CITS2200 Project Report

Khang Siang Tay – 21884623
Joshua Ng – 20163079

For this project, we used a unique data structure to return our results. The data structure is an array of array list. This data structure allowed us to store an array list for different components.

Constructor, $O(n^2)$ – n is the number of nodes
The constructor reads the input and makes a HashMap, this map assigns each node to an index which will be used to make an adjacency matrix. The reason we used an adjacency matrix was used because it allowed us to access the edges easily. Besides that, we had previous methods which we wrote in the labs such as breadth first search and these methods required an adjacency matrix.

The constructor also classifies connected nodes into respective components. This was done using breadth first search since breadth first search sets a node to -1 if it is not connected. This was manipulated to assign nodes to its component.

There are multiple loops which are O(n) to read the text file. However, there is one nested loop that contained breadth first search which was used to classify the nodes into its components. Hence, the complexity is $O(n^4)$. This can be improved if Floyd Warshall's algorithm is used as it has a complexity of $O(n^3)$.

Degree Centrality, $O(n^2)$ – n is the number of nodes
For this method, we tallied the number of edges a node has as the lines were read. This tallying is done in the constructor. From the dataset given, since there were only 2 components, the complexity of this method is O(n). However, if a dataset that contains n edges and n components, the complexity in the worst case will then be $O(n^2)$.

Closeness Centrality, $O(n^4)$ – n is the number of nodes
For this method, we used breadth first search on each node and we summed the results into an array. Breadth first search was chosen as it is a good algorithm as it is easier to implement and it is $O(n^2)$. The complexity is $O(n^4)$ since it contained a nested loop and breadth first search was ran on each vertex.

This complexity can be improved by using Floyd Warshall's algorithm has a complexity of $O(n^3)$ as it contains three nested loops.

Betweenness Centrality, O(n*e) – n is the number of nodes, e is the number of edges
For this method, we adapted Brandes' algorithm to implement betweenness centrality. We chose Brandes' because it had a very efficient complexity order, O(n*e). The algorithm was also easy to implement as we referenced the pseudocode from "A faster algorithm for betweenness centrality" from the Journal of Mathematical Sociology.

Katz Centrality, $O(n^4)$ – n is the number of nodes
For this method, we used breadth first search to get this centrality measure. The data that were given was unweighted which meant that connected nodes are of distance 1 apart. Hence, breadth first search was used on the adjacency matrix representation of the graph, it returned an array of k values for each node. This k value was then used to calculate the Katz centrality. We did not use matrix multiplication to find k as it is very inefficient as it had a complexity of $O(n^3)$. The complexity of this is $O(n^4)$ as it contained a nested loop and breadth first search was ran inside the nested loop.

Once again, this complexity can be reduced if Floyd Warshall's algorithm was implemented.

getMax( ) method, $O(n^2)$ – n is the number of nodes
This method was used to return the top 5 nodes for each component. It returns an array of array list. For each component, a priority queue list was used to store the node numbers and the values as the priority. Although to enqueue an item into the priority queue is O(n), only around 5 items will be added into the queue, hence the complexity is still $O(n^2)$. If there are very little components, the complexity of this method will be O(c*n), where n is the number of nodes and c is the number of components. If c = n, the complexity will be $O(n^2)$.

getDist( ) method, $O(n^2)$ – n is the number of nodes
This method runs breadth first search and calculates the single-source shortest path. In an adjacency matrix, it has a complexity of $O(n^2)$. Since the graph is undirected and has non-negative weights, Breadth first search using adjacency list would be better as it has a complexity of O(n+e), where e is the number of edges.