Augmenting Data Structures

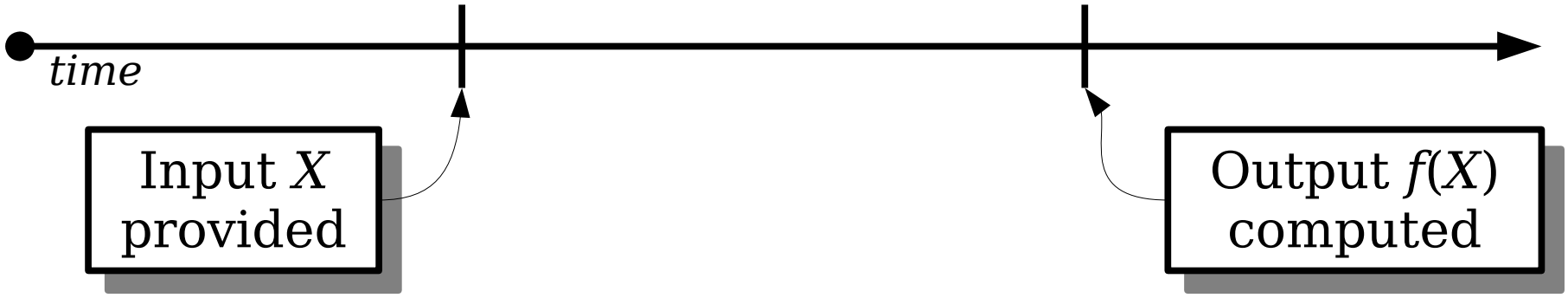
Dynamic Problems

Classical Algorithms

- The “classical” algorithms model goes something like this:

Given some input X, compute some interesting function f(X).

- The input X is provided up front, and only a single answer is produced.

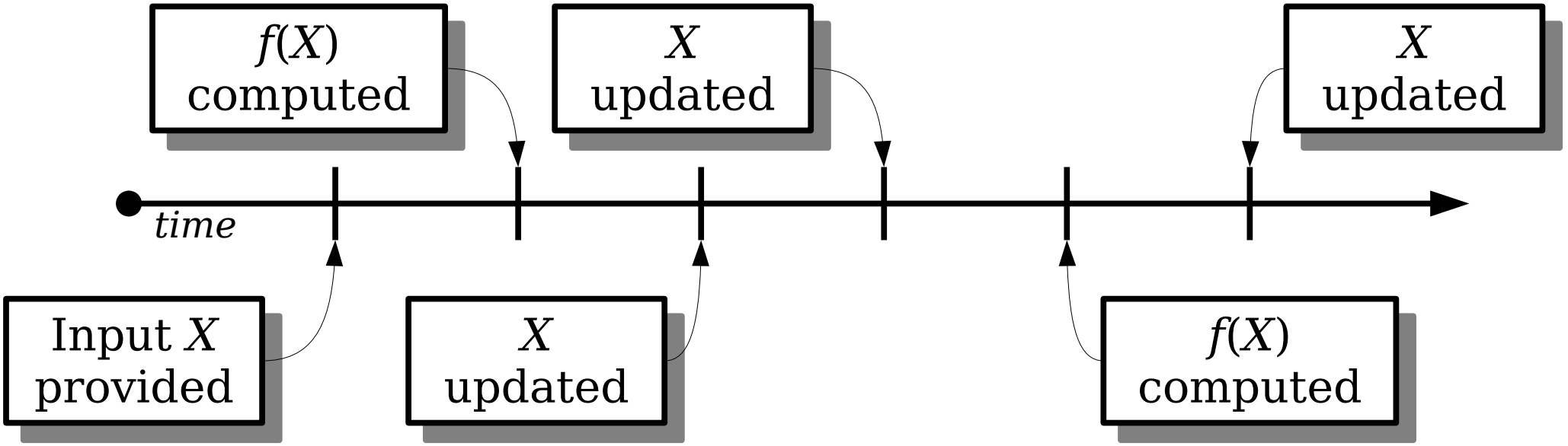


Dynamic Problems

- ***Dynamic versions*** of problems are framed like this:

  Given an input X that can change in fixed ways, maintain X while being able to compute f(X) efficiently at any point in time.

- These problems are typically harder to solve efficiently than the “classical” static versions.

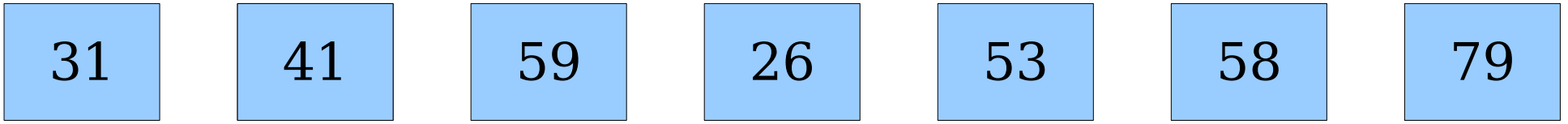


Dynamic Selection

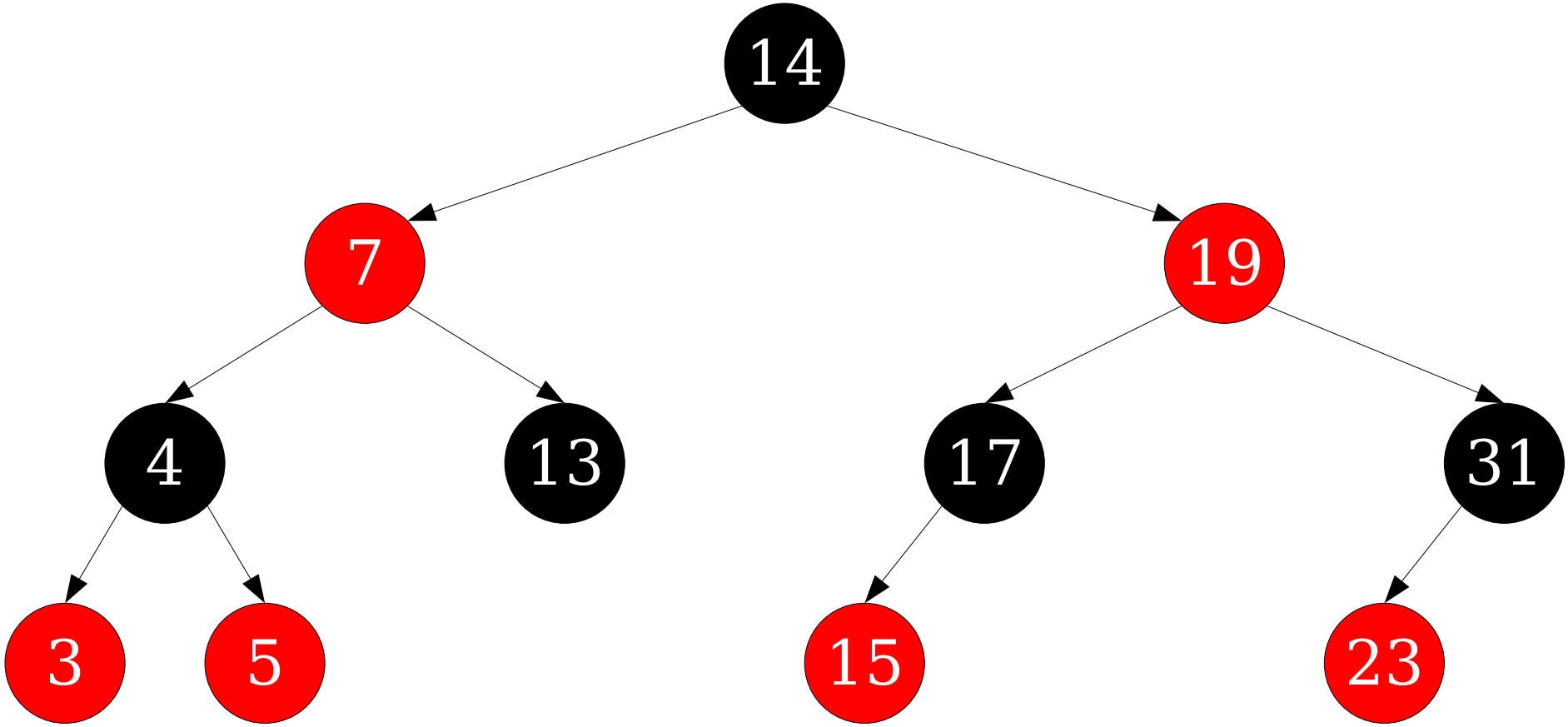
- The ***selection*** problem is the following:

  Given a list of distinct values and a number k, return the kth-smallest value.

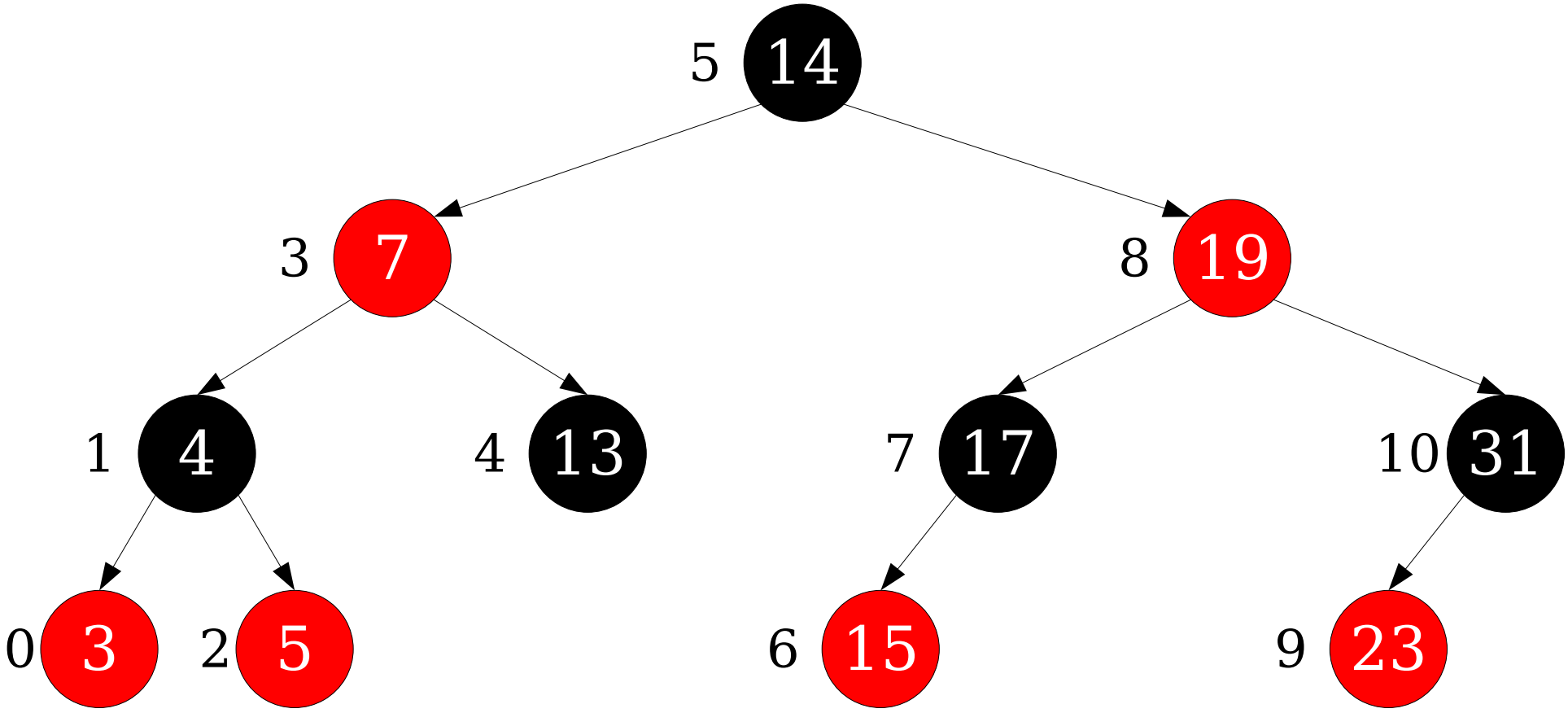
- In the static case, where the data set is fixed in advance and k is known, we can solve this in time $O(n)$ using quickselect or the median-of-medians algorithm.
- ***Goal:*** Solve this problem efficiently when the data set is changing – that is, the underlying set of elements can have insertions and deletions intermixed with queries.



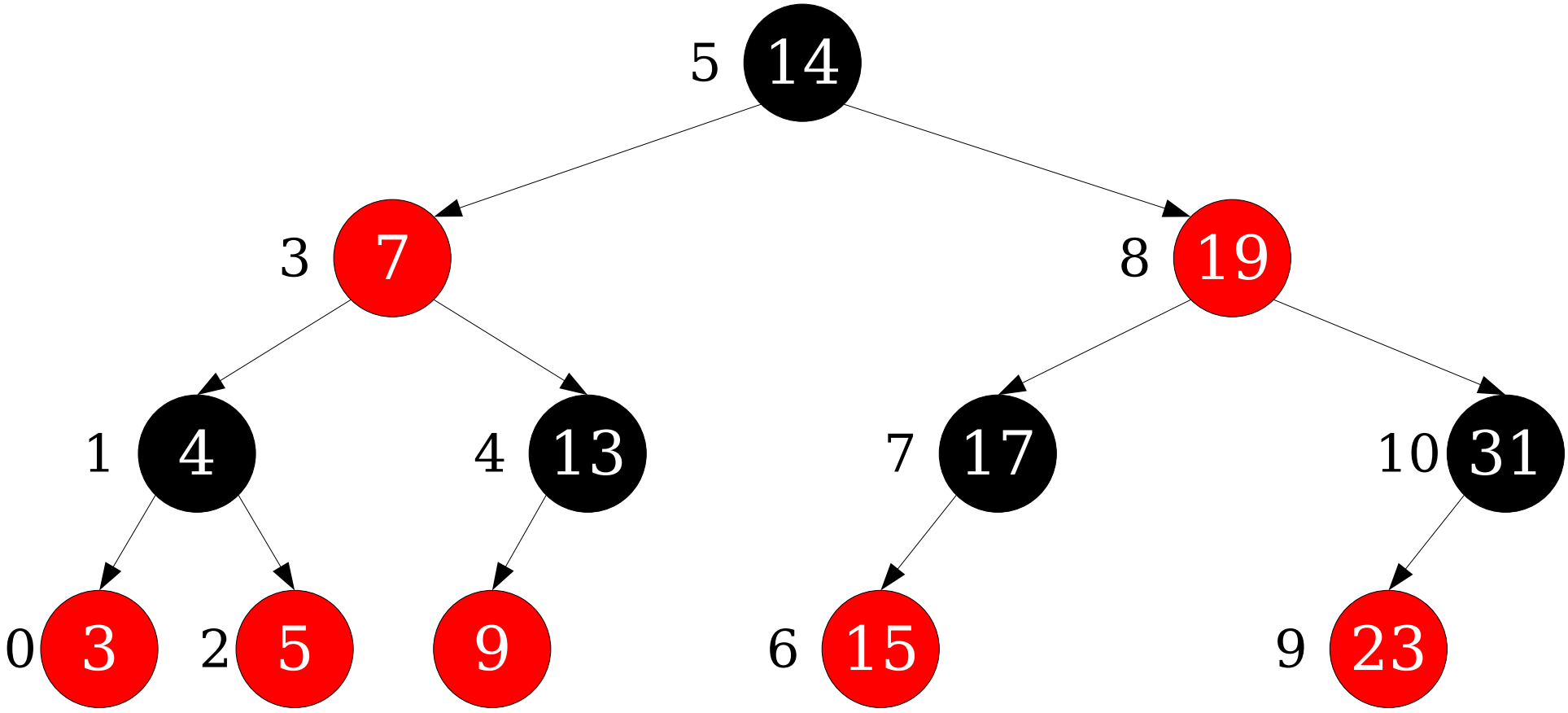
Dynamic Selection



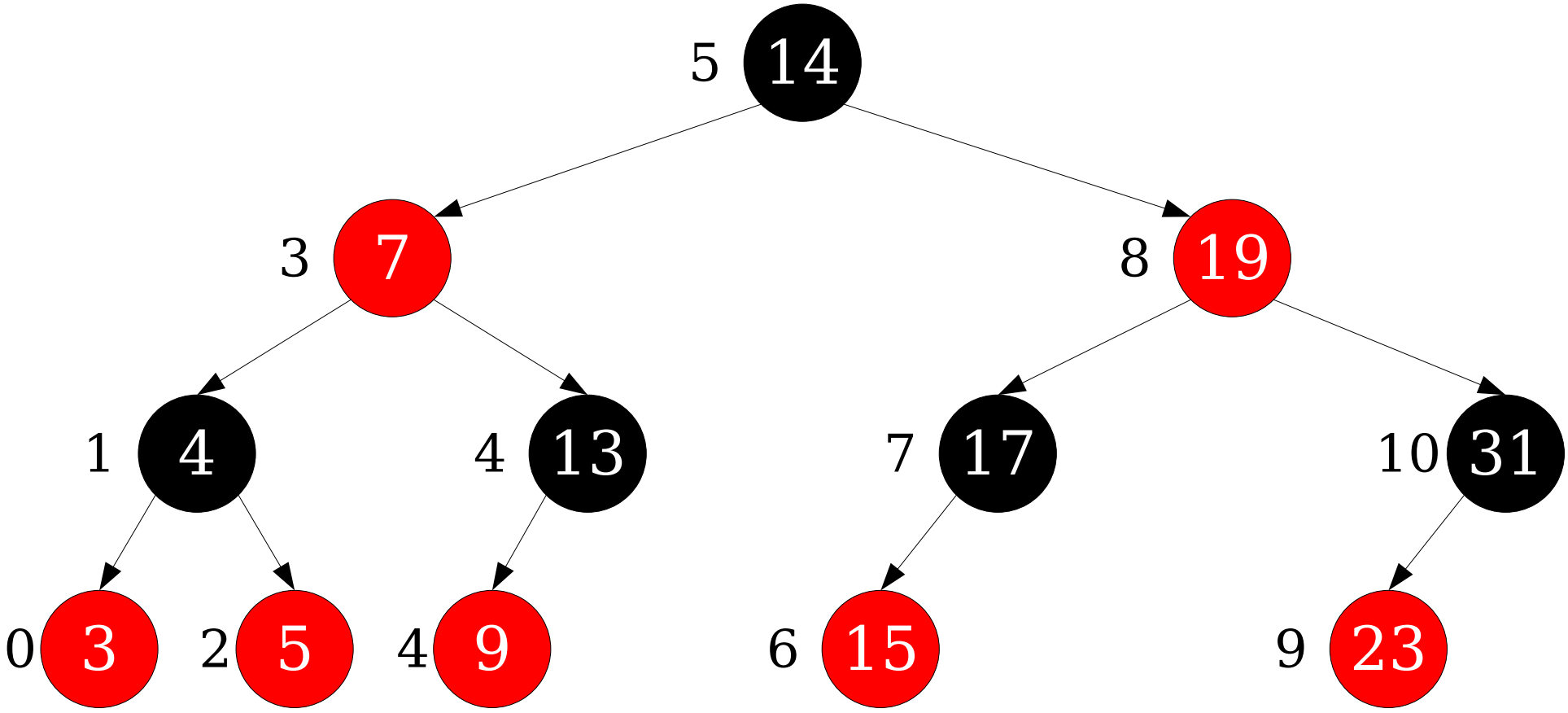
Dynamic Selection



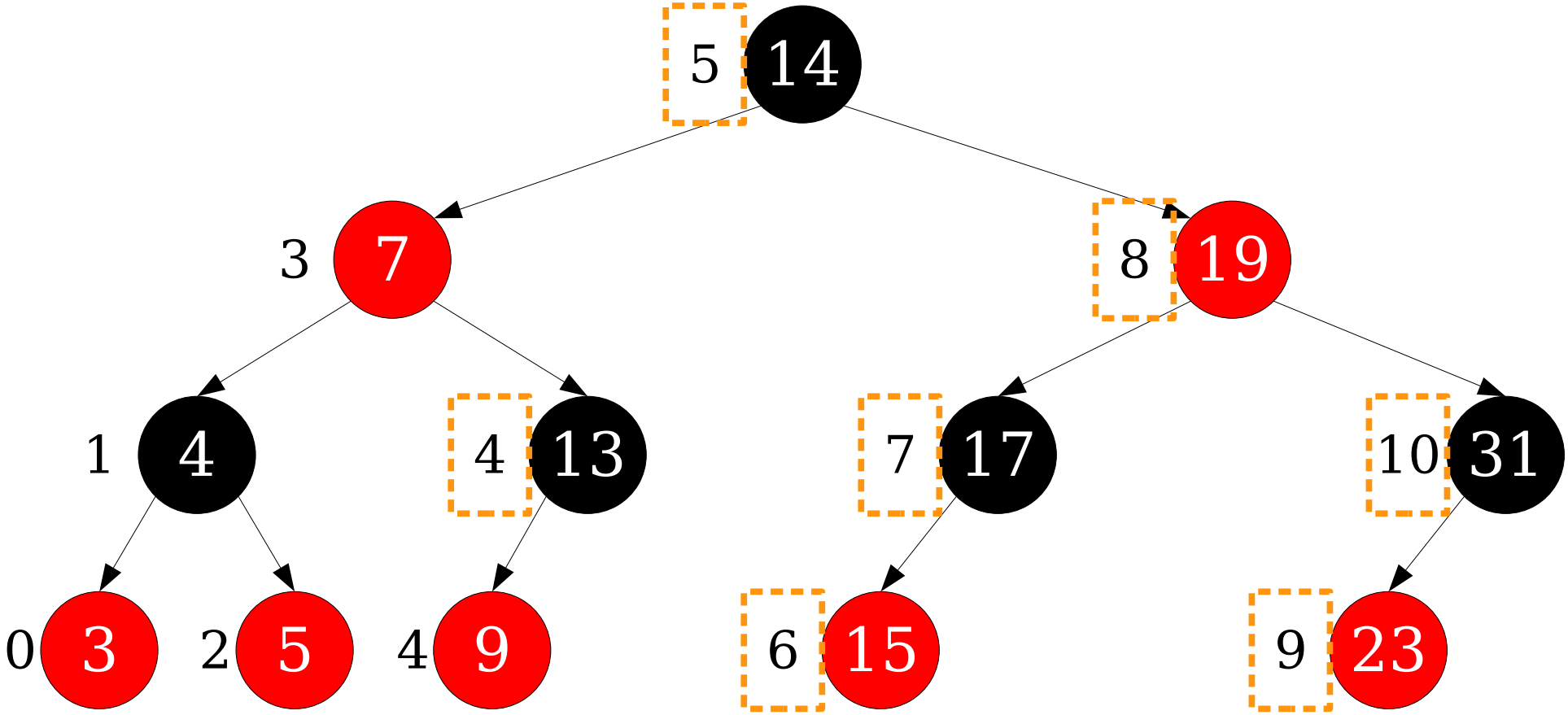
Dynamic Selection



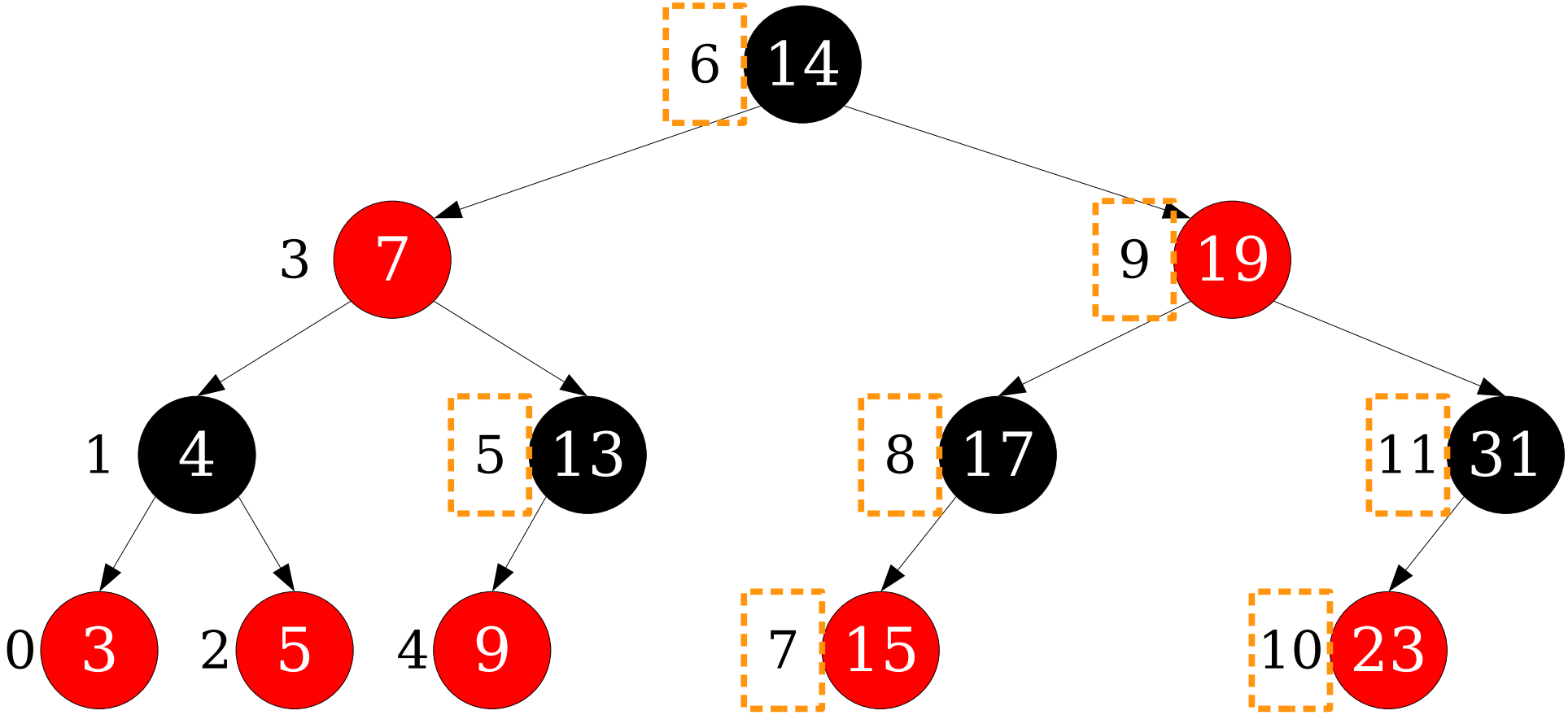
Dynamic Selection



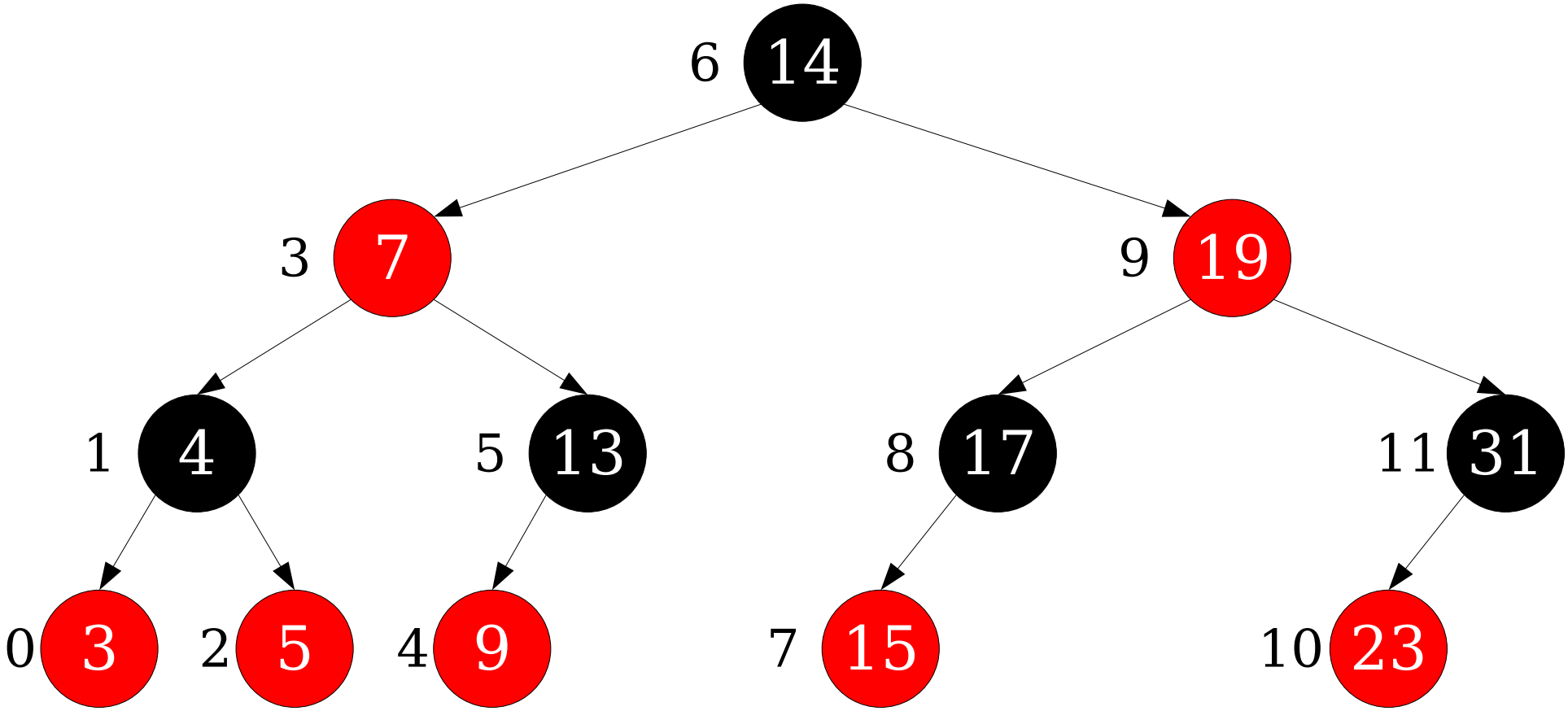
Dynamic Selection



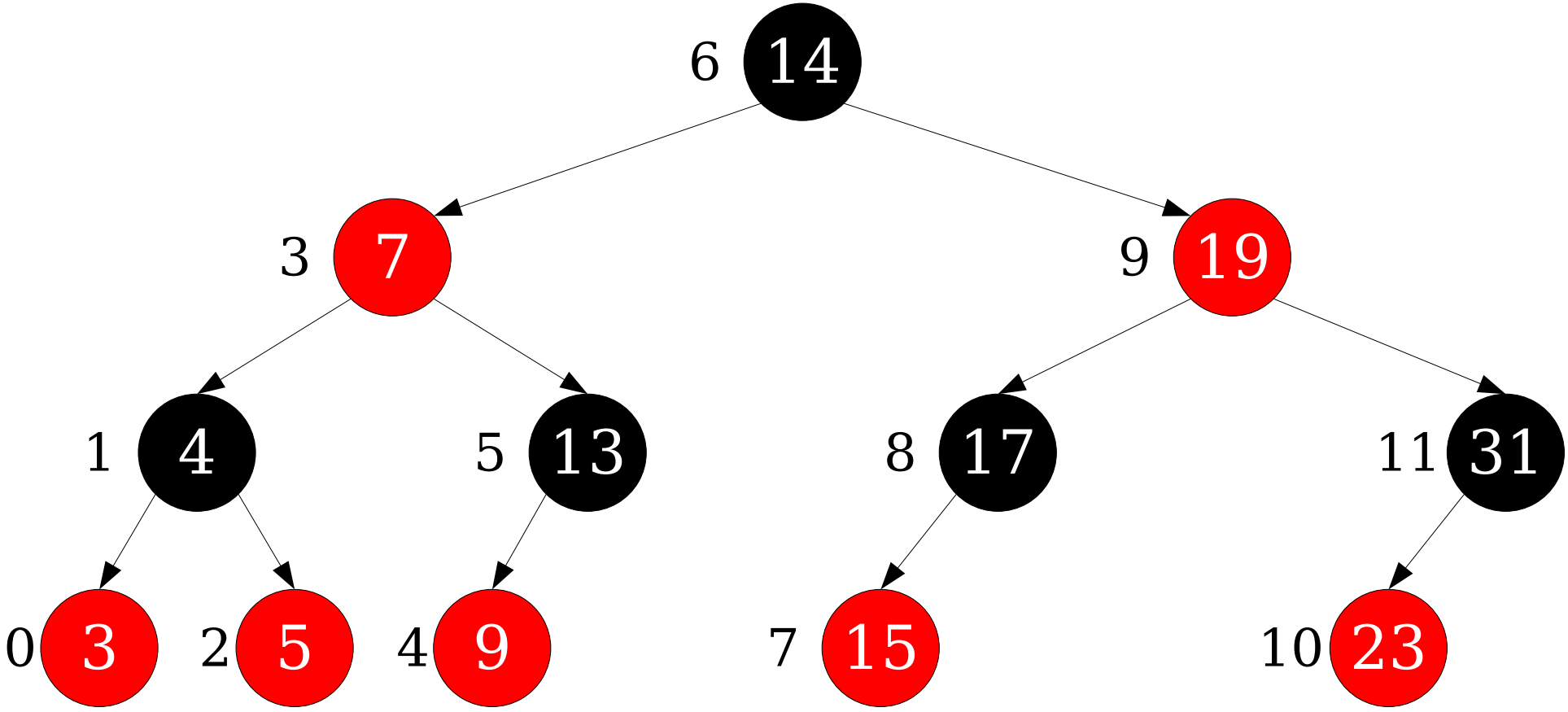
Dynamic Selection



Dynamic Selection



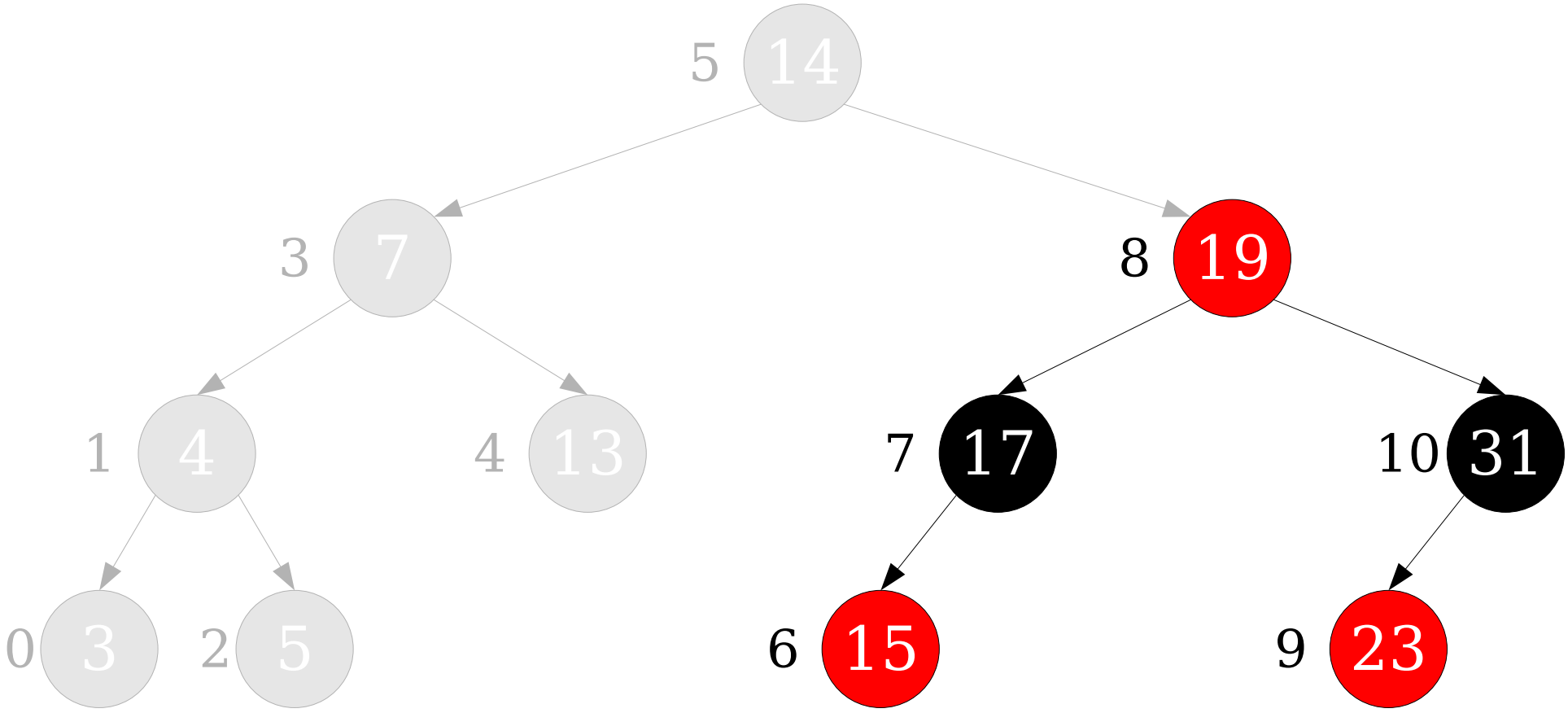
Dynamic Selection



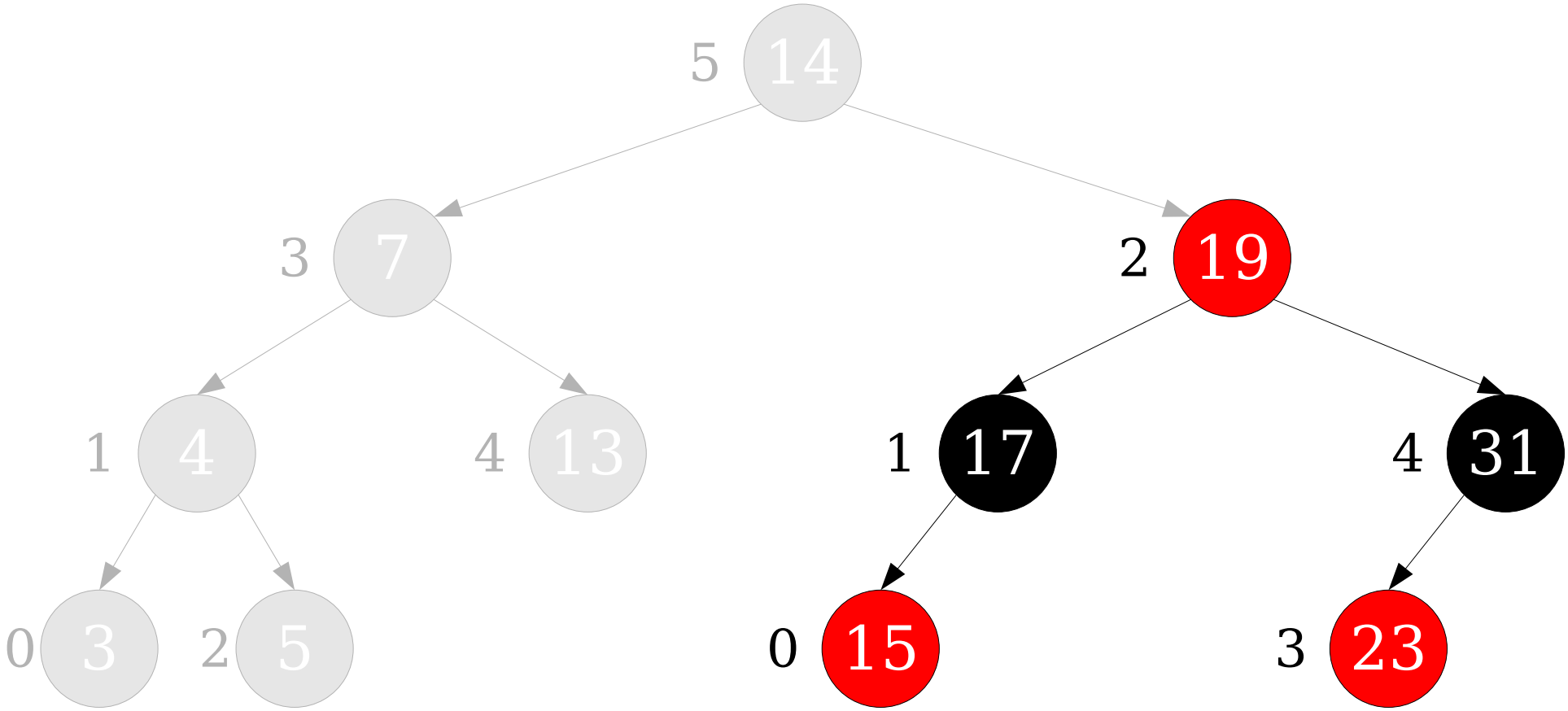
Problem: After inserting a new value, we may have to update $\Theta(n)$ values.

This is inherent in this solution route. These numbers track *global* properties of the tree.

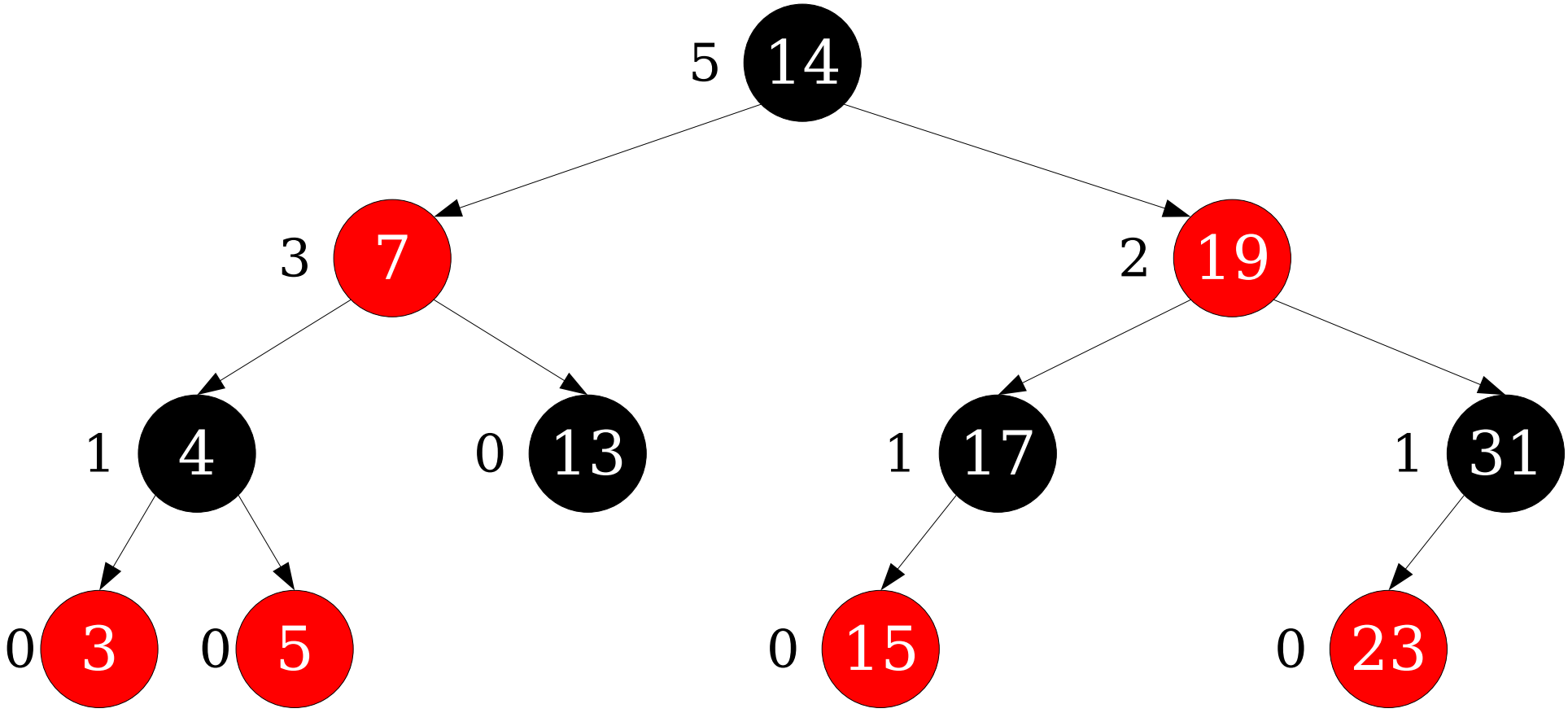
Dynamic Selection



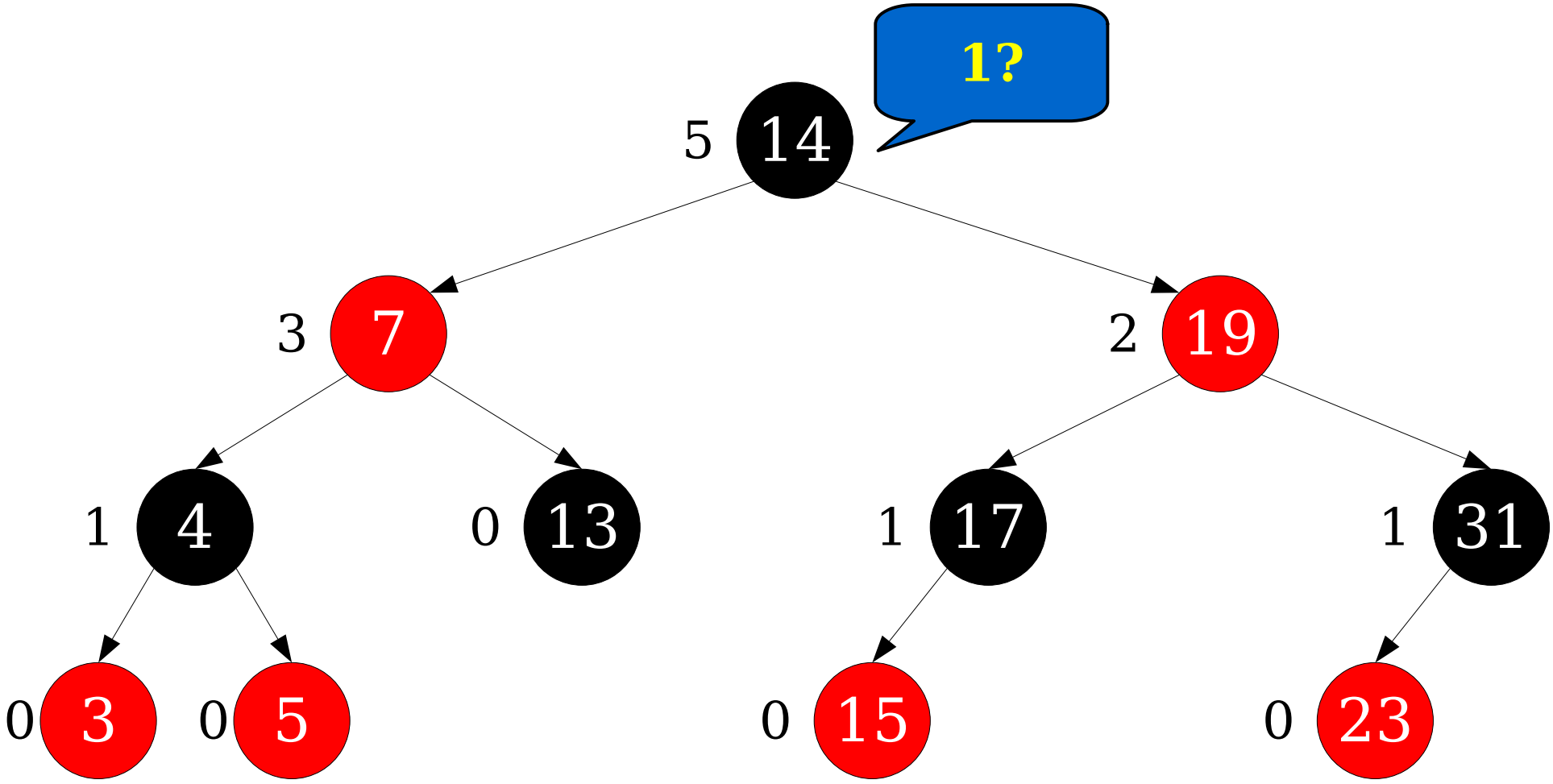
Dynamic Selection



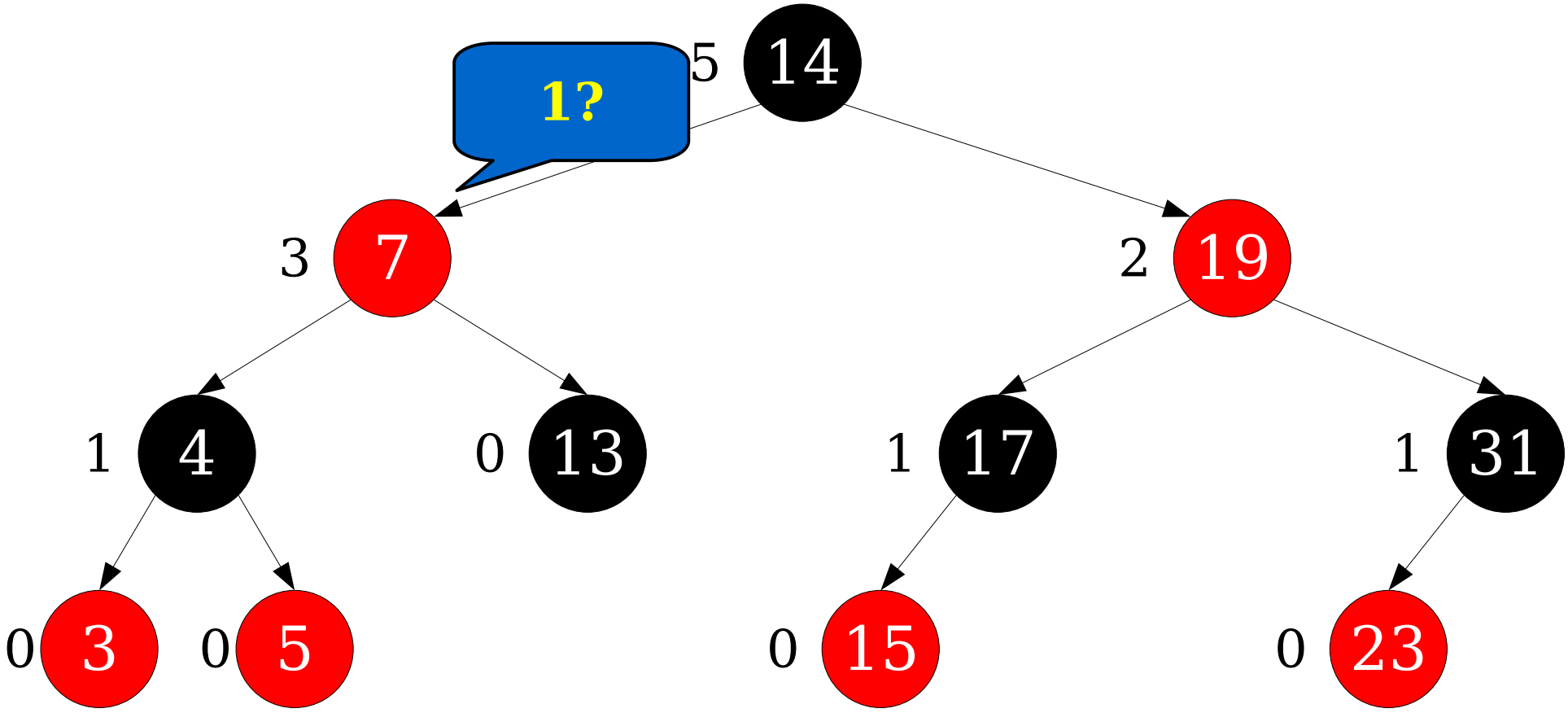
Dynamic Selection



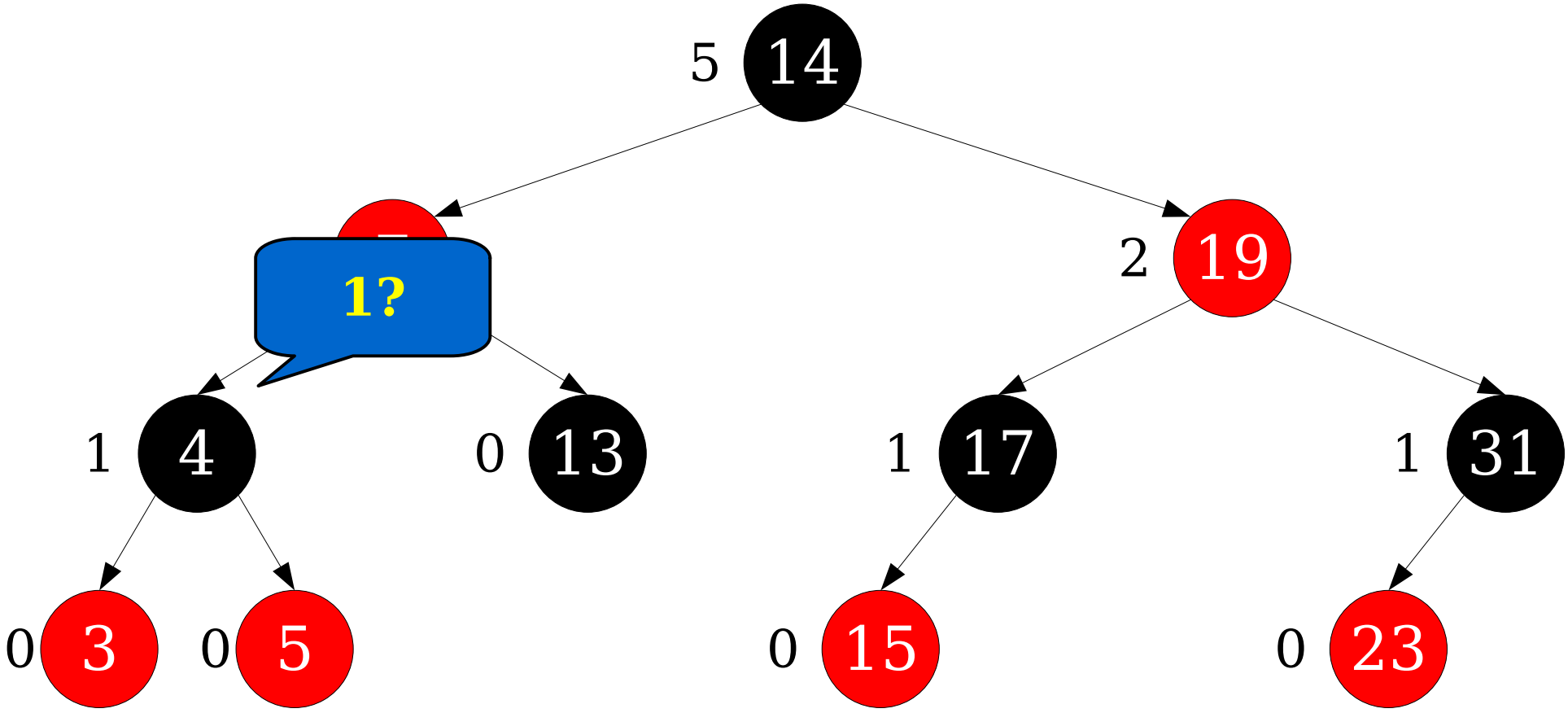
Dynamic Selection



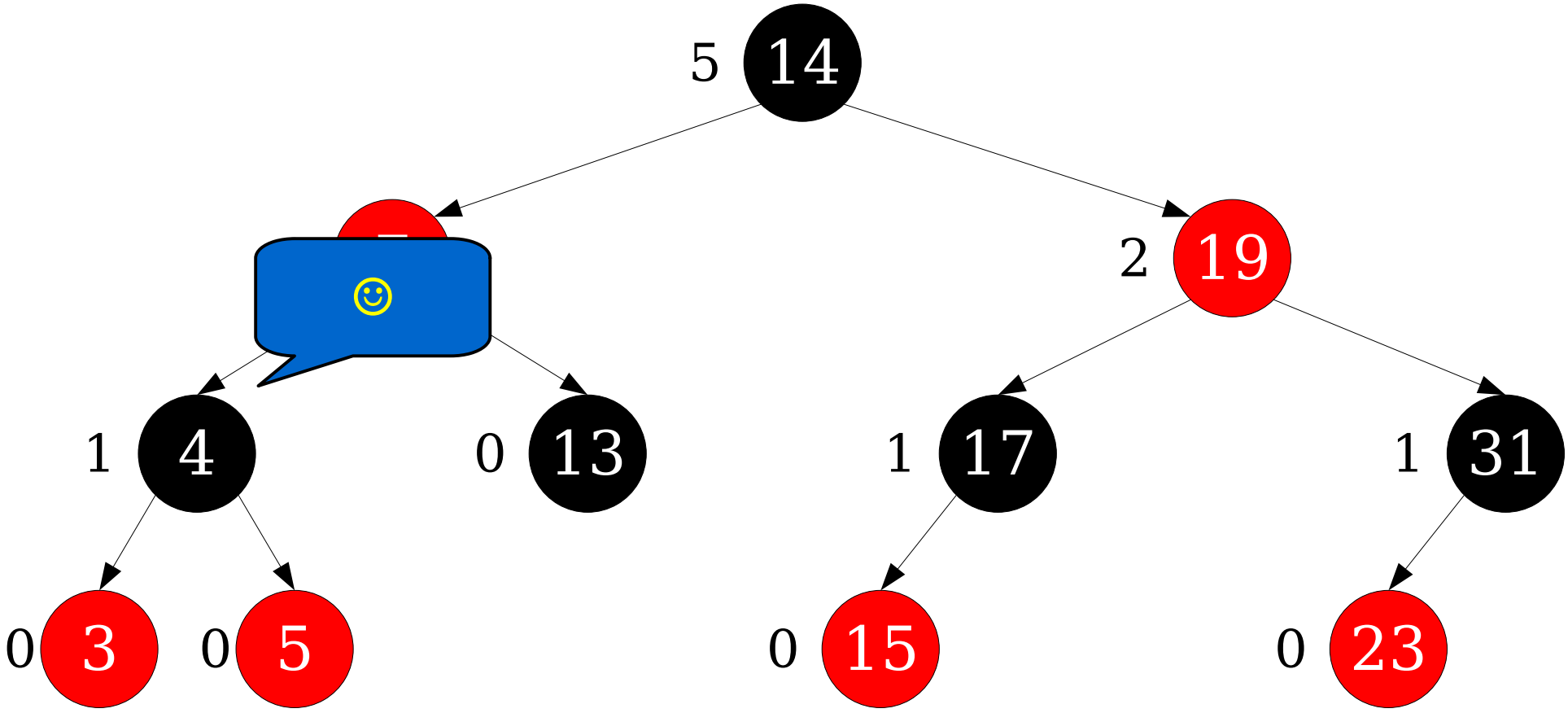
Dynamic Selection



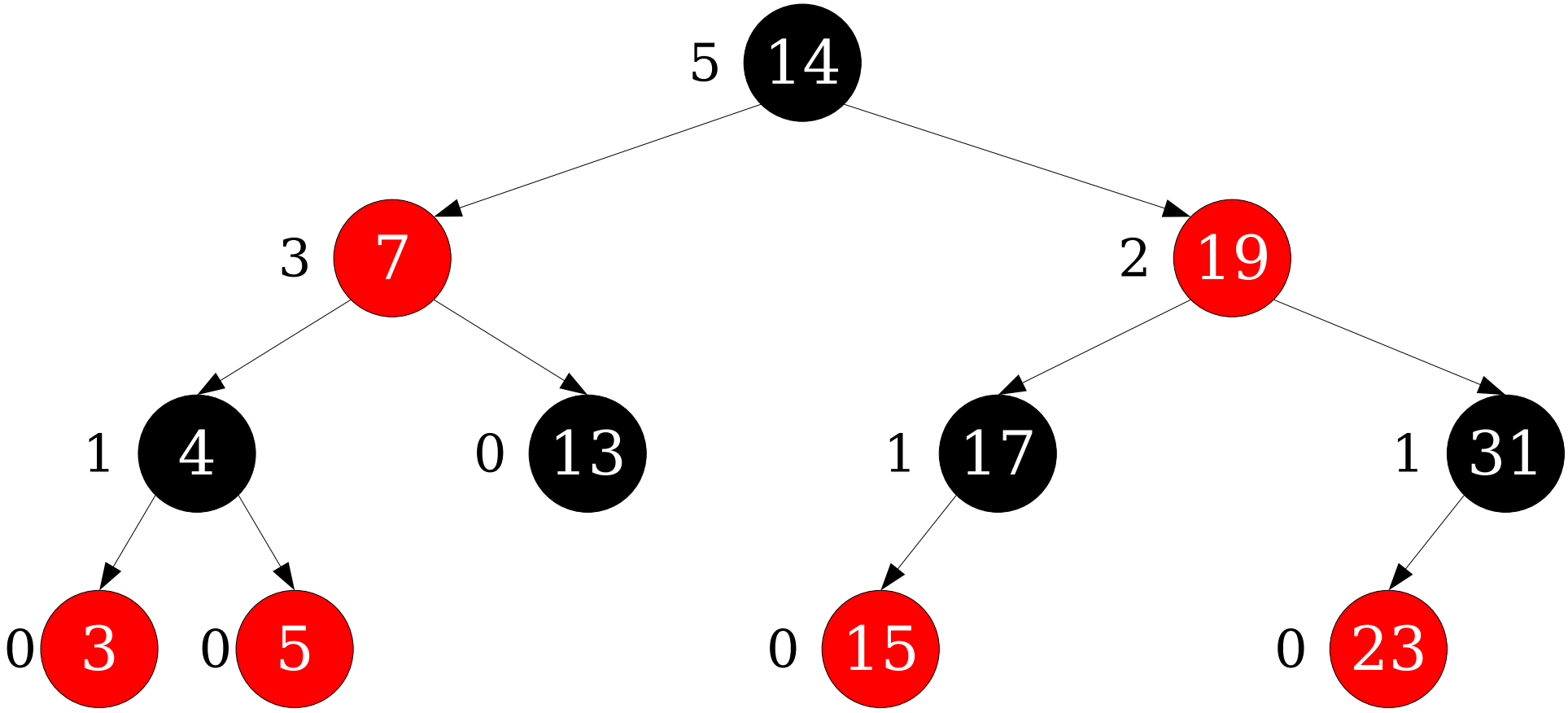
Dynamic Selection



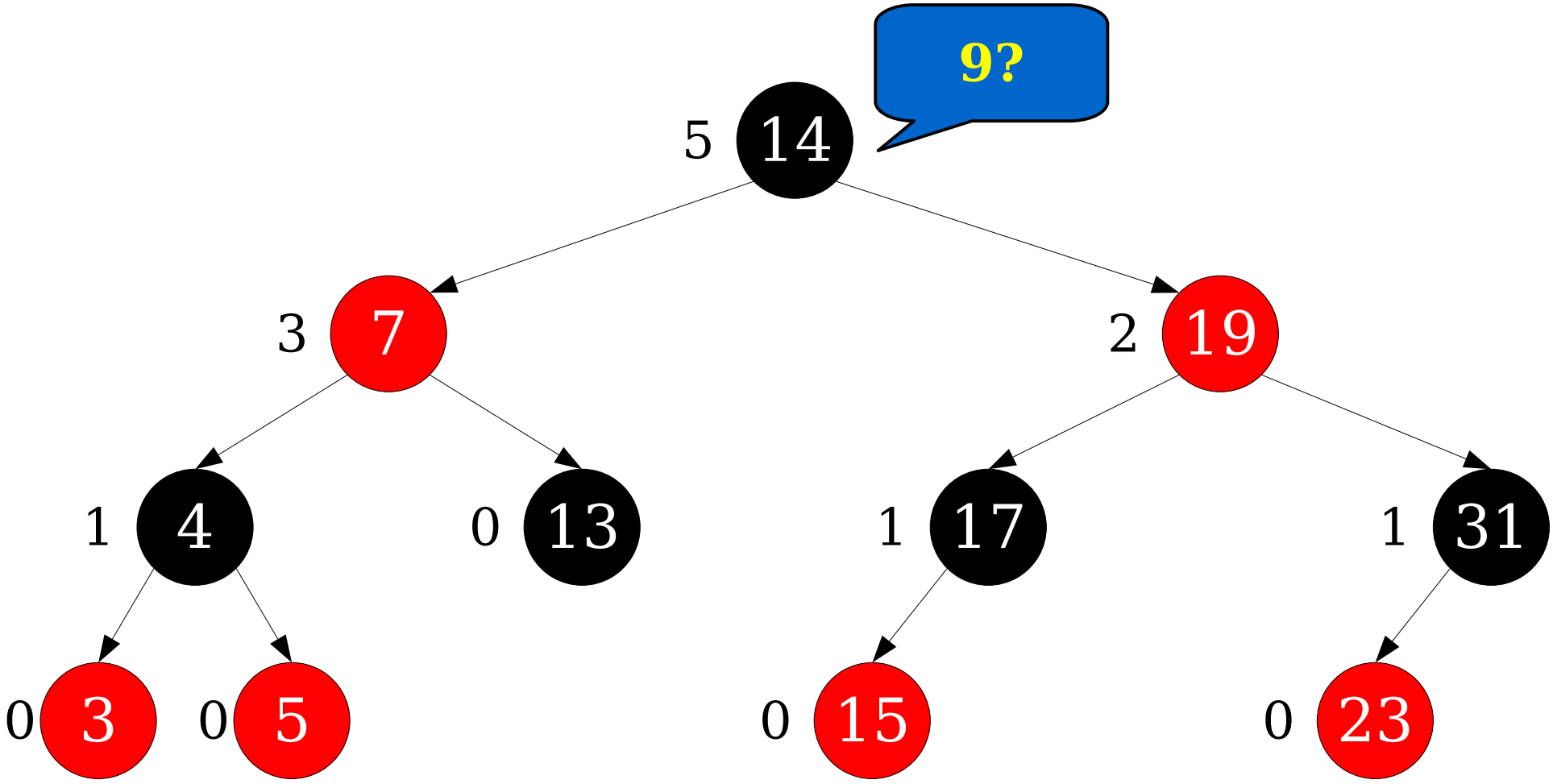
Dynamic Selection



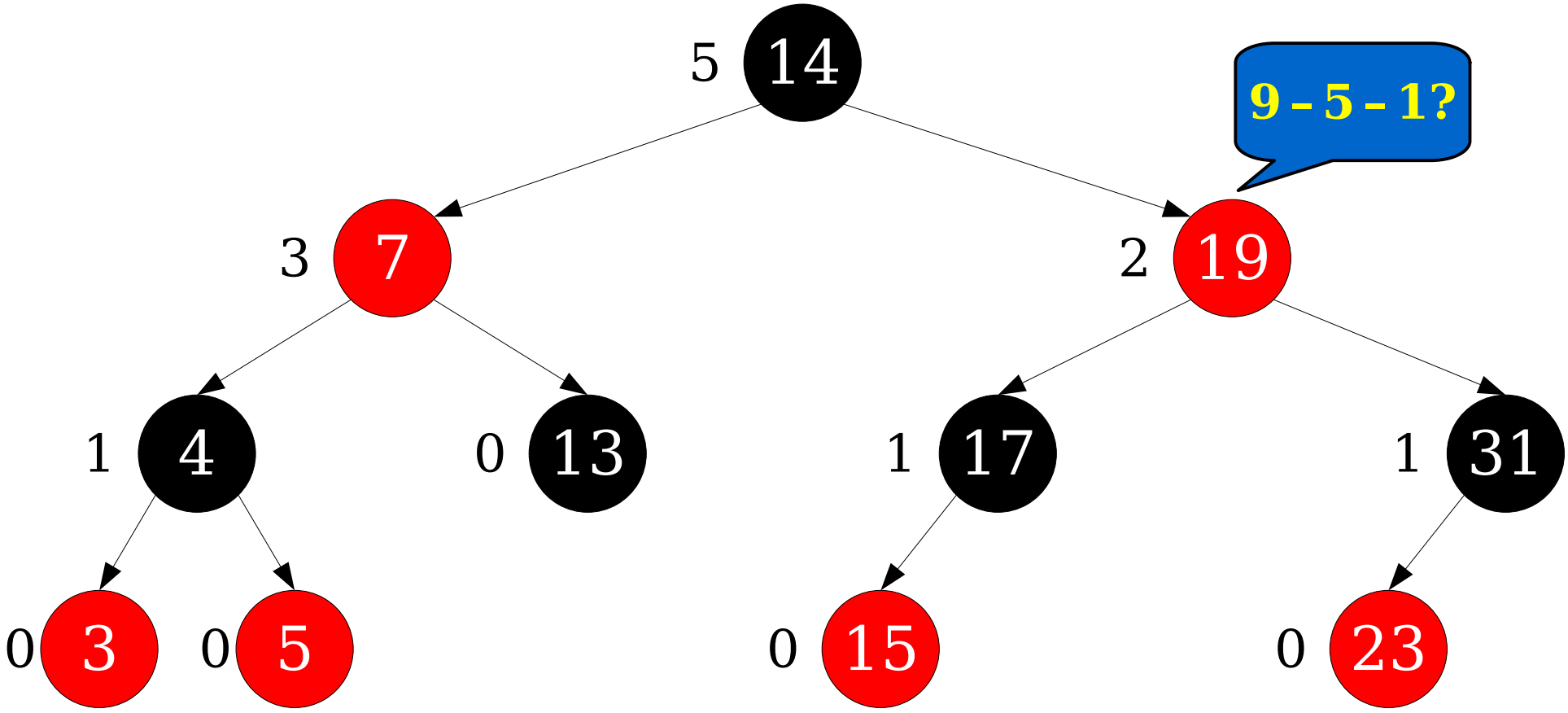
Dynamic Selection



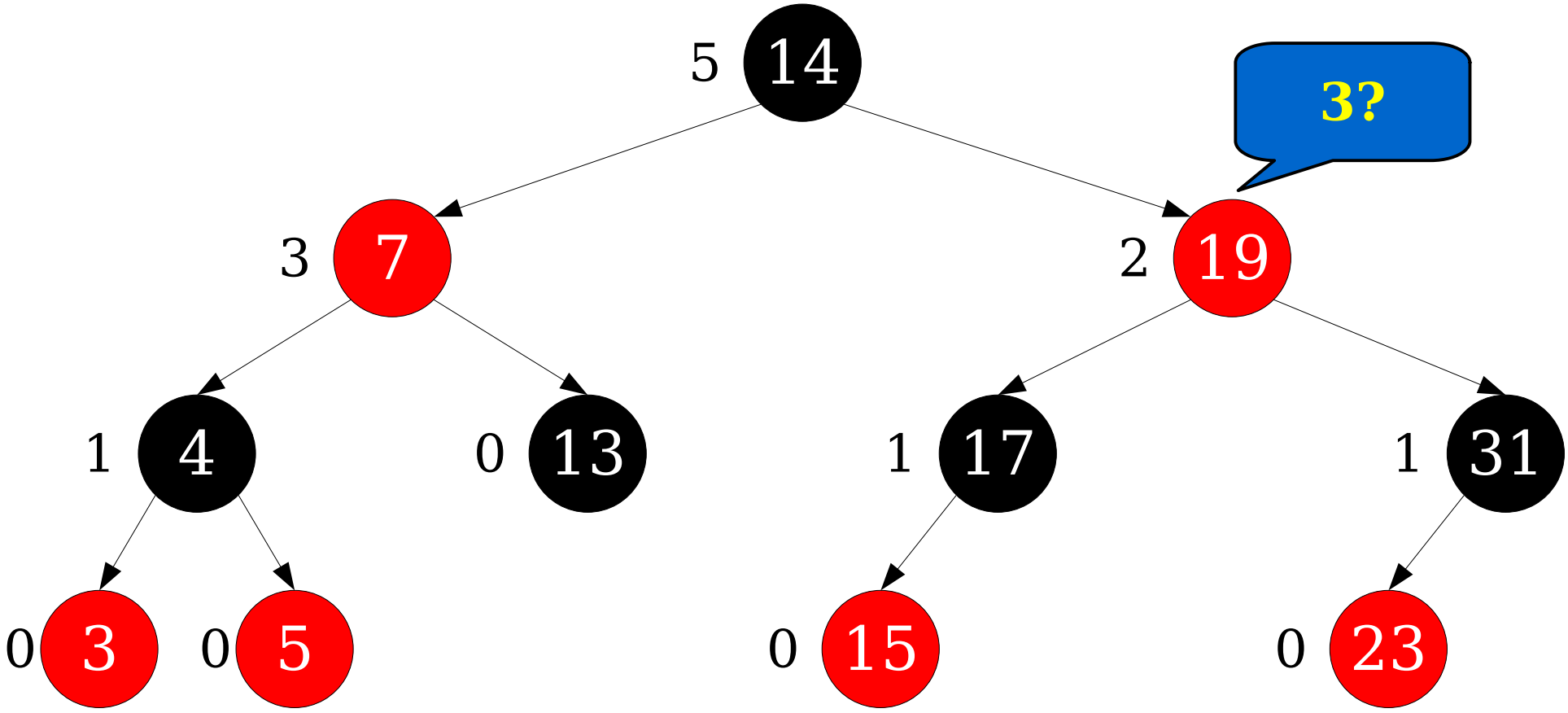
Dynamic Selection



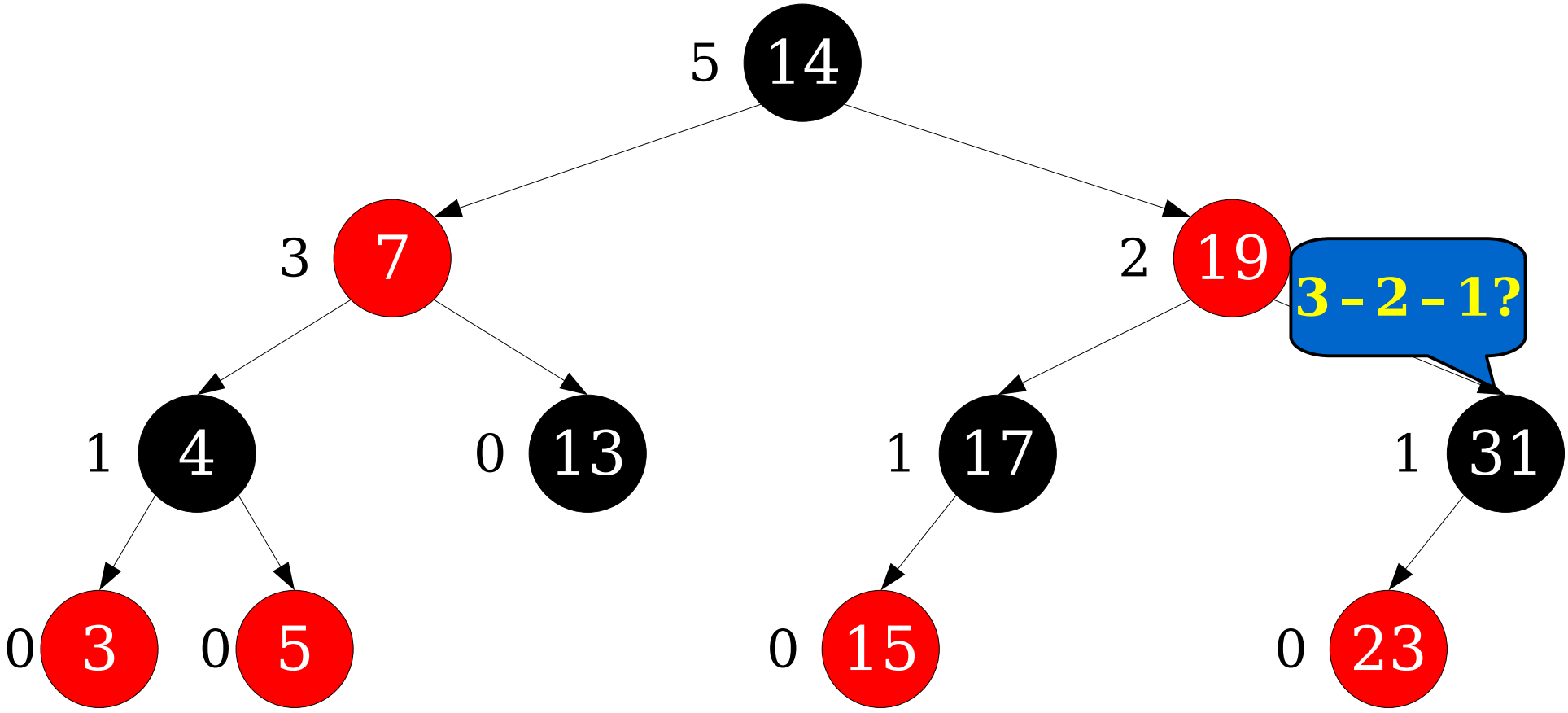
Dynamic Selection



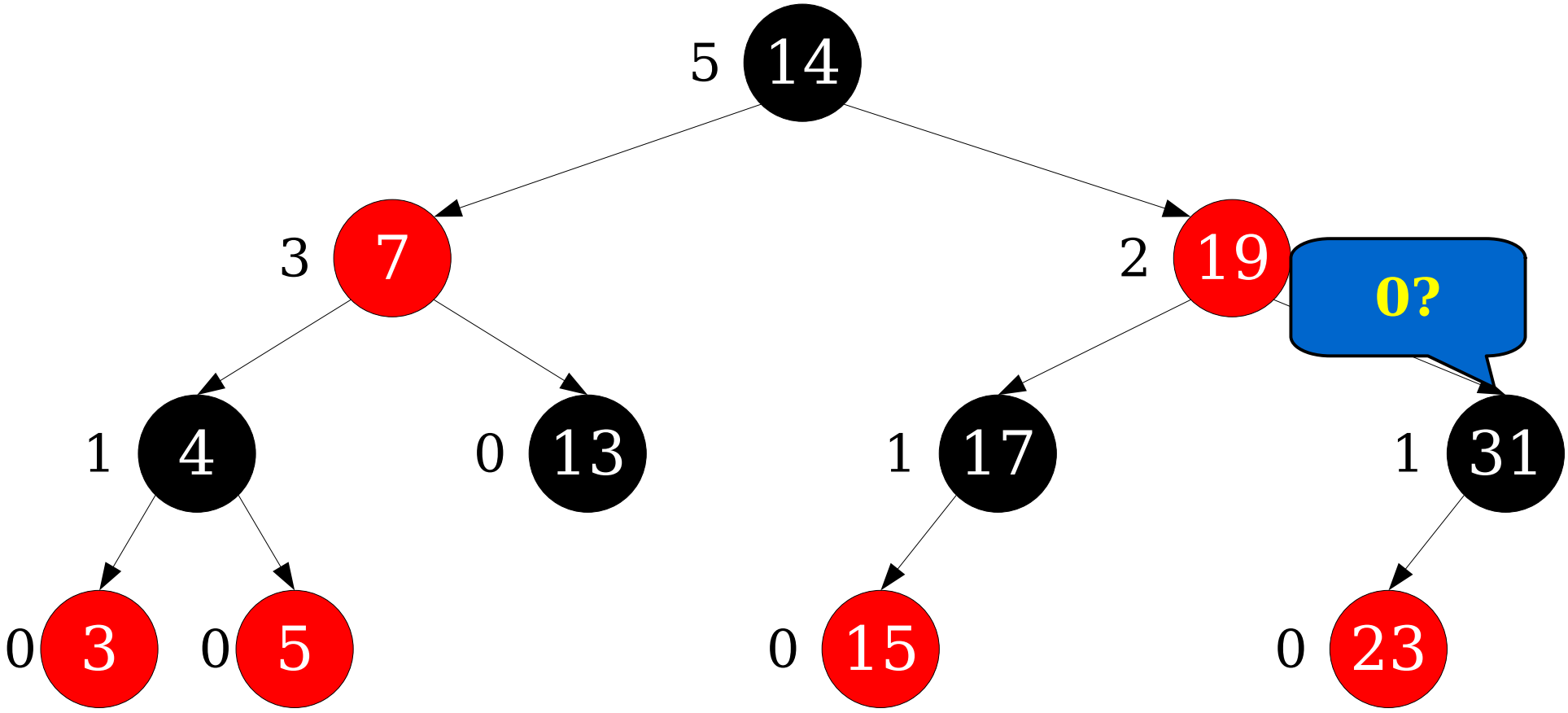
Dynamic Selection



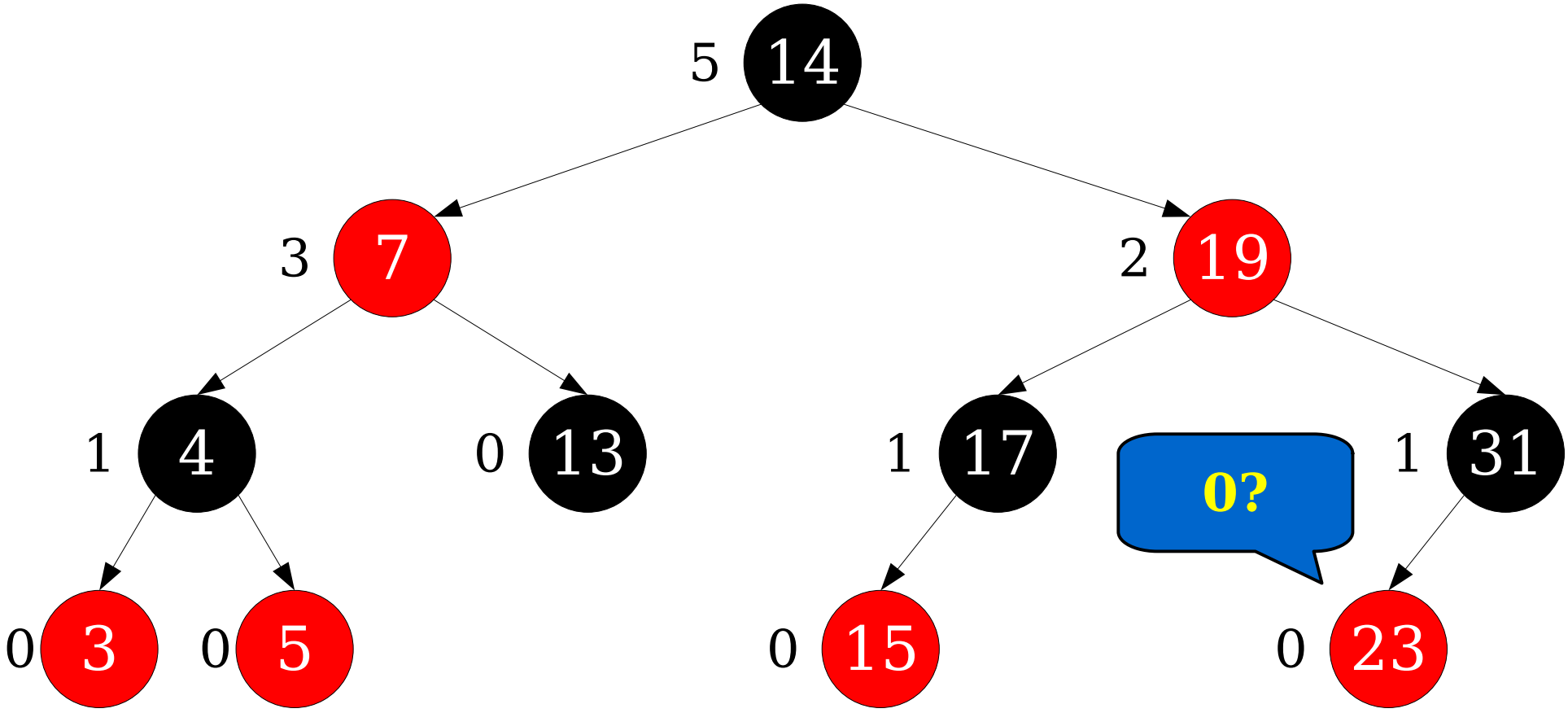
Dynamic Selection



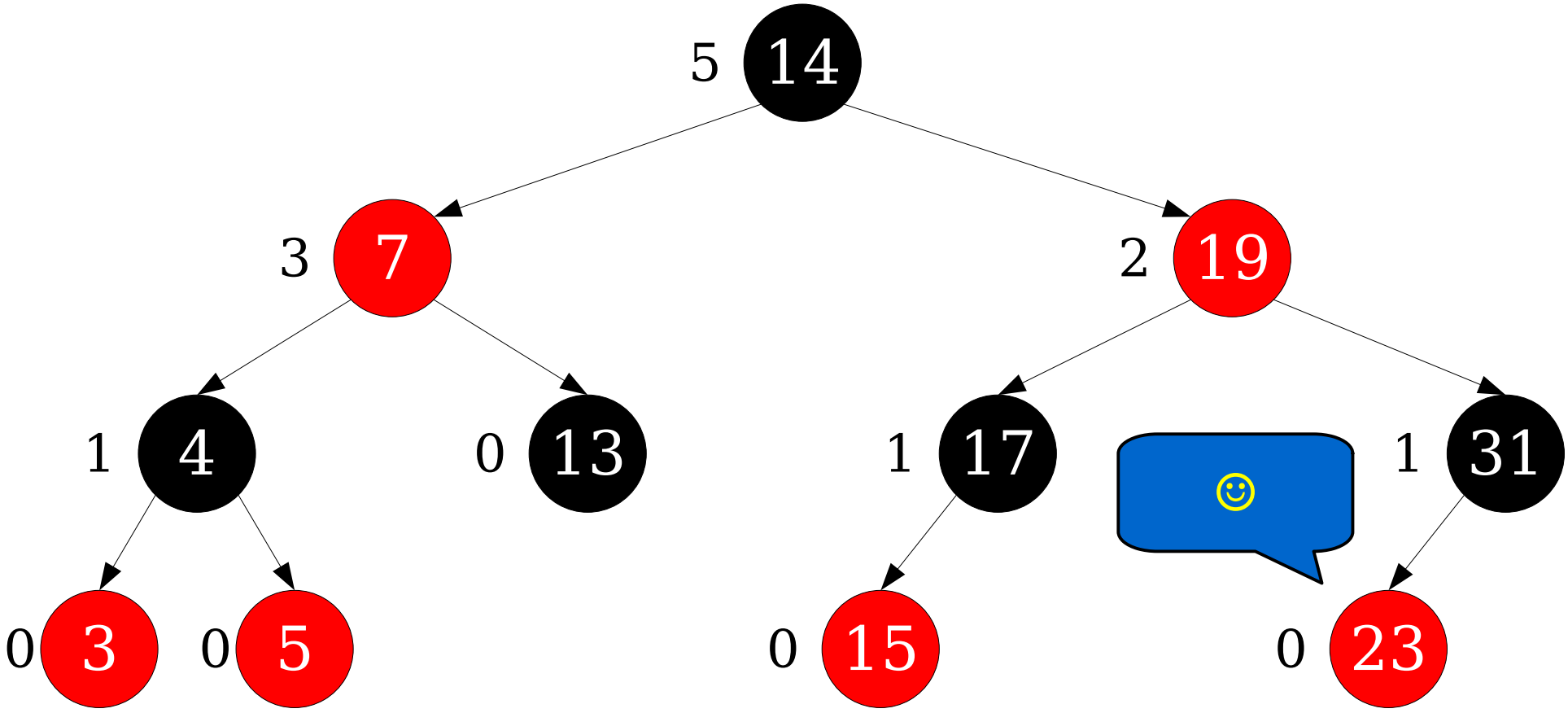
Dynamic Selection



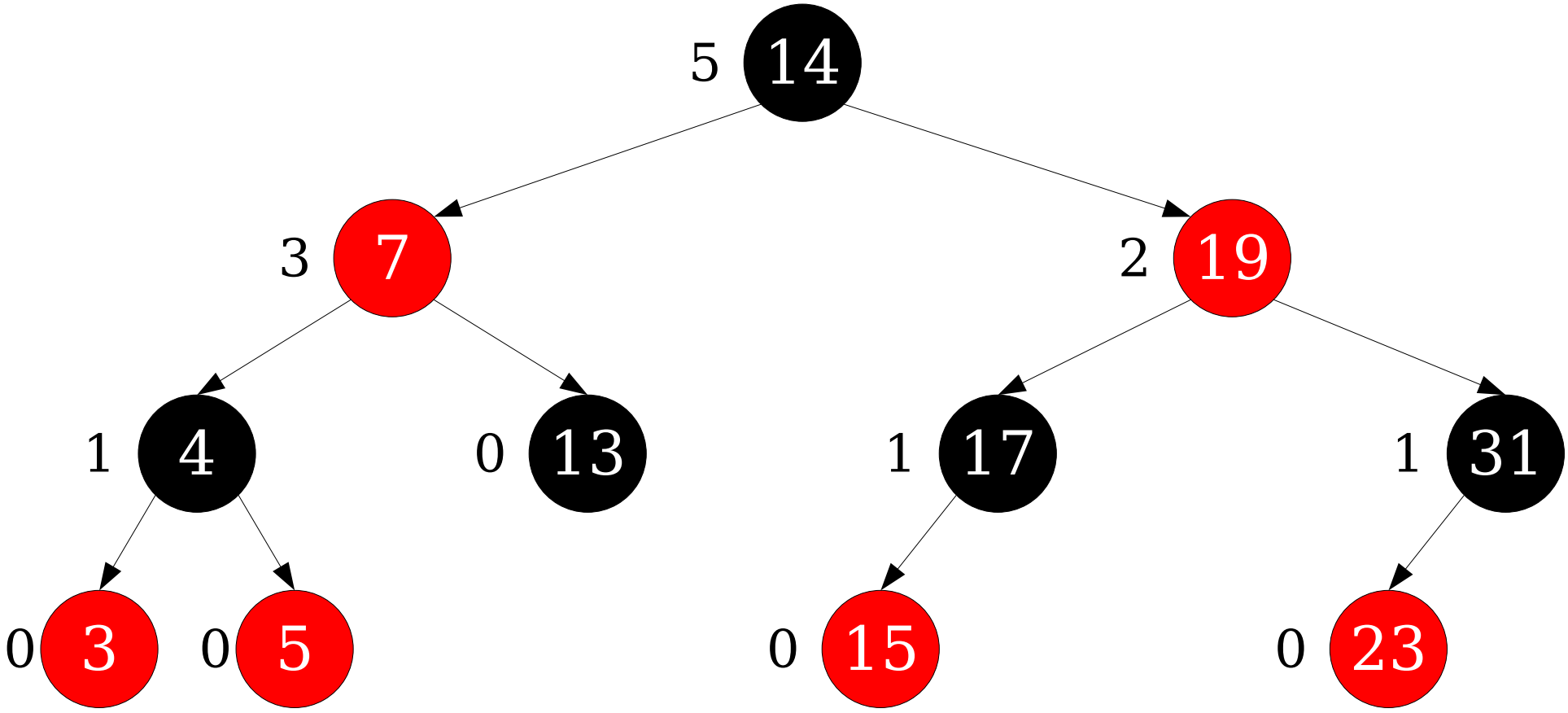
Dynamic Selection



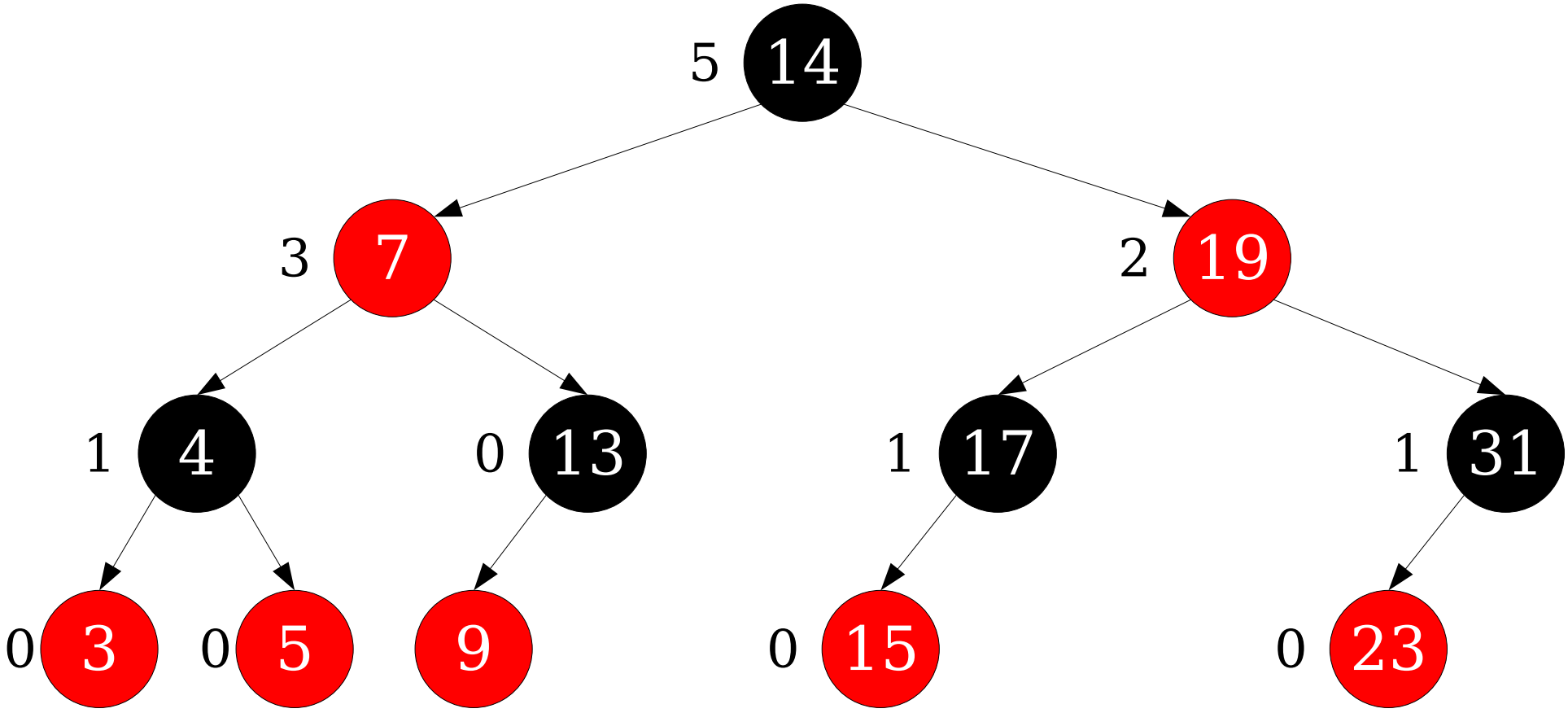
Dynamic Selection



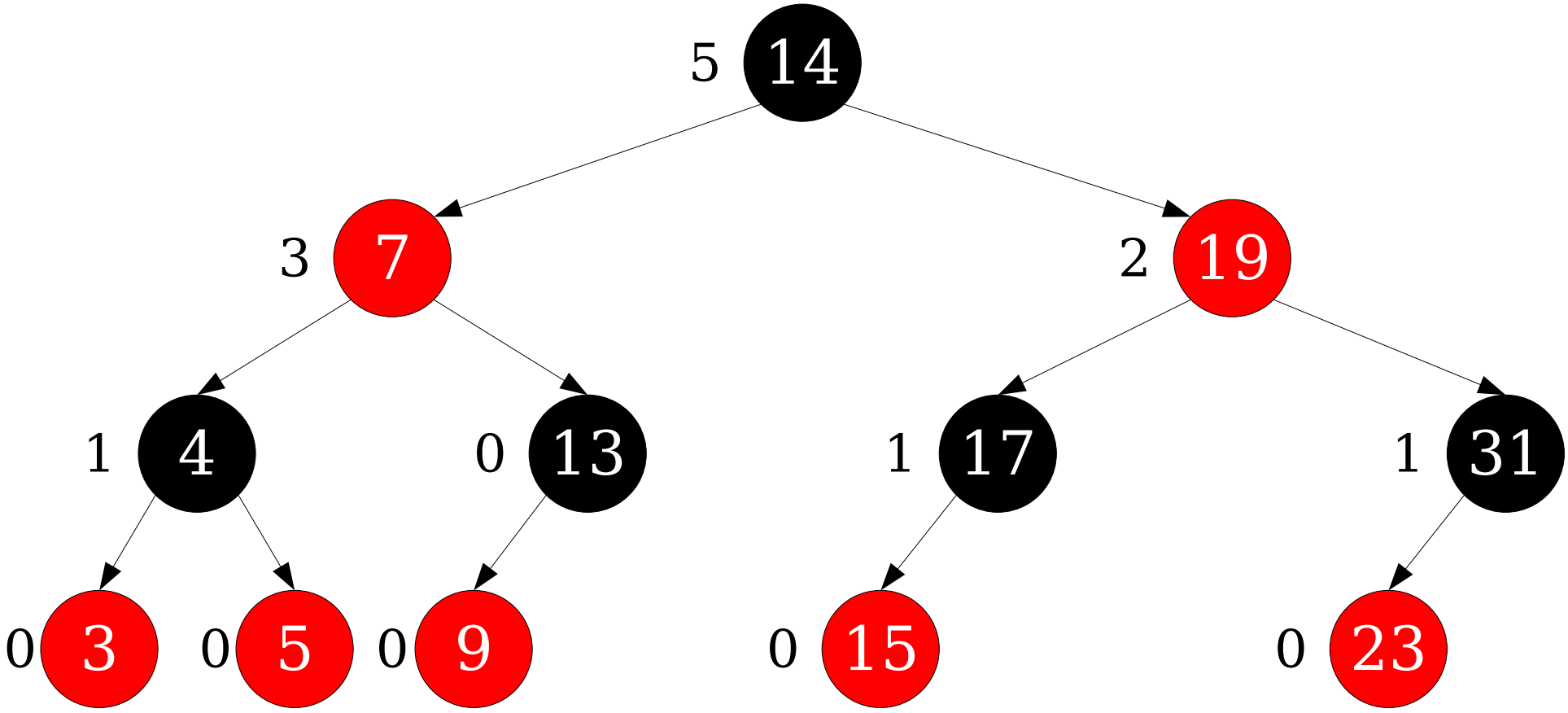
Dynamic Selection



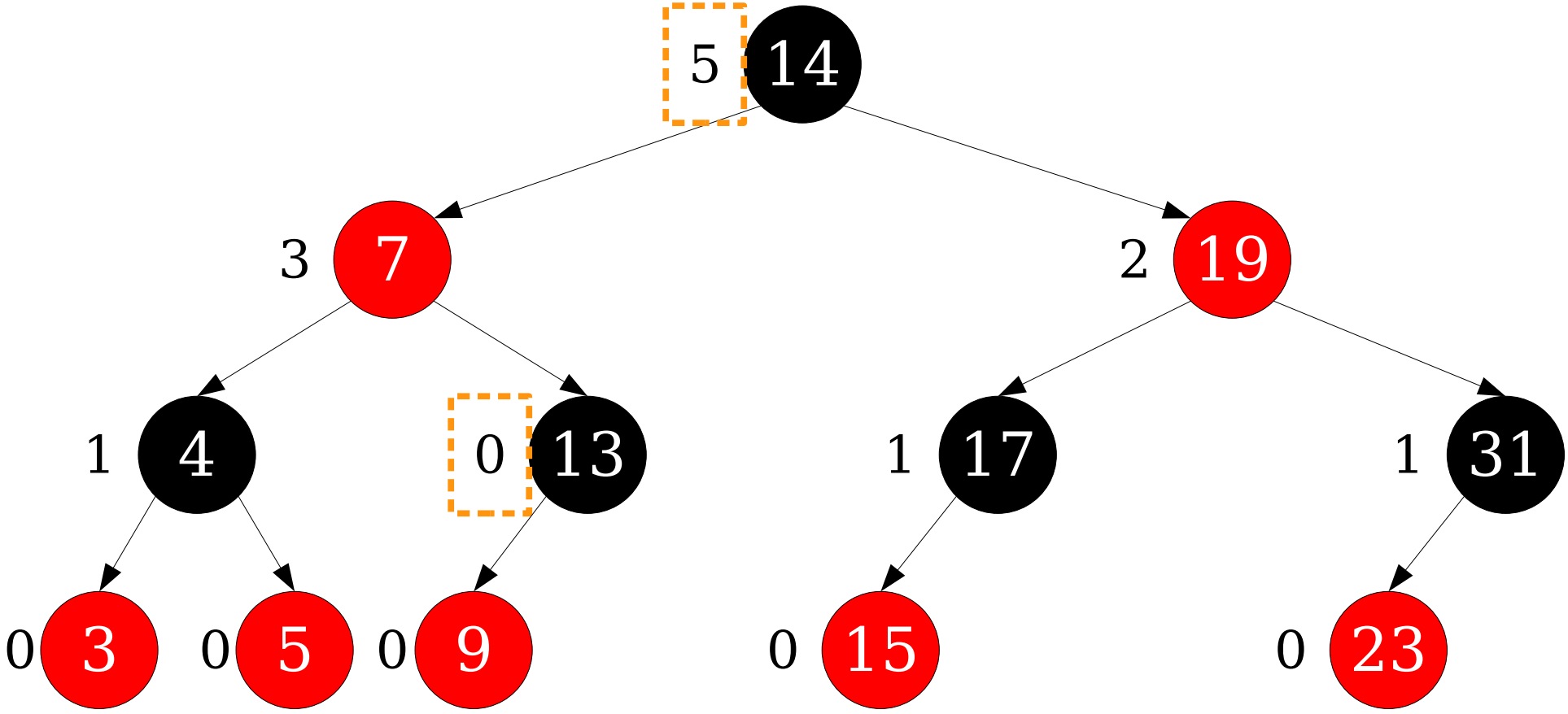
Dynamic Selection



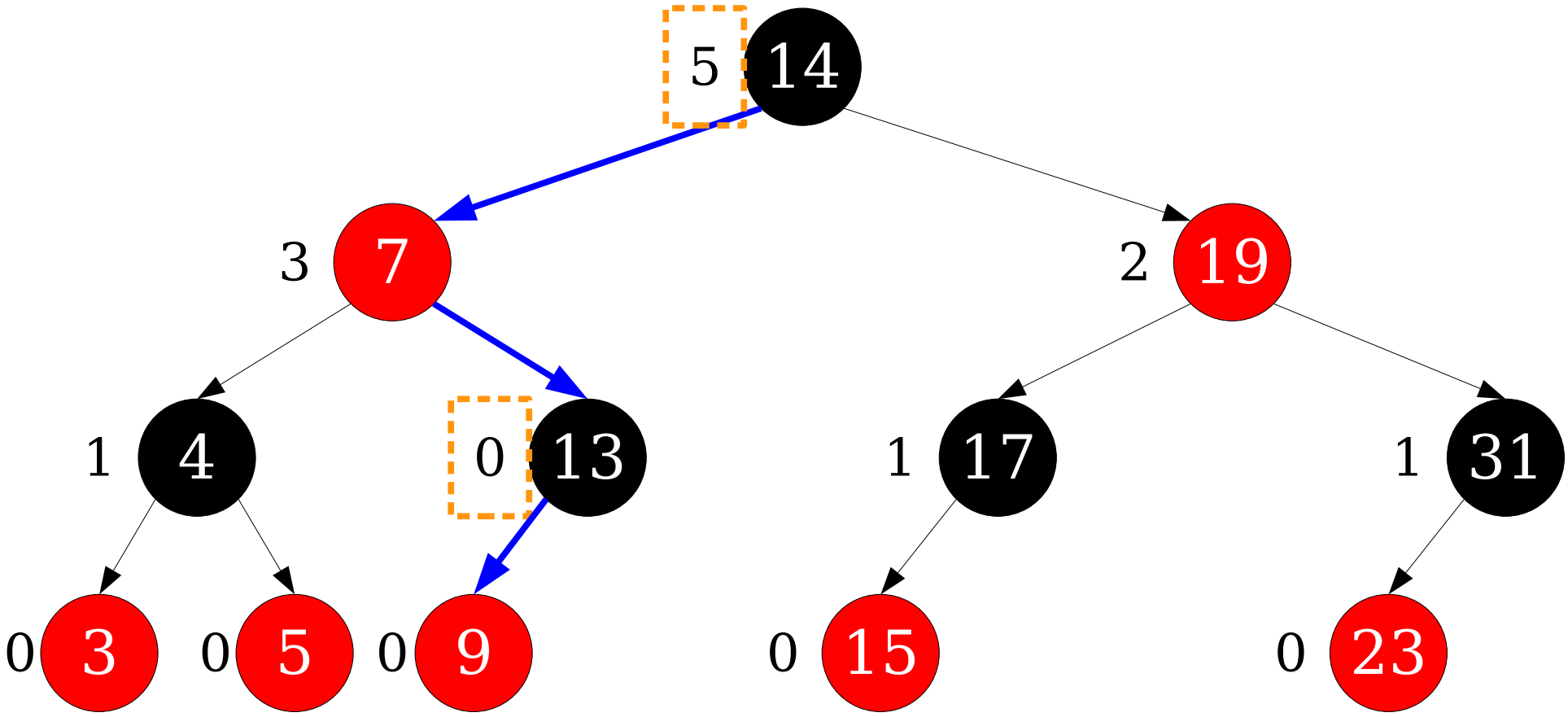
Dynamic Selection



Dynamic Selection

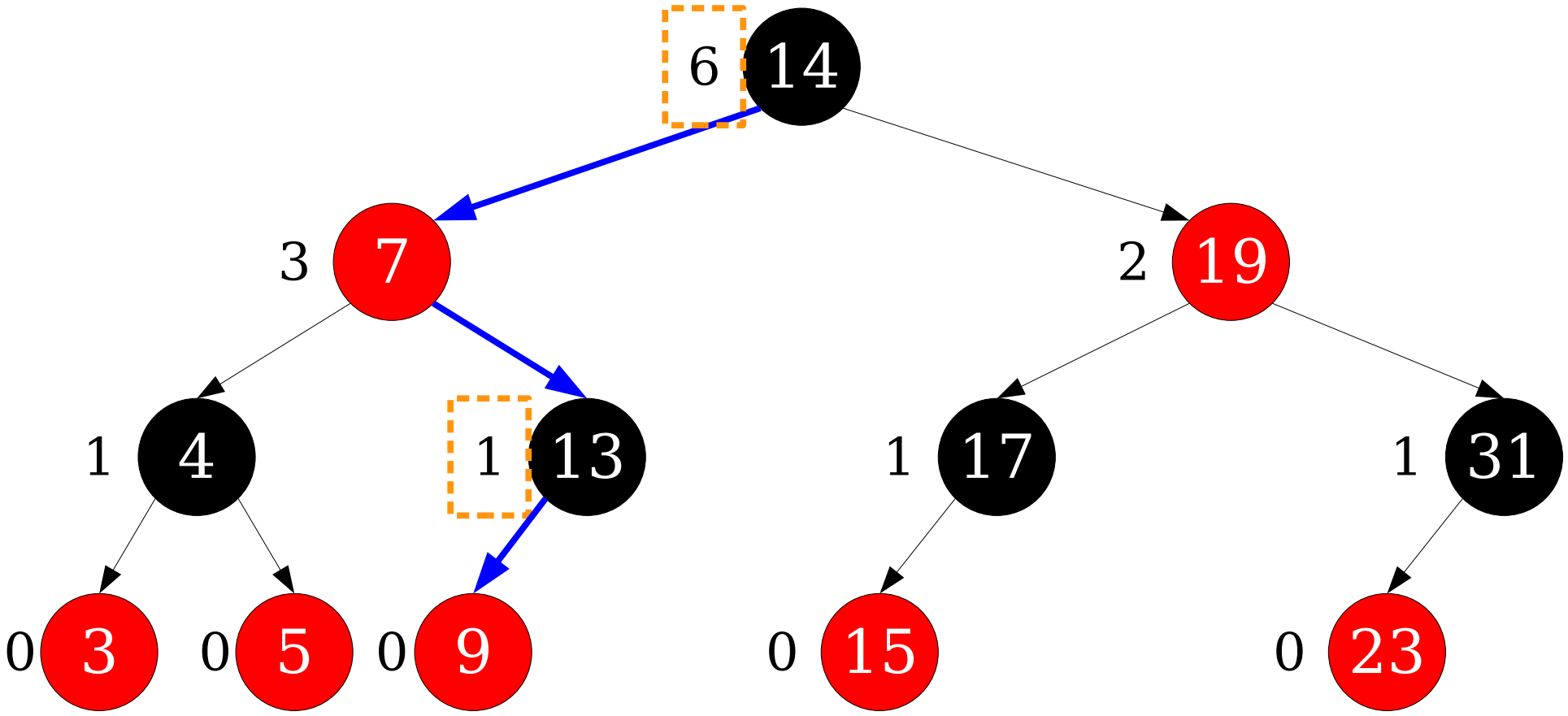


Dynamic Selection



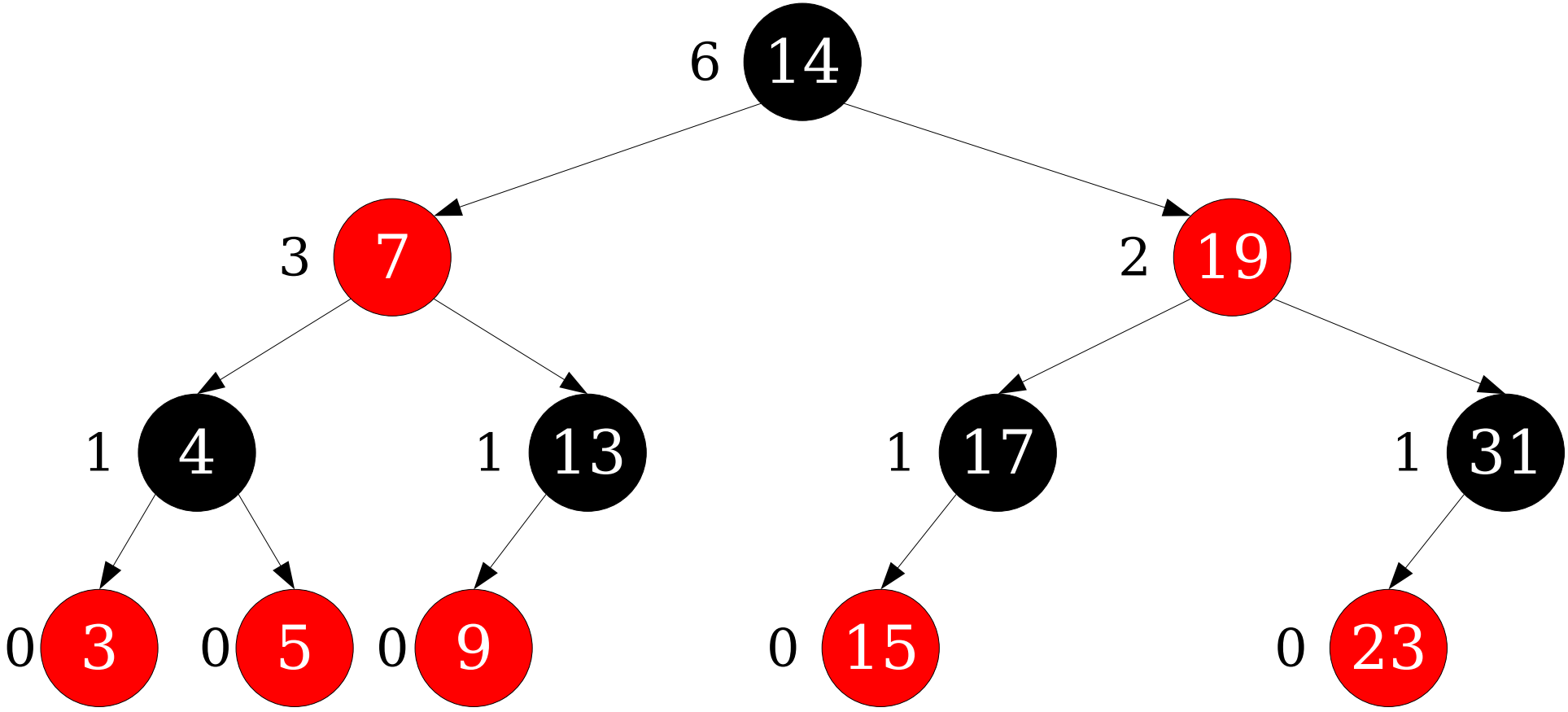
We only update values on nodes that gained a new key in their left subtree. And there are only O(log n) of these!

Dynamic Selection

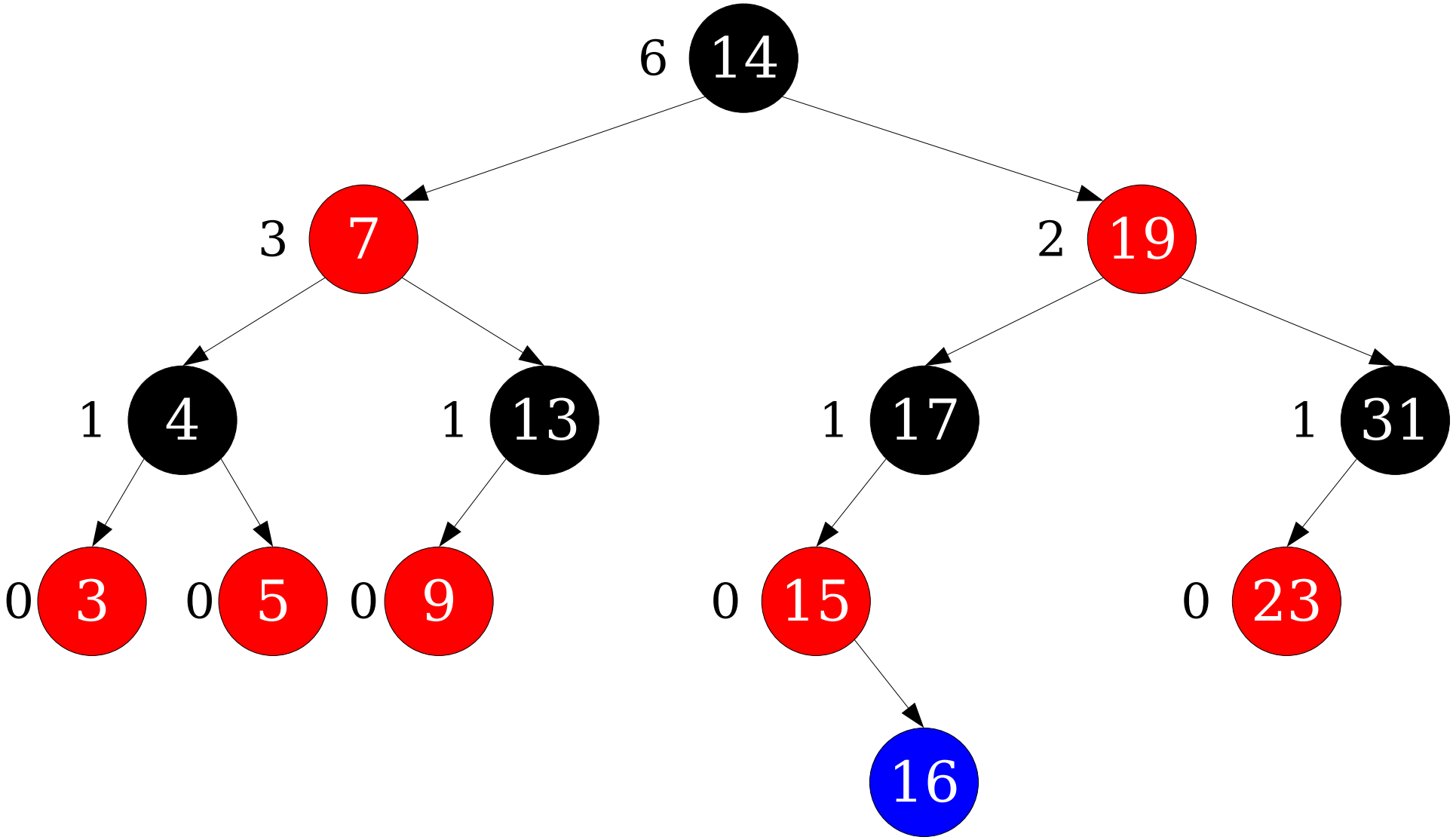


We only update values on nodes that gained a new key in their left subtree. And there are only O(log n) of these!

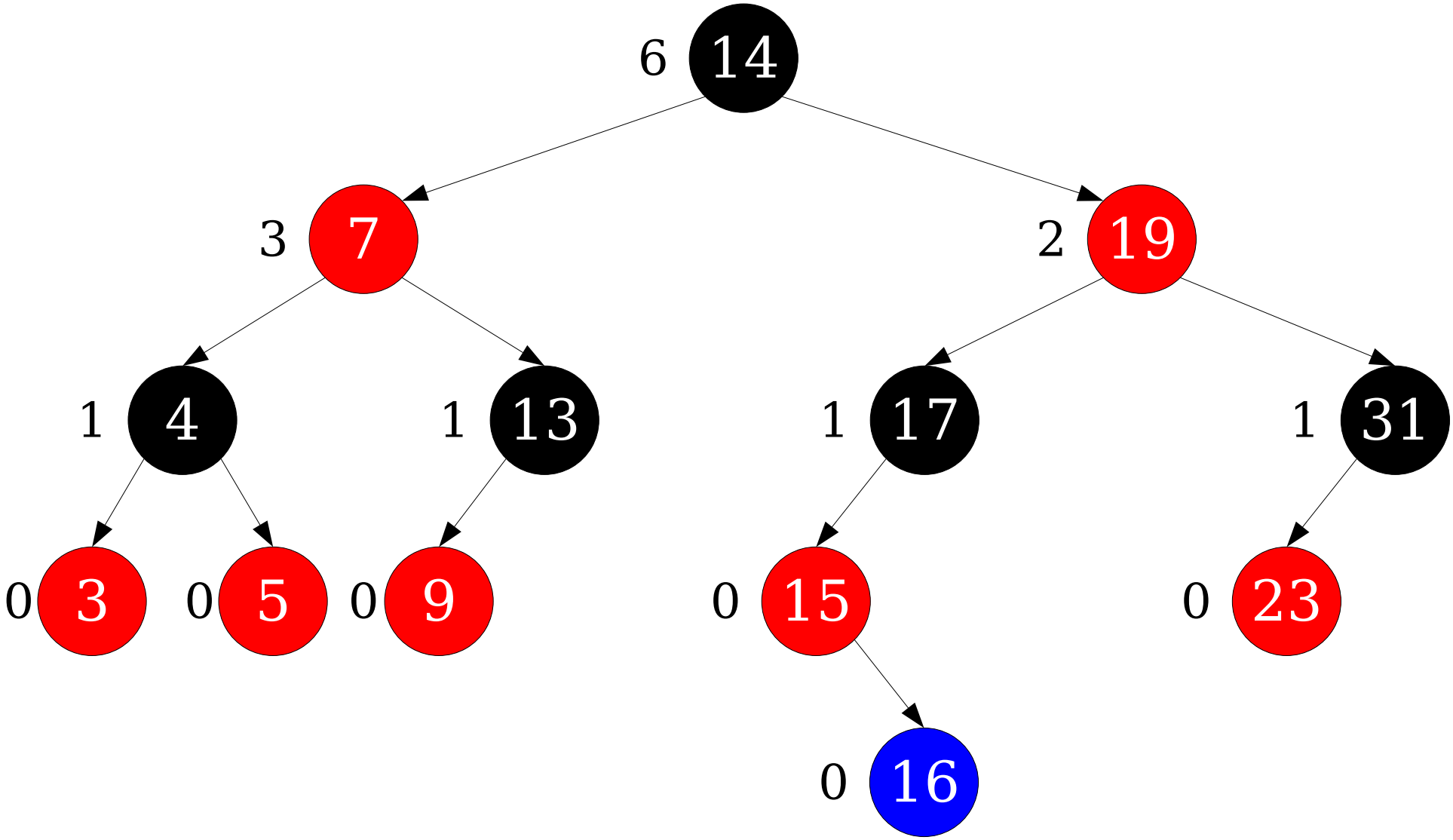
Dynamic Selection



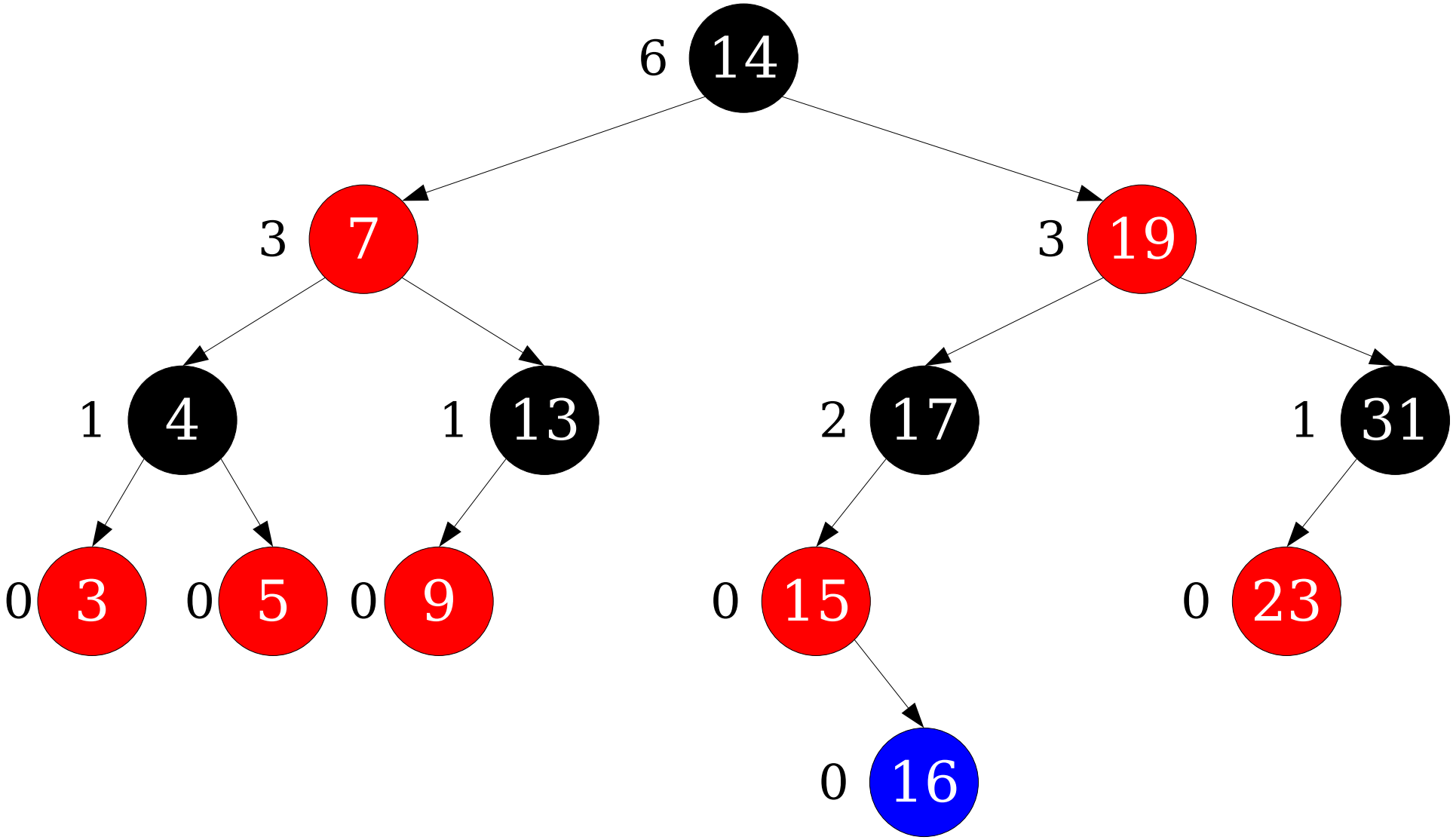
Dynamic Selection



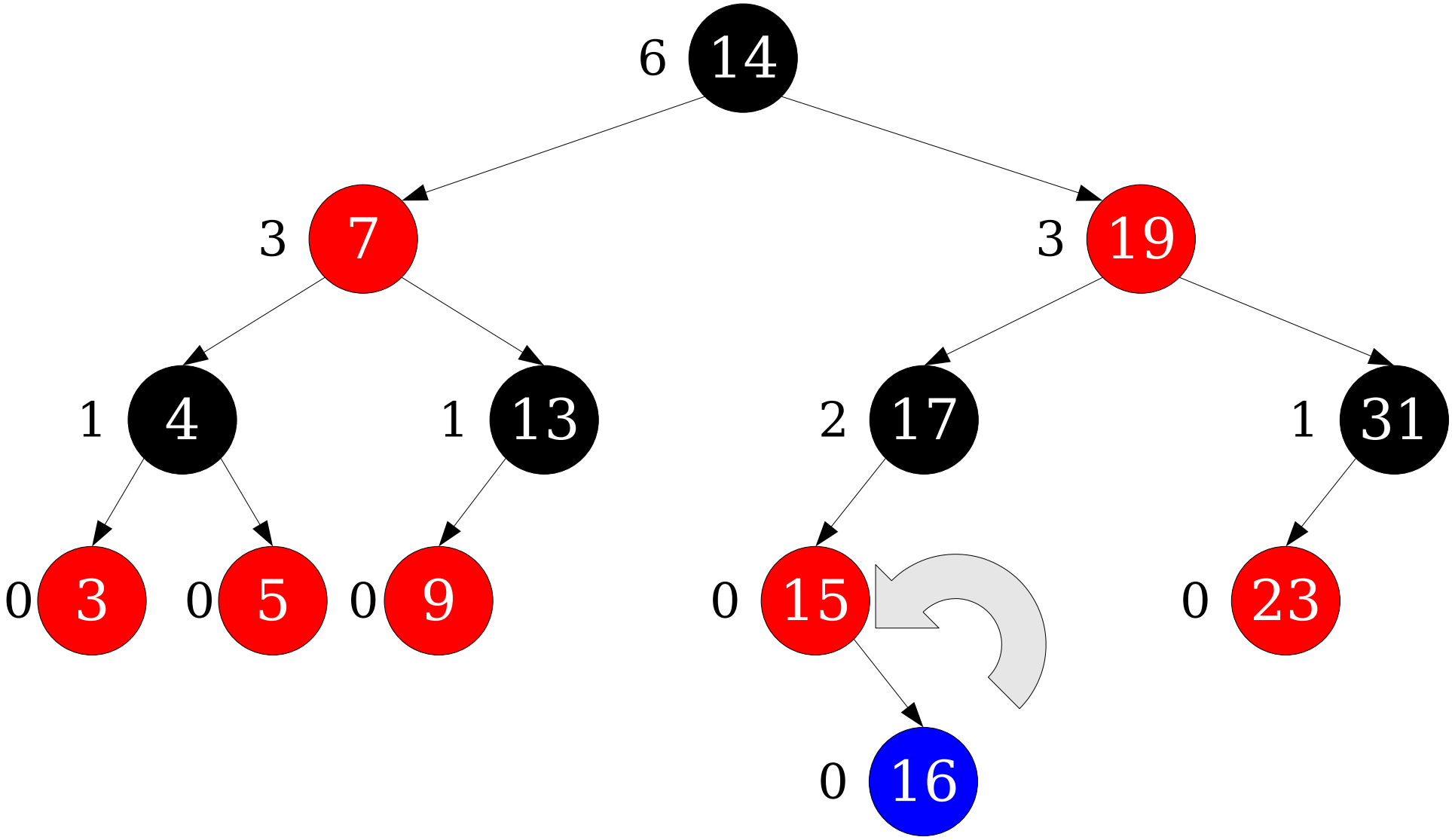
Dynamic Selection



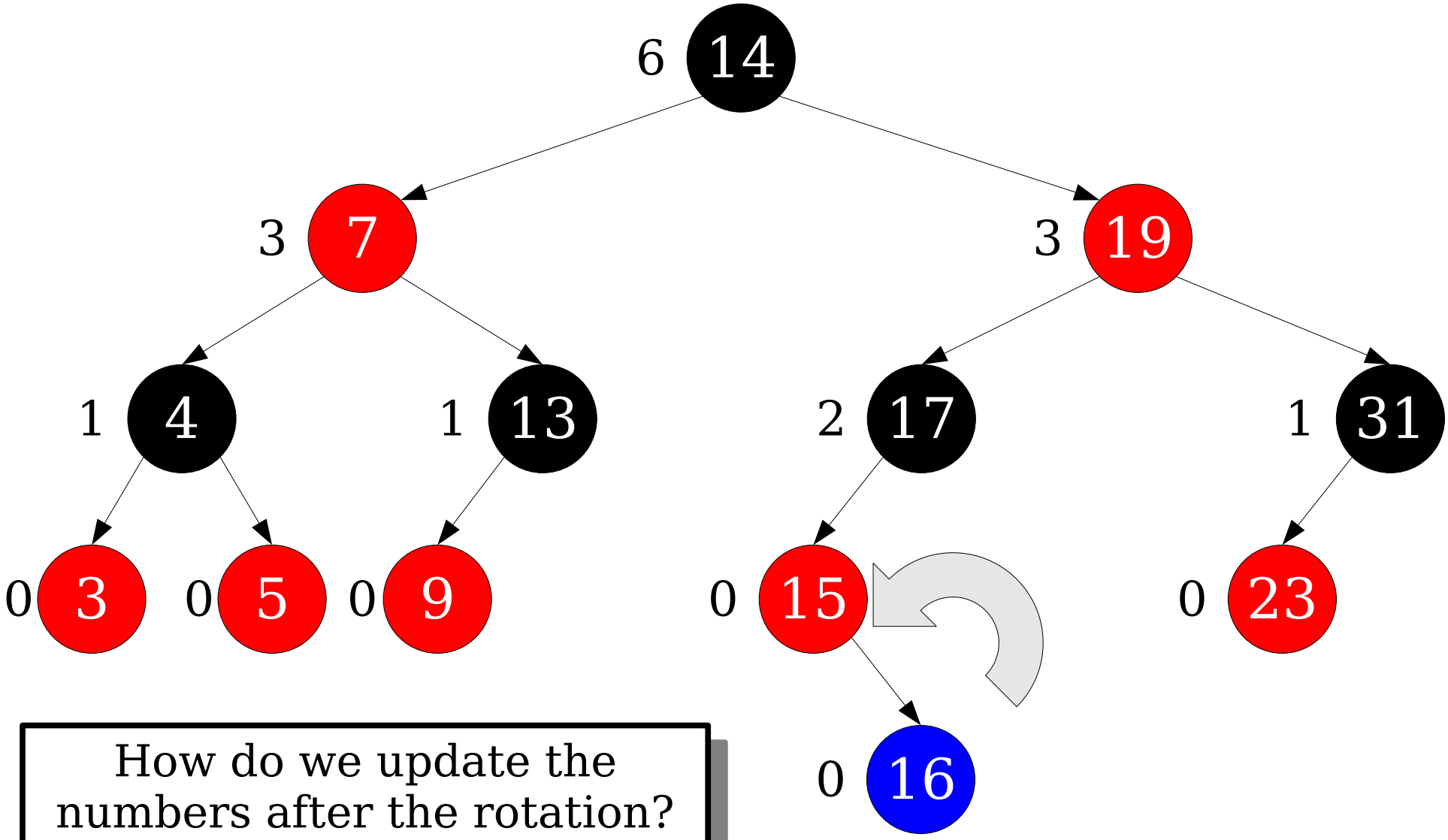
Dynamic Selection



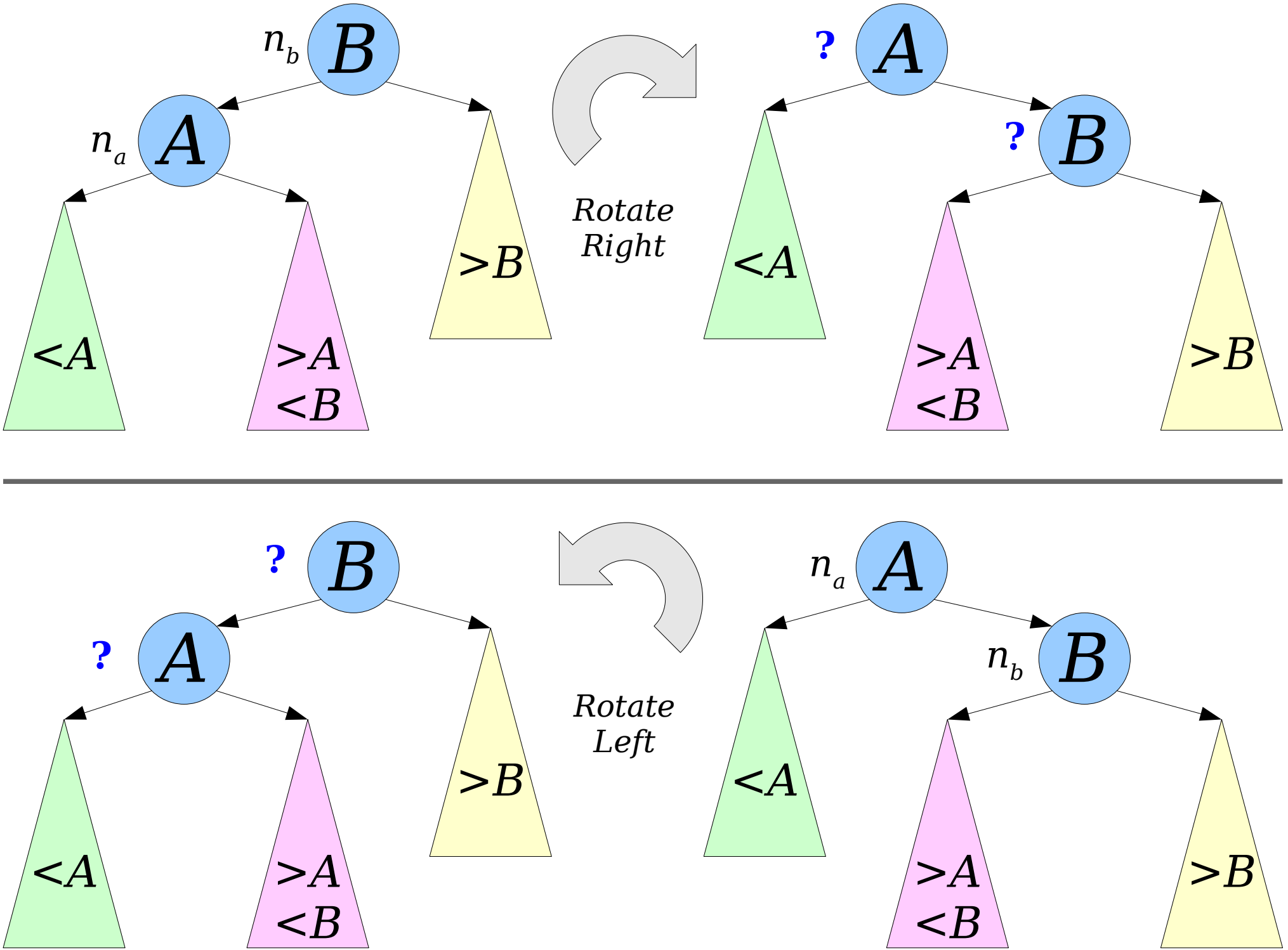
Dynamic Selection

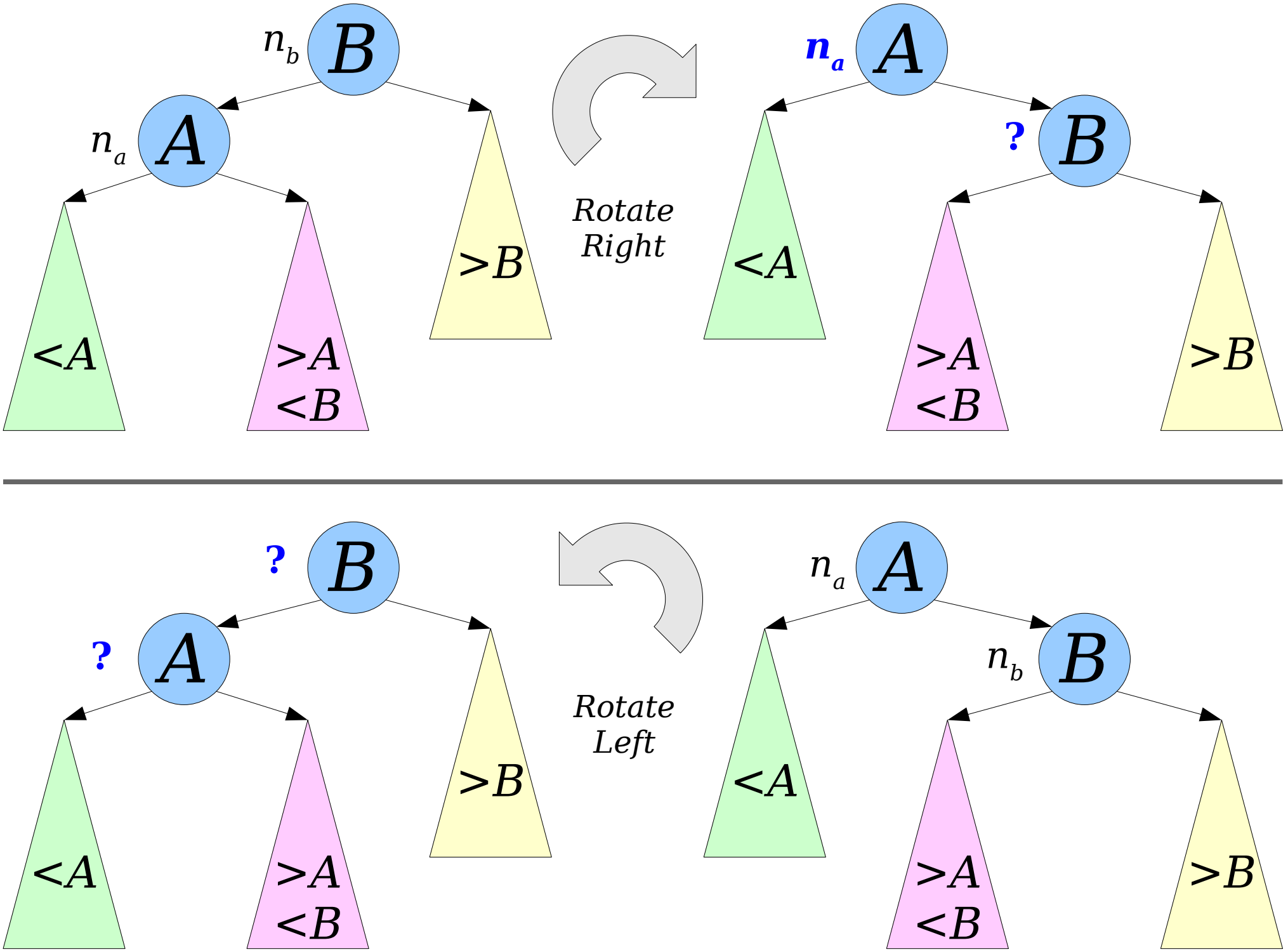


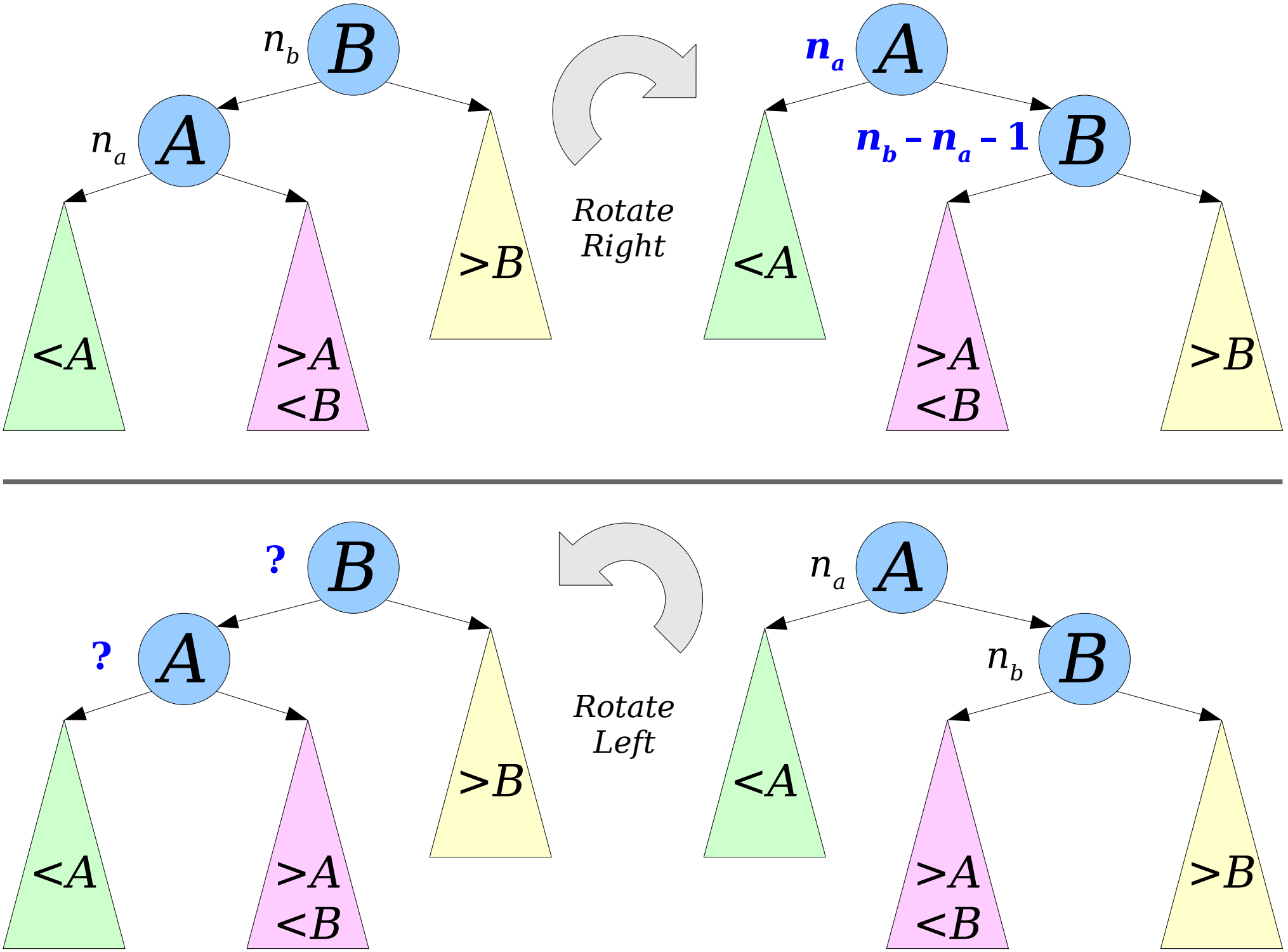
Dynamic Selection

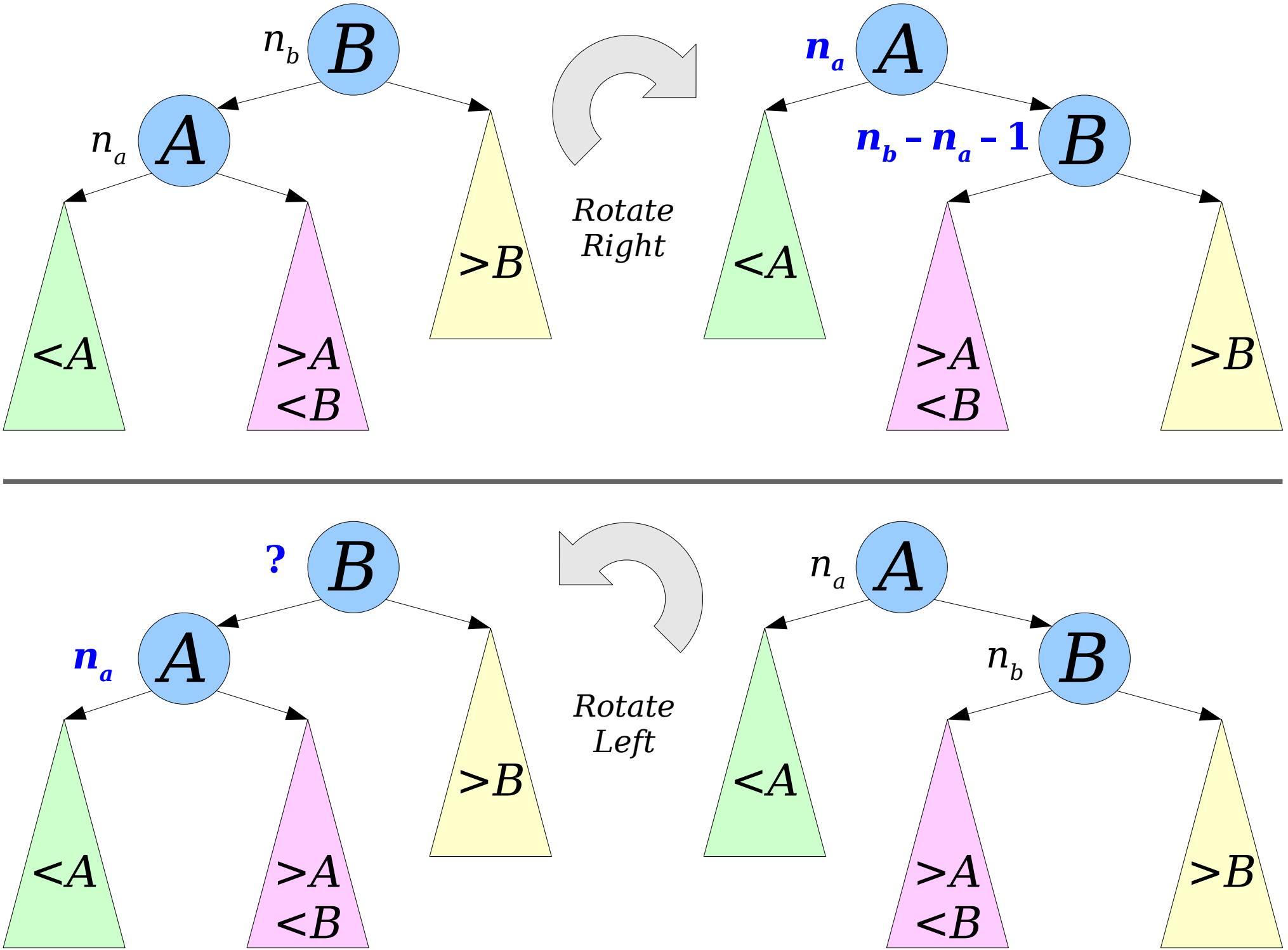


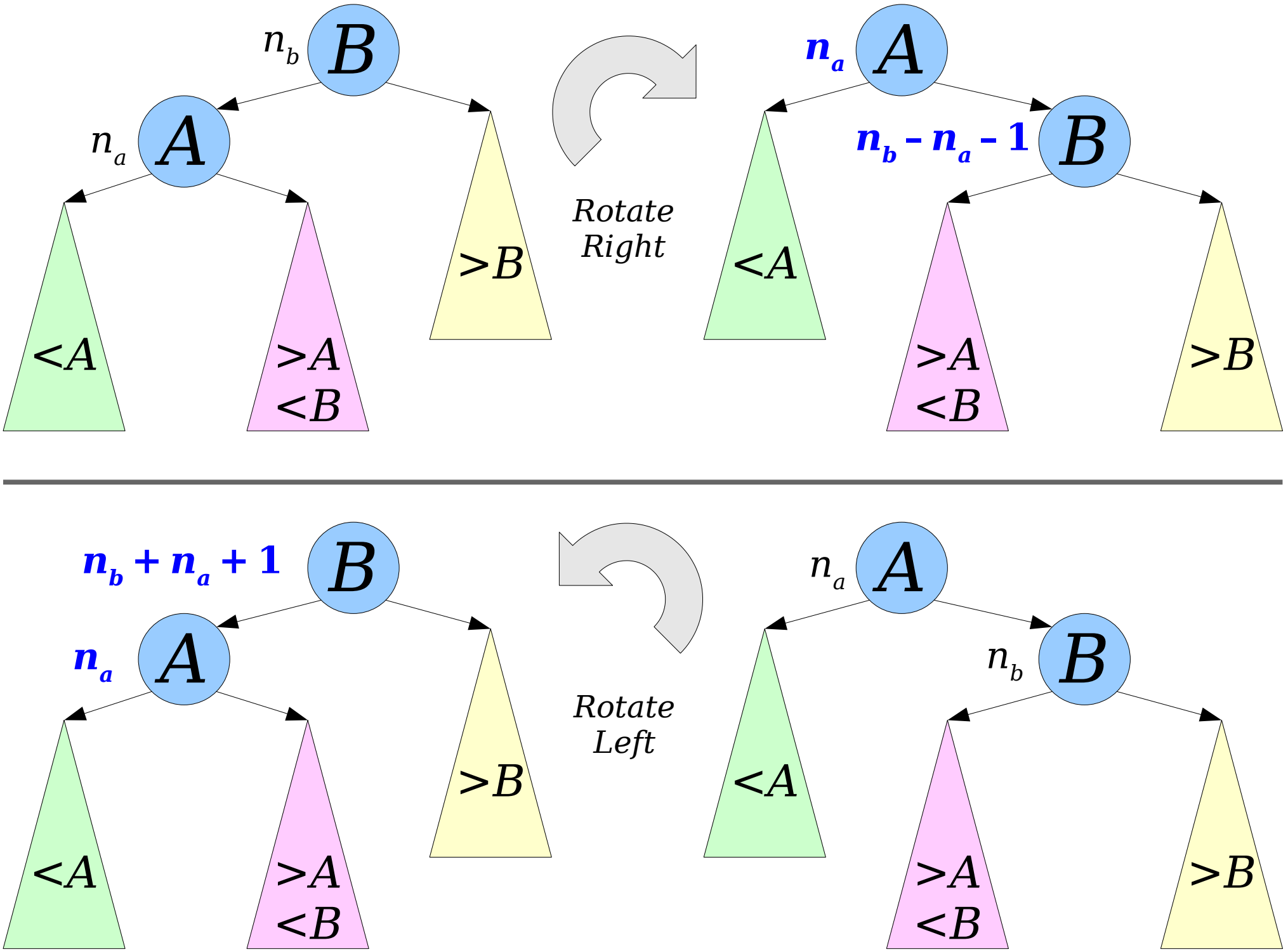
How do we update the numbers after the rotation?


n_b B
n_a A
<A
>A
<B
>B
Rotate
Right
? A
? B
<A
>A
<B
>B
? B
? A
<A
>A
<B
>B
Rotate
Left
n_a A
n_b B
<A
>A
<B
>B

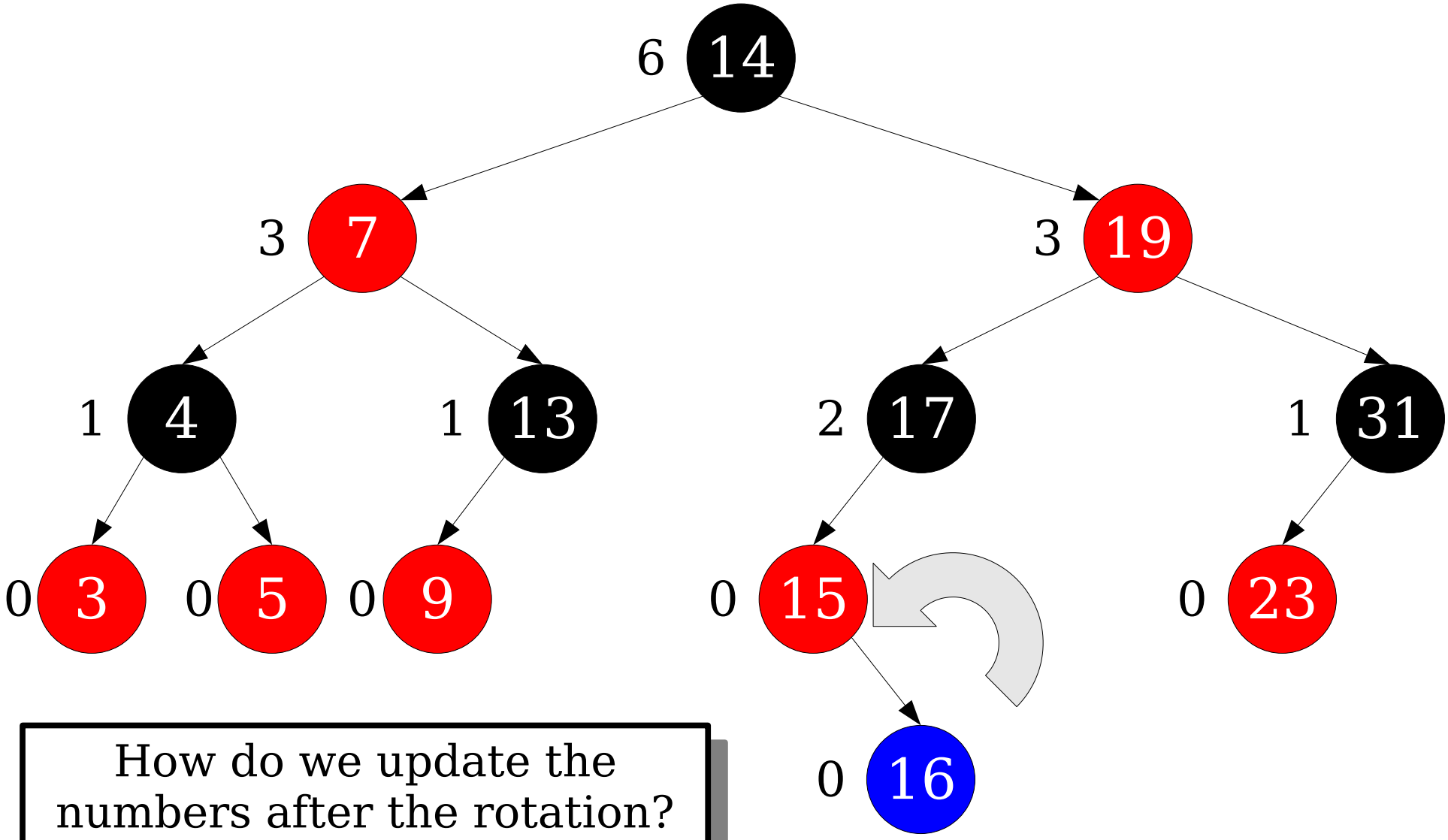

n_b B
n_a A
<A
>A
<B
>B
Rotate
Right
n_a A
?
B
<A
>A
<B
>B
? B
? A
<A
>A
<B
>B
Rotate
Left
n_a A
n_b B
<A
>A
<B
>B


n_b B
n_a A
<A
>A
<B
>B
Rotate
Right
n_a A
$n_b - n_a - 1$ B
<A
>A
<B
>B
? B
? A
<A
>A
<B
>B
Rotate
Left
n_a A
n_b B
<A
>A
<B
>B


n_b B
n_a A
<A
>A <B
>B
Rotate Right
n_a A
$n_b - n_a - 1$ B
<A
>A <B
>B
? B
n_a A
<A
>A <B
>B
Rotate Left
n_a A
n_b B
<A
>A <B
>B

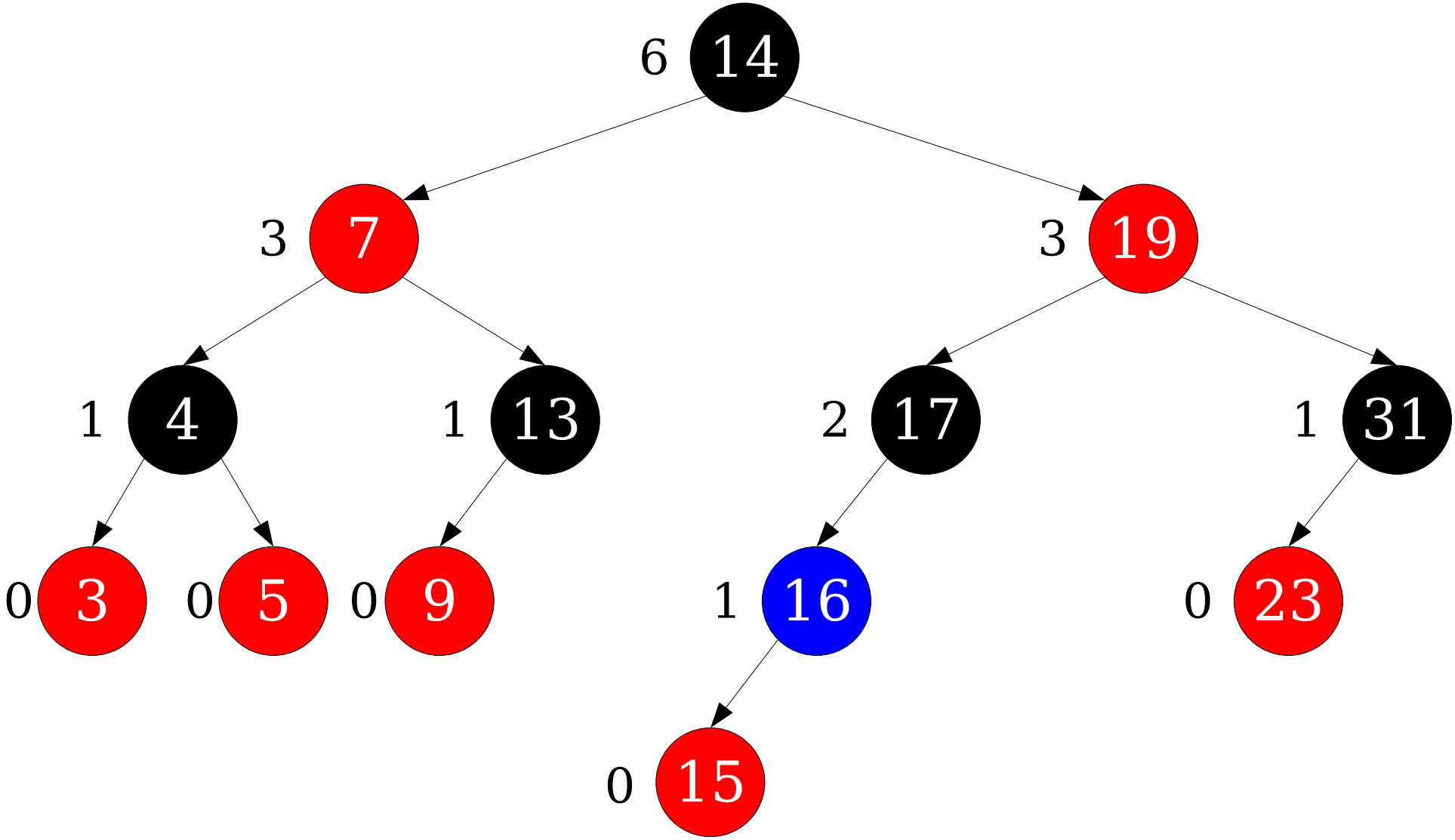

n_b B
n_a A
<A
>A
<B
>B
Rotate
Right
n_a A
$n_b - n_a - 1$ B
<A
>A
<B
>B
$n_b + n_a + 1$ B
n_a A
<A
>A
<B
>B
Rotate
Left
n_a A
n_b B
<A
>A
<B
>B

Dynamic Selection

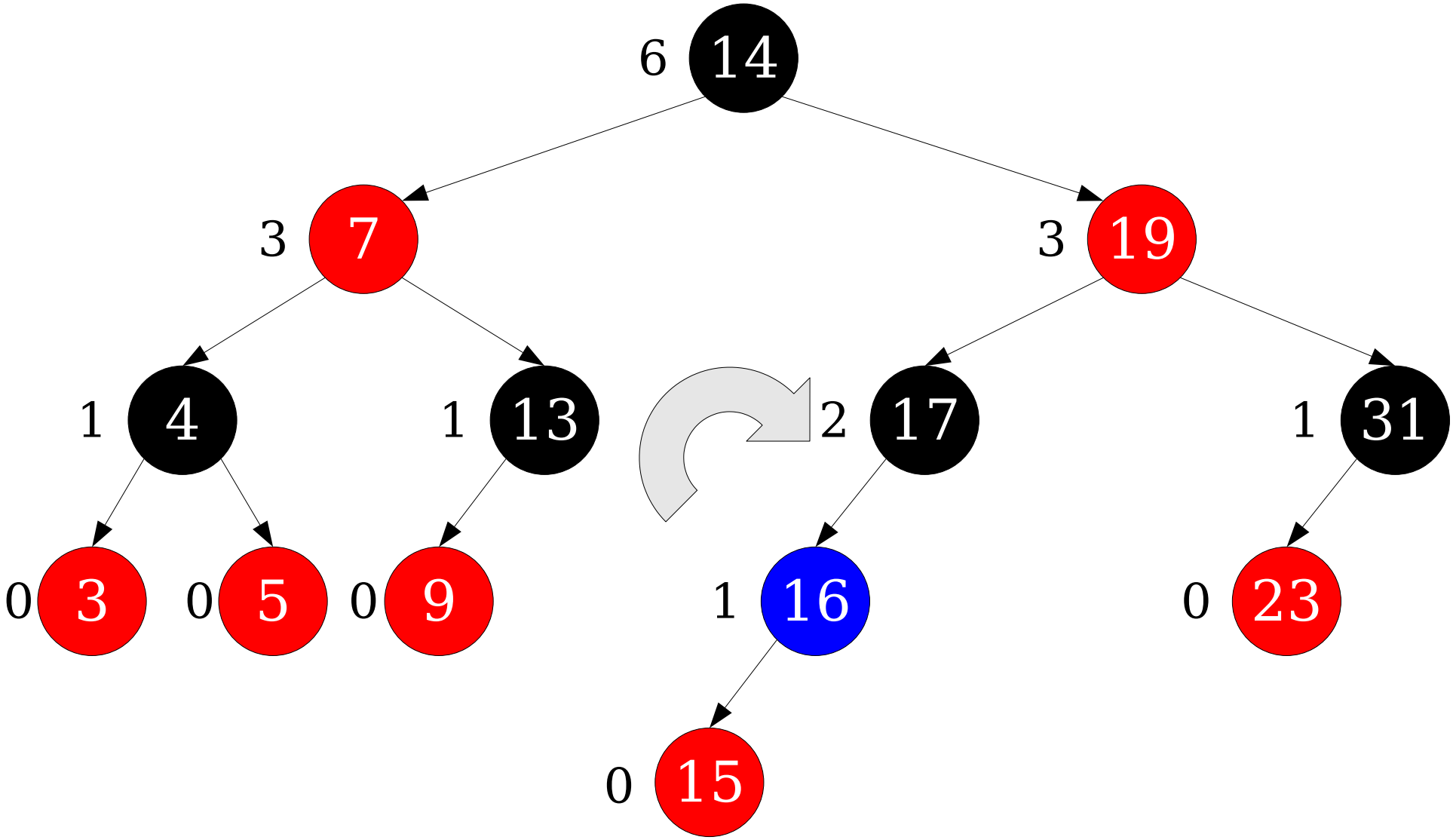


How do we update the numbers after the rotation?

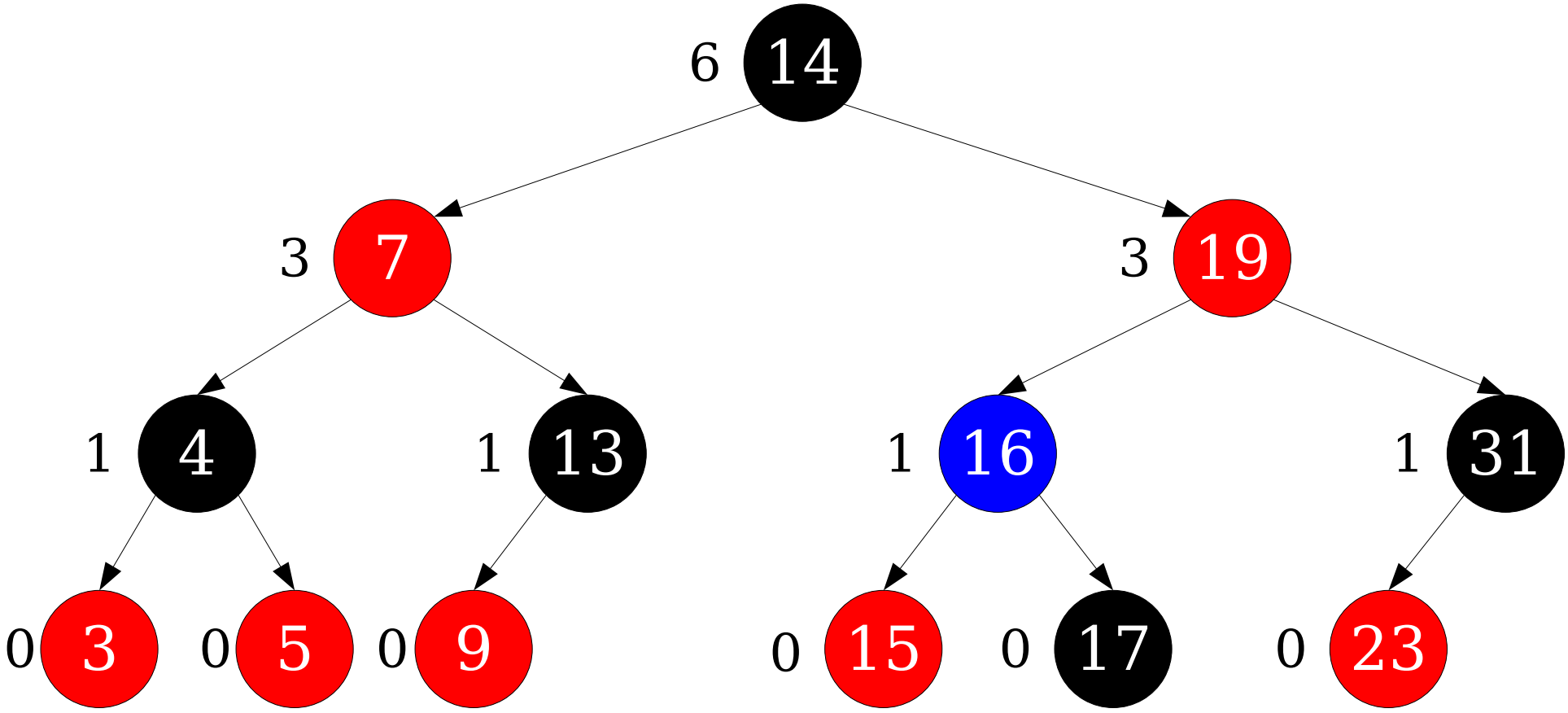
Dynamic Selection



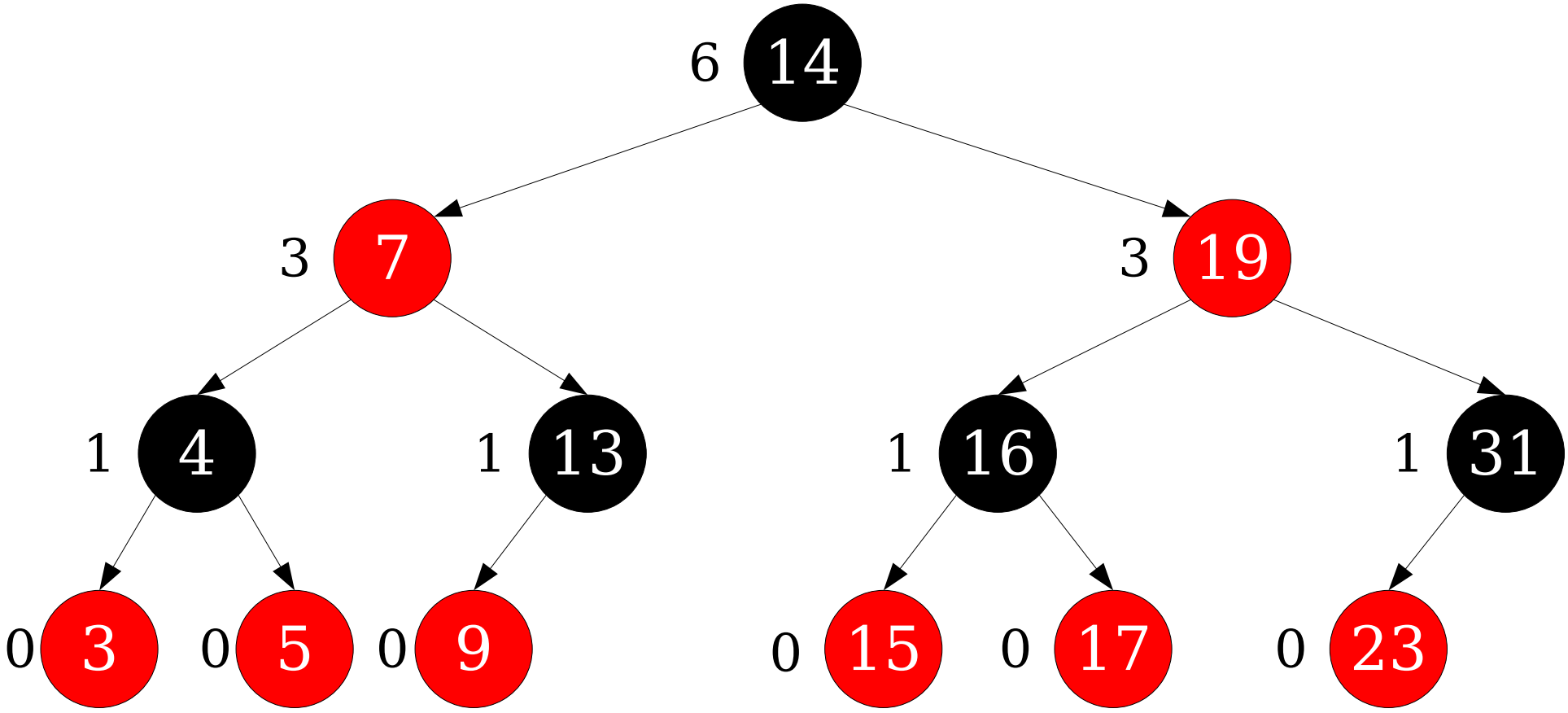
Dynamic Selection



Dynamic Selection



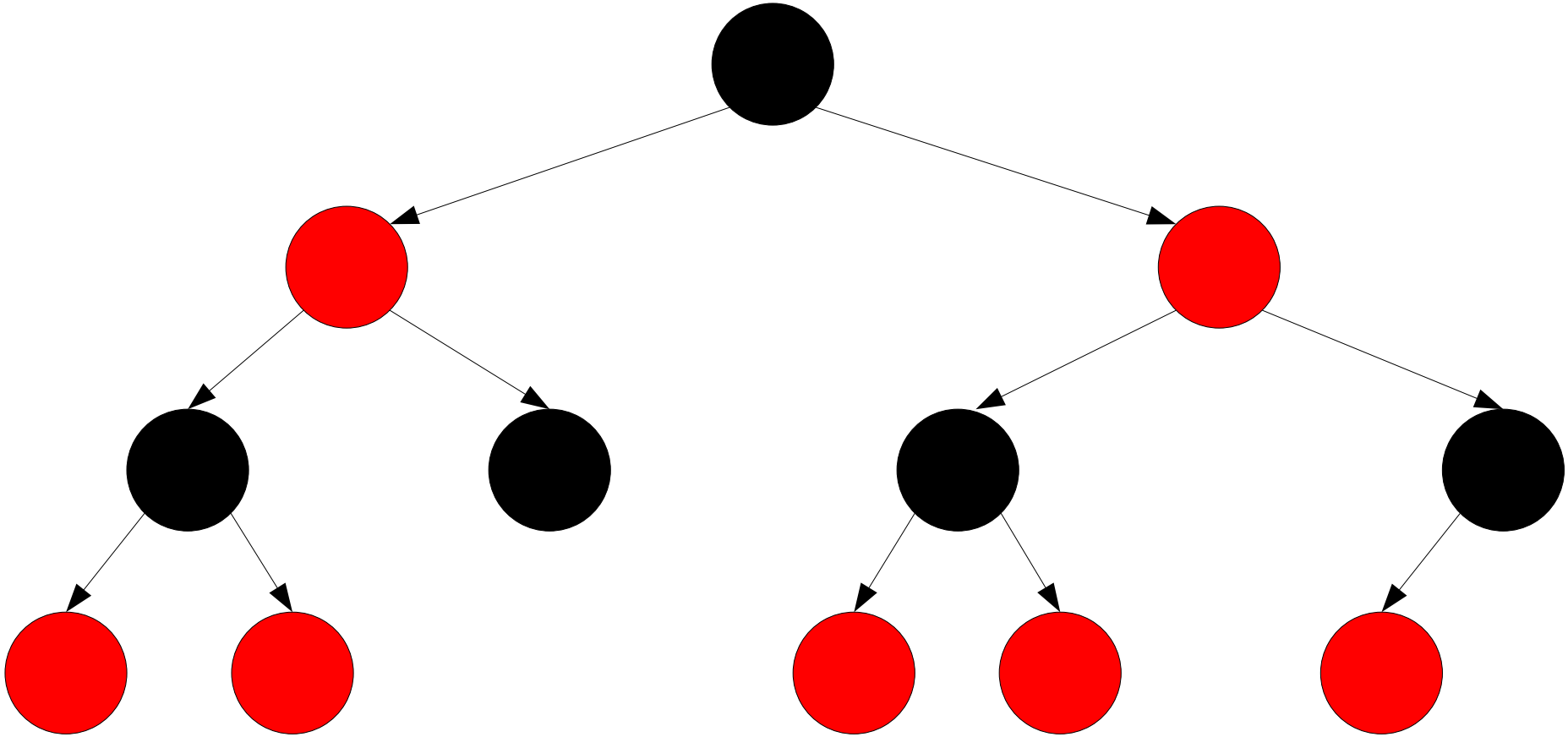
Dynamic Selection



Order Statistic Trees

- This modified red/black tree is called an ***order statistics tree***.
  - Start with a red/black tree.
  - Tag each node with the number of nodes in its left subtree.
  - Use the preceding update rules to preserve values during rotations.
  - Propagate other changes up to the root of the tree.
- Only O(log n) values must be updated on an insertion or deletion and each can be updated in time O(1).
- Supports all BST operations plus ***select*** (find kth order statistic) and ***rank*** (given a key, report its order statistic) in time O(log n).

Generalizing our Idea



Theorem: Suppose we want to cache some computed value in each node of a red/black tree. Provided that the value can be recomputed purely from the node's value and from it's children's values, and provided that each value can be computed in time O(1), then these values can be cached in each node with insertions, lookups, and deletions still taking time O(log n).