# Ch 13 RED-BLACK TREES

13.1

A Red-Black Tree is a Binary Search Tree with some additional properties.

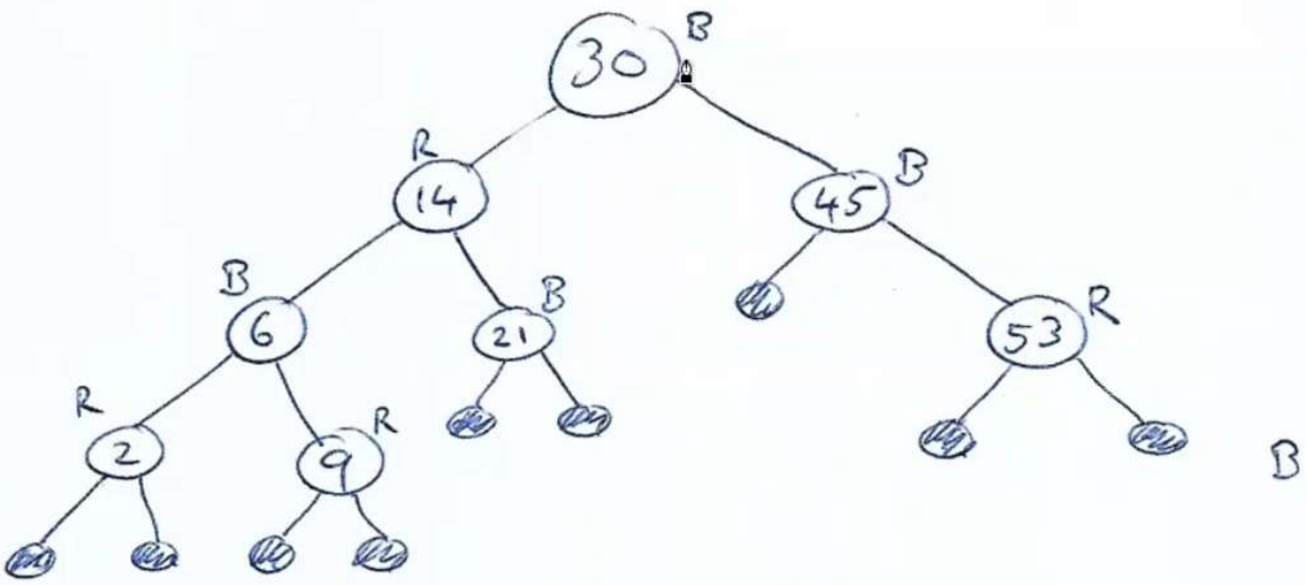
Every node has an additional attribute called colour, which is either Red or Black

```
Class RB-node
  int key
  RB-node left
  RB-node right
  RB-node p
  bool colour        (use Red / Black ).
```

A RED-BLACK TREE must satisfy :

1. Every node is either Red or Black
2. The root is Black
3. Every leaf node is Black (NIL leaves)
4. If a node is Red, then its children are Black

5. For each node, every simple path from the node to its descendent leaves contain the same number of Black nodes.


30
B
14
R
45
B
6
B
21
B
53
R
2
R
9
R
B

Introduce a special nil node:

RB-node nil

key = -1

left = NIL

right = NIL

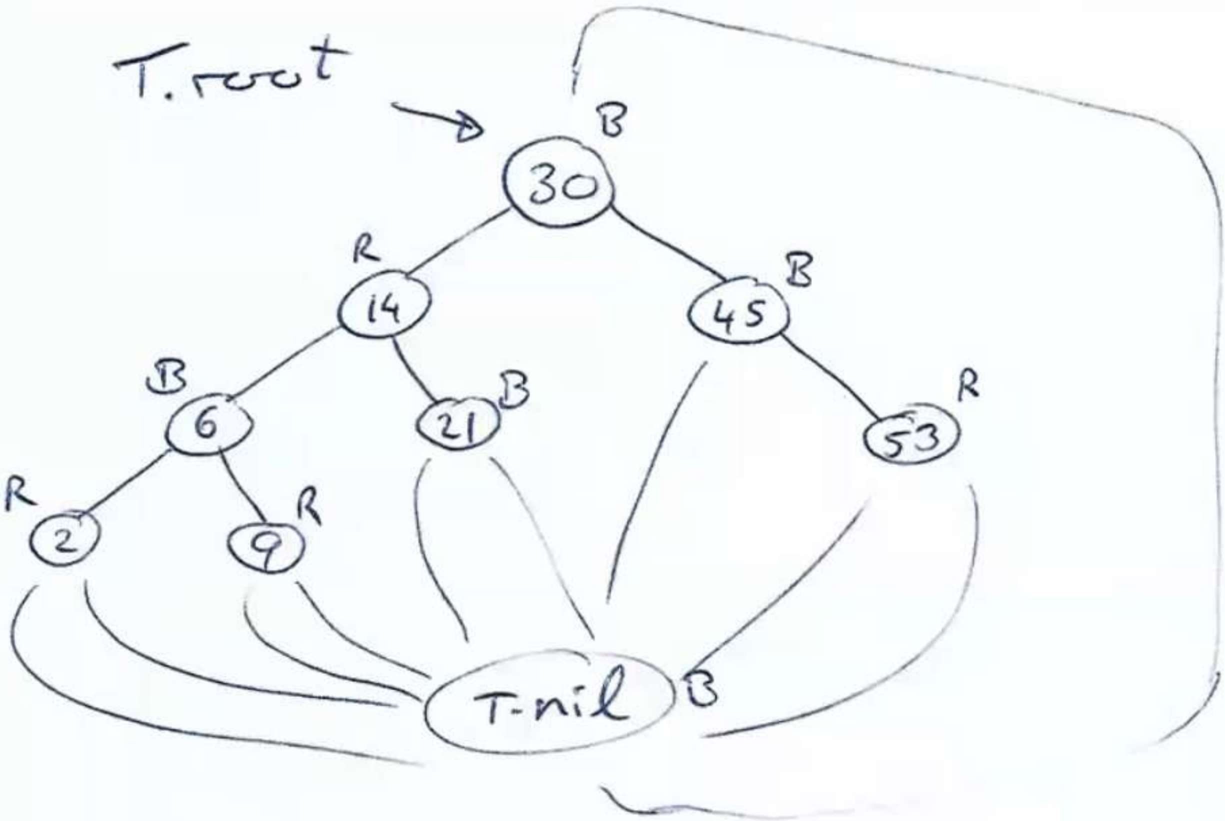
p = NIL

colour = Black

Class RB-Tree

RB-node root
RB-node nil

So a RB-tree T has a T.root
and T.nil



T.root
30
B
14
R
45
B
6
B
21
B
53
R
2
R
9
R
T.nil
B

note: T.root.p = T.nil

If a node $x$ has no left child,

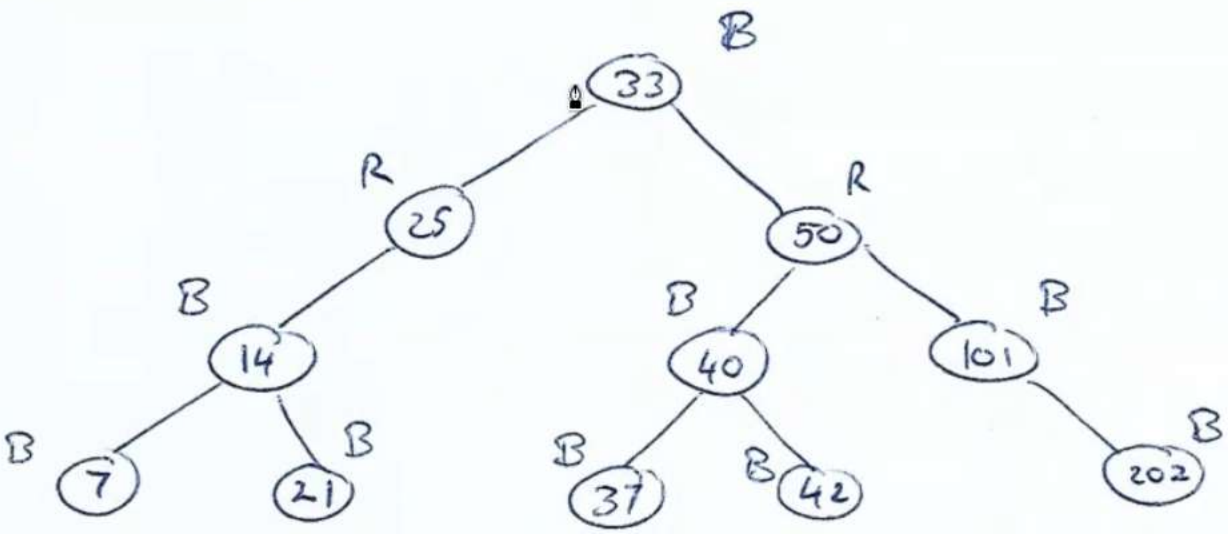
then $x$.left = T.nil

etc.

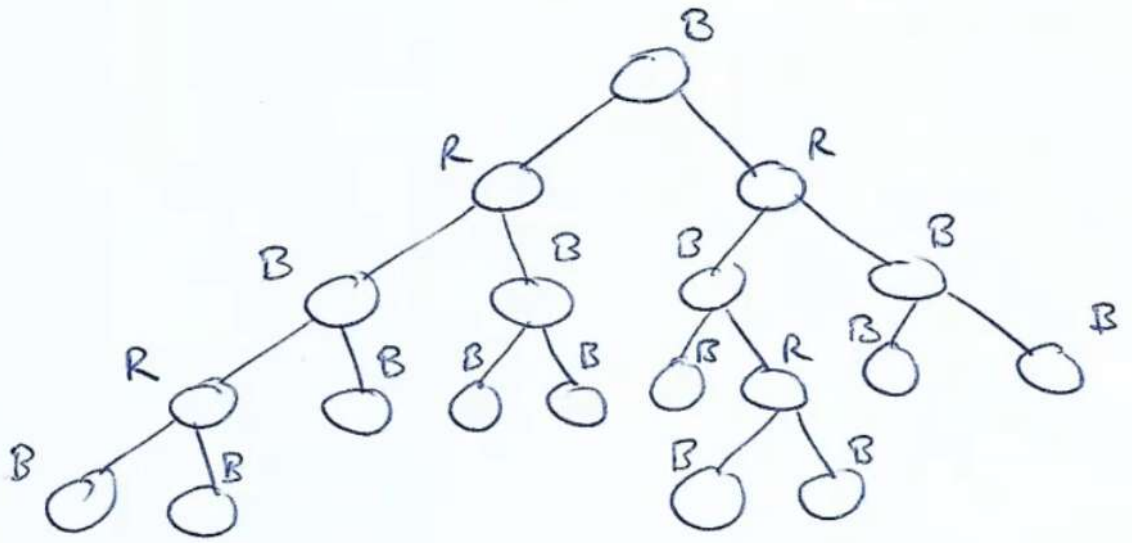
So we must always pass T ---

and replace NIL by T.nil in

most places in the BST algorithms.




not a RB-tree




RB-tree

Lemma 13.1

A RB-tree with $n$ internal nodes has height at most $2\log(n+1)$.

Proof: we first use induction to show:

Any node $x$ has a subtree containing at least $2^{bh(x)}-1$ internal nodes [bh(x) is the black-height of $x$ - it is the number of black nodes in any simple path from $x$ to a leaf - but not including $x$ itself.]

Base case: consider a leaf node.

Say $x$ = T.nil

then $bh(x) = 0$ so $2^{bh(x)} - 1 = 2^0 - 1 = 0$

which is correct since there are no internal nodes a T.nil.

Induction step: Consider any node $x$ and suppose the statement is true at $x.left$ and $x.right$.

Then # nodes in $x.left$'s subtree is

$$\geq 2^{bh(x.left)} - 1$$

# nodes in $x.right$'s subtree is

$$\geq 2^{bh(x.right)} - 1$$

∴ # nodes in $x$'s subtree is

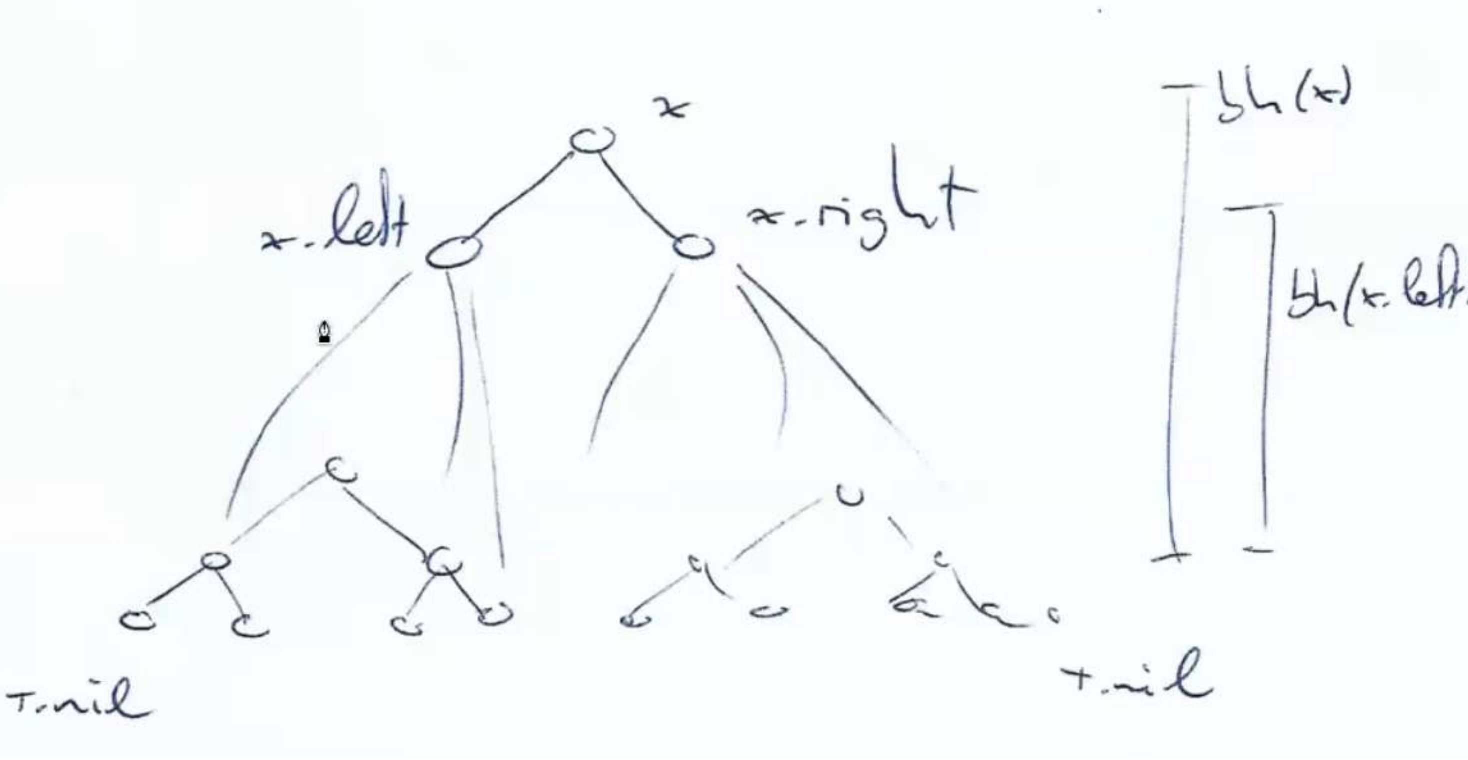
$$\geq 2^{bh(x.left)} - 1 + 2^{bh(x.right)} - 1 + 1$$

$$= 2^{bh(x.left)} + 2^{bh(x.right)} - 1$$

$$\geq 2^{bh(x)-1} + 2^{bh(x)-1} - 1$$

$$= \frac{1}{2} \cdot 2^{bh(x)} + \frac{1}{2} 2^{bh(x)} - 1$$

$$= \underline{\underline{2^{bh(x)} - 1}}$$


x
x.left
x.right
bh(x)
bh(x.left)
T.nil
T.nil

Thus the statement is true at every node in the tree, especially at the root:

$$\underbrace{\#\text{ nodes in subtree at root}}_{n} \text{ is } \geq 2^{bh(root)} - 1$$

$$\therefore \quad n \geq 2^{bh(root)} - 1$$

$$\therefore \quad n \geq 2^{\frac{h}{2}} - 1$$

$$\therefore \quad n+1 \geq 2^{\frac{h}{2}}$$

$$\therefore \quad \log(n+1) \geq \frac{h}{2}$$

$$\therefore \quad h \le 2\log(n+1)$$

Note: Since the height of a RB-tree is $O(\log n)$, the operations Search, Insert, Delete run in $O(\log n)$

However, we must maintain RB-tree properties when Inserting & Deleting