3.1 + Breath-first-search, we will start at the root node the explore each of its neighbor's before going to its child node. We can't use recursive here for BFS but we should use queue. BFS is generally good for finding shortest path. Time complexity is $O(V+E)$ if used adjacency list and $O(V^2)$ if used adjacency matrix.

The order is a, b, c, d, e, f, g, h, i, J, k, l, m, n, o, p, q

+ Depth-first-search, we start at root then explore each branch completely before going to a different child node or next branch. DFS is preferred if we want to visit each node in the graph (simpler than BFS). We uses stack data structure to implement and recursive is also used.

Time complexity is $O(V+E)$ if using adjacency list and $O(V^2)$ if used adjacency matrix.

Order: a, b, d, e, c, f, h, l, m, q, i, g, J, n, o, k, p

3.2 Clearly explain why algo work:

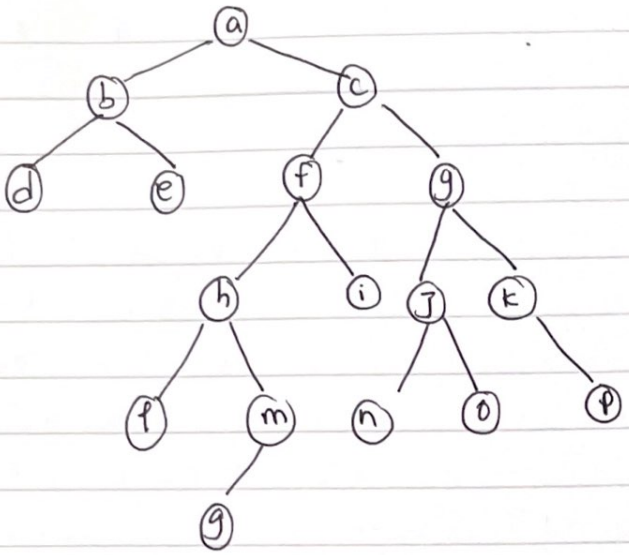
We are going to implement DFS in the problem. We will pick a vertex or node then we will go to ~~it~~ ~~node~~ explore that node in depth. If there are more than one node, we will pick the first node in an acending order then alternate between the two child nodes until we get to the leaf vetex. At the leaf vertex, we will backtract to get to the vextex that has ~~that other child~~ more than one child and pick the other child to go down (explore that child's node). We will continue until we reach the root node or starting node.

This ~~exac~~ will exactly crossing the tree twice because we will go depth on the child node until we reach the leaf node. From that leaf node, ~~from that~~ we backtracks all the way up to the node that has more than 1 child then choose that other child vertex. Continue this process, we will ensure that we cross each edge of tree exactly 5.

The runtime is O(n) where n is # of node).

TOPIC | DATE

3.3



Clearly explain why diameters of above tree is 7 :

from root node a to b, we know the dist(a, b) = 1 ; hence, in the left branch of the tree the max dist is dist (d, a) = dist (e, a) = 2.

from the right branch, we could connect left and right branch of the tree through node a, from node a to h, dist (a, h) = 3 and from h, we could go down to g ; therefore, we would get max distance from the right node of max (a, g) = 5 if we ~~put~~ ~~of~~ get the distance from left to right node which is max dist (d, g) or max dist (e, g) = 5 + 2 = 7

∴ The maximum distance of x and y chosen all aver t is 7