The Technion – Faculty of Biology
Computational Approaches for Life Scientists - 134141
Spring 2016

# Home assignment 3 – Due May 11th, 23:55

Submission instructions:

- Submission is **in pairs**.
  Honor code:
  Each student is expected to be fully involved in solving all the questions. Furthermore, although you may discuss the solutions with other students, each pair must write the whole solution separately. In particular, it is not allowed to spread any piece of code / solution.

- The answers will be submitted in 2 files:

  1. A doc/docx file with the "dry" part: answers to all the questions and the required explanations.
  2. A py file with the "wet" part: you need to write the required code in the template file **hw1_template.py**. The code in this file must support your answers and conclusions, such that when run yields the results provided in the dry part. The grade for this part will be given for a correct and reasonable code, which avoids unnecessary complications and bad styling.

- Note: the files should be named hwX_NAME1_NAME2 where X is the HW number (e.g. hw2) and NAME1 and NAME2 are the names of the students. For example: hw2_gur_hevroni_amir_rubinstein.docx.

- The assignment should be submitted via moodle. The 2 files should be uploaded to moodle by the deadline. You may submit late though, up to a maximum of 5 days for the whole semester. For that matter, submission after 23:55 is a day late.

- Note: while each question normally has a single correct answer, the code you'll write to answer them may be written in various ways. There is no single correct code.
  However, to get full score you must make an effort to write your code in a reasonable manner, in terms such as low complexity, meaningful names for variables, etc.

  You are requested to comment (#) your code, and you may even leave old code that you have written and no longer use (commented out!). This way we will be able to track your thinking process, at least partially.

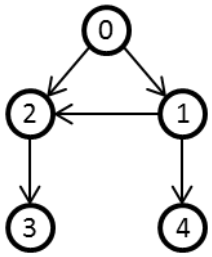
- Bonus: up to 5 points may be given as a bonus to your grade, for interesting, **non-trivial** comments, mostly biological ones, that you find relevant. For example:
  - A relevant application for the question's topic that is beyond a trivial one
  - New information that sheds interesting light on the question, or puts it in an interesting biological context

  If you added such comments and would like them to be considered for bonus, add a clear title "INSIGHTS" before the comments. Also, make sure you state your references (a website, a course you took, a book, etc.).

# ATG

## Question 1 – Betweeness of nodes in a network

The **betweeness** of a node in a graph was defined in class as follows: the number of shortest paths that pass through the node. For example, look at the following directed graph:




Note that the graph is unweighted, and therefore we consider all the edges to be of weight 1. In other words: `G[i][j]==1` where there is an edge between node `i` and node `j`, otherwise `G[i][j]==inf`. The betweeness of node 2 is 2, because there are 2 shortest paths that go though it: 0→2→3 and 1→2→3. The betweeness of node 1 is 1, and he betweeness of all other nodes is 0.

In this question you will be asked to compute the **maximal betweeness** in the network representing the **mammalian hippocampal CA1 neuron signaling network**. The network is taken from the Science magazine paper by Avi Ma'ayan *et al*. (2005): "*Formation of Regulatory Patterns During Signal Propagation in a Mammalian Cellular Network*". The network is provided in that paper in a format that was mentioned in the lecture slides. The function `CA1file2mat`, provided to you in that lecture, reads the network file into a matrix format:

```
G,nodes = CA1file2mat("./CA1.txt")
```

Note that the resulting graph is also unweighted.

a. Complete the function `find_betweeness(G,v)`. The function gets a directed graph `G` as a matrix, and an index `v` of a node in it (i.e. $0 \leq v \leq len(G) - 1$). It returns the betweeness of node `v`. Use Dijkstra's algorithm to find shortest paths between nodes, and the function `shortest_path` that we saw in class to recreate the shortest path from a given source to a given target.

   For example, for the "toy" graph above:

```
>>> find_betweeness(G, 0)
0
>>> find_betweeness(G, 1)
1
>>> find_betweeness(G, 2)
2
>>> find_betweeness(G, 3)
0
>>> find_betweeness(G, 4)
0
```

b. Complete the function `max_betweeness(G)`, that iterates over the nodes in G and calls `find_betweeness` for each one, in order to find the node with maximal betweeness.

c. What is the node with the maximal betweeness in the CA1 neuron signaling network? What is its betweeness? Note: running time for this network is expected to be relatively long, in the scale of 1-3 hours on a reasonably fast computer (!), even if you try your best to be efficient.

d. To understand if this result is exceptional, let's compute the maximal betweeness in a **random graph** of the same size. There are many models for random graphs, two of them briefly mentioned in class. Perhaps the simplest model is the Erdős–Rényi (ER) model: a graph where for every two nodes `i` and `j` there is an edge from `i` to `j` with probability $p$ ($0 \leq p \leq 1$).

Complete the function ER`graph(n,p)` in the template file. This function receives the number of nodes $n$ and the probability for an edge $p$. It returns a random graph with $n$ nodes, according to the ER model. The graph will be represented, as usual, as a matrix. An edge will be represented by the value 1, a missing edge by inf = float('Inf').

Examples:

```
>>> G = rGraph(3,1) #p=1, all edges exist
>>> G
[[1, 1, 1], [1, 1, 1], [1, 1, 1]]
>>> G = rGraph(3,0) #p=0, no edges at all
>>> G
[[inf, inf, inf], [inf, inf, inf], [inf, inf, inf]]
>>> G = rGraph(3,0.5)
>>> G
 [[inf, 1, 1], [1, 1, inf], [1, inf, 1]]
>>> G = rGraph(3,0.5)
>>> G
[[1, inf, 1], [inf, inf, inf], [1, inf, inf]]
```

```
import random

def ERgraph(n,p):
    """ generate a random directed graph with n nodes
        and prob. for edge p """

    g = [[inf]*n for i in range(n)]
    for i in range(n):
        for j in range(n):

            _____________________
            if ____________________:
                g[i][j] = 1
    return g
```

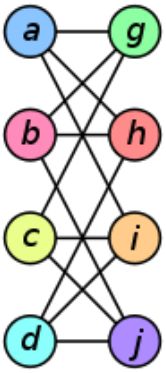
e. Generate a random ER graph with the same number of nodes as the CA1 neuron network. Set the parameter *p* such that the number of edges is roughly the same number of edges as in CA1 (find out how many edges are in CA1, and do the math). What is the node with the maximal betweeness in your random graph?

f. Can you provide any plausible explanation for the results? If you are curious, you may wish to compare additional properties related to betweeness, such as the average, median, or anything you find relevant.

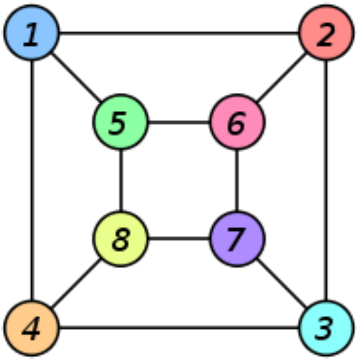
## Question 2 – Graph isomorphism

The goal of this question is to practice some logical thinking in the context of graph theory. No Python code here, only some theoretical brain exercise!

A basic notion in graph theory is **graph isomorphism**. Non-formally, two graphs are said to be isomorphic, if they actually represent the same structure of interactions (but perhaps with different names for nodes, and a different way of drawing). For example, these two graphs are isomorphic (image from Wikipedia):




Graph G




Graph H

To see this, change the names of the nodes of G in the following way:

$a \rightarrow 1$ , $b \rightarrow 6$ , $c \rightarrow 8$ , $d \rightarrow 3$ , $g \rightarrow 5$ , $h \rightarrow 2$ , $i \rightarrow 4$ , $j \rightarrow 7$

Now, the two graphs represent the exact same interactions between the nodes. For example, nodes *a* and *h* are neighbors in G, and indeed 1 and 2 are neighbors in H; nodes *a* and *b* are NOT neighbors in H, and indeed 1 and 6 are NOT neighbors in G. This relation is true for *every* two nodes. Generally, to prove that two graphs are isomorphic, one should show such a mapping (called isomorphism), and show that for every two nodes from G, they are neighbors if and only if the corresponding nodes in H are neighbors.

a. Are the following two graphs isomorphic? If you think they are, it is enough to specify the mapping. Otherwise, explain why not.

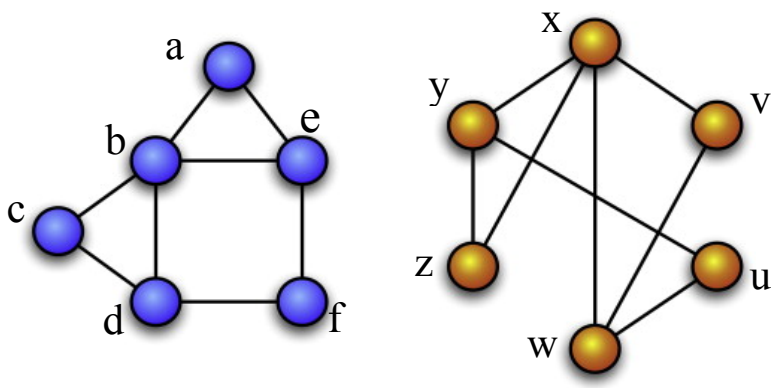



image from: http://dx.doi.org/10.1016/j.pbiomolbio.2012.05.007

b. Amir claims: for two graphs H and G to have the same number of nodes and the same number of edges is a *sufficient condition* for them to be isomorphic. Gur disagrees, and claims this is a *necessary condition*, but not a sufficient one. Which one of us is right? Or maybe both of us are wrong?

c. True or False: if two graphs have the same *degree distribution* (that is, the same number of nodes with degree 0, the same number of nodes with degree 1, and so on for every possible degree), they must be isomorphic. Explain why this is true or show a counter example.

## Question 3 – Boolean model for the simulation of regulatory networks

A. Extend the Boolean model for regulatory networks into a **discrete** model, in which nodes' states may have values between 0 and $U$=9. Note that the changes should occur only in the function `update_node`.

B. Look at the simulation of the yeast cell-cycle network in the slides. The initial vector [1,0,0,0,1,0,0,0,1,0,0] reaches a steady state with the final vector [0,0,0,0,1,0,0,0,1,0,0]. Simulate the discrete model on the initial vector [9,0,0,0,9,0,0,0,9,0,0] (1's replaced by 9's). What is the final vector of this simulation?

C. Do you observe any differences between the simulation of this initial vector with the Boolean and the discrete models, which look significant to you? No single correct answer here!

**TGA**