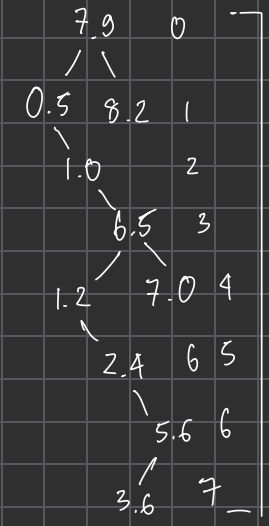
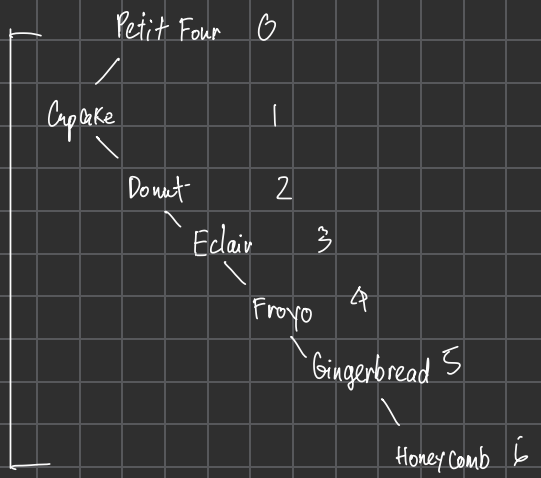
1.

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]



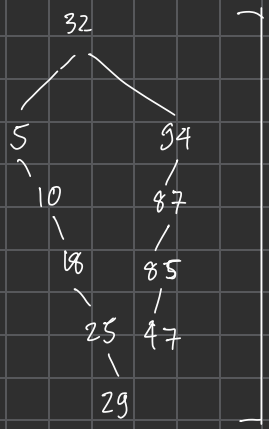
height = 7

b. ["Petit Four", "Cupcake", "Donut", "Eclair", "Froyo", "Gingerbread", "Honey Comb"]



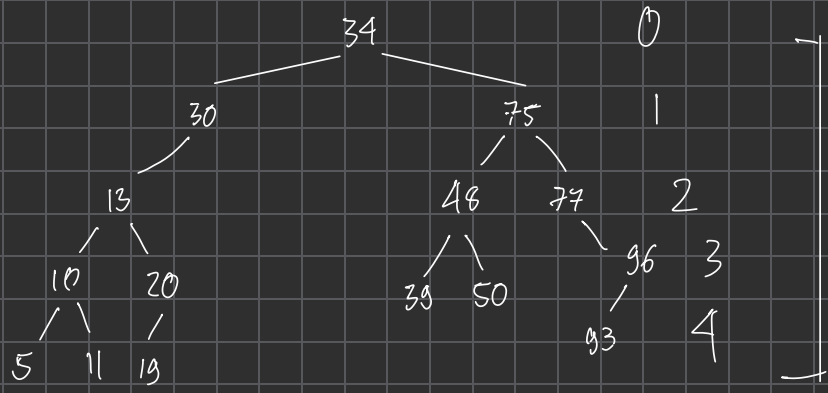
Height = 6

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



height = 5

d. [34, 30, 75, 77, 96, 48, 39, 50, 93, 13, 3, 5, 11, 20, 19]

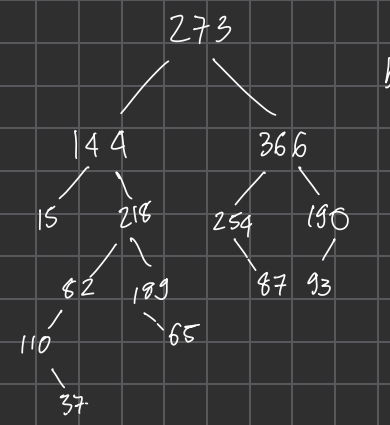


height = 4

2. First tree [68, 21, 15, 54, 46, 36, 37, 54, 46, 36, 37, 59, 65, 92, 80, 87, 97, 93]

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

a. 68 + 21 + (92 x 2) = 273
21 + 15 + (54 x 2) = 144
15 + 0 (null) + (0 x 2) (null) = 15
54 + 46 + (59 x 2) = 218
46 + 36 + (0 x 2) (null) = 82
36 + 0 + 74 = 110
37 + 0 (null) + (0 x 2) = 37
59 + 0 + 130 = 189
65 + 0 + 0 = 65
72 + 80 + 194 = 366
80 + 0 + 174 = 254
87 + 0 + 0 = 87
97 + 93 + 0 = 190
93 + 0 + 0 = 93



Preorder: [273, 144, 15, 218, 82, 110, 37, 189, 65, 366, 254, 87, 190, 93]

Inorder: [15, 144, 110, 37, 82, 218, 189, 65, 273, 254, 87, 366, 93, 190]

b. Not BST, because the node values are no longer in BST where Left < Root < Right
E.g.: Node 273 has left child 144 and right 366 ✓
But node 200 (left child of 218) has 189 as right child & 189 < 218

c. No because it's no longer a BST (AVL requires BST) and also the tree is unbalanced (Some subtrees are very deep)

5. For Election class, the time complexity is O(n log n) where n is the # of candidates. This is because we call updateMaxHeap in every method which has time complexity of O(n log n) → this method rebuild the heap every time there is a new entries. All other methods run O(n) so it's like O(n + n + ... n log n) → O(n log n)

Space is O(n) because voteMap and maxHeap store n in space (all of the candidates)

For ElectionSystem class:
- Time : O(n log n) where n is # candidates

- Space: O(n) because voteMape and MaxHeap store O(n)