Assignment 1
Social Network Analysis for Computer Scientists

Xiang He

Social Network Analysis for Computer Scientists
The Netherlands
15-09-2022

October 2, 2022

1 Exercise 1

1.1

Indegree: The number of reverse neighborhood of node v .

$$\left|N'(v)\right|$$

Outdegree: The number of neighborhood of node v.

$$|N(v)|$$

1.2

combine degree: The counts of reverse neighborhood and neighborhood of node v.

$$\left|N'(v)\right| \cup |N(v)|$$

1.3

Reverse k-neighborhood: All nodes that are between 0 and k steps reach nodes in W, which helps us to find the set of nodes that take k steps to reach a certain node.

$$N'_k(W) = N'\left(N'_{k-1}(W)\right) \cup N'_{k-1}(W)$$

1.4

Choosing a random point $v \subset S$, than recursion k-neighborhood algorithm and reverse-k-neighborhood algorithm on v, save nodes in two groups. Create a new set K from the points that exist in both groups, if $S \subseteq K$ then S is a a strongly connected component of the network.

```
def find_strongly_connected_component(graph_nodes_list, nodes_list):
    nodes_list.append(K_neighborhood(graph_nodes_list))
    nodes_list.append(reverse_K_neighborhood(graph_nodes_list))
    return find_strongly_connected_component(graph_nodes_list, nodes_list)

node_list = []
all_nodes_list = find_strongly_connected_component(S.nodes(), node_list)
if S.nodes() <= all_nodes_list:
print("The nodes set S is a strongly_connected_component")
```

1.5

The ratio of (reversed)neighborhood edges to all potential edges.

$$\frac{|N'(v)| + |N(v)|}{\frac{1}{2}n(n-1)}$$

1.6

It's a Bipartite graph.
The path in this graph always contains an even number of nodes and edges, and the shortest path should contain 4 nodes.

1.7

For the N(v) of each node are most nodes with the similar degree, extent value is close to 1, otherwise it is close to -1.

$$\frac{deg(v)}{deg_{avg}(N(v))}$$

1.8

The node v take max_k steps in $N_k(v)$ to node v_1, than node v_1 take max_k steps in $N_k(v_1)$ to node v_2 , the path of v, v_1, v_2 is the diameter.

$$max_k \subset \{N_k(W), N'_k(W)\}$$

1.9

For all $N_2(v)$ in same node

```
def dfs( graph, node, path, Limit)->int:
path.append(node)
if len(path) == Limit :
    if node == path[0]:
        return 1
    else:
        return 0
else:
    count=0
    for next_node in graph.get_adjacent(node):
        # if the next node is not in the path, go to it
        if next_node not in path[1:]:
            count=count+dfs(graph,next_node,path,Limit)

    return count
```

2 Exercise 2

All program are write in Python language

2.1

Generating graph by `nx.read_adjlist("large.tsv", create_using=nx.DiGraph())` , than calculate directed links by `.number_of_edges()`

large Network: 511718
medium Network: 13294

2.2

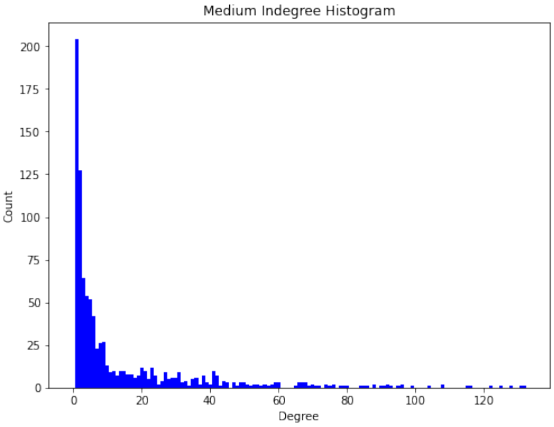
calculating by `.number_of_nodes()`

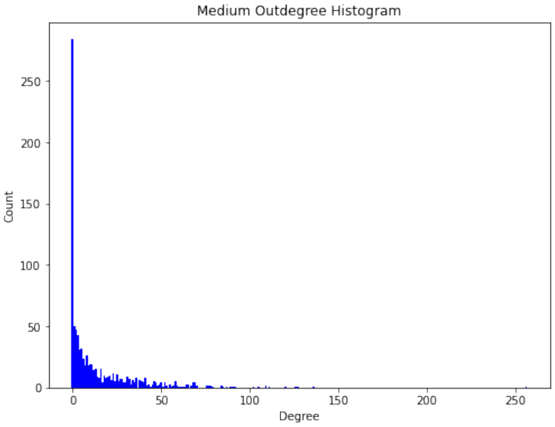
large Network: 21461
medium Network: 928

2.3

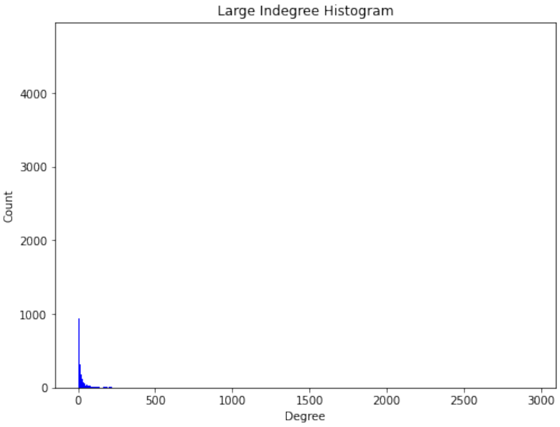
Using `collections.Counter` to calculate and save all nodes (in)out-degrees, than generate a counts list and draw by matplotlib.

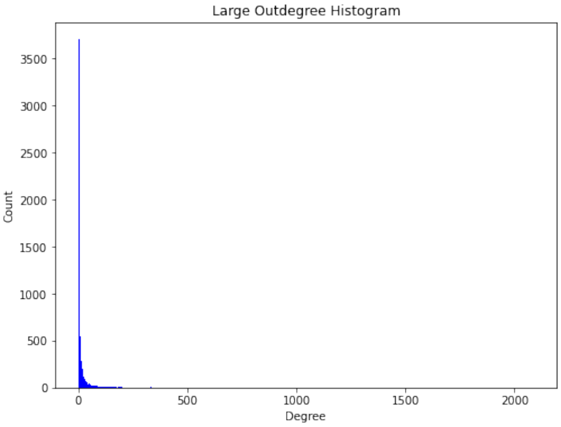
In Medium graph

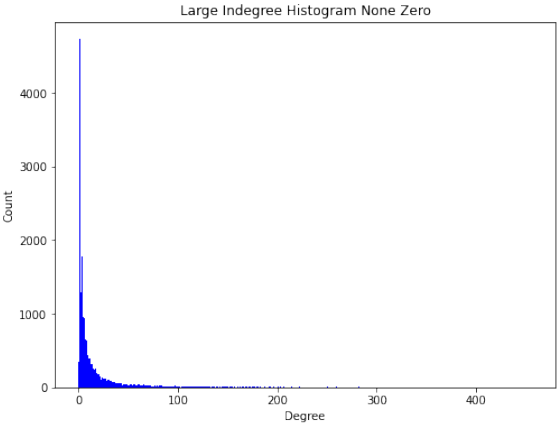


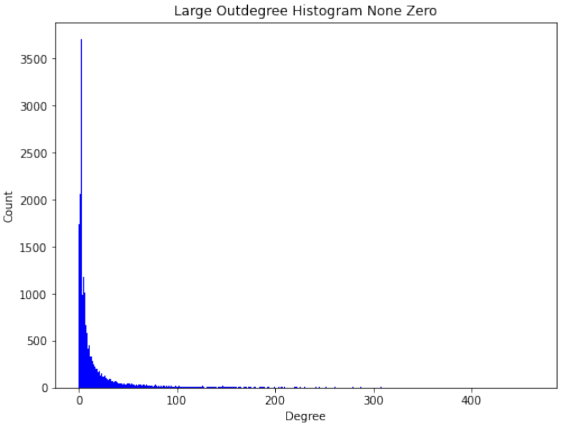


In Large graph
Some values cannot be displayed clearly due to the large number of data, therefore additional graphs are drawn that do not contain zero count degrees.









2.4

Calculating by networkx tools

```
max(nx.strongly_connected_components(G), key=len)
max(nx.weakly_connected_components(G), key=len)
nx.number_strongly_connected_components(G)
nx.number_weakly_connected_components(G)
```

number of strongly connected components in medium network: 305 nodes: 623 edges: 10512
number of weakly connected components in medium network: 1 nodes: 928 edges: 13294
number of strongly connected components in large network: 2154 nodes: 19299 edges: 505153
number of weakly connected components in large network: 14 nodes: 21434 edges: 511700

2.5

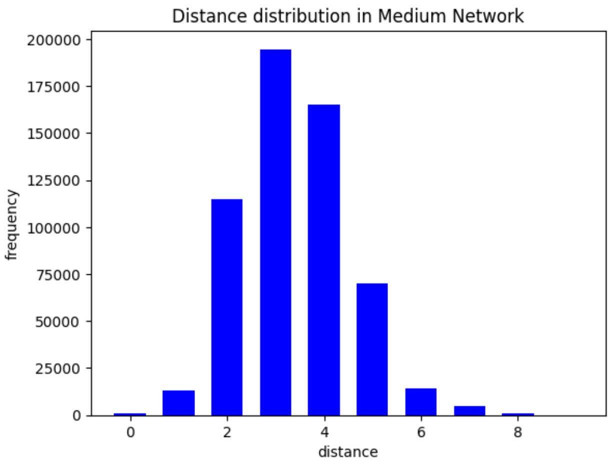
Calculating by `nx.average_clustering(G)`

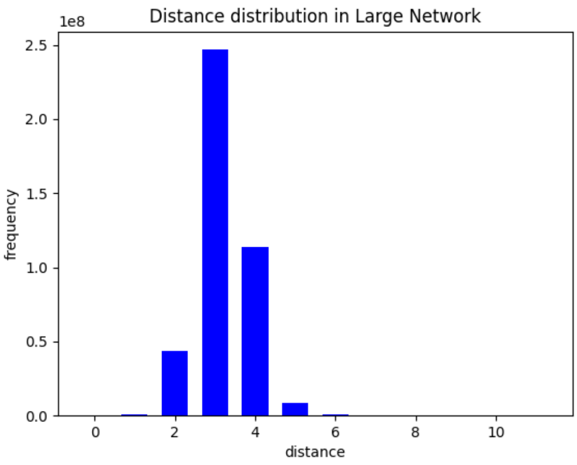
Medium graph: 0.21287579609762455
Large graph: 0.38135345230825596

2.6

Distance distribution of largest weakly connect component in Medium graph calculate by `nx.all_pairs_shortest_path_length(G)`.

To the Large graph, extract the nodes-list from largest weakly connect component, than calculate the shortest path of each node (`nx.single_source_shortest_path_length(G, i)`) with multiprocessing tools `concurrent.futures`. The nodes list is partitioned by 200 nodes per list, and each partition will be handled by a process. (almost 15min)





2.7

Graph generate by Gephi, and Use Force Atlas layout
Repulsion strength: 1500.00
Maximum displacement: 20.0
Nodes centrality partition by Eigenvector Centrality, and rank by Eigenvector Centrality in five sections bar. In the preview, Edges Opacity Setting is changed as 30