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 (no children)
4) children of red nodes are black
5) every path to leaf has same number of black nodes

Add dummy nodes so that all data nodes have 2 children; dummy nodes always black

not a red-black tree (violates #5)
red-black tree!

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

all nodes have 0 or 2 children non-leaf

if height \( O(\log_2 \text{total nodes}) \)
\[ = O(\log_2 2 \cdot \#\text{data nodes} + 1) \]
\[ = O(\log_2 2n + 1) \]
\[ = O(\log_2 n) \]

(black height)

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

shortest possible path has \( b \) nodes

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

Nov 3rd 2020 Page 2
Red-black tree with black height $b$ has $\geq 2^b - 1$ nodes

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

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

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

\[ b \leq \log_2 (n + 1) \]

\[ h \leq 2b - 1 \leq 2\log_2 n + 1 \]

\[ h \text{ is } O(\log_2 n) \]
Insert

\[ O(\log n) \]

1) Do normal BST insert, color new node red (w/ black dummy nodes as leaves)

2) If parent of new node exists and is black, DONE!
   Else
   let z = new node
   while z is not root and z → parent → color = red
     if parents sibling is also red
       [other cases symmetric]
       e → red → color = black
   \[ O(\log n) \] iterations

\[ O(1) \] per iteration