Task 1

Proposition: Every binary tree on n nodes where each has either zero or two children has precisely n+1/2 leaves.

Base case

A tree with 1 leaf has 1 node altogether. A tree with 2 leaves has 3 nodes altogether.

When L=1,N=1 and L=2, N=3. (L is leaf and N is node).

Strong Inductive hypothesis

Let's assume that any full binary tree with L leaves has N=2L−1 total nodes for all 0≤L≤K.

Inductive step

Let T be a full binary tree with K+1 leaves.
Seeing how T is a full binary tree, its subtrees

$T_{left}$and T_{right}must have at least 1 leaf each.

This means $T_{left}$and T_{right}have a number of leaves <K+1. Call this number

$L_{left}$and L_{right}respectively.

By our strong IH, they must have
$2(L_{left})-1$ and $2(L_{right})-1$ total nodes respectively.

T(tree) therefore has a number of nodes equal to the sum of the number of nodes in each subtree plus its 'root' which appears not in either subtree.

T's subtree is an internal node and not a leaf, the number of leaves in

$T=L_{left}+L_{right}$.

$2(L_{left})-1+2(L_{right})-1+1=2(L_{left}+L_{right})-2+1=2(K+1)-1=2(L+1)-1$

Thus proving by induction that all full binary trees with L leaves have

N=2L−1 total nodes for all L, L=$\frac{n+1}{2}$.

Task 3.2

Subtask II: You will also analyze the running time of this algorithm and argue why the tree generated meets the depth requirement.

Analyze the running time

Total number of nodes in a tree:

$$n = 2^{d+1} - 1$$

Solving for d, we get:

$$n+1 = 2^{d+1}$$

$$log_2(n + 1)=log_2(2^{d+1})$$

$$log_2(n + 1)=d + 1(log_2 2)$$

$$log_2(n + 1) = d + 1$$

$$d=log_2(n + 1)\text{-}1$$

$$d=log_2 n$$

The depth of a full binary search tree with 16 nodes is 4. In other words, the depth of a binary search tree with n nodes must be at least then log(n) and so the running time of the find, insert and delete algorithms can be at least log(n).

A full binary search tree is said to be balanced because every node's proper ‘descendants’ are divided evenly between its left and right subtrees.

Thus, the search algorithm employed by the find, insert and delete operations perform like a binary search.

Why the tree generated meets the depth requirement

When the tree becomes unbalanced the worst case is O(n), where n is the number of nodes in the tree. For example, the number of comparisons needed to find the node marked A in the binary search trees below is 3, which is equal to the number of nodes or the depth requirement.