1. Centrality measures for undirected networks (6 pt)

In this exercise, we will get familiar with some common centrality measures by applying them to undirected networks (although these measures can all be generalized to directed networks).

Submit your solution as a pdf file in MyCourses.

Below, we list and define the measures used in this exercise:

1. degree $k(i)$:
   Number of neighbors of node i.

2. betweenness centrality $bc(i)$:
   Number of shortest paths between other nodes of the network that pass through node i. If there are several shortest paths between a given pair of nodes, then the contribution of that node pair to the betweenness of i is given by the fraction of those paths that contain i. Betweenness scores are normalized by $(N-1)(N-2)$, i.e. the number of node-pairs in the network, excluding pairs that contain i (because paths starting or ending in node i do not contribute to the betweenness of i), which is the maximum possible score. If σ_{st} is the number of shortest paths from s to t and σ_{sit} the number of such paths that contain i, then
   $$bc(i) = \frac{1}{(N-1)(N-2)} \sum_{s \neq i} \sum_{t \neq i} \frac{\sigma_{sit}}{\sigma_{st}}.$$

3. closeness centrality $C(i)$:
   The inverse of the average shortest path distance to all other nodes except i:
   $$C(i) = \frac{N-1}{\sum_{v \neq i} d(i,v)}.$$

4. k-shell $k_s(i)$:
   Node i belongs to the k-shell if it belongs to the k-core of the network but does not belong to the $k+1$-core. The k-core is the maximal subnetwork (i.e. the largest possible subset of nodes and links between them) where all nodes have at least degree k. Explicitly, the 1-core is formed by removing nodes of degree 0 (isolated nodes) from the network; the 2-core is formed by removing nodes of degree 1 and iteratively removing the nodes that get degree 1 or 0 because of the removal; the 3-core is formed by removing nodes of degree less than 3 and those nodes that get degree less than 3 because of the removal, and so on. The 1-shell is then the set of nodes that was removed from the 1-core to obtain the 2-core.

5. eigenvector centrality $e(i)$:
   Eigenvector centrality is a generalization of degree that takes into account the degrees of the node's neighbors, and recursively the degrees of the neighbors of neighbors, and so on. It is defined as the eigenvector of the adjacency matrix that corresponds to the largest eigenvalue.

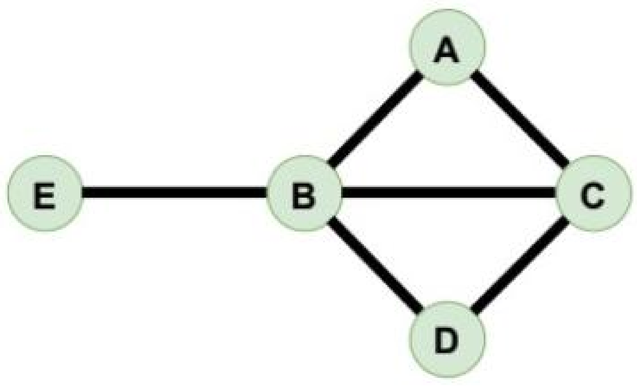
a) (2 pt) Your first task is to compute the requested centralities by hand (without using a computer) for the selected nodes in the network shown in Fig. 1.

i) Betweenness centrality of node **B** (include the intermediate calculation steps in your report).

ii) Closeness centrality of node **B** (include the intermediate calculation steps in your report).

iii) k-shell centrality of all nodes in the network shown in Fig. 1.

Note that you are not asked to report the degree and eigenvector centralities, because the first one is trivial and the latter one might be too difficult to calculate by pen-and-paper (you would need to compute the eigenvalues of a 5×5 matrix!)



i) Let's compute, for each node, σ_{st} and σ_{sBt}, where σ_{st} is the number of shortest paths from s to t and σ_{sBt} the number of such paths that contain B

NODE A

$\frac{\sigma_{ABC}}{\sigma_{AC}} = \frac{0}{1} = 0$; $\frac{\sigma_{ABD}}{\sigma_{AD}} = \frac{1}{2}$; $\frac{\sigma_{ABE}}{\sigma_{AE}} = \frac{1}{1} = 1$

NODE C

$\frac{\sigma_{CBA}}{\sigma_{CA}} = \frac{0}{1} = 0$; $\frac{\sigma_{CBD}}{\sigma_{CD}} = \frac{0}{1} = 0$; $\frac{\sigma_{CBE}}{\sigma_{CE}} = \frac{1}{1} = 1$

NODE D

$\frac{\sigma_{DBA}}{\sigma_{DA}} = \frac{1}{2}$; $\frac{\sigma_{DBC}}{\sigma_{DC}} = \frac{0}{1} = 0$; $\frac{\sigma_{DBE}}{\sigma_{DE}} = \frac{1}{1} = 1$

NODE E

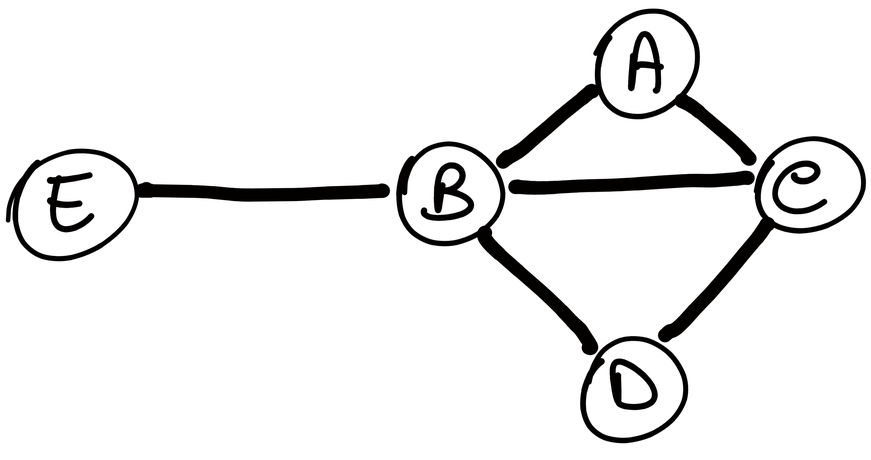
$\frac{\sigma_{EBA}}{\sigma_{EA}} = \frac{1}{1} = 1$; $\frac{\sigma_{EBC}}{\sigma_{EC}} = \frac{1}{1} = 1$; $\frac{\sigma_{EBD}}{\sigma_{ED}} = \frac{1}{1} = 1$

Finally $bc(B) = \frac{1}{(N-1)(N-2)} \sum_{s \neq B} \sum_{t \neq B} \frac{\sigma_{sBt}}{\sigma_{st}} = \frac{7}{12} \simeq 0.583.$

ii) Let's compute $d(B,A) = 1$, $d(B,C) = 1$, $d(B,D) = 1$, $d(B,E) = 1$, so $C(B) = \frac{N-1}{\sum_{i \neq B} d(B,i)} = \frac{4}{4} = 1$

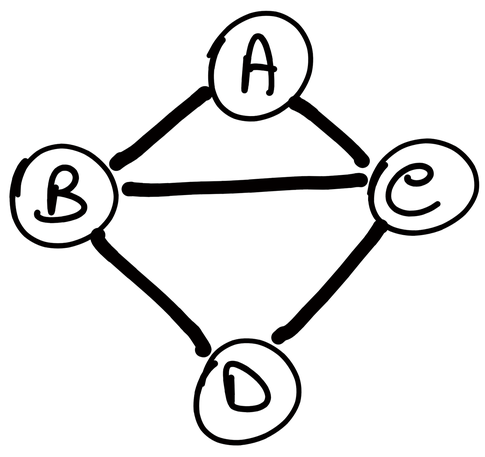
iii)

1-core



$\{A, B, C, D, E\}$

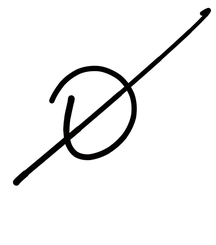
2-core



$\{A, B, C, D\}$

* even if we remove the node E, with degree 1 this don't affect the degrees of the other nodes, which remain greater or equal than 2.

3-core



$\{\emptyset\}$

* In this case, removing the node (D), with degree equals to 2, effects also the degree of the other nodes, and no nodes with degree equal or higher than 3 remain.

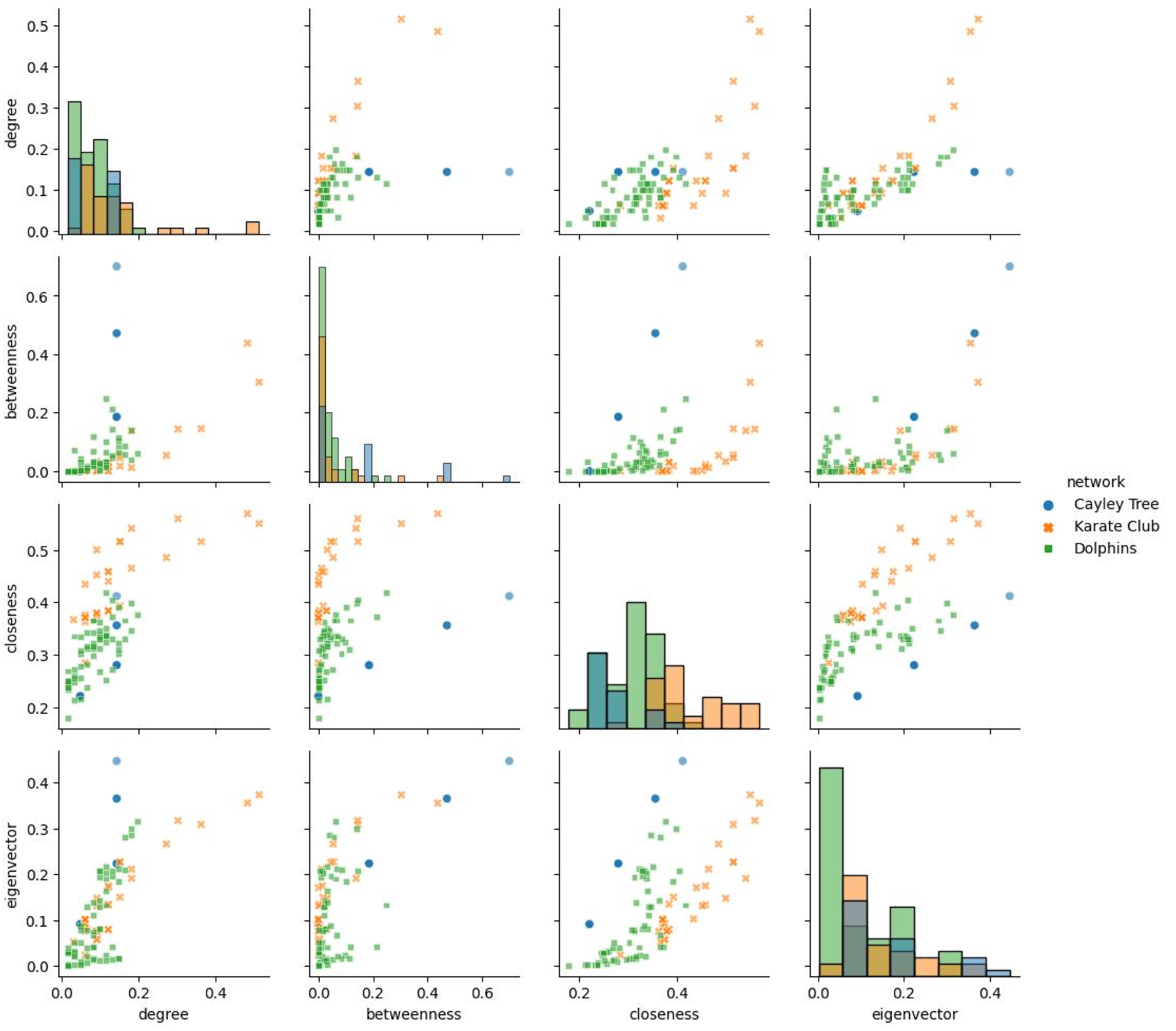
the k-shell for each node are:

k-shell(A) = 2
k-shell(B) = 2
k-shell(C) = 2
k-shell(D) = 2
k-shell(E) = 1

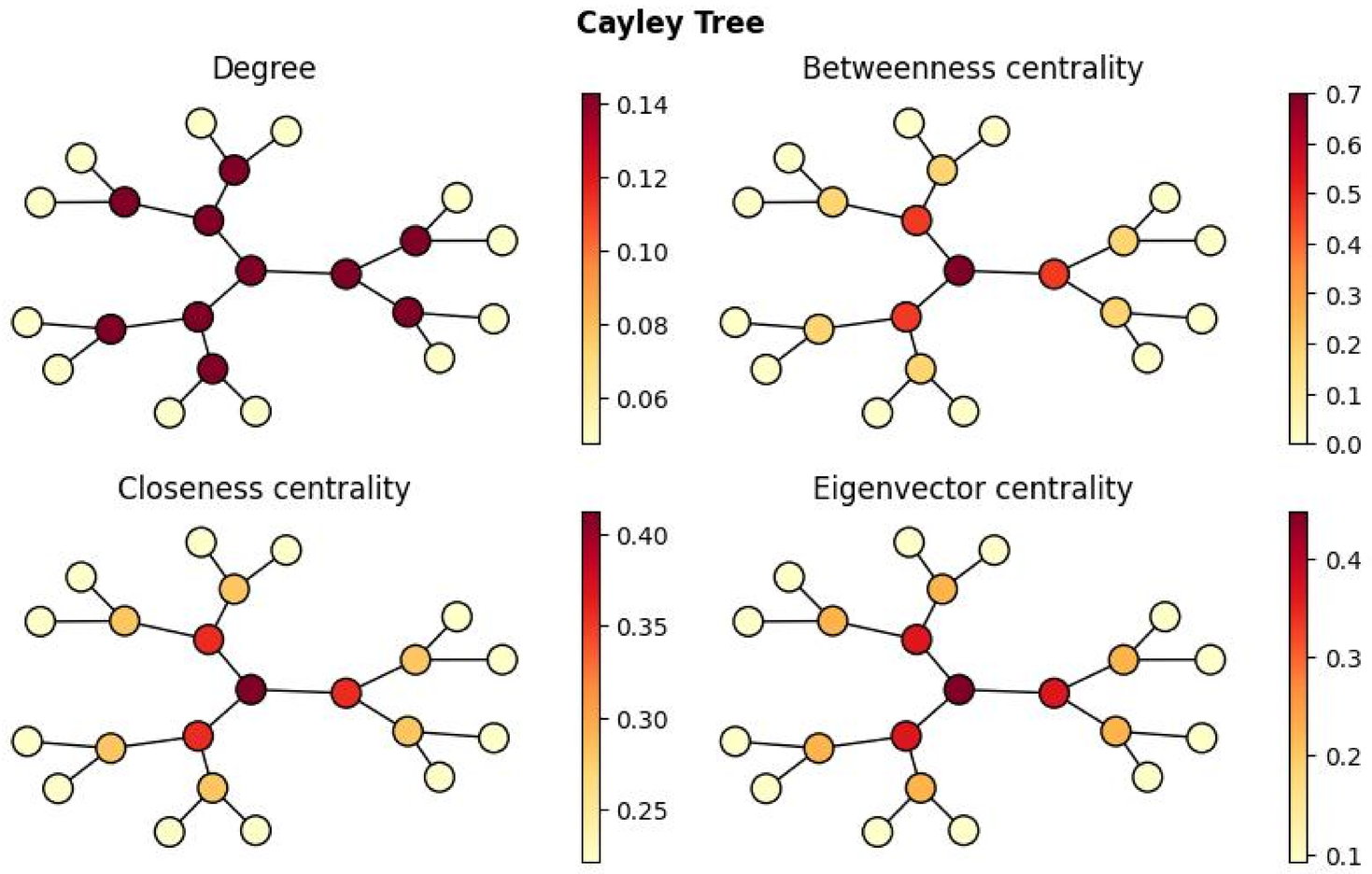
b) (2 pt) Use NetworkX to compute the degree, betweenness, closeness, and eigenvector centralities for the networks for the three networks (see also Fig. 2):

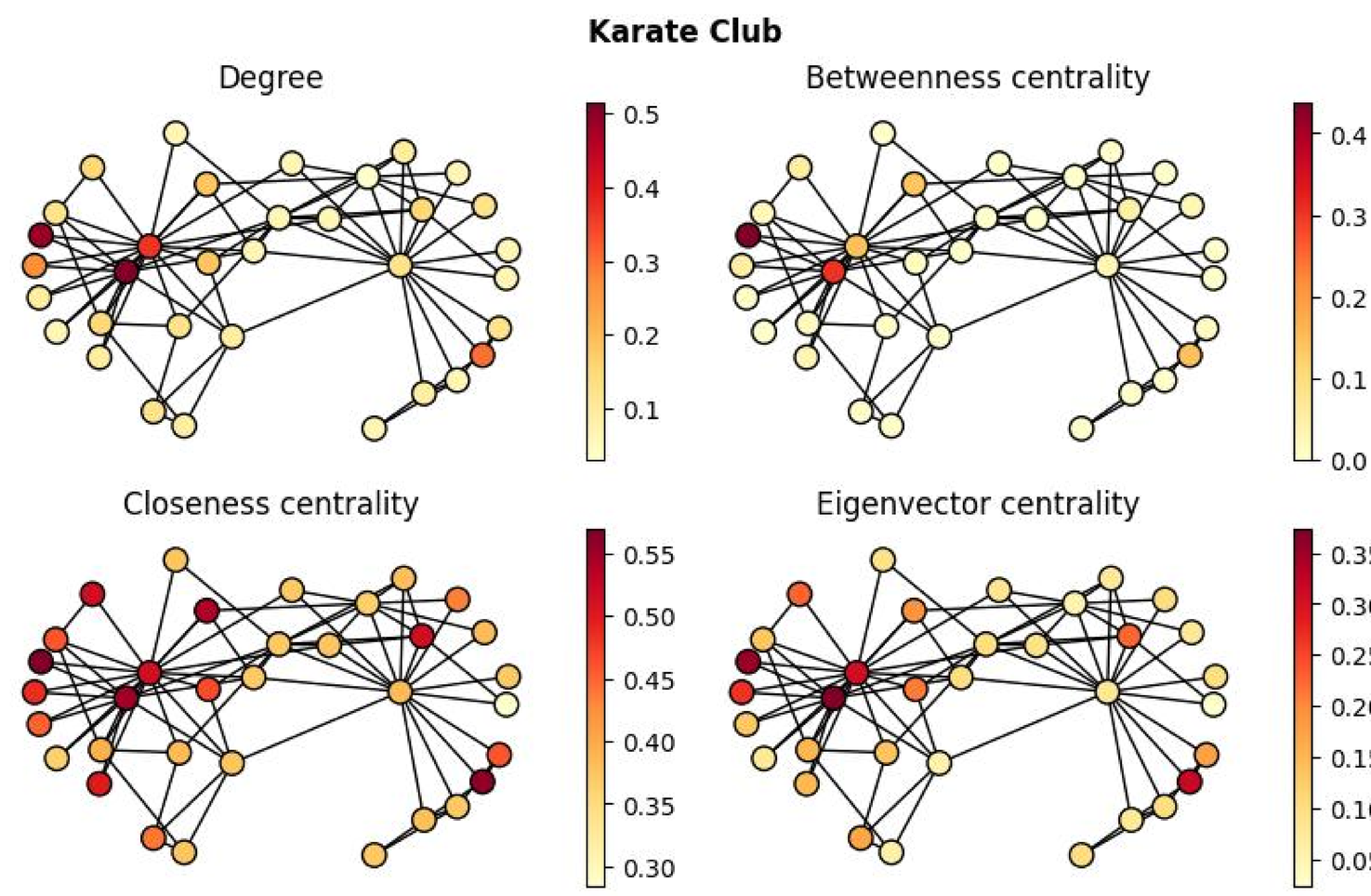
- Cayley tree with $k = 3$ and $l = 3$ (`cayley_tree_edge_list.edg`). A Cayley tree is a tree in which every node except the leaves has k neighbors. The parameter l is the number of layers in the tree, i.e., the path length between the root node and the leaves.
- Zachary karate club network (`karate_club_edge_list.edg`). The Zachary karate club network is a well-known social network studied by sociologist Zachary (1977). The nodes represent members of a karate club and the links represent interaction between members.
- Dolphin social network (`dolphins_edge_list.edg`). A social network between bottlenose dolphins living off New Zealand compiled by Lusseau et al. (2003). The nodes represent dolphins and the links represent frequent associations between dolphins.

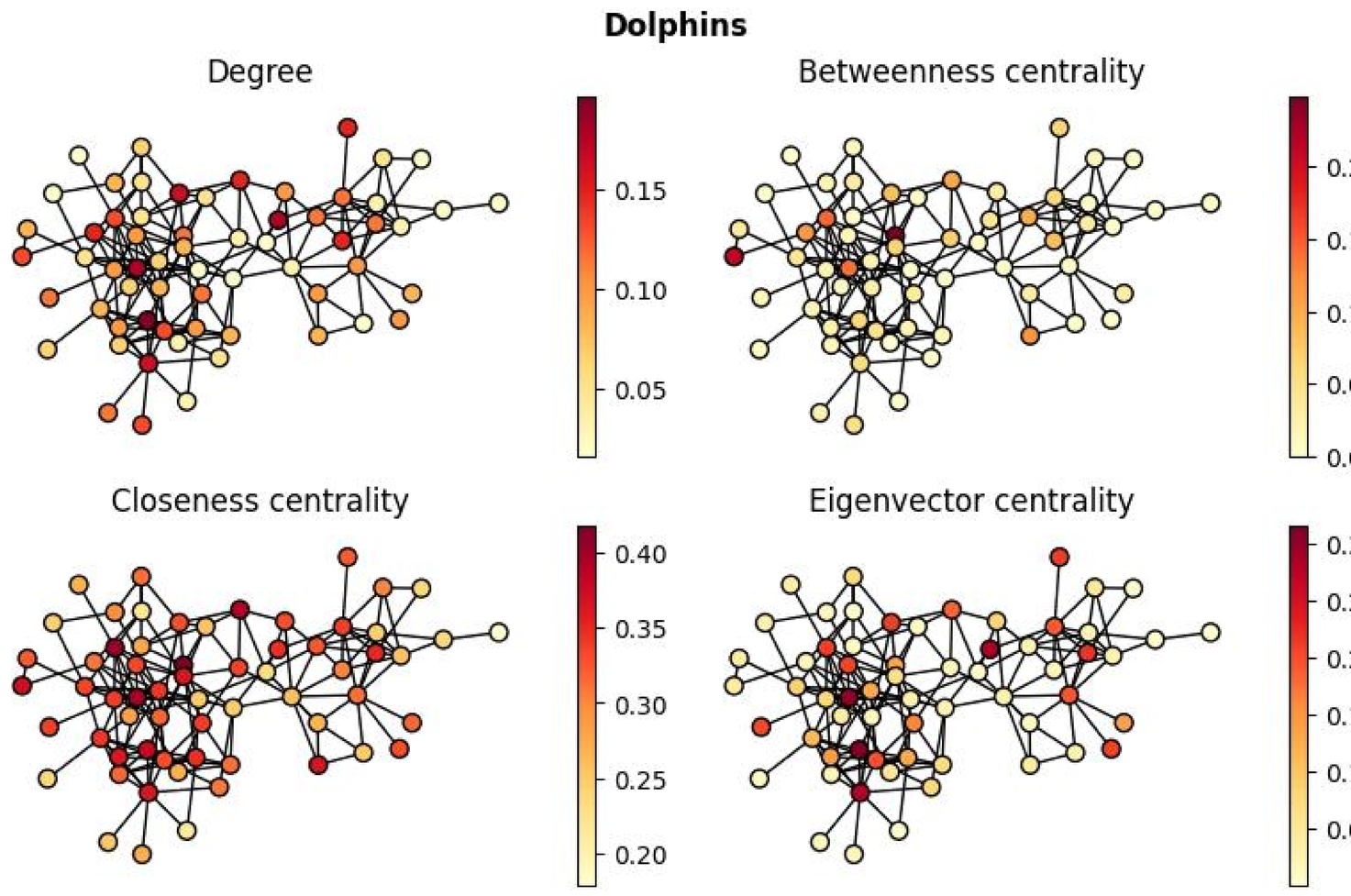
Visualize the relationships between the four centralities as scatter plots in a pairwise manner. You can use the code template provided for this and the following tasks.



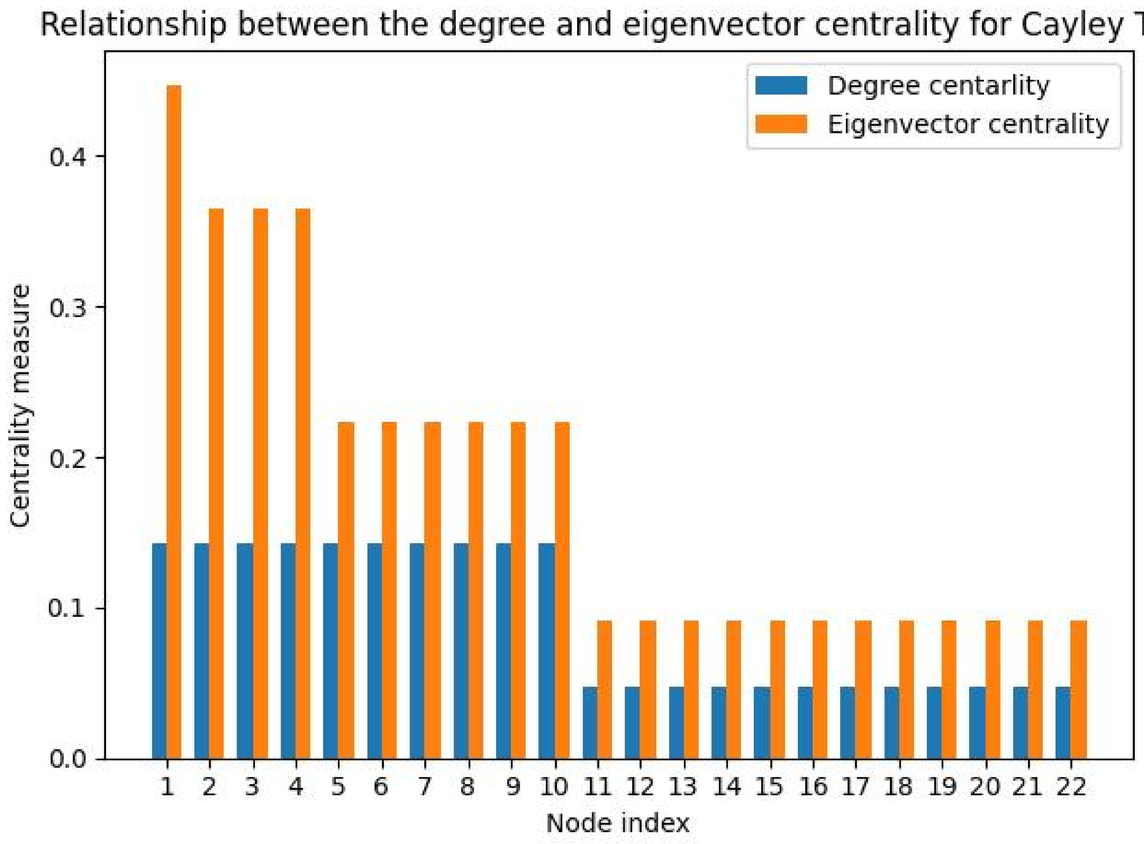
c) (1 pt) To highlight the differences between the centrality measures, visualize each network studied in part b) (the Cayley tree, karate club network, and dolphin social network), using each of the centrality measures to define the colors of the network nodes. To make the visualization easier, coordinates of the nodes are provided in `.pkl` files (`cayley_tree_coords.pkl`, `karate_club_coords.pkl`, `dolphines_coords.pkl`).

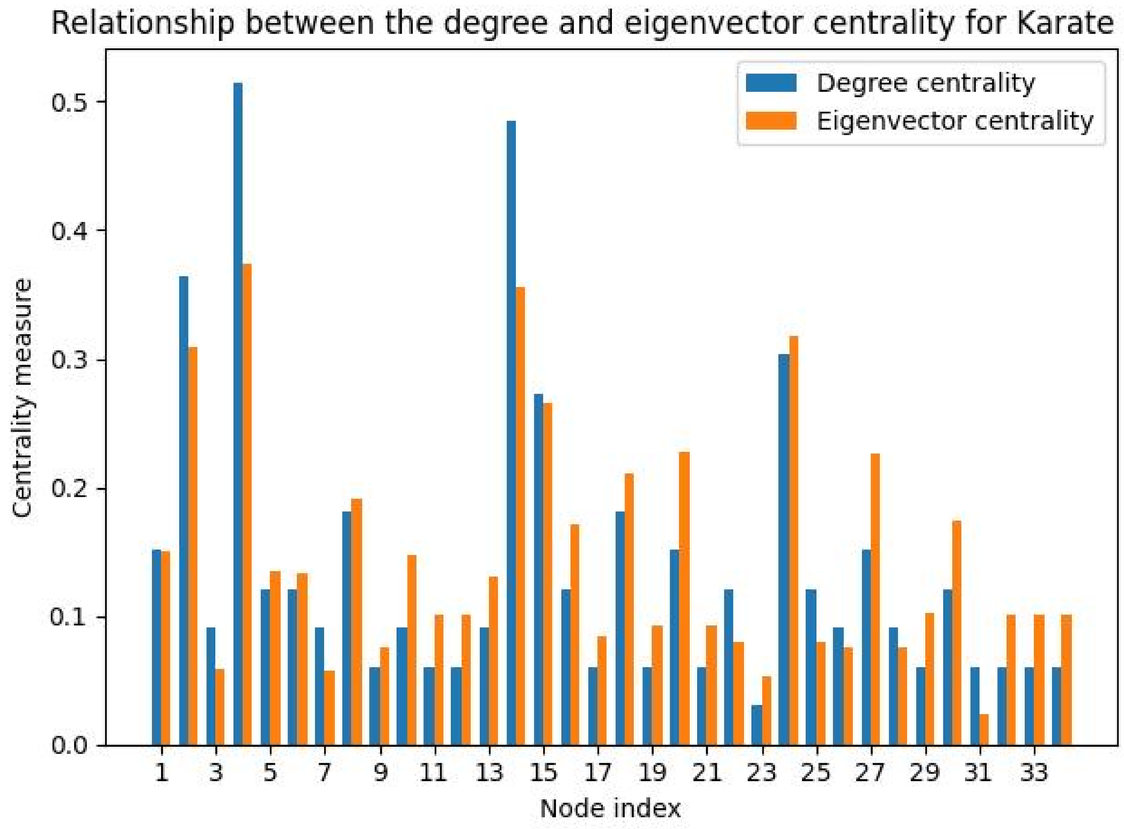


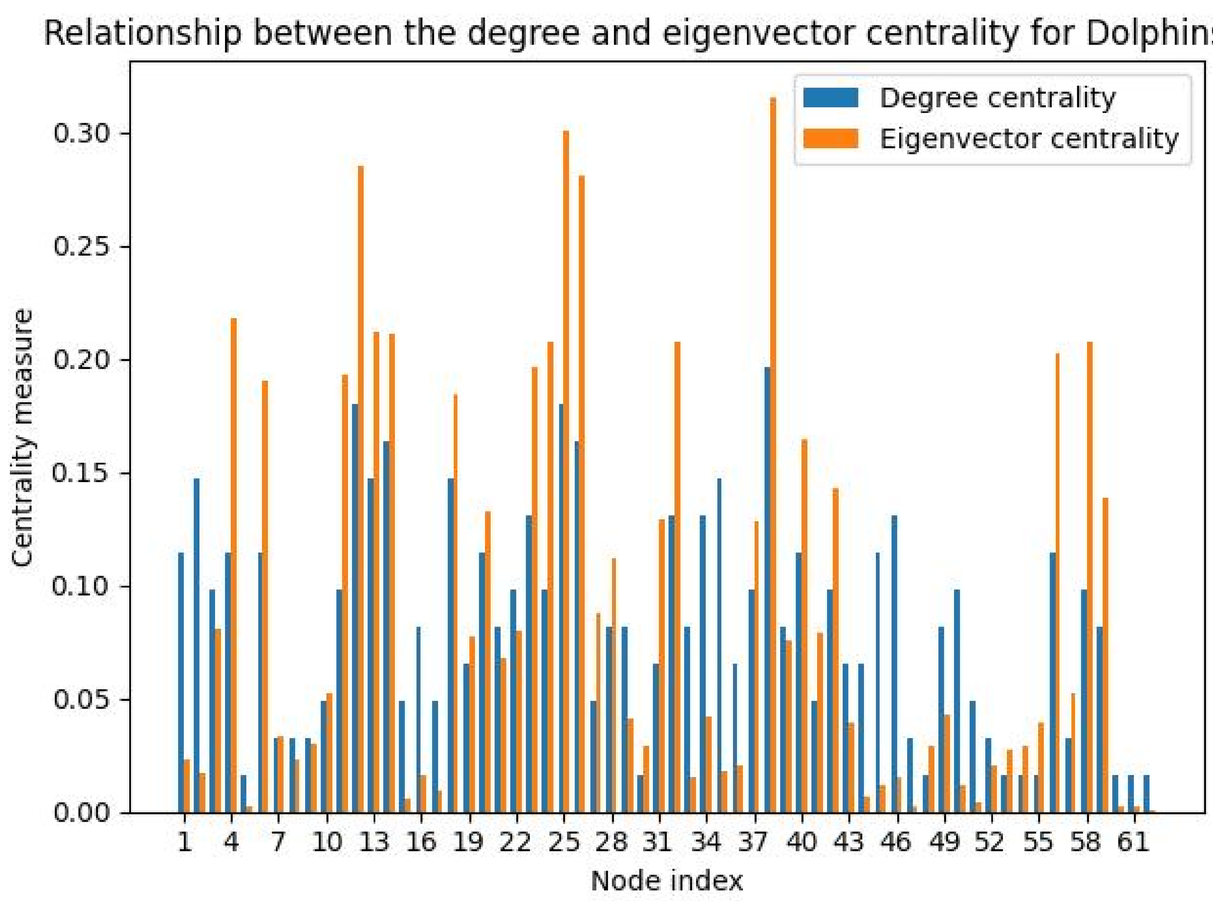




d) (1 pt) Explain, in 3-10 sentences, the relationship between the degree and eigenvector centrality in the three networks studied in parts b) and c). Describe the differences between the two centrality measures and explain why those differences occur.







Degree centrality and eigenvector centrality are two measures of node importance in networks. Degree centrality is a local measure that focuses on the number of immediate connections, while eigenvector centrality is a global measure that considers the importance of both a node's connections and its neighbors' connections. As can be seen from the graphs, nodes with a high degree centrality tend to have a similarly high eigenvector centrality value, but this is not always the case. The CayleyTree histogram illustrates how nodes with the same degree centrality can have different eigenvector centrality values. This happens because while degree centrality looks at the closest neighbors, eigenvector centrality allows each node to gain importance from its less immediate predecessors, which may not necessarily be the same as those of other nodes sharing the same degree centrality value.

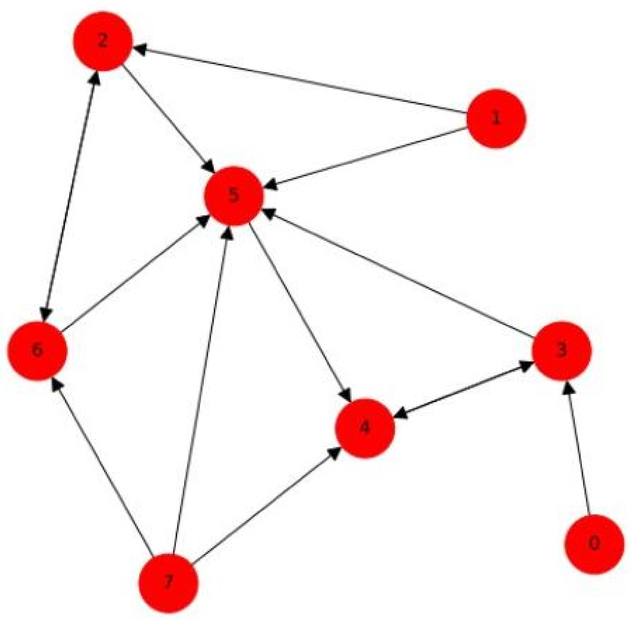
2. PageRank for directed networks (12 pts)

PageRank, a generalization of eigenvector centrality for directed networks, was first developed for search engines like Google to determine the centrality of web pages. If we consider a random walker that with probability d moves to one of the successors of the current node (i.e., the nodes *to* which the current node is linked) and with probability $1 - d$ teleports to a random node, the PageRank of each node is equal to the fraction of time that the random walker has spent at that node.

In this exercise, we will investigate the behavior of PageRank in a simple directed network (see Fig. 3). To get started, you can use the provided code template.

Submit your solutions by taking a quiz in MyCourses.

a) (1 pt) Load the network given in file `pagerank_network.edg` and, as a sanity check, visualize it with `nx.draw`.



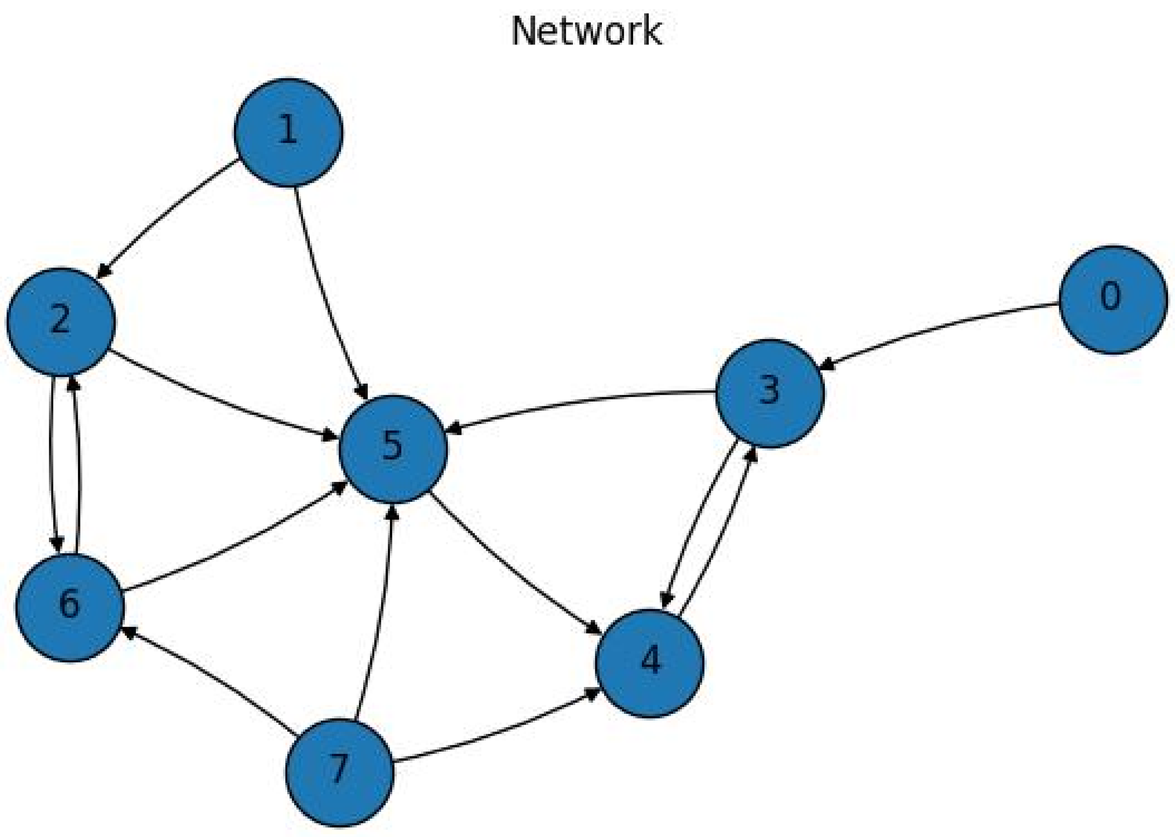
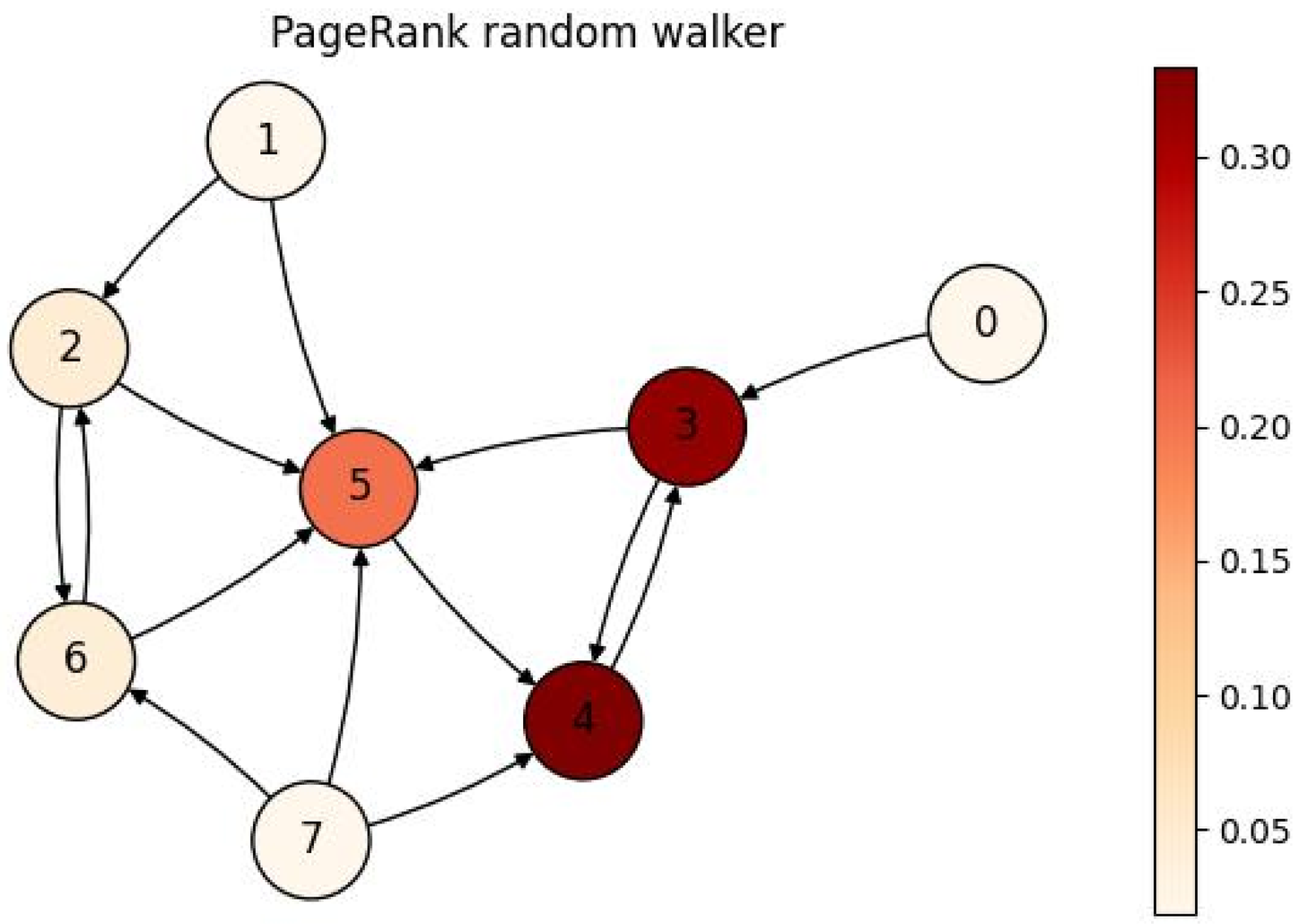


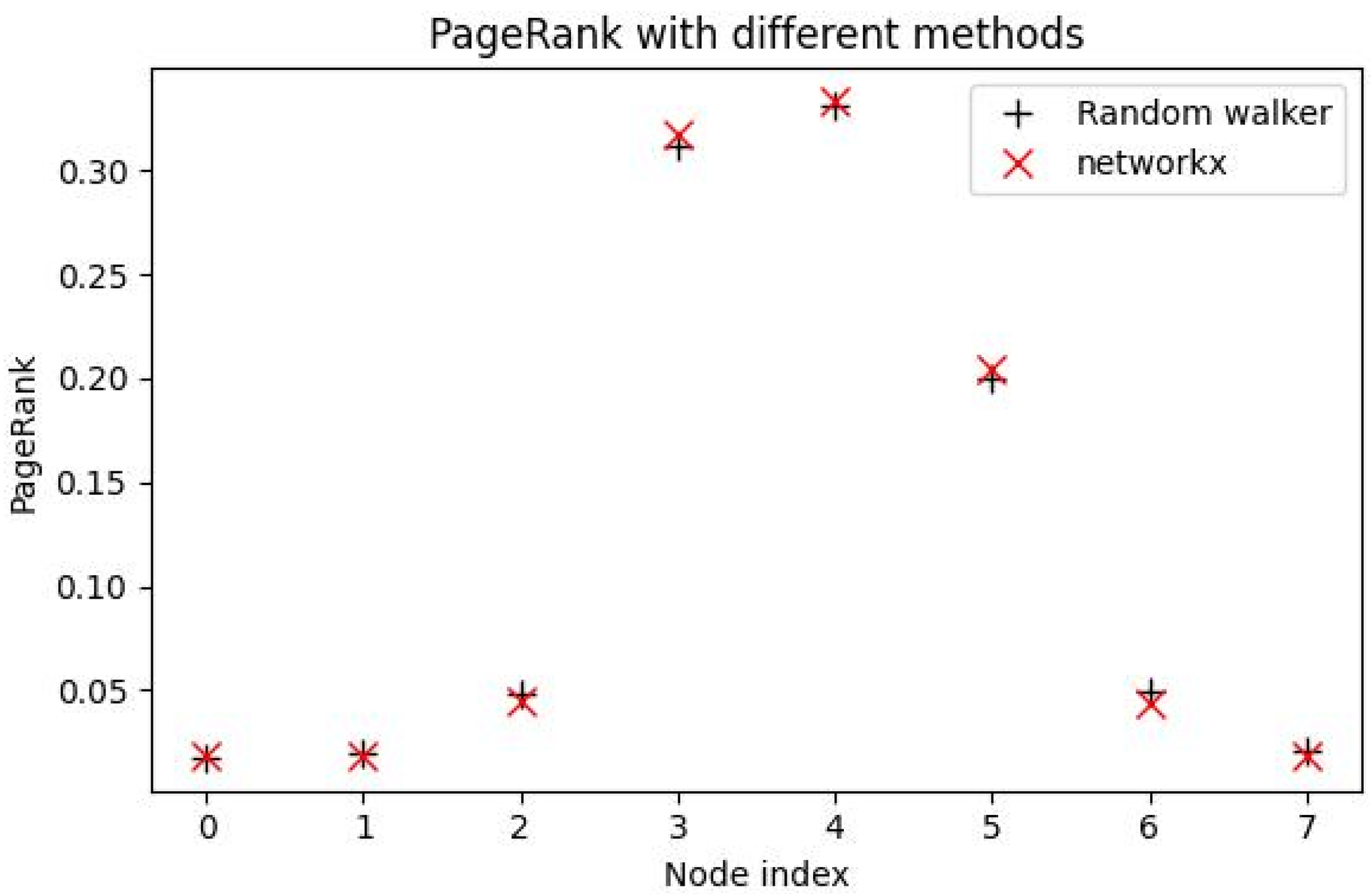
Figure 3: A simple directed network.

b) (4 pt) Write a function that computes the PageRank on a network by simulating a random walker. In more detail,

1. Initialize the PageRank of all nodes to 0.
2. Pick the current node (the starting point of the random walker) at random.
3. Increase the PageRank of the current node by 1.
4. Select the node to which the random walker will move next:
   * Draw a random number $p \in [0, 1]$.
   * If $p < d$, the next node is one of the successors of the current one. Choose it at random.
   * Else, the random walker will teleport. Pick the next node randomly from all nodes in the network.
5. Repeat steps 3-4 N_{steps} times.
6. Normalize the PageRank values by N_{steps}.

Use your function to compute PageRank in the example network. Visualize the result on the network by using the PageRank values as node color values. Compare your results with `nx.pagerank` by plotting both results as a function of node index.





It's possible to notice that the two models agree, and the results are very similar.

c) (4 pt) The above algorithm is a naive way of computing PageRank. The actual algorithm behind the success of Google, introduced by its founders, Larry Page and Sergey Brin, is based on power iteration [1].

Power iteration finds the leading eigenvector for the "Google matrix" (or other matrices) very fast under certain conditions. An intuitive way of thinking about the power iteration algorithm is to think that at time $t-1$ you have a vector $x(t-1)$ where each element gives the probability of finding the walker. You use the rules of the random walk/teleportation process to find out what are the probabilities of finding the random walker at each node at time t. That is, you increase the time t and calculate $x(t)$ based on $x(t-1)$ until the vector x doesn't change anymore.

Write a function that computes the PageRank by using power iteration. In more detail,

1. Initialize the PageRank of all nodes to $\frac{1}{n}$, where n is the number of nodes in the network. That is, at time step $t = 0$ your PageRank vector contains the same value for all nodes, so it is equally likely to find the walker at any node. (Any other initialization strategy is possible as long as the sum of all entries is 1, and the closer the initial vector is to the final vector, the faster you will find the stationary PageRank values)
2. Increase the time step t by 1 and create a new empty PageRank vector $x(t)$.
3. Fill in each element of the new vector PageRank vector $x(t)$ using the old PageRank vector $x(t-1)$ and the formula:

$$x_i(t) = (1-d)\frac{1}{n} + d\sum_{j\in\nu_i}\frac{x_j(t-1)}{k_j^{\text{out}}},$$

where ν_i is the set of nodes that have a directed link ending at i. For each node $j \in \nu_i$, k_j^{out} is j's out-degree. In summary, for each node i you need to calculate their entry in the new PageRank vector $x(t)$ as a sum of two parts:

   * the probability that the walker will randomly teleport into the node, given by

$$(1-d)\frac{1}{n},$$

   and

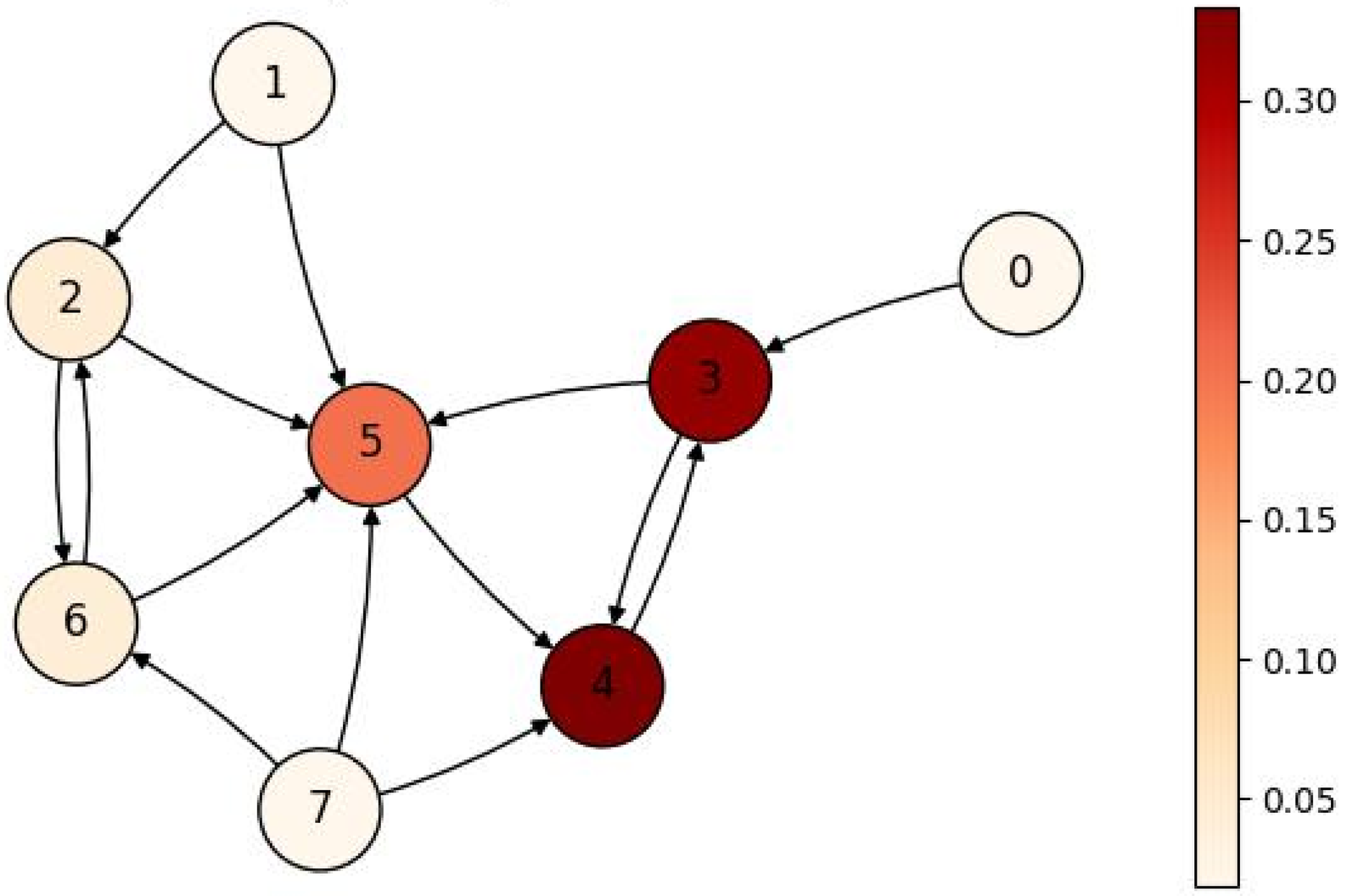
   * the probability that the walker will move from a neighbor j to node i. Iterate over each in-neighbor j of the node i (i.e., there is a link from j to i) and add the neighbor's contribution

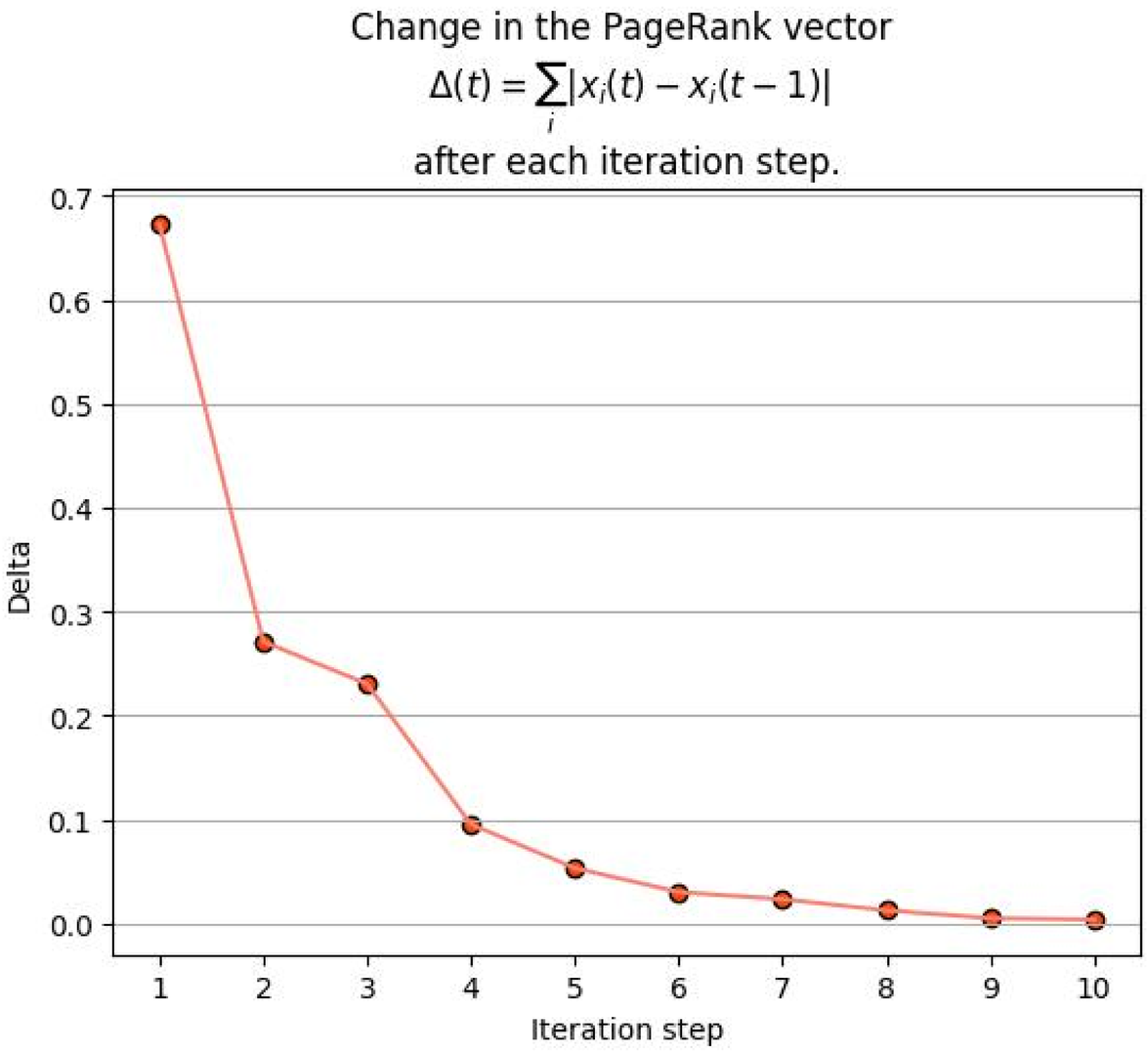
$$d\frac{x_j(t-1)}{k_j^{\text{out}}}$$

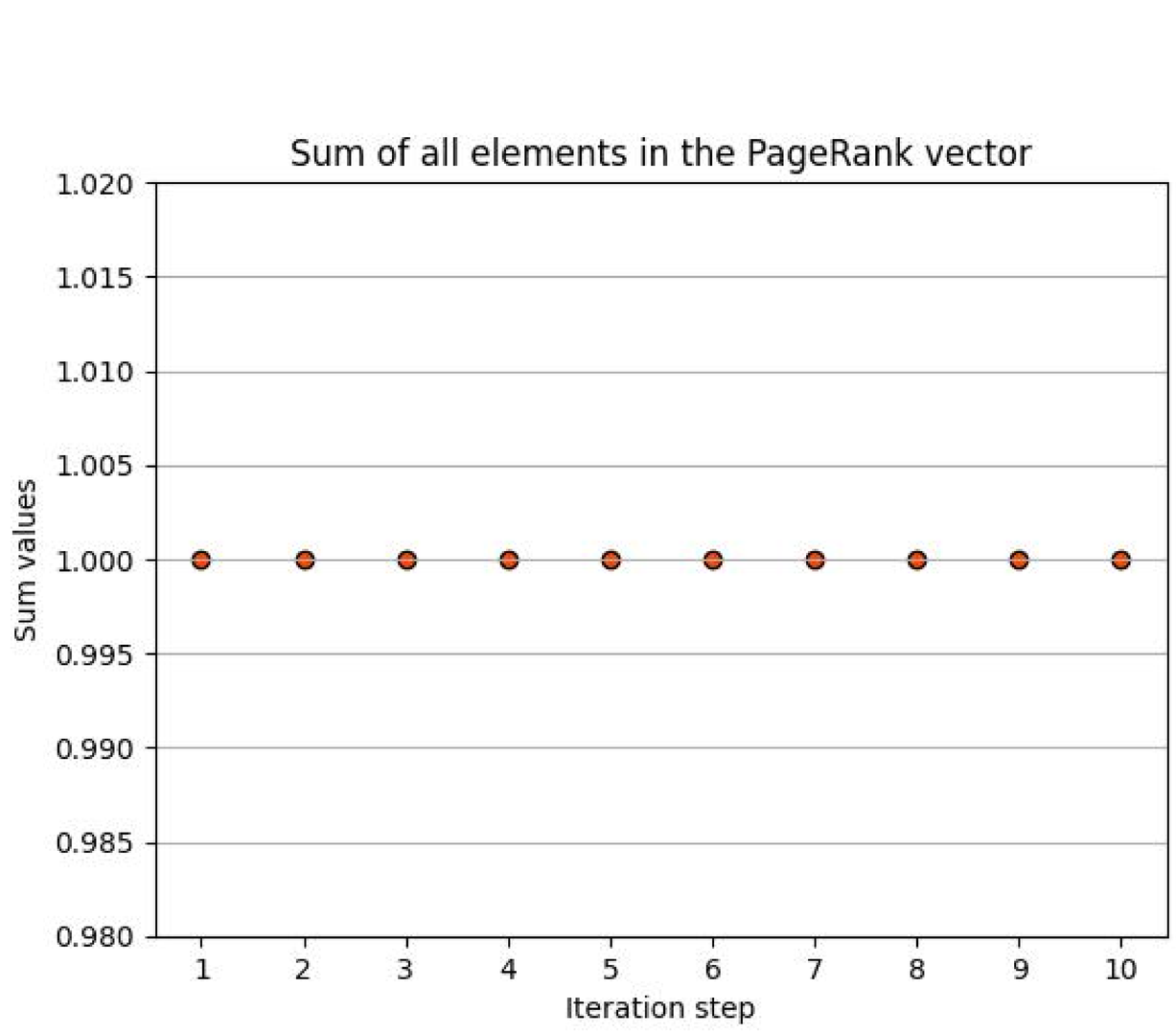
   to the entry of node i in the new PageRank vector $x(t)$.

4. Repeat steps 2 and 3 $N_{\text{iterations}}$ times.

Use your function to compute PageRank in the sample network and visualize the result on top of the network as in b).







We can see a good progess of the algorithm because the change in the page rank vector converges to zero, so it is a decreasing function of t and $N_{it} = 10$ are enough as expected.

This plot confirms that the sum of all elements in the PageRank vector is always equal to one, as it should be.

d) (3 pt) The aim of this task is to understand how the structure of a network relates to PageRank. Answer the following questions.

i) Choose all the correct statements about the relationship between the PageRank and the degree of a node:

A The PageRank of a node is always higher if its in-degree is higher.

B The PageRank of a node is always higher if its out-degree is lower.

C ~~X~~ The PageRank of a node is affected by its in-degree, but not by its out-degree.

D ~~X~~ The PageRank of a node is dependent on the PageRanks of its predecessors and their out-degrees.

E The PageRank of a node is dependent on the PageRanks of its successors, but not on their in-degrees.

F The PageRank of a node with zero in-degree is zero.

ii) If a node belongs to a strongly connected component, how does that affect the PageRank of the node? Explain in 2-5 sentences.

iii) Explain why the PageRank of nodes 3 and 4 in the example network is higher than that of node 5 in 2-5 sentences.

iv) How could the information about the network's structure be used in improving the power iteration algorithm given in part c)? Explain in 1-3 sentences.

ii)

If a node belongs to an SCC, this means that it is reachable from many other nodes in the network, which can boost its PageRank. However, the impact of the SCC on the PageRank of a node also depends on the PageRank of the other nodes in the component. In fact, if the other nodes have a high PageRank, then the node's PageRank will be boosted more.
In other words, belonging to an SCC is generally a good thing for a node's PageRank, but the specific impact depends on the other nodes in the SCC. If the other nodes are important, then the node's PageRank will be boosted more.

iii)

Despite node 5 having a higher degree of centrality than nodes 3 and 4, the latter two have higher PageRank values. This is because a node's PageRank also depends on the PageRank of other nodes in the network. In this case, the neighbors of node 5 have low PageRank values, while nodes 3 and 4, both having high PageRank values, are mutually connected, mutually boosting their scores. In fact, it can be observed that the PageRank of node 5 is increased precisely due to its connection with node 3, whose PageRank value is much higher than the PageRank values of the other nodes (1, 2, 6, 7) that point to node 5.

iv)

The information about the network's structure can be used to improve the power iteration algorithm of PageRank by using a better initial guess for the dominant eigenvector. For example, we can use the degree of each node as the initial guess, since the degree of a node is a good indicator of its importance in the network.

Challenge exercises (4 pts)

e) (2 pt) The Google search engine indexes billions of websites and the algorithm for calculating the PageRank needs to be extremely fast. In the original paper about PageRank [1], by Google founders Larry Page and Sergey Brin, they claim that their "iterative algorithm" is able to calculate the PageRank for 26 million webpages in a few hours using a normal desktop computer (in 1998).

Come up with a rough estimate of how long it would take for your power iteration algorithm (part c) and naive random walker algorithm (part b) to do the same. You can assume that the average degree of the 26 million node network is small and that the power iteration converges in the same number of time steps as it does for your smaller networks. For the random walk, you can assume you need to run enough time steps such that the walker visits each node 1000 times, on average. You can also omit any considerations of fitting the large network in memory, or the time it takes to read it from disk, etc. Under these assumptions, you can simply calculate the time it takes to run the algorithm in a reasonably-sized network, and then multiply the result by the number of times the 26 million node network is larger than your reasonably-sized network.

Report all calculations and parameters you use (such as size of the network, number of time steps, etc.). Simply reporting the result without telling how you obtained it will not get you any points.

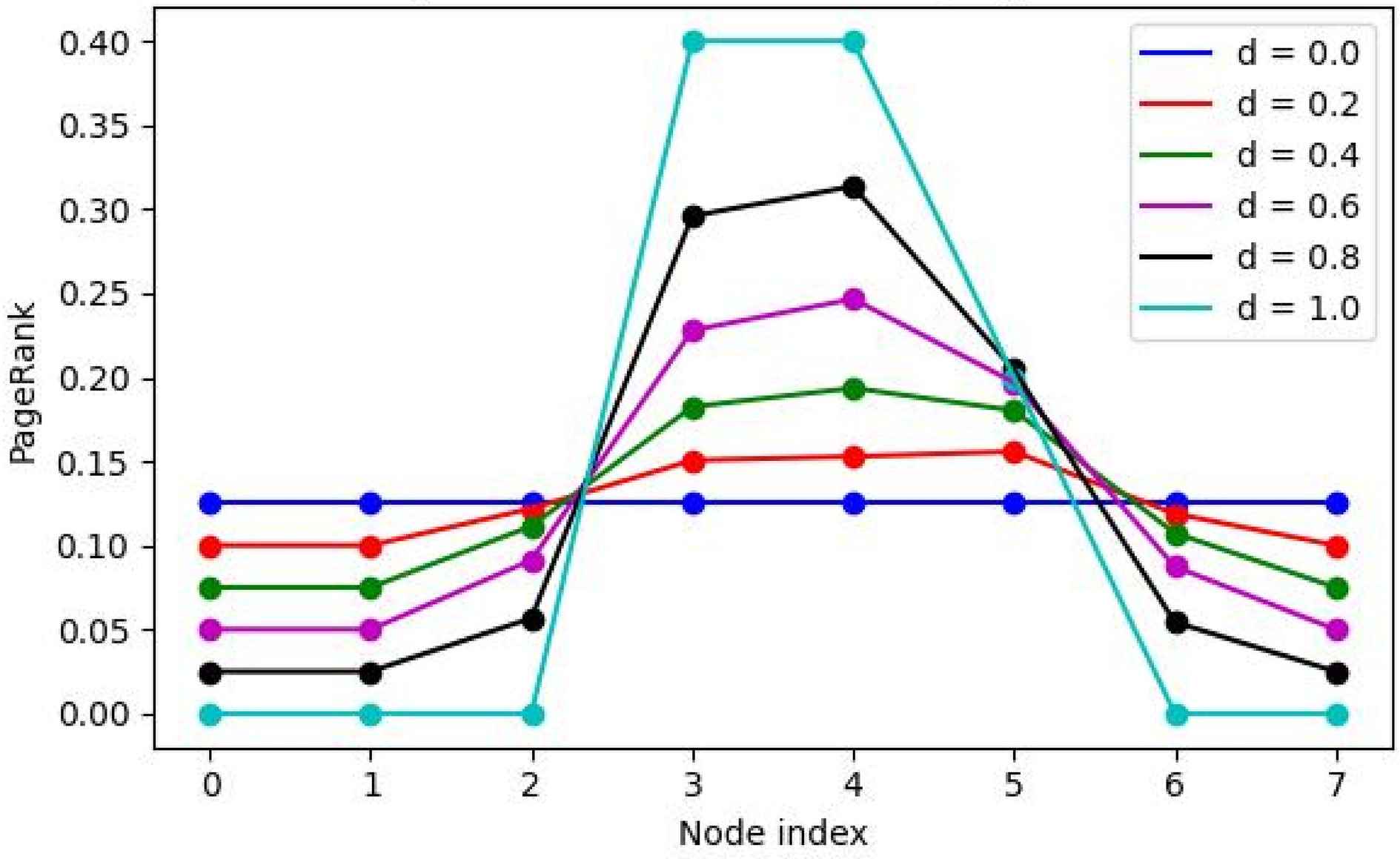
I ran 10 tests for each function (n_iterations = 10 for power iteration and n_steps = 10^6 for random walker function), with a network of size n_nodes = 10^4. I measured the running time of each test with time.time() and, with the averge running time of the 10 tests I computed the estimated running times for a network of size N_nodes = $26 \cdot 10^6$ as $\text{estimated\_time} = \frac{\text{avg\_time}}{\text{n\_nodes}} \cdot \frac{\text{N\_nodes}}{3600}$ obtaining the following results

Function	Average running time ($n = 10^4$)	Estimated running time ($N = 26 \cdot 10^6$)
Power iteration	0.879 seconds	0.635 hours
Random walker	140.255 seconds	101.295 hours

f) (2 pt) Investigate the effect of the damping factor d on the PageRank values of the network used in parts a)-c). Repeat the PageRank calculation with, e.g., 5 different values of $d \in [0, 1]$ and plot the PageRank as a function of node index (plots of all values of d in the same figure).

Interpret the results. How and why does the change of d affect the rank of the nodes and the absolute PageRank values? Explain what happens when $d = 0$ and when $d = 1$.

In case you do not trust your implementations in b) and c), you can use PageRank values obtained with `nx.pagerank` in this last task.



When the damping factor d is set to 0, it means that the random surfer never follows links and always jumps to a random page. As showed in the plot above, the PageRank algorithm essentially becomes a uniform distribution, where all nodes have equal Page Rank values.

Otherwise, when the damping factor d is set to 1, it means that the random surfer always follows links and never jumps to a random page. In this case the Page Rank algorithm becomes purely link-based and it doesn't incorporate any randomization, so values are solely determined by the network's link structure and node degrees such that nodes with more incoming links will have higher Page Rank values.