CS2040S Semester 1 2023/2024

Data Structures and Algorithms

Tutorial+Lab 07

Table ADT 2: Balanced BST

For Week 09

Document is last modified on: July 18, 2023

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 learn Adelson-Velskii Landis (AVL) Tree as one such possible balanced BST implementation (it is a legacy... Prof Halim learned this AVL Tree when he was an undergraduate, hence this is what he pass down for now... Maybe one day, he will add Red-Black Tree (this is the one that is used inside Java TreeMap and TreeSet instead of AVL Tree) visualization at VisuAlgo and change the lecture content too :).

In this tutorial, we will first discuss bBST augmentations by discussing operations that utilizes its 'ordered' property: Select and Rank, as well as a simple augmentation to make the standard bBST able to handle duplicate elements.

Then, we will show (again) the versatility of balanced BST 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 07 Questions

Basic Operations of (balanced) Binary Search Tree: AVL 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, but on a balanced BST: AVL Tree. The tutor will first open `https://visualgo.net/en/avl`, click Create → Random. Then, the tutor will ask students to Search for some integers, find Successor of existing integers, perform Inorder Traversal, Insert a few random integers, and also Remove existing integers (details in Q2).

Q2). Draw a valid AVL Tree and nominate a vertex to be deleted such that if that vertex is deleted:
a). **No rotation** happens
b). **Exactly one** of the four rotation (L, R, LR, RL) cases happens
c). **Exactly two** of the four rotation cases happens (you can**not** use the sample given in VisuAlgo which is `https://visualgo.net/en/bst?mode=AVL&create=8,6,16,3,7,13,19,2,11,15,18,10`, delete vertex 7; think of your own test case)
d). **Exactly three** of the four rotation cases happens

Extra BST Operations (After Augmentations)

Q3). There are two important bBST operations: Select and Rank that have just been included in VisuAlgo (during May-July 2023 holiday, 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'.

Q4). 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). Prof Halim has just added this feature at `https://visualgo.net/en/bst` during May-July 2023 holiday.

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

Q5 and Q6 are just quick review of an interesting topic that has been discussed in lecture.

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?

Q6). Quick follow up from the question. above:
Now revisit Q6). of Tut04 (Priority Queue ADT Tutorial).
Would you answer that question differently now?

Hash Table or Balanced BST?

Q7). 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 7

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

- PS4 Debrief (Quick one)

- 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`
- 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.
As usual, we still have next week, so the official discussion is shorter.