Building an AVL Tree With All Related Operations

Group Members:

Ahmet Can Ömercikoğlu (211015038)
Mustafa Mansur Yönügül (211015033)

Introduction:

The AVL Tree Implementation project aims to develop a Python program that implements an AVL (Adelson-Velsky and Landis) tree data structure. AVL trees are self-balancing binary search trees that maintain their balance through rotations, ensuring efficient insertion, deletion, and searching operations. This project provides a complete implementation of an AVL tree, including methods for insertion, deletion, searching, and traversing the tree.

Work Distrubition:

Ahmet Can Ömercikoğlu: Implemented the core AVL tree data structure and methods, including insertion, deletion, searching, balancing, and tree traversal. Also contributed in construction of main function.

Mustafa Mansur Yönügül: Developed the user interface for interacting with the AVL tree. Implemented the main program loop that takes user input and performs the corresponding AVL tree operations. Also helped with balancing and traverse operations.

Flowchart:

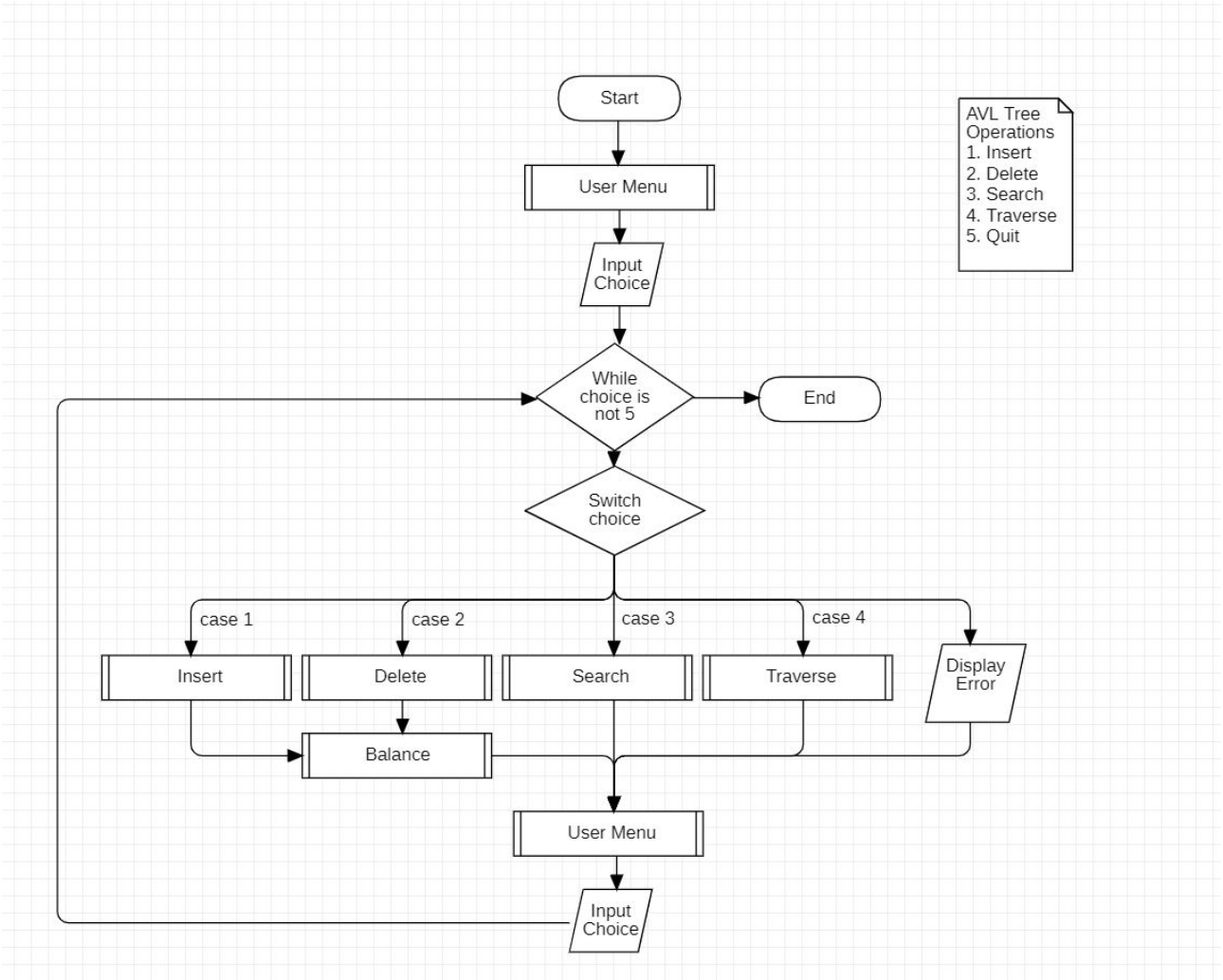

Table:

Function/Variable/File	Data Type	Return Type	Arguments	Description
AVLNode	Class	none	key: any	Represents a node in the AVL tree with attributes for the key, left child, right child, and height of the node.
AVLTree	Class	none	none	Represents the AVL tree and provides methods for AVL tree operations.
AVLTree.insert	Method	none	key: any	Inserts a key into the AVL tree.
AVLTree._insert	Method	none	key: any node: AVLNode	Recursive helper method for inserting a key into the AVL tree.
AVLTree.min_value_node	Method	AVLNode	node: AVLNode	Finds and returns the node with the minimum value in a subtree.
AVLTree.delete	Method	bool	key: any	Deletes a key from the AVL tree and returns True if successful, False otherwise.
AVLTree._delete	Method	tuple	key: any node: AVLNode	Recursive helper method for deleting a key from the AVL tree.
AVLTree.search	Method	bool	key: any	Searches for a key in the AVL tree and returns True if found, False otherwise.
AVLTree._search	Method	bool	key: any node: AVLNode	Recursive helper method for searching for a key in the AVL tree.
AVLTree.balance_factor	Method	int	node: AVLNode	Calculates and returns the balance factor of a node.
AVLTree.height	Method	int	node: AVLNode	Retrieves and returns the height of a node.
AVLTree.rotate_left	Method	AVLNode	node: AVLNode	Performs a left rotation on a node and returns the new root of the rotated subtree.
AVLTree.rotate_right	Method	AVLNode	node: AVLNode	Performs a right rotation on a node and returns the new root of the rotated subtree.
AVLTree.balance	Method	AVLNode	node: AVLNode	Balances a node if its balance factor is violated and returns the new root of the subtree.
AVLTree.traverse	Method	list	node: AVLNode	Performs an in-order traversal of the AVL tree and returns a list of keys.
main	Function	none	none	Main program that interacts with the user and performs AVL tree operations based on user input.

Conclusion:

In conclusion, the AVL Tree Implementation project has successfully implemented an AVL tree data structure in Python. The project provides a well-organized and efficient implementation of AVL tree operations, including insertion, deletion, searching, and tree traversal. The team members collaborated effectively, with one member focusing on the core AVL tree implementation and the other member developing the user interface. The resulting program provides a user-friendly interface for utilizing the AVL tree data structure.