Karim Itani
00272-22810
ECE 36800

PA04 - RE-ROOTING

Time complexity-

For the re-rooting section of this code, the time complexity is O[n].

Reason:
The re-rooting occurs on each cutline node. The re-rooting is conducted with first the left child, and then the right child. However, even the leaf nodes have a recursive call in order to conduct re-rooting, but nothing happens in this call as they do not have any children. This means that every cutline node (besides the root node) will have 2 recursive calls, and every leaf node will have one recursive call. The total number of cutline nodes (excepting the root node) is ((n-1) - 1), which is equal to (n - 2). Since each of these cutline nodes have two re-roots, we end up with (2n - 4) recursive calls. Each leaf node will add one additional recursive call that does nothing, so this adds n additional calls. Hence the total number of calls is (3n - 4). This number is directly proportional to n. Hence, the overall time complexity is O[n].

Space complexity-

For the re-rooting section of this code, the space complexity is O[log(n)].

Reason:
For the re-rooting of the pre-existing binary tree, no additional memory is required. Only the pointers are moved, and calculations are made. While doing this, the tree is traversed in a post order fashion. Hence, in the worst case scenario, the space complexity will be O[h], where h is the height of the binary tree. This height will be equal to log(n).