Computing Betweenness Centrality in an Undirected Graph

Assignment #3 for E0 251 Due date : 12-Oct-2019, 11.59 PM

This assignment deals with **betweenness centrality computation** in networks. A network is represented as an undirected graph. The graph to be used is to be downloaded (see below).

- **Part 1**: To be used as a test graph until other graphs as discussed in part 3 are ready. Make sure that your program runs on this graph. **No marks**
  - The compressed directory is *contact-high-school-proj-graph.tar.gz* and is available for download from **https://www.cs.cornell.edu/~arb/data/contact-high-school/index.html.** (Do not cut and paste the URL - the "tilde" character - ~ will not appear properly.)
  - There are two files in the folder/directory. You need to consider the file *contact-high-school-proj-graph.txt* only. There are 5,818 lines in the file; each line corresponds to a weighted edge of the graph in the form i j w where i and j are the two nodes of the undirected edge and w is its weight.
  - This dataset corresponds to a social network of 327 high school students and two nodes have an undirected edge between them if the associated students are friends. Observe that j values are listed in non-decreasing order.
  - Note that there are 327 nodes in the graph. So, $|V| = 327$ and $|E| = 5818$ in G. Ignore all the w values. **Use this edge information to store the given undirected graph G as an adjacency list**.
- **Part 2**: Use the adjacency list to compute the *betweenness centrality*(BC) of vertices using algorthms as below.
  1. Floyd-Warshall all-pairs-shortest-path algorithm. You are required to modify this algorithm to include BC computation. This basically requires counting the number of shortest paths passing through a vertex. Remember that you are dealing with undirected graphs and Floyd-Warshall algorithm is meant for directed graphs (this means some care is needed here). **40% marks**
  2. Breadth-First-Search algorithm to find the shortest paths between pairs of vertices. Rest as in the description above for F-W algorithm. **35% marks**
- **Part 3**: Use a publicly available Erdos-Renyi model random graph generator and test your programs on large graphs with 5000 - 20000 vertices, and plot the execution time results for the three programs. BC values from both the algorithms above must be identical. Experiment with different number of edges and vertices, ranging from very sparse (but connected) to very dense. Erdos-Renyi graph generators need the parameters to be chosen carefully so that one *giant* connected component is generated. There can be many other small connected components. Choose the giant connected component and make sure that it has as many vertices as necessary. Ignore the other smaller components. This means that you must experiment with the parameters and also run a connected components algorithm to find the giant component. Do this without fail. The test graphs that you use must be connected. **25% marks**

Reference: Wikipedia on Betweenness Centrality and Centrality.

Note: There is a better algorithm due to Brandes but the assignment does not cover it. So, do not implement it.