

Programming Assignment 2

Modify the BST-based dictionary to be a Threaded BST-based dictionary.

Files You're Given

- BinNode.h
- book.h
- BSTNode.h
- BST.h
- dictionary.h

Steps (1:5)

1. Make sure that you can compile and run the files you've been given. You'll have to create a .cpp file with main() and create a BST object.
2. As part of your “approach” determination, determine what objects must be modified to create a threaded-BST.
 - BinNode – what change do we need to make to BinNode relative to threads?
 - BSTNode implements BinNode – what changes are required?

Steps (2:5) - Approach (cont.)

- BST.h – this is where most of the work is done
- Focus on inserthelp() first, since it must work or nothing else will
 - You can use recursion here
 - Question – when I insert a new node, what is the state of the “root” I pass into inserthelp and how does that affect the insert?
 - Is “root” empty, in which case I merely append my new node to the left or right side as appropriate?
 - Is one of the “root” pointers a regular pointer or a thread and is it the side I need to insert on?
 - Figure out the possible states of “root” and then determine what action you need to take to do an insert when root is in that state

Steps (3:5)

- inserthelp example

1. “root” is 79 and “newNode” (78) is less than root

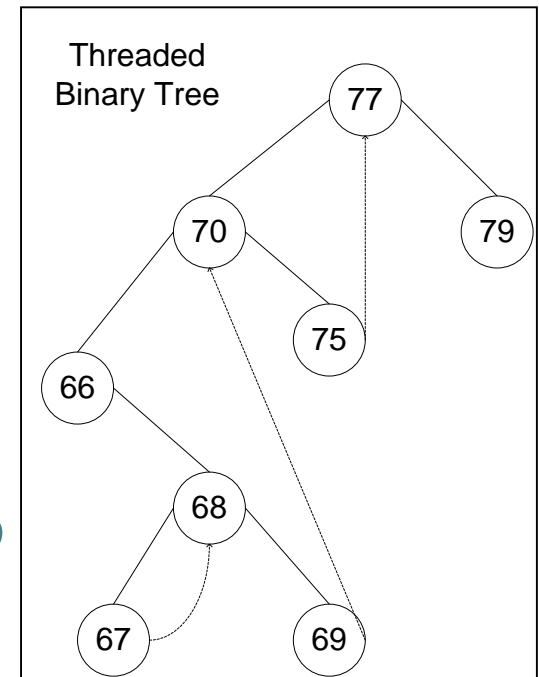
- a) root->setleft(newNode, regular)

- b) newNode->setright(root, thread)

2. “root” (69) has a right thread, no children, and “newNode” (69.5) is greater than root

- a) newNode->setright(root->right(), thread)

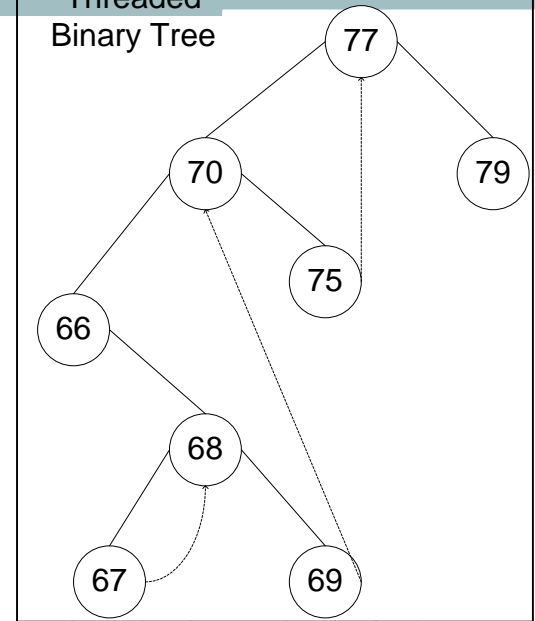
- b) root->setright(newNode, regular)



Steps (4:5)

- `inorderPrint()`
 - Do this next, so that you can print your tree using threads
 - Rules
 - Start at root and use a loop to go as far down the left side of the tree as you can following **regular** pointers. You are now at the smallest node – print it
 - Now follow the node's right pointer until you can go left again using the following rules
 - If following a **thread** don't go left even if the left subtree is not empty
 - If following a **regular** pointer go left if the left subtree is not empty
- `reverseOrder()` – rules exact opposite of `inorderPrint()`

Threaded
Binary Tree



Steps (5:5)

- `printhelp()`
 - Uses recursion and threads break its current implementation
 - Your last task is to modify it so that it will continue to work as designed and ignore threads

Programming Assignment Tip

- You can create a text file of your programs output using the “>” redirect operator rather than taking a lot of screen shots to do the same thing.
- You still need to turn in a screen shot so that I can see you ran your program but instead of more screen shots you can include your program’s output.
- Your program must run or you won’t get credit for it.