

**Algorithm 1** INSERT,DELETE,SEARCH INTERVAL TREES

---

```

1: procedure LEFT-ROTATE( $T, x$ )
2:    $y \leftarrow \text{right}[x]$ 
3:    $\text{right}[x] \leftarrow \text{left}[y]$ 
4:   if  $\text{left}[y] \neq \text{NIL}$  then
5:      $p[\text{left}[y]] \leftarrow x$  ▷  $p$  is parent
6:    $p[y] \leftarrow p[x]$ 
7:   if  $p[x] \leftarrow \text{NIL}$  then
8:      $\text{root}[T] \leftarrow y$ 
9:   else if  $x = \text{left}[p[x]]$  then
10:     $\text{left}[p[x]] \leftarrow y$ 
11:   else
12:     $\text{right}[p[x]] \leftarrow y$ 
13:    $\text{left}[y] \leftarrow x$ 
14:    $p[x] \leftarrow y$ 
15:    $\text{max}[x] = \text{MAX}(\text{max}[\text{left}[x]], \text{max}[\text{right}[x]], \text{max}[x])$  ▷ reconfiguring the augmented values
16:    $\text{max}[y] = \text{MAX}(\text{max}[\text{left}[y]], \text{max}[\text{right}[y]], \text{max}[y])$ 
17:    $\text{max}[p[y]] = \text{MAX}(\text{max}[\text{left}[p[y]]], \text{max}[\text{right}[p[y]]], \text{max}[p[y]])$ 
18:
19: procedure RIGHT-ROTATE( $T, y$ ) ▷ analogous to LEFT-ROTATE
20:    $x \leftarrow \text{left}[y]$ 
21:    $\text{left}[y] \leftarrow \text{right}[x]$ 
22:   if  $\text{right}[x] \neq \text{NIL}$  then
23:      $p[\text{right}[x]] \leftarrow y$ 
24:    $p[x] \leftarrow p[y]$ 
25:   if  $p[y] \leftarrow \text{NIL}$  then
26:      $\text{root}[T] \leftarrow x$ 
27:   else if  $y = \text{right}[p[y]]$  then
28:      $\text{right}[p[y]] \leftarrow x$ 
29:   else
30:      $\text{left}[p[y]] \leftarrow x$ 
31:    $\text{right}[x] \leftarrow y$ 
32:    $p[y] \leftarrow x$ 
33:    $\text{max}[x] = \text{MAX}(\text{max}[\text{left}[x]], \text{max}[\text{right}[x]], \text{max}[x])$ 
34:    $\text{max}[y] = \text{MAX}(\text{max}[\text{left}[y]], \text{max}[\text{right}[y]], \text{max}[y])$ 
35:    $\text{max}[p[x]] = \text{MAX}(\text{max}[\text{left}[p[x]]], \text{max}[\text{right}[p[x]]], \text{max}[p[x]])$ 
36:
37: procedure BST-INSERT( $T, z$ )
38:    $y \leftarrow \text{NIL}$ 
39:    $x \leftarrow \text{root}[T]$ 
40:   while  $x \neq \text{NIL}$  do
41:      $y \leftarrow x$ 
42:     if  $\text{low}[z] < \text{low}[x]$  then
43:        $x \leftarrow \text{left}[x]$ 
44:     else
45:        $x \leftarrow \text{right}[x]$ 
46:    $p[z] \leftarrow y$ 
47:   if  $y = \text{NIL}$  then
48:      $\text{root}[T] \leftarrow z$  ▷  $z$  is the only node
49:   else if  $\text{low}[z] < \text{low}[y]$  then ▷ setting up the pointers to  $z$ 
50:      $\text{left}[y] \leftarrow z$ 
51:   else
52:      $\text{right}[y] \leftarrow z$ 
53:    $\text{max}[z] = \text{high}[z]$ 

```

---

---

```

55: procedure RB-INSERT( $T, x$ )                                ▷ inserting a node in interval(Red Black) tree
56:    $BST - INSERT(T, x)$                                      ▷ First insert  $x$  as normally inserted into BST and color it red
57:    $color[x] \leftarrow RED$ 
58:   while  $x \neq root[T]$  and  $color[p[x]] = RED$  do
59:     if  $p[x] = left[p[p[x]]]$  then
60:        $y \leftarrow right[p[p[x]]]$                                 ▷ uncle
61:       if  $color[y] = RED$  then                                    ▷ Case a
62:          $color[p[x]] \leftarrow BLACK$ 
63:          $color[y] \leftarrow BLACK$ 
64:          $color[p[p[x]]] \leftarrow RED$ 
65:          $x \leftarrow p[p[x]]$                                 ▷ Change  $x$  to grandparent
66:       else if  $x = right[p[x]]$  then                            ▷ Case b: Left Right Case
67:          $x \leftarrow p[x]$ 
68:          $LEFT - ROTATE(T, x)$ 
69:          $color[p[x]] \leftarrow BLACK$                                 ▷ Follow Case b: Left Left Case
70:          $color[p[p[x]]] \leftarrow RED$ 
71:          $RIGHT - ROTATE(T, p[p[x]])$ 
72:       else                                                    ▷ Case b: Left Left Case
73:          $color[p[x]] \leftarrow BLACK$ 
74:          $color[p[p[x]]] \leftarrow RED$ 
75:          $RIGHT - ROTATE(T, p[p[x]])$ 
76:     else
77:       (do the same thing in then in line 94 clause with "right" and "left" swapped)  ▷ Case b: Right
Left and Right Right Case
78:    $color[root[T]] \leftarrow BLACK$                                 ▷ Since root is always black
79:
80: procedure RB-DELETE( $T, z$ )                                ▷ Deleting a node in RB-Tree
81:   if  $left[z] = nil[T]$  or  $right[z] = nil[T]$  then
82:      $y \leftarrow z$ 
83:   else
84:      $y \leftarrow RB - SUCCESSOR(z)$ 
85:   if  $left[y] \neq nil[T]$  then
86:      $x \leftarrow left[y]$ 
87:   else
88:      $x \leftarrow right[y]$ 
89:    $p[x] \leftarrow p[y]$ 
90:   if  $p[y] = nil[T]$  then
91:      $root[T] \leftarrow x$ 
92:   else if  $y = left[p[y]]$  then
93:      $left[p[y]] \leftarrow x$ 
94:   else
95:      $right[p[y]] \leftarrow x$ 
96:   if  $y \neq z$  then
97:      $low[z] \leftarrow low[y]$ 
98:   if  $color[y] = BLACK$  then
99:      $RB - DELETE - FIXUP(T, x)$ 
    return  $y$ 
100: procedure RB-SUCCESSOR( $x$ )                                ▷ helper for finding successor for a node in tree
101:   if  $right[x] \neq NIL$  then return  $RB - MINIMUM(right[x])$ 
102:    $y \leftarrow p[x]$ 
103:   while  $y \neq NIL$  and  $x = right[y]$  do
104:      $x \leftarrow y$ 
105:      $y \leftarrow p[y]$ 
106:   return  $y$ 

```

---

```

111: procedure RB-DELETE-FIXUP( $T, x$ )
112:   while  $x \neq \text{root}[T]$  and  $\text{color}[x] = \text{BLACK}$  do
113:     if  $x = \text{left}[p[x]]$  then
114:        $w \leftarrow \text{right}[p[x]]$ 
115:       if  $\text{color}[x] = \text{RED}$  then
116:          $\text{color}[w] \leftarrow \text{BLACK}$ 
117:          $\text{color}[p[x]] \leftarrow \text{RED}$ 
118:          $\text{LEFT} - \text{ROTATE}(T, p[x])$ 
119:          $w \leftarrow \text{right}[p[x]]$ 
120:       if  $\text{color}[\text{left}[w]] = \text{BLACK}$  and  $\text{color}[\text{right}[w]] = \text{BLACK}$  then
121:          $\text{color}[w] \leftarrow \text{RED}$ 
122:          $x \leftarrow p[x]$ 
123:       else if  $\text{color}[\text{right}[w]] = \text{BLACK}$  then
124:          $\text{color}[\text{left}[w]] \leftarrow \text{BLACK}$ 
125:          $\text{color}[w] \leftarrow \text{RED}$ 
126:          $\text{RIGHT} - \text{ROTATE}(T, w)$ 
127:          $w \leftarrow \text{right}[p[x]]$ 
128:       else
129:          $\text{color}[w] \leftarrow \text{color}[p[x]]$ 
130:          $\text{color}[p[x]] \leftarrow \text{BLACK}$ 
131:          $\text{color}[\text{right}[w]] \leftarrow \text{BLACK}$ 
132:          $\text{LEFT} - \text{ROTATE}(T, p[x])$ 
133:          $x \leftarrow \text{root}(T)$ 
134:     else
135:       (same as then clause with "right" and "left" exchanged)
136:    $\text{color}[x] \leftarrow \text{BLACK}$ 
137:

```

---



---

```

138: procedure SEARCH( $\text{root}, \text{interval}$ )                                 $\triangleright$  interval to be searched has attributes low and high
139:   if  $\text{root} = \text{NULL}$  then return  $\text{NULL}$ 
140:   if  $\text{Interval}[\text{root}].\text{low} \leq \text{interval}.\text{high}$  and  $\text{interval}.\text{low} \leq \text{Interval}[\text{root}].\text{high}$  then           $\triangleright$  Checking for
   overlaps return  $\text{Interval}[\text{root}]$ 
141:   if  $\text{left}[\text{root}] \neq \text{NULL}$  and  $\max[\text{left}[\text{root}]] \geq \text{interval}.\text{low}$  then return  $\text{SEARCH}(\text{left}[\text{root}], \text{interval})$   $\triangleright$ 
   interval may overlap with an interval in left subtree
   return  $\text{SEARCH}(\text{right}[\text{root}], \text{interval})$                                  $\triangleright$  Otherwise recur for right subtree

```

---