Binary Search Tree Product Documentation

v1.1 (2/25/2019)

# Introduction

This program is a sample implementation of a self-balancing binary search tree (BST) (using the AVL method) using generic programming in C++. It will store a value in the node, and each node keeps track of its parent node, its left and right child nodes, and its height in the BST.

A binary search tree is a data structure that stores values in a hierarchy based on value comparison. As new values are added to the tree, the tree sorts these values by comparing the new value to values already present in the tree. Each value is stored in its own node, and each node can have no more than two children: a left child, and a right child. The value stored a node’s left child is always less than the value stored in the initial node; likewise, the value stored in a node’s right child is always greater than the value stored in the initial node.

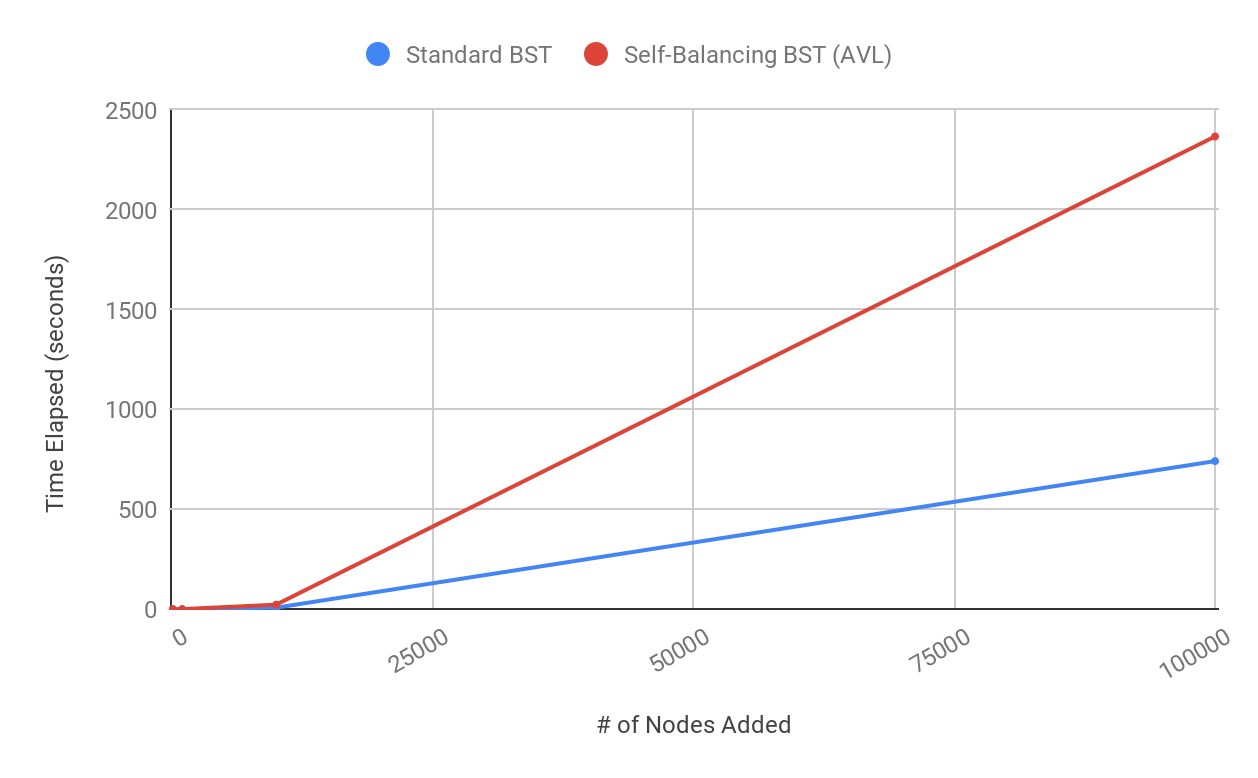
The user interacts with the program through a simple console-based menu. They are able to perform the following functions:

* **Insert** a new node into the tree
* **Find** a node containing a key value matching a value of the user’s choosing.
* Display the lowest (**minimum**) key value stored in the BST.
* Display the highest (**maximum**) key value stored in the BST.
* **Display** **detailed information about a node** containing a key value matching a value of the user’s choosing.
* **Display** a visual representation of the BST.
* **Delete** a node from the BST that has a key value matching a value of the user’s choosing. T(he program uses the Find function to locate this node).
* **Delete all** the nodes contained in the BST.
* **Auto-populate** the BST with a sequence values manually entered by the developer. (These values were used for manually testing the BST.)
* **Auto-populate** a specified number of nodes with a sequenced value starting from zero. These functions will also delete any nodes already contained in the BST.
* **Exit** the program.

# Code Analysis

This code is very poorly optimized when dealing with large data sets. As you can see from the included data tables and chart, the tree does not meet the expected complexity of O(log n). This is likely due to the fact that the tree will always attempt to balance itself if it has a height of at least 2, and each attempt to balance itself checks from the newly created node, all the way up to the root of the tree. In addition, each check for a node’s balance also checks the height of each of the node’s subtrees.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **# of Nodes Added** | **Standard BST** | | |  | **AVL BST** | | |
| **Clicks** | **Seconds** | **Minutes** | **Clicks** | **Seconds** | **Minutes** |
| **100** | 0 | 0 | 0 | 3 | 0.003 | 0 |
| **1000** | 68 | 0.068 | 0 | 220 | 0.22 | 0 |
| **10000** | 7400 | 7.4 | 0.12 | 22486 | 22.486 | 0.37 |
| **100000** | 741369 | 741.369 | 12.36 | 741369 | 741.369 | 39.45 |



# Programmer’s Guide

## <BSNode.h>

template <typename Type> class BSNode

A custom-built node class for use with <BSTree.h>. The class is templated as to allow for use with a wide variety of data types.

|  |  |  |
| --- | --- | --- |
| Attributes/Data Members (private) | | |
| Name | Type | Description |
| value | Type | The value held by this node. Initializes to NULL. |
| height | int | The height this node resides at in the tree. Initializes to -1. |
| parent | BSNode<Type>\* | A pointer to the this node’s parent node. Initializes to nullptr. |
| leftChild | BSNode<Type>\* | A pointer to the this node’s left child node. Initializes to nullptr. |
| rightChild | BSNode<Type>\* | A pointer to the this node’s right child node. node. Initializes to nullptr. |

|  |  |  |  |
| --- | --- | --- | --- |
| Member Functions (private) | | | |
| Signature | |  | |
| Function Name | Return Type | Description & Parameters | |
| SetHeight | void | Changes the height this node resides at in the tree. Also recursively adjusts the height of any child nodes under this node.  int newHeight - The new height at which this node resides in the tree. | |

### 

|  |  |  |  |
| --- | --- | --- | --- |
| Member Functions (public) | | | |
| Signature | |  | |
| Function Name | Return Type | Description & Parameters | |
| ~BSNode | (Desctructor) | Default destructor. | |
| BSNode | (Constructor) | Default constructor. | |
| BSNode | (Constructor) | Constructor with an initialized value.  Type initValue - The initial value to be stored in this node. | |
| GetHeight | int | Returns the height this node resides at in the tree. | |
| GetLeftChild | BSNode<Type>\* | Returns a pointer to the node that is this node's left child node. | |
| GetParent | BSNode<Type>\* | Returns a pointer to this node’s parent node. | |
| GetRightChild | BSNode<Type>\* | Returns a pointer to the node that is this node's right child node. | |
| GetValue | Type | Returns the key value stored in this node. | |
| SetHeight | void | Changes the height this node resides at in the tree.  int newHeight - The new height at which this node resides in the tree. | |
| SetLeftChild | void | Changes which node is this node's left child node. Also adjusts the child's height.  BSNode<Type>\* newPtr - The pointer to be set as this node’s left child node. | |
| SetParent | void | Changes which node is this node's parent node. Also adjusts the node's height value.  BSNode<Type>\* newPtr - The pointer to be set as this node’s parent node. | |
| SetRightChild | void | Changes which node is this node's right child node. Also adjusts the child's height.  BSNode<Type>\* newPtr - The pointer to be set as this node’s right child node. | |
| SetValue | void | Changes the value stored in this node.  Type newValue - The new value to be stored in this node. | |

### 

## <BSTree.h>

template <typename Type> class BSTree

A custom-built binary search tree class for use with <BSNode.h>. The class is templated as to allow for use with a wide variety of data types.

|  |  |  |
| --- | --- | --- |
| Attributes/Data Members (private) | | |
| Name | Type | Description |
| root | BSNode<Type>\* | A pointer to the root (first) node of the tree. Initializes to nullptr. |

|  |  |  |  |
| --- | --- | --- | --- |
| Member Functions (private) | | | |
| Signature | |  | |
| Function Name | Return Type | Description & Parameters | |
| BalanceTree | void | Attempts to balance the tree (or subtree), starting from the node passed into the function.BSNode<Type>\* nodeToCheck - The node from which to check the tree's balance. | |
| CheckTreeBalance | int | Checks the balance of the branches beneath the node passed into the function.Returns the difference between the heights of the left and right branches below the node being checked.  BSNode<Type>\* nodeToCheck - The node at which the tree's (or subtree's) balance is checked. | |
| GetTreeHeight | int | Checks for the height of the tree (or subtree), by running a depth-first search starting at the node passed into the function. Returns the largest height value in the tree (or subtree).  BSNode<Type>\* startingNode - The node from which to begin the height check. | |
| MaxNode | BSNode<Type>\* | Finds the node in the tree (or subtree) containing the maximum (largest) value. Returns a pointer to the node containing the maximum value stored in the tree (or subtree).  BSNode<Type>\* startNode - The node from which to start the search. | |
| MinNode | BSNode<Type>\* | Finds the node in the tree (or subtree) containing the minimum (smallest) value. Returns a pointer to the node containing the minimum value stored in the tree (or subtree).  BSNode<Type>\* startNode - The node from which to start the search. | |
| PrintNode | void | Prints the contents of a node to an output file when called by Print().  BSNode<Type>\* node - The node whose value will be printed to the console.  int space - The number of spaces to be inserted before printing the node's value. An call to this function from outside PrintNode should use the SPACE\_COUNT global value in this file.  bool verbose - Whether or not to output the node’s height along with it’s key value. Defaults to false. | |
| RotateLeft | BSNode<Type>\* | Rotates the provided node to the to the left subtree. Returns the node that was rotated into the original node's position.  BSNode<Type>\* rotateNode - The node to be rotated to the left. | |
| RotateRight | BSNode<Type>\* | Rotates the provided node to the to the right subtree. Returns the node that was rotated into the original node's position.  BSNode<Type>\* rotateNode - The node to be rotated to the right. | |
| SwapValues | void | Swaps two key values between two different nodes.  BSNode<Type>\* node1 - A node that will be switching key values with node2.  BSNode<Type>\* node2 - A node that will be switching key values with node1. | |

|  |  |  |  |
| --- | --- | --- | --- |
| Member Functions (public) | | | |
| Signature | |  | |
| Function Name | Return Type | Description & Parameters | |
| ~BsTree | (Destructor) | Default destructor. | |
| BSTree | (Constructor) | Default constructor. | |
|  |  |  | |
| GetRoot | BSNode<Type>\* | The pointer to the root node of this tree. Returns a nullptr if there is no root. | |
| Delete | void | Deletes a node from the tree.  Type delValue - The key value of the node to be deleted.  bool verbose - Whether or not to output status messages to the console. Defaults to true. | |
| DeleteAll | void | Deletes all the nodes from the tree.  bool verbose - Whether or not to output status messages to the console. Defaults to true. | |
| Find | BSNode<Type>\* | Attempts to find a node containing the given value. If the value is found, returns a pointer to the node object where the value is stored. Otherwise, returns a nullptr.  Type valueToFind - The value to attempt to find in the tree.  bool verbose - Whether or not to output status messages to the console. Defaults to true. | |
| IsEmpty | bool | Determines whether or not the tree is empty. If the tree is empty, returns true. Otherwise, returns false.  bool verbose - Whether or not to output status messages to the console. Defaults to true. | |
| Insert | void | Inserts a value into the tree.  int newValue - The new value to be inserted into the tree.  bool verbose - Whether or not to output status messages to the console. Defaults to true. | |
| Maximum | Type | Finds the maximum (largest) value stored in the tree.  bool verbose - Whether or not to output status messages to the console. Defaults to true. | |
| Minimum | Type | Finds the minimum (smallest) value stored in the tree.  bool verbose - Whether or not to output status messages to the console. Defaults to true. | |
| Print | void | Prints the tree to an output file for visual debugging. | |