IT5003 Semester 2 2024/2025

Data Structures and Algorithms

Tutorial 06
(Balanced) BST
For Week 08 (Sat)/09 (Mon)

Document is last modified on: March 12, 2025

1 Introduction and Objective

The purpose of this tutorial is to reinforce the concepts of Binary Search Tree (BST) and the importance of having a balanced BST. In CS2040/C/S, we properly learn the details of Adelson-Velskii Landis (AVL) Tree as one such possible balanced BST implementation. However, IT5003 students (Python users) will mainly focus on the BST structure - important in technical interview - (and just a short exposure on the existence of self-balancing Search Trees like AVL Tree). When applicable, Python users can rely on the property that the average depth of a vertex in a **randomly-built** BST with n vertices is $O(\log n)$ (this is not officially proven but can be used verbatim in IT5003).

In this tutorial, we will also discuss bBST augmentations by reviewing a simple augmentation to make the standard bBST able to handle duplicate elements (multiset) and another augmentation to support operations that utilizes its 'ordered' property: Select and Rank.

Then, we will show (again) the versatility of bBST data structure as an alternative implementation of Priority Queue ADT that we have learned earlier.

We will also discuss balanced BST versus Hash Table (discussed in the previous tutorial) as implementation for Table ADT.

2 Tutorial 06 Questions

Basic Operations of (Balanced) Binary Search Tree

Q1). (Optional, only when many are still not comfortable with basic bBST operations): We will start this tutorial with a quick review of basic BST operations. The tutor will first open `https://visualgo.net/en/bst`, click Create → **Random** (important so that the height of the BST is

$O(\log N)$). Then, the tutor will ask students to Search for some Integers (exist or not exist, exact vs lower_bound, maximum vs minimum), find Successor (or Predecessor) of existing Integers, perform Inorder (or PreOrder/PostOrder) Traversal, Insert a few more **random** Integers, and also Remove existing Integers (details in Q2).

Q2). What sequence does a preorder traversal of the BST in Figure 1 yield?
Preorder traversal is very similar to Inorder traversal that we have seen in lecture.
Preorder traversal is just like this:

```
PreOrder(T)
  if T is null, stop
  Visit/Process T (see, the visitation is done first)
  PreOrder(T.left)
  PreOrder(T.right)
```

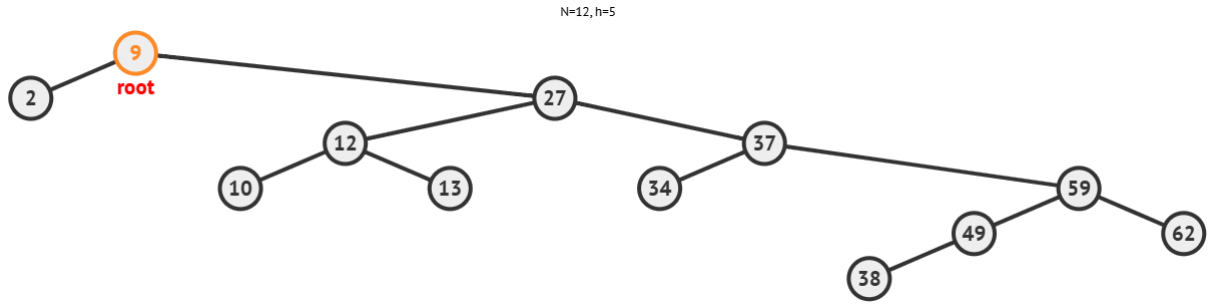

Figure 1: BST for Q4

What about a postorder traversal of the same BST?
Postorder traversal is just like this:

```
PostOrder(T)
  if T is null, stop
  PostOrder(T.left)
  PostOrder(T.right)
  Visit/Process T (see, the visitation is done last)
```

Extra BST Operations (After Augmentations)

Q3). [Should be just a quick review]: What if there are duplicate elements in our bBST? Please discuss on how to implement this feature efficiently after we augment each bBST vertex with yet another 'extra attribute' (different from above).

Q4). There are two important bBST operations: Select and Rank that are now in VisuAlgo (see `https://visualgo.net/en/bst?slide=5-1`) but can be quite useful for some **order-statistics** problems. Please discuss the details on how to implement these two operations efficiently after we augment each bBST vertex with an 'extra attribute'.

Binary Heap... or Not? (Quick Review)

Q5). We know that Binary (Max) Heap can be used as Priority Queue and can do `ExtractMax()` in $O(\log n)$ time. What modifications/additions/alterations are required so that both `ExtractMax()` and `ExtractMin()` can be done in $O(\log n)$ time for the set of n elements and every other Priority Queue related-operations, especially Insert/Enqueue retains the same $O(\log n)$ running time? The elements are not necessarily distinct. Hint: What is the topic of this tutorial?

Hash Table or Balanced BST?

Q6). As of now, you have been exposed with both possible implementations of Table ADT: Hash Table (and its variations) and BST (including Balanced BST like AVL Tree). Now write down four potential usage scenarios of Table ADT. Two scenarios should favor the usage of Hash Table whereas for the other two scenarios, using Balanced BST is better.

Hands-on 6

TA will run the second half of this session with a few to do list:

- PS4 Debrief (Quick one)
- Only for CS2040S/Java (no such library for IT5003/Python, we skip this): Very quick review of Java TreeSet and TreeMap,
- Do a(nother) sample speed run of VisuAlgo online quiz that are applicable so far, e.g., `https://visualgo.net/training?diff=Medium&n=5&tl=5&module=bst` (there should not be AVL Tree question, so this mode is suitable for both CS2040S+IT5003 lab; remember that CS2040S side *will* have AVL Tree questions whereas IT5003 students will have AVL Tree questions turned off *nearer to the execution of VA OQ 2*)
  PS: TAs may choose to review heap and/or hashtable modules too for VA OQ 2 preparation
- Then, live solve another chosen Kattis problem involving BST/AVL Tree/augmented BST...

Problem Set 5

We will end the tutorial with **high-level** discussion of PS5.
Note that we still have next tut+lab07 (22/24 March) and lec9 (26 March),
before PS5 is due on Due: Thu, 27 Mar 25, 11.59pm (last minute before Well-Being Holiday).
So, this early discussion is mostly just to help students **understand the problem** first.