Red-Black Tree Pseudocode

Here are the procedures for red-black trees discussed in the book.

LEFT-ROTATE(T, x)

```
 1  y = x.right // set y
 2  x.right = y.left // turn y's left subtree into x's right subtree
 3  if y.left ≠ T.nil
 4        y.left.p = x
 5  y.p = x.p // link x's parent to y
 6  if x.p == T.nil
 7        T.root = y
 8  elseif x == x.p.left
 9        x.p.left = y
10  else
11        x.p.right = y
12  y.left = x // put x on y's left
13  x.p = y
```

RIGHT-ROTATE(T, x)

```
 1  x = y.left // set x
 2  y.left = x.right // turn x's left subtree β into y's right subtree
 3  if x.right ≠ T.nil // if β is a populated subtree, y points to x
 4        x.right.p = y
 5  x.p = y.p // link y's parent to x
 6  if y.p == T.nil // make x new root if y was the root
 7        T.root = x
 8  elseif y == y.p.right
 9        y.p.right = x
10  else
11        y.p.left = x
12  x.right = y // put y on x's right
13  y.p = x
```

TREE-MINIMUM(x)

```
1  while x.left ≠ NIL
2        x = x.left
3  return x
```

```
RB-INSERT(T, z)
 1  y = T.nil
 2  x = T.root
 3  while x ≠ T.nil // perform regular binary search
 4        y = x
 5        if z.key < x.key
 6              x = x.left
 7        else
 8              x = x.right
 9  z.p = y // make y the parent, insert z in the appropriate place
10  if y == T.nil
11        T.root = z
12  elseif z.key < y.key
13        y.left = z
14  else
15        y.right = z
16  z.left = T.nil
17  z.right = T.nil
18  z.color = RED
19  RB-INSERT-FIXUP(T, z)
```

```
RB-INSERT-FIXUP(T, z)
 1  while z.p.color == RED
 2      if z.p == z.p.p.left
 3          y = z.p.p.right
 4          if y.color = RED // case 1
 5              z.p.color = BLACK
 6              y.color = BLACK
 7              z.p.p.color = RED
 8              z = z.p.p
 9          else
10              if z == z.p.right // case 2
11                  z = z.p
12                  LEFT-ROTATE(T, z)
13              z.p.color = BLACK // case 2 leads into case 3
14              z.p.p.color = RED
15              RIGHT-ROTATE(T, z.p.p)
16      else
17          y = z.p.p.left
18          if y.color = RED // case 1
19              z.p.color = BLACK
20              y.color = BLACK
21              z.p.p.color = RED
22              z = z.p.p
23          else
24              if z == z.p.left // case 2
25                  z = z.p
26                  RIGHT-ROTATE(T, z)
27              z.p.color = BLACK // case 2 leads into case 3
28              z.p.p.color = RED
29              LEFT-ROTATE(T, z.p.p)
30  T.root.color == BLACK
```

```
RB-TRANSPLANT(T, u, v)
1  if u.p == T.nil
2      T.root = v
3  elseif u == u.p.left
4      u.p.left = v
5  else
6      u.p.right = v
7  v.p = u.p
```

```
RB-DELETE(T, u, v)
 1  y = z
 2  y-original-color = y.color
 3  if z.left == T.nil
 4       x = z.right
 5       RB-TRANSPLANT(T, z, z.right) // replace z with right non-nil subtree
 6  elseif z.right == T.nil
 7       x = z.left
 8       RB-TRANSPLANT(T, z, z.left) // replace z with left non-nil subtree
 9  else
10       y = TREE-MINIMUM(z.right)
11       y-original-color = y.color
12       x = y.right
13       if y.p == z
14            x.p = y
15       else RB-TRANSPLANT(T, y, y.right)
16            y.right = z.right
17            y.right.p = y
18       RB-TRANSPLANT(T, z, y)
19       y.left = z.left
20       y.left.p = y
21       y.color = z.color
22  if y-original-color == BLACK
23       RB-DELETE-FIXUP(T, x)
```

```
RB-DELETE-FIXUP(T, u, v)
 1  while x ≠ T.root and x.color == BLACK
 2      if x == x.p.left
 3          w = x.p.right
 4          if w.color == RED // case 1 in then clause
 5              w.color = BLACK
 6              x.p.color = red
 7              LEFT-ROTATE(T, x.p)
 8              w = x.p.right
 9          if w.left.color == BLACK and w.right.color == BLACK // case 2
10              w.color = red
11              x = x.p
12          else
13              if w.right.color == BLACK // case 3
14                  w.left.color = BLACK
15                  w.color = red
16                  RIGHT-ROTATE(T, w)
17                  w = x.p.right
18              w.color = x.p.color // case 4
19              x.p.color = BLACK
20              w.right.color = BLACK
21              LEFT-ROTATE(T, x.p)
22              x = T.root
23      else
24          w = x.p.left
25          if w.color == RED // case 1 in then clause
26              w.color = BLACK
27              x.p.color = red
28              RIGHT-ROTATE(T, x.p)
29              w = x.p.left
30          if w.right.color == BLACK and w.left.color == BLACK // case 2
31              w.color = red
32              x = x.p
33          else
34              if w.left.color == BLACK // case 3
35                  w.right.color = BLACK
36                  w.color = red
37                  LEFT-ROTATE(T, w)
38                  w = x.p.left
39              w.color = x.p.color // case 4
40              x.p.color = BLACK
41              w.left.color = BLACK
42              RIGHT-ROTATE(T, x.p)
43              x = T.root
44  x.color = BLACK
```