 **HiLCoE School of Computer Sciences**

9/30/2022

**And Technology**

* Data Structure and Algo Analysis
* Dictionary(English to English) Documentation

Batch: DRB-2002

**Prepared by**

* **Surafel Zewdu**
* **Agi Kelbessa**
* **Yosef Tigstu**

.

**public struct Dictionary**//Dictionary Record Struct

{

public char []word;

public char []pron;

public char []type;

public char []meaning;

**}**

**public struct DicInde**x //Dictionary Index Struct (to be Inserted as Data Part of AVLNode in AVL Tree

**{**

public string word;

public long posIndex;

**}**

**public class AVLNode**

**{**

public DicIndex data;

public int height;

public AVLNode left;

public AVLNode right;

**}**

**public partial class AVLTree**

**{**

**private AVLNode roo**t; //top root of the tree

**public AVLTree();** //Constructor

**public AVLNode getRoot()**; //Returns Root

//Inserts Dictionary Record in to both file and AVL Tree

**public void insert(Dictionary newWordDictionaryRecord)**

**private void insertNodeOnlyToAVLTree(DicIndex dicIndex)** //Insert into AVLTree only DicIndex

**private AVLNode insertNode(AVLNode current, AVLNode n)** // Recursively insert into AVLTree

**public void delete(DicIndex target)**

//Recursively Deletes a Node

**private AVLNode deleteNode(AVLNode current, DicIndex target)**

//Edits a Dictionary Record in File, and Updates a DicIndex in Node

**public void edit(DicIndex oldDicIndex, Dictionary newEditedDictionaryRecord)**

//Balaces a Tree based if(Height of left sub Tree- Height of right SubTree > |2|)

**private AVLNode balance\_tree(AVLNode current)**

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***Rotation/BalanceFactor and Height**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//Recurisvly calculates height of the Given Tree

**private int getHeight(AVLNode n)**

//returns height of left subTree-height of Right SubTree

**private int getBalanceFactor(AVLNode n)**

**private AVLNode singleRightRotate(AVLNode parent)**

// (A) (B)

// / \ / \

// (B) AR (C) (A)

// / \ -> / \ / \

// (C) BR CL CR BR AR

// / \

//CL CR

**private AVLNode DoubleLeftRotate(AVLNode parent)**

// (A) (C)

// / \ / \

// (B) AR (B) (A)

// / \ -> / \ / \

// BL (C) BL CL CR AR

// / \

// CL CR

**private AVLNode DoubleRightRotate(AVLNode parent)**

// (A) (C)

// / \ / \

// AL (B) (A) (B)

// / \ -> / \ / \

// (C) BR AL CL CR BR

// / \

// CL CR

**public AVLNode findMax(AVLNode r**)//returns the Right Most Node of the Tree

**public AVLNode findMin(AVLNode r)**//returns the Left Most Node of the Tree

**}**

//\*\*\*\*\*\*\*\*\***\* Additional Methods of AVLTree Class we Implemented\***\*\*\*\*\*

**public partial class AVLTree**

**{**

//Reads Dictionary.bin file and Ineserts Into Tree all Records(as DicIndex)

**public void loadAVLTreeFromFile()**

//Adds Dicionary Record to File

**private long AddDictionaryRecordintoFileAndReturnPostionIndexOfTheWordAdded(Dictionary dictionary)**

//loads Data From AVLTree into new File,then deletes old file

**public void saveDataFromAVLTreeToFile()**

//Inorder Travese and Write to File Each Node Visisted

**private void InOrderTraverseToWriteToNewFileEachNodeVisited(AVLNode rootNode, FileStream oldFileStream, FileStream newFileStream, BinaryFormatter binaryFormatter**)

//finds Nodes that match searchString and Returns there Address in Queue List

**public Queue<DicIndex> find(String searchStr)**

//searches for Exact Matching Word to Search Input **O(Logn)**

**private void findNodeBST(AVLNode rootNode,String searchStr, Queue<DicIndex> resultsQueue)**

//Inorder Travese and Return All Nodes **O(n)**

**private void findAllNodes(Queue<DicIndex> resultsQueue, AVLNode rootNode**)

//Return Prefix Matching Node (Inorder Travese) **O(n)**

**private void findNodesPrefix(AVLNode rootNode, String searchStr, Queue<DicIndex> resultsQueue)**

//Return PostFix Matching Nodes (Inorder Travese) **O(n)**

**private void findNodesPostfix(AVLNode rootNode, String searchStr, Queue<DicIndex> resultsQueue)**

//Return Infix Matching Nodes (Inorder Travese) **O(n)**

**private void findNodesInfix(AVLNode rootNode, String searchStr, Queue<DicIndex> resultsQueue)**

**}**

s