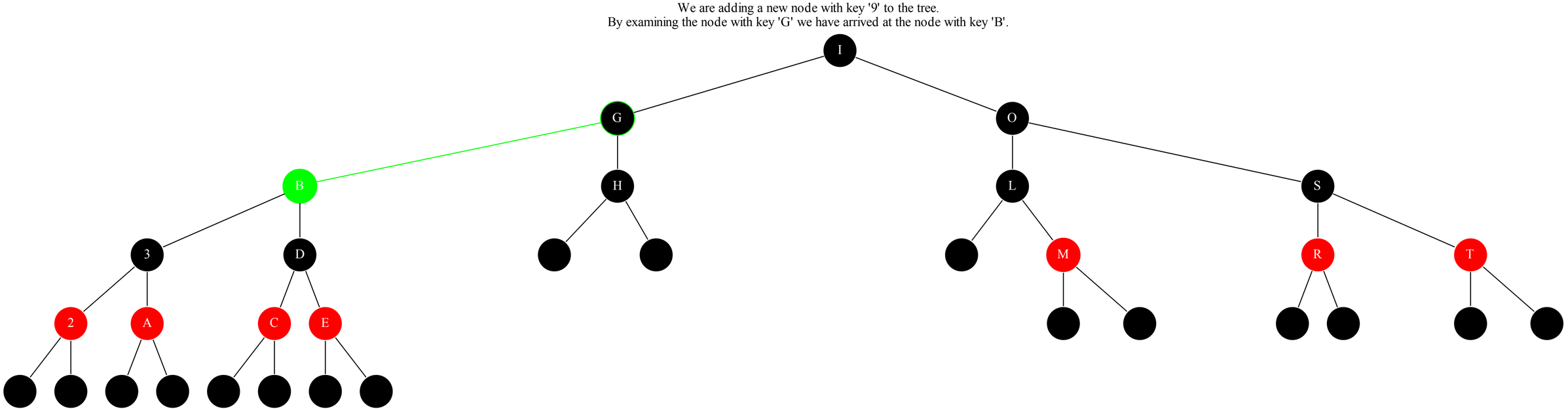


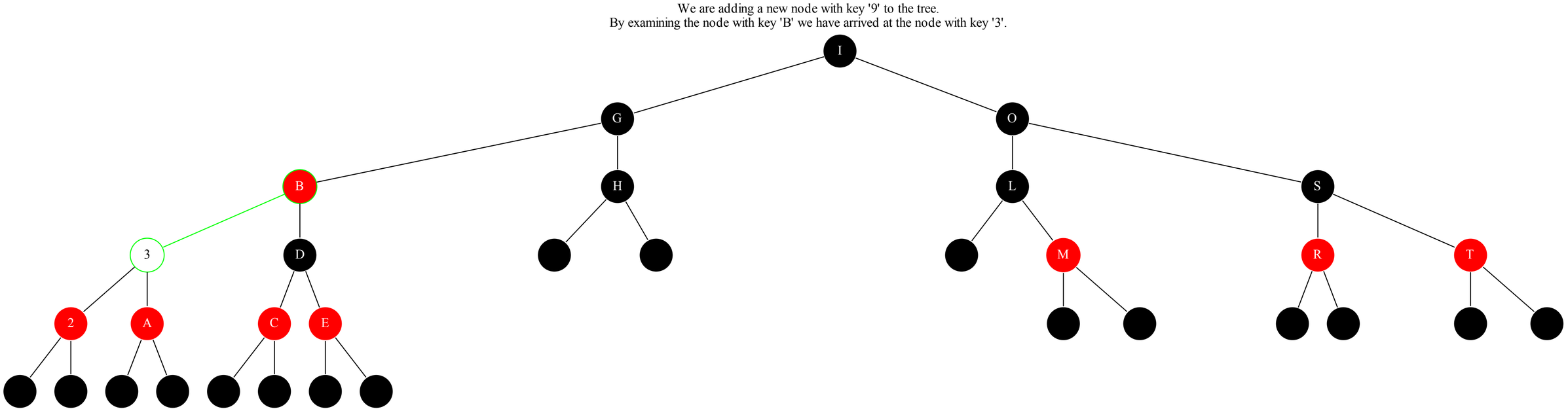
Since all of the required red-black properties are satisfied, we have arrived to a correctly formed red-black tree.

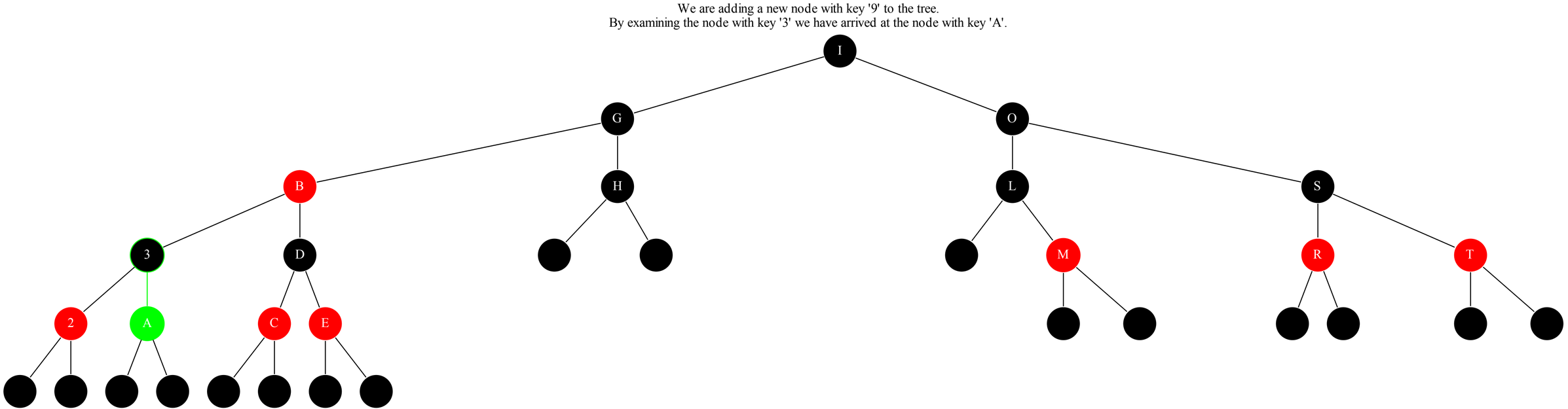


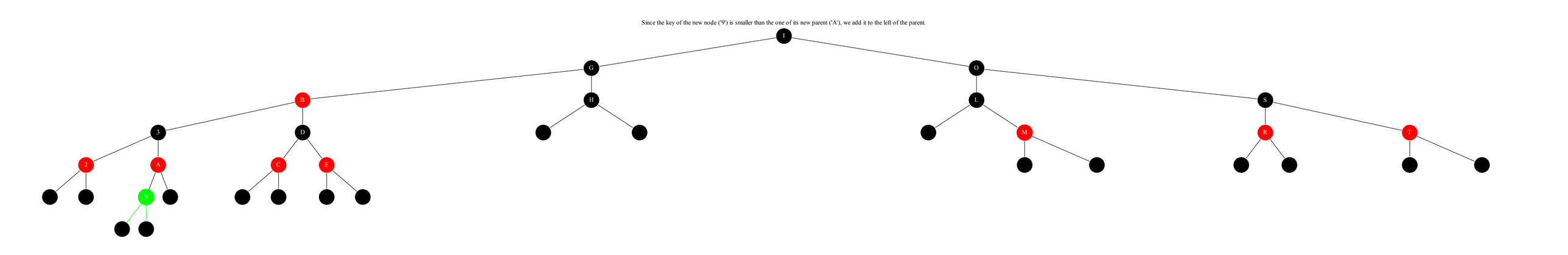

We are adding a new node with key '9' to the tree.
The first step is to find a leaf (Nill), which the new node will replace.
We begin by examining the root (key 'I').
I
G
O
B
H
L
S
3
D
M
R
T
2
A
C
E

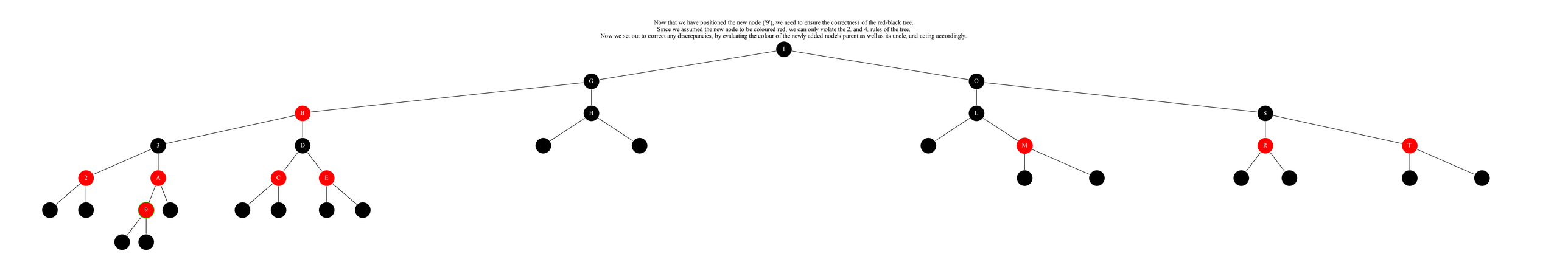

We are adding a new node with key '9' to the tree.
By examining the node with key 'I' we have arrived at the node with key 'G'.
I
G
O
B
H
L
S
3
D
M
R
T
2
A
C
E

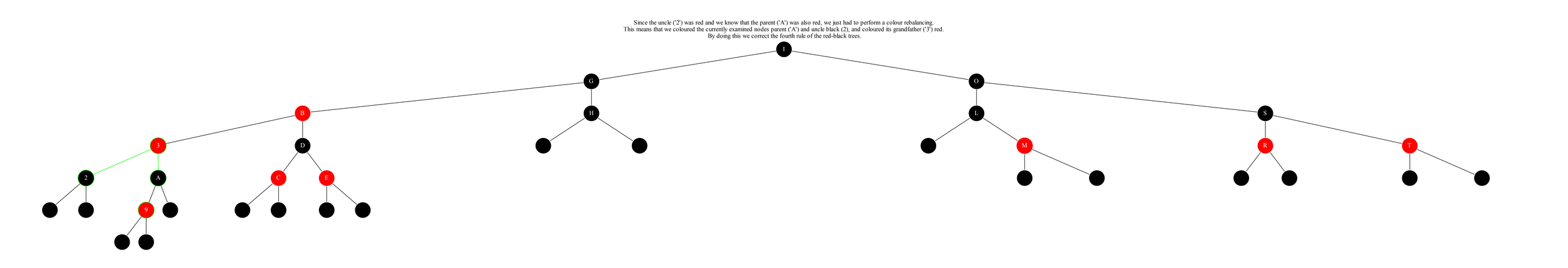

We are adding a new node with key '9' to the tree.
By examining the node with key 'G' we have arrived at the node with key 'B'.
I
G
O
B
H
L
S
3
D
M
R
T
2
A
C
E


We are adding a new node with key '9' to the tree.
By examining the node with key 'B' we have arrived at the node with key '3'.
I
G
O
B
H
L
S
3
D
M
R
T
2
A
C
E

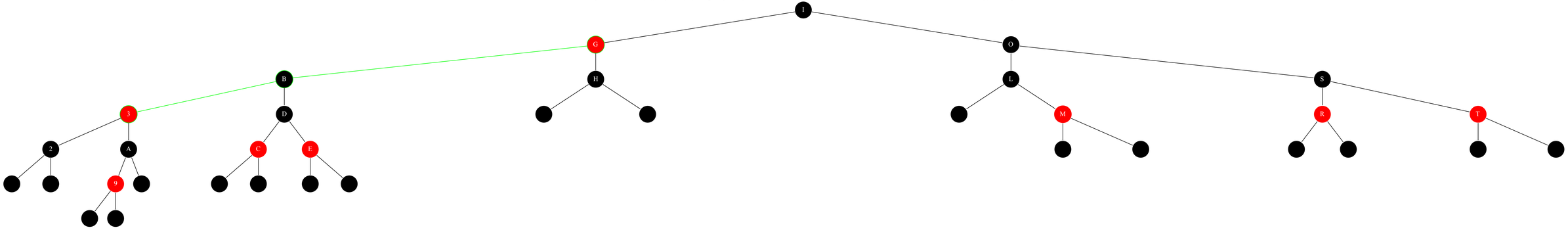

We are adding a new node with key '9' to the tree.
By examining the node with key '3' we have arrived at the node with key 'A'.
I
G
O
B
H
L
S
3
D
M
R
T
2
A
C
E


Since the key of the new node ('9') is smaller than the one of its new parent ('A'), we add it to the left of the parent.
I
G
O
B
H
L
S
3
D
M
R
T
2
A
C
E
9

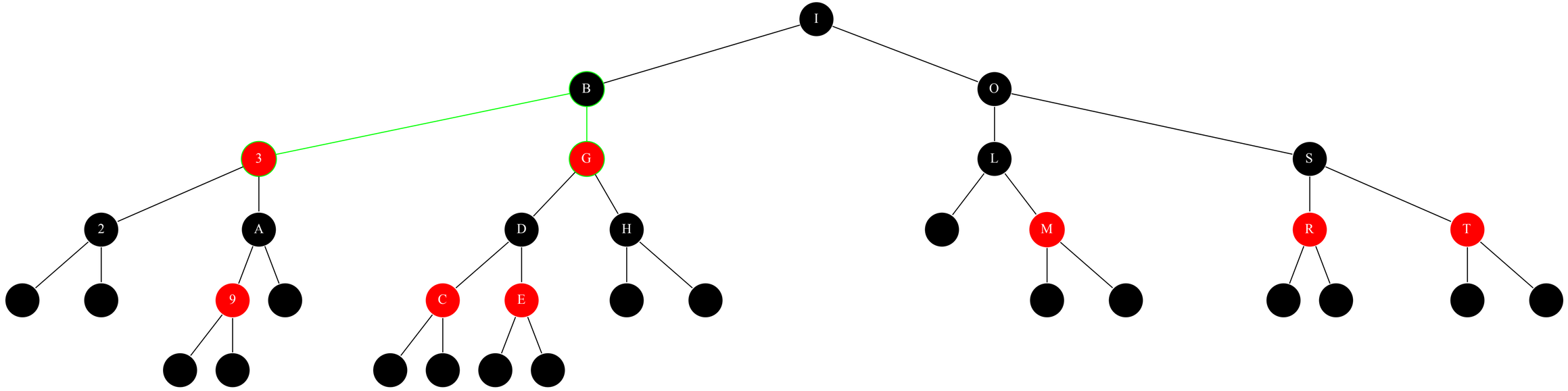

Now that we have positioned the new node ('9'), we need to ensure the correctness of the red-black tree.
Since we assumed the new node to be coloured red, we can only violate the 2. and 4. rules of the tree.
Now we set out to correct any discrepancies, by evaluating the colour of the newly added node's parent as well as its uncle, and acting accordingly.
I
G
O
B
H
L
S
3
D
M
R
T
2
A
C
E
9


Since the uncle ('2') was red and we know that the parent ('A') was also red, we just had to perform a colour rebalancing.
This means that we coloured the currently examined nodes parent ('A') and uncle black (2), and coloured its grandfather ('3') red.
By doing this we correct the fourth rule of the red-black trees.
I
G
O
B
H
L
S
3
D
M
R
T
2
A
C
E
9

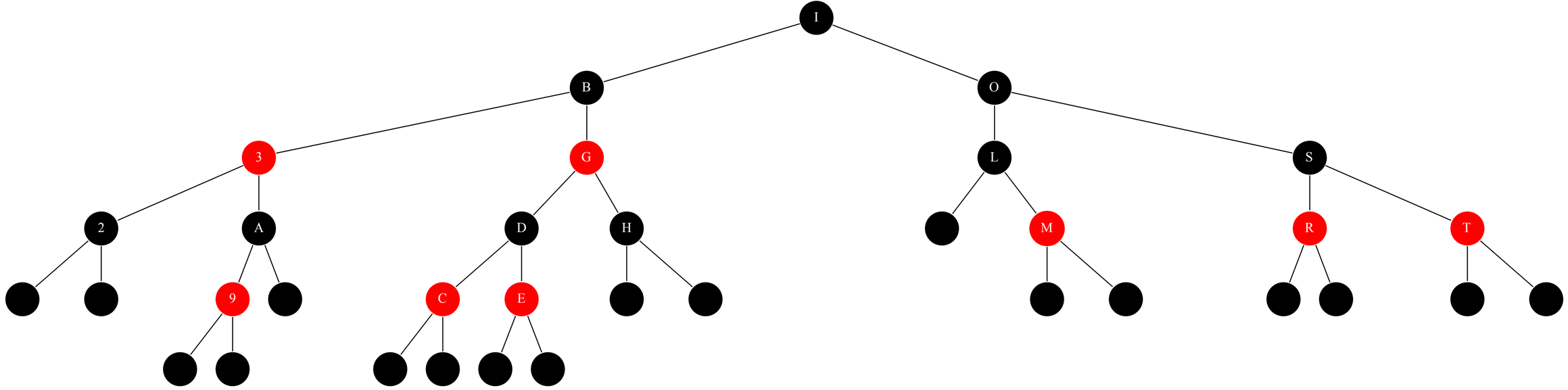
Since the currently examined node ('3'), its parent ('B') and grandparent ('G') are 'in-line', we now had to recolour them, so that when we rotate them in the next step, we will get a correct tree.

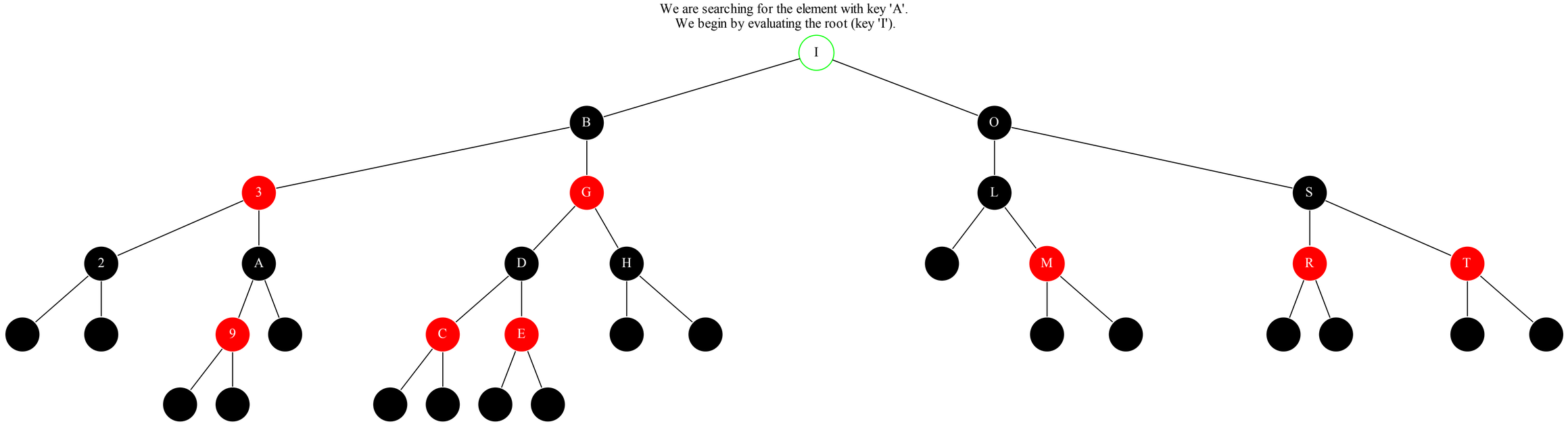


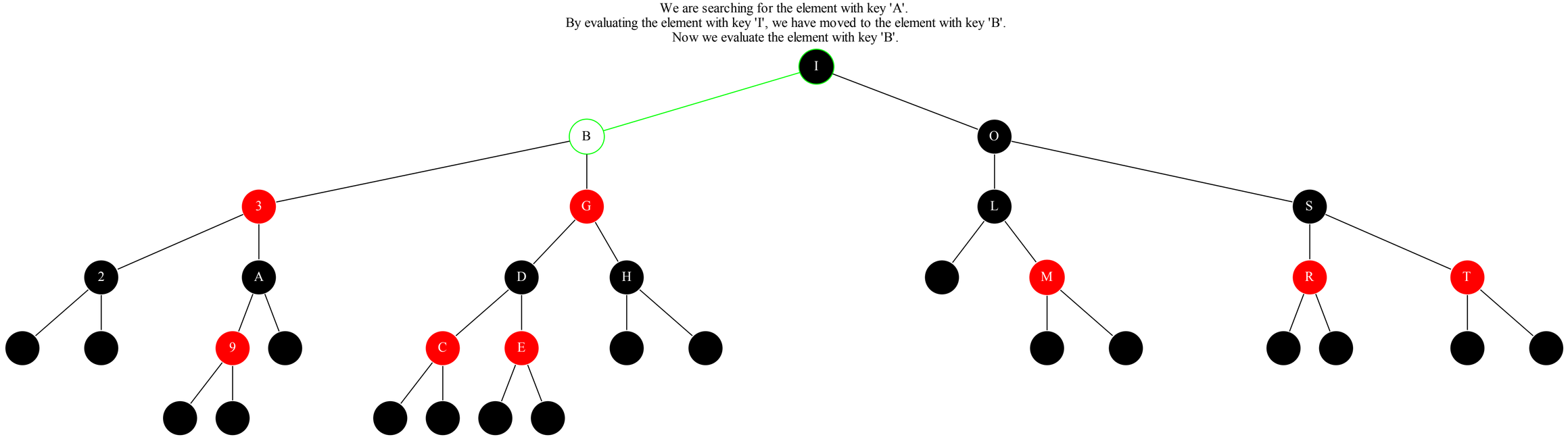
Since the currently examined node ('3'), its parent ('B') and grandparent ('3') are 'in-line' and properly coloured now, just one more rotation gives a correct red-black tree.

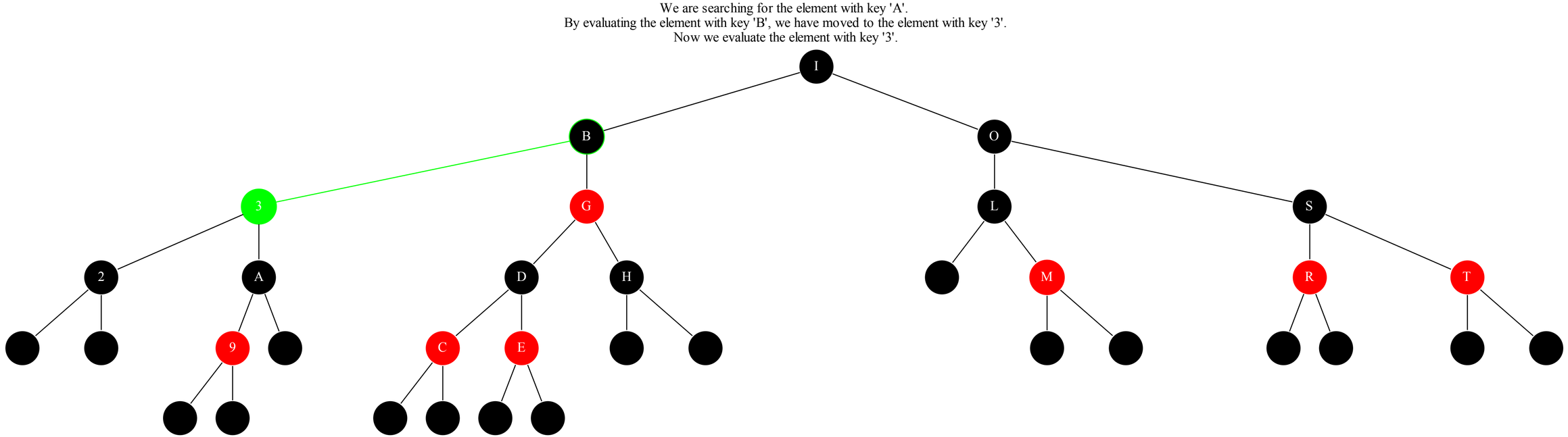


Since all of the required red-black properties are satisfied, we have arrived to a correctly formed red-black tree.

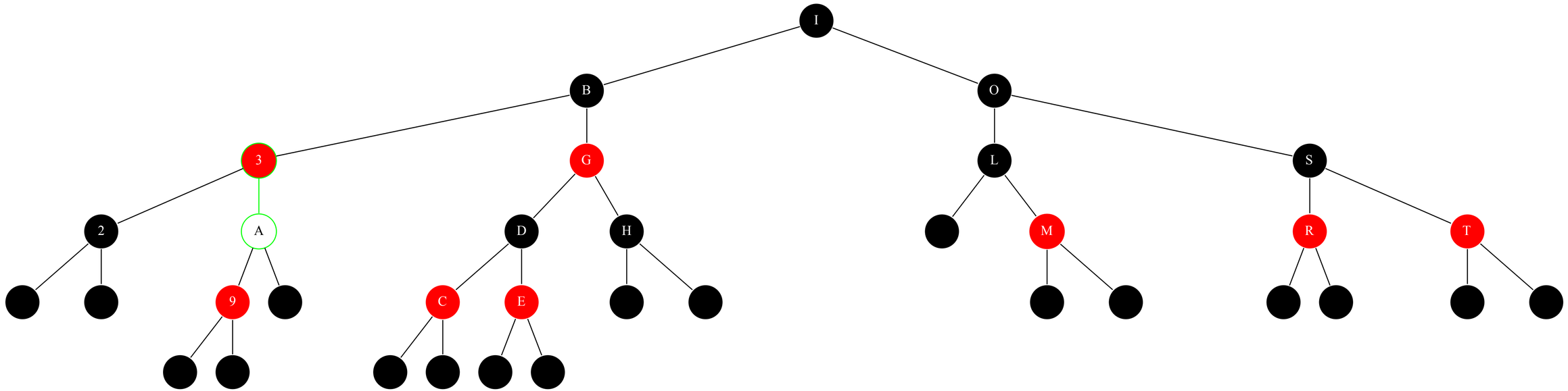


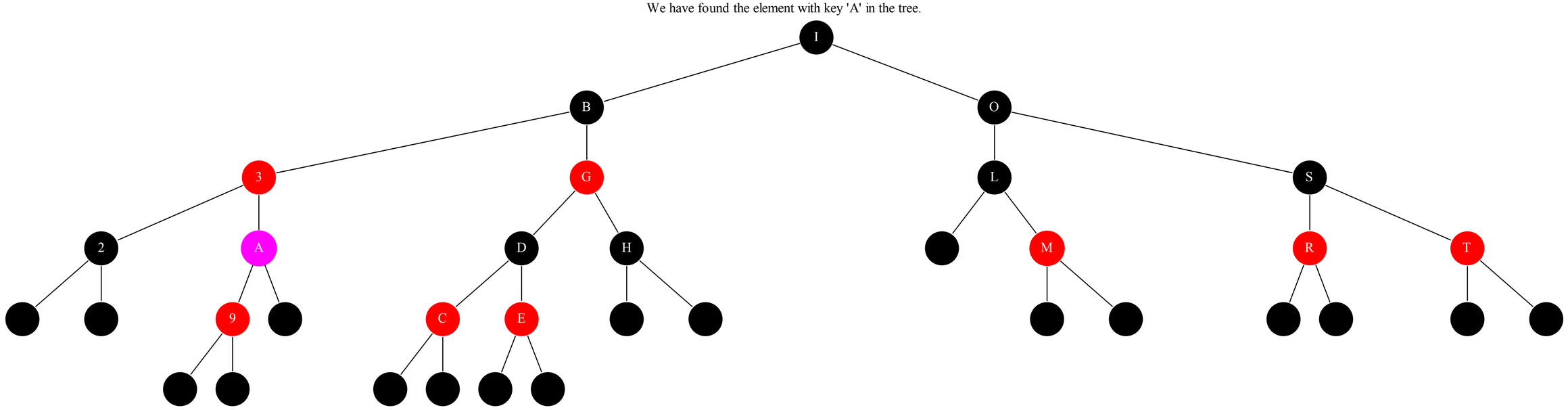

We are searching for the element with key 'A'.
We begin by evaluating the root (key 'I').
I
B
O
3
G
L
S
2
A
D
H
M
R
T
9
C
E

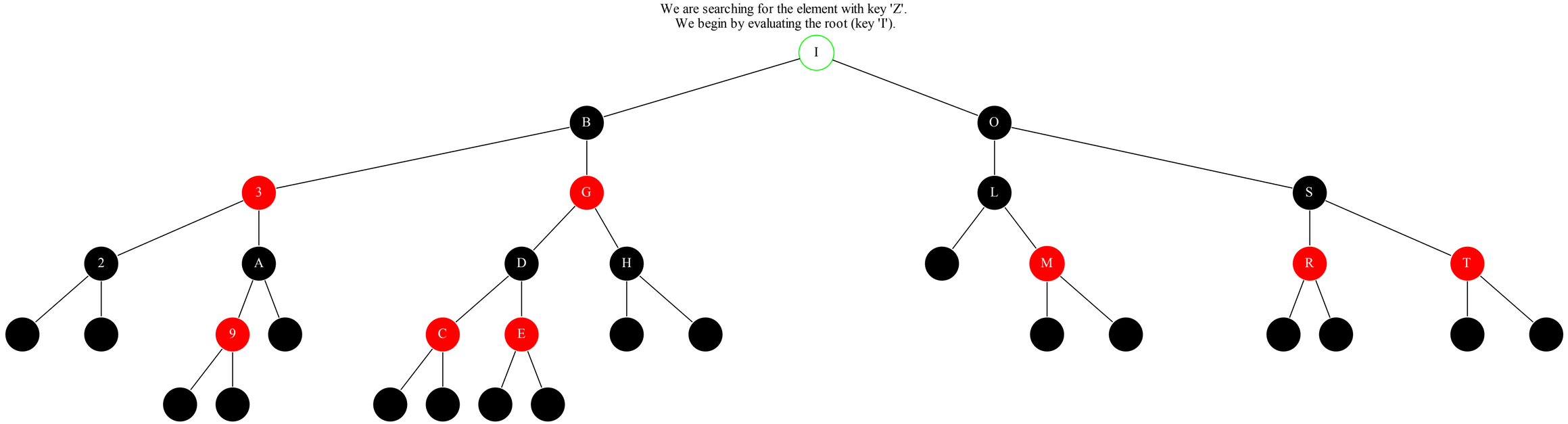

We are searching for the element with key 'A'.
By evaluating the element with key 'I', we have moved to the element with key 'B'.
Now we evaluate the element with key 'B'.
I
B
O
3
G
L
S
2
A
D
H
M
R
T
9
C
E

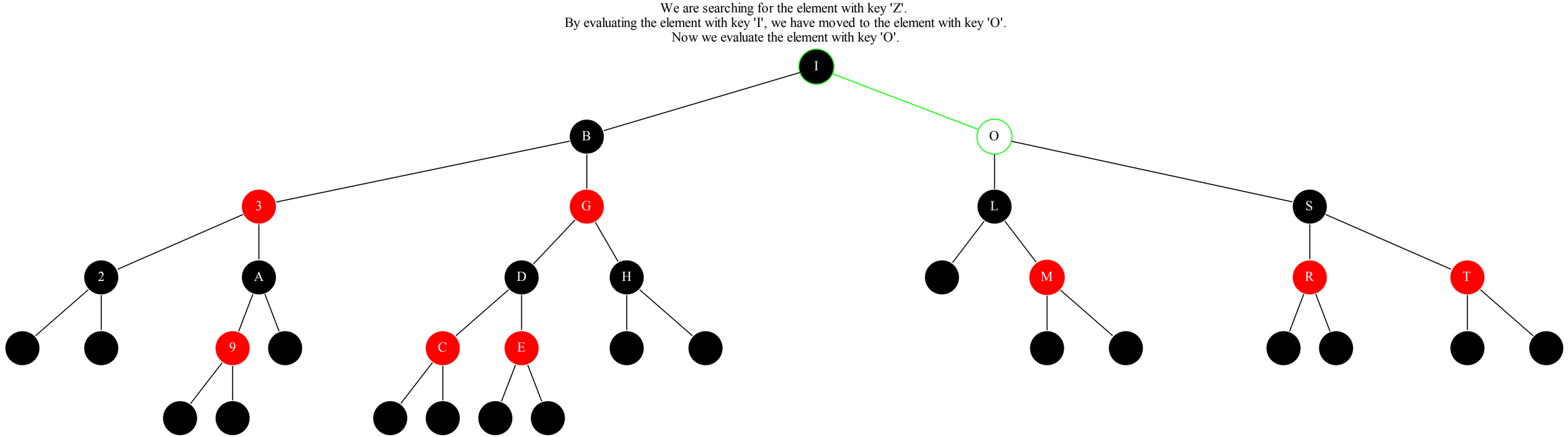

We are searching for the element with key 'A'.
By evaluating the element with key 'B', we have moved to the element with key '3'.
Now we evaluate the element with key '3'.
I
B
O
3
G
L
S
2
A
D
H
M
R
T
9
C
E

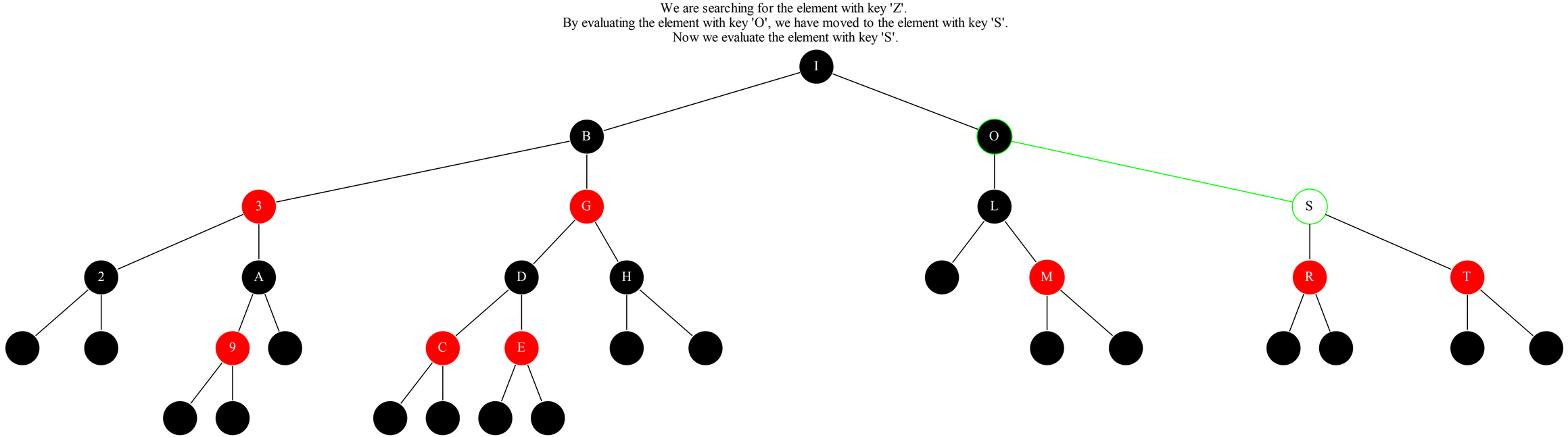
We are searching for the element with key 'A'.
By evaluating the element with key '3', we have moved to the element with key 'A'.
Now we evaluate the element with key 'A'.



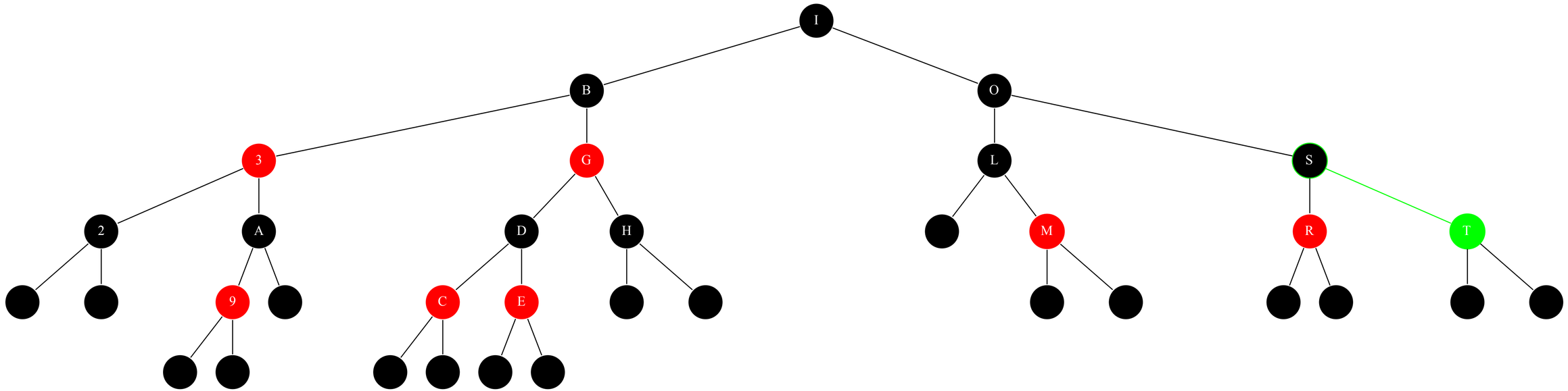

We have found the element with key 'A' in the tree.
I
B
O
3
G
L
S
2
A
D
H
M
R
T
9
C
E

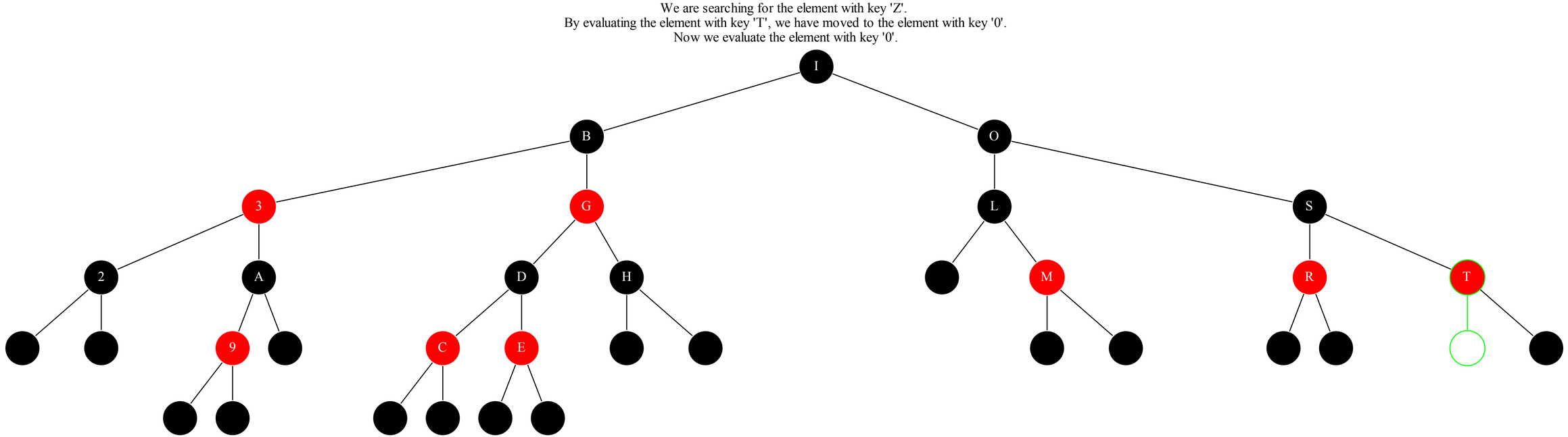

We are searching for the element with key 'Z'.
We begin by evaluating the root (key 'I').
I
B
O
3
G
L
S
2
A
D
H
M
R
T
9
C
E

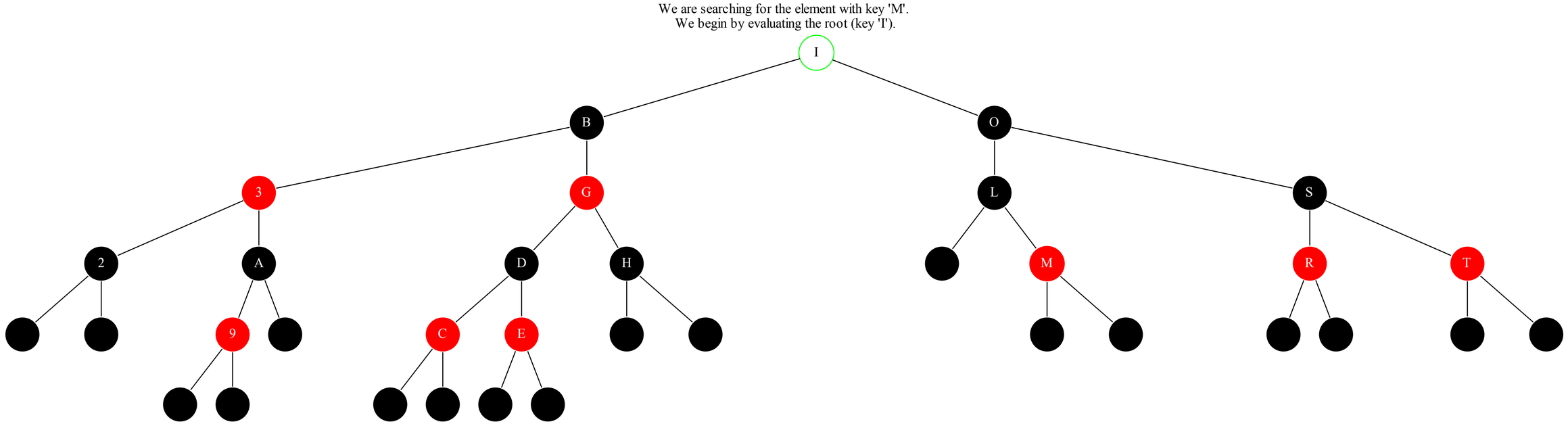

We are searching for the element with key 'Z'.
By evaluating the element with key 'I', we have moved to the element with key 'O'.
Now we evaluate the element with key 'O'.
I
B
O
3
G
L
S
2
A
D
H
M
R
T
9
C
E

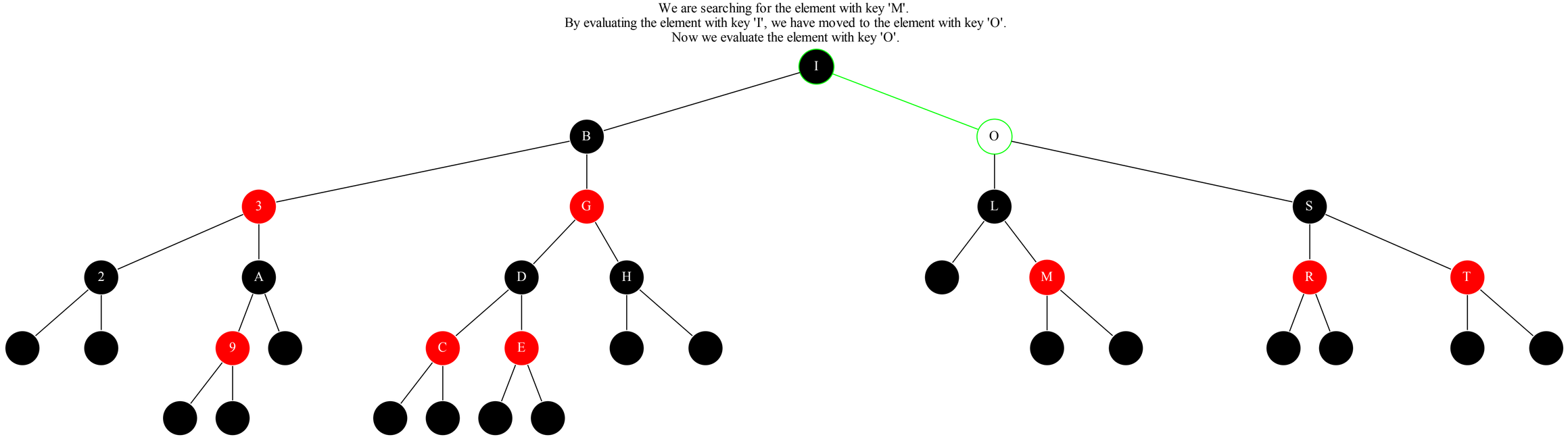

We are searching for the element with key 'Z'.
By evaluating the element with key 'O', we have moved to the element with key 'S'.
Now we evaluate the element with key 'S'.
I
B
O
3
G
L
S
2
A
D
H
M
R
T
9
C
E

We are searching for the element with key 'Z'.
By evaluating the element with key 'S', we have moved to the element with key 'T'.
Now we evaluate the element with key 'T'.

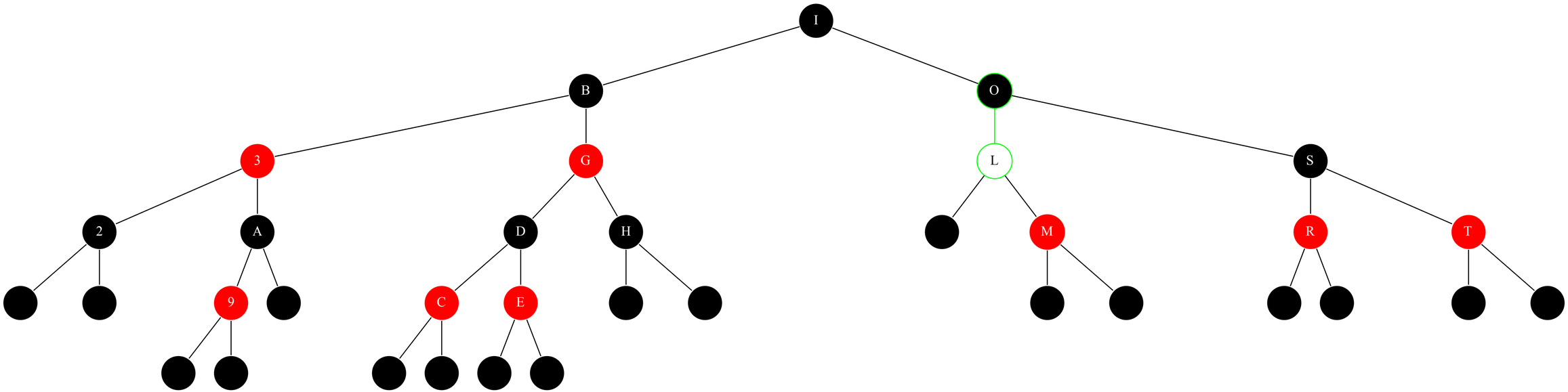



We are searching for the element with key 'Z'.
By evaluating the element with key 'T', we have moved to the element with key '0'.
Now we evaluate the element with key '0'.
I
B
O
3
G
L
S
2
A
D
H
M
R
T
9
C
E

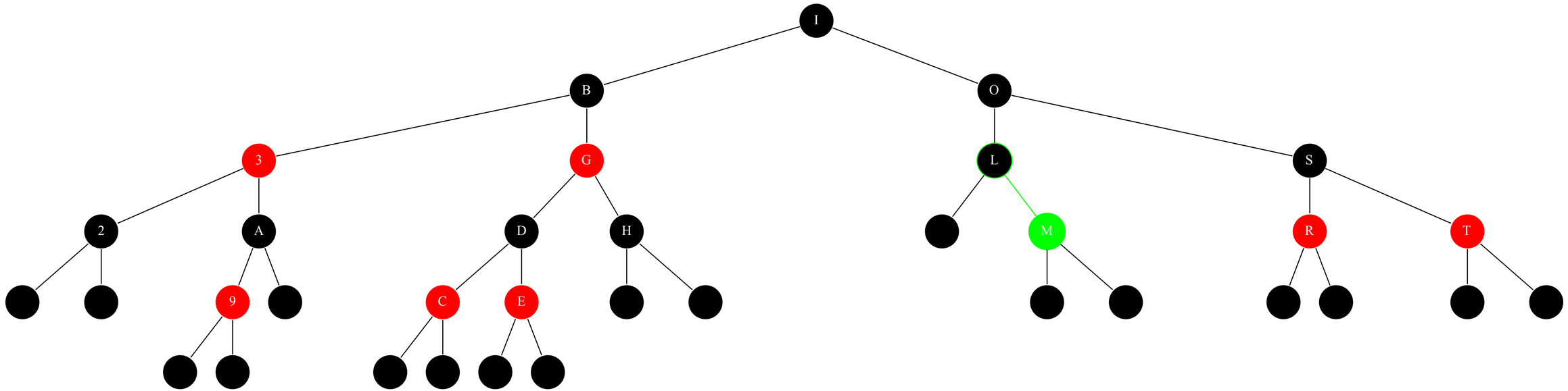

We are searching for the element with key 'M'.
We begin by evaluating the root (key 'I').
I
B
O
3
G
L
S
2
A
D
H
M
R
T
9
C
E

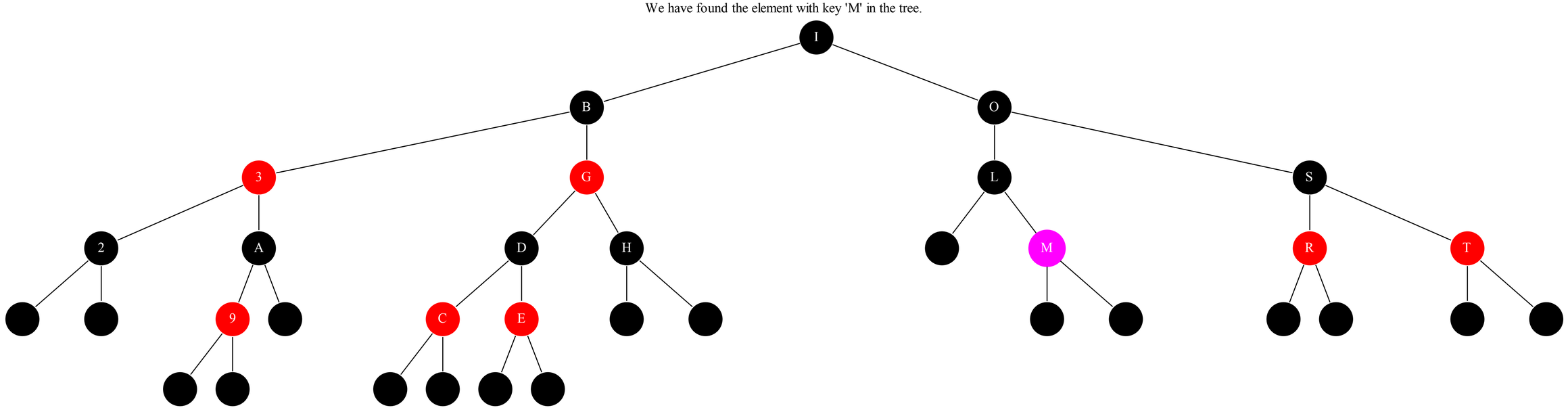

We are searching for the element with key 'M'.
By evaluating the element with key 'I', we have moved to the element with key 'O'.
Now we evaluate the element with key 'O'.
I
B
O
3
G
L
S
2
A
D
H
M
R
T
9
C
E

We are searching for the element with key 'M'.
By evaluating the element with key 'O', we have moved to the element with key 'L'.
Now we evaluate the element with key 'L'.



We are searching for the element with key 'M'.
By evaluating the element with key 'L', we have moved to the element with key 'M'.
Now we evaluate the element with key 'M'.




We have found the element with key 'M' in the tree.
I
B
O
3
G
L
S
2
A
D
H
M
R
T
9
C
E

We have found the parent of the node with the key 'M', because we have previously found the node in question, and its parent exists.