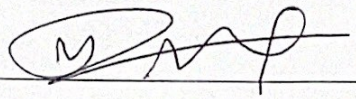
Quiz 03

COSC 211: Data Structures, Fall 2021

Instructions. This quiz is open book and open note—you may freely use your notes, lecture notes, or textbook while working on it. You may *not* consult any living resources such as other students or web forums. The quiz must be submitted by the beginning of class on Thursday, *October 21st, 2021.* If you do not attend class in person, you may email your scanned or typeset solution to the professor using the subject line `[COSC 211] Quiz 03`.
Affirmation. I attest that that work presented here is mine and mine alone. I have not consulted any disallowed resources while taking this quiz.

Name: Dhyey Dharmendrakumar Mavani

Signature: 

Question 1. Consider an implementation of the `SimpleList` ADT that stores elements in a binary tree according to the following scheme: if a node v stores the element at index j and u is a left descendant of v, then u stores an element at index i with $i < j$. If w is a right descendant of v, then the index k of the element stored in w satisfies $k > j$.

Given such a binary tree structure, what information should each tree node store so that the `SimpleList` operations can be implemented in $O(h)$ time (where h is the height of the tree)? How could you implement `get(i)` and `add(i, x)` in $O(h)$ time?

- The case is of Balanced BST (AVL) by indexes.

- We should store items in Balanced binary tree, sorted by key. Additionally, we will add ~~each~~ pointer to the lowest priority element within the subtree rooted at that node. This will allow all three operations in O(logn).

- When we add, we can follow a path from element to root by updating the additional pointer described above as we go. This makes add(i, x) ≡ O(logn).

- Moreover, any interval of keys can be expressed as union of O(logn) subtrees, so the get(i) method will only examine the data inside the O(logn) nodes and thus will take place in O(logn) time.