Name: Sang Hun Kim
PUID: 0025808690
3/3/2017

Programming Assignment 3 Report

From this programming assignment, it was given to implement a program that involves generating a binary tree, typically a strictly binary tree, and compute the 2D packing of rectangles represented by a binary tree. Leaf nodes of binary tree have widths and heights of rectangles that are needed to be packed, as well as their labels assigned from the input file. Nodes that are not leaf nodes do not have widths and heights, but they are assigned with Horizontal 'H', or Vertical 'V', to indicate where should the rectangles be allocated and packed.

Given with this assignment, a singly linked list was created at the very first that was read from the input file. Typically, input text file has all the information about the binary nodes in pre-order form. Here, creating a linked list from the input text file has time-complexity of $O(n^2)$ because the program is reading all the nodes that are needed to be created as binary tree, in which it takes O(n) of time complexity, and creating linked list from those information requires another O(n) within a while loop. Thus, generating singly linked list requires $O(n^2)$ for time complexity and has O(n) as space complexity.

With the linked list generated from the program, create a strictly binary tree in a sense that has the format that only leaf nodes have information about the rectangles and the rest nodes only have information about where to (path) insert the rectangle for packing. Creating strictly binary tree require O(n) time complexity, as well as space complexity at its worst case because it has to go through n number of nodes to create the binary tree and the tree might be heavily weighted on either right side or left side. However, in average cases, generating binary tree has O(n) for time complexity and O(log(n)) for space complexity.

The main challenge from this assignment was to compute each rectangles' coordinates. However, searching for each rectangle's corresponding coordinates did not require a lot of time nor space. This function's time complexity specifically have O(n/2) because it only functions for the leaf nodes of the binary tree, in which there are n/2 leaf nodes in tree. Also, space complexity of this function is O(n) at its worst case if the tree is heavily weighted on either right side or left side; however, typically it will have O(log(n)) as space complexity because the level, or the height of the binary tree is typically log(n), in which n is the number of nodes existing in the binary tree.

This assignment could have been performed better if used pop, push on the stack function, instead of creating new singly linked list because by performing pop, push on the stack, the time complexity and space complexity will significantly decrease, in which it will be more efficient in terms of time of performance and space in memory.