ADM Homework 4

Group – 25

Alessandra Griesi, Malik Bekmurat, Venakta Naga Sai Krishna Abhinay Pochiraju
https://github.com/pvnskabhinay/ADM-HW4-Group-25

The Problem:

In this homework we are dealing with networks. In particular, we will carry out some information from Computer Scientists network, by applying various graph methodologies. Here we use DBLP dataset, we are given two json files. full_DBLP (which contains the whole network) and reduced_DBLP(which is a reduced version of the full_DBLP). In the first part, we process the json and parse it. We create a graph G, whose nodes are authors. The nodes are connected if the authors have a common publication, and, the weight of the edges are calculated by Jaccard Similarity. Then, the second part is computing some statistics and visualizing them. And finally the third part is computing generalized version of the Erdös number.

Our Approach:

We have written a file: modules.py, which has the functions that we use in our main code. The final main code is named: ADM-HW4-Group-25.py. The file, modules.py has the following functions:

- jaccard_distance(list1, list2)
- plot_graph(graph)
- dijkstra(graph, source, destination)
- dijkstra2(graph, sub_gr)

So, we import modules.py, for using its member functions in our code. We did that by: 'import modules as mo', and then use the functions as: mo.plot_graph(graph).

Our first step was to parse the data from the given json files. The given json files (both the full_dblp and the reduced_dblp) has a list of the following keys for every index of the list:

- authors: it is a list of dictionaries, with keys author (name of the author) and author_id
- id_conference: the conference id
- id_conference_int: the numerical conference id
- id_publication: the publication id
- id_publication_int: the numerical publication id
- title: the title of the publication

Some of the parsed data is explained below:

- *publications* : It is a dictionary, which has publications id's and the list of authors that published in that particular publication.
- *authors* : It is of type defaultdict(list), which we created for easy access of authors.

We worked on both the json files and so we will present the results for both in this document for comparison.

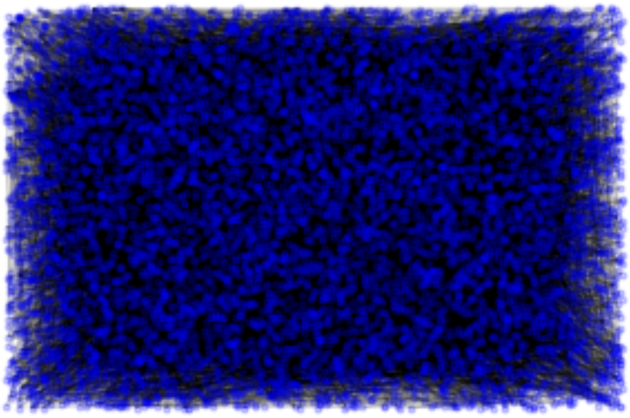
1st part: Parsing and plotting the Graph of given data

We used networkx module to work on graphs here. Adding nodes, and connecting them if they have a publication in common, we got the following results.

	reduced_dblp.json	full_dblp.json
graph info	Type: Graph Number of nodes: 7771 Number of edges: 16489 Average degree: 4.2437	Type: Graph Number of nodes: 904646 Number of edges: 3679297 Average degree: 8.1342

Plot of the graph:

reduced_dblp



full_dblp

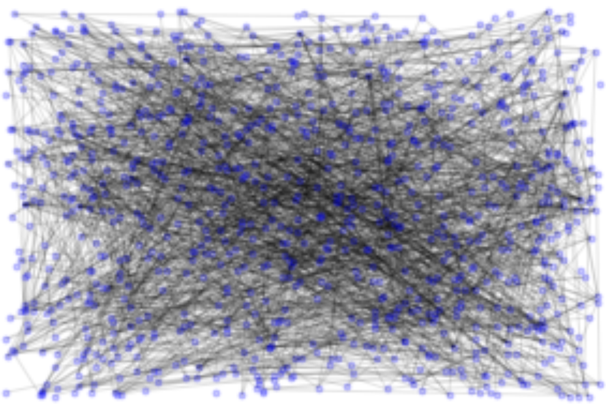
2[nd] Part: Computing statistics and visualizing them

2a) Here we take conference id as input and we looked through the parsed data and made a list of all the authors who attended that conference. And then using the subgraph() function in the networkx module we plotted the following subgraph.

	reduced_dblp.json	full_dblp.json
Input	conference id: 3052	conference id: 1
graph info	Type: Graph Number of nodes: 120 Number of edges: 117 Average degree: 1.9500	Type: Graph Number of nodes: 949 Number of edges: 1849 Average degree: 3.8967

reduced_dblp

full_dblp



- **DEGREE CENTRALITY**

Degree is a simple centrality measure that counts how many neighbours a node has. We calculated the degree centrality of the full graph and the induced subgraph. The plots of degree centrality are given below.

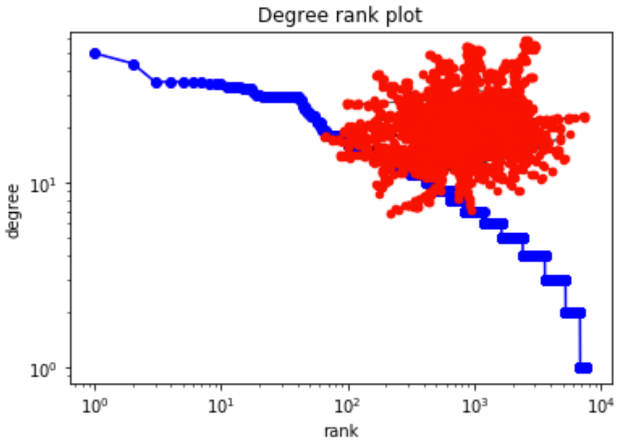
First we plotted a log-log plot with graph in inset. And then we plotted histograms for degree centrality.

We calculated the degree centrality measure using the following logic:

```
# DEGREE CENTRALITY
degree_centrality={}
n=len(H)
for each in H:
    degree_centrality[each]=len(H.edges(each))/(n-1)
print(degree_centrality)
```

Degree rank plot for the whole graph:

reduced_dblp

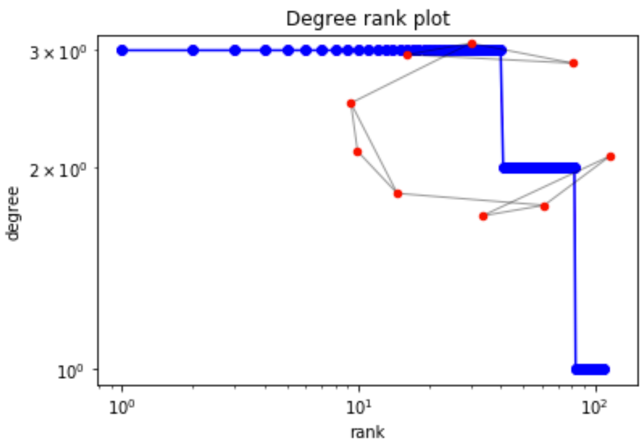


full_dblp

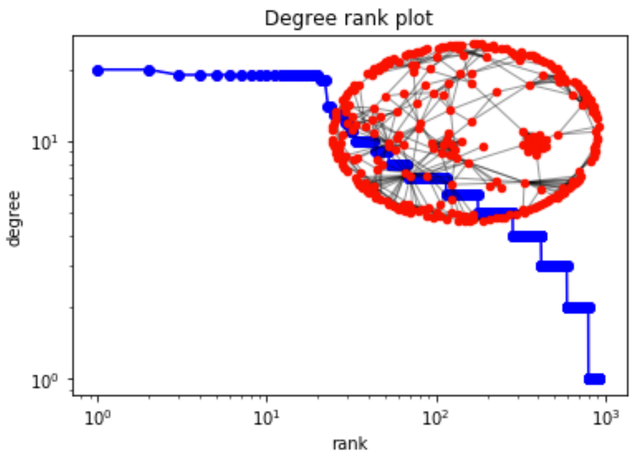
data rate exceeded for full_dblp

Degree rank plot for the induced subgraph:

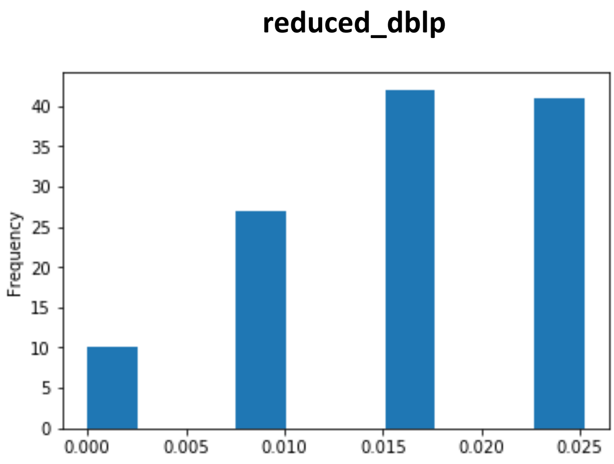
reduced_dblp

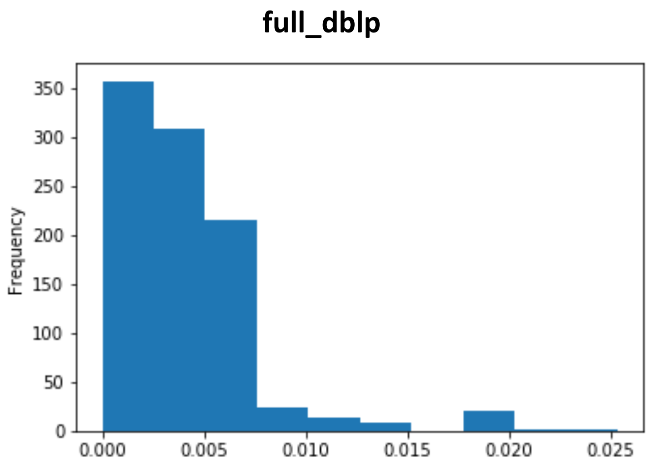


full_dblp



The histogram plots for degree centrality measures:

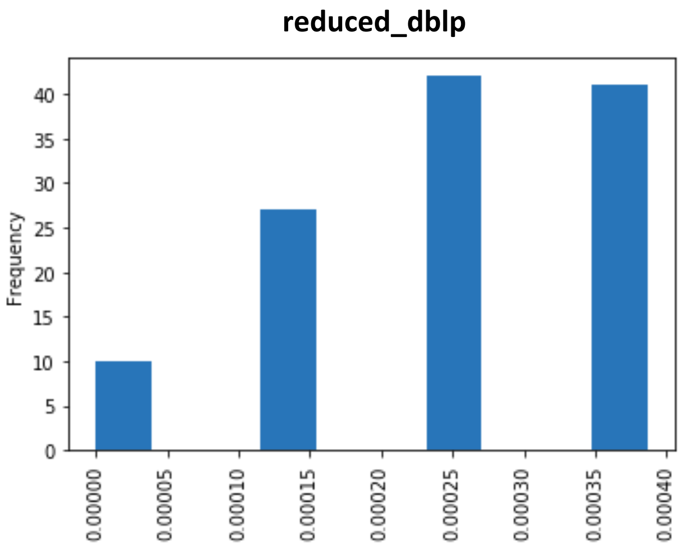


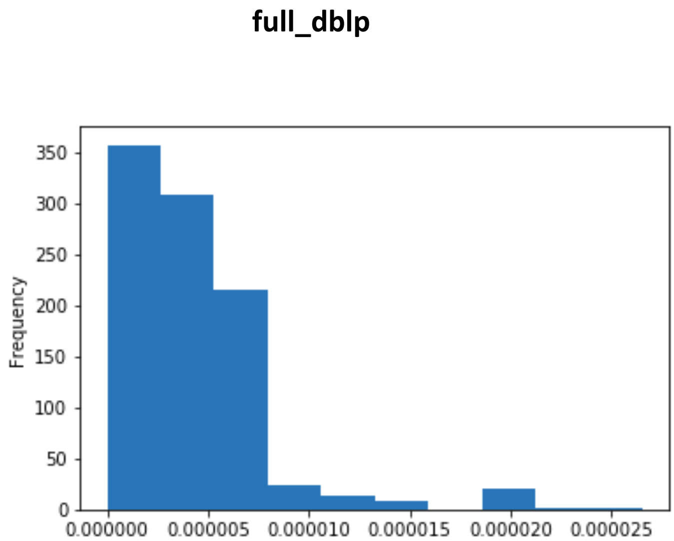


- **ABSOLUTE CENTRALITY**

We plotted the following plots for absolute centrality measures using the following logic:

```
#ABSOLUTE CENTRALITY
centrality={}
s=1.0/(len(graph)-1.0)
centrality=dict((n,d*s) for n,d in H.degree_iter())
```



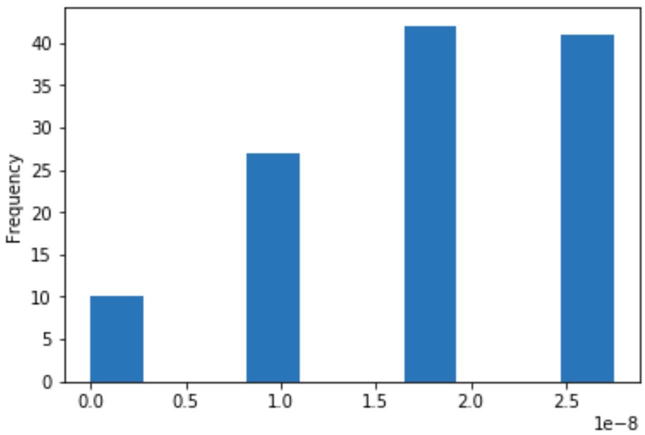


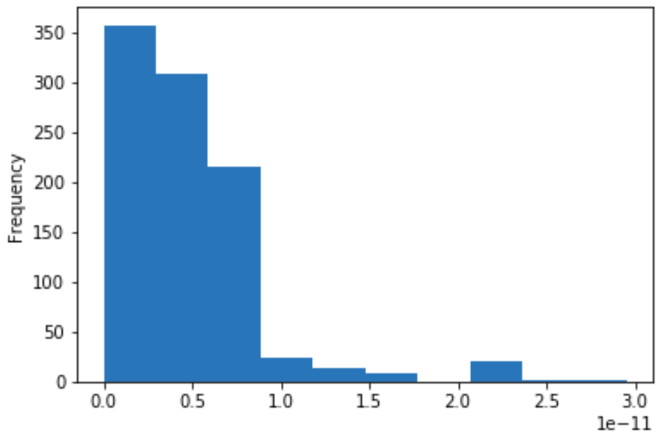
- **BETWEENESS CENTRALITY**

We plotted the following plots for betweeness centrality measures using the following logic:

```
# BETWEENNESS CENTRALITY
betweeness_centrality={}
n=len(H)
denom=n**2 - 3*n + 2
for id_author in centrality.keys():
   betweeness_centrality[id_author]=centrality[id_author]/denom
```

reduced_dblp **full_dblp**



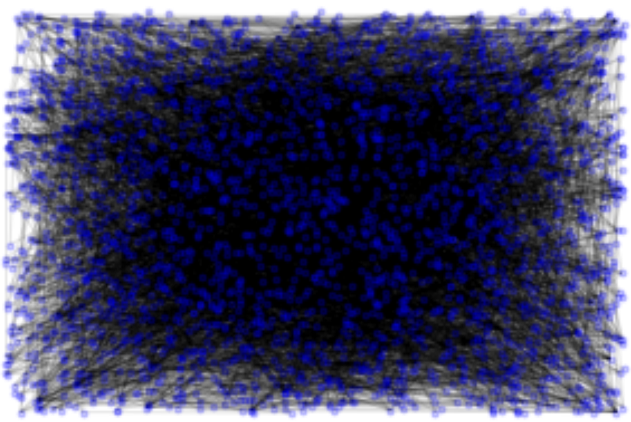


2b) Here we take an author and an integer d as input, and plot the subgraph induced by the nodes that have hop distance (i.e., number of edges) at most equal to d with the input author.
Using the inbuilt function ego_graph from the networkx module, we plotted the induced subgraph and it is as follows:

	reduced_dblp.json	full_dblp.json
Input	author id :256176 d: 10	author id :256176 d: 10
graph info	Type: Graph Number of nodes: 2706 Number of edges: 7657 Average degree: 5.6593	Type: Graph Number of nodes: 791863 Number of edges: 3553849 Average degree: 8.9759

Plots:

reduced_dblp **full_dblp**



3[rd] Part: Computing a generalized version of the Erdös number

3a) Here we take author_id as an input and find the weight of the shortest path that connects the input author with Aris. We implemented a function dijkstra() which calculates the weight of the shortest path from a source node to any given node in the graph. The function is implemented using heap, where we store the list of nodes in a structured manner. We store the node and minimum distance in the heap. The function returns the weight of the shortest path that connects the input author with Aris. Following are our observations:

	reduced_dblp.json	full_dblp.json
Input	Enter author_id:202882	Enter author_id:202882
Output	The weight of the shortest path that connects 202882 with Aris is: 8.632570207570208	The weight of the shortest path that c onnects 202882 with Aris is: 4.6769623131903835

3b) Here we take a subset of nodes (Cardinality smaller than 21) as input and calculate for each node of the graph, its Group Number. We implemented a function dijkstra2() which calculates the weight of the shortest path from closest node in sub group to each node in graph. This function is also implemented using heap. The function returns a dictionary with all the nodes as keys and their respective values as Group Number for that particular node. Following are our observations:

	reduced_dblp.json
Input	Enter a subset of nodes (Cardinality smaller than 21) separated by spaces: 256176 273893 44955 256177 523303
Output	The Group number of the node '256176' is : 0 The Group number of the node '365188' is : 0.5 The Group number of the node '20793' is : 0.6842105263157895 The Group number of the node '18263' is : 0.8 The Group number of the node '365027' is : 0.8421052631578947 The Group number of the node '269977' is : 0.85 The Group number of the node '272068' is : 0.85 The Group number of the node '20994' is : 0.5 The Group number of the node '21131' is : 0.8947368421052632 The Group number of the node '273893' is : 0 The Group number of the node '356527' is : 0.8947368421052632 The Group number of the node '226733' is : 0.9090909090909091 The Group number of the node '16617' is : 0.9473684210526316 The Group number of the node '16618' is : 0.9473684210526316 The Group number of the node '44955' is : 0 The Group number of the node '53221' is : 0.9473684210526316 The Group number of the node '220576' is : 0.0 The Group number of the node '255654' is : 0.9473684210526316 The Group number of the node '256177' is : 0 The Group number of the node '272067' is : 0.9473684210526316 The Group number of the node '396772' is : 0.0 The Group number of the node '433893' is : 0.9473684210526316 The Group number of the node '449967' is : 0.9473684210526316 The Group number of the node '456091' is : 0.0 The Group number of the node '490807' is : 0.9473684210526316 etc.

full_dblp.json	
Input	Enter a subset of nodes (Cardinality smaller than 21) separated by spaces: 256176 273893 44955
Output	The Group number of the node '256176' is : 0 The Group number of the node '365188' is : 0.7777777777777778 The Group number of the node '20793' is : 0.8125 The Group number of the node '365027' is : 0.8928571428571429 The Group number of the node '273893' is : 0 The Group number of the node '272068' is : 0.9189189189189189 The Group number of the node '20994' is : 0.9 The Group number of the node '269977' is : 0.9361702127659575 The Group number of the node '272067' is : 0.9473684210526316 The Group number of the node '356527' is : 0.9473684210526316 The Group number of the node '695406' is : 0.9473684210526316 The Group number of the node '523303' is : 0.875 The Group number of the node '523302' is : 0.75 The Group number of the node '18263' is : 0.9545454545454546 The Group number of the node '456091' is : 0.9 The Group number of the node '256177' is : 0.9565217391304348 The Group number of the node '396772' is : 0.9090909090909091 The Group number of the node '44955' is : 0 The Group number of the node '255654' is : 0.96 The Group number of the node '220576' is : 0.75 The Group number of the node '21131' is : 0.9649122807017544 The Group number of the node '226733' is : 0.9655172413793104 The Group number of the node '433893' is : 0.967741935483871 The Group number of the node '449967' is : 0.967741935483871 The Group number of the node '518711' is : 0.96875 The Group number of the node '16617' is : 0.9696969696969697 The Group number of the node '490807' is : 0.9705882352941176 The Group number of the node '53221' is : 0.9714285714285714 The Group number of the node '490865' is : 0.972972972972973 The Group number of the node '255275' is : 0.9761904761904762 The Group number of the node '490729' is : 0.9777777777777777 The Group number of the node '225947' is : 0.9803921568627451 The Group number of the node '255328' is : 0.9824561403508771 The Group number of the node '16618' is : 0.9830508474576272 The Group number of the node '19211' is : 0.9850746268656716 The Group number of the node '114821' is : 0.9871794871794872 The Group number of the node '638767' is : 0.75 The Group number of the node '612633' is : 1.5401785714285714 The Group number of the node '20794' is : 1.5744047619047619 The Group number of the node '52567' is : 1.601058201058201 The Group number of the node '451516' is : 1.6125 The Group number of the node '664683' is : 0.8571428571428572 The Group number of the node '364863' is : 1.623931623931624 The Group number of the node '264763' is : 1.6339285714285714 The Group number of the node '264765' is : 1.6339285714285714 etc.