Data Structures(Project Summer 2020)

Its a program that gets a file as an input creates 3 data structures(Binary tree, AVL tree , Hash table) and prints the times it took to create each.

The code has been written and tested to work in Windows with c++ 11.

**In main there a #define MAXWORDS that sets the maximum amount of**

**words that can be read from the file.**

**Main Functions(menuFunctions.cpp) :**

**bool fileExists(string) Checks if the file exists or not and returns true or false respectively.**

**string getFilename() Waits for user to input the name of the file in project folder to read from. If the given file doesn’t exists or enter is pressed with no input the program automatically loads the default file (small-file.txt)**

**void runHashing(HashMap &,string\* &,int &) Run hashing adding words from the given string array and then prints the time taken to add all the words.**

**void runBstNode(BstNode \*,string\* &,int &) Run binary tree adding words from the given string array and then prints the time taken to add all the words.**

**void runAVL(AvlTreeNode \*,string\* &,int &) Run AVL tree adding words from the given string array and then prints the time taken to add all the words.**

**Hash Map (HashMap.h) :**

**bool AddElement(string) Adds a given element to the Hash, returns true if the element was successfully added.**

**bool FindElement(string) Searches for the given string, if found returns true else returns false.**

**void print() Outputs the hash data in HashOutput.txt file in project folder.**

**Hashmap.h has a constant (const int Mapsize) sets the Hash size.**

**Binary Tree (BsgNode.h) :**

**BstNode\* Insert(BstNode\* root, string) Adds a node in the BS tree, in the correct place.**

**bool BstSearch(BstNode\* root, string) Returns whether or not the word has been found.**

**BstNode\* DeleteEllement(BstNode\* root, string,int) Deletes an element/node from the BS tree.**

**void PreOrder(BstNode\* ) Performs preorder.**

**void PostOrder(BstNode\* ) Performs postorder.**

**void InOrder(BstNode\* ) Performs inorder.**

**AVL Tree (AvlTree.h) :**

**AvlTreeNode \*AvlInsertion(AvlTreeNode \*, string) Inserts a new element in the Avl Tree.**

**bool AvlSearch(AvlTreeNode \*, string) Returns weather or not a word has been found.**

**AvlTreeNode \*AVLdelete(AvlTreeNode \*, string,int) Deletes-subtracts a word from the AVL tree.**

**void preorder(AvlTreeNode \*) Performs preorder.**

**void postorder(AvlTreeNode \*) Performs postorder.**

**void inorder(AvlTreeNode \*) Performs inorder.**