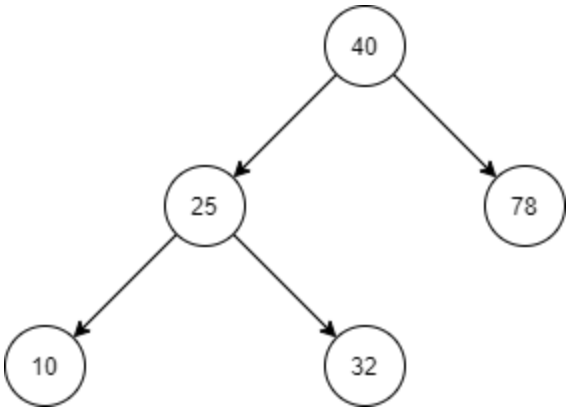
Tree Abstract Data Structure

On Blackboard you can find the code for a binary tree. Binary trees are typically implemented by creating a single class, which represents a node in the tree, and contains two pointers to its child nodes.



Tree data structures can be traversed in two ways: Depth-First or Breadth-First. When traversing a tree in a depth-first fashion, we again have three options: pre-order, in-order and post-order traversal.

If we use these traversal techniques on the three shown above, we get the following results:

- Pre-Order: 40, 25, 10, 32, 78
- In-Order: 10, 25, 32, 40, 78
- Post-Order: 10, 32, 25, 78, 40
- Breadth-First: 40, 25, 78, 10, 32

Code

1. First, look up how every tree traversal method works exactly.
2. For every tree traversal method (Pre-Order, In-Order, Post-Order and Breadth-First), implement some unit tests.
3. For every tree-traversal method, implement a method that takes a binary tree as an argument, and returns a vector of all data elements from the nodes you visited.
4. Implement a function that determines the height of a binary tree. The height of a binary tree is the number of edges in the longest possible path from the root of the tree to a leaf node.
5. Implement a function that determines the depth of a node in a binary tree. The depth of a node in a binary tree is the number of edges between this node and the root of the tree.

Report

1. We can use a binary tree to represent a mathematical expression, which traversal method should we use to evaluate the expression? Why can't we use the others?
2. If you assume a sorted binary tree (Such as the one shown in the figure), what is the time-complexity of finding a node with a given value?
3. We can also use binary trees to build N-ary trees, how would you tackle this?