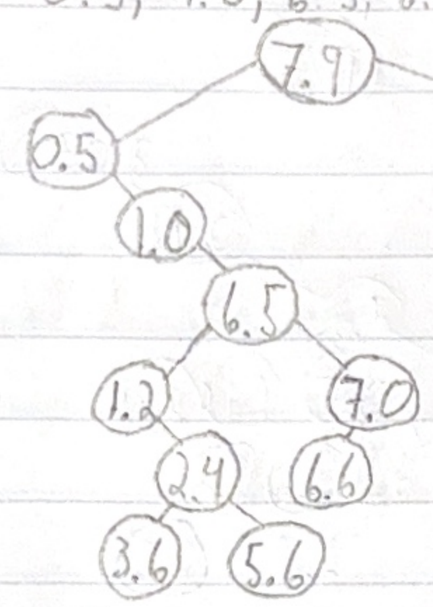
Assignment 6 - Trees and Heaps

Jeremiah McDonald
Algorithms
12 April 2025

Type of Tree

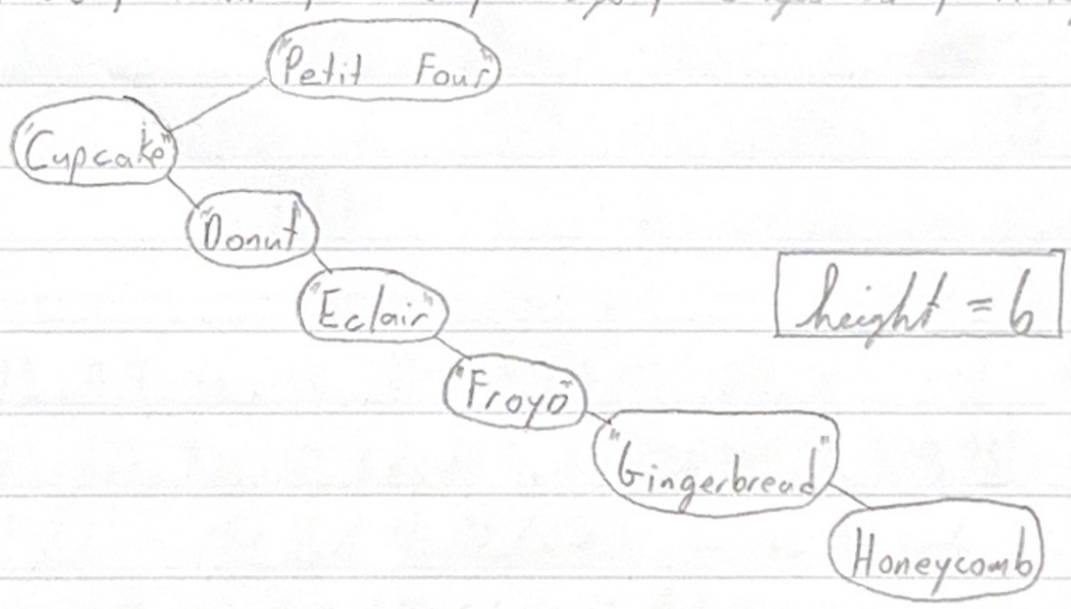
2. Given each of the following arrays, create a BST by adding each element of the array in the order it appears in the array. Afterwards, indicate what is the height of the resulting tree.

a.) [7.9, 0.5, 1.0, 6.5, 8.2, 7.0, 6.6, 9.9, 1.2, 2.4, 5.6, 3.6]



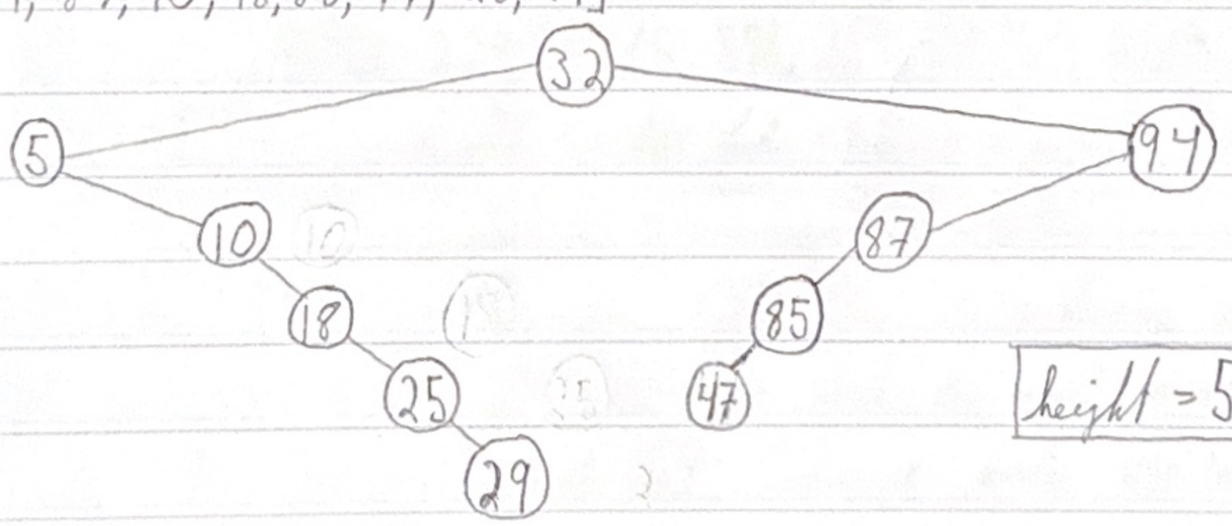
height = 6

b) ["Petit Four", "Cupcake", "Donut", "Eclair", "Froyo", "Gingerbread", "Honeycomb"]



height = 6

c.) [32, 5, 94, 87, 10, 18, 85, 47, 25, 29]

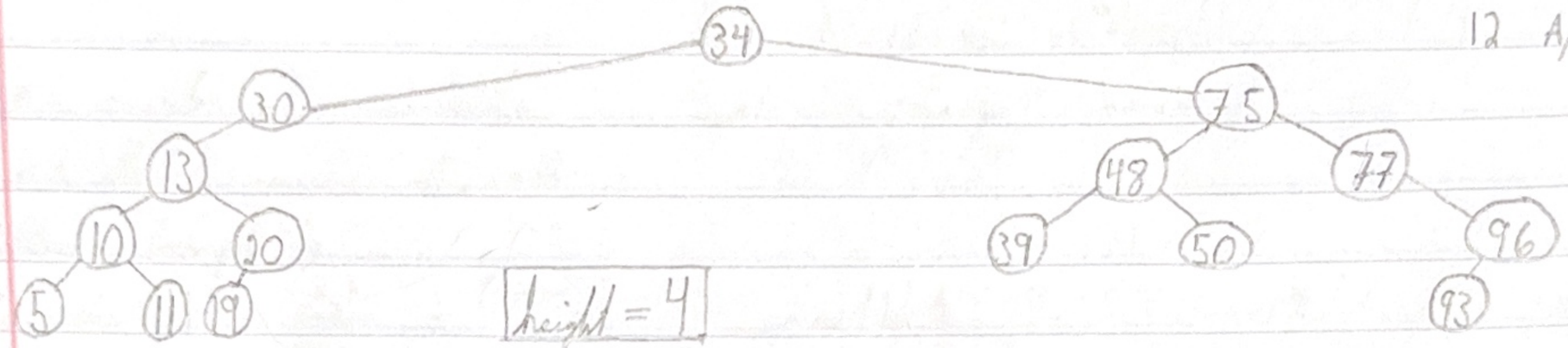


height = 5

Jeremiah McDonald
Algorithms
12 April 2025

1. ~~Type~~ of Tree continued

d.) [34, 30, 75, 77, 96, 48, 39, 50, 93, 13, 10, 5, 11, 20, 19]

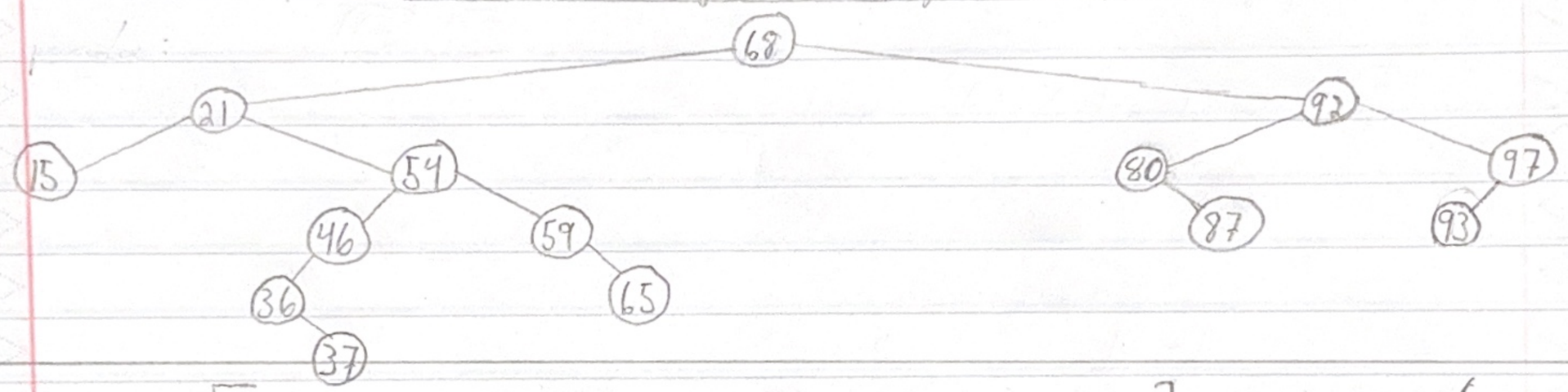


height = 4

2. BST Traversal: Given the following BST, in which the data on an empty node is 0.

a.) What will be the resulting tree after doing preorder and inorder traversal and applying the following operation on each node. Note round up the node values.

node.data += left.data + (right.data * 2)



Preorder: [68, 21, 15, 54, 46, 36, 37, 59, 65, 92, 80, 87, 97, 93] ← No operation (Only traversal)

▷ Preorder: [273, 144, 15, 218, 82, 110, 37, 189, 65, 366, 254, 87, 190, 93] ← Traverse Preorder and apply operation node.data += left.data + (right.data * 2), on each node.

Inorder: [15, 21, 36, 37, 46, 54, 59, 65, 68, 80, 87, 92, 93, 97] ← No operation (Only traversal)

▷ Inorder: [15, 144, 110, 37, 156, 328, 189, 65, 396, 254, 87, 540, 93, 190] ← Traverse Inorder and apply operation node.data += left.data + (right.data * 2) on each node

b.) Are the resulting trees BST's? No, a BST has an ordering property that any node's left subtree values ≤ the current nodes value and ≤ right subtree value. After traversal the structure did not change only the nodes values changed.

c.) An AVL tree is a BST with an additional height balance requirement. Since neither tree is a BST, they are also not an AVL tree

Jeremiah McDonald

5. Algorithm Analysis

Time Complexity: $O(pn)$
Space Complexity: $O(n+k)$

n = number of candidates
p = total number of vote
Time Complexity:
initializeCandidates : $O(n)$, + castVote : $O(n)$ + castRandomVote $O(n)$ +
getElection : $O(n)$ + getTopKCandidate $O(n+k)$ + audit Election $O(n) = O(pn+k)$

Space: heap[] = $O(n)$ + getTopKCandidates() standings $O(K)$ = $O(n+k)$