AST 20105 Data Structures & algorithms

Group Project

Group Member: Lam Ho 55718164
Ling Tsz Wo 55703058
Leung Ho Lun 55750987
Wong Lam 55705212

Introduction

Develop a program that performs basic operations such as file reading, searching for a record, inserting new records, deleting records, searching records in a batch, and/or other requested functions. We choice the Link-list data structure as the structure.

Link-list structure

We decide to use the Link-list structure is it is easy

Insertion

We will compare the value key (ID), like BST. The smaller one will place on the left, which mean store from the beginning. And then we will check it is balance or not. If it is balance, we do nothing. If not, we will compare it one by one by using the ID. If ID is still larger than next one, as we use the link-list, we will find the address of next node. And compare with the value of the next node until the value of next node is larger than the previous one.

Searching

We will ask user input the data that he/she want to find, and we will do the searching one by one until all node is checked. It is a slowest way but it is the most reliable. If it was find, we will print all the data.

Sorting

We use avl tree to do the sorting. The following implementation uses the recursive BST insert to insert a new node. In the recursive BST insert, after insertion, we get pointers to all ancestors one by one in a bottom-up manner. So we don't need parent pointer to travel up. The recursive code itself travels up and visits all the ancestors of the newly inserted node.

1) Perform the normal BST insertion.
2) The current node must be one of the ancestors of the newly inserted node. Update the height of the current node.

3) Get the balance factor (left subtree height – right subtree height) of the current node.
4) If balance factor is greater than 1, then the current node is unbalanced and we are either in Left Left case or left Right case. To check whether it is left left case or not, compare the newly inserted key with the key in left subtree root.
5) If balance factor is less than -1, then the current node is unbalanced and we are either in Right Right case or Right-Left case. To check whether it is Right Right case or not, compare the newly inserted key with the key in right subtree root.
So, we will have four operation to do:
1)Left Left Case
2)Left Right Case
3)Right Right Case
4)Right Left Case