2023 Spring DS210 Project

This project is aiming to analyze the public biking transportation network in Helsinki (Finland's capital). A main question that this project will seek to answer is "what is the most important location within the bike transportation network". To view the bike transportation network as a graph, each biking station can be viewed as a node. In the data, each station has a station ID. The connections or routes between stations can be viewed as the edges.

There are 2 modules written for this project:

I. Read txt to graph: The module takes a txt file, which will be turned into a graph object.

II. Centrality Function Modules : Containing 3 functions

1. Degree Centrality: A station that has many direct neighbor stations can be considered as an important station (node). The function degree centrality return a hashmap, with key as the node index and the value as the degree centrality. The degree centrality for each node = $\frac{node\ i's\ number\ of\ directed\ connected\ neghibors}{total\ number\ of\ nodes\ -1}$

The division helps with normalizing the degree centrality score. To be more precise, the degree centrality will range from 0 to 1. While 0 meaning there is no neghibors connected to the node, 1 represents that a node is directly connected the neghibors with all other nodes in the network.

2. Eigenvector centrality: The eigenvector centrality considers the importance of the neghiboring nodes. For instance, a node with a large number of connections (for example: a high degree of centrality) may have a low eigenvector centrality score if all of its connections are with low-score nodes.
The eigenvector function begins by initializing a vector centrality with an initial value of 0.5 for each node in the graph. I uses a loop to iteratively update the centrality scores for each node in the graph based on the centrality scores of its neighbors. For each node in the graph, the function calculates its centrality score by summing the centrality scores of its neighbors and dividing by the degree of the neighbor. The function iterates through the graph until convergence or the maximum number of iterations is reached. The updated result is stored a hashmap that maps each node index to its eigenvector centrality score. At last, the eigenvector scores are normalized.

3. Betweness Centrality:

Reference https://www.cl.cam.ac.uk/teaching/1617/MLRD/handbook/brandes.pdf
In the case of transportation networks, the betweenness centrality measure could be useful as knowing the shortest path could be crucial in identifying the routes that minimize cost (if the fare is calculated by distance) or time. The function calculates the betweenness centrality score by initializing 0 for each node and then calculates the betweenness centrality for each node by iterating over all the nodes in the graph. For this function, I referenced the Brandes algorithm, which is a algorithm to compute the betweeness centrality. The scores were normalized so that the scores could be between 0~1.

Main Results

```
Station with highest degree centrality: 21.0
Max degree centrality: 0.3769633507853403
Station with highest eigenvector centrality: 21.0
Max eigenvector centrality: 1
Station with highest betweeness centrality: 126
Max betweeness centrality: 0.35139734504804093
```

The station with ID 21 has the highest degree centrality, which implies that station 21 has the most directly connected neighboring stations. To see whether that station 21 is connected to the stations with other highly important stations, the eigenvector centrality score could be helpful in this case. The node 21 also has the highest eigenvector centrality score, meaning that station 21 is directly connected to many other important nodes. In regards to betweeness centrality, station with ID 126 has the highest betweenness centrality, which means that it appears on many shortest paths between pairs of nodes in the graph.