Depth First Search
This method is implemented according to the pseudo code provided in lecture 6[1]. We used recursion in which each node is visited only once. In result, the time complexity is O(V + E) where V and E represent the number of nodes and edges respectively in a given graph.

Breadth First Search
This method is implemented according to a video[2]. We used Queue based approach in which each node is visited only once. In result, the time complexity is O(V + E).

Transitive Closure
This method is implemented according to the pseudo code provided in lecture 6. We visit each node once and perform **Depth First Search** operation on it to find all the reachable nodes. In result, the time complexity is O(N x (V + E)) where N represents the number of iterations, i.e., we iterate through each node, therefore N == V.

Connected Components
This method is implemented more or less according to the pseudo code provided in lecture 6. We iterate over all nodes and perform **Depth First Search** operation on those nodes that are not visited. **Depth First Search** operation gives us a set of nodes that we marked as visited. Hence, we do not have to run **Depth First Search** operation on all the nodes. Then we check if set of nodes is part of another component. If yes, we merge it, otherwise we add it as new component. In result, the time complexity is O(N x (log V + log E)). We used log because we do not run **Depth First Search** operation on all nodes.

[1] https://mymoodle.lnu.se/pluginfile.php/3293457/course/section/383694/2DV600_Lecture6.pdf
[2] https://www.youtube.com/watch?v=bIA8HEEUxZI