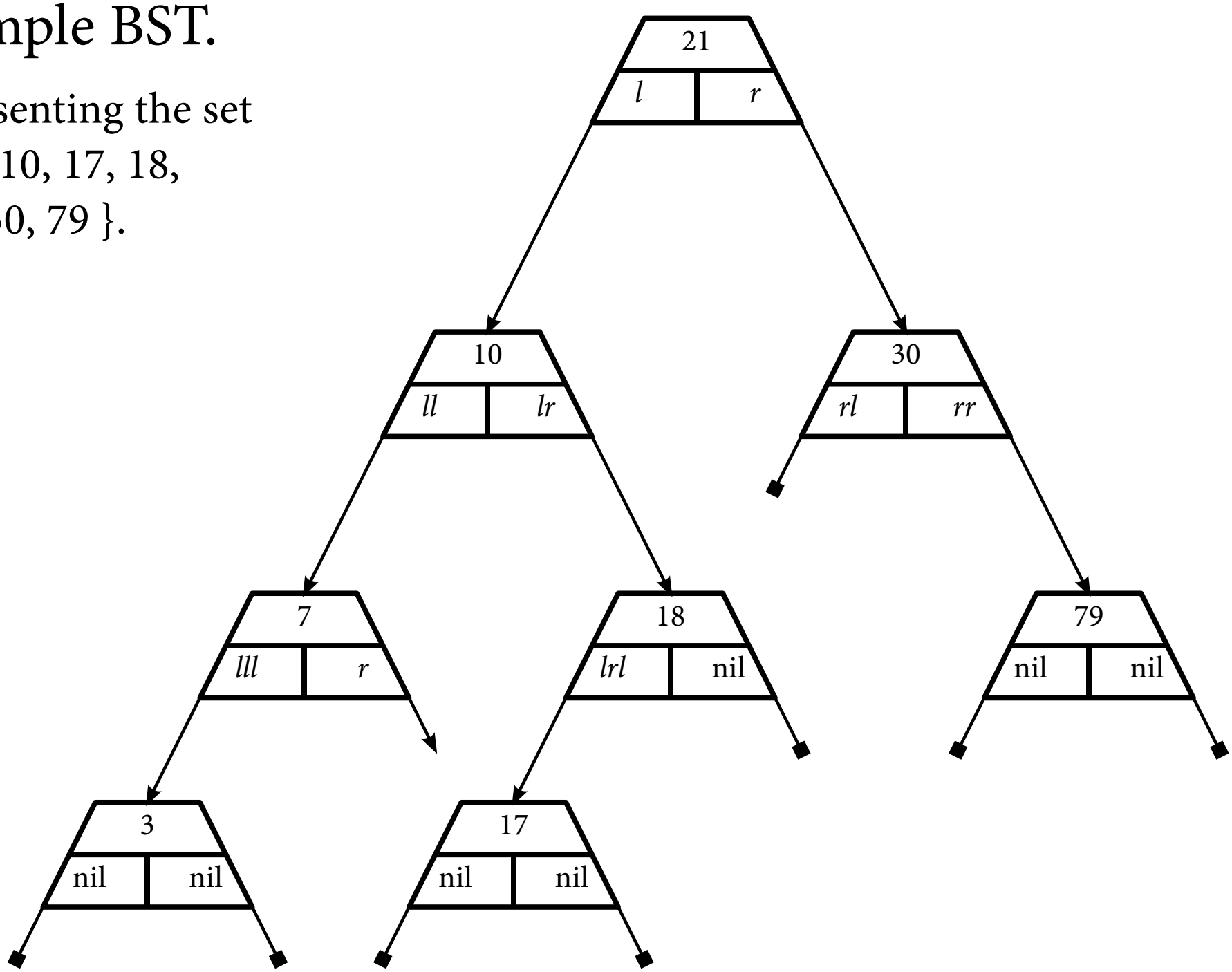
Example BST.

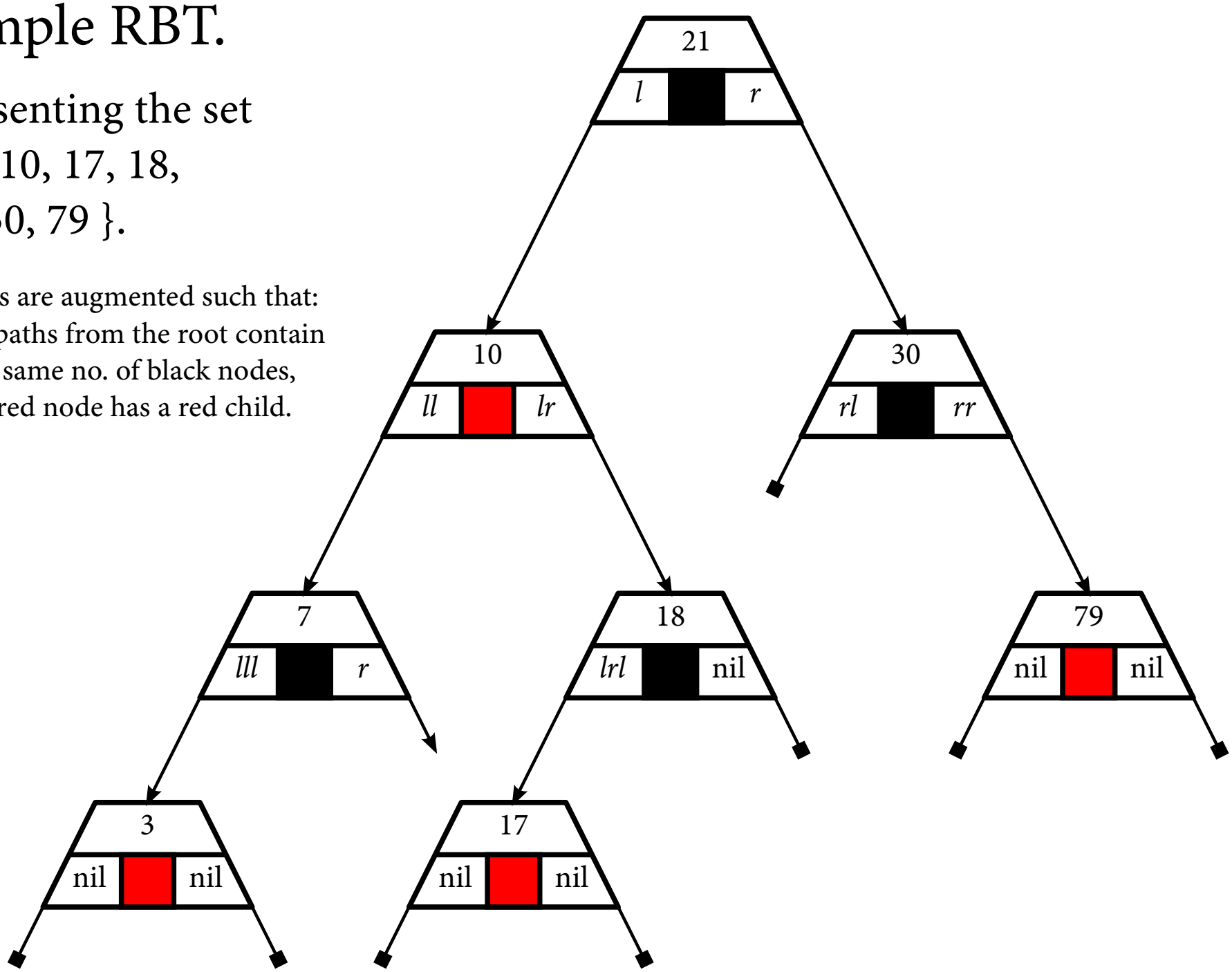
Representing the set
{ 3, 7, 10, 17, 18,
21, 30, 79 }.



Example RBT.

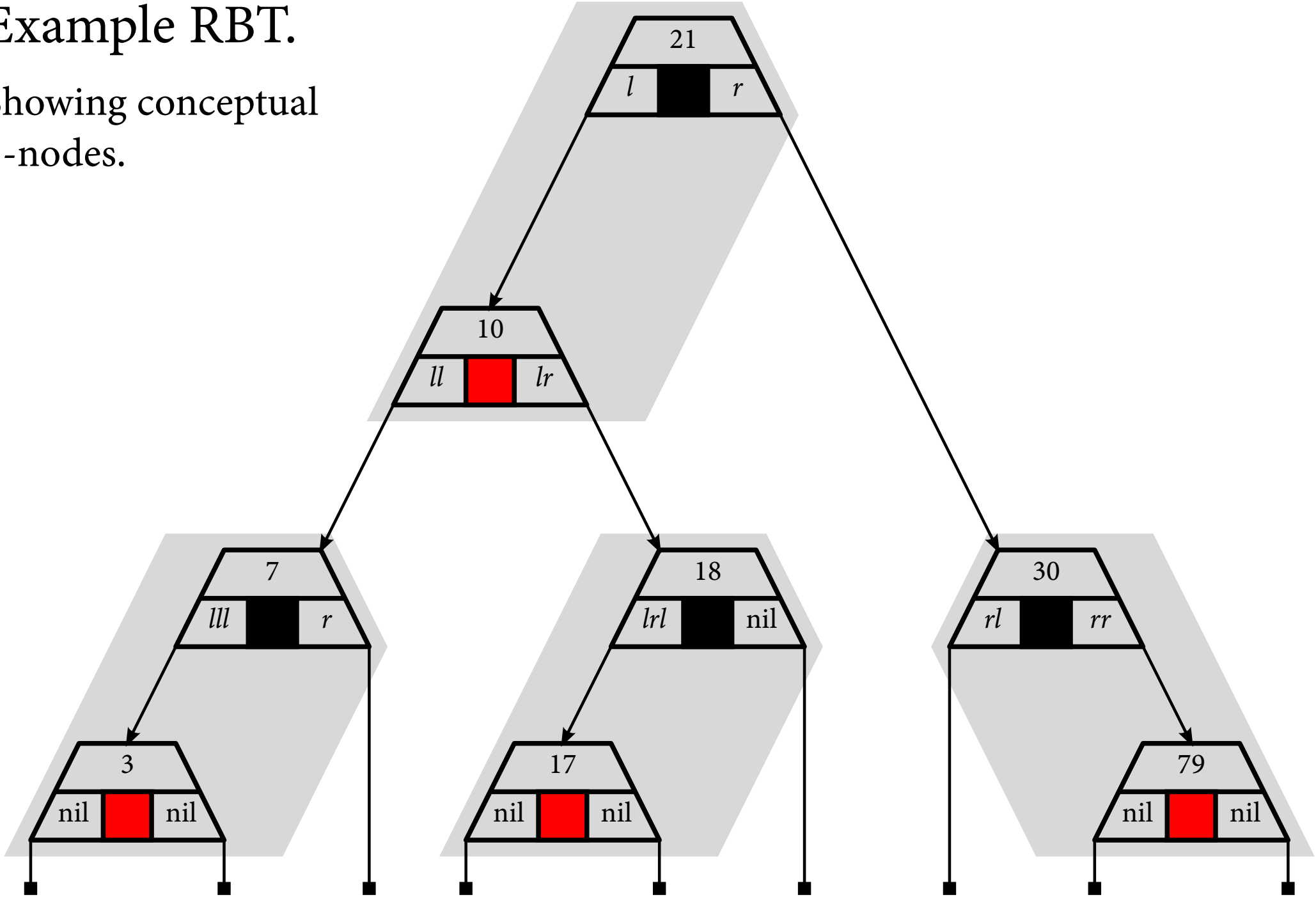
Representing the set
{ 3, 7, 10, 17, 18,
21, 30, 79 }.

The nodes are augmented such that:
(1) all paths from the root contain the same no. of black nodes,
(2) no red node has a red child.

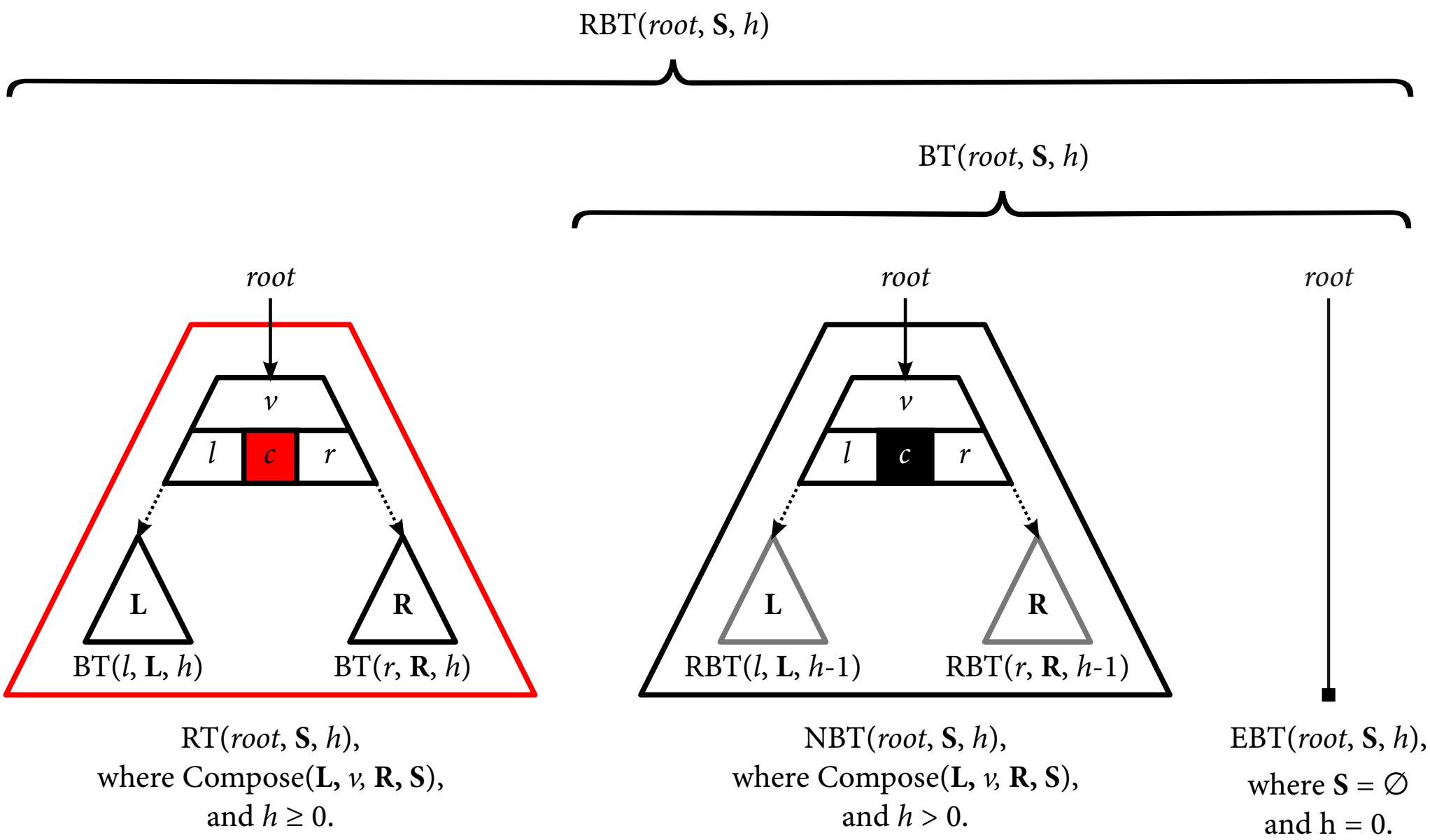


Example RBT.

Showing conceptual 3-nodes.



RBT predicates.



RT(*root*, **S**, *h*),
where Compose(**L**, *v*, **R**, **S**),
and $h \geq 0$.

NBT(*root*, **S**, *h*),
where Compose(**L**, *v*, **R**, **S**),
and $h > 0$.

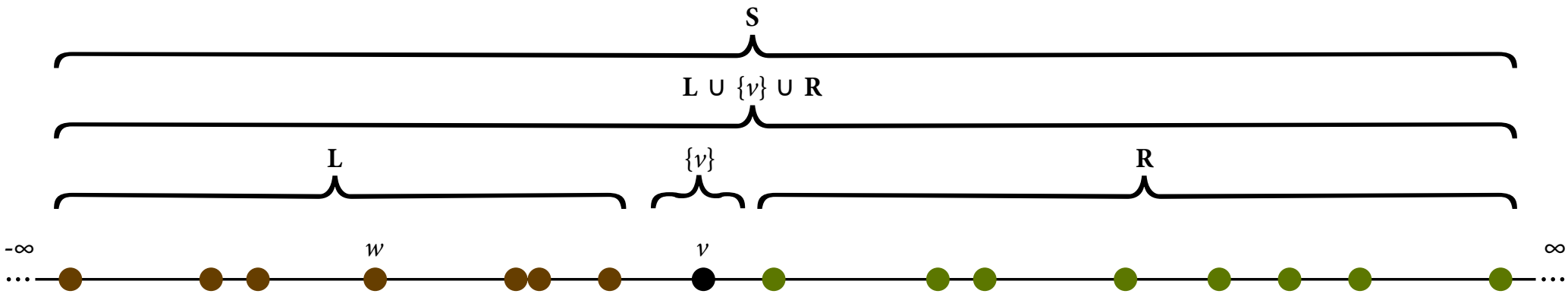
EBT(*root*, **S**, *h*),
where **S** = ∅
and h = 0.

Compose($\mathbf{L}$, v, $\mathbf{R}$, $\mathbf{S}$)

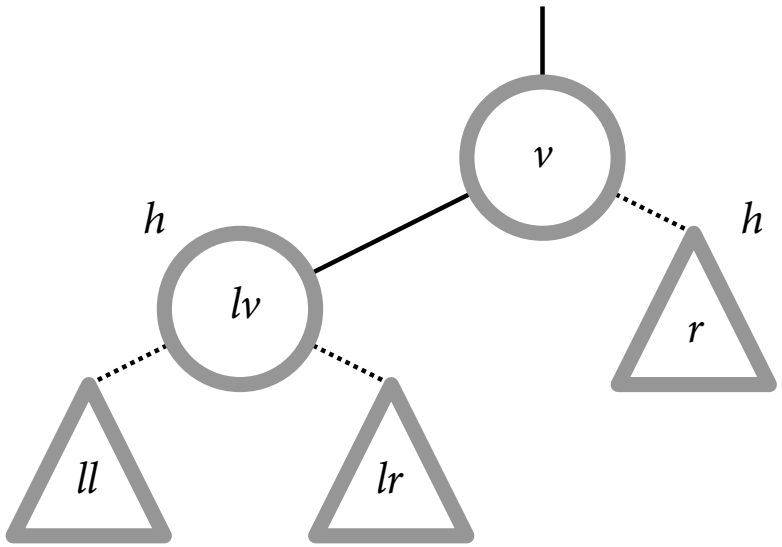
$\mathbf{L} \cup \{v\} \cup \mathbf{R} = \mathbf{S} \wedge$

$\forall l \in \mathbf{L}.\ l < v \wedge$

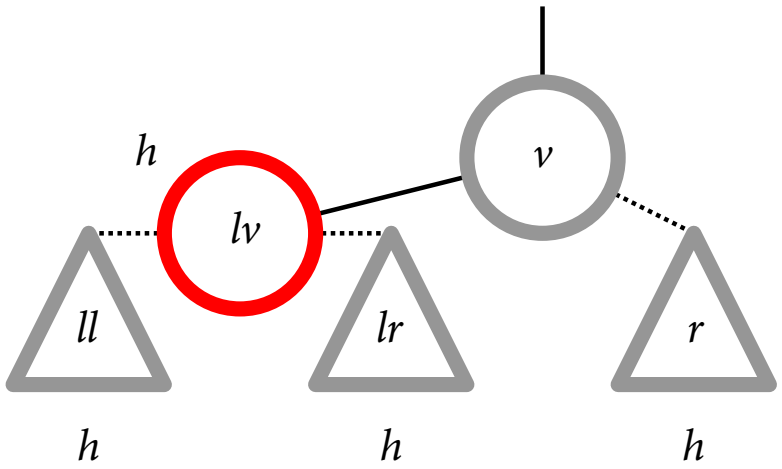
$\forall r \in \mathbf{R}.\ v < r.$



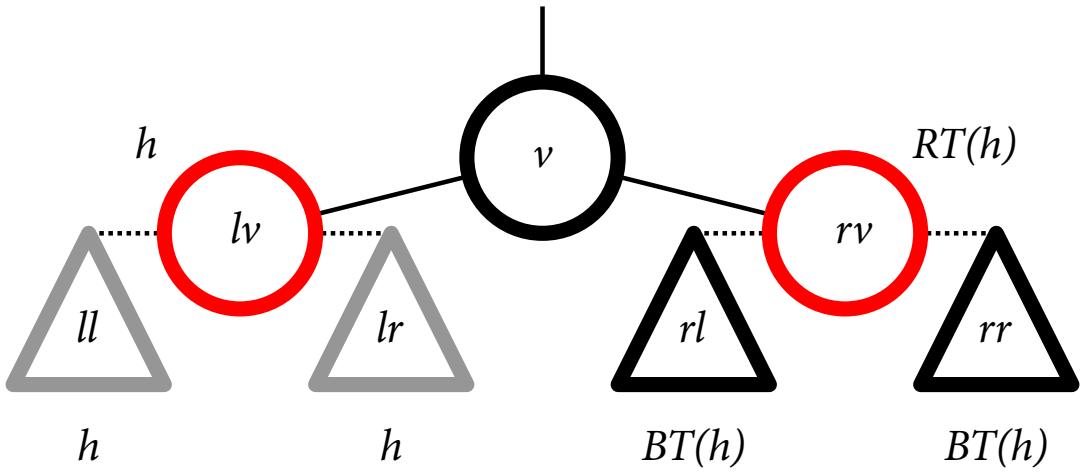
```
Node* balance(Node* root, int dir) {
  if (red(root.left)) {
    if (red(root.right)) {
      root.left  = blacken(root.left);
      root.right = blacken(root.right);
      root.black = false;
    }
    else {
      if (red(root.left.left)) {
        root = rb.rotate.single(
          root, RIGHT);
      }
      else {
        if (red(root.left.right)) {
          root = rb.rotate.dbl(
            root, RIGHT);
        }
        else {
        }
      }
    }
  }
  else {
  }
  return root;
}
```



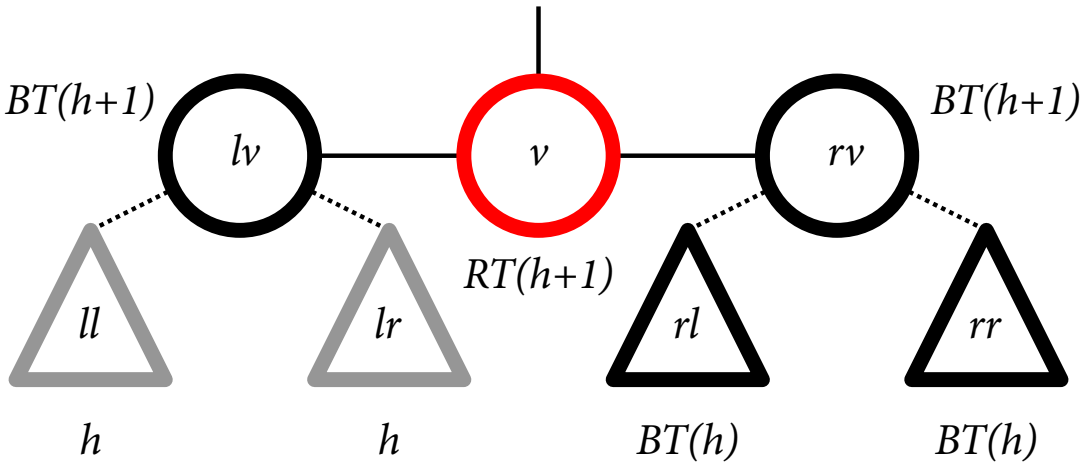
```
Node* balance(Node* root, int dir) {
  if (red(root.left)) {
    if (red(root.right)) {
      root.left  = blacken(root.left);
      root.right = blacken(root.right);
      root.black = false;
    }
    else {
      if (red(root.left.left)) {
        root = rb.rotate.single(
          root, RIGHT);
      }
      else {
        if (red(root.left.right)) {
          root = rb.rotate.dbl(
            root, RIGHT);
        }
        else {
        }
      }
    }
  }
  else {
  }
  return root;
}
```



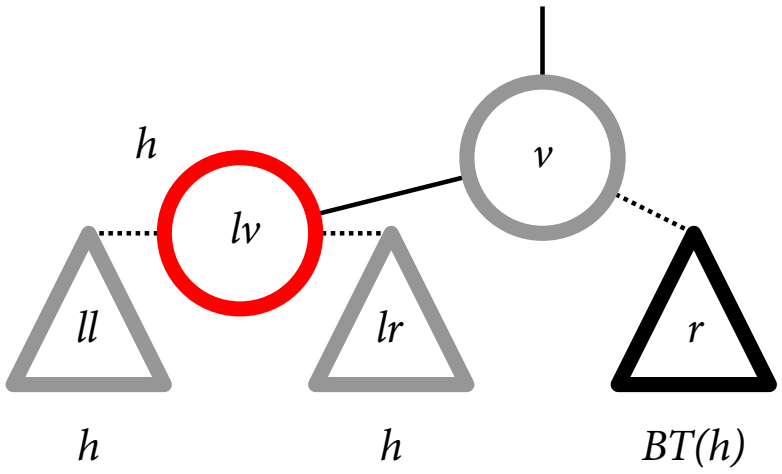
```
Node* balance(Node* root, int dir) {
  if (red(root.left)) {
    if (red(root.right)) {
      root.left  = blacken(root.left);
      root.right = blacken(root.right);
      root.black = false;
    }
    else {
      if (red(root.left.left)) {
        root = rb.rotate.single(
          root, RIGHT);
      }
      else {
        if (red(root.left.right)) {
          root = rb.rotate.dbl(
            root, RIGHT);
        }
        else {
        }
      }
    }
  }
  else {
  }
  return root;
}
```



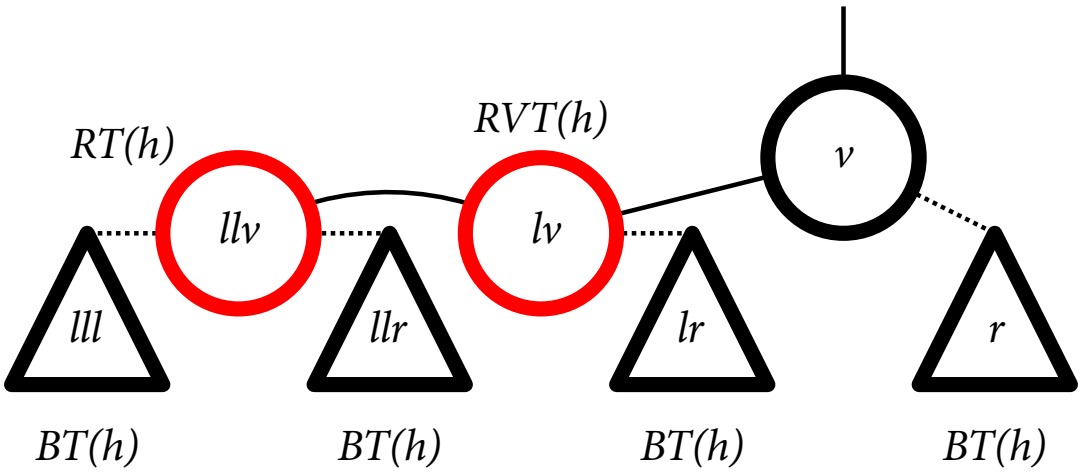
```
Node* balance(Node* root, int dir) {
  if (red(root.left)) {
    if (red(root.right)) {
      root.left  = blacken(root.left);
      root.right = blacken(root.right);
      root.black = false;
    }
    else {
      if (red(root.left.left)) {
        root = rb.rotate.single(
          root, RIGHT);
      }
      else {
        if (red(root.left.right)) {
          root = rb.rotate.dbl(
            root, RIGHT);
        }
        else {
        }
      }
    }
  }
  else {
  }
  return root;
}
```



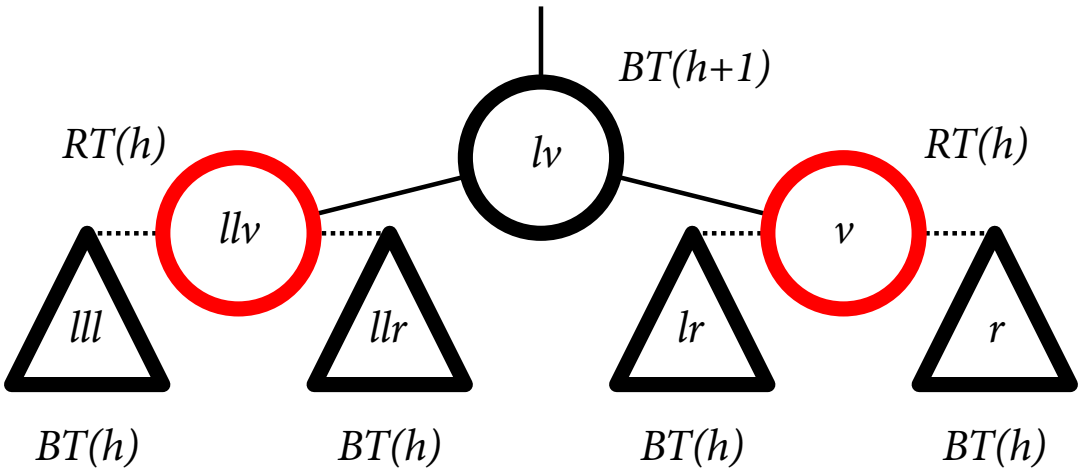
```
Node* balance(Node* root, int dir) {
  if (red(root.left)) {
    if (red(root.right)) {
      root.left  = blacken(root.left);
      root.right = blacken(root.right);
      root.black = false;
    }
    else {
      if (red(root.left.left)) {
        root = rb.rotate.single(
          root, RIGHT);
      }
      else {
        if (red(root.left.right)) {
          root = rb.rotate.dbl(
            root, RIGHT);
        }
        else {
        }
      }
    }
  }
  else {
  }
  return root;
}
```



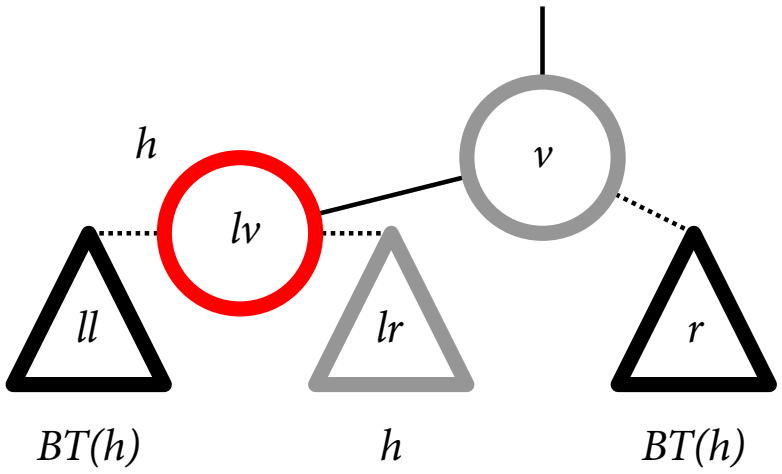
```
Node* balance(Node* root, int dir) {
  if (red(root.left)) {
    if (red(root.right)) {
      root.left  = blacken(root.left);
      root.right = blacken(root.right);
      root.black = false;
    }
    else {
      if (red(root.left.left)) {
        root = rb.rotate.single(
          root, RIGHT);
      }
      else {
        if (red(root.left.right)) {
          root = rb.rotate.dbl(
            root, RIGHT);
        }
        else {
        }
      }
    }
  }
  else {
  }
  return root;
}
```



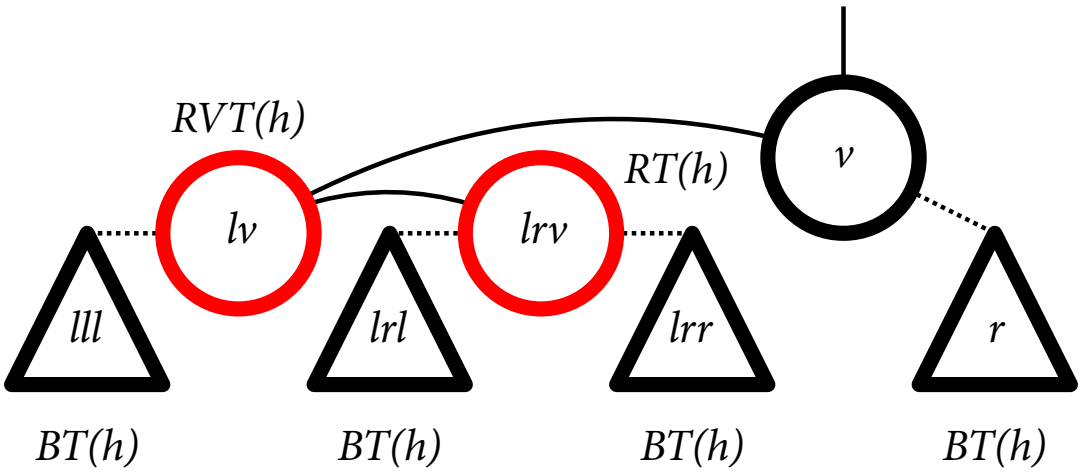
```
Node* balance(Node* root, int dir) {
  if (red(root.left)) {
    if (red(root.right)) {
      root.left  = blacken(root.left);
      root.right = blacken(root.right);
      root.black = false;
    }
    else {
      if (red(root.left.left)) {
        root = rb.rotate.single(
          root, RIGHT);
      }
      else {
        if (red(root.left.right)) {
          root = rb.rotate.dbl(
            root, RIGHT);
        }
        else {
        }
      }
    }
  }
  else {
  }
  return root;
}
```



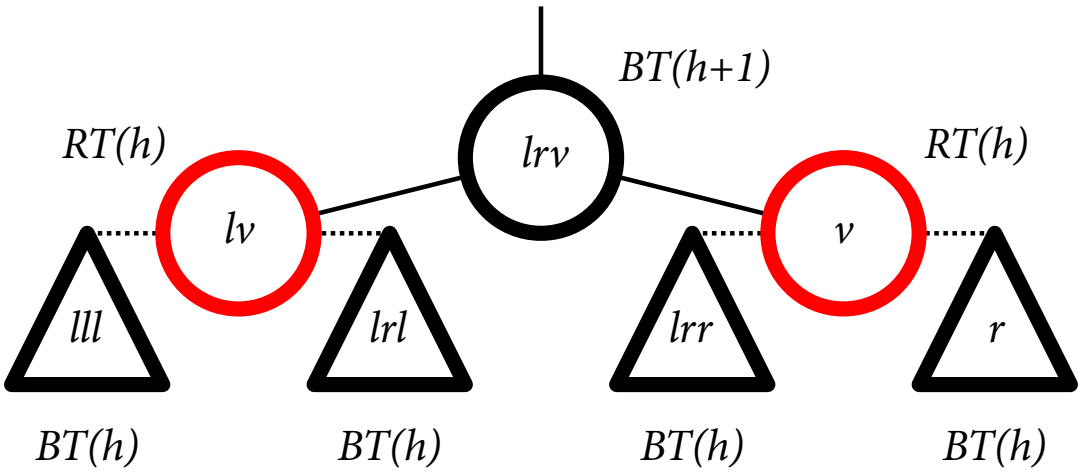
```
Node* balance(Node* root, int dir) {
  if (red(root.left)) {
    if (red(root.right)) {
      root.left  = blacken(root.left);
      root.right = blacken(root.right);
      root.black = false;
    }
    else {
      if (red(root.left.left)) {
        root = rb.rotate.single(
          root, RIGHT);
      }
      else {
        if (red(root.left.right)) {
          root = rb.rotate.dbl(
            root, RIGHT);
        }
        else {
        }
      }
    }
  }
  else {
  }
  return root;
}
```



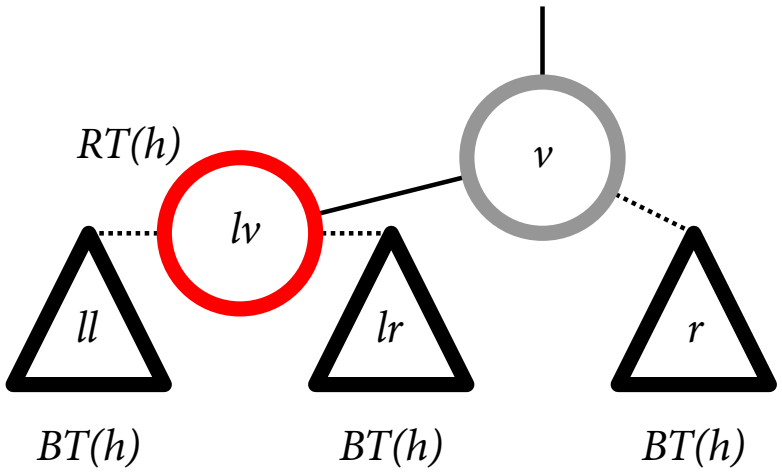
```
Node* balance(Node* root, int dir) {
  if (red(root.left)) {
    if (red(root.right)) {
      root.left  = blacken(root.left);
      root.right = blacken(root.right);
      root.black = false;
    }
    else {
      if (red(root.left.left)) {
        root = rb.rotate.single(
          root, RIGHT);
      }
      else {
        if (red(root.left.right)) {
          root = rb.rotate.dbl(
            root, RIGHT);
        }
        else {
        }
      }
    }
  }
  else {
  }
  return root;
}
```



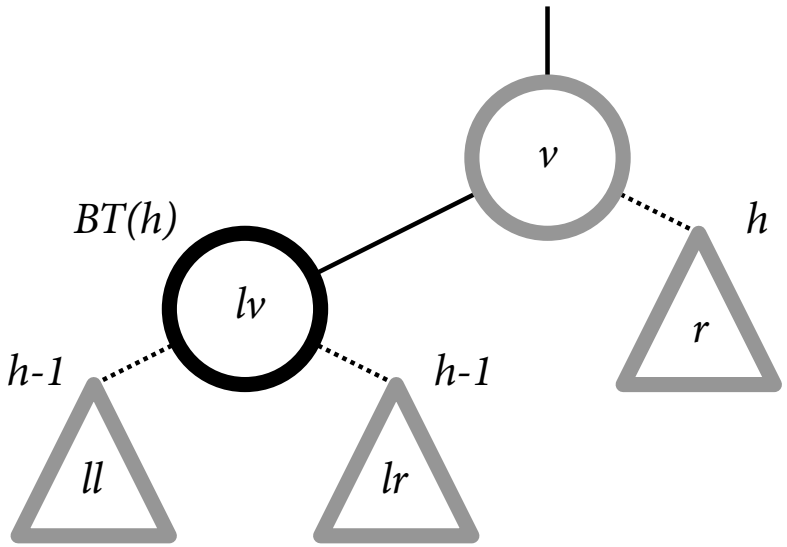
```
Node* balance(Node* root, int dir) {
  if (red(root.left)) {
    if (red(root.right)) {
      root.left  = blacken(root.left);
      root.right = blacken(root.right);
      root.black = false;
    }
    else {
      if (red(root.left.left)) {
        root = rb.rotate.single(
          root, RIGHT);
      }
      else {
        if (red(root.left.right)) {
          root = rb.rotate.dbl(
            root, RIGHT);
        }
        else {
        }
      }
    }
  }
  else {
  }
  return root;
}
```



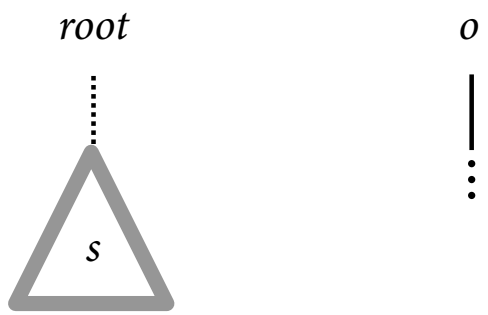
```
Node* balance(Node* root, int dir) {
  if (red(root.left)) {
    if (red(root.right)) {
      root.left  = blacken(root.left);
      root.right = blacken(root.right);
      root.black = false;
    }
    else {
      if (red(root.left.left)) {
        root = rb.rotate.single(
          root, RIGHT);
      }
      else {
        if (red(root.left.right)) {
          root = rb.rotate.dbl(
            root, RIGHT);
        }
        else {
        }
      }
    }
  }
  else {
  }
  return root;
}
```



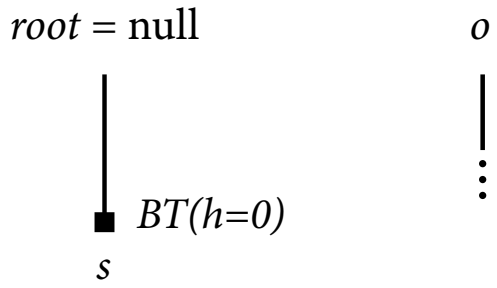
```
Node* balance(Node* root, int dir) {
  if (red(root.left)) {
    if (red(root.right)) {
      root.left  = blacken(root.left);
      root.right = blacken(root.right);
      root.black = false;
    }
    else {
      if (red(root.left.left)) {
        root = rb.rotate.single(
          root, RIGHT);
      }
      else {
        if (red(root.left.right)) {
          root = rb.rotate.dbl(
            root, RIGHT);
        }
        else {
        }
      }
    }
  }
  else {
  }
  return root;
}
```



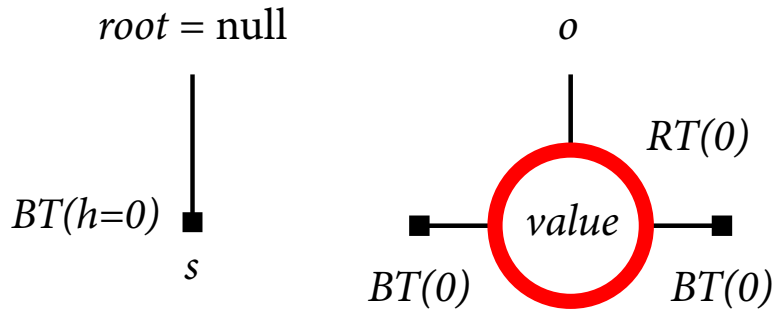
```
Node* insert_aux(Node* root, int value) {
  Node* o;
  if (root == null) {
    o = new Node(value);
  }
  else {
    if (root.value == value) {
      o = root;
    }
    else {
      if (value < root.value) {
        root.left = insert_aux(root.left, value);
        root.left = balance(root, LEFT);
        o = root;
      }
      else {
        // Symmetrical
      }
    }
  }
  return o;
}
```



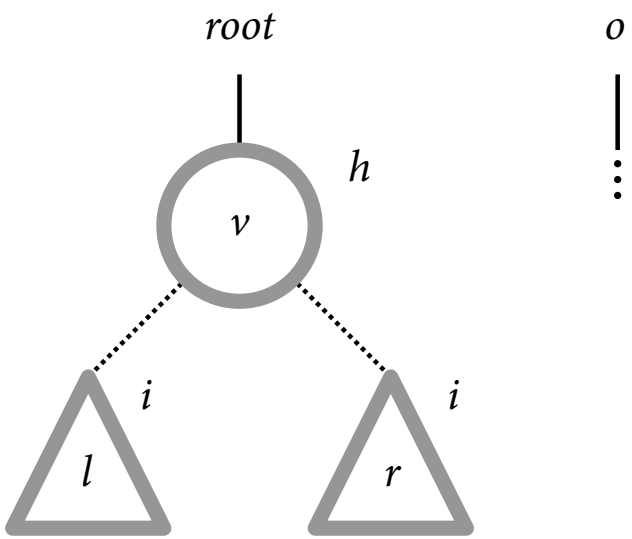
```
Node* insert_aux(Node* root, int value) {
  Node* o;
  if (root == null) {
    o = new Node(value);
  }
  else {
    if (root.value == value) {
      o = root;
    }
    else {
      if (value < root.value) {
        root.left = insert_aux(root.left, value);
        root.left = balance(root, LEFT);
        o = root;
      }
      else {
        // Symmetrical
      }
    }
  }
  return o;
}
```



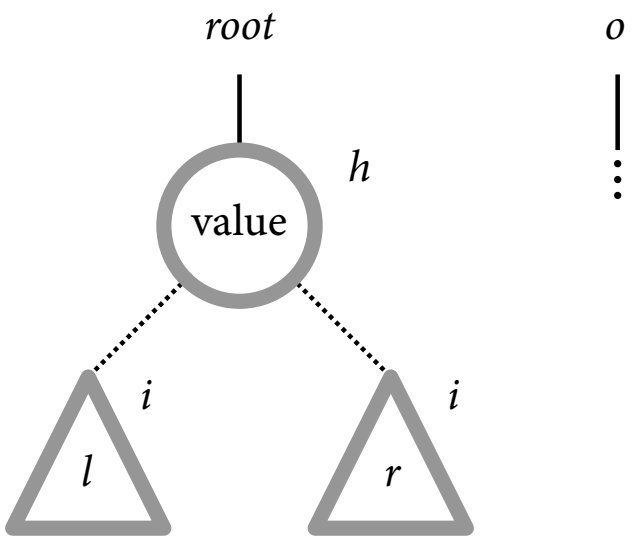
```
Node* insert_aux(Node* root, int value) {
  Node* o;
  if (root == null) {
    o = new Node(value);
  }
  else {
    if (root.value == value) {
      o = root;
    }
    else {
      if (value < root.value) {
        root.left = insert_aux(root.left, value);
        root.left = balance(root, LEFT);
        o = root;
      }
      else {
        // Symmetrical
      }
    }
  }
  return o;
}
```



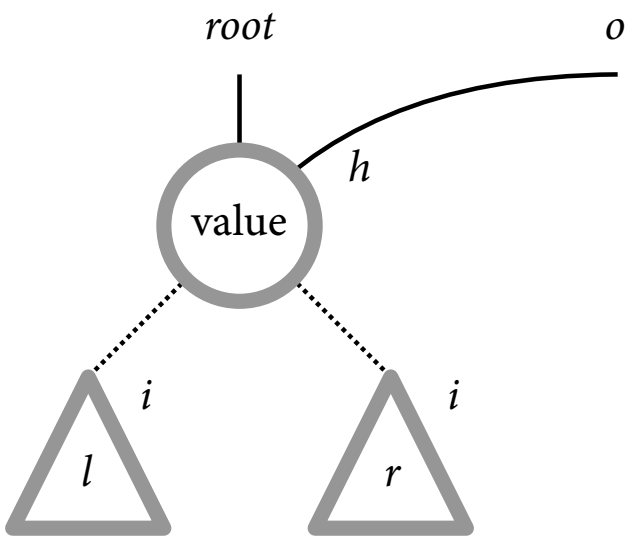
```
Node* insert_aux(Node* root, int value) {
  Node* o;
  if (root == null) {
    o = new Node(value);
  }
  else {
    if (root.value == value) {
      o = root;
    }
    else {
      if (value < root.value) {
        root.left = insert_aux(root.left, value);
        root.left = balance(root, LEFT);
        o = root;
      }
      else {
        // Symmetrical
      }
    }
  }
  return o;
}
```



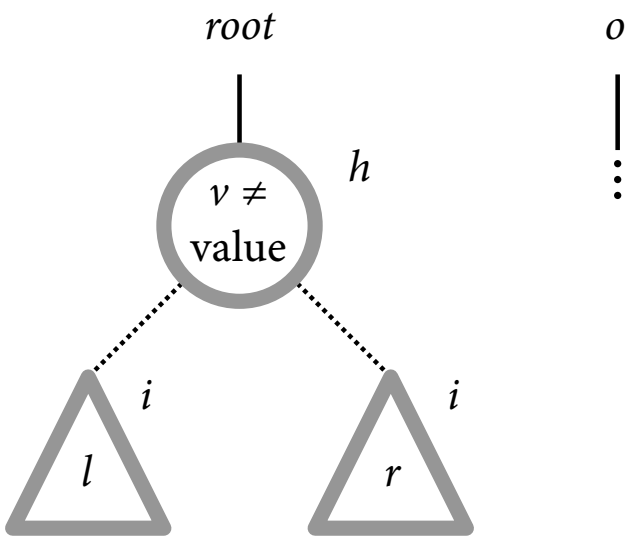
```
Node* insert_aux(Node* root, int value) {
  Node* o;
  if (root == null) {
    o = new Node(value);
  }
  else {
    if (root.value == value) {
      o = root;
    }
    else {
      if (value < root.value) {
        root.left = insert_aux(root.left, value);
        root.left = balance(root, LEFT);
      }
      else {
        // Symmetrical
      }
    }
  }
  return o;
}
```



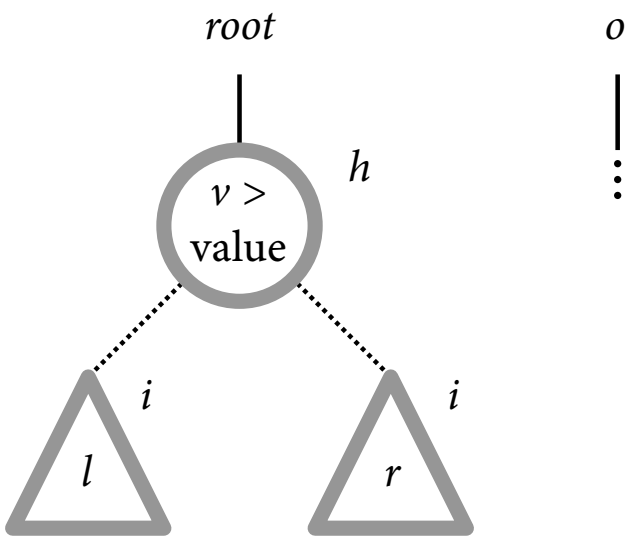
```
Node* insert_aux(Node* root, int value) {
  Node* o;
  if (root == null) {
    o = new Node(value);
  }
  else {
    if (root.value == value) {
      o = root;
    }
    else {
      if (value < root.value) {
        root.left = insert_aux(root.left, value);
        root.left = balance(root, LEFT);
        o = root;
      }
      else {
        // Symmetrical
      }
    }
  }
  return o;
}
```



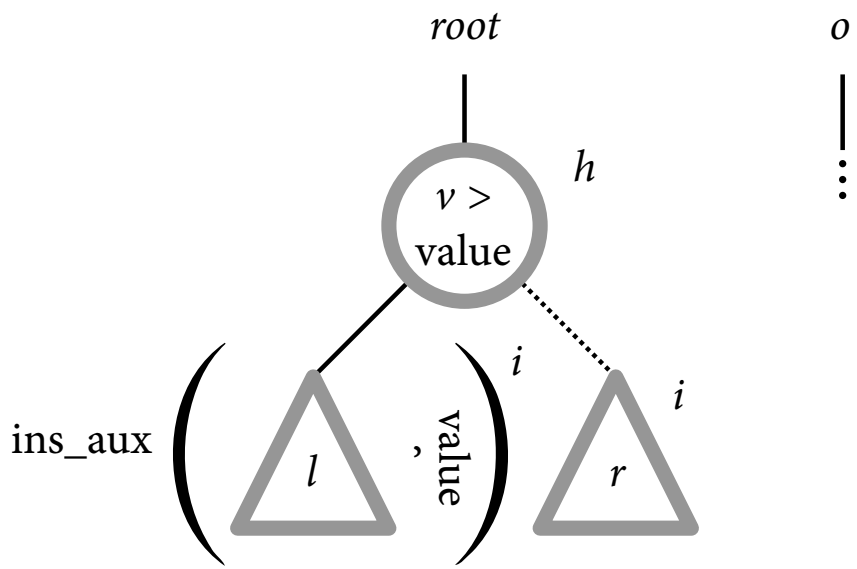
```
Node* insert_aux(Node* root, int value) {
  Node* o;
  if (root == null) {
    o = new Node(value);
  }
  else {
    if (root.value == value) {
      o = root;
    }
    else {
      if (value < root.value) {
        root.left = insert_aux(root.left, value);
        root.left = balance(root, LEFT);
        o = root;
      }
      else {
        // Symmetrical
      }
    }
  }
  return o;
}
```



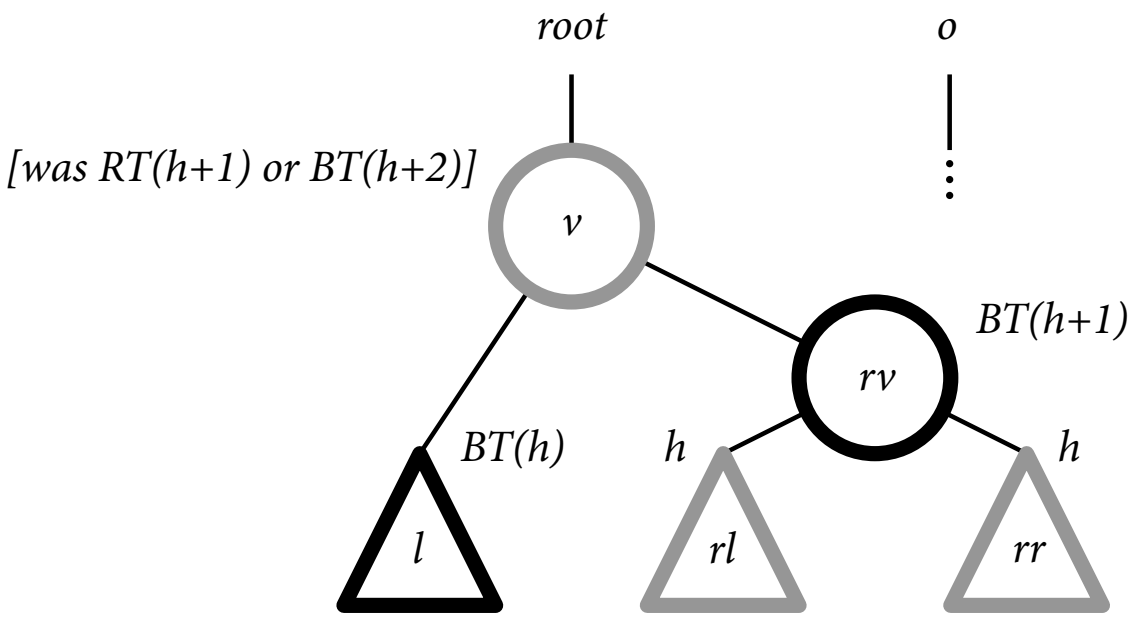
```
Node* insert_aux(Node* root, int value) {
  Node* o;
  if (root == null) {
    o = new Node(value);
  }
  else {
    if (root.value == value) {
      o = root;
    }
    else {
      if (value < root.value) {
        root.left = insert_aux(root.left, value);
        root.left = balance(root, LEFT);
        o = root;
      }
      else {
        // Symmetrical
      }
    }
  }
  return o;
}
```



```
Node* insert_aux(Node* root, int value) {
  Node* o;
  if (root == null) {
    o = new Node(value);
  }
  else {
    if (root.value == value) {
      o = root;
    }
    else {
      if (value < root.value) {
        root.left = insert_aux(root.left, value);
        root.left = balance(root, LEFT);
        o = root;
      }
      else {
        // Symmetrical
      }
    }
  }
  return o;
}
```



```
Node* balB_left(Node* root, bool* fixed) {
  Node* o;
  if (red(root.right.right)) {
    bool oldColor = root.black;
    root = rb.rotate.single_left(root);
    o = clr(oldColor, root, fixed);
  }
  else {
    if (red(root.right.left)) {
      o = clr(root.black, rb.rotate.dbl_left(root), fixed);
    }
    else {
      *fixed = !root.black;
      root.black = true;
      root.right.black = false;
      o = root;
    }
  }
  return o;
}
```



```
Node* balB_left(Node* root, bool* fixed) {
  Node* o;
  if (red(root.right.right)) {
    bool oldColor = root.black;
    root = rb.rotate.single_left(root);
    o = clr(oldColor, root, fixed);
  }
  else {
    if (red(root.right.left)) {
      o = clr(root.black, rb.rotate.dbl_left(root), fixed);
    }
    else {
      *fixed = !root.black;
      root.black = true;
      root.right.black = false;
      o = root;
    }
  }
  return o;
}
```