Red-Black Trees

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 \rightarrow leaf has same # of black nodes

Add empty leaves everywhere - all data nodes have 2 children

[Diagram of red-black tree]

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

\[ \text{if height is } \log_2 \text{ total nodes} = O(\log_2 \text{ data nodes}) \]

(longest path)

# black nodes in path root \rightarrow leaf is \( b \)

shortest possible path has \( b \) nodes (all black)

longest possible path has \( 2^b - 1 \) nodes

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

\[ n \geq 2^b - 1 \]

\[ b = 4 \quad n+1 \geq 2^b \]

\[ \log_2 n+1 \geq b \]

\[ b = O(\log_2 n) \]
\[ h \gg 0 \left( T_n \right) \]
(assume tree is a red-black tree before insert)

1) Do normal BST insert, color new node red (w/ 4 leaf laws) (doesn't affect pop 5)
2) If new node's parent exists and is black, DONE (pop 4 still holds)

Else

let z = new node

while z is not root and z → parent → color = red

if x black nodes

then

else

if

then

[other cases symmetric]
\( e \rightarrow \text{root} \rightarrow \text{color} = \text{black} \) (all black node count root to leaf?)