Subject: Find BST
Posted by cdyancey on Sat, 25 Feb 2017 22:11:00 GMT

Concerning the following functions:

```c
int findBST(bst *tree, void *value); //returns 1 if found, 0 otherwise
bstNode *findBSTNode(bst *tree, void *value); //returns 0 if not found
```

Are these essentially the same operation, only with different return types? If so, wouldn't (int) findBST be redundant, since you could just run findBSTNode and test for a NULL value?

Also, it seems like it would be a good strategy then to write findBSTNode first, then implement findBST as a call to findBSTNode that returns 0 if findBSTNode returns a NULL value and 1 otherwise.

Am I wrong?

---

Subject: Re: Find BST
Posted by davidmccoy on Sun, 26 Feb 2017 23:27:43 GMT

It looks like they would both use a helper function that found a node with the desired value (or null if not found), and then for findBST return an 1 if a node was found or 0 if not, and for findBSTNode just return the node pointer found.

---

Subject: Re: Find BST
Posted by nltollman on Mon, 27 Feb 2017 05:11:40 GMT

I don't see the point of having the parameter of findBSTNode to be the BST itself rather than a BSTNode. If you have the parameter as BST, you would need to create a private function that's recursively called to traverse down the tree and find the node. However, if you have the parameter as a BSTNode, you wouldn't need a private function at all. You would simply have that function itself, findBSTNode, become a recursive function.

---

Subject: Re: Find BST
Posted by lusth on Mon, 27 Feb 2017 20:26:00 GMT

findBSTNode is meant for the application (vbst and rbt) and those applications won't know the internals of a bst (such was what the root pointer is called).