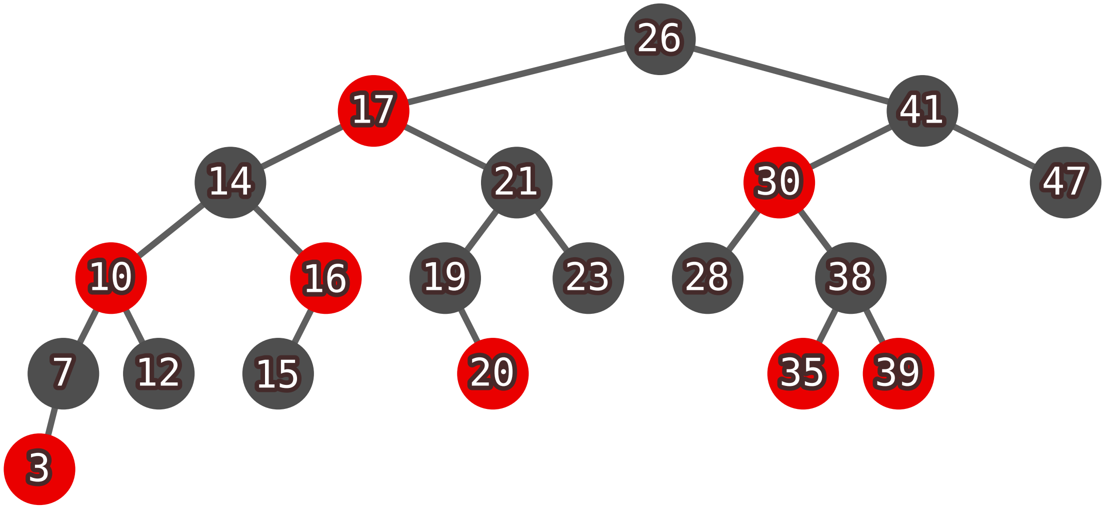
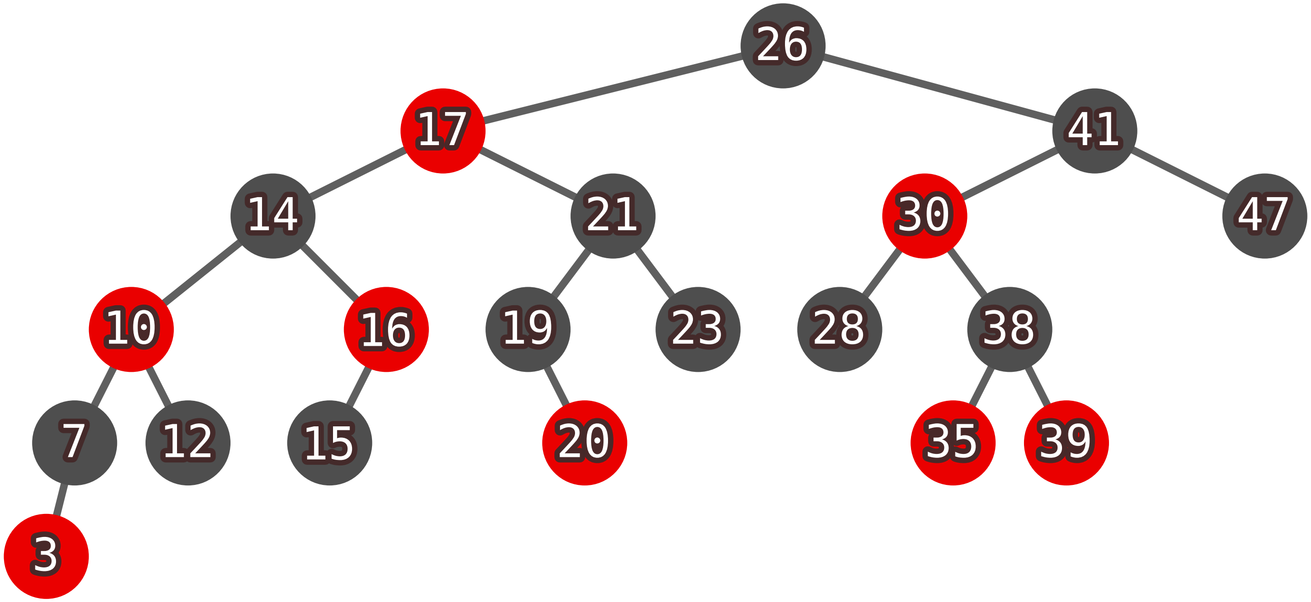
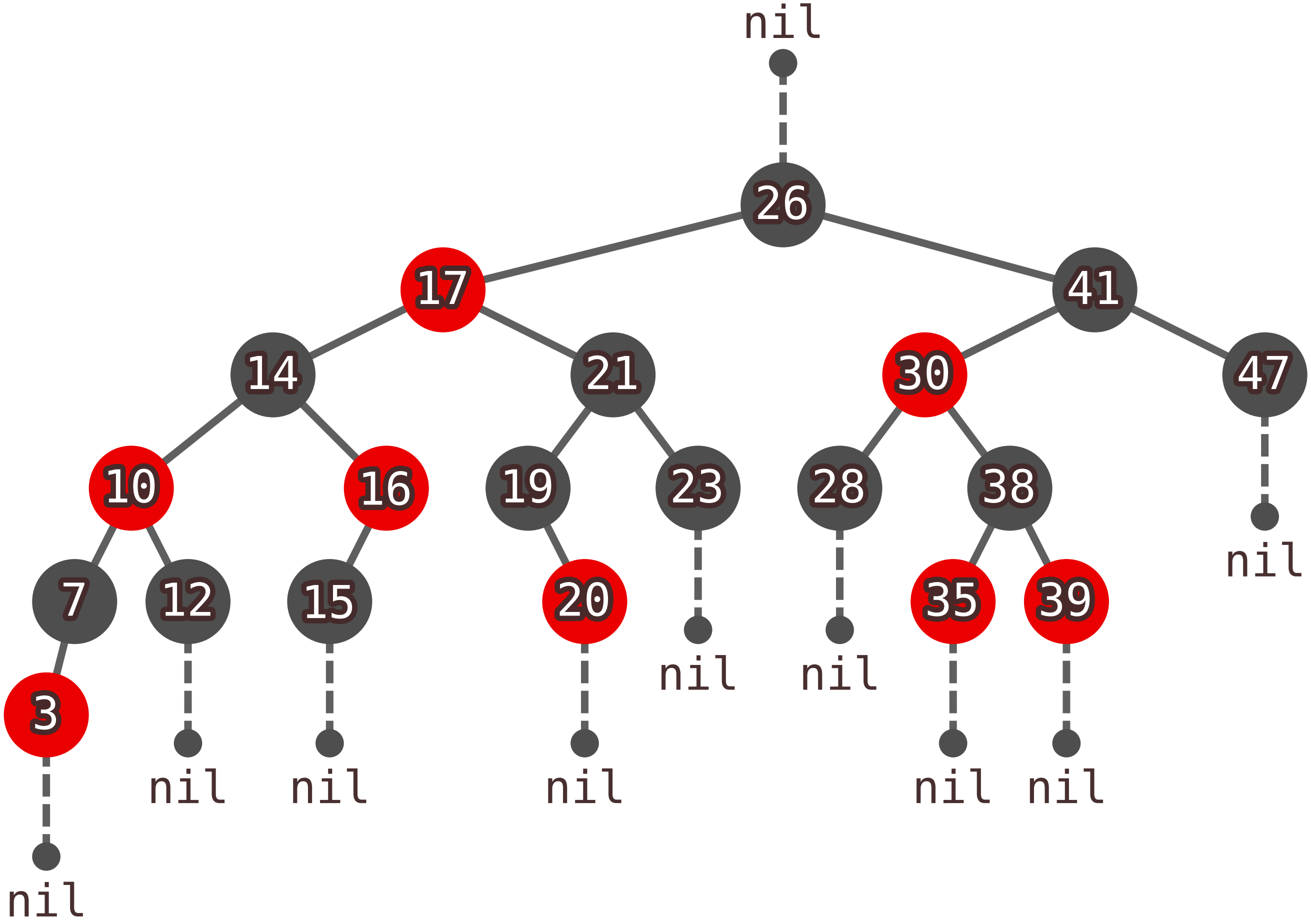
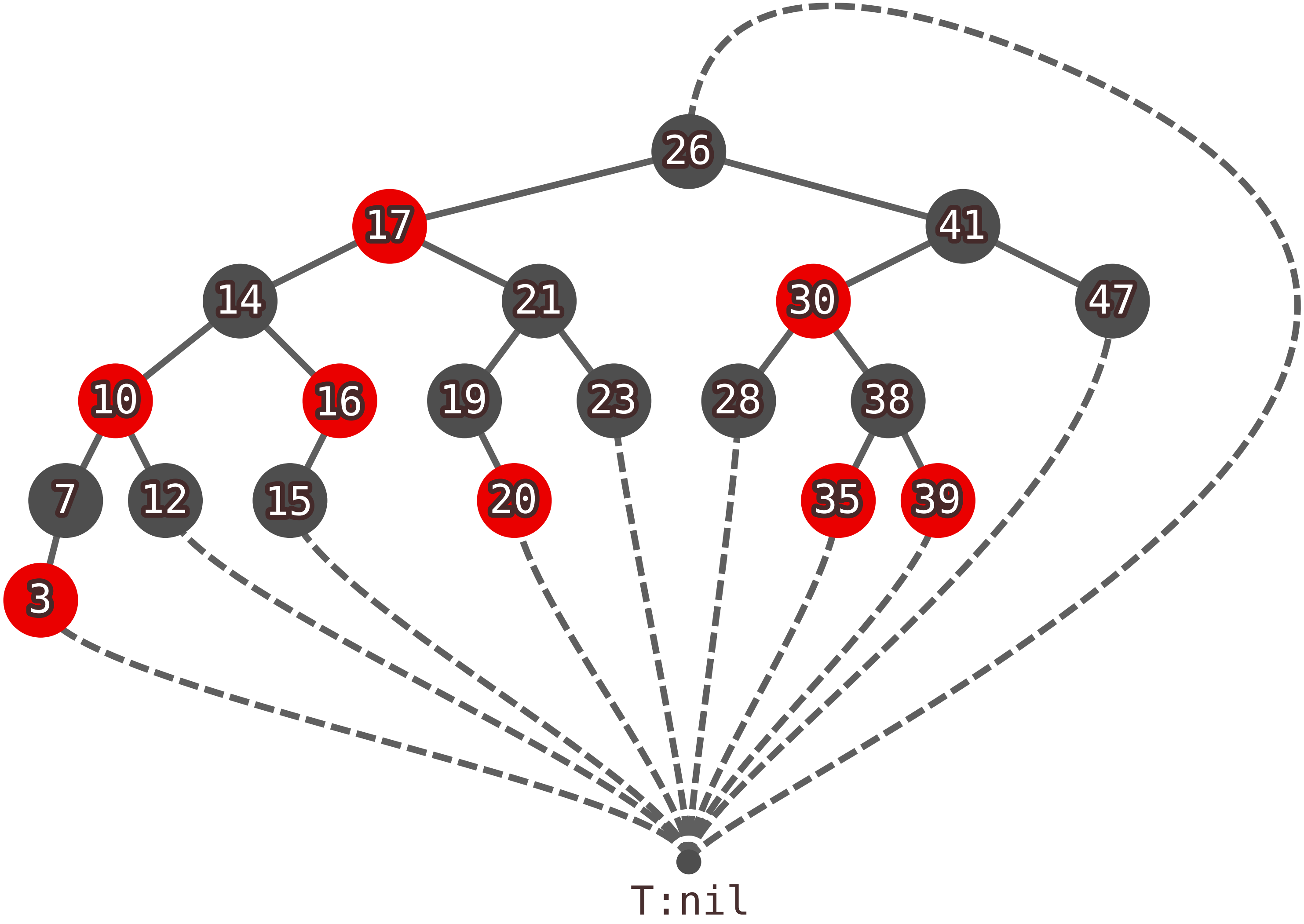
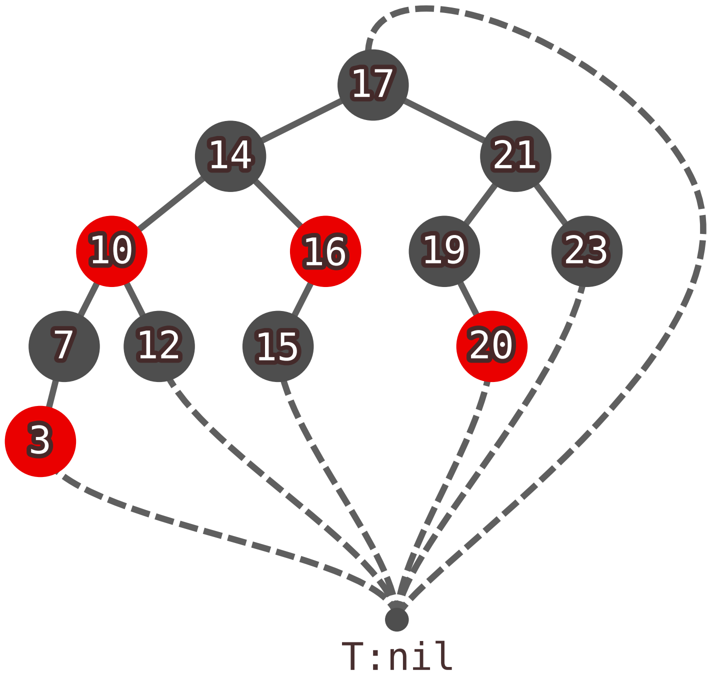
Red-black trees




26
17
41
14
21
30
47
10
16
19
23
28
38
7
12
15
20
35
39
3


nil
26
17
41
14
21
30
47
10
16
19
23
28
38
nil
7
12
15
20
35
39
nil
nil
3
nil
nil
nil
nil
nil
nil


26
17
41
14
21
30
47
10
16
19
23
28
38
7
12
15
20
35
39
3
T:nil



1. Every node is either red or black.
2. The root is black.
3. Every leaf (*T:nil*) is black.
4. If a node is red, then both its children are black. (Hence no two reds in a row on a simple path from the root to a leaf.)
5. For each node, all paths from the node to descendant leaves contain the same number of black nodes.

Height of a red-black tree

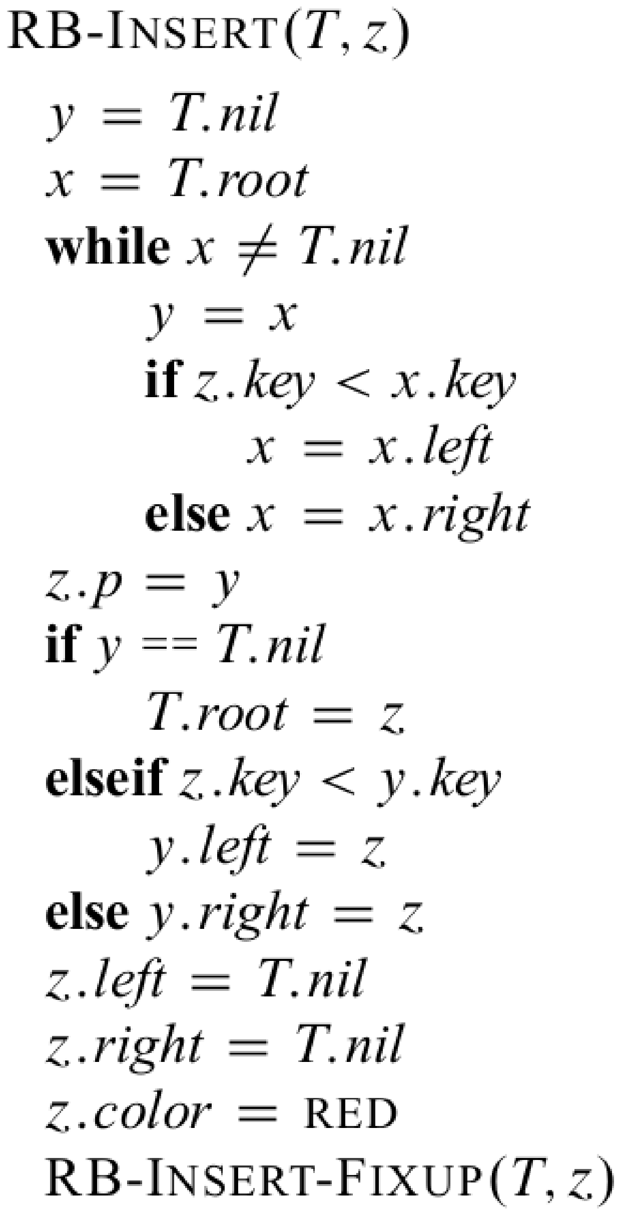
Height of a node, ***h(x)***,
is the number of edges in a longest path to a leaf.

Black-height of a node, ***bh(x)***,
is the number of black nodes (including *T:nil*) on the path from *x* to leaf, not counting *x*.

Claim 1: Any node with height h has black-height greater or equal to $h/2$.

Claim 2: The subtree rooted at any node x contains at least $2^{bh(x)} - 1$ internal nodes.

Lemma: The height of a red-black tree with n internal nodes is less or equal to $2\ log(n) + 1$.



```
RB-INSERT$(T, z)$
    $y = T.nil$
    $x = T.root$
    while $x \neq T.nil$
            $y = x$
            if $z.key < x.key$
                    $x = x.left$
            else $x = x.right$
    $z.p = y$
    if $y == T.nil$
            $T.root = z$
    elseif $z.key < y.key$
            $y.left = z$
    else $y.right = z$
    $z.left = T.nil$
    $z.right = T.nil$
    $z.color = $ RED
    RB-INSERT-FIXUP$(T, z)$
```

```
RB-INSERT-FIXUP$(T, z)$
  while $z.p.color$ == RED
      if $z.p == z.p.p.left$
          $y = z.p.p.right$
          if $y.color$ == RED
              $z.p.color$ = BLACK                          // case 1
              $y.color$ = BLACK                            // case 1
              $z.p.p.color$ = RED                          // case 1
              $z = z.p.p$                                  // case 1
          else if $z == z.p.right$
                  $z = z.p$                                // case 2
                  LEFT-ROTATE$(T, z)$                      // case 2
              $z.p.color$ = BLACK                          // case 3
              $z.p.p.color$ = RED                          // case 3
              RIGHT-ROTATE$(T, z.p.p)$                     // case 3
      else (same as then clause with “right” and “left” exchanged)
  $T.root.color$ = BLACK
```

```
RB-DELETE$(T, z)$
    $y = z$
    $y$-original-color $= y.color$
    if $z.left == T.nil$
        $x = z.right$
        RB-TRANSPLANT$(T, z, z.right)$
    elseif $z.right == T.nil$
        $x = z.left$
        RB-TRANSPLANT$(T, z, z.left)$
    else $y =$ TREE-MINIMUM$(z.right)$
        $y$-original-color $= y.color$
        $x = y.right$
        if $y.p == z$
            $x.p = y$
        else RB-TRANSPLANT$(T, y, y.right)$
            $y.right = z.right$
            $y.right.p = y$
        RB-TRANSPLANT$(T, z, y)$
        $y.left = z.left$
        $y.left.p = y$
        $y.color = z.color$
    if $y$-original-color == BLACK
        RB-DELETE-FIXUP$(T, x)$
```

```
RB-DELETE-FIXUP$(T, x)$
  while $x \neq T.root$ and $x.color == \text{BLACK}$
      if $x == x.p.left$
          $w = x.p.right$
          if $w.color == \text{RED}$
              $w.color = \text{BLACK}$                      // case 1
              $x.p.color = \text{RED}$                      // case 1
              LEFT-ROTATE$(T, x.p)$                      // case 1
              $w = x.p.right$                               // case 1
          if $w.left.color == \text{BLACK}$ and $w.right.color == \text{BLACK}$
              $w.color = \text{RED}$                        // case 2
              $x = x.p$                                     // case 2
          else if $w.right.color == \text{BLACK}$
                  $w.left.color = \text{BLACK}$             // case 3
                  $w.color = \text{RED}$                    // case 3
                  RIGHT-ROTATE$(T, w)$                   // case 3
                  $w = x.p.right$                           // case 3
              $w.color = x.p.color$                         // case 4
              $x.p.color = \text{BLACK}$                    // case 4
              $w.right.color = \text{BLACK}$                // case 4
              LEFT-ROTATE$(T, x.p)$                      // case 4
              $x = T.root$                                  // case 4
      else (same as then clause with “right” and “left” exchanged)
  $x.color = \text{BLACK}$
```