1 Question 1

Since the graph G has M related components and 2M nodes, we can deduce that a related component of G has exactly 2 nodes. Each connected component is therefore a complete graph, since its two nodes are linked by an edge.

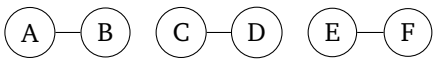

Figure 1: Example graph G with M = 3

Let x_{ij} be a node with $i \in \{1, ..., M\}$ the index of the connected component to which the node belongs and $j \in \{1, 2\}$ the number of the node in the connected component.
The random walk produced from node x_{ij} will always oscillate between the node and the other node x_{ik} (with $k \neq j$ and $k \in \{1, 2\}$) to which it is connected within the same connected component. Its composition will be as follows (before random shuffling):

$$x_{ij}x_{ik}x_{ij}x_{ik}x_{ij}x_{ik}...$$

Nodes from different connected components will therefore never appear together in a random walk, whereas nodes within the same connected component will always appear together in a random walk.
Within a connected component For two nodes of the same related component, the random walks have the same composition; consequently, the cosine similarity of two nodes of the same component is very high, close to 1.
Between two connected components For two nodes of different connected components, the random walks have no elements in common; consequently, the cosine similarity of two nodes of the same component is very low.

2 Task 8

We use logistic regression for classification in both methods. We obtain an accuracy of **1.0 for the Deep Walk** method and an accuracy of **0.857 for the spectral decomposition** of the random walk normalized graph Laplacian method.

3 Question 2

3.1 DeepWalk

The DeepWalk algorithm has a time complexity of $O(|V| * d * E)$, where:

- $|V|$ is the number of nodes in the graph
- d is the length of the random walks
- E is the number of edges in the graph

This complexity can be further reduced to $O(|V| * d) + O(E * d^2)$ by using a more efficient implementation of the SkipGram algorithm. [3]

3.2 Spectral Embedding

The spectral embedding technique has a time complexity of $O(|E| * d^2)$, where:

- $|E|$ is the number of edges in the graph
- d is the desired embedding dimension

This complexity is due to the need to compute the eigenvectors of the adjacency matrix or the Laplacian matrix of the graph. [1]

4 Question 3

The omission of self-loops in a graph would have a significant impact on the architecture of a GNN and on the evolution of hidden states within the network. [2]

4.1 Impact on a single-layer GNN

In a single-layer GNN, the absence of self-loops would deprive the network of the possibility of directly incorporating the characteristics specific to each node into its hidden representation. This would limit the GNN's ability to capture the intrinsic characteristics of each node, and could hamper its performance on tasks that rely heavily on node-level information.
Take, for example, a node classification task where the objective is to predict the category of each node based on its characteristics. Without self-loops, the GNN would rely solely on the characteristics of the node's neighbors to make its prediction. This could be problematic if the node's own characteristics are highly informative about its category.

4.2 Impact on a multi-layer GNN

In a multi-layer GNN, the absence of self-loops would have a cumulative effect on the evolution of hidden states. As information propagates through the layers, the network would gradually lose the ability to distinguish between nodes and their neighbors, leading to a more homogeneous representation of the graph.
This could be detrimental to tasks requiring GNN to maintain a clear distinction between nodes, such as node classification or link prediction. In node classification, the aim is to group nodes according to their similarities.Without self-looping, GNN may struggle to identify distinct groups due to the loss of node-specific information.Similarly, in link prediction, where the objective is to predict which node pairs are likely to be connected, GNN may struggle to identify pairs sharing strong node-level features.

5 Question 4

5.1 Cycle graph

$$A = \begin{bmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{bmatrix} \qquad A + \mathbb{I} = \begin{bmatrix} 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \end{bmatrix}$$

$$D = \begin{bmatrix} 3 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 3 \end{bmatrix} \qquad D^{-\frac{1}{2}} = \begin{bmatrix} 0.5773 & 0 & 0 & 0 \\ 0 & 0.5773 & 0 & 0 \\ 0 & 0 & 0.5773 & 0 \\ 0 & 0 & 0 & 0.5773 \end{bmatrix}$$

$$\mathbf{A} = \begin{bmatrix} 0.3333 & 0.3333 & 0 & 0.3333 \\ 0.3333 & 0.3333 & 0.3333 & 0 \\ 0 & 0.3333 & 0.3333 & 0.3333 \\ 0.3333 & 0 & 0.3333 & 0.3333 \end{bmatrix} \qquad \mathbf{A}X = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

$$\mathbf{A}XW_0 = \begin{bmatrix} 0.5 & 0.2 \\ 0.5 & 0.2 \\ 0.5 & 0.2 \\ 0.5 & 0.2 \end{bmatrix} \qquad Z_0 = \begin{bmatrix} 0.5 & 0.2 \\ 0.5 & 0.2 \\ 0.5 & 0.2 \\ 0.5 & 0.2 \end{bmatrix}$$

$$\mathbf{A}Z_0 = \begin{bmatrix} 0.5 & 0.2 \\ 0.5 & 0.2 \\ 0.5 & 0.2 \\ 0.5 & 0.2 \end{bmatrix} \qquad \mathbf{A}Z_0W_1 = \begin{bmatrix} -0.07 & -0.08 & 0.38 & 0.39 \\ -0.07 & -0.08 & 0.38 & 0.39 \\ -0.07 & -0.08 & 0.38 & 0.39 \\ -0.07 & -0.08 & 0.38 & 0.39 \end{bmatrix}$$

$$Z_1 = \begin{bmatrix} 0 & 0 & 0.38 & 0.39 \\ 0 & 0 & 0.38 & 0.39 \\ 0 & 0 & 0.38 & 0.39 \\ 0 & 0 & 0.38 & 0.39 \end{bmatrix}$$

5.2 Star graph

$$A = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \qquad A + \mathbb{I} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{bmatrix}$$

$$D = \begin{bmatrix} 4 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 2 \end{bmatrix} \qquad D^{-\frac{1}{2}} = \begin{bmatrix} 0.5000 & 0 & 0 & 0 \\ 0 & 0.7071 & 0 & 0 \\ 0 & 0 & 0.7071 & 0 \\ 0 & 0 & 0 & 0.7071 \end{bmatrix}$$

$$\mathbf{A} = \begin{bmatrix} 0.2500 & 0.3535 & 0.3535 & 0.3535 \\ 0.3535 & 0.5000 & 0 & 0 \\ 0.3535 & 0 & 0.5000 & 0 \\ 0.3535 & 0 & 0 & 0.5000 \end{bmatrix} \qquad \mathbf{A}X = \begin{bmatrix} 1.3106 \\ 0.8535 \\ 0.8535 \\ 0.8535 \end{bmatrix}$$

$$\mathbf{A}XW_0 = \begin{bmatrix} 0.6553 & 0.2621 \\ 0.4267 & 0.1707 \\ 0.4267 & 0.1707 \\ 0.4267 & 0.1707 \end{bmatrix} \qquad Z_0 = \begin{bmatrix} 0.6553 & 0.2621 \\ 0.4267 & 0.1707 \\ 0.4267 & 0.1707 \\ 0.4267 & 0.1707 \end{bmatrix}$$

$$\mathbf{A}Z_0 = \begin{bmatrix} 0.6164 & 0.2465 \\ 0.4450 & 0.1780 \\ 0.4450 & 0.1780 \\ 0.4450 & 0.1780 \end{bmatrix} \qquad \mathbf{A}Z_0W_1 = \begin{bmatrix} -0.0863 & -0.0986 & 0.4685 & 0.4808 \\ -0.0623 & -0.0712 & 0.3382 & 0.3471 \\ -0.0623 & -0.0712 & 0.3382 & 0.3471 \\ -0.0623 & -0.0712 & 0.3382 & 0.3471 \end{bmatrix}$$

$$Z_1 = \begin{bmatrix} 0 & 0 & 0.4685 & 0.4808 \\ 0 & 0 & 0.3382 & 0.3471 \\ 0 & 0 & 0.3382 & 0.3471 \\ 0 & 0 & 0.3382 & 0.3471 \end{bmatrix}$$

In both cases, we can see that GNN produces Z_1 output that is relevant for node classification, for example. By associating each line of Z_1 with a node in sequence, we see that the attributes in Z_1 for similar nodes are equal. In the case of the cyclic graph, where all nodes have the same degree, all nodes also have the same values in Z_1. This allows us to distinguish the central node of the star graph from its adjacent nodes.

References

[1] Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps for dimensionality reduction and graph clustering. In *Proceedings of the 15th international conference on neural information processing systems*, pages 589–596. MIT Press, 2002.

[2] Thomas N. Kipf and Max Welling. Graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[3] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: A unified approach to embedding representations of large graphs. *arXiv preprint arXiv:1406.5721*, 2014.