CITS2200 Centrality Project

Kieron Ho, 20500057

Uses Java Runtime Environment 1.8.0_161

Program Output

The program I have developed calculates the centrality values for a dataset that can be loaded into the system via an adjacent file. The centrality values calculated are degree centrality, closeness centrality, betweenness centrality and Katz centrality. My program presents the vertices with the top 5 centrality values of each centrality.

Program Interaction ("Main.java" & "DataParser.java")

The program is started through the code "Main.java" which contains the main method. The program provides a simple interface that allows the user to add new datasets that are placed adjacent to the program files. At any point, a new data set can be added and incorporated into the existing data. There is also an option to clear the data stored in the program to allow for new analysis. The new data is analysed and modified as required in the code "DataParser.java".

At entry or change of data, the program will show the output of the centrality calculations, showing the user the 5 vertices with the top 5 calculated centralities.

It is possible to change the alpha value used to calculate the Katz centrality if desired, otherwise the default is set to 0.5. The data for the Katz centrality is recalculated and all data is redisplayed if the alpha value is altered.

Finally, the program can be closed to exit the program safely.

Program Data Storage ("CentralityData.java")

During operation the program stored input data including a list of unique edges, an edge matrix, and values for each centrality, re-calculated if the dataset changes.

The vertices are stored as ordinally generated numbers in the edge matrix, with their true integer references being stored in a "vertexReferences" array.

Explanation for Centrality Calculations

Degree Centrality ("DegreeCentrality.java")

The degree centrality is the sum of the edges connected to unique vertices. The higher the number of connected vertices, the higher the value of the degree centrality. This was found by using an edge matrix and finding the total number of edges connected to the vertex being tested.

Closeness Centrality ("ClosenessCentrality.java")

The closeness centrality is the inverse of the distance from a vertex to all other connected vertices (called the farness). This value is found by making a shortest weight spanning tree with the root as the chosen vertex and summing the distance to each connected vertex. This process used Dijkstra's algorithm for a priority first search.

Betweenness Centrality ("BetweennessCentrality.java")

The betweenness centrality is a measure of how important a chosen vertex is for the connection between other vertex pairs. The centrality is calculated as the number of all possible shortest paths that require the chosen vertex divided by the total number of shortest paths between two vertices. This process used Ulrik Brandes' Algorithm to calculate the betweenness centrality of each vertex. (Ulrik Brandes. 2001. *A Faster Algorithm for Betweenness Centrality*. [ONLINE] Available at: http://www.algo.uni-konstanz.de/publications/b-fabc-01.pdf. [Accessed 15 May 2018]).

Katz Centrality ("KatzCentrality.java")

The Katz centrality measures a form of connection density, in similarity to the degree centrality, but employing a diminishing additive weight for connected vertices as they move away from the chosen vertex through the shortest path. Each distance from the chosen vertex has its added "degree weight" multiplied with an alpha value with an ordinally increasing index, starting with the root vertex being α^1, moving outwards as α^2 and α^3 etc. For each Katz centrality the shortest paths to each vertex is found using the chosen vertex as the root, with Dijkstra's algorithm.

Additional Strategies Utilised

New PriorityQueue Entry Classes ("QueueEntry.java" & "QueueEntryFloat.java")

These two new classes are used as the stored class type in priority queues. They have two fields:

1. "vertexReference", an integer that primarily refers to an ordinally given vertex reference
2. "key", used as the value to compare in the priority queue. It is assigned this property using a Comparator interface. In the "QueueEntry.java" file it is an integer for use in the degree centrality and the closeness centrality. In the "QueueEntryFloat.java" file it is a float, and used in the betweenness centrality and the Katz centrality

Extended PriorityQueue functionality ("PQMethods.java")

Due to the use of priority queues required in centrality calculations, I found it apt to create additional methods for use with these priority queues. These methods are:

contains(PriorityQueue<QueueEntry> queue, ***int*** *itemValue)*

Takes a provided priority queue and checks if any stored objects contain a selected item value, returning true if found.

changePriority(PriorityQueue<QueueEntry> queue, ***int*** *itemValue,* ***int*** *newPriority)*

Takes a provided priority queue and checks if any stored objects contain a selected item value, and if so removing it from the priority queue and replacing it with a new priority key.