Binary Trees

A binary tree is a tree in which every node has at most two children.

A binary search tree is a binary tree with the additional requirement that for each node the values in the left subtree are smaller than the node’s value and the values in the right sub-tree are greater than the node’s value.

A leaf node is a node that has no children.

A internal node is a node that is not a leaf node.

The height of a node is the longest path (i.e. number of edges) from the node to a leaf node.

The height of a binary tree is the height of the root node.

Height of a Binary Tree is $O(\log n)$

We showed this for a special type of binary tree called perfect binary tree. A perfect binary tree is binary tree in which all internal nodes have exactly two children and all leaves are at the same level.

Let $n$ be the number of nodes in a perfect binary tree and let $l_k$ denote the number of nodes on level $k$. Note that:

- $l_k = 2l_{k-1}$, i.e. each level has exactly twice as many nodes as the previous level (since each internal node has exactly two children)
- $l_0 = 1$, i.e. on the “first level” we have only one node (the root node).
- the leaves are at the last level, $l_h$, where $h$ is the height of the tree.

The total number of nodes, $n$, in the tree is equal to the sum of the nodes on all the levels:

$$1 + 2^1 + 2^2 + 2^3 + ... + 2^h = n$$

From CS 201 we know that:

$$1 + 2^1 + 2^2 + 2^3 + ... + 2^h = 2^{h+1} - 1$$

Therefore:

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

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

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

$$(h + 1) \log_2 2 = \log_2 (n + 1)$$

$$h + 1 = \log_2 (n + 1)$$

$$h = \log_2 (n + 1) - 1$$

Therefore $h$ is $O(\log n)$
Now that we know the height of the tree we can compute the number of leaves, \( l_h \), in the tree. We observed earlier that \( l_h = 2^h \) so we can substitute the value of \( h \) in this expressions:

\[
2^h = 2^{\log_2(n+1)-1} = 2^{\log_2(n+1)}/2 = (n + 1)/2
\]

In the above expression we used the fact that \( a^{b-c} = a^b/a^c \) and \( a^{\log_a b} = b \).

In summary, in a perfect binary tree with \( n \) nodes:

- the height is \( h = \log_2(n + 1) - 1 \), i.e. \( h \) is \( O(\log n) \)
- the number of leaves is \( l_h = (n + 1)/2 \), i.e. roughly half of the nodes are at the leaves.

Examples of Recursive Methods

Adding the values of the nodes in a binary tree:

```plaintext
procedure ADD(root):
    if root is nil:
        return 0
    else:
        s1 = ADD(left[root])
        s2 = ADD(right[root])
        return data[root] + s1 + s2
```

Calculating the height of the tree (for empty tree defined height to be -1):

```plaintext
procedure HEIGHT(root):
    if root is nil:
        return -1
    else:
        h1 = HEIGHT(left[root])
        h2 = HEIGHT(right[root])
        return 1 + MAX(h1, h2)
```