*Red Black Tree – Insertion*

*Description:*

In this Chapter, we would introduce the algorithm about how to insert a new node into Red Black Tree. *However, in order to finish the procedure, we need to keep the original structure of Red Black Tree and insert the node with the color Red.*

Here, two functions are used to realize the insertion stage, include *RB-Insert(T, z)* and *RB-Insert-FixUp(T, z)*.

*Procedure:*

During the procedure of insertion, there includes Two Steps:

1. Insert the node into the right position with the color Red.
2. Adjust the color of node from bottom to top and left – rotate | right – rotate the nodes to keep the public five properties of Red Black Tree.

*Structure:*

In Red Black Tree below, there exists one Red Black Tree with unique key.

*struct Node {*

*int key; Key stands for the value of the node.*

*string color; Two kinds of color, Red and Black.*

*Node \* p; parent child.*

*Node \* left; left child.*

*Node \* right; right child.*

*};*

*Example:*

A picture containing indoor, sitting, looking, clock

Description automatically generated

Here, we need to insert the node with key 4. We are using the normal Binary Search Method in Red Black Tree to get the right position to insert the node. However, we need to keep in mind that the node has the color of Red. After insertion, we can find that there exist two nodes with color Red in Parent – Child Relationship.

A picture containing indoor, sitting, looking, clock

Description automatically generated

In next step, we need to adjust the sequence of nodes and make it obey the public five properties in Red Black Tree.

*Procedures:*

* Check that the color of node 4 and 5 have the same color as Red, and also the right node of its parent equals to black. Change the color of node 5 and 8 as Black and convert the color of node 7 as white.

A picture containing indoor, sitting, photo, looking

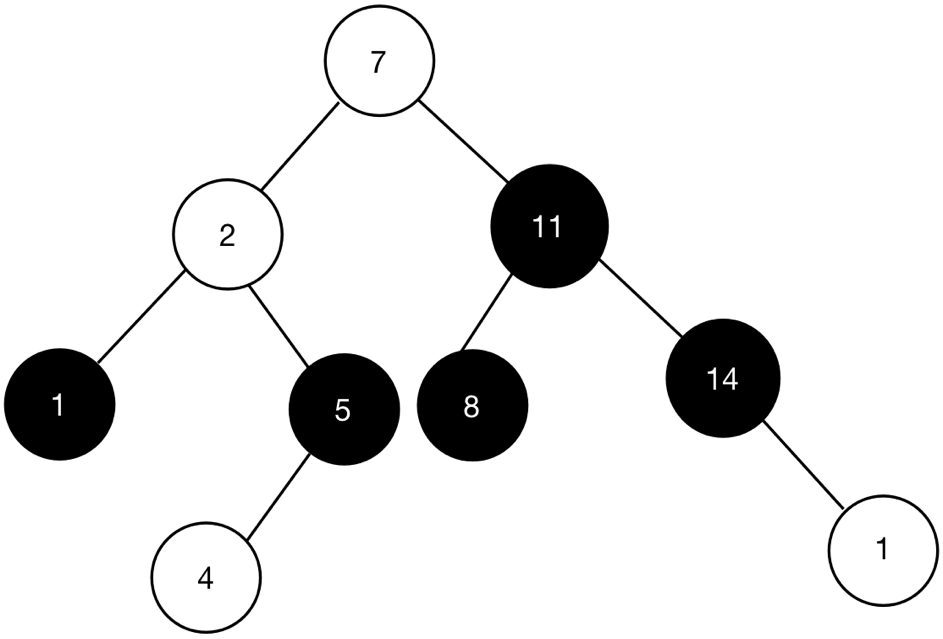
Description automatically generated

* Go to the upper level of Red Black Tree, and check that the node 7 and node 2 have the same color white. Therefore, rotate node 2 and node 7 and make the structure below.

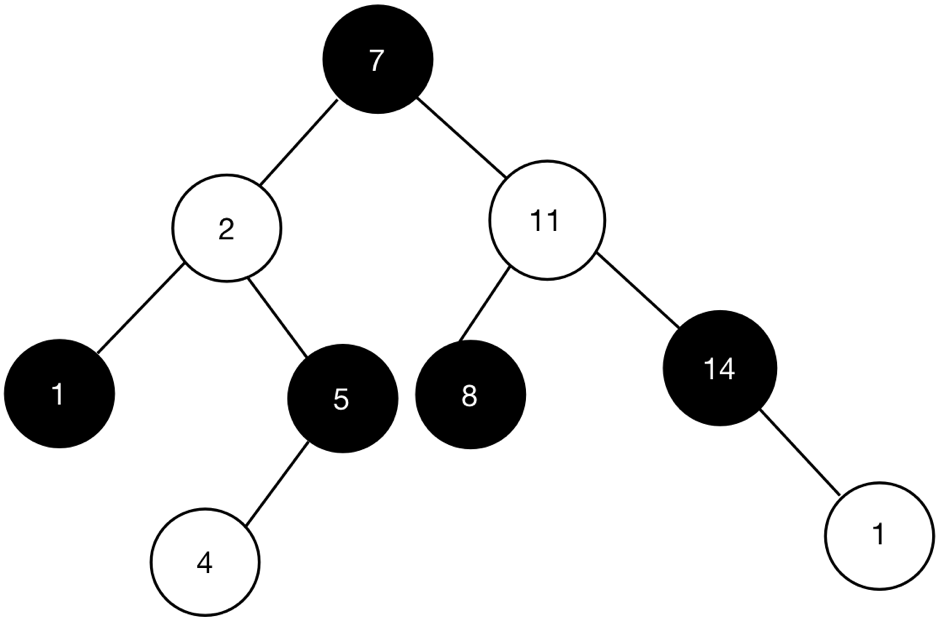
A close up of electronics

Description automatically generated

* Now, rotate the parent node 7 of node 2, parent node 11 and the uncle node 14 of node 2.



* In next step, we intend to change the color of nodes, including the node 7, and node 11. The color of node 7 change to Black, and the color of node 11 changes to white.

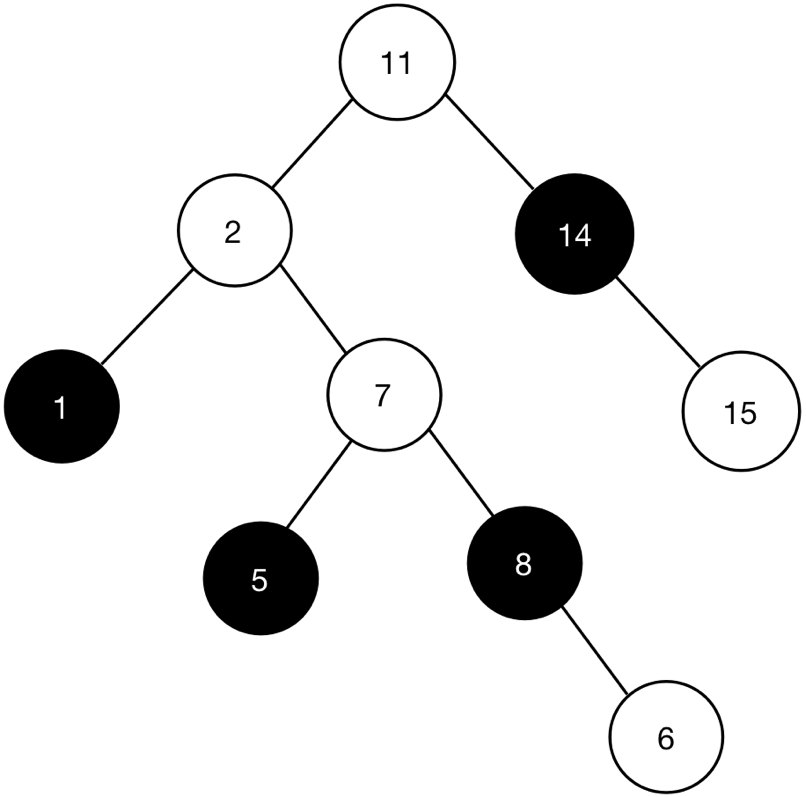


*Example:*

A picture containing indoor, sitting, looking, glasses

Description automatically generated

At first, we need to check the color of parent node 8 and the color of uncle node 5. In the first scenario, we find that the node 5 and 8 are all RED, therefore, we need to convert the color to BLACK. Also, we need to convert the color of node 7 to RED.



In this scenario, we need to rotate the node 2, since in such way, we can find that the total number of black node are all equal, which equals to 1.

A picture containing sitting, clock

Description automatically generated

In this scenario, we need to right rotate the node 11 to make the Red Black Tree much more balanced.

A picture containing computer

Description automatically generated

After finishing the rotation, then we need to change the color of root node 7 as BLACK.

A picture containing electronics, sitting, computer

Description automatically generated

*Code:*

void RB\_Insert (Tree T, Node z) {

Node x = T.nil; // Node x is the parent node of node y.

Node y = T.root;

// Move both node x and y to the right location.

while (y != x) {

if (z.value > y.value) {

y = y.right;

} else if (z.value < y.value) {

y = y.left;

}

}

// Set up the parent node of z.

z.p = x;

if (y == x) {

T.root = z;

} else if (z.value < x.value) {

x.left = z;

} else {

x.right = z;

}

z.left = T.nil;

z.right = T.nil;

z.color = RED;

RB\_INSERT\_ROTATE (T, z);

}

*Instruction:*

In order not to break the rule of Red Black Tree, after insertion, we make the node z with color RED. With the color RED, we can persist the property of Red Black Tree.

*Code:*

void RB\_INSERT\_ROTATE (Tree T, Node z) {

while (z.p && z.p.color == RED) {

if (z.p == z.p.p.left) {

Node y = z.p.p.right;

if (y.color == RED) {

z.p.color = BLACK;

y.color = BLACK;

z.p.p.color = RED;

z = z.p.p;

} else if (y.color == BLACK) {

if (z == z.p.right) {

z = z.p;

Left\_Rotation(T, z);

} else {

z.p.color = BLACK;

z.p.p.color = RED;

Right\_Rotation(T, z.p.p);

}

}

} else if (z.p == z.p.p.right) {

Node y = z.p.p.left;

if (y.color == RED) {

z.p.color = BLACK;

y.color = BLACK;

z.p.p.color = RED;

z = z.p.p;

} else if (y.color == BLACK) {

if (z == z.p.left) {

z = z.p;

Right\_Rotation(T, z);

} else {

z.p.color = BLACK;

z.p.p.color = RED;

Left\_Rotation(T, z.p.p);

}

}

}

}

T.root.color = BLACK;

}