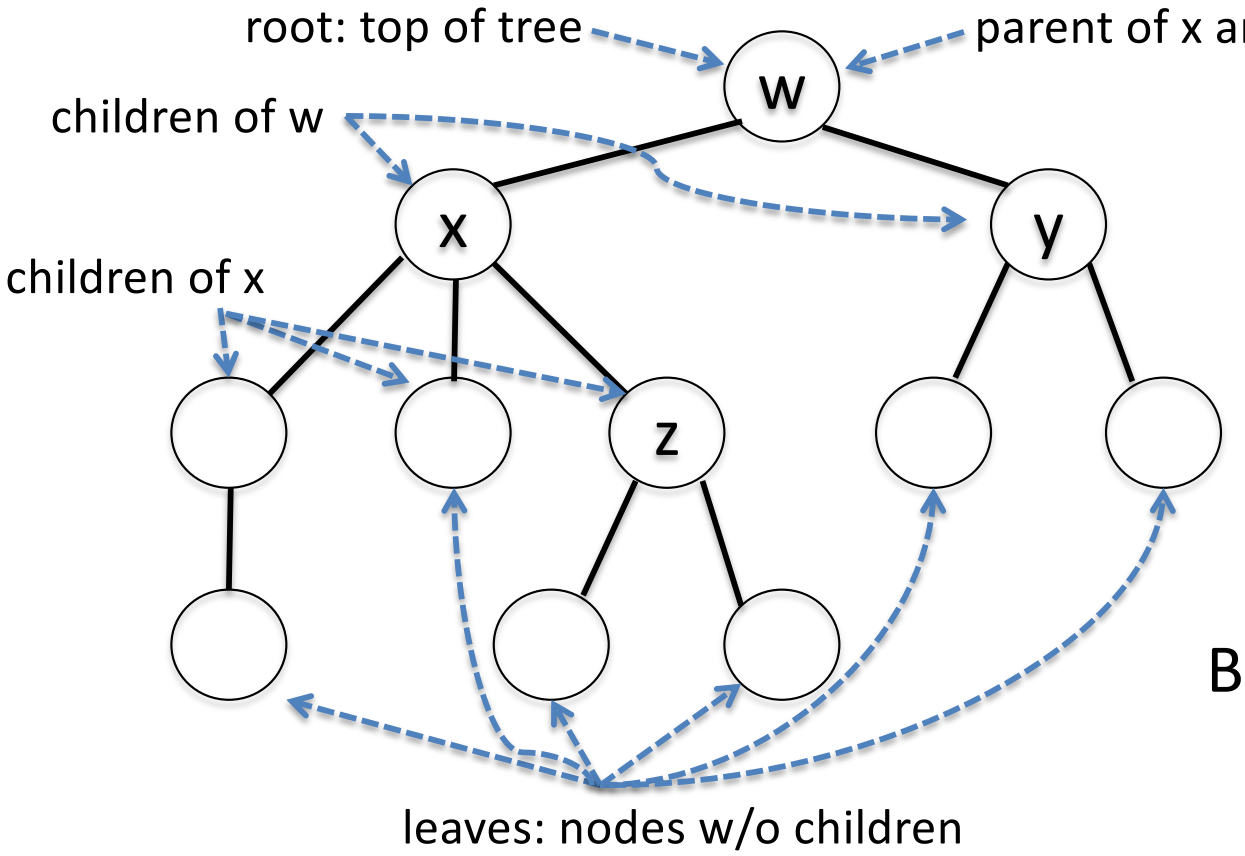
More trees

5/1/26

Administrivia

- HW 5 (sorting and search order) due tonight (5/1)

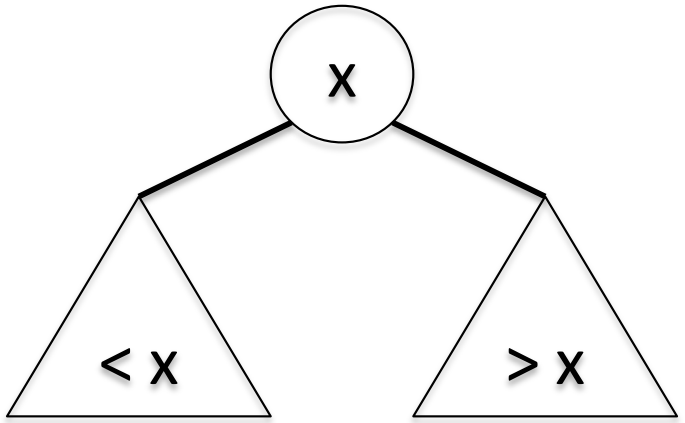
Recall: Trees



Other familial relationships:

sibling, grandparent, grandchild,
uncle (or aunt), ancestor, descendant

Binary Search Tree (BST)

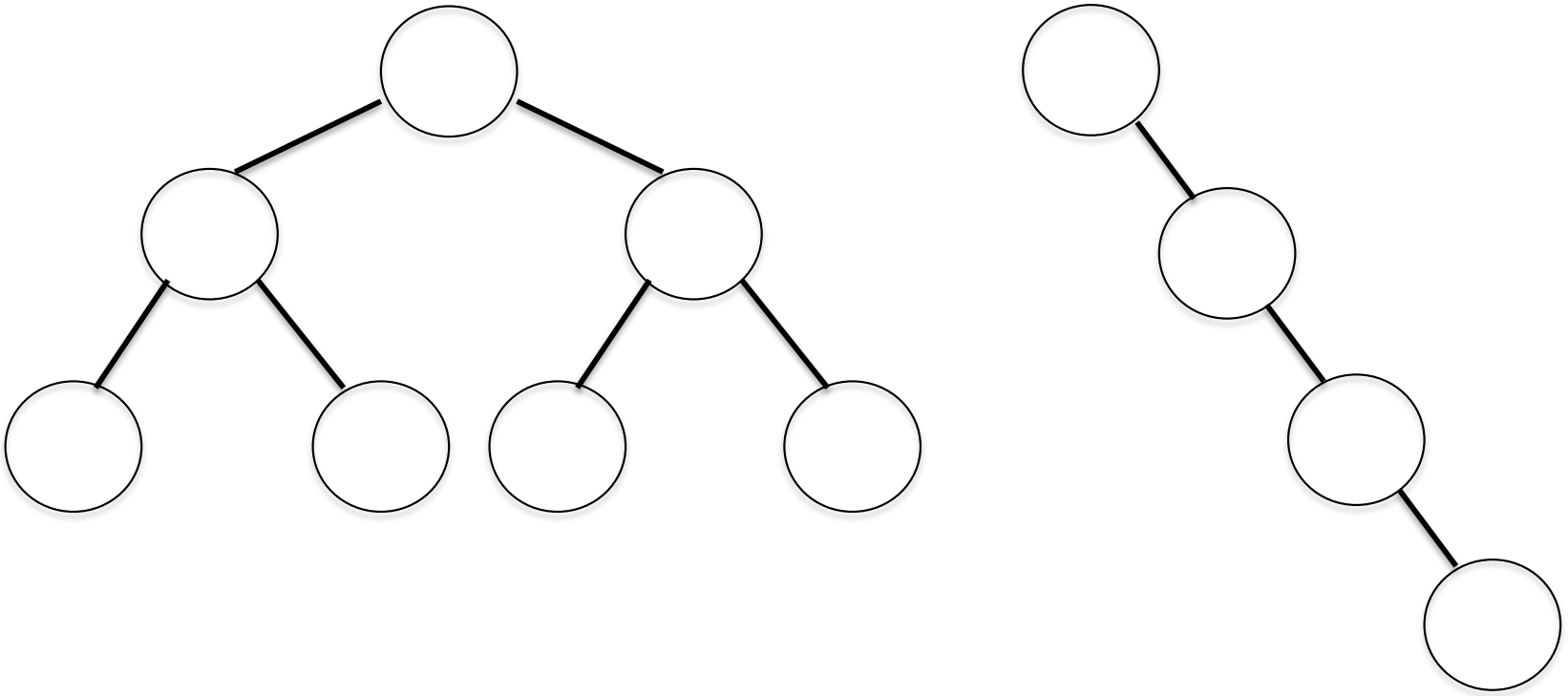


Other tree terminology

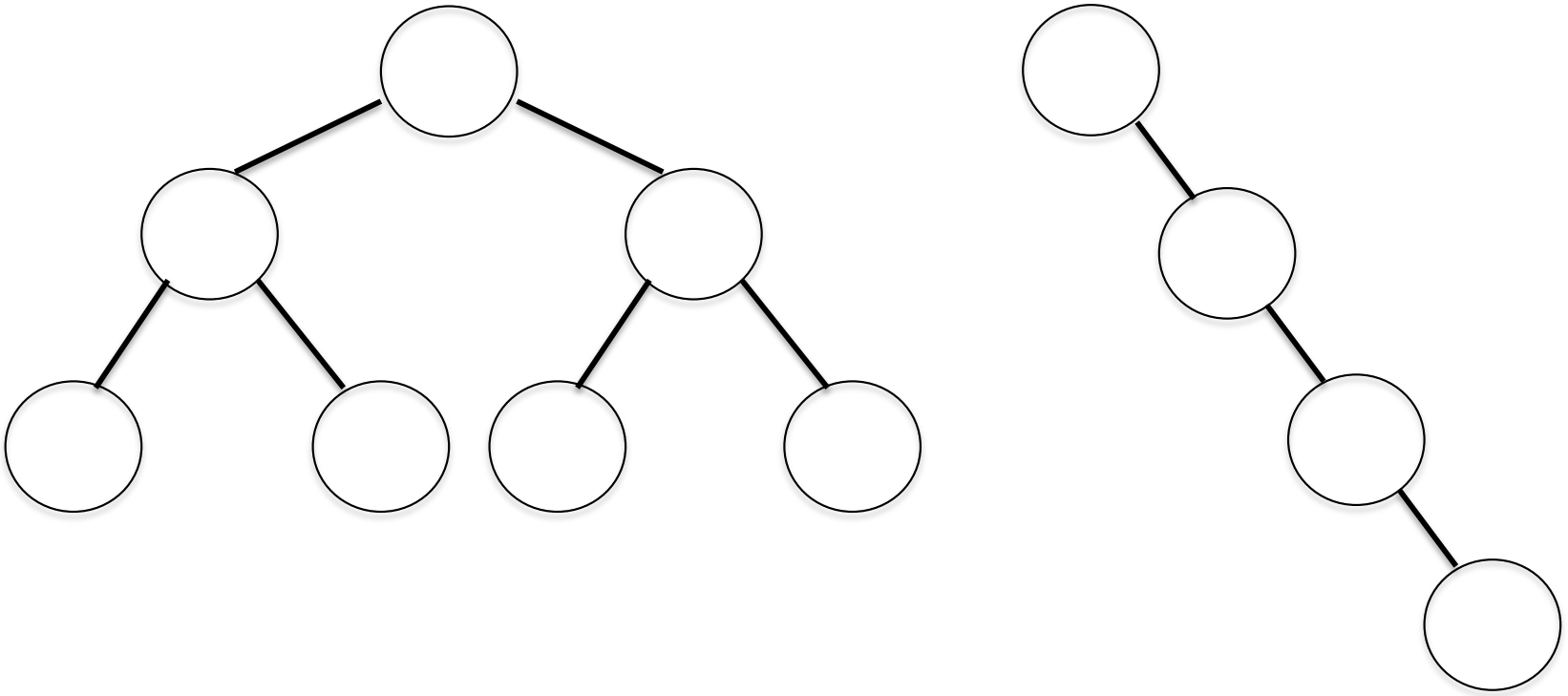
(book differs slightly; they count nodes)

- The *depth* of a node is the number of edges between it and the tree's root
- The *height* of a tree is the maximum number of edges from the root to a leaf

O(log n) ≤ height of n-node BST ≤ n-1



O(log n) ≤ height of n-node BST ≤ n-1



A tree is *balanced* if its height is O(log n)

How can a BST implement add?

- Follow contains until curr is about to become a null reference (until “about to fall off”)
- Replace that null with the new node

How can a BST implement add?

- Follow contains until curr is about to become a null reference (until “about to fall off”)
- Replace that null with the new node

What is the running time of a BST add?

A. O(1)
B. O(log n)
C. O(tree height)
D. O(n)
E. None of the above

How can a BST implement add?

- Follow contains until curr is about to become a null reference (until “about to fall off”)
- Replace that null with the new node

What is the running time of a BST add?

A. O(1)
B. O(log n)
C. O(tree height)
D. O(n)
E. None of the above

How can a BST implement remove?

- Begin by finding node with the key to remove
- Case 1: This node has 0 or 1 non-null children

How can a BST implement remove?

- Begin by finding node with the key to remove
- Case 1: This node has 0 or 1 non-null children
  - Remove node and replace it with child or null

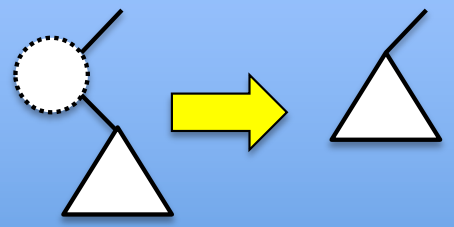
How can a BST implement remove?

- Begin by finding node with the key to remove

- Case 1: This node has 0 or 1 non-null children
  - Remove node and replace it with child or null

How long does it take to move a subtree whose parent is removed?

A. O(1)
B. O(log n)
C. O(n)
D. Not enough information
E. None of the above



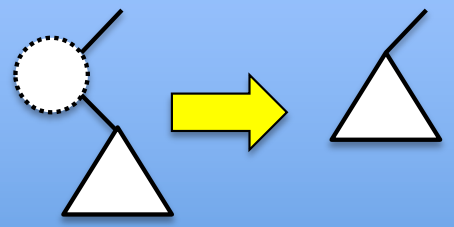
How can a BST implement remove?

- Begin by finding node with the key to remove

- Case 1: This node has 0 or 1 non-null children
  - Remove node and replace it with child or null

How long does it take to move a subtree whose parent is removed?

A. O(1)
B. O(log n)
C. O(n)
D. Not enough information
E. None of the above



How can a BST implement remove?

- Begin by finding node with the key to remove
- Case 1: This node has 0 or 1 non-null children
  - Remove node and replace it with child or null
- Case 2: This node has 2 non-null children

How can a BST implement remove?

- Begin by finding node with the key to remove
- Case 1: This node has 0 or 1 non-null children
  - Remove node and replace it with child or null
- Case 2: This node has 2 non-null children
  - Replace the key with its *in-order successor* (next key in sorted order)
  - Remove the successor

How can we advance a Node reference to the next key?

- Case 1: Node has right subtree
  - Go one step right and then left as far as possible

- Case 2: Right subtree is null
  - Go up tree until reaching a node whose left child is the ancestor of your starting place

    (requires parent references in the Node)
  - If you try to go up from the root, there is no next key

What is the running time of remove in a balanced BST?

A. $O(1)$

B. $O(\log n)$

C. $O(\log^2 n) = O((\log n)^2)$

D. $O(n)$

E. None of the above

What is the running time of remove in a balanced BST?

A. O(1)

B. O(log n)

C. $O(\log^2 n) = O((\log n)^2)$

D. O(n)

E. None of the above

Recall: Desired properties

- Ability to quickly “jump” later in the sequence of keys (like binary search)

- Ability to easily add keys into the appropriate position (like linked lists)