DESIGN AND ANALYSIS OF ALGORITHMS

Surabhi Narayan

Department of Computer Science & Engineering

DESIGN AND ANALYSIS OF ALGORITHMS

RED BLACK TREE

Surabhi Narayan

Department of Computer Science & Engineering

Red Black Tree

- A Red Black Tree is a category of the self-balancing binary search tree.
- Created in 1972 by Rudolf Bayer who termed them **"symmetric binary B-trees.**"
- A red-black tree is a Binary tree where a particular node has color as an extra attribute, either red or black.
- These colours are used to ensure that the tree remains balanced during insertions and deletions.
- Although the balance of the tree is not perfect, it is good enough to reduce the searching time and maintain it around O(log n) time, where n is the total number of elements in the tree.

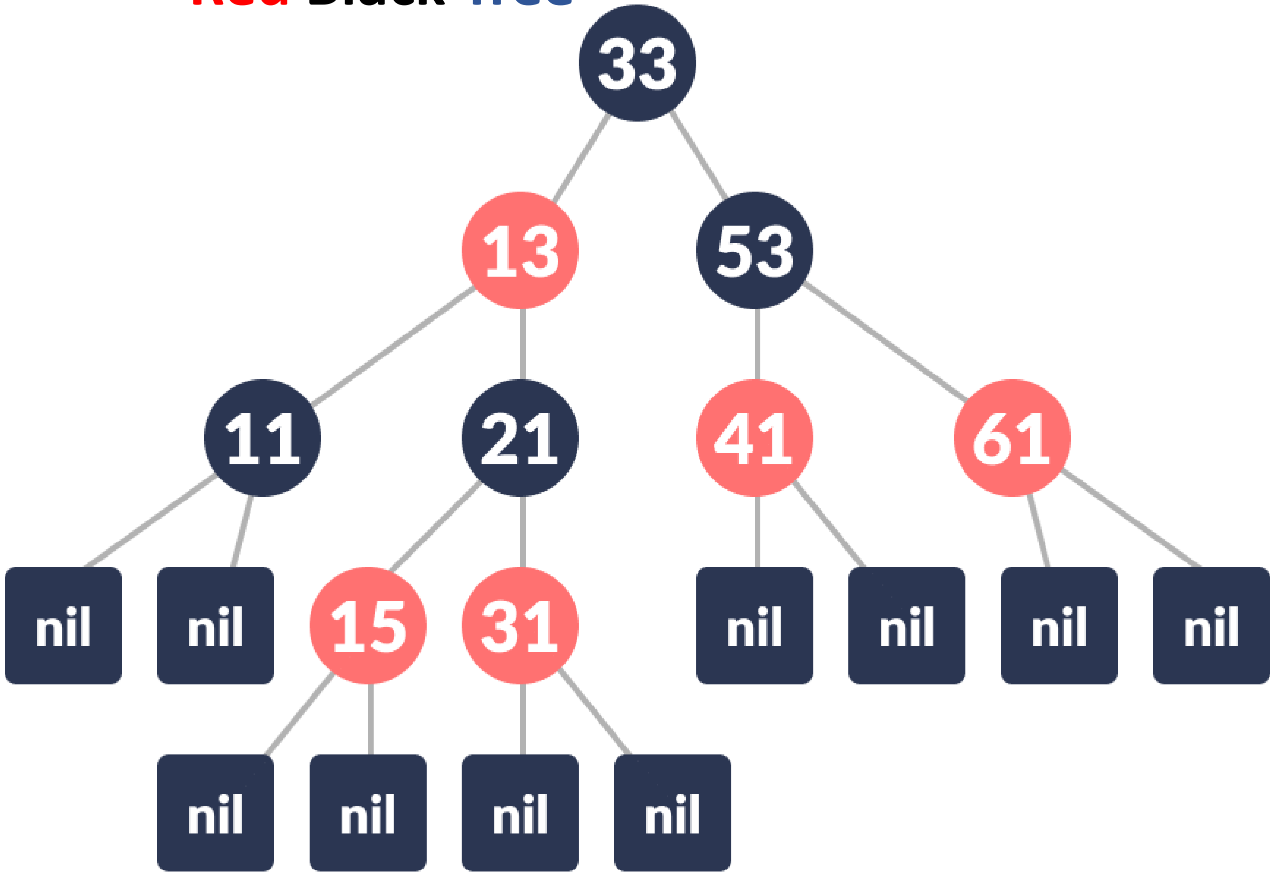
Red Black Tree

A red-black tree satisfies the following properties:

- **Red/Black Property:** Every node is colored, either red or black.
- **Root Property:** The root is black.
- **Leaf Property:** Every leaf (NIL) is black.
- **Red Property:** If a red node has children then, the children are always black.
- **Depth Property:** For each node, any simple path from this node to any of its descendant leaf has the same black-depth (the number of black nodes).

Red Black Tree

Red Black Tree



Each node has the following attributes:

- Color
- key
- leftChild
- rightChild
- parent (except root node)

Interesting points about Red-Black Tree:

- Black height of the red-black tree is the number of black nodes on a path from the root node to a leaf node. Leaf nodes are also counted as black nodes. So, a red-black tree of height h has black height >= h/2. *Number of nodes from a node to its farthest descendant leaf is no more than twice as the number of nodes to the nearest descendant leaf.*

- Height of a red-black tree with n nodes is $h \le 2 \log_2(n + 1)$.
- All leaves (NIL) are black.
- The black depth of a node is defined as the number of black nodes from the root to that node i.e the number of black ancestors.
- Every red-black tree is a special case of a binary tree.

Applications:

- Most of the self-balancing BST library functions like map and set in C++ (OR TreeSet and TreeMap in Java) use Red-Black Tree.
- It is used to implement CPU Scheduling Linux. Completely Fair Scheduler uses it.
- Besides they are used in the K-mean clustering algorithm for reducing time complexity.
- Moreover, MySQL also uses the Red-Black tree for indexes on tables.

Operations on a Red-Black Tree

- insert a key value (insert)
- determine whether a key value is in the tree (lookup/search)
- remove key value from the tree (delete)
- print all of the key values in sorted order (print)

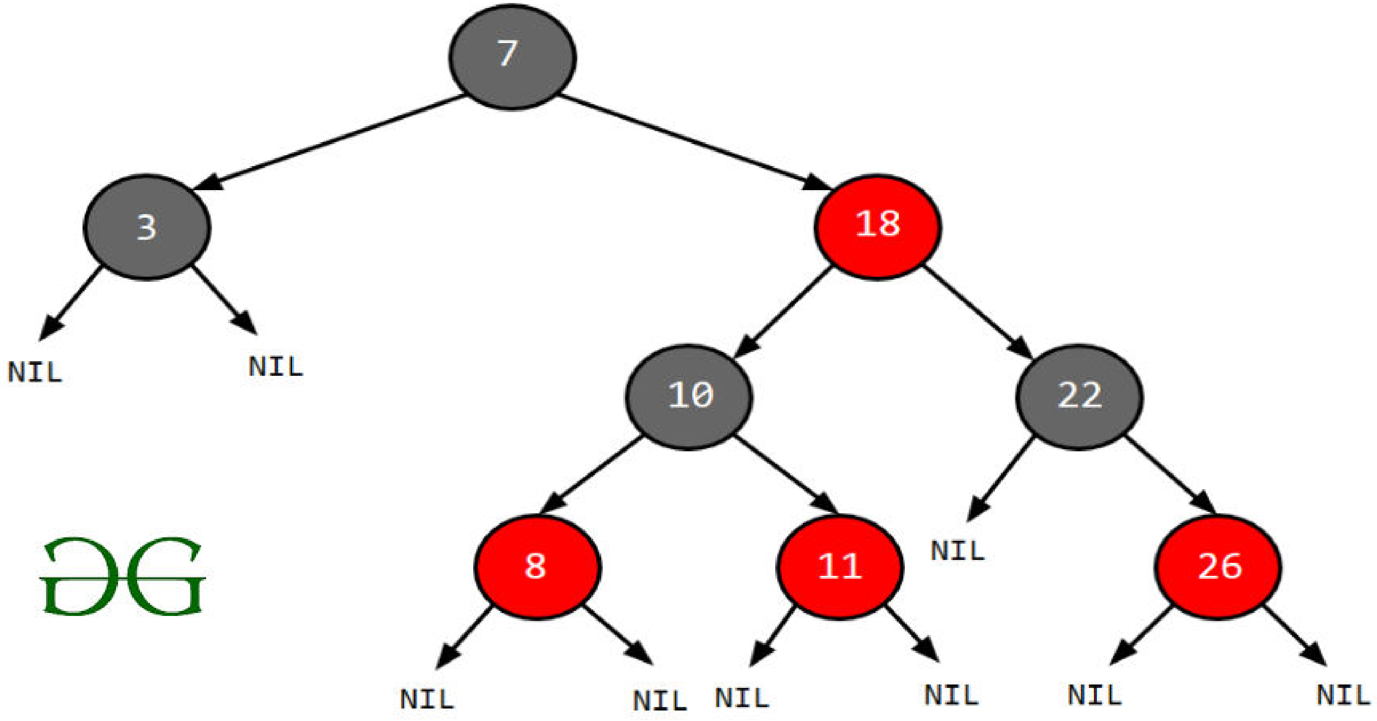
Search in Red-black Tree:

As every red-black tree is a special case of a binary tree so the searching algorithm of a red-black tree is similar to that of a binary tree.

```
searchElement (tree, val)
Step 1: If tree -> data = val OR tree = NULL
Return tree
    Else If val data
               Return searchElement (tree -> left, val)
           Else Return searchElement (tree -> right, val)
         [ End of if ]
[ End of if ]

Step 2: END
```

Search in Red-black Tree: search for the key 11



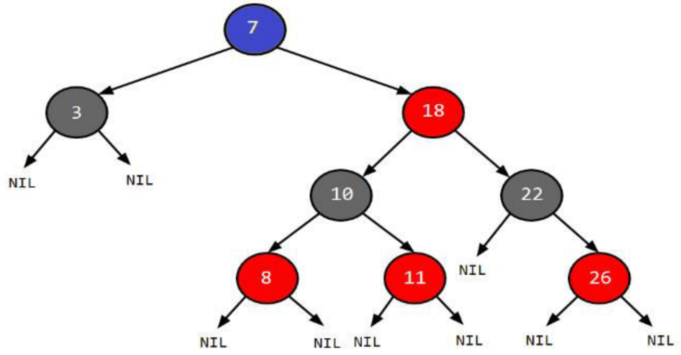
1.Start from the root.

2.Compare the inserting element with root, if less than root, then recurse for left, else recurse for right.

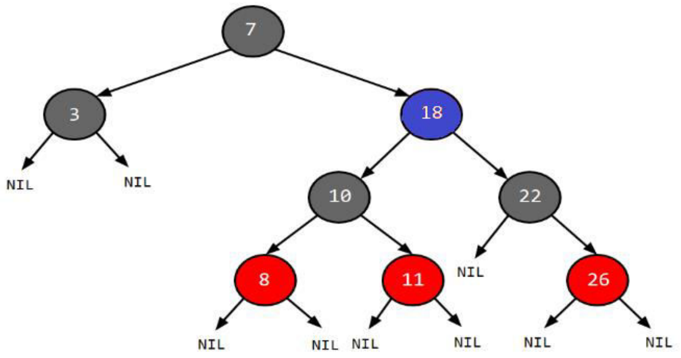
3.If the element to search is found anywhere, return true, else return false.

Search in Red-black Tree: search for the key 11

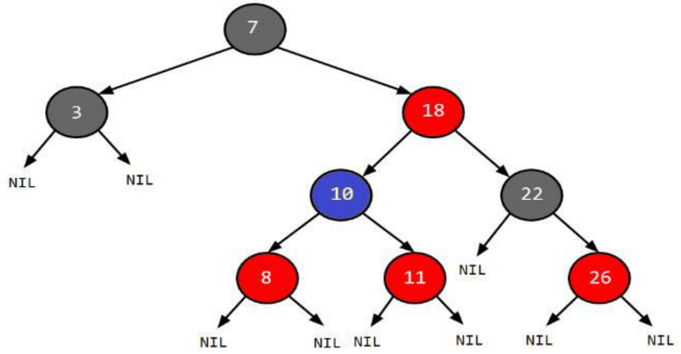
Step-1



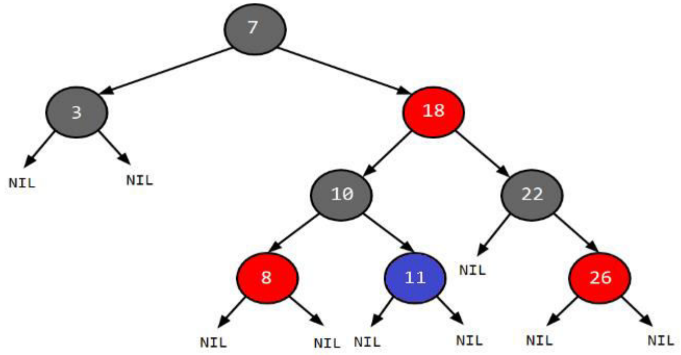
Step-2



Step-3



Step-4



Red Black Tree - Insertion

Let x be the newly inserted node.

1.Perform standard BST insertion and make the colour of newly inserted nodes as RED.

2.If x is the root, change the colour of x as BLACK (Black height of complete tree increases by 1).

3.Do the following if the color of x's parent is not BLACK **and** x is not the root.

a) If x's uncle is RED (Grandparent must have been black from property 4)

(i) Change the colour of parent and uncle as BLACK.

(ii) Colour of a grandparent as RED.

(iii) Change x = x's grandparent, repeat steps 2 and 3 for new x. **b) If x's uncle is BLACK**, then there can be four configurations for x, x's parent (**p**) and x's grandparent (**g**) (This is similar to AVL Tree)

(i) Left Left Case (p is left child of g and x is left child of p)

(ii) Left Right Case (p is left child of g and x is the right child of p)

(iii) Right Right Case (Mirror of case i)

(iv) Right Left Case (Mirror of case ii)

Red Black Tree - Insertion

Let x be the newly inserted node.

1.Perform standard BST insertion and make the colour of newly inserted nodes as RED.

2.If x is the root, change the colour of x as BLACK (Black height of complete tree increases by 1).

3.Do the following if the color of x's parent is not BLACK **and** x is not the root.

a) If x's uncle is RED (Grandparent must have been black from property 4)

(i) Change the colour of parent and uncle as BLACK.

(ii) Colour of a grandparent as RED.

(iii) Change x = x's grandparent, repeat steps 2 and 3 for new x. **b) If x's uncle is BLACK**, then there can be four configurations for x, x's parent (**p**) and x's grandparent (**g**) (This is similar to AVL Tree)

(i) Left Left Case (p is left child of g and x is left child of p)

(ii) Left Right Case (p is left child of g and x is the right child of p)

(iii) Right Right Case (Mirror of case i)

(iv) Right Left Case (Mirror of case ii)

Red Black Tree - Insertion

Let x be the newly inserted node.

1.Perform standard BST insertion and make the colour of newly inserted nodes as RED.

2.If x is the root, change the colour of x as BLACK (Black height of complete tree increases by 1).

3.Do the following if the color of x's parent is not BLACK **and** x is not the root.

a) If x's uncle is RED (Grandparent must have been black from property 4)

(i) Change the colour of parent and uncle as BLACK.

(ii) Colour of a grandparent as RED.

(iii) Change x = x's grandparent, repeat steps 2 and 3 for new x. **b) If x's uncle is BLACK**, then there can be four configurations for x, x's parent (**p**) and x's grandparent (**g**) (This is similar to AVL Tree)

(i) Left Left Case (p is left child of g and x is left child of p)

(ii) Left Right Case (p is left child of g and x is the right child of p)

(iii) Right Right Case (Mirror of case i)

(iv) Right Left Case (Mirror of case ii)

Red Black Tree - Insertion

Let x be the newly inserted node.

1.Perform standard BST insertion and make the colour of newly inserted nodes as RED.

2.If x is the root, change the colour of x as BLACK (Black height of complete tree increases by 1).

3.Do the following if the color of x's parent is not BLACK **and** x is not the root.

a) If x's uncle is RED (Grandparent must have been black from property 4)

(i) Change the colour of parent and uncle as BLACK.

(ii) Colour of a grandparent as RED.

(iii) Change x = x's grandparent, repeat steps 2 and 3 for new x. **b) If x's uncle is BLACK**, then there can be four configurations for x, x's parent (**p**) and x's grandparent (**g**) (This is similar to AVL Tree)

(i) Left Left Case (p is left child of g and x is left child of p)

(ii) Left Right Case (p is left child of g and x is the right child of p)

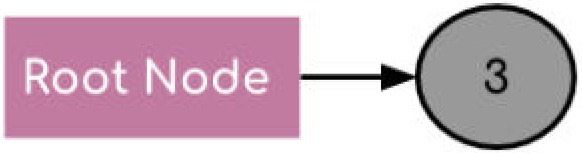
(iii) Right Right Case (Mirror of case i)

(iv) Right Left Case (Mirror of case ii)

Red Black Tree - Insertion

Create a red-black tree with elements 3, 21, 32 and 15.

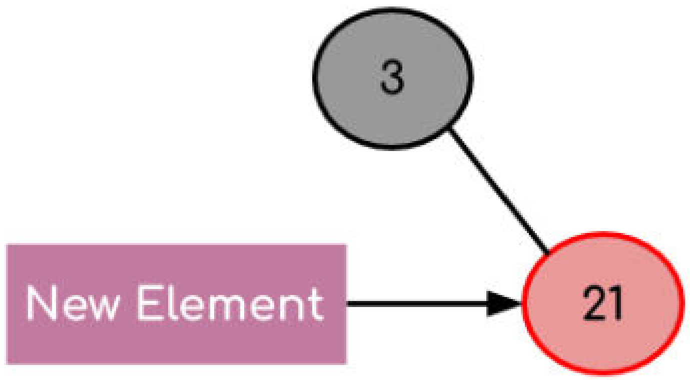
Step 1: Inserting element 3 inside the tree.



Red Black Tree - Insertion

Create a red-black tree with elements 3, 21, 32 and 15.

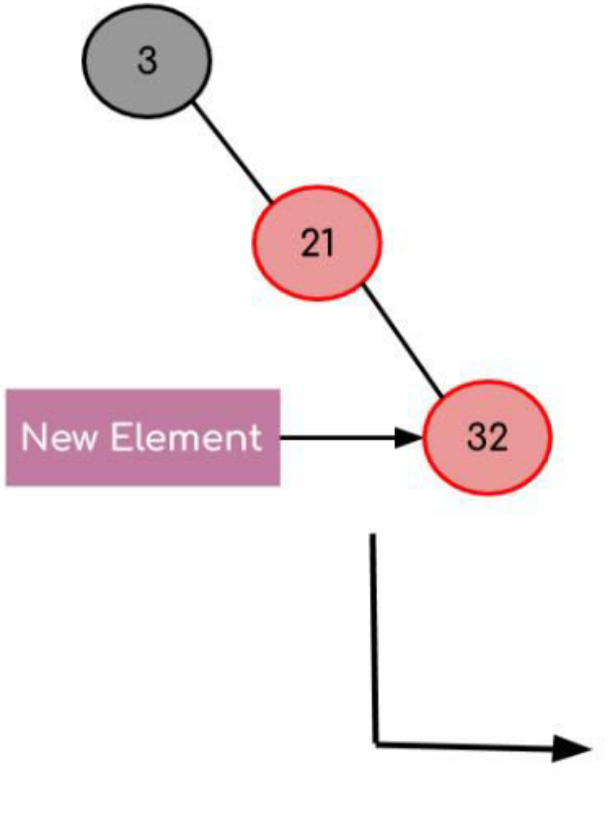
Step 2: Inserting element 21 inside the tree.



Red Black Tree - Insertion

Create a red-black tree with elements 3, 21, 32 and 15.

Step 3: Inserting element 32 inside the tree.

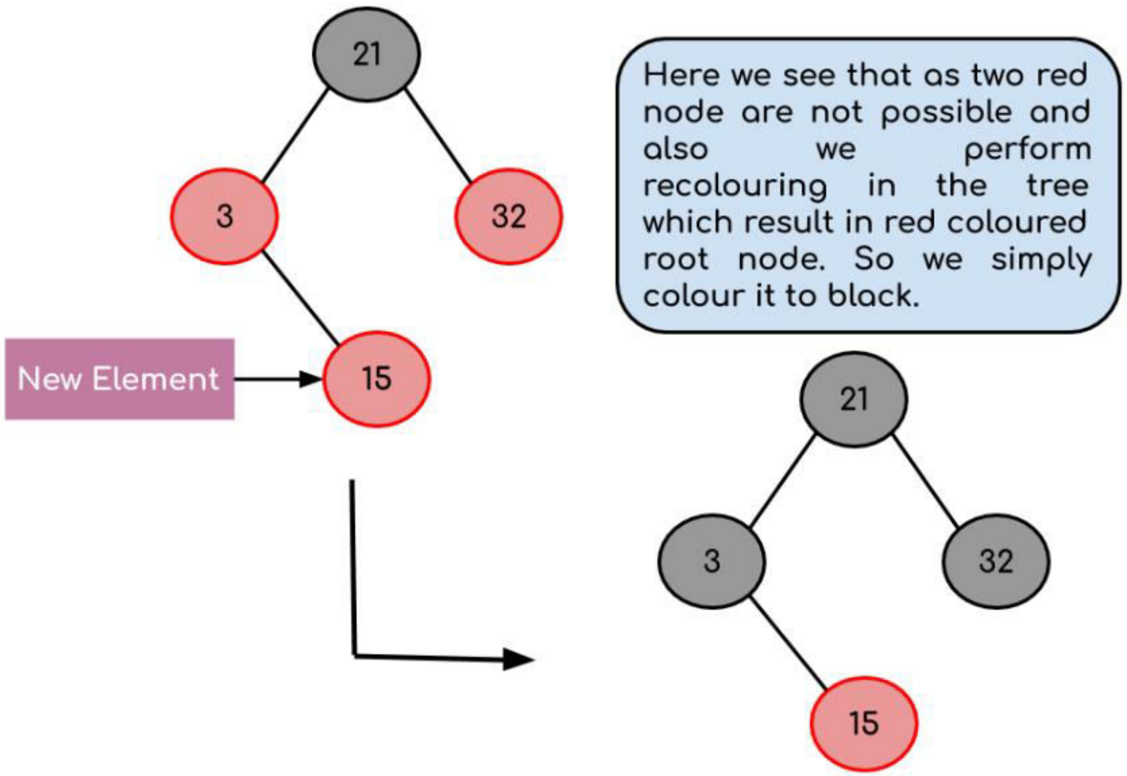


Here we see that as two red node are not possible and also we can see the conditions of RR rotation so it will follow RR rotation and recolouring to balance the tree.

Red Black Tree - Insertion

Create a red-black tree with elements 3, 21, 32 and 15.

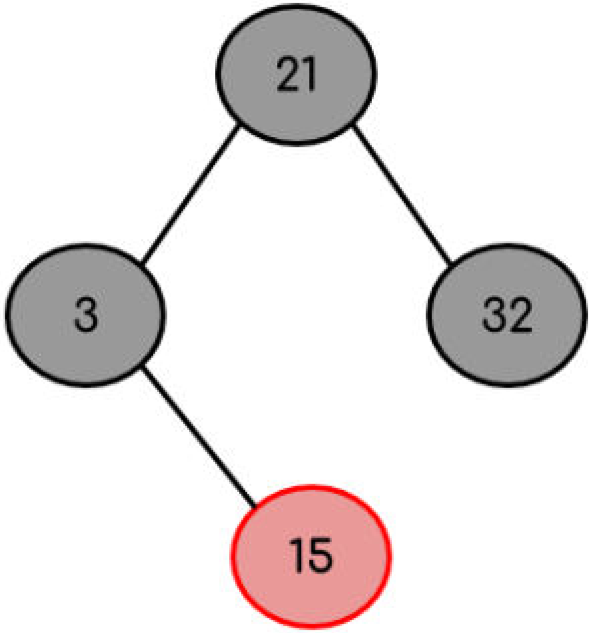
Step 4: Inserting element 17 inside the tree.



Here we see that as two red node are not possible and also we perform recolouring in the tree which result in red coloured root node. So we simply colour it to black.

Red Black Tree - Insertion

Create a red-black tree with elements 3, 21, 32 and 15.



PES
UNIVERSITY
ONLINE

THANK YOU

Surabhi Narayan
Department of Computer Science &Engineering

surabhinarayan@pes.edu