CSE 101 2-23-23

PA6:

Ex from string count

"#@@@7" → "7"

Dictionary ADT (PA7, PA8)

a state: a set of ordered

Pairs: (key, value)

Keys are unique

Operations :

- lookup (key)
  return value assoc. with key
  or nil if key does not exist.

- insert (key, value)
  adds new pair to dict.
  Pre: lookup(key) = nil

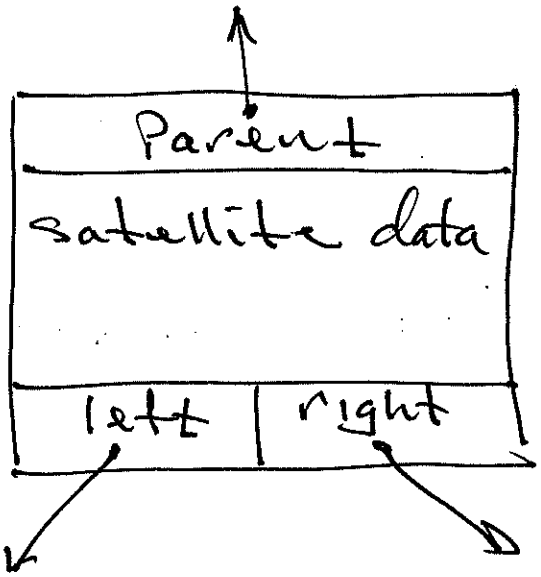
- delete (key)
  removes (key, value)
  Pre: lookup(key) ≠ nil

How to implement Dictionary

- BST : chap 12 (Pa 7)
- RBT : chap 13 (Pa 8)
- HashTable : chap 11

Binary Search Tree (BST)

Node Object



Dictionary: (key, value)

simplify: key

BST Properties:

Let x, y be nodes. Then

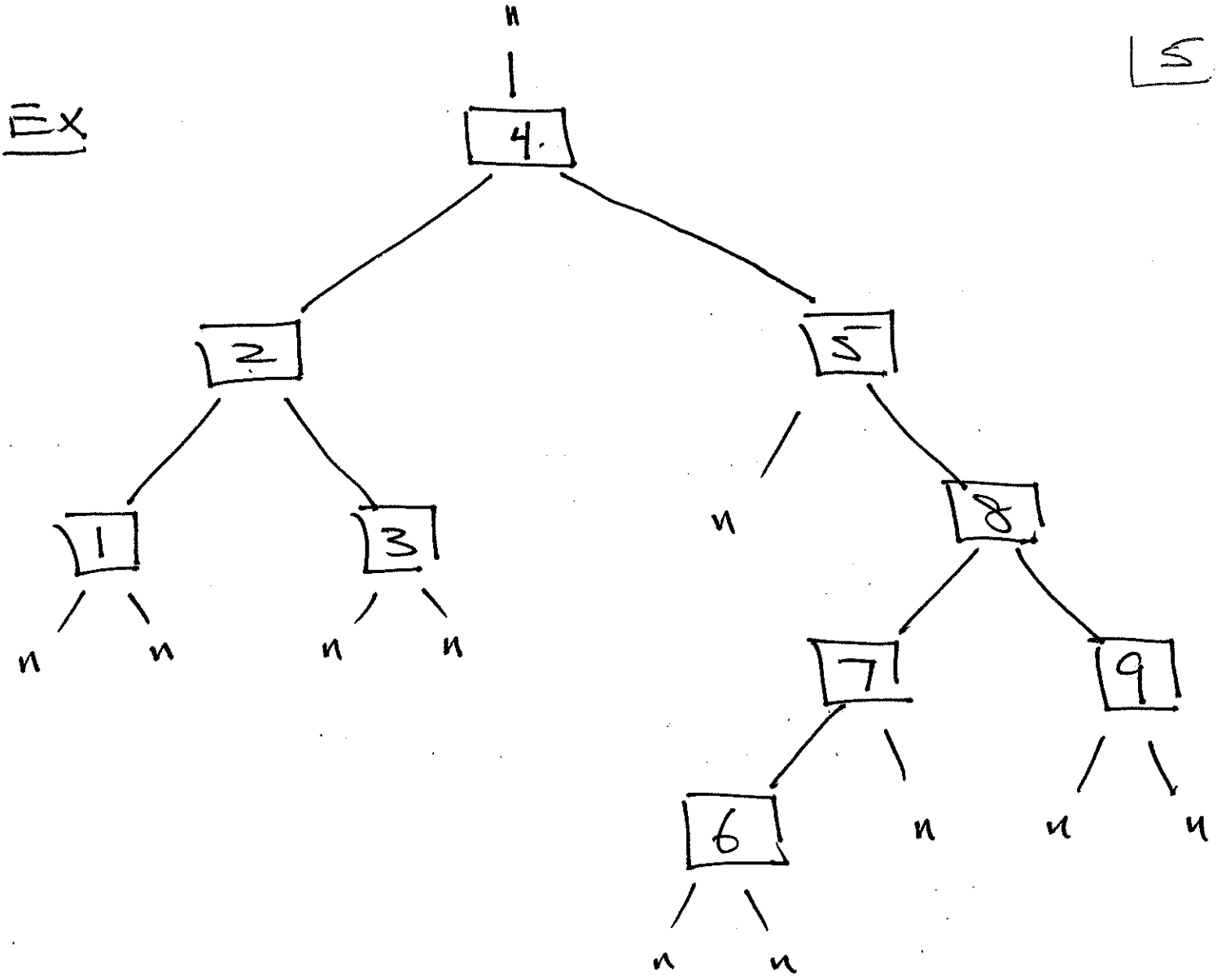
(1) if y is in the left subtree of x, then

$$key[y] \leq key[x]$$

(2) if y is in the right subtree of x, then

$$key[x] \leq key[y]$$

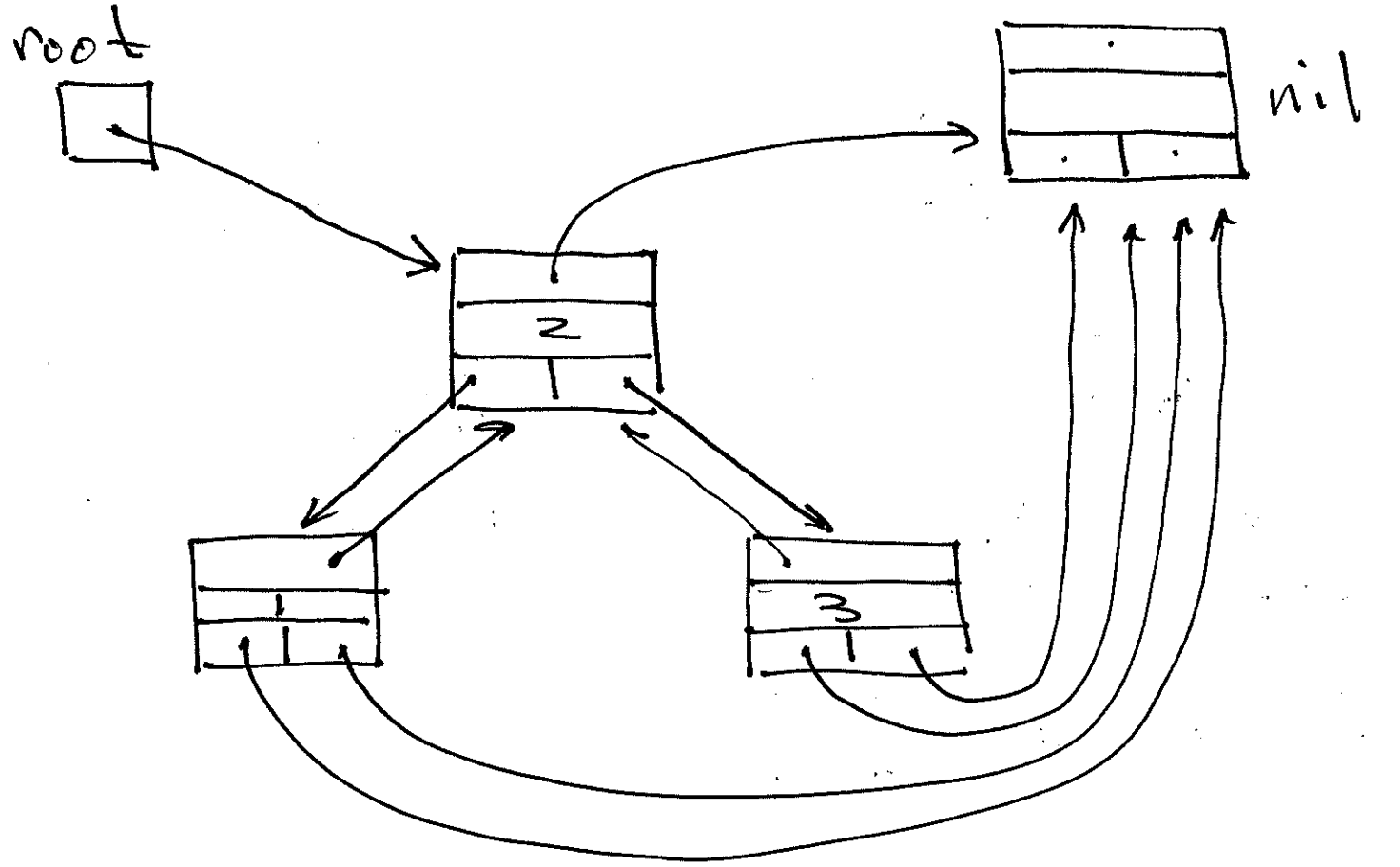
Ex

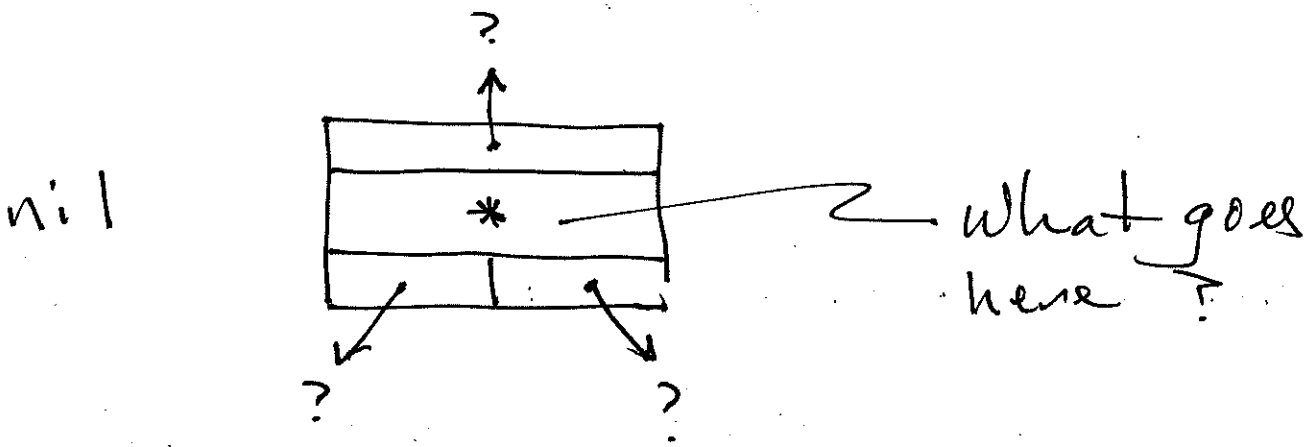


Defn

- a leaf is a node with no children
- an internal node is a non-leaf

sentinel nil node :





- tree traversals

  - InOrderTreeWalk(x) ← node
  - PreOrderTreeWalk(x)
  - PostOrderTreeWalk(x)

- output of InOrderTreeWalk on example:

  1 2 3 4 5 6 7 8 9

- PreOrderTreeWalk:

  4 2 1 3 5 8 7 6 9

- post order tree walk:

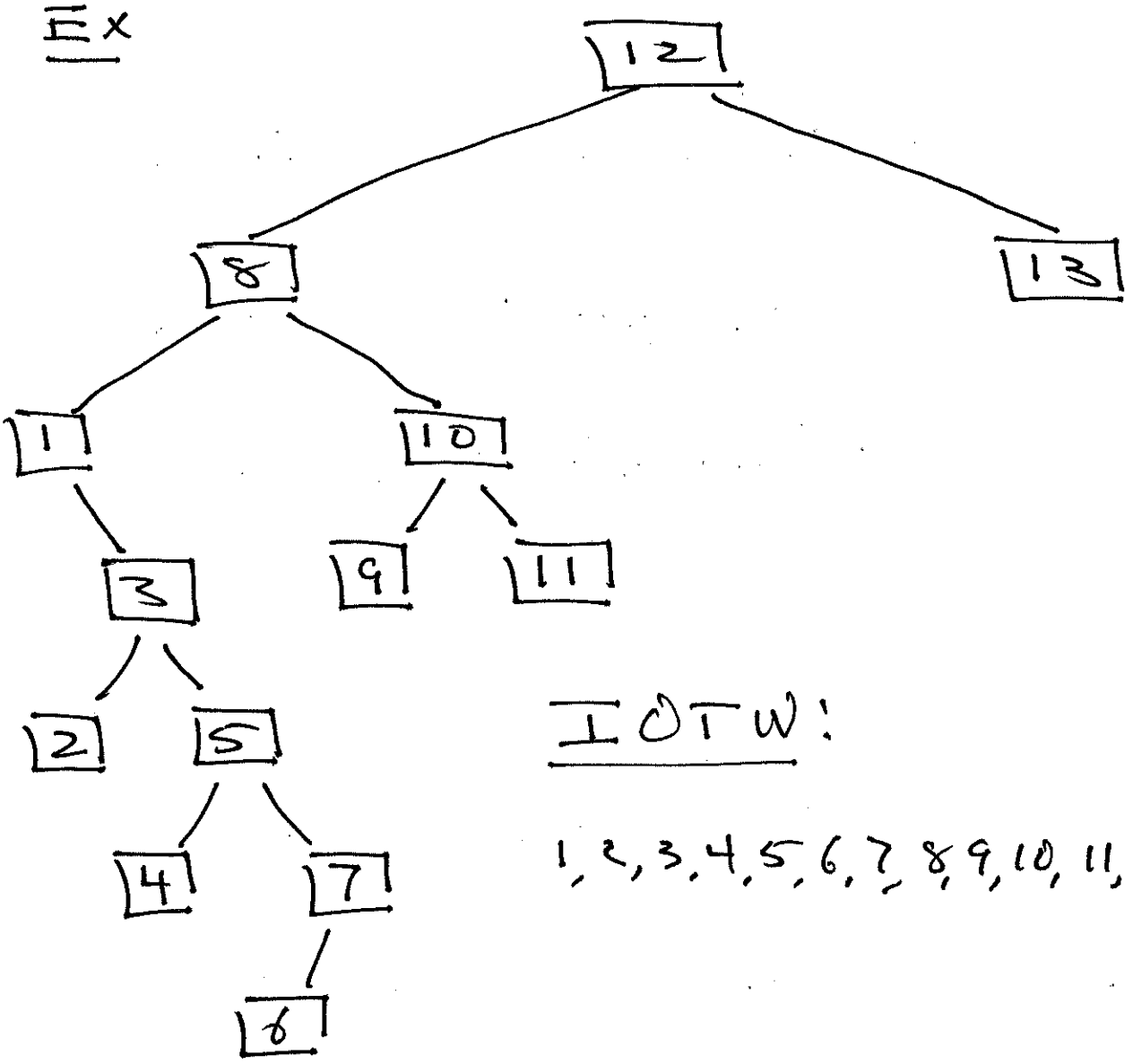
  1 3 2 6 7 9 8 5 4

- Queries
  - TreeSearch(x, k)
  - TreeMinimum(x)
  - TreeMaximum(x)

Def:

The <u>successor</u> of a node x is next node after x to be processed in an in-order tree walk

Ex



IOTW:

1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13

successor(1) = 2 case 1

successor(8) = 9 case 1

successor(7) = 8 case 2

Two Cases:

Case 1: x has a right child

successor of x is the leftmost descendant of that child.

Case 2: x has no right child

successor of x is the lowest ancestor of x whose left child is also an ancestor of x. (note: x is its own ancestor.)