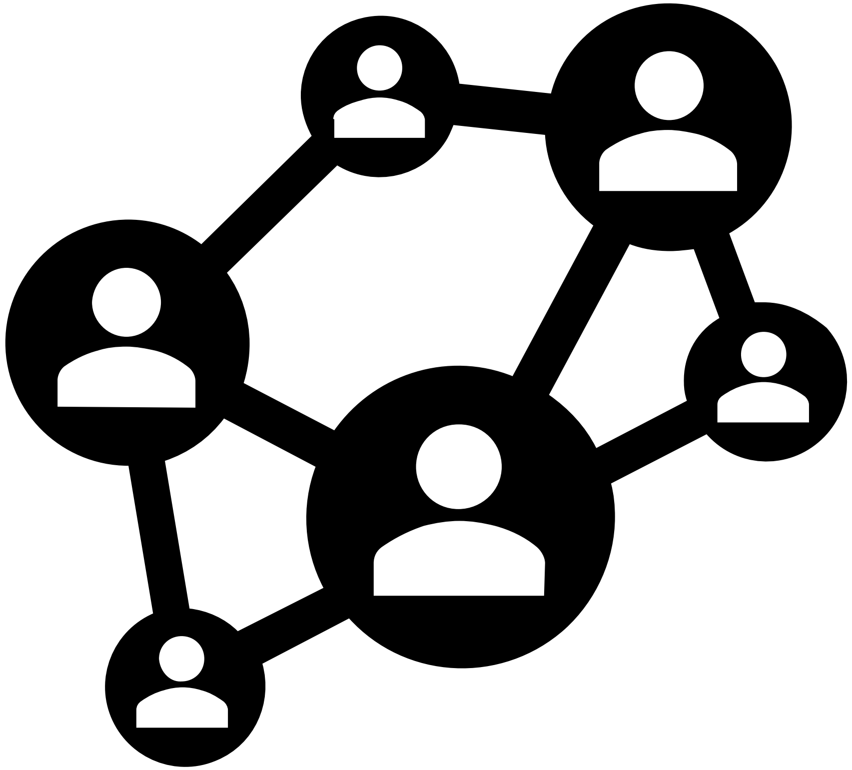
GRAPH DATA MODELLING

CSMODEL T3 AY 2024 - 2025

Arren Matthew C. Antioquia
arren.antioquia@dlsu.edu.ph
Department of Software Technology
De La Salle University, Manila, Philippines

GRAPH DATA

- Graphs are a general language for describing systems of interacting entities.
- A graph is a collection of objects where some pairs of objects are connected by links.



GRAPH DATA

Networks (Natural Graphs):

- Communication systems link electronic devices
- Interaction between genes/proteins regulate life
- Thoughts are connected through neurons in our brain.

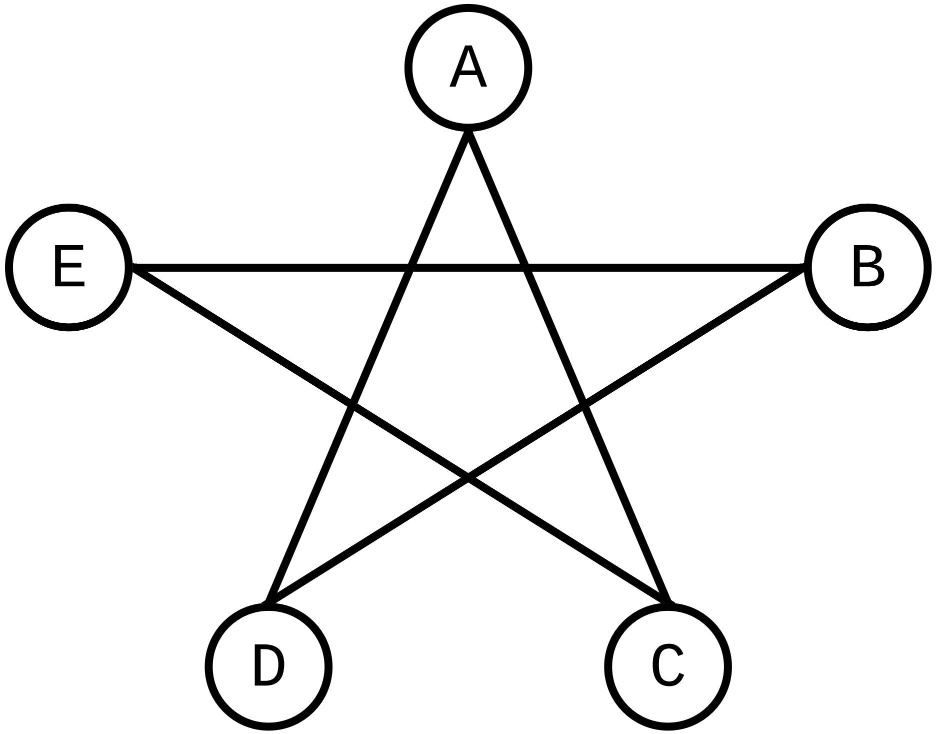
Information Graphs:

- Information/knowledge are organized and linked
- Scene graphs: how objects in a scene relate
- Similarity networks: take data, connect similar points.

GRAPH DATA

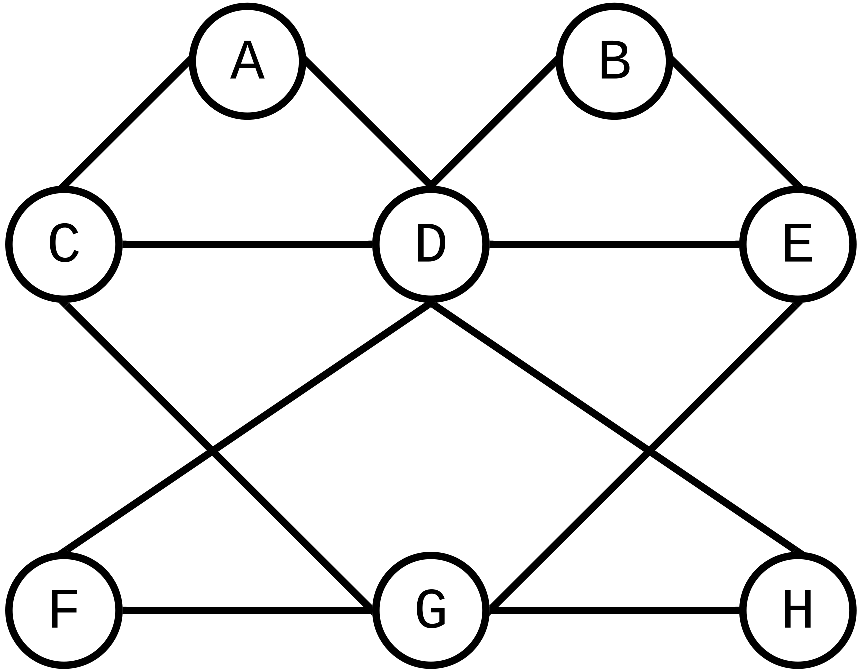
Components of a Graph

- **Objects:** nodes, vertices
- **Interactions:** links, edges
- **System:** network, graph

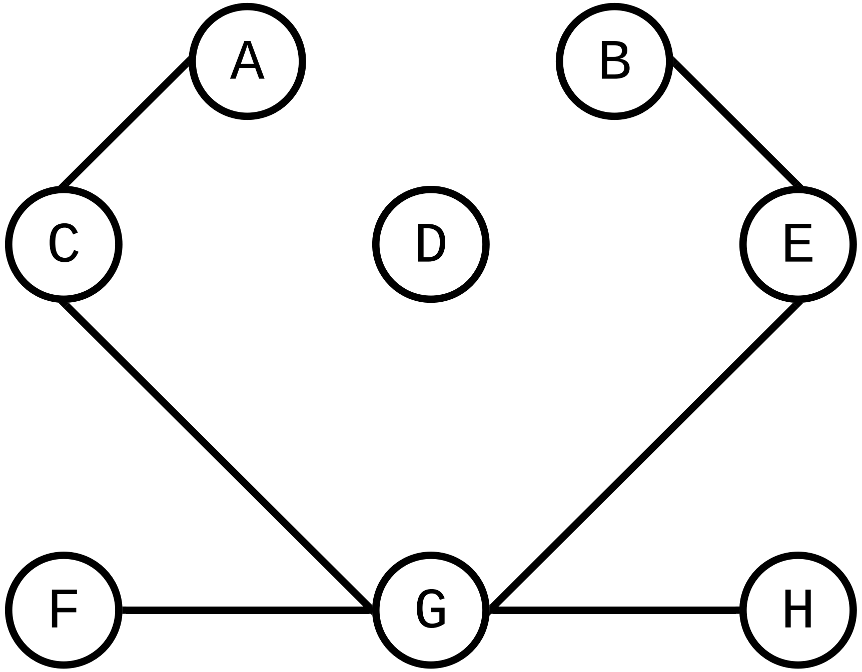


GRAPH DATA

A graph is **connected** if every two vertex has a path between them. G_1 is a connected graph, while G_2 is not.



$Graph\ G_1$



$Graph\ G_2$

GRAPH DATA

Network often refers to real systems

- Web, social network, metabolic network
- Jargon: Network, node, link

Graph is a mathematical representation of a network

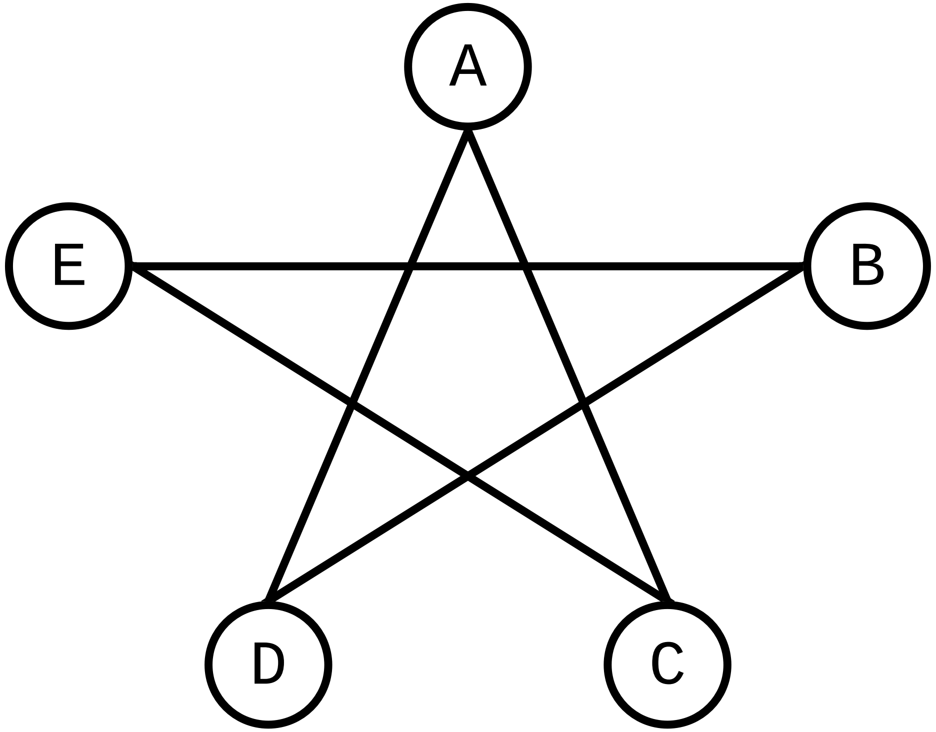
- Web graph, social graph, knowledge graph
- Jargon: Graph, vertex, edge

UNDIRECTED GRAPHS

UNDIRECTED GRAPHS

Undirected graphs are composed of edges which do not have any specific direction.

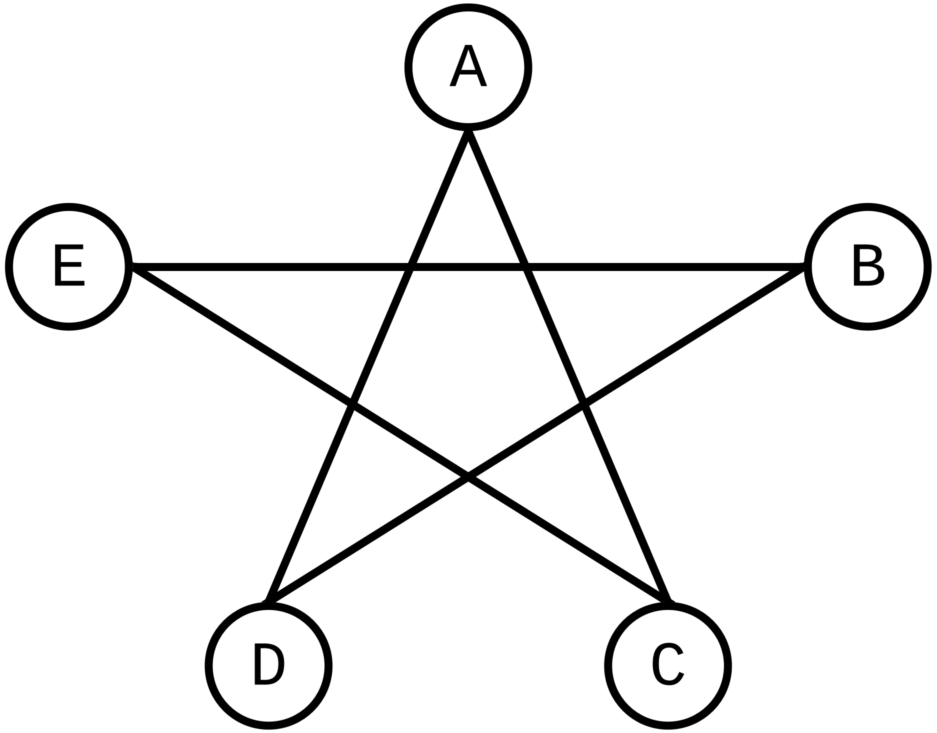
These edges, instead, represents a two-way relationship between the vertices.



$Graph\ G_1$

UNDIRECTED GRAPHS

An undirected graph G can be represented as $G = (V, E)$, where V is the set of vertices and E is the set of edges



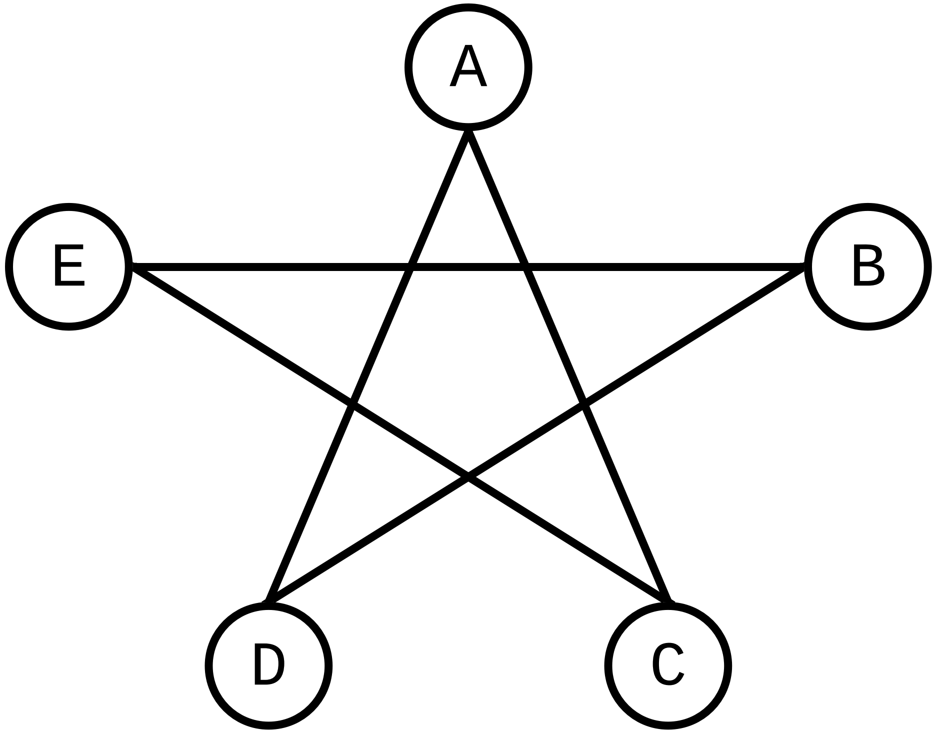
$Graph\ G_1$

$$G_1 = (V_1, E_1)$$

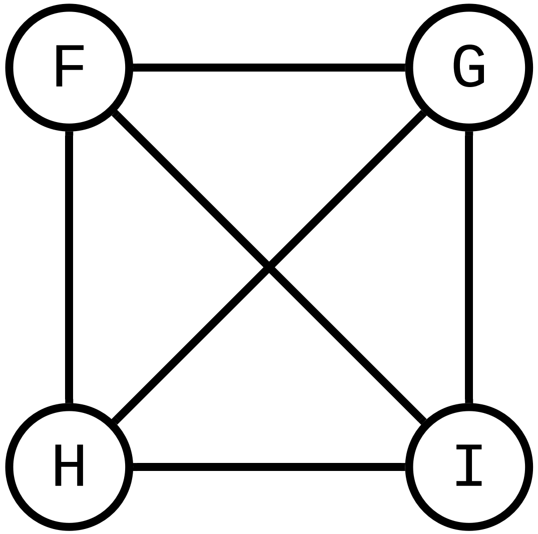
$$V_1 = \{A, B, C, D, E\}$$
$$E_1 = \{(A, C), (A, D), (B, D), (B, E), (C, E)\}$$

UNDIRECTED GRAPHS

Degree refers to the number of edges at a vertex.



Graph G_1



Graph G_2

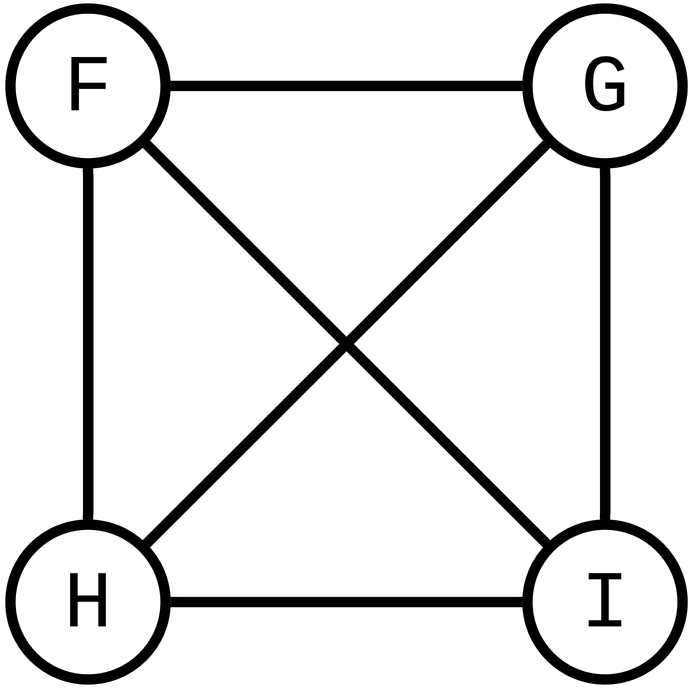
In G_1, all vertices have a degree 2.

In G_2, all vertices have a degree 3.

UNDIRECTED GRAPHS

The maximum number of edges in any n-vertex undirected graph is $\frac{n(n-1)}{2}$.

An n-vertex undirected graph with exactly $\frac{n(n-1)}{2}$ edges is a **complete undirected graph**. Graph G_1 is a complete undirected graph



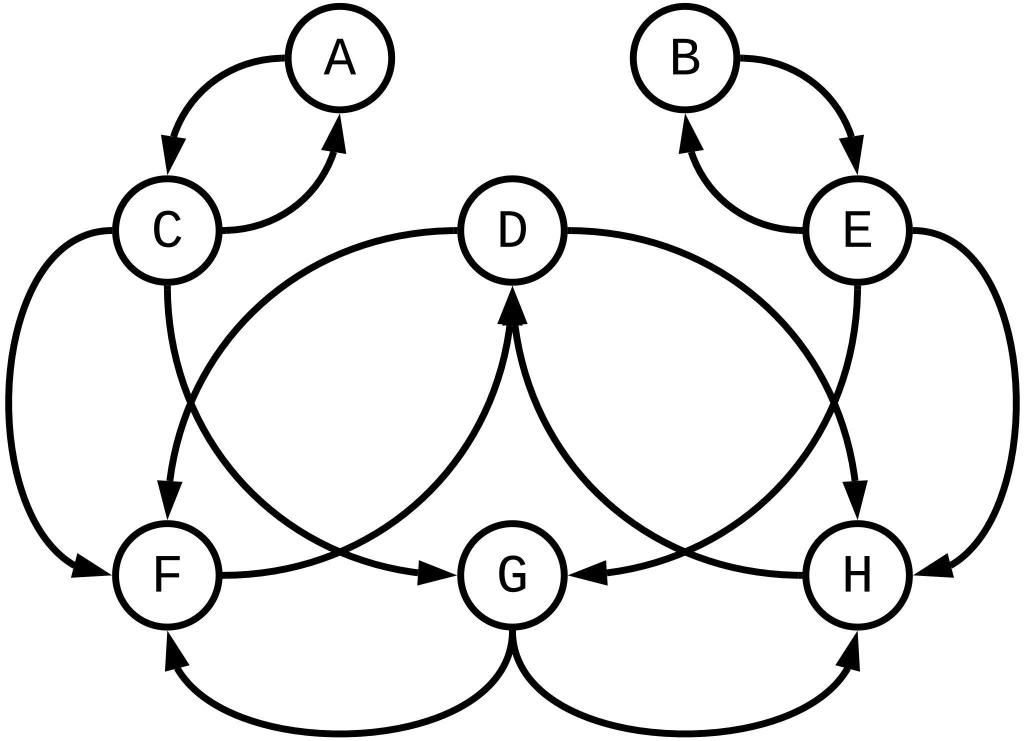
$Graph\ G_1$

DIRECTED GRAPHS

DIRECTED GRAPHS

Directed graphs are composed of edges which have specified direction.

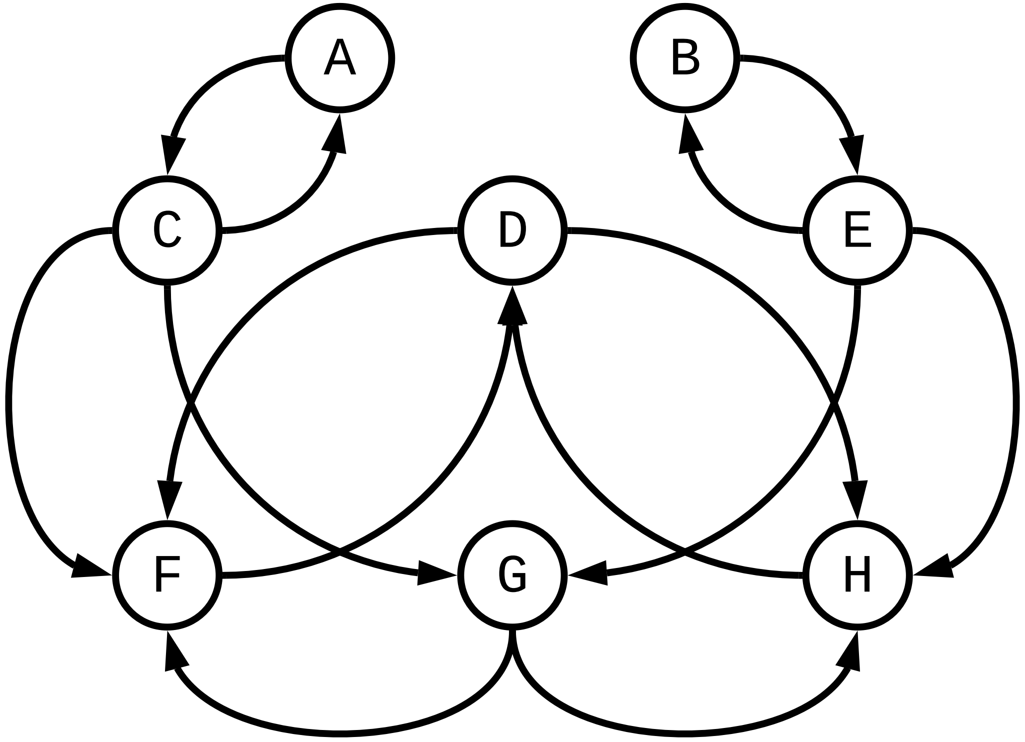
Thus, at most 2 edges of different direction might be used to connect 2 different vertices.



$Graph\ G_1$

DIRECTED GRAPHS

The directed graph G can be represented as $G = (V, E)$, where V is the set of vertices and E is the set of edges

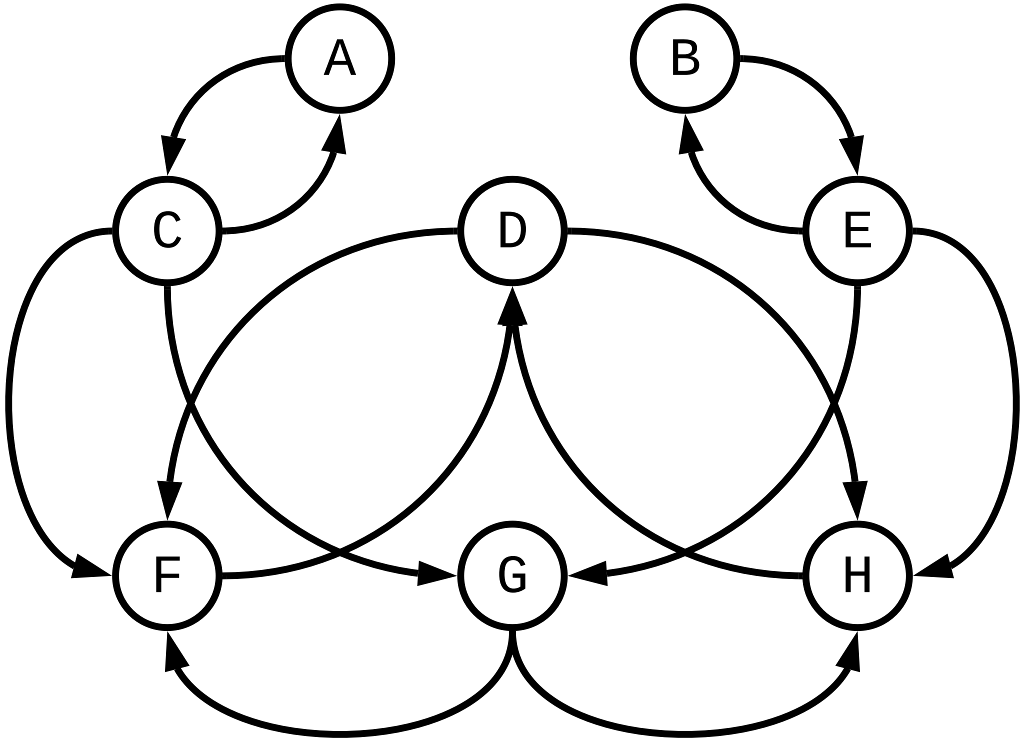


$Graph\ G_1$

$G_1 = (V_1, E_1)$
$V_1 = \{A, B, C, D, E, F, G, H\}$
$E_1 = \{<A, C>, <B, E>, <C, A>, <C, F>, <C, G>, <D, F>, <D, H>, <E, B>, <E, G>, <E, H>, <F, D>, <G, F>, <G, H>, <H, D>\}$

DIRECTED GRAPHS

Out-degree – number of arrows originating from a vertex



$Graph\ G_1$

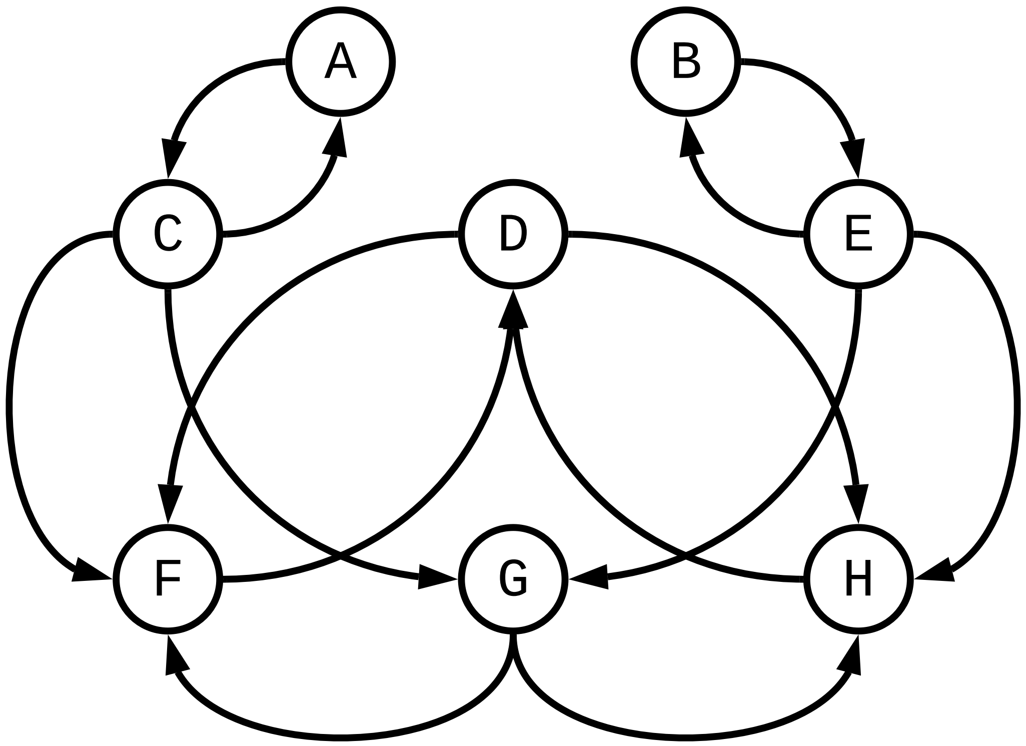
Out-degree of vertex A is 1.

Out-degree of vertex D is 2.

Out-degree of vertex C is 3.

DIRECTED GRAPHS

In-degree – number of arrows pointing to a vertex



$Graph\ G_1$

In-degree of vertex B is 1.

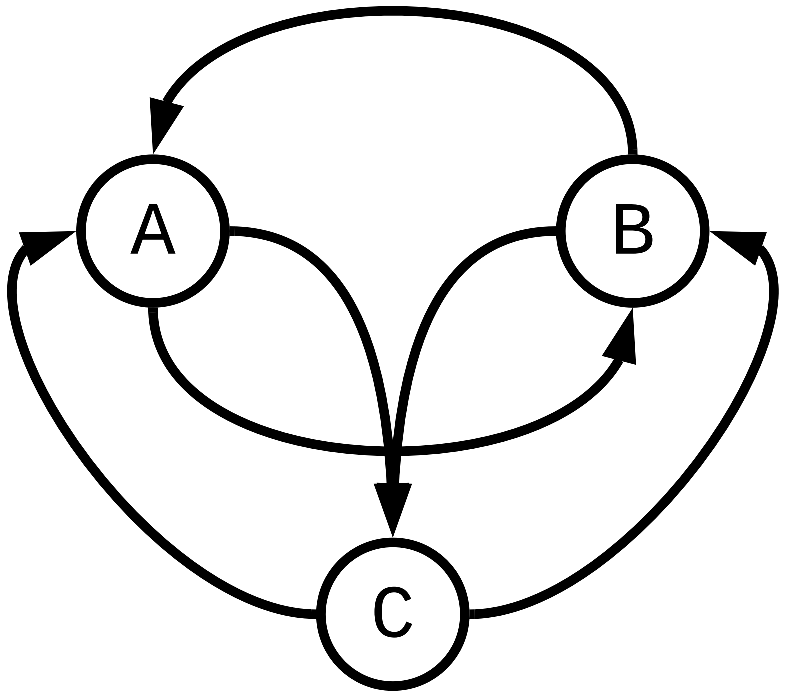
In-degree of vertex G is 2.

In-degree of vertex F is 3.

DIRECTED GRAPHS

The maximum number of edges in any n-vertex directed graph is $n(n-1)$.

An n-vertex directed graph with exactly $n(n-1)$ edges is a **complete directed graph**. Graph G_1 is a complete directed graph.

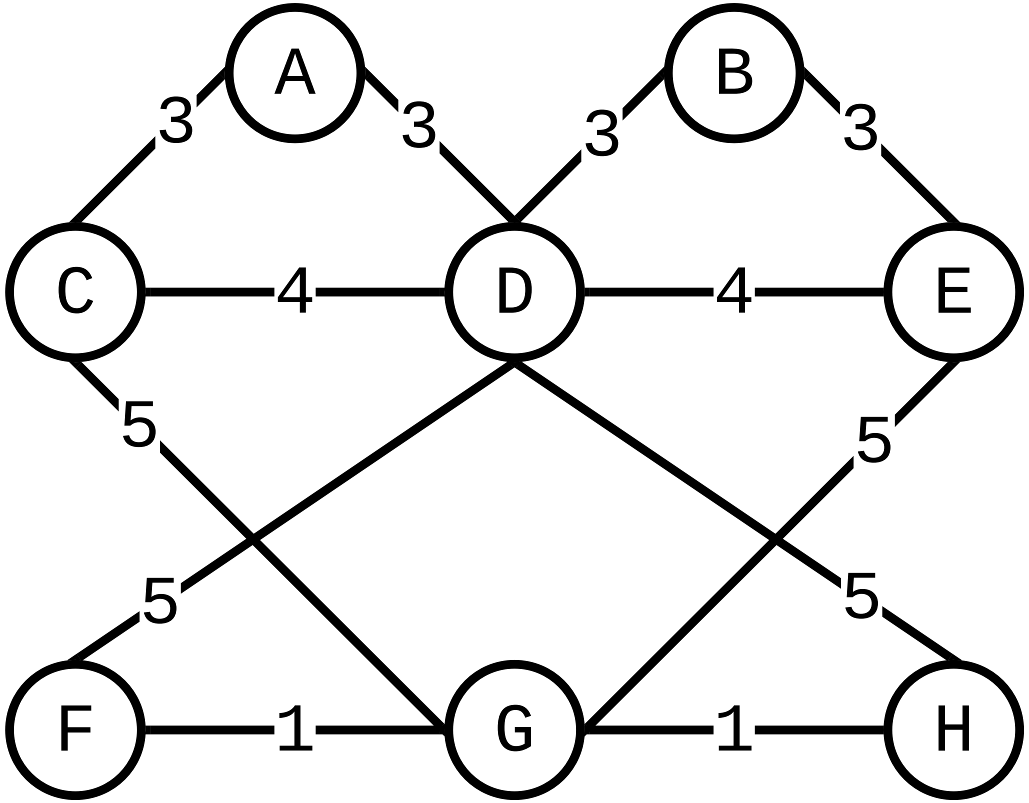


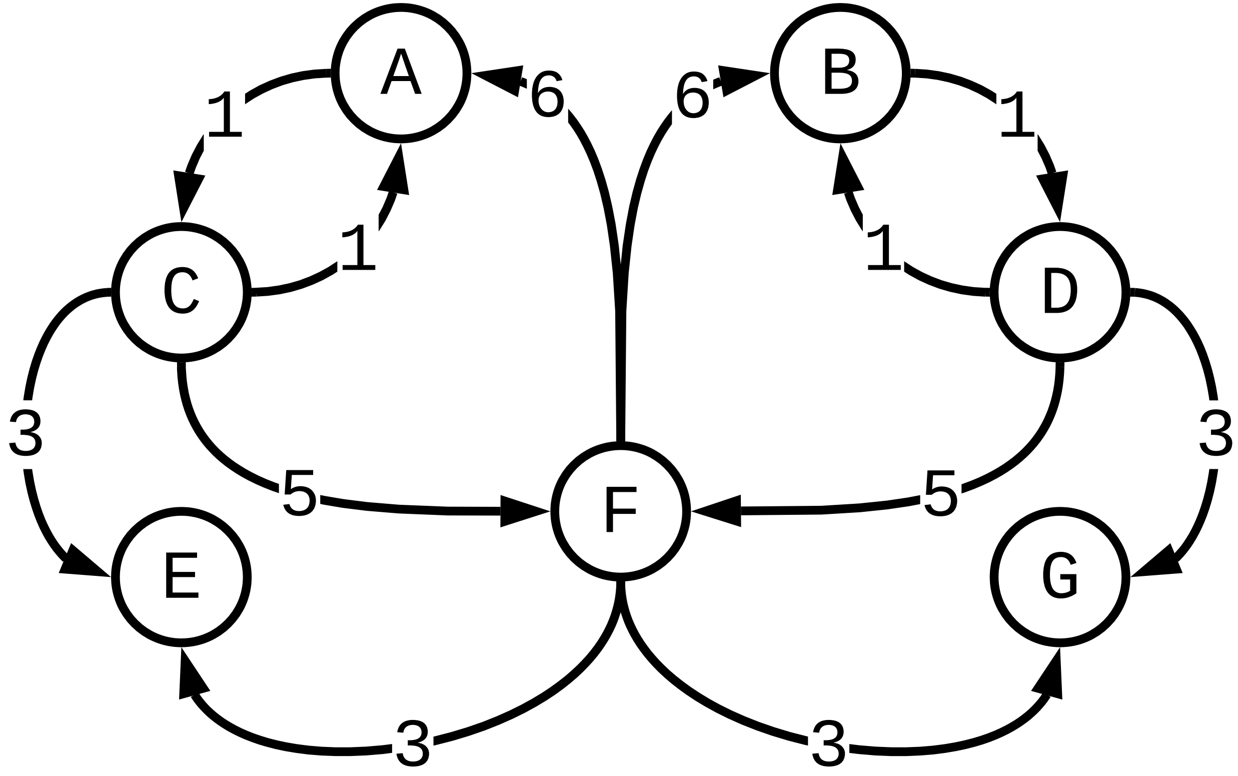
$Graph\ G_1$

WEIGHTED GRAPHS

WEIGHTED GRAPHS

Graphs for which each edge has an associated weight, typically given by a weighted function $w: E \rightarrow R$





GRAPH REPRESENTATIONS

GRAPH REPRESENTATIONS

Collection of Adjacency Lists

Adjacency list representation is usually preferred, since it provides a compact way to represent sparse graphs (i.e., $|E| < |V|^2$).

Adjacency Matrix

Adjacency matrix representation is preferred if the graph is dense (i.e., $|E|$ is close to $|V|^2$).

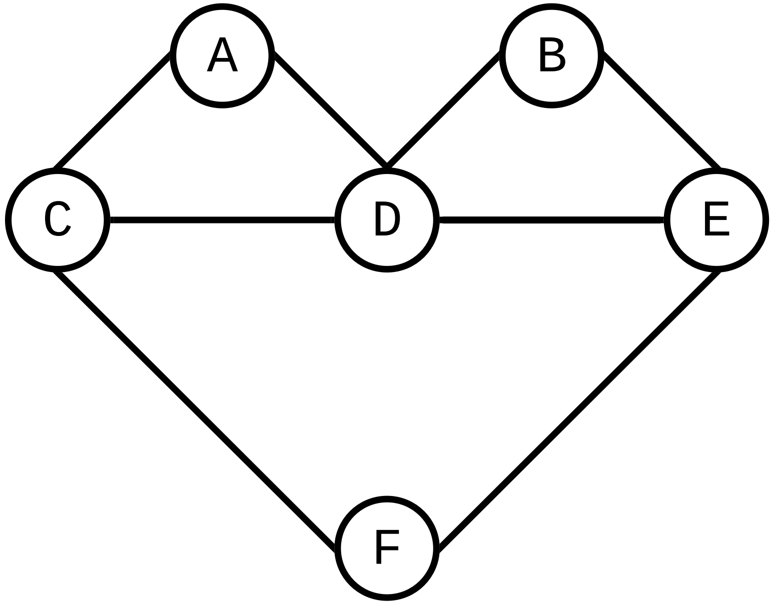
GRAPH REPRESENTATIONS

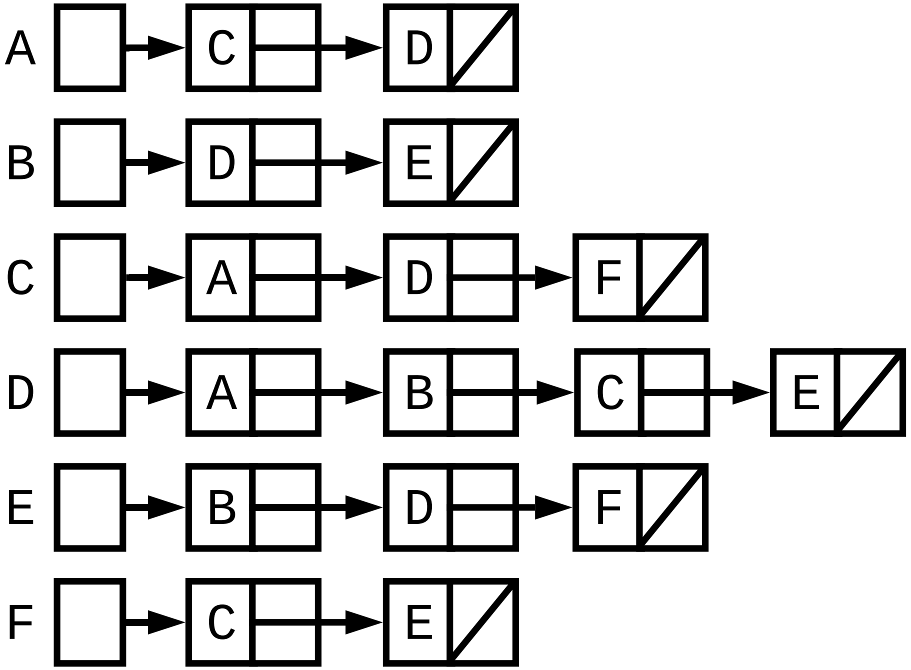
Adjacency List

- The adjacency list representation of graph $G = (V, E)$ consists of an array A with $|V|$ number of lists, one for each vertex in V.
- For each vertex $u \in V$, the adjacency list $A[u]$ contains all vertices v such that there is an edge $(u, v) \in E$.
- The adjacency list representation's memory requirement is $O(V + E)$.

GRAPH REPRESENTATIONS

Adjacency List

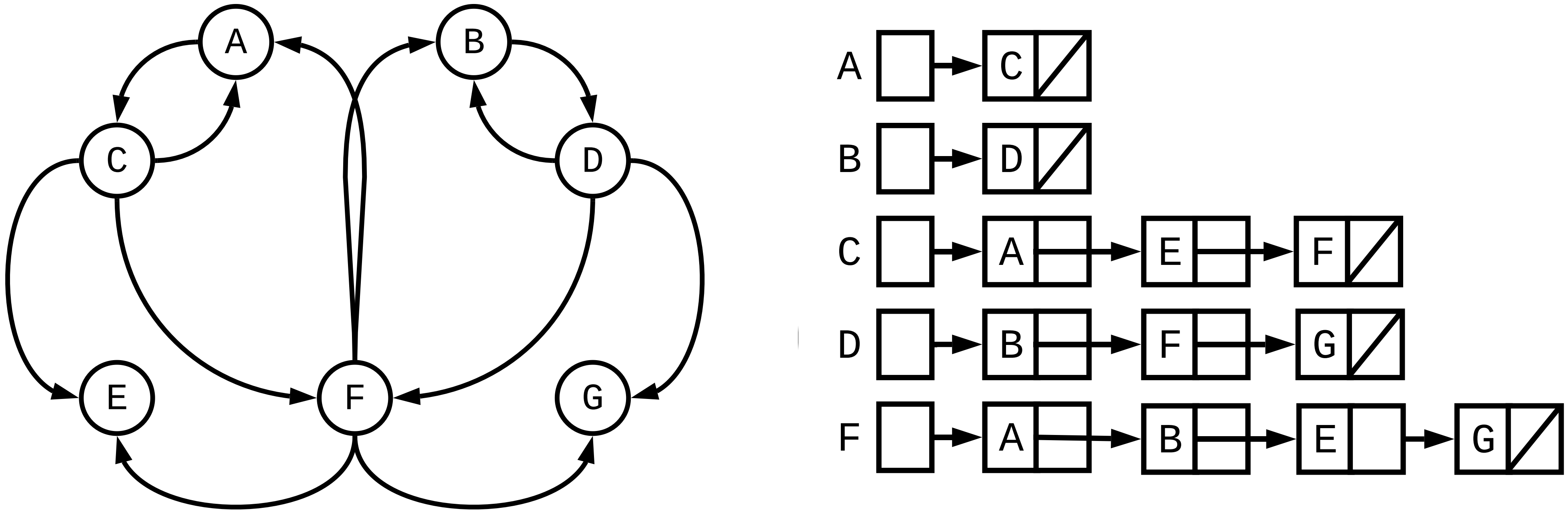




In an **undirected graph**, the sum of the lengths of all the adjacency lists is $2|E|$.

GRAPH REPRESENTATIONS

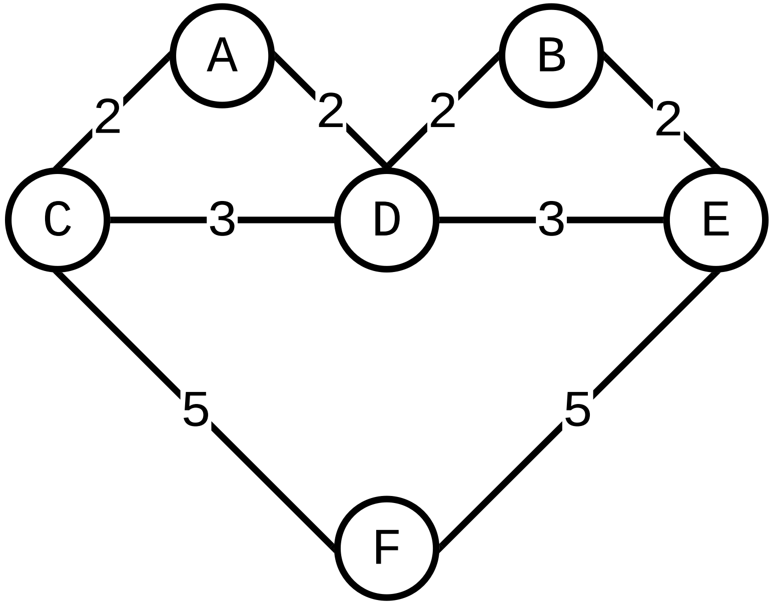
Adjacency List

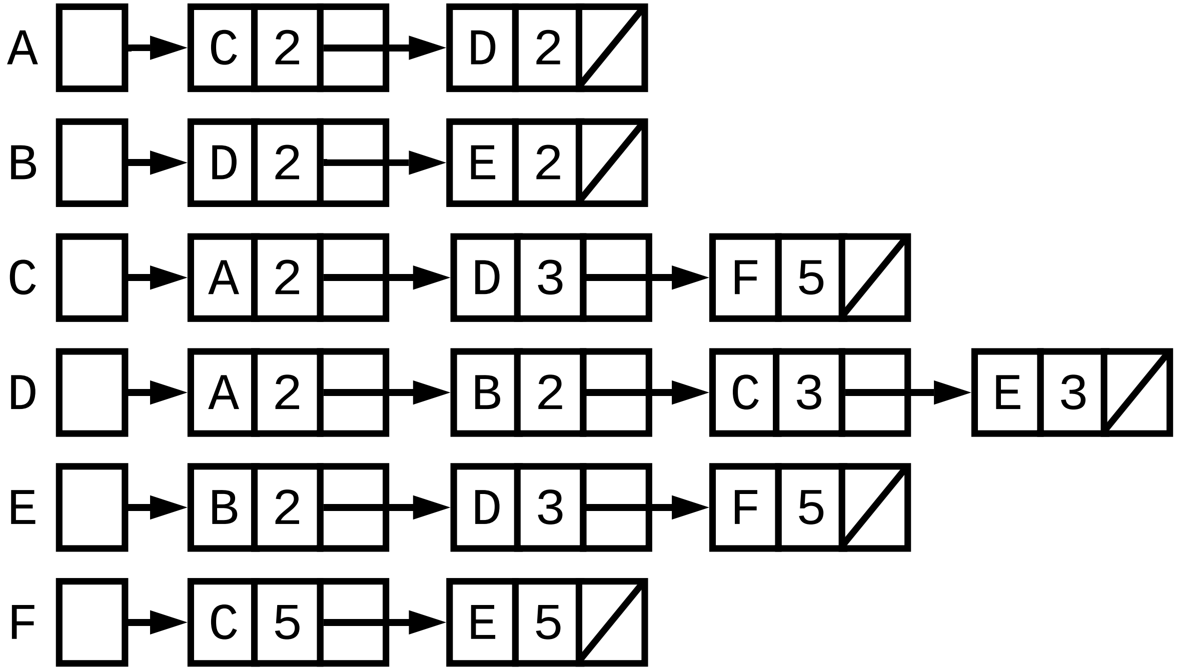


In a **directed graph**, the sum of the lengths of all the adjacency lists is $|E|$.

GRAPH REPRESENTATIONS

Adjacency List

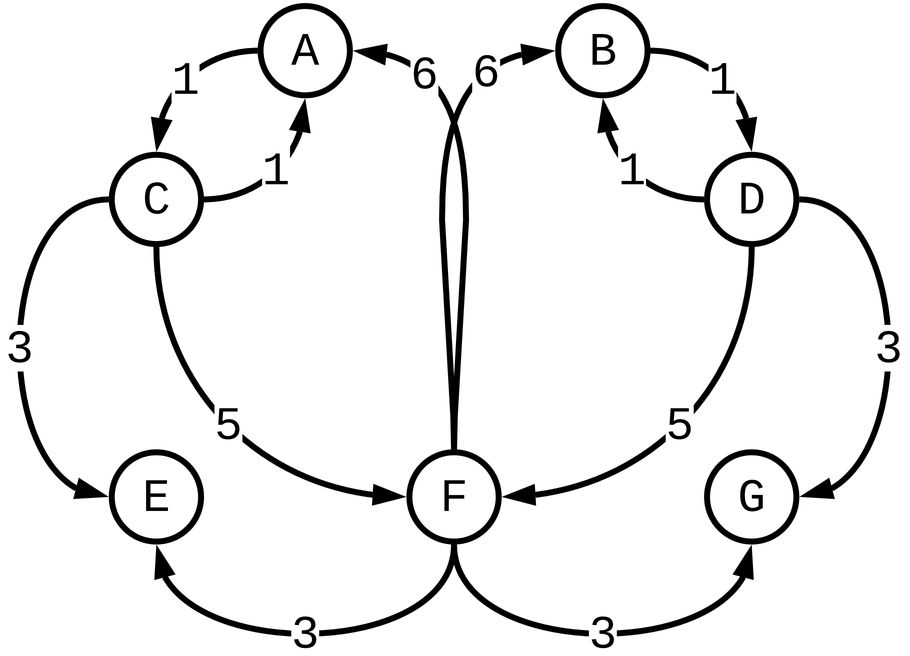


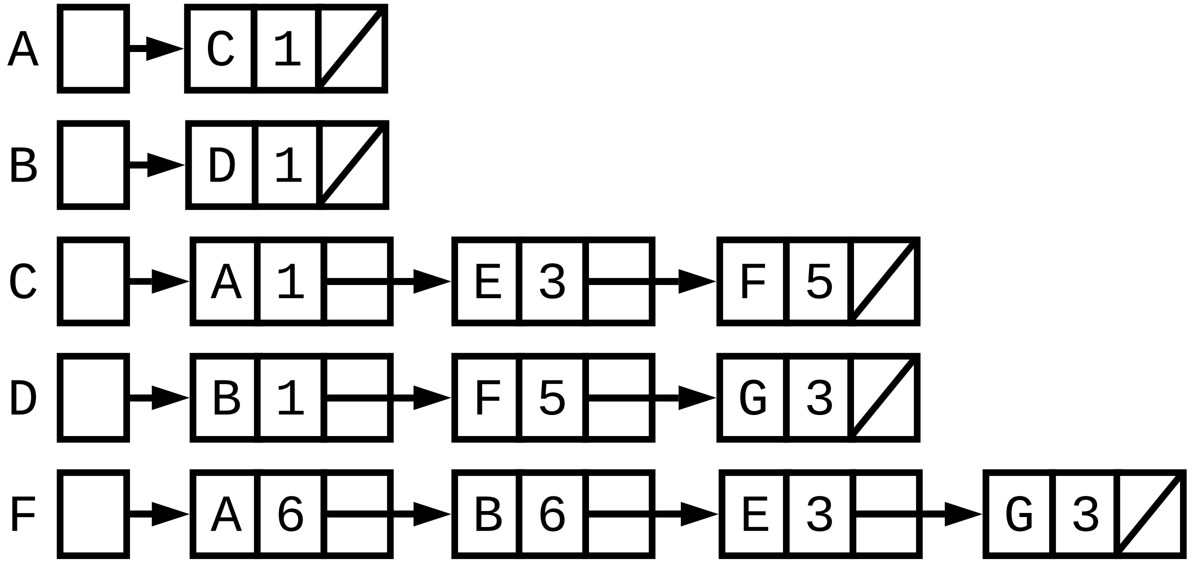


The weight $w(u, v)$ of the edge $(u, v) \in E$ is stored with vertex v in u's adjacency list and vice versa.

GRAPH REPRESENTATIONS

Adjacency List





The weight $w(u, v)$ of the edge $(u, v) \in E$ is stored with vertex v in u's adjacency list.

GRAPH REPRESENTATIONS

Adjacency Matrix

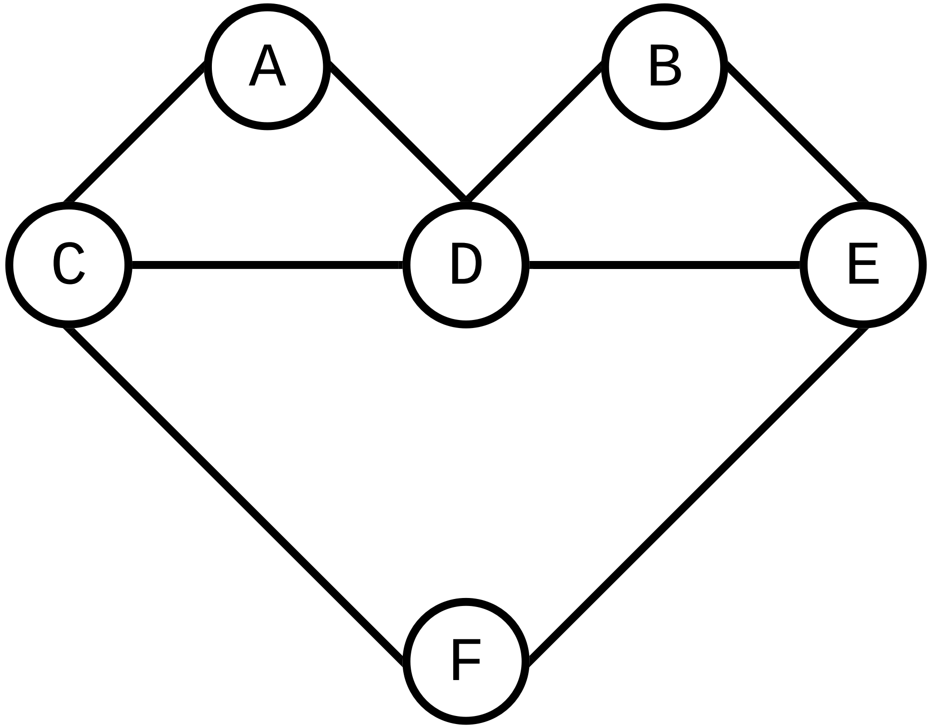
The adjacency matrix representation of graph $G = (V, E)$ consists of a $|V| \times |V|$ matrix $A = (a_{ij})$ such that:

$$a_{ij} = \begin{cases} 1 & if\ (i, j)\ \epsilon\ E \\ 0 & otherwise \end{cases}$$

An adjacency matrix representation of a graph requires $O(|V|^2)$ memory, independent of the edges in the graph.

GRAPH REPRESENTATIONS

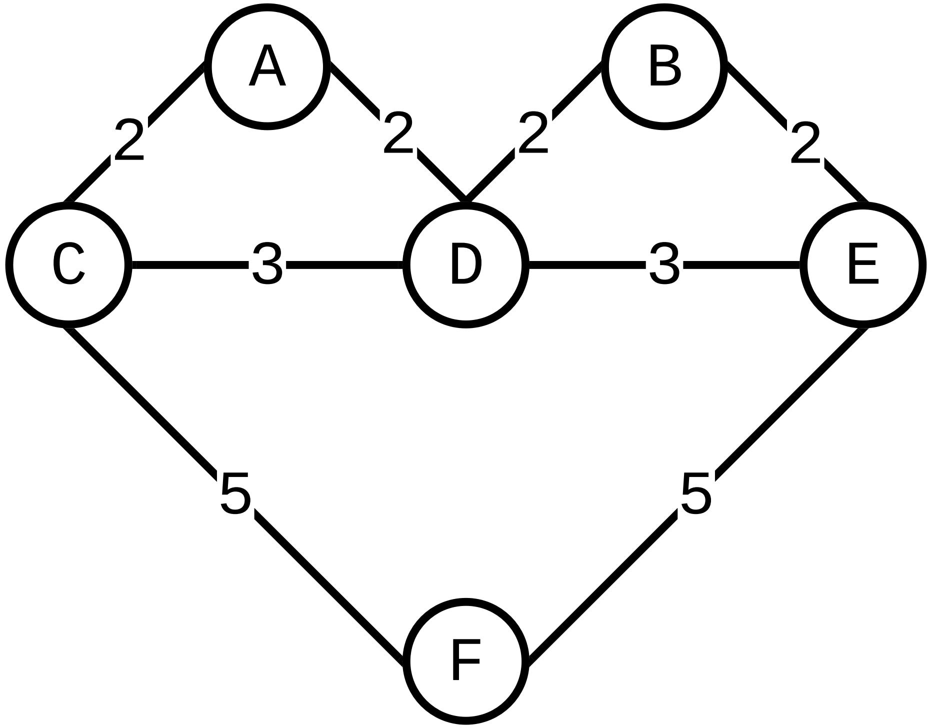
Adjacency Matrix



	A	B	C	D	E	F
A	0	0	1	1	0	0
B	0	0	0	1	1	0
C	1	0	0	1	0	1
D	1	1	1	0	1	0
E	0	1	0	1	0	1
F	0	0	1	0	1	0

GRAPH REPRESENTATIONS

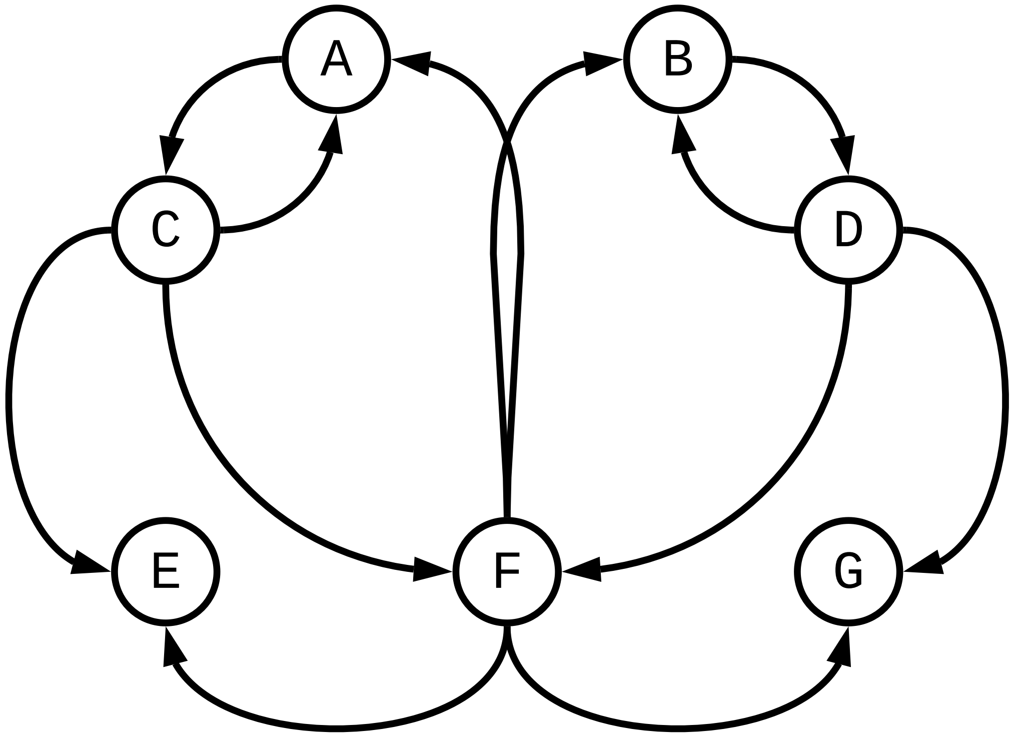
Adjacency Matrix



	A	B	C	D	E	F
A	0	0	2	2	0	0
B	0	0	0	2	2	0
C	2	0	0	3	0	5
D	2	2	3	0	3	0
E	0	2	0	3	0	5
F	0	0	5	0	5	0

GRAPH REPRESENTATIONS

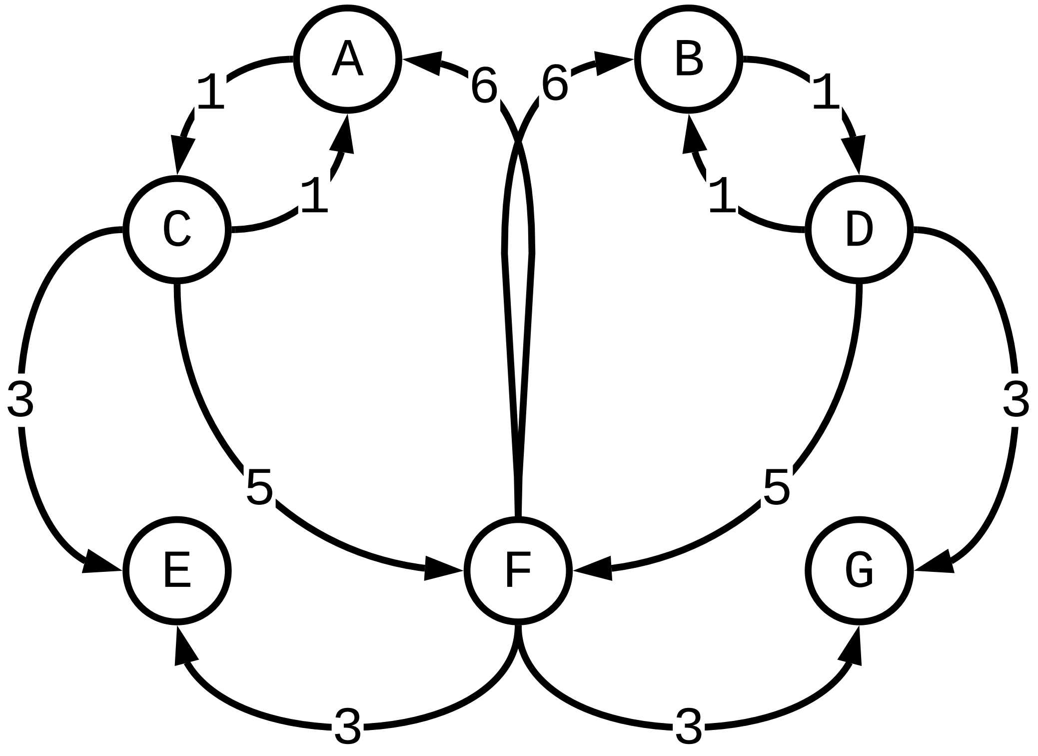
Adjacency Matrix



	A	B	C	D	E	F	G
A	0	0	1	0	0	0	0
B	0	0	0	1	0	0	0
C	1	0	0	0	1	1	0
D	0	1	0	0	0	1	1
E	0	0	0	0	0	0	0
F	1	1	0	0	1	0	1
G	0	0	0	0	0	0	0

GRAPH REPRESENTATIONS

Adjacency Matrix



	A	B	C	D	E	F	G
A	0	0	1	0	0	0	0
B	0	0	0	1	0	0	0
C	1	0	0	0	3	5	0
D	0	1	0	0	0	5	3
E	0	0	0	0	0	0	0
F	6	6	0	0	3	0	3
G	0	0	0	0	0	0	0

GRAPH EMBEDDINGS

GRAPH EMBEDDINGS

- **Graph embedding** transforms graphs to a lower dimensional representation of the graph, while preserving its topology.
- Its goal is to turn graphs into a format that machine learning algorithms can understand and process.
- Machine learning algorithms are tuned for continuous data; thus, we need to convert graphs, which are discrete by nature, in a continuous vector space.

GRAPH EMBEDDINGS

Recent algorithms used to produce graph embeddings:

- DeepWalk (Perozzi et al., 2014)
- Node2Vec (Grover & Leskovec, 2016)

GRAPH EMBEDDINGS

DeepWalk (Perozzi et al., 2014)

Deepwalk belongs to the family of graph embedding techniques that uses **walks**.

GRAPH EMBEDDINGS

DeepWalk (Perozzi et al., 2014)

Graphs are like texts.

	the	dog	is	cute	cat	also	red	but	blue	not
“cat”	0	0	0	0	1	0	0	0	0	0

	A	B	C	D	E	F	G	H	I	J
A	0	0	0	1	0	1	0	0	0	0

GRAPH EMBEDDINGS

DeepWalk (Perozzi et al., 2014)

Language modeling estimates the likelihood of a specific sequence of words appearing in a corpus.

Suppose we have a sequence of words:

$$W_1^n = (w_0, w_1, \ldots, w_n)$$

We want to maximize:

$$\Pr(w_n | w_0, w_1, \ldots, w_{n-1})$$

GRAPH EMBEDDINGS

DeepWalk (Perozzi et al., 2014)

DeepWalk generalizes language modeling to explore the graph through a stream of short random walks.

Suppose we have a sequence of visited vertices:

$$V_1^n = (v_0, v_1, \dots, v_n)$$

We want to estimate the likelihood of:

$$\Pr(v_n | v_0, v_1, \dots, v_{n-1})$$

GRAPH EMBEDDINGS

DeepWalk (Perozzi et al., 2014)

The goal is to **learn a latent representation**, not only a probability distribution of node co-occurrences. Thus, they introduced the mapping function:

$$\Phi\colon v \in V \to \mathbb{R}^{|V| \times d}$$

which represents the latent social representation associated with each vertex v in the graph.

GRAPH EMBEDDINGS

DeepWalk (Perozzi et al., 2014)

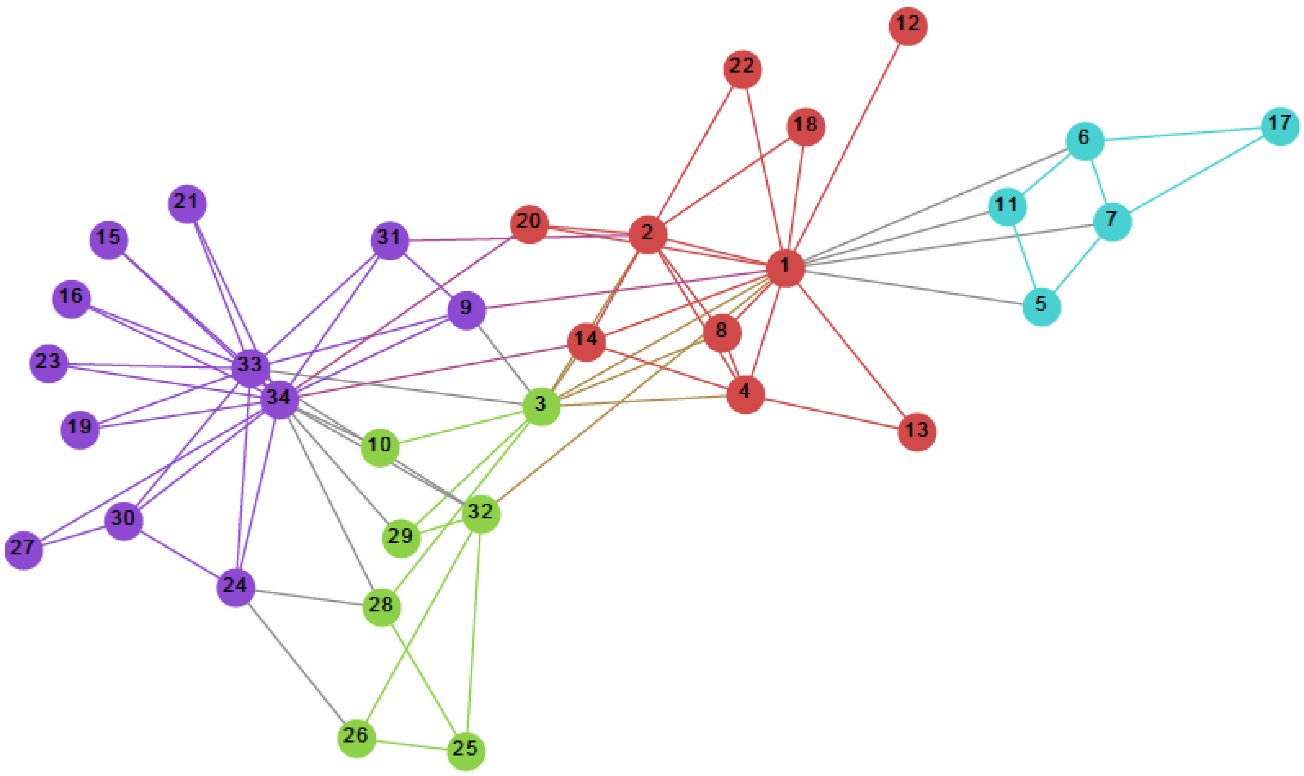
Thus, DeepWalk estimates the likelihood:

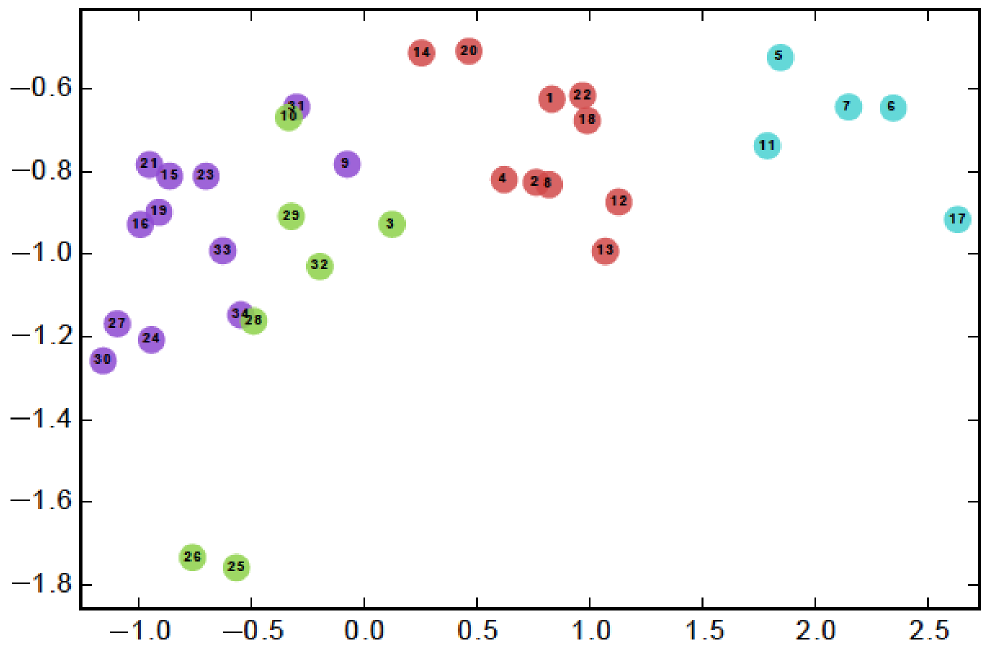
$$\Pr(\Phi(v_n)|\Phi(v_0), \Phi(v_1), \ldots, \Phi(v_{n-1}))$$

The goal is to estimate the likelihood of observing node v_n given all the previous nodes visited so far in the random walk.

GRAPH EMBEDDINGS

DeepWalk (Perozzi et al., 2014)





GRAPH EMBEDDINGS

Node2Vec (Grover & Leskovec, 2016)

- Node2vec is one of the first Deep Learning attempts to learn embedding from graph data.
- Node2Vec, like DeepWalk, utilizes walks to learn graph embeddings.
- Compared to DeepWalk, Node2vec **incorporates a search bias** variable α, parameterized by p and q, which allows it to interpolate between BFS and DFS.

GRAPH EMBEDDINGS

Node2Vec (Grover & Leskovec, 2016)

Formally, given a source code u, simulate a random walk of fixed length l. Let c_i denote the ith node in the walk, starting with $c_0 = u$. Nodes c_i are generated by the following distribution:

$$P(c_i = x \mid c_{i-1} = v) = \begin{cases} \frac{\pi_{vx}}{Z}, & if\ (v, x) \in E \\ 0, & otherwise \end{cases}$$

where $P(x, v)$ is the transition probability between v and x

GRAPH EMBEDDINGS

Node2Vec (Grover & Leskovec, 2016)

Suppose the walk has just traversed the edge (t, v) and now resides at node v. The walk needs to decide on the next step to evaluate the transition probability π_{vx} on edges (v, x) leading to v.

$$\pi_{vx} = \alpha_{pq}(t, x) \times w_{vx}$$

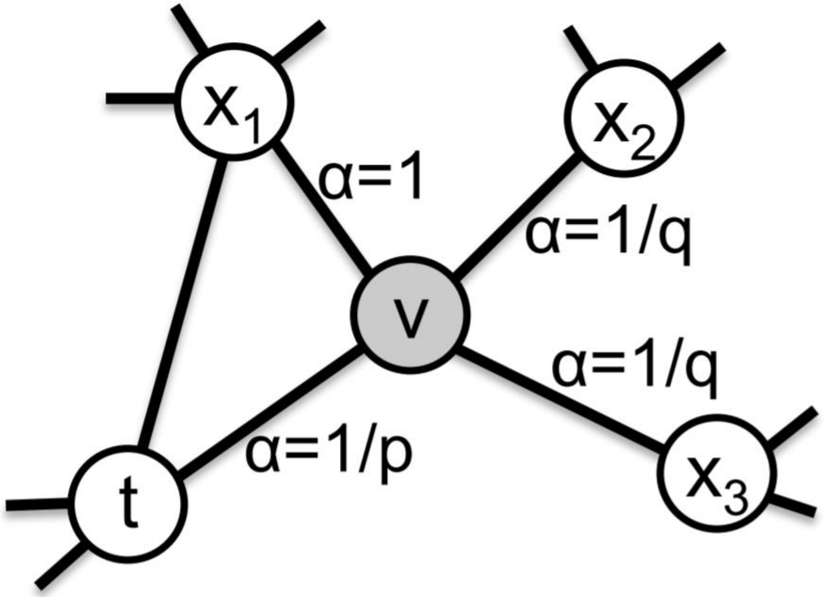
where w_{vx} is the weight of the edge going from v to x

GRAPH EMBEDDINGS

Node2Vec (Grover & Leskovec, 2016)

Search Bias

$$\alpha_{pq}(t,x) = \begin{cases} 1/p & if\ d_{tx} = 0 \\ 1 & if\ d_{tx} = 1 \\ 1/q & if\ d_{tx} = 2 \end{cases}$$

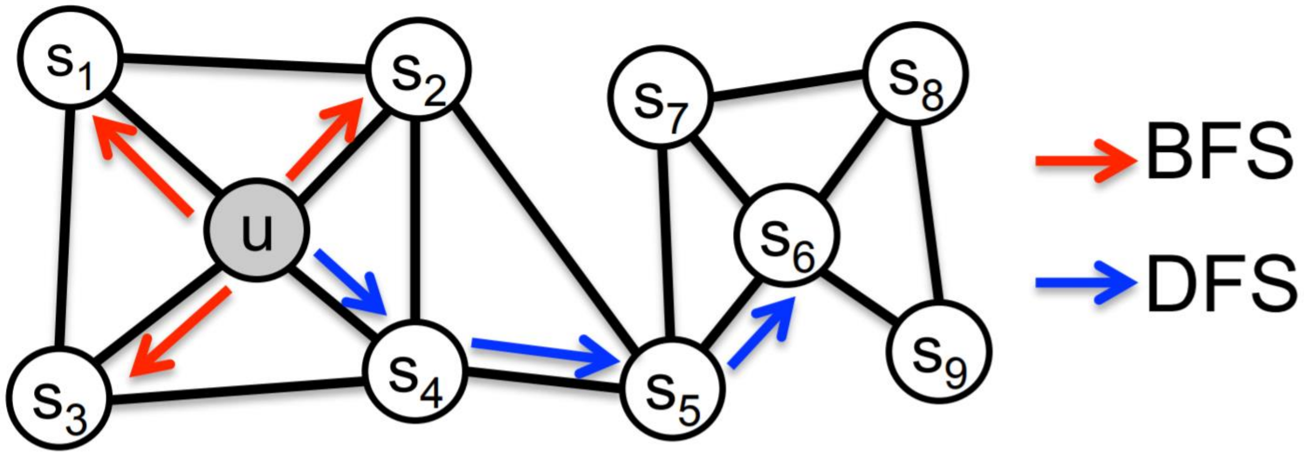


where d_{tx} denotes the shortest distance between t and x

GRAPH EMBEDDINGS

Node2Vec (Grover & Leskovec, 2016)

BFS is ideal for learning local neighbors.

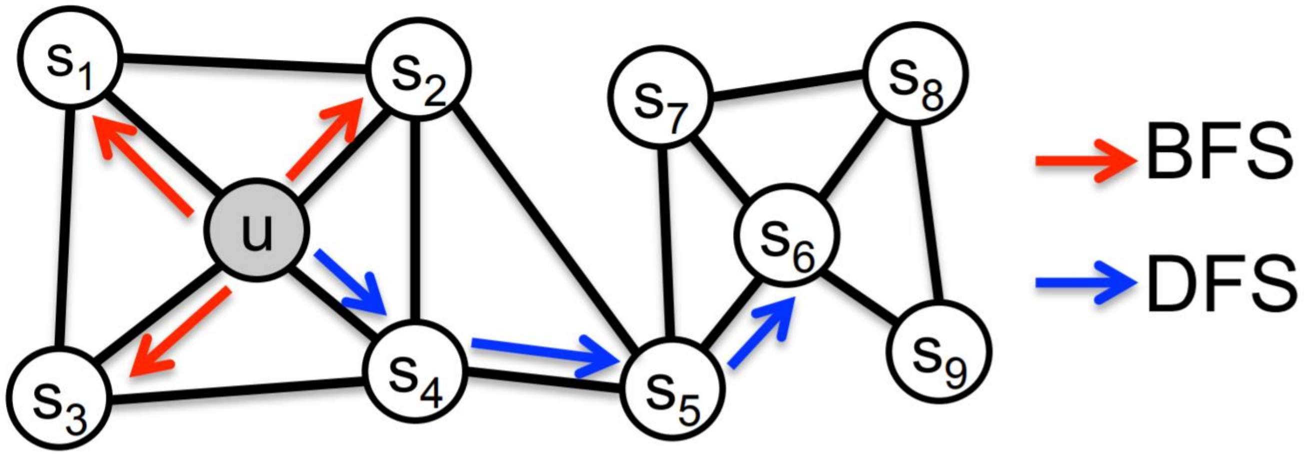


The neighborhood N_s is restricted to nodes which are immediate neighbors of the source.

GRAPH EMBEDDINGS

Node2Vec (Grover & Leskovec, 2016)

DFS is better for learning global variables.

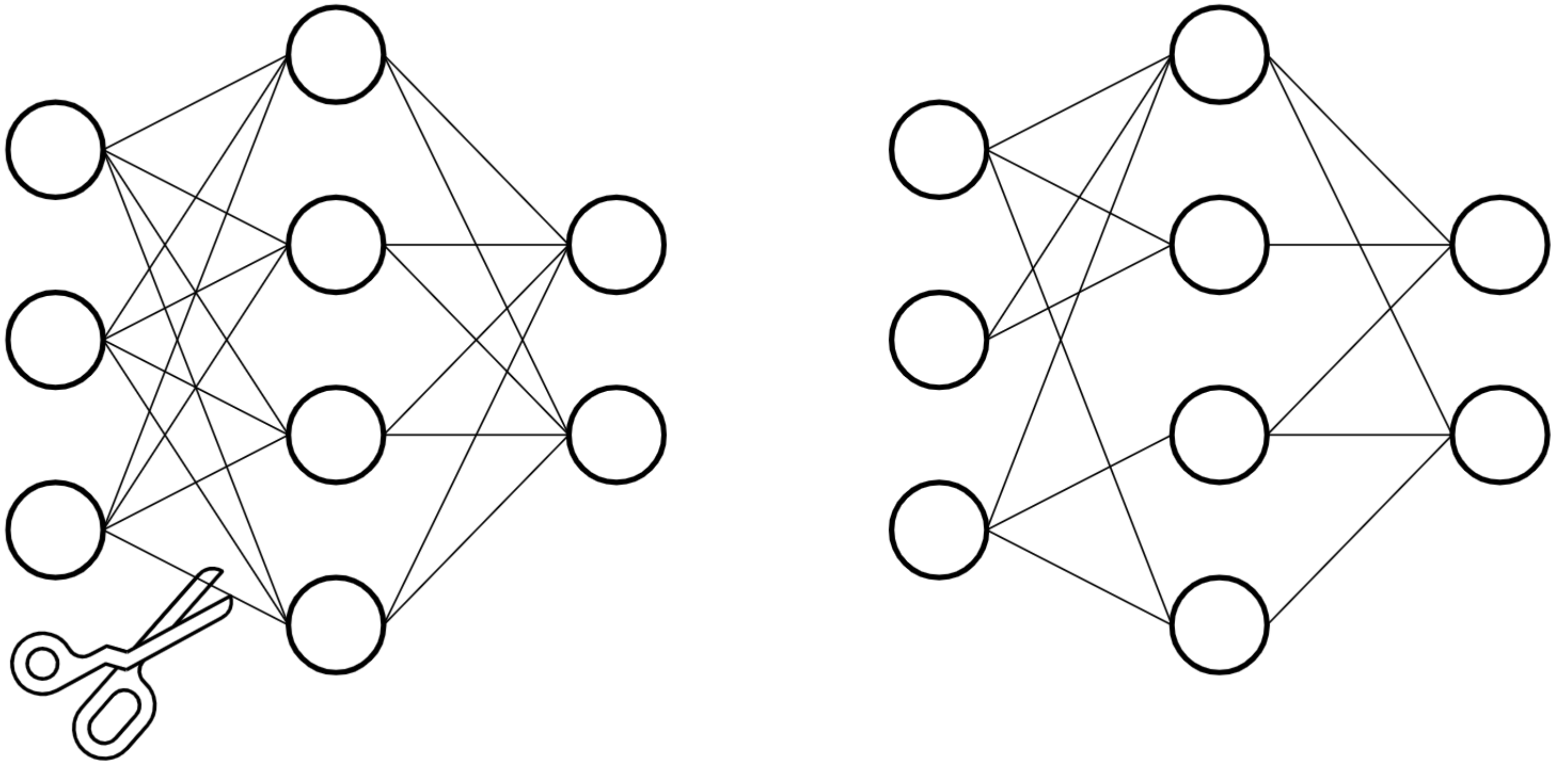


The neighborhood consists of nodes sequentially sampled at increasing distances from the source node.

APPLICATIONS

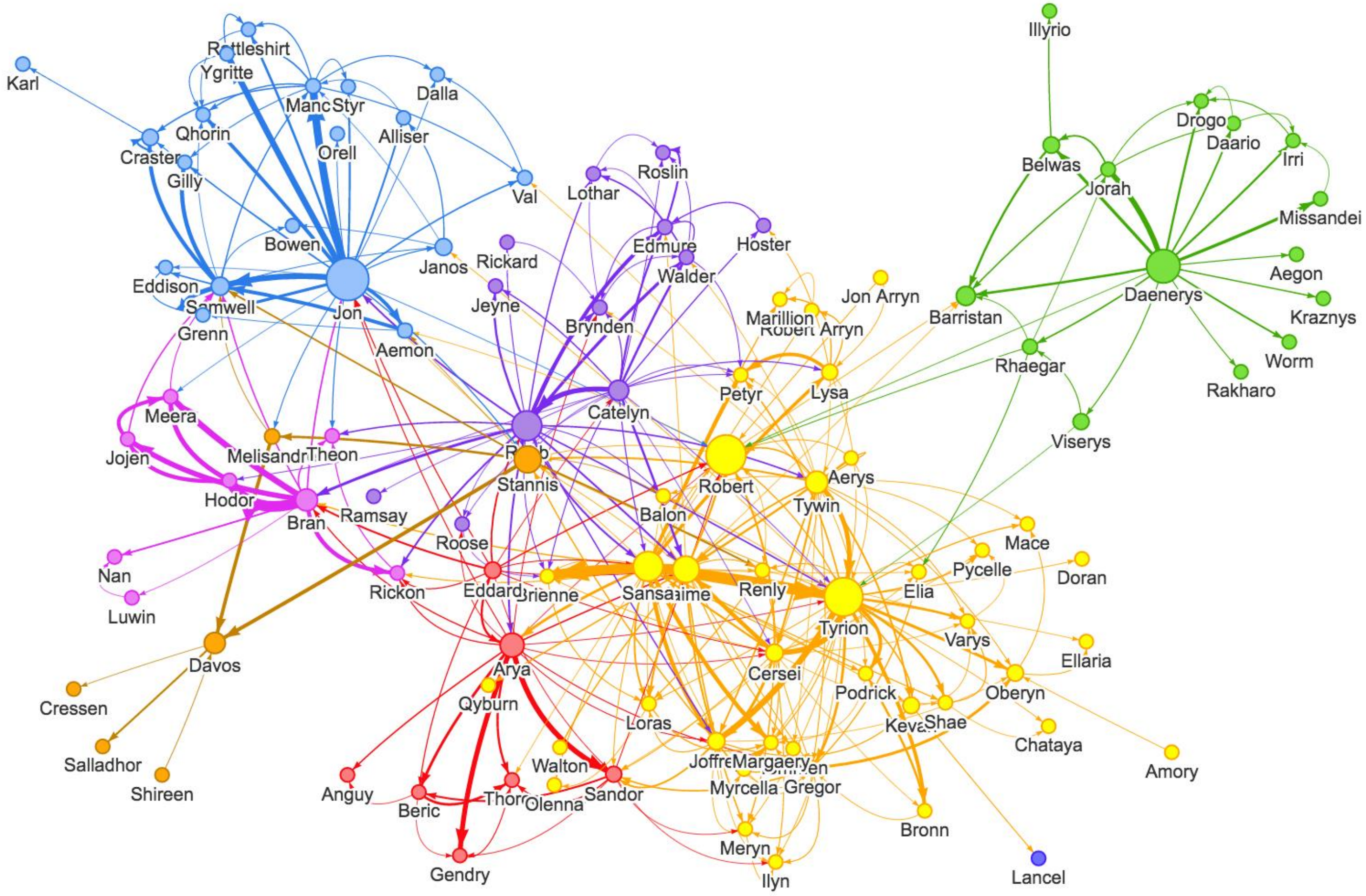
APPLICATIONS

Network Compression



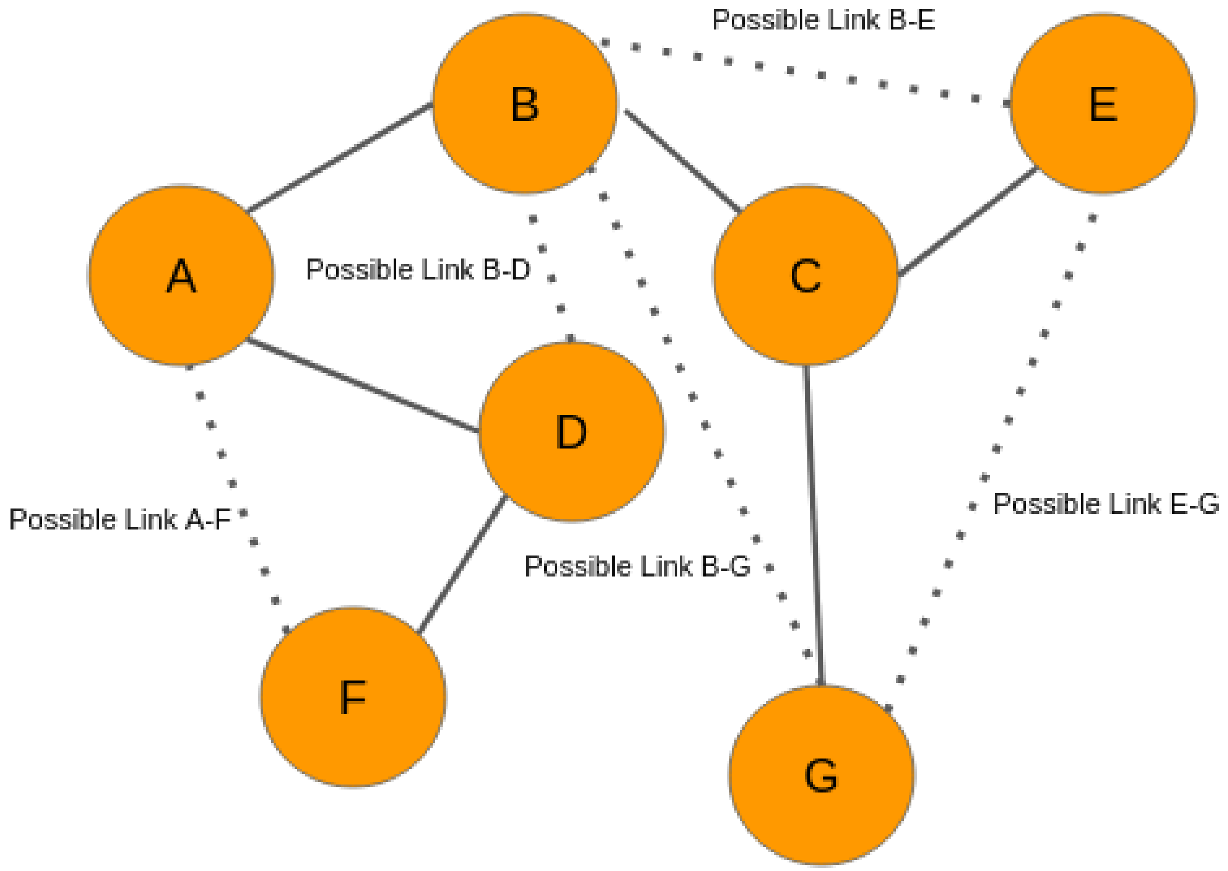
APPLICATIONS

Clustering



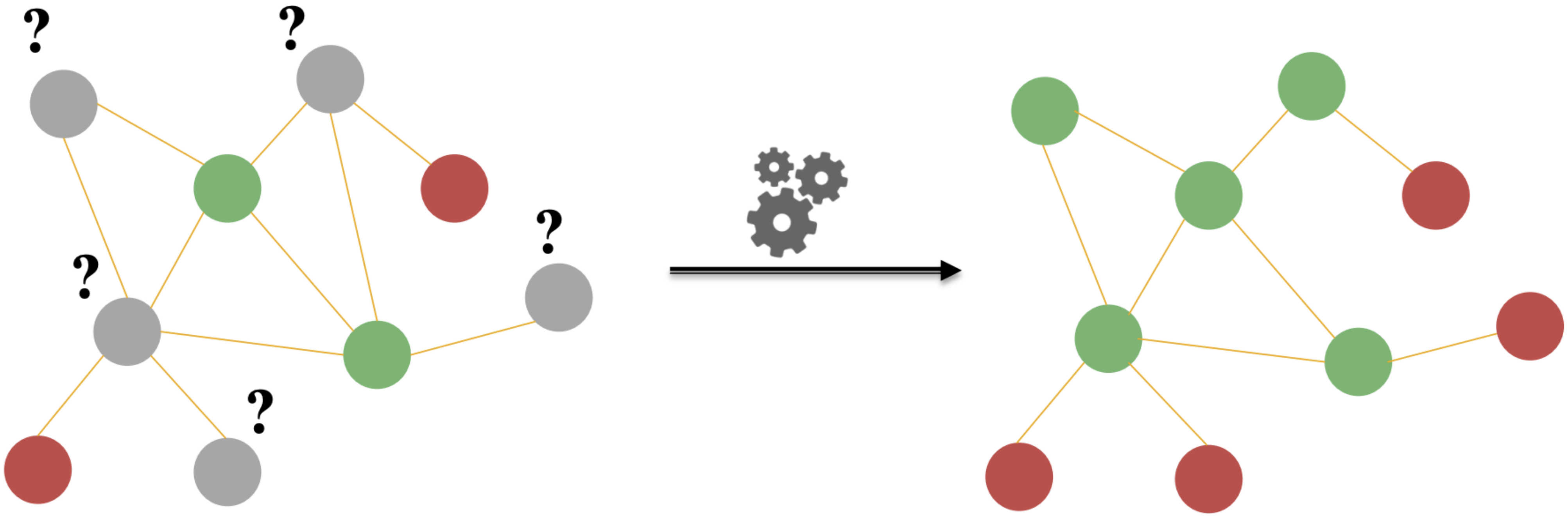
APPLICATIONS

Link Prediction



APPLICATIONS

Node Classification



GRAPH DATA MODELLING

CSMODEL T3 AY 2024 - 2025

Arren Matthew C. Antioquia
arren.antioquia@dlsu.edu.ph
Department of Software Technology
De La Salle University, Manila, Philippines