50.039 Theory and Practice of Deep Learning

W9-S2 Pre-Reading on Graph Theory

Matthieu De Mari

SINGAPORE UNIVERSITY OF TECHNOLOGY AND DESIGN

About this pre-reading

1. What are **graph objects**?
2. How do we **define** a graph object **mathematically**?
3. What are some **basic notions of graph theory** that could prove useful for Deep Learning?
4. What are **typical graph problems** and **graph datasets**?

Outline

In this lecture

- Introduction to graph theory
- Definitions for key concepts
- Typical problems in Graph Theory
- A few practical applications for graph theory (social networks, neural networks, transport networks, etc.)

In the next lectures

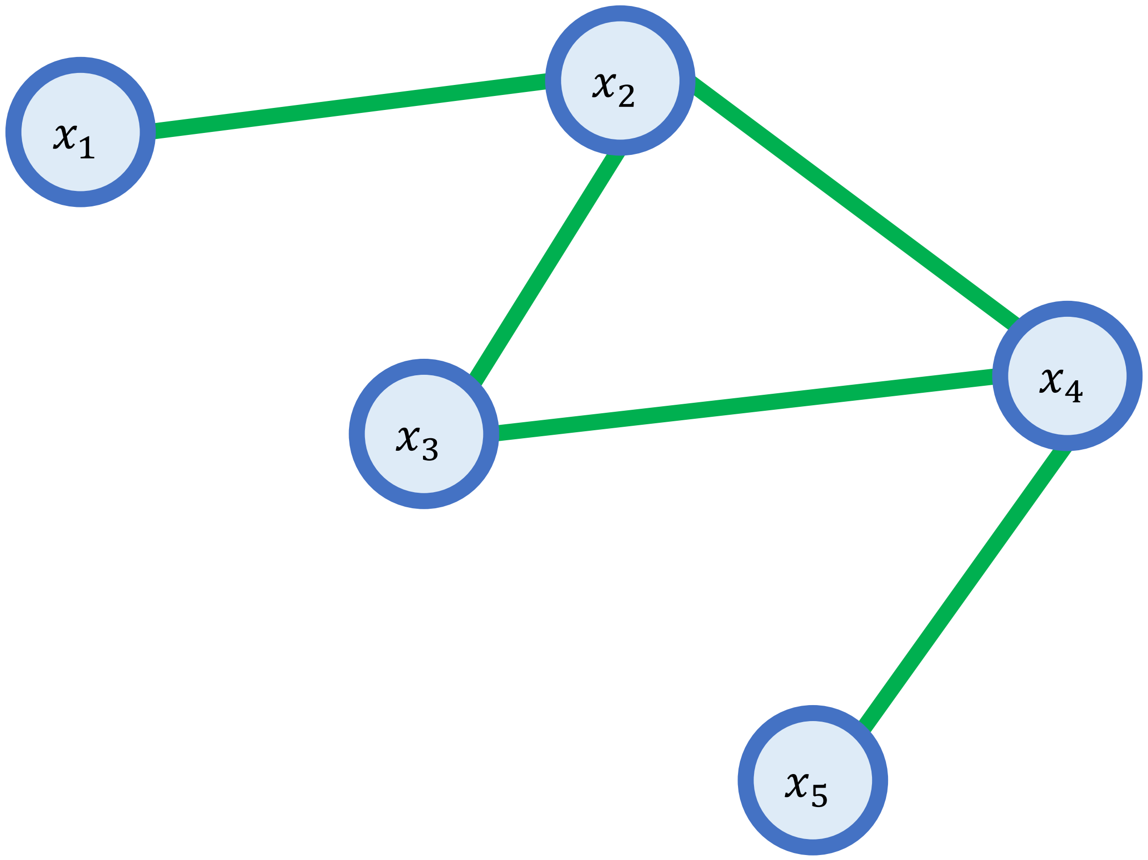
- Using graph types datasets
- Graph convolutions and graph embeddings
- Graph Convolutional Neural Networks
- Graph Convolutional Neural Networks with Attention Mechanisms
- Some more advanced embeddings

Graph theory

Definition (graph theory):

In mathematics, **graph theory** is the study of **graphs** objects, which are mathematical structures used to model pairwise relations between objects.

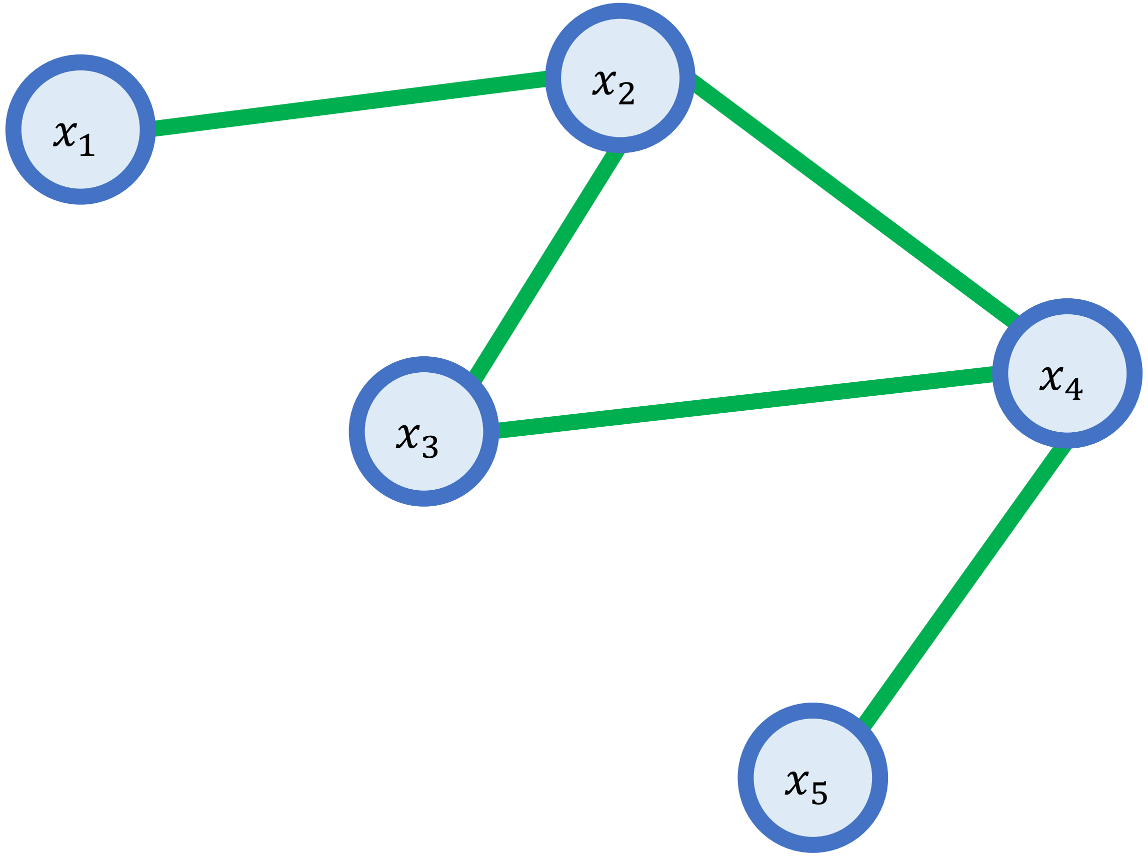
Graphs are one of the principal objects of study in **discrete mathematics**.



Graphs: a general and minimal definition

Definition (graph): A **graph** is a mathematical object, defined by an ordered pair $\boldsymbol{G} = (\boldsymbol{V}, \boldsymbol{E})$, with

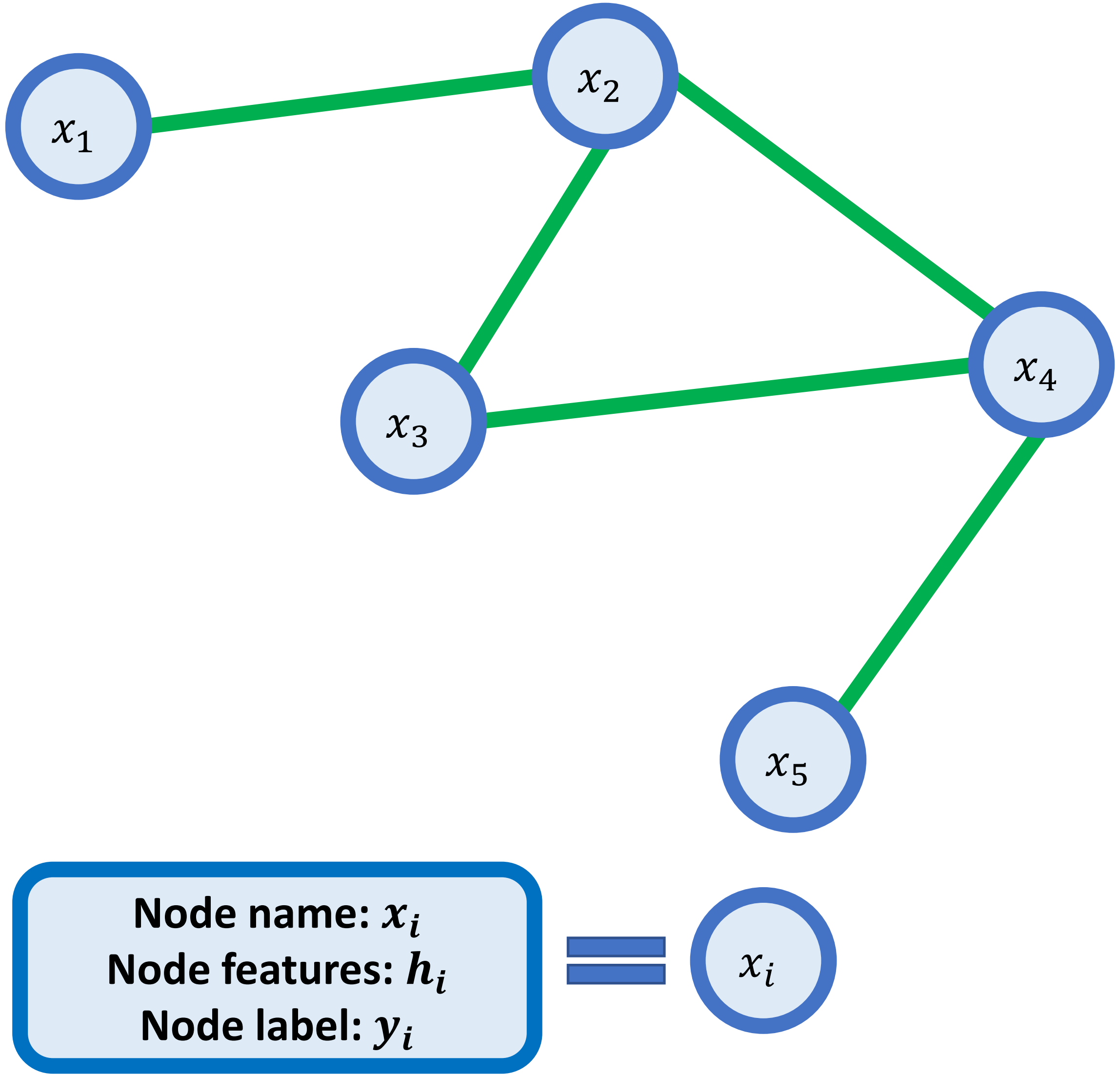
- $\boldsymbol{V} = \{\boldsymbol{x_1}, \boldsymbol{x_2}, \ldots, \boldsymbol{x_N}\}$ a set of N **vertices** (also called **nodes** or **points**),
- And $\boldsymbol{E}$ a set of **edges** (also called **links** or **lines**), defined as a subset of $\{(\boldsymbol{i}, \boldsymbol{j}) \mid \forall \boldsymbol{i} \in [\boldsymbol{1}, \boldsymbol{N}], \forall \boldsymbol{j} \in [\boldsymbol{1}, \boldsymbol{N}]\}$.



Nodes definition and attributes

Definition (nodes): A **node $\boldsymbol{x_i}$** is a point in the **graph**.

- A **node** has a **name $\boldsymbol{x_i}$**, which is used for indexing and differs from one node to another.
- A **node** may also have **attributes**, for instance:
    - Some **node features**, defined, for instance, as a vector $\boldsymbol{h_i} \in \mathbb{R}^{\boldsymbol{F}}$, with F elements,
    - Some **node label** y_i, defining a class for the node.

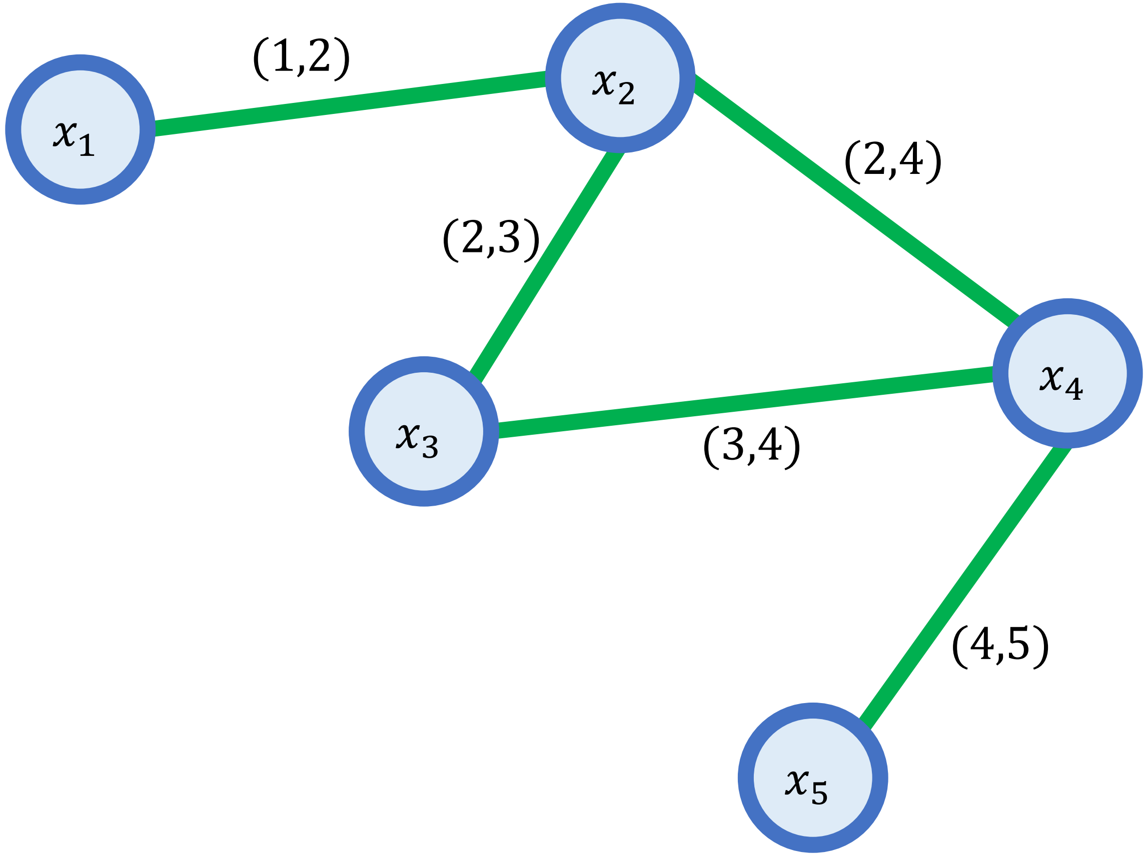


Edges definition

Definition (edges): An **edge $(\boldsymbol{i}, \boldsymbol{j})$** defines a connection from **node $\boldsymbol{x_i}$** to **node $\boldsymbol{x_j}$**.

- If **edge $(\boldsymbol{i}, \boldsymbol{j}) \in \boldsymbol{E}$**, then nodes x_i and x_j are connected in the **graph $\boldsymbol{G}$**.
- In our example, we have

$$\boldsymbol{E} = \{(1,2),(2,3),(2,4),(3,4),(4,5)\}$$



Undirected graph definition

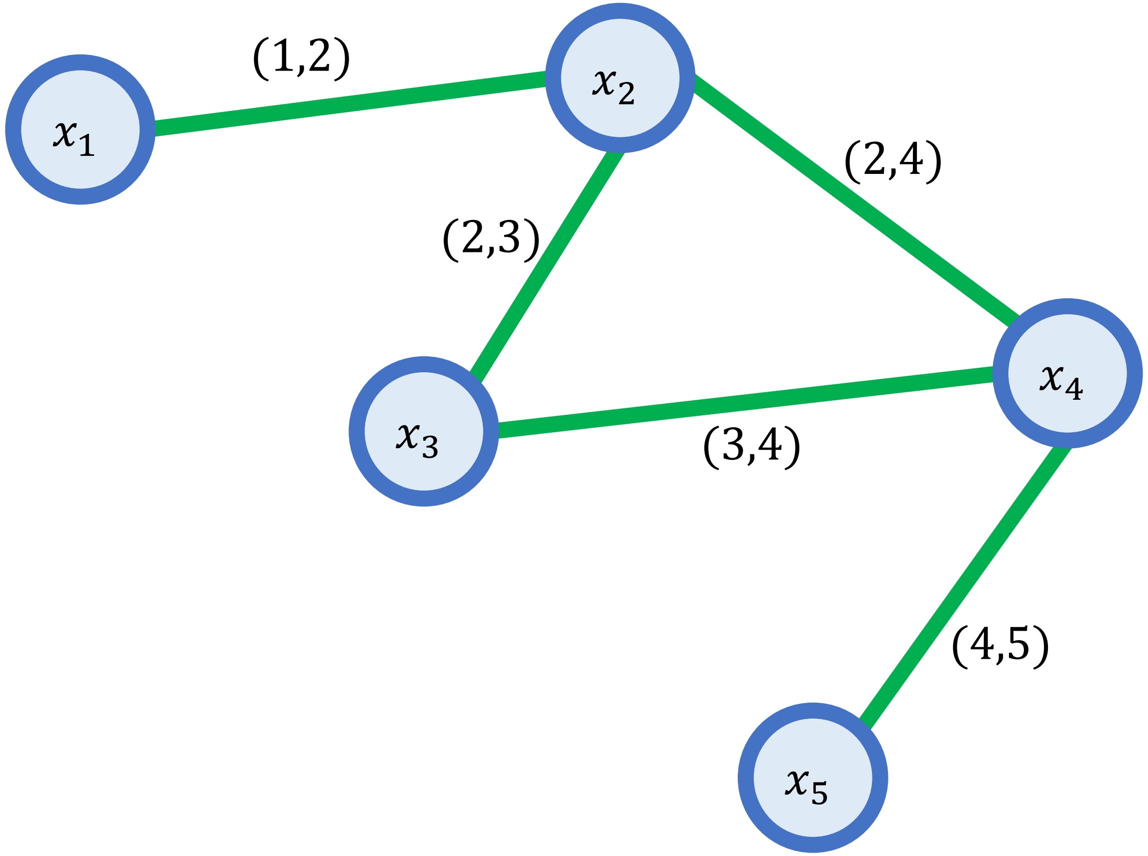
Definition (undirected graph): A **graph** G is **undirected**, if connections go both ways.

- **Undirected property:** "if node x_i is connected to node x_j, then node x_j is also connected to node x_i".
- Our example is an undirected graph, and the edges set writes as

$$\boldsymbol{E} = \{(1,2),(2,1),(2,3),(3,2),(2,4),(4,2),(3,4),(4,3),(4,5),(5,4)\}$$

Or, to avoid redundancy,

$$\boldsymbol{E} = \{(1,2),(2,3),(2,4),(3,4),(4,5)\}$$



Examples of an undirected graph

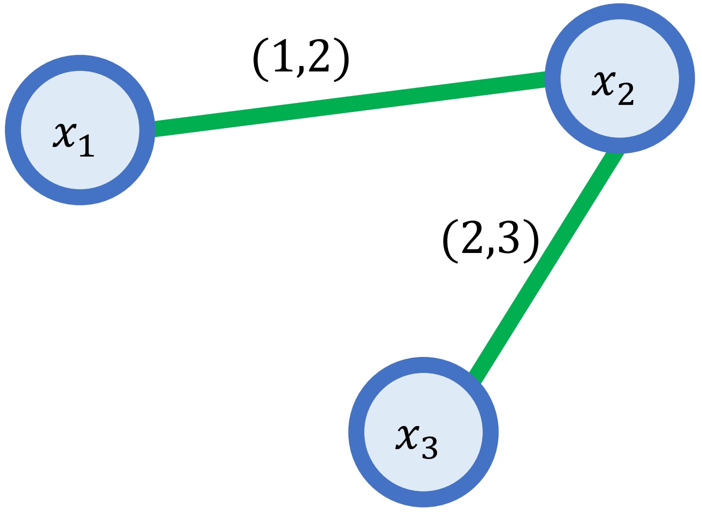
facebook

An examples of an undirected graph is... **Facebook!**

- A **node $\boldsymbol{x_i}$** simply consists of a Facebook user, and its features are user data.
- If two users i and j are friends, then there exist an **edge $(\boldsymbol{i, j})$** connecting both users.
- **Undirected property:** "if node x_i is connected to node x_j, then node x_j is also connected to node x_i".

Node name: x_i = User ID

Node features: h_i = (user first name, user family name, date of birth, age, etc.)



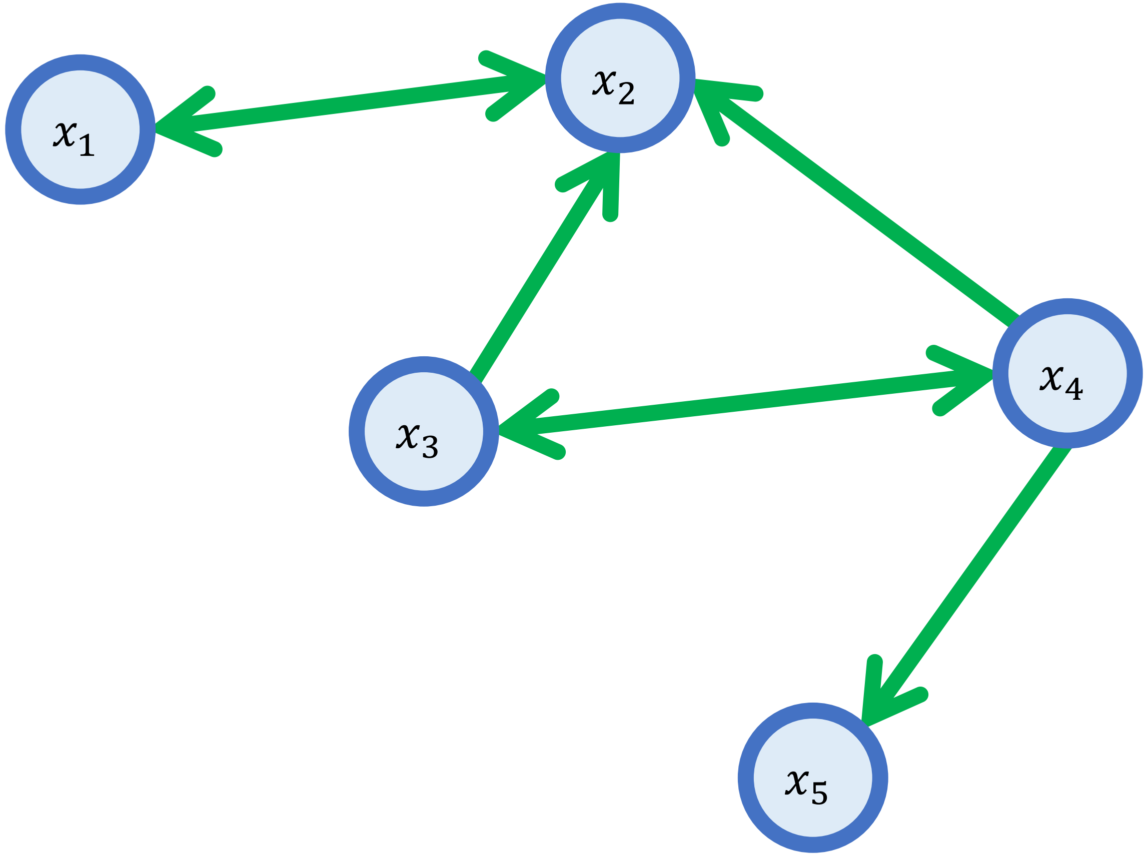
Directed graph definition

Definition (directed graph): A **graph** G is **directed**, if the undirected property does not hold.

- **Undirected property:** "if node x_i is connected to node x_j, then node x_j is also connected to node x_i".

- Our example is a directed graph, and our edges set writes as

$$\boldsymbol{E} = \{(\mathbf{1},\mathbf{2}),(\mathbf{2},\mathbf{1}),(\mathbf{3},\mathbf{2}),(\mathbf{4},\mathbf{2}),(\mathbf{3},\mathbf{4}),(\mathbf{4},\mathbf{3}),(\mathbf{4},\mathbf{5})\}$$



Examples of a directed graph

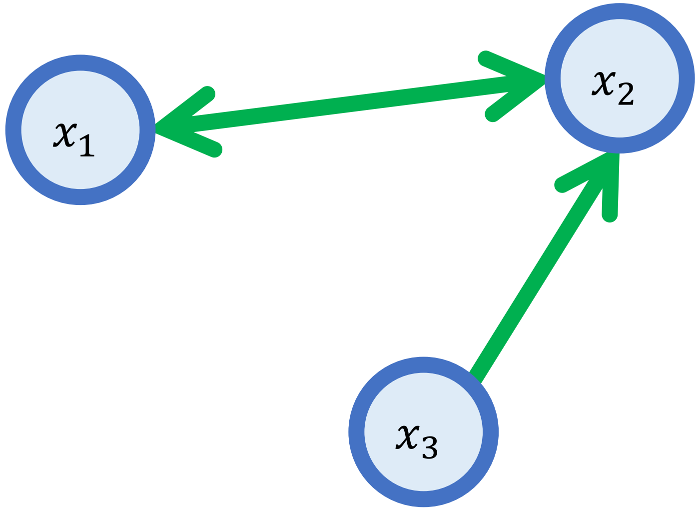
An examples of a directed graph is…
Twitter/Instagram!

- As before, a **node x_i** consists of a Twitter user, and its features are user data.

- On Twitter, the **undirected property** does not hold: you can follow people, but they do not have to follow you back.

Node name: x_i = User ID

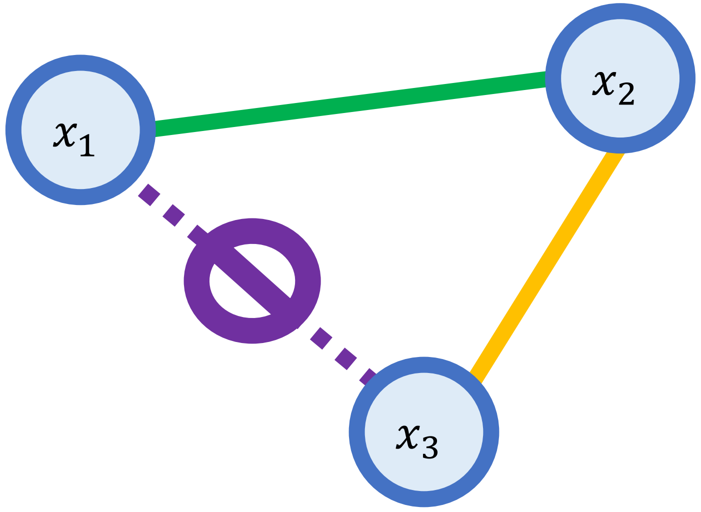
Node features: h_i = (user first name, user family name, date of birth, age, etc.)



Edges set and adjacency matrix

Definition (adjacency matrix):
The **adjacency matrix** of a graph G, is the square matrix $\boldsymbol{A}$, with general term a_{ij}

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



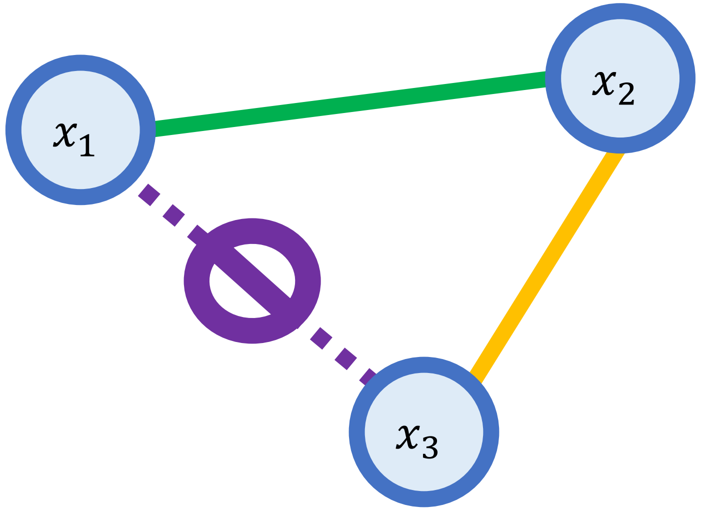
- In our example, we have

$$A = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$

Edges set and adjacency matrix

Property (adjacency matrix in <u>undirected</u> graphs):

The **adjacency matrix** of an <u>undirected</u> graph G is **symmetric**.



- In our example, we have

$$A = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$

Edges set and adjacency matrix

Property (adjacency matrix in undirected graphs):

The **adjacency matrix** of an undirected graph G is **symmetric**.

- In our example, we have

$$A = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$

Definition (Hermitian matrix):

A real-valued matrix A is **Hermitian** if and only if $a_{ij} = \overline{a_{ji}}$. (a.k.a. symmetric matrix)

Theorem (Spectral Theorem):

A **Hermitian matrix** is **unitarily diagonalizable** (i.e. it admits a basis of orthonormal vectors) with **real eigenvalues**.

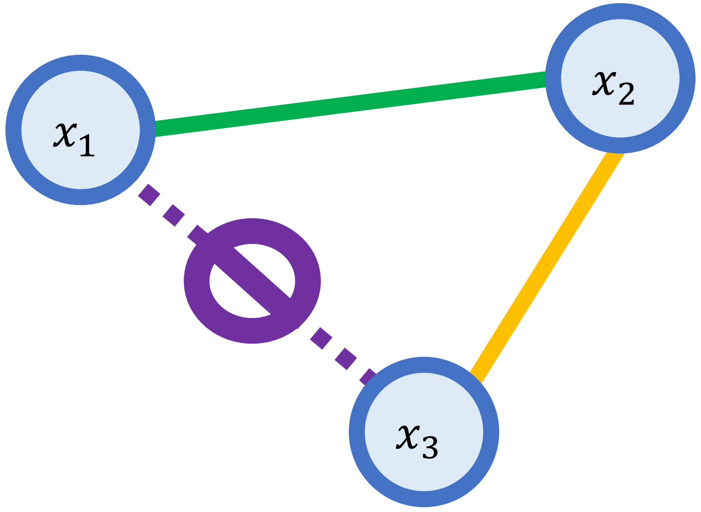
Edges set and adjacency matrix

Property (adjacency matrix in <u>undirected</u> graphs):

The **adjacency matrix** of an <u>undirected</u> graph G is **symmetric**.

- In our example, we have

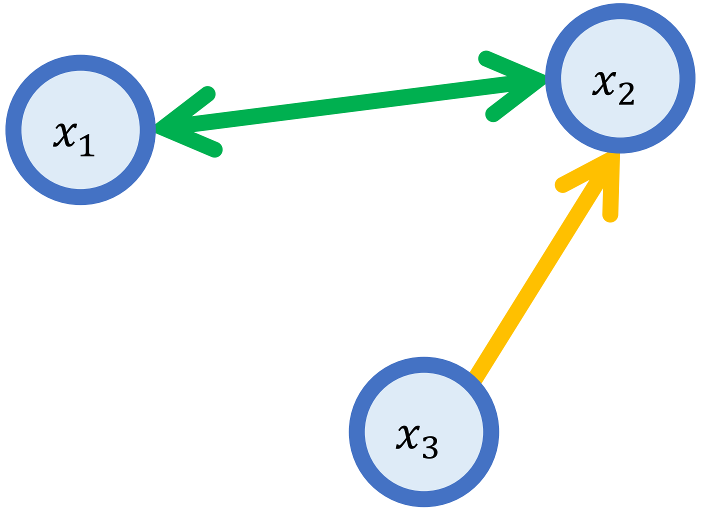
$$A = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$



Edges set and adjacency matrix

Property (adjacency matrix in <u>directed</u> graphs):

The **adjacency matrix** of a <u>directed</u> graph G is NOT necessarily **symmetric**.



- In our example, we have

$$A = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}$$

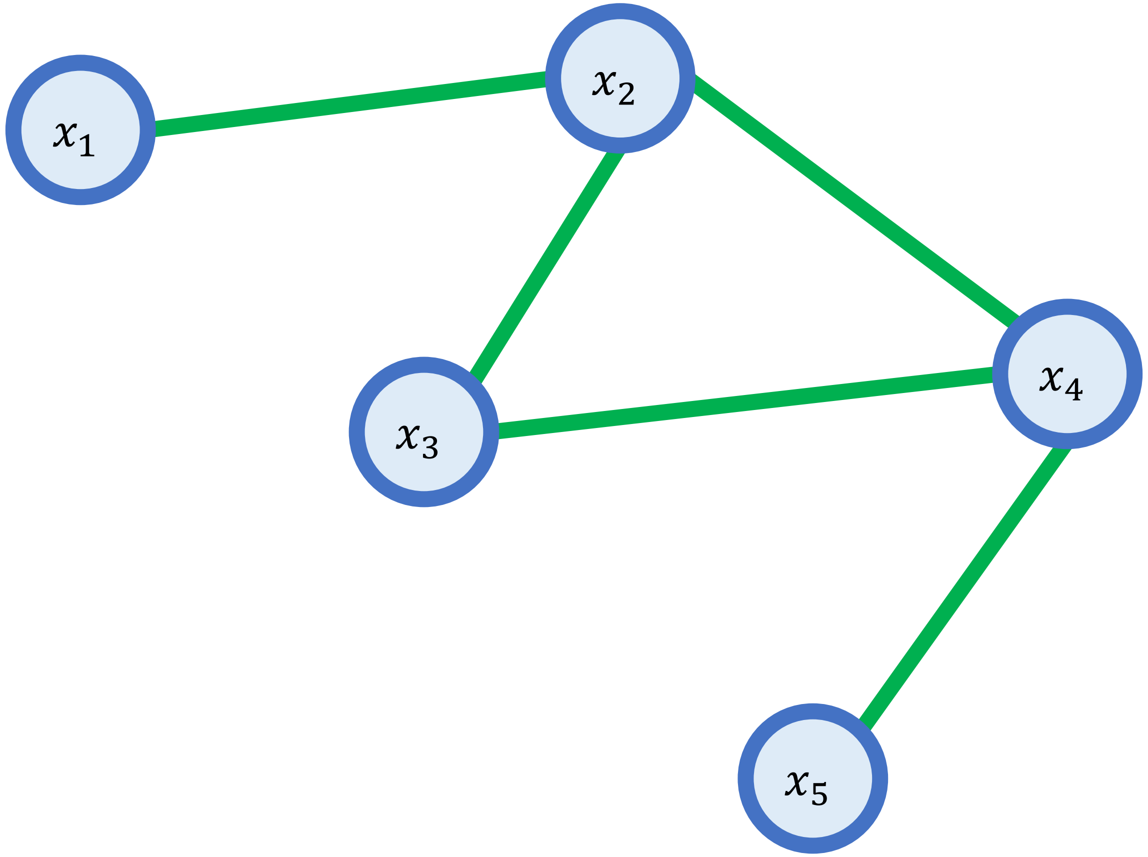
Degree of a node and degree matrix

Definition (degree matrix): The **degree matrix** of an undirected graph G, is the diagonal square matrix $\boldsymbol{D}$, with general term d_{ii}

$$d_{ii} = \text{number of nodes connected to node } \mathrm{x_i}$$

In practice, we call d_{ii} the **degree** of the node x_i.

In the Facebook example, the **degree** of node x_i is simply the **number of friends** of user x_i.

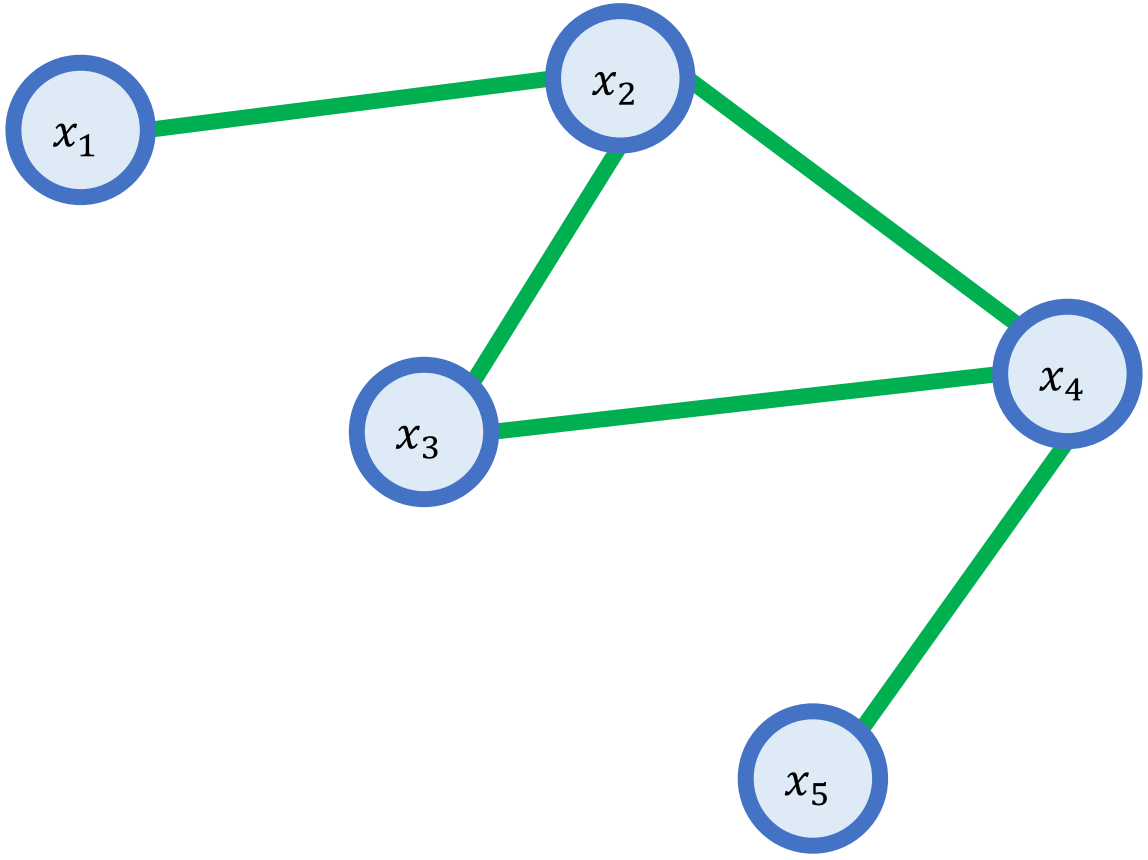


Degree of a node and degree matrix

Definition (degree matrix): The **degree matrix** of an undirected graph G, is the diagonal square matrix $\boldsymbol{D}$, with general term d_{ii}

$$d_{ii} = \text{number of nodes connected to node } \mathrm{x_i}$$

$$D = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

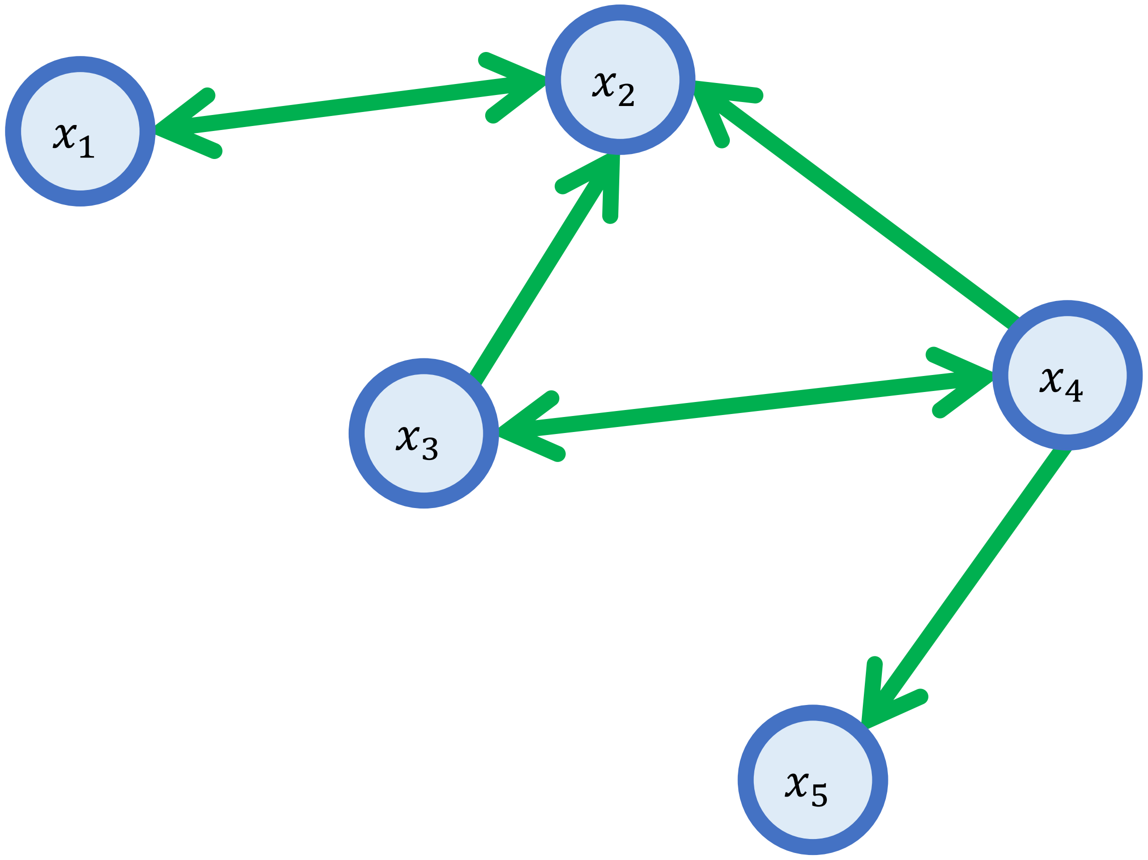


In-degree/out-degree of a node

Note: this is applicable only for directed graphs.

Definition (in-degree of a node):
The **in-degree d_{ii}^{+} of node i** of a directed graph G is the number of edges incoming to node x_i.

- In our example, the in-degree of node x_2 is 3.

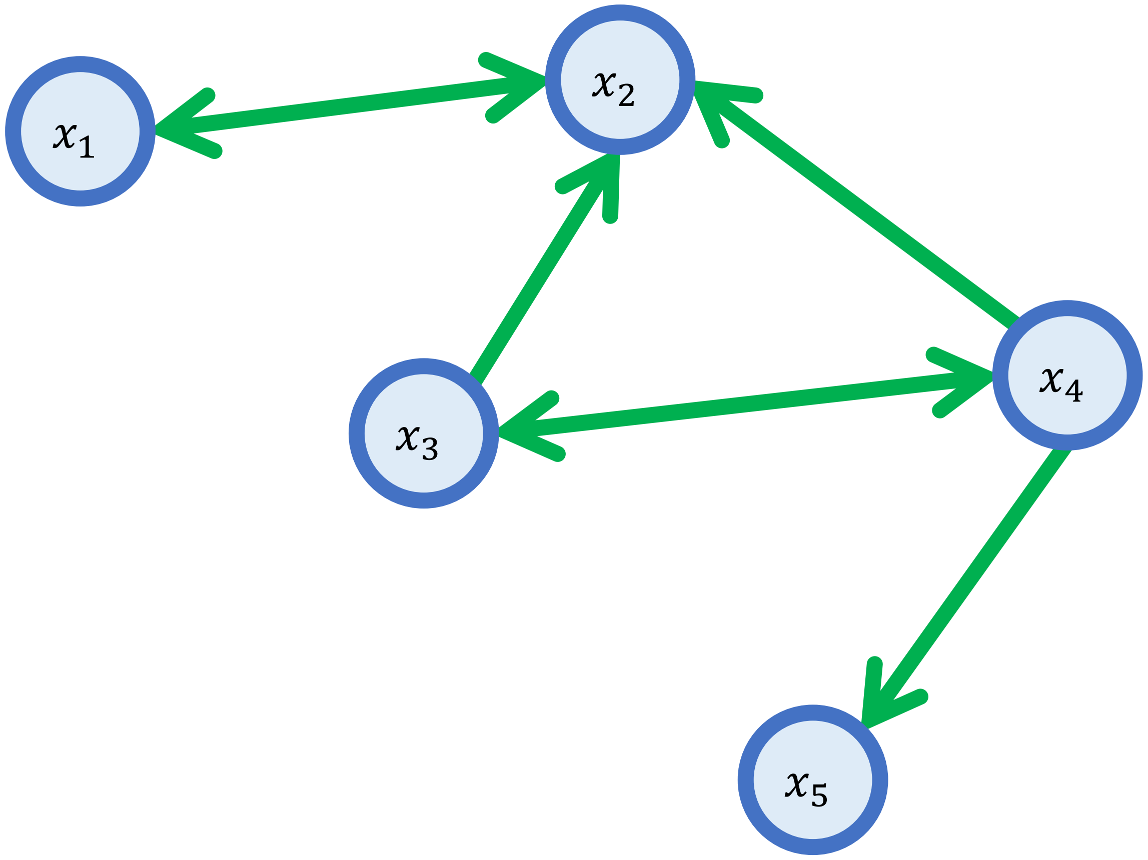


In-degree/out-degree of a node

Note: this is applicable only for directed graphs.

Definition (out-degree of a node): The **out-degree** $\boldsymbol{d_{ii}^{-}}$ **of node** $\boldsymbol{i}$ of a directed graph G is the number of edges outgoing from node x_i.

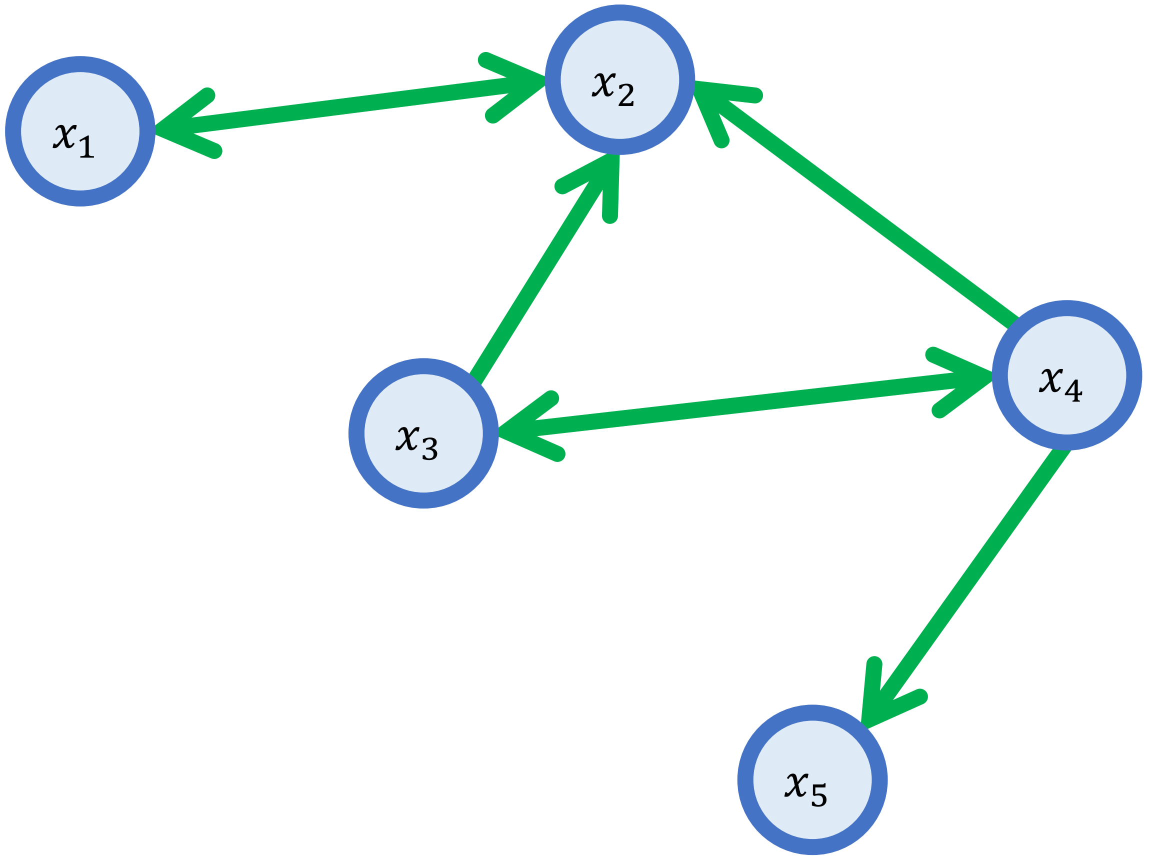
- In our example, the out-degree of node x_2 is 1.



In-degree/out-degree of a node

- **Note:** this is applicable only for <u>directed</u> graphs.
- In our Twitter example,
  - the in-degree is the number of accounts following user x_i,
  - and the out-degree the number of accounts that user x_i follows.

Definition (degree of a node): The **degree of a node $\boldsymbol{x_i}$** in a <u>directed</u> graph G, is the **sum** of its **in-** and **out-degrees**.

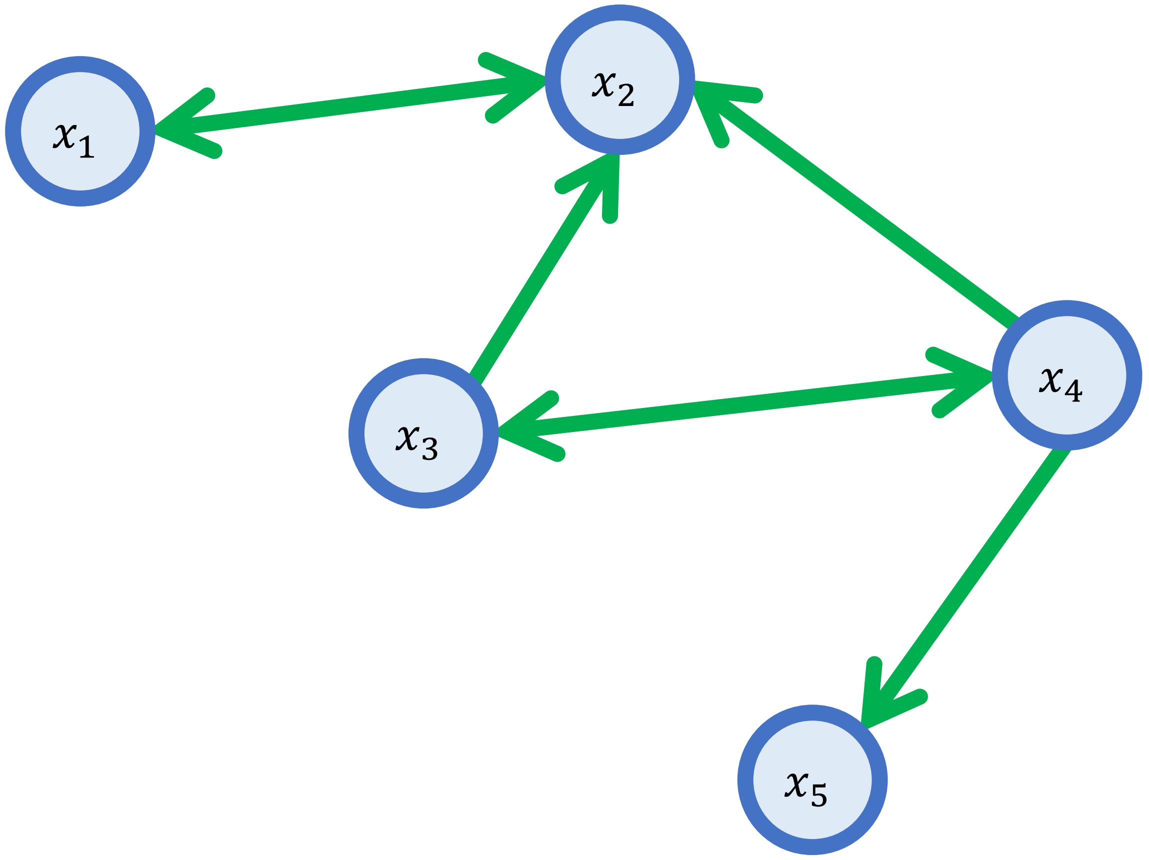


Restricted

Degree of a graph

Definition (degree of a graph):
The **degree of a graph G** is the sum of all nodes degrees.

- Undirected: $\sum_i d_{ii} = Tr(D)$
- Directed: $\sum_i d_{ii} = \sum_i (d_{ii}^{+} + d_{ii}^{-})$



Laplacian matrix (undirected graphs)

Definition (Laplacian matrix):
The **Laplacian matrix** of a graph is defined as $L = D - A$.

(Following our previous notation, we have D denoting the degree matrix of the graph, and A its adjacency matrix)

- The Laplacian matrix L has very **interesting properties**:
  - Some of these I will list as challenges in the next slide (only for the braves, who feel like practicing!)
  - Some others properties, we will discuss in the next lectures.

Challenges (for the brave only – out of class)

- **Challenge question #1 (easy):**

Consider an undirected graph G, with a Laplacian matrix L. Prove that **every row sum** and **column sum** of L is **zero**.

Challenges (for the brave only – out of class)

- **Challenge question #2 (medium):**

Prove that the Laplacian matrix L of an undirected graph G is **symmetric** and **positive semidefinite** (i.e. all its eigenvalues are positive)**.**

Challenges (for the brave only – out of class)

- **Challenge question #3 (good luck!):**

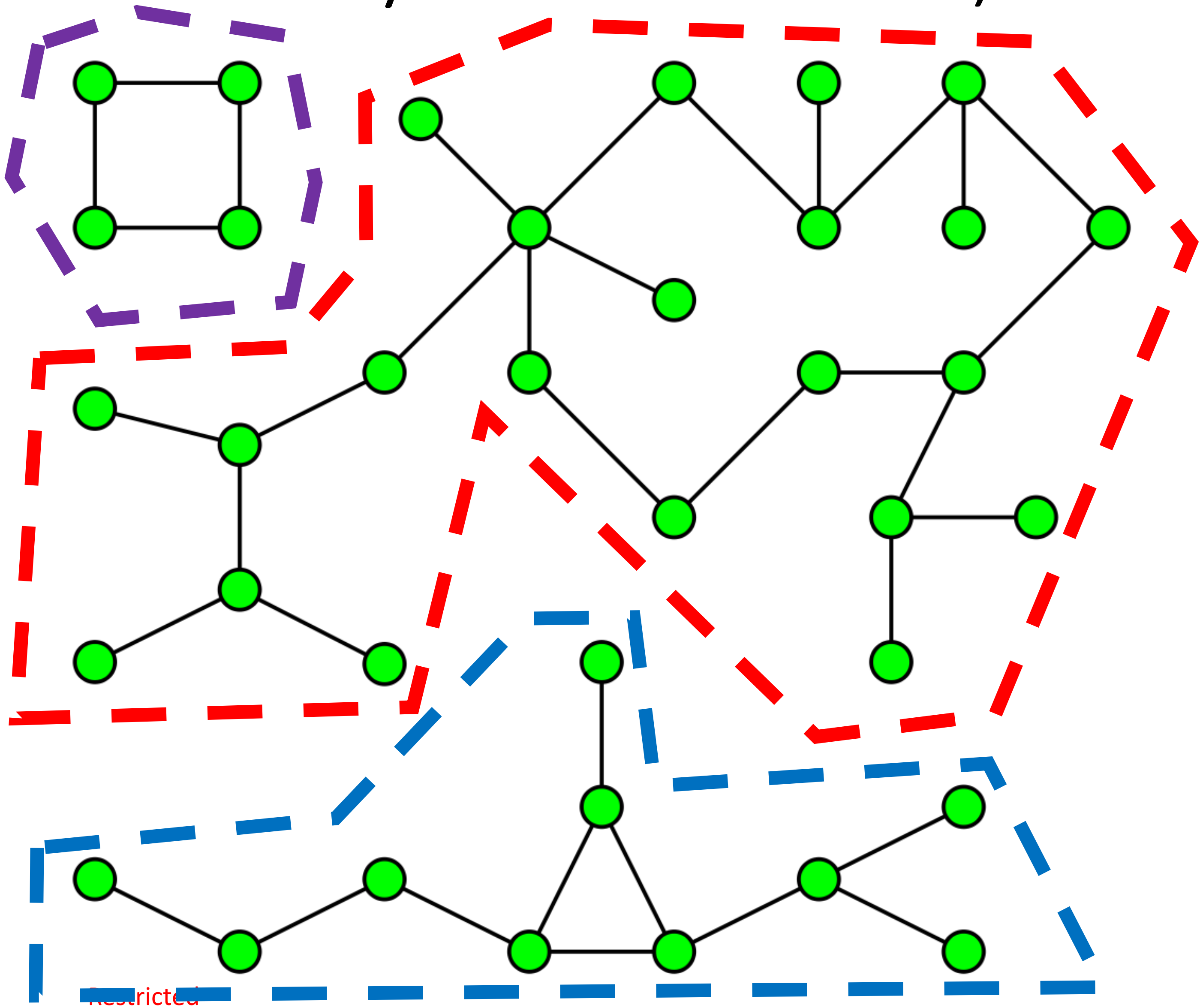
Consider an undirected graph G, with a Laplacian matrix L. Prove that the **number of connected <u>components</u>** in the graph is the **dimension of the nullspace of matrix L**, and the **algebraic multiplicity of its 0 eigenvalue**.

Definition (component of a graph): a component is a set of nodes in the graph (also called a subgraph), in which

1. All nodes in the subgraph are connected to each other.
2. None of the nodes in the subgraph are connected to any of the nodes in the supergraph (i.e. the other nodes in the graph, which are not part of the subgraph).

Challenges (for the brave only – out of class)

The graph, in our example on the right, has **three components**, in **red**, **blue** and **purple**.



Reachability in a graph

Definition (reachability):

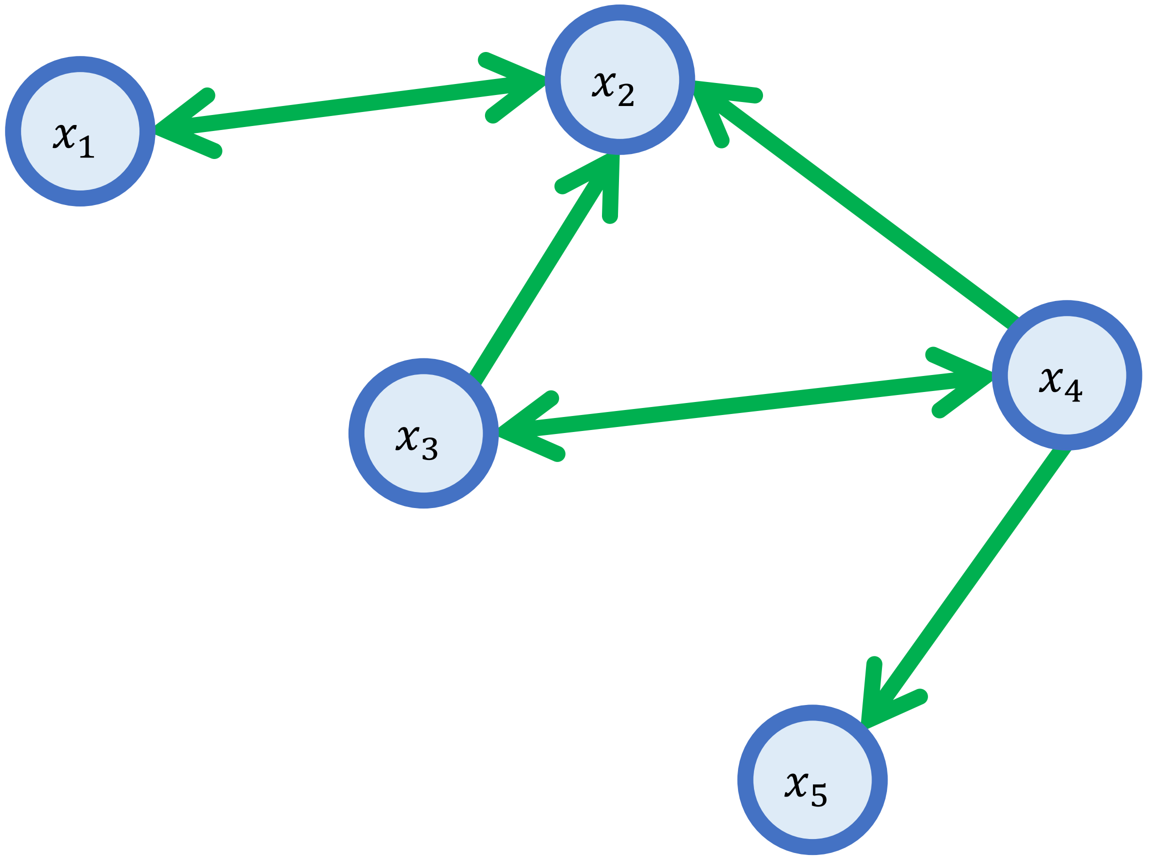
A node x_j is **reachable** from node x_i, if and only if **there exists a sequence**

$$((k_1, k_2), (k_2, k_3), \ldots (k_{m-1}, k_m)),$$

such that

- $k_1 = x_i$
- $k_m = x_j$
- And $\forall t \in [1, m-1], (k_t, k_{t+1}) \in E$

- In our example, x_5 is reachable from x_3. The converse is not true.



Distances in a graph: hops/jumps

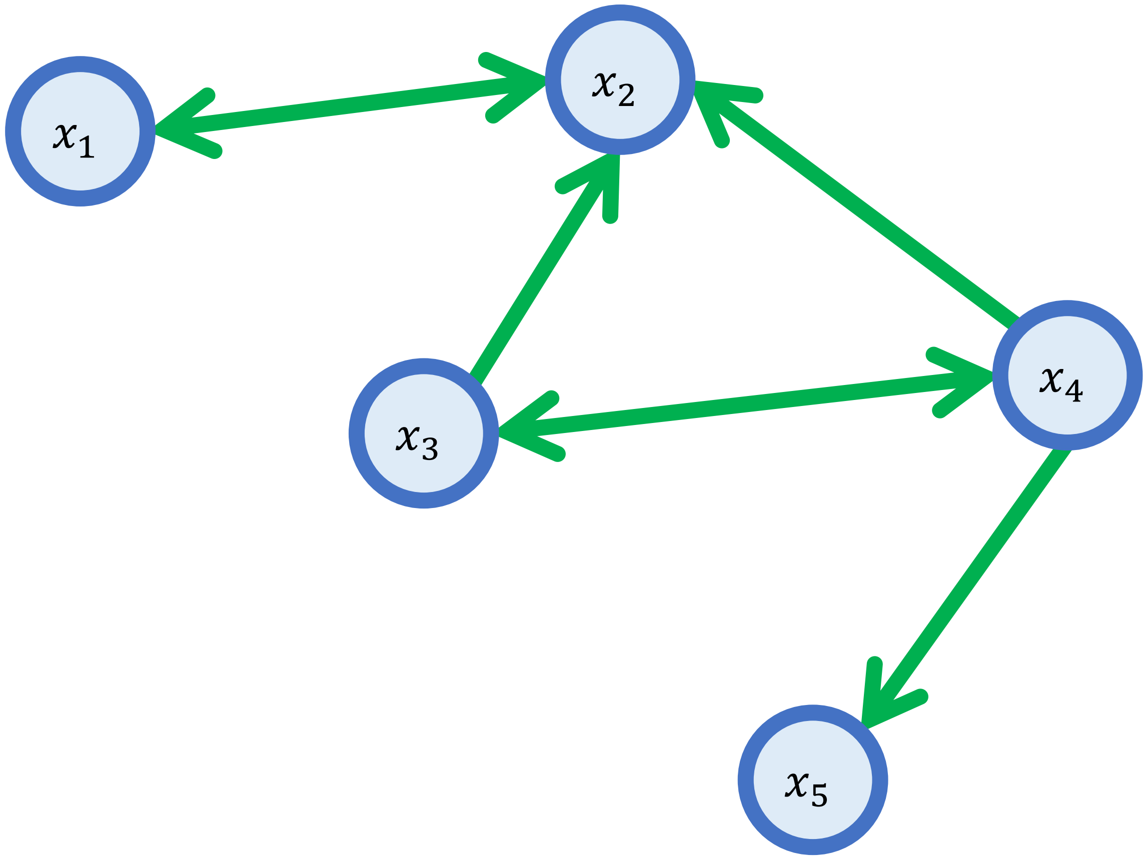
Definition (hop-distance):

Consider a node x_j reachable from x_i. The **hop-distance** $\boldsymbol{d(x_i \rightarrow x_j)}$ from node x_i to node x_j is the **length of the smallest sequence**

$$\big((k_1, k_2), (k_2, k_3), \dots (k_{m-1}, k_m)\big),$$

such that

- $k_1 = x_i$
- $k_m = x_j$
- And $\forall t \in [1, m-1], (k_t, k_{t+1}) \in E$



In our example above, the hop-distance from x_4 to x_1 is 2.

Distances in a graph: hops/jumps

2nd-degree connections - People who are connected to your 1st-degree connections. You'll see a **2nd** degree icon next to their name in search results and on their profile. You can send them an invitation by clicking the Connect button on their profile page, or by contacting them through an InMail. **Learn more about InMail.**

Jun Liu • 2nd

Assistant Professor, Singapore University of Technology and Design

Singapore

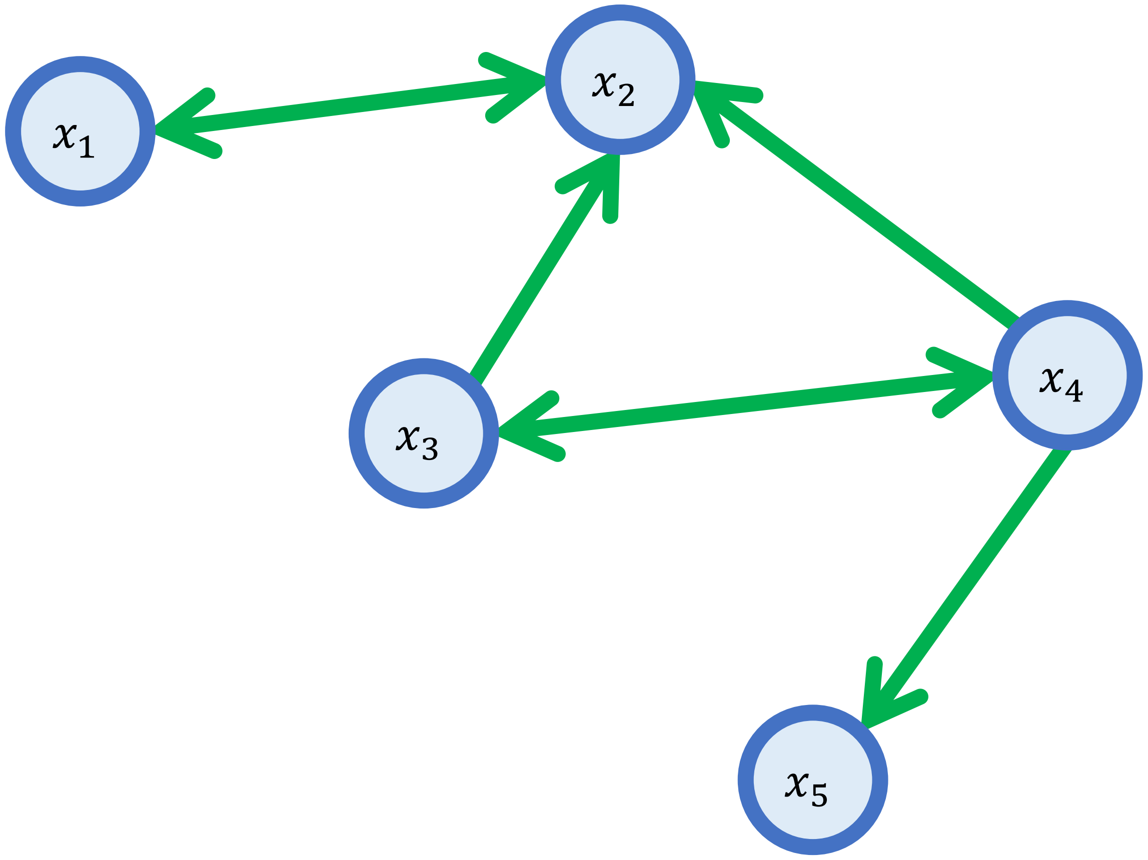
Tianruo Shen, Pranav Agarwal, and 8 other shared connections

Connect

Distances in a graph: hops/jumps

Note: in directed graphs,

- We must to take into account the direction of the edges!
- Reachability might exist from x_i to x_j, but not from x_j to x_i!
- Number of hops to get from x_i to x_j is not necessarily the same as the number of hops to get from x_j to x_i!



Edge attributes

We have seen earlier that:

- A **node** has a **name** $\boldsymbol{x_i}$, which is used for indexing and differs from one node to another.
- A node may also have **attributes**, for instance:
  - Some **node features**, defined as a vector $\boldsymbol{h_i}$,
  - Some **node label** y_i, defining a class for the node.

- An **edge** $(\boldsymbol{i}, \boldsymbol{j})$ may also have **attributes**, for instance:
  - Some **edge features**, defined as a vector $\boldsymbol{e_{ij}} \in \mathbb{R}^{\boldsymbol{F'}}$, with F' elements,
  - Some **edge label** l_{ij}, defining a class for the edge.

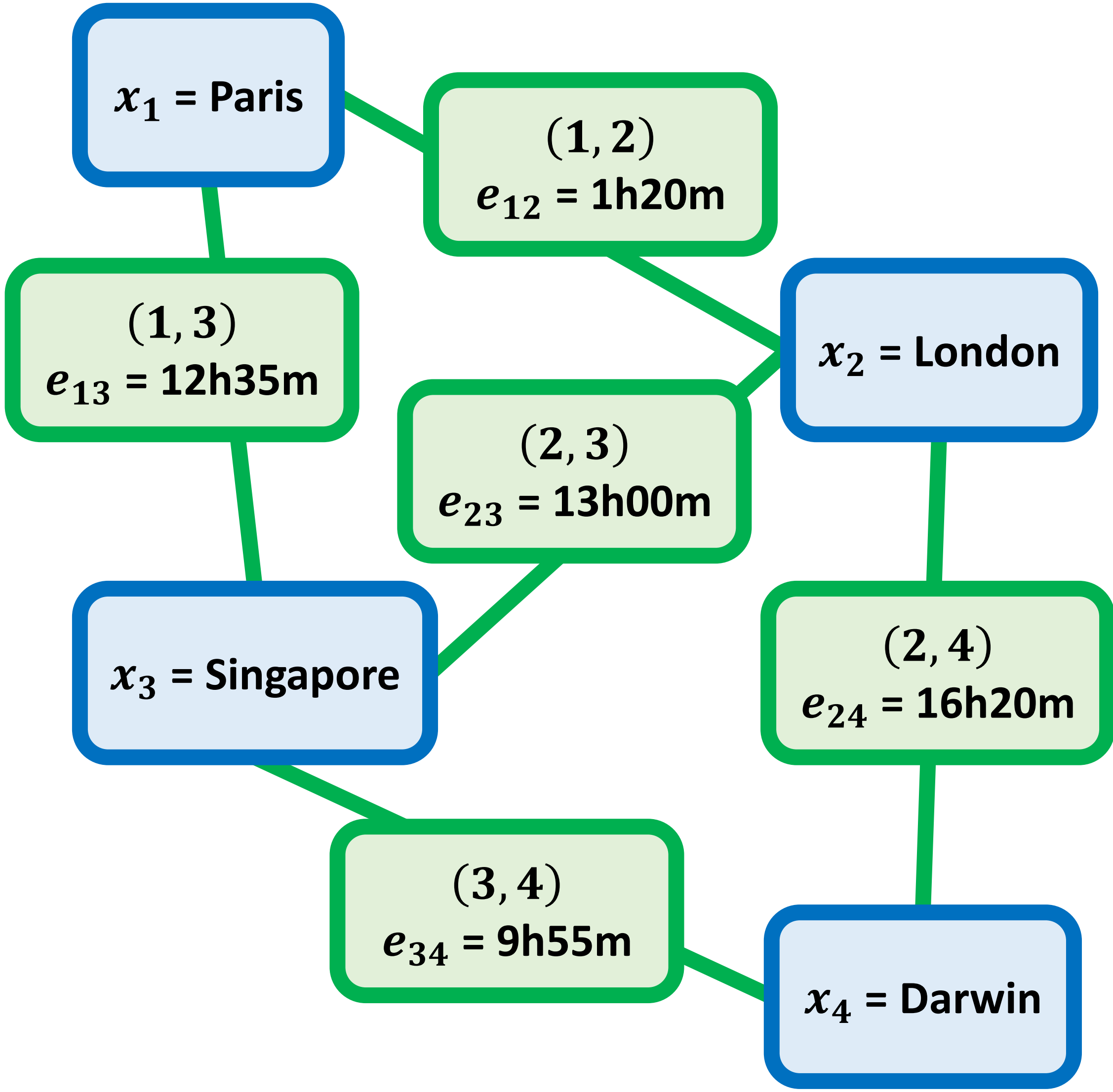
- **Open question:** Could you think of possible edge attributes for our Facebook/Twitter/Instagram examples?

Some illustrative examples

- On top of the examples that we have seen so far (mostly social networks, such as Facebook, Twitter, LinkedIn,…)
- Here are a few examples of graphs and some interesting underlying problems, which could be investigated.

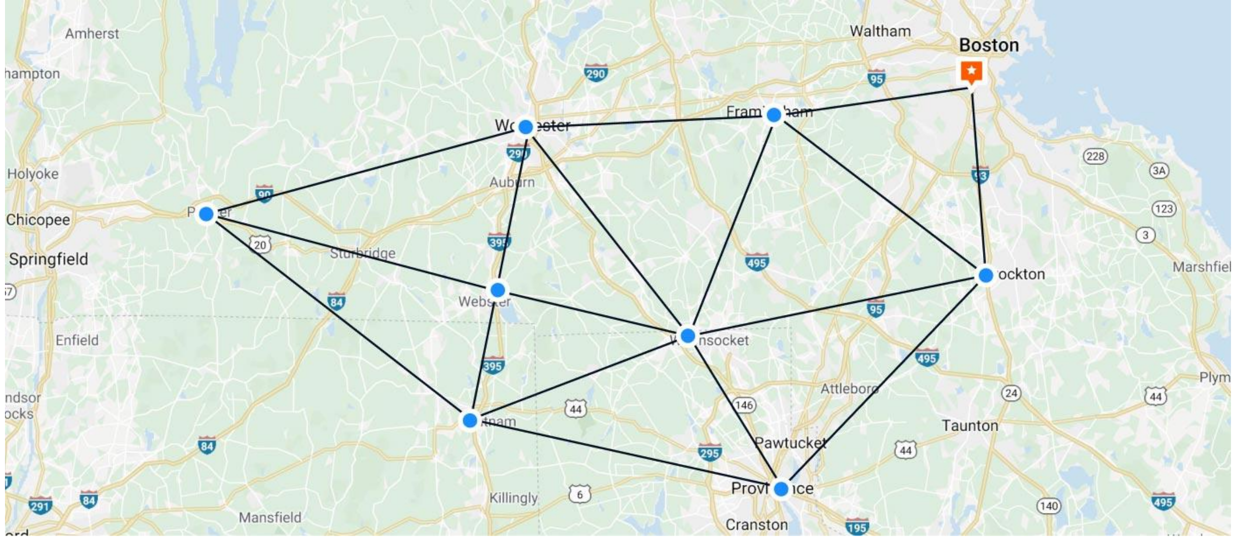
Distances in a graph: using edge features

- It is very common for the edge features to carry an information related to the distance between the nodes connected by this edge.
- This information can be used to define a distance metric between two nodes.
- Typical problem: define the shortest path/distance between two places.



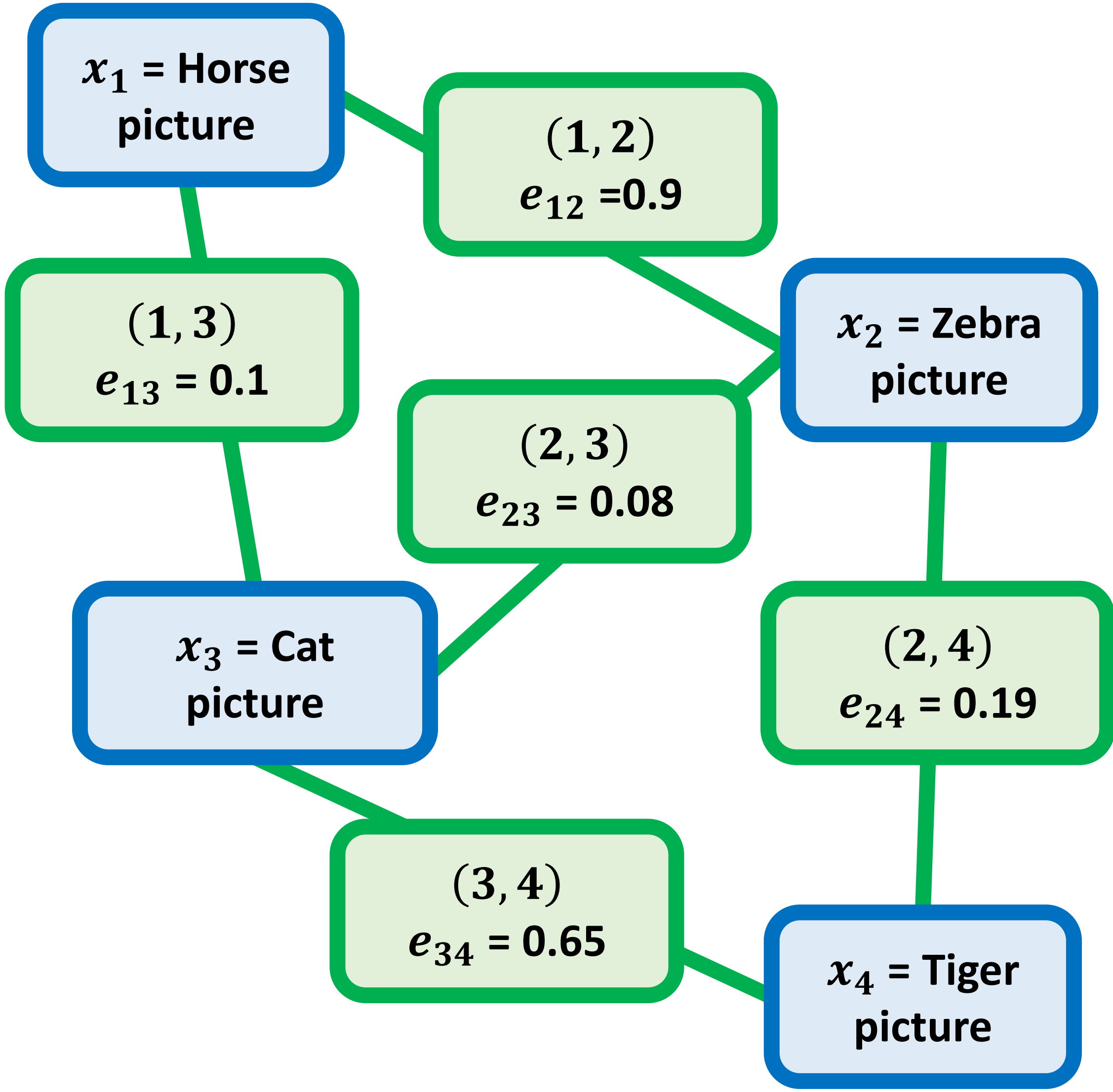
Traveling salesman problem

- Consider a **fully connected graph** (i.e. a graph with all possible edges drawn).
- Edges weight e_{ij} consists of the amount of time needed to go from node x_i to node x_j.
- What is the best sequence to visit all nodes, starting from x_1 and ending in x_1?
- It should **minimize** the sum of visited edges.

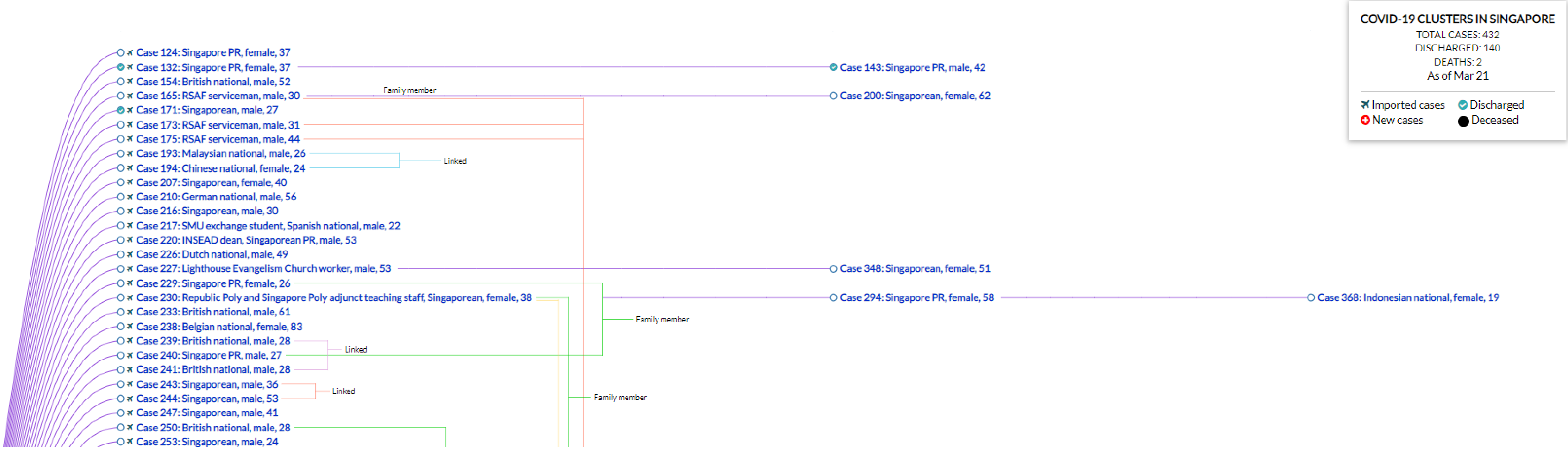


Distances in a graph: an images example

- In computer vision, you might also need to compute a similarity between pictures of a same dataset.
- This similarity measure (e.g. triplet loss) can also be interpreted as a distance measure on edges connecting images (nodes).



COVID-19 contagion graph



Source: https://infographics.channelnewsasia.com/covid-19/coronavirus-singapore-clusters.html

COVID-19 contagion graph

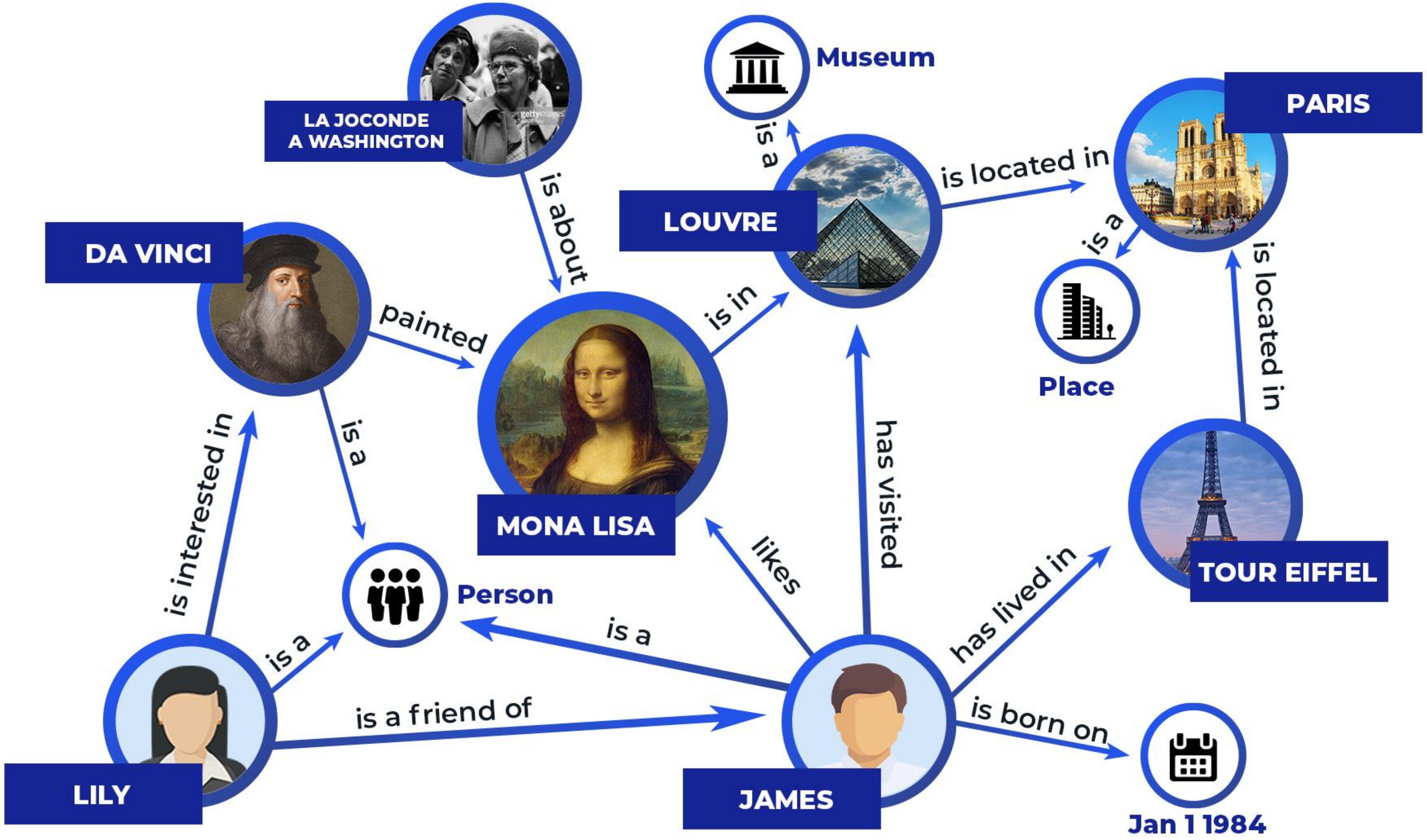
- Nodes are all Singaporean individuals
- Nodes names: person’s name
- Nodes features: age, work, address, etc.
- Node labels: (infected, not infected)
- Edges indicate contact links between individuals
- Edge features could be details related to the contact between individuals (work/leisure, cluster address, etc.)
- Edge might be directed, to indicate the contagion direction
- Edge labels are 1 if this contact led to an infection from one person/node to another, and 0 otherwise.

COVID-19 contagion graph

Many application for such a COVID-19 contagion graph.

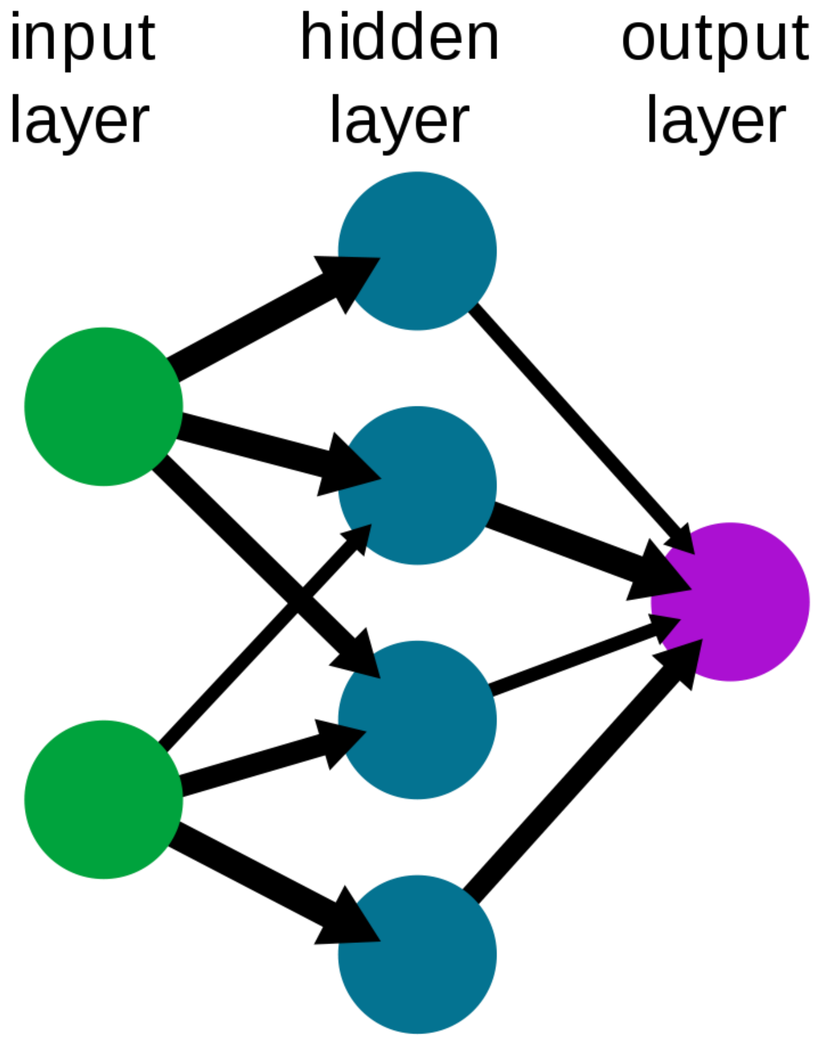
- **Contact tracing:** reconstructing the contagion links in the graph as close to the reality as possible.
- **Confinement and social distancing:** limiting the number of edges between nodes, to prevent contagion.
- **Finding patient zero:** consists of looking for the one infected node that can reach all other infected nodes through infected edges.
- **Contagion inference:** attempt to predict the next infected person based on current graph state.
- Etc.

Knowledge graphs: a.k.a. Wikipedia

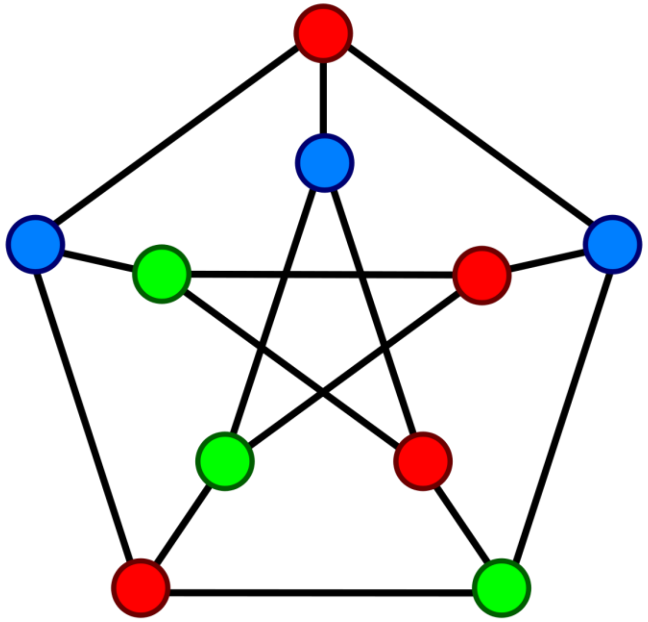


Neural Networks are graphs as well?!

- Yes, neural networks can also be modeled as directed graphs…
  - Nodes = neurons
  - Nodes features = logits, input/output nodes values, etc.
  - Edges = node connections (might be erased by dropout)
  - Edges features = weights, activation functions, propagation rules, etc.
  - Forward propagation and backpragation mechanisms…

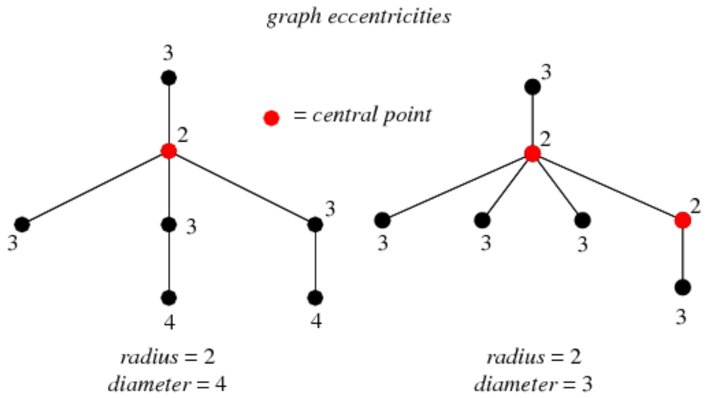


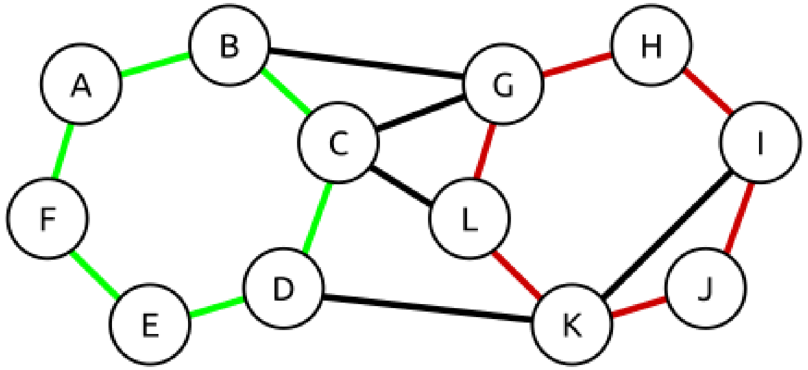
More problems on Graph Theory



Graph theory has many more open problems, which are currently unsolved.

- Graph coloring
- Radius and diameter of a graph
- Graph cycles detection
- Many other optimization problems on graphs…
- Stochasticity on Graphs…





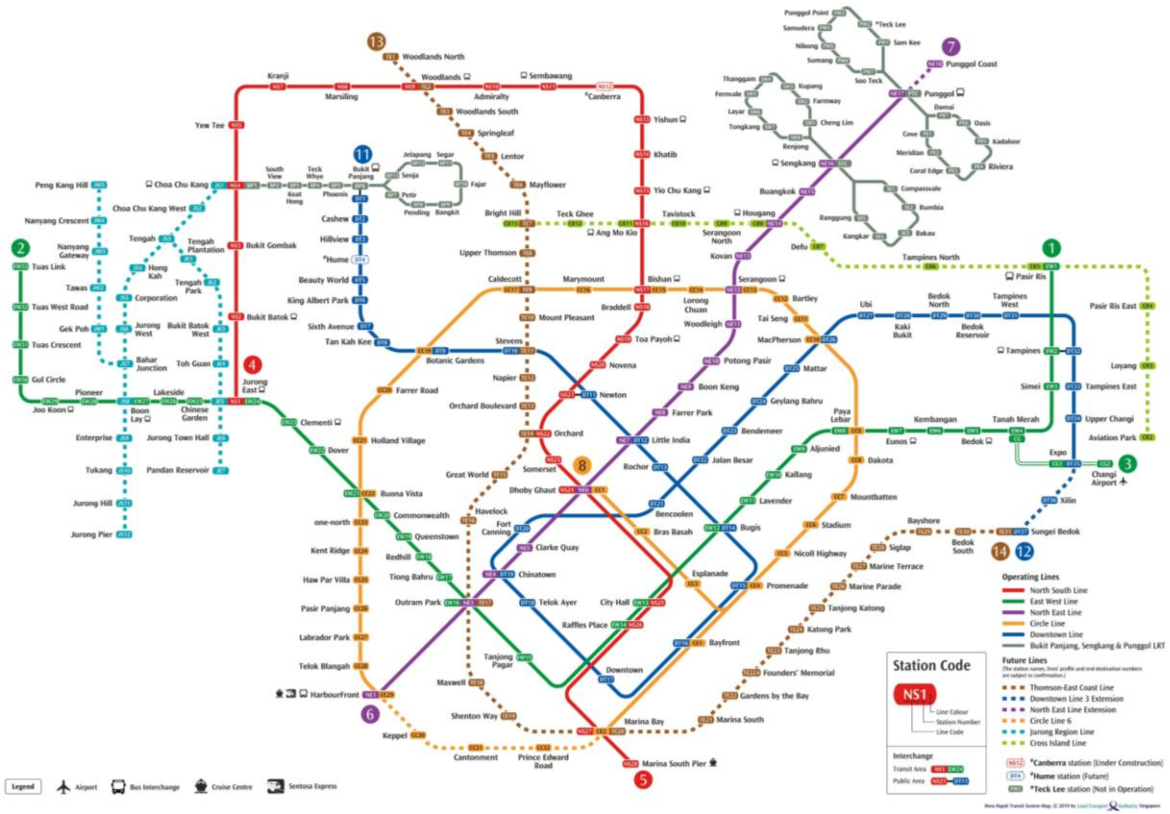
Restricted

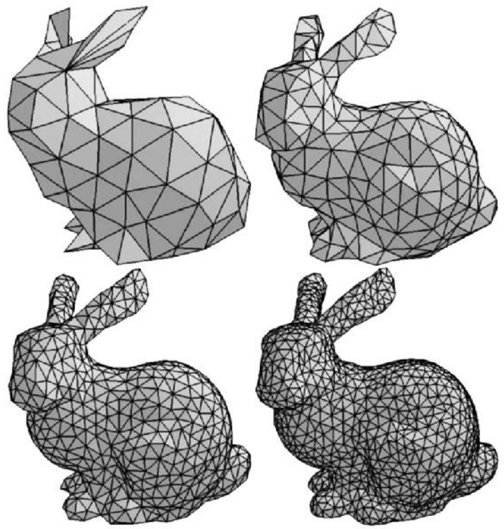
Conclusion

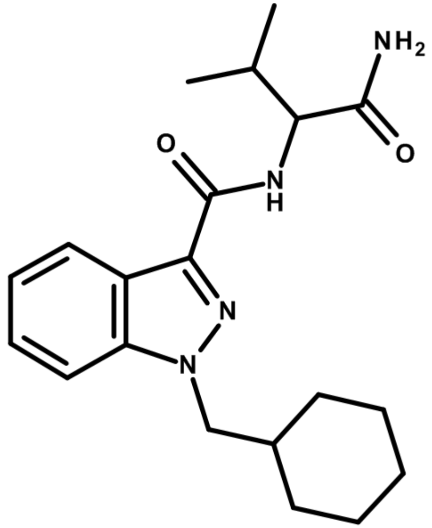
In this lecture

- Introduction to graph theory
- Definitions for key concepts
- A few practical applications for graph theory (social networks, neural networks, transport networks, molecular structures, 3D mappings, etc.)

- Answers to challenges: ***[Luxburg]*** U. von Luxburg, A Tutorial on Spectral Clustering. (refer to Chapter 3 of paper)







Learn more about these topics

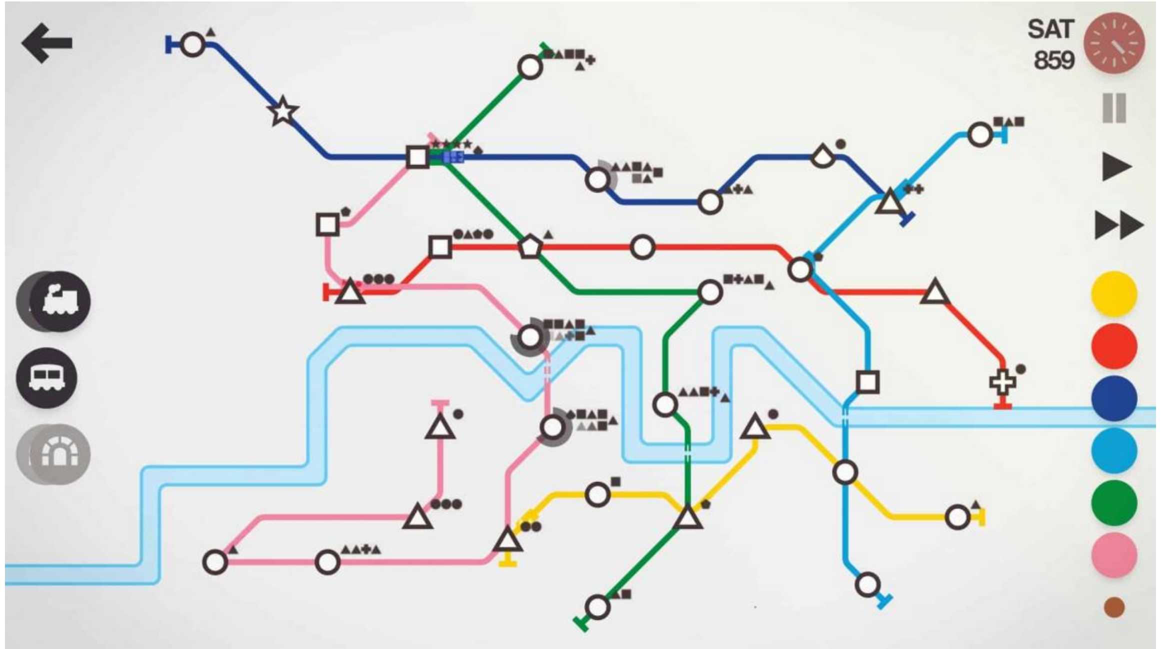
Out of class, for those of you who are curious

- [Bondy1976] Bondy et al., “Graph Theory with applications”, 1976.
- [Bondy2008] Bondy and Murty, “Graph Theory”, 2008.
- [Medium1] “A Gentle Introduction To Graph Theory”, 2017. https://medium.com/basecs/a-gentle-introduction-to-graph-theory-77969829ead8

Learn more about these topics

Out of class, for those of you who are curious

- [MiniMetro] MiniMetro game, using graphs-based transportation/flow networks for urban planning (Web Demo, available on Steam/Phone) https://www.coolmathgames.com/0-mini-metro-london



A bit of Practice

Have a look at the practice exercises!