CSCB63 Tutorial, Week 3

Yufei Cui

January 21, 2019

1 Binary Trees

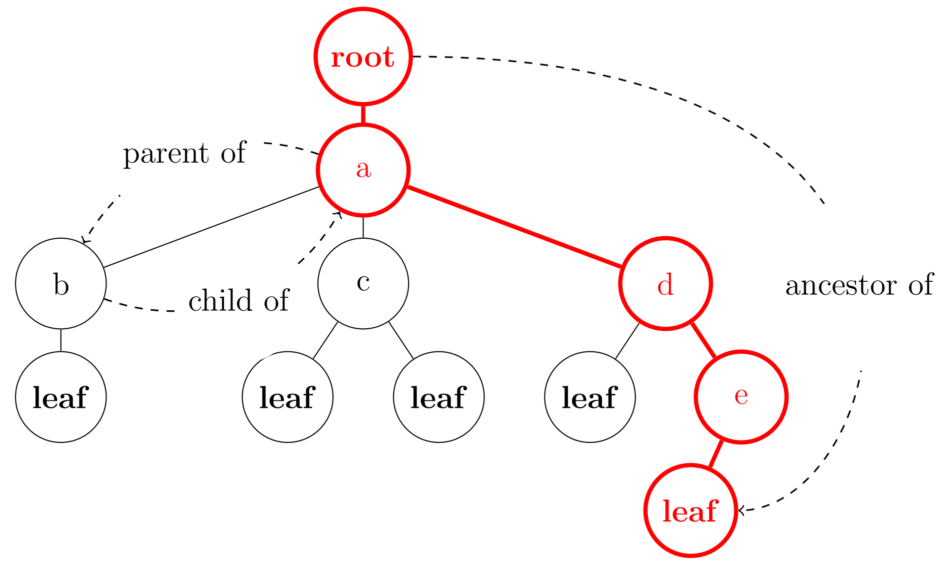
1.1 Terminologies[1]

Node:	Most basic unit in a tree that holds some sort of data.
Edge:	A connection between 2 nodes.
Parent/Child:	Node u is the parent to a child node v if there's an edge that directly connects u to v away from the root.
Tree:	Either empty or has a node that has 0 or more trees connected to it. (Notice the recursive definition here)
Root:	The top node on a non-empty tree.
Leaf:	Node with no children.
Binary Tree:	Tree with at most 2 children for every node called the left and right child.
Path:	A path from v_1 to v_n is a collection of nodes and edges $v_1, e_1, v_2, \ldots, e_{n-1}, v_n$ that starts from v_1 and ends at v_n where each v_i and v_{i+1} is connected by an edge e_i.
Ancestor/ Descendent:	Node v_1 is an ancestor to a descendent v_n if there's a path from v_1 to v_n where each node v_i is the parent of v_{i+1}.
Subtree:	Subtree S of a tree T is a tree that consists of a node in T and all of its descendants in T.
Height:	Number of edges on the longest path from the root to a leaf.[2]
Size:	Number of nodes in a tree. We will denote $\|T\|$.
Weight:	Size of the tree plus 1.

[1] `https://en.wikipedia.org/wiki/Tree_(data_structure)`

[2] This can change depending on convention.

Example 1.1. Terminologies is best understood with diagrams. It's okay if you skipped those definitions above, try to understand the following tree example.



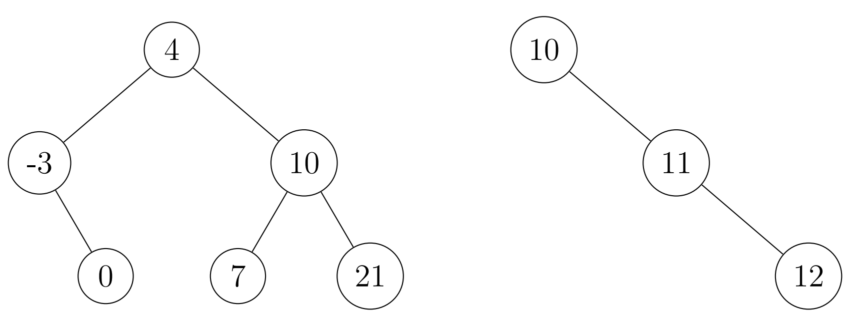
The longest path is highlighted in red with 4 edges. The size of the tree is 11, the weight is therefore 12. There are 5 leafs which are all descendants of nodes a and $root$.

1.2 Binary Search Trees

A binary search tree (BST) is a binary tree with the additional requirements:

- For each node u, it contains a *comparable key* that is greater than or equal to the keys of its left subtree and less than or equal to the keys of its right subtree.

Example 1.2.



Note. The min and max height of a BST T with n nodes is $\lfloor \log_2 n \rfloor$ and $n-1$, respectively.

$$\therefore height(T) \in \Omega(\log n) \ \wedge \ height(T) \in O(n). \tag{1}$$

1.3 AVL Trees

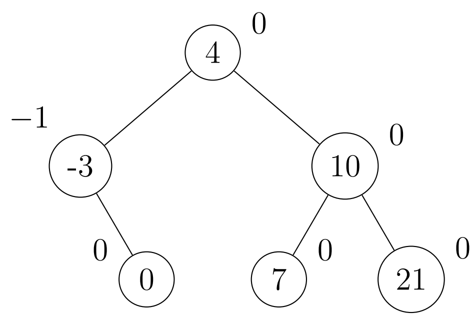
An AVL tree is a **BST** with the additional requirement:

- The *balance factor* of every node is either -1, 0, or 1

Definition. Let R and L be the right and left subtrees of u. The balance factor or $BF(u)$ is defined to be:

$$BF(u) = height(R) - height(L) \tag{2}$$

Example 1.3. This is the first tree from Example 1.2 with each node's balance factor's drawn.



1.4 Weight Balanced Trees

A weight balanced tree (WBT) is a **BST** with the additional requirement:

- For every node u:

$$3 \times weight(R) \leq weight(L) \quad \textbf{and} \quad weight(R) \leq 3 \times weight(L) \tag{3}$$

  where R and L are the right and left subtrees of u, respectively.

Note. Remember that $weight(T) = size(T) + 1$

Example 1.4. Both Trees from Example 1.2 are WBT's.

Exercise 1.1. Prove or disprove: AVL trees are also WBT's.

1.5 Rotations

I made a separate document focusing only on rotations. $\leftarrow$ Click