# horizontal lineData Structures

Homework Assignment 8 - AVL and 2-4 Tree

**Update**

* Problem 2: The delete function has to return the tree's root.

Problem 1 – 2-4 Tree Insertion - 25 Points

Problem 2 – 2-4 Tree Deletion - 25 Points

Problem 3 – AVL Single Rotations - 25 Points

Problem 4 – AVL Double Rotations - 25 Points

**Notes and Requirements**

* Your submission must be your effort. You can not copy other students' code.
* This worksheet is graded on performance; Implementations must be correct.
* You are encouraged to visit our office hours to ask coding questions.
* Only the latest (most recent) submission is graded.
* Late submissions are not considered for grading.
* You can not use any third-party libraries.

**Some assignments on this worksheet are manually graded.**

## Problem 1 – 2-4 Tree Insertion - 25 Points

Implement the member function *insert(self, key)* for a 2-4 tree class that maintains the B-tree properties.

**Requirements**

* Time complexity: **O(h)** per insertion, where h is the tree height.
* Space complexity: **O(1)** additional (aside from recursion stack).
* Assume all inserted keys are distinct and comparable.

## 

## Problem 2 – 2-4 Tree Deletion - 25 Points

Implement the member function *delete(self, key)* for a 2-4 tree class that maintains the B-tree properties. Your function has to return the root of the tree.

**Requirements**

* Time complexity: **O(h)** per deletion, where h is the tree height.
* Space complexity: **O(1)** additional (aside from recursion stack).
* Do not rebuild the tree from scratch.

## 

## Problem 3 – AVL Tree Single Rotations - 25 Points

Implement the two fundamental rotation operations for an AVL tree class:

* *left\_rotate(self, node)*
* *right\_rotate(self, node)*

**Requirements**

* Both rotations must run in **O(1)** time.
* The AVL insert must guarantee height = **O(log n)**.
* You cannot use recursion stacks deeper than **O(log n)**.
* Do not use built-in tree libraries; manage pointers and height fields yourself.

## 

## Problem 4 – AVL Tree Double Rotations - 25 Points

Extend your AVL tree implementation to handle double rotations:

* **Left-Right (LR):** when inserting into the right subtree of a left child.
* **Right-Left (RL):** when inserting into the left subtree of a right child.

**Requirements**

* All rotations (single and double) must maintain **O(1)** rotation cost.
* Insertion must still run in **O(log n)** time overall.
* Your code must correctly rebalance for all cases (LL, RR, LR, RL).