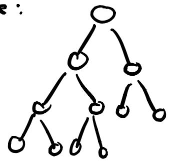
Agenda:

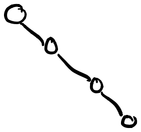
- → Radix Sorting ↗ stability } Poll
- → Sorting Review }
- → Balanced Trees!
  - → problem w/ BST (insert order)
    - ↳ motivation?

BST Problems

Best Case:
"balanced"



worst case:
"spindly"



height: Θ(n) → Linked List!

height: Θ(logn)

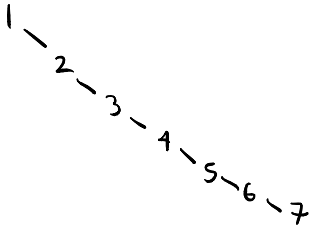
→ height grows linearly OR better!

Insert order

Create an insert order for 1, 2, 3, 4, 5, 6, 7
to make the tree

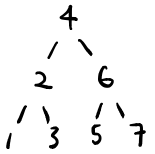
1.) Spindly

→ add(1)
→ add(2)
⋮
→ add(7)



2.) Balanced

add(4), add(2), 1, 3, 6, 5, 7



→ How do we ensure trees are bushy?

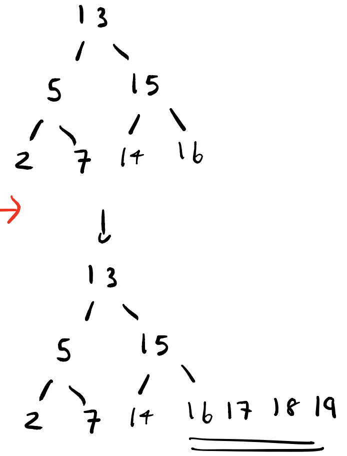
B-trees

→ 2-3 trees

thats what increases height

→ problem is in adding leaves

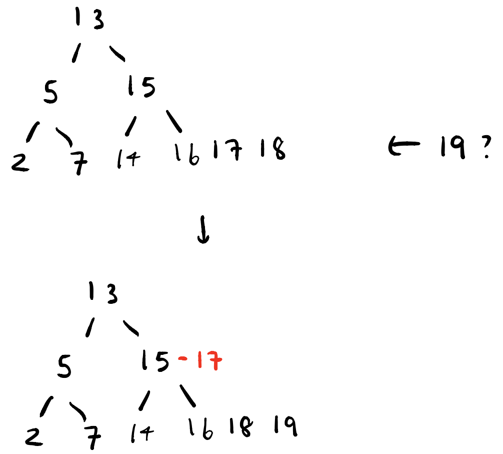
↳ Sol : stack new items at leaves



Solution: set limit, say L=3

→ if node # > 3, give node to parent

→ arbitrarily left middle (makes most sense)



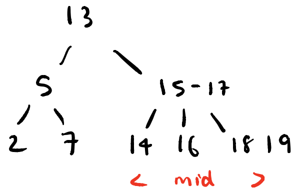
new problem : 16 > 17 ?

→ solution : size 2 node has 3 children

max k+1 children

=) size k node has k+1 children (true for k=1)

=)



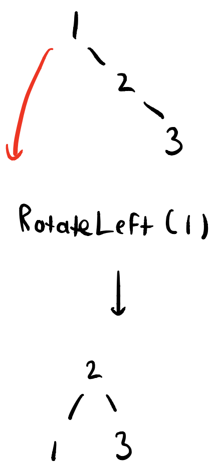
Do Prob 2, Do Prob 3

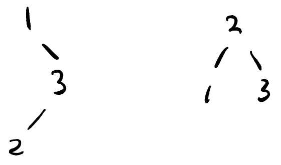
Problem: 2-3 hard to implement → not very easy to "send nodes up"

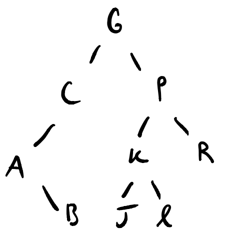
→ Left Leaning Red Black Trees! → use Rotations instead!

Rotation:

- note diff tree BST depending on insertion order
- You can get from one BST shape to anothur!



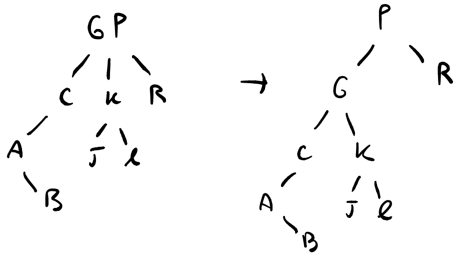


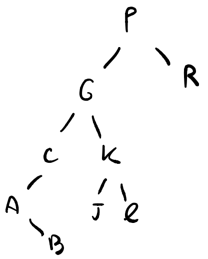


Rotate Left (G)

→ temp. merge

→ send down { L

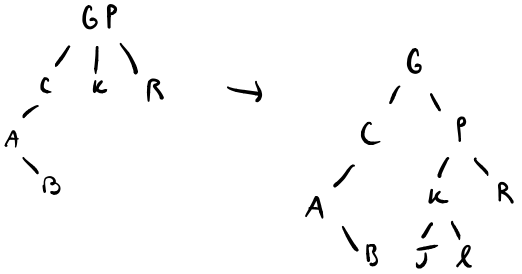




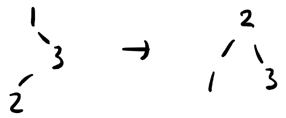
Rotate Right (P)

→ temp merge

→ send down {R

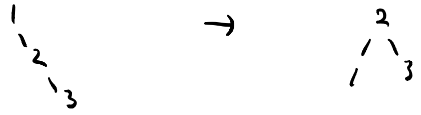


Test:



Rotate Right (3)

Rotate Left (1)



Do Animation

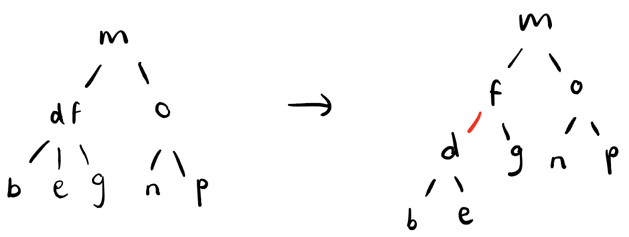
Why Rotations?

↙ our goal

→ easy to implement → can result in Balanced Trees

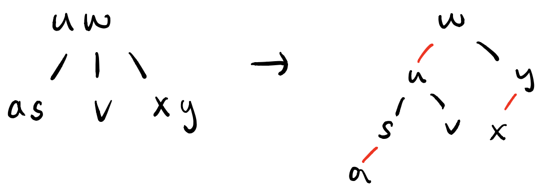
LLRBs - like 2-3 but easier to implement

- any 2-3 tree can be converted to an LLRB (vice versa!)



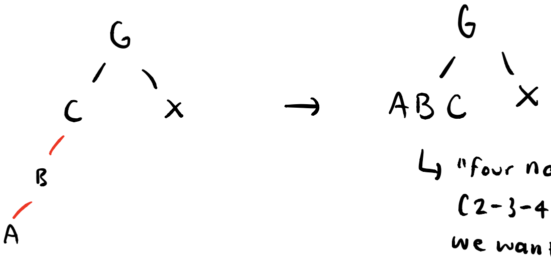
→ store smaller item off to left (left leaning)

→ just convenient book keeping (code: store a marker on each edge)



Rules:

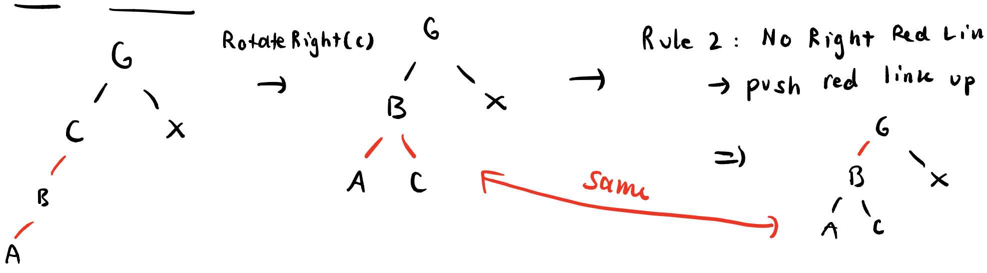
Problems: Must Always agree w/ 2-3 structure (1-1 correspondence)



↳ "four node" (2-3-4 ✓) we want 2-3

Rule 1: Two Red nodes cannot appear in a row

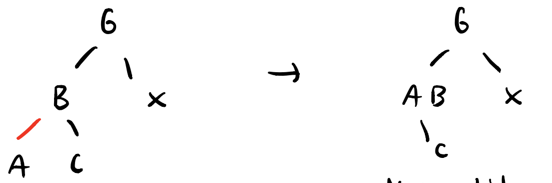
Fix: Rotate!



Rule 2: No Right Red Links!

→ push red link up

⇒



Rule 3: Root to all leaves must have same # of black links

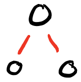
Not valid
=> not balanced

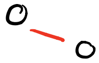
→ 1 link to A
→ 2 to C
→ 1 to x

} Not allowed

Summary : LLRBs

1.) LLRB 1-1 correspondence w/ 2-3 trees

2.) One node cannot have 2 red links  → violates 2-3 property

3.) No Right Red Links 

4.) Every path from Root to leaf has same # of black links → ensure balance

5.) Always insert using Red Link → merge into 2-3 tree leaf