CIS 313:
Intermediate Data Structure

week of Feb 11

sixth week of the term
red-black tree rules

1. every node is either red or black
2. the root is black
3. every leaf (null) is black
4. if a node is red, both of its children are black
5. for each node, all simple paths from the node to descendant leaves contain the same number of black nodes
red-black tree height

• (too) simple analysis:
• the black-height is at most $\log_2 n$
• the actual height is at most twice the black height
• so total at most $2\log_2 n$
• OK, text says at most $2\log_2 (n+1)$
red-black tree insertion

• to insert new key $x$
• as always, search to the bottom of the tree for where $x$ would go
• put $x$ there and color it red (to maintain black-height)
• this might cause a problem: two reds in a row
• if no such problem, then done
• if double-red problem, then fix using
  • color shifts or
  • rotation

example: insert 7
RB-INSERT-FIXUP

• section 13.3 of text
• this deals with the double red case after an insertion
• let y be the current node, both it and its parent are red
• let z be the “uncle” of y: the sibling of y’s parent’s parent
• two cases:
  • z is red
    • color shift
    • then check again for double red, possibly continue
  • z is black
    • rotate
    • done
z is red

color shift

this swaps a black and red level, preserving black height along these paths, but may create another double-red at the new y
z is black

rotation (double)

done now
z is black (again, different case)

there are two other cases similar to these needing single and double left rotations
turn a binary search tree into a red-black tree

code taken online
sample BST