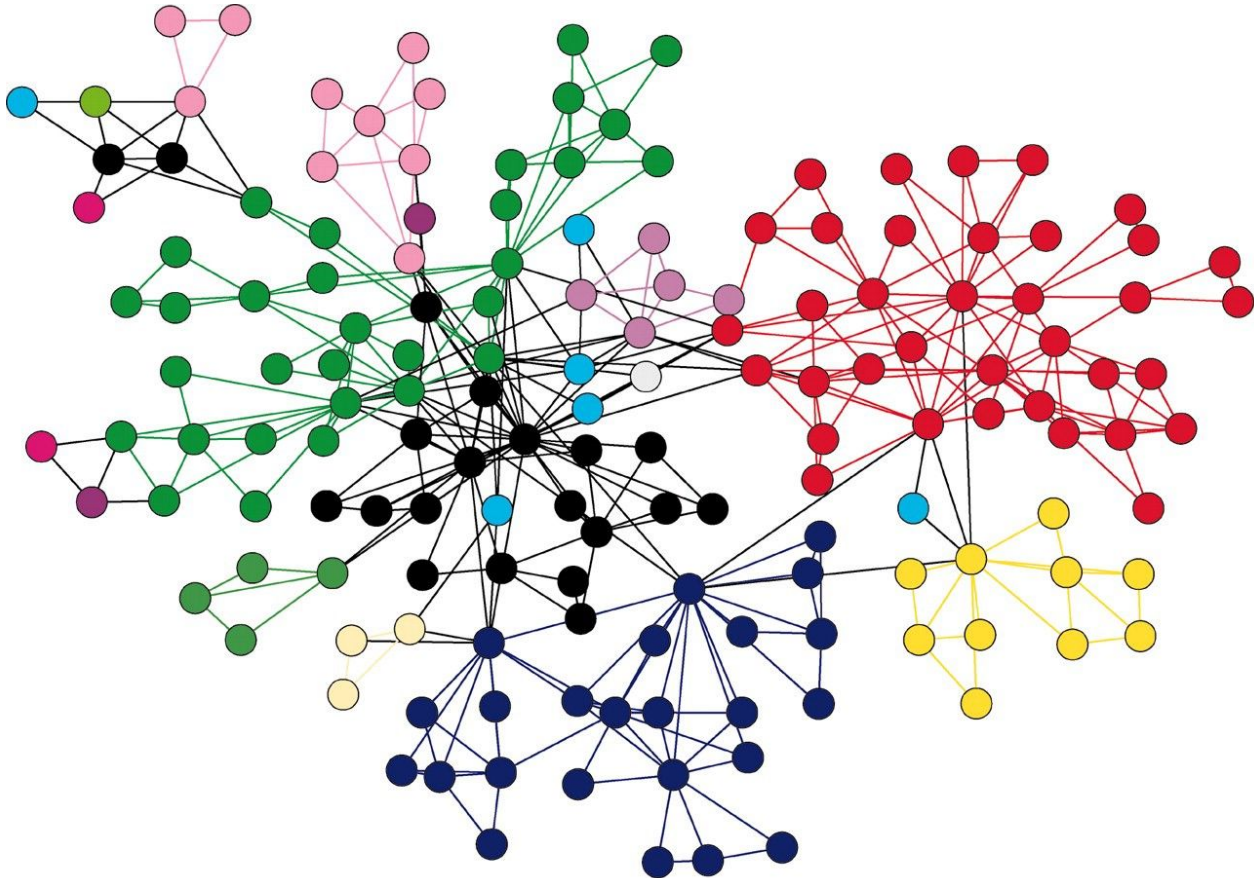
Algorithms and Data Structures - ITCS6114

GRAPH STREAMING ANALYSIS



Dhananjay Arora

Jasmine Maria Corea

Kshitij Jaju

Naman Manocha

Namra Desai

Introduction

The real-time prediction of new connections given the previous connections in a graph is a notoriously difficult task. The proposed technique avoids this difficulty by modeling statistics computed from the graph over time. Vertex statistics summarize topological information as real numbers, which allows us to leverage the existing fields of computational statistics and machine learning. This creates a modular approach to analysis in which methods can be developed that are agnostic to the metrics and algorithms used to process the graph.

Benefits of Graph Summarization

—Reduction of data volume and storage:

Graphs of real-world datasets are often massive. For example, as of August 2017, the Facebook social network had 2 billion users, and more than 100 billion emails were exchanged daily. Summarization techniques produce small summaries that require significantly less storage space than their original counterparts. Graph summarization techniques can decrease the number of I/O operations, reduce communication volume between clusters in a distributed setting, allow loading the summary graph into memory, and facilitate the use of graph visualization tools while avoiding the "hairball" visualization problem.

—Speedup of graph algorithms and queries:

While a plethora of graph analysis methods exist, many cannot efficiently handle large graphs. Summarization techniques produce smaller graphs that maintain the most salient information from the original graph. The resultant summary graph can be queried, analyzed, and understood more efficiently with existing tools and algorithms.

—Interactive analysis support:

As the systems side makes advancements in interactive graph analysis, summarization is introduced to handle information extraction and speed up user analysis. The resultant graph summaries make it possible to visualize datasets that are originally too large to load into memory. —Noise elimination: Real graph data are frequently large scale and considerably noisy with many hidden, unobserved, or erroneous links and labels. Such noise hinders analysis by increasing the workload of data processing and hiding the more “important” information. Summarization serves to filter out noise and reveal patterns in the data.

Graph Statistics

Graph Kernel

A graph kernel as an algorithm that builds a data structure or index on a graph. We define a vertex statistic as a function from the vertex set to the real numbers that depends on the edge set and any information contained by the edges, i.e. edge weight.

The data for each statistic can be stored as an $|V| \times |T|$ array, which is indexed by vertex set V and time steps T = t1, t2, . . . , tT . These dense matrices are amenable to parallel processing using techniques from high performance linear algebra.

These statistics can also be visualized over time, which is a challenge for graphs with more than thousands of vertices. These statistics can give an analyst a dynamic picture of the changes in the graph without showing the overwhelming amount of topological information directly.

Betweenness Centrality

In graph theory, betweenness centrality is a measure of centrality in a graph based on shortest paths. For every pair of vertices in a connected graph, there exists at least one shortest path between the vertices such that either the number of edges that the path passes through (for unweighted graphs) or the sum of the weights of the edges (for weighted graphs) is minimized. The betweenness centrality for each vertex is the number of these shortest paths that pass through the vertex.

Betweenness centrality finds wide application in network theory: it represents the degree of which nodes stand between each other. For example, in a telecommunications network, a node with higher betweenness centrality would have more control over the network, because more information will pass through that node. Betweenness centrality was devised as a general measure of centrality:[1] it applies to a wide range of problems in network theory, including problems related to social networks, biology, transport and scientific cooperation.

The betweenness centrality of a node {\displaystyle v} is given by the expression:

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

$$\mathrm{normal}(g(v)) = \frac{g(v) - \min(g)}{\max(g) - \min(g)}$$

which results in:

max(normal)=1

min(normal)=0

Closeness Centrality

In a connected graph, closeness centrality (or closeness) of a node is a measure of centrality in a network, calculated as the reciprocal of the sum of the length of the shortest paths between the node and all other nodes in the graph. Thus, the more central a node is, the closer it is to all other nodes.

Closeness was defined by Bavelas (1950) as the reciprocal of the farness, that is:

$$C(x) = \frac{1}{\sum_y d(y,x)}.$$

where d(y,x) is the distance between vertices x and y. When speaking of closeness centrality, people usually refer to its normalized form which represents the average length of the shortest paths instead of their sum. It is generally given by the previous formula multiplied by N-1, where N is the number of nodes in the graph. For large graphs this difference becomes inconsequential so the -1 is dropped resulting in:

$$C(x) = \frac{N}{\sum_y d(y,x)}.$$

This adjustment allows comparisons between nodes of graphs of different sizes.

Taking distances from or to all other nodes is irrelevant in undirected graphs, whereas it can produce totally different results in directed graphs (e.g. a website can have a high closeness centrality from outgoing link, but low closeness centrality from incoming links).

Clustering Coefficient

In graph theory, a clustering coefficient is a measure of the degree to which nodes in a graph tend to cluster together. Evidence suggests that in most real-world networks, and in particular social networks, nodes tend to create tightly knit groups characterised by a relatively high density of ties; this likelihood tends to be greater than the average probability of a tie randomly established between two nodes.

$$C = \frac{\text{number of closed triplets}}{\text{number of all triplets (open and closed)}}.$$

Clustering Coefficient is a measure of how tightly knit the graph is. It is used to detect communities and measure cohesiveness of those communities. It can also be used to determine the stability of graph.

There are two types of clustering coefficient:

- Local clustering coefficient

The local clustering coefficient of a node is the likelihood that its neighbours are also connected. The computation of this score involves triangle counting.

- Global clustering coefficient

The global clustering coefficient is the normalized sum of those local clustering coefficients.

We can leverage linear regression to find vertices that are significant. Once we have computed the statistic at two distinct time-steps, the line of best fit can be used to determine what a typical change was due to the edges that were inserted. The distance from each vertex to that best-fit surface could be used as a score for each vertex. The vertices with large scores could then be processed as anomalous.

The value of Clustering coefficient ranges from 0 to 1, The value can also be helpful to predict the shape of the graph as when it is close to 0 the graph shape will be like a star or else if it closer to 1 it will be a clique.

In the dataset on windows authentication system, most of the nodes have clustering coefficient value of 0 because it is extremely rare for authentication nodes to form any triangle patterns. As all the relation are mostly one to one in nature or one to many one in some cases.

Degree of a Node

In the study of graphs and networks, the degree of a node in a network is the number of connections it has to other nodes.

The degree distribution is very important in studying both real networks, such as the Internet and social networks, and theoretical networks. The simplest network model, for example, the (Bernoulli) random graph, in which each of n nodes is connected (or not) with independent probability p (or 1 − p), has a binomial distribution of degrees k:

$$P(k) = \binom{n-1}{k} p^k (1-p)^{n-1-k},$$

(or Poisson in the limit of large n). Most networks in the real world, however, have degree distributions very different from this. Most are highly right-skewed, meaning that a large majority of nodes have low degree but a small number, known as "hubs", have high degree. Some networks, notably the Internet, the world wide web, and some social networks are found to have degree distributions that approximately follow a power law: $P(k) \sim k{-}\gamma$, where γ is a constant. Such networks are called scale-free networks and have attracted particular attention for their structural and dynamical properties.

NetworkX Library

NetworkX is a Python package for the creation, manipulation, and study of the structure, dynamics, and functions of complex networks.

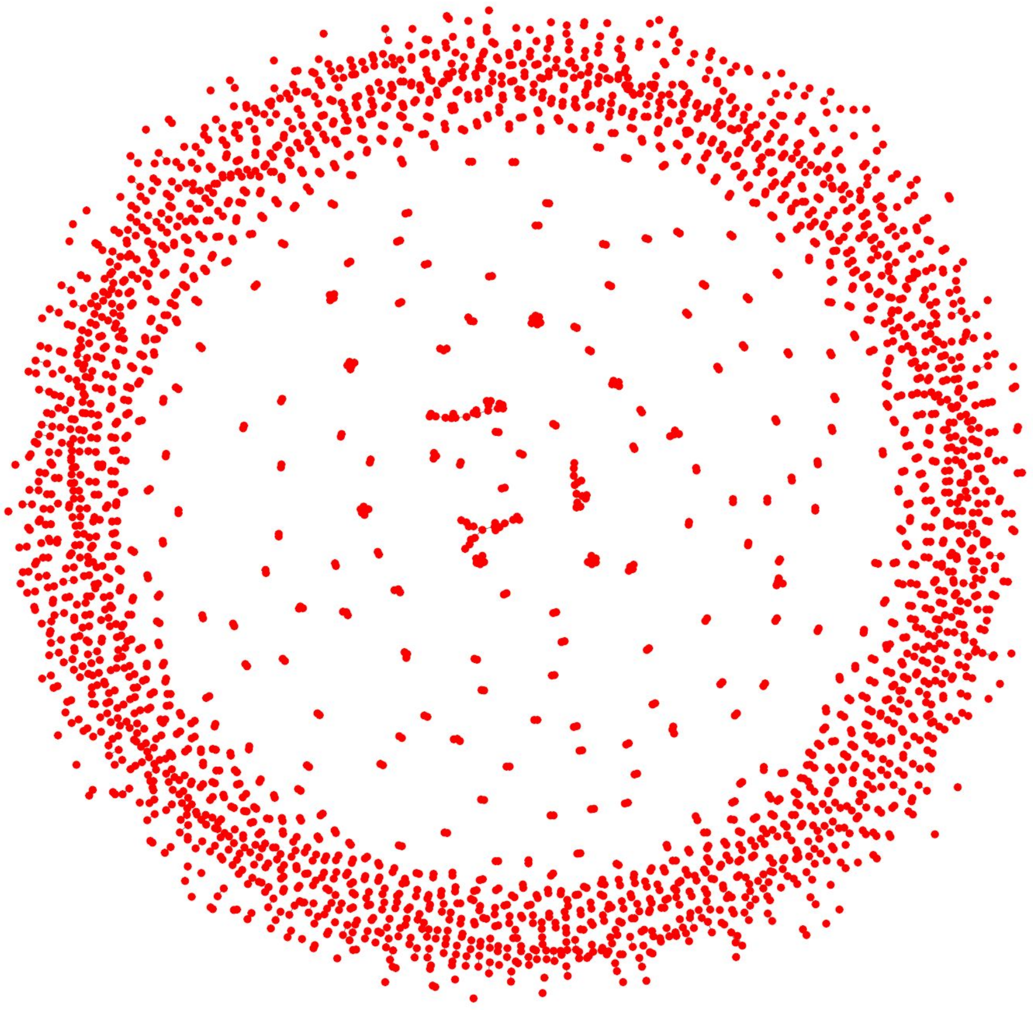
With NetworkX you can load and store networks in standard and nonstandard data formats, generate many types of random and classic networks, analyze network structure, build network models, design new network algorithms, draw networks, and much more.

NetworkX provides:

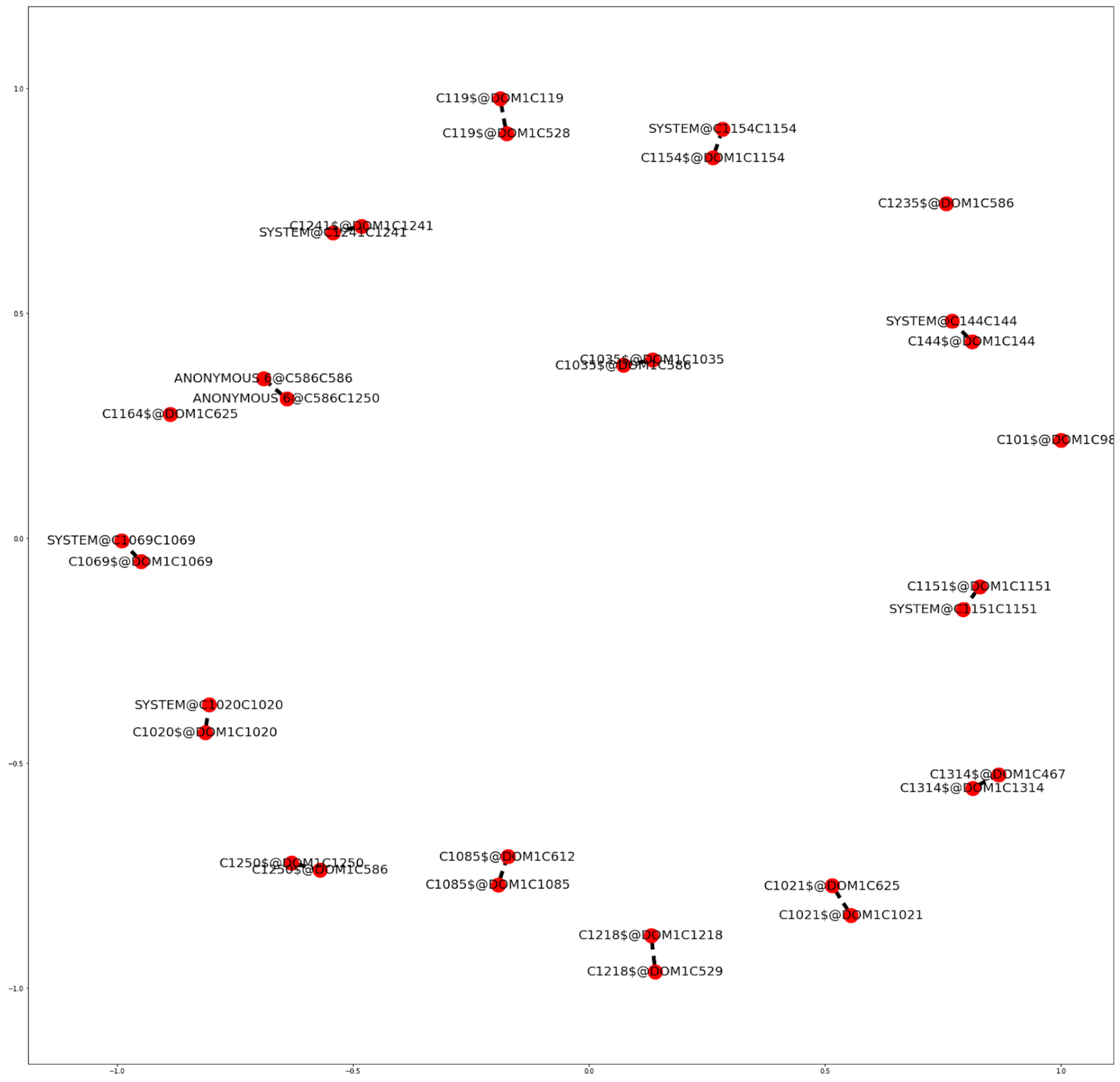
- tools for the study of the structure and dynamics of social, biological, and infrastructure networks;
- a standard programming interface and graph implementation that is suitable for many applications;
- a rapid development environment for collaborative, multidisciplinary projects;
- an interface to existing numerical algorithms and code written in C, C++, and FORTRAN; and
- the ability to painlessly work with large nonstandard data sets.

Screenshots from Code

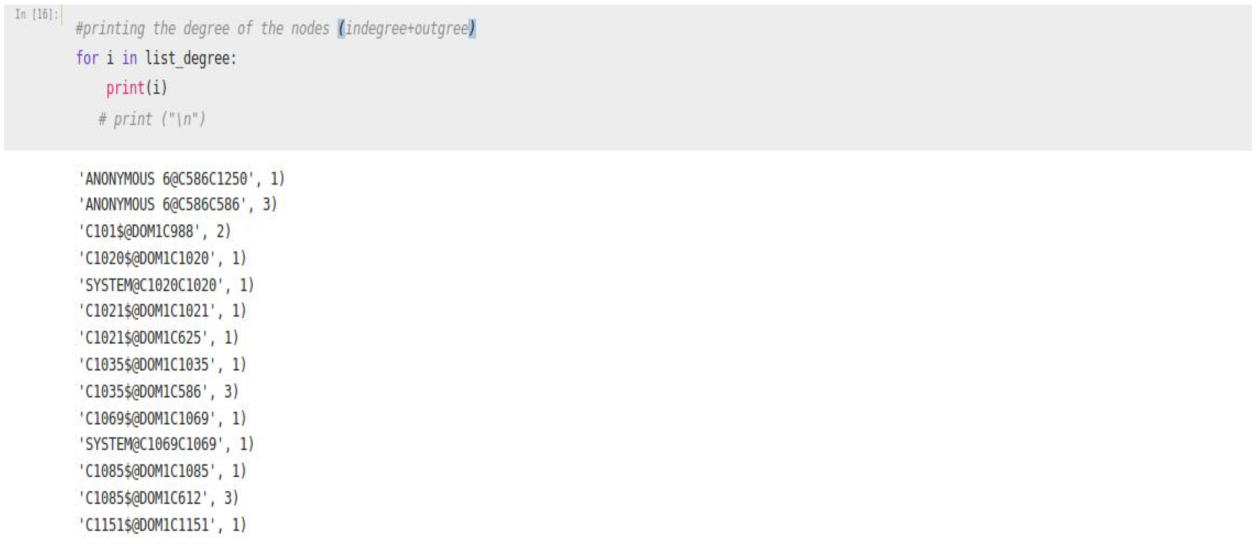
Graph - 2000 rows



Graph - 50 nodes



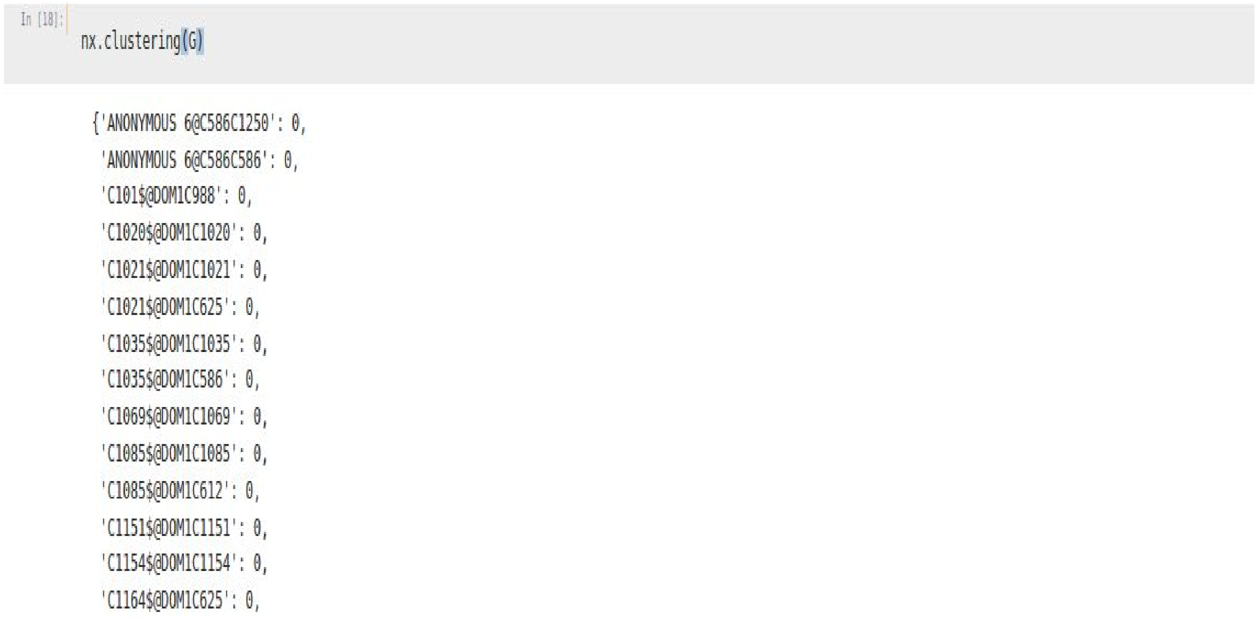
Degree of a Node



```
In [16]:
#printing the degree of the nodes (indegree+outgree)
for i in list_degree:
    print(i)
   # print ("\n")
```

```
'ANONYMOUS 6@C586C1250', 1)
'ANONYMOUS 6@C586C586', 3)
'C101$@DOM1C988', 2)
'C1020$@DOM1C1020', 1)
'SYSTEM@C1020C1020', 1)
'C1021$@DOM1C1021', 1)
'C1021$@DOM1C625', 1)
'C1035$@DOM1C1035', 1)
'C1035$@DOM1C586', 3)
'C1069$@DOM1C1069', 1)
'SYSTEM@C1069C1069', 1)
'C1085$@DOM1C1085', 1)
'C1085$@DOM1C612', 3)
'C1151$@DOM1C1151', 1)
```

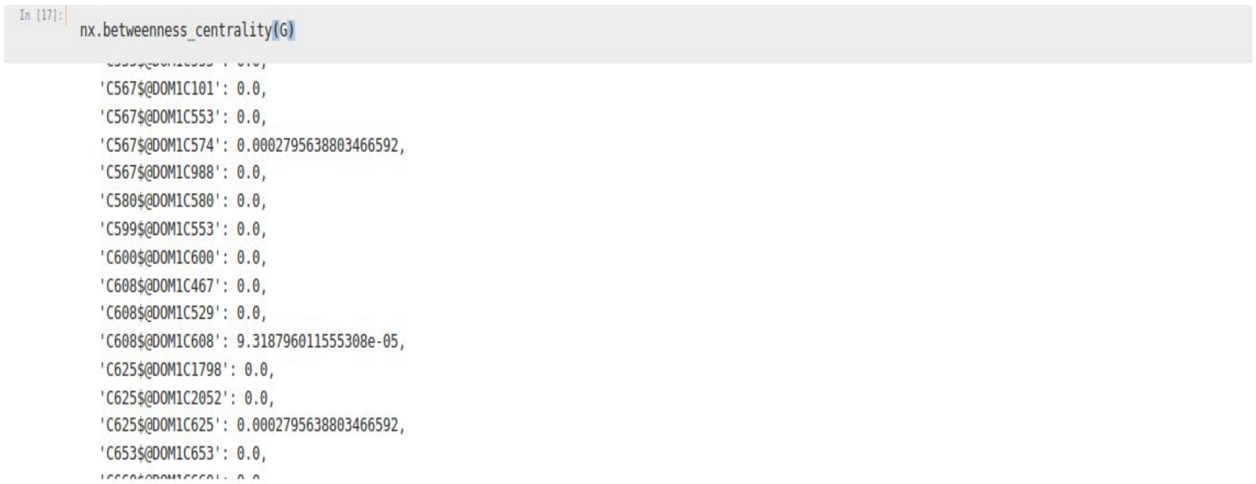
Clustering Coefficient



```
In [18]:
nx.clustering(G)
```

```
{'ANONYMOUS 6@C586C1250': 0,
 'ANONYMOUS 6@C586C586': 0,
 'C101$@DOM1C988': 0,
 'C1020$@DOM1C1020': 0,
 'C1021$@DOM1C1021': 0,
 'C1021$@DOM1C625': 0,
 'C1035$@DOM1C1035': 0,
 'C1035$@DOM1C586': 0,
 'C1069$@DOM1C1069': 0,
 'C1085$@DOM1C1085': 0,
 'C1085$@DOM1C612': 0,
 'C1151$@DOM1C1151': 0,
 'C1154$@DOM1C1154': 0,
 'C1164$@DOM1C625': 0,
```

Betweenness Centrality



```
In [17]:  nx.betweenness_centrality(G)
```

```
 'C567$@DOM1C101': 0.0,
 'C567$@DOM1C553': 0.0,
 'C567$@DOM1C574': 0.0002795638803466592,
 'C567$@DOM1C988': 0.0,
 'C580$@DOM1C580': 0.0,
 'C599$@DOM1C553': 0.0,
 'C600$@DOM1C600': 0.0,
 'C608$@DOM1C467': 0.0,
 'C608$@DOM1C529': 0.0,
 'C608$@DOM1C608': 9.318796011555308e-05,
 'C625$@DOM1C1798': 0.0,
 'C625$@DOM1C2052': 0.0,
 'C625$@DOM1C625': 0.0002795638803466592,
 'C653$@DOM1C653': 0.0,
```

References:

1. https://tsafavi.github.io/assets/pdf/liu-2018-graph-summarization.pdfhttps://tsafavi.github.io/assets/pdf/liu-2018-graph-summarization.pdf
2. https://en.wikipedia.org/wiki/Degree_distribution
3. https://pdfs.semanticscholar.org/2371/3212d5f2c4b2aef18489f6af21be05341060.pdf
4. https://books.google.com/books?isbn=1118099524
5. http://comp.social.gatech.edu/papers/Streaming-Twitter-Stats.pdf
6. https://en.wikipedia.org/wiki/Betweenness_centrality
7. https://hk.saowen.com/a/23f1121eeb4bca9dd45943b8a7010e1318af2431cdc9329195d56e7807fa158a
8. https://en.wikipedia.org/wiki/Closeness_centrality
9. https://en.wikipedia.org/wiki/Clustering_coefficient
10. https://neo4j.com/docs/graph-algorithms/current/algorithms/triangle-counting-clustering-coefficient/
11. http://barabasi.com/f/623.pdf
12. https://networkx.github.io/documentation/stable/
13. https://arxiv.org/ftp/arxiv/papers/1603/1603.07624.pdf

Acknowledgement

To complete this project, we received help from multiple sources and we are extremely grateful for the same.

Firstly, we would like to thank our Algorithms and Data Structures Professor, Dr. Qiong Cheng for giving us the opportunity to work this state of the art topic and for constantly guiding us and giving us the required material throughout the project.

Secondly, we are grateful to our teammates for showing enthusiasm and always making time for this project.