.

Midterm Exam

EPFL EE-452 — Network Machine Learning — Spring 2024/2025

Name: ______________________________ **SCIPER:** __________

Instructions:

- You have 2 hours to complete the exam. Do not start before the teaching team gives you permission.
- The midterm is open-book: you can use any printed or handwritten resources. Electronic devices are not permitted.
- Multiple Choice Questions should be answered with only one option. Selecting more than one option leads to 0 credit for that question. No negative points are assigned to wrong answers.
- In the Open Questions, justify your answers to receive full credit.
- The exam will be graded out of 30 points in total, with 14 points allocated to the Multiple Choice Questions and 16 points to the Open Questions.
- Good luck!

Part A: Multiple Choice Questions (14 points)

A.1 (1 point): A graph without any cycles is called a forest. How many edges does a forest with N vertices and C connected components have?

- □ $N(1 + 1/C)$.
- □ $N(1 + C)$.
- □ $N - C$.
- □ $2N + C$.

Solution: The forest graph can be seen as C different connected acyclic graphs, meaning they are trees. The number of edges in one of the tree T_i is $n_i - 1$, where n_i is the number of nodes in tree T_i. Then if you sum up all edges: $\sum_{i=1}^{C}(n_i - 1) = n - C$

A.2 (1 point): The betweenness centrality for a node $v \in V$ is defined as: $g(v) = \sum_{s \neq v \neq t \in V} \frac{\sigma_{st}(v)}{\sigma_{st}}$, where σ_{st} is the number of shortest paths from s to t and $\sigma_{st}(v)$ is the number of shortest paths from s to t going through v. Which one of the following statements is always true?

- □ The betweenness centrality of nodes in a complete graph follows a Poisson distribution with parameter $\lambda = 1/N$.
- □ To compute the betweenness centrality of a given node, it is enough to run BFS on this node only.
- □ The betweenness centrality of a node can be interpreted as the probability that the node lies on the shortest path of any two other randomly sampled nodes.
- □ Removing a node with non-zero betweenness centrality increases the average shortest path distance.

Solution: The true answer is the third, directly from the definition. The first answer is false because all nodes will have a centrality value of zero. The second is false because you need to compute the shortest paths among all nodes in the graphs. A BFS on one node will only give you the shortest path starting from this node. The last answer is false, think of graph with four vertices connected as a square.

A.3 (1 point): Given a graph G with vertex set $V = \{v_1, ..., v_n\}$, we define the degree sequence of G to be the list $d(v_1), ..., d(v_n)$ of degrees in decreasing order. Select the degree sequence corresponding to an undirected, unweighted graph without self-loops.

- □ 3, 3, 2, 2, 2, 1.
- □ 7, 3, 3, 2, 2, 1.
- □ 6, 6, 6, 4, 4, 3, 3.
- □ 6, 6, 6, 4, 4, 2, 2.

Solution: The true answer is the third. You can proceed by elimination. The first answer is false because the number of nodes with odd degree should be even. The second is wrong because the first degree in the sequence is higher than the number of nodes. The last answer is wrong, because having three nodes with degree 6 would require all other nodes to have a degree of at least three, which is not the case.

A.4 (1 point): Given a graph on N nodes, which of the following is always an eigenvalue of the combinatorial Laplacian?

□ N

□ 2

□ $1/N$

□ 0

Solution: 0 is always an eigenvalue for the constant eigenvector.

A.5 (1 point): Let χ_2 be the eigenvector of the second smallest eigenvalue λ_2 of the combinatorial Laplacian of the graph in Figure 1. Which of the following statements about the entries corresponding to nodes 1 and 10 is always true?

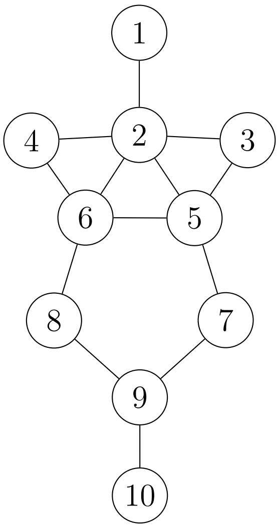


Figure 1: An unweighted and undirected graph. Numbers represent node IDs.

□ They are both positive.

□ They are both negative.

□ They have opposite signs.

□ They are equal.

Solution: They have opposite signs, as they are the farthest nodes in the graph and χ_2 corresponds to the signal with the smoothest variation on the graph, orthogonal to the constant vector.

A.6 (1 point): Using the PageRank algorithm on the graph from Figure 1, which node would get the highest ranking?

□ Node 1.

□ Node 2.

□ Node 9.

□ Node 10.

Solution: Node 2 has the highest PR ranking.

A.7 (1 point): Let us define a smoothness function $F(f_G, G) = f_G^T L f_G$, where f_G denotes a function defined over the graph G, whose combinatorial Laplacian is denoted by L. Then, the following holds:

□ If $F(h_G, G) = F(g_G, G) = 0$, then $h_G = g_G$ over G.

□ If $F(h_G, G) > F(g_G, G)$, then h_G varies more smoothly than g_G over G.

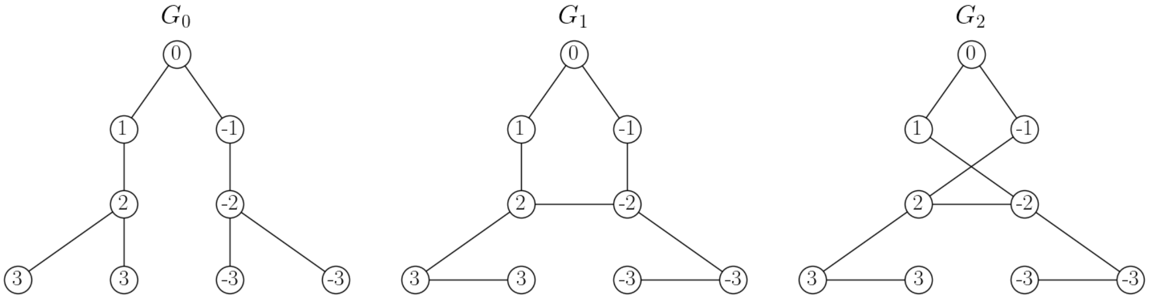
□ Let y be a noisy graph signal. $\hat{h}_G$, defined as $\hat{h}_G = \operatorname{argmin}_{h_G} \|y - h_G\|_2^2 + \lambda F(h_G, G)$, corresponds to a high-pass filtered version of y.

□ None of the above.

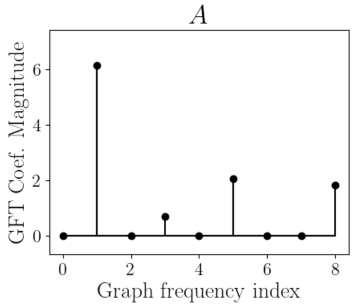
Solution:

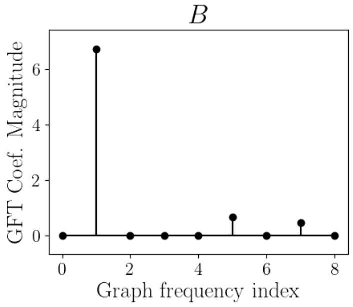
- **False.** It only means that both and h_G and g_G are constant, but they can be different.
- **False.** The opposite relation, g_G varies more smoothly than h_G.
- **False.** $\hat{h}$ is a low-pass version filter.
- **True.** From above.

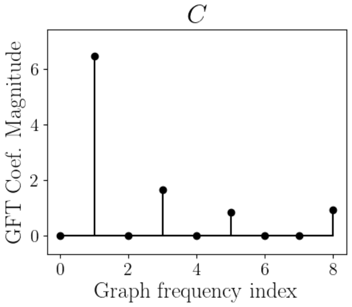
A.8 (1 point): Consider the three graphs G_0, G_1, and G_2 represented below. The signal defined over each graph is given by the numbers inscribed in its nodes.



Which of the following combinations correctly matches each graph to the corresponding plot of the magnitude of its Graph Fourier Transform (GFT) coefficients?







- □ $G_0 \to A, \quad G_1 \to B, \quad G_2 \to C$
- □ $G_0 \to B, \quad G_1 \to C, \quad G_2 \to A$
- □ $G_0 \to C, \quad G_1 \to A, \quad G_2 \to B$
- □ $G_0 \to B, \quad G_1 \to A, \quad G_2 \to C$

Solution: Between G_0 and G_1: In G_1, the edges contributing between nodes of signal 3 do not have any variation, but there is one extra link (comparatively with G_0) connecting nodes of signal 2 and -2 ($\Delta = 4$). In contrast, G_0 has two edges connecting nodes with different values of nodes, but just with $\Delta = 1$. Thus, G_1 will contain stronger high frequency components than G_0.

Between G_1 and G_2: In G_1, the edges that differ from G_2 have $\Delta = 1$, while the ones from G_2 have $\Delta = 3$. Thus, G_2 will contain stronger high frequency components than G_1.

Considering these, we necessarily have the correspondence: $G_0 \to B, \quad G_1 \to C, \quad G_2 \to A$ (Option B).

A.9 (1 point):

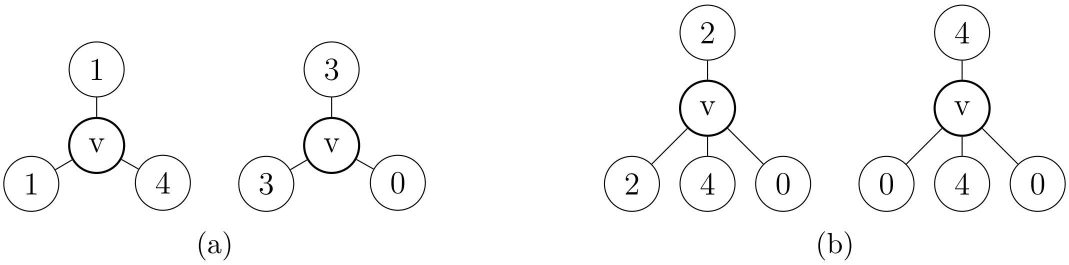


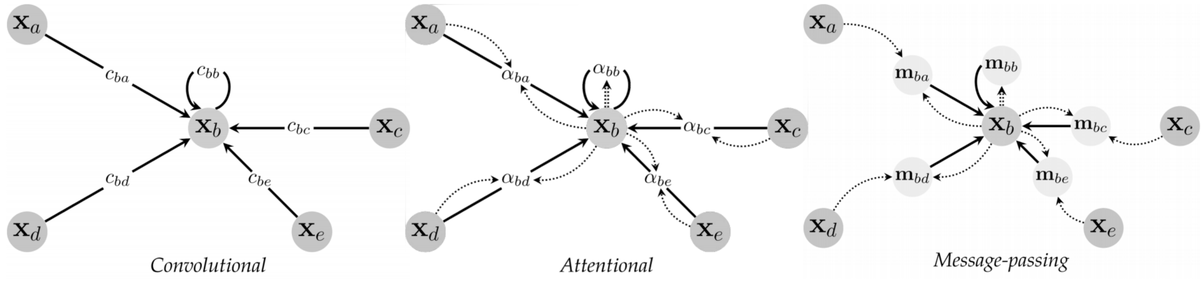
Figure 2: Various graph structures.

Figures 2a and Figure 2b each show two graphs with node features indicated by the numbers inside the nodes. From the perspective of node v, which statement accurately describes the ability of the Mean, Max, Min, and Standard Deviation (STD) aggregators to distinguish the neighborhood of v?

- □ STD fails both in Figure 2a and 2b.
- □ Min fails in Figure 2a but can differentiate in Figure 2b.
- □ Max fails in Figure 2a but can differentiate in Figure 2b.
- □ Mean fails both in Figure 2a and 2b.

Solution: In Figure 2a, Mean and STD fail whereas Max and Min are able to differentiate the neighborhood of v. In Figure 2b, only STD is able to differentiate the neighborhood of v. Hence, the correct option is that Mean fails in both cases.

A.10 (1 point):



$$\mathbf{h}_i = \phi\left(\mathbf{x}_i, \bigoplus_{j \in \mathcal{N}_i} c_{ij}\psi(\mathbf{x}_j)\right) \qquad \mathbf{h}_i = \phi\left(\mathbf{x}_i, \bigoplus_{j \in \mathcal{N}_i} a(\mathbf{x}_i, \mathbf{x}_j)\psi(\mathbf{x}_j)\right) \qquad \mathbf{h}_i = \phi\left(\mathbf{x}_i, \bigoplus_{j \in \mathcal{N}_i} \psi(\mathbf{x}_i, \mathbf{x}_j)\right)$$

Figure 3: An illustration of different GNN layers. Each node i has a node feature $\mathbf{x}_i$ and a node embedding after the GNN layer denoted as $\mathbf{h}_i$. The set $\mathcal{N}_i$ denotes the direct neighbors of node i. The operator $\bigoplus$ indicates a permutation-invariant aggregation. The functions ψ and ϕ are learnable transformations. c_{ij} denote fixed convolutional weights whereas $a(\mathbf{x}_i, \mathbf{x}_j)$ are learned attention weights.

Which of the following statements correctly describes the relative representational power of the GNN layers in Figure 3? The notation architecture X $\subseteq$ architecture Y means that every function that can be represented by architecture X can also be represented by architecture Y.

□ attentional $\subseteq$ message-passing $\subseteq$ convolutional

□ message-passing $\subseteq$ convolutional $\subseteq$ attentional

□ convolutional $\subseteq$ attentional $\subseteq$ message-passing

□ attentional $\subseteq$ convolutional $\subseteq$ message-passing

Solution: As seen in Figure 3, a convolutional layer updates the embedding $\mathbf{h}_i$ as

$$\mathbf{h}_i = \phi\Big(\mathbf{x}_i, \bigoplus_{j \in \mathcal{N}_i} c_{ij}\psi(\mathbf{x}_j)\Big)$$

Here, c_{ij} is a fixed weight. One can also see that an attention layer updates the embedding $\mathbf{h}_i$ as

$$\mathbf{h}_i = \phi'\Big(\mathbf{x}_i, \bigoplus_{j \in \mathcal{N}_i} a(\mathbf{x}_i, \mathbf{x}_j)\psi'(\mathbf{x}_j)\Big)$$

Since $a(\mathbf{x}_i, \mathbf{x}_j)$ depends on node features, the model can adaptively weight each neighbor, and any convolutional layer can be recovered by choosing

$$a(\mathbf{x}_i, \mathbf{x}_j) = c_{ij}$$

Formally,

$$\forall (c_{ij}, \psi, \phi) \quad \exists (a(\cdot), \psi', \phi') \quad \text{such that} \quad \mathbf{h}_i^{(\text{att})} = \mathbf{h}_i^{(\text{conv})}$$

Hence, convolutional $\subseteq$ attentional. Similarly, a message-passing layer is defined as

$$\mathbf{h}_i = \phi^*\Big(\mathbf{x}_i, \bigoplus_{j \in \mathcal{N}_i} \psi^*\big(\mathbf{x}_i, \mathbf{x}_j\big)\Big)$$

The term $a(\mathbf{x}_i, \mathbf{x}_j)\psi'\big(\mathbf{x}_j\big)$ in the attentional layer is one particular way of defining $\psi^*\big(\mathbf{x}_i, \mathbf{x}_j\big)$ in the message-passing layer. Formally,

$$\forall (a(\cdot), \psi', \phi') \quad \exists (\psi^*, \phi^*) \quad \text{such that} \quad \mathbf{h}_i^{(\text{mp})} = \mathbf{h}_i^{(\text{attn})}$$

Hence, attentional $\subseteq$ message-passing. Therefore the correct answer is convolutional $\subseteq$ attentional $\subseteq$ message-passing.

A.11 (1 point):

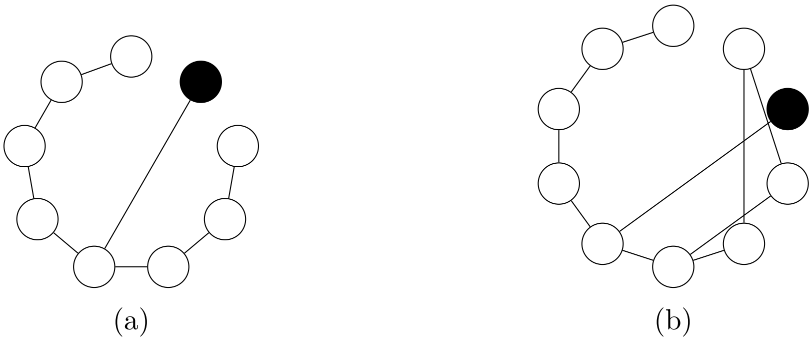

(a) (b)

Figure 4: Various graph structures.

Consider the graphs in Figure 4, where all the nodes have 1-dimensional initial feature vector $\mathbf{x}_i = [1]$. Assume that there is a simplified GNN version with no nonlinearity, no learned linear transformation, and sum aggregation. Specifically, at every layer, the embedding of node i is updated as the sum over the embeddings of its neighbors ($\mathcal{N}_i$), and its current embedding $\mathbf{h}_i^k$ to get $\mathbf{h}_i^{k+1}$. This simple GNN computes node embeddings for the black nodes in each graph. How many message-passing layers are needed to distinguish these black nodes, i.e., to have different GNN embeddings?

□ 2

□ 3

□ 4

□ 5

Solution: We need three layers. Observe that after the first layer, the black nodes will both have a feature vector of [2]. After the second layer, they will both have a feature vector of [6]. However, after the third later, 4a will have a feature vector of [18], whereas 4b will have a feature vector of [19].

A.12 (1 point): Oversquashing is a problem in graph neural networks where information from distant nodes is overly compressed as it passes through narrow parts (bottlenecks) of the graph. Consider the following graph scenarios. Three scenarios typically suffer from oversquashing, and one does not. Identify the scenario that does NOT typically experience oversquashing.

□ A large communication graph among students from three universities (i.e., a network with dense connections within the same university and highly sparse interactions between different universities).

□ A hierarchical organization graph with many management layers (i.e., a tree graph, with several intermediate levels between the root and lead nodes).

□ A grid-structured graph for image segmentation tasks (i.e., a 2D lattice graph where nodes represent pixels and edges encode spatial adjacency).

□ A protein structure graph characterized by a long sequential chain (i.e., a graph with a linear backbone and possible short-range side-chain connections).

Solution: (A) In the student communication graph, interactions are dense within each class but sparse between classes, causing bottlenecks at inter-class edges. (B) Hierarchical organization chart: tends to have oversquashing, as decisions require information flow across multiple hierarchical layers. (C) Grid-like image segmentation graph: usually does not experience oversquashing because predictions mainly depend on immediate neighbors. (D) Protein chain graph: prone to oversquashing, as node interactions frequently depend on distant residues along the chain. The correct answer is (C).

A.13 (1 point): Oversmoothing in GNNs refers to the phenomenon where increasing the number of layers leads to homogeneous node representations. Multiple strategies have been developed to overcome the oversmoothing problem in graph neural networks. From the definition above, please select the method that is the most effective in addressing oversmoothing.

- □ Add random noise to node features at each layer.
- □ Increase the number of hidden GNN layers.
- □ Add random edges between distant nodes to improve information flow.
- □ Perform graph sparsification by deleting edges that do not influence the core properties of the graph.

Solution: (A) (B) Adding random noise and increasing layers generally do not mitigate oversmoothing. Noise addition may degrade embedding quality, while more layers exacerbate the smoothing effect. (C) Adding random edges will increase information aggregation, which leads to more severe over-smoothing. (D) Graph sparsification reduces unnecessary neighborhood averaging, thus preserving distinct node features. The answer is (D).

A.14 (1 point): Graph rewiring is a preprocessing or structural modification technique used in GNNs. Virtual nodes are an example of rewiring, where an additional node is introduced and connected to all existing graph nodes to enhance message passing. Based on the definition above, which of the following correctly describes the primary purpose of adding virtual nodes?

- □ Decrease oversmoothing by adding more edges to encourage wider information aggregation.
- □ Avoid oversquashing by improving information flow.
- □ Increase computational efficiency by removing all redundant nodes from the graph.
- □ Convert directed graphs into undirected graphs to simplify the training of GNNs.

Solution: Graph rewiring aims to enhance information propagation efficiency and address bottlenecks such as oversquashing by structurally adjusting the original graph. The other choices describe incorrect or unrelated objectives. The correct answer is (B).

Part B: Open Questions (16 points)

B.1 (4 points): Consider a triangle graph with nodes A, B, and C. Each node $i \in \{A, B, C\}$ is associated with a scalar feature $h_i^{(0)} \in \mathbb{R}$ at the initial iteration ($l = 0$). For each iteration $l \geq 0$, the feature of each node is updated by taking the average of its neighbors' features from the previous iteration. Explicitly, for $l \geq 0$:

$$\begin{aligned} h_A^{(l+1)} &= \frac{1}{2}\big(h_B^{(l)} + h_C^{(l)}\big), \\ h_B^{(l+1)} &= \frac{1}{2}\big(h_A^{(l)} + h_C^{(l)}\big), \\ h_C^{(l+1)} &= \frac{1}{2}\big(h_A^{(l)} + h_B^{(l)}\big). \end{aligned}$$

Answer the following questions.

1. **(1 point)** Prove that the average value of the node features remains constant, i.e.,

$$\frac{h_A^{(l)} + h_B^{(l)} + h_C^{(l)}}{3} = \frac{h_A^{(0)} + h_B^{(0)} + h_C^{(0)}}{3}, \forall l.$$

2. **(1 point)** Prove that the difference between any two node features decreases by a factor of $\frac{1}{2}$ at each iteration. For example, for node A and node B, we have:

$$\left\| h_A^{(l+1)} - h_B^{(l+1)} \right\| = \frac{1}{2} \left\| h_A^{(l)} - h_B^{(l)} \right\|, \forall l.$$

3. **(1 point)** Deduce the final stable value of each node as $l \to \infty$.
4. **(1 point)** Interpret the results.

Solution:

1.

$$\begin{aligned} h_A^{(l+1)} + h_B^{(l+1)} + h_C^{(l+1)} &= \frac{1}{2}\Big[(h_B^{(l)} + h_C^{(l)}) + (h_A^{(l)} + h_C^{(l)}) + (h_A^{(l)} + h_B^{(l)})\Big] \\ &= \frac{1}{2}\Big[2h_A^{(l)} + 2h_B^{(l)} + 2h_C^{(l)}\Big] \\ &= h_A^{(l)} + h_B^{(l)} + h_C^{(l)}. \end{aligned}$$

Thus, the sum is invariant to l and the average,

$$\frac{h_A^{(l)} + h_B^{(l)} + h_C^{(l)}}{3},$$

remains constant. By induction, we have the results.

Grading: 0.5 points for invariant sum and 0.5 points for induction.

2.

$$\begin{aligned} h_A^{(l+1)} - h_B^{(l+1)} &= \frac{1}{2}\left(h_B^{(l)} + h_C^{(l)}\right) - \frac{1}{2}\left(h_A^{(l)} + h_C^{(l)}\right) \\ &= \frac{1}{2}\left(h_B^{(l)} - h_A^{(l)}\right) \\ &= -\frac{1}{2}\left(h_A^{(l)} - h_B^{(l)}\right). \end{aligned}$$

By applyting the absolute value on both sides of the equality and proceeding similarly for the differences $h_B^{(l)} - h_C^{(l)}$ and $h_C^{(l)} - h_A^{(l)}$, we retrieve the intended result.

Grading: 1.0 point for proving the relation between any 2 pairs of nodes.

3. Since the differences between any two node features decay geometrically (by a factor of $\frac{1}{2}$ per iteration), we have

$$\lim_{l\to\infty}\left(h_A^{(l)} - h_B^{(l)}\right) = \lim_{l\to\infty}\left(h_B^{(l)} - h_C^{(l)}\right) = \lim_{l\to\infty}\left(h_C^{(l)} - h_A^{(l)}\right) = 0.$$

Thus, as $l \to \infty$, the node features converge to a common value L. By the invariance of the sum from part (a):

$$3L = h_A^{(0)} + h_B^{(0)} + h_C^{(0)},$$

which implies

$$L = \frac{h_A^{(0)} + h_B^{(0)} + h_C^{(0)}}{3}.$$

Hence, every node converges to the average of the initial features.

Grading: 1 point for either giving the convergence value of feature distance or for giving the convergence value of feature value.

4. This result highlights that repeated averaging (or message passing) leads to all nodes becoming indistinguishable from one another, which is typically referred to as the *over-smoothing* phenomenon in literature.

Grading: 1 point for any interpretation related to over-smoothing, information vanishing, or GNN's limitation. 0.5 points for other reasonable interpretations.

B.2 (4 points): Let $x \in \mathbb{R}^n$ be a graph signal defined on a graph with symmetric Laplacian matrix $L \in \mathbb{R}^{n\times n}$. The Graph Fourier Transform of x is defined as:

$$\hat{x} = U^\top x,$$

where $U \in \mathbb{R}^{n\times n}$ is the matrix of eigenvectors of L, assumed to be orthonormal.

1. **(1 point)** Derive the inverse Graph Fourier Transform (expression) under the assumption that U is an orthonormal matrix.
2. **(2 points)** Describe one scenario where this inverse formulation does **not** hold, and briefly explain why.
3. **(1 point)** Provide an example graph for the scenario described above. You may draw the graphs or describe them clearly.

Solution: Let $L \in \mathbb{R}^{n\times n}$ be a symmetric matrix and $U \in \mathbb{R}^{n\times n}$ its matrix of orthonormal eigenvectors, so that $U^\top = U^{-1}$.

(a) **Inverse Graph Fourier Transform:**

Given the Graph Fourier Transform:

$$\hat{x} = U^\top x,$$

and using the fact that $U^\top = U^{-1}$, we recover x by:

$$x = U\hat{x}. \tag{1}$$

This follows directly from:

$$U\hat{x} = UU^\top x = Ix = x. \tag{2}$$

Grading: 0.5 points for Equation (2) and 0.5 points for Equation (1).

(b) The previous inverse formulation is not valid when it is impossible to find an orthonormal basis of eigenvectors for the Laplacian. The Spectral Theorem guarantees that any real symmetric matrix admits a complete orthonormal basis of eigenvectors. Consequently, the absence of such a basis may only occur when the Laplacian is not symmetric, as in the case of directed graphs, where asymmetric edges yield a non-symmetric Laplacian matrix.

Grading: 1 point for an invalid setting and 1 point for the justification.

(c) Directed graph (asymmetric Laplacian): A simple directed graph with two nodes:

$$(1) \to (2)$$

Hence:

$$A = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}, \quad D = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}, \quad L = D - A = \begin{bmatrix} 1 & -1 \\ 0 & 0 \end{bmatrix}.$$

L is not symmetric and does not have a orthonormal eigenbasis.

Grading: 1 point for a valid example.

B.3 (4 points): This question aims to investigate the expressive power of GNNs for learning simple graph algorithms. Consider breadth-first search (BFS), where at each step, all nodes that are connected to already visited nodes become visited. Suppose that a GNN is used to learn to execute the BFS algorithm and that the embeddings are 1-dimensional. Initially, all nodes have input feature 0, except a source node which has input feature 1. At every BFS step, all nodes that are directly connected to a visited node should become visited (i.e. get an embedding of 1), and nodes that have not been reached should remain at 0. Describe a message function, an aggregation function, and an update rule for the GNN such that it learns the task perfectly. Note that the input to the message function should be the current node embeddings and the output of the update rule should be the updated node embeddings.

Solution: We define the following components for the 1-dimensional GNN to simulate BFS.

Message Function: For each edge (j, i), define the message to be

$$m_{j \to i} = \psi(x_j) = x_j$$

Since $x_j \in \{0, 1\}$ (0 for not visited, 1 for visited), the message indicates whether or not node j is visited.

Aggregation Function: For node i, aggregate messages from all its direct neighbors $\mathcal{N}_i$ by taking the maximum:

$$m_i = \bigoplus_{j \in \mathcal{N}_i} m_{j \to i} = \max_{j \in \mathcal{N}_i} x_j$$

This aggregation function is permutation invariant and ensures that if at least one neighbor is visited, i.e., has value 1, then $m_i = 1$.

Grading: 2 points for correctly identifying the message and the aggregation functions. If confusion exists regarding the message function, 0.5 points are deducted.

Update Rule: Update the node i's embedding by combining its current state with the aggregated message:

$$x_i' = \phi\big(x_i, m_i\big) = \max\big(x_i, m_i\big).$$

This rule guarantees that once a node becomes visited, it remains visited in subsequent iterations. In this way, after a sufficient number of iterations, the GNN exactly mimics the BFS process by marking all nodes reachable from the source as 1. Thus, the described message function, aggregation operator, and update rule allow the GNN to learn the BFS algorithm perfectly.

Grading: 2 points for correct update rules.

B.4 (4 points): A graph $G = (V, E)$ is *bipartite* if it has two set of vertices V_1, V_2 such that $V_1 \cup V_2 = V$ and $V_1 \cap V_2 = \emptyset$, and all edges of the graph go from V_1 to V_2, i.e. $\forall e = (v_1, v_2) \in E$. Given an undirected bipartite graph G, show that for any eigenvalue λ of the adjacency matrix $\boldsymbol{A}$, $-\lambda$ is also an eigenvalue, and describe the relationship between the corresponding eigenvectors.

Solution: Let the nodes be ordered so that the first rows of $\boldsymbol{A}$ correspond to the N_1 nodes in V_1, and the last ones to the N_2 nodes in V_2. By definition of a bipartite graph, we can write

$$\boldsymbol{A} = \begin{bmatrix} \mathbf{0}_{N_1 \times N_2} & \boldsymbol{A}_0^\top \\ \boldsymbol{A}_0 & \mathbf{0}_{N_2 \times N_1} \end{bmatrix}, \tag{3}$$

where A_0 has size $N_1 \times N_2$ and represents the connections between V_1 and V_2. Then, let $\boldsymbol{u}$ be an eigenvector of $\boldsymbol{A}$, with $\boldsymbol{A}\boldsymbol{u} = \lambda \boldsymbol{u}$, it follows that

$$\boldsymbol{A}\boldsymbol{u} = \begin{bmatrix} \mathbf{0}_{N_1 \times N_2} & \boldsymbol{A}_0^\top \\ \boldsymbol{A}_0 & \mathbf{0}_{N_2 \times N_1} \end{bmatrix} \begin{bmatrix} \boldsymbol{u}_1 \\ \boldsymbol{u}_2 \end{bmatrix} = \begin{bmatrix} \boldsymbol{A}_0^\top \boldsymbol{u}_2 \\ \boldsymbol{A}_0 \boldsymbol{u}_1 \end{bmatrix} = \lambda \begin{bmatrix} \boldsymbol{u}_1 \\ \boldsymbol{u}_2 \end{bmatrix}, \tag{4}$$

where $\boldsymbol{u}_1$ and $\boldsymbol{u}_2$ are the eigenvector entries corresponding to nodes in V_1 and V_2 respectively.

Defining $\boldsymbol{u}' = \left[\boldsymbol{u}_1^\top, -\boldsymbol{u}_2^\top\right]^\top$, we observe that

$$\boldsymbol{A}\boldsymbol{u}' = \boldsymbol{A} \begin{bmatrix} \boldsymbol{u}_1 \\ -\boldsymbol{u}_2 \end{bmatrix} = \begin{bmatrix} -\boldsymbol{A}_0^\top \boldsymbol{u}_2 \\ \boldsymbol{A}_0 \boldsymbol{u}_1 \end{bmatrix} = (-\lambda) \begin{bmatrix} \boldsymbol{u}_1 \\ -\boldsymbol{u}_2 \end{bmatrix}. \tag{5}$$

Therefore, $\boldsymbol{u}'$ is also an eigenvector of $\boldsymbol{A}$, with eigenvalue $-\lambda$.

Grading: 1 point for adjacency $\boldsymbol{A}$, i.e. Equation (3); 1 point for relation between $\boldsymbol{u}_1$ and $\boldsymbol{u}_2$, i.e. Equation (4); 1 point for finding $\boldsymbol{u}'$; 1 point for proving the eigenpair, i.e. Equation (5).