

## Final Exam (Graph Mining – Spring 2025)

Full Name:


Student ID:


- The formula and solution process should be presented with the answer.
- The answer should be written in English.


### 1. (Graphlet Kernel) (5pt)

- a) Given the example graph G with 5 nodes, compute the graphlet count vector  $f_G$  for graphlets of size  $k=3$

Answer:

: (A, B, C) = 1

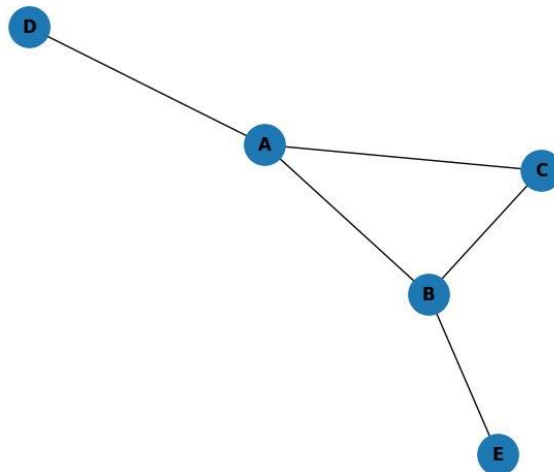
: (A, D, C), (A, B, D), (B, A, E), (B, C, E) = 4

: (A, D, E), (E, B, D), (A, C, E), (B, C, D) = 4

  
: (C, D, E) = 1

- b) What is its main principle, what computational challenge does it face, and how does it address graph isomorphism when dealing with unlabeled versus labeled graphs?

Answer: Graphlet kernel represents graphs by counting small subgraphs (graphlets) and computes similarity based on their frequency vectors. Compute all graphlets is heavy. Graph isomorphism with unlabeled graph use canonical forms or hashing to identify isomorphic graphlets, with labeled graph, incorporate node/edge labels into graphlet matching.



2. (Shortest Path Kernel) (5pt)

a) Given the two graphs, apply the shortest path kernel. (label R as red node and B as blue node)

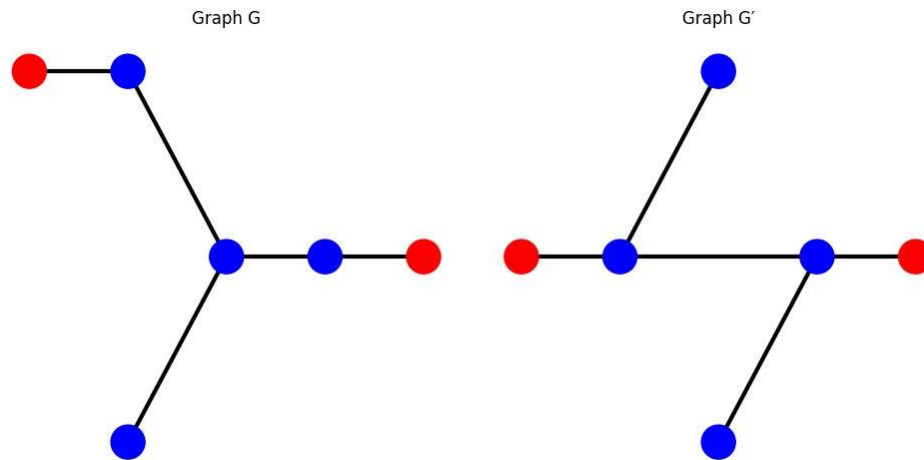
Answer: (B-B), (R-R), (B-R), (B-X-B), (R-X-R), (R-X-B), (B-X-X-B), (R-X-X-R), (B-X-X-R), (B-X-X-X-B), (R-X-X-X-R), (B-X-X-X-R)

$$G = [3, 0, 2, 3, 0, 2, 0, 0, 4, 0, 1, 0]$$

$$G' = [3, 0, 2, 2, 0, 4, 1, 1, 2, 0, 0, 0]$$

b) Why is it faster than checking for subgraph isomorphisms?

Answer: Because the SPK compares graphs using precomputed all-pairs shortest path distances, avoiding the NP-complete problem of subgraph isomorphism, making it more efficient for large graphs.

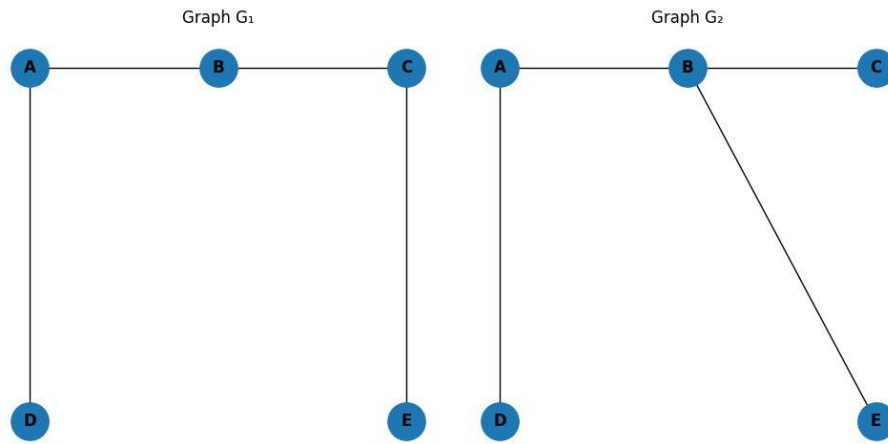


3. (WL Kernel) (5pt)

a) Given two graphs, draw whether the graphs are isomorphic

b) Explain the underlying idea of the Weisfeiler-Lehman kernel. How does the iterative relabeling process help capture graph structure?

Answer: WL kernel captures graph structure by iterative updating node labels based on their neighbor's labels, effectively encoding local substructure. This relabeling increasingly expressive features that reflect multi-hop neighborhoods, enabling efficient and powerful graph comparison.



4. (Node2Vec) (10pt)

- a) Consider an undirected graph with eight nodes in the following figure. A biased random walk (Node2Vec algorithm) has the return parameter  $p = 0.5$  and the in-out parameter  $q = 0.5$ . Assume that all edge weights of the graph are 1 and the walker is currently on node C by departing from node E. Calculate transition probabilities from node C to its neighbors.

There are five neighbors of node C: B, D, E, F, and H. Since the walker starts from E to C, the transition probabilities from C to its neighbors can be calculated as follows:

$$P_{C \rightarrow B} = 1 \times \frac{1}{q} = \frac{1}{0.5} = 2$$

$$P_{C \rightarrow D} = 1 \times \frac{1}{q} = \frac{1}{0.5} = 2$$

$$P_{C \rightarrow E} = 1 \times \frac{1}{p} = \frac{1}{0.5} = 2$$

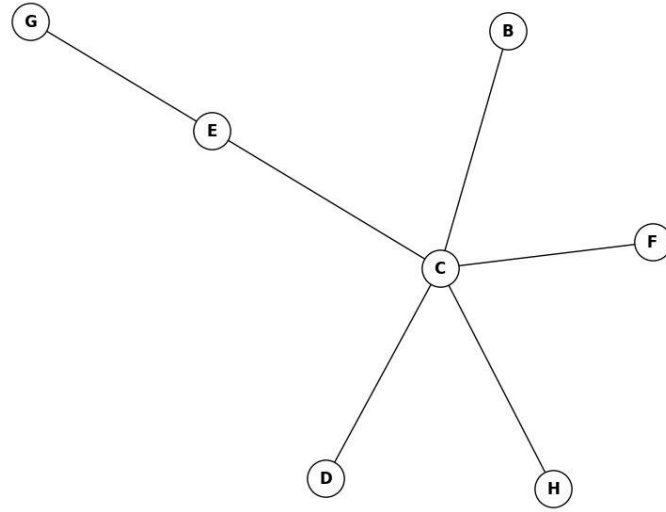
$$P_{C \rightarrow F} = 1 \times \frac{1}{q} = \frac{1}{0.5} = 2$$

$$P_{C \rightarrow G} = 1 \times 1 = 2$$

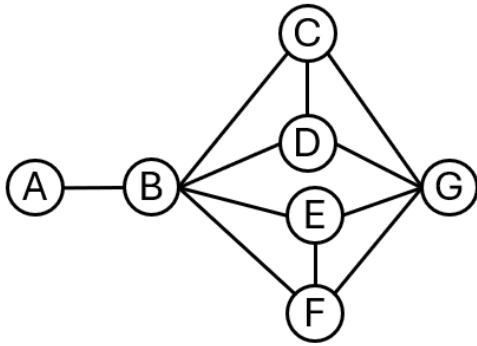
$$P_{C \rightarrow H} = 1 \times \frac{1}{q} = \frac{1}{0.5} = 2$$

- b) Describe how Node2Vec uses two hyperparameters — the return parameter  $p$  and the in-out parameter  $q$  — to interpolate between Breadth-First Search (BFS) and Depth-First Search (DFS). How does each parameter influence the walk behavior?

Answer: Return parameter ( $p$ ) controls the likelihood of revisiting a node, higher  $p$  discourages backtracking, promoting exploration (DFS-like behavior). In-out parameter guides the trade-off between staying close (BFS) or moving far (DFS). Lower  $q$  favors BFS (local neighborhood), while higher  $q$  encourages DFS (global exploration)



5. (LINE) Consider an undirected graph  $G$  of seven nodes A, B, C, D, E, F, and G given in the following figure. Let  $x_i$  is the initial vector representations of a node  $i$ , as shown in Eq. 1. (10pt)



$$x_i = (w_{i1}, w_{i2}, \dots, w_{i|V|}) \quad (1)$$

where  $w_{ik} = \begin{cases} 1 & \text{if } (i, k) \in E, \\ 0 & \text{otherwise} \end{cases}$ ,  
 $|V|$  denotes the number of nodes in the graph.

- a) Calculate the initial vectors of all the nodes in graph  $G$  based on Eq. 1.

$$x_A = (0, 1, 0, 0, 0, 0, 0)$$

$$x_B = (1, 0, 1, 1, 1, 1, 0)$$

$$x_C = (0, 1, 0, 1, 0, 0, 1)$$

$$x_D = (0, 1, 1, 0, 0, 0, 1)$$

$$x_E = (0, 1, 0, 0, 0, 1, 1)$$

$$x_F = (0, 1, 0, 0, 1, 0, 1)$$

$$x_G = (0, 0, 1, 1, 1, 1, 0)$$

- b) Calculate the second-order proximity between pairs of nodes (A, D) and (B, G) based on Manhattan Distance (the distance between two data points is computed as  $D_{(x,y)} = \sum_{i=1}^n |x_i - y_i|$ , where  $n$  is the number of dimensions).

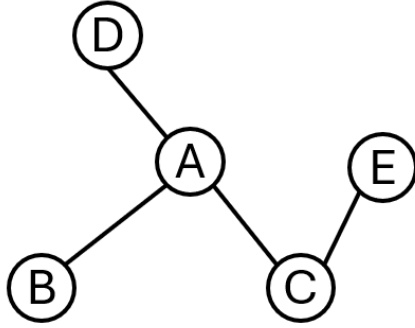
$$\begin{aligned}
 D_{AD} &= \sum_{i=1}^n |x_{A,i} - x_{D,i}| \\
 &= |(0, 1, 0, 0, 0, 0, 0) - (0, 1, 1, 0, 0, 0, 1)| \\
 &= 2
 \end{aligned}$$

$$\begin{aligned}
D_{BG} &= \sum_{i=1}^n |x_{B,i} - x_{G,i}| \\
&= |(1,0,1,1,1,0) - (0,0,1,1,1,0)| \\
&= 1
\end{aligned}$$

c) How does LINE separately model and preserve both proximities in its objective functions?

Answer: First-order proximity by preserving observed direct connections between node pairs using joint probability in its objective, second-order proximity by modeling similarity in neighborhood structures, using conditional probabilities over context nodes.

6. (HOPE) Consider an undirected graph  $G$  of five nodes A, B, C, D, and E given in the following figure. (10pt)



Equation (1):

$$S = (M_g)^T \cdot M_l,$$

$$M_g = I - \beta \cdot A,$$

$$M_l = \beta \cdot A,$$

where  $I$  refers to the Identity matrix.

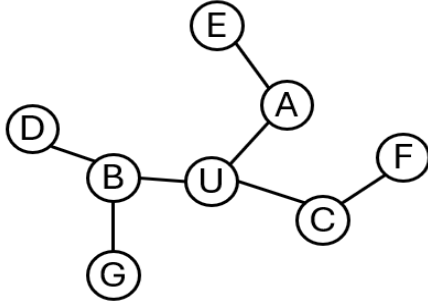
- a) (Asymmetric Transitivity Preserving Graph Embedding), a high-order proximity matrix  $S$  is defined in Eq. (1). Calculate the  $S$  matrix based on the Katz proximity measurement with  $\beta = 1$ .

$$\begin{aligned}
S &= \left( \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} - \begin{bmatrix} 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix} \right)^T \begin{bmatrix} 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix} \\
&= \begin{bmatrix} 1 & -1 & -1 & -1 & 0 \\ -1 & 1 & 0 & 0 & 0 \\ -1 & 0 & 1 & 0 & -1 \\ -1 & 0 & 0 & 1 & 0 \\ 0 & 0 & -1 & 0 & 1 \end{bmatrix}^T \begin{bmatrix} 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix} \\
&= \begin{bmatrix} -3 & 1 & 1 & 1 & -1 \\ 1 & -1 & -1 & -1 & 0 \\ 1 & -1 & -2 & -1 & 1 \\ 1 & -1 & -1 & -1 & 0 \\ -1 & 0 & 1 & 0 & -1 \end{bmatrix}
\end{aligned}$$

- b) What specific challenge does HOPE aim to solve that is not adequately addressed by methods like LINE or DeepWalk?

Answer: HOPE addresses preserving asymmetric transitivity in directed graphs, which methods like LINE and DeepWalk fail to capture due to their focus on symmetric or undirected relationships.

7. (Struc2Vec) Consider an undirected, unweighted graph given in the following figure. From the Struc2Vec method, let  $R_k(U)$  denote the set of neighbor nodes within  $k$ -hop distance rooted at node  $U$ . Let  $S(v)$  denotes the ordered degree sequence of a node set  $v \subset V$  (from the minimum to maximum values). Let  $f_k(u, v)$  denotes the structural distance between  $u$  and  $v$ . (10pt)



$$f_k(u, v) = f_{k-1}(u, v) + g(S(R_k(u)), S(R_k(v))) \quad (1)$$

where  $g(\cdot)$  measures the distance between the ordered degree sequences, which is based on the Manhattan Distance ( $g(x, y) = \sum_{i=1}^n |x_i - y_i|$ , with  $n$  is the number of dimensions).  
 $f_0(u, v) = 0$

- Calculate  $R_0(B)$ ,  $R_1(B)$ ,  $S(R_0(B))$ , and  $S(R_1(B))$ .
- Calculate the structural distance  $f_1(E, G)$  between two nodes  $E$  and  $G$ .
- Why is structural identity important in certain graph learning tasks?

Answer:

a)

$$R_0(B) = \{B\}$$

$$R_1(B) = \{D, U, G\}$$

$$S(R_0(B)) = \{3\}$$

$$S(R_1(B)) = \{1, 3, 1\}$$

b)

$$\begin{aligned} f_1(E, G) &= f_0(E, G) + g(S(R_0(E)), S(R_0(G))) \\ &= 0 + g(\{2\}, \{3\}) \\ &= 0 + 1 \\ &= 1 \end{aligned}$$

- Structure identity is crucial in graph learning tasks where node roles of functions are defined by their structural rather than their neighbors. In social networks, individuals with similar roles (like interns or managers) may not be directly connected but share similar structural positions.

8. (metapath2vec) Consider a heterogeneous graph given in the following figure. There are three types of nodes in the academic network: *Author* (A), *Paper* (P), and *Venue* (V). (10pt)

- List all the meta-path APA and APVPA.

Answer:

APA:

$$a_1 p_1 a_2$$

$$a_2 p_1 a_1$$

$a_2 p_2 a_3$

$a_2 p_2 a_4$

$a_3 p_2 a_2$

$a_3 p_2 a_4$

$a_4 p_2 a_2$

$a_4 p_2 a_3$

$a_4 p_3 a_5$

$a_5 p_3 a_4$

## APVPA

$a_1 p_1 v_1 p_2 a_2$

$a_1 p_1 v_1 p_2 a_3$

$a_1 p_1 v_1 p_2 a_4$

$a_2 p_1 v_1 p_1 a_1$

$a_2 p_1 v_1 p_2 a_3$

$a_2 p_1 v_1 p_2 a_4$

$a_2 p_1 v_1 p_2 a_4$

$a_3 p_2 v_1 p_1 a_1$

$a_3 p_2 v_1 p_1 a_2$

$a_3 p_2 v_1 p_2 a_2$

$a_3 p_2 v_1 p_2 a_4$

$a_4 p_2 v_1 p_1 a_1$

$a_4 p_2 v_1 p_1 a_2$

$a_4 p_2 v_1 p_2 a_2$

$a_4 p_2 v_1 p_2 a_2$

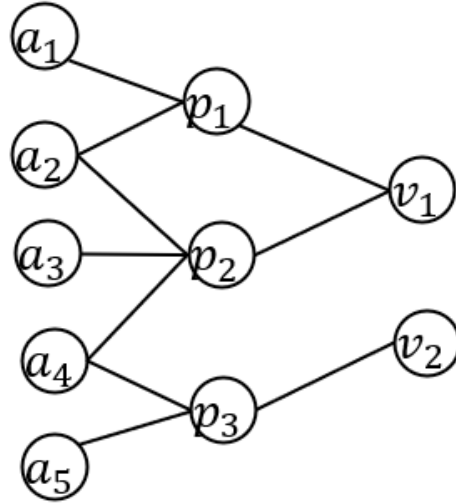
$a_4 p_3 v_2 p_3 a_5$

$a_5 p_3 v_2 p_3 a_4$

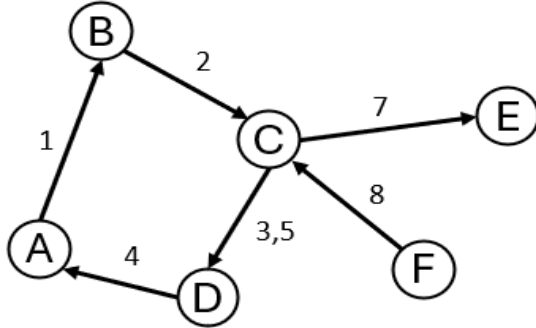
- b) How does meta-path-guided sampling help capture semantic and structural correlations between heterogeneous node types?

Answer: Metapath2vec sampling enhances the capture of semantic and structural correlations in heterogeneous networks by directing random walks along predefined sequences of node types and relations, known as meta-paths. Metapath2vec ensures that the generated node sequences reflect meaningful relationships between different types of nodes, facilitating the learning of embeddings that preserve both the network's structure and semantics. Metapath2vec effectively models complex interactions in heterogeneous networks, outperforming traditional methods that do not account for node and relation heterogeneity.

Author (A)    Paper (P)    Venue (V)



9. (CTDNE) Consider a dynamic graph given in the following figure. The edges are labeled by time. (10pt)



Equation (1):

$$N_t(v) = \{(u, t') | e = (v, u, t') \in E_T \wedge T(e) > t\},$$

where  $T(e)$  refers to the timestamp of the edge  $e$

- From the CTDNE method, the temporal neighbors of a node  $v$  at time  $t$  can be computed as Eq. (1). Calculate the set of temporal neighbors of the node D at time  $t = 0$ .
- List all the temporal random walks from node D to other nodes with length 4.
- What are the main limitations of static embedding methods like DeepWalk, Node2Vec, or LINE when applied to temporal (time-evolving) networks?

Answer:

a)  $N_D^{t=0} = \{A\}$

b) DABCE, DABCD

c) Static embedding methods like DeepWalk, Node2Vec, LINE assume fixed graph structure and cannot capture temporal dynamics, leading to outdated embeddings as network evolves, failing to model changes in node roles, edge information, and temporal patterns over time.

10. (dynnode2vec) Consider two snapshots of a dynamic graph with structural evolution from time  $t=1$  to  $t=2$ , as shown in the following figure. The evolving nodes in the timestamp  $t$  are defined as in Eq. 1 based on the Dynnode2vec method. (10pt)

- Calculate  $V_{add}$ ,  $E_{add}$ ,  $V_{del}$ , and  $E_{del}$  at timestamp  $t=2$ .
- Calculate  $\Delta V_2$ .



- c) Describe key challenges in dynamic networks such as temporal evolution, node/edge addition or removal, and the need for temporal consistency in embeddings.

Answer:

a)

$$V_{add} = \{6,7\}$$

$$E_{add} = \{e_{16}; e_{17}; e_{57}\}$$

$$V_{del} = \{2\}$$

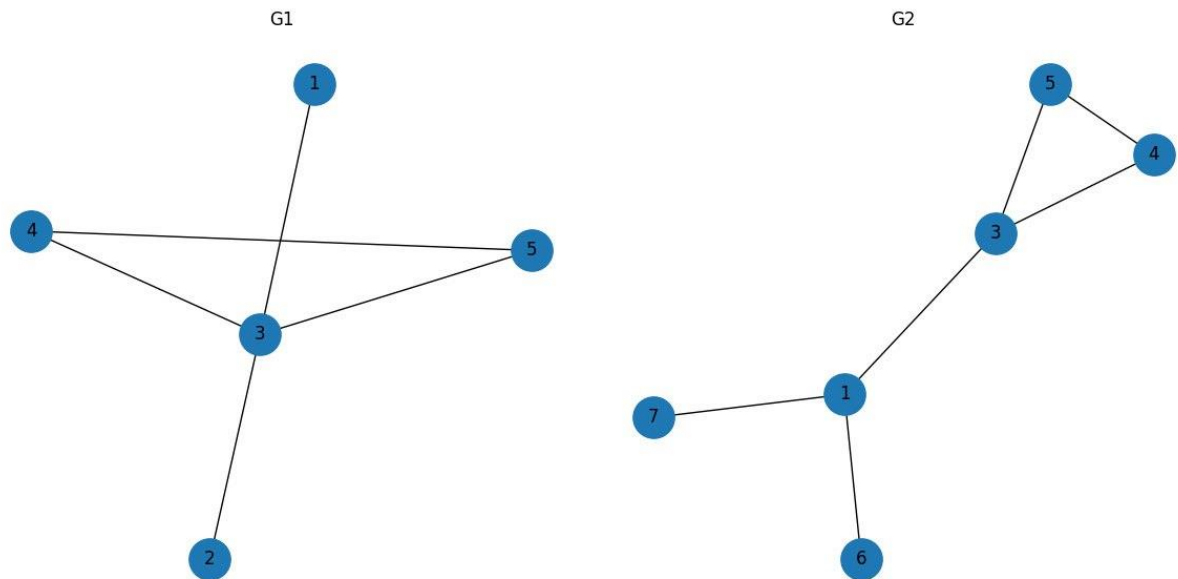
$$E_{del} = \{e_{23}\}$$

b)

$$\Delta V_2 = \{6,7\} \cup \{1,6,7\}$$

$$= \{1,6,7\}$$

- c) Key challenges in dynamic network include temporal evolution which is capturing changes in structure over time, node/ edge update which is handling additions and removals efficiently, and temporal consistency which is maintaining stable embeddings across time steps.



Equation (1):  $\Delta V_t = V_{add} \cup \{v_i \in V_t | \exists e_i = (v_i, v_j) \in (E_{add} \cup E_{del})\}$ , where  $V_{add}$  and  $E_{add}$  denote the sets of new nodes and edges that are added, respectively.  $V_{del}$  and  $E_{del}$  are the sets of new nodes and edges that are deleted, respectively.

11. (Div2Vec) (5pt)

- a) Given graph, use Div2Vec to perform a random walk of length 3 starting from node A.

Answer: Node A (k=1) => Node C

Node C (k=3)

Neighbors: A (k=1), B (k=1), D (k=3)

Div2Vec bias favors higher-degree nodes => Node D

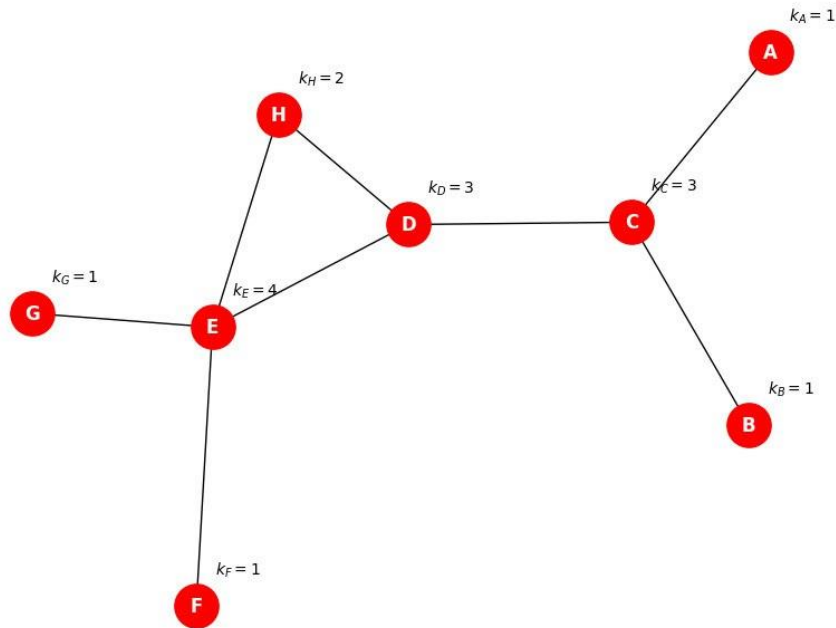
Node D (k=3)

Neighbors: C (visited), E (k=4), H (k=2) => Node E

Final, A=>C=>D=>E

- b) How Div2Vec balances accuracy vs. diversity using its sampling function?

Answer: Div2Vec balances accuracy and diversity by using sampling function that interpolates between traditional random walk probabilities and a diversity-promoting score (entropy or dissimilarity). A tunable parameter controls this trade-off, ensuring walks are both informative and diverse.



12. (APP) (5pt)

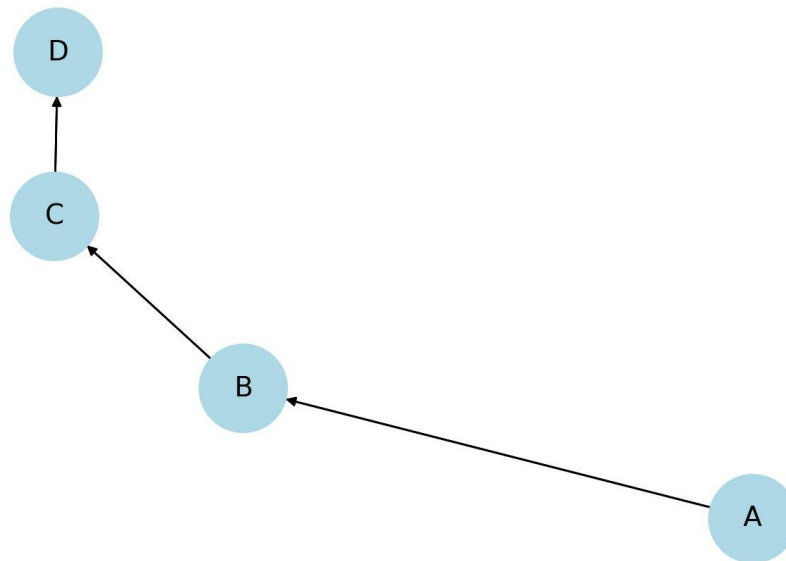
- a) Given directed graph with 4 nodes, apply APP method to simulate two sample random walks starting at node A, with restart probability  $\alpha=0.3$ , and apply path sharing optimization.

Answer: Depend on your choice, it could be  $A \Rightarrow B \Rightarrow C \Rightarrow A$ , or  $A \Rightarrow B \Rightarrow A$

- b) Compare how APP differs from Node2Vec in both walk structure and training objective.

Answer: APP uses restart-based walks for direct, asymmetric proximity, Node2Vec uses biased walks (BFS/DFS) for structural/homophilic patterns. APP learns asymmetric (source-context) embeddings, while Node2Vec learns symmetric embeddings via skip-gram

Directed Graph:  $A \rightarrow B \rightarrow C \rightarrow D$



13. (JUST) (5pt)

- a) Simulate a 2-step walk starting from node A1 using the JUST algorithm with  $\alpha=0.8$ ,  $\ell=1$ , with 0.5, 0.6 for walk 1, and walk 2, respectively.

Answer: Depend on your choice, it could be  $A1 \Rightarrow P1 \Rightarrow V1$  or  $A1 \Rightarrow A2 \Rightarrow P2$

- b) How JUST's probabilistic transition strategy (jump/stay decision with decay functions) addresses the limitations of hand-crafted meta-paths.

Answer: JUST's probabilistic transition with decay functions replaces rigid, hand-crafted meta-paths by allowing flexible, data-driven walks that adaptively weigh node type transitions. This captures richer semantics and avoids reliance on manually defined paths, improving generalization and scalability

Heterogeneous Graph (A: Author, P: Paper, V: Venue)

