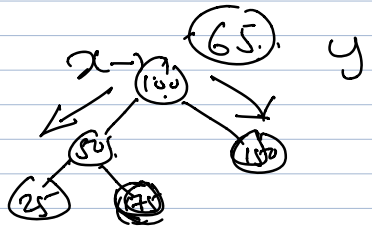
RB-INSERT(T, z)  T → RB tree, z → new node to be inserted in RB tree

```
y = T.nil          y → pointer to RB node.
x = T.root         x → p_rb_tree → p_root_node.

while x ≠ T.NIL    Until x becomes a T.nil.
    y = x
    if z.key < x.key
        x = x.left
    else
        x = x.right.
z.p = y.
if y == T.NIL.
    T.root = z
else if z.key < y.key
    y.left = z
else
    y.right = z
z.left = T.nil
z.right = T.nil
z.color = RED.   !!
```




x = A(100) y = nil.
y = A(100) x = A(50)
y = A(50) x = A(75)
y = A(75) x = T.NIL

→ z is at appropriate place as far as satisfying BST properties are concerned.

RB-Insert-Fixup(T, z).

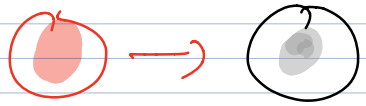
(1) Every node must be R or B.

(2) Root must be B.

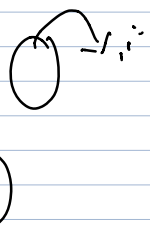
(3) Sentinel node : B.

(4) children of Red = B

(5) from any node, black heights of all paths from given node to sentinel leaf node must be identical.

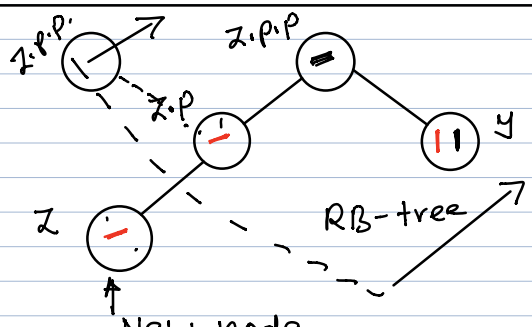


RB-INSERT-FIXUP(T,z)




```
while z.p.color == RED
    if z.p = z.p.p.left
        y = z.p.p.right
        if y.color == RED
            z.p.color = BLACK          .... 1
            y.color = BLACK.           .... 1
            z.p.p.color = RED          .... 1
            z = z.p.p.                 ... 1
        else if z == z.p.right
                z = z.p                .... 2
                LEFT-ROTATE(T,z)       .... 2
            z.p.color = BLACK          .... 3
            z.p.p.color = RED          .... 3
            RIGHT-ROTATE(T, z.p.p)     .... 3
    else (Symmetic)
```

---




y.color == BLACK

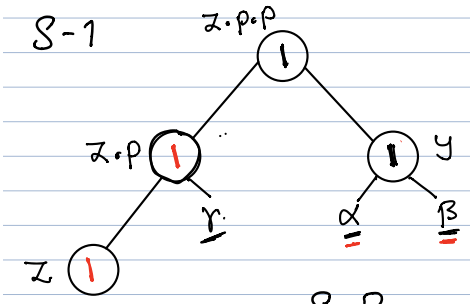
CASE-I: y.color == RED.

CASE-II: y.color == BLACK ∧ z == z.p.right

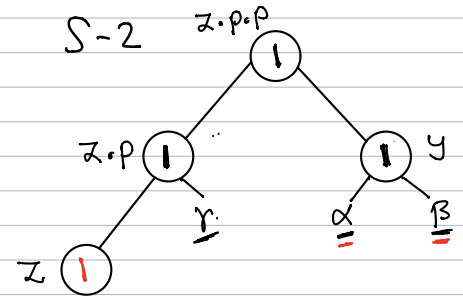
CASE-III: y.color == BLACK ∧ z == z.p.left

CASE-III

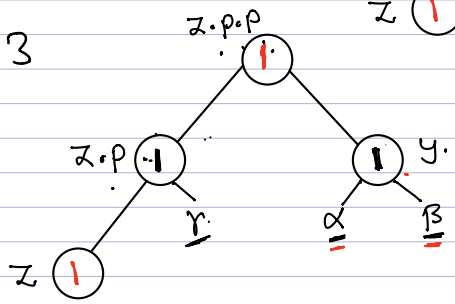
S-1

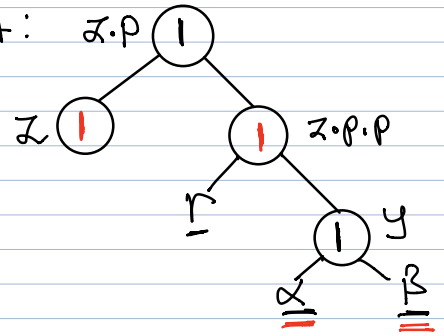



S-2

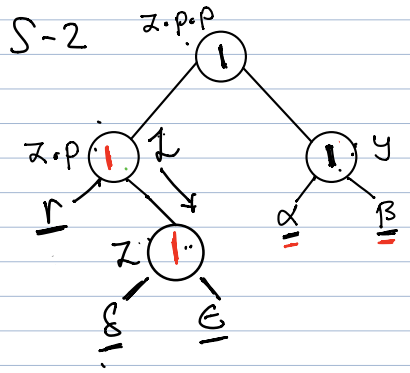


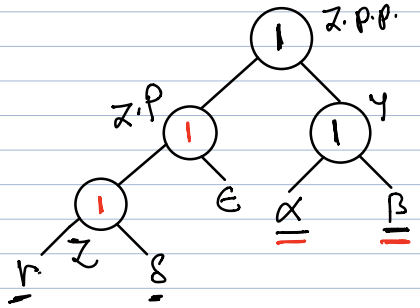

S-3



Step-4: 


CASE-II Analysis: $y.color \equiv BLACK$ and $z == z.p.right$

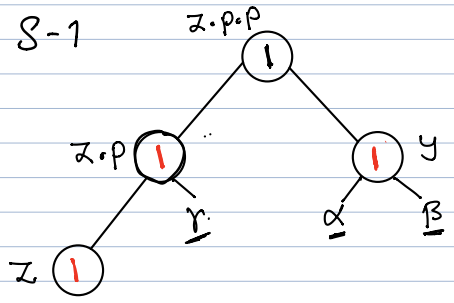
S-2 


$y.color == BLACK \wedge$

$z == z.p.left \equiv$ CASE-III Cond!

CASE-I: $y.color == RED$.

S-1 


S-1