Red Black Trees

Mohsin Abbas

Red-Black Properties

- The *red-black is a BST such that*:
  1. Every node is either red or black
  2. Every NULL pointer is black
     - Note: this means every node has 2 children
  3. If a node is red, both children are black
     - Note: can’t have 2 consecutive reds on a path
  4. Every path from node to descendent leaf contains the same number of black nodes
  5. The root is always black

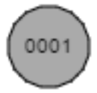
Red Black Trees

Let’s do an example from scratch

Always insert nodes as red (except the root)

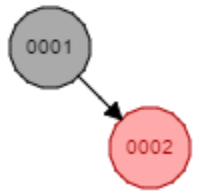
Insert 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11

- Always insert nodes as red (except the root)



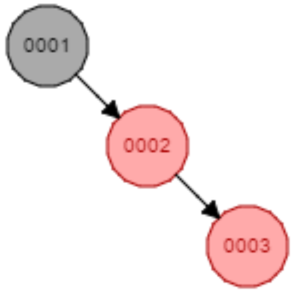
1. Every node is either red or black
2. Every NULL pointer is black
3. If a node is red, both children are black
4. Every path from node to descendent leaf contains the same number of black nodes
5. The root is always black

Insert 1, **2**, 3, 4, 5, 6, 7, 8, 9, 10, 11



1. Every node is either red or black
2. Every NULL pointer is black
3. If a node is red, both children are black
4. Every path from node to descendent leaf contains the same number of black nodes
5. The root is always black

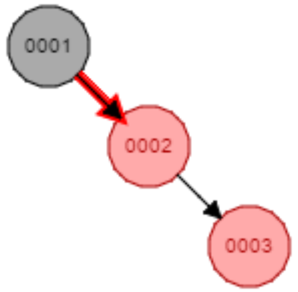
Insert 1, 2, **3**, 4, 5, 6, 7, 8, 9, 10, 11



1. Every node is either red or black
2. Every NULL pointer is black
3. If a node is red, both children are black
4. Every path from node to descendent leaf contains the same number of black nodes
5. The root is always black

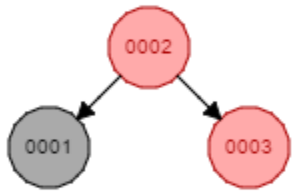
Insert 1, 2, **3**, 4, 5, 6, 7, 8, 9, 10, 11

Rotate



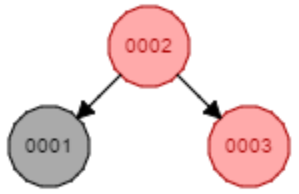
1. Every node is either red or black
2. Every NULL pointer is black
3. If a node is red, both children are black
4. Every path from node to descendent leaf contains the same number of black nodes
5. The root is always black

Insert 1, 2, **3**, 4, 5, 6, 7, 8, 9, 10, 11



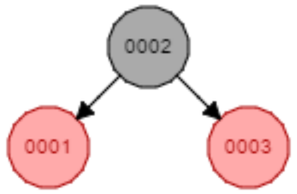
1. Every node is either red or black
2. Every NULL pointer is black
3. If a node is red, both children are black
4. Every path from node to descendent leaf contains the same number of black nodes
5. The root is always black

Insert 1, 2, **3**, 4, 5, 6, 7, 8, 9, 10, 11



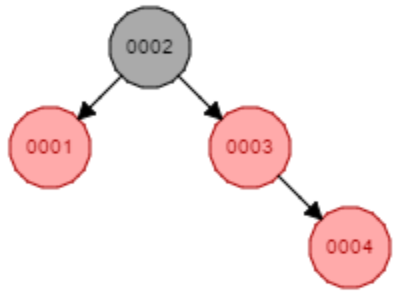
1. Every node is either red or black
2. Every NULL pointer is black
3. If a node is red, both children are black
4. Every path from node to descendent leaf contains the same number of black nodes
5. The root is always black

Insert 1, 2, **3**, 4, 5, 6, 7, 8, 9, 10, 11



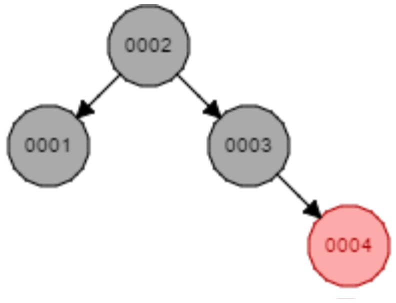
1. Every node is either red or black
2. Every NULL pointer is black
3. If a node is red, both children are black
4. Every path from node to descendent leaf contains the same number of black nodes
5. The root is always black

Insert 1, 2, 3, **4**, 5, 6, 7, 8, 9, 10, 11



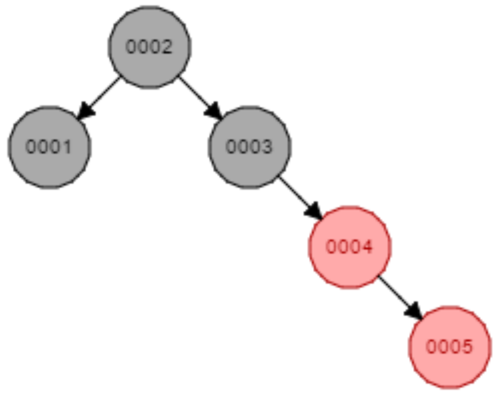
1. Every node is either red or black
2. Every NULL pointer is black
3. If a node is red, both children are black
4. Every path from node to descendent leaf contains the same number of black nodes
5. The root is always black

Insert 1, 2, 3, **4**, 5, 6, 7, 8, 9, 10, 11



1. Every node is either red or black
2. Every NULL pointer is black
3. If a node is red, both children are black
4. Every path from node to descendent leaf contains the same number of black nodes
5. The root is always black

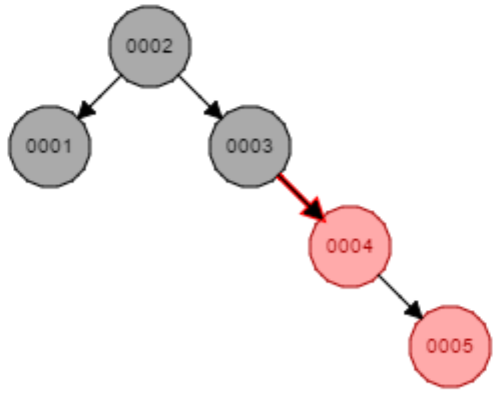
Insert 1, 2, 3, 4, **5**, 6, 7, 8, 9, 10, 11



1. Every node is either red or black
2. Every NULL pointer is black
3. If a node is red, both children are black
4. Every path from node to descendent leaf contains the same number of black nodes
5. The root is always black

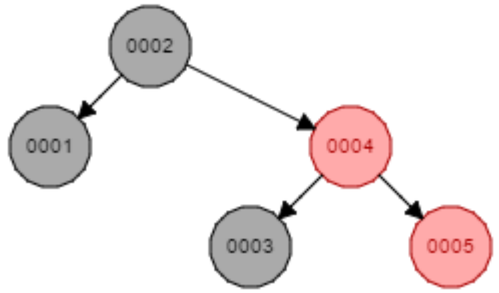
Insert 1, 2, 3, 4, **5**, 6, 7, 8, 9, 10, 11

Rotate



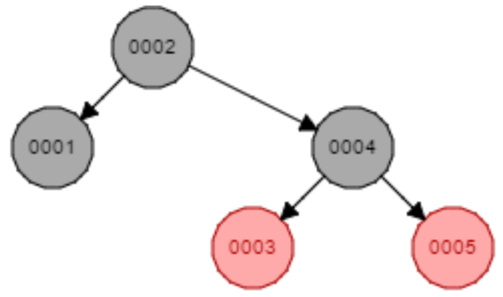
1. Every node is either red or black
2. Every NULL pointer is black
3. If a node is red, both children are black
4. Every path from node to descendent leaf contains the same number of black nodes
5. The root is always black

Insert 1, 2, 3, 4, **5**, 6, 7, 8, 9, 10, 11



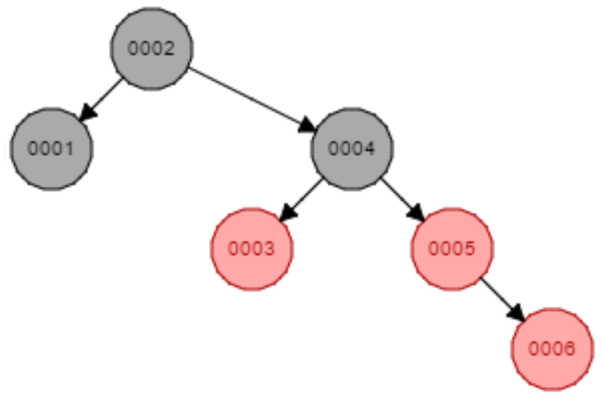
1. Every node is either red or black
2. Every NULL pointer is black
3. If a node is red, both children are black
4. Every path from node to descendent leaf contains the same number of black nodes
5. The root is always black

Insert 1, 2, 3, 4, **5**, 6, 7, 8, 9, 10, 11



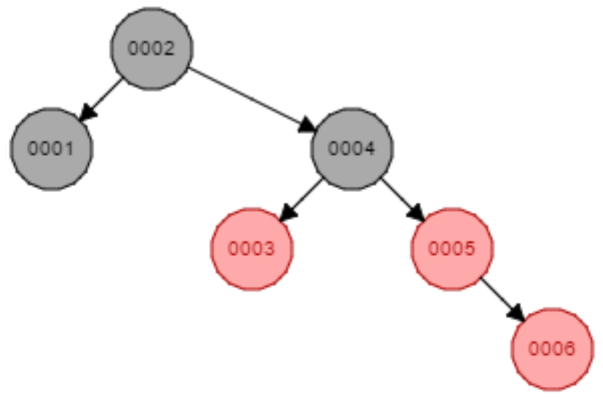
1. Every node is either red or black
2. Every NULL pointer is black
3. If a node is red, both children are black
4. Every path from node to descendent leaf contains the same number of black nodes
5. The root is always black

Insert 1, 2, 3, 4, 5, **6**, 7, 8, 9, 10, 11



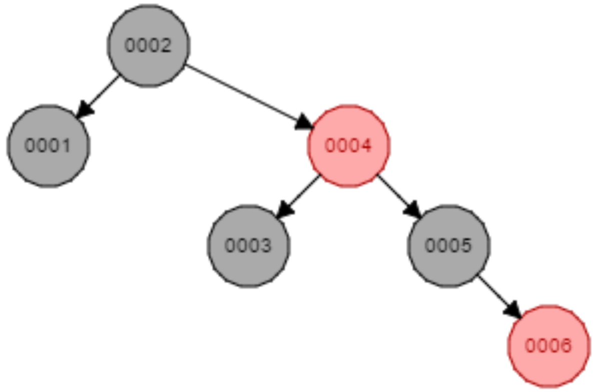
1. Every node is either red or black
2. Every NULL pointer is black
3. If a node is red, both children are black
4. Every path from node to descendent leaf contains the same number of black nodes
5. The root is always black

Insert 1, 2, 3, 4, 5, **6**, 7, 8, 9, 10, 11



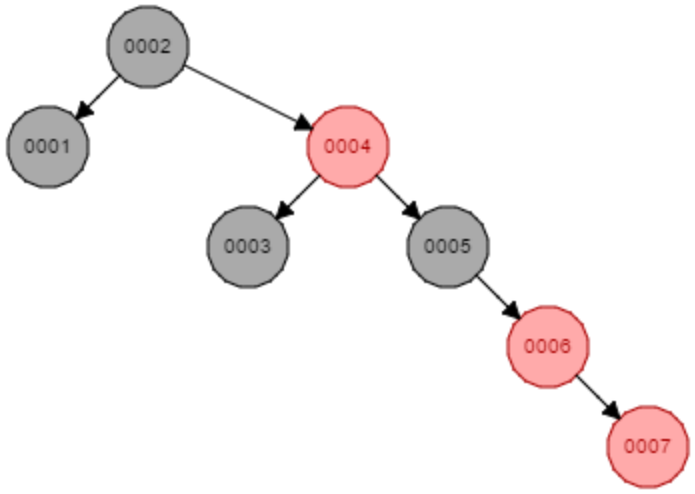
1. Every node is either red or black
2. Every NULL pointer is black
3. If a node is red, both children are black
4. Every path from node to descendent leaf contains the same number of black nodes
5. The root is always black

Insert 1, 2, 3, 4, 5, **6**, 7, 8, 9, 10, 11



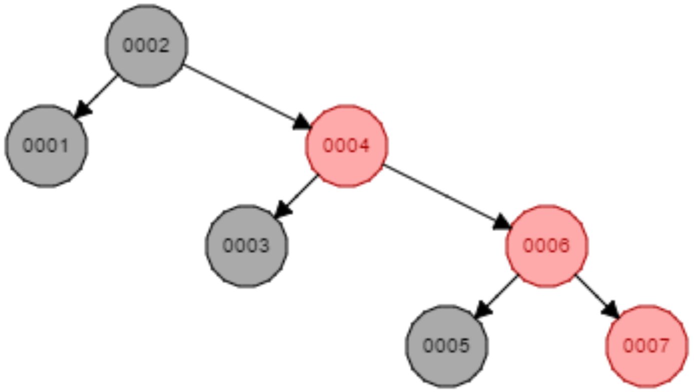
1. Every node is either red or black
2. Every NULL pointer is black
3. If a node is red, both children are black
4. Every path from node to descendent leaf contains the same number of black nodes
5. The root is always black

Insert 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11



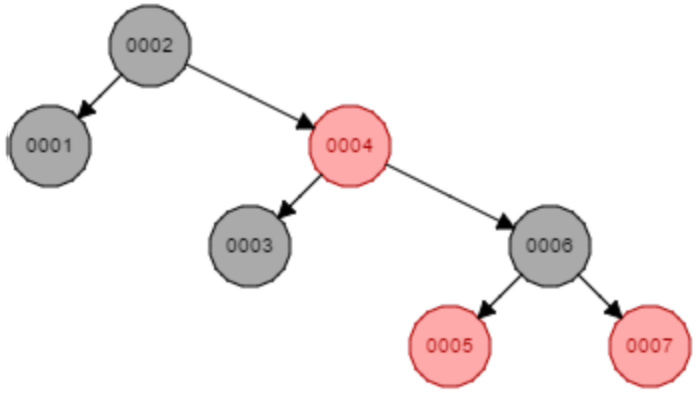
1. Every node is either red or black
2. Every NULL pointer is black
3. If a node is red, both children are black
4. Every path from node to descendent leaf contains the same number of black nodes
5. The root is always black

Insert 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11



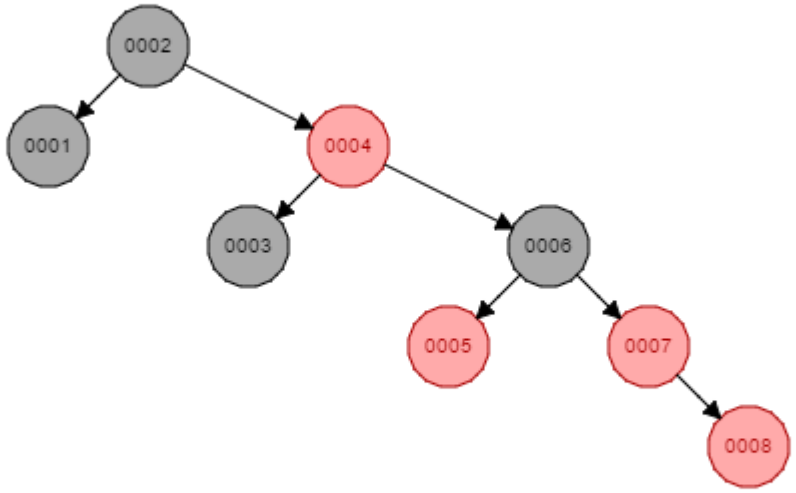
1. Every node is either red or black
2. Every NULL pointer is black
3. If a node is red, both children are black
4. Every path from node to descendent leaf contains the same number of black nodes
5. The root is always black

Insert 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11



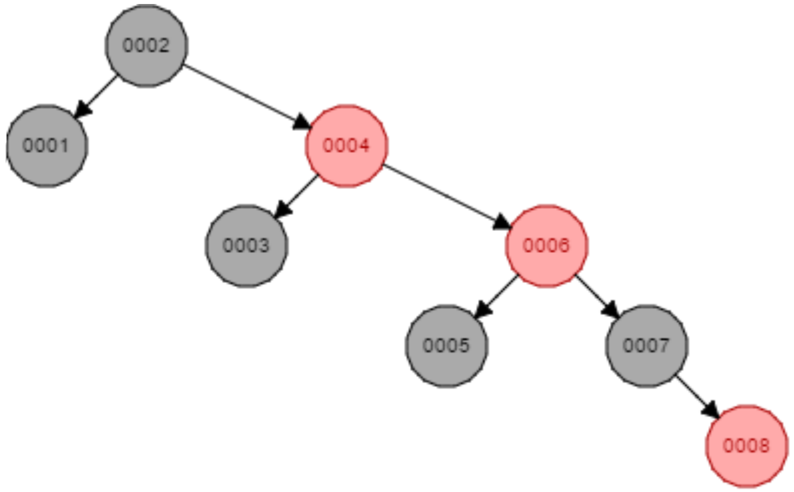
1. Every node is either red or black
2. Every NULL pointer is black
3. If a node is red, both children are black
4. Every path from node to descendent leaf contains the same number of black nodes
5. The root is always black

Insert 1, 2, 3, 4, 5, 6, 7, **8**, 9, 10, 11



1. Every node is either red or black
2. Every NULL pointer is black
3. If a node is red, both children are black
4. Every path from node to descendent leaf contains the same number of black nodes
5. The root is always black

Insert 1, 2, 3, 4, 5, 6, 7, **8**, 9, 10, 11

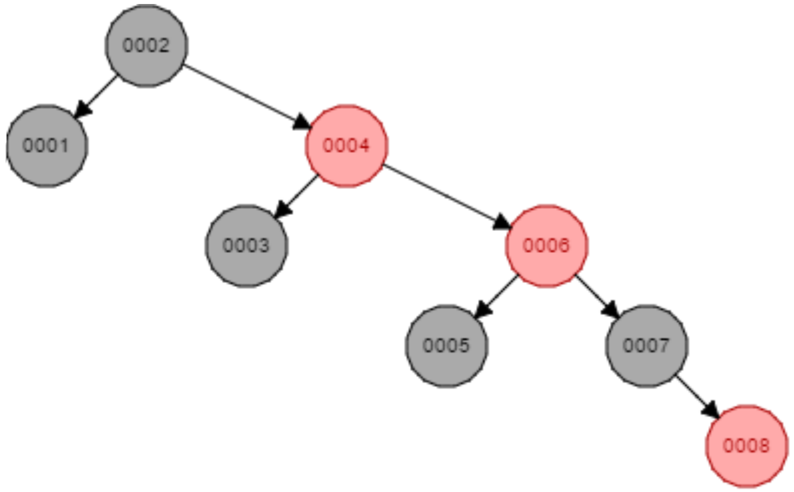


1. Every node is either red or black
2. Every NULL pointer is black
3. If a node is red, both children are black
4. Every path from node to descendent leaf contains the same number of black nodes
5. The root is always black

Insert 1, 2, 3, 4, 5, 6, 7, **8**, 9, 10, 11

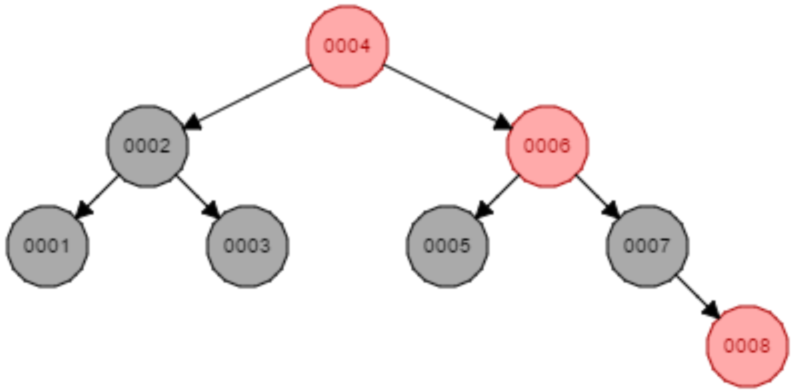
Caution:

See the next step very carefully



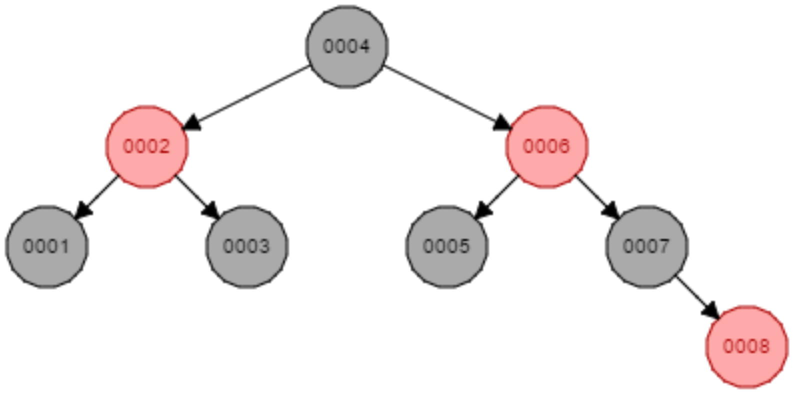
1. Every node is either red or black
2. Every NULL pointer is black
3. If a node is red, both children are black
4. Every path from node to descendent leaf contains the same number of black nodes
5. The root is always black

Insert 1, 2, 3, 4, 5, 6, 7, **8**, 9, 10, 11



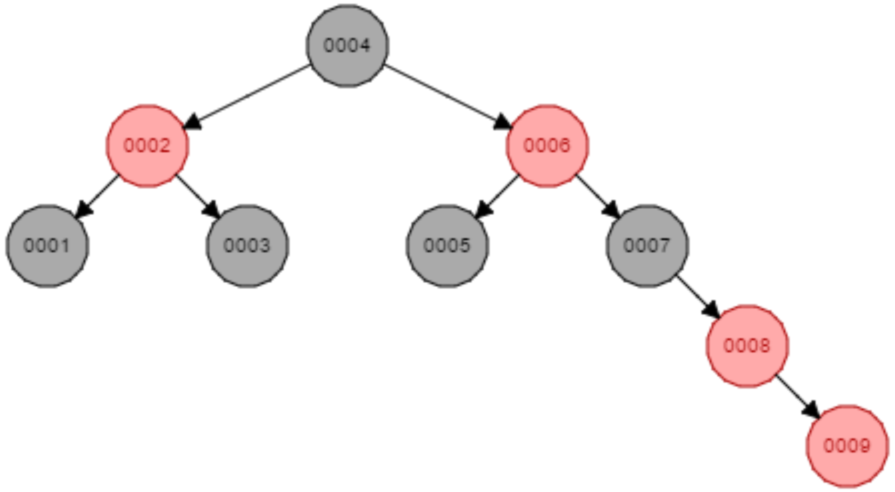
1. Every node is either red or black
2. Every NULL pointer is black
3. If a node is red, both children are black
4. Every path from node to descendent leaf contains the same number of black nodes
5. The root is always black

Insert 1, 2, 3, 4, 5, 6, 7, **8**, 9, 10, 11



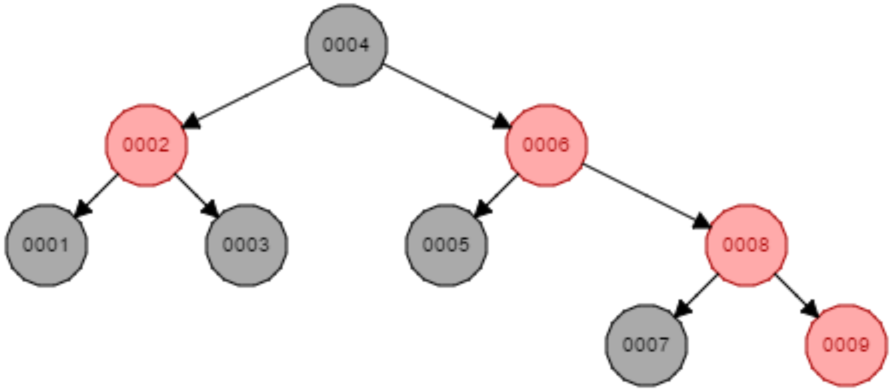
1. Every node is either red or black
2. Every NULL pointer is black
3. If a node is red, both children are black
4. Every path from node to descendent leaf contains the same number of black nodes
5. The root is always black

Insert 1, 2, 3, 4, 5, 6, 7, 8, **9**, 10, 11



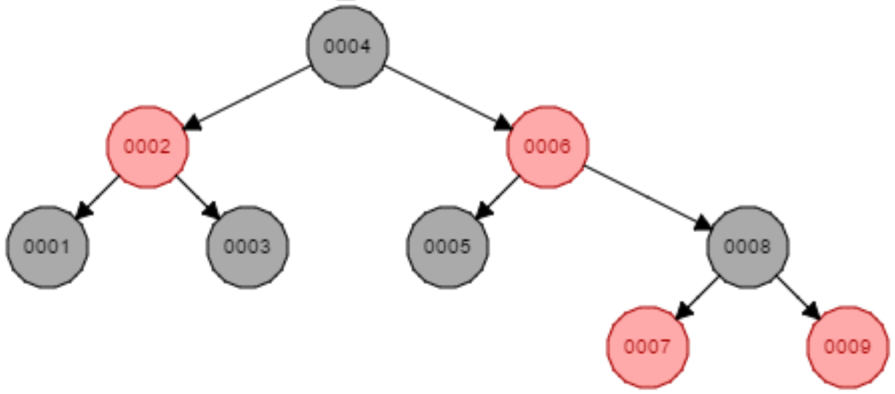
1. Every node is either red or black
2. Every NULL pointer is black
3. If a node is red, both children are black
4. Every path from node to descendent leaf contains the same number of black nodes
5. The root is always black

Insert 1, 2, 3, 4, 5, 6, 7, 8, **9**, 10, 11



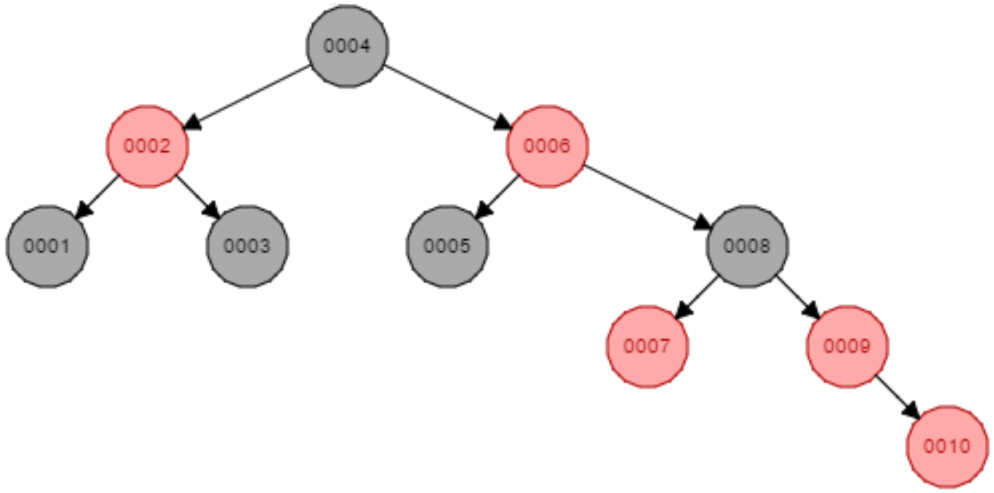
1. Every node is either red or black
2. Every NULL pointer is black
3. If a node is red, both children are black
4. Every path from node to descendent leaf contains the same number of black nodes
5. The root is always black

Insert 1, 2, 3, 4, 5, 6, 7, 8, **9**, 10, 11



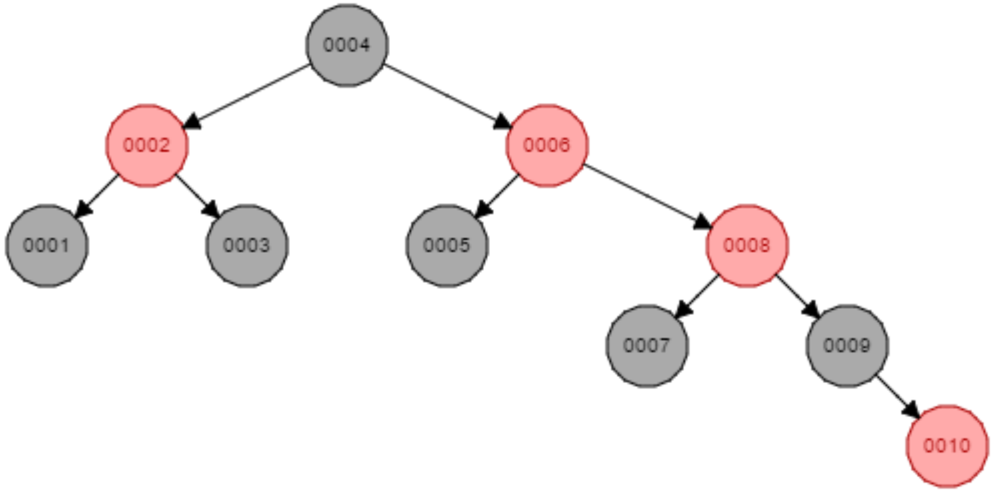
1. Every node is either red or black
2. Every NULL pointer is black
3. If a node is red, both children are black
4. Every path from node to descendent leaf contains the same number of black nodes
5. The root is always black

Insert 1, 2, 3, 4, 5, 6, 7, 8, 9, **10**, 11



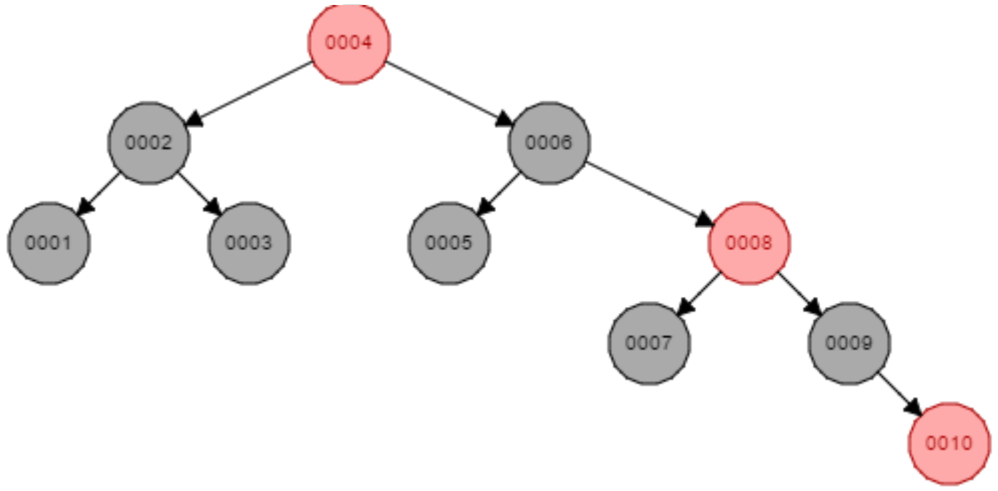
1. Every node is either red or black
2. Every NULL pointer is black
3. If a node is red, both children are black
4. Every path from node to descendent leaf contains the same number of black nodes
5. The root is always black

Insert 1, 2, 3, 4, 5, 6, 7, 8, 9, **10**, 11



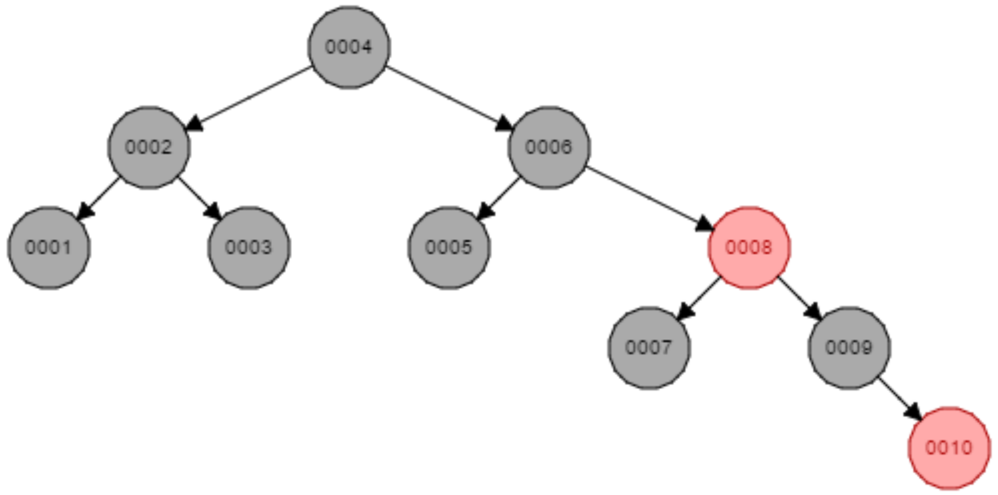
1. Every node is either red or black
2. Every NULL pointer is black
3. If a node is red, both children are black
4. Every path from node to descendent leaf contains the same number of black nodes
5. The root is always black

Insert 1, 2, 3, 4, 5, 6, 7, 8, 9, **10**, 11



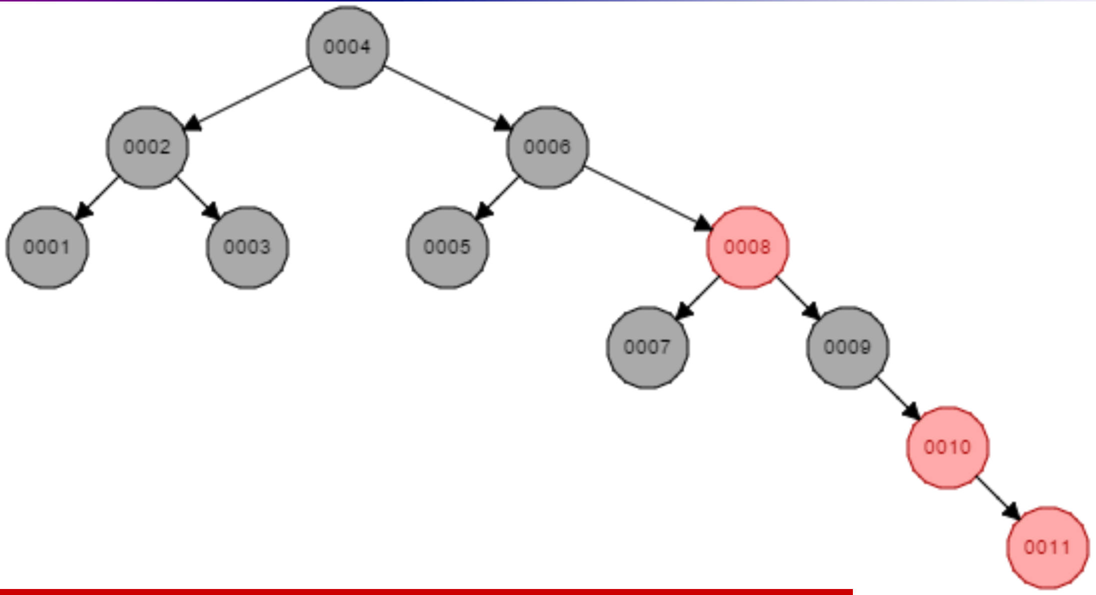
1. Every node is either red or black
2. Every NULL pointer is black
3. If a node is red, both children are black
4. Every path from node to descendent leaf contains the same number of black nodes
5. The root is always black

Insert 1, 2, 3, 4, 5, 6, 7, 8, 9, **10**, 11



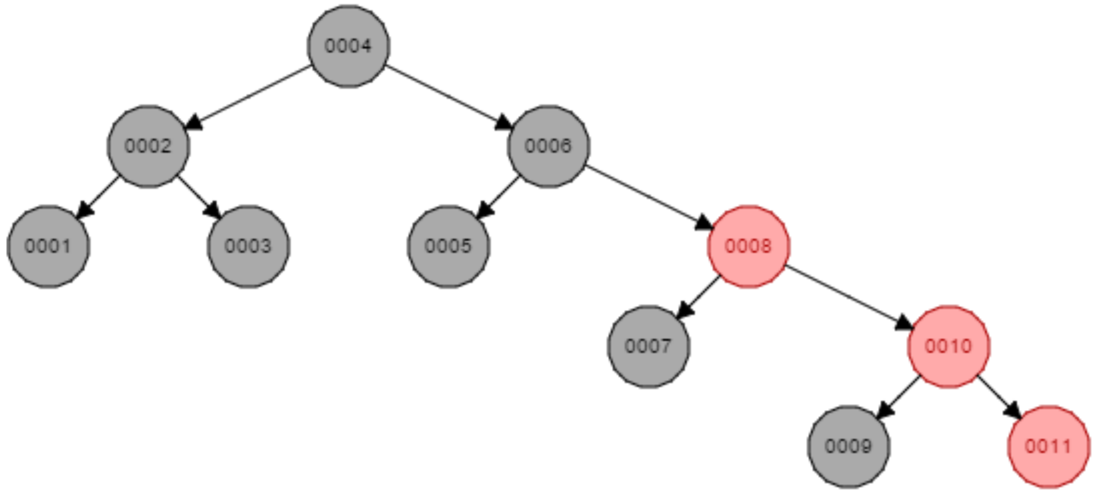
1. Every node is either red or black
2. Every NULL pointer is black
3. If a node is red, both children are black
4. Every path from node to descendent leaf contains the same number of black nodes
5. The root is always black

Insert 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11



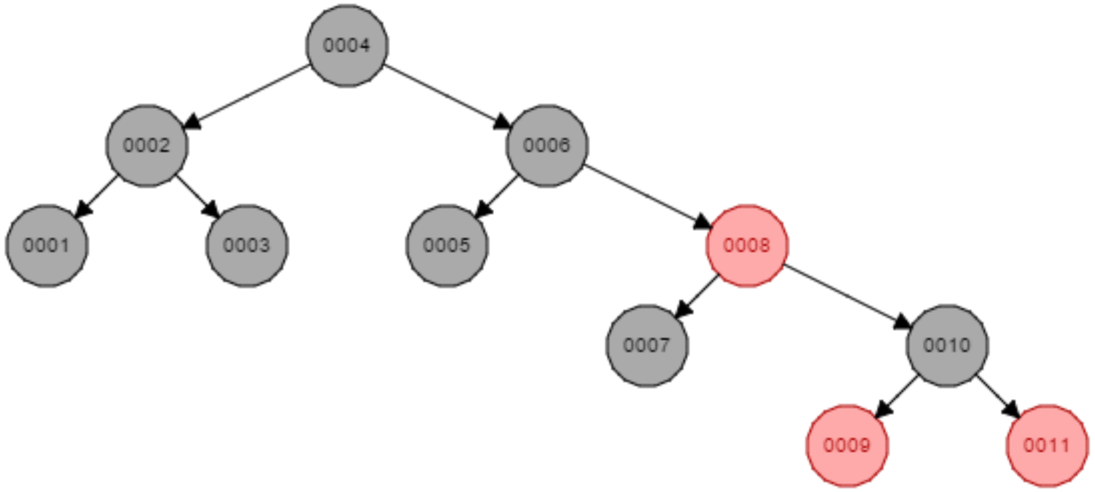
1. Every node is either red or black
2. Every NULL pointer is black
3. If a node is red, both children are black
4. Every path from node to descendent leaf contains the same number of black nodes
5. The root is always black

Insert 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, **11**



1. Every node is either red or black
2. Every NULL pointer is black
3. If a node is red, both children are black
4. Every path from node to descendent leaf contains the same number of black nodes
5. The root is always black

Insert 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, **11**



1. Every node is either red or black
2. Every NULL pointer is black
3. If a node is red, both children are black
4. Every path from node to descendent leaf contains the same number of black nodes
5. The root is always black

Red Black Trees

Insertion

5 Cases of Insertion

- **N** is the root node, i.e., first node of red–black tree
- **N**'s parent (**P**) is black
- **N**'s parent (**P**) and uncle (**U**) are red
- **N** is added to right of left child of grandparent, or **N** is added to left of right child of grandparent (**P** is red and **U** is black)
- **N** is added to left of left child of grandparent, or **N** is added to right of right child of grandparent (**P** is red and **U** is black)

Case 1

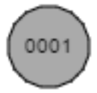
Insertion

Case 1

- All nodes are initially inserted as Red (except the root node)

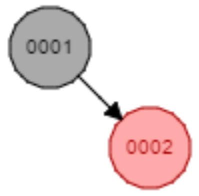
Insert 1, 2

- Always insert nodes as red (except the root)



1. Every node is either red or black
2. Every NULL pointer is black
3. If a node is red, both children are black
4. Every path from node to descendent leaf contains the same number of black nodes
5. The root is always black

Insert 1, 2



1. Every node is either red or black
2. Every NULL pointer is black
3. If a node is red, both children are black
4. Every path from node to descendent leaf contains the same number of black nodes
5. The root is always black

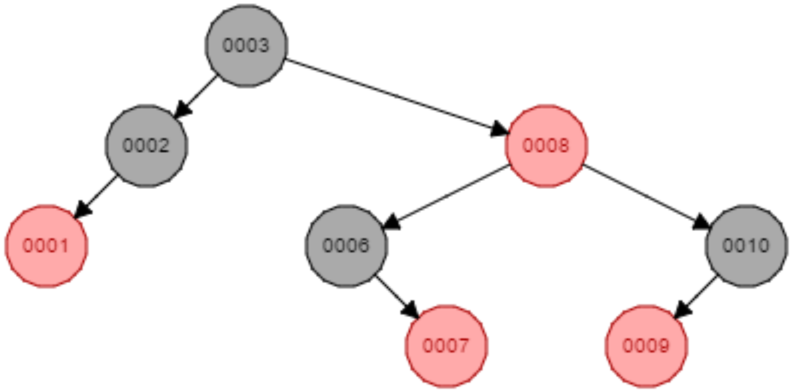
Case 2

Insertion

Case 2

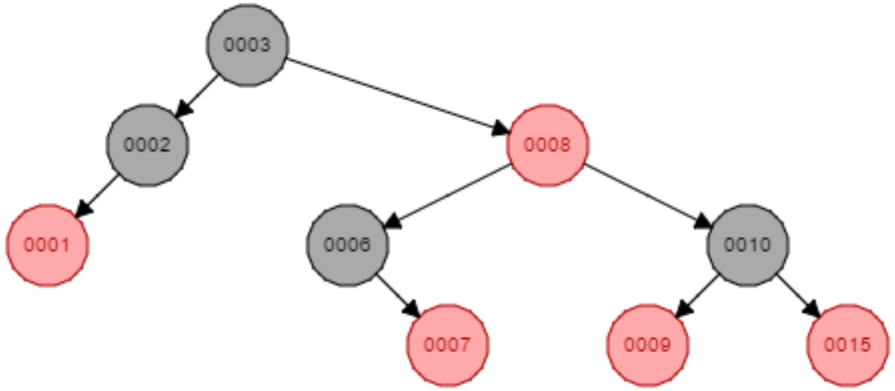
- **N**'s parent (**P**) is black
- No matter how complex the tree already is, if new node’s parent is already black, the tree will always remain a Red-Black tree.

Insert **15, 5**



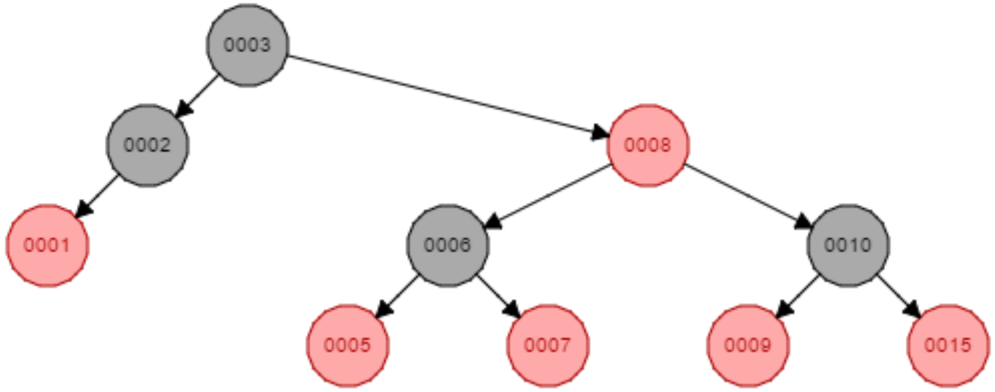
1. Every node is either red or black
2. Every NULL pointer is black
3. If a node is red, both children are black
4. Every path from node to descendent leaf contains the same number of black nodes
5. The root is always black

Insert 15, 5



1. Every node is either red or black
2. Every NULL pointer is black
3. If a node is red, both children are black
4. Every path from node to descendent leaf contains the same number of black nodes
5. The root is always black

Insert **15**, **5**



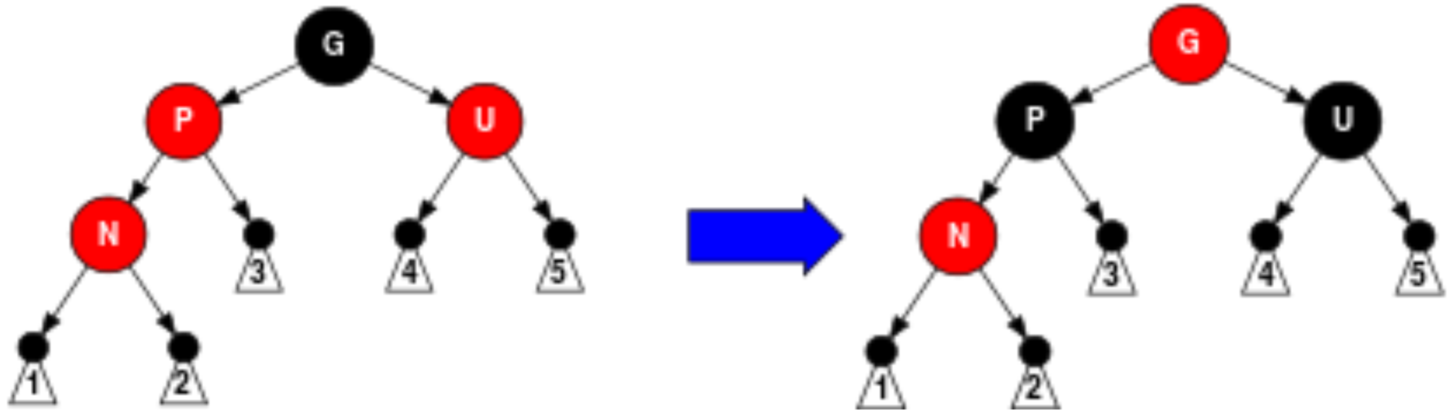
1. Every node is either red or black
2. Every NULL pointer is black
3. If a node is red, both children are black
4. Every path from node to descendent leaf contains the same number of black nodes
5. The root is always black

Case 3

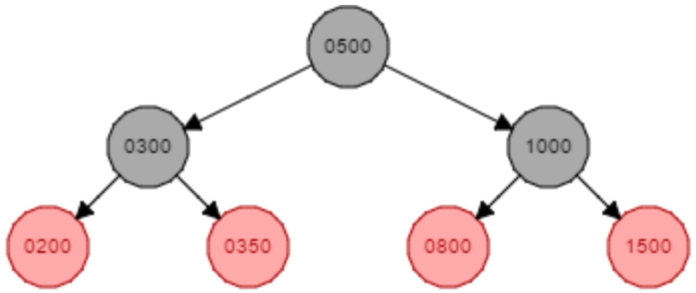
Insertion

Case 3

- **N**'s parent (**P**) and uncle (**U**) are red
- Solution: Re-colour

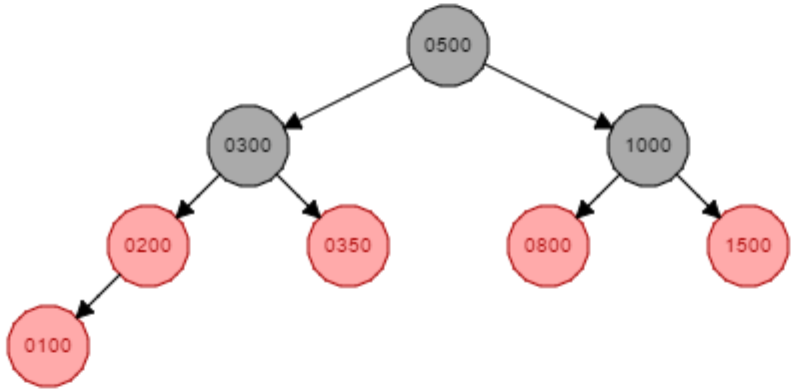


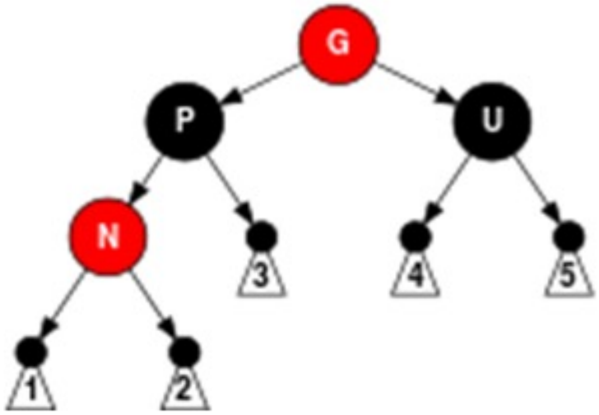
Insert **100, 1800**



1. Every node is either red or black
2. Every NULL pointer is black
3. If a node is red, both children are black
4. Every path from node to descendent leaf contains the same number of black nodes
5. The root is always black

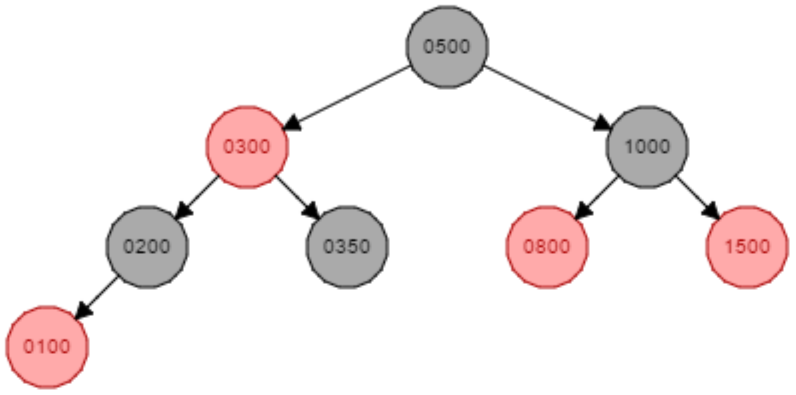
Insert 100, 1800





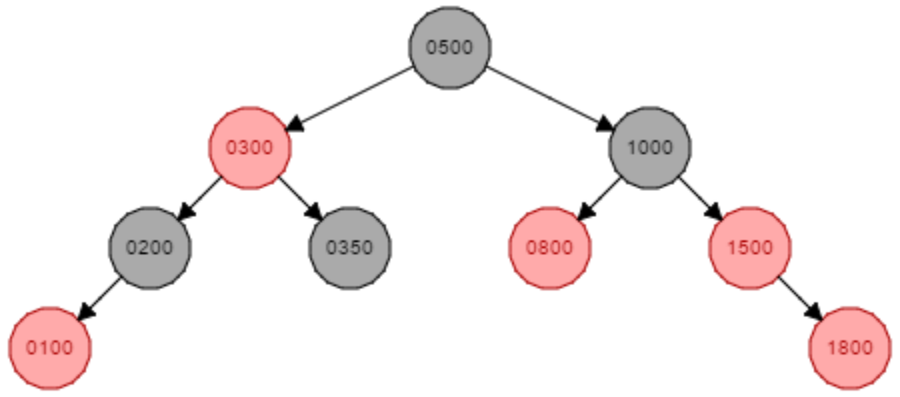
1. Every node is either red or black
2. Every NULL pointer is black
3. If a node is red, both children are black
4. Every path from node to descendent leaf contains the same number of black nodes
5. The root is always black

Insert 100, 1800



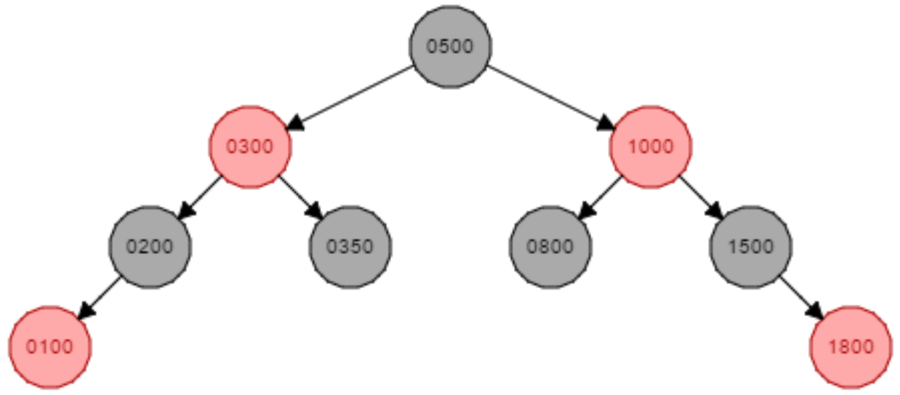
1. Every node is either red or black
2. Every NULL pointer is black
3. If a node is red, both children are black
4. Every path from node to descendent leaf contains the same number of black nodes
5. The root is always black

Insert **100**, **1800**



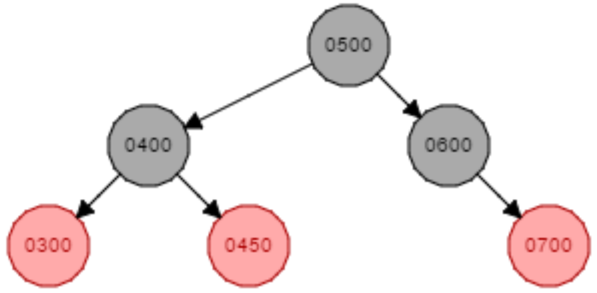
1. Every node is either red or black
2. Every NULL pointer is black
3. If a node is red, both children are black
4. Every path from node to descendent leaf contains the same number of black nodes
5. The root is always black

Insert **100**, **1800**



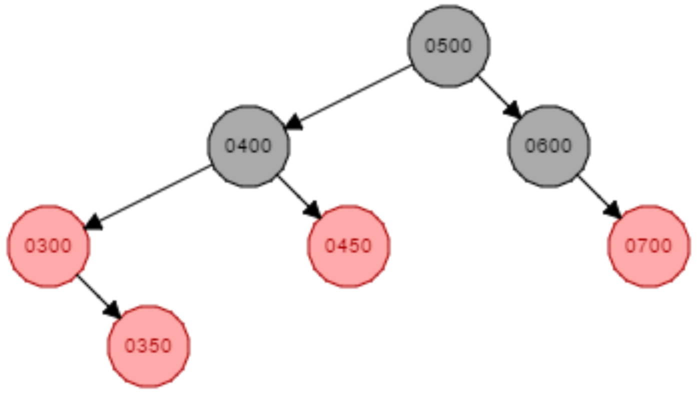
1. Every node is either red or black
2. Every NULL pointer is black
3. If a node is red, both children are black
4. Every path from node to descendent leaf contains the same number of black nodes
5. The root is always black

DIY: Insert **350**



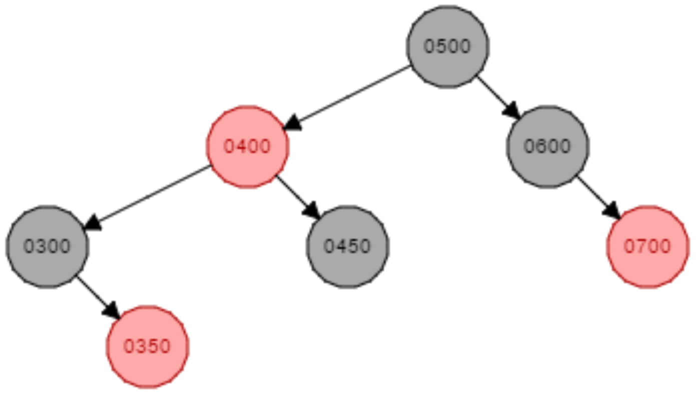
1. Every node is either red or black
2. Every NULL pointer is black
3. If a node is red, both children are black
4. Every path from node to descendent leaf contains the same number of black nodes
5. The root is always black

DIY: Insert **350**



1. Every node is either red or black
2. Every NULL pointer is black
3. If a node is red, both children are black
4. Every path from node to descendent leaf contains the same number of black nodes
5. The root is always black

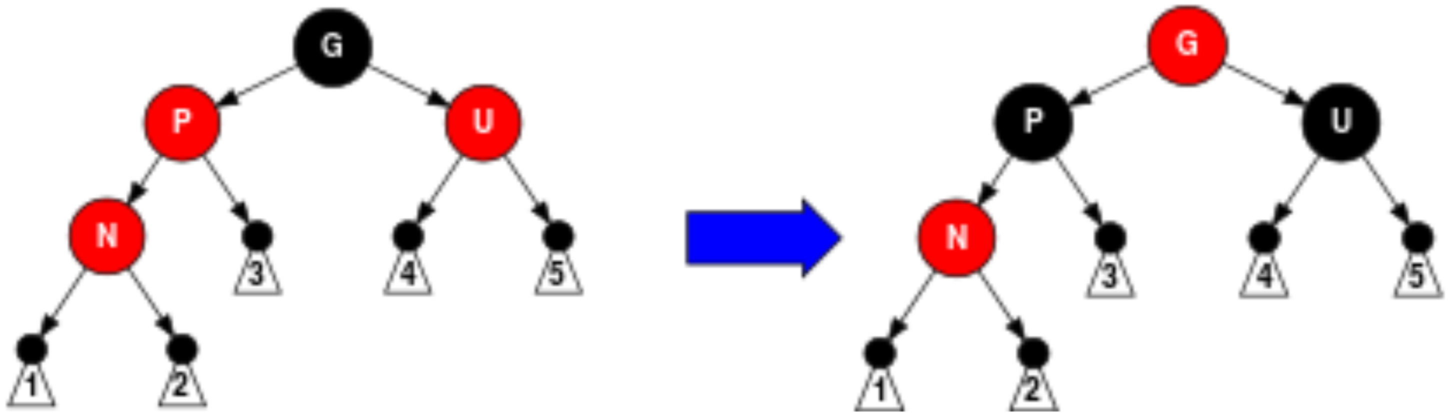
DIY: Insert **350**



1. Every node is either red or black
2. Every NULL pointer is black
3. If a node is red, both children are black
4. Every path from node to descendent leaf contains the same number of black nodes
5. The root is always black

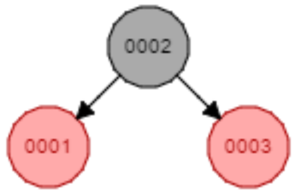
Case 3 (in case of root)

- **N**'s parent (**P**) and uncle (**U**) are red
- Solution: Re-colour



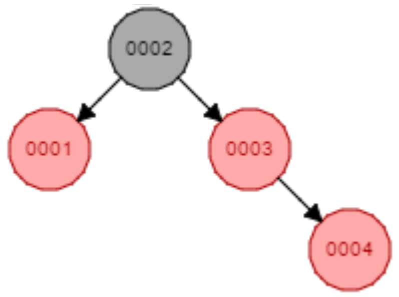
- Never forget the root’s case

Insert **4**



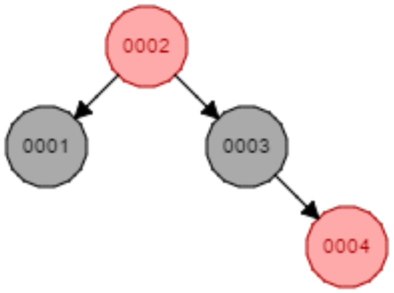
1. Every node is either red or black
2. Every NULL pointer is black
3. If a node is red, both children are black
4. Every path from node to descendent leaf contains the same number of black nodes
5. The root is always black

Insert 4



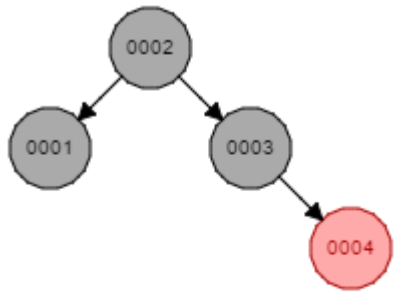
1. Every node is either red or black
2. Every NULL pointer is black
3. If a node is red, both children are black
4. Every path from node to descendent leaf contains the same number of black nodes
5. The root is always black

Insert 4



1. Every node is either red or black
2. Every NULL pointer is black
3. If a node is red, both children are black
4. Every path from node to descendent leaf contains the same number of black nodes
5. The root is always black

Insert 4



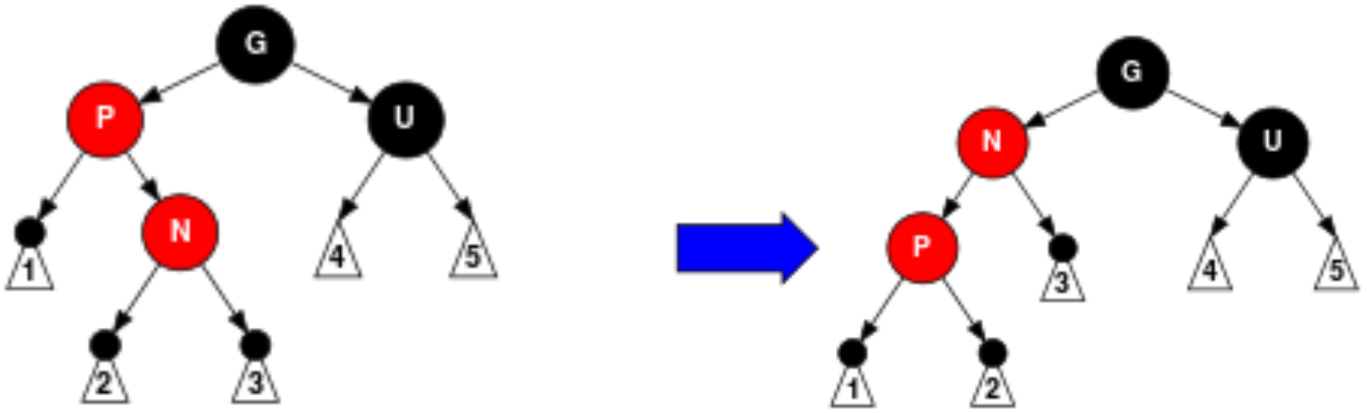
1. Every node is either red or black
2. Every NULL pointer is black
3. If a node is red, both children are black
4. Every path from node to descendent leaf contains the same number of black nodes
5. The root is always black

Case 4

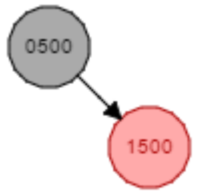
Insertion

Case 4

- **N** is added to right of left child of grandparent, or **N** is added to left of right child of grandparent (**P** is red and **U** is black)
- Intermediate Solution: Rotation of 2 nodes
- Move to Case 5

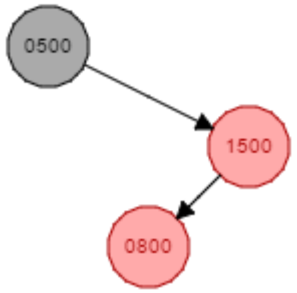


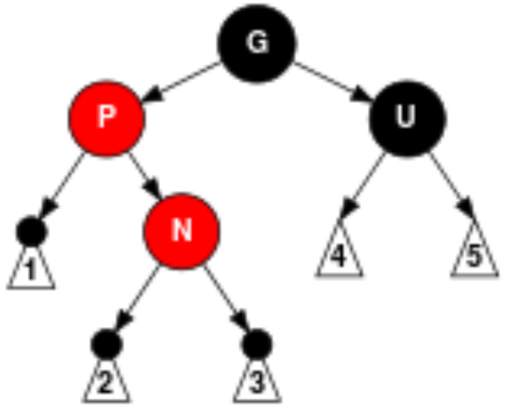
Insert **800**



1. Every node is either red or black
2. Every NULL pointer is black
3. If a node is red, both children are black
4. Every path from node to descendent leaf contains the same number of black nodes
5. The root is always black

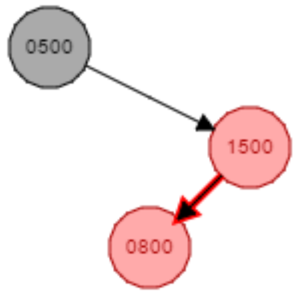
Insert **800**

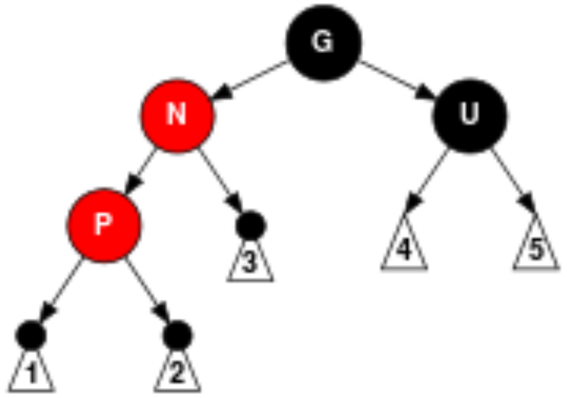




1. Every node is either red or black
2. Every NULL pointer is black
3. If a node is red, both children are black
4. Every path from node to descendent leaf contains the same number of black nodes
5. The root is always black

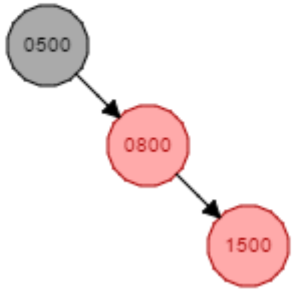
Insert **800**

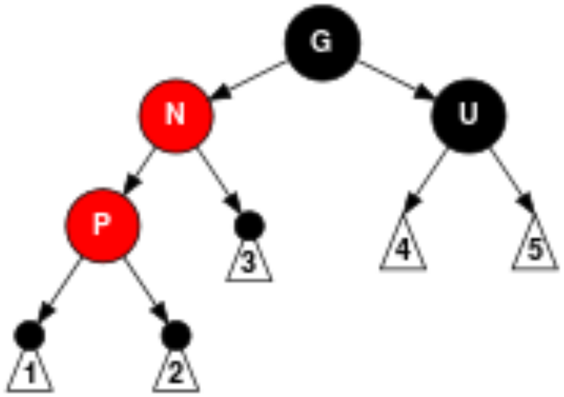




1. Every node is either red or black
2. Every NULL pointer is black
3. If a node is red, both children are black
4. Every path from node to descendent leaf contains the same number of black nodes
5. The root is always black

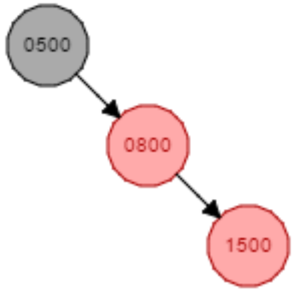
Insert **800**

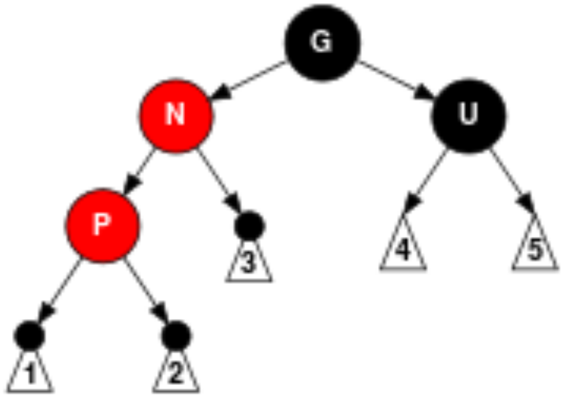




1. Every node is either red or black
2. Every NULL pointer is black
3. If a node is red, both children are black
4. Every path from node to descendent leaf contains the same number of black nodes
5. The root is always black

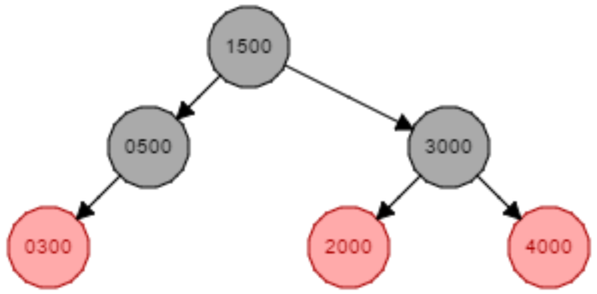
Insert **800**





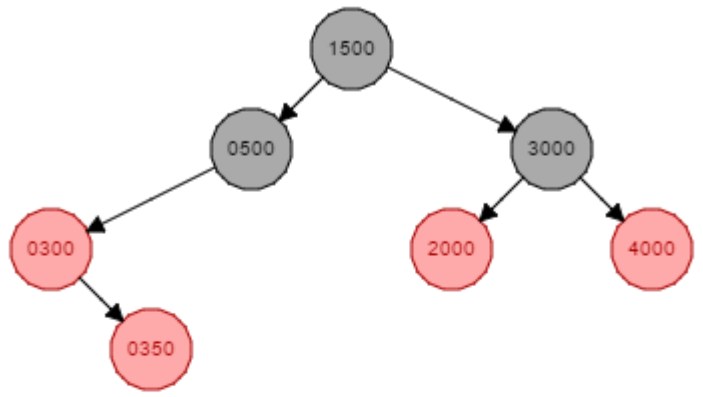
1. Every node is either red or black
2. Every NULL pointer is black
3. If a node is red, both children are black
4. Every path from node to descendent leaf contains the same number of black nodes
5. The root is always black

DIY: Insert **350**



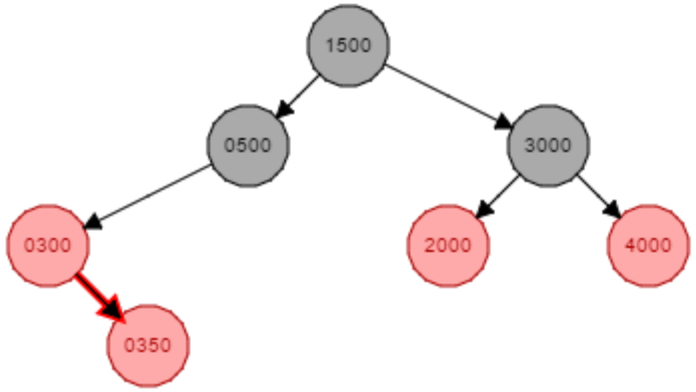
1. Every node is either red or black
2. Every NULL pointer is black
3. If a node is red, both children are black
4. Every path from node to descendent leaf contains the same number of black nodes
5. The root is always black

DIY: Insert **350**



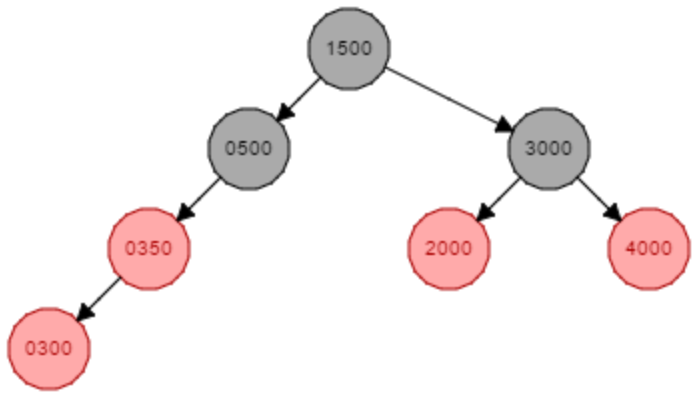
1. Every node is either red or black
2. Every NULL pointer is black
3. If a node is red, both children are black
4. Every path from node to descendent leaf contains the same number of black nodes
5. The root is always black

DIY: Insert 350



1. Every node is either red or black
2. Every NULL pointer is black
3. If a node is red, both children are black
4. Every path from node to descendent leaf contains the same number of black nodes
5. The root is always black

DIY: Insert **350**



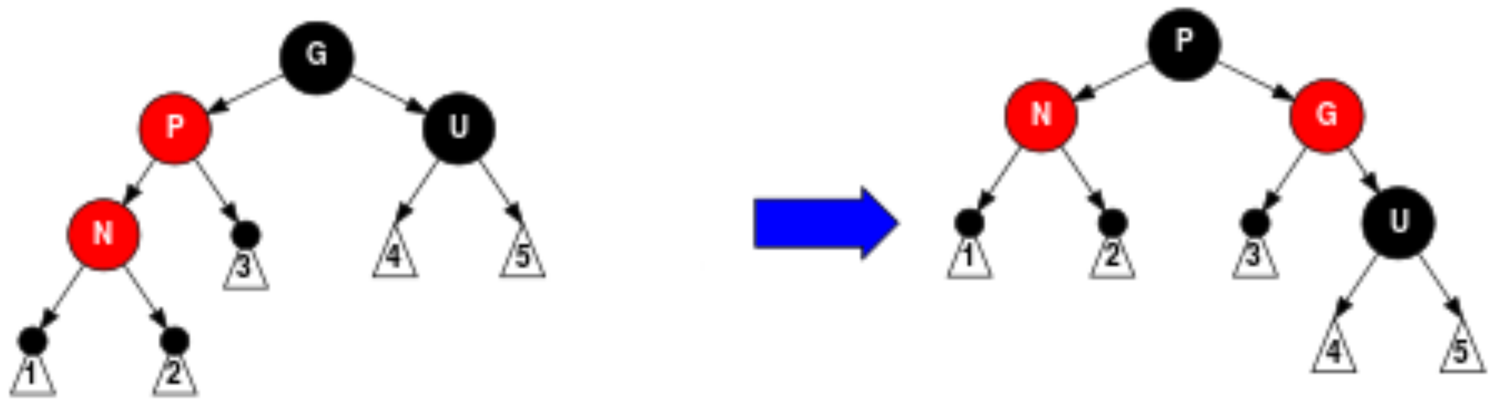
1. Every node is either red or black
2. Every NULL pointer is black
3. If a node is red, both children are black
4. Every path from node to descendent leaf contains the same number of black nodes
5. The root is always black

Case 5

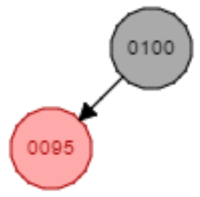
Insertion

Case 5

- **N** is added to left of left child of grandparent, or **N** is added to right of right child of grandparent (**P** is red and **U** is black)
- Solution: Rotation

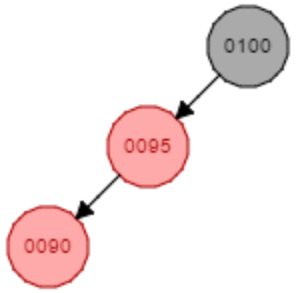


Insert **90**



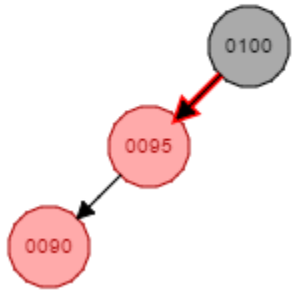
1. Every node is either red or black
2. Every NULL pointer is black
3. If a node is red, both children are black
4. Every path from node to descendent leaf contains the same number of black nodes
5. The root is always black

Insert **90**



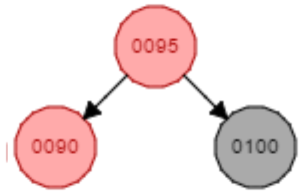
1. Every node is either red or black
2. Every NULL pointer is black
3. If a node is red, both children are black
4. Every path from node to descendent leaf contains the same number of black nodes
5. The root is always black

Insert **90**



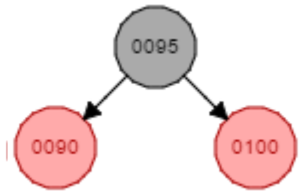
1. Every node is either red or black
2. Every NULL pointer is black
3. If a node is red, both children are black
4. Every path from node to descendent leaf contains the same number of black nodes
5. The root is always black

Insert **90**



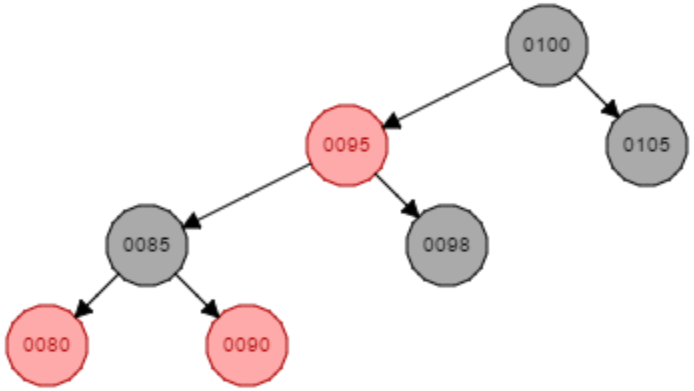
1. Every node is either red or black
2. Every NULL pointer is black
3. If a node is red, both children are black
4. Every path from node to descendent leaf contains the same number of black nodes
5. The root is always black

Insert **90**



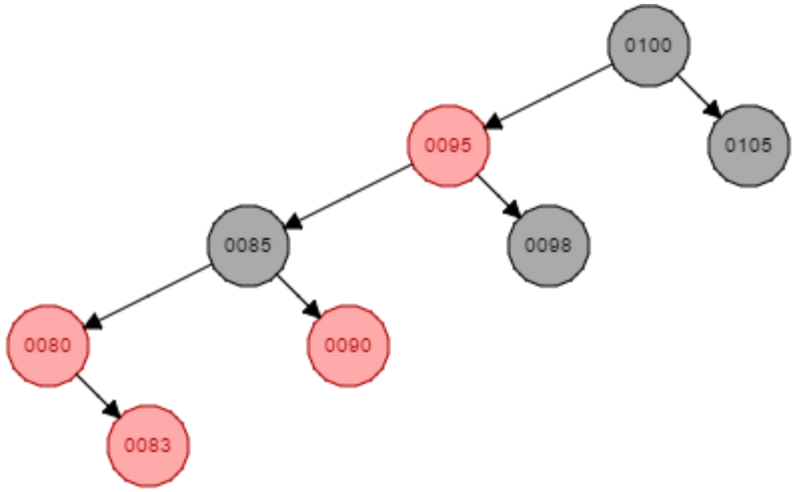
1. Every node is either red or black
2. Every NULL pointer is black
3. If a node is red, both children are black
4. Every path from node to descendent leaf contains the same number of black nodes
5. The root is always black

Do It Yourself (DIY): Insert **83**



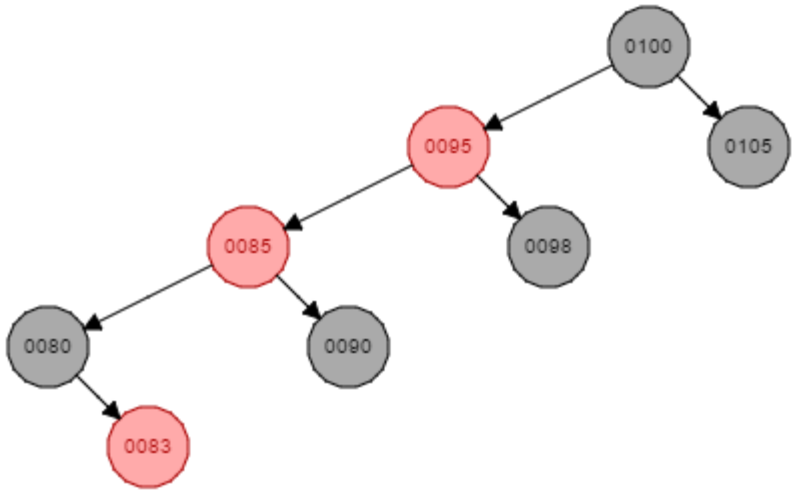
1. Every node is either red or black
2. Every NULL pointer is black
3. If a node is red, both children are black
4. Every path from node to descendent leaf contains the same number of black nodes
5. The root is always black

DIY: Insert **83**



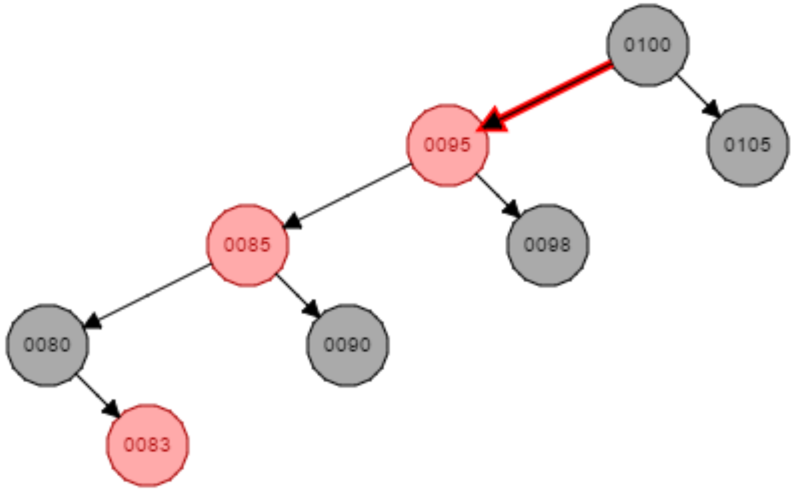
1. Every node is either red or black
2. Every NULL pointer is black
3. If a node is red, both children are black
4. Every path from node to descendent leaf contains the same number of black nodes
5. The root is always black

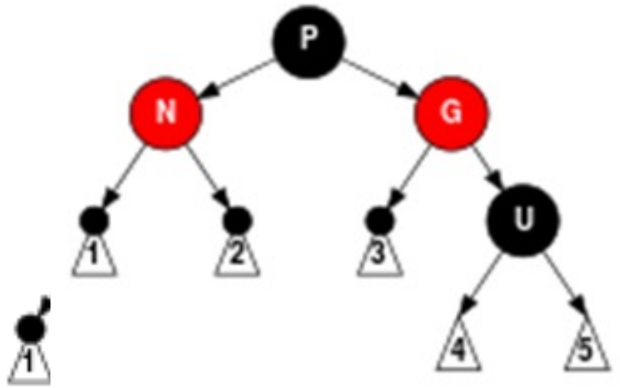
DIY: Insert **83**



1. Every node is either red or black
2. Every NULL pointer is black
3. If a node is red, both children are black
4. Every path from node to descendent leaf contains the same number of black nodes
5. The root is always black

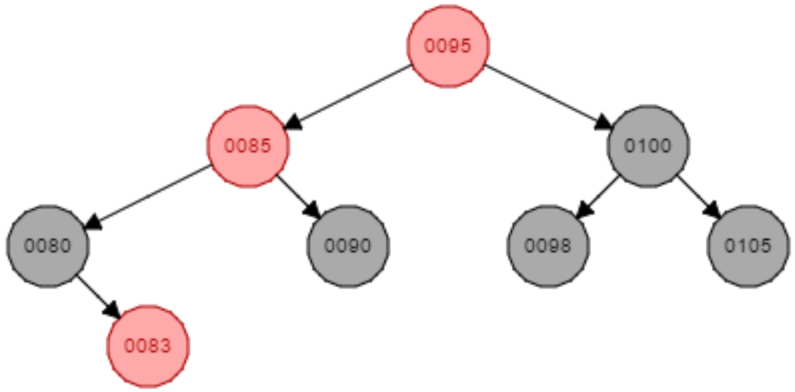
DIY: Insert **83**

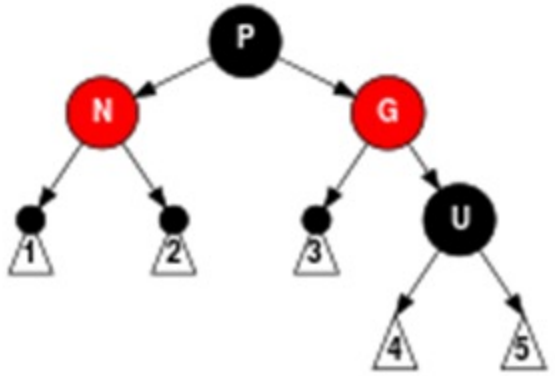




1. Every node is either red or black
2. Every NULL pointer is black
3. If a node is red, both children are black
4. Every path from node to descendent leaf contains the same number of black nodes
5. The root is always black

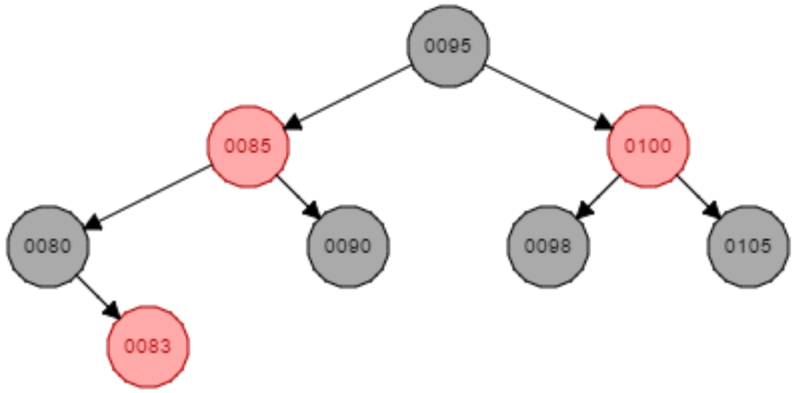
DIY: Insert 83





1. Every node is either red or black
2. Every NULL pointer is black
3. If a node is red, both children are black
4. Every path from node to descendent leaf contains the same number of black nodes
5. The root is always black

DIY: Insert **83**

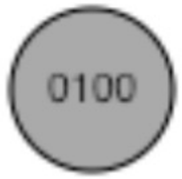


1. Every node is either red or black
2. Every NULL pointer is black
3. If a node is red, both children are black
4. Every path from node to descendent leaf contains the same number of black nodes
5. The root is always black

Problem with Inserting Duplicates

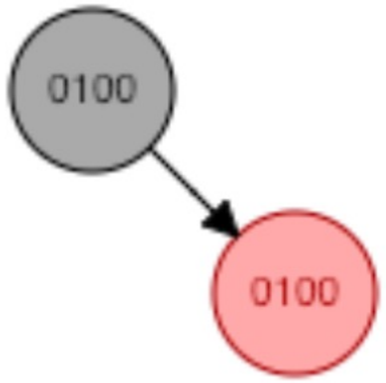
Insert: 100, 100, 100

Insert **100,** 100, 100



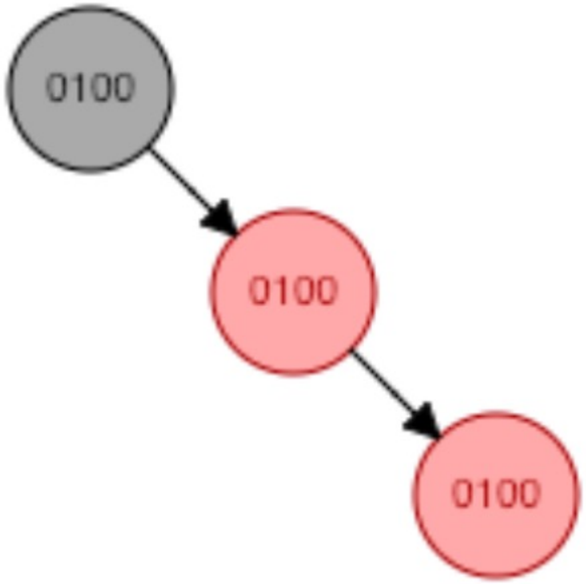
1. Every node is either red or black
2. Every NULL pointer is black
3. If a node is red, both children are black
4. Every path from node to descendent leaf contains the same number of black nodes
5. The root is always black

Insert 100, **100,** 100



1. Every node is either red or black
2. Every NULL pointer is black
3. If a node is red, both children are black
4. Every path from node to descendent leaf contains the same number of black nodes
5. The root is always black

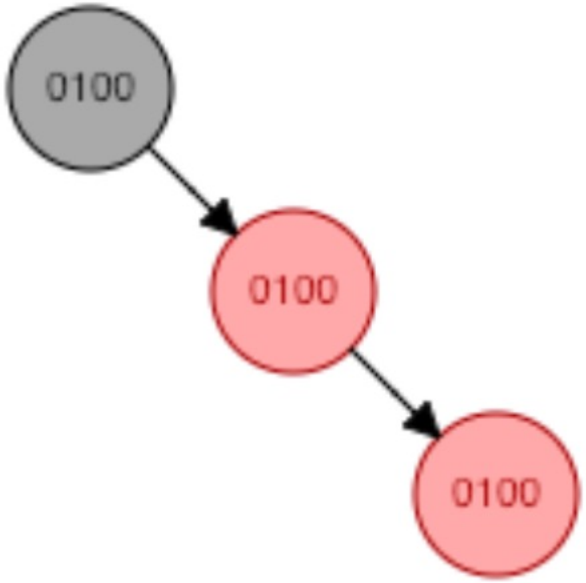
Insert 100, 100, **100**



1. Every node is either red or black
2. Every NULL pointer is black
3. If a node is red, both children are black
4. Every path from node to descendent leaf contains the same number of black nodes
5. The root is always black

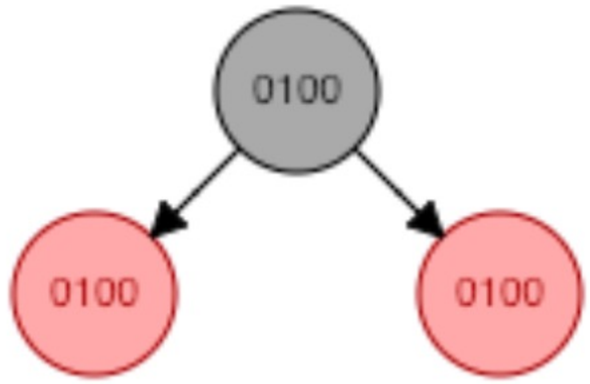
Insert 100, 100, **100**

Rotation (rule 5)



1. Every node is either red or black
2. Every NULL pointer is black
3. If a node is red, both children are black
4. Every path from node to descendent leaf contains the same number of black nodes
5. The root is always black

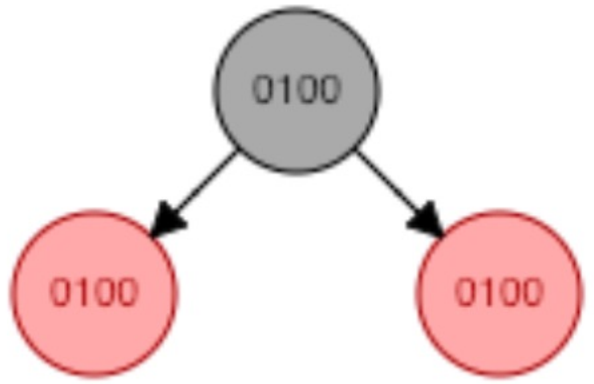
Insert 100, 100, **100**



1. Every node is either red or black
2. Every NULL pointer is black
3. If a node is red, both children are black
4. Every path from node to descendent leaf contains the same number of black nodes
5. The root is always black

Insert 100, 100, **100**

See the problem?

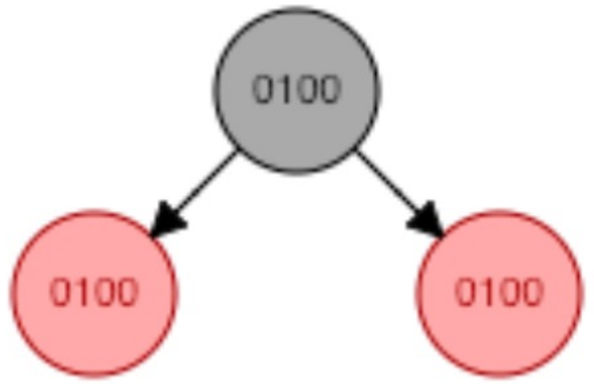


1. Every node is either red or black
2. Every NULL pointer is black
3. If a node is red, both children are black
4. Every path from node to descendent leaf contains the same number of black nodes
5. The root is always black

Insert 100, 100, **100**

See the problem?

- Values greater or equal to parent node must be inserted at right

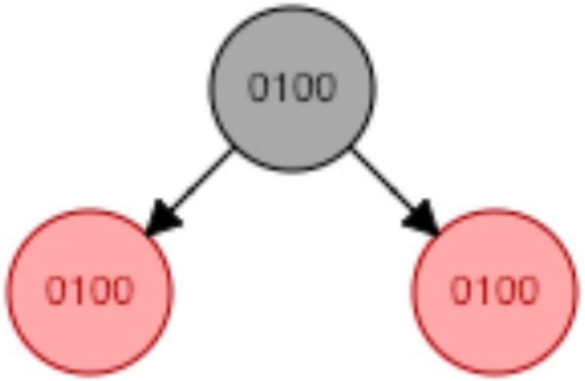


1. Every node is either red or black
2. Every NULL pointer is black
3. If a node is red, both children are black
4. Every path from node to descendent leaf contains the same number of black nodes
5. The root is always black

Insert 100, 100, **100**

See the problem?

- Values greater or equal to parent node must be inserted at right (NOT left)

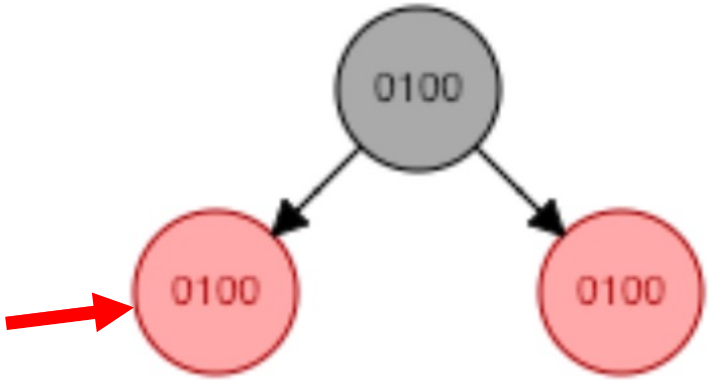


1. Every node is either red or black
2. Every NULL pointer is black
3. If a node is red, both children are black
4. Every path from node to descendent leaf contains the same number of black nodes
5. The root is always black

Insert 100, 100, **100**

See the problem?

- Values greater or equal to parent node must be inserted at right (NOT left)



1. Every node is either red or black
2. Every NULL pointer is black
3. If a node is red, both children are black
4. Every path from node to descendent leaf contains the same number of black nodes
5. The root is always black

SOLUTION

```
template <class T>
struct Node
{
        T data;
        Node<T> *left;
        Node<T> *right;
        char colour;
        int count;
};
```

SOLUTION

There is an extra attribute namely COUNT that would be incremented if an already existing value is added to the tree.

SOLUTION

- Count > = 1
- Count will never be 0.
- All duplicate values can be deleted by making the value of COUNT equal to 1 in all the existing nodes.
- If a node has to be deleted all together, its memory will be deleted using the DELETE reserved word.