13.3 INSERTION

RECALL THAT TREEINSERT(T, Z) INSERTS A NEW NODE Z INTO A TREE SO AS TO PRESERVE THE BST PROPERTIES.

WHICH OF THE RBT PROPERTIES MAY BE VIOLATED AFTER AN INSERTION?

1.) ALL NODES BLACK OR RED
2.) ALL NILS ARE BLACK
3.) ROOT IS BLACK
4.) RED NODES HAVE ONLY BLACK CHILDREN
5.) # OF BLACK NODES ON ANY TWO DOWNWARD PATHS FROM A NODE TO A DESCENDENT LEAF IS THE SAME.

(1) IS CERTAINLY TRUE, AND (2) HOLDS SINCE IT IS A CONVENTION ANYWAY. (3) IS ALSO TRUE.

WE CAN MAKE (5) TRUE BY COLORING THE NEWLY INSERTED NODE RED, SINCE THEN WE REPLACE A BLACK (nil) WITH A RED (Z) WHICH HAS TWO BLACK (nil) CHILDREN.

(4) MAY BE FALSE HOWEVER.

RB-INSERT (T, z) is a simple modification of Tree-Insert(T, z) which colors the new node red, then calls a subroutine RB-Insert-Fixup (T, z) to re-establish property (4) if necessary.

RB-Insert(T, z)   (Pre: key[z] has been set)

```
1.  y←nil
2.  x←root[T]
3.  while x ≠ nil
4.          y←x
5.          if key[z]<key[x]
6.                  x←left[x]
7.          else
8.                  x←right[x]
9.  p[z]←y
10. if y = nil
11.         root[T]←z
12. else if key[z]<key[y]
13.         left[y]←z
14. else
15.         right[y]←z
16. left[z]←nil
17. right[z]←nil
18. color[z]←red
19. RB-Insert-Fixup(T, z)
```

NOTE: lines 1-15 ARE JUST BST-INSERT.

```
RB-Insert-Fixup(T, z)
while color[p[z]] = red
        if p[z] = left[p[p[z]]]
                y←right[p[p[z]]]
                if color[y] = red
                        color[p[z]]←black       ) case 1
                        color[y]←black          )   "
                        color[p[p[z]]]←red      )   "
                        z←p[p[z]]               )   "
                else
                        if z = right[p[z]]
                                z←p[z]          ) case 2
                                LeftRotate(T, z) )  "
                        color[p[z]]←black       ) case 3
                        color[p[p[z]]]←red      )   "
                        RightRotate(T, p[p[z]]) )   "
        else
                y←left[p[p[z]]]
                if color[y] = red
                        color[p[z]]←black       ) case 4
                        color[y]←black          )   "
                        color[p[p[z]]]←red      )   "
                        z←p[p[z]]               )   "
                else
                        If z = left[p[z]]
                                z←p[z]          ) case 5
                                RightRotate(T, z) ) "
                        color[p[z]]←black       ) case 6
                        color[p[p[z]]]←red      )   "
                        LeftRotate(T, p[p[z]])  )   "
color[root[T]]←black
```

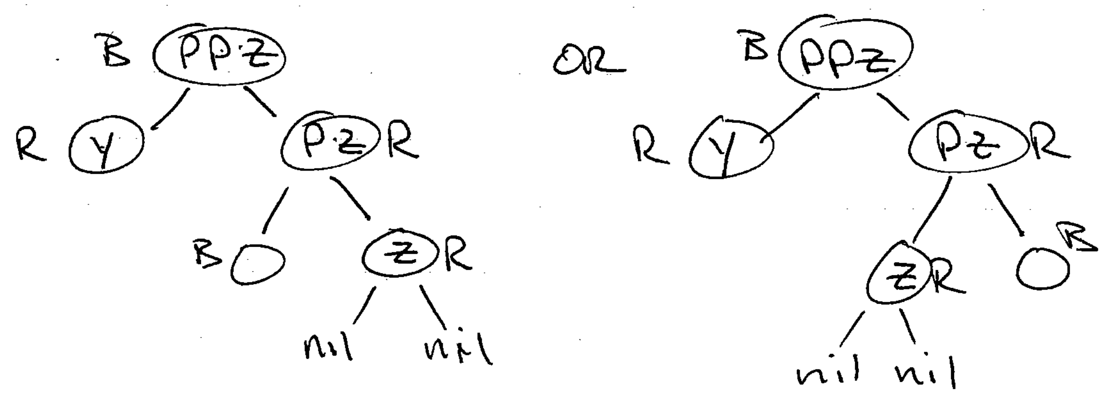
RB-Insert-Fixup(T, z):

IF color p[z] is RED THEN WE HAVE 2 REDS IN A ROW p[z] AND z, AND HENCE A VIOLATION OF RBT PROPERTY 4. WE BREAK THE PROBLEM INTO TWO ~~cases~~ CASES:

- p[z] is left child of its parent. Let y be p[z]'s right sibling, i.e. z's uncle. This case breaks further into 3 subcases {1,2,3}
- p[z] is right child of its parent. Then let y be p[z]'s left sibling, i.e. y is z's uncle. Again we have 3 subcases {4,5,6}.

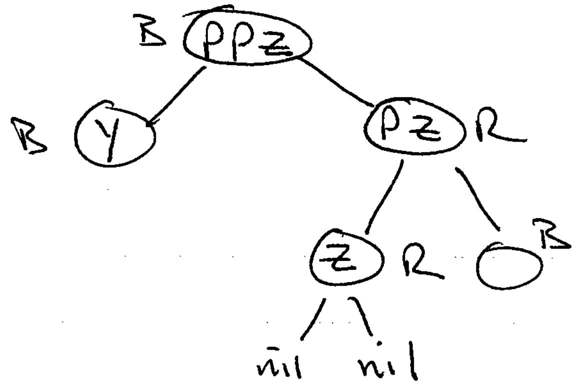
The book does {1,2,3} so we concentrate on {4,5,6}. So assume $p[z] = right[p[p[z]]]$ and set $y \leftarrow left[p[p[z]]]$.

Case 4: y is RED

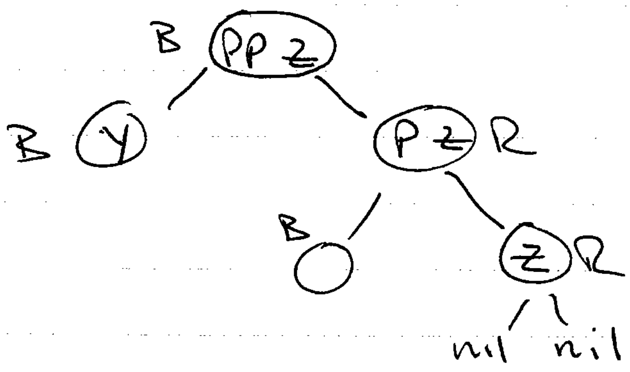


NOTE: IN ANY CASE P[P[Z]] MUST BE BLACK. OTHERWISE RBT PROP 4 WOULD HAVE BEEN FALSE BEFORE INSERTION OF Z.

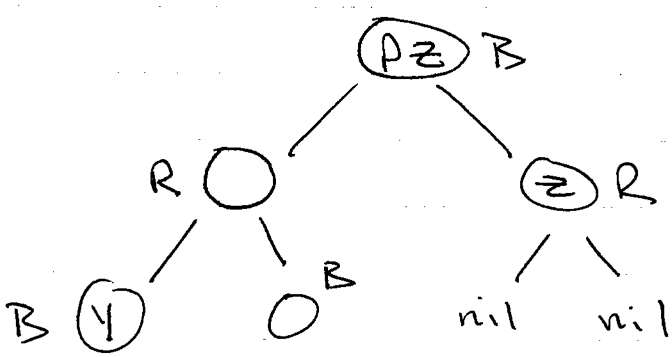
CASE 5: Y IS BLACK, Z IS IT'S PARENT'S LEFT CHILD.



CASE 6: Y is BLACK, Z is P[Z]'s right child

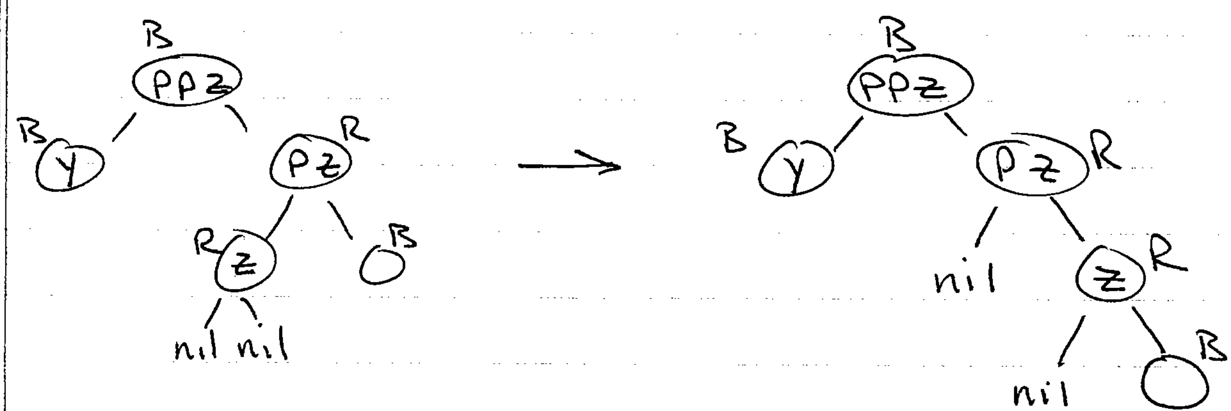


IN CASE 6 WE COLOR P[Z] BLACK, color P[P[Z]] RED, THEN LEFT ROTATE ABOUT P[P[Z]] TO GET:

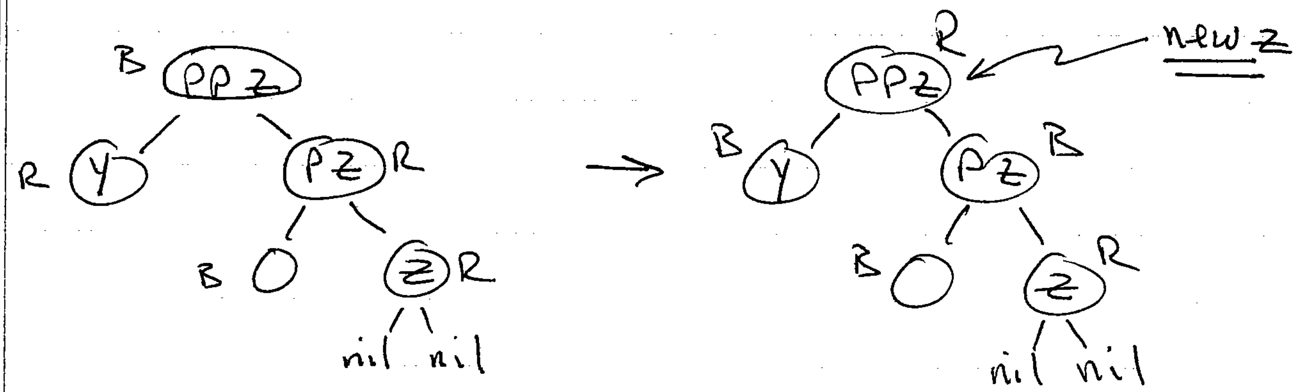


OBSERVE THAT PROPERTY 4 IS NOW SATISFIED, AND PROP. 5 HAS NOT BEEN VIOLATED. ALSO NOTE THAT THE (ABSOLUTE) HEIGHT OF THE TREE MAY BE LOWERED IN THIS STEP. THE while loop TERMINATES IF WE REACH THIS POINT.

IN CASE 5 WE SIMPLY CONVERT TO CASE 6. THIS IS DONE BY LETTING z 'climb up' TO P[z], THEN RIGHT ROTATE ABOUT z.



IN CASE 4 WE FIX THE COLOR RELATIONSHIPS AMONG y, P[z], P[P[z]], THEN LET THE VARIABLE z 'climb up' TO P[P[z]].



OBSERVE THAT RBT PROP. 5 IS PRESERVED, BUT PROP. 4 MAY BE VIOLATED SINCE BOTH z AND p[z] (FORMERLY p[p[z]] AND p[p[p[z]]]) MAY BE RED. IF SO WE ~~[illegible]~~ REPEAT THE BODY OF while LOOP.

NOTE IF p[z] = nil ~~[illegible]~~ (i.e. z is ROOT) THEN color[z] = BLACK AND THE LOOP WILL TERMINATE.