**CIS 263 Assignment 3**

By: Jack Lukomski

I started out by creating two binary trees, one being a valid binary tree, and the other one not. The valid tree was created using a BinaryTree class that I created and using the insert method (the method takes in one parameter, data, and creates, and inserts the node in the correct position). The invalid tree utilized the Node class I created, and I manually inserted the nodes to the left and right (if I used my BinaryTree class, the binary tree would be valid). For readability, the invalid binary tree is inverted, meaning that values greater than the parent are on the left, and values less than the parent are on the right. I did this because when printed using the inorder tree walk method, the invalid tree will print the values in reverse order.

A diagram of a diagram

Description automatically generated

*Figure 1: The valid binary tree*

A diagram of a flowchart

Description automatically generated

*Figure 2: The invalid binary tree*

After creating both binary trees, I wrote a recursive algorithm that runs in linear time that returns true if the binary tree supports the search order property, and false if it does not. I then wrote a quick test routine that prints a binary tree and tells if it follows the search order property. Obviously, the valid tree passed the test, and the invalid tree did not.

A screenshot of a computer

Description automatically generated

*Figure 3: The test routine*

bool isValidTree(Node \*n)

{

return checkOrderProperty(n, INT\_MIN, INT\_MAX);

}

bool checkOrderProperty(Node \*curr, int min, int max)

{

if (curr == nullptr) {

return true;

}

if (curr->getData() < min || curr->getData() >= max) {

return false;

}

return (checkOrderProperty(curr->getLeft(), min, curr->getData()) && checkOrderProperty(curr->getRight(), curr->getData(), max));

}

*Figure 4: The recursive order property check algorithm*

As seen in *figure 4,* this is the , recursive algorithm to check if the order tree property is followed. It works by having the base case that returns true if the current node is a null pointer, if the current node’s data is less than the minimum or the current node’s data is more than the max, the algorithm returns false. The recursive call ANDS two calls to the function together, one passing the left node, the min, and the current nodes data as the max parameter, the other call passes the right node, the current nodes data as the min parameter, and the max. Once one of the two base cases are met, the function terminates, and returns either true or false. This runs in linear time because each node in the binary tree is only being accessed once.