# 2020/10/28 - Binary Search Tree (Part 1)

28 Tháng Mười 2020 7:08 SA

## SYNOPSIS

- Go over Lab 6.

## LAB 6

- As always, on Canvas, go to the "Lab 6.1" assignment and read the "lab6.html" file attached. All lab details will be there.

- There are three parts split into two submissions. This will work in your favour in the long run.

- There is template code... as a TXT file. You are not writing a CPP this time... but a BST.h to interact with 3 given cpp files. Not bad.

- This is one of few lab assignments where you submit more than just code. In addition, send 3 drawings:

  1. Illustrate bst::iterator::operator++, which is inorder traversal. The tricky part is that this is a single step. Thus, think iteratively, not recursively.

  2. Illustrate bst::lower_bound.

  3. Illustrate bst::upper_bound.

- Submit those drawings outside the tar file. Submit the BST.h inside the tar. FACE DEDUCTIONS OTHERWISE...

## SUBMISSION COMMAND

LAB 6.1

```
tar -cvf lab6_1.tar BST.h
```

LAB 6.2

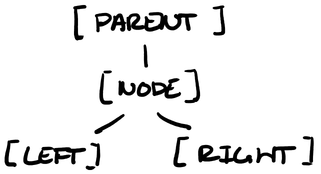
```
tar -cvf lab6_2.tar BST.h
```

## BST.h - PART 1

- Copy BST.txt to BST.h. We got some work to do...

★ BST::NODE

- Add a node *parent. This will used to point to a node above the current one.



- Add an int id. This will store a unique id for each node. First node has ID 1.

- Constructor allows setting id upon object creation. By default, set id to 0. Also set parent to NULL.

☆ BST

- For part 1, you only need the following (comment out the rest):
  - bst()
  - ~bst()
  - bool empty()
  - void insert(Tkey &)
  - void print_bylevel()
  - void clear(node *)
  - node *insert(node *, Tkey &)

- To your relief, you aren't actually coding much.

- Add int id and set it to 0 in the constructor
  - This is how you will know what id to give to new nodes as they are inserted into the tree.

  - First node inserted starts with an id of 1. If you want, you can set this to 1 initially and increment post-insertion.

- In insert (the recursive one), make 3 changes:
  1. If T is NULL, we are inserting a node. Recall that the bst::node constructor takes an integer. Set the ID of the node accordingly. First node's ID is 1.
  2. We need to set node.parent. We don't have access to the previous node during insertion. However, recall that this function

is recursive. Look at where T→left is assigned. Set T→left's parent to T.

3. Same as #2 but with T→right.

★ BST::NODE

- In print, make 3 changes:
  1. Print both the id and key. Both at setw(3), with a space between them.
  2. Print out parent→id if parent isn't NULL. If it is NULL, print " ROOT ".
     - If you want a shortcut, just C+P the left if-else code and change the variables used.
  3. For parent, left, and right, make it so id is printed instead of key.

- C+P your print function. Make a template specialisation specifically for bst<string>
  - Remember hash table lab where the handout had 3 hash functions (int, float, string)?
  - template <>
    void bst<string>:: ...
  - In this function, simply make the setw before the key be 20.

## ★ OUTPUT BREAKDOWN

- cat test1_int.txt

```
4
2
1
3
2   ← IGNORED. ALREADY IN BST.
6
5
7
KEY
```

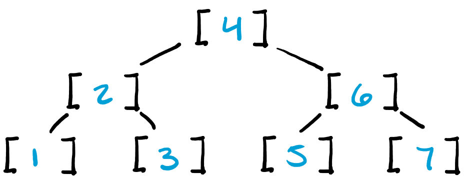
```
./BST1 test1_int.txt

1      4   :   ROOT        L =   2    R =   5
2      2   :   P =    1    L =   3    R =   4
5      6   :   P =    1    L =   6    R =   7
3      1   :   P =    2
4      3   :   P =    2
6      5   :   P =    5
7      7   :   P =    5
ID     KEY          KEY          KEY          KEY
```

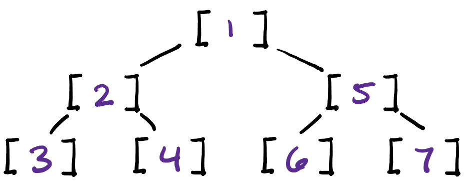
### TREE VISUALISATION

|  |  |  |  |  |  |  |  |  |
|---|---|---|---|---|---|---|---|---|
| KEY | 4 | 2 | 1 | 3 | ✗ | 6 | 5 | 7 |
| ID | 1 | 2 | 3 | 4 |  | 5 | 6 | 7 |

BY KEY:



BY ID:



## ★ TESTING FOR STRING KEYS

- Copy BST1_usage.cpp to BST1_string.cpp.
- In this new file, change:

```
int key;           ⇒   string key;
bst<int> T;            bst<string> T;
```

- Compile via:

```
g++ -o BST1_string BST1_string.cpp
```

NOTICE HOW BST.h ISN'T IN THE COMMAND... HMM...

- If only there were a special script to test string input. Hmm...