CPTS 223 Advanced Data Structure C/C++ Fall 2024

MA2: Binary Search Trees

1 Learning Objectives

At the conclusion of this programming assignment, participants should be able to:

- Understand the logic of binary search trees (BSTs);
- Implement C++ code for BSTs from a starter code.

2 Prerequisite

Before starting this programming assignment, participants should be able to:

- Analyze a basic set of requirements and apply top-down design principles for a problem;
- Describe and analyze BSTs;
- Design, implement and test medium programs in an object-oriented language.

3 Overview and Requirements

3.1 Overview

The starter code "main.cpp" has implemented a main test procedure for the BST class and its member functions. Since some important member functions are not implemented yet, we can only see some outputs as in Figure 1 at this moment.

The starter code "BST.h" has implemented the below three member functions following the example code provided in the textbook, which can be used as good examples for other member functions.

- ~`BST` destructor (refer to textbook, Figure 4.27);
- `findMin` (refer to textbook, Figure 4.20);
- `findMax` (refer to textbook, Figure 4.21);

```
(base) yanyan@Yans-MacBook-Air-2 starter_code % ./build/MA2
**TODO**: insert function
**TODO**: insert function
**TODO**: insert function
**TODO**: insert function
**TODO**: insert function
**TODO**: insert function
(1.1) Print BST (in-order traversal):
 **TODO**: printInOrder function

(1.2) Print BST in level order:
**TODO**: printLevels function
(1.3) Is 100 in BST? true (1) or false (0): **TODO**: contains function
0
(1.4) Is 9 in BST? true (1) or false (0): **TODO**: contains function
0
(1.5) BST size: **TODO**: treeSize function
0
(1.6) Height of BST: **TODO**: treeHeight function
0
(1.7) Print max path:
 **TODO**: printMaxPath function

**TODO**: remove function
(1.8) Removing 11, print BST (in-order traversal):
 **TODO**: printInOrder function

(1.9) Print BST in level order:
**TODO**: printLevels function
(1.10) BST size: **TODO**: treeSize function
0
```

Figure 1: Run starter code.

3.2 Requirements

```
(base) yanyan@Yans-MacBook-Air-2 MA2 % ./build/MA2
(1.1) Print BST (in-order traversal):
 -10 -> -1 -> 1 -> 6 -> 11 -> 100 -> </s>
(1.2) Print BST in level order:
 Level 0: 11
 Level 1: 1 100
 Level 2: -1 6
 Level 3: -10
(1.3) Is 100 in BST? true (1) or false (0): 1
(1.4) Is 9 in BST? true (1) or false (0): 0
(1.5) BST size: 6
(1.6) Height of BST: 3
(1.7) Print max path:
 11 -> 1 -> -1 -> -10 -> </s>
(1.8) Removing 11, print BST (in-order traversal):
 -10 -> -1 -> 1 -> 6 -> 100 -> </s>
(1.9) Print BST in level order:
 Level 0: 100
 Level 1: 1
 Level 2: -1 6
 Level 3: -10
(1.10) BST size: 5
(base) yanyan@Yans-MacBook-Air-2 MA2 %
```

Figure 2: Run an implemented code.

In this coding assignment, we will (i) implement 8 member functions in "BST.h" and (ii) answer a question in "main.cpp". All required tasks are summarized as follows:

1. (10 pts) `contains` (refer to textbook, Figure 4.18): return a bool variable. It returns 1 (true) if the BST contains the queried element, while it returns 0 (false) if the BST does not contain the queried element.
2. (10 pts) `insert` (refer to textbook, Figure 4.17, 4.23): insert a new element into BST.
3. (10 pts) `remove` (refer to textbook, Figure 4.17, 4.26). This function deletes an existing element from BST if the BST contains this target element. It will do nothing if the BST does not contains this to-delete element.
4. (10 pts) `treeSize` returns an integer that is the total number of elements in the BST.
5. (10 pts) `treeHeight` returns an integer that is the height of the BST.
6. (10 pts) `printInOrder` (refer to textbook, Figure 4.60) prints all elements in the BST in the in-order traversal, i.e., from the smallest element to the largest one.
7. (20 pts) `printLevels` prints all elements in a level-wise way. Specifically, it first prints the node at the level 0 (the node with depth=0), i.e., the root node. Then it prints the nodes at the level 1 (the nodes with depth=1), level 2 (the nodes with depth=2), etc.
8. (10 pts) `printMaxPath` prints the longest path in the BST, starting with the root node and stopping at the leaf node in the longest path.
9. (10 pts) Answer the question at the beginning of the "main.cpp" file in the starter code:

   > After generating random integers and inserting them into a BST, for the average-case analysis, the height of the BST is in the order of?

   Please write your answer to this question directly in the beginning of "main.cpp".

After the above 8 member functions are successfully implemented, we can get some useful outputs from the program, e.g., as in Figure 2.

4 How to Submit: Github (and Share with TA)

1. In your Git repository for this class's coding assignments, **create a new branch called "MA2"** (Refer to this YouTube video about how to create a branch from Github web interface or the terminal). In the current working directory, also create a new directory called "MA2" and place all MA2 files in the "MA2" directory. All files for MA2 should be added, committed and pushed to the remote origin which is your private GitHub repository created when during PA1 (NO NEED TO CREATE A NEW REPO).

2. You should submit at least the following files:

   - the header file ("BST.h"), where you will implement the BST class.
   - the main C++ source files ("main.cpp"), where you will have all tests passed.
   - a "CMakeLists.txt" file containing your CMake building commands on Linux/WSL/MacOS which can compile your code.

3. You can refer the example GitHub template project for how to use CMake at `https://github.com/DataOceanLab/CPTS-223-Examples`.

4. Please invite the GitHub accounts of TAs (see Syllabus page and check TA's names as well as their Github usernames before submitting) as the collaborators of your repository. You should submit a URL link to the branch of your private GitHub repository on Canvas. Otherwise, we will not be able to see your repository and grade your submission.

5. Please push all commits of this branch before the due date for submitting your Github link to Canvas portal. Otherwise it might be considered as late submission.

Here is a checklist for submitting your code:

- Invite all TAs and instructor as collaborators of your created repository "CPTS223_assignments". Their github username can be found in Syllabus on Canvas.
- Commit and push your local code to your repository.
- Make sure that your repository reflects and contains all your latest files.
- Copy the link to your repository in the Canvas submission portal.

5 Grading Guidelines

This assignment is worth 100 points. We will grade according to the following criteria: See Section 3.2 for individual points.