CSE222 – DATA STRUCTURES – HW7 REPORT

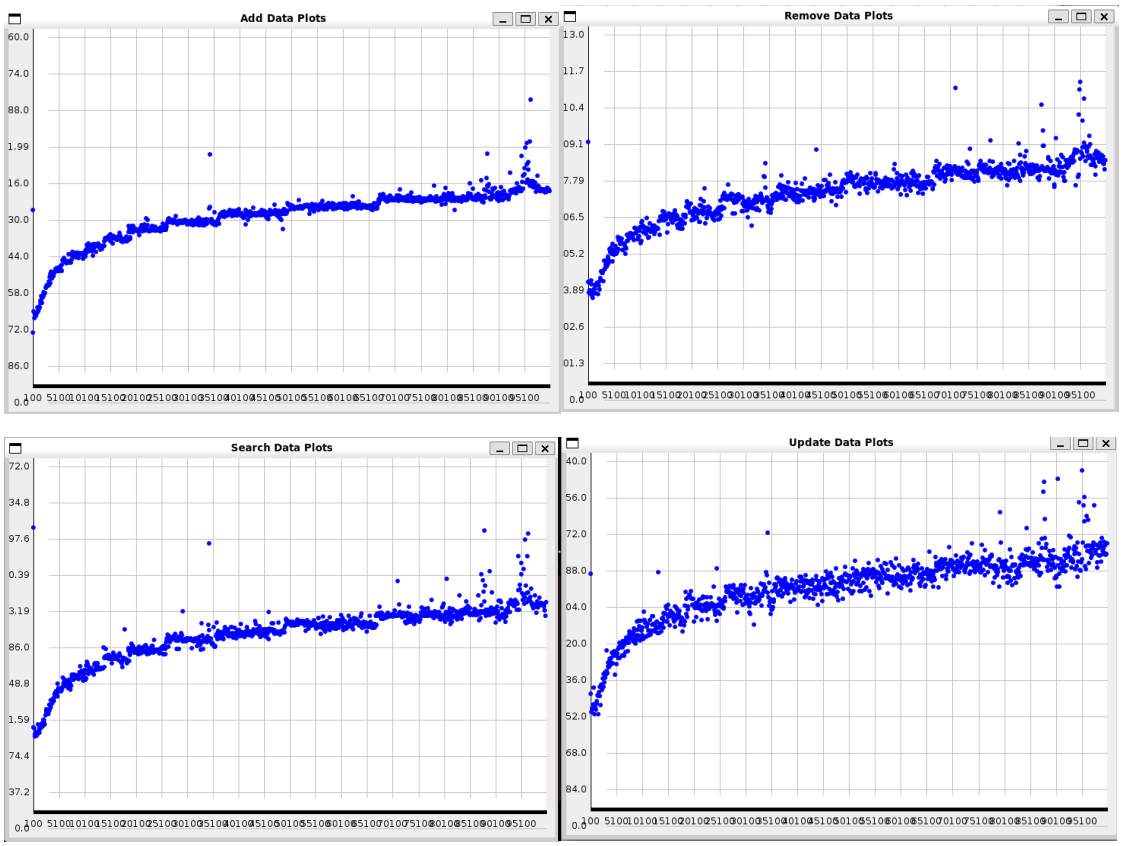
BERKAY EMRE KESKİN - 210104004032

In this homework, we were asked to implement our own AVL Tree according to the Stock class and perform analysis according to add, remove, search and update elements in this tree. I implemented the tree using rotations and balance factors to stable the balance and time complexity of O(log (n)). While i was implementing the add and delete functions (line 171 – 281), i used the BST (Binary Search Tree) implementation to add or remove an element from the tree. After that for each addition or removal from the tree, I checked the balance. If the tree is not balanced, I balanced it using rotation.I also implemented the traversal functions (inorder, preorder, postorder) (line 117 – 164).

We were also responsible for generating an input file to perform analysis on the implementation. I created a class called InputFileGenerator and implemented my functions. According to PDF of the homework, I generated commands using the given size. I get my input file name and each operation with it's size from the arguements and I create commands for each. After that I shuffle the commands and print them to the file.

In the GUIVisualization class, I used the given file and made some changes on the class. I changed the constructor of the GUIVisualization class and added three more variables. I added my data points (each x and y) and title of the graph. I set the fields of datapoints to my data points and I set the title according to the given title.

In the Main class, I hold my all data points for each operation. I get two arguements in the main function, input file and add file. Add file is for adding elements before analysis (In the PDF, we were told to create an AVL Tree with 1000, 10000... sizes). I generated these add files for ease of use in my project (add_100.txt, add_1000.txt, add_10000.txt) using InputFileGenerator. Input file is the randomly generated operation file. At first, I add the elements from the add file. Then I process the input file. In the Main class, we have performPerformanceAnalysis function. For the analysis I use that for filling my data points. I call this function in a for loop to create each average time for the given size. After filling my data fields, I send them with creating a GUIVisualization object for each operation. I send them with their titles to generate a table for each operation. For clearly seeing the table, I tried different sizes of operations. For smaller size of operations the table does not look like a logarithmic table but when I tried to increase the size, I get a clear table to analyze. These are my generated tables for the operations for the size of 10000 operation:



As we see from the graphs, we can get some exceptional data plots but for general, we get an logarithmic table for each operation. We were expecting to get this because of the AVL Tree's speciality which is balancing the tree for each insertion and removal. We get a time complexity for each operation of O(log(n)).