Justin-Anthony Joco
ECE 368

Programming Assignment 4 Report

As this assignment is a continuation of PA3, the purpose of this assignment is to optimize the packing. Based upon the binary tree constructed from PA3, the code output is to print out the information from PA3 and the smallest rectangle that can fit the input rectangles.

This can be accomplished by rerooting the non-leaf nodes within the original tree. If we have a strictly binary tree with n total nodes, the rerooting algorithm should theoretically have a time complexity of O(n) since there are n-1 tree edges. Note that if we have m leaf nodes, the number of different representations of the same tree is 2m-3. Ideally, the memory complexity should be O(log(n)) because the length of the longest stack frame is the height of the tree, in which h = log(n+1) = O(log(n))

Let's say I have a root node and it's a subtree of a bigger tree. I will denote its left child and right child as left and right respectively. Each child will have a weight on its left and right, which I will denote as left_left, left_right, right_left, and right_right, respectively. Here are four cases if I want to reroot a particular edge:

- Left ->left_left:
    - Set left_right to be left child of root
    - Set root as new right child of left
- Left -> left_right:
    - Set left_left to be left child of root
    - Set root as new left child of left
- Right->right_left:
    - Set right_right to be right child of root
    - Set root as new right child of right
- Right->right
    - Set right_left to be right child of root
    - Set root as new left child of right

I compare each of these cases to the original area of this root node; if the area is smaller, then I set the root's dimensions to that of the smallest rectangle calculated. These dimensions should be sent to the parent, which will calculate the best width and height of the bigger tree that includes this subtree. I continue these comparisons until my root node is each leaf node once.

Unfortunately, my code does not accomplish what I stated above recursively. Thus, my code outputs the correct information for only specific cases. Had I done what I stated above, my time complexity and memory complexity should theoretically be O(n) and O(log(n)), respectively.