**CSCI 6444 Intro to Big Data & Analytics**

**Projcet 1 ---- Group 5**

**Group Member:**

**Yunfei Xu G21534523**

**Xiaohan Li G47313418**

**Siyuan Huang G43575455**

**Yanlong Shi G25529891**

Description:

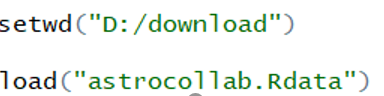
1. Data Set: (on Blackboard)

Collaboration Network of Astrophysicists’ Papers in ArXiv

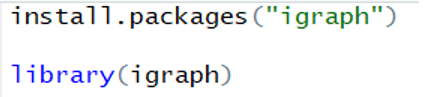
astrocollab.RData: Co-authorship network between scientists posting preprints on the Astrophysics E-Print archive.

Edge attribute 'weight' Edge weights, based on the number of common papers and the number of authors of these papers. See the publication(s) for the definition. Vertex attribute 'name' Author name.

Now we start to handle the Rdata file. The first thing we need to do is to set the index into the index we store Rdata. In my computer, I save astrocollab.Rdata into D:\download



2. Install the igraph package from one of the CRAN mirrors. You may also use igraphdata package and rgraph (included in the SNA package) as well.



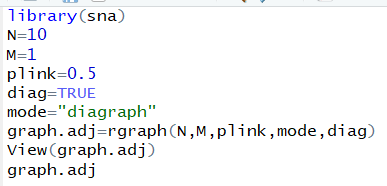
3. You will have to determine how to load the data into a data structure usable by the graph packages.



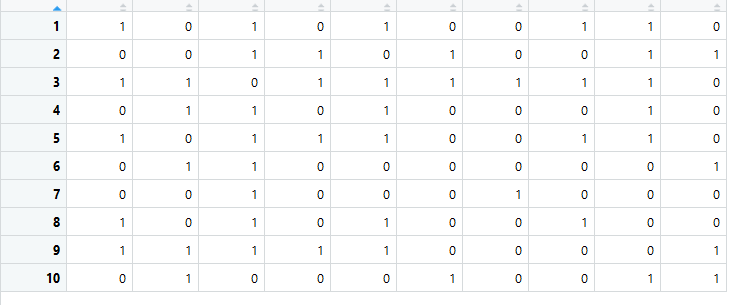
Problem 4.

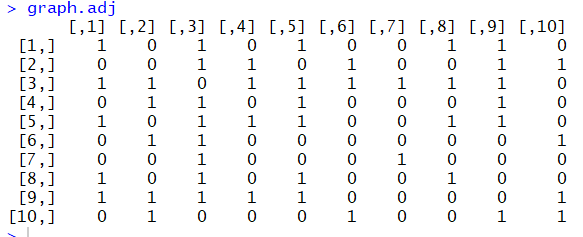
a. Create the graph

For better see the result totally, we decrease the number of vectors into 10.



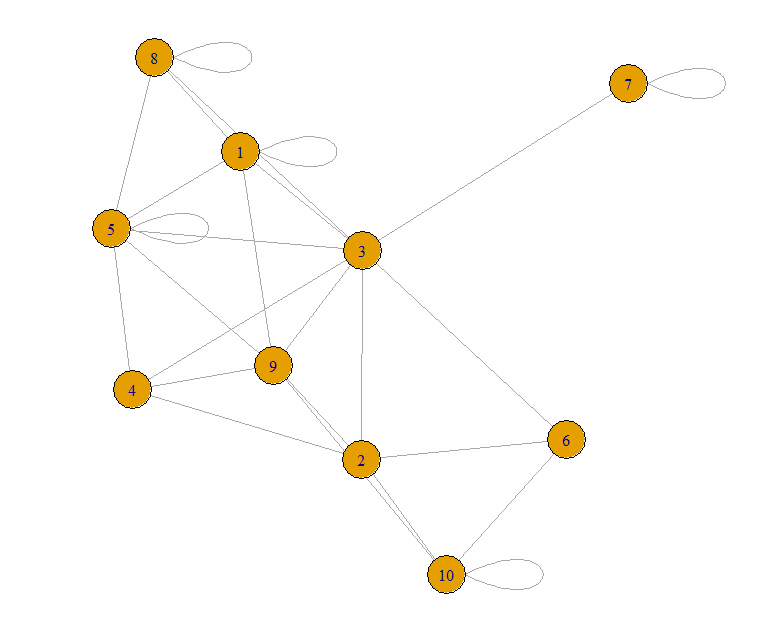
This is the result of View(graph.adj) and graph.adj





b. Plot the graph



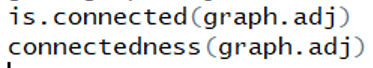


c. Shows the density of graph



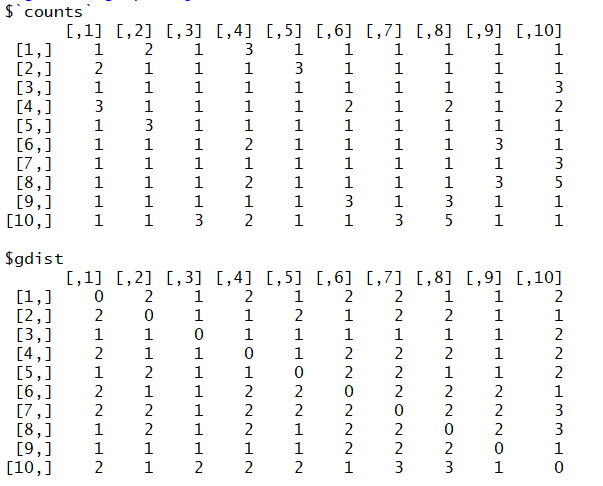


d. Shows the connectedness of graph



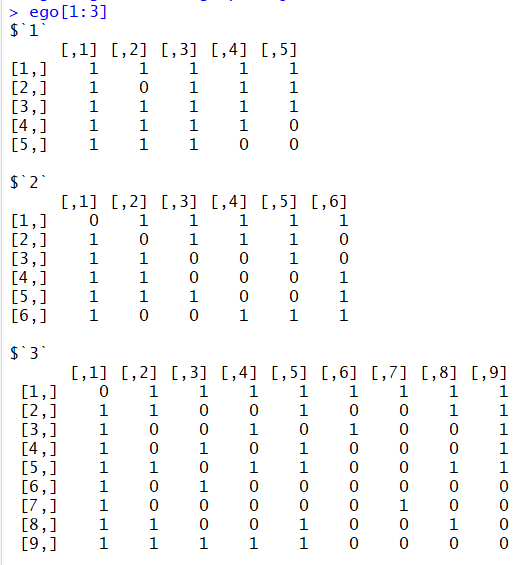
e. Shows the geodesic



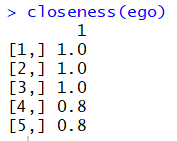


f. Shows the egocentric of graph



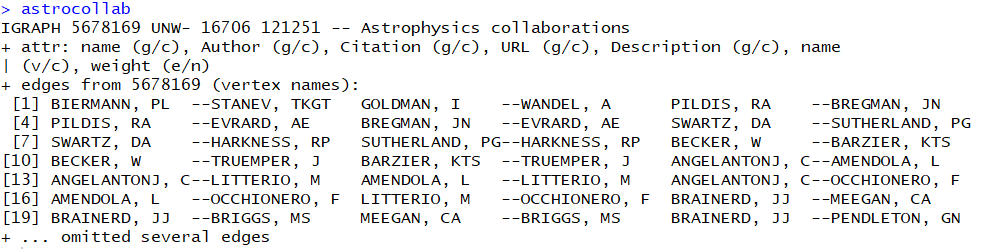


g. Shows the closeness centrality



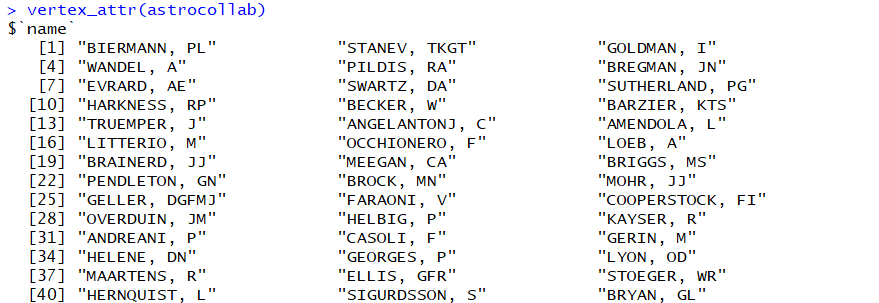
h. Load a Rdata file





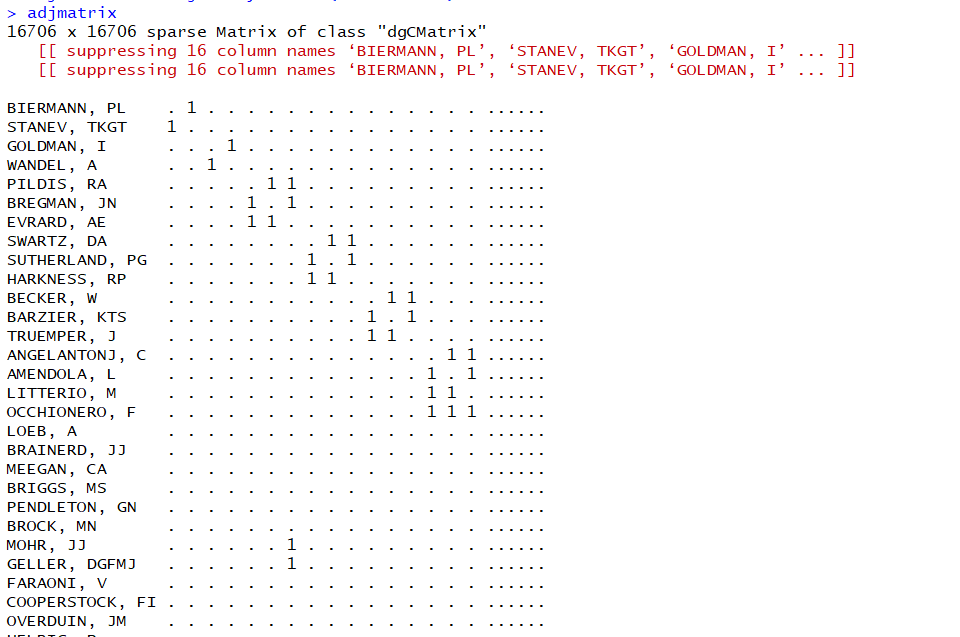
i. Finding Vertex Attributes





j. Represent the graph as a adjacency matrix

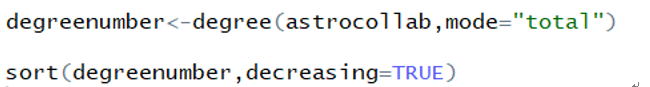


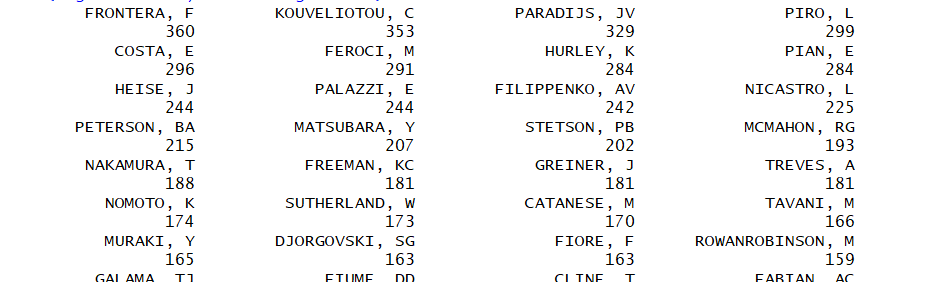


Problem 6.

a.

Central person means a person with the most degree. After upgrade the astrocollab using upgrade\_graph. We use the degree function in igraph package to get degree information for all vertexes. We choose mode as total because we don’t need to divide the in-degree or out-degree. We want to find the most degree person, we need to use sort function to sort all degree information (using decreasing). The result shown in next picture. We can see that the central person is Frontera, F with 360 degree. There is only one result.





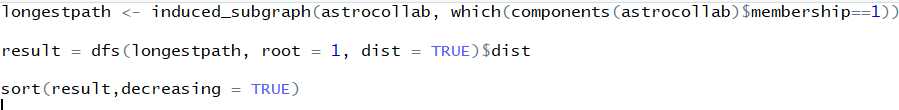
b.

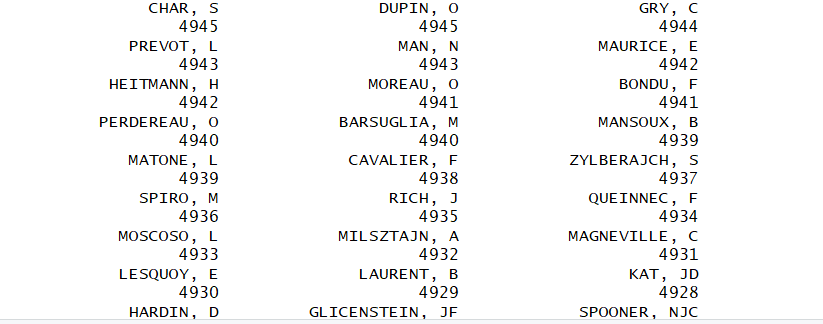
We need to calculate all vertices’ paths in maximal connected components of a graph to find a max value.

We use induced\_subgraph to create a subgraph of astrocollab, using components to get the largest connected component, save the result into membership and find all vertices which membership is 1.

After finish that, we use dfs algorithm and set dist = TRUE to return the distance from the root of the search tree, save the result into dist(result).

Finally, sort like a problem. You can find that CHAR, S and DUPIN, O both have 4945 distance, which are the longest paths. They are multiple.



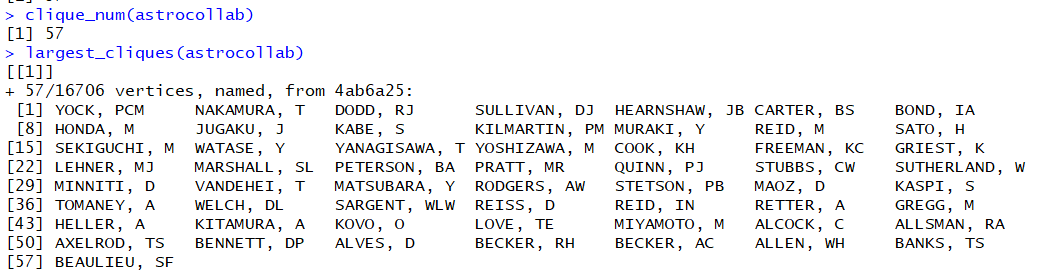


c.

To find the largest cliques, we use clique\_num(astrocollab) to calculate the size of the largest clique(s), then use largest\_cliques(astrocollab) to shows all largest cliques.

It shows that there is only one largest clique, which size is 57.





Using function count\_max\_cliques(astrocollab) to counts the maximal cliques. The result is 15794. So we can say that there are multiple cliques with different degree.





d.

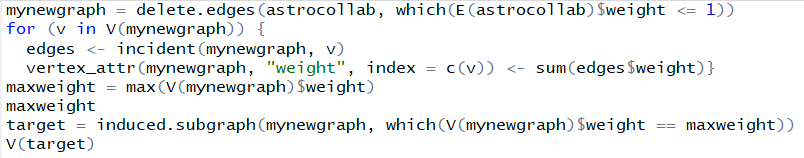
The weight of edges are different, it means we can’t just use ego() function to find the highest ego. We also need to consider the influence of weight of edges.

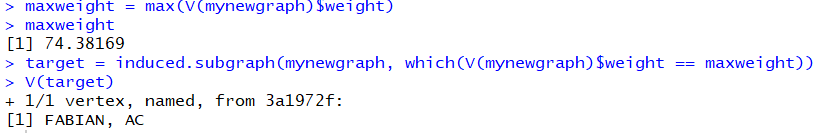
Because there are too many edges, we have to delete some un-important (weight is low) edges. So I delete all edges with weight lower than 1. E() shows a collection of edges.

I set a for loop, traverse all vertices in rest of the graph, store all incident edges of that vertices into edges. After sum all weight of edges of that node, using function Vertex\_attr to query graph in weight attribute.

We find the max weight of the graph. The result is 74.38169.

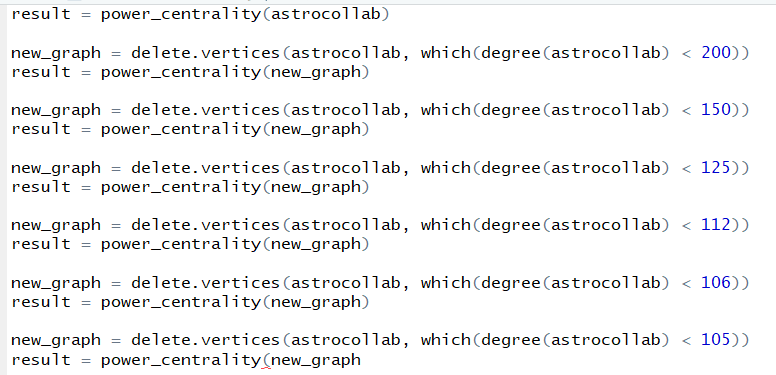
Find all target which weight is equal to the max weight and list all of the result, we find that there is only one result. So there is only one person that have the highest ego.



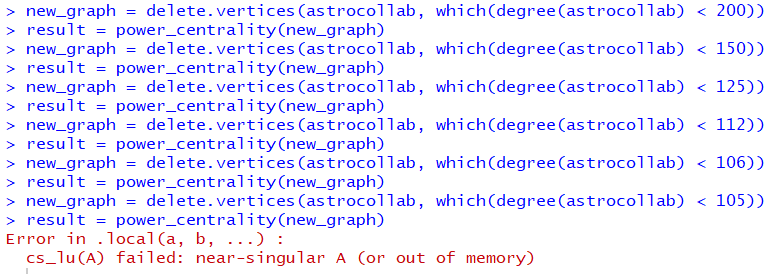


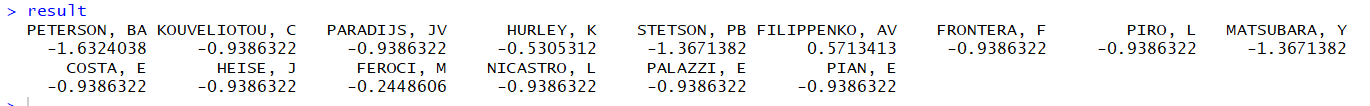
e.

When we use power\_centrality to calculate the value, we find that if whole graph is included, that will be a error. So we use delete.vertice to limit the scale of graph by different degree. And you can see that from 200-106, all is available, but when the degree is <=105, there will have error. So we can find that the power centrality is equal to 106.

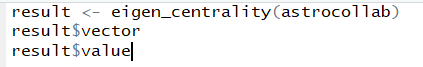






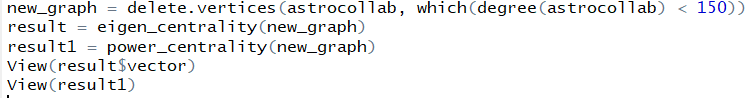


For the betweenness centrality. We find that we use function eigen\_centrality, which can be used by any scale of graph. With the value of 33.75974. That is the result of betweenness centrality.





I compare the node in two set. In same scale, which is degree < 150, I find the nodes are totally equal, except the value is different.



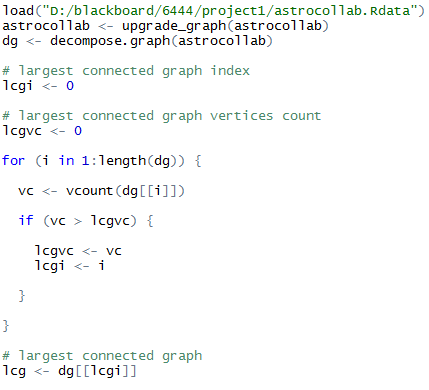
Betweenness Centrality: identifies an entity's position within a network in terms of its ability to make connections to other pairs or groups in a network.

Closeness: nodes that can reach other nodes via the fewest intermediaries are relatively independent/autonomous of others.

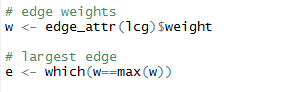
For the concept of two centrality, any node connected to others that will be displayed into both centrality list. That is why two list nodes are totally equal.

Problem 7.

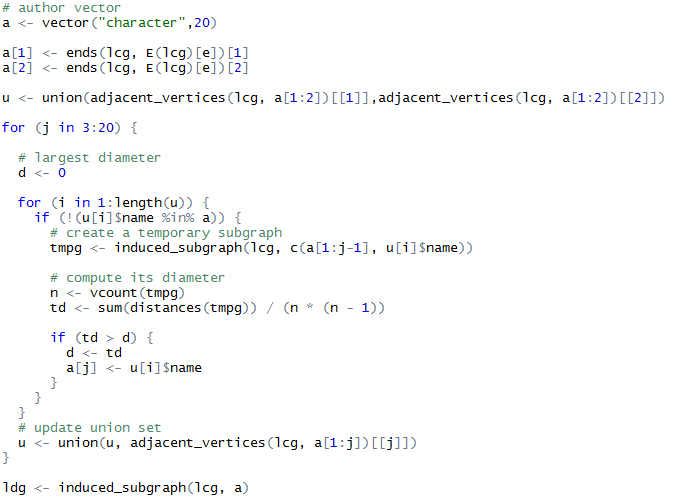
Firstly, find the largest connected graph index then obtain the largest connected graph.



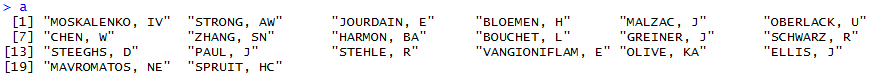
Then, compare weights of each edge and get the largest edge



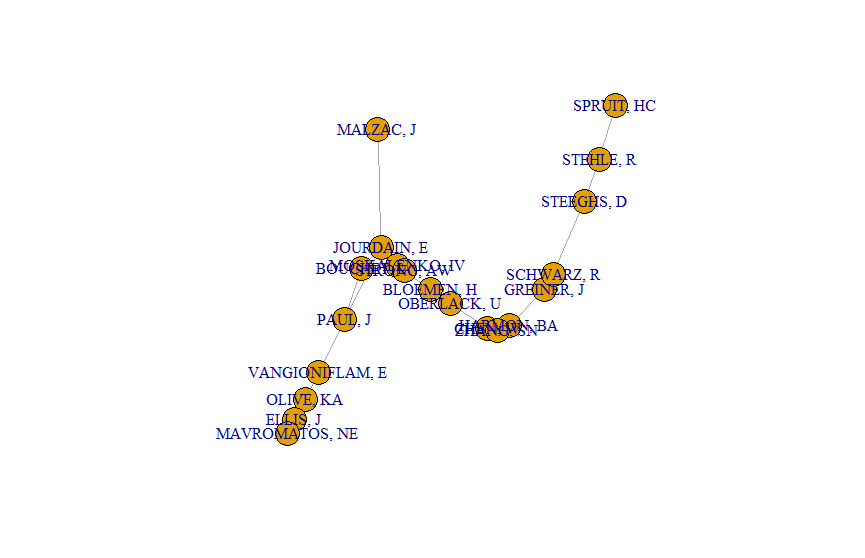
To get the authors’ network, we create an author vector first and then use for loop to find their largest diameter and keep update the vector.



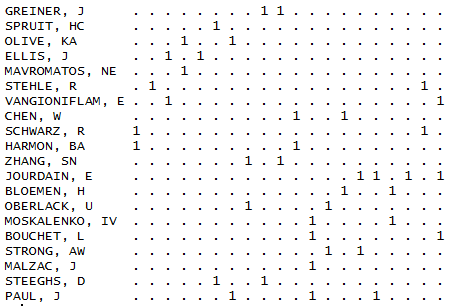
The greatest diameters for 20 nodes in the network is 17.578. The 20 authors are:



Their network is like this:



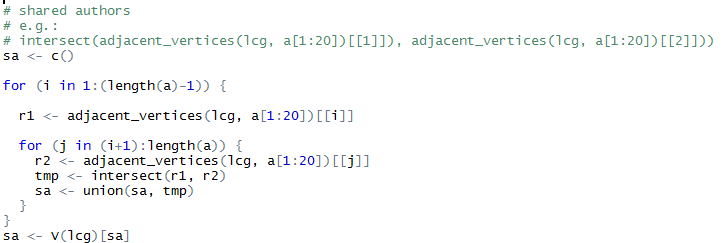
And its adjacency matrix:



From above we can see that if a subgraph has largest diameter, the nodes in it basically would not overlap. And its more like a spanning tree, since tree’s nodes would have least possibility to overlap, which lower the diameter. However, noted that a subgraph has largest diameter doesn’t mean that it is a minimum spanning tree, though it’s also a spanning tree. The reason is simple: an edge of a minimum spanning tree is not necessary the shortest path of these 2 nodes, because maybe these 2 nodes can have a shorten path by through other nodes. Therefore, a subgraph has largest diameter is not necessary a minimum spanning tree.

c.

To calculate shard common authors, we use intersect function to compare each pair of vertices and store their common subset.



They have 232 shared authors in total:

