Practical Exercise 1

TA in charge -Oded Salton

All the data and assumptions in this exercise are for educational purposes only. Please follow official guidelines and scientific advise regarding COVID-19.

Due date:26/04/20

In this exercise you will practice some of the network concepts learned in class.

background story :

during the beginning of 2020 , the coronavirus has begun to spread outside of China and it was a question of time until the virus would begin to spread in Israel. As a preparation for the coming of the plague the government recruited you as a specialists for netwok behavior in order to simulate of the behavior of the contagion among the Israeli community in order to choose the best way in order to prevent a crisis .

Guideinstructions :

There are 3 python files :

1. Main_students.py - includes the signature of the function you need to implement.
2. Hw1_part1.py - contains the functions you need to implement in the mandatory part of the exercise (you are allowed to add other functions in this file).

In addition you will receive a (small) dataset representing the social connections in one particular area. The data contains two files.

links_dataset.csv: this file represents the topology of the graph (11781 connections inside the community, between 5878 nodes). The file contains three columns for the ID of the two users (source ,target) and the weight of the connections .

infection_information_set.csv : the file is including the latest COVID-19 test results for every node of the network. This input may differ between questions. The title of the columns specifies the relevant question.

Python packages that you are allowed to use (all in their python 3.6 version) :

networkx, numpy , pandas , matplotlib , scipy

*if when you run the program in the VM and it doesn't find the packages use the command /usr/bin/python3.6 <filename>.py

Part A (25 points)

In part A you should use the links_datadset.csv file in order to create an **undirected unweighted graph** in which each row represent an edge between two nodes in the network. In part A we will ignore the weight of each edge and will not use the infection information set.

Q1:

Implement the function graphA(links_data).
stage 1: The function should accept the dataset and build a undirected unweighted graph.
implement function : calc_best_power_law(G)
stage 2 :after creating the graph, the function need to calculate the power law distribution with the parameters alpha and beta that minimize the square error between the theoretical and empirical distriburions. It should return alpha and beta as a float parameters. (alpha and beta are called "c" and "a" respectively in the notation of Section 18.2 in the book)

Stage 3 :

Implement the function

plot_histogram(G, alpha, beta).

The function should plot a histogram of G nodes' degrees as a log-log graph (as in fig. 18.2 in the book), and the power law with parameters alpha,beta.

Stage 4 : You should submit a pdf with a plot of the histogram and the power law approximation. (the plot picture would be open only on your computer)

Q2 :

Implement the function G_features (G) that receives the unweighted graph G and returns for every node in G :

a. Closeness centrality - n-1 / sum of distances , where n is the number of nodes in the certain connected component .
and we if there is no path between two nodes it considered as 0

b. Betweenness centrality

The output must be in a **dictionary of dictionary format**. see example in the appendix.

Part B (55 points):

In class we learned about the Linear Threshold model (click here) on this part we will use the model in order to simulate the spread of the virus inside the community.
In questions 3 and 4 we will use the following assumption:

- The threshold of all nodes is 0.41

Q3. Implement the function create_undirected_weighted_G(links_data,infection_data,'Q3')

The function should return an **undirected weighted** graph WG. In addition, each node should have an attribute for whether it is infected or not. Initialize its value according to Q3 in infection_data.csv.
Implement the function run_k_iterations(WG, k, Threshold)
in this function you'll run k iterations of contagion which on every iteration if a node was exposed to the virus in bigger or equal level of exposure the node would get infected.
The function will return a dictionary S of length k+1 where the key is the number of iteration and the value is the list of nodes that got infected (S_0 is the initial subset of infected nodes from the data).

Implement the function calculate_branching_factor (S,k) in order to calculate the branching factor for every level of the infection spread.
The function returns an array of length k, where the branching factor of level i is defined to be the (number of nodes in level i)/(number of nodes in level I-1). That is:

$R_i := \frac{|S_i|}{|S_{i-1}|}$ (branching at level i)

Try the average branching factor (R) under different thresholds. Check for yourself: How many nodes are eventually infected when R<1 and when R>1 ?

Q4. After a successful simulation of contagion process, your next mission is to find a good mitigation strategy, you need to do it according to the following assumption :
placing a node in quarantine removes it from the graph, so it cannot be infected or infect others.

Implement the functions:

find_maximal_h_deg(WG,h) [find h nodes with maximal degree]

calculate_clustering_coefficient(WG,candidate_nodes) [compute CC for each node]
* you must calculate it without using the function clustering of network

infected_nodes_after_k_iterations(WG, k , Threshold) [The total number of nodes that are infected after k iterations]

slice_dict(dict_nodes,number) [return first [number] nodes]

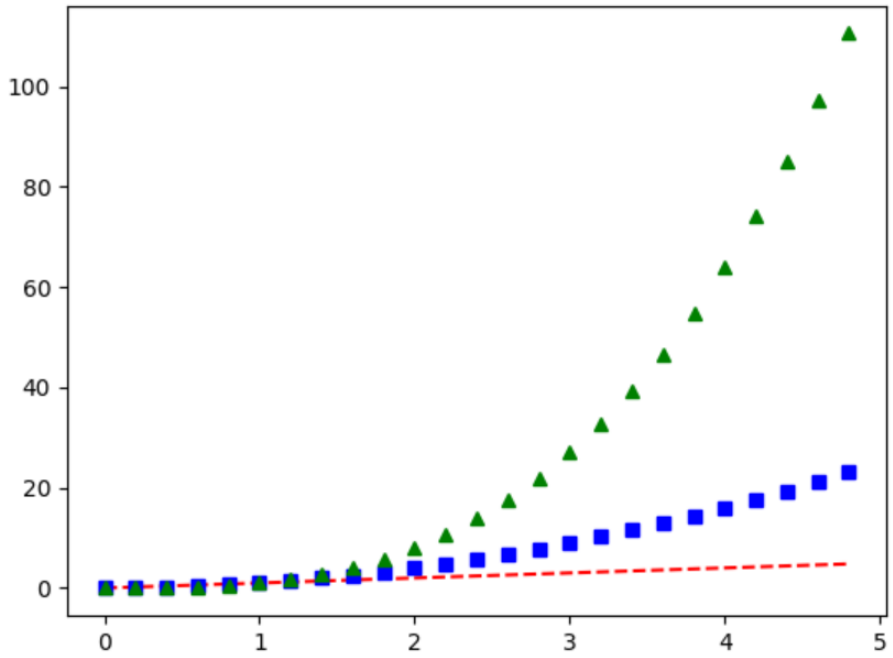
nodes_removal(WG,nodes_list) [remove nodes from the graph]

The code in main_students.py uses these functions to isolate nodes in three different orders: at random, and by ascending / descending order of their CC.
Finally, implement the function

graphB(number_nodes_1, number_nodes_2, number_nodes_3) – you should run this function only on your private computer .

It should plot three graphs, each representing the final number of infected nodes (on the Y axis) as a function of the number of removed nodes (on the X axis).

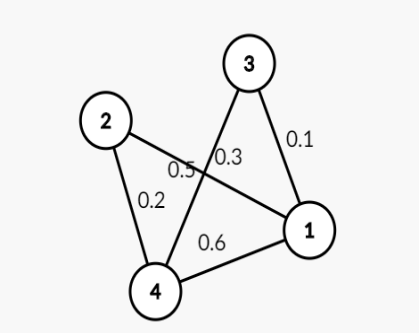
Add the plot to your pdf and describe what was the most successful strategy . explain why do you think that it was the most successful one.



Appendix 1 : create graph from the data

source	target	weight C1
1	2	0.3
1	3	0.1
2	4	0.2
3	4	0.5
1	4	0.6

The created graph WG



Appendix 2 : Centrality measures for the above graph (without weights)
{‘ Closeness’:{1:1,2:0.75,3:0.75,4:1}, ’Betweeness’:{1:0.333, 2:0.0, 3:0.0, 4:0.333}}

submission -

for the non-competitive part :

you should create a zip file names – hw1_studentID1_studentID2.zip that carries :

hw1_part1.py (your solution)

hw1_ID1_ID2.pdf

hw1_competition.csv (the removed edges)

Make sure there are no sub folders in the zip file. Make sure your file name does not have a double suffix, e.g. hw1_studentID1_studentID2.zip.zip (funny but happens a lot).

Only one student in a group should submit

We may test your code with a different input. The format will be the same (same column and data types), but do not assume anything about the number of rows or their content.