2020/04/15 - Binary Search Trees

14 Tháng Tư 2020 7:08 CH

Synopsis

- We're almost at the end of the semester!
- Two labs left, both involving BST [Binary Search Tree].

Getting Started

- Run the following commands:

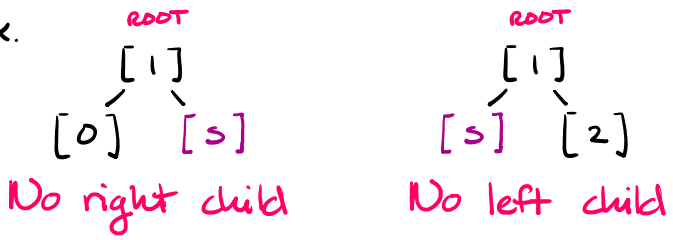
```
mkdir obj bin
cp -r ~jplank/cs140/Notes/Trees/include .
cp -r ~jplank/cs140/Notes/Trees/src .
cp ~jplank/cs140/Notes/Trees/makefile .
```

- This lab is not so bad... but it may catch you off guard a few.

The BST "Rules"

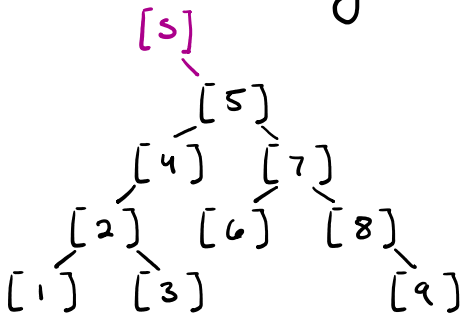
- Every tree has a sentinel node, just like linked lists do. Below, I notate them as [S].
- Tree root is sentinel->right. If the tree is empty, then sentinel->right = sentinel.
- Nodes that don't have a specific child will have pointers to the sentinel node.

(Ex.



- DEPTH: How deep a node is in the tree.
Can be found by going up to the sentinel node.

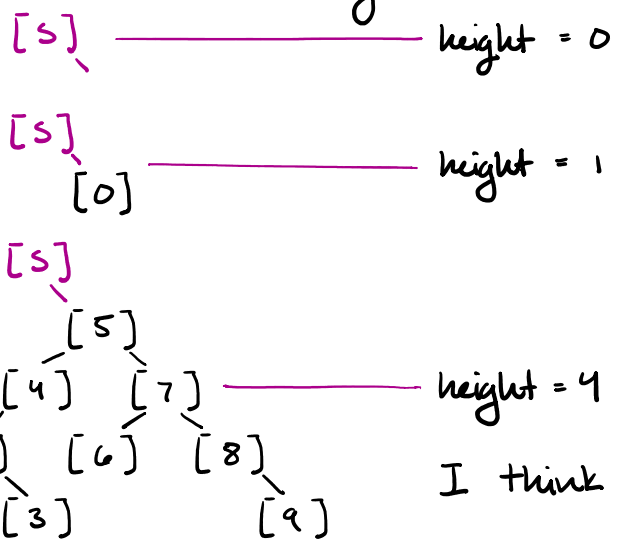
(Ex. Assume following tree:



Depth(5) = 0
Depth(4) = 1
Depth(6) = 2
Depth(9) = 3

- HEIGHT: Maximum depth of the tree + 1.
If there are no nodes, height is 0.
Recursion makes this trivial.

(Ex. Assume the following trees:



I think you get the idea...

Submission Command

- tar -cvf labA.tar src/bstree_lab.cpp

Suggested Order

- Follow the gradescript for proper implementation steps. 30-100 generally are "All or Nothing".

- Depth — GS 1 - 10
- Height — GS 11 - 19 (Not 20)
  - recursive_find_height — Recursive Helper
- Ordered_keys — GS 20 - 29 (Not 30)
  - make_key_vector — Recursive Helper
- BSTree::BSTree(const BSTree &) — GS 41-50
  - operator = — GS 30 - 40
    - make_balanced_tree — Recursive Helper

Copy Constructor "Trick"

- C+P the orginal constructor in btree_notes.cpp from lines 12 → 18 (the entire thing).
- Manually call operator=(). C++ will let you do this... seriously:

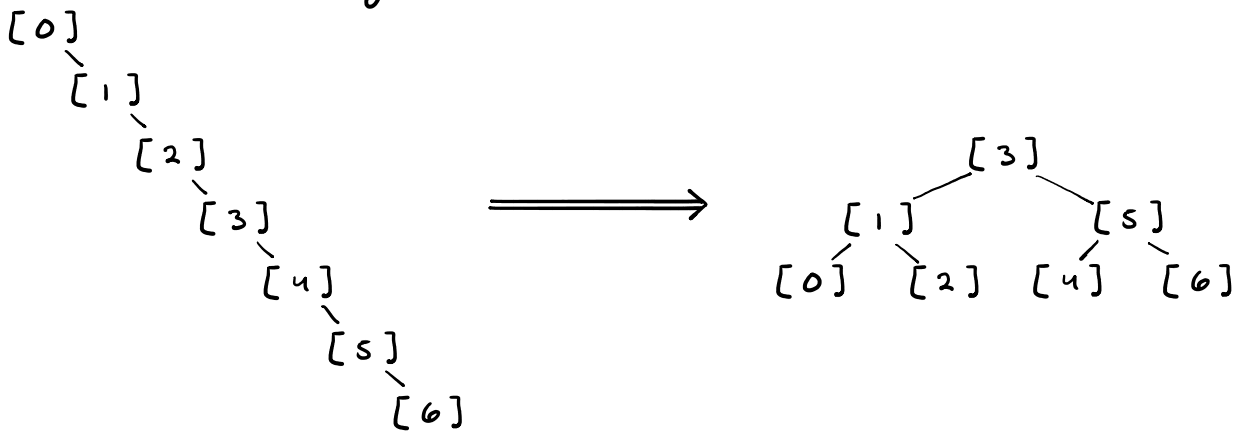
```
operator=(t);  //Invoke operator on "t"
```

OR

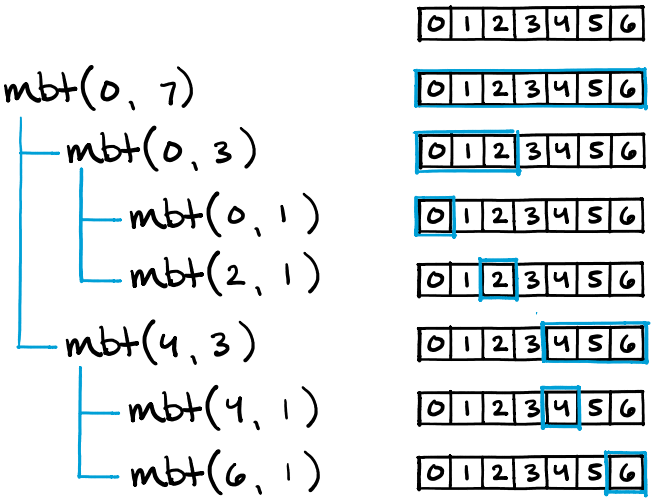
```
*this = t;  //Deref self-ptr and assign.
```

MORE ON "MAKE_BALANCED_TREE"

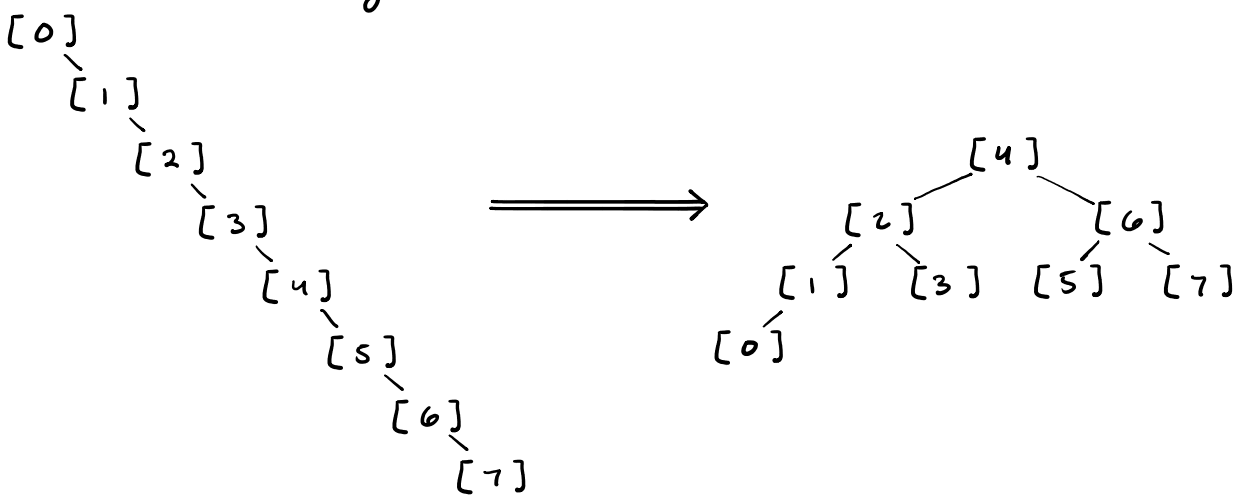
- Assume the following tree



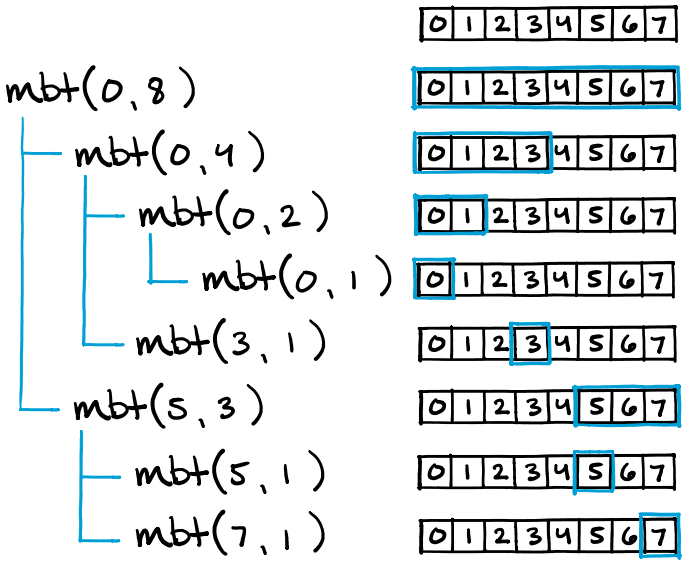
Recursive Order (mbt = make_balanced_tree)



- Assume the following tree



Recursive Order (mbt = make_balanced_tree)



- Try to make your program match function calls as shown above.

- You are making BSTNodes manually. Pay attention to the fact this function is const. You can't call Insert. Yet, this function returns BSTNode *?

Recursion Pseudocode

```
left_child = right_child = sentinel
n = num_indices

if (n > 1) {
    left_child = make_balanced_tree(..)

    if we can split right
        right_child = make_balanced_tree(..)
}
```

- make new BSTNode "i".

  i -> left = left_child
  i -> right = right_child
  i -> parent = sentinal
  i -> key and value, y'know.

- Also set "parent" of left and right children to i.

  If we are in the first call (just check if n = sorted_keys.size() or something), then set sentinel -> right = i.

- return i.