CSE 101 5-12-20

Pas: ext 1 day

Today

- Deletion in a BST
- Pat
- Red-Black tree

Deletion in a BST

- Transplant(T, u, v) replaces the subtree rooted at u with subtree rooted at v.

Pa 6

Dictionary ADT again, but now based on RBT.

Note: ADT operation

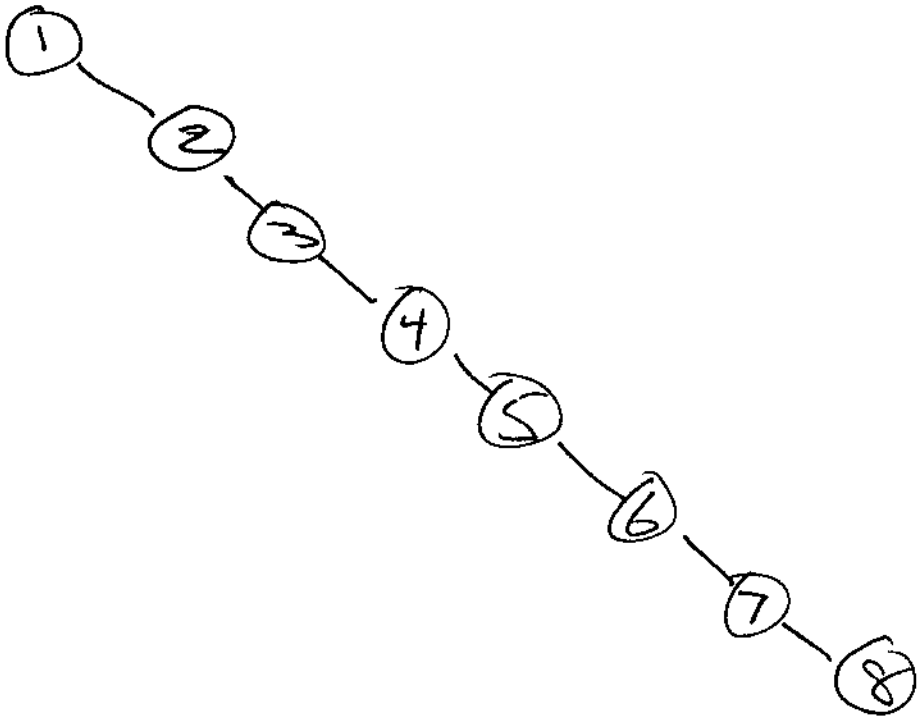
printDictionary( )

will have one new parameter, a char* giving "in", "pre", "post" to specify type of tree walk.

Ex.

Insert Keys 1 2 3 4 5 6 7 8

into an initially empty BST



Cost: $1+2+3+4+5 \ldots +8 = \frac{8 \cdot 9}{2} = 36$

recall: $1 + \ldots + n = \frac{n(n+1)}{2}$

$= \frac{1}{2}n^2 + \frac{1}{2}n = \Theta(n^2)$

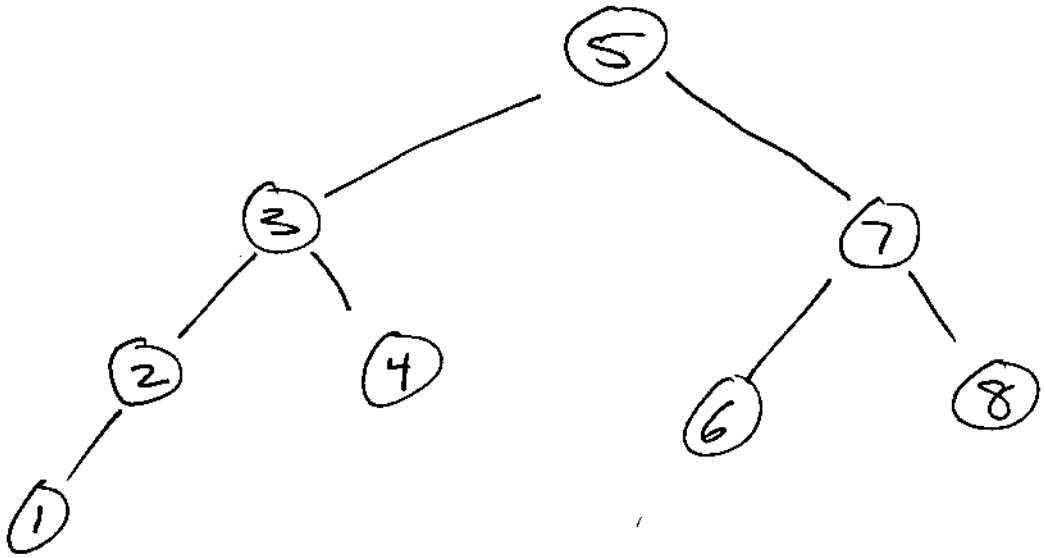
In: 1 2 3 ---- 8

Pre: 1 2 3 --- 8

Post: 8 7 6 ---- 1

Ex. same keys, now in order:

5 3 7 2 1 6 4 8



In: 1 2 3 8

Pre: 5 3 2 1 4 7 6 8

Post: 1 2 4 3 6 8 7 5

Chapter 13 Red-Black Trees (RBT)

Defn

an RBT is BST satisfying the RBT Properties below. By convention the leaves in a RBT are considered to be the nil-children of the Key bearing nodes.

RBT Properties

1. each node has color RED or BLACK.
2. root is BLACK.
3. each leaf (nil child) is BLACK.
4. every RED node has 2 BLACK children, i.e. no RED node has a RED child.

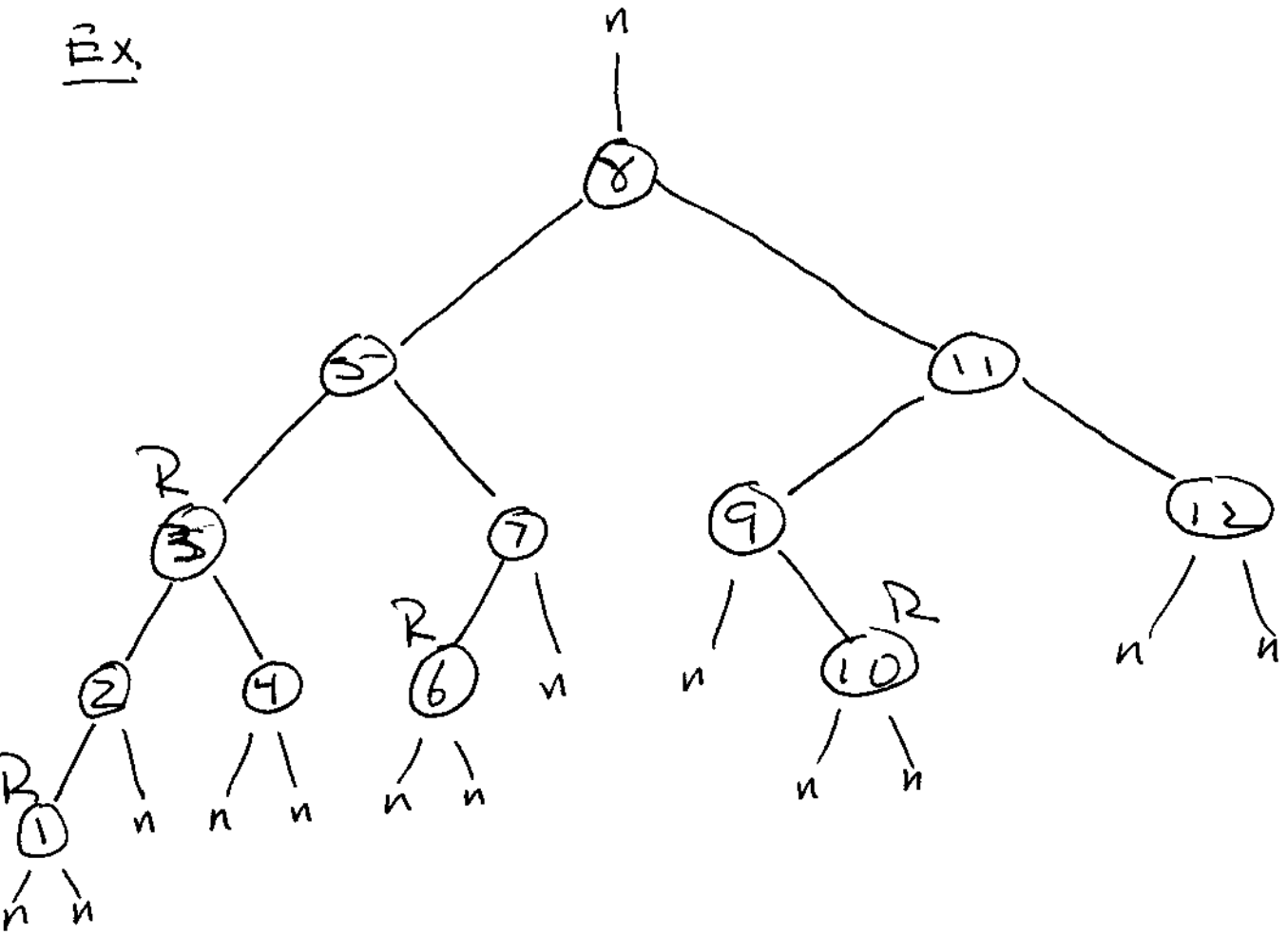
5. for any node x, every descending path from x to a leaf has the same number of black nodes.

Defn

The black height of a node x, denoted bh(x) is the # of black nodes in any descending path from x to a leaf (not counting x itself.)

note: bh(x) is well defined by RBT property (5).

Ex.



nodes	black height
8	3
5, 11, 3	2
1, 2, 4, 6, 7, 9, 10, 12	1
all nil leaves	0

note: $bh(x) = 0$ iff $height(x) = 0$ iff x is a leaf.

- how ?
- why ? ✓ (good run time)

Theorem

A RBT with n internal (non-nil) nodes and height h satisfys

$$h \leq 2 \lg(n+1)$$

Rmks

- recall any B.T. satisfies $h \geq \lfloor \lg n \rfloor$.
  so, a RBT satisfies

$$\lfloor \lg n \rfloor \leq \text{height}(T) \leq 2 \lg(n+1)$$

$$\text{so} \quad \frac{\lfloor \lg n \rfloor}{\lg(n)} \leq \frac{\text{height}(T)}{\lg(n)} \leq \frac{2\lg(n+1)}{\lg(n)}$$

$\lg(n)$ → const_1 ; $\frac{2\lg(n+1)}{\lg(n)}$ → const_2

∴ if T is a RBT on n nodes, then

$$height(T) = \Theta(\log n)$$

∴ run time of: TreeMin(), TreeMax(), TreeSearch(), TreeSuccessor(), TreePredecessor() is $\Theta(\log n)$

- Furthermore its possible to alter insert & delete so that they preserve RBT properties, and run in time $\Theta(height(T)) = \Theta(\log n)$.

Sec. B.2 : Rotations

Pick up here ...

- See CLRS
- See Pseudo-code in Examples