Programming Assignment #03 - Union of Two Binary Search Trees

Data Structures #41, School of Software, SKKU
Susung Park (2014311254), School of Mechanical Engineering, SKKU

1. How to Use

To compile the program, simply move the terminal into the project directory, and then type `gcc main.c -o main.exe` .
After compilation, type `./main.exe` to actually execute the program. Input file should have the name `input.txt` , and should be placed inside the `{PROJECT_DIR}/io` directory.

2. Algorithms

For the reading operation, the program first reads the text file line-by-line, and uses `sscanf()` function to parse the numbers at each line. Since the `sscanf()` function returns the numbers of tokens successfully read, this can be an indicator whether the line was about a normal edge or the root of a tree.

After the reading operation, the program tries to merge two BSTs. However, there were no possible algorithm to generally unify two BSTs into one, except the one iterating the smaller BST and inserting each node into the larger BST (at least within the range I've learned).

In the PDF file provided, there is a binary search tree, which is in an extremely special case; simply connecting the second BST as the left child of 18 of first BST is enough. This is possible because the rightmost of left subtree of 5 has smaller key than the leftmost of second BST. This allows the two BSTs to be merged into one with only single operation. However, in the following figure, it is not the case.

As shown in the figure, the values are 'coupled', so that simply moving one BST beneath the other is impossible. Perhaps the method of determining the range of keys by each subtree and moving in units of them could be possible, but that also causes large amount of computation, and the algorithm becomes complicated.

Therefore, in this assignment, the algorithm simply iterating through the smaller BST and then add each node into the first BST was used.

3. Time Complexity of the Algorithm

Since there was no repetitive instructions during the reading operation, the time complexity of reading operation can be considered as O(l), having l as the length of the text file.

For the merging operation, each element requires independent time for insertion operation. Therefore, the expected time complexity of merging operation is O(nlog(m)), having m as the number of nodes of the larger BST and n that of smaller BST.