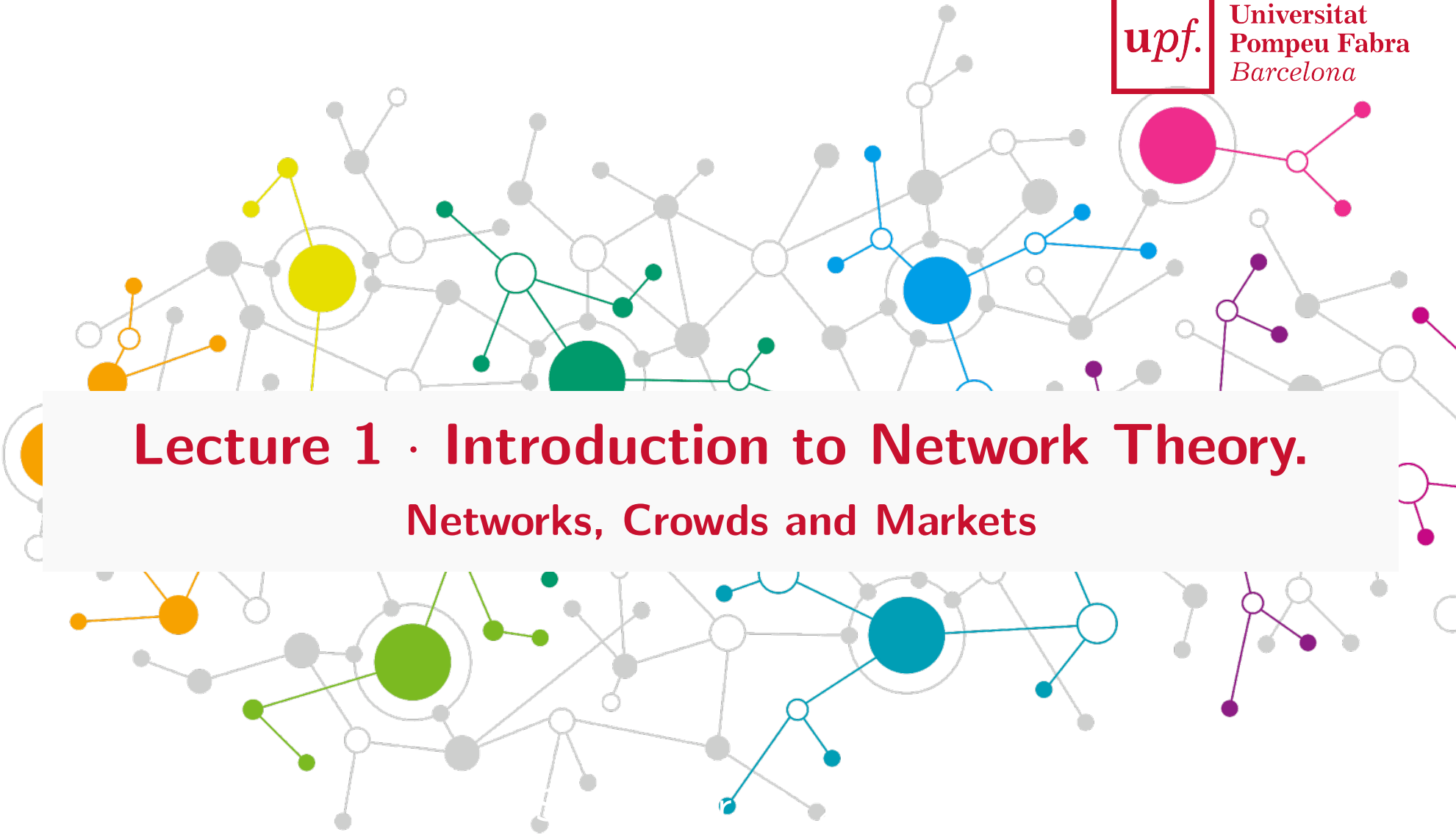


Universitat Pompeu Fabra Barcelona

Lecture 1 · Introduction to Network Theory.

Networks, Crowds and Markets

Course Introduction

Goals of the course:

- Understand the structure and dynamics of networks.
- Develop intuition through examples and simple simulations.
- Learn mathematical tools to analyze networks
  - graph theory, probability, and linear algebra
- Study applications to economics, social sciences, and data science.

Prerequisites:

- No formal prerequisites beyond Math I-III.
- Helpful background: basic calculus, linear algebra, and probability.
- Familiarity with simple programming (e.g., Python or R) will make it easier to follow examples.

Schedule and Office Hours

Instructor: Piotr Zwiernik (piotr.zwiernik@upf.edu)

Office Hours: Tuesday 2-3pm (20.202)

Lectures: Monday and Tuesday 3-4:30 pm

Seminars: Wednesdays, weeks 3,4,5,6,7,8.

- group 1, 3-4:30pm; group 2, 4:30-6pm

Bibliography

BK Easley & Kleinberg (2010). *Networks, Crowds, and Markets*. Cambridge University Press.
`https://www.cs.cornell.edu/home/kleinber/networks-book/networks-book.pdf`

B Barabási (2016). *Network Science*. Cambridge University Press.
`https://networksciencebook.com/`

MFD Menczer, Fortunato, David (2020). *A First Course in Network Science*. Cambridge University Press.
`https://cambridgeuniversitypress.github.io/FirstCourseNetworkScience/`

and also

- Saoub (2017). *A Tour Through Graph Theory*. Springer.
- Newman (2010). *Networks: An Introduction*. Oxford University Press.

Evaluation

- Final Exam (50%) – planned for 11 December.
- Midterm (30%) – planned for 28 October.
- Report and Presentation (20%)

$$G = \begin{cases} 0.5G_{FE} + 0.3G_M + 0.2G_{RP} & \text{if } G_{FE} \geq 4 \\ G_{FE} & \text{if } G_{FE} < 4 \end{cases}$$

You must score 4 or above in your final exam in order to pass the course.

Report and Presentation

- The aim of the report is to explore a topic related to Networks that hasn't been covered in the course lectures. Applied or theoretical.
- A comprehensive list of suggested topics and detailed instructions will be made available next week.
- You have two more weeks to choose a topic **or propose one**.
- Reports will be presented in groups consisting of ~ 4 students who belong to the same seminar group.
- The deadline for report submission is 12th November.
- Report presentations will take place during seminars 5 and 6, scheduled for 12 and 19 November, respectively.

Final Exam and Recovery Exam

- The final exam is scheduled on December 11, 3-5pm.
- In the second term, a recovery exam will be conducted (date to be announced) for students who have not successfully passed the course.
  - The recovery exam will be available to students who meet the following conditions:
    - Attainment of a grade of 4 or higher for your report and presentation.
    - Attendance of the least three seminars of the course.
    - Have average at least 3 in the final exam and midterm.
  - Students who do not fulfil all the conditions listed above will not be eligible to take the Recovery Exam.

First examples and motivation

Why Economists Care About Networks

- **Labor Markets:** Weak ties (Granovetter, 1973) explain how people find jobs and why networks matter for inequality.
- **Financial Contagion:** 2008 crisis showed how bank exposures form a network where shocks spread.
- **Trade and Production:** Countries and firms are linked in global supply chains; disruptions spread along these networks.
- **Spread of information:** Ideas, news, and technologies often spread only through network connections.

Takeaway: Networks are not only mathematical objects; they capture the structure of labor markets, finance, trade, and innovation.

The ZOO of Networks

Networks are everywhere and they come in different forms.

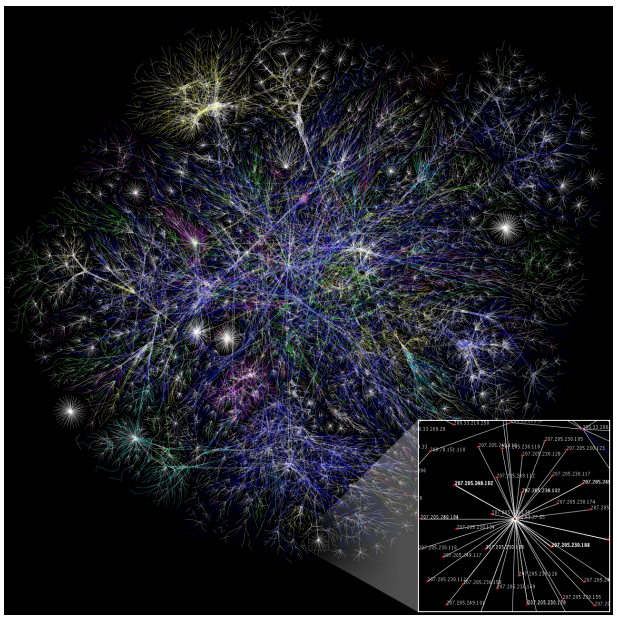
- Technological/Transportation Networks
- Social Networks
- Information Networks
- Market and Interaction Networks
- Biological Networks

Form of networks are highly application dependent and so are the relevant questions.

- Network Models
- Structural Properties
- Network Dynamics
- Network Behaviour

Technological Networks

Public Global Computer Network



- **What:** Physical and logical infrastructure of the Internet.
- **Vertices:** Routers, servers, computers, autonomous systems.
- **Edges:** Physical links (fiber, cables) or logical connections (IP routing).
- **Directed?** Typically modeled as undirected (connectivity) or directed (routing/traffic).
- **Special:** Hierarchical (backbones, ISPs), scale-free degree distribution, massive growth (2.1B users in 2014, billions more today).

Technological Networks: Electrical Power Grid

Vulnerability and Cascading Failures

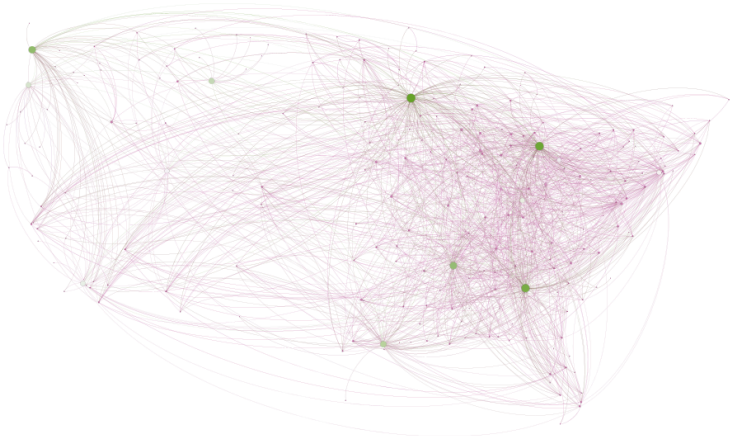
- **Vertices:** Power plants, substations.
- **Edges:** High-voltage transmission lines.
- **Problem:** Highly connected networks bring efficiency, but may spread failures.
- **Themes:** Robustness, redundancy, targeted attacks, cascading failures.
- **Dramatic example:** *Apagón in Spain (2025).*

Another example, from North America, is described in [B], Section 1.1.1.

Cascading failures are one of the topics for the report.

Technological Networks

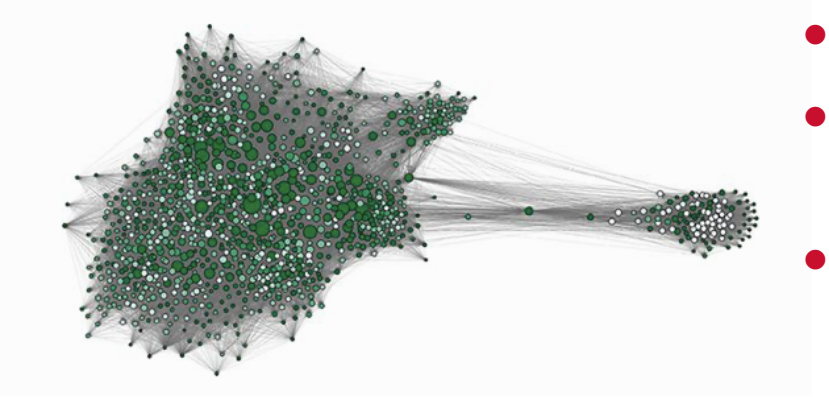
US Air Transportation Network



- **What:** Flight connectivity among airports.
- **Vertices:** Airports.
- **Edges:** Scheduled flight routes.
- **Directed?** Yes (A$\rightarrow$B may exist without B$\rightarrow$A; plus weighted by flights/passengers).
- **Special:** Strong hub-and-spoke structure; seasonal/temporal dynamics.

Social Networks: Sociogram

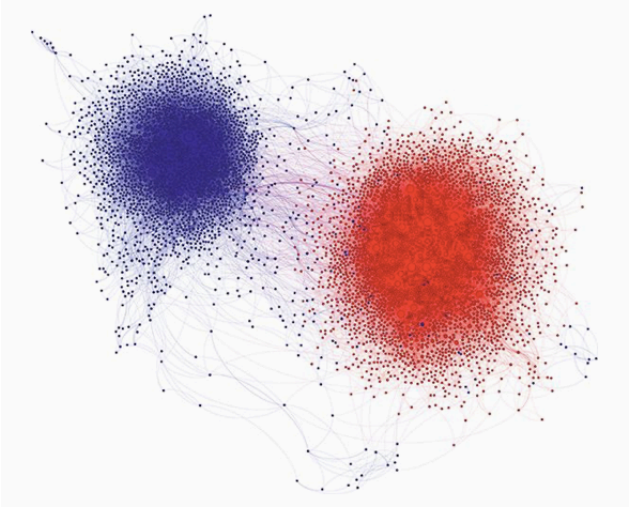
Social Interaction



- **What:** Communication ties among individuals in a group.
- **Vertices:** People (e.g., employees/classmates).
- **Edges:** Social/communication links (e.g., emails, interactions, being "friends").
- **Directed?** Often directed (sender→receiver) and weighted (counts). But friendship networks are undirected.
- **Special:** Community structure, homophily, triadic closure.

Social Networks: Twitter

Social Communication

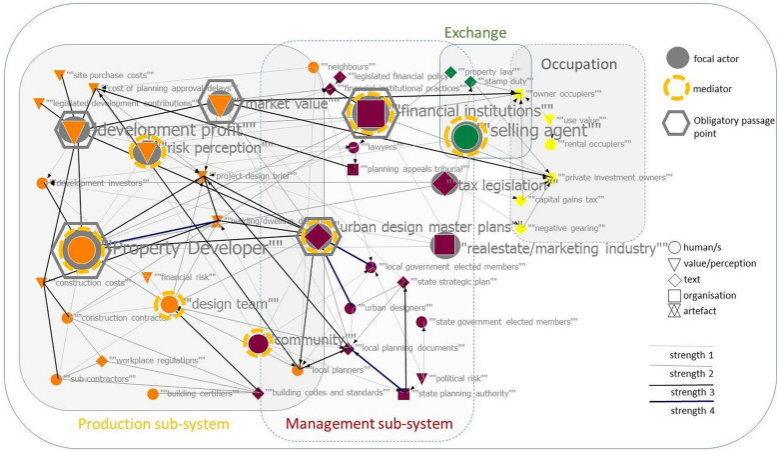


- **What:** Online social network with information diffusion.
- **Vertices:** Accounts/users.
- **Edges:** Follows (directed), interactions (mentions/retweets/replies).
- **Directed?** Yes (follow graph is directed).
- **Special:** Heavy-tailed degree, influencers/hubs, cascades.

Information diffusion on networks is one of the topics for the report.

Market interaction network: Actor–Network Theory (ANT)

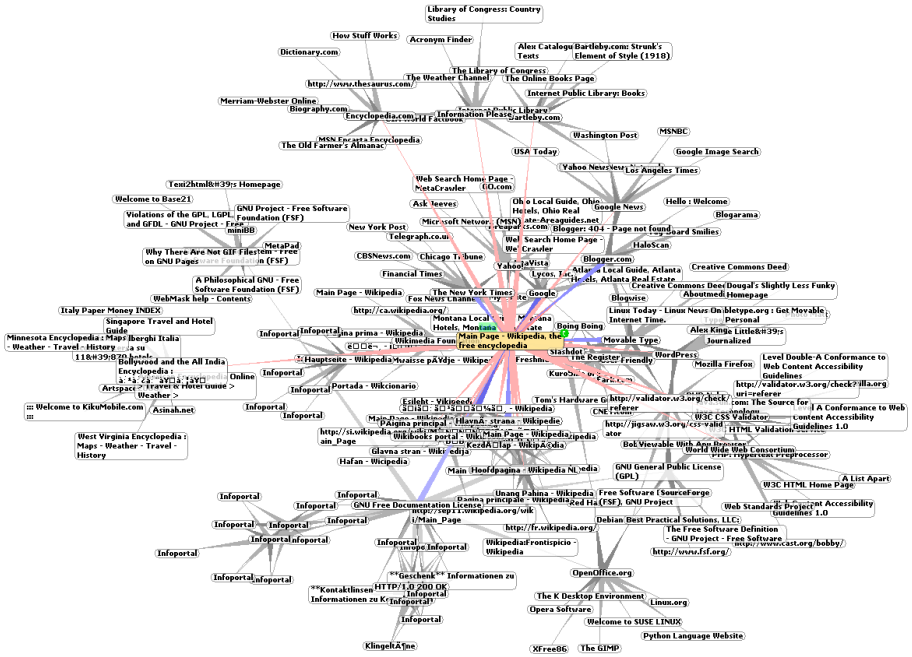
Socio–Economic Interactions



- **What:** heterogeneous network of human and non-human actors.
- **Vertices:** People, institutions, policies, artefacts, perceptions.
- **Edges:** Influence, dependency, flow of resources or constraints.
- **Directed?** Often directed, with varying strengths.
- **Special:** Shows interplay of technical, social, and economic forces.

Information Networks: The World Wide Web

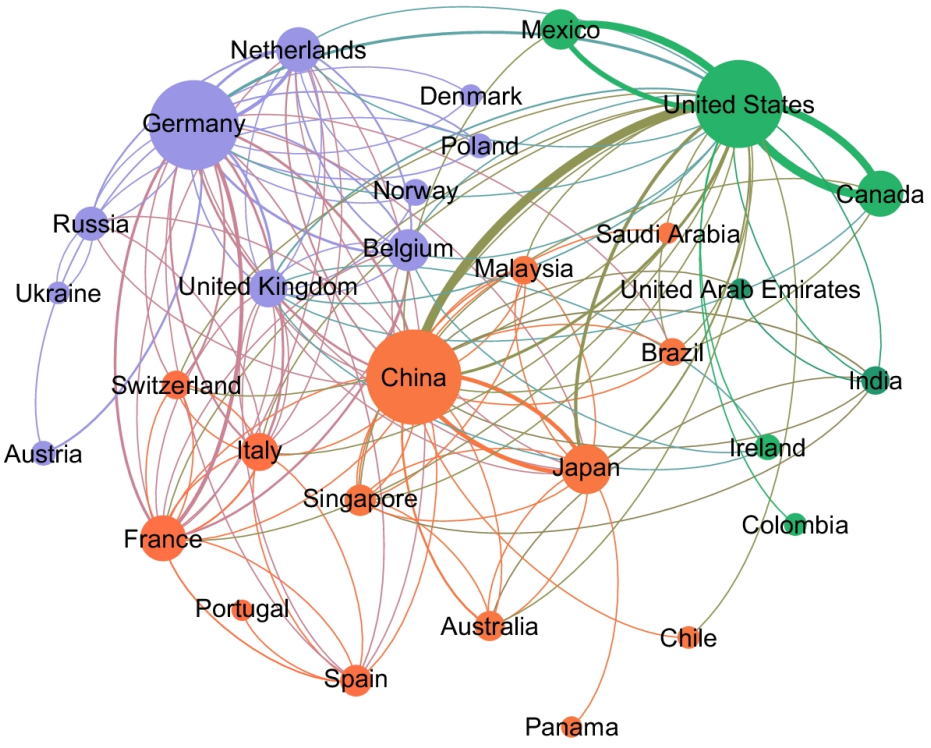
Universal Linked Information System



- **What:** Hyperlink network of web pages/sites.
- **Vertices:** Pages (or domains).
- **Edges:** Hyperlinks.
- **Directed?** Yes (A links to B).
- **Special:** Scale-free in-degree; PageRank; massive, evolving.

Economic Networks: World Trade Network

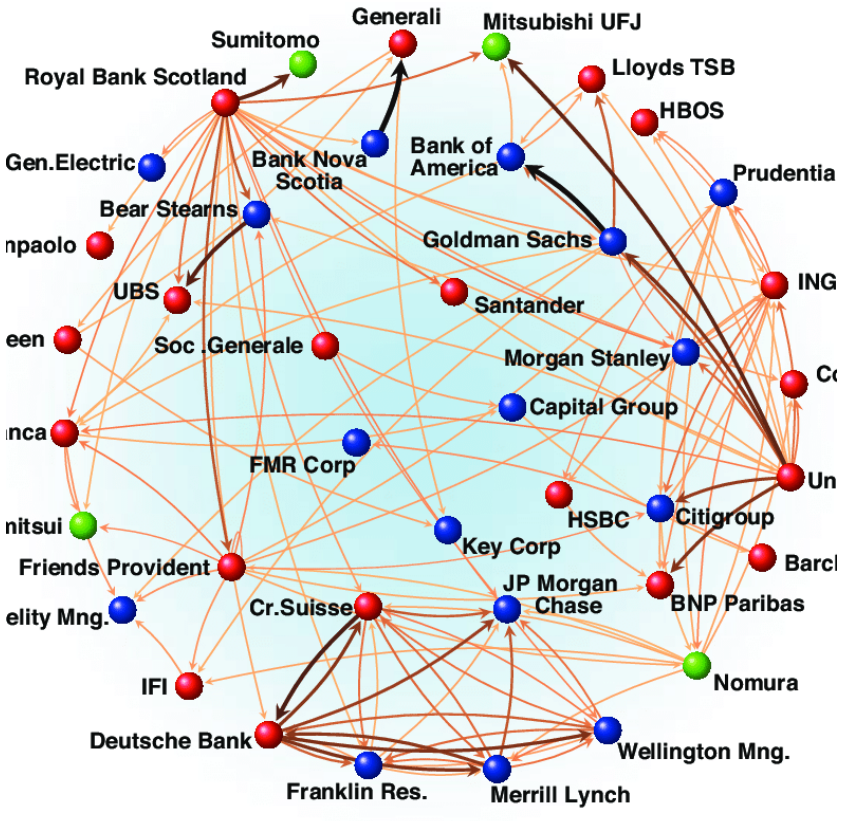
Geographical Market Flow



- **What:** International trade flows.
- **Vertices:** Countries/economies.
- **Edges:** Export/import links (weighted by volume/value).
- **Directed?** Yes (exports A→B vs B→A differ).
- **Special:** Weighted, multiplex by commodity; core–periphery structure.

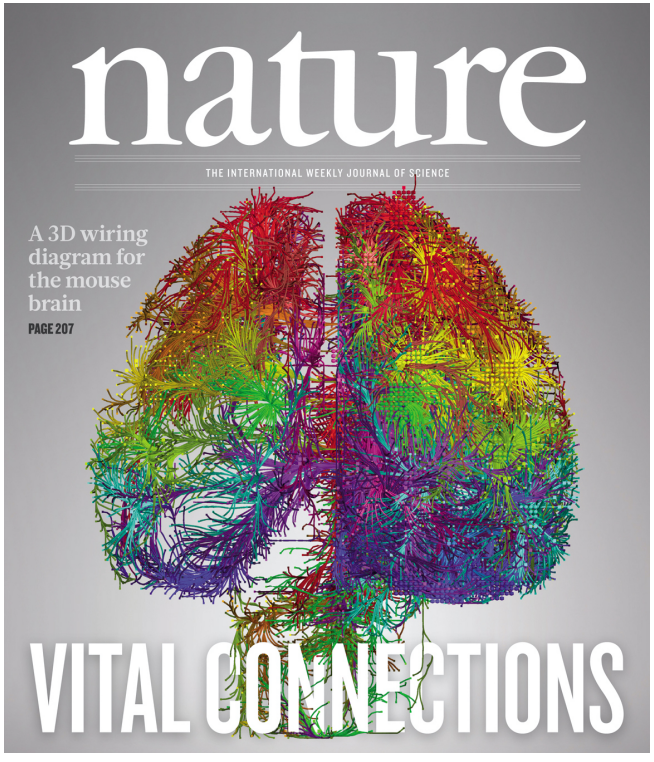
Economic Networks: Financial Networks

Institutional Financial Flow



- **What:** Financial exposures and transactions across institutions.
- **Vertices:** Banks/funds/financial entities.
- **Edges:** Loans, derivatives, interbank payments (weighted).
- **Directed?** Often directed (obligations/exposures).
- **Special:** Systemic risk, contagion/cascades, clearing networks.

Biological Networks: Brain Neurons Network



- **What:** Neural connectivity (micro/meso/macro scale).
- **Vertices:** Neurons or brain regions.
- **Edges:** Synapses/tracts (structural) or correlations (functional).
- **Directed?** Structural often directed; functional may be undirected/weighted.
- **Special:** Small-world, modularity, rich-club hubs.

SPREAD OF H1N1 VIRUS

- First successful real-time prediction of a pandemic (H1N1, 2009).
- Based on data from the worldwide transportation network.
- Predicted that H1N1 would peak in **October 2009**, while seasonal flu was expected in **January–February**.
- Vaccines, available only by November 2009, had little impact.
- This case illustrates the **power of network science** in addressing global challenges.

The course outline

Part I: Network Analysis

The course outline

Part I: Network Analysis

1. Introduction to Graph Theory

The course outline

Part I: Network Analysis

1. Introduction to Graph Theory
2. Network Models
   - 2.1 Random Networks
   - 2.2 Scale - Free Networks

The course outline

Part I: Network Analysis

1. Introduction to Graph Theory
2. Network Models
   - 2.1 Random Networks
   - 2.2 Scale - Free Networks
3. Centrality Measures

The course outline

Part I: Network Analysis

1. Introduction to Graph Theory
2. Network Models
   - 2.1 Random Networks
   - 2.2 Scale - Free Networks
3. Centrality Measures
4. Network Analysis & Community Detection

The course outline

Part I: Network Analysis

1. Introduction to Graph Theory
2. Network Models
   - 2.1 Random Networks
   - 2.2 Scale - Free Networks
3. Centrality Measures
4. Network Analysis & Community Detection

Part II: Network Applications

The course outline

Part I: Network Analysis

1. Introduction to Graph Theory
2. Network Models
   - 2.1 Random Networks
   - 2.2 Scale - Free Networks
3. Centrality Measures
4. Network Analysis & Community Detection

Part II: Network Applications

5. Social Networks

The course outline

Part I: Network Analysis

1. Introduction to Graph Theory
2. Network Models
   - 2.1 Random Networks
   - 2.2 Scale - Free Networks
3. Centrality Measures
4. Network Analysis & Community Detection

Part II: Network Applications

5. Social Networks
6. Economic and Financial Networks

What is graph theory?

A basic mathematical model

We will discuss useful mathematical abstractions for Network Science.

For most of the course we use undirected graphs:

- $G = (V, E)$ a graph
- V set of vertices (nodes); often $[N] := \{1, \ldots, n\}$ for some $N \in \mathbb{N}$
- E set of edges (links, pairs of nodes); $E \subseteq [N] \times [N]$

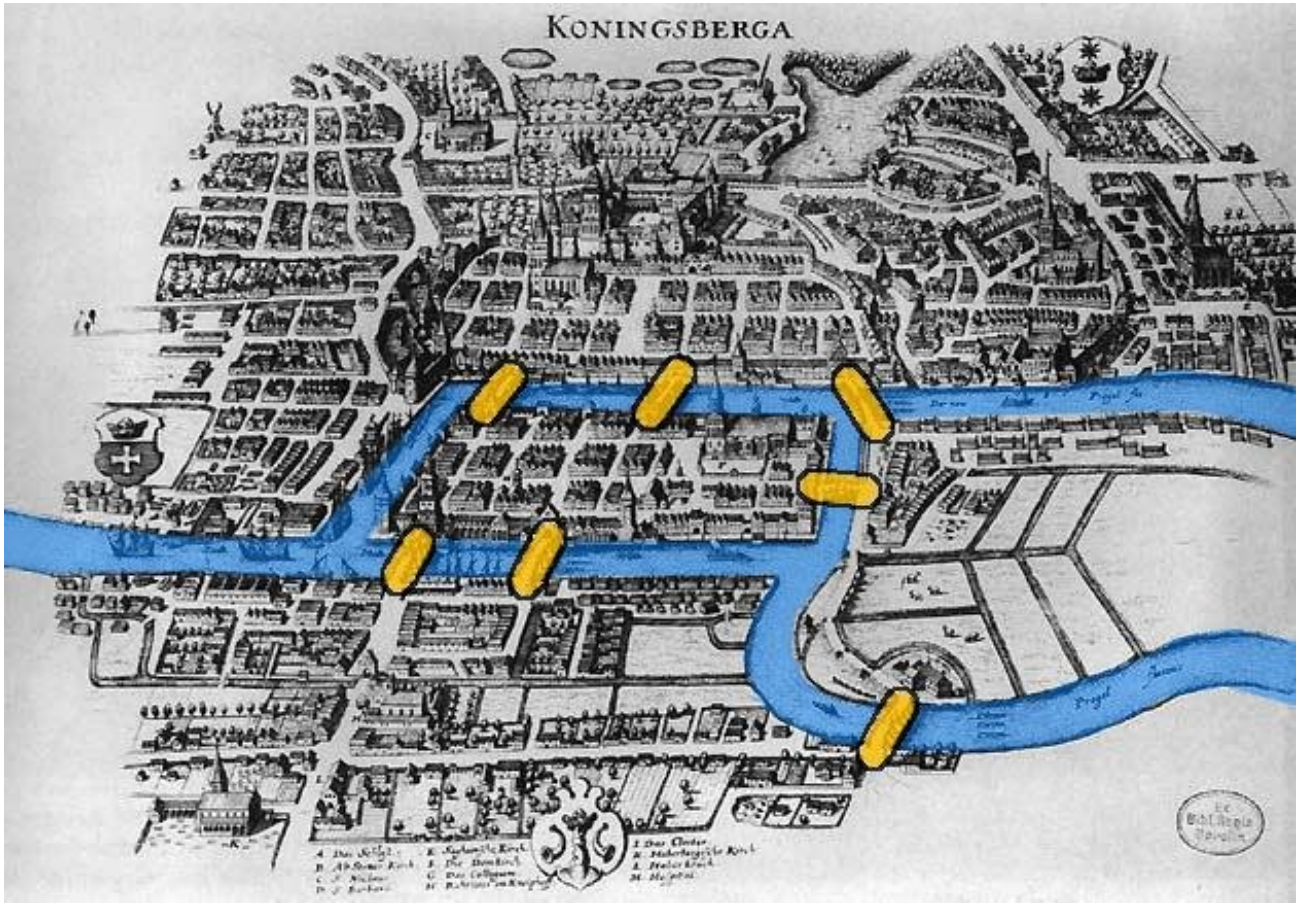
e.g. $G = (\{1, 2, 3, 4\}, \{12, 13, 14, 23, 24\})$

Note

Throughout the course, $N = |V|$ denotes the number of nodes and $L = |E|$ the number of edges in the network.

The Bridges of Königsberg

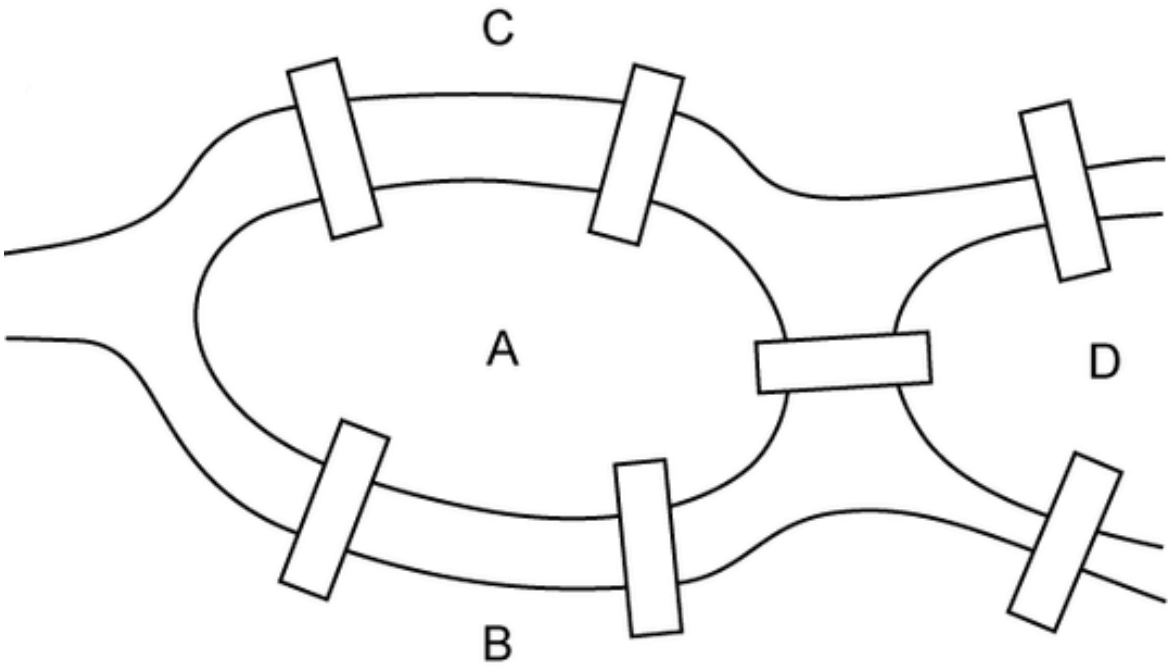
It all started with Leonhard Euler circa 1735

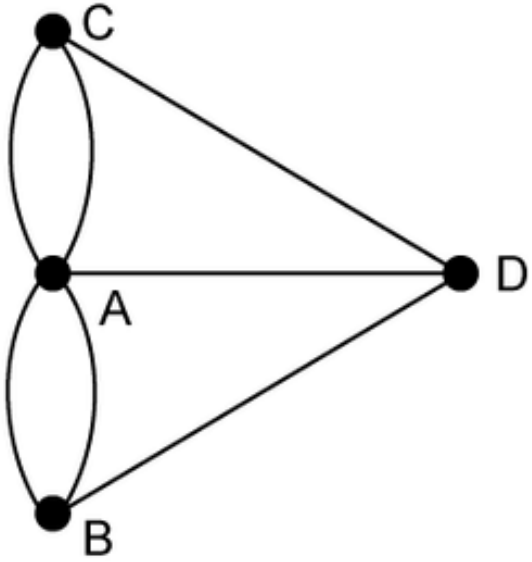


Can one walk across all seven bridges and never cross the same one twice?

The Königsberg Bridge

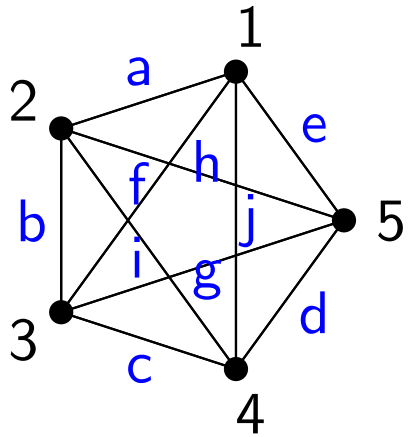
Euler's formalization





Eulerian tour

An **Eulerian tour** in a graph $G = (V, E)$ is a closed walk in G which uses each edge exactly once, but vertices may be visited multiple times.



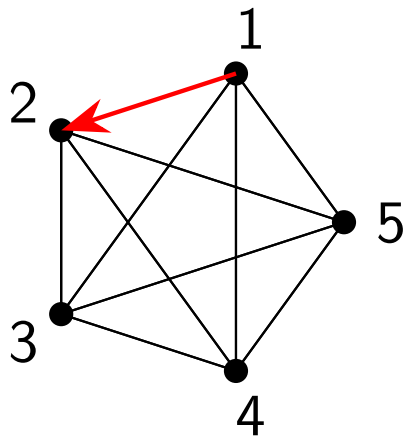
$\rightarrow$ $1a2b3c4d5e1f3g5h2i4j1$

Example (Chinese Postman Problem)

- Route inspection: postal delivery, garbage collection, snow plowing.
- DNA sequencing: Eulerian paths in de Bruijn graphs.
- Network monitoring and flow design.

Eulerian tour

An **Eulerian tour** in a graph $G = (V, E)$ is a closed walk in G which uses each edge exactly once, but vertices may be visited multiple times.

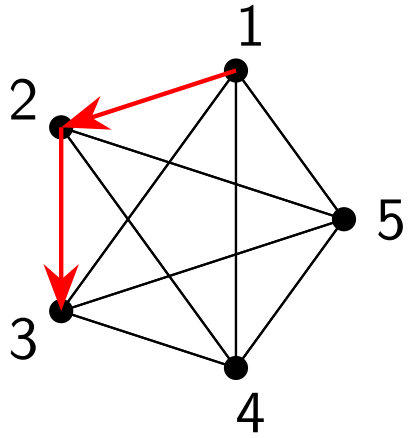


Example (Chinese Postman Problem)

- Route inspection: postal delivery, garbage collection, snow plowing.
- DNA sequencing: Eulerian paths in de Bruijn graphs.
- Network monitoring and flow design.

Eulerian tour

An **Eulerian tour** in a graph $G = (V, E)$ is a closed walk in G which uses each edge exactly once, but vertices may be visited multiple times.

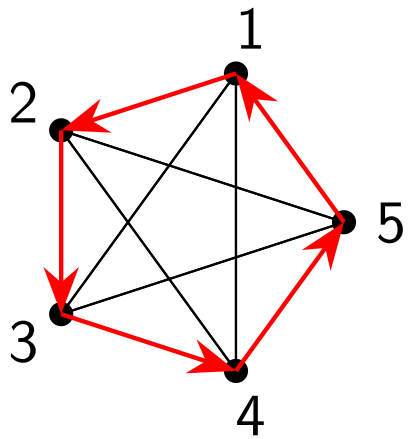


Example (Chinese Postman Problem)

- Route inspection: postal delivery, garbage collection, snow plowing.
- DNA sequencing: Eulerian paths in de Bruijn graphs.
- Network monitoring and flow design.

Eulerian tour

An **Eulerian tour** in a graph $G = (V, E)$ is a closed walk in G which uses each edge exactly once, but vertices may be visited multiple times.

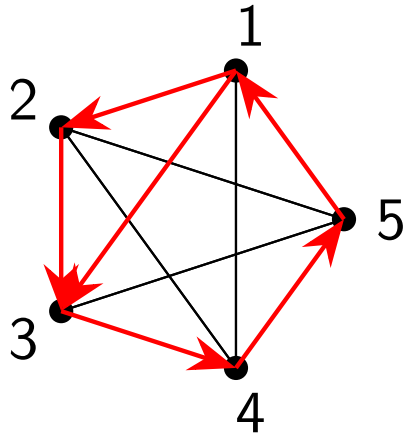


Example (Chinese Postman Problem)

- Route inspection: postal delivery, garbage collection, snow plowing.
- DNA sequencing: Eulerian paths in de Bruijn graphs.
- Network monitoring and flow design.

Eulerian tour

An **Eulerian tour** in a graph $G = (V, E)$ is a closed walk in G which uses each edge exactly once, but vertices may be visited multiple times.

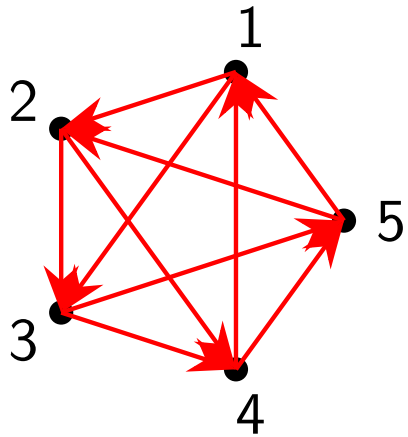


Example (Chinese Postman Problem)

- Route inspection: postal delivery, garbage collection, snow plowing.
- DNA sequencing: Eulerian paths in de Bruijn graphs.
- Network monitoring and flow design.

Eulerian tour

An **Eulerian tour** in a graph $G = (V, E)$ is a closed walk in G which uses each edge exactly once, but vertices may be visited multiple times.



A graph is Eulerian if it admits an Eulerian tour.

Example (Chinese Postman Problem)

- Route inspection: postal delivery, garbage collection, snow plowing.
- DNA sequencing: Eulerian paths in de Bruijn graphs.
- Network monitoring and flow design.

Euler's Theorem

Definition

The degree $\deg(v)$ of a vertex v is the number of incident edges.

Theorem

A connected graph is Eulerian if and only if every vertex has even degree.

- If there is an Eulerian Tour, when traversing a node we use two edges every time. The total number of edges incident to a vertex must be even.

Euler's Theorem

Definition

The degree $\deg(v)$ of a vertex v is the number of incident edges.

Theorem

A connected graph is Eulerian if and only if every vertex has even degree.

- If there is an Eulerian Tour, when traversing a node we use two edges every time. The total number of edges incident to a vertex must be even.
- If all vertices have even degree, we can always extend a walk:
  - Start at any vertex and keep following unused edges.
  - At each step, if you arrive at a vertex with unused incident edges, you can leave again (since degrees are even).
  - The walk closes when you return to the starting vertex.

  (There is a way to pick a walk so that no unused edges remain.)

The Königsberg Bridge

Theorem

A connected graph contains an Eulerian (open) walk if and only if there are exactly two nodes with odd number of edges.

If that is the case, these are the starting and ending points.

- If all but two of the degrees are even, then add an extra edge joining them. Now all vertices have even degree: there is an Eulerian Tour in the resulting graph. Cut the Tour by the added edge to obtain an Eulerian walk in G.

The Königsberg Bridge

Theorem

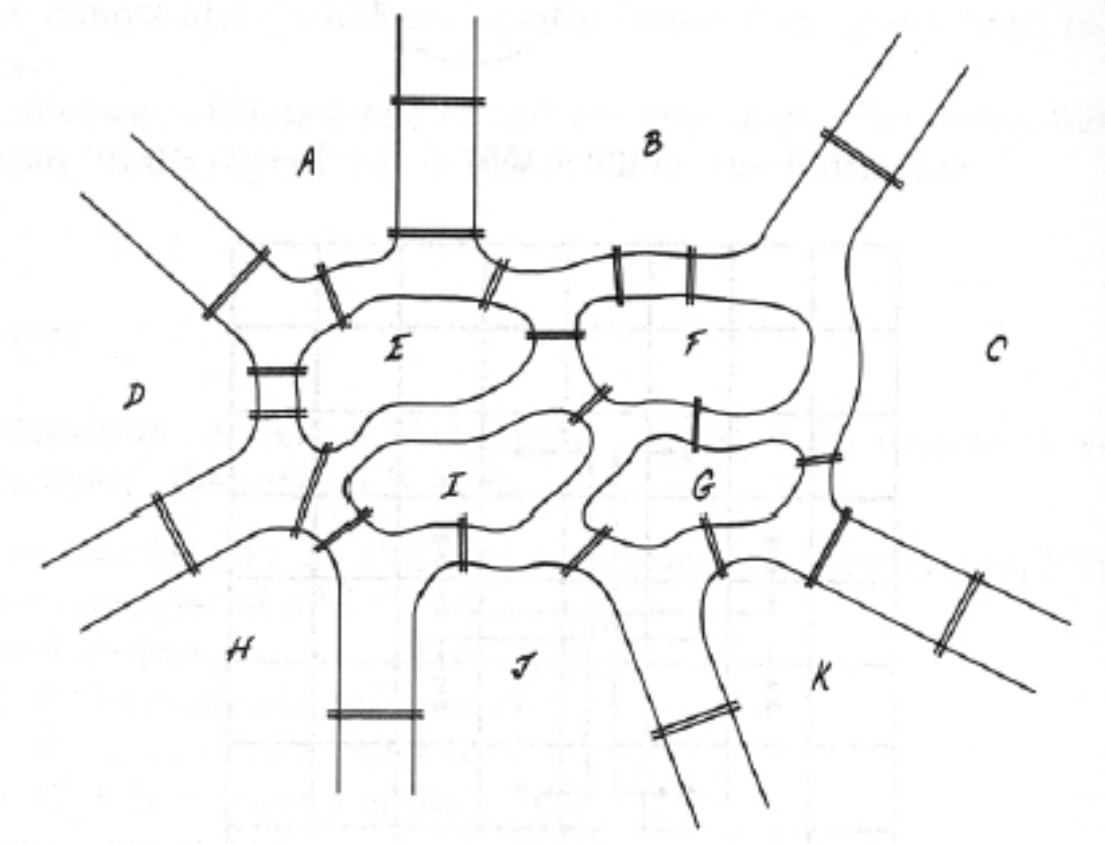
A connected graph contains an Eulerian (open) walk if and only if there are exactly two nodes with odd number of edges.

If that is the case, these are the starting and ending points.

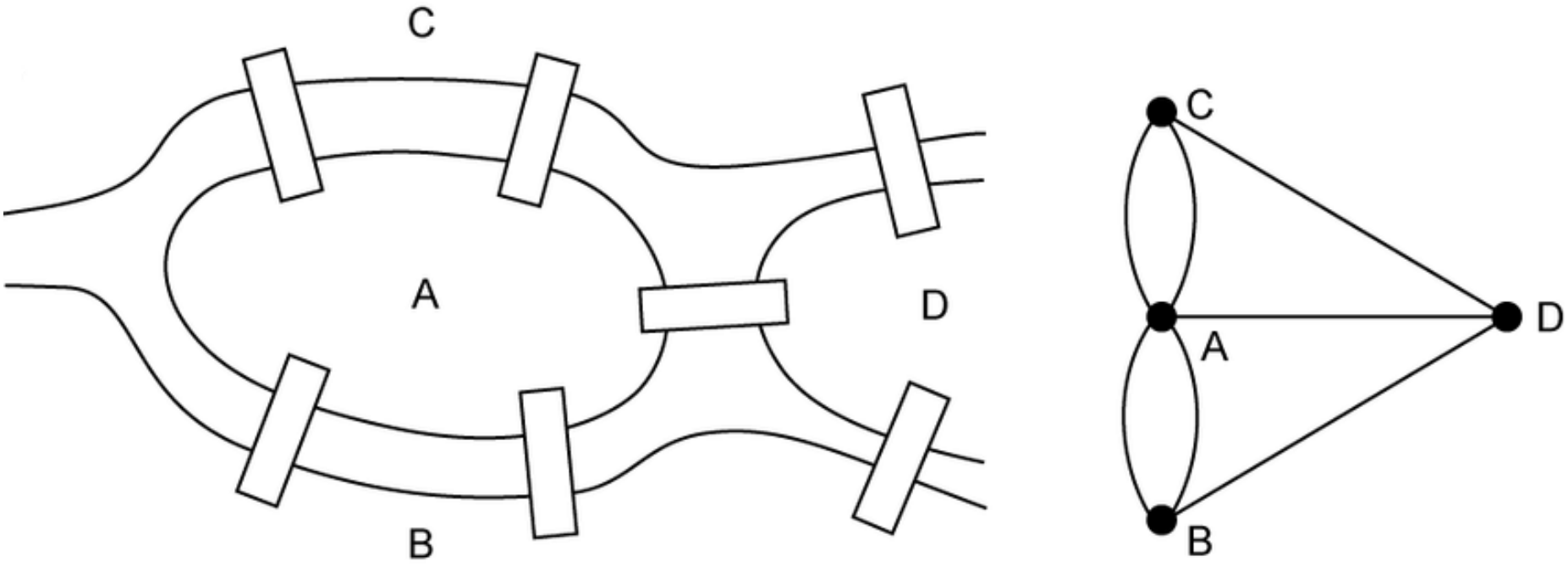
- If all but two of the degrees are even, then add an extra edge joining them. Now all vertices have even degree: there is an Eulerian Tour in the resulting graph. Cut the Tour by the added edge to obtain an Eulerian walk in G.
- If there is an Eulerian walk starting at u and ending at v, we can complete it to an Eulerian Tour by adding the edge uv. In the resulting graph all vertices have even degree.

Exercise 1

Can you find an Eulerian Path in the following graph? If so, draw such path.



The Königsberg Bridge: Coda



Four vertices of odd degree: no Eulerian Tour, no Eulerian walk.

Google Colab: your coding environment

- Google Colab = online Jupyter notebook.
- Runs entirely in your browser: no installation needed.
- You just need a Google account (UPF).
- You can run small Python code snippets.
- We use it to explore networks hands-on. No serious coding needed.

To do:

- Go to Google Colab and create account if needed.
- Appendix A in "A First Course in Network Science" may be useful.

First steps with NetworkX in Colab

- NetworkX is a Python library for studying graphs.
- Lets us build graphs, compute statistics, and draw them.

Please check our colab1.

- You can modify edges/nodes and see the result immediately.
- Already enough to test Eulerian walks!

NetworkX commands to remember

- `import networkx as nx` – the canonical import
- `G = nx.Graph()` or `nx.DiGraph()` – create an (undirected or directed) graph
- `G.add_node(n)` / `G.add_nodes_from([...])` – add node(s)
- `G.add_edge(u, v)` / `G.add_edges_from([...])` – add edge(s)
- `nx.shortest_path(G, a, b)` – get shortest path
- `nx.degree(G, n)` – degree of node n
- `nx.draw(G, with_labels=True)` – draw graph (simple plotting)