Programming Project (2016)
CITS 2200 - Data Structures and Algorithms

1 Description

The aim of this project is to explore *centrality* of vertices in a graph. Centrality is an important measure of the global influence of a vertex in a graph. This measure is used extensively in analysing large social network graphs for information diffusion and marketing. There are many different measures of centrality and we will explore four of those in this project.

Degree centrality: Degree centrality is the highest number of edges incident on a node. In other words, a node n_i has the highest degree centrality if it has the highest number of edges incident on it. It should be clear that it is easy to compute the degree centrality of a graph in $O(E)$ time for a graph $G = (V, E)$.

Closeness centrality: If a graph $G = (V, E)$ is connected, we can define a distance between every pair of vertices (i, j), which is the shortest path between (i, j). Let us denote this by $SP(i, j)$. The *farness* of a node j, denoted by $far(j)$, is defined as $far(j) = \sum_{i=1}^{N} SP(i, j)$. What this means is that we add up all the distances to all other nodes from the node j. The closeness centrality, $close(j) = \frac{1}{far(j)}$, in other words, the inverse of $far(j)$. Please understand these definitions clearly.

Betweenness centrality: Betweenness centrality is the measure how important a vertex is in terms of how many shortest paths in the graph pass through that vertex. Suppose s_{ij} is the total number of shortest paths between vertices i and j. Suppose $s_{ij}(v)$ is the number of those shortest paths in s_{ij} that passes through another vertex v. The betweenness centrality is the sum:

$$bet(v) = \sum_{(i,j)} \frac{s_{ij}(v)}{s_{ij}}$$

We are basically summing up the betweenness centrality of vertex v for all other vertex pairs (i, j). Note that the computation of $bet(v)$ will require you to compute all possible shortest paths between every pair of vertices (i, j), not just one shortest path between (i, j).

Katz Centrality: Katz centrality is used for degree of influence of a node (or a person in Facebook or Twitter). The idea is to extend the notion of degree centrality to nodes that are more than one hop away. Central to Katz centrality is an attenuation factor α, where, $0 < \alpha < 1$. Consider a vertex i. For every edge that is incident on i, we add a weight α. So if there are k edges incident on i, then we have to add a weight $k\alpha$. Let us call the neighbors of i as $neighbor(i)$. Next we consider all the edges that are incident on the vertices in $neighbor(i)$ and add a factor of α^2 for each such edge. In general, for an edge t hop away from node i, we add a factor of α^t. We continue in this way until we have finished all the vertices and add these weights to get $Katz(i)$, the Katz centrality for vertex i. Usually Katz centrality is written mathematically using the powers of the adjacency matrix A of the graph. For every vertex $i \in G$, A, the adjacency matrix gives the neighbor information, A^2, that is A multiplied by A gives the information of nodes that are two hops away, A^3 gives the information of the nodes that are 3 hops away and so on. So, we can write:

$$Katz(i) = \sum_{k=1}^{\infty} \sum_{j=1}^{N} \alpha^k A^k$$

Note that the first summation is from 1 to ∞ just for convenience, as we don't know after how many hops we will exhaust all the nodes in the graph. Eventually the node with the highest Katz centrality is chosen as the central node in the graph.

2 Tasks

- The project carries 20% of the total marks for this unit.
- Your task is to implement these four centrality measures. Note that it is very easy to implement these algorithms using shortest path algorithms that we will discuss in the lectures shortly. So, a basic correct implementation will fetch you only about 70% of the marks, the remaining 30% of the marks will be awarded on efficiency of your algorithm and the data structures you have used. There is good scope of using some of the data structures tht we have studied in the lectures.
- We will discuss the basic implementation in terms of the shortest path algorithms like Dijkstra's shortest path algorithm, but any sophistication in your implementation is purely your responsibility.

- The project can be done in a group of 2.
- I will provide a Twitter data set from Stanford University in the beginning and I am looking for other datasets as well for testing your codes.

3 Project deliverables

The following are the deliverables for the project:

- **The Java code:** You can write your code in as many files as you like, but there should be a `main` method and you should explain your code briefly in the submitted document. You should also mention the Java version that your code requires. I will check your code from the linux command line, i.e., using *javac* and `java`. You should make sure that your code compiles and runs from the command line.
- You should explain the output your code produces. What I need is the top five nodes according to each of the centrality measures.
- For Katz centrality, the mathematical formula that I have used is just a formulation of the problem. You should not preferably use matrix multiplication, rather something based on breadth-first or depth-first traversal.
- You can use any reliable code either from the Java class library or some third-party library. However you should understand and explain any such code briefly in the report.
- The usual shortest path algorithms report only a single shortest path between vertex pairs (s, t). I found this article for finding all shortest paths between a pair of vertices http://stackoverflow.com/questions/14144071/finding-all-the-shortest-paths-between-two-nodes-in-unweighted-undirected-graph

  If you want to use it, it is your responsibility to understand and check its correctness.
- The report should contain all the information mentioned above and should also explain any innovative strategy or data structure you have used. Please submit the report in pdf format. It is easy to convert word documents into pdf.