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 Git, repo, team

x2 available soon

p2 available ~~soon~~ due before 10pm 2/21

h3 available soon and due before 10pm on Monday 2/18

Peer Mentors: will help students practice ~~Git and GitHub~~ commands Set up P2 and BST & AVL tree

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

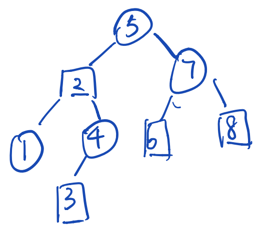
- **B-Tree**
  - **2-3 Tree**
  - **2-3-4 Tree**
  - **B+ Tree**
- **x2 due next week**

Red-Black Trees (RBT)

RBT:

A BST that stays balanced

Example:



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

(5)
[2] (7)
(1) (4) [6] [8]
[3]

Red-Black Tree Properties

root property the root is black

red property red nodes must (or no children) have black children

black property every path from root null child must have the same number of black trees

Red-Black Tree Operations

print
lookup
same as BST

insert

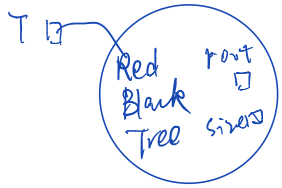
delete

H O(log n)

Inserting into a Red-Black Tree

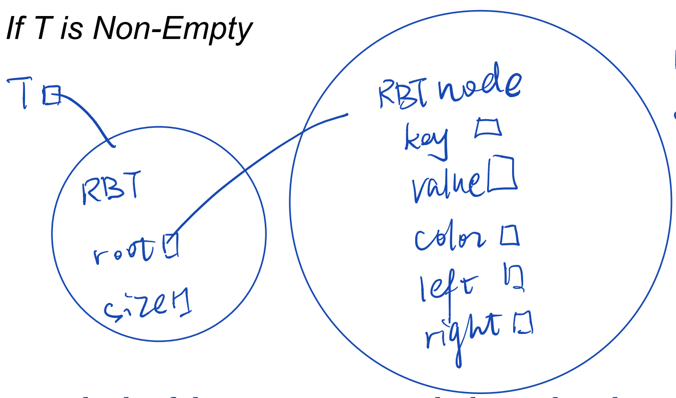
Goal: maintain balance when inserting & deleting

If T is Empty



To insert into an empty tree.
1. add a new RBT node root = newnode
2. color it black

If T is Non-Empty



1. step down as BST
2. Add a new leaf node
3. color it red
4. rebalancing

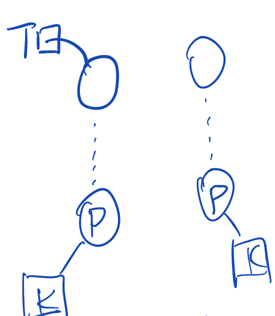
Which of the properties might be violated as a result of inserting a red leaf node?

root property a new leaf node is red, won't affect root property

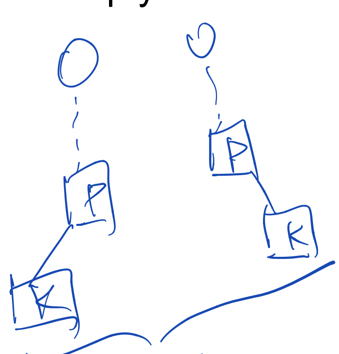
black property . . . - - - - - - - - - - - - - - black . . . .

red property may affect red property
If New node's parent is red, must fix (R.P.V) ←violation

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

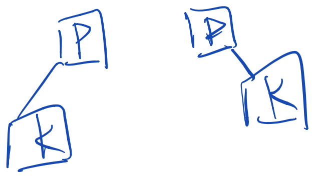


No RPV.
no need fix an insert



RPV
need fix

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

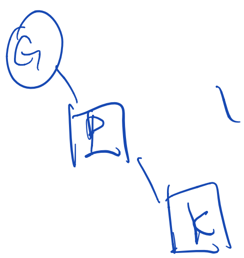


RPV,

Fixing an RBT

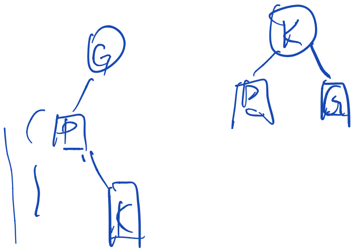
(T-N-R)

Tri-Node Restructuring if P's sibling is null

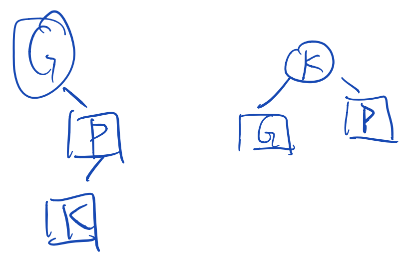


PPV, S is null
TNR

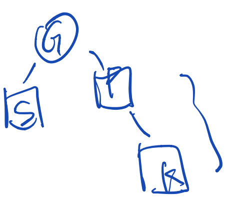
⇒



RPV, S is null
TN



Recoloring is done if P's sibling is red

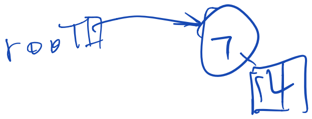


Recolor
(R?)

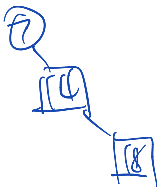
1. Set G -> red
   S, T, black
2. leave K as red
3. set root to black

RBT Insert Practice I

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

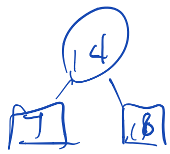


2. Redraw the tree from above and then show the result from inserting 18.



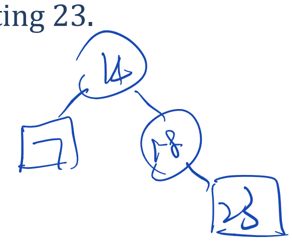
RPV S sibling is null

TNR

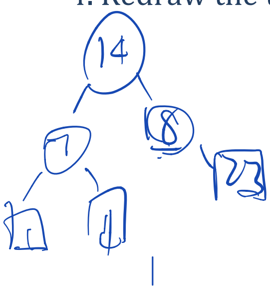


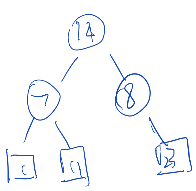
3. Redraw the tree from above and then show the result from inserting 23.

RPV S is red, R-C



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

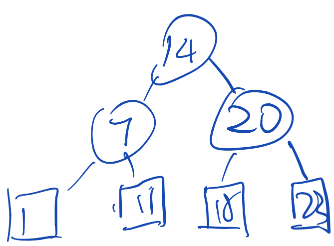




5. Redraw the tree from above and then show the result from inserting 20.

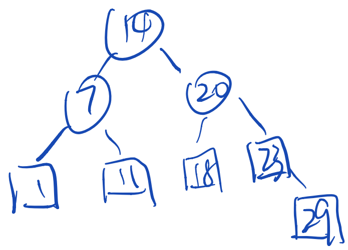
RPV S is null

TNR

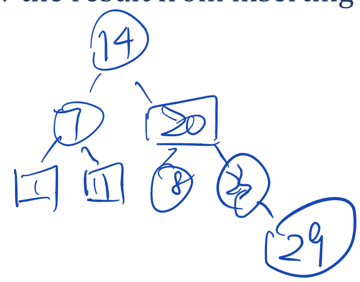


RBT Insert Practice II

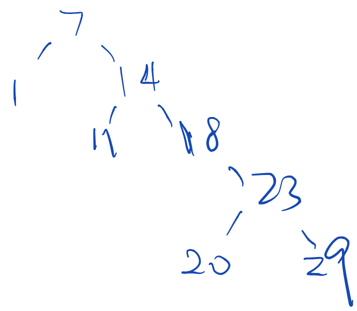
6. Redraw the tree from the previous page and then show the result from inserting 29.



} RPV Sibling is red
R-C.



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?

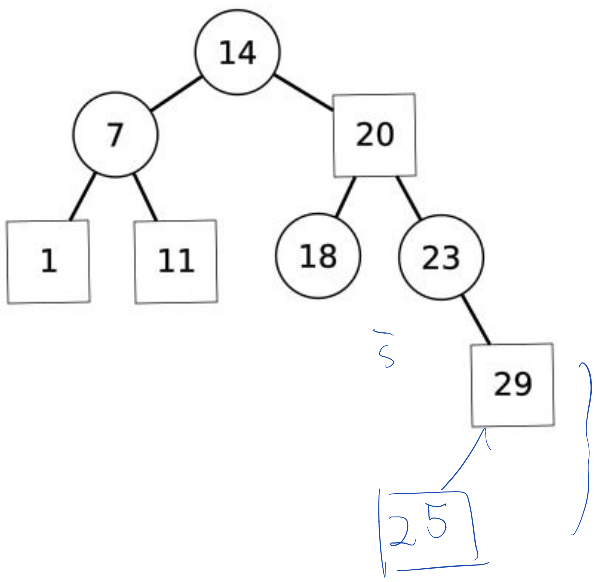
maintain the balance

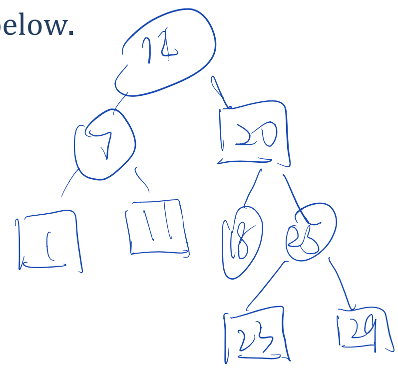
grow at. H $O(\log_2 N)$

$2\log_2 N$?

RBT Practice III

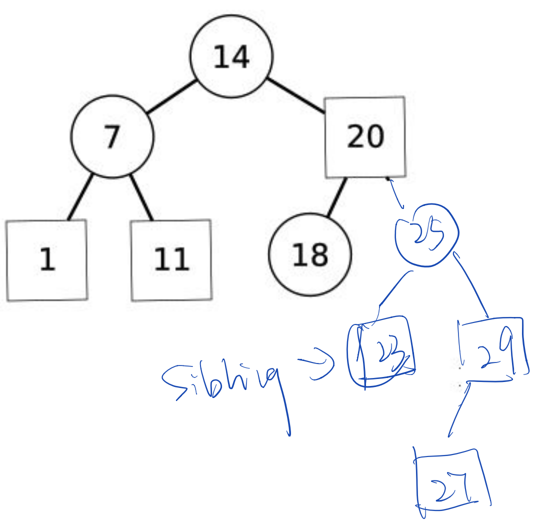
8. Show the result from inserting 25 in the RBT below.



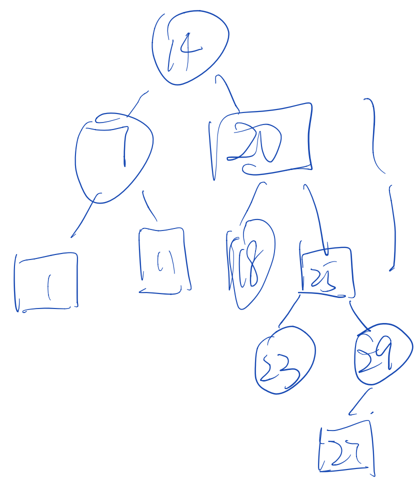


P
RPV, S is null

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



RPV, sibling is red.



RPV

Cascading Fixes

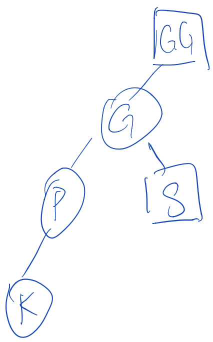
Fixing an RBT UPDATED!

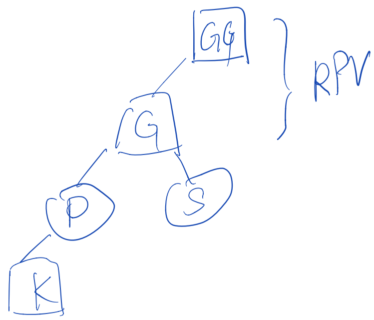
parent of new node

Recoloring is done if P's sibling S is red

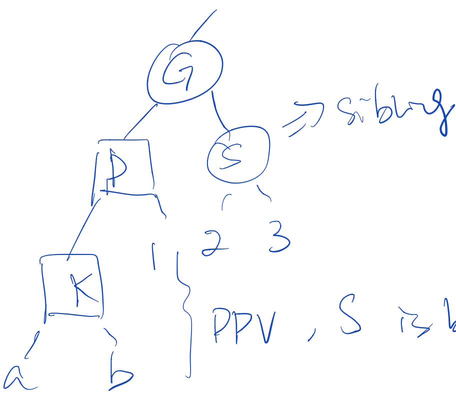
recoloring

R-C may cause "casade"
Fix black up the tree.

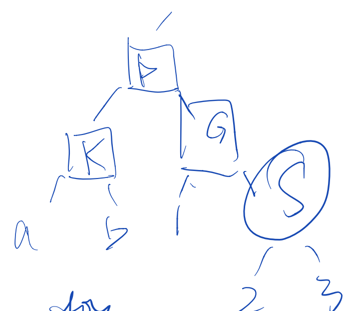




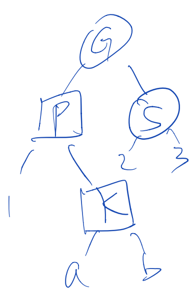
Tri-Node Restructuring is done if P's sibling S null or Black



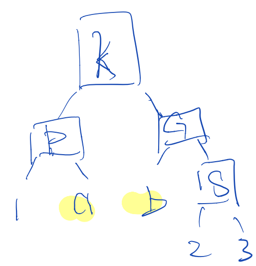
PPV, S is black, try node restructor



G.left = P.right
P.right = G
return P.



PPV, S is black, tNR



P.right = K.left
K.left = P
G.left = K.right
K.right = G
return K.

Return to previous page and cascade the fixes.

RBT Complexity

print $O(N)$ visit each node

lookup $O(H) = O(\log_2 N)$ H-B ? X NO !!!!!

$H = 2\log(N)$ bounded

insert $O(\log_2 N)$ lookup + linked node + detect + cascade.

$(\log_2 N) + O(1) + O(1000) + O(\log_2 N)$

$= O(\log_2 N)$

delete $O(\log_2 N)$

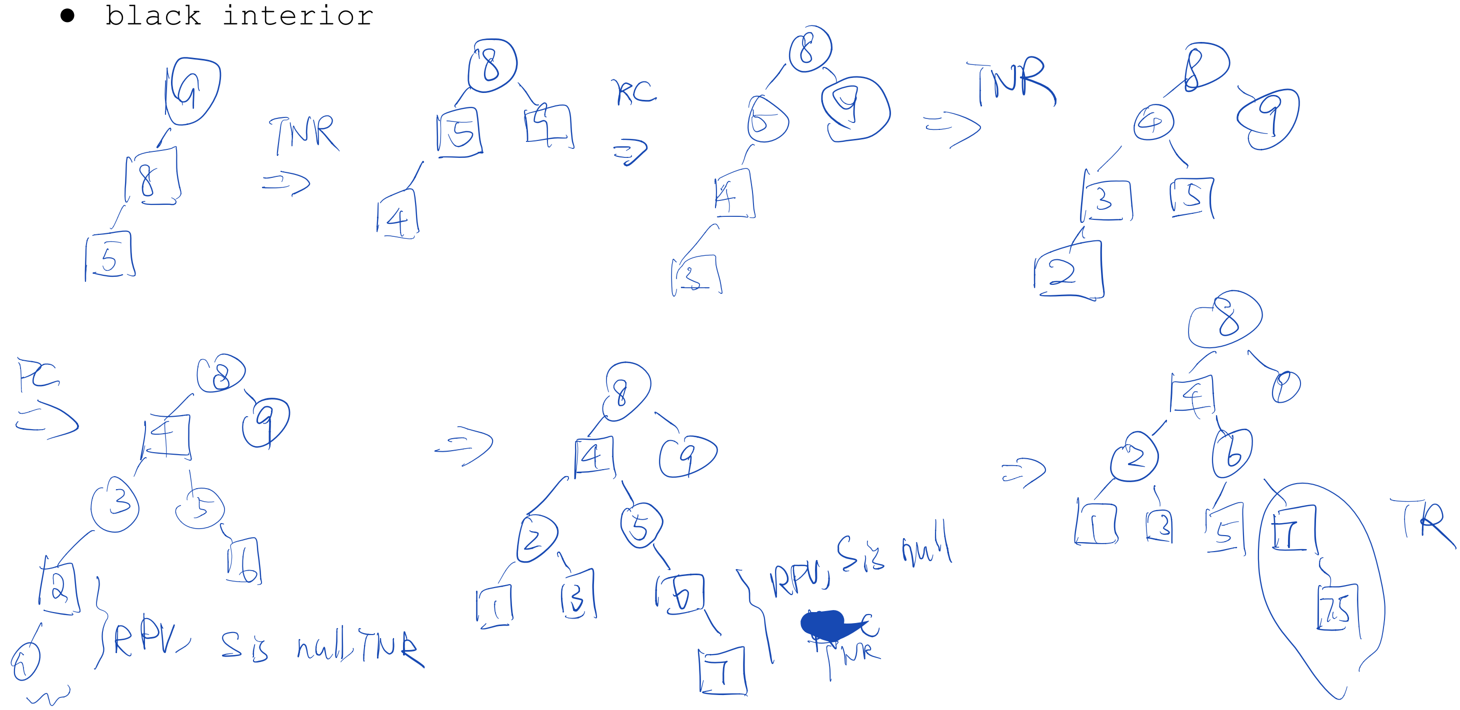
RBT Delete Practice

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

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)