LIBRARY BOOK RECORDS USING DOUBLY LINKED LIST

This program uses a doubly Linked List to create and maintain a library book record. The books can be inserted into the list dynamically (after an initial list is created). The book can be inserted at any desired position in the list. The program displays the list in both orders – from oldest to latest and vice versa. The program also uses the fstream library, which is used to give output in a .txt (text) file. The updated book list will be displayed and saved in a text file on the system. This program can in fact be used to maintain any list of n items.

The program has four functions. The ‘create’ function is used to take input from the user and create the book list. It uses four node pointers to traverse through the linked list. The ‘insert’ function has three cases – insertion at first, insertion at end of the list and insertion between two nodes. Three pointers are used in this function. The ‘del’ function also has three cases and uses three pointers. The ‘show’ function gives the output on the screen and in the text file.

Conclusion: Doubly Linked List can be used to efficiently input and manage large amounts of data as it is comparatively easy to traverse in a doubly linked list in comparison with other data structures.

Also, search, insert, delete, min and max operations all run in O(logN) (for an AVL tree). So, it has many uses, where all operations need to have same time complexity, particularly look up intensive applications like inventory management.

Lookup in an AVL tree is typically faster than Red black tree, but this comes at the cost of slower insertion and deletion due to more rotation operations. Upon insertion, AVL tree may need O(log(N)) operations to rebalance the tree, but red-black tree requires O(1). This means AVL tree is probably faster on searching but slower on insertion. So use an AVL tree if you expect the number of lookups to dominate the number of updates to the tree.

AYUSSH GUPTA

12BCE0129