Exam Conflicts: Check your course and exam schedule for Midterm and Final and report any conflicts via the Exam Conflicts Form.

WACM Explains... Linux - Intermediate: Monday 4/11 5:30-7:00pm in CS1240

Week 4

ASSIGNMENTS
x2 available soon™
p2 available soon™
h3 available soon™ and due before 10pm on Monday 2/18
Peer Mentors: will help students practice Git and GitHub commands

Module: Week 4 (start on week 5 before next week)

THIS WEEK

- AVL Summary (from Week 3 outline)
- **Red-Black Tree**
  - **insert**
  - **lookup**
  - **delete**
- **Git and GitHub (x2)**
  - version control
  - centralized and decentralized

NEXT WEEK

- **B-Tree**
  - **2-3 Tree**
  - **2-3-4 Tree**
  - **B+ Tree**
- **x2 due next week**

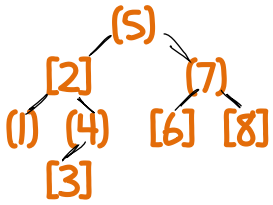
Red-Black Trees (RBT)

RBT:

A BST that stays balanced — general

Example:

Black node ( ) circle
Red Node [ ] square
(draw a red black tree)



Red-Black Tree Properties

root property
The root is always a Black Node

red property
Red nodes always have black children
Black Nodes do not always have red children

black property
For every Node that is not nil, the # of black nodes on the path from the root to that Node is the same

Red-Black Tree Operations

print O(N)

lookup
insert
delete
} O(log N)

Inserting into a Red-Black Tree

Goal: Use BST insert, check 3 properties, fix

If T is Empty

insert @ root
Set Node color to black

If T is Non-Empty

Perform the insert method as a BST
(super.insert(c))
- Set Node to red,
- Check 3 properties, fix

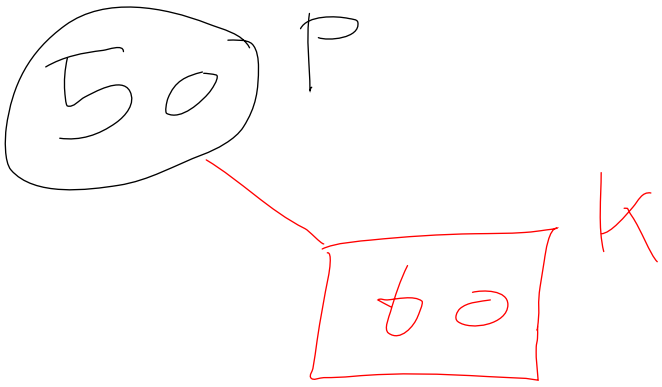
Which of the properties might be violated as a result of inserting a red leaf node?

`root property` Tree is non-empty... this does not change the root

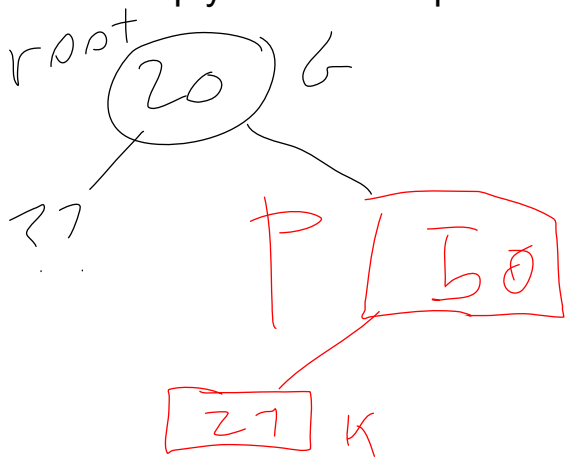
`black property` inserting a red leaf, No worries

`red property` this one we need to check & fix

Non-Empty Case 1: K's parent P is black



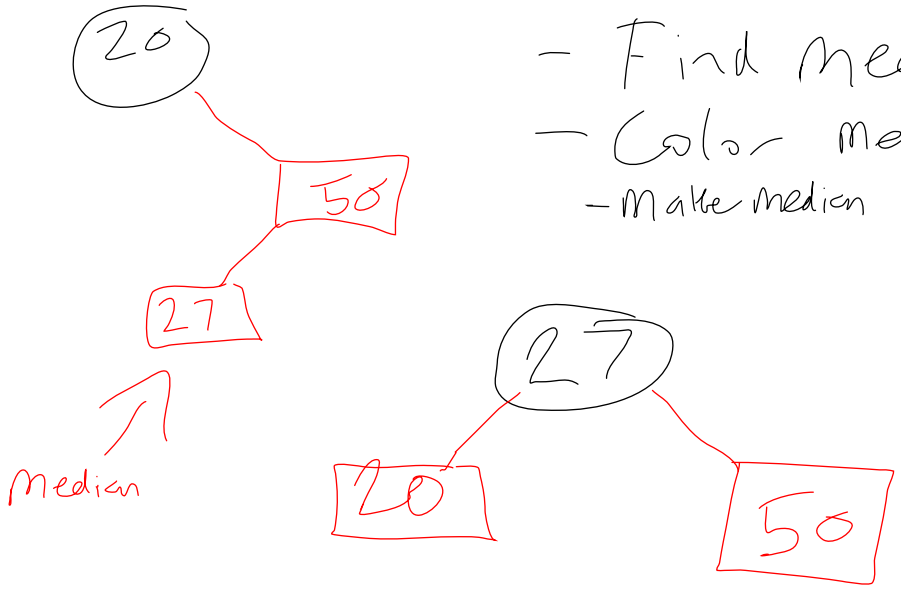
Non-Empty Case 2: K's parent P is red



fix depends on p's sibling

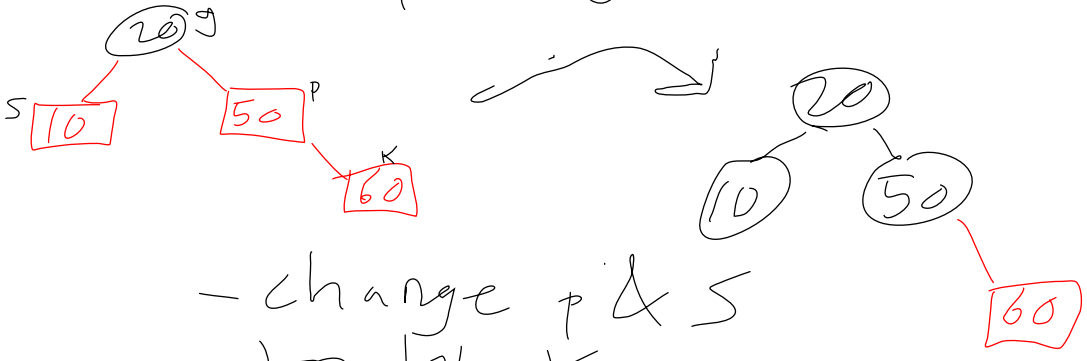
Fixing an RBT

Tri-Node Restructuring if p's sibling is Black or Null



- Find median
- Color median black
- Make median the parent of other two

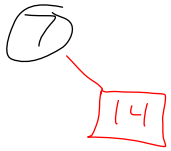
Recoloring is done if p's sibling is red



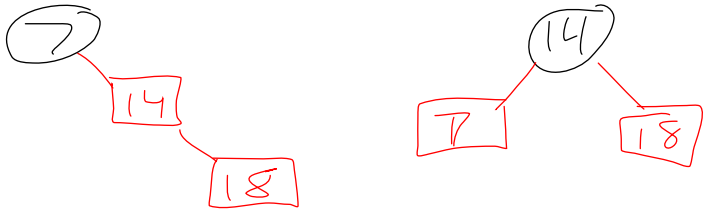
- change p & s to black
- change g to red unless it's the root
- Check g's parent for red violation through recursion

RBT Insert Practice I

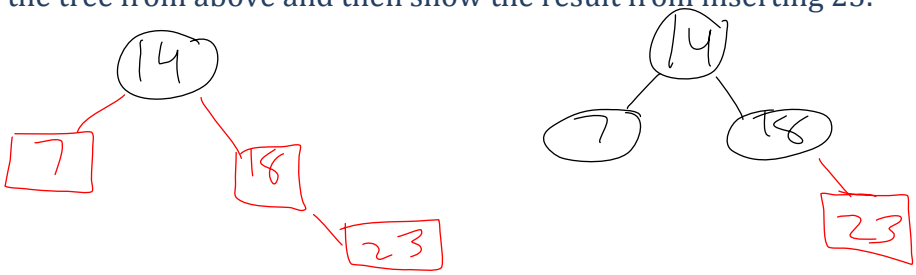
1. Start with an empty RBT, show the RBT that results from inserting 7 and 14.



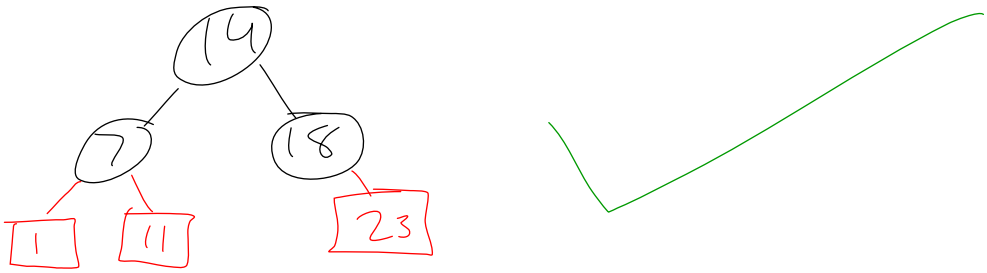
2. Redraw the tree from above and then show the result from inserting 18.



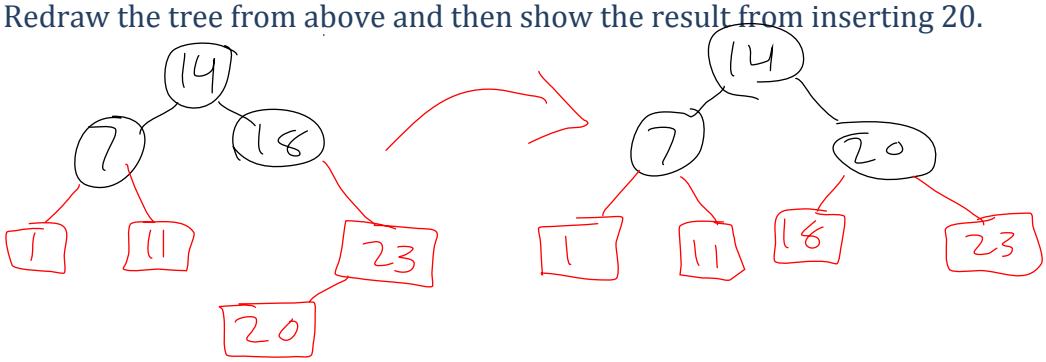
3. Redraw the tree from above and then show the result from inserting 23.



4. Redraw the tree from above and then show the result from inserting 1 and 11.

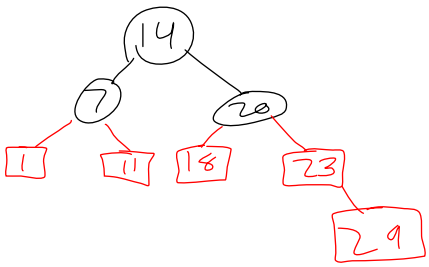


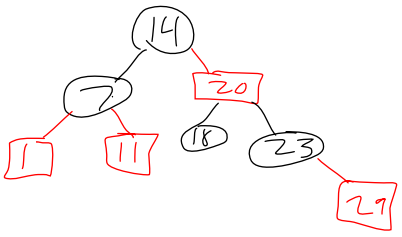
5. Redraw the tree from above and then show the result from inserting 20.



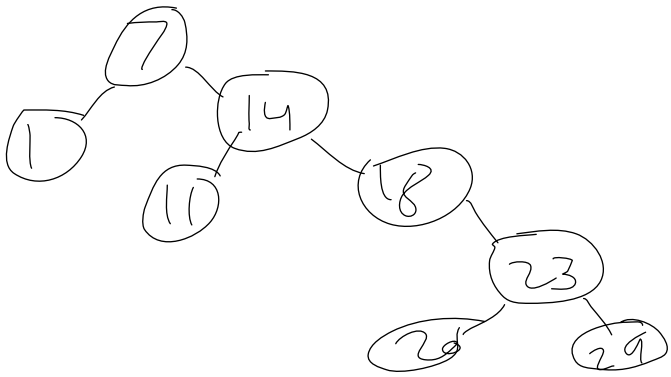
RBT Insert Practice II

6. Redraw the tree from the previous page and then show the result from inserting 29.





7. Insert the same list of values into an empty BST: 7, 14, 18, 23, 1, 11, 20, 29

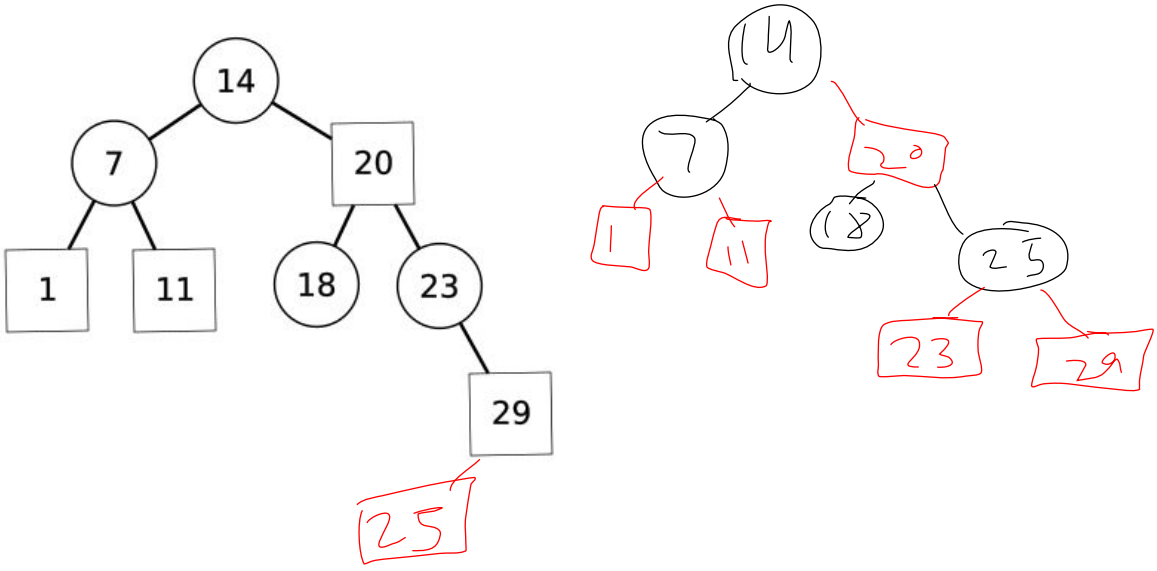


What does this demonstrate about the differences between a BST and RBT?

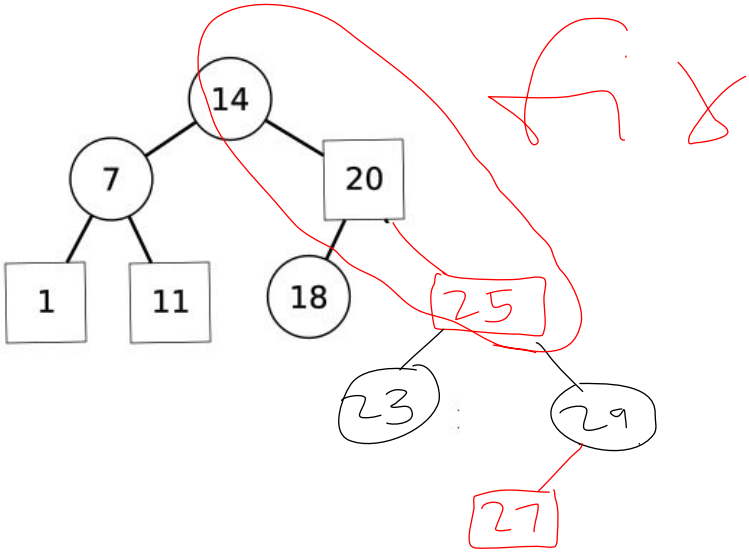
A red-black tree is balanced & achieves $\log_2 N$ whereas a BST can be linear

RBT Practice III

8. Show the result from inserting 25 in the RBT below.



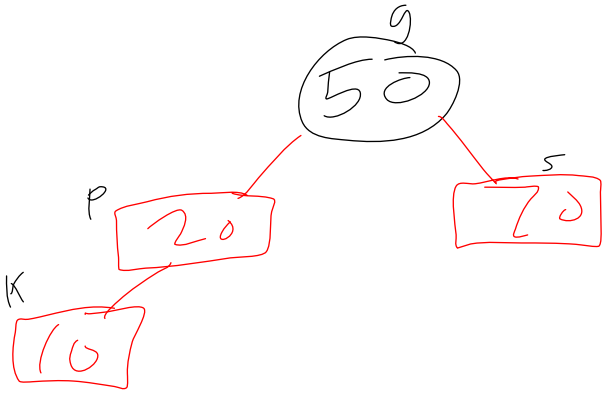
9. Redraw the tree from above and then show the result from inserting 27.



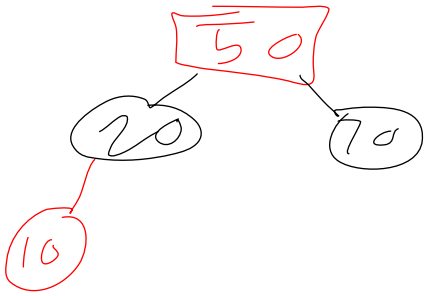
Cascading Fixes

Fixing an RBT UPDATED!

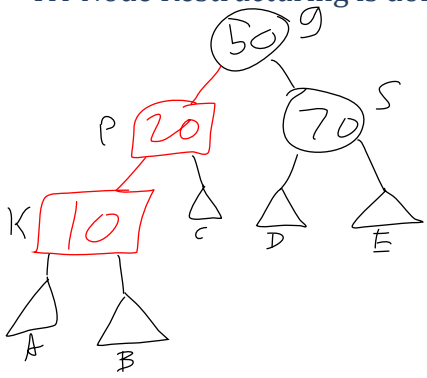
Recoloring is done if P's sibling S is red




```
g.isRed=true
P.isRed=false;
S.isRed=false;
if(g == root)
   g.isRed=false;
```

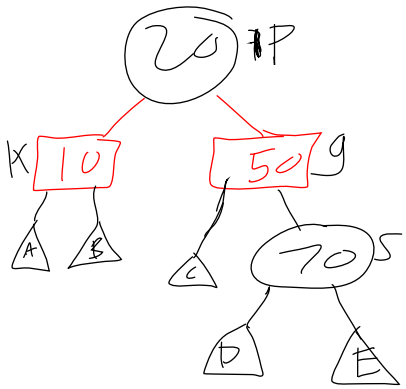


Tri-Node Restructuring is done if P's sibling S null or Black




```
P.isRed=false;
K.isRed=true;
g.isRed=true;
g.left=P.right
P.right=g
return P;
```

P is the median hand to g



K is the median

Return to previous page and cascade the fixes.

$height \leq 2 \cdot \log_2 N$

RBT Complexity

print $O(N)$

lookup $O(height) = O(2 \cdot \log_2 N) = O(\log_2 N)$

insert look-up + Make new node + fix + cascade up

$O(\log_2 N) + O(1) + O(1) + O(\log_2 N)$

$= O(\log_2 N)$

delete

$O(\log_2 N)$

RBT Delete Practice

Delete as from BST and then fix RBT properties

Visualize inserts and deletes at:
https://www.cs.usfca.edu/~galles/visualization/RedBlack.html

Insert 9, 8, 5, 4, 3, 2, 6, 1, 7 → See Next page

Practice deleting

- leaf nodes
- red interior
- black interior

Git and GitHub

git commands

- clone
- status
- log
- init
- config
- add
- commit
- push
- pull

GitHub

1. Create account with wisc.edu
2. Install Student Pack (unlimited free private repositories)
3. Create a repository
4. clone it it to your your CS account
5. config
6. add/edit a file
7. add
8. commit
9. push to GitHub repository
10. add a collaborator (for working in teams)