Project Report

The re-rooting algorithm works by traversing the tree in a post order traversal. It re-roots at the left child of the current node, packs the re-rooted tree and then propagates to the left child using recursion till a leaf node is reaches. When a leaf node is reached it returns to the call stack and then revert the tree back to the same state the tree was in in the call stack before the re-rooting. Once all the left nodes have been re-rooted the right nodes are re-rooted and the process is repeated. During each packing the area occupied by each re-root is calculated and if the area is smaller the Best Width and Best Height variables over written to the width and height of the of the current packing.

If there are n leaf nodes in the tree the re-rooting algorithm performs operation on each of the internal. No operations happen on the root node and the leaf nodes. Since its a strictly binary tree there are n-2 internal node, given that there are n leaf nodes. The operation is performed on each node is essentially just changing the edge connections depending on the location of the nodes, which is of time complexity O(1)Since this operation is performed twice on each internal node (once operation for each child node) the time complexity becomes O(2n-4). n-2 is the number of internal nodes. Therefore the time complexity of the re-rooting algorithm is O(n).

The re-root function performs the above mentioned re-rooting algorithm in given binary tree and occupies no auxiliary space, therefore its space complexity is O(1).