Time Complexity Estimates

Depth-First Search as implemented in X.MyDFS.dfs(DirectedGraph<E> graph)

I have used the recursive approach.
We are visiting each node only once. And in the worst case we need to visiting all edges once to. Therefor the time complexity will be O(N+E).

Breadth-First Search as implemented in X.MyBFS.bfs(DirectedGraph<E> graph)

I have used the Queue-approach.
And again, all node and edges in worst case need to be visited one time. Therefor the time complexity again will be O(N+E).

Transitive Closure as implement in X.MyTransitiveClosure.computeClosure(DirectedGraph<E> graph)

I have used the Slide-approach from the lecture. This is as I know a really slow solution (50-60 seconds) and I was trying to use a Tarjan to solve this but didn’t succeed so that’s why I will go with this solution instead.
Here we are looping through each node once and on each node we are running a dfs.
Therefor the time complexity will be O(N*N+E).

Connected Components as implement in X.MyConnectedComponents.computeComponents(DirectedGraph<E> graph)

I have used an approach where I loop through all nodes running a dfs on it and then I check if a node is already part of an component then we just merge it together. With this approach I only need to visit each node once. Therefor the time complexity will be O(n log n + log e).