CSE 101 2-11-21

- Pa5 is due wed Feb. 17

Continue Successor of node x:

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

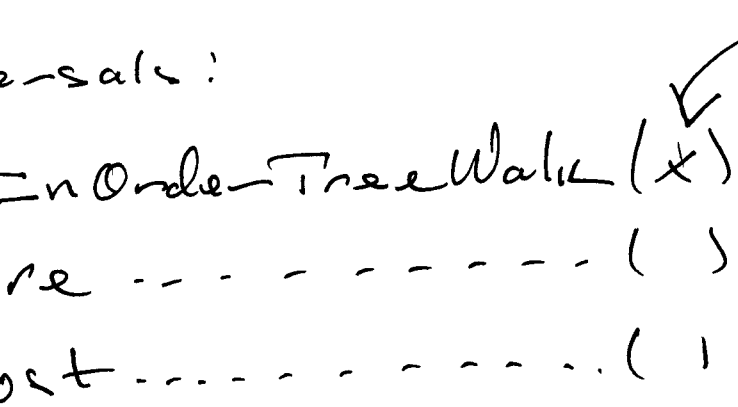
Exercise:

- write definition of Predecessor of x
- write the function

  TreePredecessor(x).

Runtimes:

Traversals:



cost = $\Theta(n)$, where n = # nodes in T

Queries.

- TreeMinimum(x) ← root[T]
- ... Maximum(·)

} cost = Θ(height(x))

- ... Successor(·)
- ... Predecessor(·)

} cost = Θ(height(Success(x)))

- TreeSearch( ) } cost = Θ(height(x))

Defn height(T) = distance from root to deepest leaf.

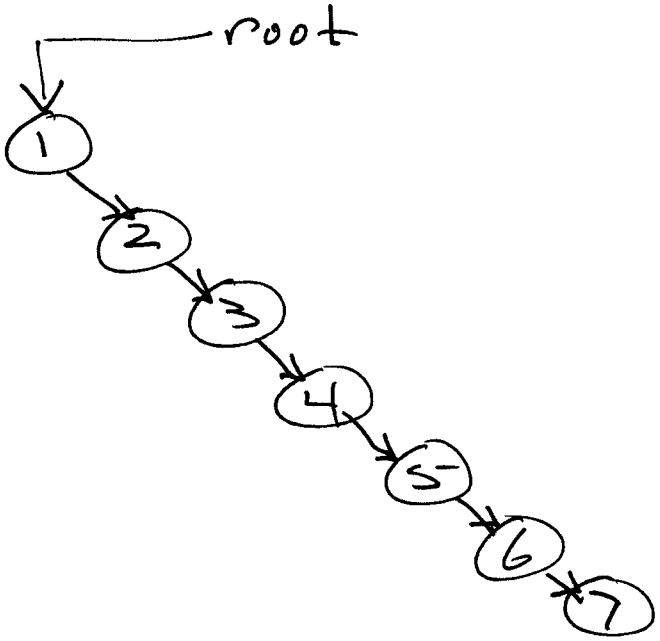
Defn depth(x) = distance from x to root.

Defn height(x) = height of subtree rooted at x, i.e. distance from x to its deepest descendant leaf.

In general (worst case) cost of a query is

Θ(height(T)).

Worst Possible BST for queries!



$n = 7$

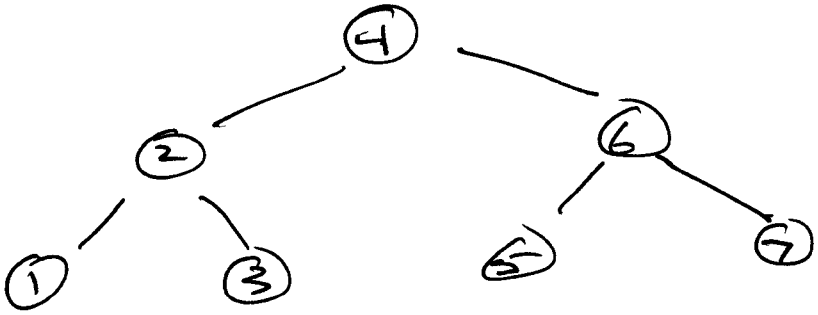
height(T) = 6

In general:

height(T) = n − 1

$= \Theta(n)$

Best BST for queries:



$n = 7$

height(T) = 2

In general: height(T) = $\Theta(\log n)$

Sec 12.3 : Insertion & Deletion

To insert a new node z into a BST, and maintain BST properties,

- set key[z] = k
- set p[z] = left[z] = right[z] = nil.

- find a node in T to adopt z as left or right child.

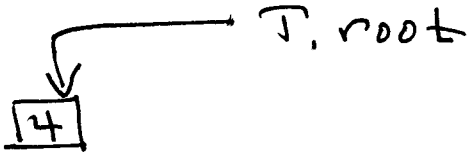
  strategy: do a search for key k in T, find the nil child where k would reside, then put it there.

see Pseudo code for

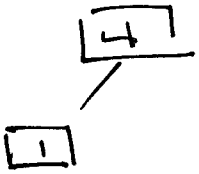
TreeInsert(T, z)

Ex. insert keys: 4, 1, 7, 2, 8, 3, 5, 6

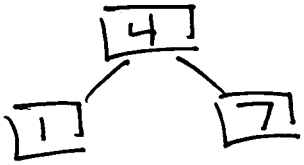
initially T.root = nil : empty tree



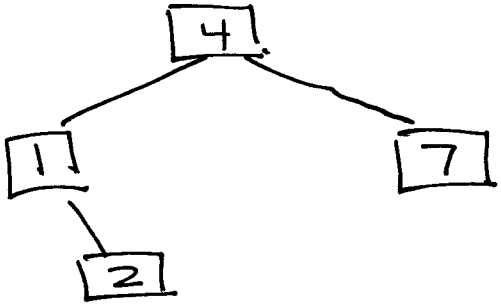
insert(4)



insert(1)

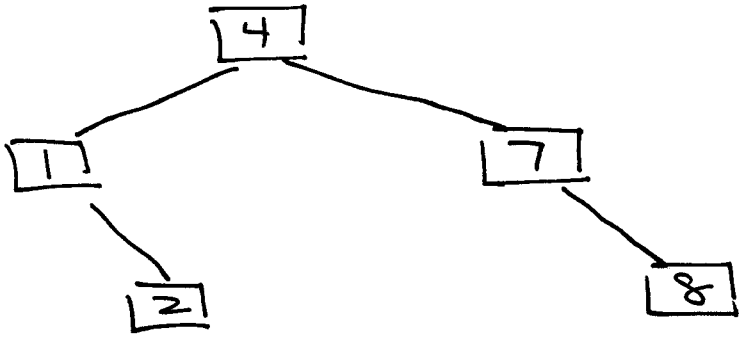


insert(7)

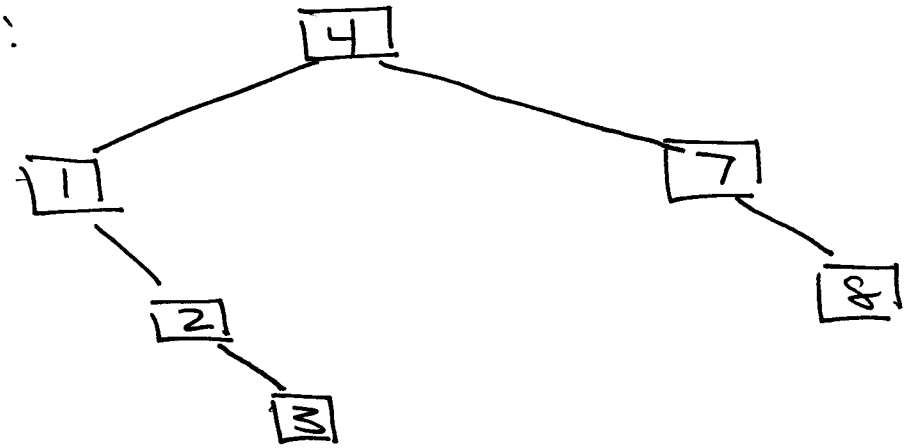


insert(2)

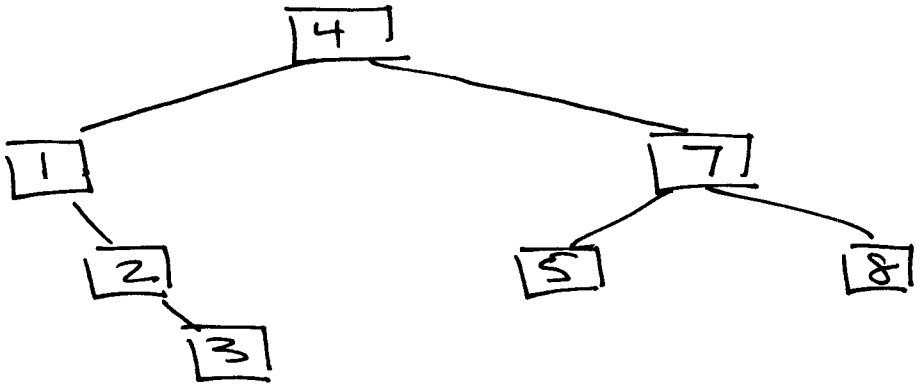
Insert(8):



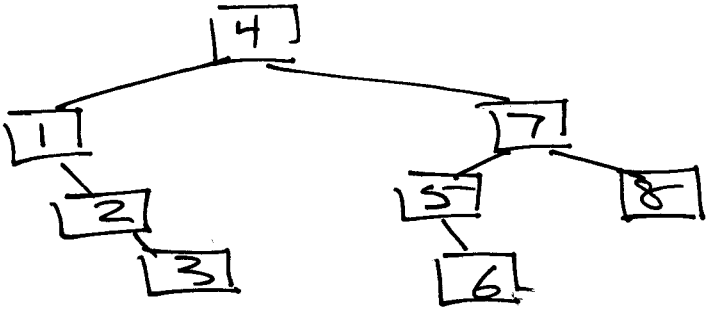
Insert(3):



Insert (5)



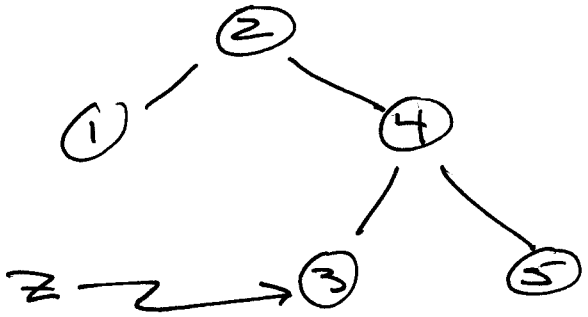
Insert(6)



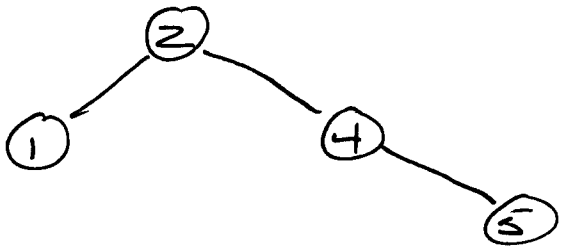
Deletion : delete z from T.

3 cases

- Case 1 : z is a leaf, i.e. no children.
  detach z from its parent

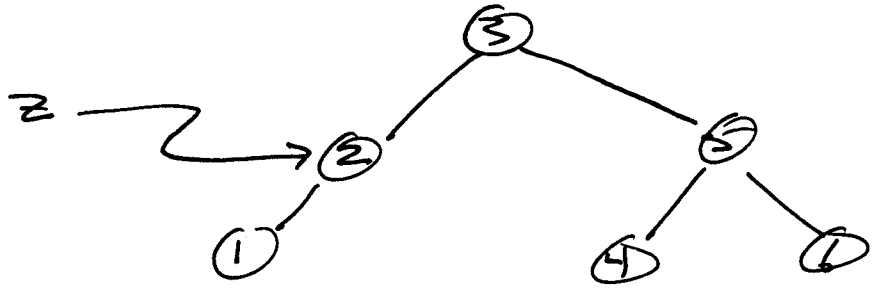


delete(3)



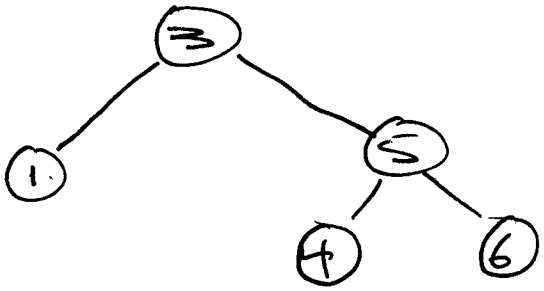
Case 2: z has 1 child

- subcase 2.1 : z has right but no left child
- subcase 2.2 : z ... left " " right " .



delete(z)

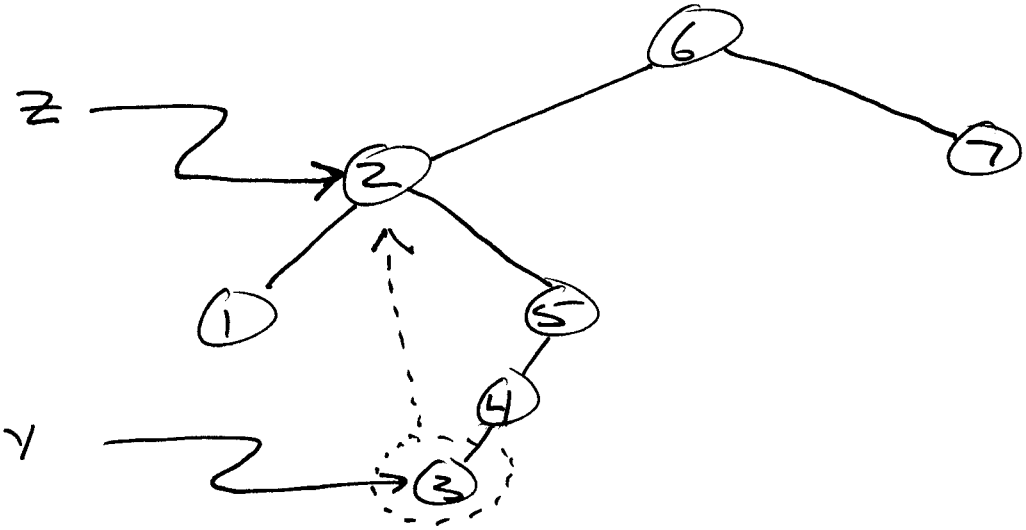
(subcase 2.2)



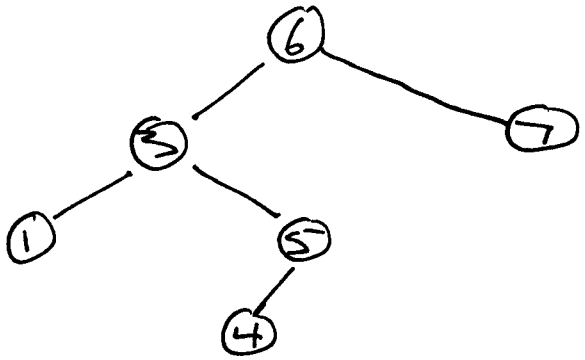
Exercise: draw example of subcase 2.1

Case 3: z has 2 children

splice out z's successor y
(which can have no left child since
it is leftmost descendant of z.right),
then replace z with y.

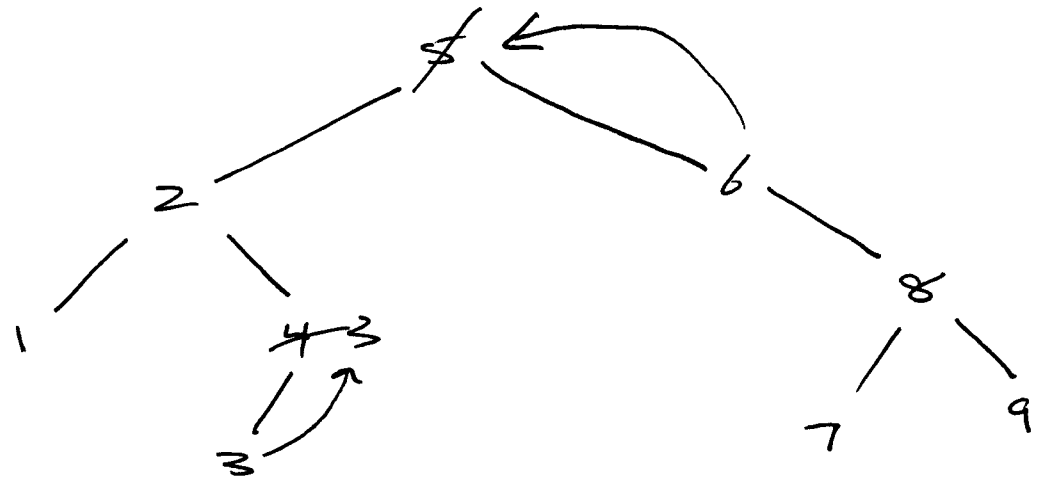


delete(2)

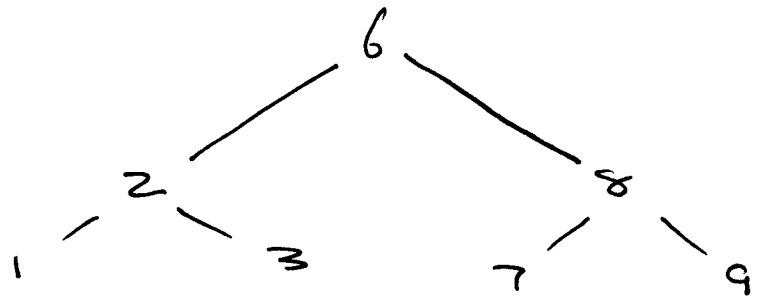


insert:

Ex. 4, 2, 6, 1, 8, 3, 7, 5, 9

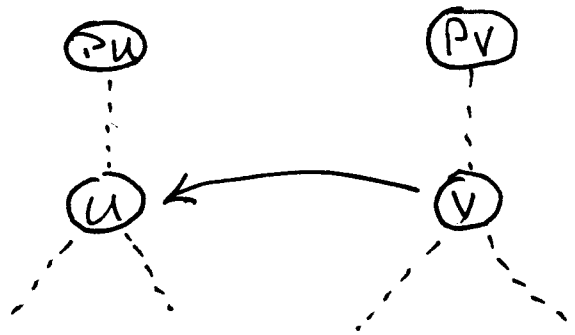


delete: 1 4



helper fcn for delete:

Transplant(T, u, v)



continue ....

Ex.