PROJECT REPORT

TOPIC : NETWORK TOPOLOGY CHARACTERISTICS

Group Members:-

Vinit Girkar 80194118

Dhruvil Patel 801219704

• Introduction:

In this Project we have implemented Network Topology Characteristics using Java. Each complex network (or class of networks) presents specific topological features which characterize its connectivity and highly influence the dynamics of processes executed on the network. The Characteristics are viz Degree of a node and its distribution, Strength of a node and its distribution, Average Weight of a node, Characteristic path length of a network. For the following characteristics we have given explanation, pseudocode, defined its time and space complexity and we also implemented the algorithm. The tool used is PWMiner for the graph visualization, Further we referred the paper from Topological centrality and its applications and Classification of graph metrics as our building block for the project. Biological Network data is used for the topologies evaluation. There are many characteristics possible for evaluation of topologies. The analysis, discrimination, and synthesis of complex networks therefore rely on the use of measurements capable of expressing the most relevant topological features.

Formal Definition of Topology Parameters implemented(GIVEN):

Degree of a node and its distribution:

1. Any graph or network has two basic components - nodes and edges. An un-weighted network can be represented as an adjacency matrix (A). Any element of adjacency matrix $a_{ij.}$
2. Degree distribution is the probability distribution of these degrees over the network. N the total number of nodes in the network. Higher average degree implies good

   inter-connectivity among the nodes in the network.

3. Degree of node is the number of connections it has with the other nodes.

$$k_i = \sum_j a_{ij}.$$

$$P(k) = \frac{N(k)}{\sum N(k)}.$$

Strength of a node and its distribution

1. If wij is the number of possible interactions between any ith and jth nodes, then the strength (si) of a node i is given by:

$$s_i = \sum_j a_{ij} w_{ij}.$$

2. The spread in the strength of a node has been characterized by a distribution

function P(s); where

$$P(s) = \frac{N(s)}{\sum N(s)}$$

Average Weight of a node

1. If wij is the number of possible interactions between any ith and jth atoms, then the average weight < wij > of an atom i is given by

$$< w_{ij} >= \frac{\sum_i \sum_j w_{ij}}{M}.$$

Where M is the total number of interactions among atoms.

Characteristic path length of a network

1. The characteristic path length (L) of a network is the shortest path length between

two nodes averaged over all pairs of nodes and is given by:

$$L = \frac{\sum_i \sum_j L_{i,j}}{N(N-1)}$$

2. where Li,j is the shortest path length between ith node and jth node.
Higher characteristic path length implies network is almost in liner chain and lower characteristic path length shows the network is in compact form.

<u>Pseudo code:</u>

<u>Degree of a node and its distribution:</u>

Store the Graph nodes in getGraphnodes functions and create a 2X2 Adjacency Matrix to store the directed Graph.
Define a function getDegreeOfGraph(String[] graphNodes, **int**[][] adjacencyMatrix)

Create a HashMap to store graph node and corresponding degree

```
for (int i = 0; i < graphNodes.length; i++) { String graphNode = graphNodes[i]; int[]
edges = adjacencyMatrix[i];
int degree = 0;

for (int edge : edges)

{

 if (edge > 0) {

degree++;

          }

}

graphDegree.put(graphNode, degree);

}

return graphDegree;

 }
```

<u>Strength of a node and its distribution</u>

Define a function getStrengthOfNode
Create a HashMap graphStength
Parse through graphnodes to store them in string

```
for (int i = 0; i < graphNodes.length; i++) {

 String graphNode = graphNodes[i];
int[] edges = adjacencyMatrix[i];

//Initialize strength

int strength = 0;



 for (int edge : edges) {

if (edge > 0) { strength += edge;

}

 }

graphStrength.put(graphNode, strength);

}

return graphStrength;

 }
```

# Average Weight of a node

Define a method getAverageWeight for calculating Average Weight of a node Use Hash map for storing the weights

```
for (int i = 0; i < graphNodes.length; i++) {

String graphNode = graphNodes[i];
int[] edges = adjacencyMatrix[i];

int degree = 0;

 int strength = 0;

for (int edge : edges) {
```

```
if (edge > 0) {

strength += edge;

degree++;

}

}

int avgStrength = 0;

if (degree > 0) {
avgStrength = strength / degree;

}

averageWeight.put(graphNode, avgStrength);

 }

return averageWeight;

}
```

# Characteristic path length of a network

Define allShortestPathLenth method, location 1-d array, distance 2-d array

```
large = 1;
for (k=1; k<=m; k++)

large += weight[k];

for (i=1; i<=n; i++) for (j=1; j<=n; j++)

distance[i][j] = (i == j) ? 0 : large;

for (k=1; k<=m; k++) {

i = nodei[k];
j = nodej[k];
if (directed[k])

distance[i][j] = weight[k];
```

```
else

distance[i][j] = distance[j][i] = weight[k];

}

if (root != 1) {



for (i=1; i<=n; i++) {
temp = distance[1][i];

distance[1][i] = distance[root][i];

distance[root][i] = temp;

}

for (i=1; i<=n; i++) {
temp = distance[i][1];

 distance[i][1] = distance[i][root];

distance[i][root] = temp;

}

}

nodeu = 1;
n2 = n + 2;
for (i=1; i<=n; i++) {

location[i] = i;

mindistance[i] = distance[nodeu][i];

}

for (i=2; i<=n; i++) {

 k = n2 - i;
minlen = large;
for (j=2; j<=k; j++) {
```

```
nodev = location[j];
temp = mindistance[nodeu] + distance[nodeu][nodev];
if (temp < mindistance[nodev]) mindistance[nodev] = temp; if (minlen >
mindistance[nodev]) {

minlen = mindistance[nodev]; minv = nodev;
minj = j;

} }

nodeu = minv;

location[minj] = location[k];

}

if (root != 1) {
mindistance[1] = mindistance[root]; mindistance[root] = 0;
for (i=1; i<=n; i++) {

temp = distance[1][i]; distance[1][i] = distance[root][i]; distance[root][i] = temp;

}

for (i=1; i<=n; i++) {

temp = distance[i][1]; distance[i][1] = distance[i][root]; distance[i][root] = temp;

}

 }

}
```

Time Complexity

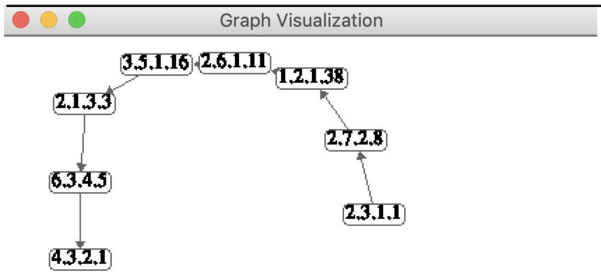
In terms of runtime, our implementation for each method is as follow: where N is the number of nodes in the network.

1. Degree of node: O(N)
2. Strength of node: O(N)
3. Average Weight of node: O(N)
4. Characteristic path length of node: O(N^3)

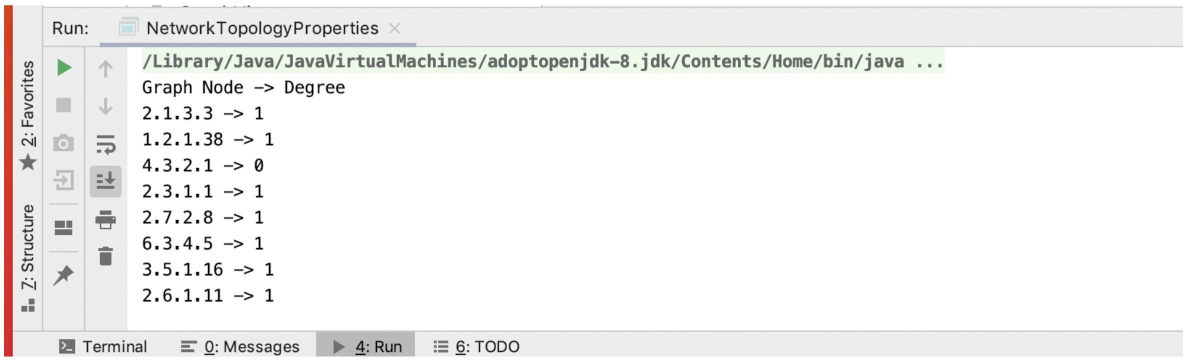
Overall Time Complexity is : O(N^3)

Network analysis of a given graph example and Results

Given Graph:

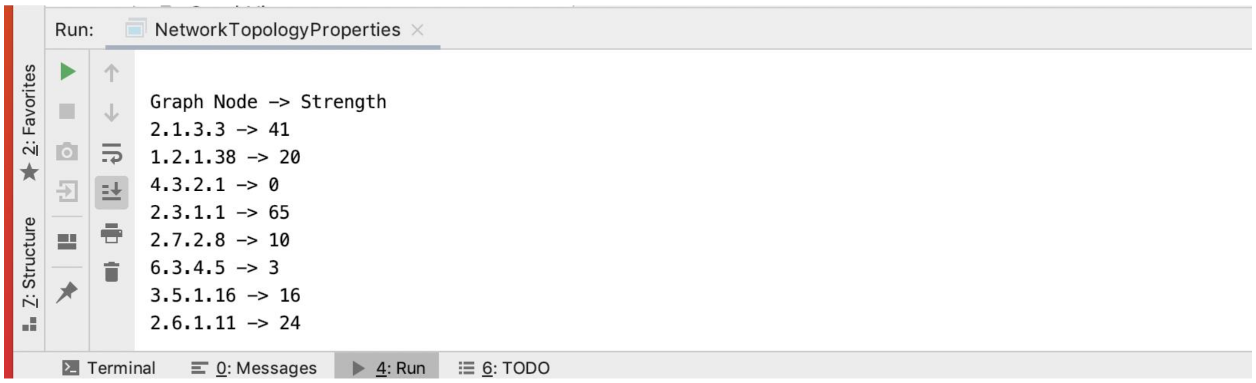


Results: Degree of node:



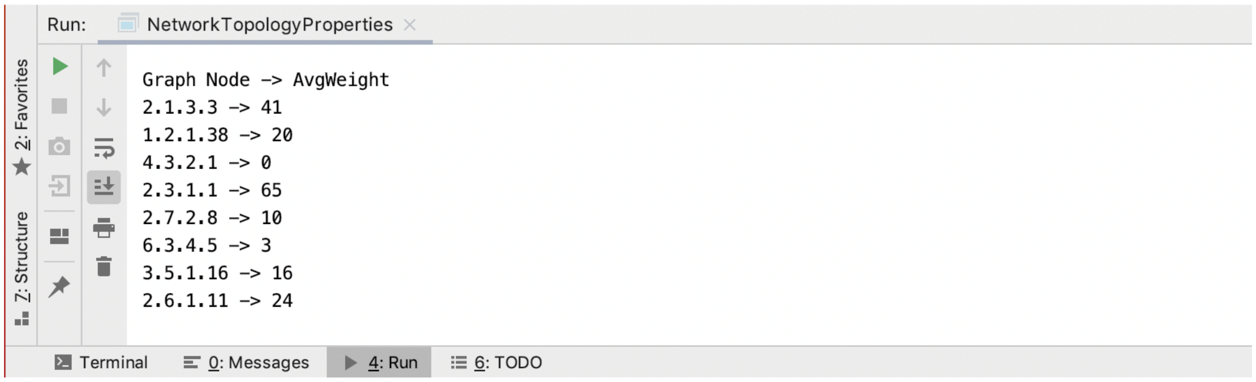
```
/Library/Java/JavaVirtualMachines/adoptopenjdk-8.jdk/Contents/Home/bin/java ...
Graph Node -> Degree
2.1.3.3 -> 1
1.2.1.38 -> 1
4.3.2.1 -> 0
2.3.1.1 -> 1
2.7.2.8 -> 1
6.3.4.5 -> 1
3.5.1.16 -> 1
2.6.1.11 -> 1
```

Strength of node:



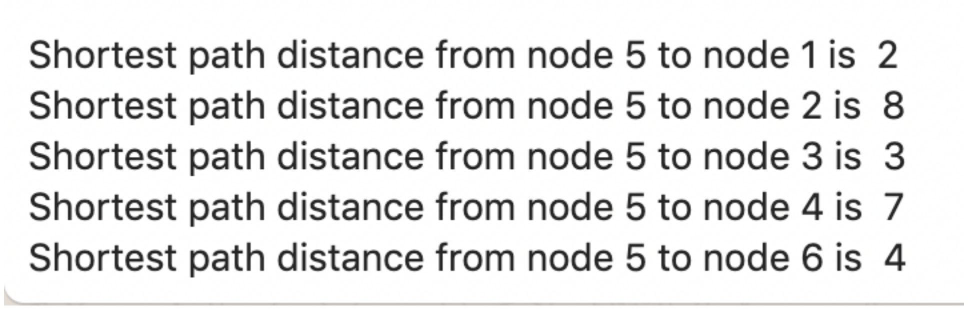
```
Graph Node -> Strength
2.1.3.3 -> 41
1.2.1.38 -> 20
4.3.2.1 -> 0
2.3.1.1 -> 65
2.7.2.8 -> 10
6.3.4.5 -> 3
3.5.1.16 -> 16
2.6.1.11 -> 24
```

Average Weight of node:



```
Graph Node -> AvgWeight
2.1.3.3 -> 41
1.2.1.38 -> 20
4.3.2.1 -> 0
2.3.1.1 -> 65
2.7.2.8 -> 10
6.3.4.5 -> 3
3.5.1.16 -> 16
2.6.1.11 -> 24
```

Characteristic path length:



```
Shortest path distance from node 5 to node 1 is  2
Shortest path distance from node 5 to node 2 is  8
Shortest path distance from node 5 to node 3 is  3
Shortest path distance from node 5 to node 4 is  7
Shortest path distance from node 5 to node 6 is  4
```