CSE 101-01 5-12-22

- mid2 : Thur 5/19
- Pa6 : ext. one day (Tue)
- Pa7 : Posted, due mon 5/23

Dictionary ADT (or Map)

a state is a set of ordered pairs

(key, value)

i.e.

$\{(K_1, V_1), (K_2, V_2), \ldots, (K_n, V_n)\}$

Option: Keys are unique.

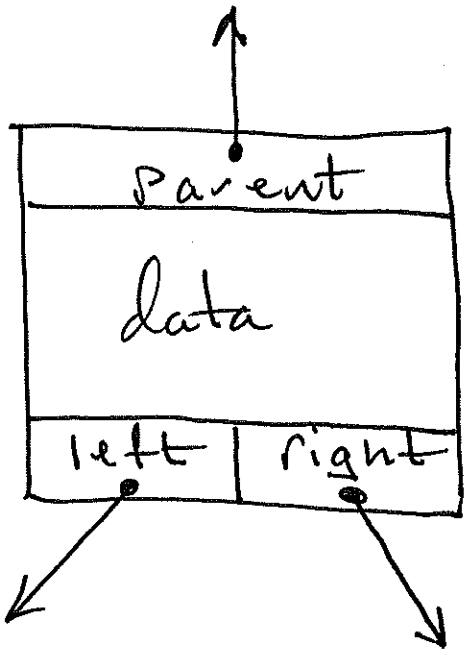
main operations:

- insert(K, V): adds (K, V) to a dictionary. Pre: lookup(K) == nil

- lookup(K): returns V s.t. (K, V) is in dict. or nil if no such pair exists.

- delete(K): removes (K, V) from dict. Pre: lookup(K) ≠ nil

Binary Search Tree (BST)



BST Properties

Let x, y be nodes in a BST

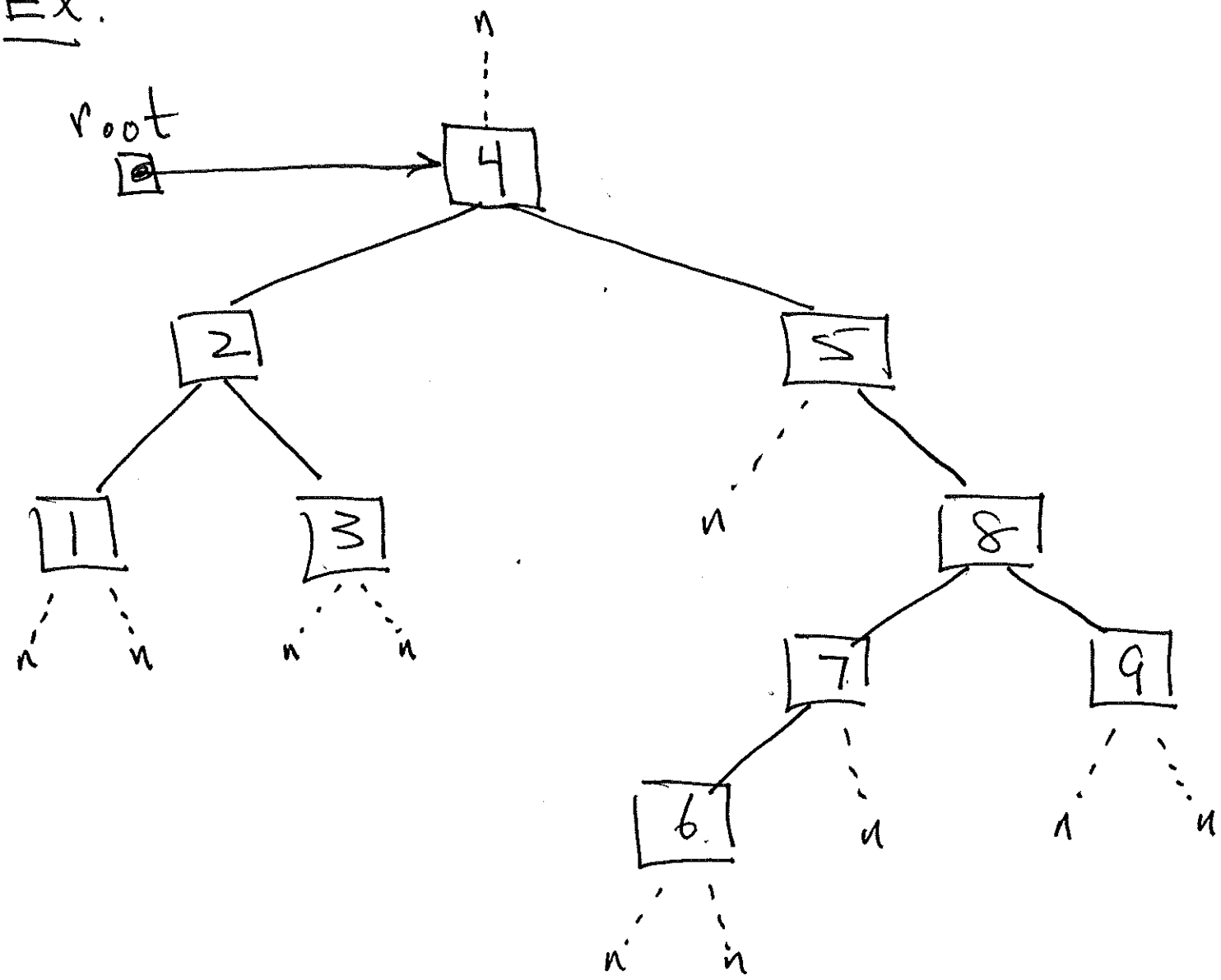
(1) if y is in the left subtree of x

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

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

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

Ex.



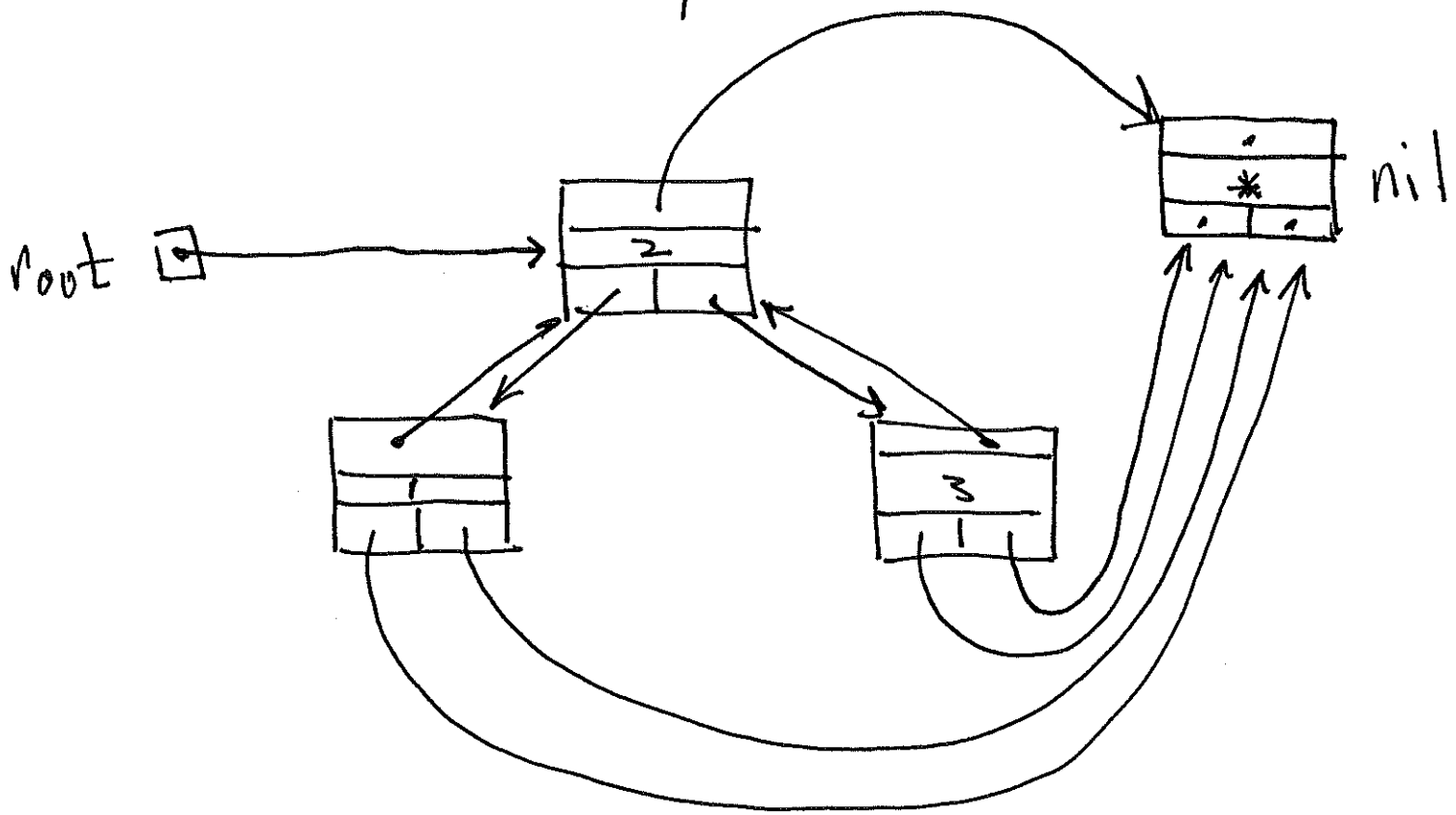
Defn

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

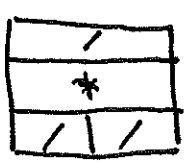
Implementation option:

Have a dummy (or sentinal) node called nil in our dictionary fields.

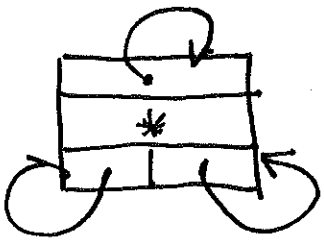
Ex. with dummy nil.



nil options



or

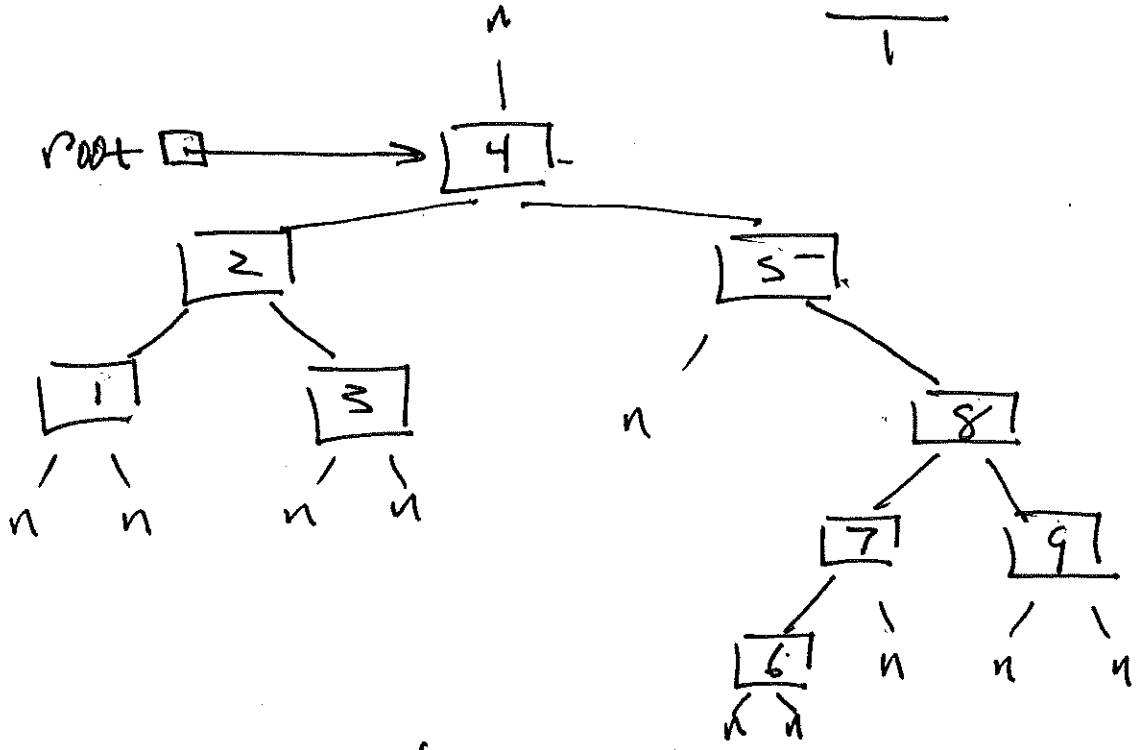


Tree Traversals

Node

- InOrderTreeWalk(x)
- PreOrderTreeWalk(x)
- PostOrderTreeWalk(x)

Ex.



InOrderTreeWalk(root[T]):

1 2 3 4 5 6 7 8 9

PreOrderTreeWalk(root[T]):

4 2 1 3 5 8 7 6 9

PostOrderTreeWalk(root[T]):

1 3 2 6 7 9 8 5 4

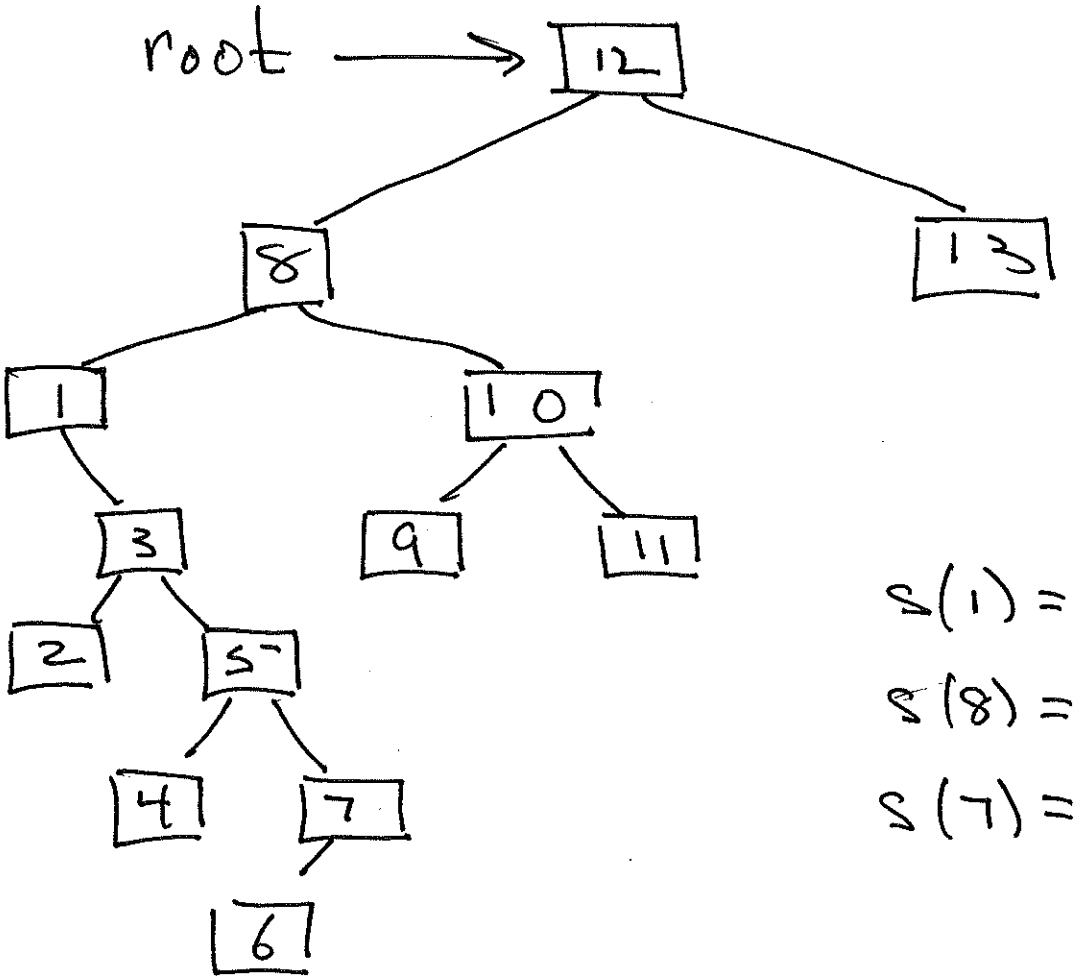
Queries

- TreeSearch(x, k)
- TreeMinimum(x)
- TreeMaximum(x)
- TreeSuccessor(x)
- TreePredecessor(x)

Def^n

The successor of a node x is the next node after x to be processed in an I.O.T.W.

Ex.



s(1) = 2] Case 1
s(8) = 9]
s(7) = 8] Case 2

I.O.T.W. : 1 2 3 4 5 6 7 8 9 10 11 12 13

Have 2 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.

Exercise:

- define Predecessor of x
- write TreePredecessor(x)

Runtime

- Traversals: in, pre, post } cost = $\Theta(n)$

where n = # nodes.

- Queries:

Depth

height(T) = dist. from root to deepest leaf

height(x) = height of subtree rooted at x.

note: height(x) ≤ height(T)