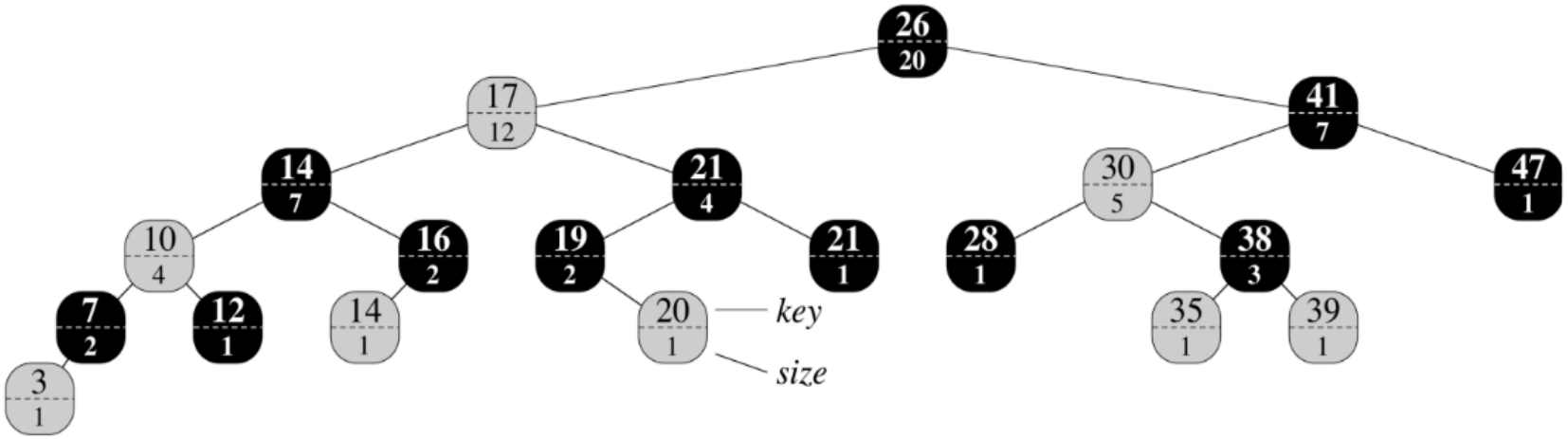
Binary search trees with extra information in the nodes

Add extra information in the nodes

Concrete example of extra information in nodes:

> All nodes store the size of their subtree (i.e. the number of nodes in their subtree).

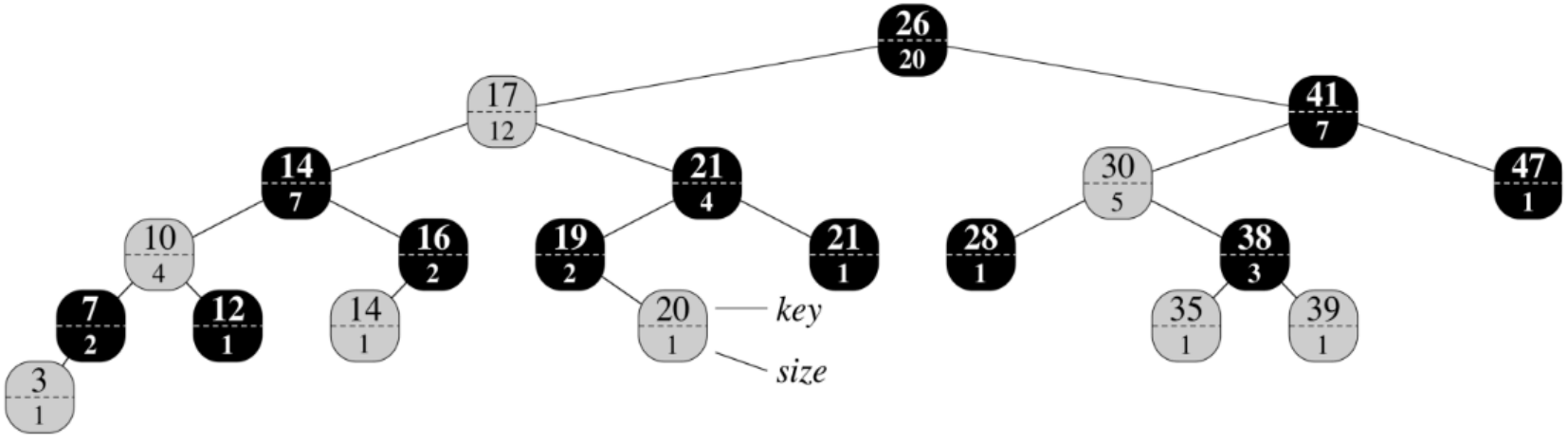
A binary search tree (red-black) with this information added:



New functionality

The goal of adding extra information to the nodes is to provide additional functionality. For example, in the above example (where the size of subtrees is stored in the nodes), the following operations can be performed in O(log n) time:

- Find the rank of a given key.
- Find the key that has a given rank.

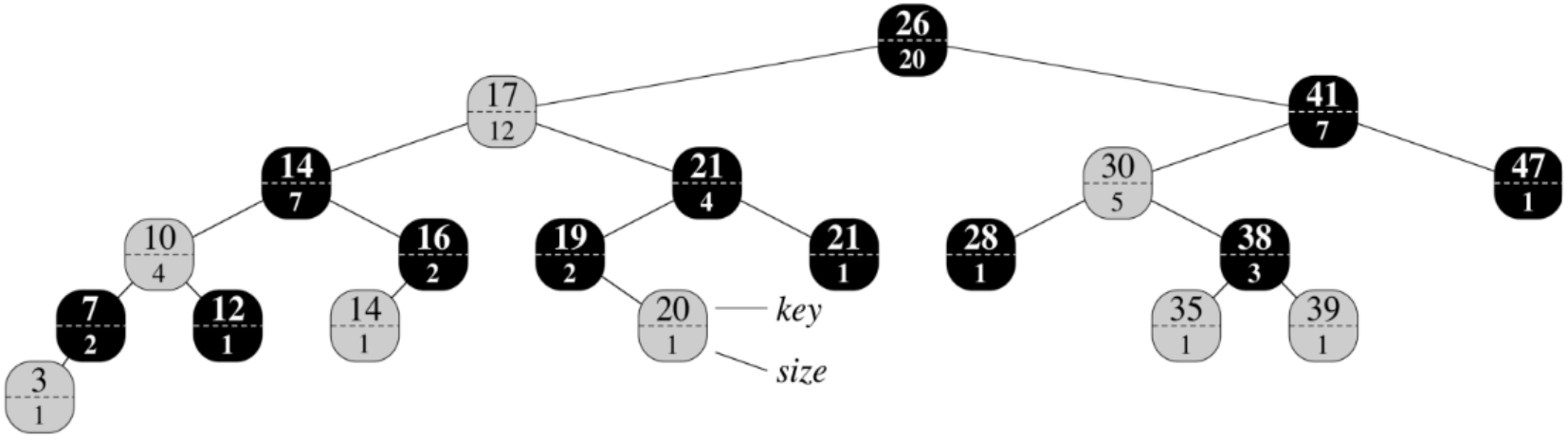


Here, the rank refers to the key's position in sorted order (among the stored ones).

Key:	3	7	10	12	14	14	16	17	19	20	21	···
RanK	1	2	3	4	5	6	7	8	9	10	11	···

New functionality

- Find the rank of a given key.
- Find the key that has a given rank.



OS-RANK(T, x)

$r = x.left.size + 1$
$y = x$
while $y \neq T.root$
  if y == $y.p.right$
    $r = r + y.p.left.size + 1$
  $y = y.p$
return r

OS-SELECT(x, i)

$r = x.left.size + 1$
if i == r
  return x
elseif $i < r$
  return OS-SELECT$(x.left, i)$
else return OS-SELECT$(x.right, i - r)$

Maintain the extra information in the nodes

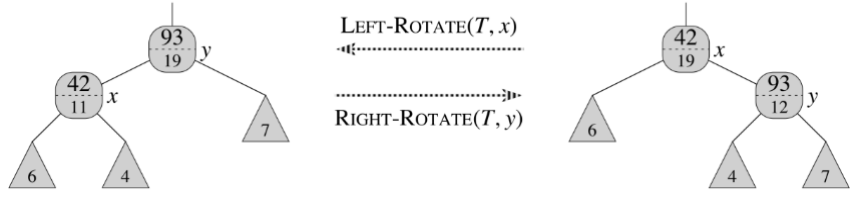
Assume the following property applies to the additional information:

> If a node's two children's values k1 and k2 are already correct, the node's own value k can be calculated in O(1) time. The values of leaves can be calculated in O(1) time.

In the example above:k can be found as $1 + k_1 + k_2$.
From the assumption it follows that the values can be maintained during updates in red-black search trees, and this without changing the O(log n) running time:

- Nodes outside the path from the inserted/deleted node to the root should not have their value changed (bottom-up calculation gives an unchanged result).

- During rebalancing: recalculate for affected nodes after each rotation and after each step upward. Finally, recalculate on the path from the last rebalancing point to the root.



Other examples

The requirement for the additional information in nodes:

> If a node's two children's values k1 and k2 are already correct, the node's own value k can be calculated in O(1) time. The values of leaves can be calculated in O(1) time.

More examples of information that fulfill this:

- Maximum of data values in the subtree.
- Minimum of data values in the subtree.
- Sum of data values in the subtree.

With this information, additional functionality can be added to the tree. For example, one can search for elements with the maximum data value. Or one can find the average of the data values in a node's subtree:
(average = sum of values / number of nodes).