Ain Shams University
Faculty of Engineering
Computers and Systems Eng. Dept.

Project
Social Media Analysis

CSE 323
Programming with Data Structures
Spring 2019

Background

Fundamentally, “computer science is a science of abstraction.” Computer scientists must create abstractions of real-world problems that can be represented and manipulated in a computer.

Graph Theory is a relatively new area of mathematics, first studied by the super famous mathematician Leonhard Euler in 1735. Since then it has blossomed into a powerful tool used in nearly every branch of science and is currently an active area of mathematics research. A graph is a set of points (we call them vertices or nodes) connected by lines (edges or arcs).

Some of the applications of graph theory are:

- Networks: A network consists of sites that send and receive messages of various types.
- Maps: A map consists of places that are connected by roads.
- Social Media platforms: The relations (friendship) among users can be modeled as a graph of nodes and edges.

Ain Shams University
Faculty of Engineering
Computers and Systems Eng. Dept.

Project
Social Media Analysis

CSE 323
Programming with Data Structures
Spring 2019

Task

The project aims at applying algorithms based on graph theory to compute the **centrality** property for each node in the graph.
Definition of 'central' varies by context/purpose. However, the main aim is to give a score to each node. This score is used to determine the most important nodes in the graph. These techniques are used in social media analysis to determine the set of influencers out of a graph containing millions of users and connections.
Centrality has many definitions. Each definition gives scores to nodes in different ways.
You are asked to implement three centrality metrics on **undirected connected graphs**.

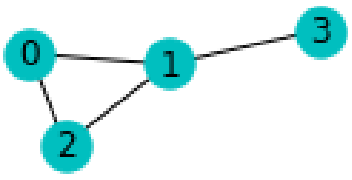
NOTE: ALL THE CENTRALITY ALGORITHMS SHOULD BE **IMPLEMENTED FROM SCRATCH**.

Centrality in graphs

Degree Centrality

Degree centrality is the simplest centrality measure to compute. Recall that a node's degree is simply a count of how many social connections (i.e., edges) it has. The degree centrality for a node is simply its degree. A node with 10 social connections would have a degree centrality of 10. A node with 1 edge would have a degree centrality of 1.

Example:



Node	Degree Centrality
0	2
1	3
2	2
3	1

Ain Shams University
Faculty of Engineering
Computers and Systems Eng. Dept.

Project
Social Media Analysis

CSE 323
Programming with Data Structures
Spring 2019

Closeness centrality

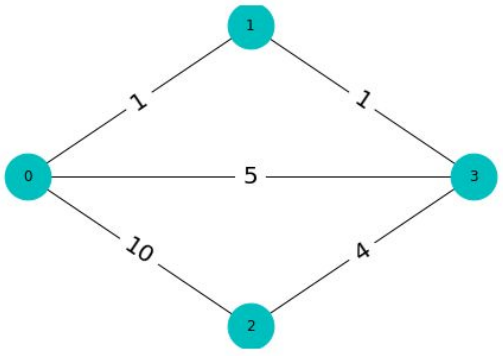
Closeness centrality indicates how close a node is to all other nodes in the network. It is calculated as the average of the shortest path length from the node to every other node in the network.
For a node x and a graph of N connected nodes, The centrality C(x) is defined as:

$$C(x) = \frac{N-1}{\sum_{y} d(y,x)}$$

where $d(y,x)$ is the shortest distance between node x and all other nodes y that are connected to it

---

Example:



For x = 0,
Node 0 is connected to nodes 1, 2, 3.

- d(0, 1) = 1
- d(0, 2) = 6 (The direct edge has weight 10 which isn't the shortest path)
- d(0, 3) = 2

C(0) = N-1/ (1+6+2) = 3/9 = 1/3

Node	Closeness Centrality
0	3/9 = 0.33333
1	3/7 = 0.42857143
2	3/15 = 0.2
3	3/7 = 0.42857143

Ain Shams University
Faculty of Engineering
Computers and Systems Eng. Dept.

Project
Social Media Analysis

CSE 323
Programming with Data Structures
Spring 2019

Betweenness Centrality

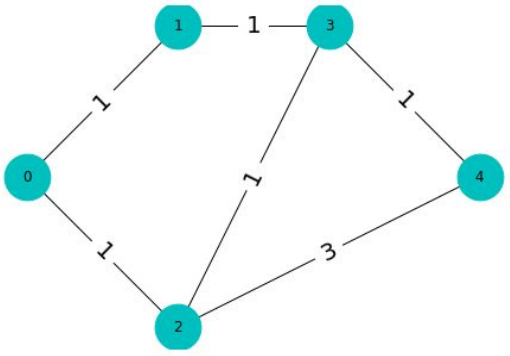
Betweenness centrality measures the number of times a node lies on the shortest path between other nodes.
According to wikipedia:

The betweenness centrality of a node v is given by the expression:

$$g(v) = \sum_{s \neq v \neq t} \frac{\sigma_{st}(v)}{\sigma_{st}}$$

where σ_{st} is the total number of shortest paths from node s to node t and $\sigma_{st}(v)$ is the number of those paths that pass through v .

Example:



SRC, DES	Paths	Shortest path weight
0,1	0,1	1
0,2	0,2	1
0,3	0,1,3 OR 0,2,3	2
0,4	0,1,3,4 OR 0,2,3,4	3
1,2	1,0,2 OR 1,3,2	2
1,3	1,3	1
1,4	1,3,4	2
2,3	2,3	1
2,4	2,3,4	2
3,4	3,4	1

For node 3:
It's lies as an intermediate node in the following paths:

- 0,4 (+ 2/2 = +1)
- 1,2 (+ 1/2 = +0.5)
- 1,4 (+ 1/1 = +1)
- 2,4 (+ 1/1 = +1)

Therefore, the betweenness centrality for node 3 = g(3) = 3.5.

Similarly: g(0)=0.5, g(1)=1, g(2) =1, g(4)=0

Ain Shams University
Faculty of Engineering
Computers and Systems Eng. Dept.

Project
Social Media Analysis

CSE 323
Programming with Data Structures
Spring 2019

General Rules

- Each team should consist of **2 to 4 members**.
- You should register your teams using the following form: https://forms.gle/8wvgfF8jN7kkbmMG6
- Final Evaluation will be during the second week of the summer vacation.
- Each centrality metric will have its set of problems on codeforces (one or more problem for each metric).
  Each problem will have different constraints.
  Contest URL: WILL OPEN BEFORE the 6th of May.
- All centrality measures should be **implemented from scratch.**
- You are free to use any tool/ language in order to visualize the graph such that nodes with higher centrality are clear.
- You won't be asked to **visualize** a graph of **size > 100 nodes**.
- You are encouraged to design your codes such that the computation and the visualization parts aren't dependent.

Constraints for problems:

The contest URL is: https://codeforces.com/group/nyuWrPiFje/contest/243176

- Degree Centrality:
  - Small Input: 2<=n<=1,000 and n-1<=m <=n*(n-1)/2
  - Large Input: 2<=n<=1,000,000 and n-1<=m <=n*(n-1)/2
- Closeness Centrality:
  - Small Input: 2<=n<=10 and n-1<=m <=n*(n-1)/2
  - Large Input: 2<=n<=100 and n-1<=m <=n*(n-1)/2
- Betweenness Centrality:
  - Small Input: 2<=n<=10 and n-1<=m <=n*(n-1)/2
  - Large Input: 2<=n<=100 and n-1<=m <=n*(n-1)/2

Ain Shams University
Faculty of Engineering
Computers and Systems Eng. Dept.

Project
Social Media Analysis

CSE 323
Programming with Data Structures
Spring 2019

Grades Distribution (To be confirmed)

Item	Grade (Percentage of the project's degree or absolute marks)
Degree Centrality	50%
Closeness Centrality	30%
Graph Visualization (less than 100 nodes)	20%
Betweenness Centrality	Bonus +20%
Using Github	- Meaningful commits: +0.5 mark - Using branches: +0.5 mark - Using Markdown in README file: +0.5 mark - Detailed steps in README for installing the requirements/dependencies: +0.5 mark
Deliverable	A single executable file/ single entry point for both the computations and the visualization: +1 mark
Submissions on Codeforces	- Getting an Accepted verdict from first submission: +0.25 mark - Getting an Accepted verdict in less than four submissions: +0.125 mark - Getting an Accepted verdict in more than ten submissions: -0.25 mark
Creative ideas/ solutions/ codes	+1 mark (You shouldn't ask for it, I will give it if I think you deserve it)
Positive spirit regarding the project/ Facebook posts/ etc	+0.01 mark ;)

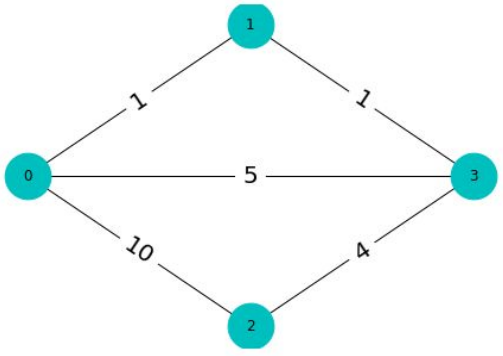
Ain Shams University
Faculty of Engineering
Computers and Systems Eng. Dept.

Project
Social Media Analysis

CSE 323
Programming with Data Structures
Spring 2019

Input description

The program should read from a file describing the graph.
The first line will contain two integers n,m where n is the number of nodes and m is the number of edges in the graph (1 < n < V, 0 < m < E).
V, E will depend on the problem (Further information will be provided on codeforces).
This will be followed by m lines describing the edges of the graph.
Each of the m lines will contain two integers a, b, c that represents an undirected edge between nodes a and b with weight c.




```
4 5
0 2 10
1 0 1
0 3 5
3 1 1
2 3
```

Output description

Print n lines showing the centrality measure for all the nodes.
The output will take the form C(NODE).
Centrality for nodes should be printed ascendingly according to the node's id.
NOTE: The difference in precision between your codes and the judges' ones should be less than $\mathbf{10^{-6}}$, you can print centralities with higher precisions (more than 6 decimal places).

Example:
If the problem needed to compute the closeness centrality for the example input file shown above, the output should be:

```
0.333333333333
0.42857143
0.2000
0.42857143
```