Data Structures

Homework Assignment 6 - Trees

Problem 1 – Swap - 25 Points
Problem 2 – Duplicates - 25 Points
Problem 3 – Isomorphic - 25 Points
Problem 4 – One-Zero-Tree - 25 Points

25% of Gradescope Autograder test cases are hidden for this assignment.

Spring 2024 - Revision 1 - *November*

Problem 1 – Swap - 25 Points

Implement the member `swap(self, A, B)`, which swapes the nodes A and B in a given binary tree.

Requirements

- Your function has to be in O(1) time complexity.
- Your function has to be in O(1) space complexity.
- Node children and parents have to stay in their original position.
- You are not allowed just to swap node elements.
- You have to maintain the original binary tree.

Problem 2 – Duplicates - 25 Points

Implement the member `has_duplicates(self)`, which returns True if a binary tree contains nodes with identical values (duplicated nodes). Return False otherwise. You cannot modify the input.

Requirements

- Your function has to be in O(n^2) time complexity.
- Your function has to be in O(1) space complexity.
- You cannot use Python lists or any other built-in data structures.

Problem 3 – Isomorphic - 25 Points

Implement the member `is_isomorphic(self, other)`, which analyzes whether *self* and *other* are isomorphic representations. If so, return true. Otherwise, return false. Isomorphic trees have one or more children flipped for nodes. Flips swap the left and right child of a node.

Requirements

- Your function has to be in O(n^2) time complexity.
- Your function has to be in O(1) space complexity.
- You cannot use Python lists or any other built-in data structures.

Problem 4 – One-Zero-Tree - 25 Points

Implement the function one_zero_update(tree) to update the tree's internal nodes based on the following rules. Please note that the function is not a class member:

- If a node has a depth of 0,2,4,6,.. its element equals the maximum of its children.
- If a node has a depth of 1,3,5,7,... its element equals the minimum of its children.

Requirements

- Your function has to be in O(n) time complexity.
- Your function has to return the same tree with all internal nodes updated.
- You cannot use Python lists or any other built-in data structures.