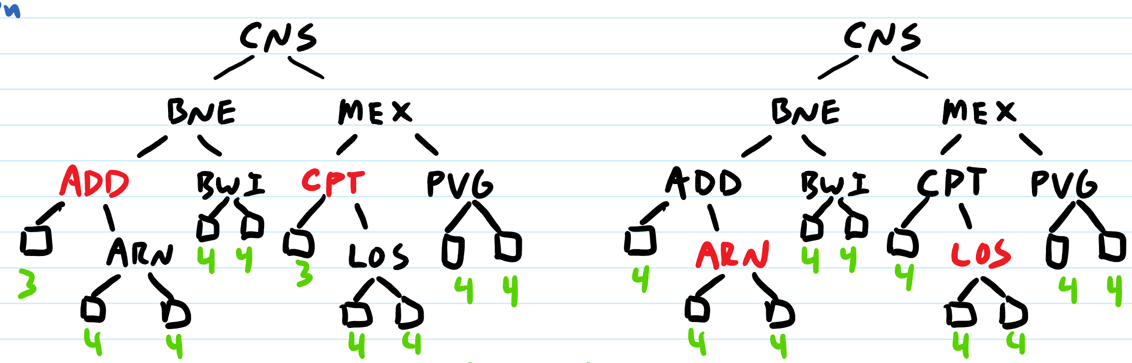
Binary Search Trees such that

1) each node has a color: red or black
2) root is colored black
3) all leaves are colored black
4) children of red nodes are black
5) every path root → leaf has same # of black nodes

add empty leaves everywhere – all data nodes have 2 children

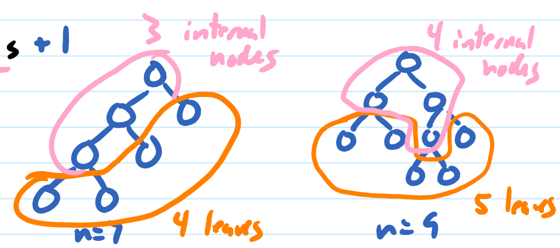


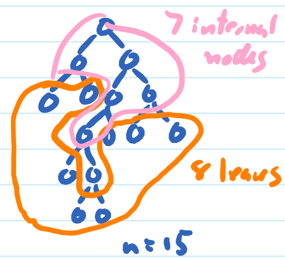
→ 0 or 2 children per node

leaves in a non-empty full binary tree = # internal nodes (non-leaves) + 1

if height is $O(\log_2 \text{total nodes}) = O(\log_2 2\cdot \text{data nodes} + 1)$

$= O(\log_2 \text{data nodes})$

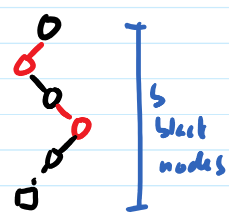




suppose # black nodes in path root → leaf is b (black height)

shortest possible path has b nodes (all black)

longest possible path has 2b-1 nodes



Red-black tree with black height b has $\geq 2^b - 1$ nodes

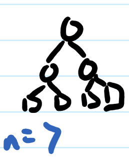
$n \geq 2^b - 1$

$n+1 \geq 2^b$

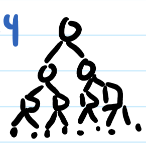
$\log_2 n+1 \geq b$

b is $O(\log_2 n)$

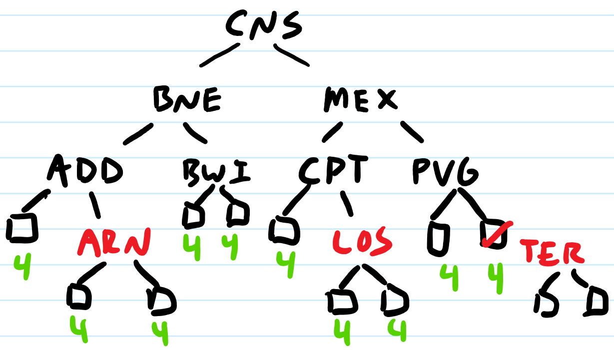
b=3



b=4



h is $O(\log_2 n)$ since $h \le 2b$



Insert

(assume tree is a red-black tree before insert)

1) Do normal BST insert, color new node red (w/ black leaves)
(doesn't affect prop 5)

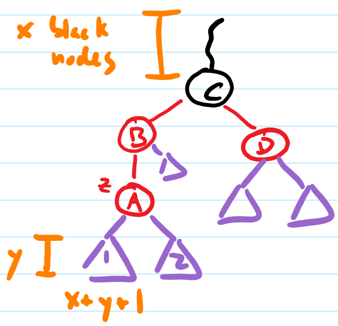
2) If new node's parent exists and is black, DONE (prop 4 still holds)

Else

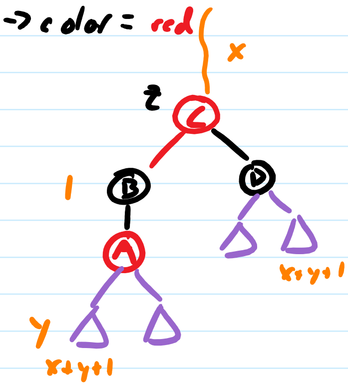
let z = new node

while z is not root and z -> parent -> color = red

if

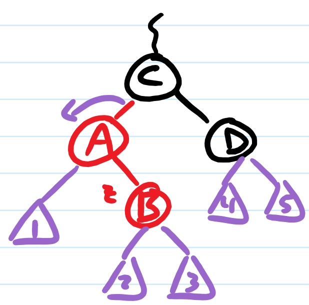


then

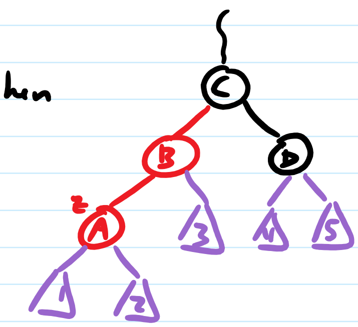


else

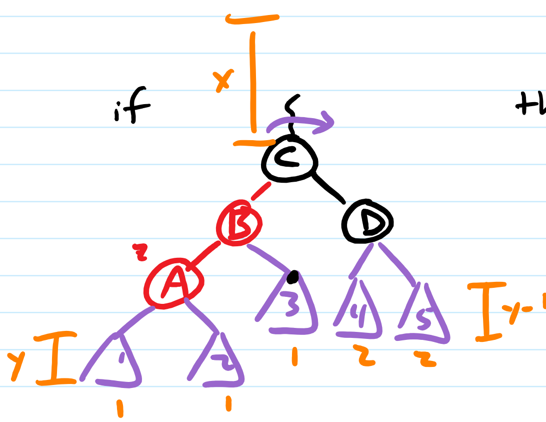
if



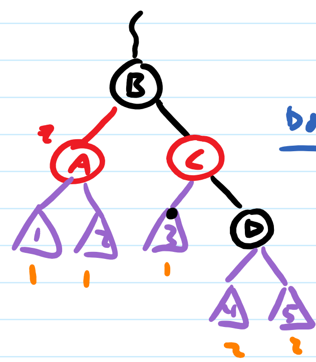
then



if



then



DONE

[other cases symmetric]

t → root → color = black (all black node counts root → leaf ↑ 1)

Example

ARN CNS CPT BWI MEX ADD PVG LOS BNE EZE