Lab 11: Graph centralities

Professor: Ronaldo Menezes

TA: Ivan Bogun

Department of Computer Science
Florida Institute of Technology

November 17, 2014

1 Problem statement

When analyzing graph structure sometimes it is useful to find which vertices in a graph are more important than others. For example, in a social network a person with many followers is a better choice for advertisement deal than the one having fewer connections. A measure of vertex importance within the graph is known as *centrality*. In this lab the task is to implement two such centralities which are based on shortest paths in the graph. *Closeness* is a centrality which takes into account the value of shortest path from a given vertex to every other, while *betweennees* takes into account shortest paths structure.

2 Assignment

Assume $G = G(V, E)$ is a weighted directed graph. Let $d(n_i, n_j)$ be a function which denotes a shortest path between nodes n_i, n_j, define closeness of the node n_i as

$$C_c(n_i) = \frac{|V|}{\sum_{i \neq j} d(n_i, n_j)} \tag{1}$$

where $|V|$ is a total number of vertices in the graph. Function 1 is called *closeness centrality* of a node n_i and is used to determine "central" node in a graph.

Betweennees is another centrality measure based on the shortest paths. Formally betweennees for the vertex v is defined by

$$C_b(v) = \sum_{v \neq s, v \neq t} \frac{\sigma_{st}(v)}{\sigma_{st}} \tag{2}$$

where $\sigma_{st}(v)$ is the total amount of shortest paths that start from s, end in t and pass through v, σ_{st} is the total amount of shortest paths that start from s and end in t. s, t, v are vertices in the graph.

3 Implementation

Download classes *Graph.java*[1] and *ErdosRenyi.java*[2]. Implement the class *CentralityMeasures.java*[3]

```
// CentralityMeasures.java

public class CentralityMeasures {

       private double[][] D;
   private Integer[][] Pi;

   private Graph g;
```

[1] Graph.java
[2] ErdosRenyi.java
[3] CentralityMeasures.java

```
    public double[][] getD() {
            return D;
        }

        public void setD(double[][] d) {
            D = d;
        }

        public Integer[][] getPi() {
            return Pi;
        }

        public void setPi(Integer[][] pi) {
            Pi = pi;
        }

        public CentralityMeasures(Graph g_) {
            this.g=g_;
        }

        public void calculateAllPairsShortestPaths() {
            // calculate all pairs shortest path
        }

        public double betweenness( int k) {
        // calculate betweenness for the vertex k
        }

        public double closeness(int i) {
        // calculate closeness for the vertex i
        }

        public void printMatrix(Integer[][] M) {

                for (int i = 0; i < M.length; i++) {
                        for (int j = 0; j < M.length; j++) {
                                System.out.print(M[i][j] + " ");
                        }
                        System.out.println("");
                }
        }

        public void printMatrix(double[][] M) {

                for (int i = 0; i < M.length; i++) {
                        for (int j = 0; j < M.length; j++) {
                                if (M[i][j] == Integer.MAX_VALUE) {
                                        System.out.print("Inf ");
                                } else {
                                        System.out.print((double)Math.round(M[i][j]*100)/100 + " ");
                                }
                        }
                        System.out.println("");
                }
        }

}
```

4 Sample input-output

Modified version of *Driver.java*[4] will be used for grading.

4.1 Input

```
public class Driver {

	public static void main(String[] args) {
		ErdosRenyi er=new ErdosRenyi(1);
		Graph G=er.randomGraph(7, 0.45);


		CentralityMeasures centrality = new CentralityMeasures(G);

		centrality.calculateAllPairsShortestPaths();

		double[][] D = centrality.getD();
		Integer[][] Pi = centrality.getPi();


		double betweenneesVar=0;
		double closenessVar=0;
		System.out.println("Matrix D");
		System.out.println("---------------");
		centrality.printMatrix(D);
		System.out.println("---------------");
		System.out.println("Matrix Pi");
		System.out.println("---------------");
		centrality.printMatrix(Pi);
		System.out.println("---------------");
		for (int i = 0; i < Pi.length; i++) {

			System.out.println("Centrality measures for vertex "+i);
			closenessVar=centrality.closeness(i);
			betweenneesVar=centrality.betweenness(i);
			System.out.println("Closeness:");
			System.out.println(closenessVar);
			System.out.println("Betweennees:");
			System.out.println(betweenneesVar);
			System.out.println("---------------");
		}


	}
}
```

4.2 Output

```
Matrix D
---------------
0.0   0.41   0.13   0.17   0.15   0.53   0.01
```

[4] Driver.java

```
0.41  0.0  0.52  0.56  0.54  0.92  0.4
0.13  0.52  0.0  0.28  0.26  0.64  0.12
0.17  0.56  0.28  0.0  0.3  0.68  0.16
0.15  0.54  0.26  0.3  0.0  0.38  0.14
0.53  0.92  0.64  0.68  0.38  0.0  0.52
0.01  0.4  0.12  0.16  0.14  0.52  0.0
---------------
Matrix Pi
---------------
null  6  6  6  6  4  0
6  null  6  6  6  4  1
6  6  null  6  6  4  2
6  6  6  null  6  4  3
6  6  6  6  null  4  4
6  6  6  6  5  null  4
6  6  6  6  6  4  null
---------------
Centrality measures for vertex 0
Closeness:
5.0
Betweennees:
0.0
---------------
Centrality measures for vertex 1
Closeness:
2.08955223880597
Betweennees:
0.0
---------------
Centrality measures for vertex 2
Closeness:
3.5897435897435894
Betweennees:
0.0
---------------
Centrality measures for vertex 3
Closeness:
3.255813953488371
Betweennees:
0.0
---------------
Centrality measures for vertex 4
Closeness:
```

```
3.954802259887006
Betweennees:
0.3333333333333333
---------------
Centrality measures for vertex 5
Closeness:
1.9073569482288826
Betweennees:
0.0
---------------
Centrality measures for vertex 6
Closeness:
5.185185185185185
Betweennees:
0.9333333333333333
---------------
```

5 Grade breakdown

basis	grade
Implementation	(60)
closeness	30
betweennees	30
Comments	(20)
Javadocs	10
General	10
Overall	(20)
Compiled	5
Style	5
Runtime	10
Total	100