Brian Engel

Binary Search Tree Pseudocode

This project was to use a binary search tree to insert, delete, search, and print all nodes for Bids. Personally, I found the binary search tree much easier to deal with than a hash table. Although there were still quite a few pointers to keep track of, at least there weren’t vectors and linked lists to keep track of at the same time.

Void Insert(Bid bid) {

Create newNode with bid

If bst root is null {

root = newNode

}

Else {

current = root

while (current != null) {

if (newNode bidId < current bidId) {

if (current left = null) {

current left = newNode

current = null

}

else {

current = current left;

}

}

else {

if (current right = null) {

current right = newNode

current = null

}

else {

current = current->right

}

}

}

}

}

void Remove(string bidId) {

parent = null

current = root

while (current != null {

if (current bidId = bidId) {

if (current left = null & current right = null) { // no children

if (parent = null) {

root = null

}

else if (parent left = current) {

Parent left = null

}

else {

Parent right = null

}

}

else if (current right = null) { // left child only

if (parent = null) {

root = current left

}

else if (parent left = current) {

Parent left = current left

}

else {

Parent right = current left

}

}

else if (current->left == nullptr) { // right child only

if (parent = null) {

root = current right

}

else if (parent left = current) {

Parent left = current right

}

else {

Parent right = current right

}

}

else { // 2 children

successor = current right

while (successor left != null) {

successor = successor left

}

temp = successor bid

Remove(successor bidId)

Current bid = temp

}

return;

}

// Search right

else if (current bidId < bidId) {

parent = current

current = current right

}

// Search left

else {

parent = current

current = current left

}

}

// Node not found

return;

}

Bid Search(string bidId) {

current = root

while (current != null) {

if (bidId = current bidId) {

return current bid

}

// if bid is smaller than current node then traverse left

else if (bidId < current bidId) {

current = current left

}

// else larger so traverse right

else {

current = current right

}

}

// not found return empty bid

return empty bid

}

void PrintInOrder(Node\* node) {

if (node != nullptr) {

PrintInOrder(node left);

//output bidID, title, amount, fund

displayBid(node bid);

PrintInOrder(node right);

}

}