Juhwan Lee
Professor Karla Fant
CS202
December 1st 2020

Program 5 Efficiency

Program 5 is an extension of program 4. Program 4 is a program for one playlist, and Program 5 has another data structure that manages multiple playlists. The data structure that manages multiple playlists is one of the balanced trees, the AVL tree. In program 3, I implemented a general binary search tree, not a balanced tree, so I had to implement a balanced tree in this program.

The overall design is the same as in program 4, but the difference is that another node class has been added and the AVL tree class. The added node class is called Node2 and has playlist data, left pointer, right pointer, and height value. The height value is needed to determine if the whole tree is balanced or not. The height value is calculated by adding 1 to the largest value among the heights of the left and right child nodes of a specific node. The leaf node has a null child node. In this case, the height of the child node is set to -1. In the AVL tree, the height of the left and right child nodes of all nodes is allowed to differ by a maximum of +1 and a minimum of -1. The way to find out if a tree is balanced or unbalanced is to check if the height difference between the left and right subtrees exceeds or does not exceed the absolute value of 1. If it exceeds absolute value 1, it is unbalanced, and if it does not exceed absolute value of 1, it is balanced. If the tree is unbalanced, the tree is balanced by rotating left or right. Trees skewed to the left perform a right rotation to balance them. Conversely, a tree skewed to the right performs a left rotation. And if the left child node has only the right child node, it performs Left-Right rotation, and if the right child node has only the left child node, it performs Right-Left rotation. Program 4 has only one data file, but program 5 has multiple data files. Because there are multiple playlists. One data file means one playlist (subject). Therefore, the first time the data file is loaded, the file for each subject is loaded. And if the user is a professor (administrator), the user can add a playlist (subject).

In conclusion, it was easier than starting from nothing because program 5 continued on top of program 4, but it was very difficult to implement a balanced tree. The implementation is similar to the normal binary search tree, but it was difficult to keep the balance. However, after completing the implementation and confirming the operation through testing, I was very happy.