Lecture 6

Instructor: Karteek Sreenivasaiah

24th August 2018

Recap - Insert procedure

Insert(x) – Insert value x into the red-back tree.
High level strategy:

- Create a node X with value x and color red.
- Insert node X just like inserting into a Binary Search Tree.
- Call procedure FixInsert at node X.

Recap - Insert procedure

Only properties 2 and 4 could be broken after inserting a new red node:

1. All nodes are colored either Red or Black.
2. The root node is black.
3. The leaf nodes (NIL) are black.
4. Both children of a red node are black.
5. For any node, all paths from the node to the descendant leaves have the same number of black nodes.

Correctness of Insert

To show that Insert works as intended, it suffices to show:

- FixInsert fixes Property 2
- FixInsert fixes Property 4

FixInsert pseudocode

Algorithm 1 FixInsert called on node Z

```
while color(parent(Z)) = red do
    U ← Uncle(Z)
    if parent(Z) is the left child of the grandparent then
        if color(U) = red then
            Recolor parent, uncle and grandparent.
            Z ← grandparent(Z).
        else
            if Z is the right child then
                Z ← parent(Z); Left rotate at (Z)
            end if
            Recolor parent and grandparent.
            Right rotate at grandparent(Z).
        end if
    end if
end while
color(root)← black.
```

Proof Strategy

The proof strategy that we use is that of a loop invariant.

1. Formulate a *loop invariant*.
2. Show that the invariant:
   - 2.1 Holds before the first iteration (*Initialization*)
   - 2.2 Holds during consecutive iterations. (*Maintenance*)
   - 2.3 Holds at the end of the last iteration. (*Termination*)
3. Conclude that the algorithm is correct using loop invariant and rest of the pseudocode.

Loop Invariant - Formulation

Consider FixInsert called on a node Z.
We show that the following *invariant* holds for the loop:

1. Node Z is colored red.
2. If parent(Z) is root, then parent(Z) is black.
3. There is at most one violation and it is of Property 2 or 4. Further:
   - If Property 2 is violated, then Z itself is root.
   - If Property 4 is violated, then Z and its parent are red.

Loop Invariant - Initialization

1. Node Z is colored red.
2. If parent(Z) is root, then parent(Z) is black.
3. At most one violation. Further:
   - If Property 2 is violated, then Z itself is root.
   - If Property 4 is violated, then Z and its parent are red.

When FixInsert is called:

- FixInsert is called on a newly added node.
- The newly added node is colored red by the Insert procedure.

Loop Invariant - Initialization

1. Node Z is colored red.
2. If parent(Z) is root, then parent(Z) is black.
3. At most one violation. Further:
    - If Property 2 is violated, then Z itself is root.
    - If Property 4 is violated, then Z and its parent are red.

When FixInsert is called:

- If Z is not root, then the root was already black.
- If parent(Z) is the root, then it is black (trivially).

Loop Invariant - Initialization

1. Node Z is colored red.
2. If parent(Z) is root, then parent(Z) is black.
3. At most one violation. Further:
   - If Property 2 is violated, then Z itself is root.
   - If Property 4 is violated, then Z and its parent are red.

When FixInsert is called:

- We inserted Z into a red-black tree.
- Node Z has both children as the black colored sentinel node NIL.
- If Property 4 was violated, it must be because of Z and its parent.

Loop Invariant - Termination

1. Node Z is colored red.
2. If parent(Z) is root, then parent(Z) is black.
3. At most one violation. Further:
   - If Property 2 is violated, then Z itself is root.
   - If Property 4 is violated, then Z and its parent are red.

When the loop terminates:

- Invariant conditions 1 and 2 are trivial.

Loop Invariant - Termination

1. Node Z is colored red.
2. If parent(Z) is root, then parent(Z) is black.
3. At most one violation. Further:
   - If Property 2 is violated, then Z itself is root.
   - If Property 4 is violated, then Z and its parent are red.

When the loop terminates:

- If Z is the root, then property 2 might be violated.
- Parent of Z is colored black. So property 4 is not violated.
- The last line in the pseudocode solves violation of property 2

Loop Invariant - Termination

1. Node Z is colored red.
2. If parent(Z) is root, then parent(Z) is black.
3. At most one violation. Further:
   - If Property 2 is violated, then Z itself is root.
   - If Property 4 is violated, then Z and its parent are red.

When the loop terminates:

- If Z is the root, then property 2 might be violated.
- Parent of Z is colored black. So property 4 is not violated.
- The last line in the pseudocode solves violation of property 2

This lets us conclude that at the end of the InsertFix algorithm, all properties are satisfied.

Loop Invariant - Maintenance

To argue that the loop invariant holds for every iteration of the loop, we need to study six cases.
Modulo symmetry, we only look at 3 cases:

- Case 1: Recolor parent, uncle, grand parent. Reassign Z to grandparent.
- Case 3: Recolor parent, grandparent. Rotate grandparent.
- Case 2: Reassign Z to parent. Rotate parent.

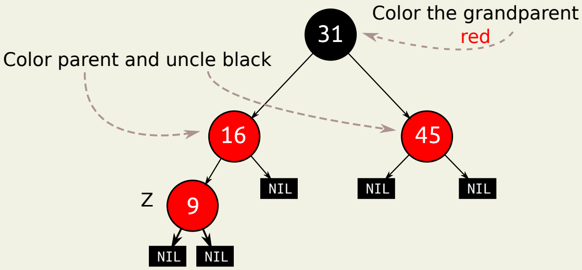
FixInsert pseudocode

Algorithm 2 FixInsert called on node Z

```
while color(parent(Z)) = red do
    U ← Uncle(Z)
    if parent(Z) is the left child of the grandparent then
        if color(U) = red then
            Recolor parent, uncle and grandparent.
            Z ← grandparent(Z).
        else
            if Z is the right child then
                Z ← parent(Z); Left rotate at (Z)
            end if
            Recolor parent and grandparent.
            Right rotate at grandparent(Z).
        end if
    end if
end while
color(root)← black.
```

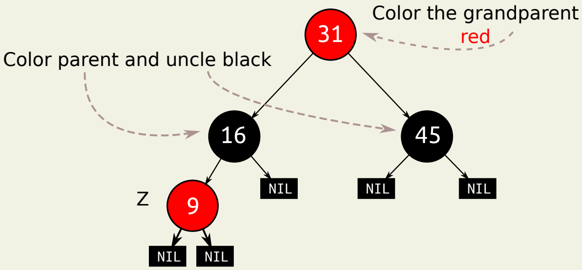
Case 1

1. Node Z is colored red.
2. If parent(Z) is root, then parent(Z) is black.
3. At most one violation. Further:
   - If Property 2 is violated, then Z itself is root.
   - If Property 4 is violated, then Z and its parent are red.



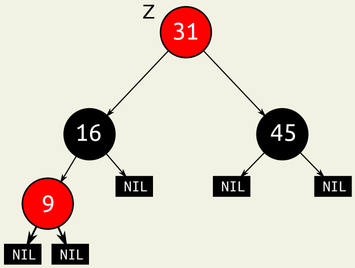
Case 1

1. Node Z is colored red.
2. If parent(Z) is root, then parent(Z) is black.
3. At most one violation. Further:
   - If Property 2 is violated, then Z itself is root.
   - If Property 4 is violated, then Z and its parent are red.



Case 1

1. Node Z is colored red.
2. If parent(Z) is root, then parent(Z) is black.
3. At most one violation. Further:
    - If Property 2 is violated, then Z itself is root.
    - If Property 4 is violated, then Z and its parent are red.



Case 1

1. Node Z is colored red.
2. If parent(Z) is root, then parent(Z) is black.
3. At most one violation. Further:
   - If Property 2 is violated, then Z itself is root.
   - If Property 4 is violated, then Z and its parent are red.

- If parent(Z) is the root, it was black already.
- We never changed its color.

Case 1

1. Node Z is colored red.
2. If parent(Z) is root, then parent(Z) is black.
3. At most one violation. Further:
   - If Property 2 is violated, then Z itself is root.
   - If Property 4 is violated, then Z and its parent are red.

- If property 2 is voilated now, it must be the case that the recoloring caused it. Hence Z points correctly to the problematic node and is the root.
- We resolved the local property 4 violation. If a new violation of property 4 happens, then it must be because of the new Z and its parent.

Case 3 and Case 2

Loop invariant:

1. Node Z is colored red.
2. If parent(Z) is root, then parent(Z) is black.
3. There is at most one violation and it is of Property 2 or 4. Further:
   - If Property 2 is violated, then Z itself is root.
   - If Property 4 is violated, then Z and its parent are red.

Look at previous lecture slides.

Exercise

Exercise

Insert elements $1, 2, 3, 4, 5, 6$ in ascending order into a red-black tree.