# Control Flow Graph

Key:

Normal node: Starting node:

Finishing node: Defines X:

Reads X:

References another graph:

X\*

X’

#

* getRoot()

Edge Coverage:

[1] 

Root’.left

createInfinityNode ()

* RBTreeWithFaults()

Edge Coverage:

[1]

Root\*

* RBTreeWithFaults(RBNode rot)

createInfinityNode (rbnode)

Edge Coverage:

[1]

Rot’

Root\*

createNullNode (rbnode)

* createInfinityNode()

Edge Coverage:

[1,2]

createInfinityNode (rbnode)

* createInfinityNode(RBNode leftchild)

Edge Coverage:

[1]

Leftchild’ node.LeftParent\*

Node\* node.black\*

* createNullNode(RBNode parent)

Edge Coverage:

[1]

parent’

Newnode\*

Newnode.Black\*

Newnode’

* search(**int** k)

Edge Coverage:

[1,2,3]

[1,2,4]

[1,2,5]

Currnode’

Currnode.key’

K’

SearchNode

This.Root’

K’

Currnode\*

Currnode.Value’

* insert(int k, string v)

newRBNode

SearchNode

y.key’

z.key’

Counter\*

y’

v’ y’

k’ z\*

This.Root’

K’

y\*

Edge Coverage:

[1,2,3,10]

[1,2,4,5]

[1,2,4,6,7,9,10]

[1,2,4,6,8,9,10]

z.key’

y.key’

This.Root.Left\*

Z.black\*

Root’

z.parent\*

y.left\*

z’

y.right\*

z’

Counter’

fixUpTree(RBNode))

z’

Counter\*

* leftChild(RBNode x,RBNode y)

x.left\* y’

y.Parent\* x’

Edge Coverage:

[1]

Leftchild (RBNode,RBNode)

* transplate(RBNode x, RBNode y)

x.Parent’

y’

x.Parent’

y’

x'

x.Parent.left’

Rightchild (RBNode,RBNode)

Edge Coverage:

[1,2]

[1,3]

x.Black’

x.Black\*

counter\*

counter’

w\* x.Parent.Right’

w.Black’

w.Black\* x.Parent.Black\*

leftRotate (RBNode)

x.Parent’

x.Parent.Right’

w\*

Counter\*

w.Left.Black’ w.Right.Black’

x.Parent’

X\*

w.Black\*

Counter\*

x\* x.Parent.Left’

w\*

x.Left’

w.Black’

w.Black\* x.Parent.Black\*

rightRotate (RBNode)

x.Parent’

w.Left.Black\*

w.Black\*

W\* x.Parent.Right’

Counter\*

rightRotate (RBNode)

x.Parent.Black’

W.Black\*

x.Parent.Black\*

LeftRotate (RBNode)

x.parent’

X\*

This.Root.Left’

Counter\*

W\*

x.Parent.Left’

counter\*

w.Left.Black’ w.Right.Black’

w.Black\* x.Parent’

x\*

counter\*

w.Black\* w.Right.Black\*

LeftRotate (RBNode)

w.Left.Black’

w \* x.Parent.Left’

counter\*

w.Black\*

x.Parent.Black’

x.Parent.Black\*

w.Left.Black\*

rightRotate (RBNode)

X\*

This.Root.Left’

Counter\*

* deleteFixup(RBNode x)

Counter\*

x’

x.Black’

This.root.Left’

w.Right.Black’

W’

x.parent’

W’

Edge Coverage:

[1,2,…][…3,+][+4,5,6,7,-][-8,14,2,…] […26,27,28]

[+4,7-][-9,10,11,13,14,2,…]

[-12,13,14,2…]

[+15,16,17,18,\*]

[+15,18,\*]

[\*19,25,2…]

[\*20,21,22,23,24,25,2…]

[\*20,23,24,25,2…]

* minimumNode(RBNode node)

minimumNode (RBNode)

Node.Left’

node'

node.left’

Edge Coverage:

[1,2]

[1,4]

[1,3]

Node’

* isNullNode(RBNode node)

Node.Key’

Edge Coverage:

[1]

* maxValue(RBNode node)

Node.value’

Node.Right’

node.right’

Edge Coverage:

[1,2] [1,3]

* ArrayOfStringsToArrayOfInts(String[] strArr)

Arr’

I’

strArr’

I\*

Arr\*

I\*

strArr’

I’

Arr\*

Edge Coverage:

[1,2,3,2,4]

* valuesToArray()

ElementsToString (RBNode,Boolean)

Edge Coverage:

[1,2]

valueString’

This.root.left’

valueString\*

* Size()

Edge Coverage:

[1,2] [1,3]

This’

ElementsToString (RBNode,Boolean)

This.root.left’

* RBNode(String value,**int** key, RBNode left, RBNode right,RBNode parent)

Edge Coverage:

[1]

This.value\*

This.key\*

This.parent\*

This.root\*

This.left\*

This.right\*

This.black\*

RightRotate(RBnode y)

x\*

y’

y\*

x.Right’

Y’

X’

RightChild (RBNode,RBNode)

leftChild (RBNode,RBNode)

Transplate (RBNode,RBNode)

x\*

y.Left’