EADS

LAB 3 Project Report

AVL Tree

Keshav Dandeva
Student ID : 302333

Introduction

This project report is about implementing template based AVL Trees with various basic operations such as storing data, deleting data, printing data and much more with an additional function of counting words that provides the unique words present in a text file with their occurrences. The counting words function works with AVL Tree as it puts the data from the file in AVL Tree.

AVL Tree Class

The AvlTree class contains various public member functions for different utilities and a private structure Node for tree. The template used for this class is *template<typename Key, typename Info>.*

```
template<typename Key, typename Info>
class AvlTree{

    struct Node{

        Key key;
        Info info;
        int balance;
        Node *left;
        Node *right;

        Node(Key key1, Info info1, Node *left1, Node *right1){

            key = key1;
            info = info1;
            balance = 0;
            left = left1;
            right = right1;

        }

    };

    Node *root;
```

Private functions :

- struct Node
  It contains Key, Info, balance (balance factor) and pointers to left node and right node. It has an object *root which is the first node of the tree.

- void Print(Node* node, int &a) ;
  This function is called in PrintTree function and is used to print the tree with balance factor, left node and right node for every node.

- void deleteAll(Node* node) ;
  This function deletes the node provided and all the sub nodes of it. It is called in public deleteAll() function and also in destructor of the class.

- int Treesize(Node* node, int a);
  This function is called in Size() function and it provides the size of the tree attached to the node provided in parameters.

- int TotalOccurrences(Node* node, int a);
  This function is called in occurenceSum() function and provides the total sum of the info i.e. the number of occurrences of all the keys attached to the node provided.

- int height(Node *node);
  This function gives the height of the node and is called in various other functions. It is important as to balance the AVL tree.

- bool keyExistsCheck(Node* node, Key k1) ;
  This function is used to check if the given key exists or not. It is called in remove function and KeyExist function in public.

- void printKeyCheck(Node* node,Key k1) ;
  This function prints the searched key and its info if it exists.

- Node* LeftRotation(Node *parent) ;
  This function does the rotation of the tree once to left. This function is used to balance the AVL tree.

- Node* RightRotation(Node *parent);
  This function does the rotation of the tree once to right. This function is used to balance the AVL tree.

- Node* LR_Rotation(Node *parent);
  This function does the double rotation of the tree first left and then right. This function is used to balance the AVL tree.

- Node* RL_Rotation(Node *parent);

This function does the double rotation of the tree first right and then right. This function is used to balance the AVL tree.

- Node* treeBalance(Node *node);
  This function is used to balance the tree with the help of the rotation functions.

- Node* insert(Node* node, Key k1, Info i1);
  This function is used to insert a new node to the tree with given key and info. It calls the balancing function to balance the AVL Tree.

- Node* insert(Node* node, Key k1);
  This is overloading function of insert where info is not provided in parameters. Hence, it takes the default value for info as 1.

- Node* findMin(Node *node);
  This function returns link to the minimum node (last) of the tree.

- Node* detachMin(Node *node);
  This function is used to remove the minimum node (last) of the tree.

Public functions :

- AvlTree();
  It is a constructor of the class and it initialises the tree to NULL.

- ~AvlTree();
  It is a destructor of the class and it uses the deleteAll() function to delete the whole tree.

- void printTree();
  This function uses the private print function and prints the tree with the balance factor and left-right link nodes.

- void showRoot();
  This function displays the root of the tree.

- void addNode(Key k, Info i);

This function uses the private insert function to add a node to the tree and also balance it simultaneously.

- void addNode(Key k);
  This function uses the private overloaded insert function (with info as 1 set to default) to add a node to the tree and balance it simultaneously.

- bool keyExists(Key k);
  This function is used to check if the given key exists in the tree.

- void printKey(Key k1);
  This function uses the printKeyCheck() function and prints the key with info if it exists in tree.

- void removeNode(Key k);
  This function calls the private remove function and removes the node with the given key.

- void deleteAll();
  This function is same as in private.

- int size();
  This function returns the size of the Tree.

- int occurenceSum();
  This function returns the sum of the info of all the nodes of the trees.

- int count_words(AvlTree<string, int>& tree, istream& source)

  This is a public function of AVL Tree class that is used to count unique words and their occurrences either from a text file or from user input via keyboard. It cleans the text document from punctations and converts the text to lower case and numbers to string. Then inserts the words to AVL Tree using add function of AVL Tree class.

Testing

1.AVL Tree Class

General Testing for the tree in source code :

1. Creating a new AvlTree class object and testing functions on an empty tree to check their appropriate behavioural response for subsequent functions.

2. Adding elements to the tree to check the balancing operations working.

3. Removing the last node, and then removing the root node, and removing random nodes to check if the tree still exists properly and balanced.

4. Deleting the whole tree and adding elements in such a way to force all types of rotation (left-right rotation, right-left rotation)

5. Creating a large tree with 23 nodes to check the balancing operations at various stages of the tree.

2. Count Words function

This function is tested for counting words in AVL Tree with string and int as the types of template for key and info respectively. In this function, most of the methods are tested while executing through following methods :

1.Input by Keyboard :

In this method, the user is asked to enter strings to count its words. Then the count_words() function is tested and the tree is printed with print() function showing all the nodes linked to left and right and balancing factor for the nodes. The size() function and occurrence_sum() function displays the size of the tree and total sum of words present in the input.

2.Text File

In this method, file handling is used to read the text file provided for testing. Then the count_words() function is tested and the list is printed with print() function showing all the nodes linked to left and right and balancing factor for the nodes. Also, user can find the occurrence of a specific key at the end of the tree. The size() function and occurrence_sum() function displays the size of the tree and total sum of words present in the input.