Performance analysis of Task 3

1. Time Complexity, O(n): In this problem, the function lca, in its algorithm, it calls another function named find_depth at the beginning, which has a linear time complexity, O(n). Through the following algorithm, there are two while loops, one is to balance the depth of a node, another one is for traversing to find the Least common ancestor. Both of these loops have linear time complexities and are separate, thus there are no nested loops. So, the algorithm to solve this task has linear time complexity, which is O(n).
2. Space Complexity, O(1): In this problem, the discussing function lca, it calls another function name find_depth at the starting of this algorithm. The function find_depth takes a node (__main__.Node type) and two variables that takes constant space. On the other hand, in the function lca, it also has some variables, but these also take constant spaces. So, all the spaces that these functions are going to take in this algorithm are going to be used by the variables only, which are fixed. Apart from that, there will be some auxiliary space consumption because of the instructions, function calls, return statements, etc. We can assume this auxiliary spaces to be constant as well, as this is not going to increase. In total, all the spaces taken by this algorithm is constant and will not increase with time and different inputs. So this algorithm has a constant space complexity, which is O(1).