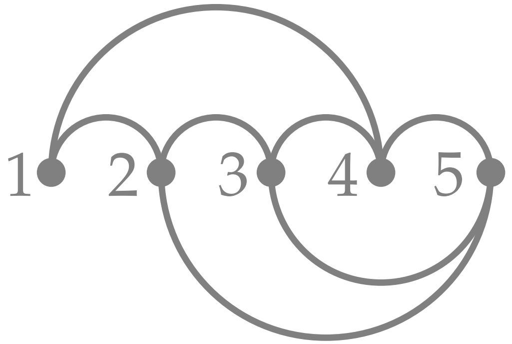
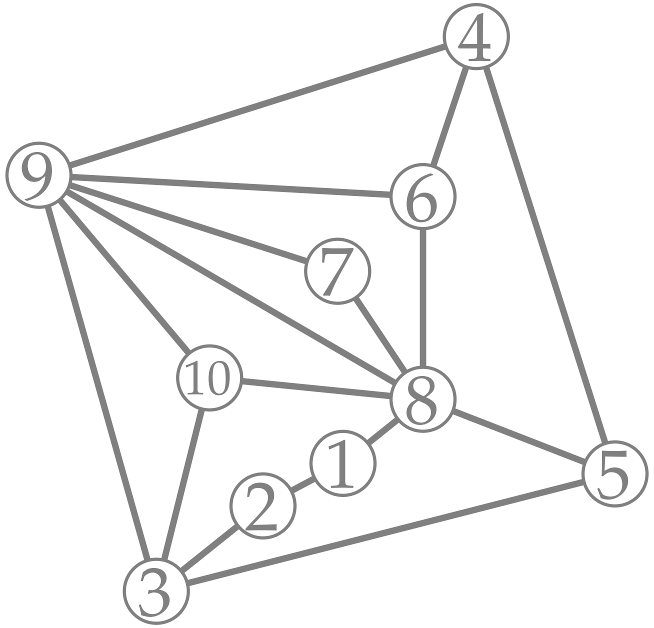
Universität Trier

Visualization of Graphs



Lecture 1:
The Graph Visualization Problem



Part I:
Organizational & Overview

Philipp Kindermann

Organizational

Lectures:

- Pre-recorded videos (as you see here)

Organizational

Lectures:

- Pre-recorded videos (as you see here)
- Release date: One week before the lecture

Organizational

Lectures:

- Pre-recorded videos (as you see here)
- Release date: One week before the lecture
- Tue 08:30 – 10:00: Questions/Discussion in BigBlueButton

Organizational

Lectures:

- Pre-recorded videos (as you see here)
- Release date: One week before the lecture
- Tue 08:30 – 10:00: Questions/Discussion in BigBlueButton
- Questions/Tasks in the Videos

Organizational

Lectures:

- Pre-recorded videos (as you see here)
- Release date: One week before the lecture
- Tue 08:30 – 10:00: Questions/Discussion in BigBlueButton
- Questions/Tasks in the Videos

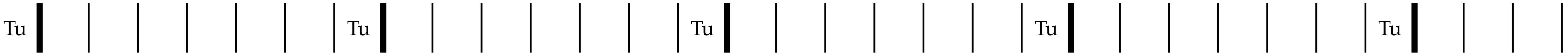
Tutorials:

- One sheet per lecture

Organizational

Lectures:

- Pre-recorded videos (as you see here)
- Release date: One week before the lecture
- Tue 08:30 – 10:00: Questions/Discussion in BigBlueButton
- Questions/Tasks in the Videos



Tutorials:

- One sheet per lecture

Organizational

Lectures:

- Pre-recorded videos (as you see here)
- Release date: One week before the lecture
- Tue 08:30 – 10:00: Questions/Discussion in BigBlueButton
- Questions/Tasks in the Videos

R: Release
D: Discussion
H: Hand In

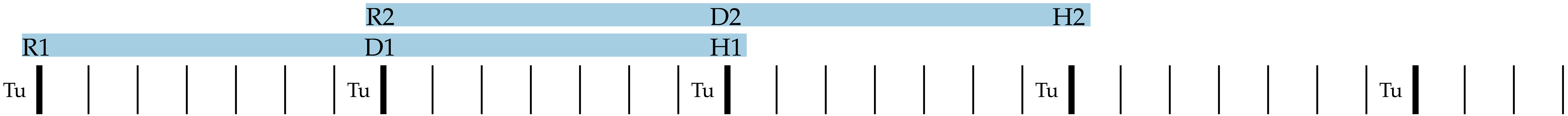
Tutorials:

- One sheet per lecture

Organizational

Lectures:

- Pre-recorded videos (as you see here)
- Release date: One week before the lecture
- Tue 08:30 – 10:00: Questions/Discussion in BigBlueButton
- Questions/Tasks in the Videos



R: Release
D: Discussion
H: Hand In

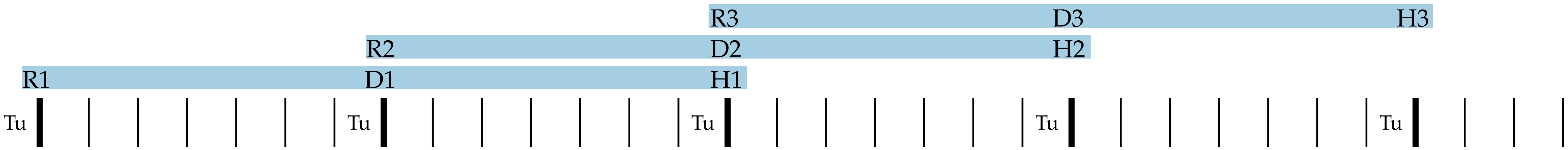
Tutorials:

- One sheet per lecture

Organizational

Lectures:

- Pre-recorded videos (as you see here)
- Release date: One week before the lecture
- Tue 08:30 – 10:00: Questions/Discussion in BigBlueButton
- Questions/Tasks in the Videos



R: Release
D: Discussion
H: Hand In

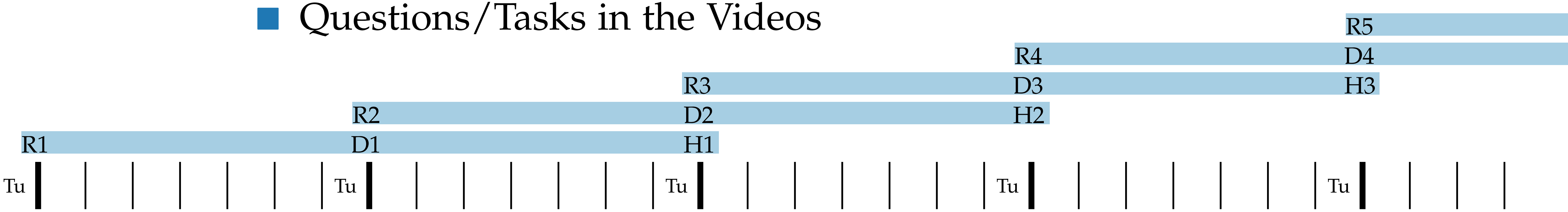
Tutorials:

- One sheet per lecture

Organizational

Lectures:

- Pre-recorded videos (as you see here)
- Release date: One week before the lecture
- Tue 08:30 – 10:00: Questions/Discussion in BigBlueButton
- Questions/Tasks in the Videos



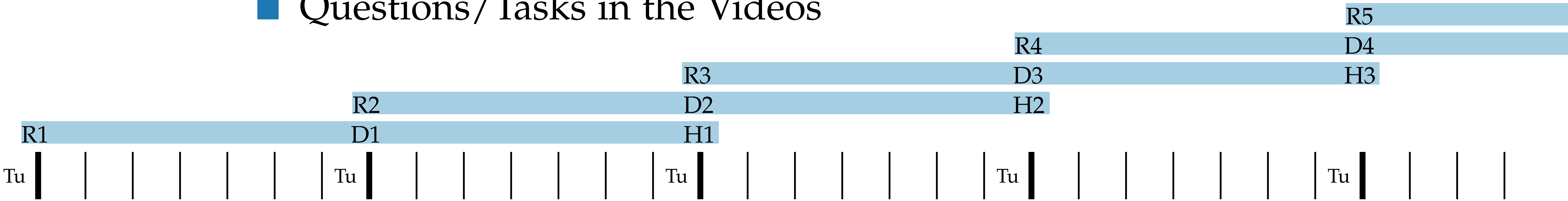
R: Release
D: Discussion
H: Hand In

Tutorials:

- One sheet per lecture

Organizational

Lectures:
- Pre-recorded videos (as you see here)
- Release date: One week before the lecture
- Tue 08:30 – 10:00: Questions/Discussion in BigBlueButton
- Questions/Tasks in the Videos



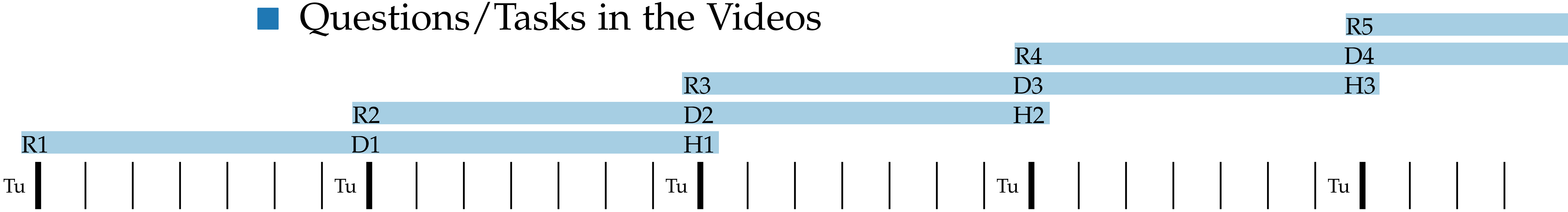
R: Release
D: Discussion
H: Hand In

Tutorials:
- One sheet per lecture
- Submit solutions online

Organizational

Lectures:

- Pre-recorded videos (as you see here)
- Release date: One week before the lecture
- Tue 08:30 – 10:00: Questions/Discussion in BigBlueButton
- Questions/Tasks in the Videos



R: Release
D: Discussion
H: Hand In

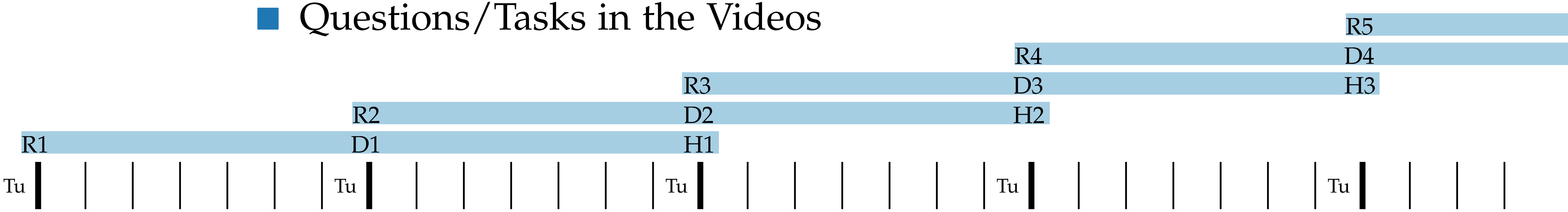
Tutorials:

- One sheet per lecture
- Submit solutions online
- Recommend LaTeX (template provided)

Organizational

Lectures:

- Pre-recorded videos (as you see here)
- Release date: One week before the lecture
- Tue 08:30 – 10:00: Questions/Discussion in BigBlueButton
- Questions/Tasks in the Videos

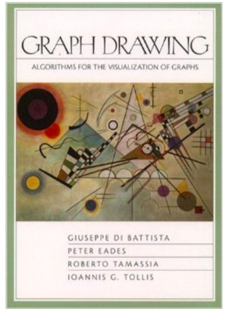


R: Release
D: Discussion
H: Hand In

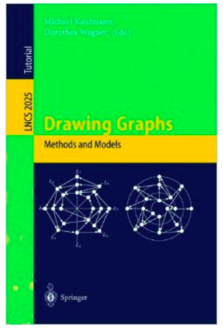
Tutorials:

- One sheet per lecture
- Submit solutions online
- Recommend LaTeX (template provided)
- Discussion and Solutions in BigBlueButton (Date: ?)

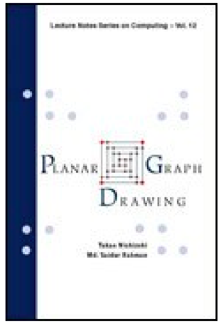
Books



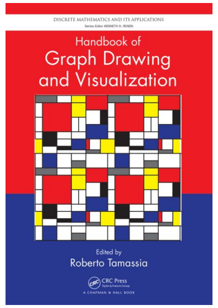
G. Di Battista, P. Eades, R. Tamassia, I. Tollis:
Graph Drawing: Algorithms for the Visualization of Graphs
Prentice Hall, 1998



M. Kaufmann, D. Wagner:
Drawing Graphs: Methods and Models
Springer, 2001



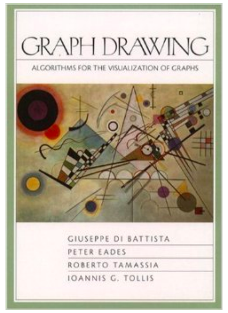
T. Nishizeki, Md. S. Rahman:
Planar Graph Drawing
World Scientific, 2004



R. Tamassia:
Handbook of Graph Drawing and Visualization
CRC Press, 2013
http://cs.brown.edu/people/rtamassi/gdhandbook/

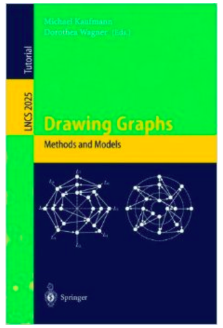
Books

[GD]



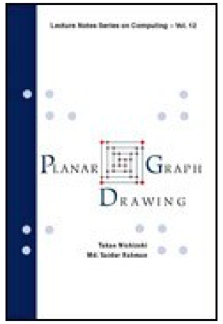
G. Di Battista, P. Eades, R. Tamassia, I. Tollis:
Graph Drawing: Algorithms for the Visualization of Graphs
Prentice Hall, 1998

[DG]



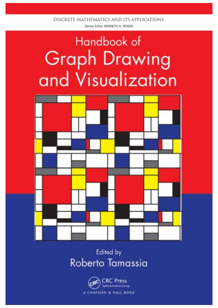
M. Kaufmann, D. Wagner:
Drawing Graphs: Methods and Models
Springer, 2001

[PGD]



T. Nishizeki, Md. S. Rahman:
Planar Graph Drawing
World Scientific, 2004

[HGDV]



R. Tamassia:
Handbook of Graph Drawing and Visualization
CRC Press, 2013
`http://cs.brown.edu/people/rtamassi/gdhandbook/`

What is this course about?

Learning objectives

What is this course about?

Learning objectives

- Overview of graph visualization

What is this course about?

Learning objectives

- Overview of graph visualization
- Improved knowledge of modeling and solving problems via graph algorithms

What is this course about?

Learning objectives

- Overview of graph visualization
- Improved knowledge of modeling and solving problems via graph algorithms

Visualization problem:

- Given a graph G, visualize it with a drawing Γ

What is this course about?

Learning objectives

- Overview of graph visualization
- Improved knowledge of modeling and solving problems via graph algorithms

Visualization problem:

- Given a graph G, visualize it with a drawing Γ

Here:

- Reducing the visualisation problem to its **algorithmic core**

What is this course about?

Learning objectives

- Overview of graph visualization
- Improved knowledge of modeling and solving problems via graph algorithms

Visualization problem:

- Given a graph G, visualize it with a drawing Γ

Here:

- Reducing the visualisation problem to its **algorithmic core**

graph class $\Rightarrow$ layout style $\Rightarrow$ algorithm $\Rightarrow$ analysis

What is this course about?

Learning objectives

- Overview of graph visualization
- Improved knowledge of modeling and solving problems via graph algorithms

Visualization problem:

- Given a graph G, visualize it with a drawing Γ

Here:

- Reducing the visualisation problem to its **algorithmic core**

$$\text{graph class} \Rightarrow \text{layout style} \Rightarrow \text{algorithm} \Rightarrow \text{analysis}$$

- modeling
- data structures

What is this course about?

Learning objectives

- Overview of graph visualization
- Improved knowledge of modeling and solving problems via graph algorithms

Visualization problem:

- Given a graph G, visualize it with a drawing Γ

Here:

- Reducing the visualisation problem to its **algorithmic core**

graph class $\Rightarrow$ layout style $\Rightarrow$ algorithm $\Rightarrow$ analysis

- modeling
- data structures

- divide & conquer, incremental
- combinatorial optimization (flows, ILPs)
- force-based algorithm

What is this course about?

Learning objectives

- Overview of graph visualization
- Improved knowledge of modeling and solving problems via graph algorithms

Visualization problem:

- Given a graph G, visualize it with a drawing Γ

Here:

- Reducing the visualisation problem to its **algorithmic core**

graph class $\Rightarrow$ layout style $\Rightarrow$ algorithm $\Rightarrow$ analysis

- modeling
- data structures
- divide & conquer, incremental
- combinatorial optimization (flows, ILPs)
- force-based algorithm
- proofs

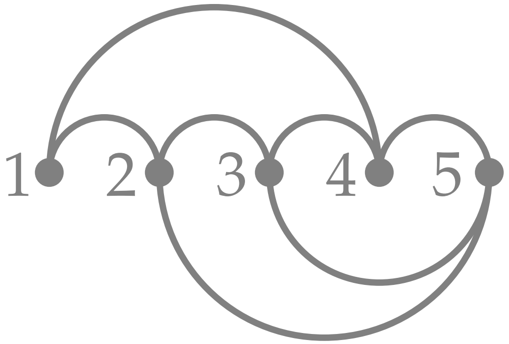
What is this course about?

Topics

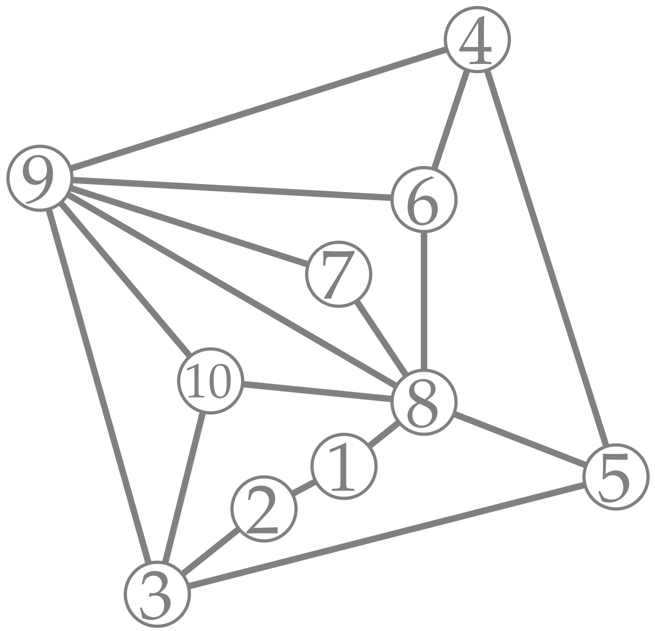
- Drawing Trees and Series-Parallel Graphs
- Straight-Line Drawings of Planar Graphs
- Orthogonal Grid Drawings
- Octilinear Drawings for Metro Maps
- Upwards Planar Drawings
- Hierarchical Layouts of Directed Graphs
- Contact Representations
- Visibility Representations
- The Crossing Lemma
- Beyond Planarity

Universität Trier

Visualization of Graphs



Lecture 1: The Graph Visualization Problem



Part II: The Layout Problem

Philipp Kindermann

Graphs and their representations

What is a graph?

- graph $G = (V, E)$
- vertices $V = \{v_1, v_2, \ldots, v_n\}$
- edge $E = \{e_1, e_2, \ldots, e_m\}$

Graphs and their representations

What is a graph?

- graph $G = (V, E)$
- vertices $V = \{v_1, v_2, \ldots, v_n\}$
- edge $E = \{e_1, e_2, \ldots, e_m\}$

Representation?

Graphs and their representations

What is a graph?

- graph $G = (V, E)$
- vertices $V = \{v_1, v_2, \ldots, v_n\}$
- edge $E = \{e_1, e_2, \ldots, e_m\}$

Representation?

- Set notation

$$V = \{v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8, v_9, v_{10}\}$$
$$E = \{\{v_1, v_2\}, \{v_1, v_8\}, \{v_2, v_3\}, \{v_3, v_5\}, \{v_3, v_9\}, \{v_3, v_{10}\}, \{v_4, v_5\}, \{v_4, v_6\}, \{v_4, v_9\}, \{v_5, v_8\}, \{v_6, v_8\}, \{v_6, v_9\}, \{v_7, v_8\}, \{v_7, v_9\}, \{v_8, v_{10}\}, \{v_9, v_{10}\}\}$$

Graphs and their representations

What is a graph?

- graph $G = (V, E)$
- vertices $V = \{v_1, v_2, \ldots, v_n\}$
- edge $E = \{e_1, e_2, \ldots, e_m\}$

Representation?

- Set notation

$$
\begin{aligned}
V &= \{v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8, v_9, v_{10}\} \\
E &= \{\{v_1, v_2\}, \{v_1, v_8\}, \{v_2, v_3\}, \{v_3, v_5\}, \{v_3, v_9\}, \\
&\quad \{v_3, v_{10}\}, \{v_4, v_5\}, \{v_4, v_6\}, \{v_4, v_9\}, \{v_5, v_8\}, \\
&\quad \{v_6, v_8\}, \{v_6, v_9\}, \{v_7, v_8\}, \{v_7, v_9\}, \{v_8, v_{10}\}, \\
&\quad \{v_9, v_{10}\}\}
\end{aligned}
$$

- Adjacency list

v_1: v_2, v_8
v_2: v_1, v_3
v_3: v_2, v_5, v_9, v_{10}
v_4: v_5, v_6, v_9
v_5: v_3, v_4, v_8
v_6: v_4, v_8, v_9
v_7: v_8, v_9
v_8: $v_1, v_5, v_6, v_7, v_9, v_{10}$
v_9: $v_3, v_4, v_6, v_7, v_8, v_{10}$
v_{10}: v_3, v_8, v_9

Graphs and their representations

What is a graph?

- graph $G = (V, E)$
- vertices $V = \{v_1, v_2, \ldots, v_n\}$
- edge $E = \{e_1, e_2, \ldots, e_m\}$

Representation?

- Set notation

$$V = \{v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8, v_9, v_{10}\}$$
$$E = \{\{v_1, v_2\}, \{v_1, v_8\}, \{v_2, v_3\}, \{v_3, v_5\}, \{v_3, v_9\}, \{v_3, v_{10}\}, \{v_4, v_5\}, \{v_4, v_6\}, \{v_4, v_9\}, \{v_5, v_8\}, \{v_6, v_8\}, \{v_6, v_9\}, \{v_7, v_8\}, \{v_7, v_9\}, \{v_8, v_{10}\}, \{v_9, v_{10}\}\}$$

- Adjacency list

v_1: v_2, v_8
v_2: v_1, v_3
v_3: v_2, v_5, v_9, v_{10}
v_4: v_5, v_6, v_9
v_5: v_3, v_4, v_8
v_6: v_4, v_8, v_9
v_7: v_8, v_9
v_8: $v_1, v_5, v_6, v_7, v_9, v_{10}$
v_9: $v_3, v_4, v_6, v_7, v_8, v_{10}$
v_{10}: v_3, v_8, v_9

- Adjacency matrix

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

Graphs and their representations

What is a graph?

- graph $G = (V, E)$
- vertices $V = \{v_1, v_2, \ldots, v_n\}$
- edge $E = \{e_1, e_2, \ldots, e_m\}$

Representation?

- Set notation

$$V = \{v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8, v_9, v_{10}\}$$
$$E = \{\{v_1, v_2\}, \{v_1, v_8\}, \{v_2, v_3\}, \{v_3, v_5\}, \{v_3, v_9\}, \{v_3, v_{10}\}, \{v_4, v_5\}, \{v_4, v_6\}, \{v_4, v_9\}, \{v_5, v_8\}, \{v_6, v_8\}, \{v_6, v_9\}, \{v_7, v_8\}, \{v_7, v_9\}, \{v_8, v_{10}\}, \{v_9, v_{10}\}\}$$

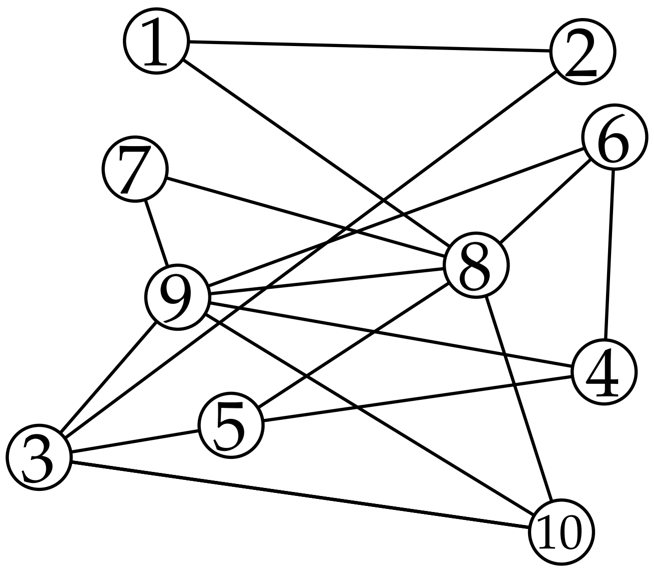
- Adjacency list

v_1: v_2, v_8
v_2: v_1, v_3
v_3: v_2, v_5, v_9, v_{10}
v_4: v_5, v_6, v_9
v_5: v_3, v_4, v_8
v_6: v_4, v_8, v_9
v_7: v_8, v_9
v_8: $v_1, v_5, v_6, v_7, v_9, v_{10}$
v_9: $v_3, v_4, v_6, v_7, v_8, v_{10}$
v_{10}: v_3, v_8, v_9

- Adjacency matrix

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

- Drawing



Graphs and their representations

What is a graph?

- graph $G = (V, E)$
- vertices $V = \{v_1, v_2, \ldots, v_n\}$
- edge $E = \{e_1, e_2, \ldots, e_m\}$

Representation?

- Set notation

$$V = \{v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8, v_9, v_{10}\}$$
$$E = \{\{v_1, v_2\}, \{v_1, v_8\}, \{v_2, v_3\}, \{v_3, v_5\}, \{v_3, v_9\}, \{v_3, v_{10}\}, \{v_4, v_5\}, \{v_4, v_6\}, \{v_4, v_9\}, \{v_5, v_8\}, \{v_6, v_8\}, \{v_6, v_9\}, \{v_7, v_8\}, \{v_7, v_9\}, \{v_8, v_{10}\}, \{v_9, v_{10}\}\}$$

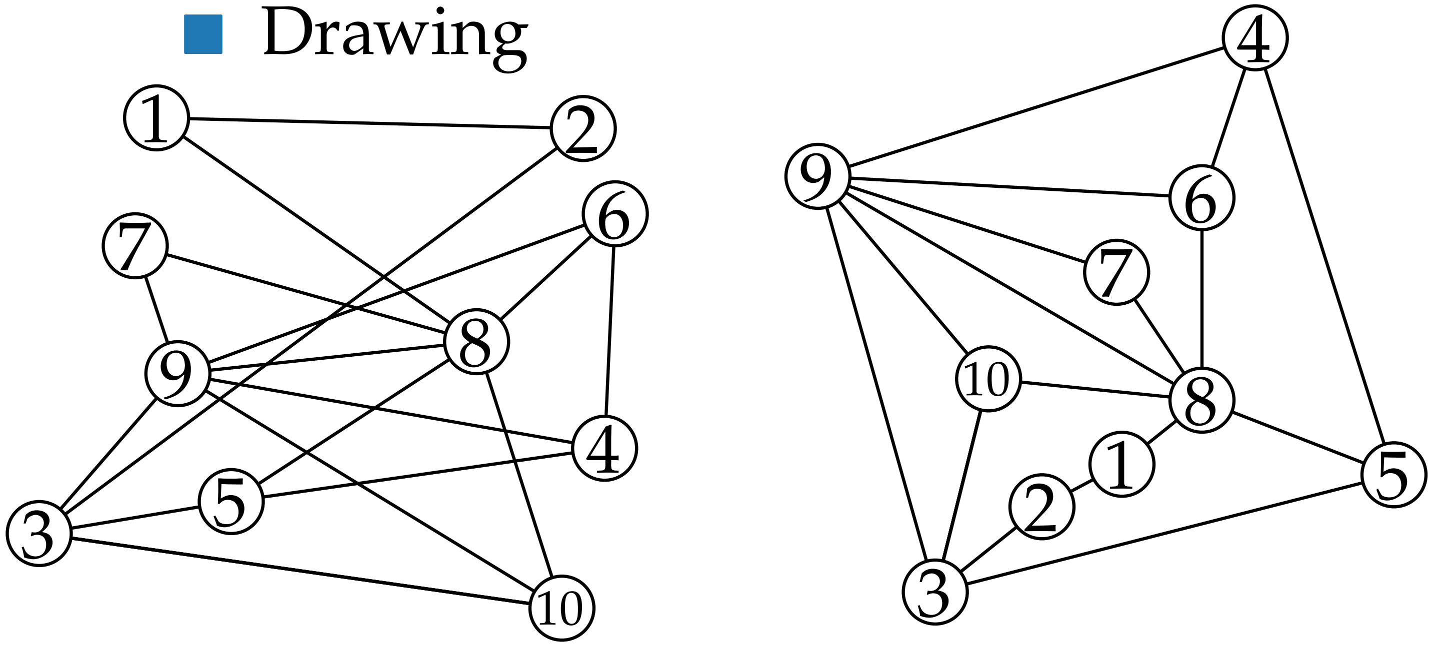
- Adjacency list

v_1: v_2, v_8
v_2: v_1, v_3
v_3: v_2, v_5, v_9, v_{10}
v_4: v_5, v_6, v_9
v_5: v_3, v_4, v_8
v_6: v_4, v_8, v_9
v_7: v_8, v_9
v_8: $v_1, v_5, v_6, v_7, v_9, v_{10}$
v_9: $v_3, v_4, v_6, v_7, v_8, v_{10}$
v_{10}: v_3, v_8, v_9

- Adjacency matrix

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

- Drawing



Why draw graphs?

Why draw graphs?

Graphs are a mathematical representation of
real physical and abstract networks.

Why draw graphs?

Graphs are a mathematical representation of
real physical and abstract networks.

Abstract networks

- Social networks
- Communication networks
- Phylogenetic networks
- Metabolic networks
- Class/Object Relation Digraphs (UML)
- …

Why draw graphs?

Graphs are a mathematical representation of
real physical and abstract networks.

Abstract networks

- Social networks
- Communication networks
- Phylogenetic networks
- Metabolic networks
- Class/Object Relation Digraphs (UML)
- ...

Physical networks

- Metro systems
- Road networks
- Power grids
- Telecommunication networks
- Integrated circuits
- ...

Why draw graphs?

Graphs are a mathematical representation of
real physical and abstract networks.

- **People think visually** – complex graphs are hard to grasp without good visualizations!

Why draw graphs?

Graphs are a mathematical representation of
real physical and abstract networks.

- **People think visually** – complex graphs are hard to grasp without good visualizations!
- Visualizations help with the **communication** and **explorati-on** of networks.

Why draw graphs?

Graphs are a mathematical representation of
real physical and abstract networks.

- **People think visually** – complex graphs are hard to grasp without good visualizations!
- Visualizations help with the **communication** and **explorati-on** of networks.
- Some graphs are too big to draw them by hand.

Why draw graphs?

Graphs are a mathematical representation of
real physical and abstract networks.

- **People think visually** – complex graphs are hard to grasp without good visualizations!
- Visualizations help with the **communication** and **explorati-on** of networks.
- Some graphs are too big to draw them by hand.

We need algorithms that draw graphs automatically to make networks more accessible to humans.

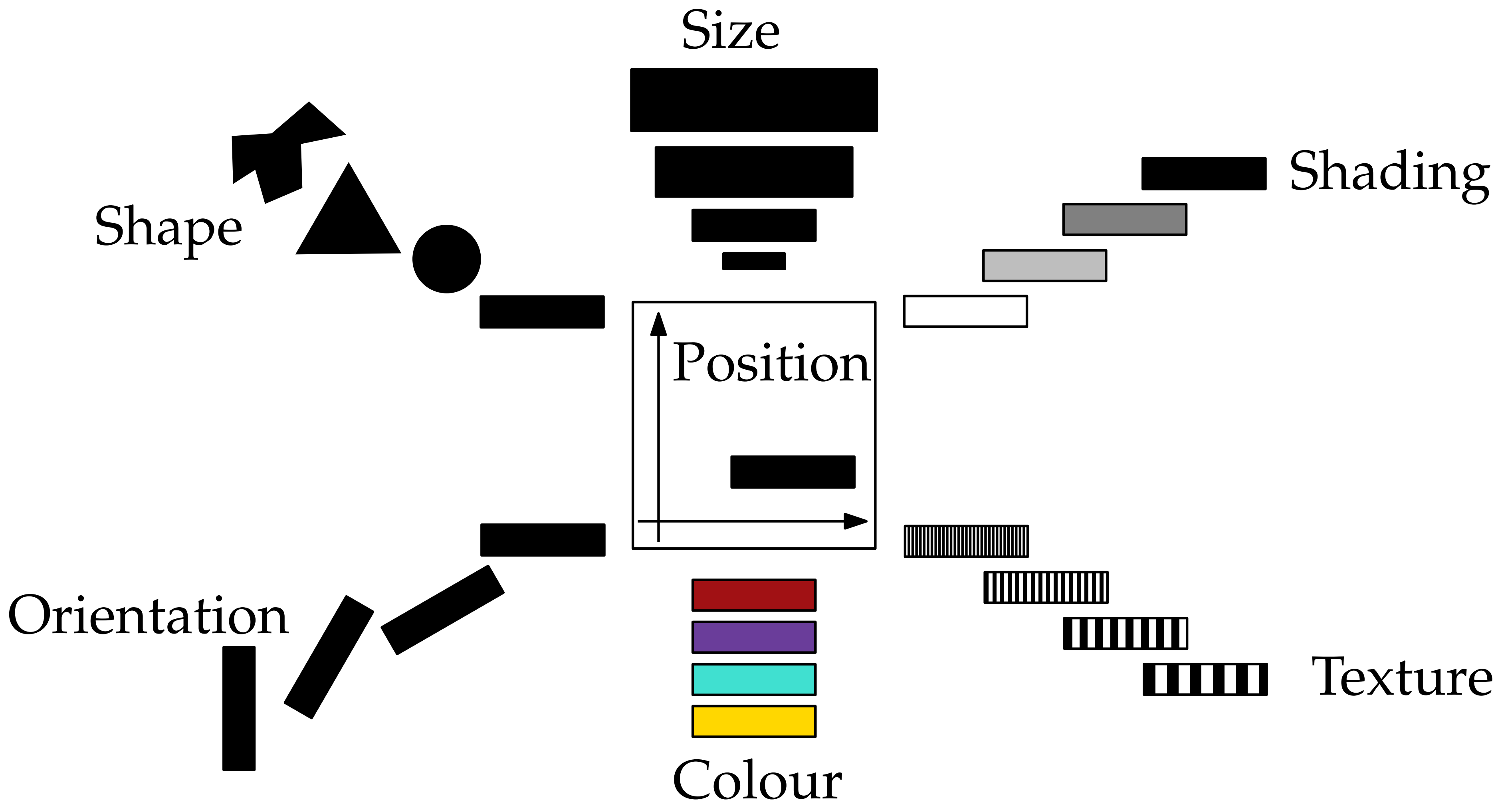
What are we interested in?

What are we interested in?

- Jacques Bertin defined visualising variables (1967)

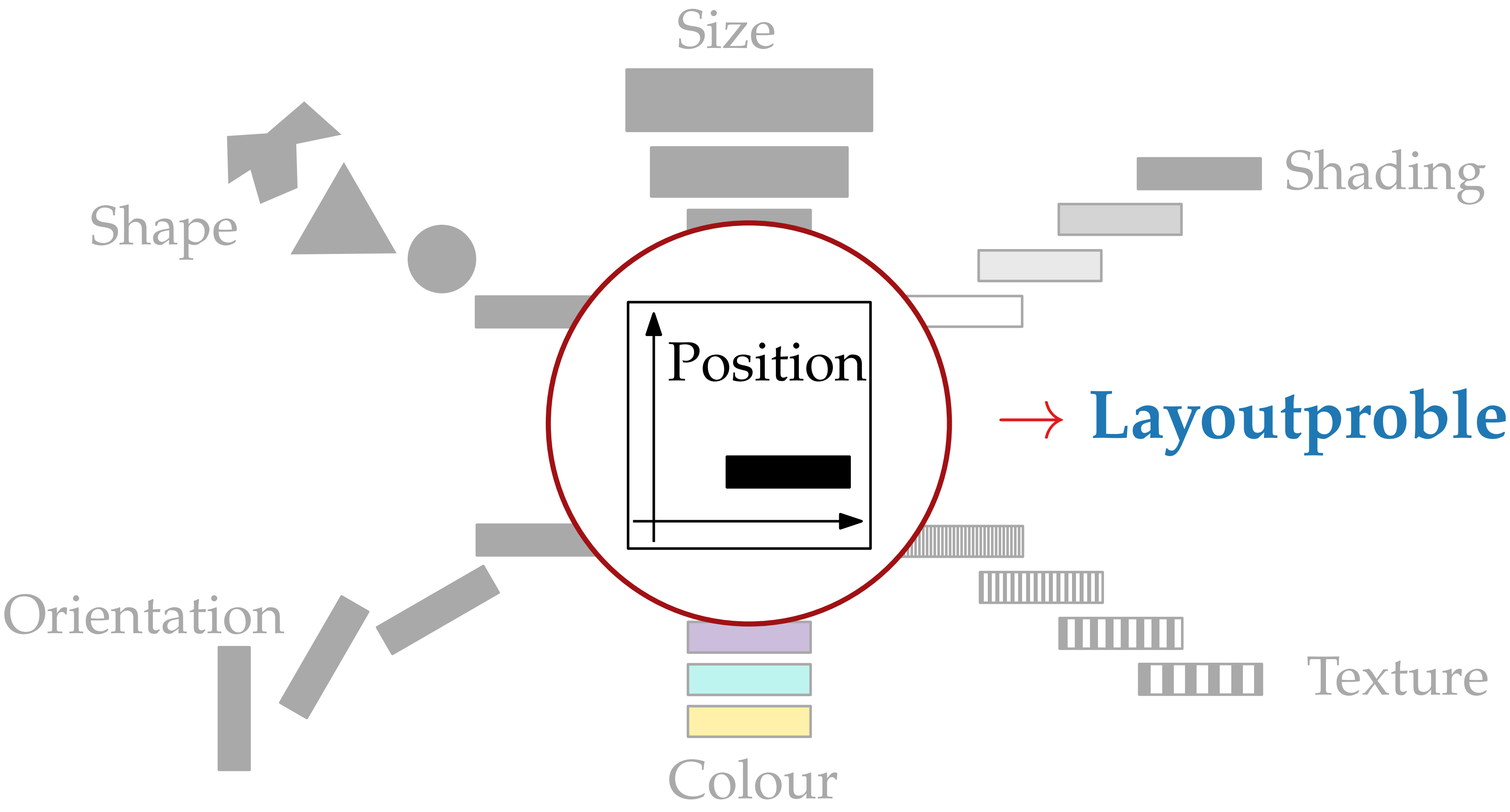
What are we interested in?

- Jacques Bertin defined visualising variables (1967)



What are we interested in?

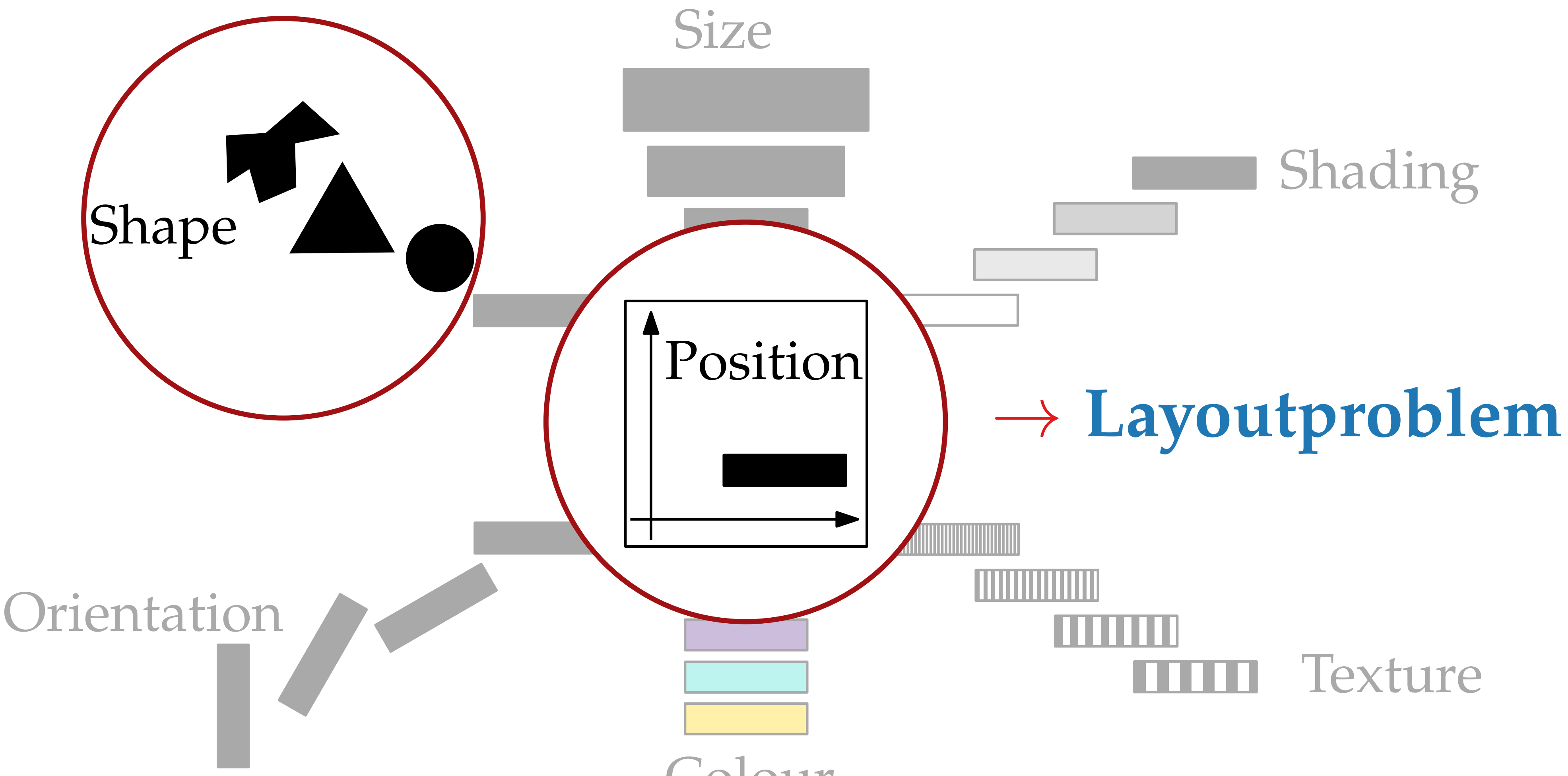
- Jacques Bertin defined visualising variables (1967)



→ **Layoutproblem**

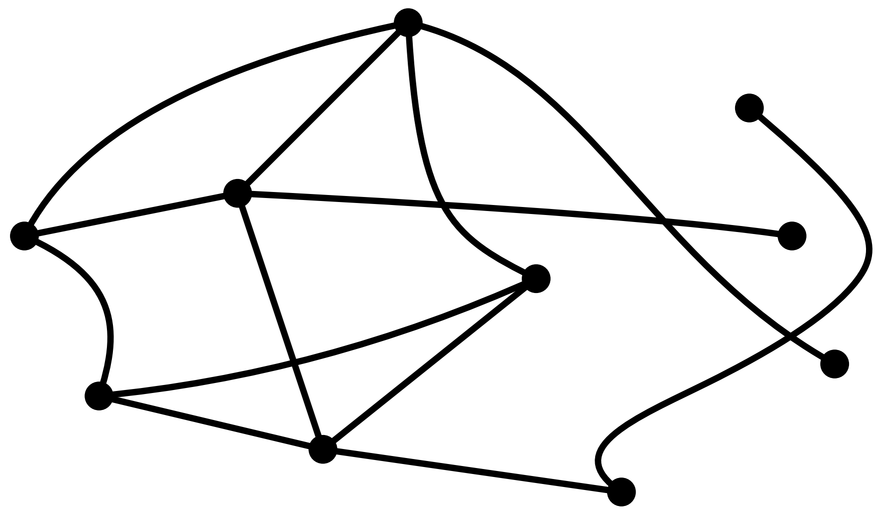
What are we interested in?

- Jacques Bertin defined visualising variables (1967)

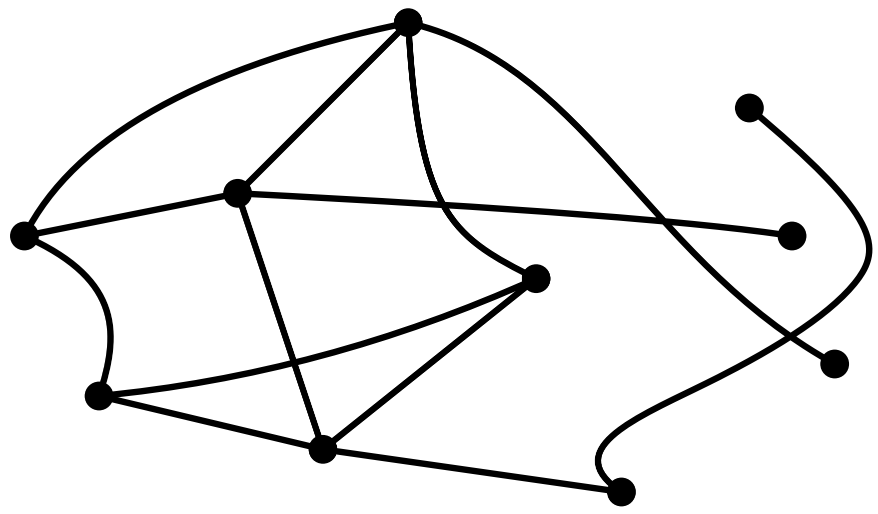


The layout problem

- *Here* restricted to the **standard representation**, so-called node-link diagrams.



The layout problem



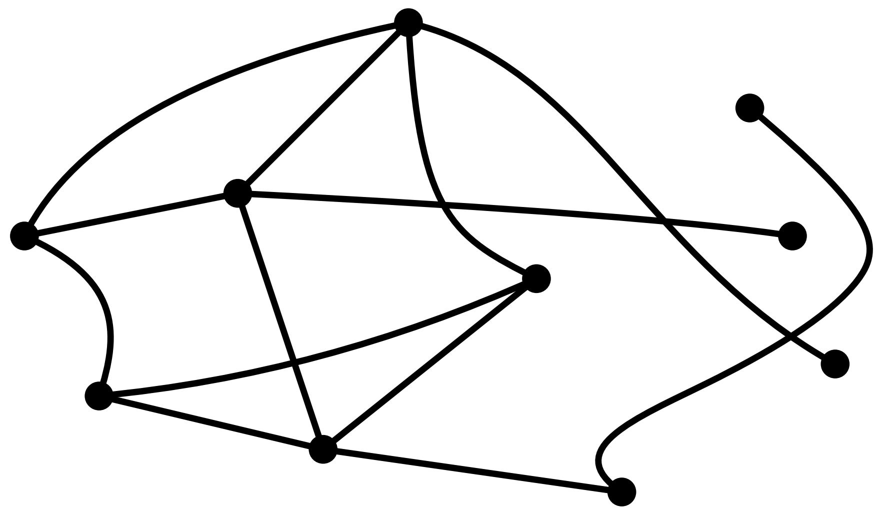
- *Here* restricted to the **standard representation**, so-called node-link diagrams.

Graph Visualization Problem

in: Graph $G = (V, E)$

out:

The layout problem



- *Here* restricted to the **standard representation**, so-called node-link diagrams.

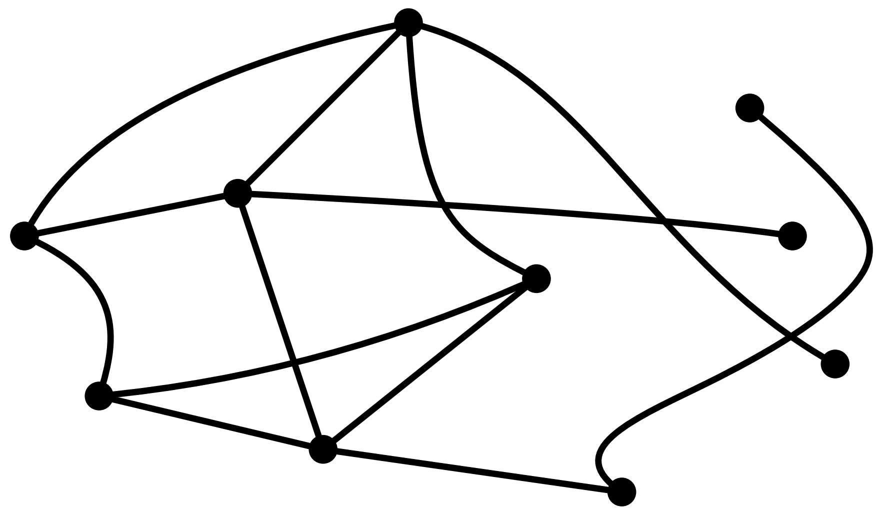
Graph Visualization Problem

in: Graph $G = (V, E)$

out: nice drawing Γ of G

- $\Gamma \colon V \to \mathbb{R}^2$, vertex $v \mapsto$ point $\Gamma(v)$
- $\Gamma \colon E \to$ curves in $\mathbb{R}^2$, edge $\{u, v\} \mapsto$ simple, open curve $\Gamma(\{u, v\})$ with endpoints $\Gamma(u)$ und $\Gamma(v)$

The layout problem?



- *Here* restricted to the **standard representation**, so-called node-link diagrams.

Graph Visualization Problem

in: Graph $G = (V, E)$

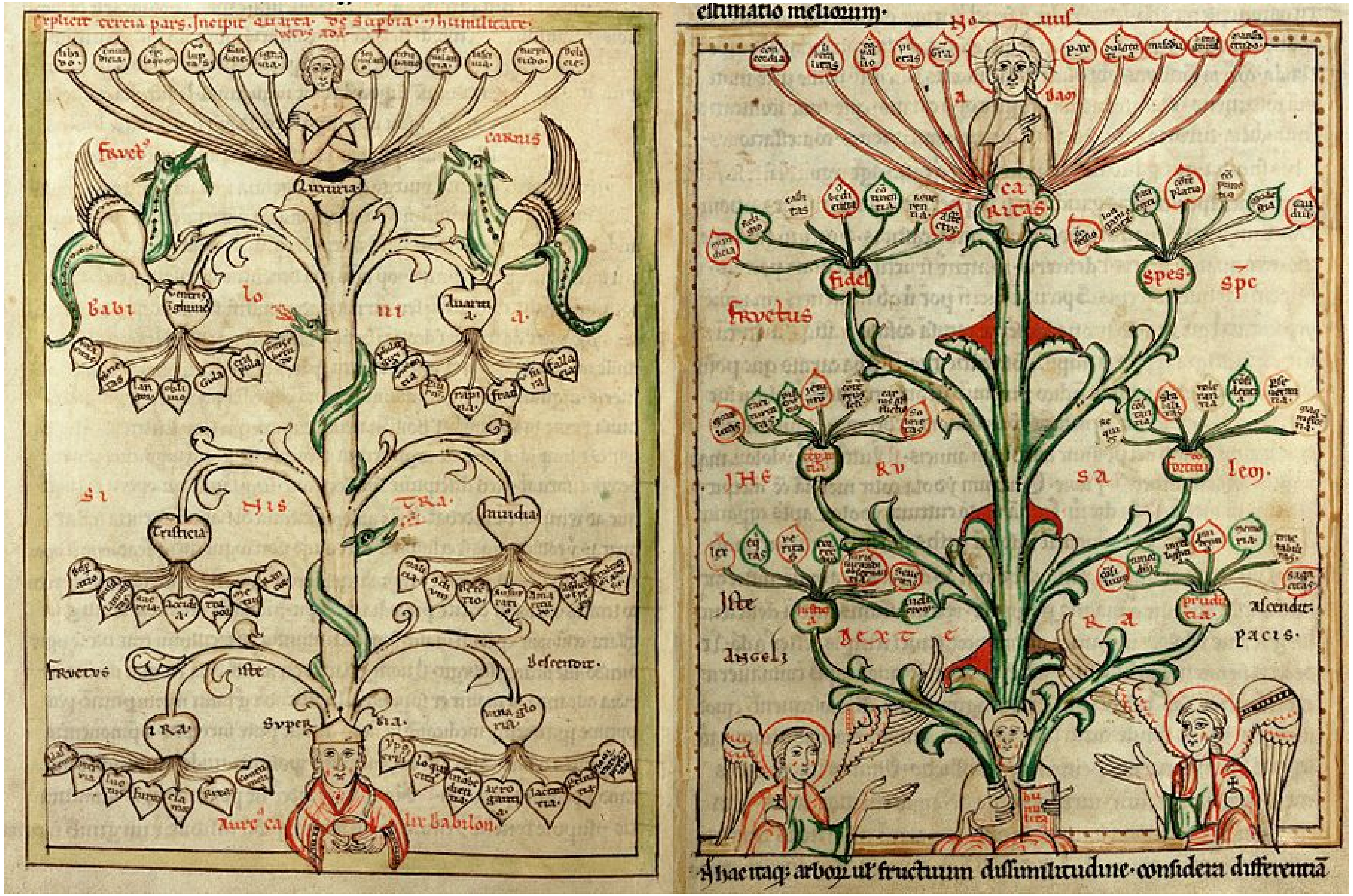
out: nice drawing Γ of G

- $\Gamma\colon V \to \mathbb{R}^2$, vertex $v \mapsto$ point $\Gamma(v)$
- $\Gamma\colon E \to$ curves in $\mathbb{R}^2$, edge $\{u, v\} \mapsto$ simple, open curve $\Gamma(\{u, v\})$ with endpoints $\Gamma(u)$ und $\Gamma(v)$

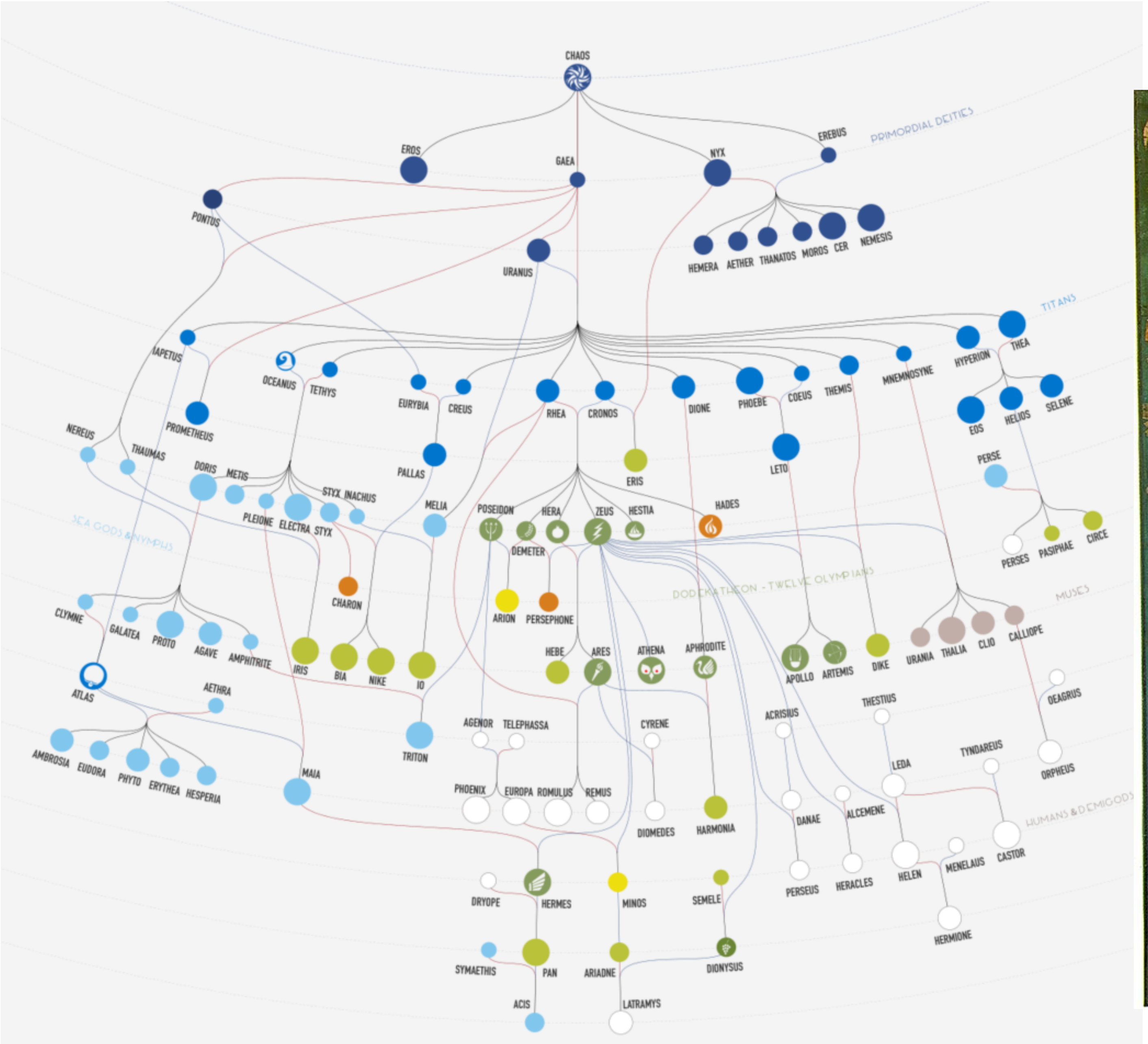
But what is a **nice** drawing?

Tree of virtues and tree of vices

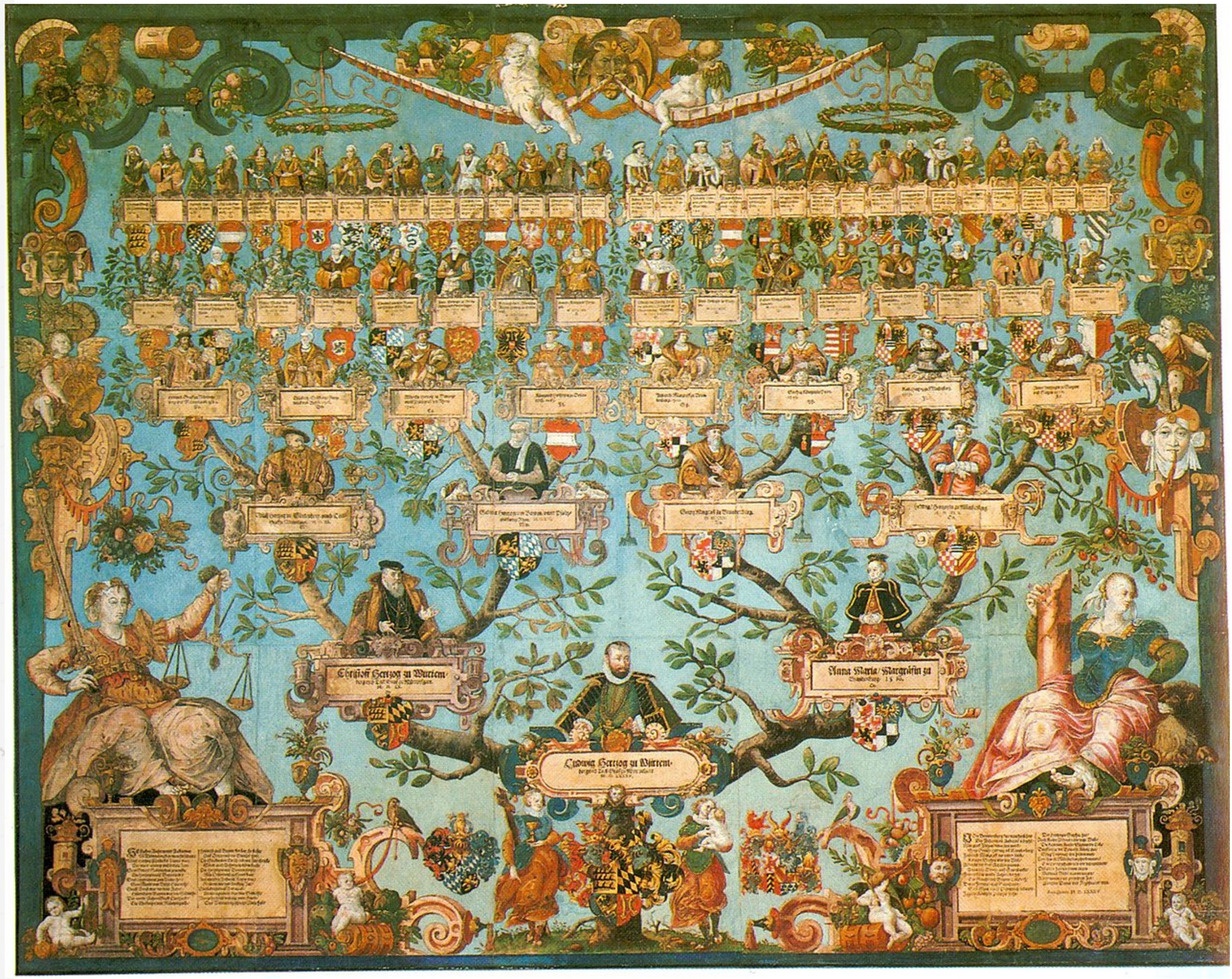
ca. 1200



Social networks - family trees

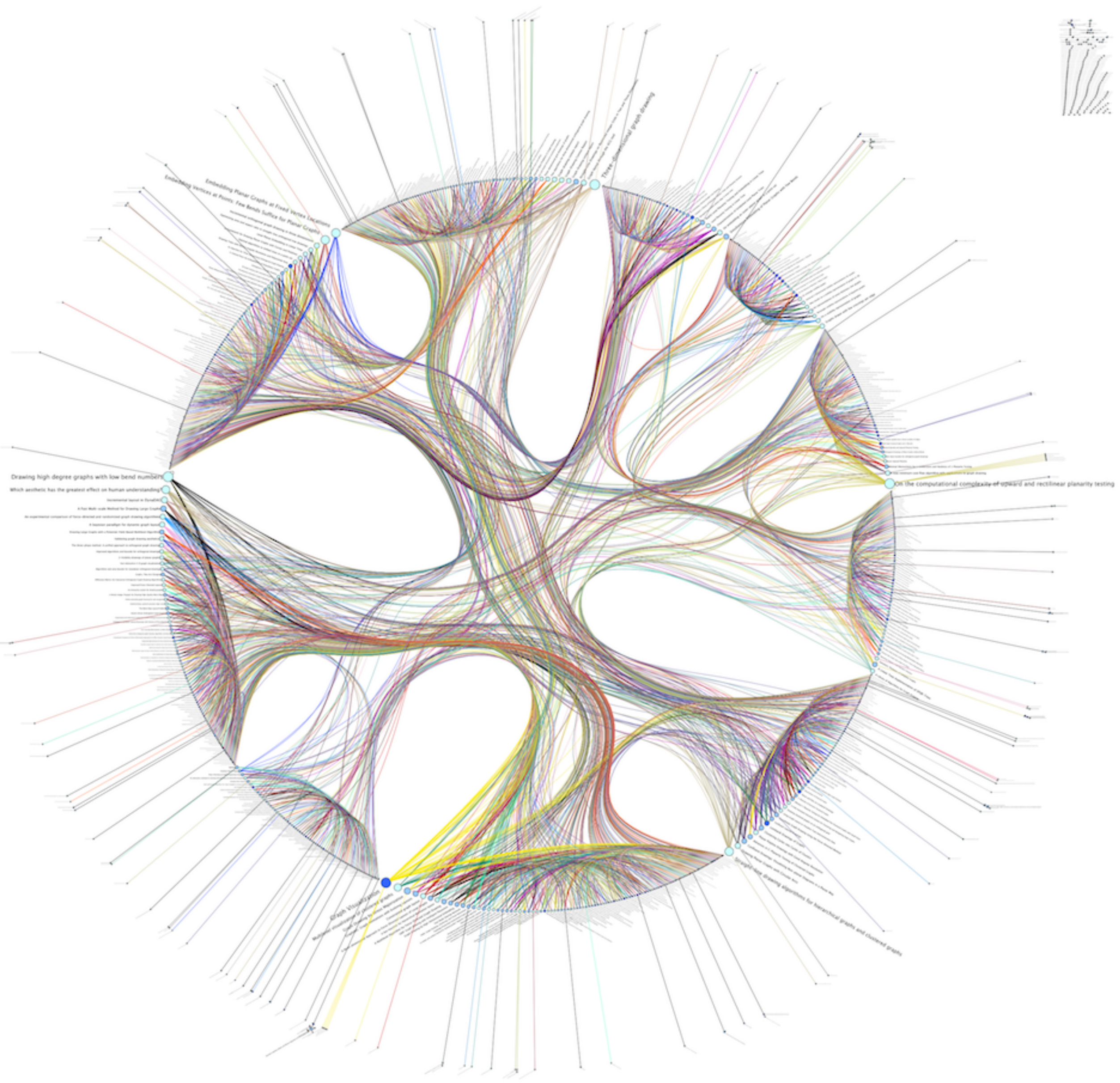


J. Klawitter, T. Mchedlidze, *Link:* go.uniwue.de/myth-poster



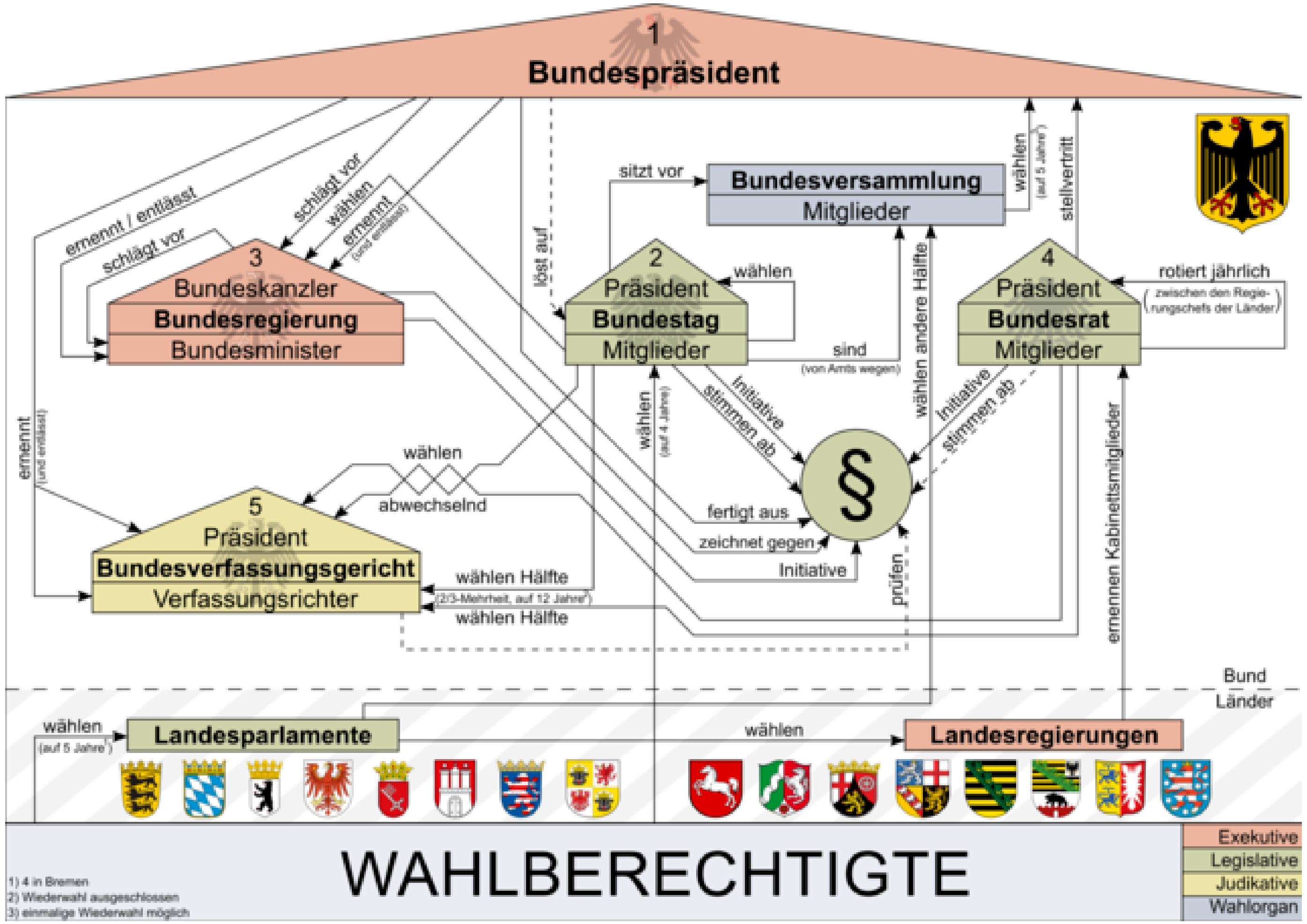
Ahnentafel Herzog Ludwig von Württemberg, 1585

Social network – citation graph

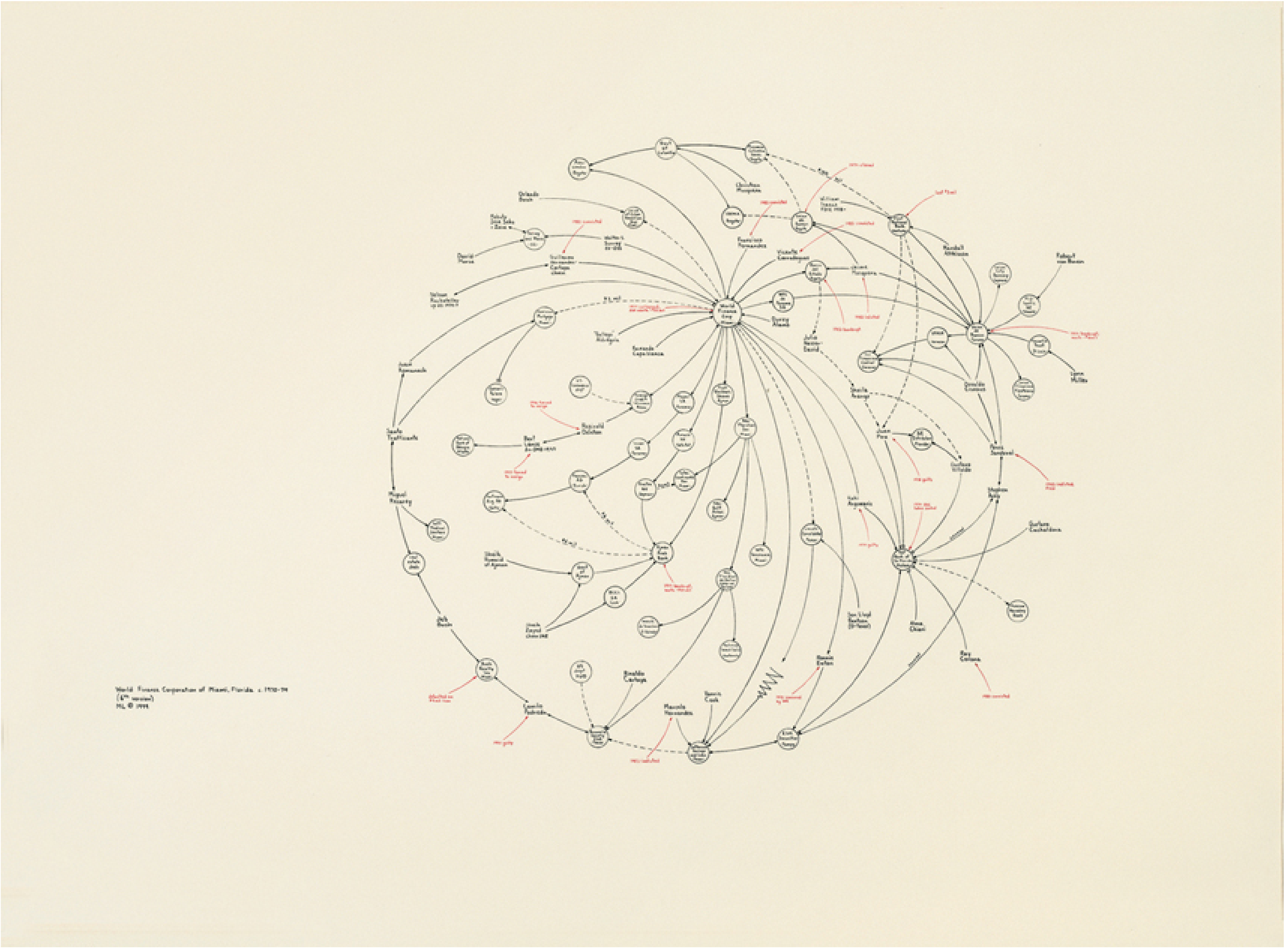


Da Ye, *Link:* https://go.uniwue.de/citation-graph

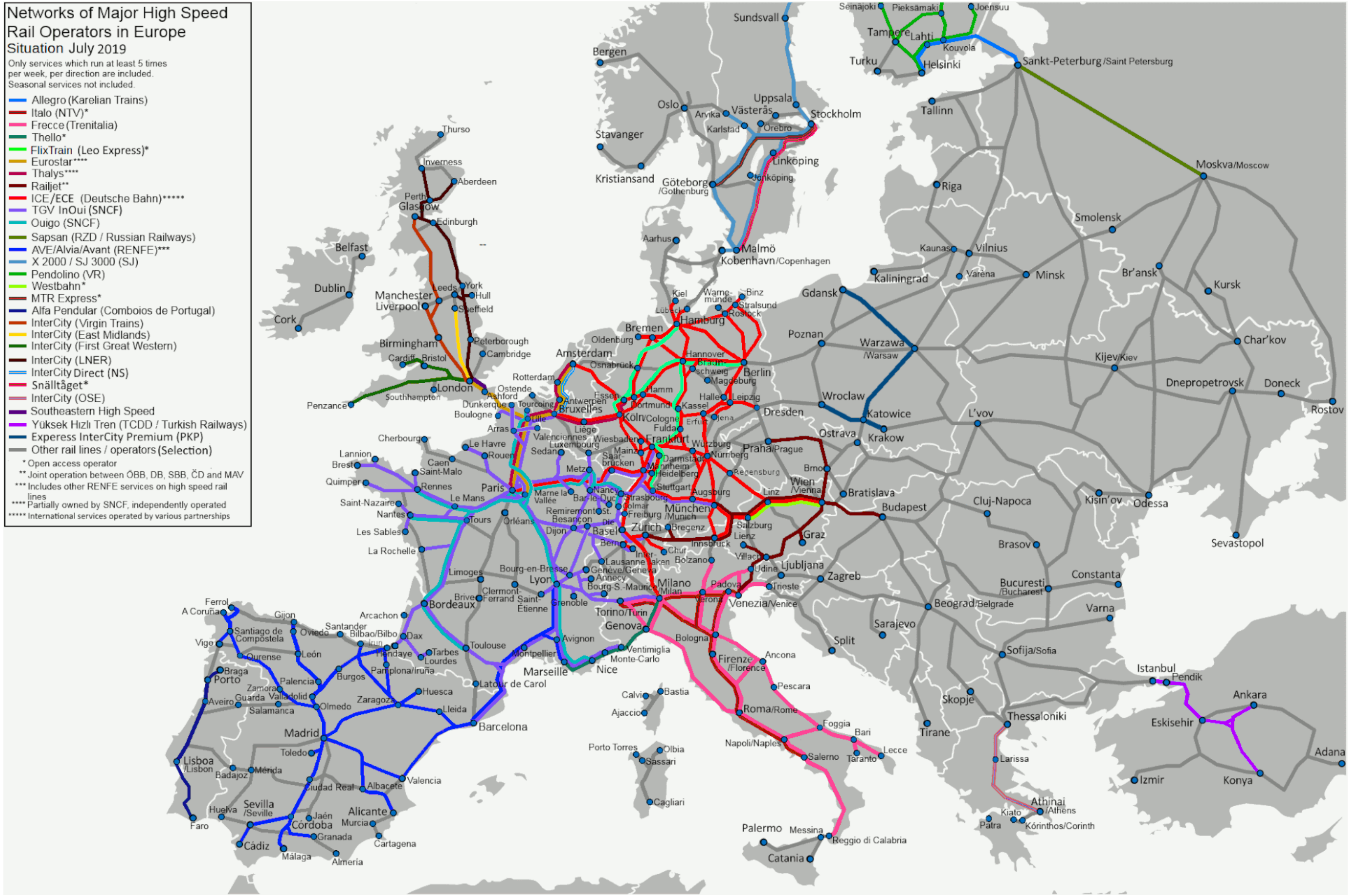
Social network - organisational chart



Social network - world finance corporation



Transportation network – European high speed railroads

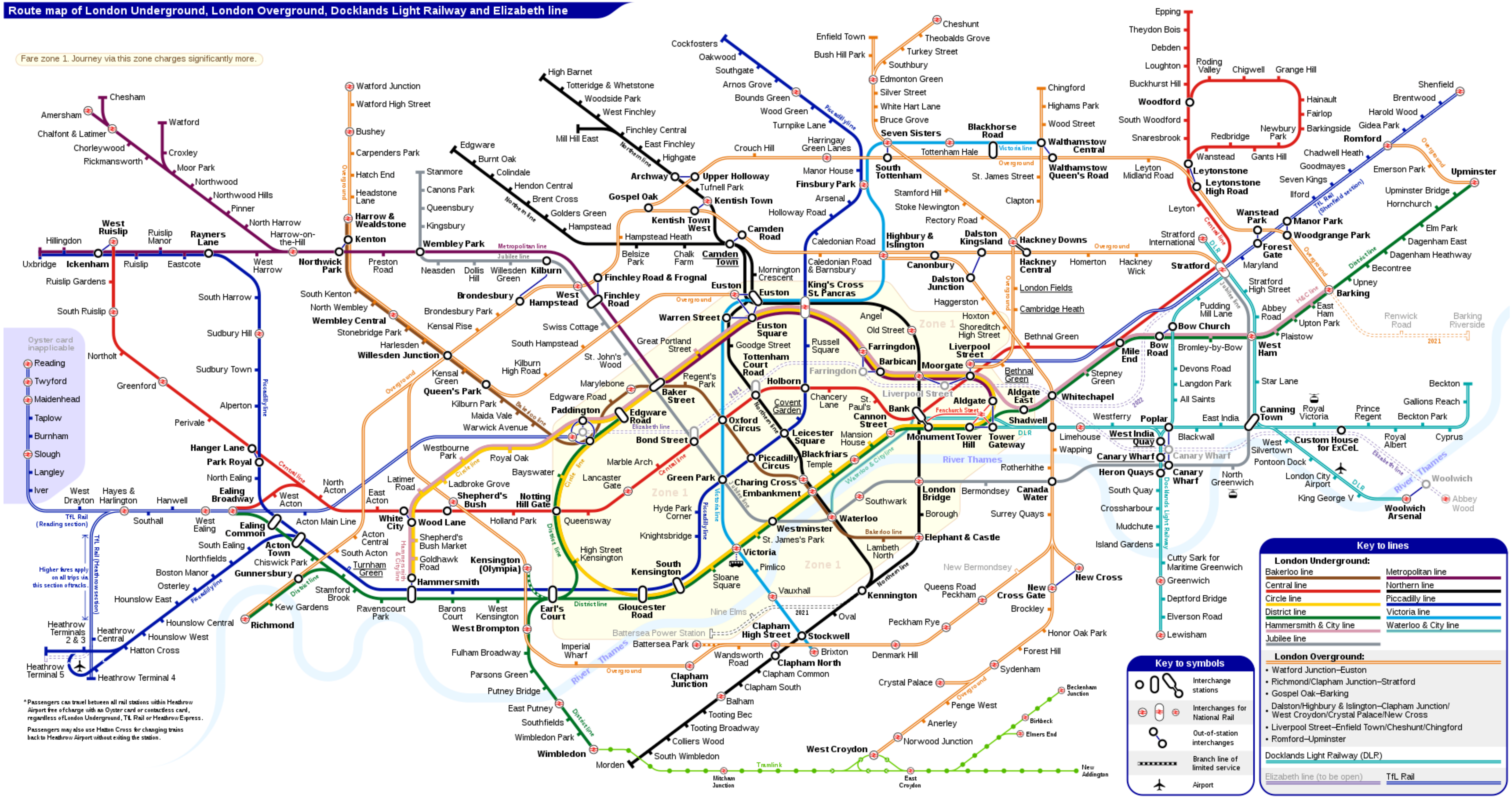


Source: Wiki Commons: Networks of Major High Speed Rail Operators in Europe - CC BY-SA 3.0

Transportation network – London Underground

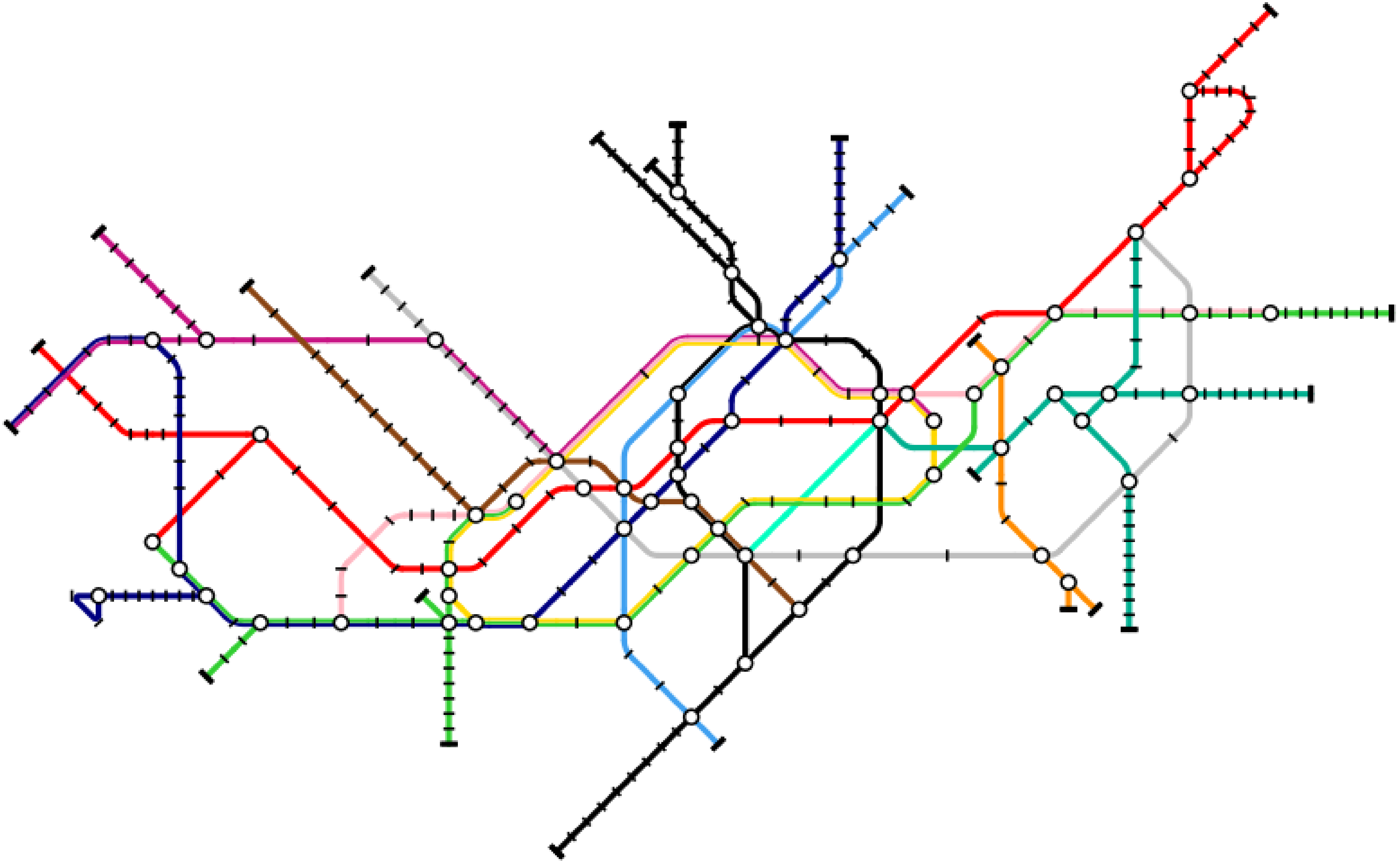
Source: Wiki Commons: London Underground full map - CC BY-SA 3.0

Transportation network – London Underground

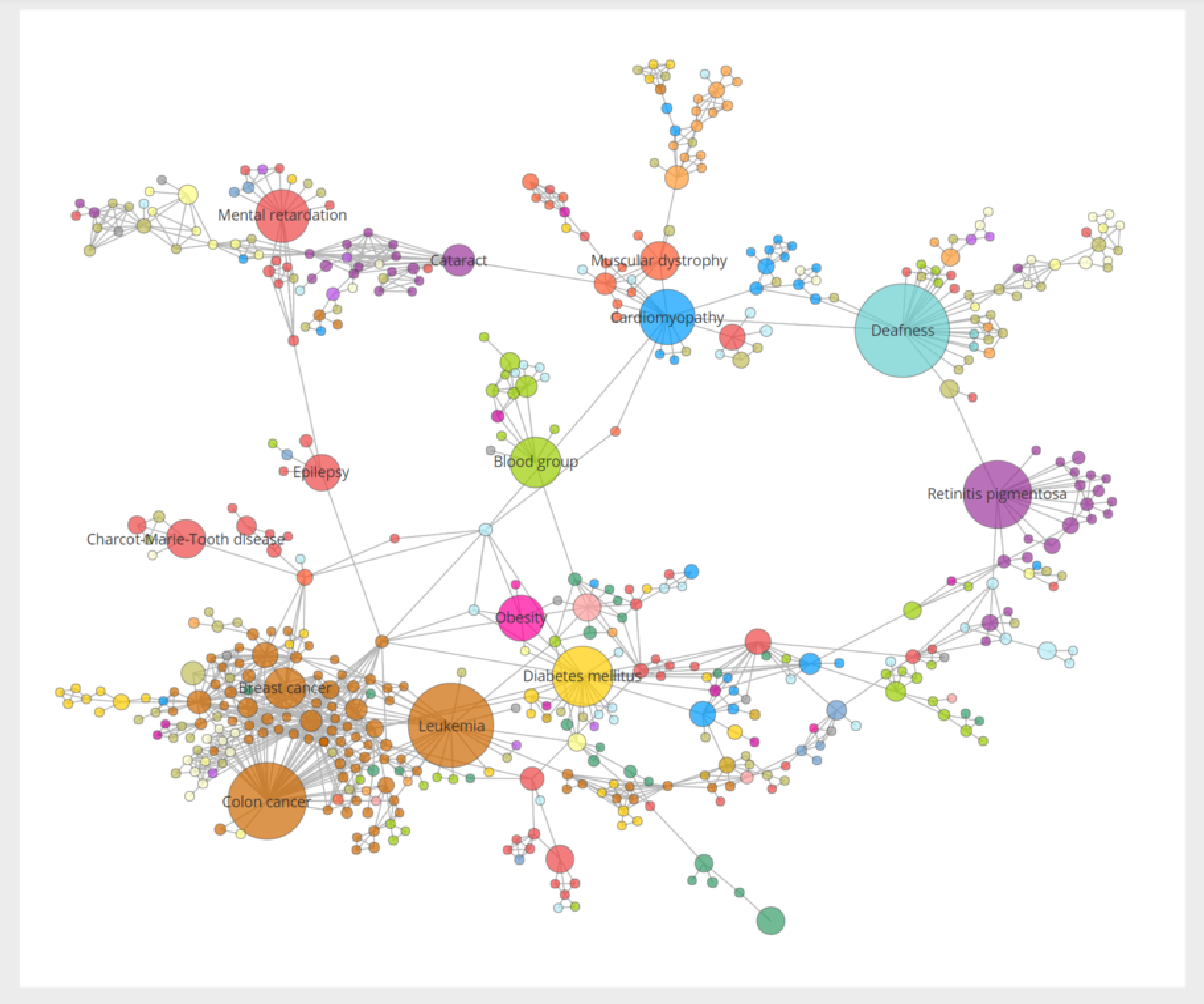


Source: Wiki Commons: London Underground Overground DLR Crossrail map - CC BY-SA 4.0

Transportation network – London Underground

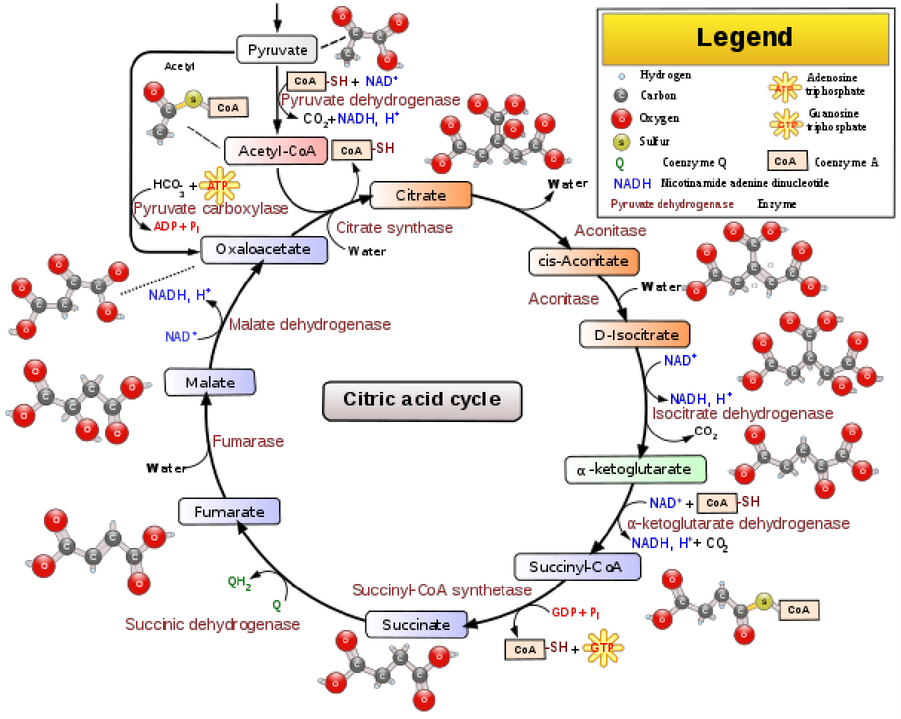


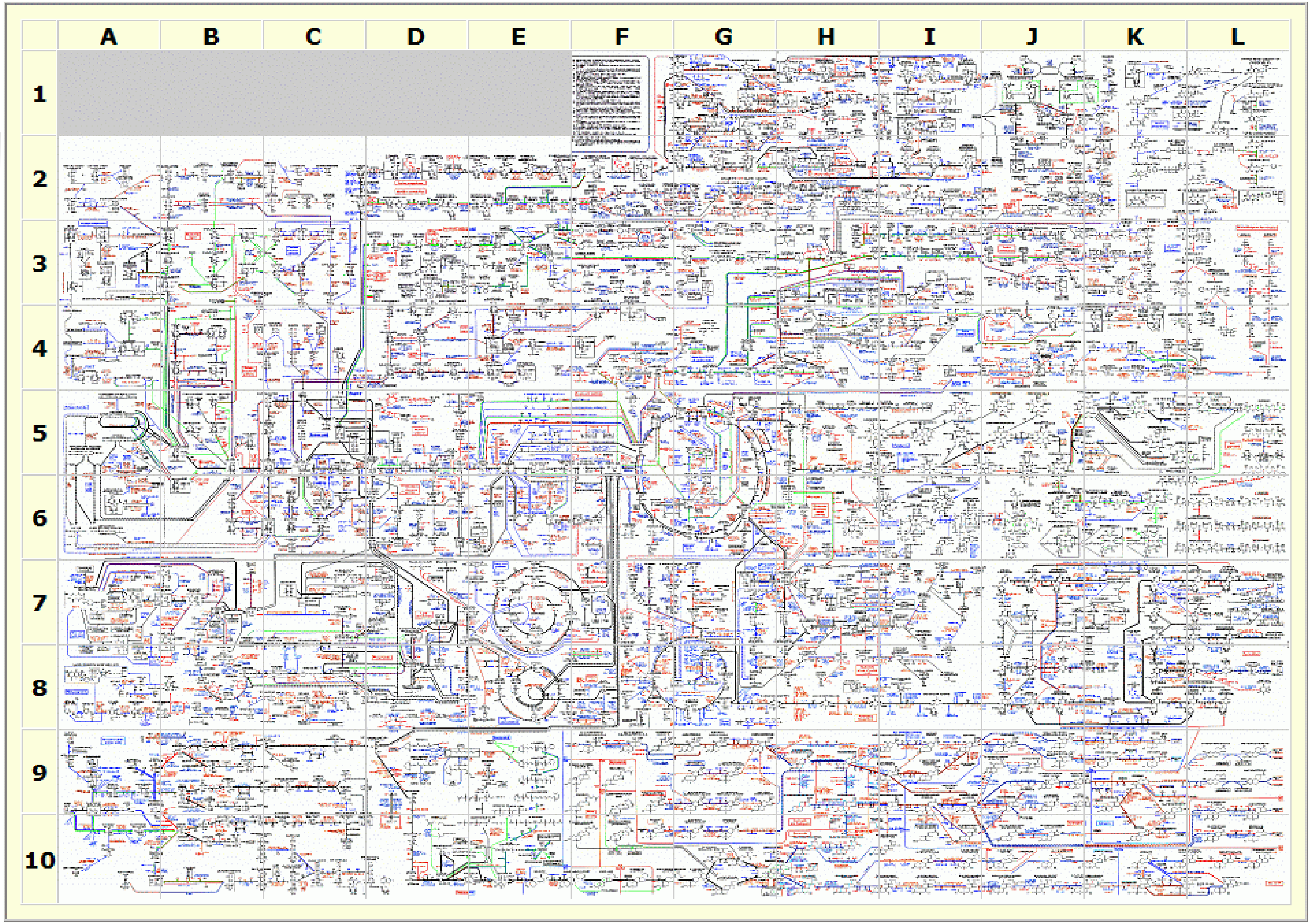
Bioinformatics – disease interaction



Source: Wiki Commons: Human disease network - CC BY-SA 4.0

Bioinformatics – molecular metabolic network

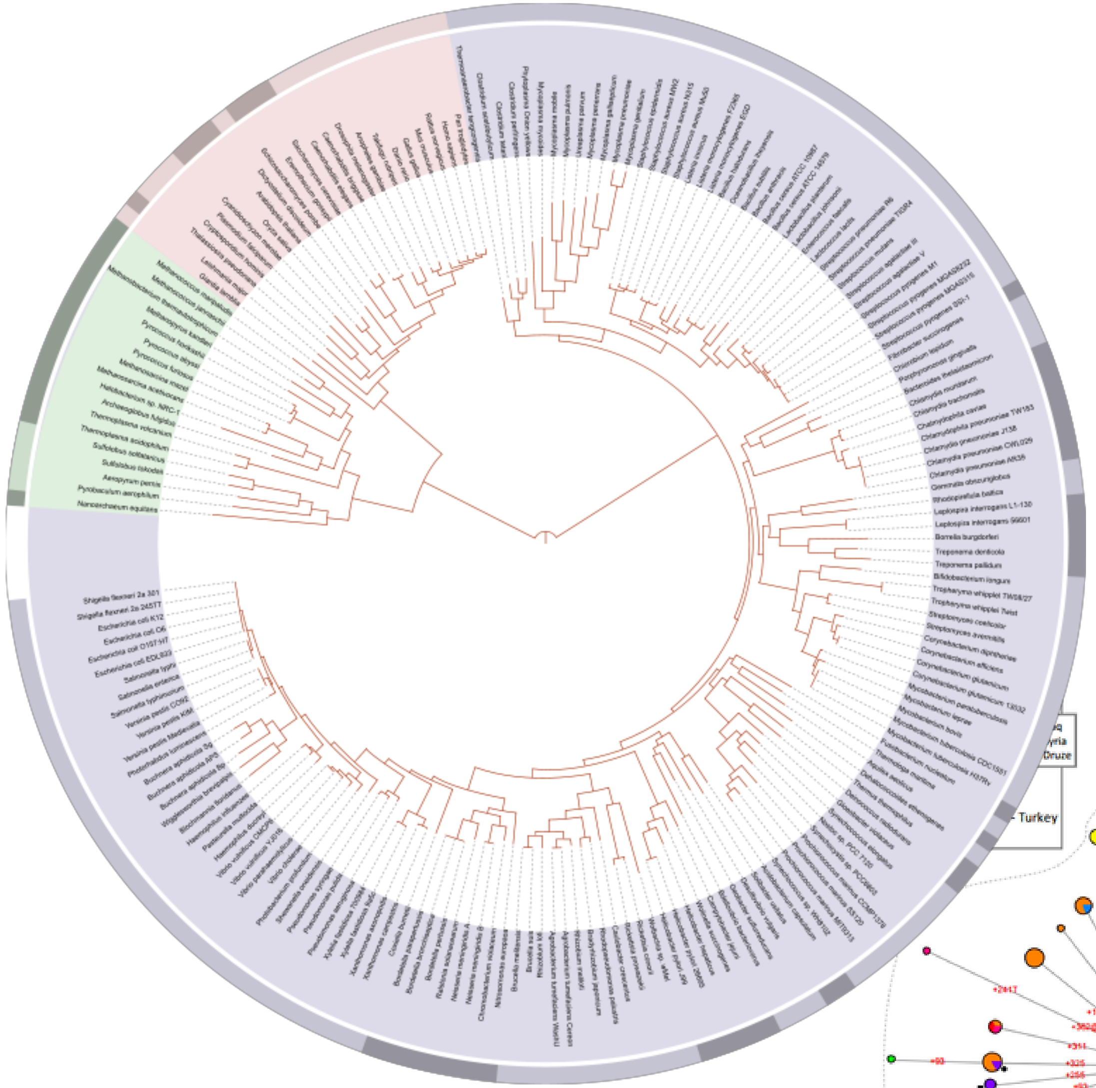


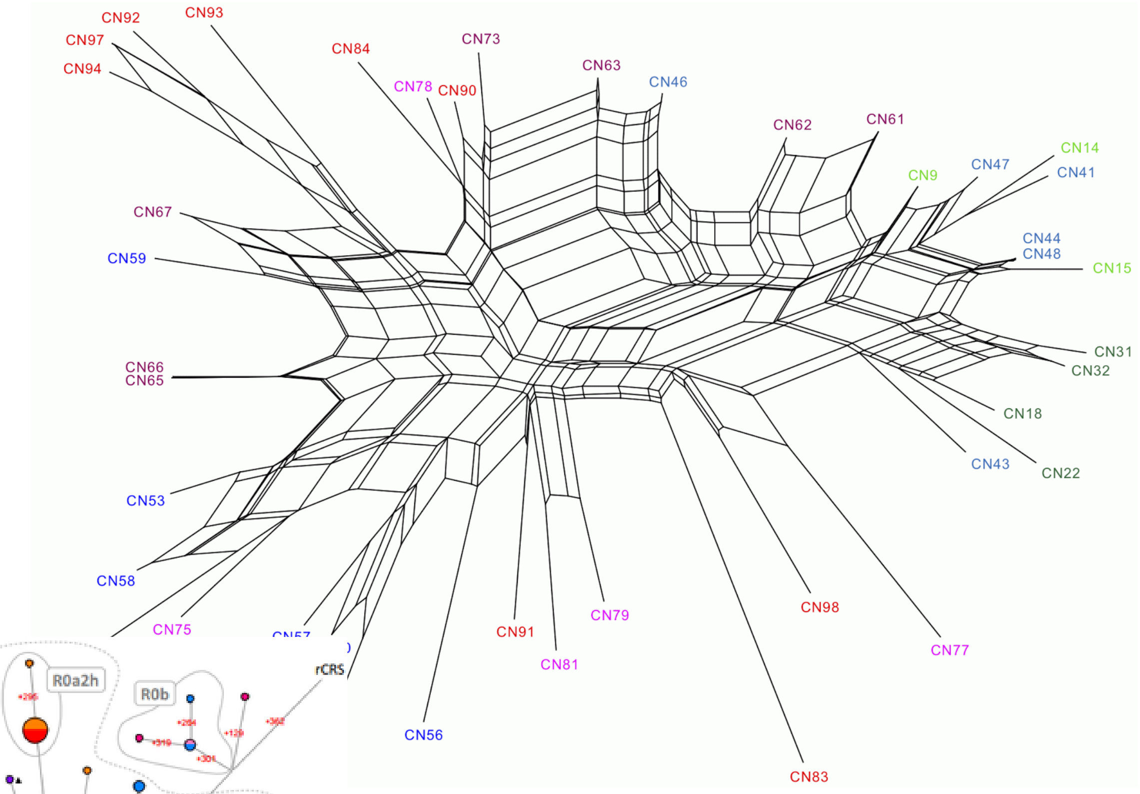


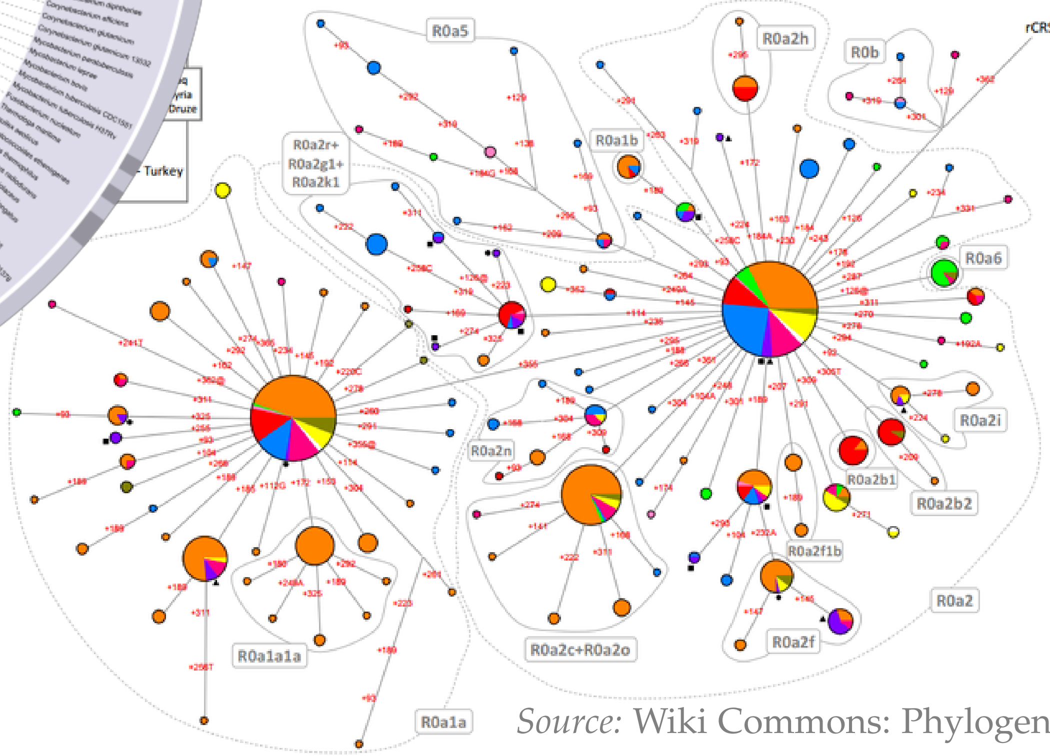
Source: Wiki Commons: Citric acid cycle with aconitate 2 - CC BY-SA 3.0

Source: Thiele et al., Nature Biotechnology 31, 419–425 (2013)

Bioinformatics – phylogenetic trees & networks

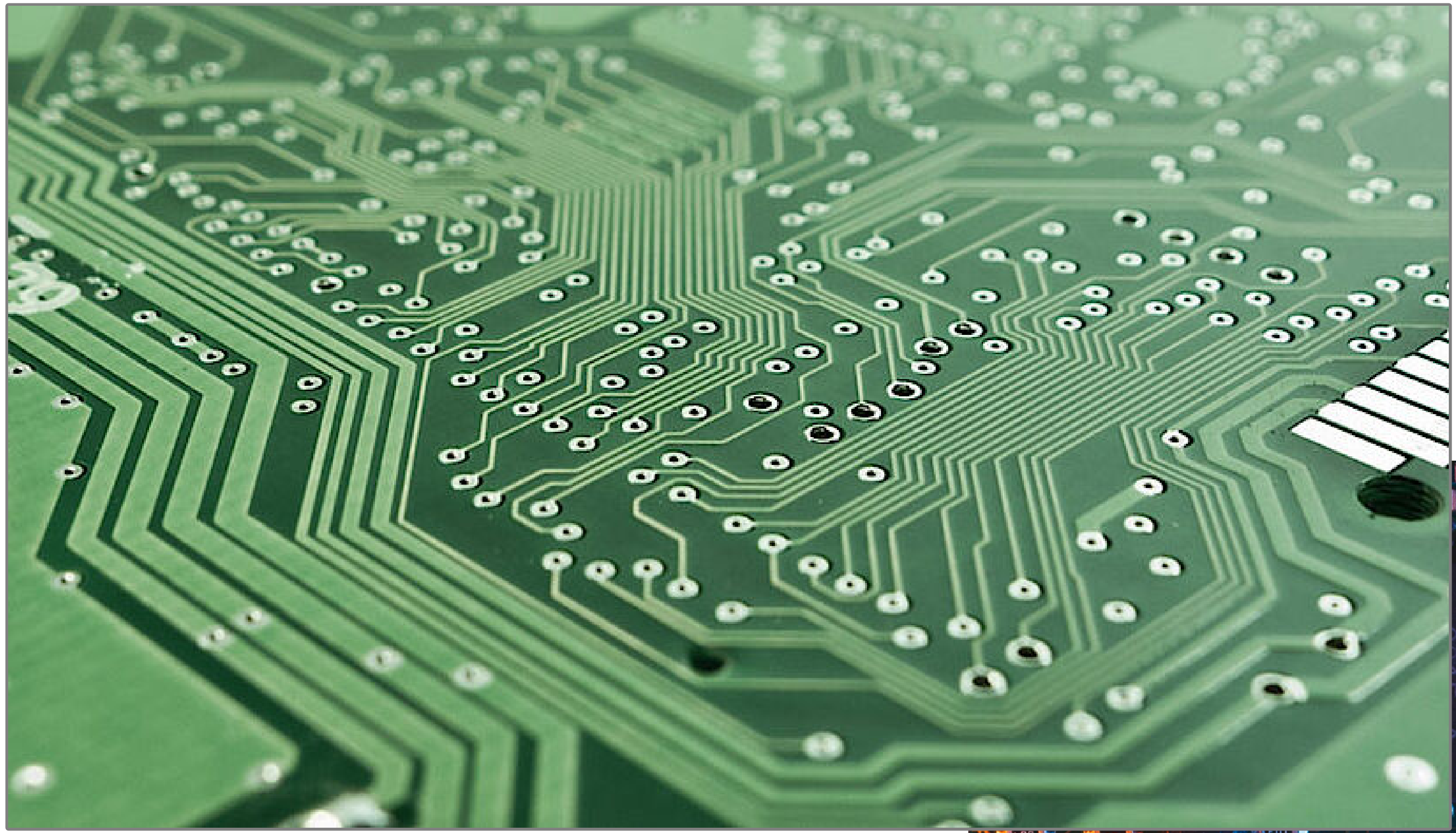




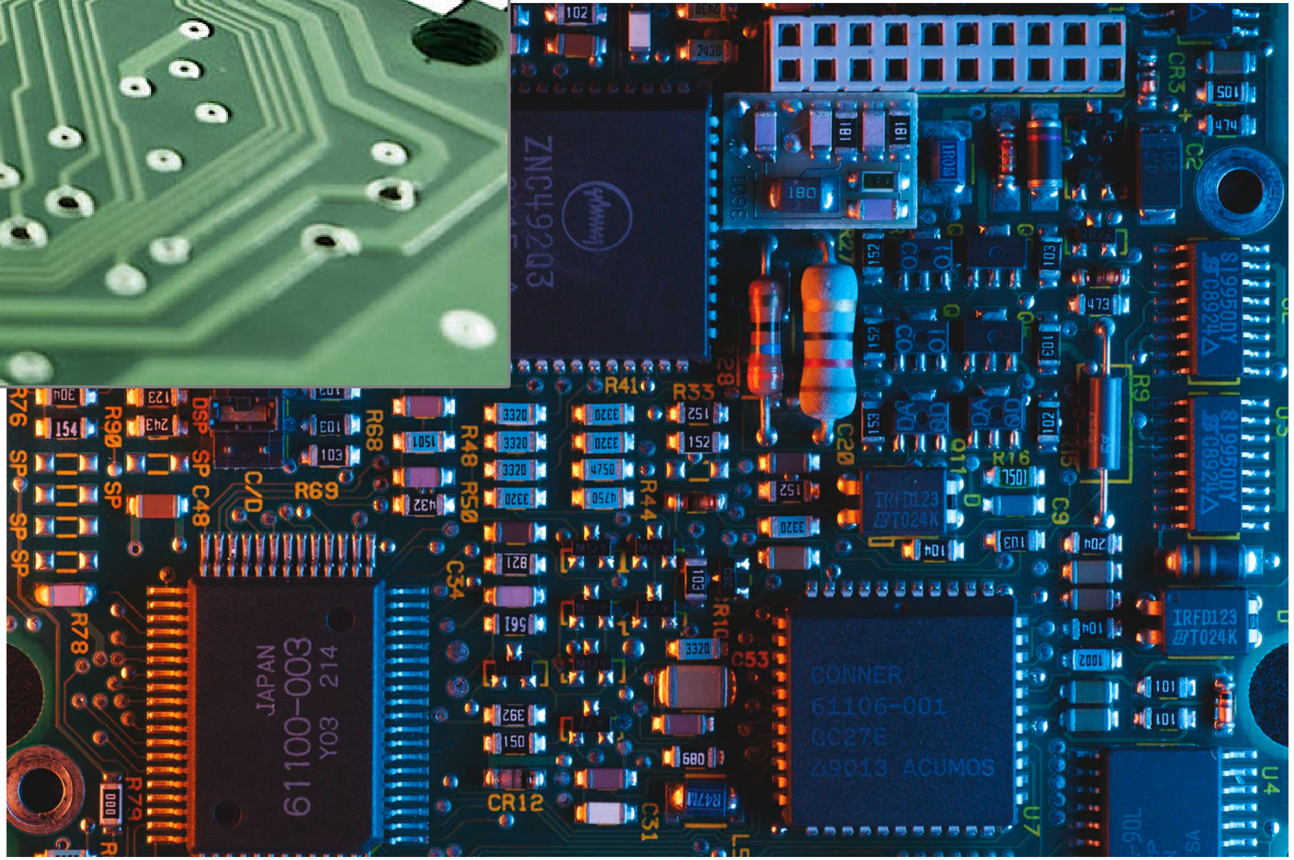


Source: Wiki Commons: Phylogenetic network of HVS-I variation - CC BY 4.0

Technical network – very large-scale integration (VLSI)

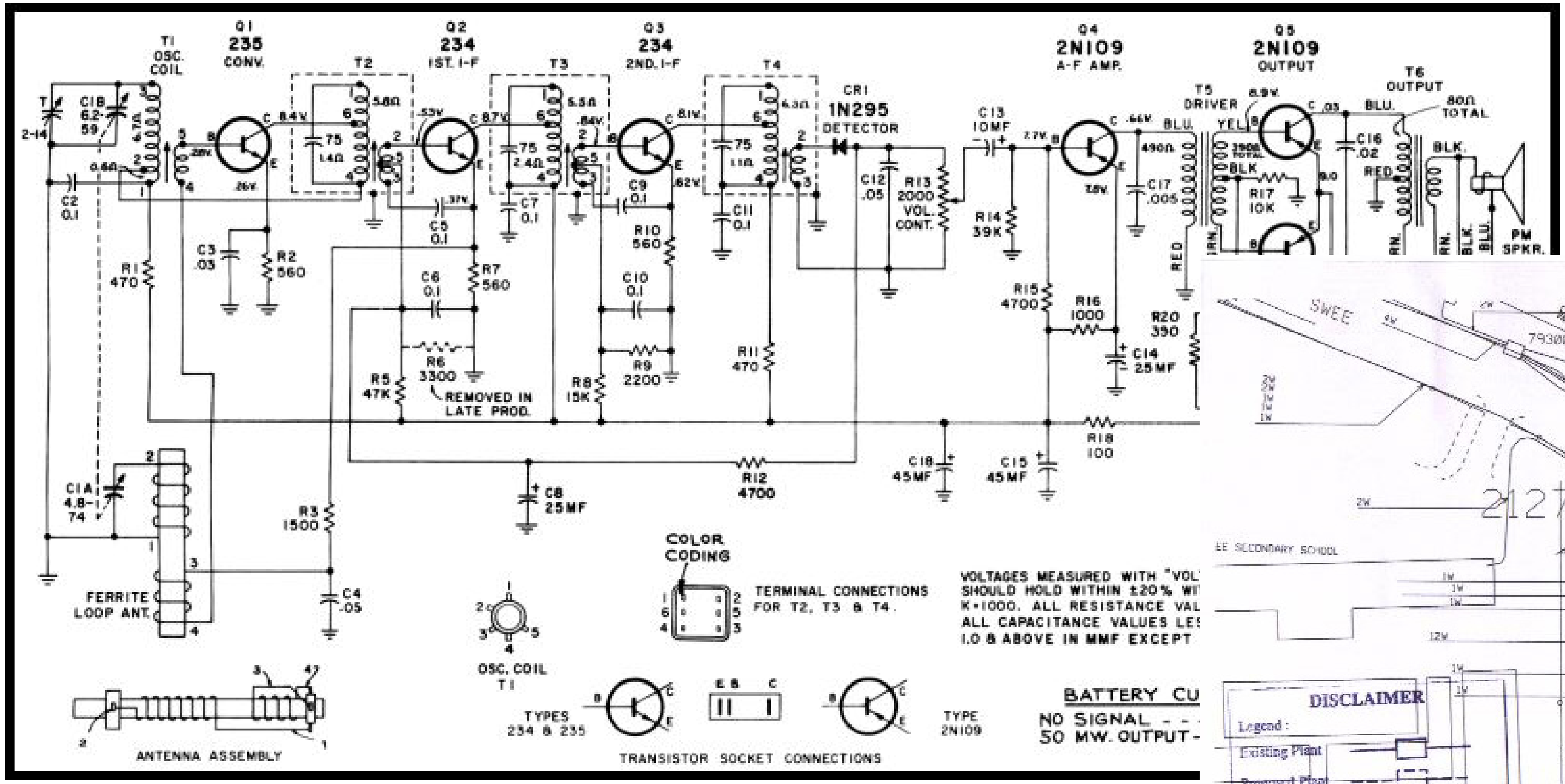


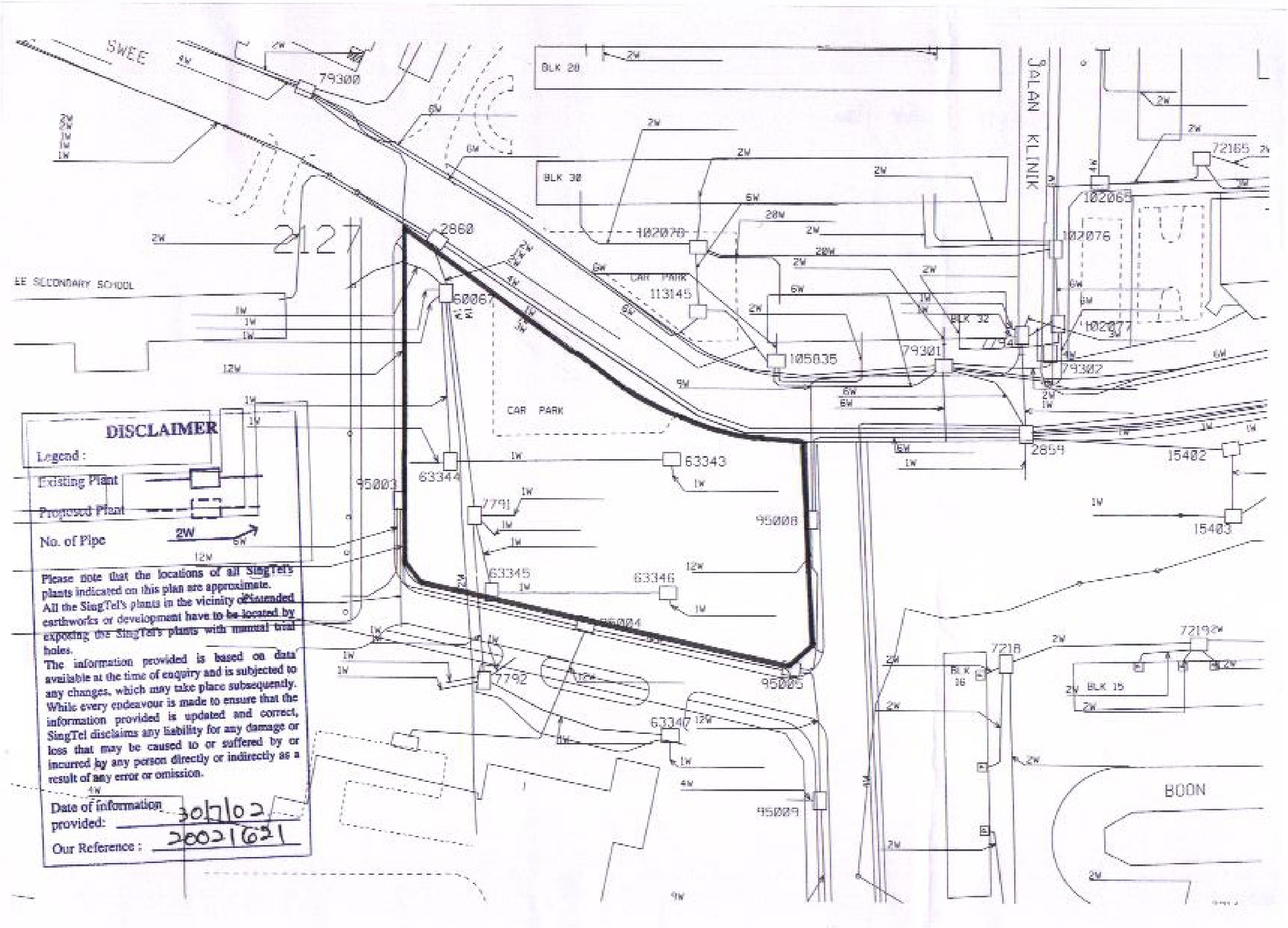
Source: Pixabay



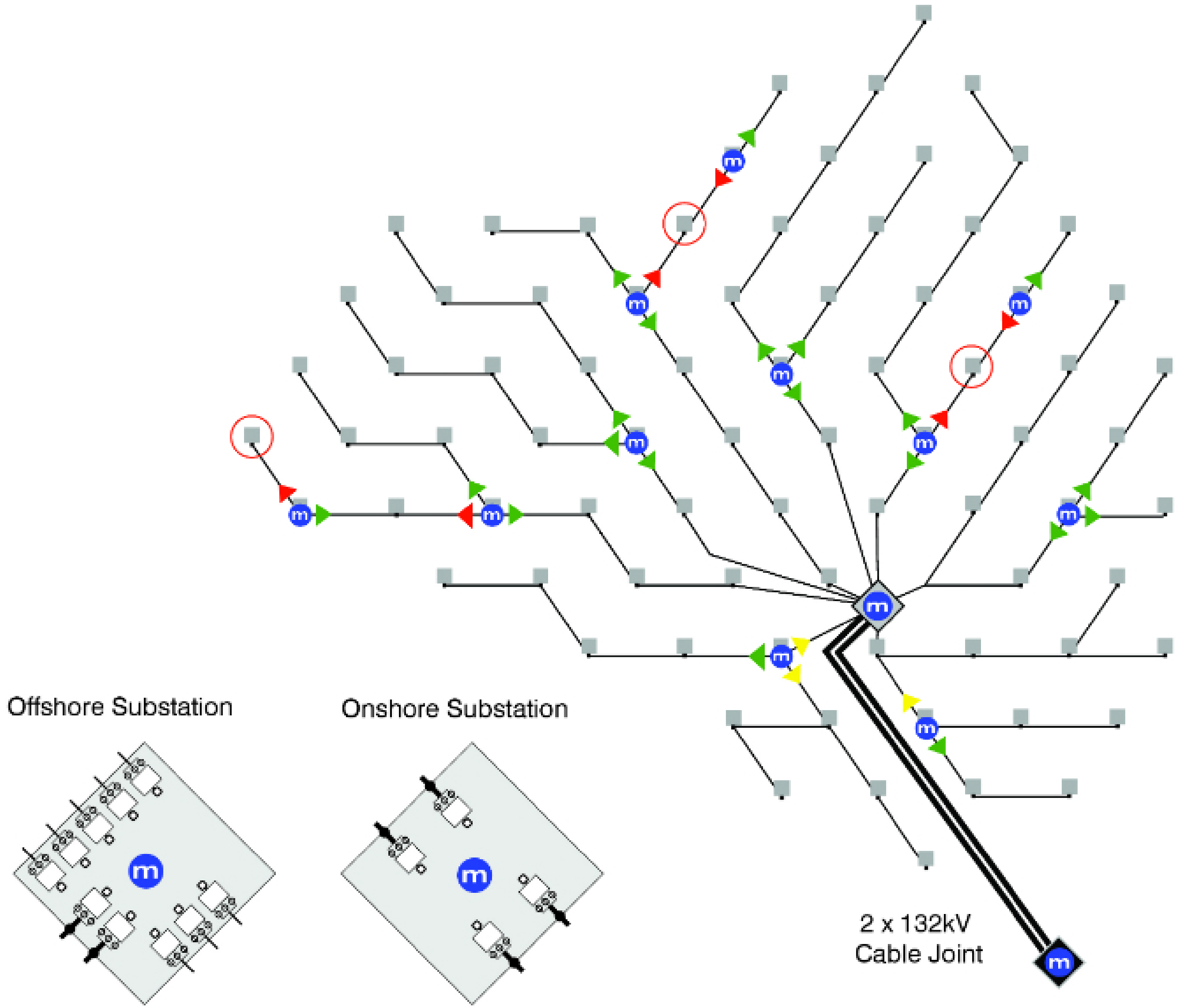
Source: Wiki Commons: Diopsis - CC BY-SA 3.0

Technical network – transistor diagram, wiring



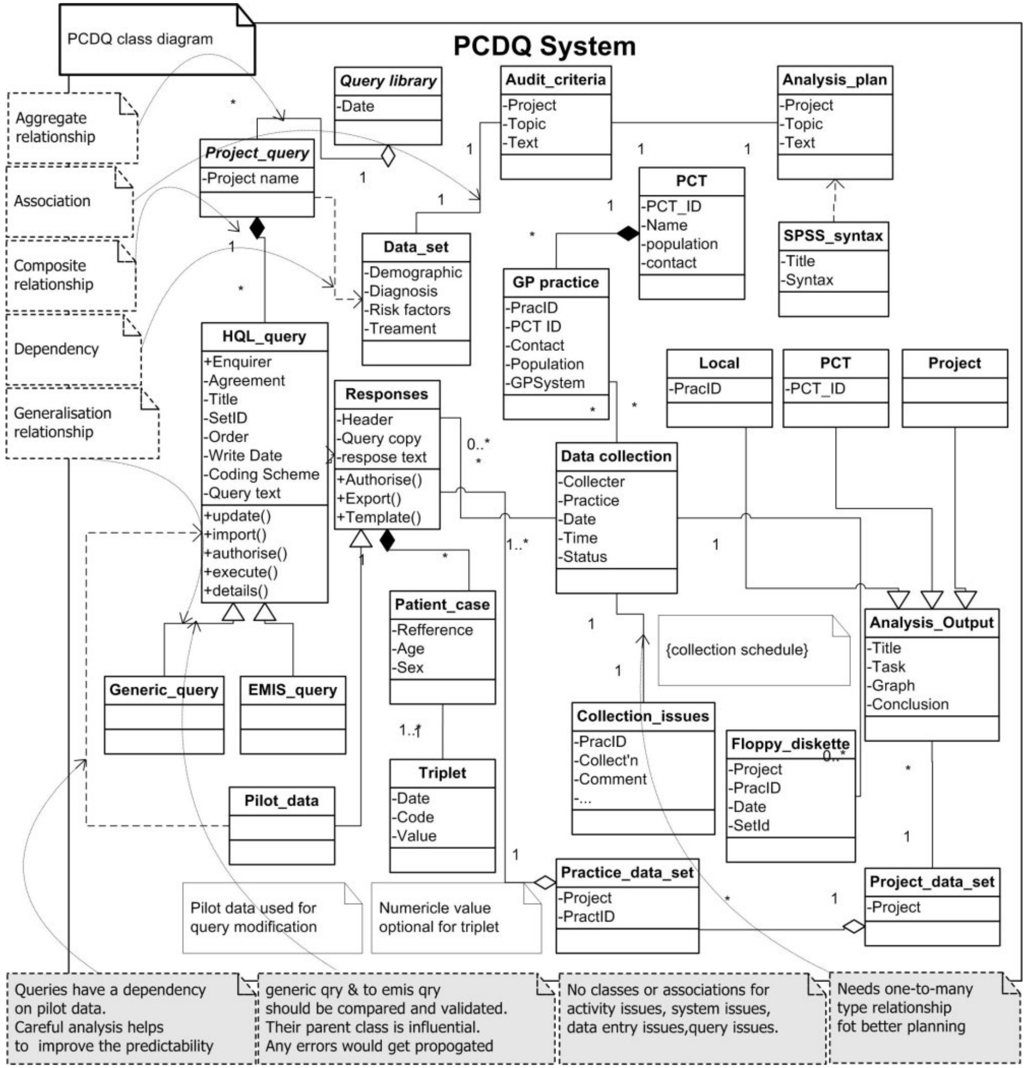


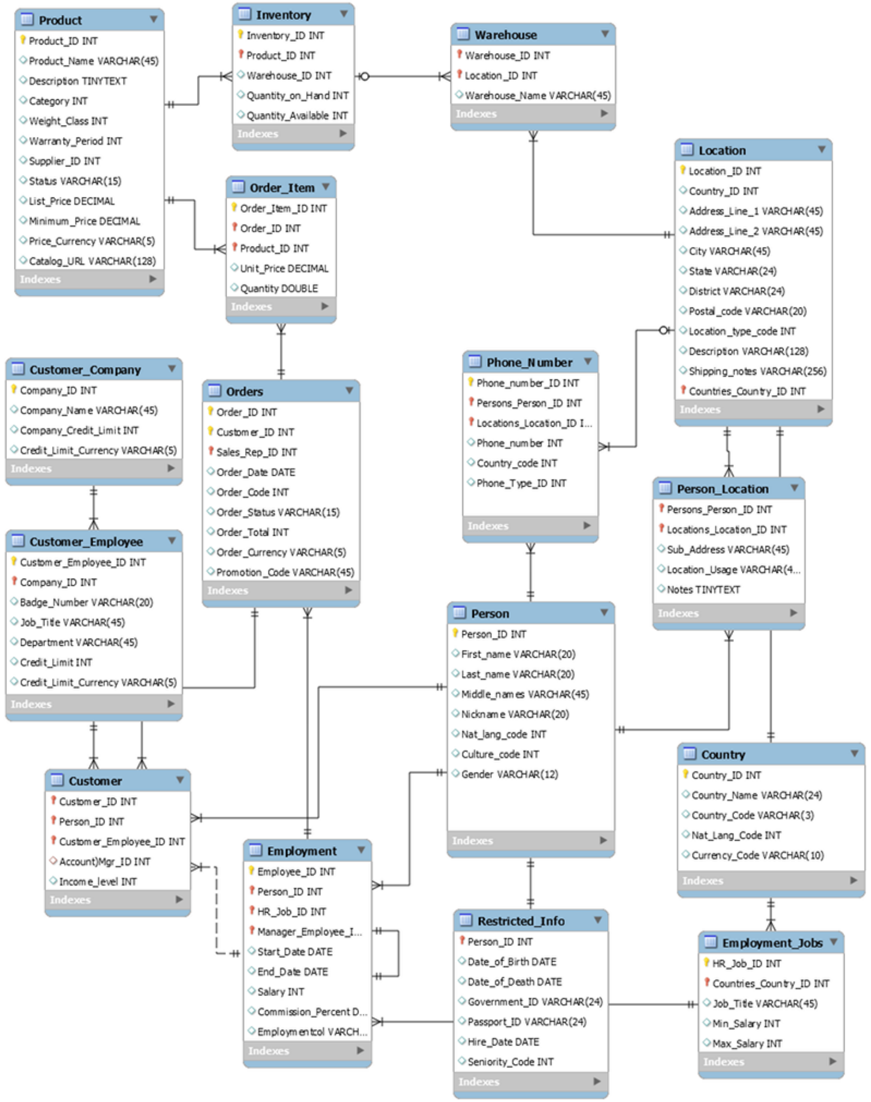
Technical networks – offshore wind farms



Source: Wiki Commons: Alpha Ventus Windmills - CC BY-SA 3.0

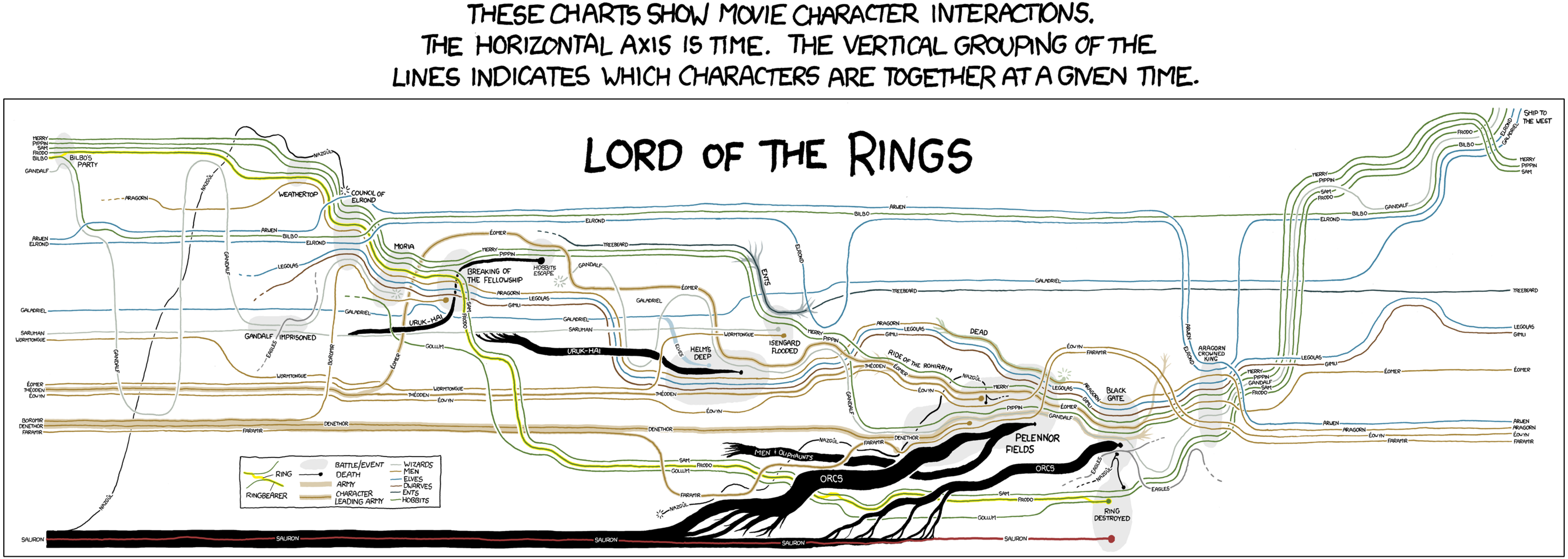
Technical network – UML diagram





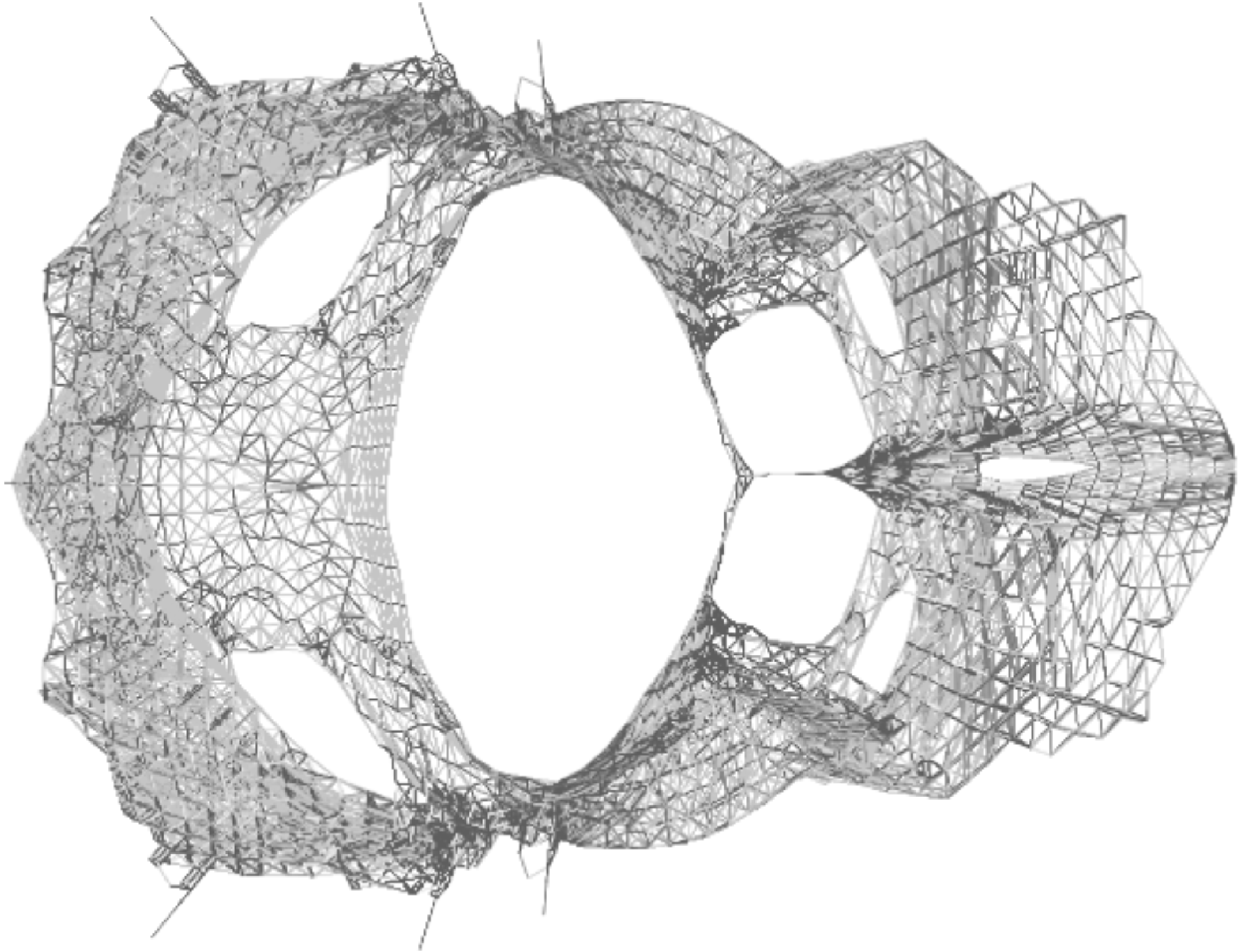
© AWS

Temporal graph layout – storylines

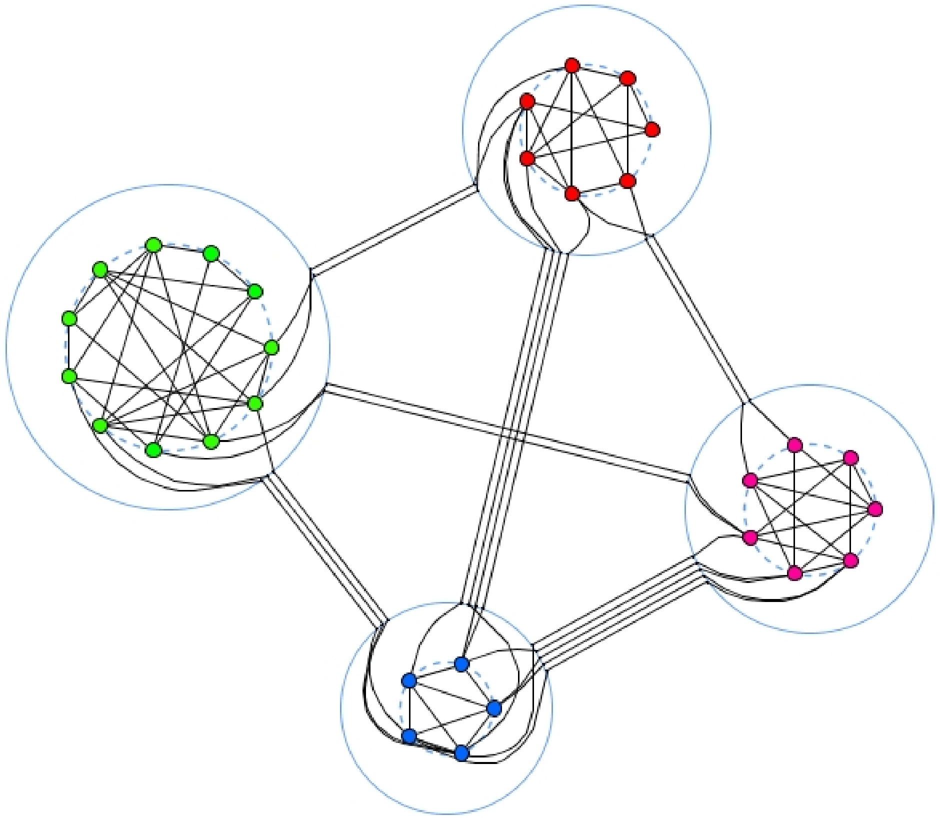


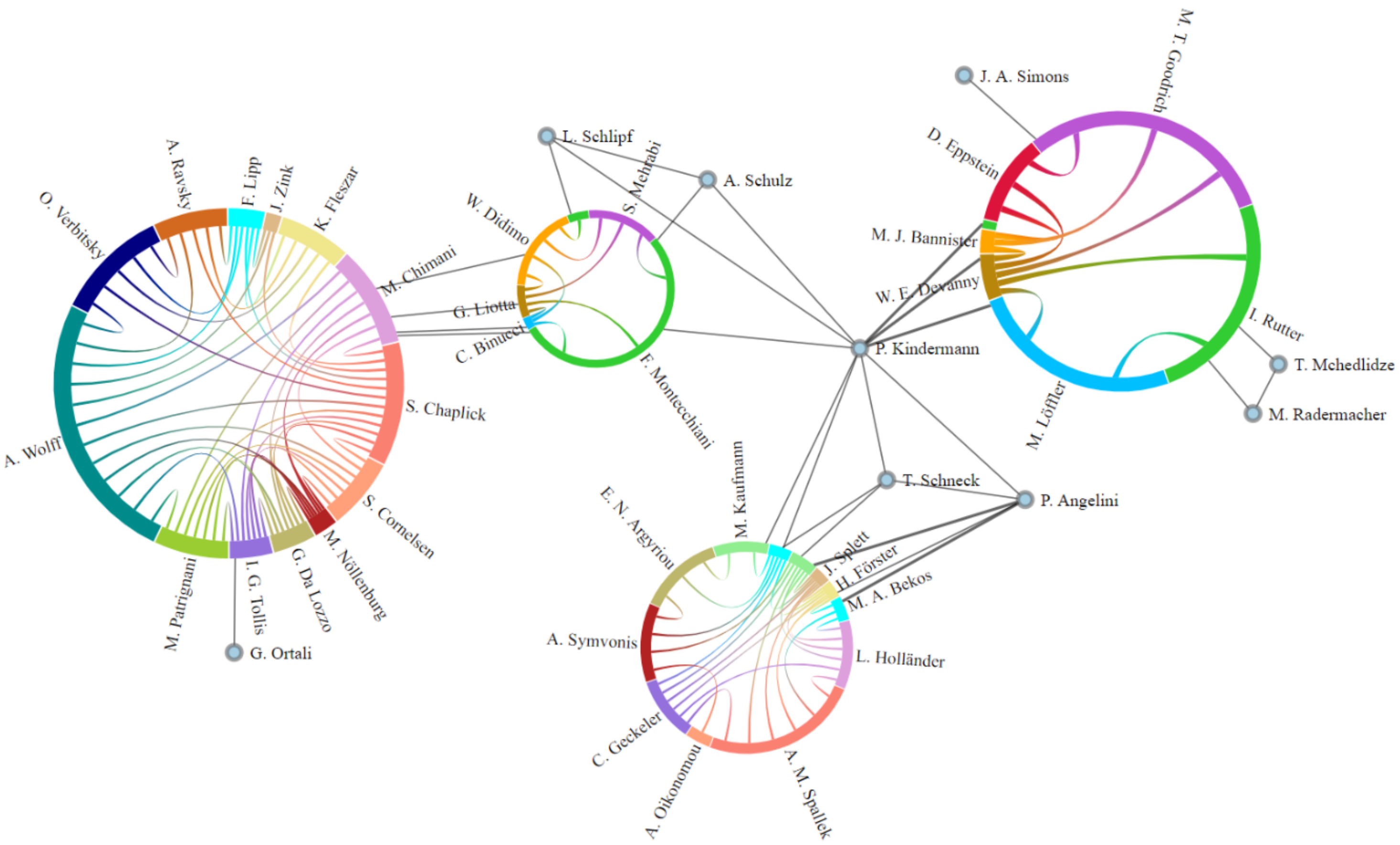
Source and more: xkcd Comic 657 – xkcd.com/657/

Large graphs – object mesh



General graphs – micro-macro layout

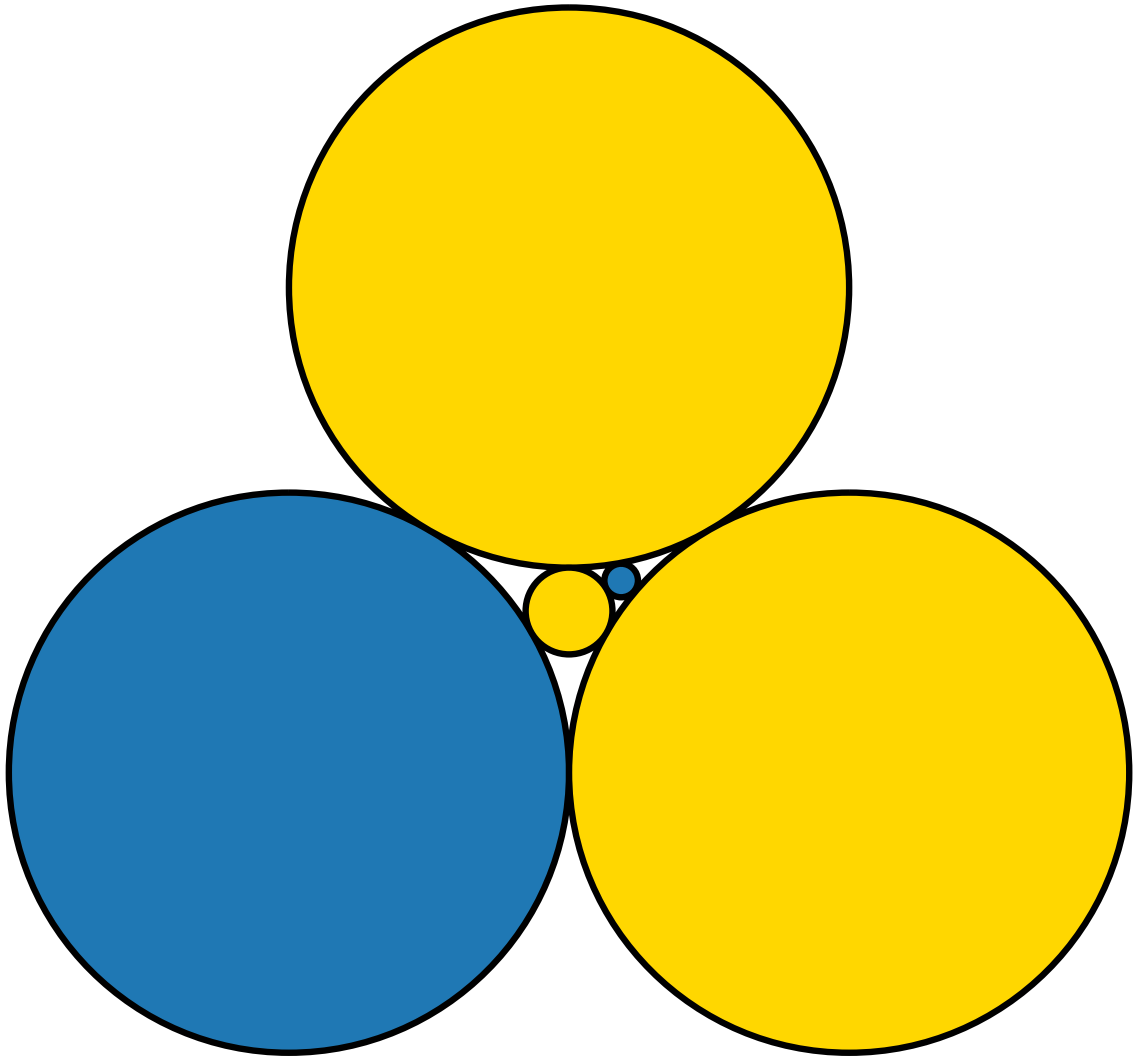


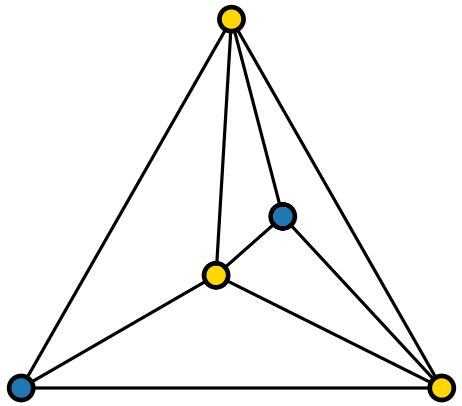


Source: Angori et al., ChordLink: A New Hybrid Visualization Model, GD’19 (2019)

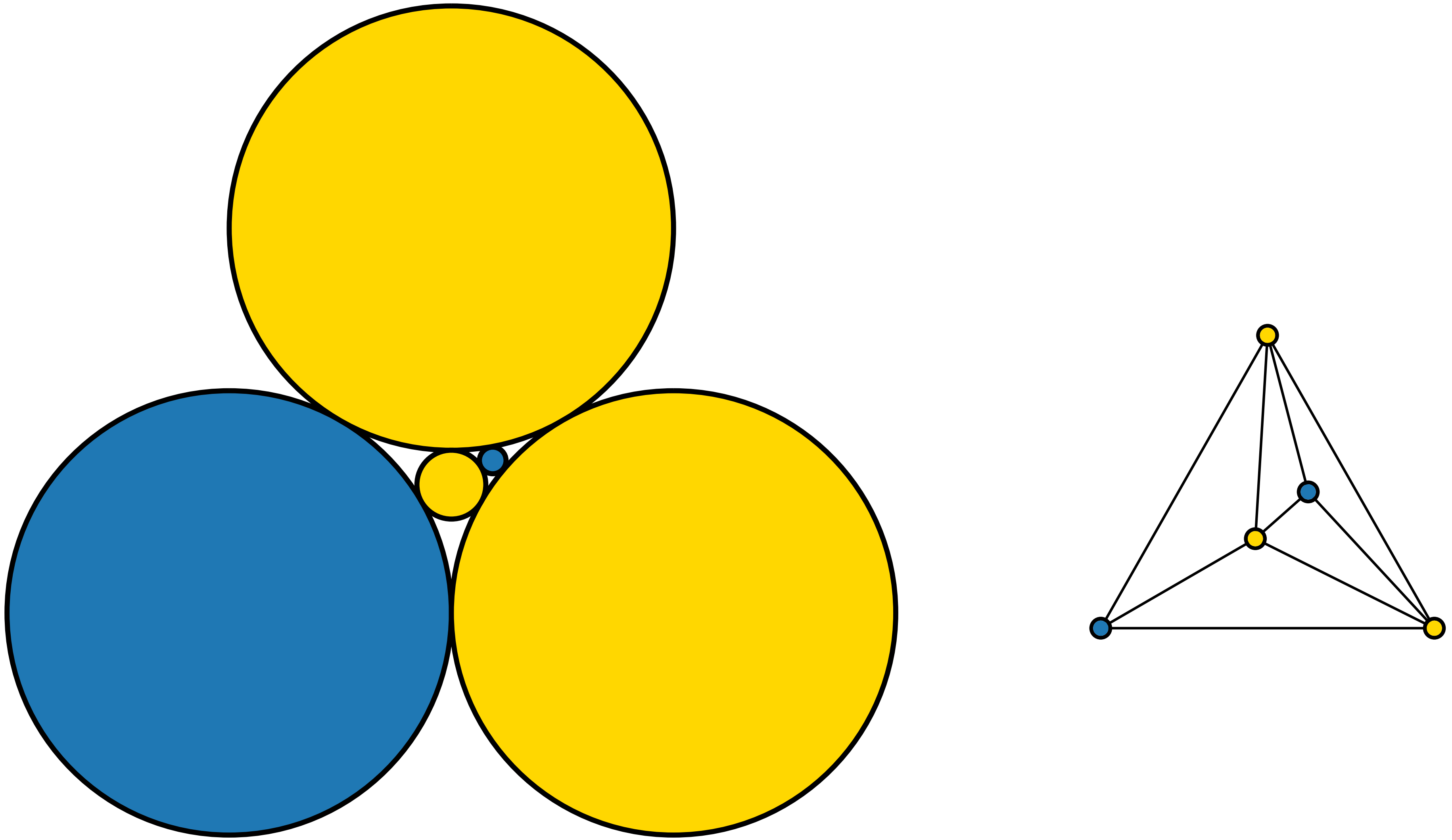
Alternative representations – treemap

Alternative representations – contact graphs





Alternative representations – contact graphs

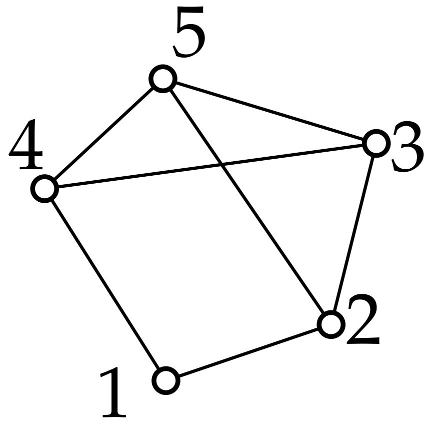


- For more examples see `visualcomplexity.com`

Requirements of a graph layout

1. Drawing conventions and requirements, e.g.,

Requirements of a graph layout



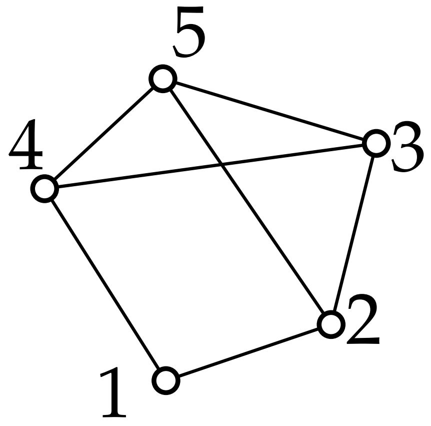
1. Drawing conventions and requirements, e.g.,

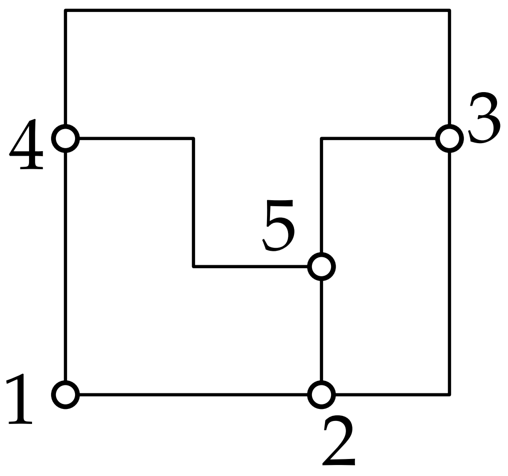
- straight edges with $\Gamma(uv) = \overline{\Gamma(u)\Gamma(v)}$

Requirements of a graph layout

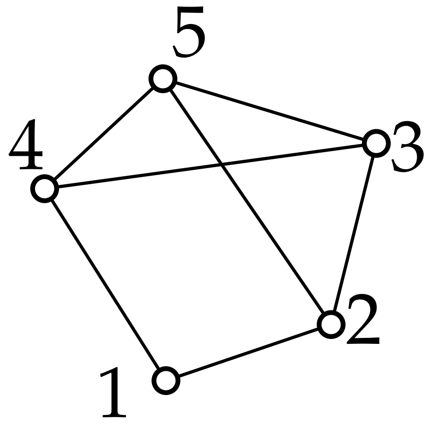
1. Drawing conventions and requirements, e.g.,

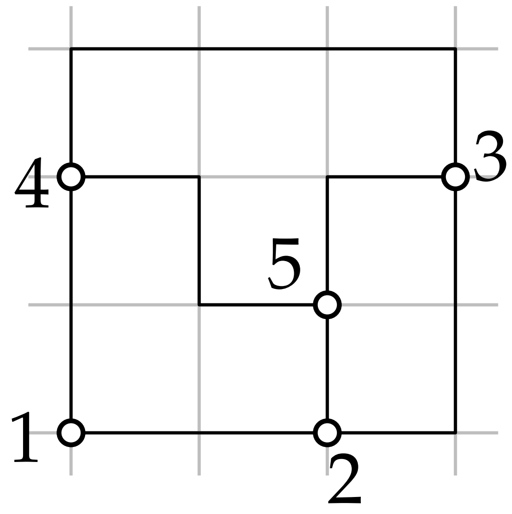
- straight edges with $\Gamma(uv) = \overline{\Gamma(u)\Gamma(v)}$
- orthogonal edges (i.e. with bends)





Requirements of a graph layout

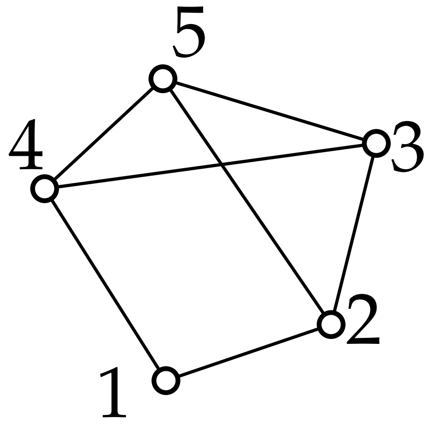


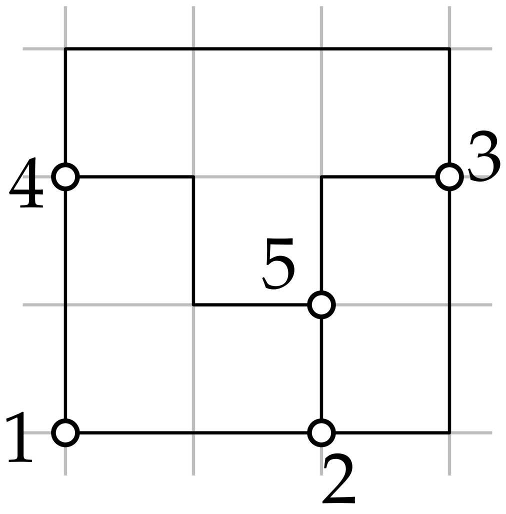


1. Drawing conventions and requirements, e.g.,
   - straight edges with $\Gamma(uv) = \overline{\Gamma(u)\Gamma(v)}$
   - orthogonal edges (i.e. with bends)
   - grid drawings

Requirements of a graph layout

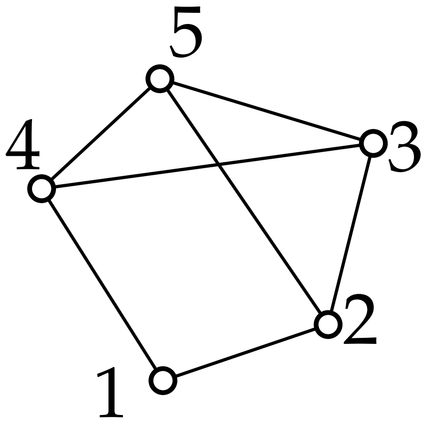
1. Drawing conventions and requirements, e.g.,
- straight edges with $\Gamma(uv) = \overline{\Gamma(u)\Gamma(v)}$
- orthogonal edges (i.e. with bends)
- grid drawings
- without crossing

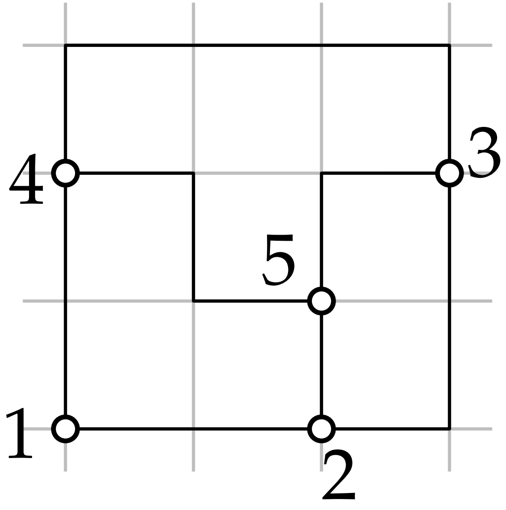


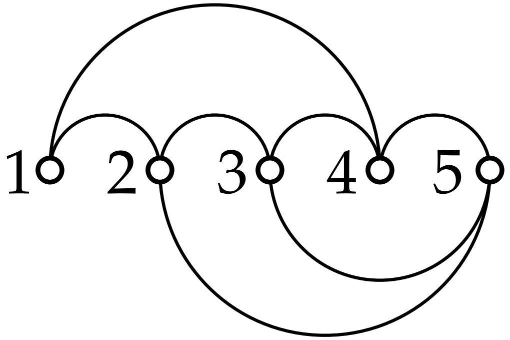


Requirements of a graph layout

1. Drawing conventions and requirements, e.g.,
- straight edges with $\Gamma(uv) = \overline{\Gamma(u)\Gamma(v)}$
- orthogonal edges (i.e. with bends)
- grid drawings
- without crossing

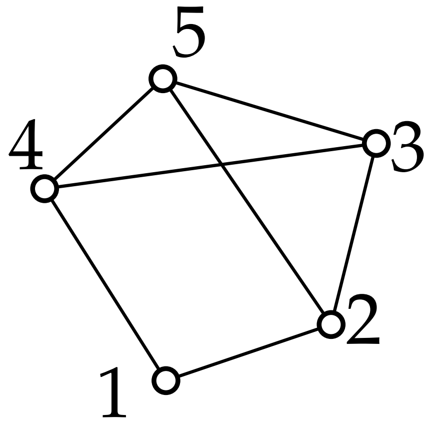


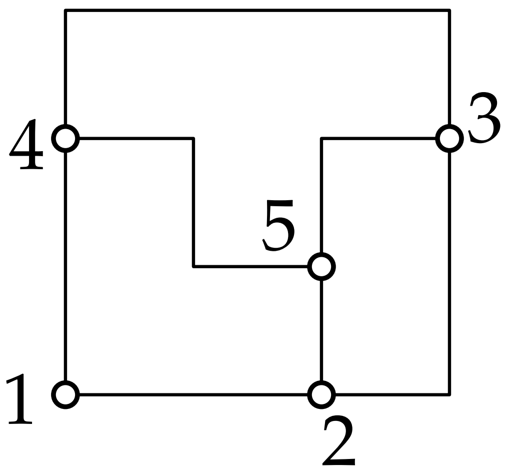


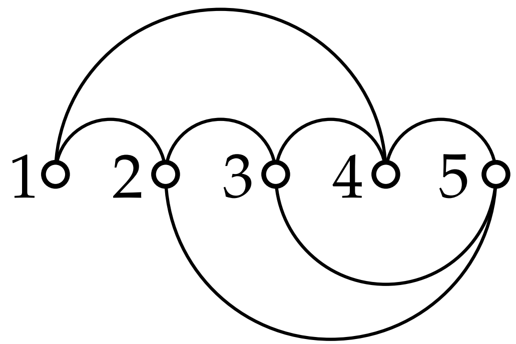


Requirements of a graph layout

1. Drawing conventions and requirements, e.g.,
   - straight edges with $\Gamma(uv) = \overline{\Gamma(u)\Gamma(v)}$
   - orthogonal edges (i.e. with bends)
   - grid drawings
   - without crossing
2. Aesthetics to be optimized, e.g.

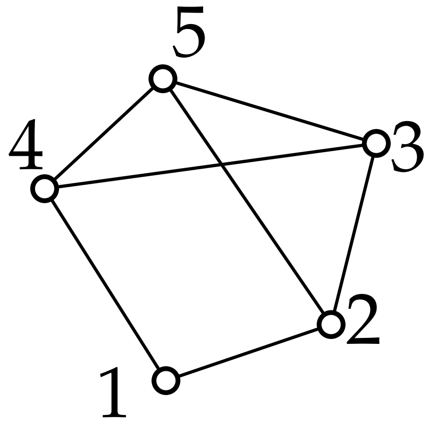


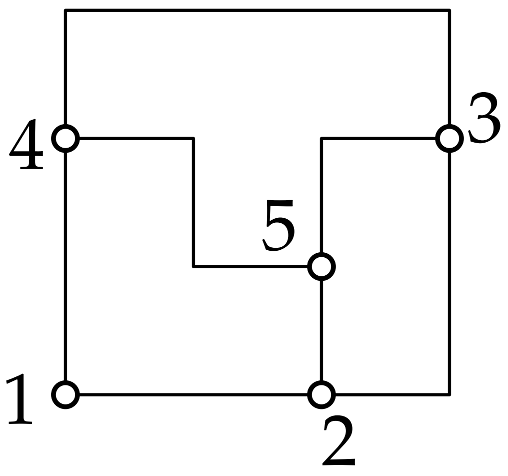


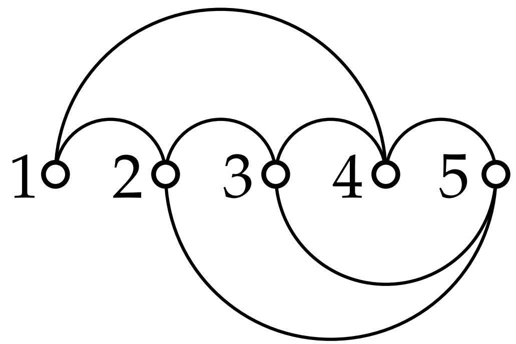


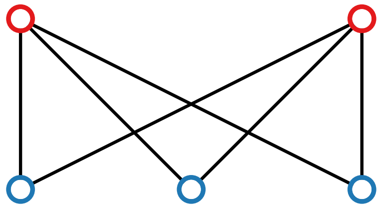
Requirements of a graph layout

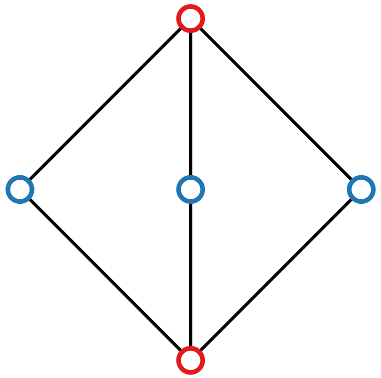
1. Drawing conventions and requirements, e.g.,
   - straight edges with $\Gamma(uv) = \overline{\Gamma(u)\Gamma(v)}$
   - orthogonal edges (i.e. with bends)
   - grid drawings
   - without crossing
2. Aesthetics to be optimized, e.g.
   - crossing/bend minimization





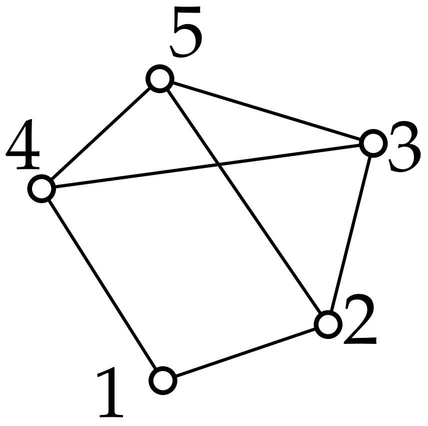


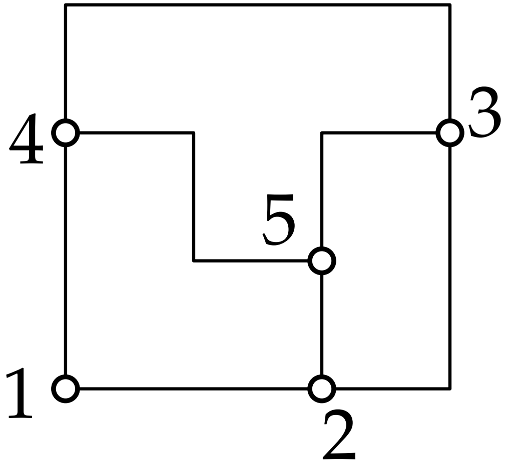


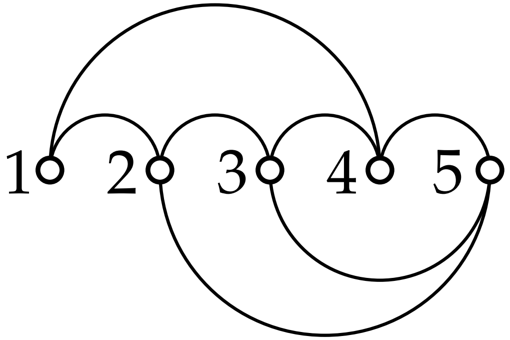


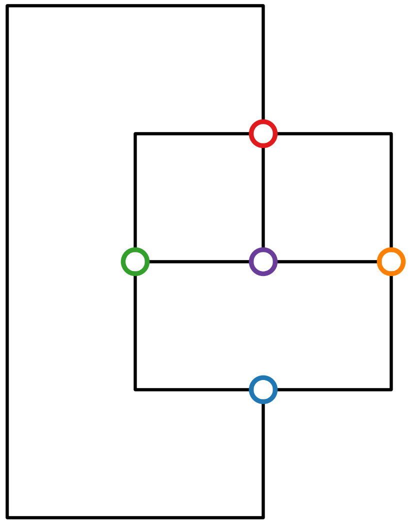
Requirements of a graph layout

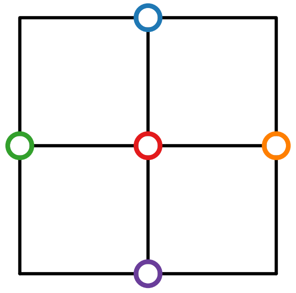
1. Drawing conventions and requirements, e.g.,
   - straight edges with $\Gamma(uv) = \overline{\Gamma(u)\Gamma(v)}$
   - orthogonal edges (i.e. with bends)
   - grid drawings
   - without crossing
2. Aesthetics to be optimized, e.g.
   - crossing/bend minimization





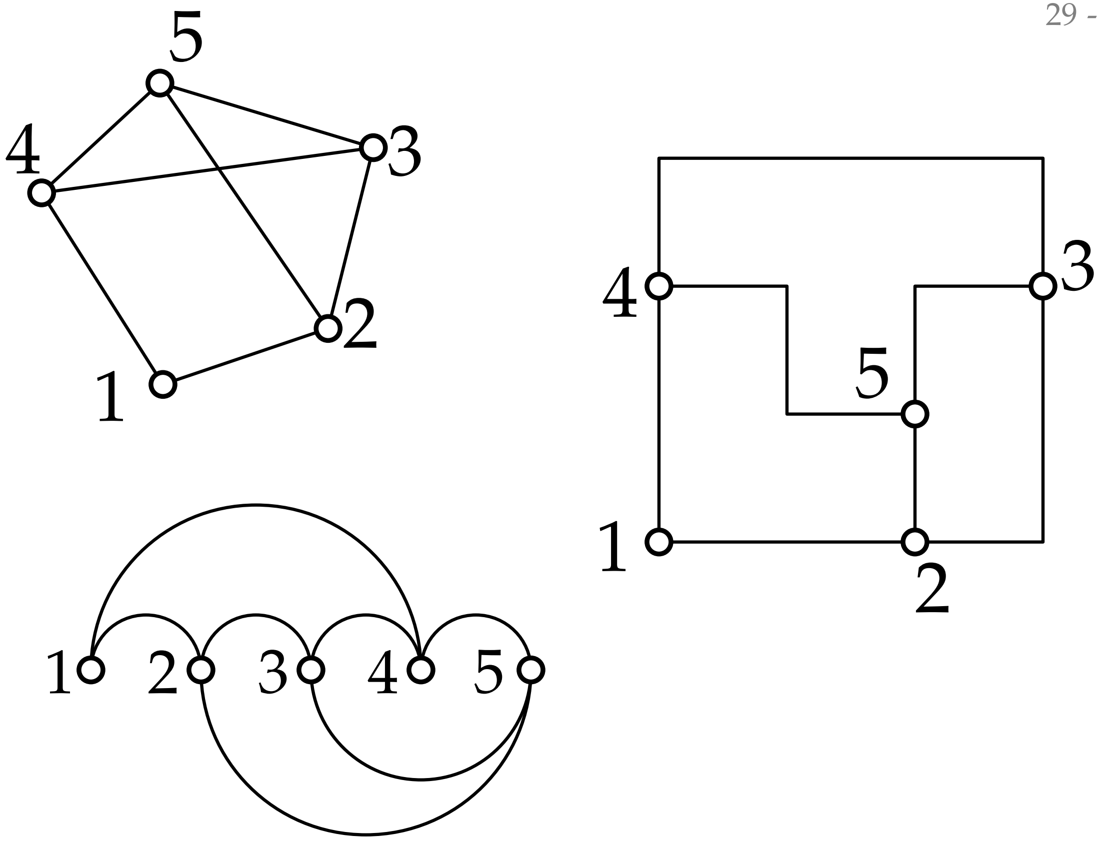


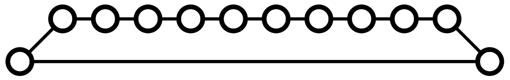


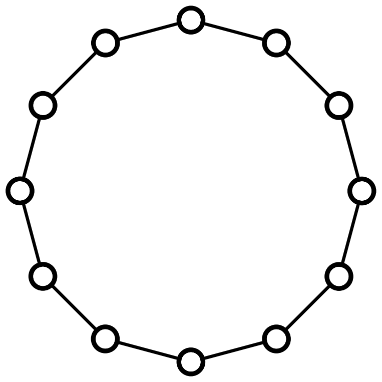


Requirements of a graph layout

1. Drawing conventions and requirements, e.g.,
   - straight edges with $\Gamma(uv) = \overline{\Gamma(u)\Gamma(v)}$
   - orthogonal edges (i.e. with bends)
   - grid drawings
   - without crossing
2. Aesthetics to be optimized, e.g.
   - crossing/bend minimization
   - edge length uniformity

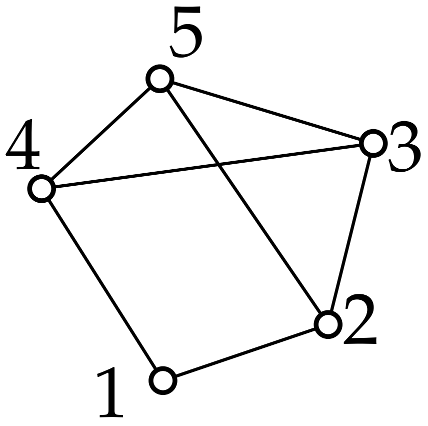


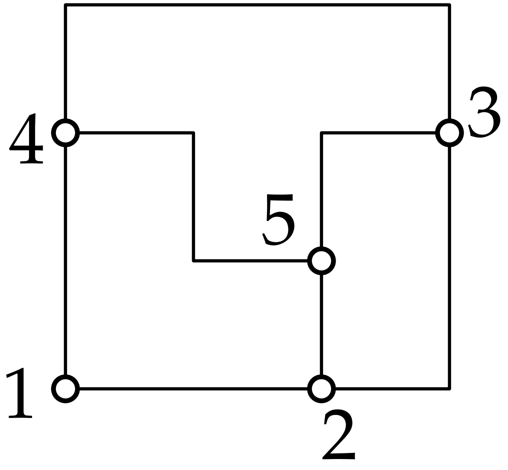


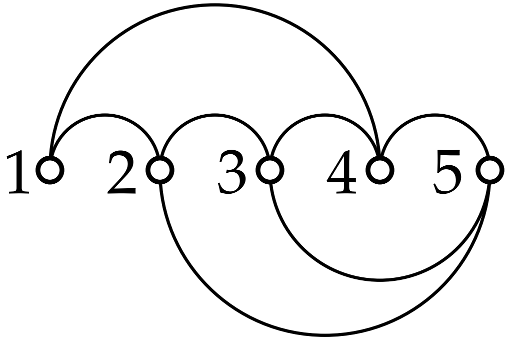


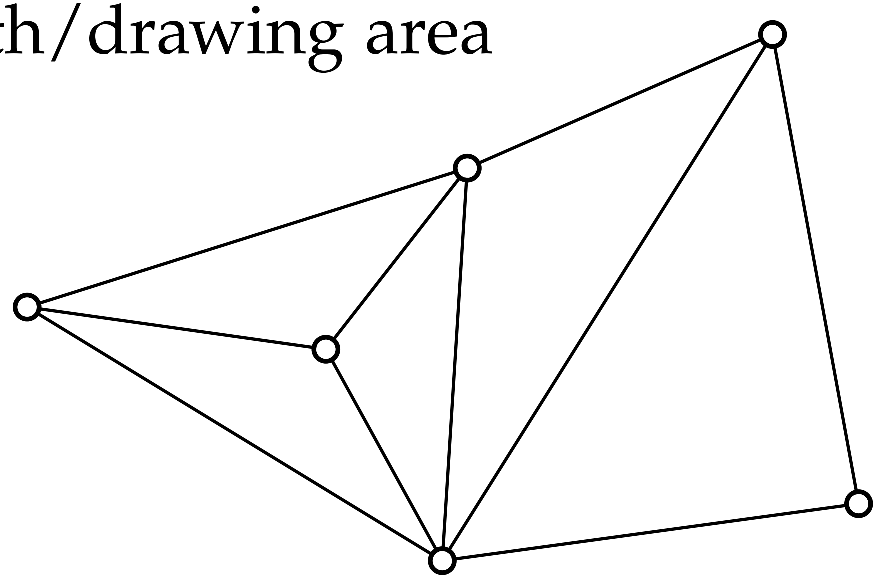
Requirements of a graph layout

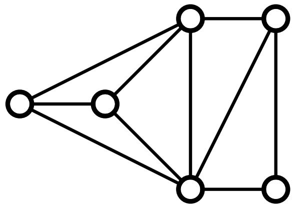
1. Drawing conventions and requirements, e.g.,
   - straight edges with $\Gamma(uv) = \overline{\Gamma(u)\Gamma(v)}$
   - orthogonal edges (i.e. with bends)
   - grid drawings
   - without crossing
2. Aesthetics to be optimized, e.g.
   - crossing/bend minimization
   - edge length uniformity
   - minimizing total edge length/drawing area



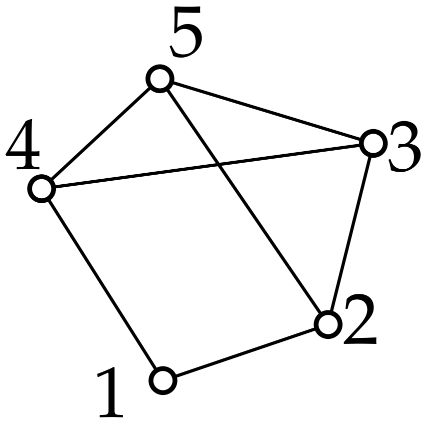


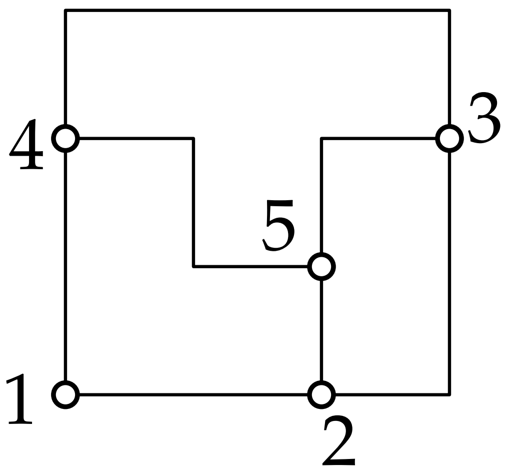


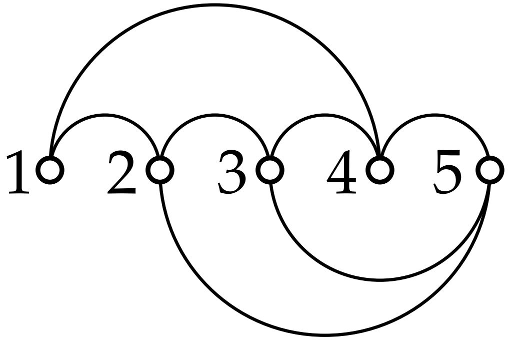




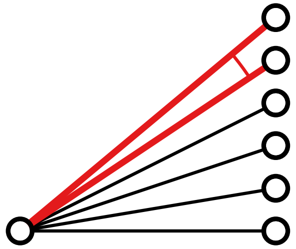
Requirements of a graph layout

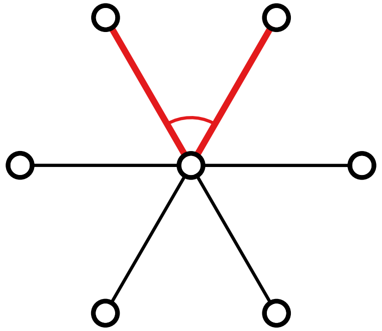






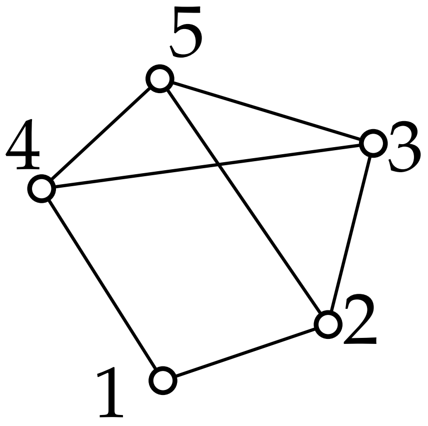
1. Drawing conventions and requirements, e.g.,
   - straight edges with $\Gamma(uv) = \overline{\Gamma(u)\Gamma(v)}$
   - orthogonal edges (i.e. with bends)
   - grid drawings
   - without crossing
2. Aesthetics to be optimized, e.g.
   - crossing/bend minimization
   - edge length uniformity
   - minimizing total edge length/drawing area
   - angular resolution

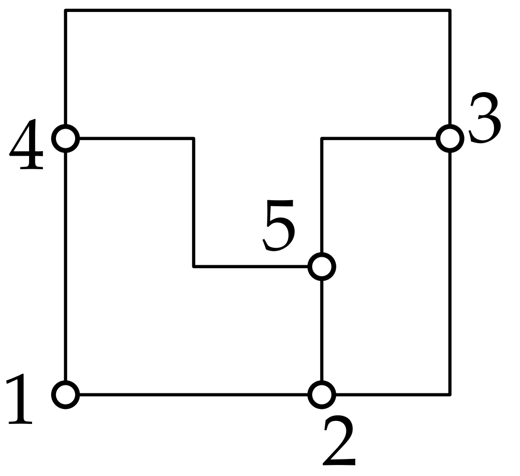


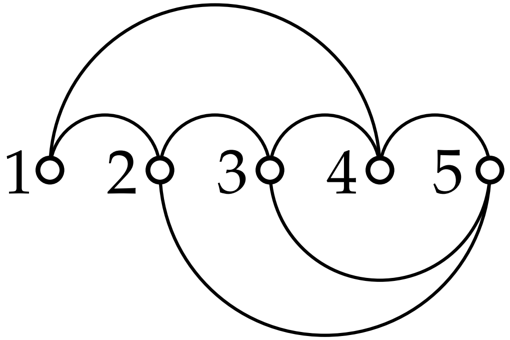


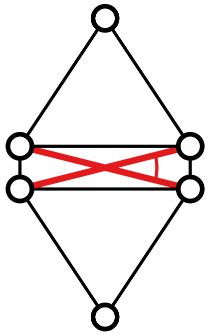
Requirements of a graph layout

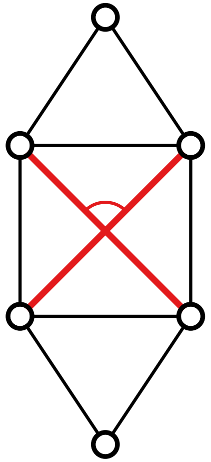
1. Drawing conventions and requirements, e.g.,
   - straight edges with $\Gamma(uv) = \overline{\Gamma(u)\Gamma(v)}$
   - orthogonal edges (i.e. with bends)
   - grid drawings
   - without crossing
2. Aesthetics to be optimized, e.g.
   - crossing/bend minimization
   - edge length uniformity
   - minimizing total edge length/drawing area
   - angular resolution



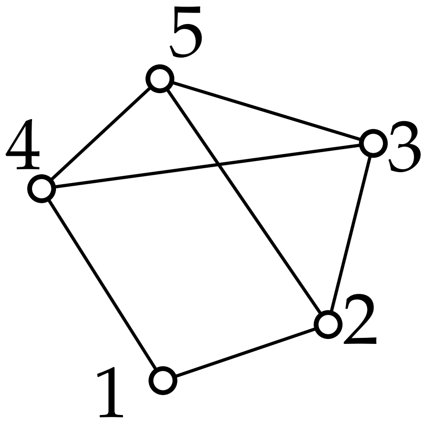


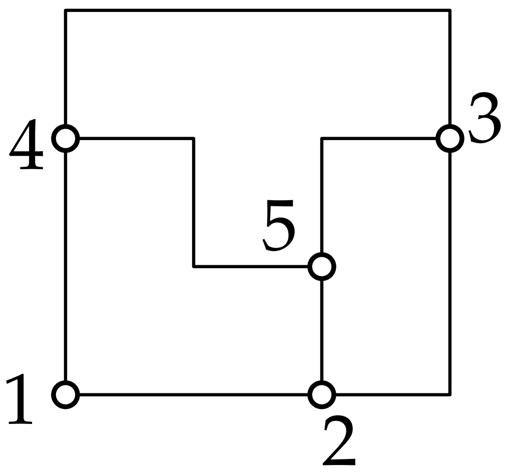




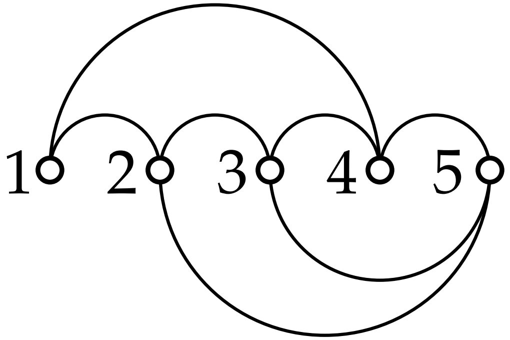


Requirements of a graph layout

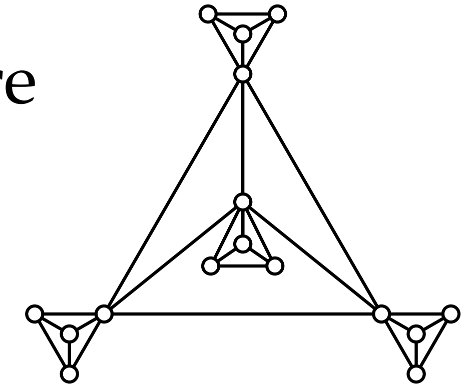


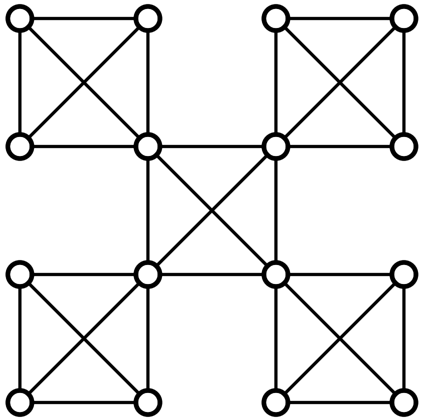


1. Drawing conventions and requirements, e.g.,
   - straight edges with $\Gamma(uv) = \overline{\Gamma(u)\Gamma(v)}$
   - orthogonal edges (i.e. with bends)
   - grid drawings
   - without crossing



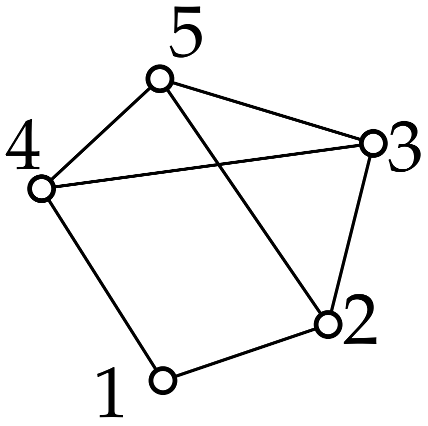
2. Aesthetics to be optimized, e.g.
   - crossing/bend minimization
   - edge length uniformity
   - minimizing total edge length/drawing area
   - angular resolution
   - symmetry/structure

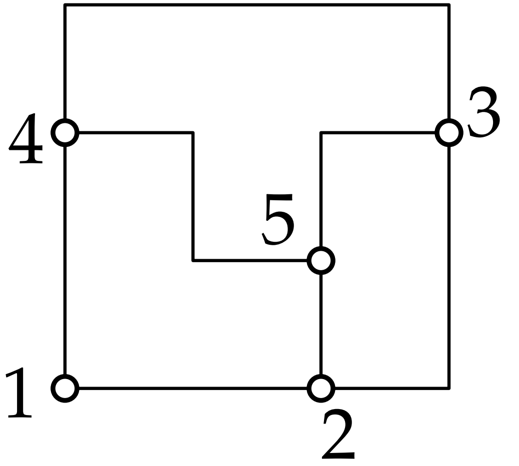


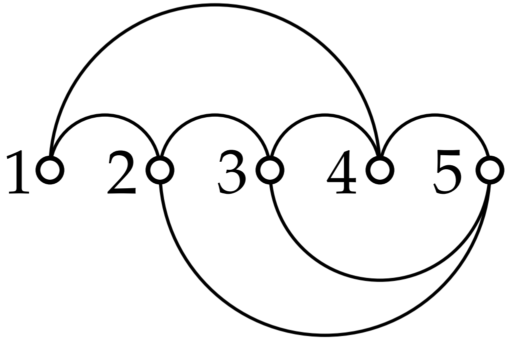


Requirements of a graph layout

1. Drawing conventions and requirements, e.g.,
   - straight edges with $\Gamma(uv) = \overline{\Gamma(u)\Gamma(v)}$
   - orthogonal edges (i.e. with bends)
   - grid drawings
   - without crossing
2. Aesthetics to be optimized, e.g.
   - crossing/bend minimization
   - edge length uniformity
   - minimizing total edge length/drawing area
   - angular resolution
   - symmetry/structure



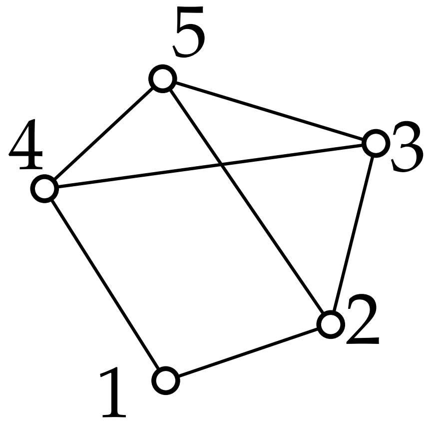


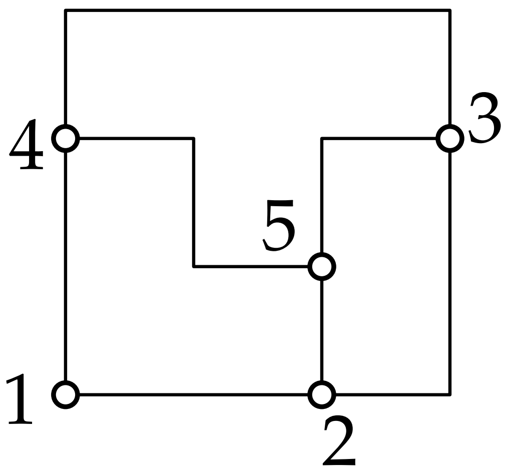


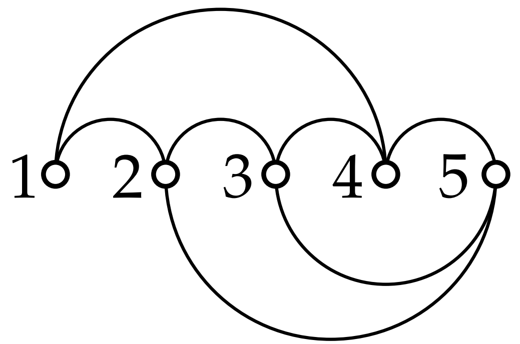
→ lead to NP-hard optimization problems

Requirements of a graph layout

1. Drawing conventions and requirements, e.g.,
   - straight edges with $\Gamma(uv) = \overline{\Gamma(u)\Gamma(v)}$
   - orthogonal edges (i.e. with bends)
   - grid drawings
   - without crossing
2. Aesthetics to be optimized, e.g.
   - crossing/bend minimization
   - edge length uniformity
   - minimizing total edge length/drawing area
   - angular resolution
   - symmetry/structure





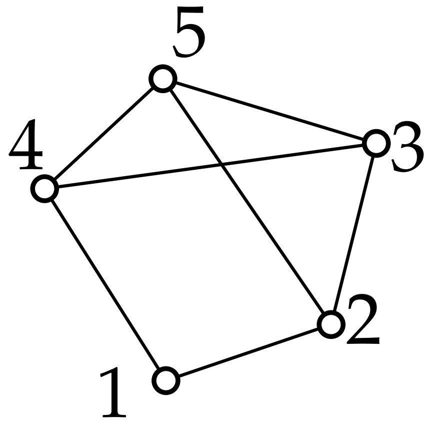


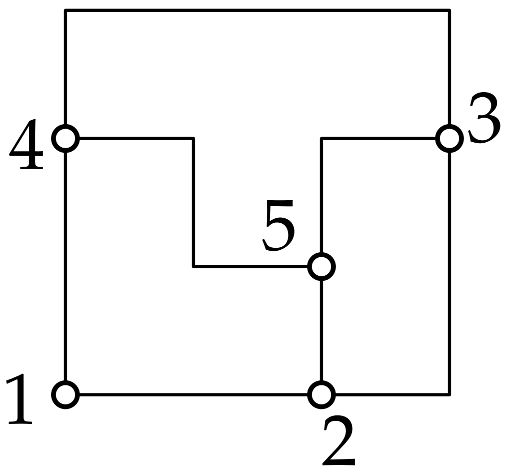
→ lead to NP-hard optimization problems

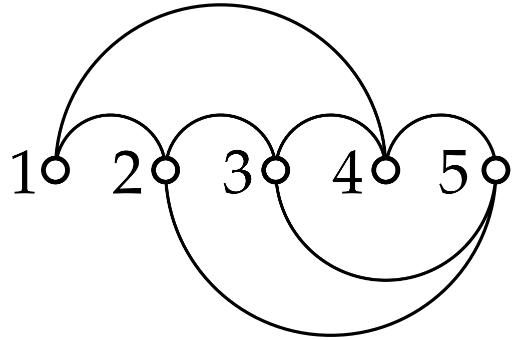
→ such criteria are often inversely related

Requirements of a graph layout

1. Drawing conventions and requirements, e.g.,
   - straight edges with $\Gamma(uv) = \overline{\Gamma(u)\Gamma(v)}$
   - orthogonal edges (i.e. with bends)
   - grid drawings
   - without crossing
2. Aesthetics to be optimized, e.g.
   - crossing/bend minimization
   - edge length uniformity
   - minimizing total edge length/drawing area
   - angular resolution
   - symmetry/structure
3. Local Constraints, e.g.





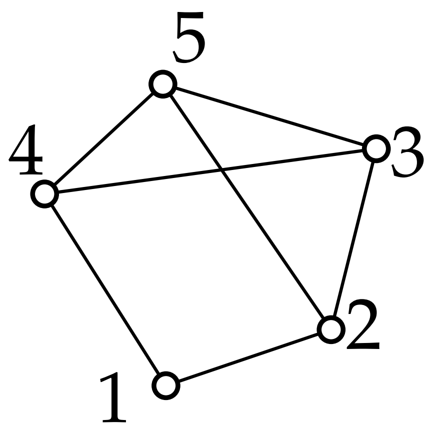


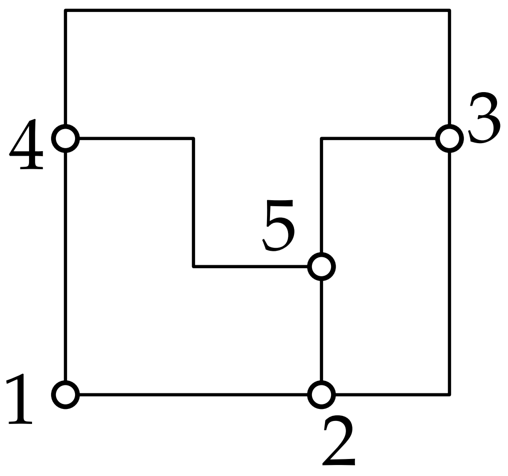
→ lead to NP-hard optimization problems

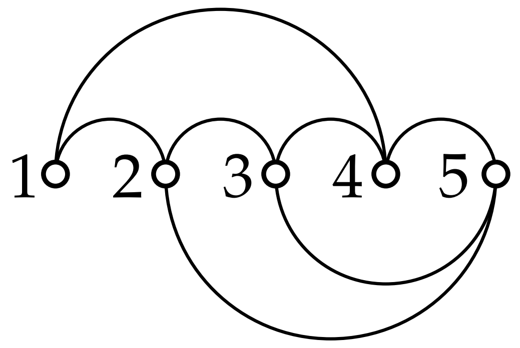
→ such criteria are often inversely related

Requirements of a graph layout

1. Drawing conventions and requirements, e.g.,
   - straight edges with $\Gamma(uv) = \overline{\Gamma(u)\Gamma(v)}$
   - orthogonal edges (i.e. with bends)
   - grid drawings
   - without crossing
2. Aesthetics to be optimized, e.g.
   - crossing/bend minimization
   - edge length uniformity
   - minimizing total edge length/drawing area
   - angular resolution
   - symmetry/structure
3. Local Constraints, e.g.
   - restrictions on neighboring vertices (e.g., “upward”).
   - restrictions on groups of vertices/edges (e.g., “clustered”).







→ lead to NP-hard optimization problems

→ such criteria are often inversely related

The layout problem

Graph Visualization Problem

in: Graph $G = (V, E)$

out: Drawing Γ of G such that

The layout problem

Graph Visualization Problem

in: Graph $G = (V, E)$

out: Drawing Γ of G such that

- **drawing conventions** are met,

The layout problem

Graph Visualization Problem

in: Graph $G = (V, E)$

out: Drawing Γ of G such that

- **drawing conventions** are met,
- **aesthetic criteria** are optimised, and

The layout problem

Graph Visualization Problem

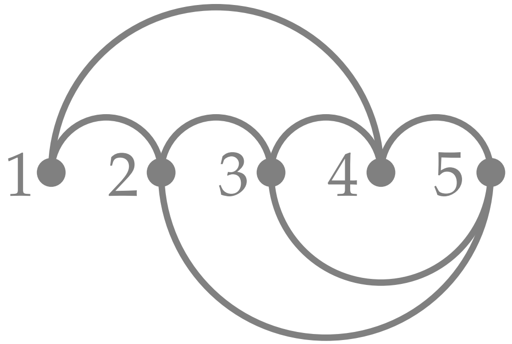
in: Graph $G = (V, E)$

out: Drawing Γ of G such that

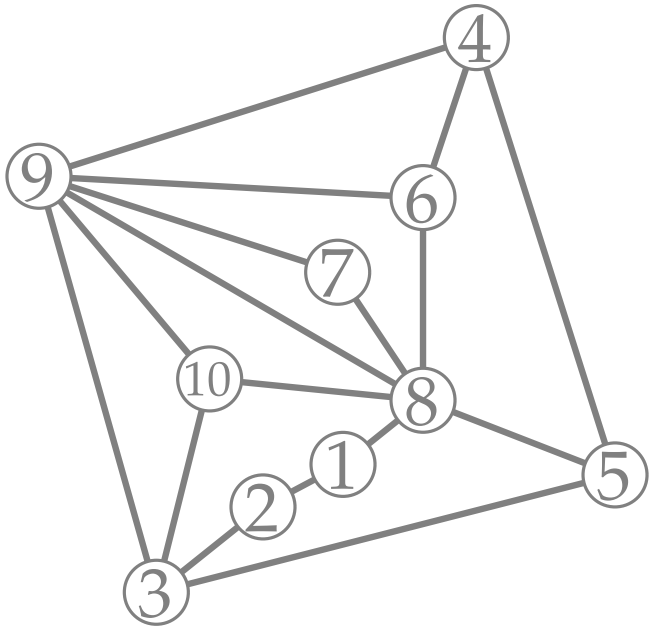
- **drawing conventions** are met,
- **aesthetic criteria** are optimised, and
- some **additional constraints** are satisfied.

Universität Trier

Visualization of Graphs



Lecture 1:
The Graph Visualization Problem

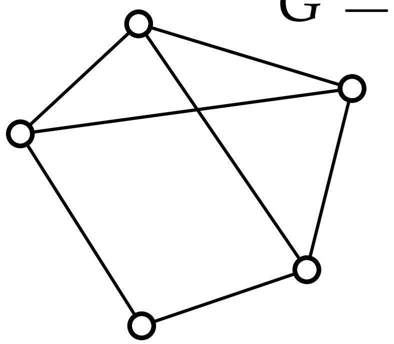


Part III:
Basics

Philipp Kindermann

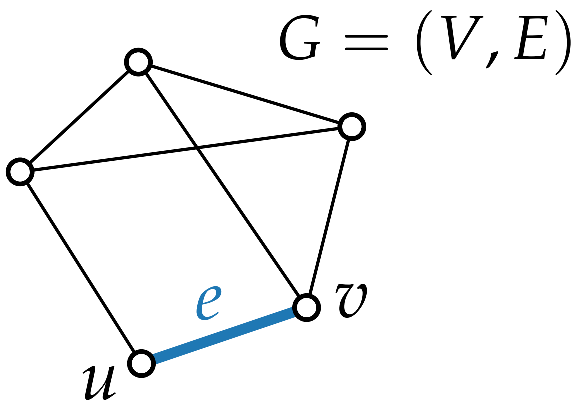
Summer Semester 2021

Basic Definitions



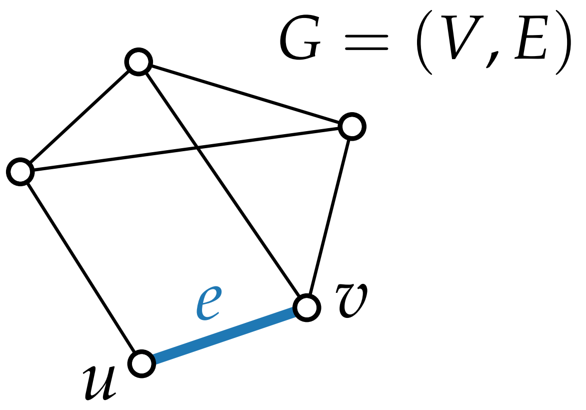
$G = (V, E)$

Basic Definitions



Edge $e = \{u, v\} \in E$:

Basic Definitions

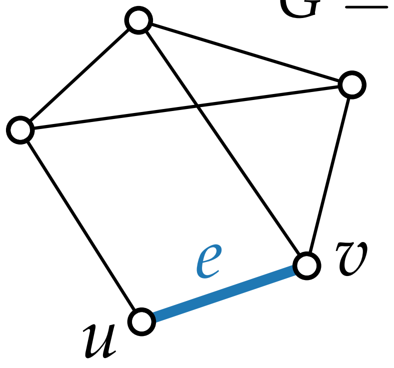


Edge $e = \{u, v\} \in E$:

- e **incident** to u and v

Basic Definitions

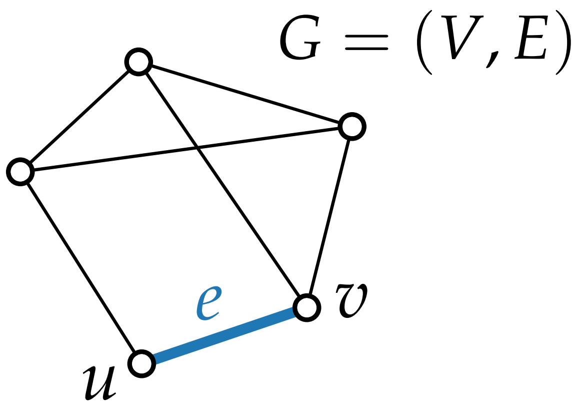
$G = (V, E)$



Edge $e = \{u, v\} \in E$:

- e **incident** to u and v
- u, v **end vertices** of e

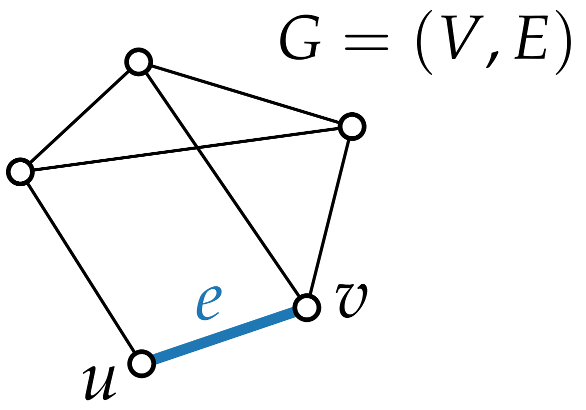
Basic Definitions



Edge $e = \{u, v\} \in E$:

- e **incident** to u and v
- u, v **end vertices** of e
- u **adjacent** to v

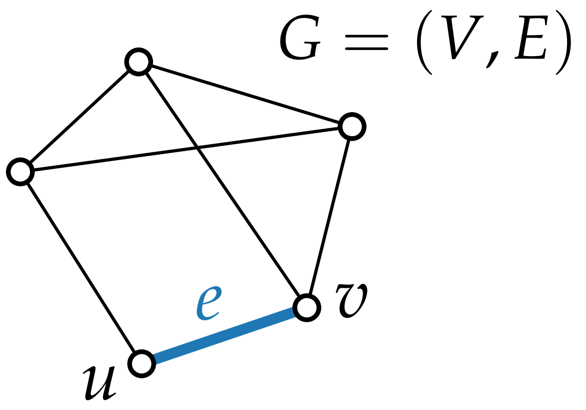
Basic Definitions



Edge $e = \{u, v\} \in E$:

- e **incident** to u and v
- u, v **end vertices** of e
- u **adjacent** to v
- u and v are **neighbors**

Basic Definitions



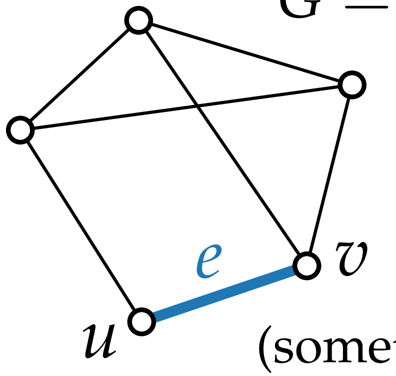
Edge $e = \{u, v\} \in E$:

- e **incident** to u and v
- u, v **end vertices** of e
- u **adjacent** to v
- u and v are **neighbors**

degree $\deg(v)$:
number of edges incident to v

Basic Definitions

$G = (V, E)$



(sometimes $e = uv$ or $e = (u, v)$)

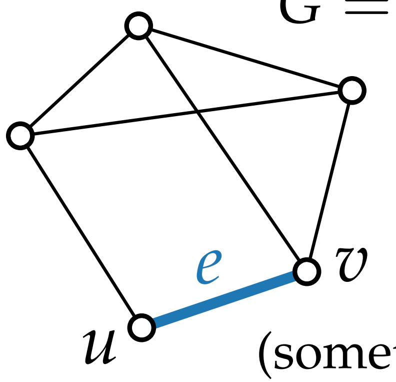
Edge $e = \{u, v\} \in E$:

- e **incident** to u and v
- u, v **end vertices** of e
- u **adjacent** to v
- u and v are **neighbors**

degree $\deg(v)$:
number of edges incident to v

Basic Definitions

$G = (V, E)$



(sometimes $e = uv$ or $e = (u, v)$)

Edge $e = \{u, v\} \in E$:

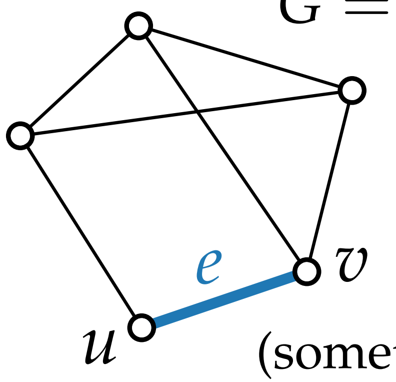
- e **incident** to u and v
- u, v **end vertices** of e
- u **adjacent** to v
- u and v are **neighbors**

degree $\deg(v)$:
number of edges incident to v

Handshaking-Lemma.
$\sum_{v \in V} \deg(v) =$

Basic Definitions

$G = (V, E)$



(sometimes $e = uv$ or $e = (u, v)$)

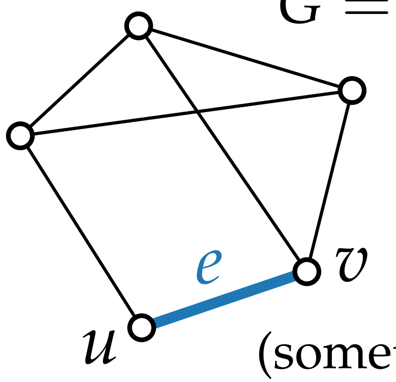
Edge $e = \{u, v\} \in E$:

- e **incident** to u and v
- u, v **end vertices** of e
- u **adjacent** to v
- u and v are **neighbors**

degree $\deg(v)$:
number of edges incident to v

Handshaking-Lemma.
$\sum_{v \in V} \deg(v) = 2|E|$

Basic Definitions



(sometimes $e = uv$ or $e = (u, v)$)

Edge $e = \{u, v\} \in E$:

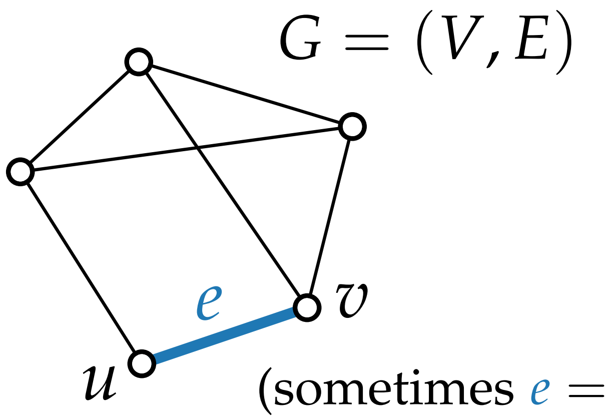
- e **incident** to u and v
- u, v **end vertices** of e
- u **adjacent** to v
- u and v are **neighbors**

degree $\deg(v)$:
number of edges incident to v

Handshaking-Lemma.
$\sum_{v \in V} \deg(v) = 2|E|$

Corollary.
The number of odd-degree vertices is even.

Basic Definitions



(sometimes $e = uv$ or $e = (u, v)$)

Edge $e = \{u, v\} \in E$:

- e **incident** to u and v
- u, v **end vertices** of e
- u **adjacent** to v
- u and v are **neighbors**

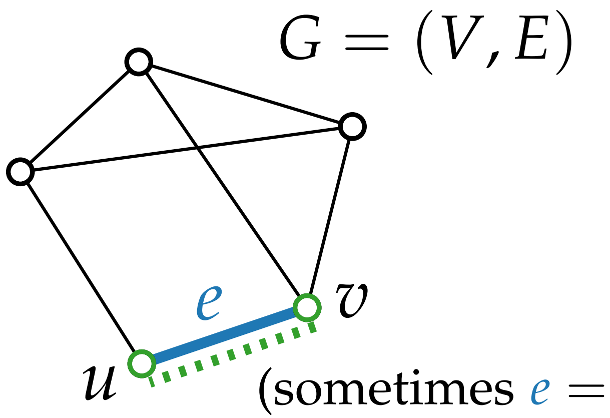
degree $\deg(v)$:
number of edges incident to v

u-v-path of length ℓ:
Sequence of $\ell + 1$ distinct adjacent vertices (and ℓ connecting edges), starting with u and ending with v:
$u - \{u, v_1\} - v_1 - \cdots - v_{\ell-1} - \{v_{\ell-1}, v\} - v$

Handshaking-Lemma.
$\sum_{v \in V} \deg(v) = 2|E|$

Corollary.
The number of odd-degree vertices is even.

Basic Definitions



(sometimes $e = uv$ or $e = (u, v)$)

Edge $e = \{u, v\} \in E$:

- e **incident** to u and v
- u, v **end vertices** of e
- u **adjacent** to v
- u and v are **neighbors**

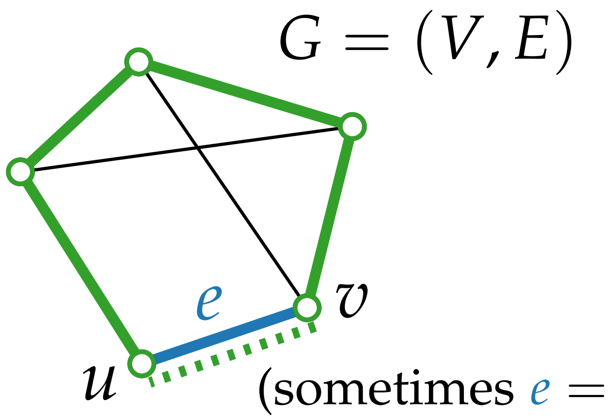
degree $\deg(v)$:
number of edges incident to v

u-v-path of length ℓ:
Sequence of $\ell + 1$ distinct adjacent vertices (and ℓ connecting edges), starting with u and ending with v:
$u - \{u, v_1\} - v_1 - \cdots - v_{\ell-1} - \{v_{\ell-1}, v\} - v$

Handshaking-Lemma.
$\sum_{v \in V} \deg(v) = 2|E|$

Corollary.
The number of odd-degree vertices is even.

Basic Definitions



(sometimes $e = uv$ or $e = (u, v)$)

Edge $e = \{u, v\} \in E$:

- e **incident** to u and v
- u, v **end vertices** of e
- u **adjacent** to v
- u and v are **neighbors**

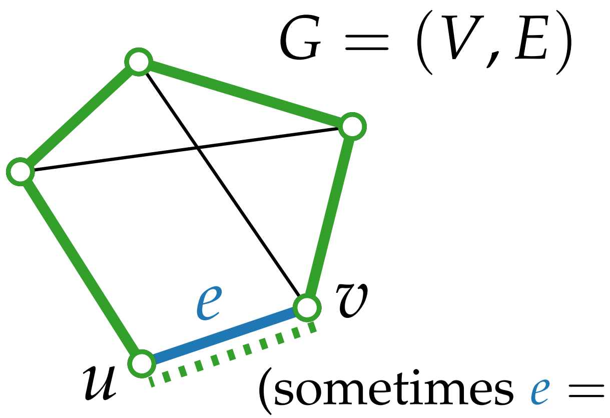
degree $\deg(v)$:
number of edges incident to v

u-v-path of length ℓ:
Sequence of $\ell + 1$ distinct adjacent vertices (and ℓ connecting edges), starting with u and ending with v:
$u - \{u, v_1\} - v_1 - \cdots - v_{\ell-1} - \{v_{\ell-1}, v\} - v$

Handshaking-Lemma.
$\sum_{v \in V} \deg(v) = 2|E|$

Corollary.
The number of odd-degree vertices is even.

Basic Definitions



(sometimes $e = uv$ or $e = (u, v)$)

Edge $e = \{u, v\} \in E$:

- e **incident** to u and v
- u, v **end vertices** of e
- u **adjacent** to v
- u and v are **neighbors**

degree $\deg(v)$:
number of edges incident to v

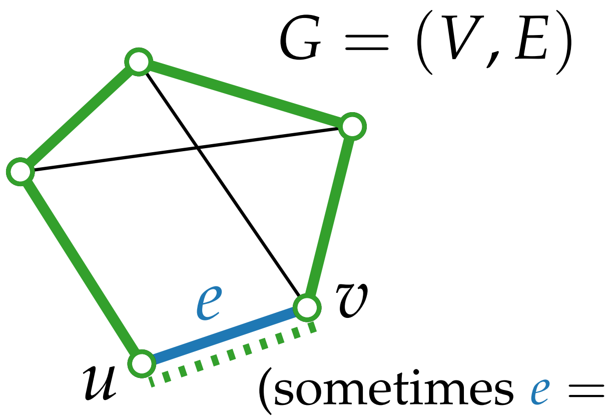
u-v-path of length ℓ:
Sequence of $\ell + 1$ distinct adjacent vertices (and ℓ connecting edges), starting with u and ending with v:
$u - \{u, v_1\} - v_1 - \cdots - v_{\ell-1} - \{v_{\ell-1}, v\} - v$

simple cycle: u-u-path

Handshaking-Lemma.
$\sum_{v \in V} \deg(v) = 2|E|$

Corollary.
The number of odd-degree vertices is even.

Basic Definitions



(sometimes $e = uv$ or $e = (u, v)$

Edge $e = \{u, v\} \in E$:

- e **incident** to u and v
- u, v **end vertices** of e
- u **adjacent** to v
- u and v are **neighbors**

degree $\deg(v)$:
number of edges incident to v

***u*-*v*-path of length ℓ:**
Sequence of $\ell + 1$ distinct adjacent vertices (and ℓ connecting edges), starting with u and ending with v:
$u - \{u, v_1\} - v_1 - \cdots - v_{\ell-1} - \{v_{\ell-1}, v\} - v$

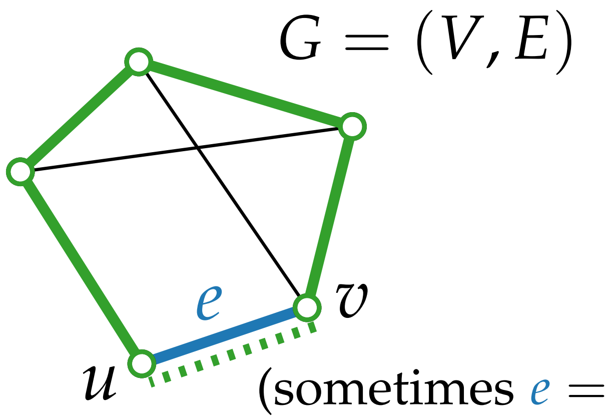
simple cycle: u-u-path

connected: There is a u-v-path for every $u, v \in V$

Handshaking-Lemma.
$\sum_{v \in V} \deg(v) = 2|E|$

Corollary.
The number of odd-degree vertices is even.

Basic Definitions



(sometimes $e = uv$ or $e = (u, v)$

Edge $e = \{u, v\} \in E$:

- e **incident** to u and v
- u, v **end vertices** of e
- u **adjacent** to v
- u and v are **neighbors**

degree $\deg(v)$:
number of edges incident to v

u-v-path of length ℓ:
Sequence of $\ell + 1$ distinct adjacent vertices (and ℓ connecting edges), starting with u and ending with v:
$u - \{u, v_1\} - v_1 - \cdots - v_{\ell-1} - \{v_{\ell-1}, v\} - v$

simple cycle: u-u-path

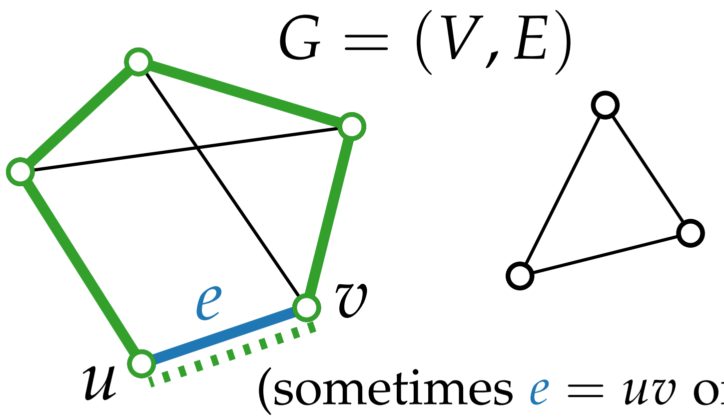
connected: There is a u-v-path for every $u, v \in V$

v **reachable** from u: There is a u-v-path

Handshaking-Lemma.
$\sum_{v \in V} \deg(v) = 2|E|$

Corollary.
The number of odd-degree vertices is even.

Basic Definitions



(sometimes $e = uv$ or $e = (u, v)$)

Edge $e = \{u, v\} \in E$:

- e **incident** to u and v
- u, v **end vertices** of e
- u **adjacent** to v
- u and v are **neighbors**

degree $\deg(v)$:
number of edges incident to v

u-v-path of length ℓ:
Sequence of $\ell + 1$ distinct adjacent vertices (and ℓ connecting edges), starting with u and ending with v:
$u - \{u, v_1\} - v_1 - \cdots - v_{\ell-1} - \{v_{\ell-1}, v\} - v$

simple cycle: u-u-path

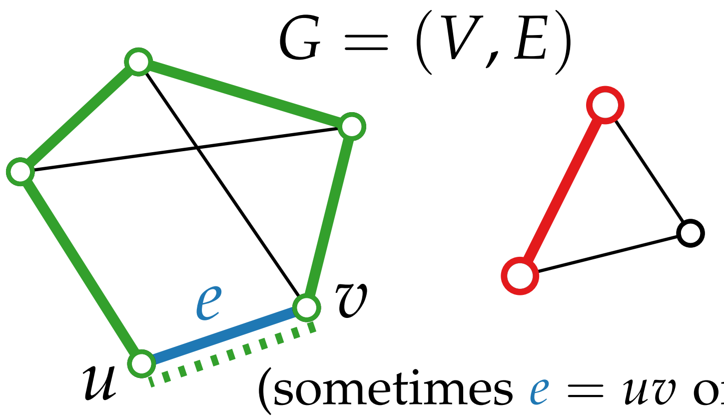
connected: There is a u-v-path for every $u, v \in V$

v **reachable** from u: There is a u-v-path

Handshaking-Lemma.
$\sum_{v \in V} \deg(v) = 2|E|$

Corollary.
The number of odd-degree vertices is even.

Basic Definitions



(sometimes $e = uv$ or $e = (u, v)$)

Edge $e = \{u, v\} \in E$:

- e **incident** to u and v
- u, v **end vertices** of e
- u **adjacent** to v
- u and v are **neighbors**

degree $\deg(v)$:
number of edges incident to v

u-v-path of length ℓ:
Sequence of $\ell + 1$ distinct adjacent vertices (and ℓ connecting edges), starting with u and ending with v:
$u - \{u, v_1\} - v_1 - \dots - v_{\ell-1} - \{v_{\ell-1}, v\} - v$

simple cycle: u-u-path

connected: There is a u-v-path for every $u, v \in V$

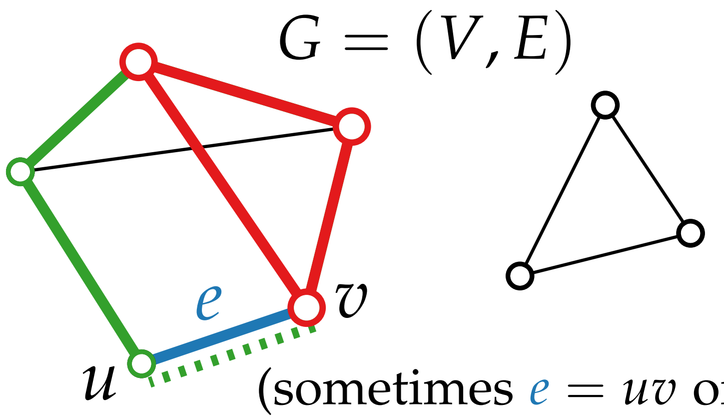
v **reachable** from u: There is a u-v-path

subgraph: graph $G' = (V', E')$ with $V' \subseteq V$ and $E' \subseteq E$

Handshaking-Lemma.
$\sum_{v \in V} \deg(v) = 2|E|$

Corollary.
The number of odd-degree vertices is even.

Basic Definitions



(sometimes $e = uv$ or $e = (u, v)$)

Edge $e = \{u, v\} \in E$:

- e **incident** to u and v
- u, v **end vertices** of e
- u **adjacent** to v
- u and v are **neighbors**

degree $\deg(v)$:
number of edges incident to v

u-v-path of length ℓ:
Sequence of $\ell + 1$ distinct adjacent vertices (and ℓ connecting edges), starting with u and ending with v:
$u - \{u, v_1\} - v_1 - \cdots - v_{\ell-1} - \{v_{\ell-1}, v\} - v$

simple cycle: u-u-path

connected: There is a u-v-path for every $u, v \in V$

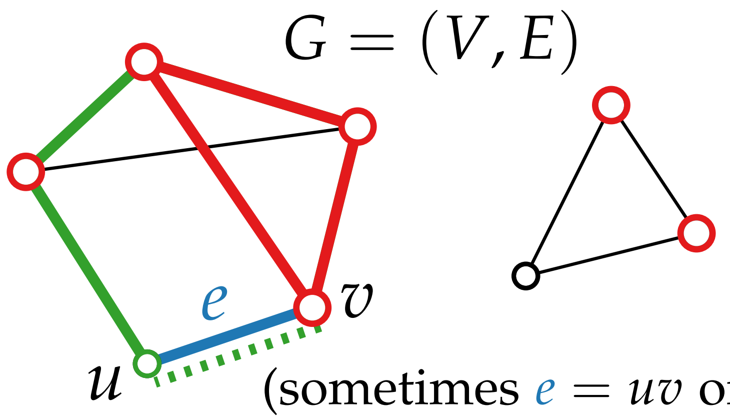
v **reachable** from u: There is a u-v-path

subgraph: graph $G' = (V', E')$ with $V' \subseteq V$ and $E' \subseteq E$

Handshaking-Lemma.
$\sum_{v \in V} \deg(v) = 2|E|$

Corollary.
The number of odd-degree vertices is even.

Basic Definitions



(sometimes $e = uv$ or $e = (u, v)$)

Edge $e = \{u, v\} \in E$:

- e **incident** to u and v
- u, v **end vertices** of e
- u **adjacent** to v
- u and v are **neighbors**

degree $\deg(v)$:
number of edges incident to v

u-v-path of length ℓ:
Sequence of $\ell + 1$ distinct adjacent vertices (and ℓ connecting edges), starting with u and ending with v:
$u - \{u, v_1\} - v_1 - \cdots - v_{\ell-1} - \{v_{\ell-1}, v\} - v$

simple cycle: u-u-path

connected: There is a u-v-path for every $u, v \in V$

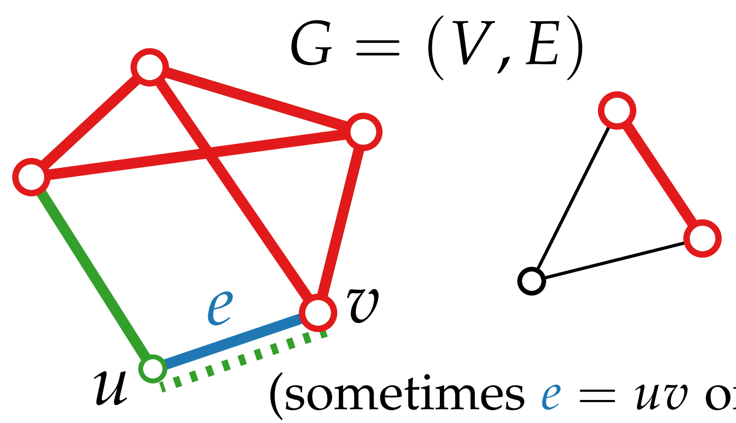
v **reachable** from u: There is a u-v-path

subgraph: graph $G' = (V', E')$ with $V' \subseteq V$ and $E' \subseteq E$

Handshaking-Lemma.
$\sum_{v \in V} \deg(v) = 2|E|$

Corollary.
The number of odd-degree vertices is even.

Basic Definitions



(sometimes $e = uv$ or $e = (u, v)$)

Edge $e = \{u, v\} \in E$:

- e **incident** to u and v
- u, v **end vertices** of e
- u **adjacent** to v
- u and v are **neighbors**

degree $\deg(v)$:
number of edges incident to v

u-v-path of length ℓ:
Sequence of $\ell + 1$ distinct adjacent vertices (and ℓ connecting edges), starting with u and ending with v:
$u - \{u, v_1\} - v_1 - \cdots - v_{\ell-1} - \{v_{\ell-1}, v\} - v$

simple cycle: u-u-path

connected: There is a u-v-path for every $u, v \in V$

v **reachable** from u: There is a u-v-path

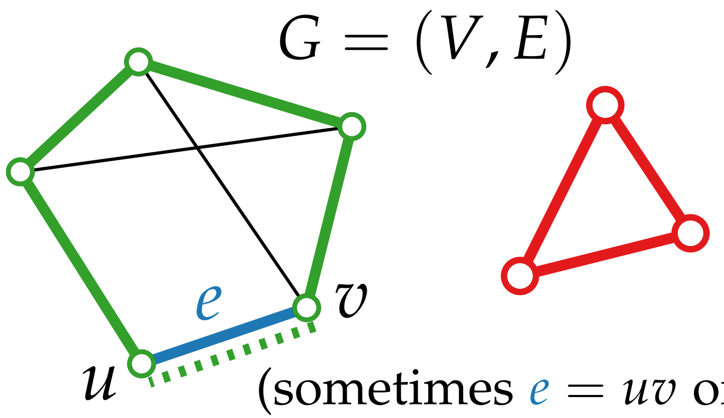
subgraph: graph $G' = (V', E')$ with $V' \subseteq V$ and $E' \subseteq E$

induced subgraph: subgraph with $E' = \binom{V'}{2} \cap E$

Handshaking-Lemma.
$\sum_{v \in V} \deg(v) = 2|E|$

Corollary.
The number of odd-degree vertices is even.

Basic Definitions



(sometimes $e = uv$ or $e = (u, v)$)

Edge $e = \{u, v\} \in E$:

- e **incident** to u and v
- u, v **end vertices** of e
- u **adjacent** to v
- u and v are **neighbors**

degree $\deg(v)$:
number of edges incident to v

u-v-path of length ℓ:
Sequence of $\ell + 1$ distinct adjacent vertices (and ℓ connecting edges), starting with u and ending with v:
$u - \{u, v_1\} - v_1 - \cdots - v_{\ell-1} - \{v_{\ell-1}, v\} - v$

simple cycle: u-u-path

connected: There is a u-v-path for every $u, v \in V$

v **reachable** from u: There is a u-v-path

subgraph: graph $G' = (V', E')$ with $V' \subseteq V$ and $E' \subseteq E$

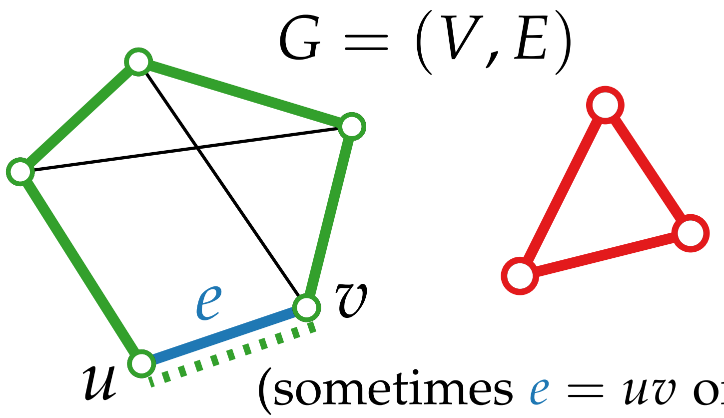
induced subgraph: subgraph with $E' = \binom{V'}{2} \cap E$

connected component: maximal connected subgraph

Handshaking-Lemma.
$\sum_{v \in V} \deg(v) = 2|E|$

Corollary.
The number of odd-degree vertices is even.

Basic Definitions



(sometimes $e = uv$ or $e = (u, v)$)

Edge $e = \{u, v\} \in E$:

- e **incident** to u and v
- u, v **end vertices** of e
- u **adjacent** to v
- u and v are **neighbors**

degree $\deg(v)$:
number of edges incident to v

u-v-path of length ℓ:
Sequence of $\ell + 1$ distinct adjacent vertices (and ℓ connecting edges), starting with u and ending with v:
$u - \{u, v_1\} - v_1 - \cdots - v_{\ell-1} - \{v_{\ell-1}, v\} - v$

simple cycle: u-u-path

connected: There is a u-v-path for every $u, v \in V$

v **reachable** from u: There is a u-v-path

subgraph: graph $G' = (V', E')$ with $V' \subseteq V$ and $E' \subseteq E$

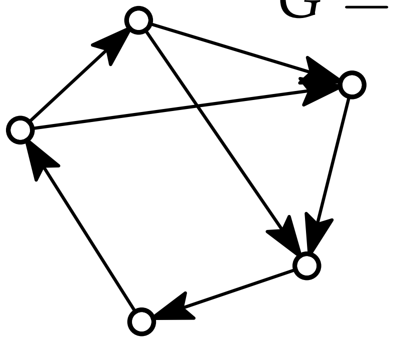
induced subgraph: subgraph with $E' = \binom{V'}{2} \cap E$

connected component: maximal connected subgraph

Handshaking-Lemma.
$\sum_{v \in V} \deg(v) = 2|E|$

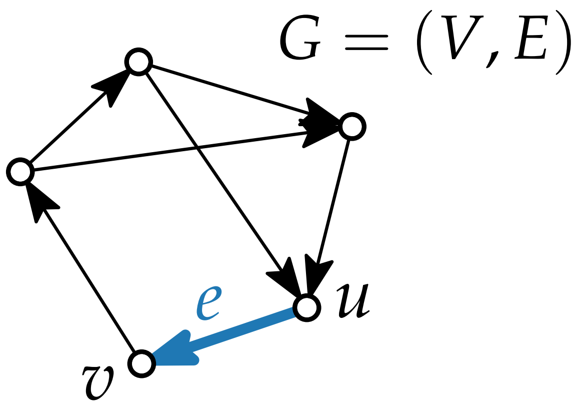
Corollary.
The number of odd-degree vertices is even.

Directed Graphs



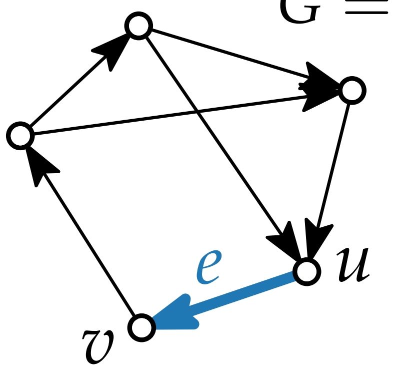
$G = (V, E)$

Directed Graphs



Edge $e = (u, v) \in E$:

Directed Graphs

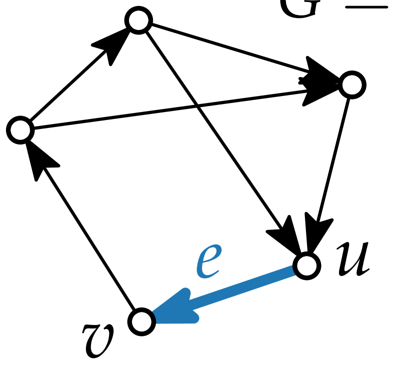


$G = (V, E)$

Edge $e = (u, v) \in E$:

- u is **source** of e

Directed Graphs

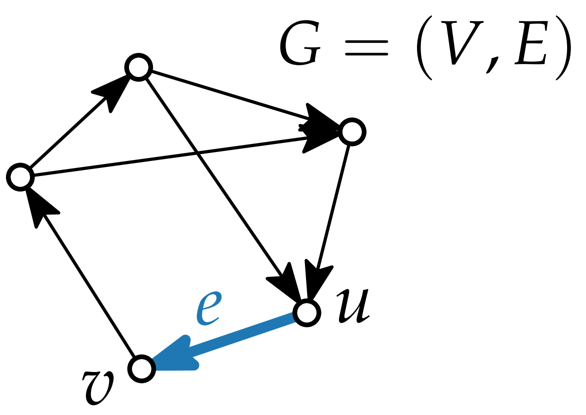


$G = (V, E)$

Edge $e = (u, v) \in E$:

- u is **source** of e
- v is **target** of e

Directed Graphs

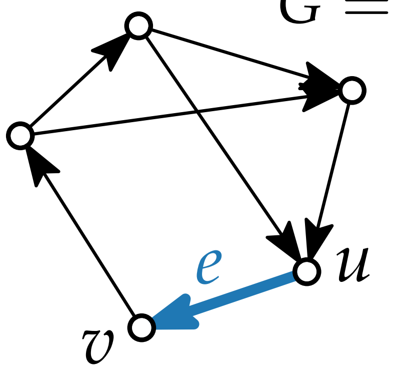


Edge $e = (u, v) \in E$:

- u is **source** of e
- v is **target** of e

indegree $\deg^-(v)$:
number of edges for which v is the target

Directed Graphs



$G = (V, E)$

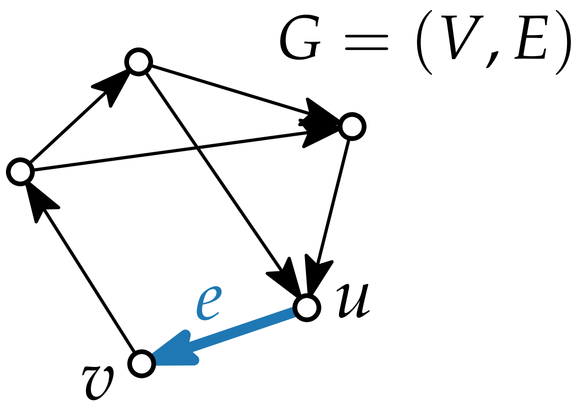
Edge $e = (u, v) \in E$:

- u is **source** of e
- v is **target** of e

indegree $\deg^-(v)$:
number of edges for which v is the target

outdegree $\deg^+(v)$:
number of edges for which v is the source

Directed Graphs



Edge $e = (u, v) \in E$:

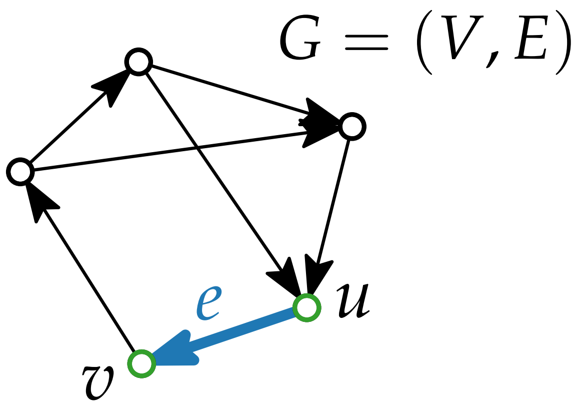
- u is **source** of e
- v is **target** of e

indegree $\deg^-(v)$:
number of edges for which v is the target

outdegree $\deg^+(v)$:
number of edges for which v is the source

Handshaking-Lemma.
$\sum_{v \in V} \deg^-(v) = \sum_{v \in V} \deg^+(v) = |E|$

Directed Graphs



Edge $e = (u, v) \in E$:

- u is **source** of e
- v is **target** of e

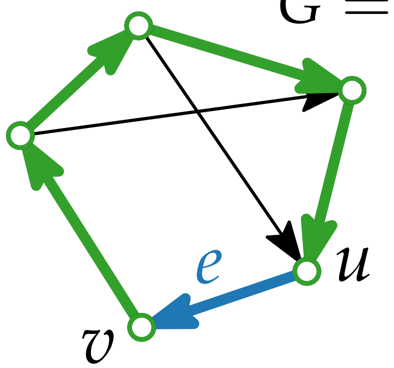
indegree $\deg^-(v)$:
number of edges for which v is the target

outdegree $\deg^+(v)$:
number of edges for which v is the source

Handshaking-Lemma.
$\sum_{v \in V} \deg^-(v) = \sum_{v \in V} \deg^+(v) = |E|$

directed u-v-path:
$u - (u, v_1) - v_1 - \cdots - v_{\ell-1} - (v_{\ell-1}, v) - v$

Directed Graphs



Edge $e = (u, v) \in E$:

- u is **source** of e
- v is **target** of e

indegree $\deg^-(v)$:
number of edges for which v is the target

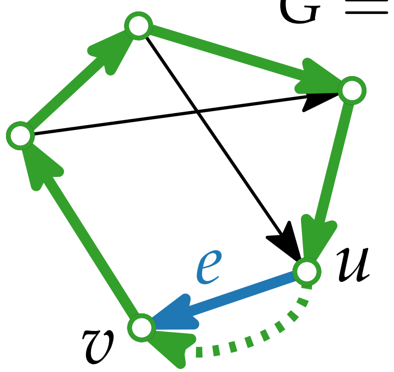
outdegree $\deg^+(v)$:
number of edges for which v is the source

Handshaking-Lemma.
$\sum_{v \in V} \deg^-(v) = \sum_{v \in V} \deg^+(v) = |E|$

directed u-v-path:
$u - (u, v_1) - v_1 - \cdots - v_{\ell-1} - (v_{\ell-1}, v) - v$

Directed Graphs

$G = (V, E)$



Edge $e = (u, v) \in E$:

- u is **source** of e
- v is **target** of e

indegree $\deg^-(v)$:
number of edges for which v is the target

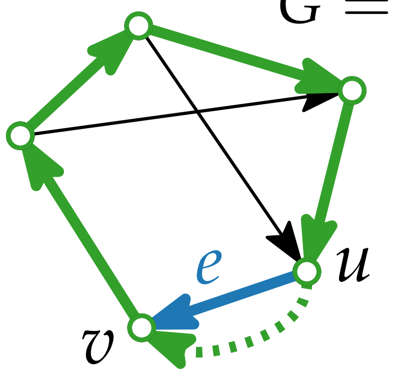
outdegree $\deg^+(v)$:
number of edges for which v is the source

Handshaking-Lemma.
$\sum_{v \in V} \deg^-(v) = \sum_{v \in V} \deg^+(v) = |E|$

directed u-v-path:
$u - (u, v_1) - v_1 - \cdots - v_{\ell-1} - (v_{\ell-1}, v) - v$

directed cycle: directed u-u-path

Directed Graphs



Edge $e = (u, v) \in E$:

- u is **source** of e
- v is **target** of e

indegree $\deg^-(v)$:
number of edges for which v is the target

outdegree $\deg^+(v)$:
number of edges for which v is the source

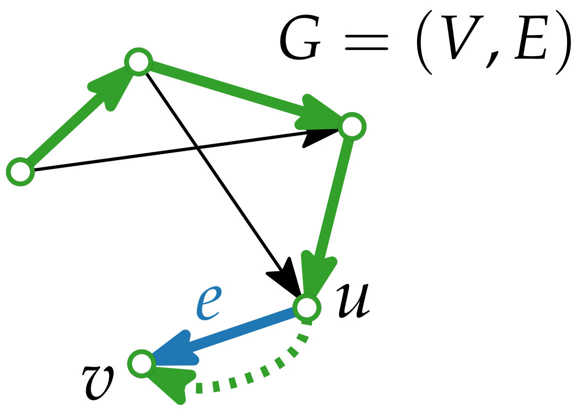
Handshaking-Lemma.
$\sum_{v \in V} \deg^-(v) = \sum_{v \in V} \deg^+(v) = |E|$

directed u-v-path:
$u - (u, v_1) - v_1 - \cdots - v_{\ell-1} - (v_{\ell-1}, v) - v$

directed cycle: directed u-u-path

acyclic: no directed cycles

Directed Graphs



Edge $e = (u, v) \in E$:

- u is **source** of e
- v is **target** of e

indegree $\deg^-(v)$:
number of edges for which v is the target

outdegree $\deg^+(v)$:
number of edges for which v is the source

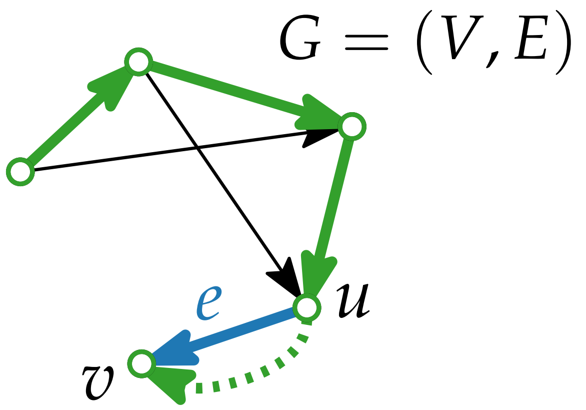
Handshaking-Lemma.
$\sum_{v \in V} \deg^-(v) = \sum_{v \in V} \deg^+(v) = |E|$

directed u-v-path:
$u - (u, v_1) - v_1 - \cdots - v_{\ell-1} - (v_{\ell-1}, v) - v$

directed cycle: directed u-u-path

acyclic: no directed cycles

Directed Graphs



Edge $e = (u, v) \in E$:

- u is **source** of e
- v is **target** of e

indegree $\deg^-(v)$:
number of edges for which v is the target

outdegree $\deg^+(v)$:
number of edges for which v is the source

Handshaking-Lemma.
$\sum_{v \in V} \deg^-(v) = \sum_{v \in V} \deg^+(v) = |E|$

directed u-v-path:
$u - (u, v_1) - v_1 - \cdots - v_{\ell-1} - (v_{\ell-1}, v) - v$

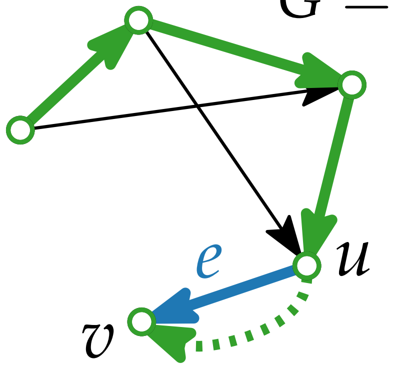
directed cycle: directed u-u-path

acyclic: no directed cycles

connected: There is a directed u-v-path or v-u-path for every $u, v \in V$

Directed Graphs

$G = (V, E)$



Edge $e = (u, v) \in E$:

- u is **source** of e
- v is **target** of e

indegree $\deg^-(v)$:
number of edges for which v is the target

outdegree $\deg^+(v)$:
number of edges for which v is the source

Handshaking-Lemma.
$\sum_{v \in V} \deg^-(v) = \sum_{v \in V} \deg^+(v) = |E|$

directed u-v-path:
$u - (u, v_1) - v_1 - \cdots - v_{\ell-1} - (v_{\ell-1}, v) - v$

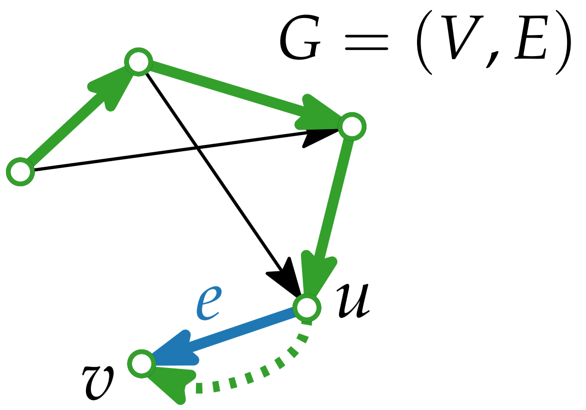
directed cycle: directed u-u-path

acyclic: no directed cycles

connected: There is a directed u-v-path or v-u-path for every $u, v \in V$

v **reachable** from u: There is a directed u-v-path

Directed Graphs



Edge $e = (u, v) \in E$:

- u is **source** of e
- v is **target** of e

indegree $\deg^-(v)$:
number of edges for which v is the target

outdegree $\deg^+(v)$:
number of edges for which v is the source

Handshaking-Lemma.
$\sum_{v \in V} \deg^-(v) = \sum_{v \in V} \deg^+(v) = |E|$

directed u-v-path:
$u - (u, v_1) - v_1 - \cdots - v_{\ell-1} - (v_{\ell-1}, v) - v$

directed cycle: directed u-u-path

acyclic: no directed cycles

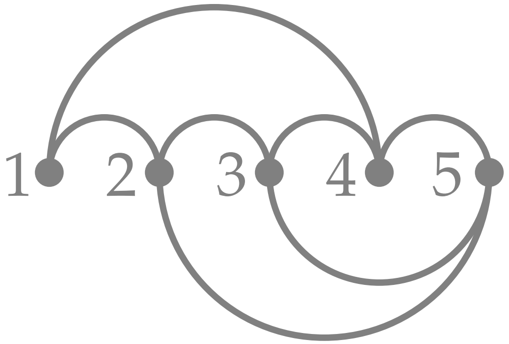
connected: There is a directed u-v-path or v-u-path for every $u, v \in V$

v **reachable** from u: There is a directed u-v-path

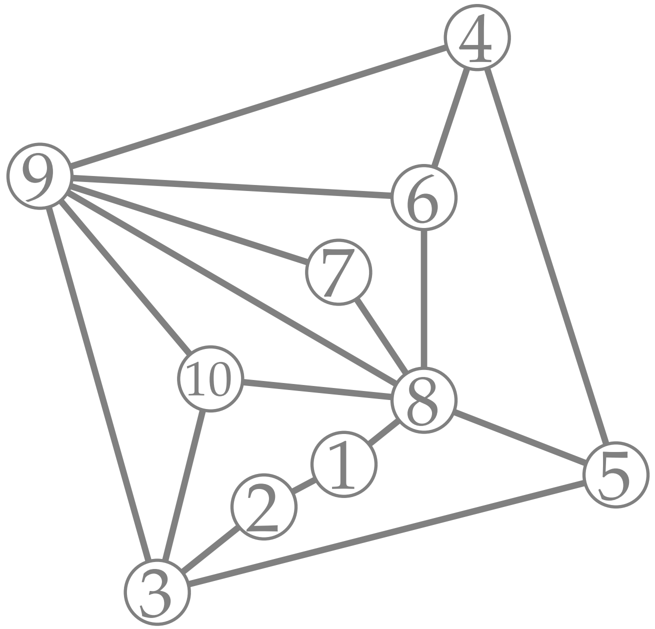
~~connected component~~ **connected component**

Universität Trier

Visualization of Graphs



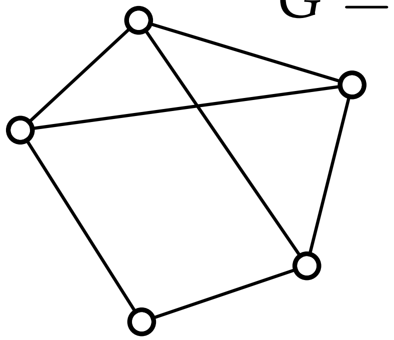
Lecture 1:
The Graph Visualization Problem



Part IV:
Planarity

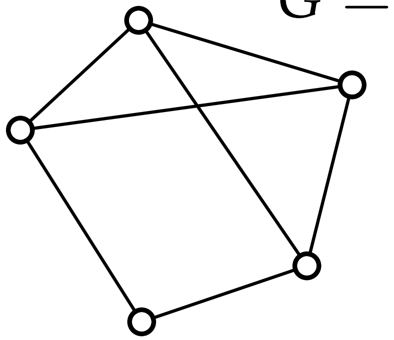
Philipp Kindermann
Summer Semester 2021

Planar Graphs



$G = (V, E)$

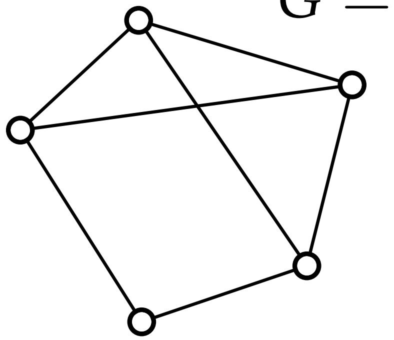
Planar Graphs



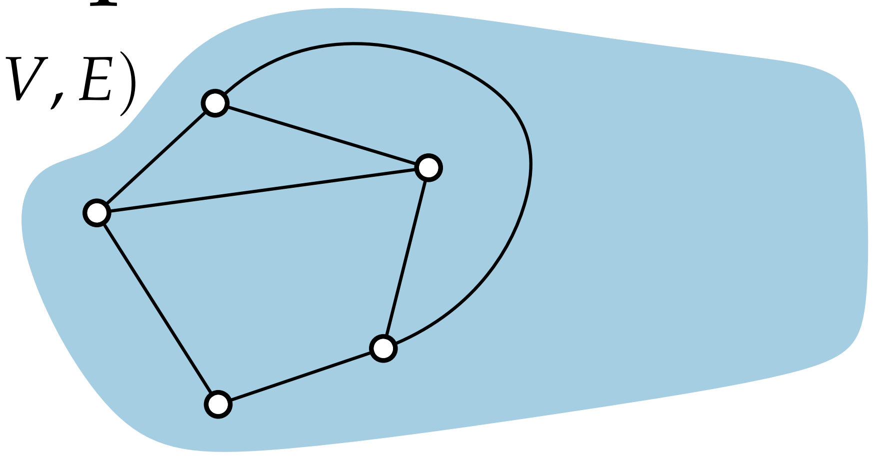
$G = (V, E)$

G is **planar**:
it can be drawn in such a way
that no edges cross each other.

Planar Graphs

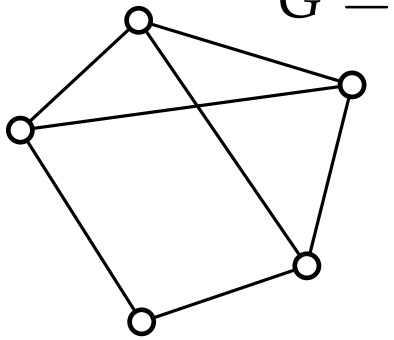


$G = (V, E)$

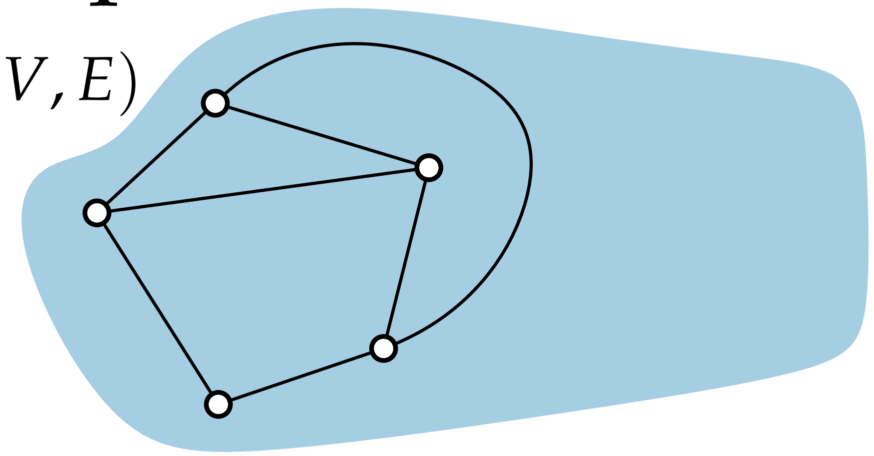


G is **planar**:
it can be drawn in such a way
that no edges cross each other.

Planar Graphs



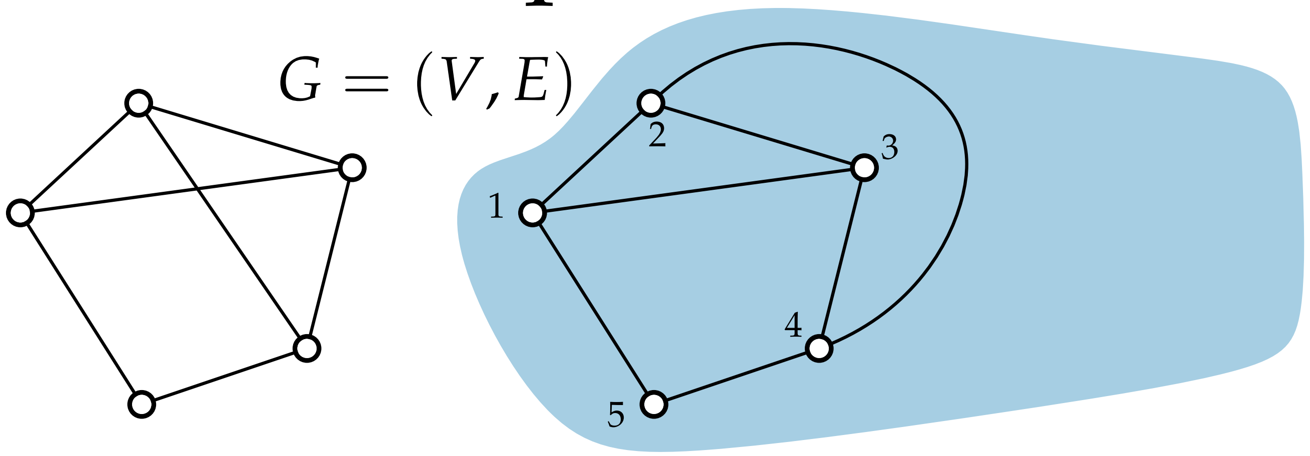
$G = (V, E)$



G is **planar**:
it can be drawn in such a way
that no edges cross each other.

planar embedding:
Clockwise orientation of adjacent
vertices around each vertex.

Planar Graphs

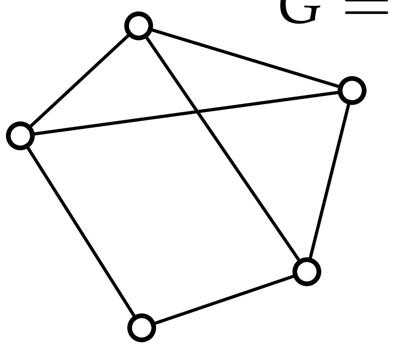


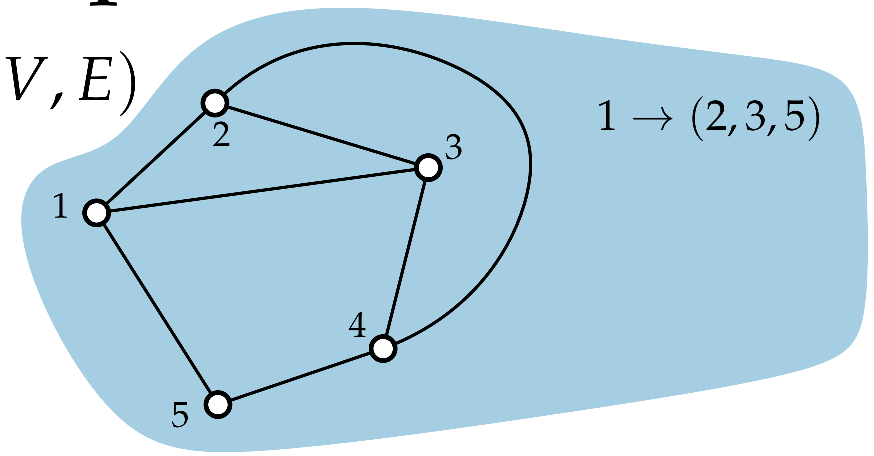
G is **planar**:
it can be drawn in such a way
that no edges cross each other.

planar embedding:
Clockwise orientation of adjacent
vertices around each vertex.

Planar Graphs

$G = (V, E)$

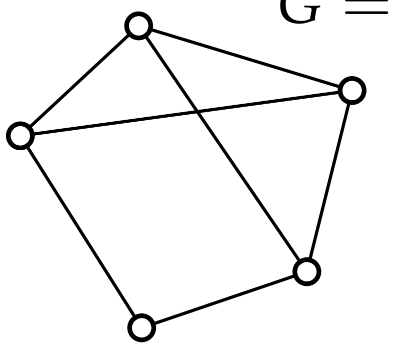




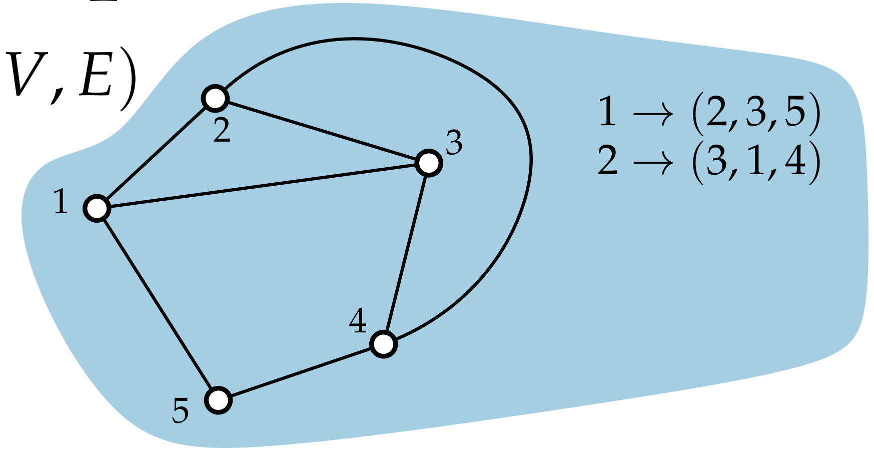
G is **planar**:
it can be drawn in such a way
that no edges cross each other.

planar embedding:
Clockwise orientation of adjacent
vertices around each vertex.

Planar Graphs



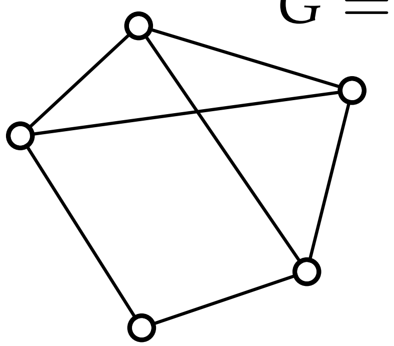
$G = (V, E)$



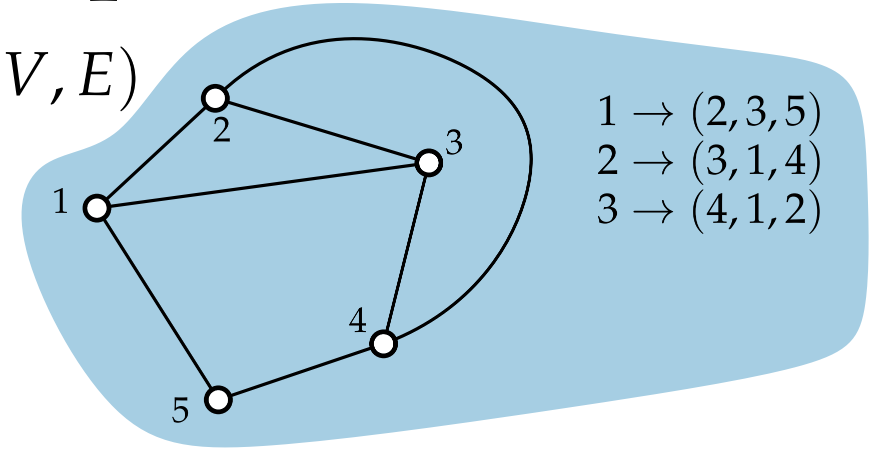
G is **planar**:
it can be drawn in such a way
that no edges cross each other.

planar embedding:
Clockwise orientation of adjacent
vertices around each vertex.

Planar Graphs



$G = (V, E)$

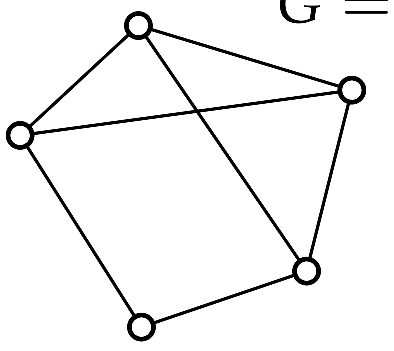


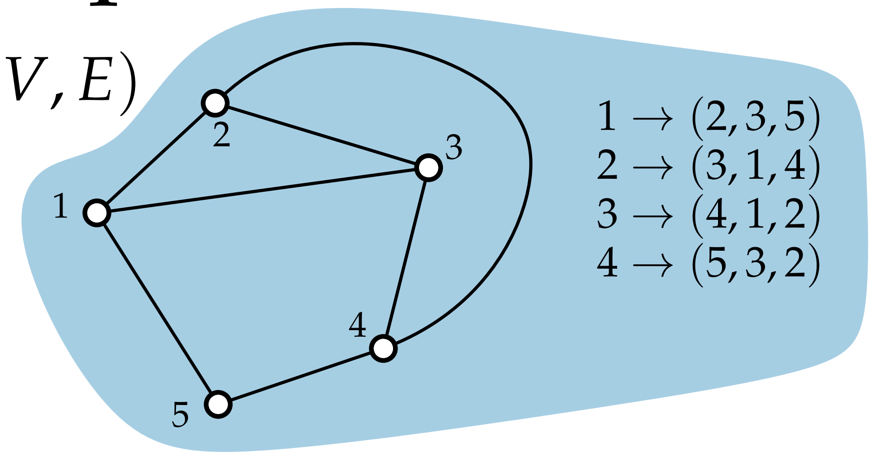
G is **planar**:
it can be drawn in such a way that no edges cross each other.

planar embedding:
Clockwise orientation of adjacent vertices around each vertex.

Planar Graphs

$G = (V, E)$

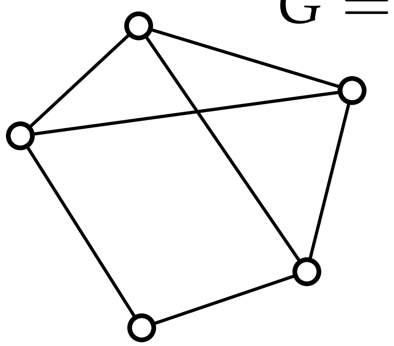




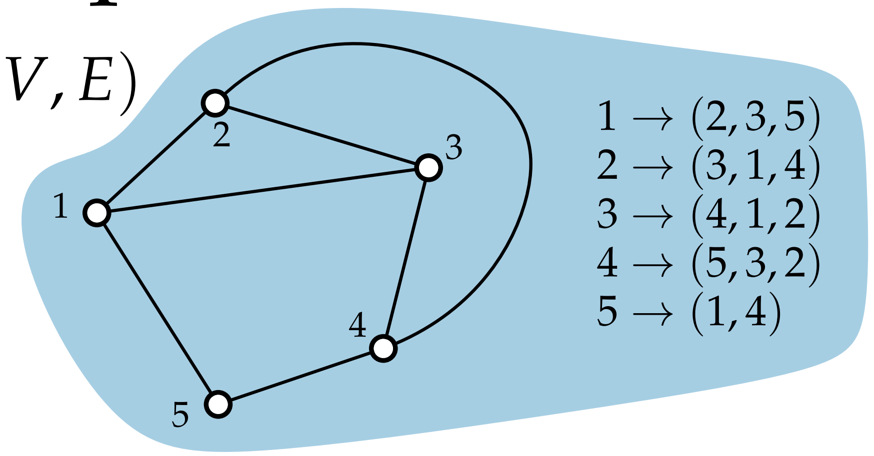
G is **planar**:
it can be drawn in such a way
that no edges cross each other.

planar embedding:
Clockwise orientation of adjacent
vertices around each vertex.

Planar Graphs



$G = (V, E)$

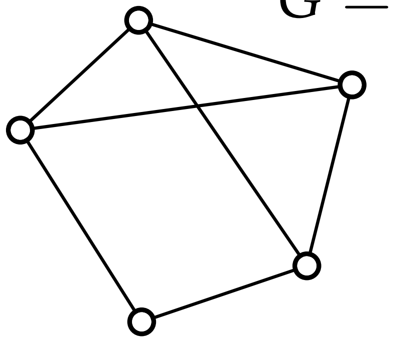


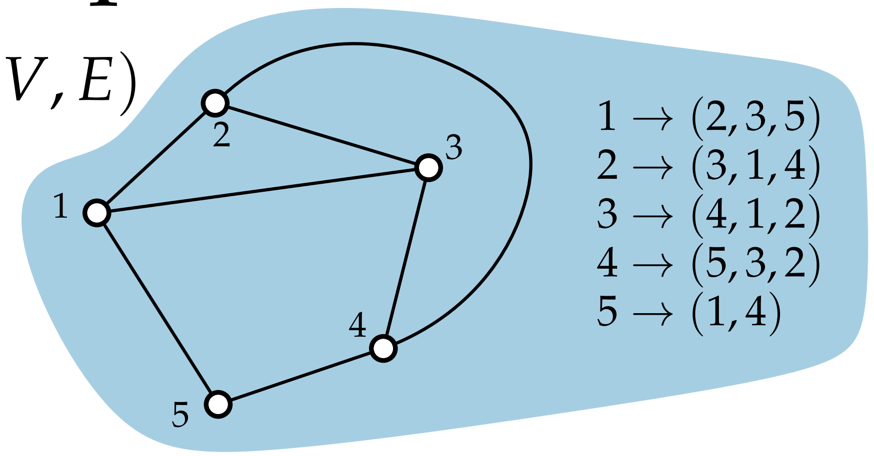
G is **planar**:
it can be drawn in such a way
that no edges cross each other.

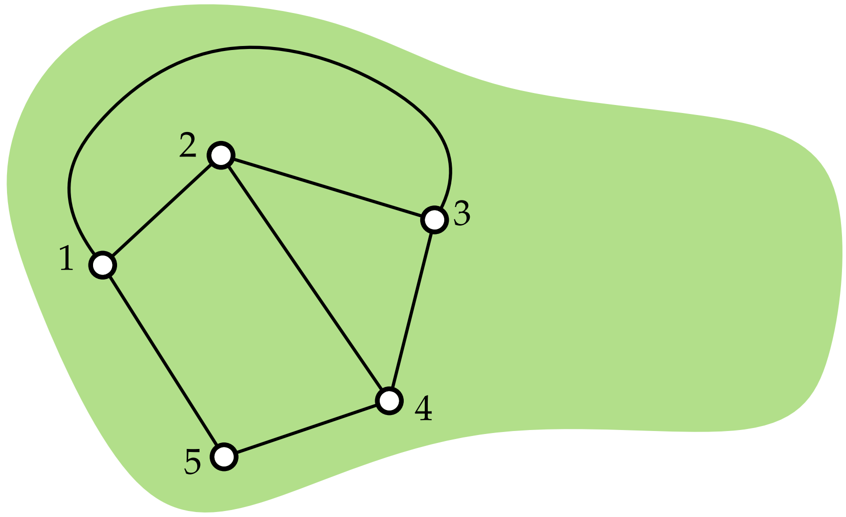
planar embedding:
Clockwise orientation of adjacent
vertices around each vertex.

Planar Graphs

$G = (V, E)$





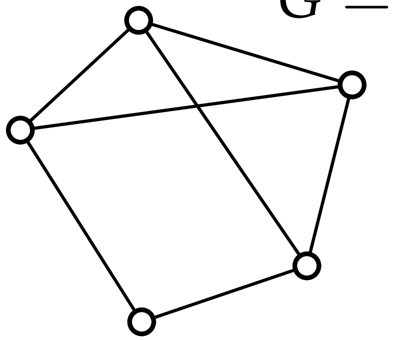


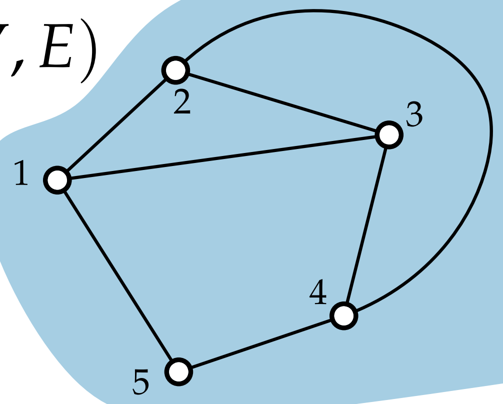
G is **planar**:
it can be drawn in such a way that no edges cross each other.

planar embedding:
Clockwise orientation of adjacent vertices around each vertex.

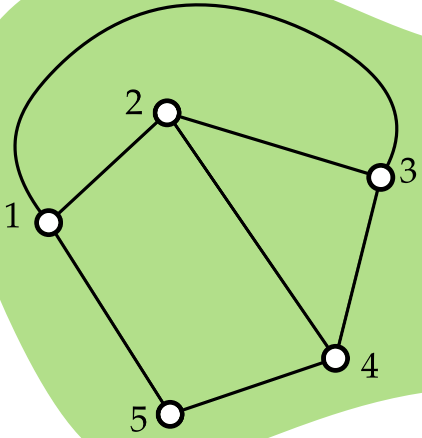
Planar Graphs

$G = (V, E)$





$1 \to (2,3,5)$
$2 \to (3,1,4)$
$3 \to (4,1,2)$
$4 \to (5,3,2)$
$5 \to (1,4)$



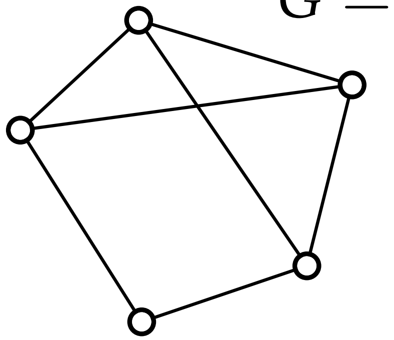
$1 \to (2,5,3)$
$2 \to (3,4,1)$
$3 \to (4,2,1)$
$4 \to (5,2,3)$
$5 \to (1,4)$

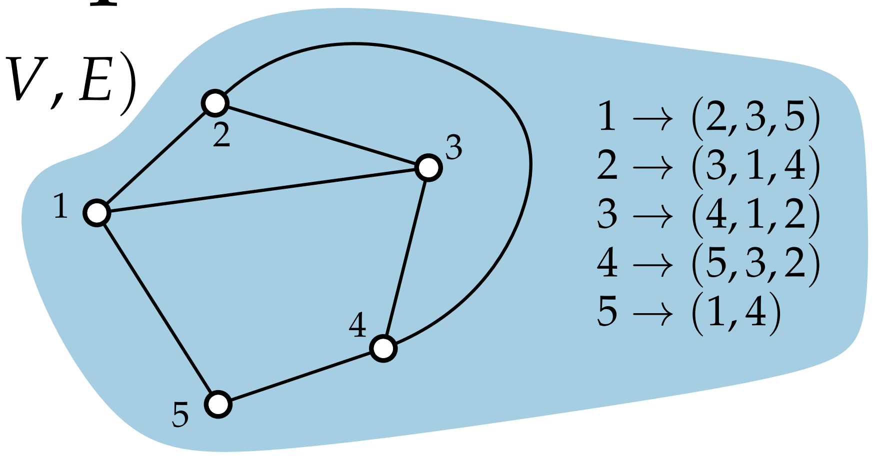
G is **planar**:
it can be drawn in such a way
that no edges cross each other.

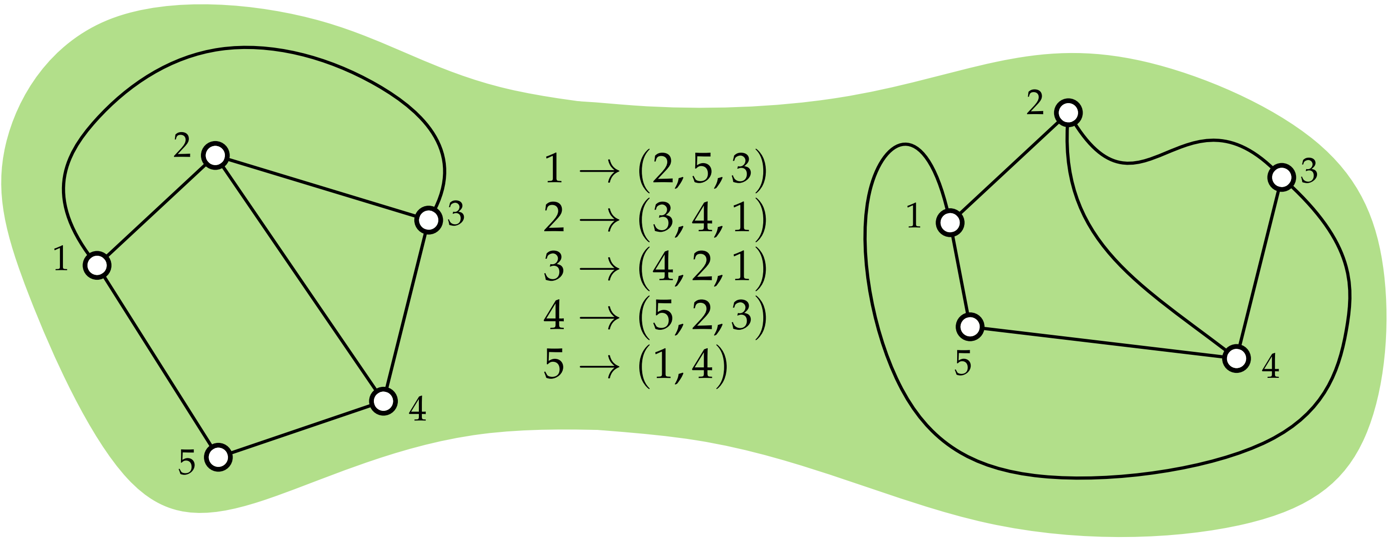
planar embedding:
Clockwise orientation of adjacent
vertices around each vertex.

Planar Graphs

$G = (V, E)$





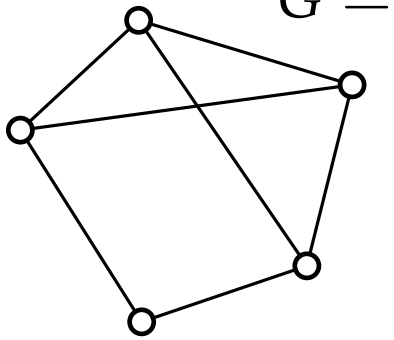


G is **planar**:
it can be drawn in such a way that no edges cross each other.

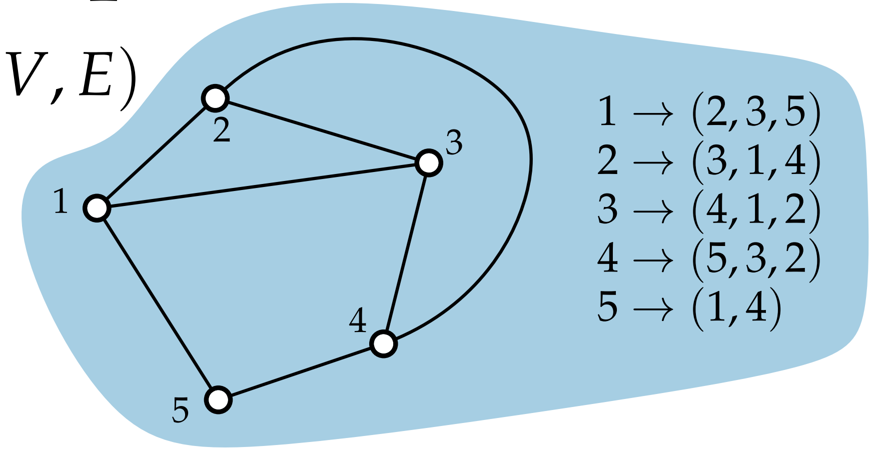
planar embedding:
Clockwise orientation of adjacent vertices around each vertex.

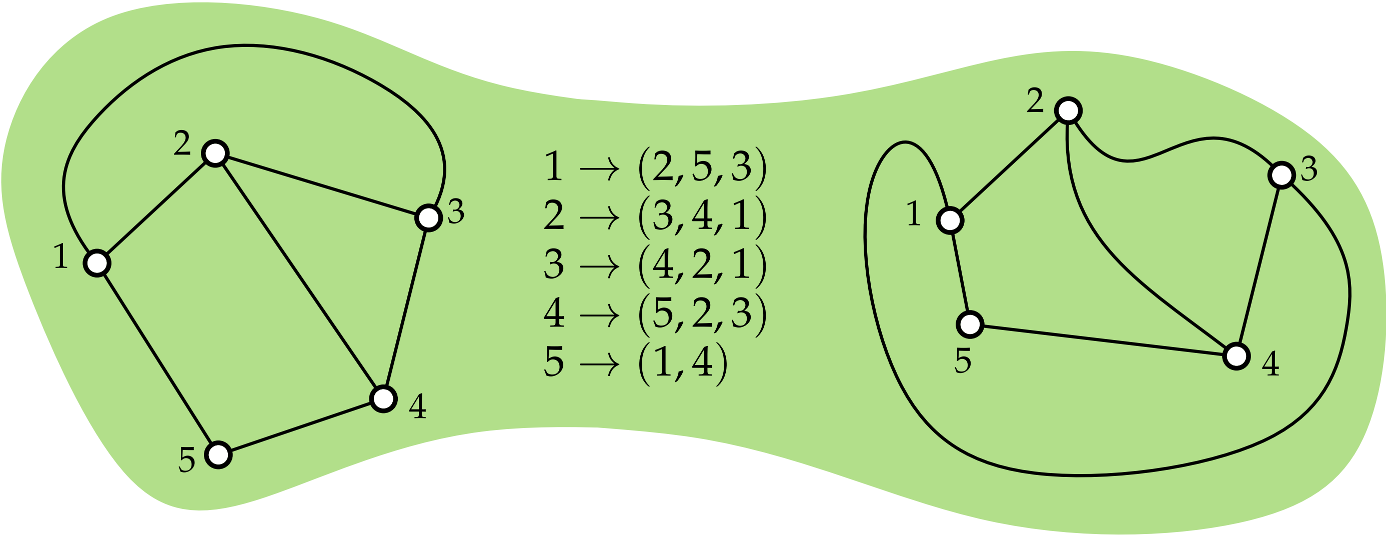
A planar graph can have many planar embeddings.

Planar Graphs



$G = (V, E)$





G is **planar**:
it can be drawn in such a way that no edges cross each other.

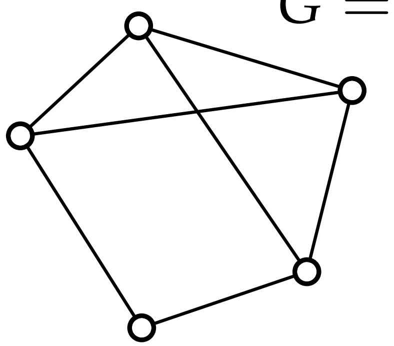
planar embedding:
Clockwise orientation of adjacent vertices around each vertex.

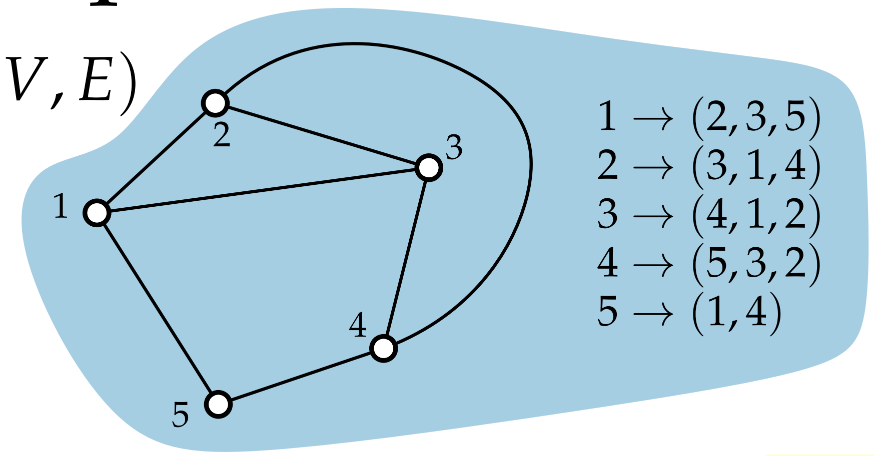
A planar graph can have many planar embeddings.

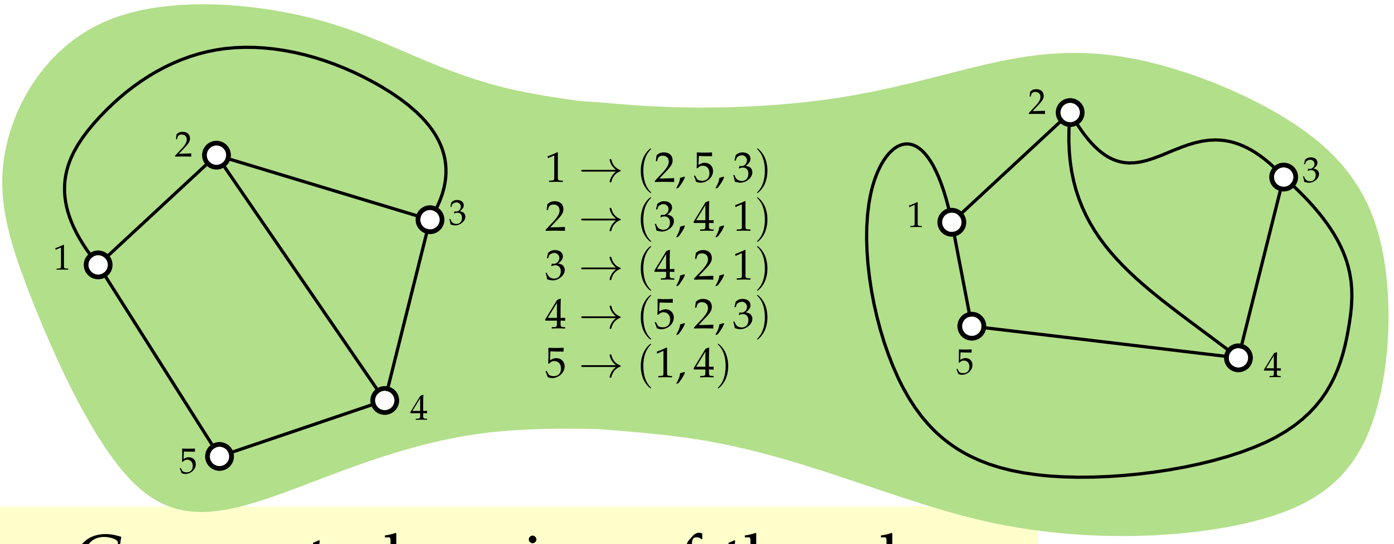
A planar embedding can have many planar drawings!

Planar Graphs

$G = (V, E)$







G is **planar**:
it can be drawn in such a way that no edges cross each other.

planar embedding:
Clockwise orientation of adjacent vertices around each vertex.

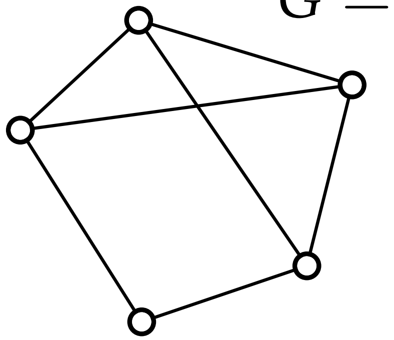
faces: Connected region of the plane bounded by edges

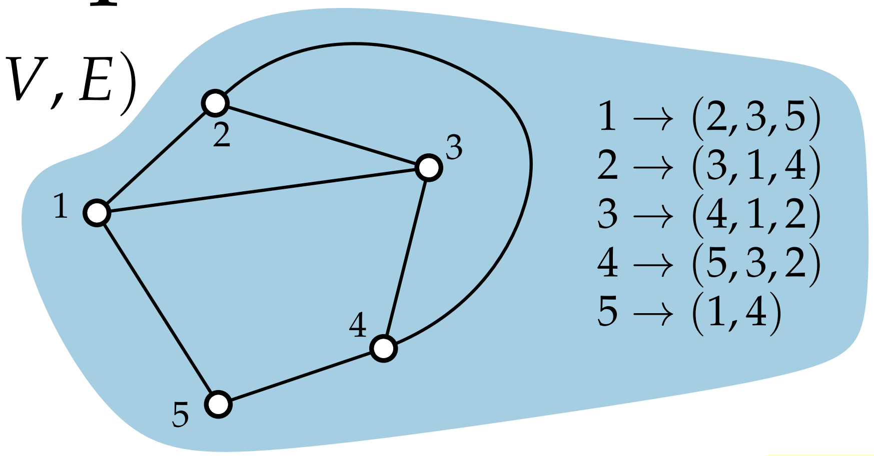
A planar graph can have many planar embeddings.

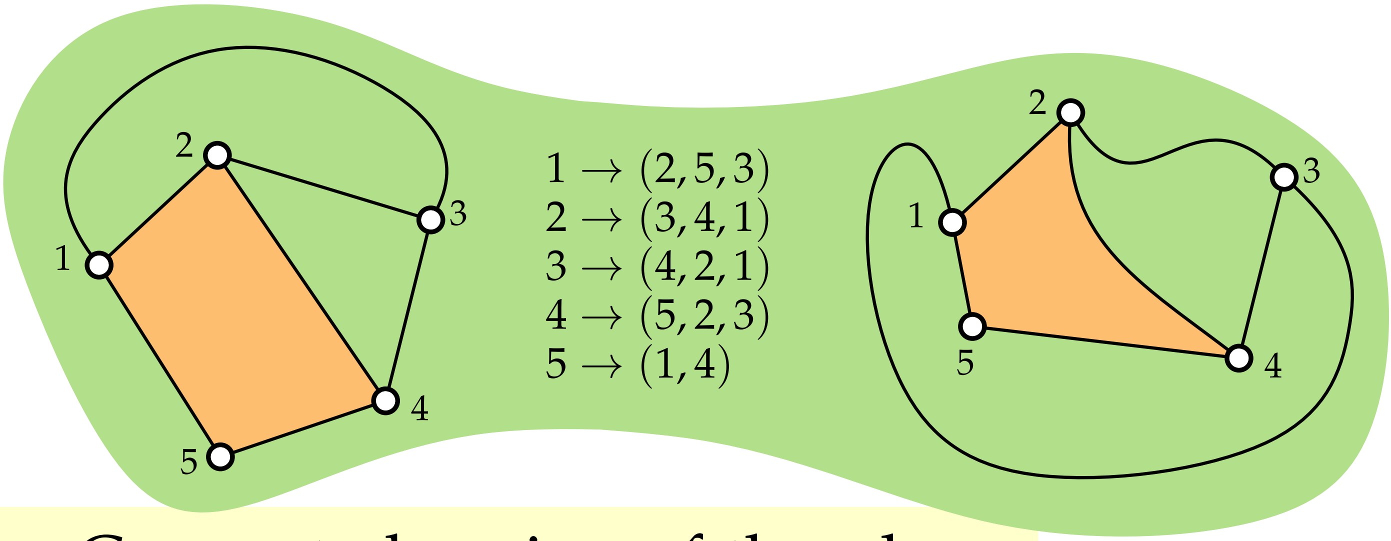
A planar embedding can have many planar drawings!

Planar Graphs

$G = (V, E)$







G is **planar**:
it can be drawn in such a way that no edges cross each other.

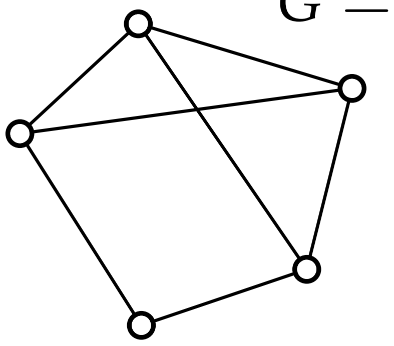
planar embedding:
Clockwise orientation of adjacent vertices around each vertex.

faces: Connected region of the plane bounded by edges

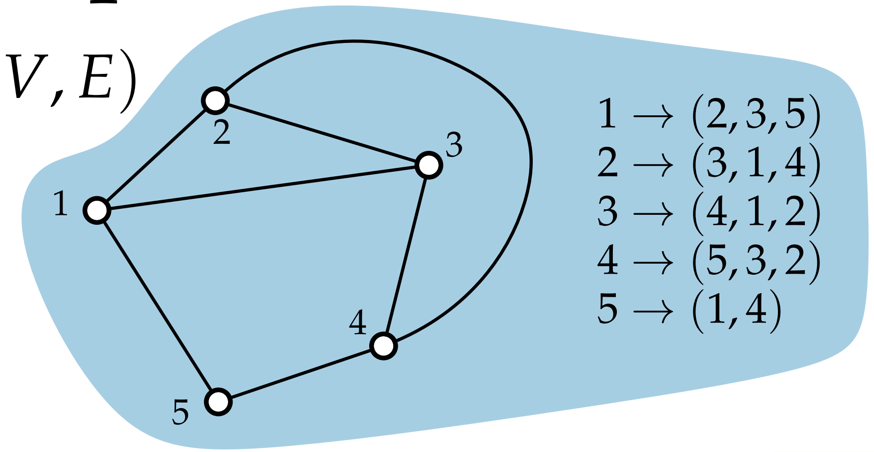
A planar graph can have many planar embeddings.

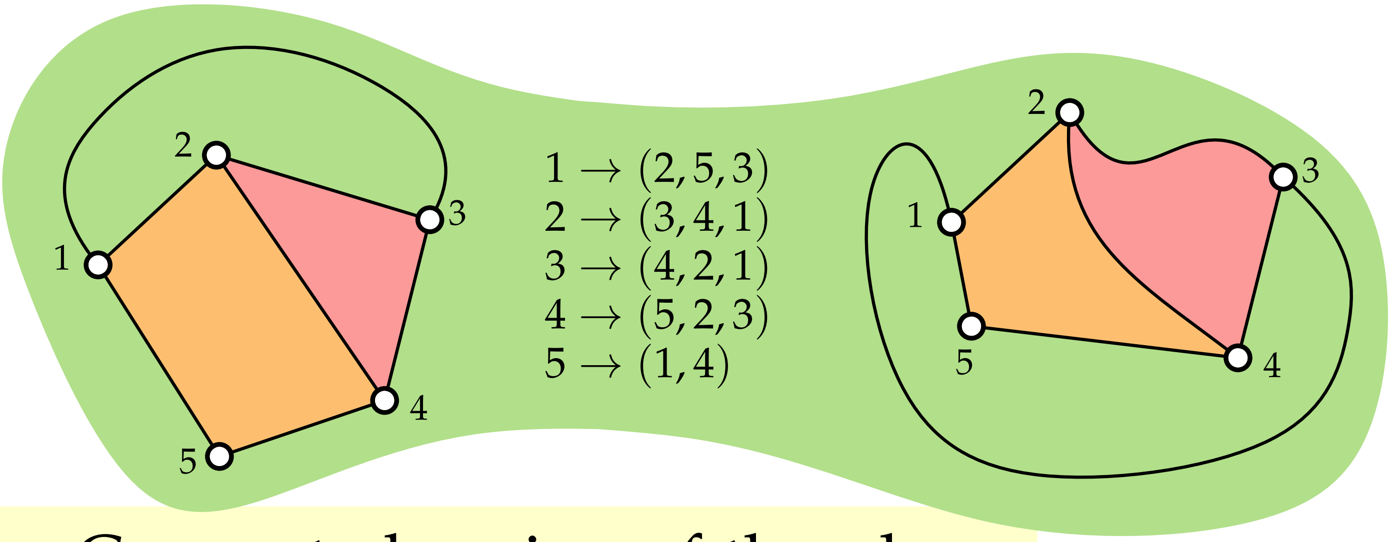
A planar embedding can have many planar drawings!

Planar Graphs



$G = (V, E)$





G is **planar**:
it can be drawn in such a way that no edges cross each other.

planar embedding:
Clockwise orientation of adjacent vertices around each vertex.

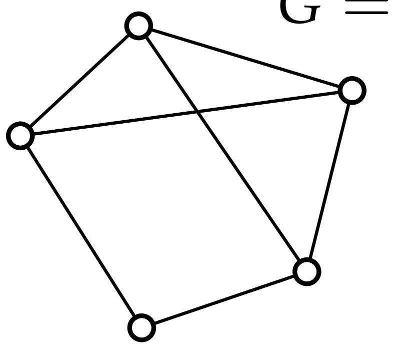
faces: Connected region of the plane bounded by edges

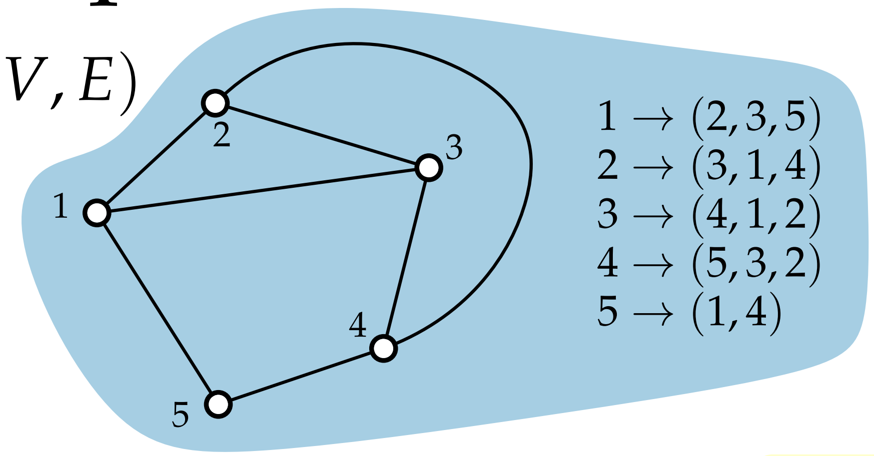
A planar graph can have many planar embeddings.

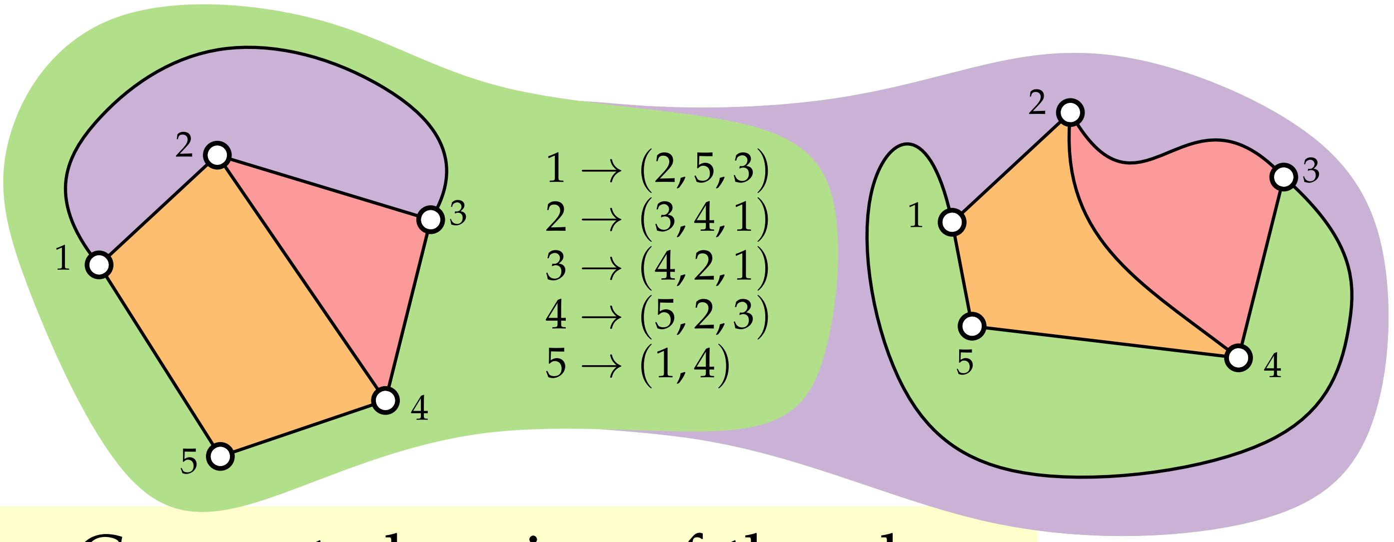
A planar embedding can have many planar drawings!

Planar Graphs

$G = (V, E)$







G is **planar**:
it can be drawn in such a way that no edges cross each other.

planar embedding:
Clockwise orientation of adjacent vertices around each vertex.

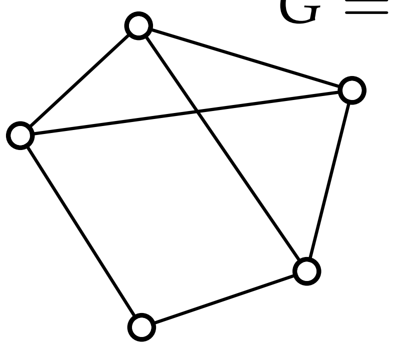
faces: Connected region of the plane bounded by edges

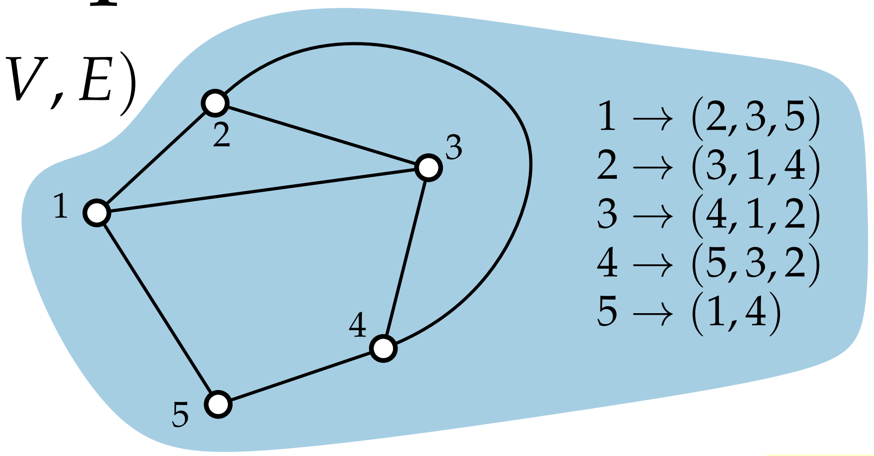
A planar graph can have many planar embeddings.

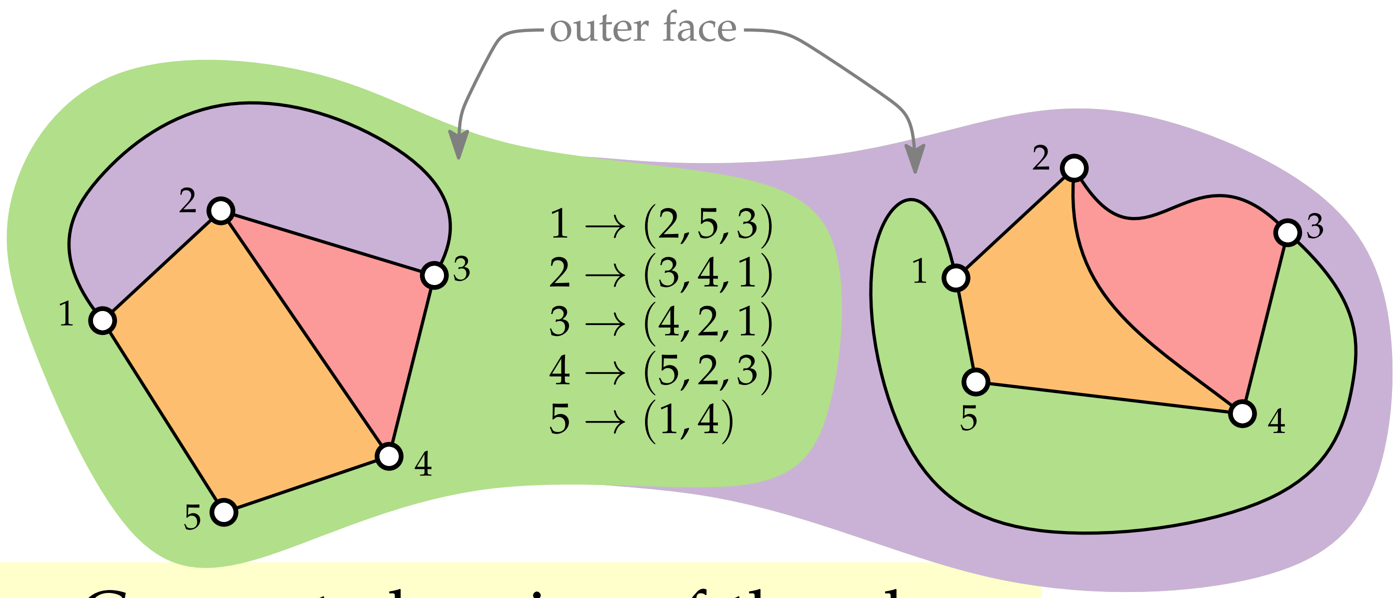
A planar embedding can have many planar drawings!

Planar Graphs

$G = (V, E)$







G is **planar**:
it can be drawn in such a way that no edges cross each other.

planar embedding:
Clockwise orientation of adjacent vertices around each vertex.

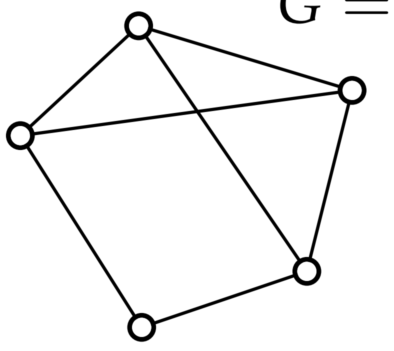
faces: Connected region of the plane bounded by edges

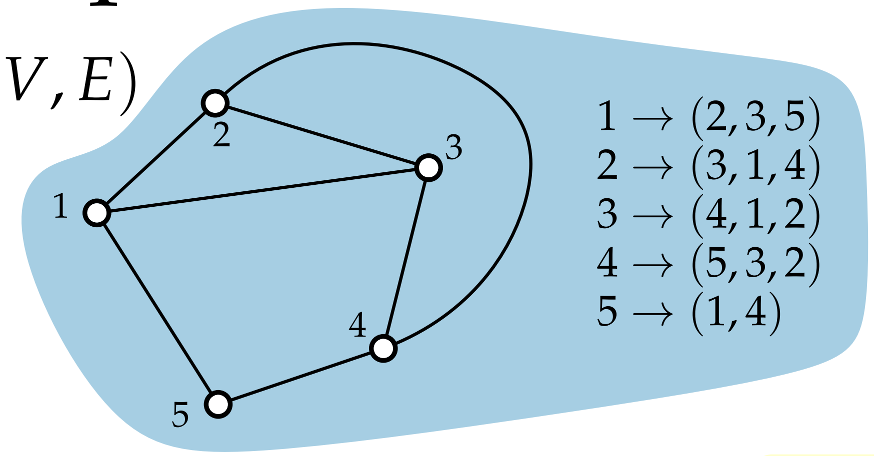
A planar graph can have many planar embeddings.

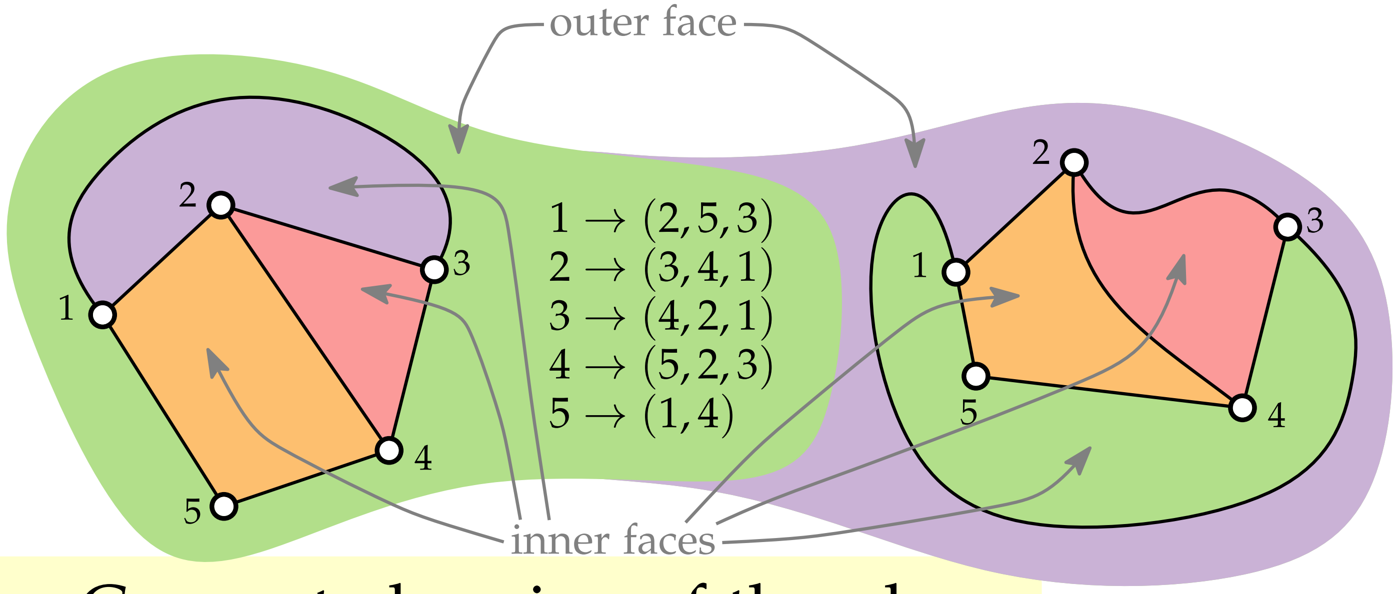
A planar embedding can have many planar drawings!

Planar Graphs

$G = (V, E)$







G is **planar**:
it can be drawn in such a way that no edges cross each other.

planar embedding:
Clockwise orientation of adjacent vertices around each vertex.

A planar graph can have many planar embeddings.

A planar embedding can have many planar drawings!

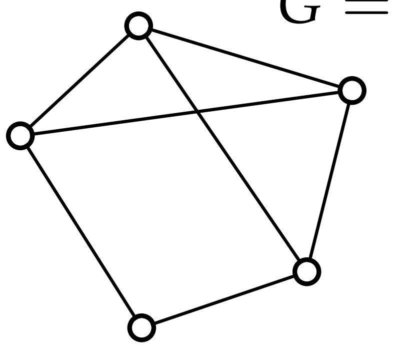
faces: Connected region of the plane bounded by edges

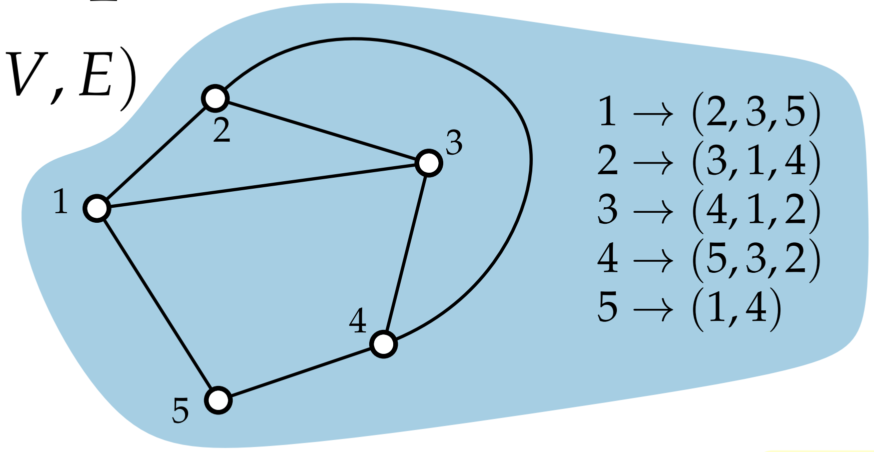
Euler's polyhedra formula.

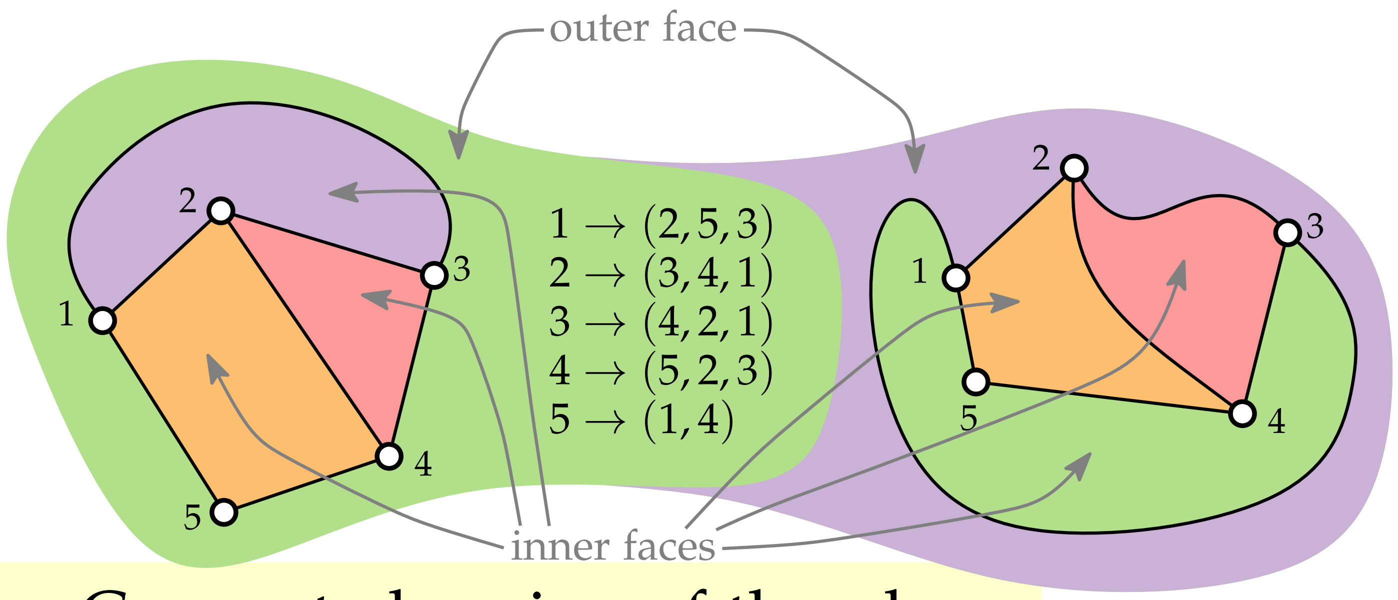
#faces - #edges + #vertices = #conn.comp. + 1

$$f - m + n = c + 1$$

Planar Graphs







G is **planar**:
it can be drawn in such a way that no edges cross each other.

planar embedding:
Clockwise orientation of adjacent vertices around each vertex.

A planar graph can have many planar embeddings.

A planar embedding can have many planar drawings!

faces: Connected region of the plane bounded by edges

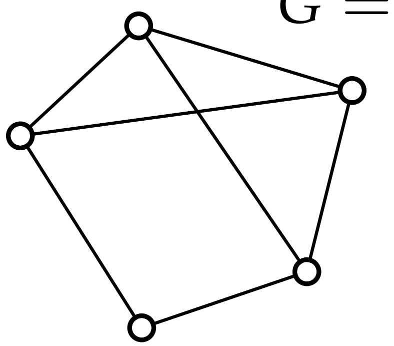
Euler's polyhedra formula.

#faces - #edges + #vertices = #conn.comp. + 1

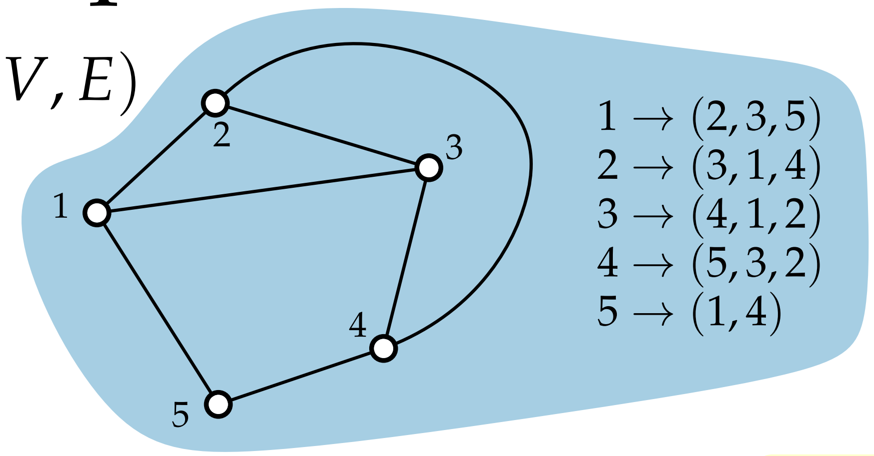
$$f - m + n = c + 1$$

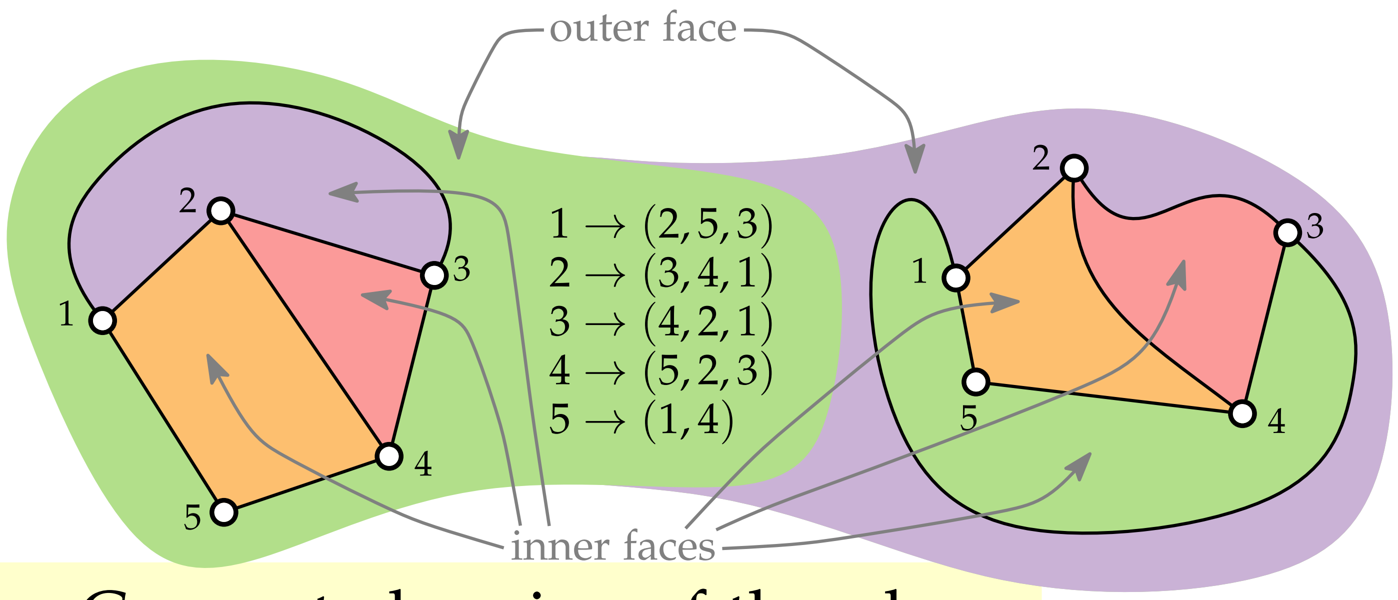
Proof.

Planar Graphs



$G = (V, E)$





G is **planar**:
it can be drawn in such a way that no edges cross each other.

planar embedding:
Clockwise orientation of adjacent vertices around each vertex.

A planar graph can have many planar embeddings.

A planar embedding can have many planar drawings!

faces: Connected region of the plane bounded by edges

Euler's polyhedra formula.

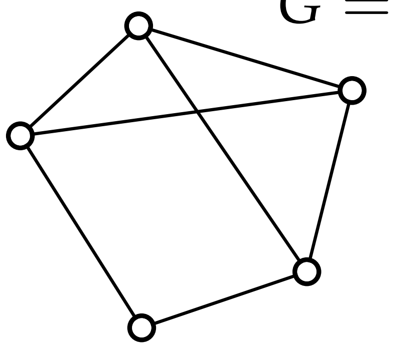
#faces - #edges + #vertices = #conn.comp. + 1

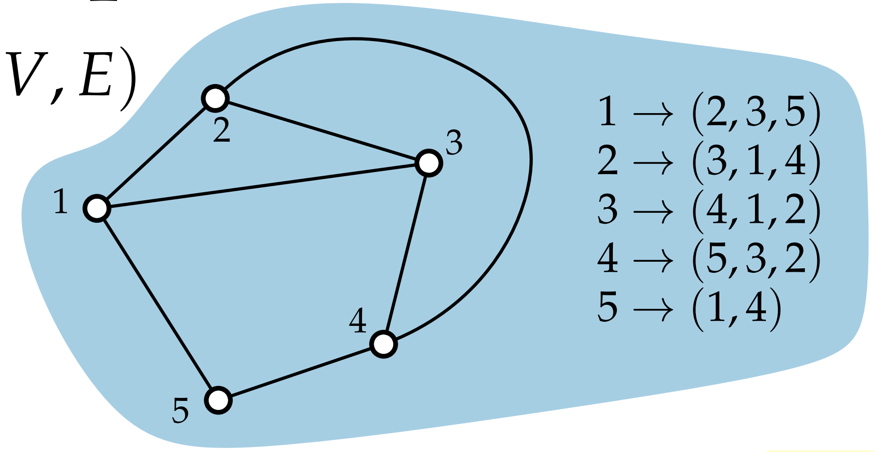
$$f - m + n = c + 1$$

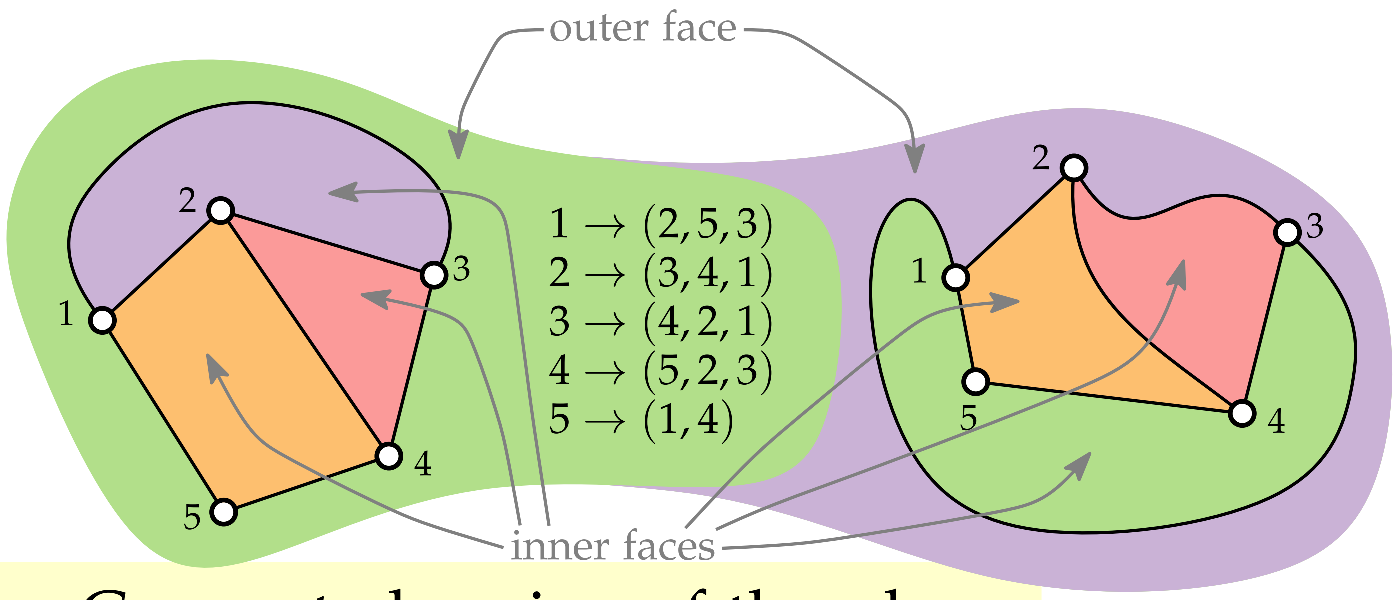
Proof. By induction on m:

Planar Graphs

$G = (V, E)$







G is **planar**:
it can be drawn in such a way that no edges cross each other.

planar embedding:
Clockwise orientation of adjacent vertices around each vertex.

A planar graph can have many planar embeddings.

A planar embedding can have many planar drawings!

faces: Connected region of the plane bounded by edges

Euler's polyhedra formula.

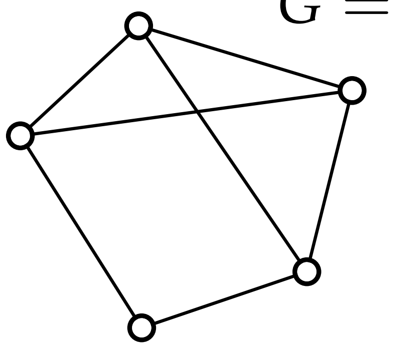
#faces - #edges + #vertices = #conn.comp. + 1

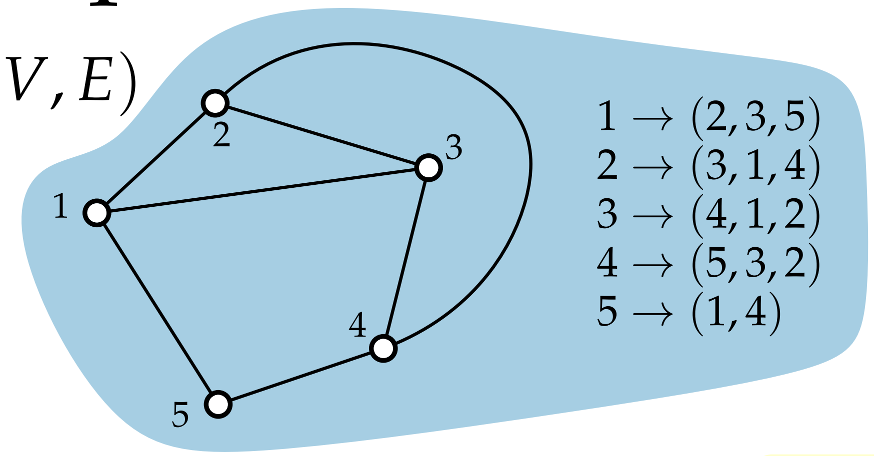
$f - m + n = c + 1$

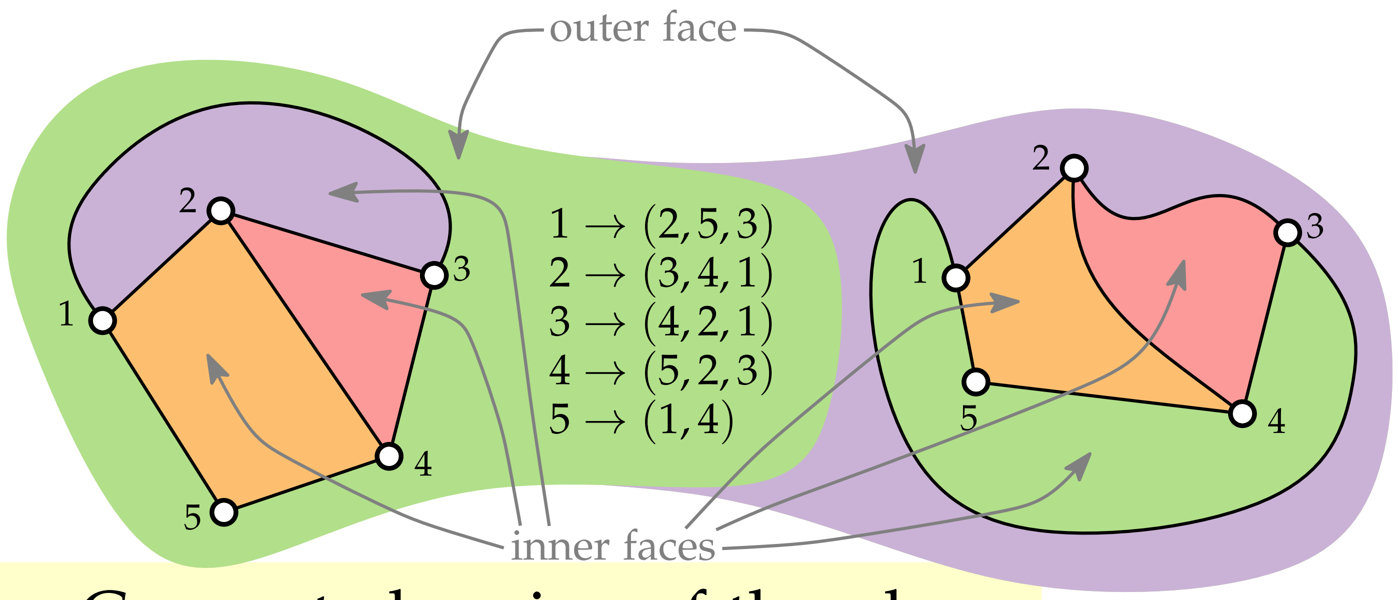
Proof. By induction on m:

$m = 0 \Rightarrow$

Planar Graphs







G is **planar**:
it can be drawn in such a way that no edges cross each other.

planar embedding:
Clockwise orientation of adjacent vertices around each vertex.

A planar graph can have many planar embeddings.

A planar embedding can have many planar drawings!

faces: Connected region of the plane bounded by edges

Euler's polyhedra formula.

#faces - #edges + #vertices = #conn.comp. + 1

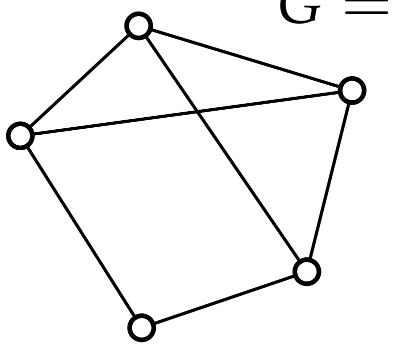
$$f - m + n = c + 1$$

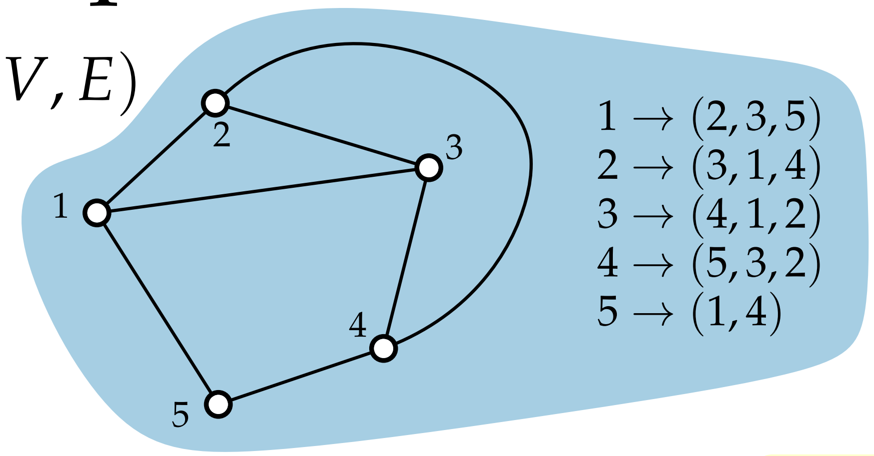
Proof. By induction on m:

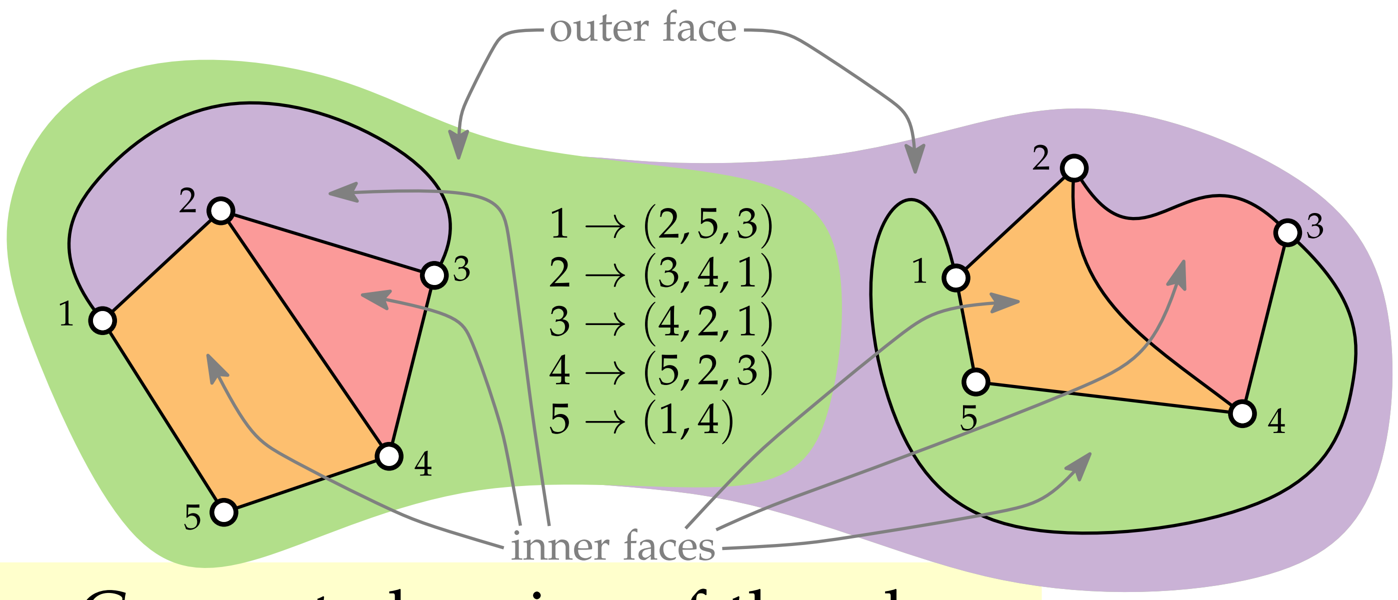
$m = 0 \Rightarrow f =$? and $c =$?

Planar Graphs

$G = (V, E)$







G is **planar**:
it can be drawn in such a way that no edges cross each other.

planar embedding:
Clockwise orientation of adjacent vertices around each vertex.

A planar graph can have many planar embeddings.

A planar embedding can have many planar drawings!

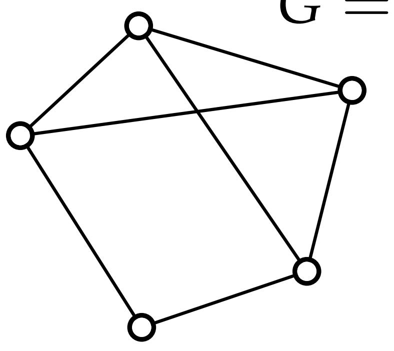
faces: Connected region of the plane bounded by edges

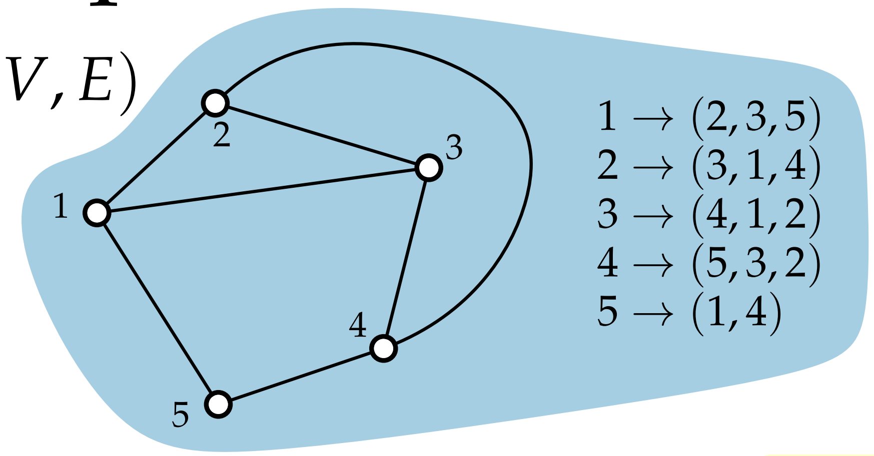
Euler's polyhedra formula.
#faces - #edges + #vertices = #conn.comp. + 1
$f - m + n = c + 1$

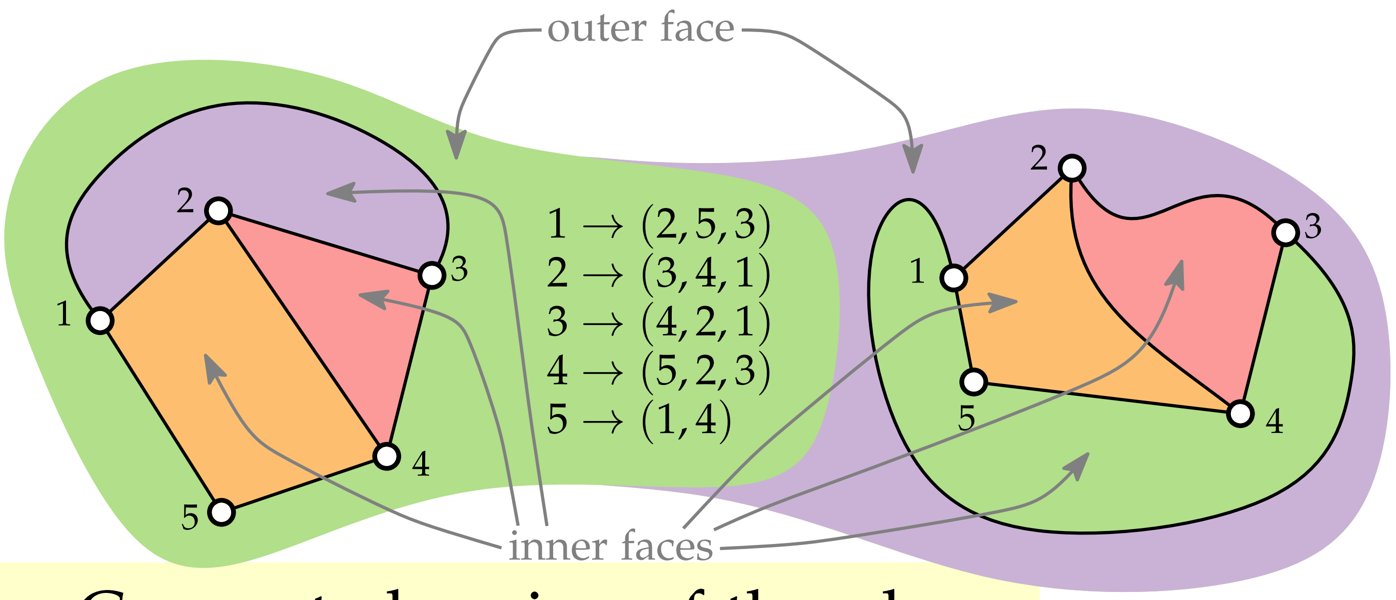
Proof. By induction on m:
$m = 0 \Rightarrow f = 1$ and $c = n$

Planar Graphs

$G = (V, E)$







G is **planar**:
it can be drawn in such a way that no edges cross each other.

planar embedding:
Clockwise orientation of adjacent vertices around each vertex.

A planar graph can have many planar embeddings.

A planar embedding can have many planar drawings!

faces: Connected region of the plane bounded by edges

Euler's polyhedra formula.

#faces - #edges + #vertices = #conn.comp. + 1

$f \quad - \quad m \quad + \quad n \quad = \quad c \quad + 1$

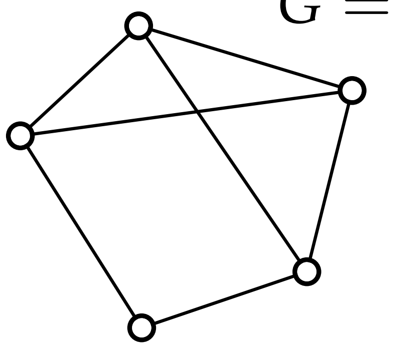
Proof. By induction on m:

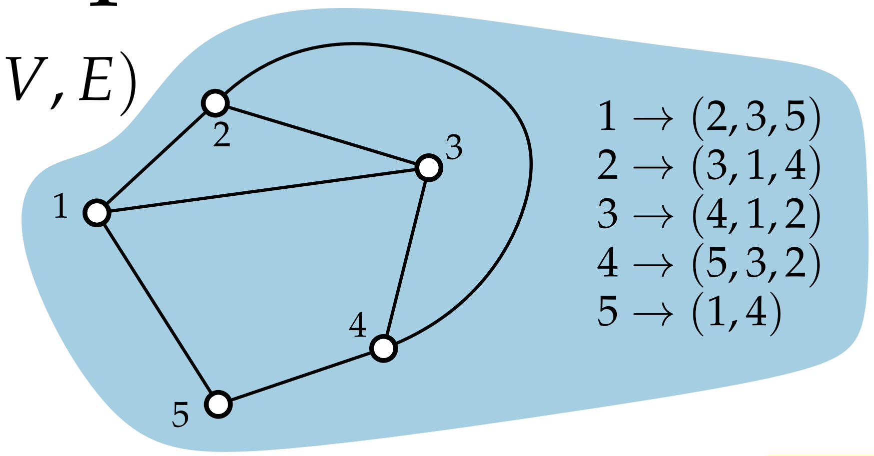
$m = 0 \Rightarrow f = 1$ and $c = n$

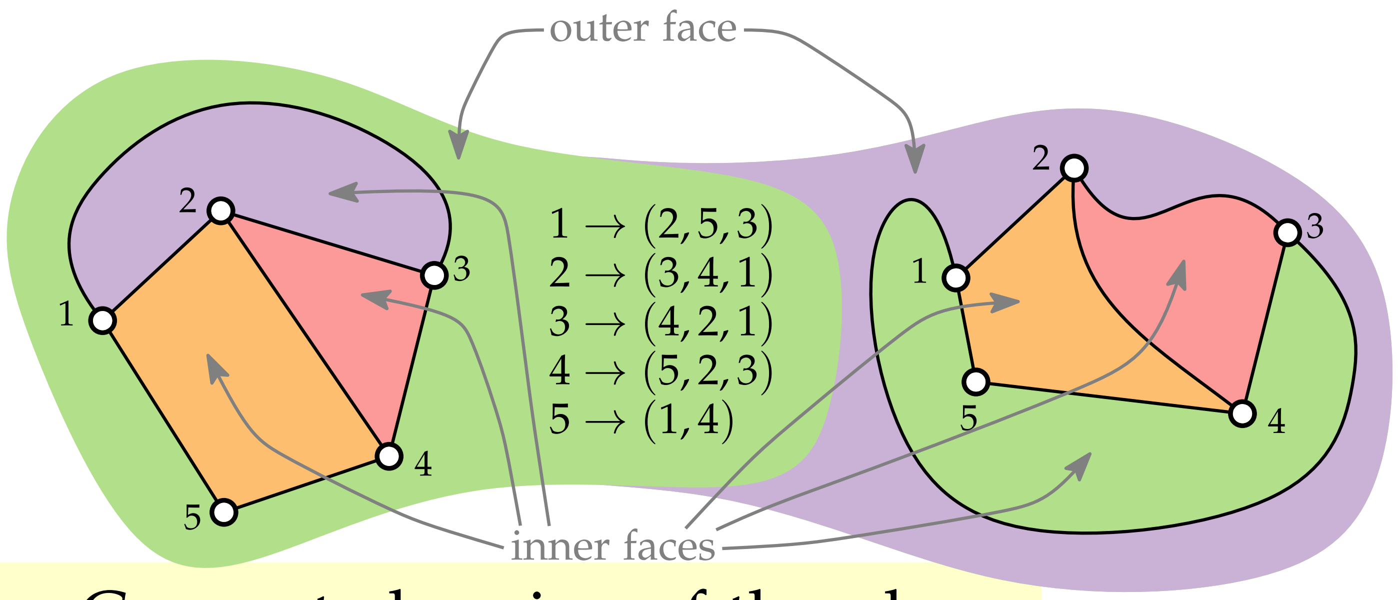
$\Rightarrow 0 - 0 + c = c + 1$

Planar Graphs

$G = (V, E)$







G is **planar**:
it can be drawn in such a way that no edges cross each other.

planar embedding:
Clockwise orientation of adjacent vertices around each vertex.

A planar graph can have many planar embeddings.

A planar embedding can have many planar drawings!

faces: Connected region of the plane bounded by edges

Euler's polyhedra formula.

#faces - #edges + #vertices = #conn.comp. + 1

$f \quad - \quad m \quad + \quad n \quad = \quad c \quad + 1$

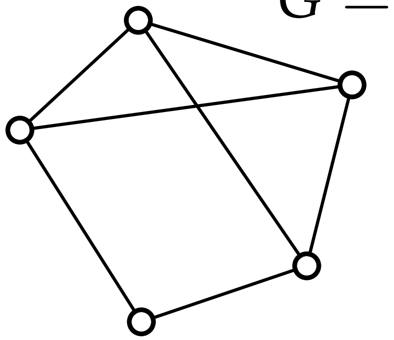
Proof. By induction on m:

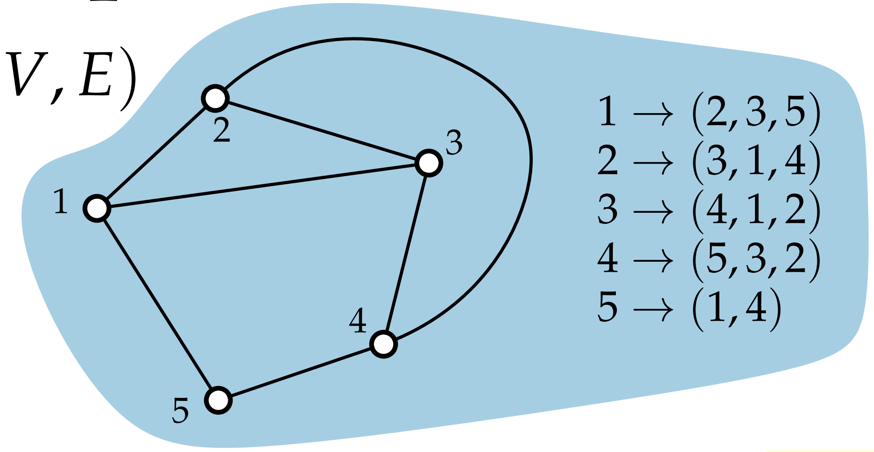
$m = 0 \Rightarrow f = 1$ and $c = n$

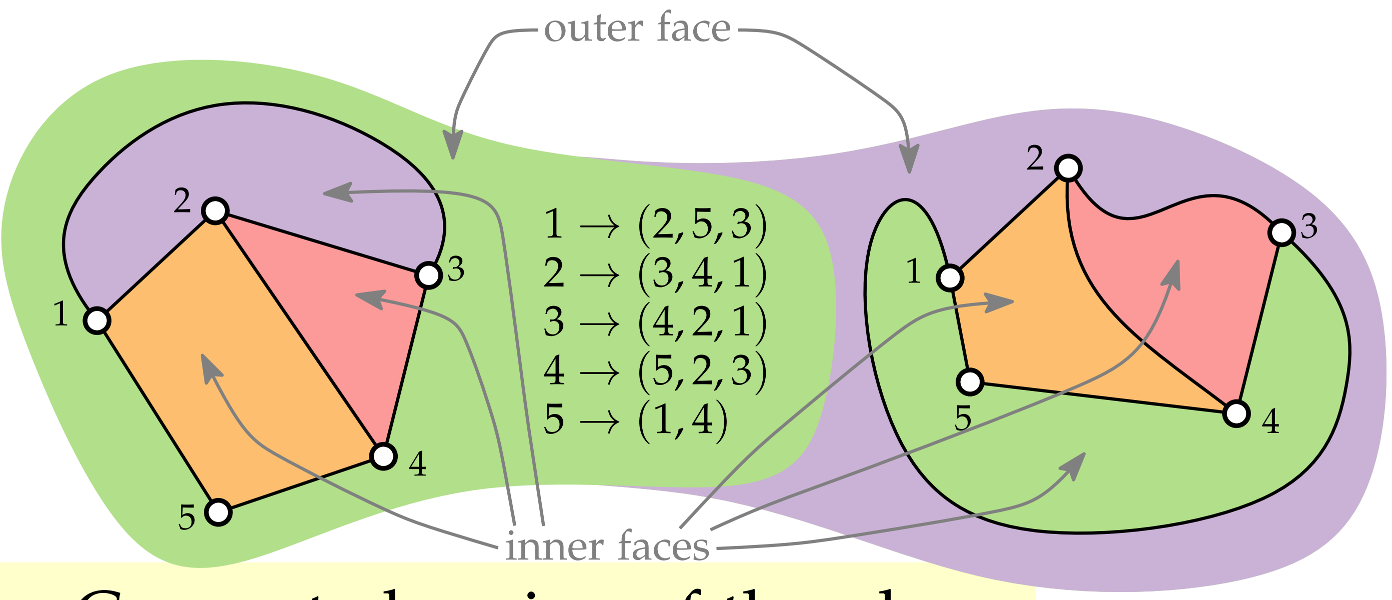
$\Rightarrow 0 - 0 + c = c + 1$ ✓

Planar Graphs

$G = (V, E)$







G is **planar**:
it can be drawn in such a way that no edges cross each other.

planar embedding:
Clockwise orientation of adjacent vertices around each vertex.

A planar graph can have many planar embeddings.

A planar embedding can have many planar drawings!

faces: Connected region of the plane bounded by edges

Euler's polyhedra formula.

#faces - #edges + #vertices = #conn.comp. + 1

$f \quad - \quad m \quad + \quad n \quad = \quad c \quad + 1$

Proof. By induction on m:

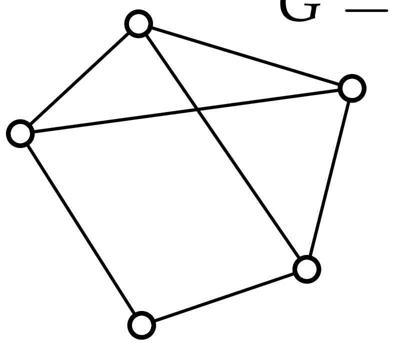
$m = 0 \Rightarrow f = 1$ and $c = n$

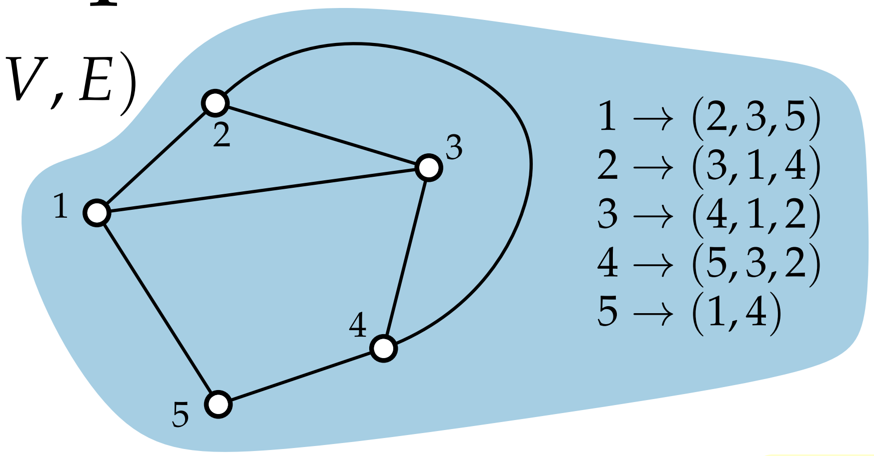
$\Rightarrow 0 - 0 + c = c + 1$ ✓

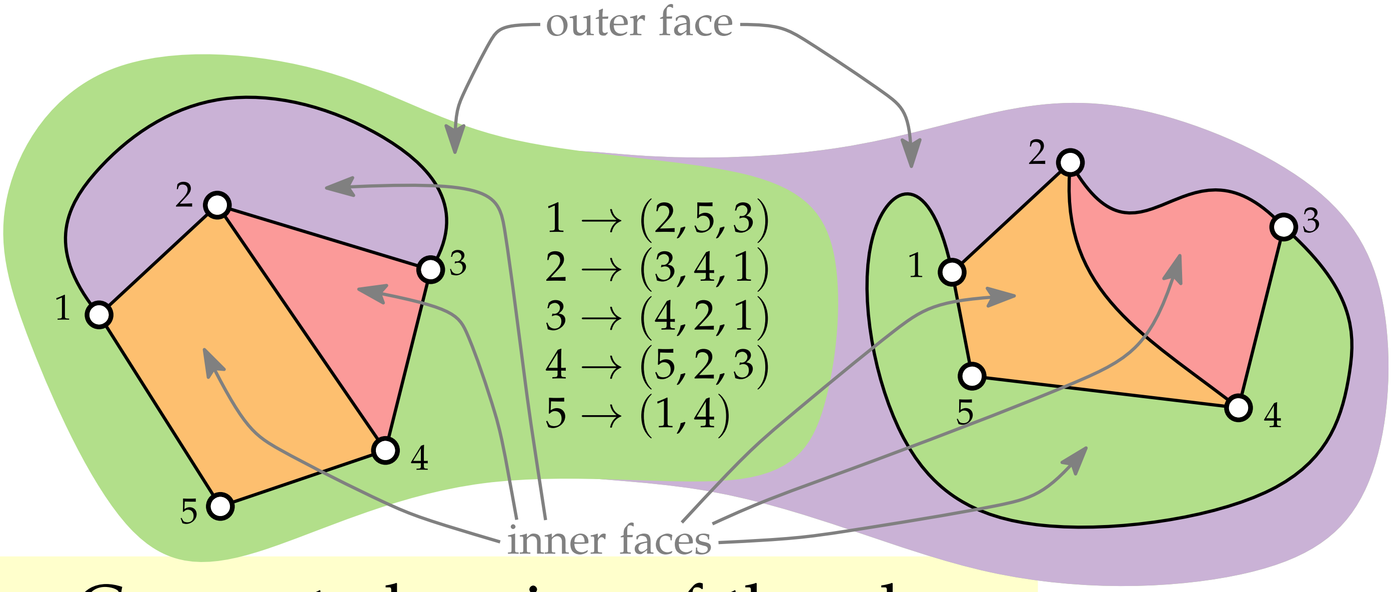
$m > 1 \Rightarrow$

Planar Graphs

$G = (V, E)$







G is **planar**:
it can be drawn in such a way that no edges cross each other.

planar embedding:
Clockwise orientation of adjacent vertices around each vertex.

A planar graph can have many planar embeddings.

A planar embedding can have many planar drawings!

faces: Connected region of the plane bounded by edges

Euler's polyhedra formula.
#faces - #edges + #vertices = #conn.comp. + 1
$f - m + n = c + 1$

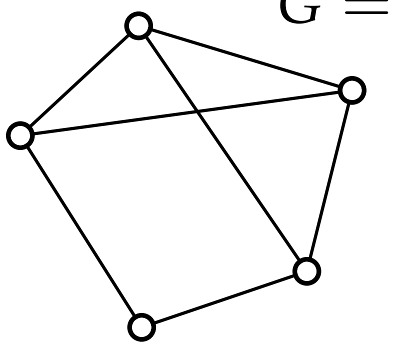
Proof. By induction on m:

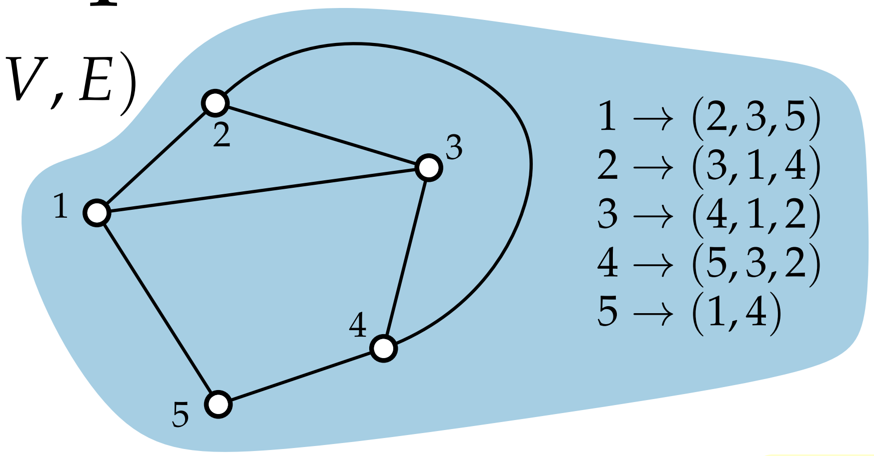
$m = 0 \Rightarrow f = 1$ and $c = n$
$\Rightarrow 0 - 0 + c = c + 1$ ✓

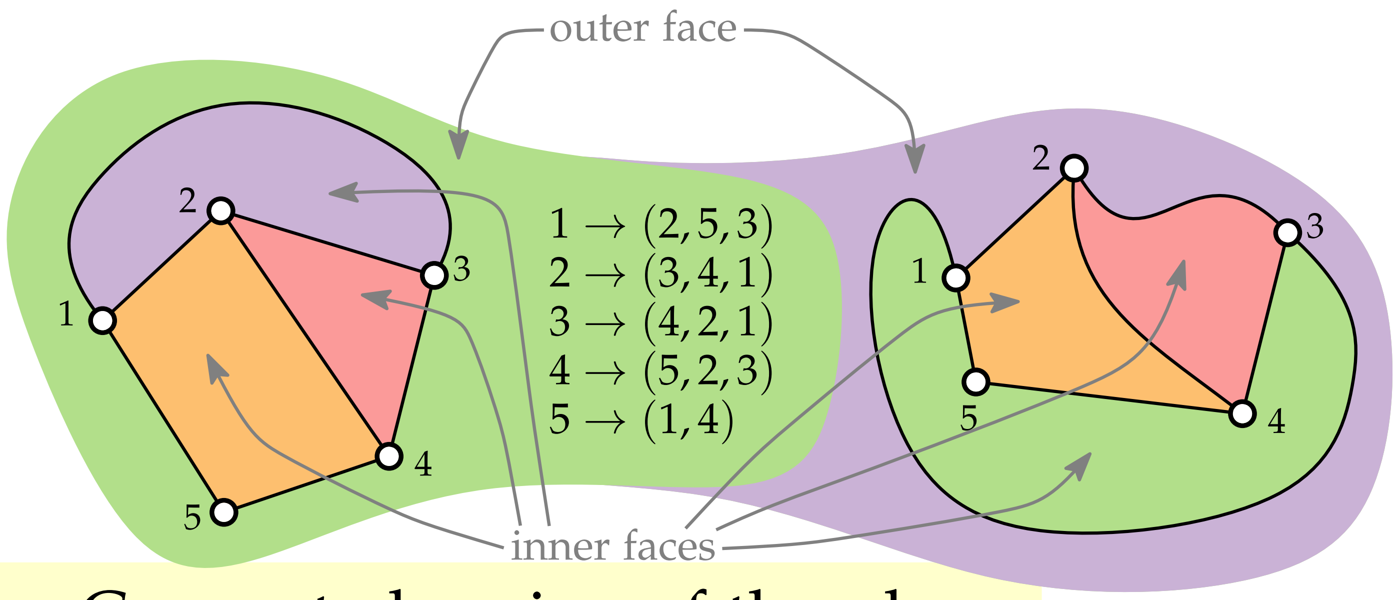
$m > 1 \Rightarrow$ remove 1 edge e

Planar Graphs

$G = (V, E)$







G is **planar**:
it can be drawn in such a way that no edges cross each other.

planar embedding:
Clockwise orientation of adjacent vertices around each vertex.

A planar graph can have many planar embeddings.

A planar embedding can have many planar drawings!

faces: Connected region of the plane bounded by edges

Euler's polyhedra formula.

#faces - #edges + #vertices = #conn.comp. + 1

$f \quad - \quad m \quad + \quad n \quad = \quad c \quad + 1$

Proof. By induction on m:

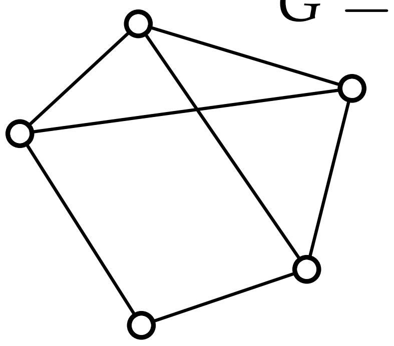
$m = 0 \Rightarrow f = 1$ and $c = n$

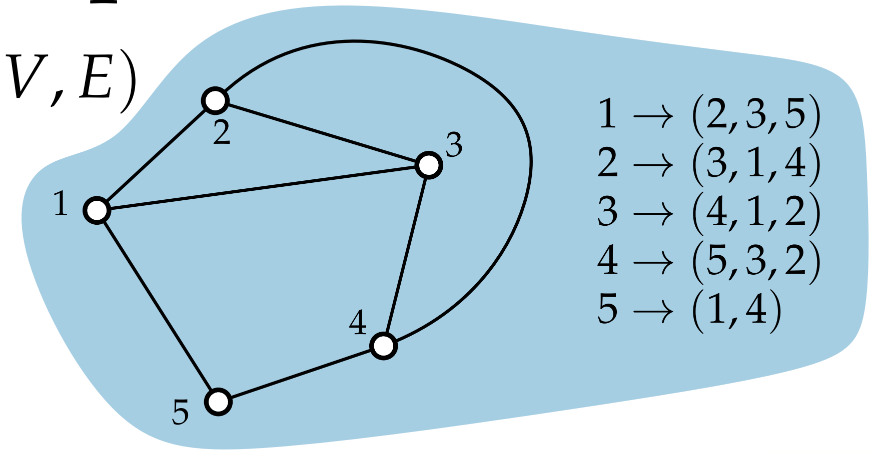
$\Rightarrow 0 - 0 + c = c + 1$ ✓

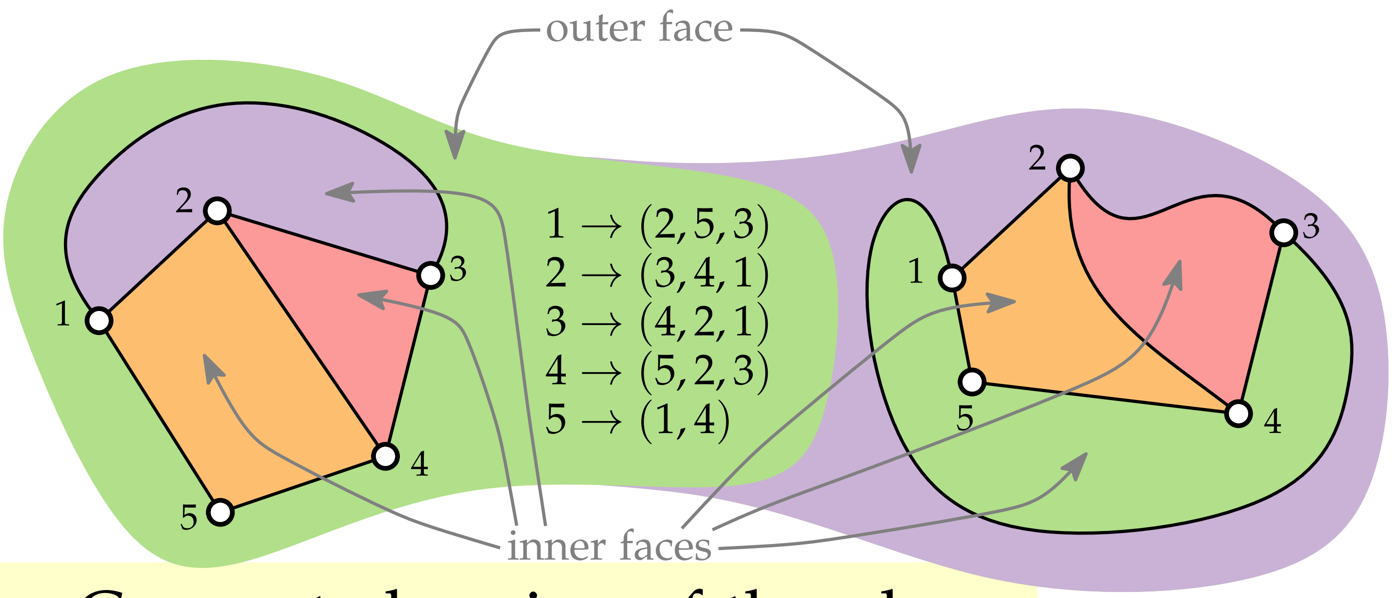
$m > 1 \Rightarrow$ remove 1 edge $e \Rightarrow m - 1$

Planar Graphs

$G = (V, E)$







G is **planar**:
it can be drawn in such a way that no edges cross each other.

planar embedding:
Clockwise orientation of adjacent vertices around each vertex.

A planar graph can have many planar embeddings.

A planar embedding can have many planar drawings!

faces: Connected region of the plane bounded by edges

Euler's polyhedra formula.

#faces - #edges + #vertices = #conn.comp. + 1

$f \quad - \quad m \quad + \quad n \quad = \quad c \quad + 1$

Proof. By induction on m:

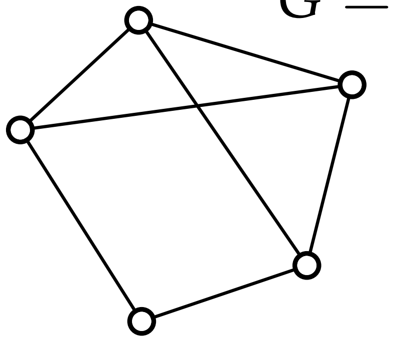
$m = 0 \Rightarrow f = 1$ and $c = n$

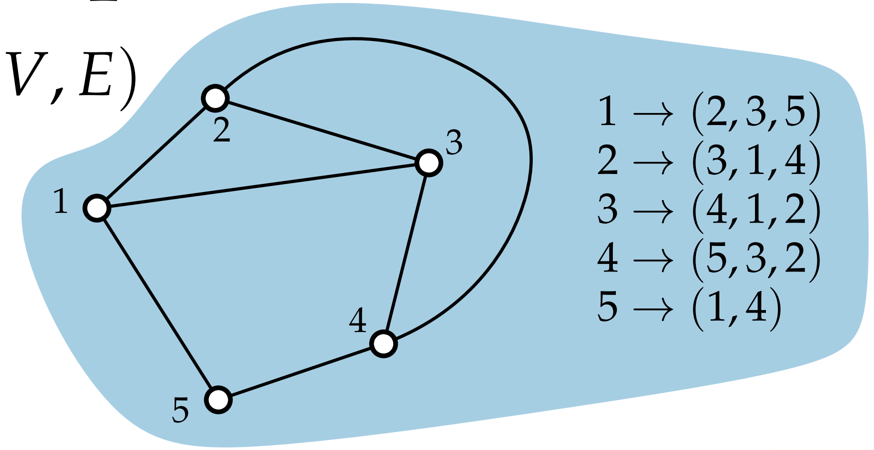
$\Rightarrow 0 - 0 + c = c + 1$ ✓

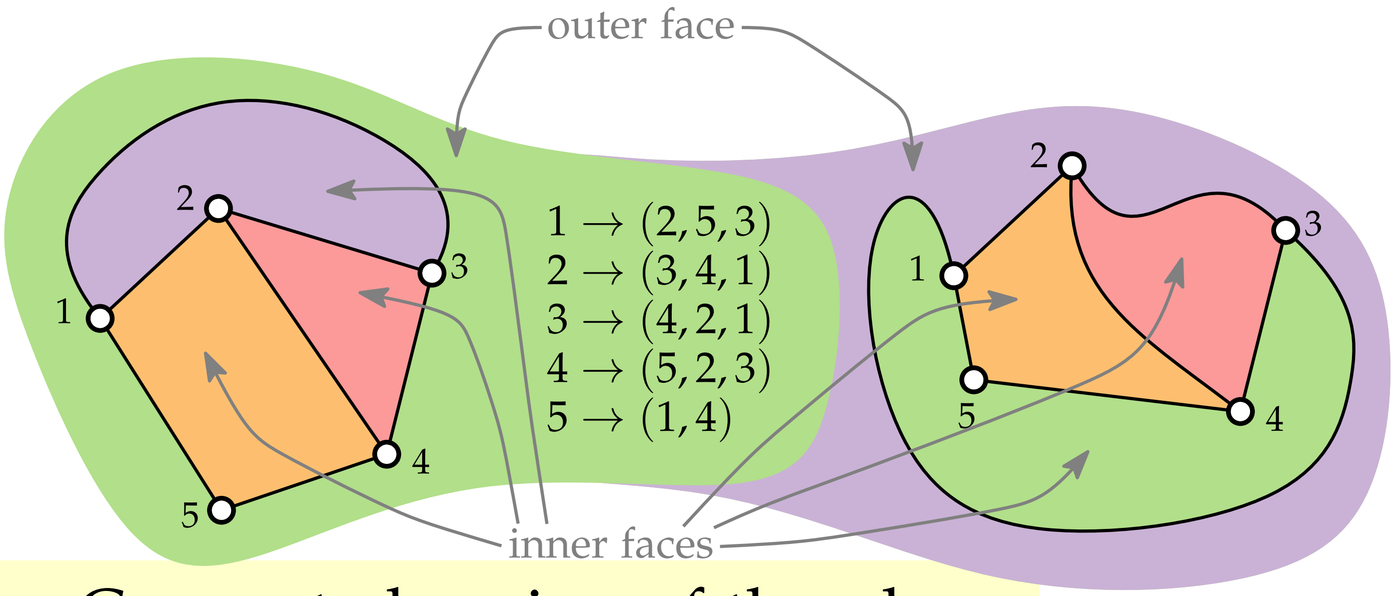
$m > 1 \Rightarrow$ remove 1 edge e $\Rightarrow m - 1$

Planar Graphs

$G = (V, E)$







G is **planar**:
it can be drawn in such a way that no edges cross each other.

planar embedding:
Clockwise orientation of adjacent vertices around each vertex.

A planar graph can have many planar embeddings.

A planar embedding can have many planar drawings!

faces: Connected region of the plane bounded by edges

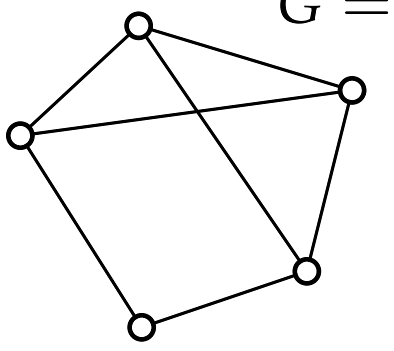
Euler's polyhedra formula.
#faces - #edges + #vertices = #conn.comp. + 1
f - m + n = c + 1

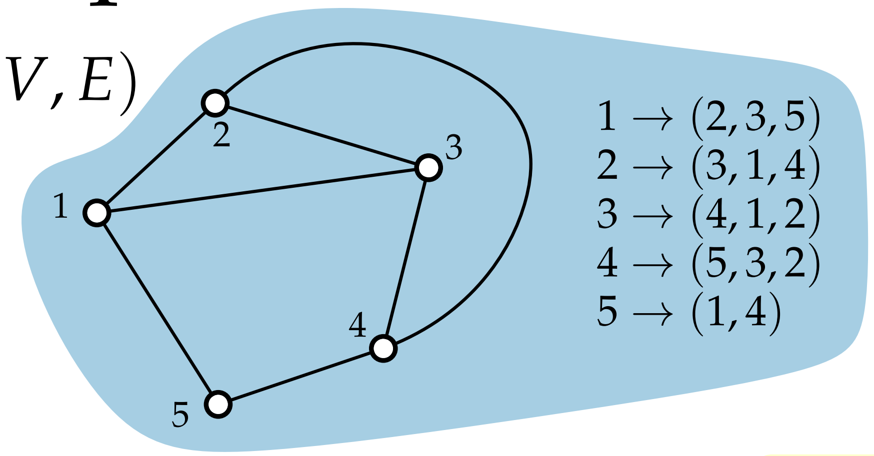
Proof. By induction on m:
$m = 0 \Rightarrow f = 1$ and $c = n$
$\Rightarrow 0 - 0 + c = c + 1$ ✓
$m > 1 \Rightarrow$ remove 1 edge e $\Rightarrow m - 1$

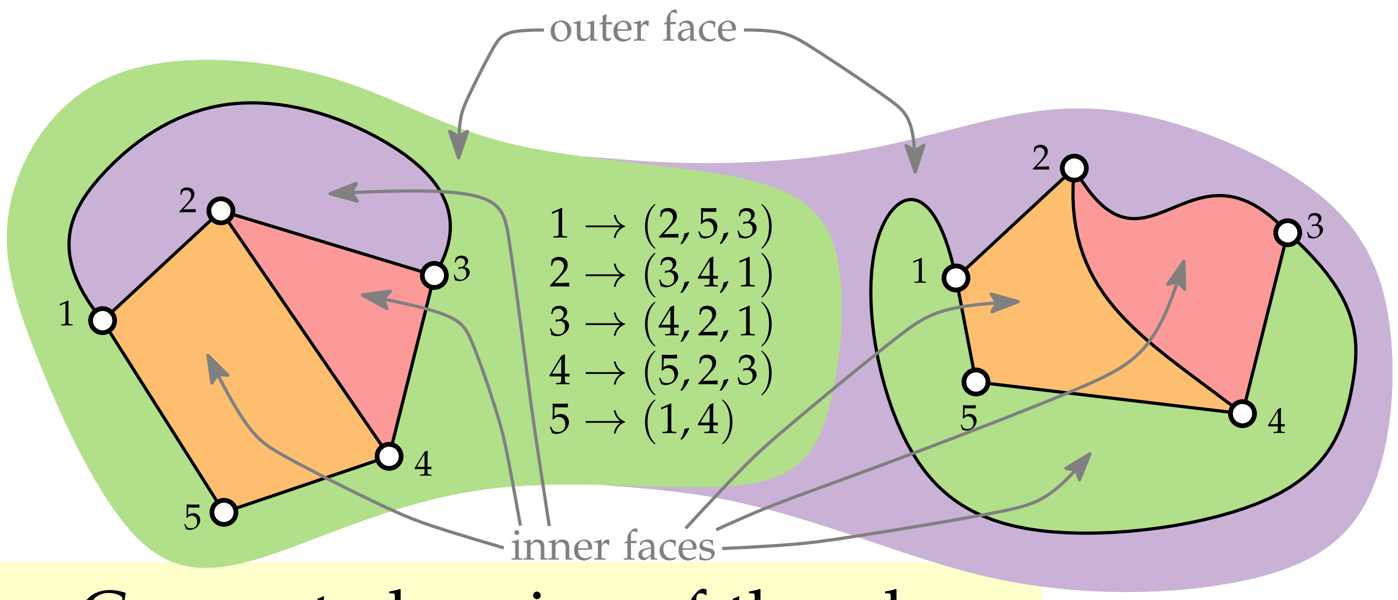
$\Rightarrow c + 1$

Planar Graphs

$G = (V, E)$







G is **planar**:
it can be drawn in such a way that no edges cross each other.

planar embedding:
Clockwise orientation of adjacent vertices around each vertex.

A planar graph can have many planar embeddings.

A planar embedding can have many planar drawings!

faces: Connected region of the plane bounded by edges

Euler's polyhedra formula.

#faces - #edges + #vertices = #conn.comp. + 1

$f - m + n = c + 1$

Proof. By induction on m:

$m = 0 \Rightarrow f = 1$ and $c = n$

$\Rightarrow 0 - 0 + c = c + 1$ ✓

$m > 1 \Rightarrow$ remove 1 edge $e \Rightarrow m - 1$

$\Rightarrow c + 1$

$\Rightarrow f + 1$

Properties of Planar Graphs

Euler's polyhedra formula.

#faces - #edges + #vertices = #conn.comp. + 1

$f - m + n = c + 1$

Properties of Planar Graphs

Euler's polyhedra formula.
#faces - #edges + #vertices = #conn.comp. + 1
$f - m + n = c + 1$

Theorem. G simple planar graph with $n \geq 3$.

Properties of Planar Graphs

Euler's polyhedra formula.

#faces - #edges + #vertices = #conn.comp. + 1

$f \quad - \quad m \quad + \quad n \quad = \quad c \quad + 1$

Theorem. G simple planar graph with $n \geq 3$.

1. $m \leq 3n - 6$

Properties of Planar Graphs

Euler's polyhedra formula.

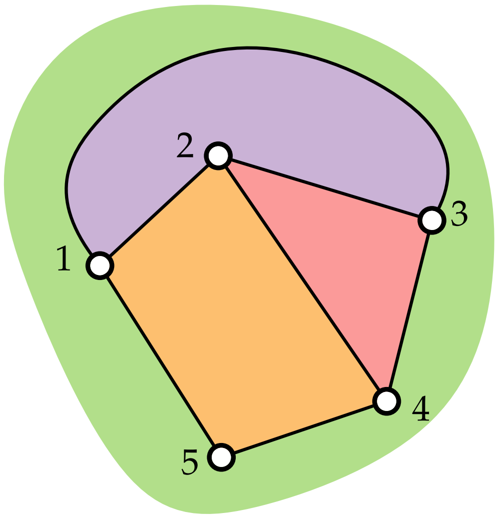
#faces - #edges + #vertices = #conn.comp. + 1

$f - m + n = c + 1$

Theorem. G simple planar graph with $n \geq 3$.

1. $m \leq 3n - 6$

Proof. 1.



Properties of Planar Graphs

Euler's polyhedra formula.

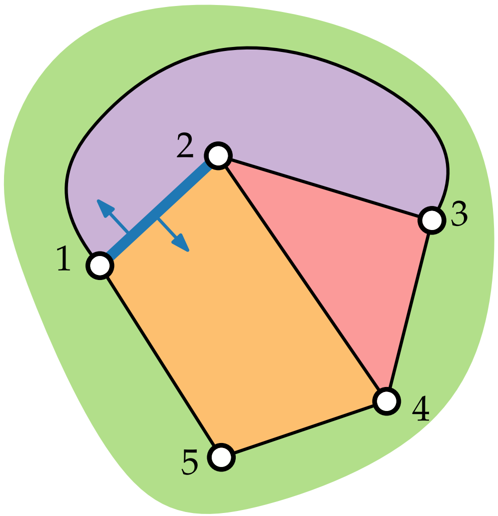
#faces - #edges + #vertices = #conn.comp. + 1

f - m + n = c + 1

Theorem. G simple planar graph with $n \geq 3$.

1. $m \leq 3n - 6$

Proof. 1. Every edge incident to ≤ 2 faces



Properties of Planar Graphs

Euler's polyhedra formula.

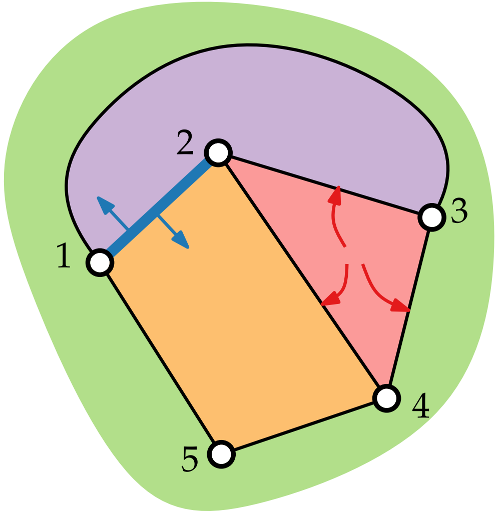
#faces - #edges + #vertices = #conn.comp. + 1

$f \quad - \quad m \quad + \quad n \quad = \quad c \quad + 1$

Theorem. G simple planar graph with $n \geq 3$.

1. $m \leq 3n - 6$

Proof. 1. Every edge incident to ≤ 2 faces
Every face incident to ≥ 3 edges



Properties of Planar Graphs

Euler's polyhedra formula.

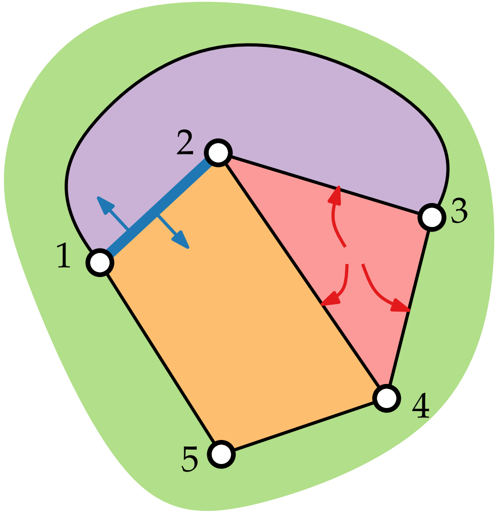
#faces - #edges + #vertices = #conn.comp. + 1

$f \quad - \quad m \quad + \quad n \quad = \quad c \quad + 1$

Theorem. G simple planar graph with $n \geq 3$.

1. $m \leq 3n - 6$

Proof. 1. Every edge incident to ≤ 2 faces
Every face incident to ≥ 3 edges
$\Rightarrow 3f \leq 2m$

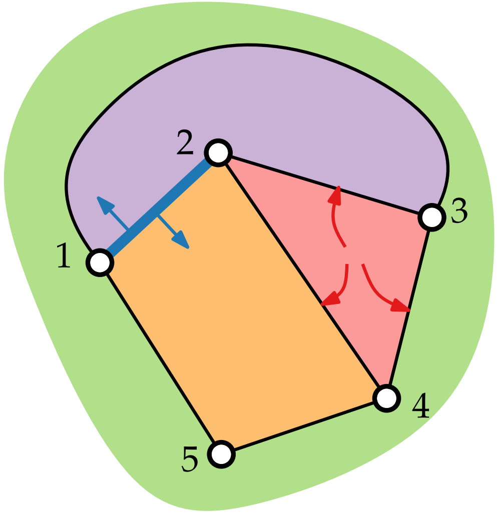


Properties of Planar Graphs

Euler's polyhedra formula.
#faces - #edges + #vertices = #conn.comp. + 1
f - m + n = c + 1

Theorem. G simple planar graph with $n \geq 3$.
1. $m \leq 3n - 6$

Proof. 1. Every edge incident to ≤ 2 faces
Every face incident to ≥ 3 edges
$\Rightarrow 3f \leq 2m$
$\Rightarrow 6 \leq 3c + 3 \leq 3f - 3m + 3n$



Properties of Planar Graphs

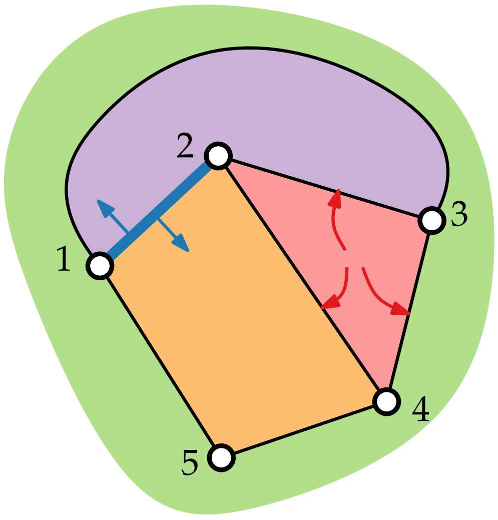
Euler's polyhedra formula.

#faces - #edges + #vertices = #conn.comp. + 1

$f - m + n = c + 1$

Theorem. G simple planar graph with $n \geq 3$.

1. $m \leq 3n - 6$



Proof. 1. Every edge incident to ≤ 2 faces

Every face incident to ≥ 3 edges

$\Rightarrow 3f \leq 2m$

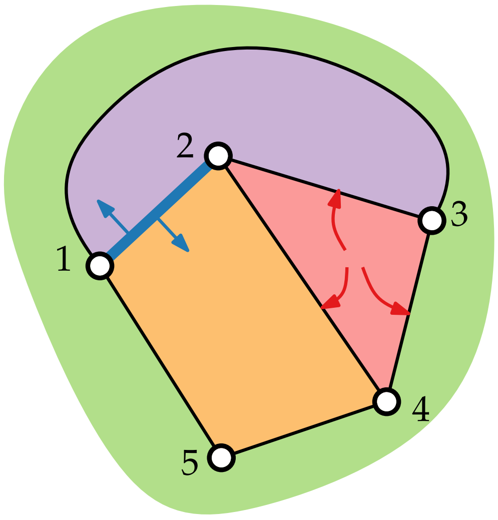
$\Rightarrow 6 \leq 3c + 3 \leq 3f - 3m + 3n$

Properties of Planar Graphs

Euler's polyhedra formula.
#faces - #edges + #vertices = #conn.comp. + 1
$f - m + n = c + 1$

Theorem. G simple planar graph with $n \geq 3$.
1. $m \leq 3n - 6$

Proof. 1. Every edge incident to ≤ 2 faces
Every face incident to ≥ 3 edges
$\Rightarrow 3f \leq 2m$
$\Rightarrow 6 \leq 3c + 3 \leq 3f - 3m + 3n \leq 2m - 3m + 3n$



Properties of Planar Graphs

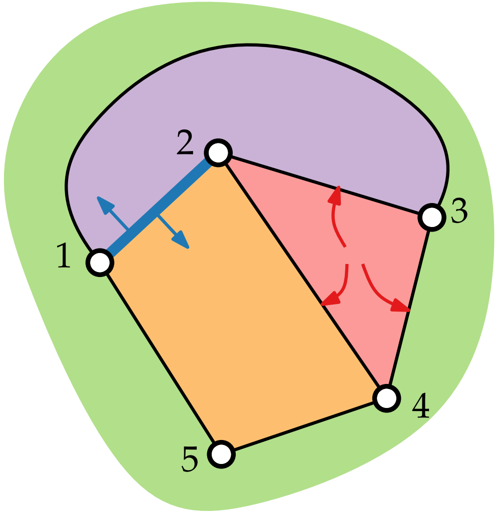
Euler's polyhedra formula.

#faces - #edges + #vertices = #conn.comp. + 1

$f \quad - \quad m \quad + \quad n \quad = \quad c \quad + 1$

Theorem. G simple planar graph with $n \geq 3$.

1. $m \leq 3n - 6$



Proof. 1. Every edge incident to ≤ 2 faces

Every face incident to ≥ 3 edges

$\Rightarrow 3f \leq 2m$

$\Rightarrow 6 \leq 3c + 3 \leq 3f - 3m + 3n \leq 2m - 3m + 3n = 3n - m$

Properties of Planar Graphs

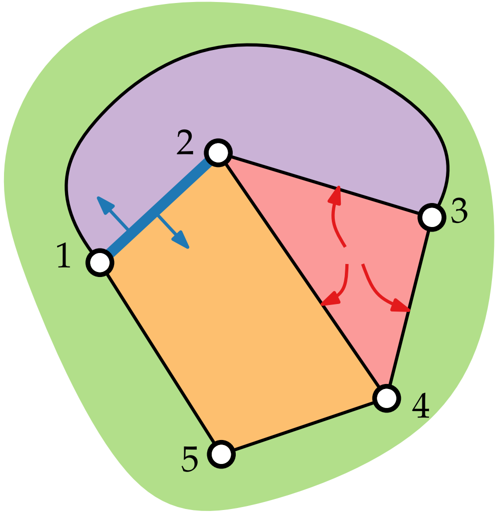
Euler's polyhedra formula.

#faces - #edges + #vertices = #conn.comp. + 1

f - m + n = c + 1

Theorem. G simple planar graph with $n \geq 3$.

1. $m \leq 3n - 6$



Proof. 1. Every edge incident to $\leq$ 2 faces

Every face incident to $\geq$ 3 edges

$\Rightarrow 3f \leq 2m$

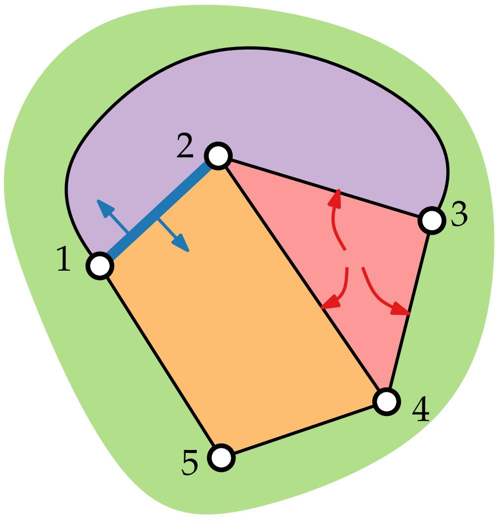
$\Rightarrow 6 \leq 3c + 3 \leq 3f - 3m + 3n \leq 2m - 3m + 3n = 3n - m$

$\Rightarrow m \leq 3n - 6$

Properties of Planar Graphs

Euler's polyhedra formula.
#faces - #edges + #vertices = #conn.comp. + 1
f - m + n = c + 1

Theorem. G simple planar graph with $n \geq 3$.
1. $m \leq 3n - 6$
2. $f \leq 2n - 4$



Proof. 1. Every edge incident to $\leq$ 2 faces
Every face incident to $\geq$ 3 edges
$\Rightarrow 3f \leq 2m$
$\Rightarrow 6 \leq 3c + 3 \leq 3f - 3m + 3n \leq 2m - 3m + 3n = 3n - m$
$\Rightarrow m \leq 3n - 6$

Properties of Planar Graphs

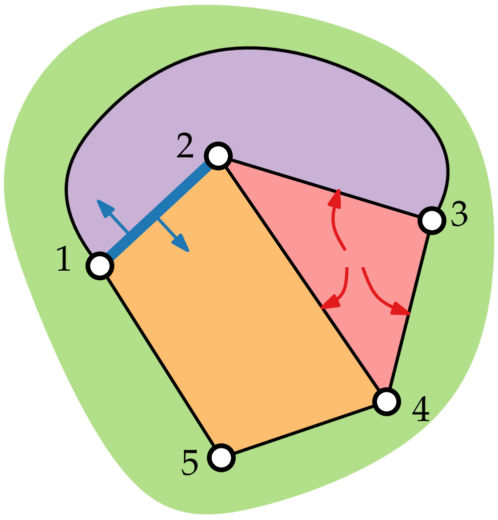
Euler's polyhedra formula.

#faces - #edges + #vertices = #conn.comp. + 1

$f - m + n = c + 1$

Theorem. G simple planar graph with $n \geq 3$.

1. $m \leq 3n - 6$ 2. $f \leq 2n - 4$



Proof. 1. Every edge incident to ≤ 2 faces

Every face incident to ≥ 3 edges

$\Rightarrow 3f \leq 2m$

$\Rightarrow 6 \leq 3c + 3 \leq 3f - 3m + 3n \leq 2m - 3m + 3n = 3n - m$

$\Rightarrow m \leq 3n - 6$

2. $3f \leq 2m$

Properties of Planar Graphs

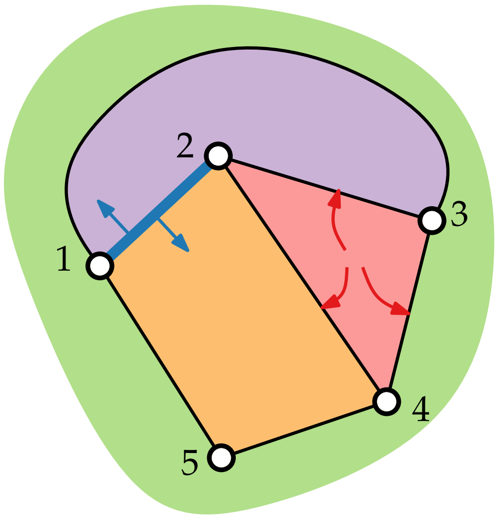
Euler's polyhedra formula.

#faces - #edges + #vertices = #conn.comp. + 1

f - m + n = c + 1

Theorem. G simple planar graph with $n \geq 3$.

1. $m \leq 3n - 6$
2. $f \leq 2n - 4$



Proof. 1. Every edge incident to ≤ 2 faces
Every face incident to ≥ 3 edges
$\Rightarrow 3f \leq 2m$
$\Rightarrow 6 \leq 3c + 3 \leq 3f - 3m + 3n \leq 2m - 3m + 3n = 3n - m$
$\Rightarrow m \leq 3n - 6$

2. $3f \leq 2m \leq 6n - 12$

Properties of Planar Graphs

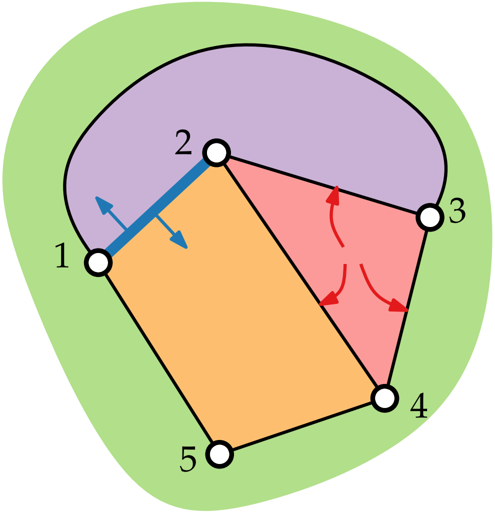
Euler's polyhedra formula.

#faces - #edges + #vertices = #conn.comp. + 1

$f - m + n = c + 1$

Theorem. G simple planar graph with $n \geq 3$.

1. $m \leq 3n - 6$
2. $f \leq 2n - 4$



Proof. 1. Every edge incident to ≤ 2 faces

Every face incident to ≥ 3 edges

$\Rightarrow 3f \leq 2m$

$\Rightarrow 6 \leq 3c + 3 \leq 3f - 3m + 3n \leq 2m - 3m + 3n = 3n - m$

$\Rightarrow m \leq 3n - 6$

2. $3f \leq 2m \leq 6n - 12 \Rightarrow f \leq 2n - 4$

Properties of Planar Graphs

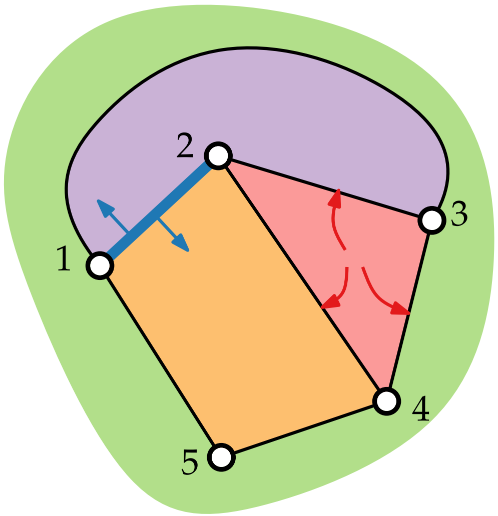
Euler's polyhedra formula.

#faces - #edges + #vertices = #conn.comp. + 1

$f - m + n = c + 1$

Theorem. G simple planar graph with $n \geq 3$.

1. $m \leq 3n - 6$ 2. $f \leq 2n - 4$
3. There is a vertex of degree at most five



Proof. 1. Every edge incident to ≤ 2 faces
Every face incident to ≥ 3 edges
$\Rightarrow 3f \leq 2m$
$\Rightarrow 6 \leq 3c + 3 \leq 3f - 3m + 3n \leq 2m - 3m + 3n = 3n - m$
$\Rightarrow m \leq 3n - 6$

2. $3f \leq 2m \leq 6n - 12 \Rightarrow f \leq 2n - 4$

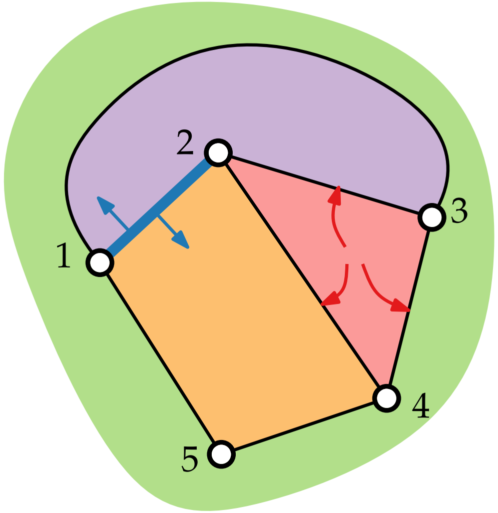
Properties of Planar Graphs

Euler's polyhedra formula.

#faces - #edges + #vertices = #conn.comp. + 1

f - m + n = c + 1

Theorem. G simple planar graph with $n \geq 3$.
1. $m \leq 3n - 6$
2. $f \leq 2n - 4$
3. There is a vertex of degree at most five



Proof. 1. Every edge incident to ≤ 2 faces
Every face incident to ≥ 3 edges
$\Rightarrow 3f \leq 2m$
$\Rightarrow 6 \leq 3c + 3 \leq 3f - 3m + 3n \leq 2m - 3m + 3n = 3n - m$
$\Rightarrow m \leq 3n - 6$

2. $3f \leq 2m \leq 6n - 12 \Rightarrow f \leq 2n - 4$

3. $\sum_{v \in V} \deg(v)$

Properties of Planar Graphs

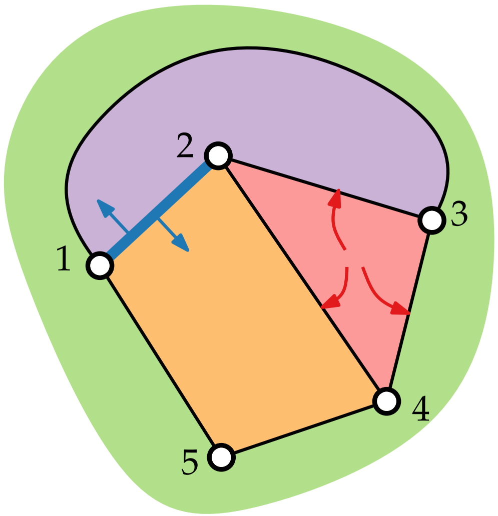
Euler's polyhedra formula.

#faces - #edges + #vertices = #conn.comp. + 1

f - m + n = c + 1

Theorem. G simple planar graph with $n \geq 3$.

1. $m \leq 3n - 6$ 2. $f \leq 2n - 4$
3. There is a vertex of degree at most five



Proof. 1. Every edge incident to ≤ 2 faces
Every face incident to ≥ 3 edges
$\Rightarrow 3f \leq 2m$
$\Rightarrow 6 \leq 3c + 3 \leq 3f - 3m + 3n \leq 2m - 3m + 3n = 3n - m$
$\Rightarrow m \leq 3n - 6$

2. $3f \leq 2m \leq 6n - 12 \Rightarrow f \leq 2n - 4$

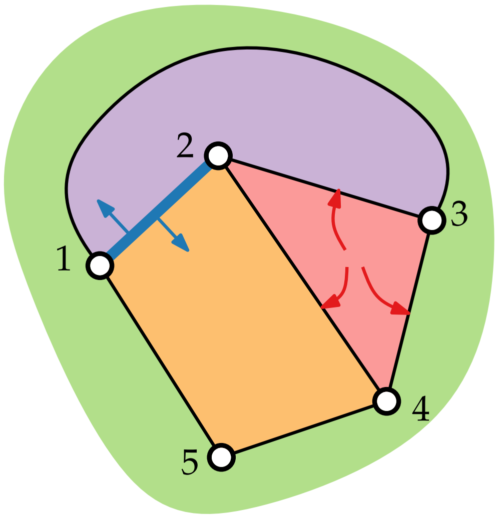
3. $\sum_{v \in V} \deg(v)$

Handshaking-Lemma.
$\sum_{v \in V} \deg(v) = 2|E|$

Properties of Planar Graphs

Euler's polyhedra formula.
#faces - #edges + #vertices = #conn.comp. + 1
f - m + n = c + 1

Theorem. G simple planar graph with $n \geq 3$.
1. $m \leq 3n - 6$
2. $f \leq 2n - 4$
3. There is a vertex of degree at most five



Proof. 1. Every edge incident to ≤ 2 faces
Every face incident to ≥ 3 edges
$\Rightarrow 3f \leq 2m$
$\Rightarrow 6 \leq 3c + 3 \leq 3f - 3m + 3n \leq 2m - 3m + 3n = 3n - m$
$\Rightarrow m \leq 3n - 6$

2. $3f \leq 2m \leq 6n - 12 \Rightarrow f \leq 2n - 4$

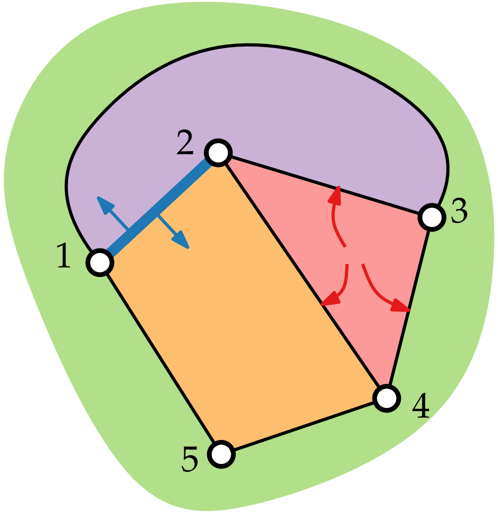
3. $\sum_{v \in V} \deg(v) = 2m$

Handshaking-Lemma.
$\sum_{v \in V} \deg(v) = 2|E|$

Properties of Planar Graphs

Euler's polyhedra formula.
#faces - #edges + #vertices = #conn.comp. + 1
f - m + n = c + 1

Theorem. G simple planar graph with $n \geq 3$.
1. $m \leq 3n - 6$ 2. $f \leq 2n - 4$
3. There is a vertex of degree at most five



Proof. 1. Every edge incident to ≤ 2 faces
Every face incident to ≥ 3 edges
$\Rightarrow 3f \leq 2m$
$\Rightarrow 6 \leq 3c + 3 \leq 3f - 3m + 3n \leq 2m - 3m + 3n = 3n - m$
$\Rightarrow m \leq 3n - 6$

2. $3f \leq 2m \leq 6n - 12 \Rightarrow f \leq 2n - 4$

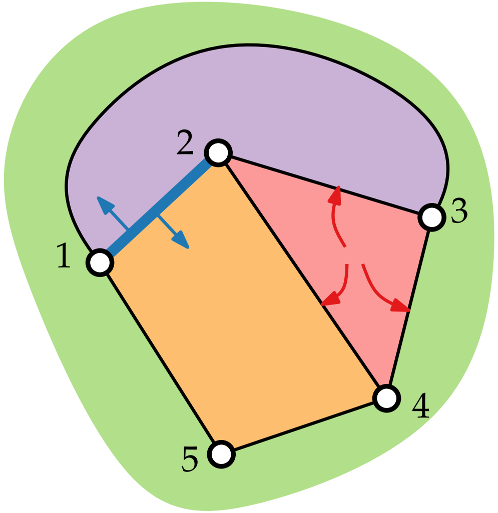
3. $\sum_{v \in V} \deg(v) = 2m \leq 6n - 12$

Handshaking-Lemma.
$\sum_{v \in V} \deg(v) = 2|E|$

Properties of Planar Graphs

Euler's polyhedra formula.
#faces - #edges + #vertices = #conn.comp. + 1
f - m + n = c + 1

Theorem. G simple planar graph with $n \geq 3$.
1. $m \leq 3n - 6$ 2. $f \leq 2n - 4$
3. There is a vertex of degree at most five



Proof. 1. Every edge incident to $\leq$ 2 faces
Every face incident to $\geq$ 3 edges
$\Rightarrow 3f \leq 2m$
$\Rightarrow 6 \leq 3c + 3 \leq 3f - 3m + 3n \leq 2m - 3m + 3n = 3n - m$
$\Rightarrow m \leq 3n - 6$

2. $3f \leq 2m \leq 6n - 12 \Rightarrow f \leq 2n - 4$

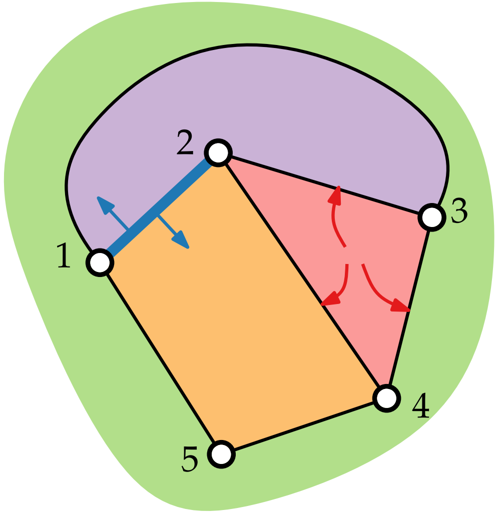
3. $\sum_{v \in V} \deg(v) = 2m \leq 6n - 12$
$\Rightarrow \min_{v \in V} \deg(v)$

Handshaking-Lemma.
$\sum_{v \in V} \deg(v) = 2|E|$

Properties of Planar Graphs

Euler's polyhedra formula.
#faces - #edges + #vertices = #conn.comp. + 1
$f - m + n = c + 1$

Theorem. G simple planar graph with $n \geq 3$.
1. $m \leq 3n - 6$ 2. $f \leq 2n - 4$
3. There is a vertex of degree at most five



Proof. 1. Every edge incident to ≤ 2 faces
Every face incident to ≥ 3 edges
$\Rightarrow 3f \leq 2m$
$\Rightarrow 6 \leq 3c + 3 \leq 3f - 3m + 3n \leq 2m - 3m + 3n = 3n - m$
$\Rightarrow m \leq 3n - 6$
2. $3f \leq 2m \leq 6n - 12 \Rightarrow f \leq 2n - 4$
3. $\sum_{v \in V} \deg(v) = 2m \leq 6n - 12$
$\Rightarrow \min_{v \in V} \deg(v) \leq 1/n \sum_{v \in V} \deg(v)$

Handshaking-Lemma.
$\sum_{v \in V} \deg(v) = 2|E|$

Properties of Planar Graphs

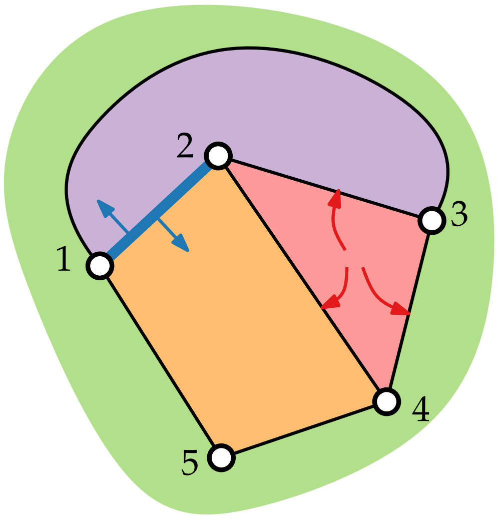
Euler's polyhedra formula.

#faces - #edges + #vertices = #conn.comp. + 1

$f \quad - \quad m \quad + \quad n \quad = \quad c \quad + 1$

Theorem. G simple planar graph with $n \geq 3$.

1. $m \leq 3n - 6$ 2. $f \leq 2n - 4$
3. There is a vertex of degree at most five



Proof. 1. Every edge incident to ≤ 2 faces

Every face incident to ≥ 3 edges

$\Rightarrow 3f \leq 2m$

$\Rightarrow 6 \leq 3c + 3 \leq 3f - 3m + 3n \leq 2m - 3m + 3n = 3n - m$

$\Rightarrow m \leq 3n - 6$

2. $3f \leq 2m \leq 6n - 12 \Rightarrow f \leq 2n - 4$

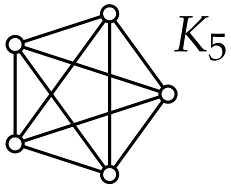
3. $\sum_{v \in V} \deg(v) = 2m \leq 6n - 12$

$\Rightarrow \min_{v \in V} \deg(v) \leq 1/n \sum_{v \in V} \deg(v) < 6$

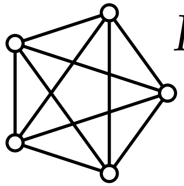
Handshaking-Lemma.

$\sum_{v \in V} \deg(v) = 2|E|$

Complete graphs



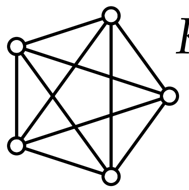
Complete graphs



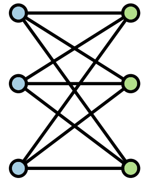
$K_n = \left(V, \binom{V}{2}\right)$ is the **complete** graph on n vertices.

Complete graphs

$K_n = \left(V, \binom{V}{2}\right)$ is the **complete** graph on n vertices.

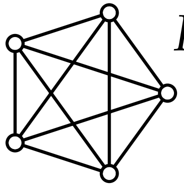


K_5

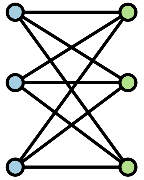


$K_{3,3}$

Complete graphs

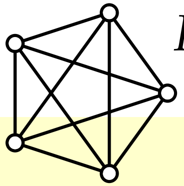


$K_n = \left(V, \binom{V}{2}\right)$ is the **complete** graph on n vertices.



$K_{n_1,n_2} = (V_1 \cup V_2, V_1 \times V_2)$ with $|V_1| = n_1$ and $|V_2| = n_2$ is a **complete bipartite** graph on $n = n_1 + n_2$ vertices.

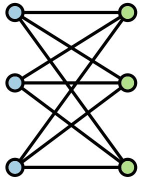
Complete graphs



K_5

$K_n = \left(V, \binom{V}{2}\right)$ is the **complete** graph on n vertices.

V_1 V_2

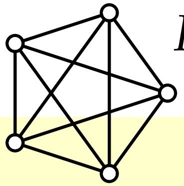


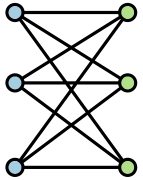
$K_{3,3}$

$K_{n_1,n_2} = (V_1 \cup V_2, V_1 \times V_2)$ with $|V_1| = n_1$ and $|V_2| = n_2$ is a **complete bipartite** graph on $n = n_1 + n_2$ vertices.

A **bipartite** graph is a subgraph of a K_{n_1,n_2}; V_1 and V_2 are called **bipartitions**.

Complete graphs





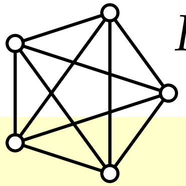
$K_n = \left(V, \binom{V}{2}\right)$ is the **complete** graph on n vertices.

$K_{n_1,n_2} = (V_1 \cup V_2, V_1 \times V_2)$ with $|V_1| = n_1$ and $|V_2| = n_2$ is a **complete bipartite** graph on $n = n_1 + n_2$ vertices.

A **bipartite** graph is a subgraph of a K_{n_1,n_2}; V_1 and V_2 are called **bipartitions**.

Theorem. K_5 and $K_{3,3}$ are not planar.

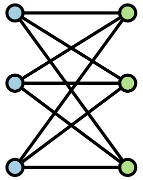
Complete graphs



K_5

$K_n = \left(V, \binom{V}{2}\right)$ is the **complete** graph on n vertices.

V_1 V_2



$K_{3,3}$

$K_{n_1,n_2} = (V_1 \cup V_2, V_1 \times V_2)$ with $|V_1| = n_1$ and $|V_2| = n_2$ is a **complete bipartite** graph on $n = n_1 + n_2$ vertices.

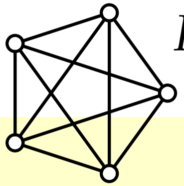
A **bipartite** graph is a subgraph of a K_{n_1,n_2}; V_1 and V_2 are called **bipartitions**.

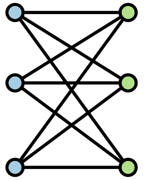
Theorem. K_5 and $K_{3,3}$ are not planar.

Proof.

K_5:

Complete graphs


K_5


V_1 V_2
$K_{3,3}$

$K_n = \left(V, \binom{V}{2}\right)$ is the **complete** graph on n vertices.

$K_{n_1,n_2} = (V_1 \cup V_2, V_1 \times V_2)$ with $|V_1| = n_1$ and $|V_2| = n_2$ is a **complete bipartite** graph on $n = n_1 + n_2$ vertices.

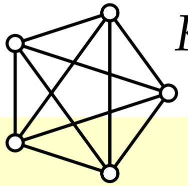
A **bipartite** graph is a subgraph of a K_{n_1,n_2}; V_1 and V_2 are called **bipartitions**.

Theorem. K_5 and $K_{3,3}$ are not planar.

Proof.

K_5: $m = \binom{5}{2}$

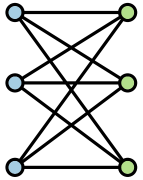
Complete graphs



K_5

$K_n = \left(V, \binom{V}{2}\right)$ is the **complete** graph on n vertices.

V_1 V_2



$K_{3,3}$

$K_{n_1,n_2} = (V_1 \cup V_2, V_1 \times V_2)$ with $|V_1| = n_1$ and $|V_2| = n_2$ is a **complete bipartite** graph on $n = n_1 + n_2$ vertices.

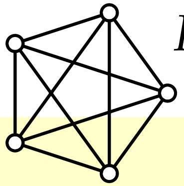
A **bipartite** graph is a subgraph of a K_{n_1,n_2}; V_1 and V_2 are called **bipartitions**.

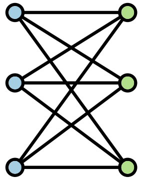
Theorem. K_5 and $K_{3,3}$ are not planar.

Proof.

K_5: $m = \binom{5}{2} = \frac{5 \cdot 4}{1 \cdot 2}$

Complete graphs





$K_n = \left(V, \binom{V}{2}\right)$ is the **complete** graph on n vertices.

$K_{n_1,n_2} = (V_1 \cup V_2, V_1 \times V_2)$ with $|V_1| = n_1$ and $|V_2| = n_2$ is a **complete bipartite** graph on $n = n_1 + n_2$ vertices.

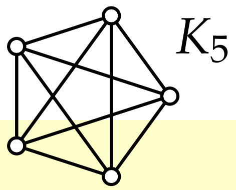
A **bipartite** graph is a subgraph of a K_{n_1,n_2}; V_1 and V_2 are called **bipartitions**.

Theorem. K_5 and $K_{3,3}$ are not planar.

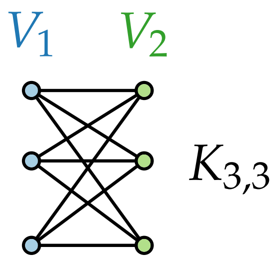
Proof.

K_5: $m = \binom{5}{2} = \frac{5 \cdot 4}{1 \cdot 2} = 10$

Complete graphs



$K_n = \left(V, \binom{V}{2}\right)$ is the **complete** graph on n vertices.



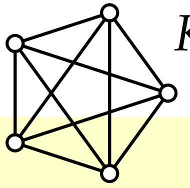
$K_{n_1,n_2} = (V_1 \cup V_2, V_1 \times V_2)$ with $|V_1| = n_1$ and $|V_2| = n_2$ is a **complete bipartite** graph on $n = n_1 + n_2$ vertices.
A **bipartite** graph is a subgraph of a K_{n_1,n_2}; V_1 and V_2 are called **bipartitions**.

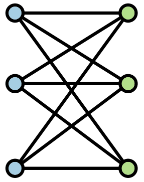
Theorem. K_5 and $K_{3,3}$ are not planar.

Proof.

K_5: $m = \binom{5}{2} = \frac{5 \cdot 4}{1 \cdot 2} = 10$ $\quad 3n - 6$

Complete graphs





$K_n = \left(V, \binom{V}{2}\right)$ is the **complete** graph on n vertices.

$K_{n_1,n_2} = (V_1 \cup V_2, V_1 \times V_2)$ with $|V_1| = n_1$ and $|V_2| = n_2$ is a **complete bipartite** graph on $n = n_1 + n_2$ vertices.

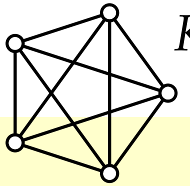
A **bipartite** graph is a subgraph of a K_{n_1,n_2}; V_1 and V_2 are called **bipartitions**.

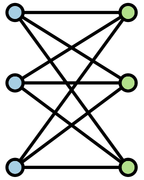
Theorem. K_5 and $K_{3,3}$ are not planar.

Proof.

K_5: $m = \binom{5}{2} = \frac{5\cdot4}{1\cdot2} = 10$ $\quad 3 \cdot 5 - 6 = 3n - 6$

Complete graphs





$K_n = \left(V, \binom{V}{2}\right)$ is the **complete** graph on n vertices.

$K_{n_1,n_2} = (V_1 \cup V_2, V_1 \times V_2)$ with $|V_1| = n_1$ and $|V_2| = n_2$ is a **complete bipartite** graph on $n = n_1 + n_2$ vertices.

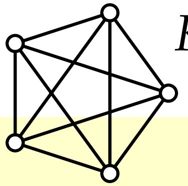
A **bipartite** graph is a subgraph of a K_{n_1,n_2}; V_1 and V_2 are called **bipartitions**.

Theorem. K_5 and $K_{3,3}$ are not planar.

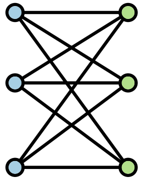
Proof.

K_5: $\quad m = \binom{5}{2} = \frac{5\cdot4}{1\cdot2} = 10 > 9 = 3 \cdot 5 - 6 = 3n - 6$

Complete graphs



$K_n = \left(V, \binom{V}{2}\right)$ is the **complete** graph on n vertices.



$K_{n_1,n_2} = (V_1 \cup V_2, V_1 \times V_2)$ with $|V_1| = n_1$ and $|V_2| = n_2$ is a **complete bipartite** graph on $n = n_1 + n_2$ vertices.
A **bipartite** graph is a subgraph of a K_{n_1,n_2}; V_1 and V_2 are called **bipartitions**.

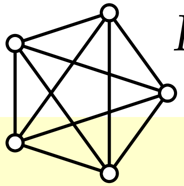
Theorem. K_5 and $K_{3,3}$ are not planar.

Theorem. G simple planar graph with $n \geq 3$.
1. $m \leq 3n - 6$ 2. $f \leq 2n - 4$
3. There is a vertex of degree at most five

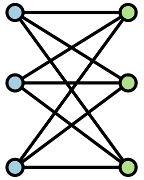
Proof.

K_5: $m = \binom{5}{2} = \frac{5 \cdot 4}{1 \cdot 2} = 10 > 9 = 3 \cdot 5 - 6 = 3n - 6$

Complete graphs



$K_n = \left(V, \binom{V}{2}\right)$ is the **complete** graph on n vertices.



$K_{n_1,n_2} = (V_1 \cup V_2, V_1 \times V_2)$ with $|V_1| = n_1$ and $|V_2| = n_2$ is a **complete bipartite** graph on $n = n_1 + n_2$ vertices.

A **bipartite** graph is a subgraph of a K_{n_1,n_2}; V_1 and V_2 are called **bipartitions**.

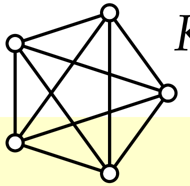
Theorem. K_5 and $K_{3,3}$ are not planar.

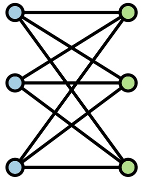
Theorem. G simple planar graph with $n \geq 3$.
1. $m \leq 3n - 6$
2. $f \leq 2n - 4$
3. There is a vertex of degree at most five

Proof.

K_5: $m = \binom{5}{2} = \frac{5 \cdot 4}{1 \cdot 2} = 10 > 9 = 3 \cdot 5 - 6 = 3n - 6$ ✓

Complete graphs





$K_n = \left(V, \binom{V}{2}\right)$ is the **complete** graph on n vertices.

$K_{n_1,n_2} = (V_1 \cup V_2, V_1 \times V_2)$ with $|V_1| = n_1$ and $|V_2| = n_2$ is a **complete bipartite** graph on $n = n_1 + n_2$ vertices.

A **bipartite** graph is a subgraph of a K_{n_1,n_2}; V_1 and V_2 are called **bipartitions**.

Theorem. K_5 and $K_{3,3}$ are not planar.

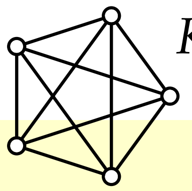
Theorem. G simple planar graph with $n \geq 3$.
1. $m \leq 3n - 6$ 2. $f \leq 2n - 4$
3. There is a vertex of degree at most five

Proof.

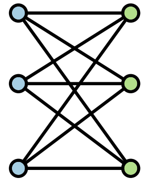
K_5: $m = \binom{5}{2} = \frac{5 \cdot 4}{1 \cdot 2} = 10 > 9 = 3 \cdot 5 - 6 = 3n - 6$ ✓

$K_{3,3}$:

Complete graphs



$K_n = \left(V, \binom{V}{2}\right)$ is the **complete** graph on n vertices.



$K_{n_1,n_2} = (V_1 \cup V_2, V_1 \times V_2)$ with $|V_1| = n_1$ and $|V_2| = n_2$ is a **complete bipartite** graph on $n = n_1 + n_2$ vertices.
A **bipartite** graph is a subgraph of a K_{n_1,n_2}; V_1 and V_2 are called **bipartitions**.

Theorem. K_5 and $K_{3,3}$ are not planar.

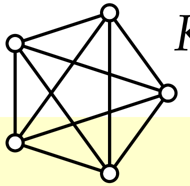
Theorem. G simple planar graph with $n \geq 3$.
1. $m \leq 3n - 6$ 2. $f \leq 2n - 4$
3. There is a vertex of degree at most five

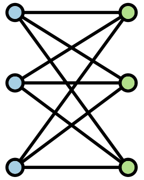
Proof.

K_5: $m = \binom{5}{2} = \frac{5 \cdot 4}{1 \cdot 2} = 10 > 9 = 3 \cdot 5 - 6 = 3n - 6$ ✓

$K_{3,3}$: $m = 3 \cdot 3$

Complete graphs


K_5



$K_{3,3}$

$K_n = \left(V, \binom{V}{2}\right)$ is the **complete** graph on n vertices.
$K_{n_1,n_2} = (V_1 \cup V_2, V_1 \times V_2)$ with $|V_1| = n_1$ and $|V_2| = n_2$ is a **complete bipartite** graph on $n = n_1 + n_2$ vertices.
A **bipartite** graph is a subgraph of a K_{n_1,n_2}; V_1 and V_2 are called **bipartitions**.

Theorem. K_5 and $K_{3,3}$ are not planar.

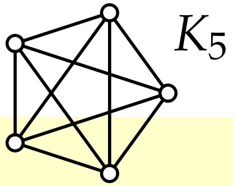
Theorem. G simple planar graph with $n \geq 3$.
1. $m \leq 3n - 6$ 2. $f \leq 2n - 4$
3. There is a vertex of degree at most five

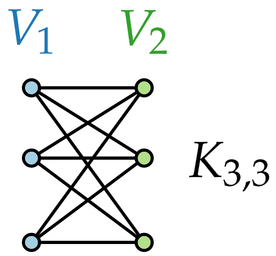
Proof.

K_5: $m = \binom{5}{2} = \frac{5\cdot 4}{1\cdot 2} = 10 > 9 = 3 \cdot 5 - 6 = 3n - 6$ ✓

$K_{3,3}$: $m = 3 \cdot 3 = 9$

Complete graphs





$K_n = \left(V, \binom{V}{2}\right)$ is the **complete** graph on n vertices.
$K_{n_1,n_2} = (V_1 \cup V_2, V_1 \times V_2)$ with $|V_1| = n_1$ and $|V_2| = n_2$ is a **complete bipartite** graph on $n = n_1 + n_2$ vertices.
A **bipartite** graph is a subgraph of a K_{n_1,n_2}; V_1 and V_2 are called **bipartitions**.

Theorem. K_5 and $K_{3,3}$ are not planar.

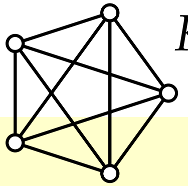
Theorem. G simple planar graph with $n \geq 3$.
1. $m \leq 3n - 6$
2. $f \leq 2n - 4$
3. There is a vertex of degree at most five

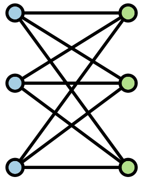
Proof.

K_5: $m = \binom{5}{2} = \frac{5\cdot 4}{1\cdot 2} = 10 > 9 = 3 \cdot 5 - 6 = 3n - 6$ ✓

$K_{3,3}$: $m = 3 \cdot 3 = 9$ $\quad 3n - 6$

Complete graphs





$K_n = \left(V, \binom{V}{2}\right)$ is the **complete** graph on n vertices.

$K_{n_1,n_2} = (V_1 \cup V_2, V_1 \times V_2)$ with $|V_1| = n_1$ and $|V_2| = n_2$ is a **complete bipartite** graph on $n = n_1 + n_2$ vertices.

A **bipartite** graph is a subgraph of a K_{n_1,n_2}; V_1 and V_2 are called **bipartitions**.

Theorem. K_5 and $K_{3,3}$ are not planar.

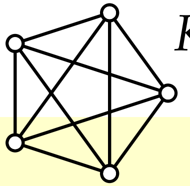
Theorem. G simple planar graph with $n \geq 3$.
1. $m \leq 3n - 6$
2. $f \leq 2n - 4$
3. There is a vertex of degree at most five

Proof.

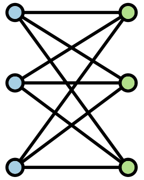
K_5: $m = \binom{5}{2} = \frac{5 \cdot 4}{1 \cdot 2} = 10 > 9 = 3 \cdot 5 - 6 = 3n - 6$ ✓

$K_{3,3}$: $m = 3 \cdot 3 = 9$ $\quad 3 \cdot 6 - 6 = 3n - 6$

Complete graphs



K_5



V_1 V_2

$K_{3,3}$

$K_n = \left(V, \binom{V}{2}\right)$ is the **complete** graph on n vertices.

$K_{n_1,n_2} = (V_1 \cup V_2, V_1 \times V_2)$ with $|V_1| = n_1$ and $|V_2| = n_2$ is a **complete bipartite** graph on $n = n_1 + n_2$ vertices.

A **bipartite** graph is a subgraph of a K_{n_1,n_2}; V_1 and V_2 are called **bipartitions**.

Theorem. K_5 and $K_{3,3}$ are not planar.

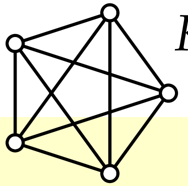
Theorem. G simple planar graph with $n \geq 3$.
1. $m \leq 3n - 6$ 2. $f \leq 2n - 4$
3. There is a vertex of degree at most five

Proof.

K_5: $m = \binom{5}{2} = \frac{5 \cdot 4}{1 \cdot 2} = 10 > 9 = 3 \cdot 5 - 6 = 3n - 6$ ✓

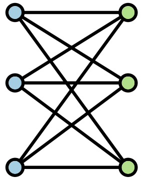
$K_{3,3}$: $m = 3 \cdot 3 = 9 < 12 = 3 \cdot 6 - 6 = 3n - 6$

Complete graphs


K_5

$K_n = \left(V, \binom{V}{2}\right)$ is the **complete** graph on n vertices.

V_1 V_2


$K_{3,3}$

$K_{n_1,n_2} = (V_1 \cup V_2, V_1 \times V_2)$ with $|V_1| = n_1$ and $|V_2| = n_2$ is a **complete bipartite** graph on $n = n_1 + n_2$ vertices.

A **bipartite** graph is a subgraph of a K_{n_1,n_2}; V_1 and V_2 are called **bipartitions**.

Theorem. K_5 and $K_{3,3}$ are not planar.

Theorem. G simple planar graph with $n \geq 3$.
1. $m \leq 3n - 6$ 2. $f \leq 2n - 4$
3. There is a vertex of degree at most five

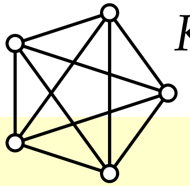
Proof.

K_5: $m = \binom{5}{2} = \frac{5 \cdot 4}{1 \cdot 2} = 10 > 9 = 3 \cdot 5 - 6 = 3n - 6$ ✓

$K_{3,3}$: $m = 3 \cdot 3 = 9 < 12 = 3 \cdot 6 - 6 = 3n - 6$

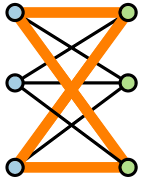
$\Rightarrow$ *no* contradiction to the theorem!

Complete graphs



$K_n = \left(V, \binom{V}{2}\right)$ is the **complete** graph on n vertices.

$K_{n_1,n_2} = (V_1 \cup V_2, V_1 \times V_2)$ with $|V_1| = n_1$ and $|V_2| = n_2$ is a **complete bipartite** graph on $n = n_1 + n_2$ vertices.



A **bipartite** graph is a subgraph of a K_{n_1,n_2}; V_1 and V_2 are called **bipartitions**.

Theorem. K_5 and $K_{3,3}$ are not planar.

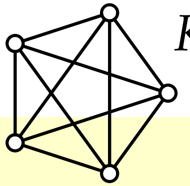
Theorem. G simple planar graph with $n \geq 3$.
1. $m \leq 3n - 6$ 2. $f \leq 2n - 4$
3. There is a vertex of degree at most five

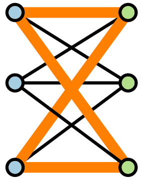
Proof.

K_5: $m = \binom{5}{2} = \frac{5 \cdot 4}{1 \cdot 2} = 10 > 9 = 3 \cdot 5 - 6 = 3n - 6$ ✓

$K_{3,3}$: $m = 3 \cdot 3 = 9 < 12 = 3 \cdot 6 - 6 = 3n - 6$
$\Rightarrow$ *no* contradiction to the theorem!
There is no cycle of length 3.

Complete graphs





$K_n = \left(V, \binom{V}{2}\right)$ is the **complete** graph on n vertices.

$K_{n_1,n_2} = (V_1 \cup V_2, V_1 \times V_2)$ with $|V_1| = n_1$ and $|V_2| = n_2$ is a **complete bipartite** graph on $n = n_1 + n_2$ vertices.

A **bipartite** graph is a subgraph of a K_{n_1,n_2}; V_1 and V_2 are called **bipartitions**.

Theorem. K_5 and $K_{3,3}$ are not planar.

Theorem. G simple planar graph with $n \geq 3$.
1. $m \leq 3n - 6$
2. $f \leq 2n - 4$
3. There is a vertex of degree at most five

Proof.

K_5: $m = \binom{5}{2} = \frac{5 \cdot 4}{1 \cdot 2} = 10 > 9 = 3 \cdot 5 - 6 = 3n - 6$ ✓

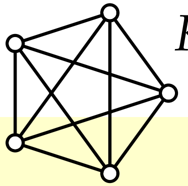
$K_{3,3}$: $m = 3 \cdot 3 = 9 < 12 = 3 \cdot 6 - 6 = 3n - 6$

$\Rightarrow$ *no* contradiction to the theorem!

There is no cycle of length 3.

Every face incident to ≥ 4 edges (in hypothetical planar drawing)

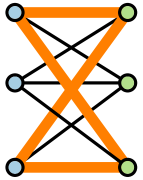
Complete graphs



K_5

$K_n = \left(V, \binom{V}{2}\right)$ is the **complete** graph on n vertices.

V_1 V_2



$K_{3,3}$

$K_{n_1,n_2} = (V_1 \cup V_2, V_1 \times V_2)$ with $|V_1| = n_1$ and $|V_2| = n_2$ is a **complete bipartite** graph on $n = n_1 + n_2$ vertices.

A **bipartite** graph is a subgraph of a K_{n_1,n_2}; V_1 and V_2 are called **bipartitions**.

Theorem. K_5 and $K_{3,3}$ are not planar.

Theorem. G simple planar graph with $n \geq 3$.
1. $m \leq 3n - 6$
2. $f \leq 2n - 4$
3. There is a vertex of degree at most five

Proof.

K_5: $m = \binom{5}{2} = \frac{5 \cdot 4}{1 \cdot 2} = 10 > 9 = 3 \cdot 5 - 6 = 3n - 6$ ✓

$K_{3,3}$: $m = 3 \cdot 3 = 9 < 12 = 3 \cdot 6 - 6 = 3n - 6$

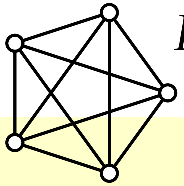
$\Rightarrow$ *no* contradiction to the theorem!

There is no cycle of length 3.

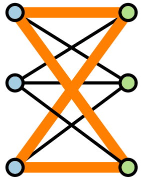
Every face incident to ≥ 4 edges (in hypothetical planar drawing)

$\Rightarrow 4f \leq 2m$

Complete graphs



$K_n = \left(V, \binom{V}{2}\right)$ is the **complete** graph on n vertices.



$K_{n_1,n_2} = (V_1 \cup V_2, V_1 \times V_2)$ with $|V_1| = n_1$ and $|V_2| = n_2$ is a **complete bipartite** graph on $n = n_1 + n_2$ vertices.
A **bipartite** graph is a subgraph of a K_{n_1,n_2}; V_1 and V_2 are called **bipartitions**.

Theorem. K_5 and $K_{3,3}$ are not planar.

Theorem. G simple planar graph with $n \geq 3$.
1. $m \leq 3n - 6$ 2. $f \leq 2n - 4$
3. There is a vertex of degree at most five

Proof.

K_5: $m = \binom{5}{2} = \frac{5 \cdot 4}{1 \cdot 2} = 10 > 9 = 3 \cdot 5 - 6 = 3n - 6$ ✓

$K_{3,3}$: $m = 3 \cdot 3 = 9 < 12 = 3 \cdot 6 - 6 = 3n - 6$

$\Rightarrow$ *no* contradiction to the theorem!

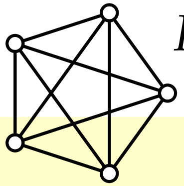
There is no cycle of length 3.

Every face incident to ≥ 4 edges (in hypothetical planar drawing)

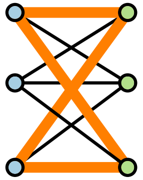
$\Rightarrow 4f \leq 2m$

$\Rightarrow 8 \leq 4c + 4 \leq 4f - 4m + 4n$

Complete graphs



$K_n = \left(V, \binom{V}{2}\right)$ is the **complete** graph on n vertices.



$K_{n_1,n_2} = (V_1 \cup V_2, V_1 \times V_2)$ with $|V_1| = n_1$ and $|V_2| = n_2$ is a **complete bipartite** graph on $n = n_1 + n_2$ vertices.
A **bipartite** graph is a subgraph of a K_{n_1,n_2}; V_1 and V_2 are called **bipartitions**.

Theorem. K_5 and $K_{3,3}$ are not planar.

Theorem. G simple planar graph with $n \geq 3$.
1. $m \leq 3n - 6$
2. $f \leq 2n - 4$
3. There is a vertex of degree at most five

Proof.

K_5: $m = \binom{5}{2} = \frac{5\cdot4}{1\cdot2} = 10 > 9 = 3 \cdot 5 - 6 = 3n - 6$ ✓

$K_{3,3}$: $m = 3 \cdot 3 = 9 < 12 = 3 \cdot 6 - 6 = 3n - 6$

$\Rightarrow$ *no* contradiction to the theorem!

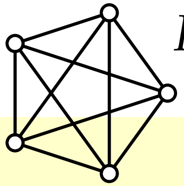
There is no cycle of length 3.

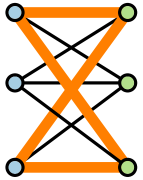
Every face incident to ≥ 4 edges (in hypothetical planar drawing)

$\Rightarrow 4f \leq 2m$

$\Rightarrow 8 \leq 4c + 4 \leq 4f - 4m + 4n \leq 2m - 4m + 4n$

Complete graphs





$K_n = \left(V, \binom{V}{2}\right)$ is the **complete** graph on n vertices.
$K_{n_1,n_2} = (V_1 \cup V_2, V_1 \times V_2)$ with $|V_1| = n_1$ and $|V_2| = n_2$ is a **complete bipartite** graph on $n = n_1 + n_2$ vertices.
A **bipartite** graph is a subgraph of a K_{n_1,n_2}; V_1 and V_2 are called **bipartitions**.

Theorem. K_5 and $K_{3,3}$ are not planar.

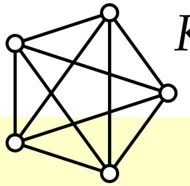
Theorem. G simple planar graph with $n \geq 3$.
1. $m \leq 3n - 6$ 2. $f \leq 2n - 4$
3. There is a vertex of degree at most five

Proof.

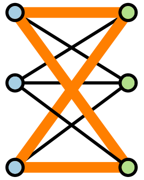
K_5: $m = \binom{5}{2} = \frac{5 \cdot 4}{1 \cdot 2} = 10 > 9 = 3 \cdot 5 - 6 = 3n - 6$ ✓

$K_{3,3}$: $m = 3 \cdot 3 = 9 < 12 = 3 \cdot 6 - 6 = 3n - 6$
$\Rightarrow$ *no* contradiction to the theorem!
There is no cycle of length 3.
Every face incident to ≥ 4 edges (in hypothetical planar drawing)
$\Rightarrow 4f \leq 2m$
$\Rightarrow 8 \leq 4c + 4 \leq 4f - 4m + 4n \leq 2m - 4m + 4n = 4n - 2m$

Complete graphs



$K_n = \left(V, \binom{V}{2}\right)$ is the **complete** graph on n vertices.



$K_{n_1,n_2} = (V_1 \cup V_2, V_1 \times V_2)$ with $|V_1| = n_1$ and $|V_2| = n_2$ is a **complete bipartite** graph on $n = n_1 + n_2$ vertices.

A **bipartite** graph is a subgraph of a K_{n_1,n_2}; V_1 and V_2 are called **bipartitions**.

Theorem. K_5 and $K_{3,3}$ are not planar.

Theorem. G simple planar graph with $n \geq 3$.
1. $m \leq 3n - 6$
2. $f \leq 2n - 4$
3. There is a vertex of degree at most five

Proof.

K_5: $m = \binom{5}{2} = \frac{5 \cdot 4}{1 \cdot 2} = 10 > 9 = 3 \cdot 5 - 6 = 3n - 6$ ✓

$K_{3,3}$: $m = 3 \cdot 3 = 9 < 12 = 3 \cdot 6 - 6 = 3n - 6$

$\Rightarrow$ *no* contradiction to the theorem!

There is no cycle of length 3.

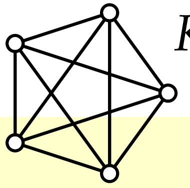
Every face incident to ≥ 4 edges (in hypothetical planar drawing)

$\Rightarrow 4f \leq 2m$

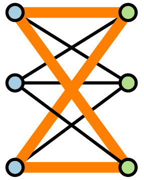
$\Rightarrow 8 \leq 4c + 4 \leq 4f - 4m + 4n \leq 2m - 4m + 4n = 4n - 2m$

$\Rightarrow m \leq 2n - 4$

Complete graphs



$K_n = \left(V, \binom{V}{2}\right)$ is the **complete** graph on n vertices.



$K_{n_1,n_2} = (V_1 \cup V_2, V_1 \times V_2)$ with $|V_1| = n_1$ and $|V_2| = n_2$ is a **complete bipartite** graph on $n = n_1 + n_2$ vertices.
A **bipartite** graph is a subgraph of a K_{n_1,n_2}; V_1 and V_2 are called **bipartitions**.

Theorem. K_5 and $K_{3,3}$ are not planar.

Theorem. G simple planar graph with $n \geq 3$.
1. $m \leq 3n - 6$
2. $f \leq 2n - 4$
3. There is a vertex of degree at most five

Proof.

K_5: $m = \binom{5}{2} = \frac{5\cdot 4}{1\cdot 2} = 10 > 9 = 3 \cdot 5 - 6 = 3n - 6$ ✓

$K_{3,3}$: $m = 3 \cdot 3 = 9 < 12 = 3 \cdot 6 - 6 = 3n - 6$

$\Rightarrow$ *no* contradiction to the theorem!

There is no cycle of length 3.

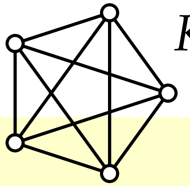
Every face incident to ≥ 4 edges (in hypothetical planar drawing)

$\Rightarrow 4f \leq 2m$

$\Rightarrow 8 \leq 4c + 4 \leq 4f - 4m + 4n \leq 2m - 4m + 4n = 4n - 2m$

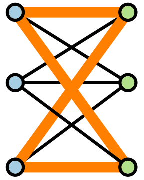
$\Rightarrow m \leq 2n - 4 = 2 \cdot 6 - 4$

Complete graphs


K_5

$K_n = \left(V, \binom{V}{2}\right)$ is the **complete** graph on n vertices.

V_1 V_2


$K_{3,3}$

$K_{n_1,n_2} = (V_1 \cup V_2, V_1 \times V_2)$ with $|V_1| = n_1$ and $|V_2| = n_2$ is a **complete bipartite** graph on $n = n_1 + n_2$ vertices.

A **bipartite** graph is a subgraph of a K_{n_1,n_2}; V_1 and V_2 are called **bipartitions**.

Theorem. K_5 and $K_{3,3}$ are not planar.

Theorem. G simple planar graph with $n \geq 3$.
1. $m \leq 3n - 6$
2. $f \leq 2n - 4$
3. There is a vertex of degree at most five

Proof.

K_5: $m = \binom{5}{2} = \frac{5\cdot 4}{1\cdot 2} = 10 > 9 = 3 \cdot 5 - 6 = 3n - 6$ ✓

$K_{3,3}$: $m = 3 \cdot 3 = 9 < 12 = 3 \cdot 6 - 6 = 3n - 6$

$\Rightarrow$ *no* contradiction to the theorem!

There is no cycle of length 3.

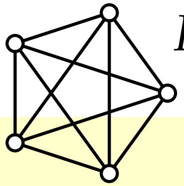
Every face incident to ≥ 4 edges (in hypothetical planar drawing)

$\Rightarrow 4f \leq 2m$

$\Rightarrow 8 \leq 4c + 4 \leq 4f - 4m + 4n \leq 2m - 4m + 4n = 4n - 2m$

$\Rightarrow m \leq 2n - 4 = 2 \cdot 6 - 4 = 8$

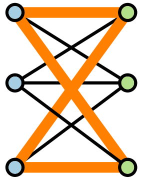
Complete graphs



$K_n = \left(V, \binom{V}{2}\right)$ is the **complete** graph on n vertices.

$K_{n_1,n_2} = (V_1 \cup V_2, V_1 \times V_2)$ with $|V_1| = n_1$ and $|V_2| = n_2$ is a **complete bipartite** graph on $n = n_1 + n_2$ vertices.

A **bipartite** graph is a subgraph of a K_{n_1,n_2}; V_1 and V_2 are called **bipartitions**.



Theorem. K_5 and $K_{3,3}$ are not planar.

Theorem. G simple planar graph with $n \geq 3$.
1. $m \leq 3n - 6$
2. $f \leq 2n - 4$
3. There is a vertex of degree at most five

Proof.

K_5: $m = \binom{5}{2} = \frac{5\cdot 4}{1\cdot 2} = 10 > 9 = 3 \cdot 5 - 6 = 3n - 6$ ✓

$K_{3,3}$: $m = 3 \cdot 3 = 9 < 12 = 3 \cdot 6 - 6 = 3n - 6$

$\Rightarrow$ *no* contradiction to the theorem!

There is no cycle of length 3.

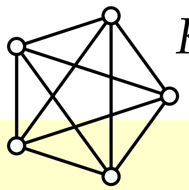
Every face incident to ≥ 4 edges (in hypothetical planar drawing)

$\Rightarrow 4f \leq 2m$

$\Rightarrow 8 \leq 4c + 4 \leq 4f - 4m + 4n \leq 2m - 4m + 4n = 4n - 2m$

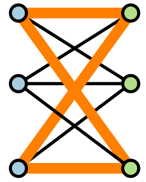
$\Rightarrow m \leq 2n - 4 = 2 \cdot 6 - 4 = 8 < 9 = m$

Complete graphs


K_5

$K_n = \left(V, \binom{V}{2}\right)$ is the **complete** graph on n vertices.

$K_{n_1,n_2} = (V_1 \cup V_2, V_1 \times V_2)$ with $|V_1| = n_1$ and $|V_2| = n_2$ is a **complete bipartite** graph on $n = n_1 + n_2$ vertices.


V_1 V_2 $K_{3,3}$

A **bipartite** graph is a subgraph of a K_{n_1,n_2}; V_1 and V_2 are called **bipartitions**.

Theorem. K_5 and $K_{3,3}$ are not planar.

Theorem. G simple planar graph with $n \geq 3$.
1. $m \leq 3n - 6$
2. $f \leq 2n - 4$
3. There is a vertex of degree at most five

Proof.

K_5: $m = \binom{5}{2} = \frac{5 \cdot 4}{1 \cdot 2} = 10 > 9 = 3 \cdot 5 - 6 = 3n - 6$ ✓

$K_{3,3}$: $m = 3 \cdot 3 = 9 < 12 = 3 \cdot 6 - 6 = 3n - 6$

$\Rightarrow$ *no* contradiction to the theorem!

There is no cycle of length 3.

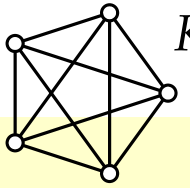
Every face incident to ≥ 4 edges (in hypothetical planar drawing)

$\Rightarrow 4f \leq 2m$

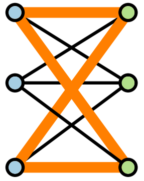
$\Rightarrow 8 \leq 4c + 4 \leq 4f - 4m + 4n \leq 2m - 4m + 4n = 4n - 2m$

$\Rightarrow m \leq 2n - 4 = 2 \cdot 6 - 4 = 8 < 9 = m$

Complete graphs



$K_n = \left(V, \binom{V}{2}\right)$ is the **complete** graph on n vertices.



$K_{n_1,n_2} = (V_1 \cup V_2, V_1 \times V_2)$ with $|V_1| = n_1$ and $|V_2| = n_2$ is a **complete bipartite** graph on $n = n_1 + n_2$ vertices.

A **bipartite** graph is a subgraph of a K_{n_1,n_2}; V_1 and V_2 are called **bipartitions**.

Theorem. K_5 and $K_{3,3}$ are not planar.

Theorem. G simple planar graph with $n \geq 3$.
1. $m \leq 3n - 6$
2. $f \leq 2n - 4$
3. There is a vertex of degree at most five

Proof.

K_5: $m = \binom{5}{2} = \frac{5\cdot4}{1\cdot2} = 10 > 9 = 3 \cdot 5 - 6 = 3n - 6$ ✓

$K_{3,3}$: $m = 3 \cdot 3 = 9 < 12 = 3 \cdot 6 - 6 = 3n - 6$

$\Rightarrow$ *no* contradiction to the theorem!

There is no cycle of length 3.

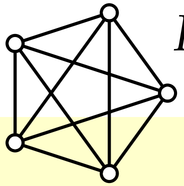
Every face incident to ≥ 4 edges (in hypothetical planar drawing)

$\Rightarrow 4f \leq 2m$

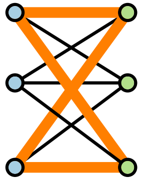
$\Rightarrow 8 \leq 4c + 4 \leq 4f - 4m + 4n \leq 2m - 4m + 4n = 4n - 2m$

$\Rightarrow m \leq 2n - 4 = 2 \cdot 6 - 4 = 8 < 9 = m$ ✓

Complete graphs



$K_n = \left(V, \binom{V}{2}\right)$ is the **complete** graph on n vertices.



$K_{n_1,n_2} = (V_1 \cup V_2, V_1 \times V_2)$ with $|V_1| = n_1$ and $|V_2| = n_2$ is a **complete bipartite** graph on $n = n_1 + n_2$ vertices.

A **bipartite** graph is a subgraph of a K_{n_1,n_2}; V_1 and V_2 are called **bipartitions**.

Theorem. K_5 and $K_{3,3}$ are not planar.

Theorem. G simple planar graph with $n \geq 3$.
1. $m \leq 3n - 6$
2. $f \leq 2n - 4$
3. There is a vertex of degree at most five

Theorem. G simp. pl. **bipartite** graph, $n \geq 3$.
1. $m \leq 2n - 4$
2. $f \leq n - 2$
3. There is a vertex of degree at most three

Proof.

K_5: $m = \binom{5}{2} = \frac{5\cdot 4}{1\cdot 2} = 10 > 9 = 3\cdot 5 - 6 = 3n - 6$ ✓

$K_{3,3}$: $m = 3\cdot 3 = 9 < 12 = 3\cdot 6 - 6 = 3n - 6$

$\Rightarrow$ *no* contradiction to the theorem!

There is no cycle of length 3.

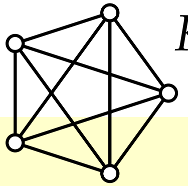
Every face incident to ≥ 4 edges (in hypothetical planar drawing)

$\Rightarrow 4f \leq 2m$

$\Rightarrow 8 \leq 4c + 4 \leq 4f - 4m + 4n \leq 2m - 4m + 4n = 4n - 2m$

$\Rightarrow m \leq 2n - 4 = 2\cdot 6 - 4 = 8 < 9 = m$ ✓

Complete graphs

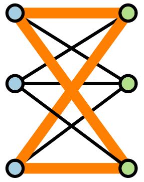


K_5

What about K_4 and $K_{2,3}$?

$K_n = \left(V, \binom{V}{2}\right)$ is the **complete** graph on n vertices.

V_1 V_2



$K_{3,3}$

$K_{n_1,n_2} = (V_1 \cup V_2, V_1 \times V_2)$ with $|V_1| = n_1$ and $|V_2| = n_2$ is a **complete bipartite** graph on $n = n_1 + n_2$ vertices.

A **bipartite** graph is a subgraph of a K_{n_1,n_2}; V_1 and V_2 are called **bipartitions**.

Theorem. K_5 and $K_{3,3}$ are not planar.

Theorem. G simple planar graph with $n \geq 3$.
1. $m \leq 3n - 6$ 2. $f \leq 2n - 4$
3. There is a vertex of degree at most five

Theorem. G simp. pl. **bipartite** graph, $n \geq 3$.
1. $m \leq 2n - 4$ 2. $f \leq n - 2$
3. There is a vertex of degree at most three

Proof.

K_5: $m = \binom{5}{2} = \frac{5 \cdot 4}{1 \cdot 2} = 10 > 9 = 3 \cdot 5 - 6 = 3n - 6$ ✓

$K_{3,3}$: $m = 3 \cdot 3 = 9 < 12 = 3 \cdot 6 - 6 = 3n - 6$

$\Rightarrow$ *no* contradiction to the theorem!

There is no cycle of length 3.

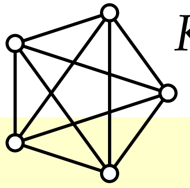
Every face incident to ≥ 4 edges (in hypothetical planar drawing)

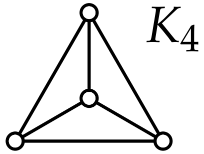
$\Rightarrow 4f \leq 2m$

$\Rightarrow 8 \leq 4c + 4 \leq 4f - 4m + 4n \leq 2m - 4m + 4n = 4n - 2m$

$\Rightarrow m \leq 2n - 4 = 2 \cdot 6 - 4 = 8 < 9 = m$ ✓

Complete graphs



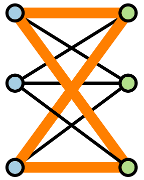


What about K_4 and $K_{2,3}$?

$K_n = \left(V, \binom{V}{2}\right)$ is the **complete** graph on n vertices.

$K_{n_1,n_2} = (V_1 \cup V_2, V_1 \times V_2)$ with $|V_1| = n_1$ and $|V_2| = n_2$ is a **complete bipartite** graph on $n = n_1 + n_2$ vertices.

A **bipartite** graph is a subgraph of a K_{n_1,n_2}; V_1 and V_2 are called **bipartitions**.



Theorem. K_5 and $K_{3,3}$ are not planar.

Theorem. G simple planar graph with $n \geq 3$.
1. $m \leq 3n - 6$
2. $f \leq 2n - 4$
3. There is a vertex of degree at most five

Theorem. G simp. pl. **bipartite** graph, $n \geq 3$.
1. $m \leq 2n - 4$
2. $f \leq n - 2$
3. There is a vertex of degree at most three

Proof.

K_5: $m = \binom{5}{2} = \frac{5 \cdot 4}{1 \cdot 2} = 10 > 9 = 3 \cdot 5 - 6 = 3n - 6$ ✓

$K_{3,3}$: $m = 3 \cdot 3 = 9 < 12 = 3 \cdot 6 - 6 = 3n - 6$

$\Rightarrow$ *no* contradiction to the theorem!

There is no cycle of length 3.

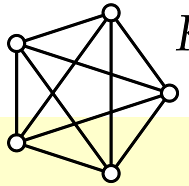
Every face incident to ≥ 4 edges (in hypothetical planar drawing)

$\Rightarrow 4f \leq 2m$

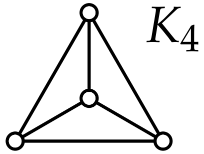
$\Rightarrow 8 \leq 4c + 4 \leq 4f - 4m + 4n \leq 2m - 4m + 4n = 4n - 2m$

$\Rightarrow m \leq 2n - 4 = 2 \cdot 6 - 4 = 8 < 9 = m$ ✓

Complete graphs



K_5



K_4

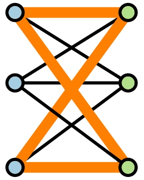
What about K_4 and $K_{2,3}$?

$K_n = \left(V, \binom{V}{2}\right)$ is the **complete** graph on n vertices.

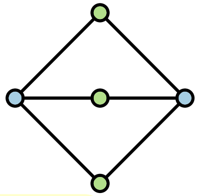
$K_{n_1,n_2} = (V_1 \cup V_2, V_1 \times V_2)$ with $|V_1| = n_1$ and $|V_2| = n_2$ is a **complete bipartite** graph on $n = n_1 + n_2$ vertices.

A **bipartite** graph is a subgraph of a K_{n_1,n_2}; V_1 and V_2 are called **bipartitions**.

V_1 V_2



$K_{3,3}$



$K_{2,3}$

Theorem. K_5 and $K_{3,3}$ are not planar.

Theorem. G simple planar graph with $n \geq 3$.
1. $m \leq 3n - 6$
2. $f \leq 2n - 4$
3. There is a vertex of degree at most five

Theorem. G simp. pl. **bipartite** graph, $n \geq 3$.
1. $m \leq 2n - 4$
2. $f \leq n - 2$
3. There is a vertex of degree at most three

Proof.

K_5: $m = \binom{5}{2} = \frac{5 \cdot 4}{1 \cdot 2} = 10 > 9 = 3 \cdot 5 - 6 = 3n - 6$ ✓

$K_{3,3}$: $m = 3 \cdot 3 = 9 < 12 = 3 \cdot 6 - 6 = 3n - 6$

$\Rightarrow$ *no* contradiction to the theorem!

There is no cycle of length 3.

Every face incident to ≥ 4 edges (in hypothetical planar drawing)

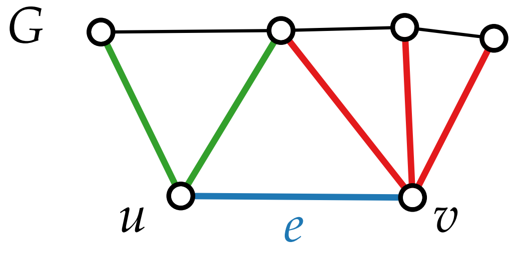
$\Rightarrow 4f \leq 2m$

$\Rightarrow 8 \leq 4c + 4 \leq 4f - 4m + 4n \leq 2m - 4m + 4n = 4n - 2m$

$\Rightarrow m \leq 2n - 4 = 2 \cdot 6 - 4 = 8 < 9 = m$ ✓

Contractions and Minors

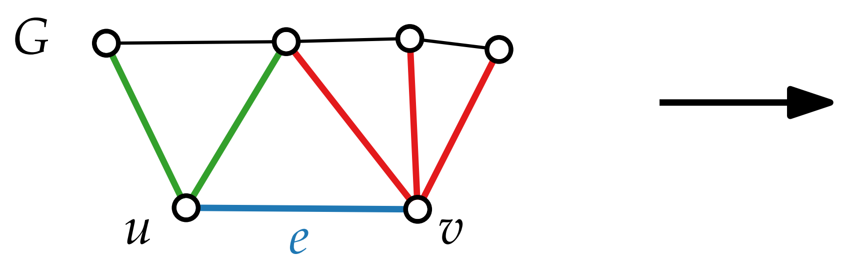
G simple graph and $e = uv \in E$



Contractions and Minors

G simple graph and $e = uv \in E$

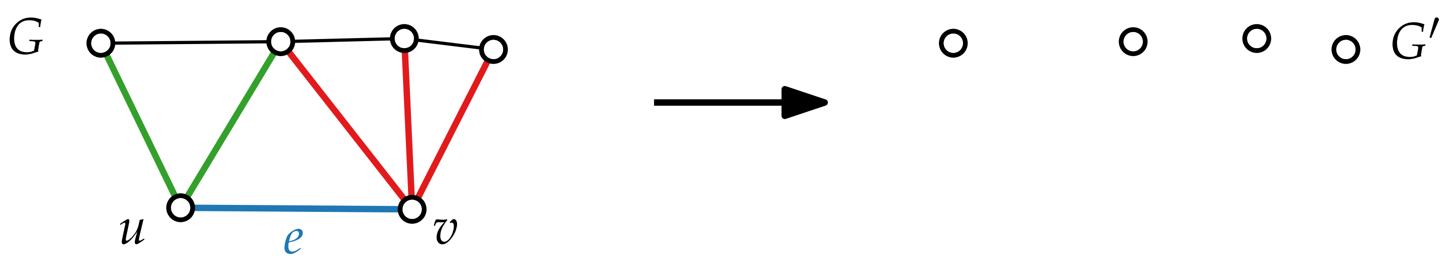
Contracting e gives the graph $G' = (V', E')$



Contractions and Minors

G simple graph and $e = uv \in E$

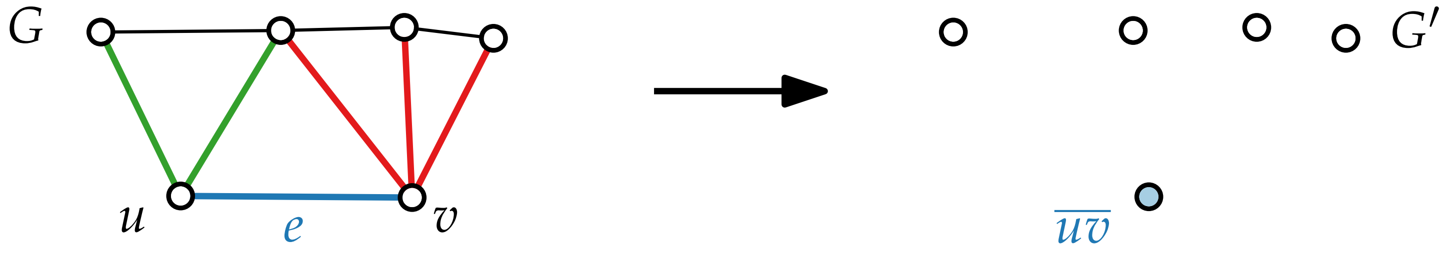
Contracting e gives the graph $G' = (V', E')$
$V' = V \setminus \{u, v\}$



Contractions and Minors

G simple graph and $e = uv \in E$

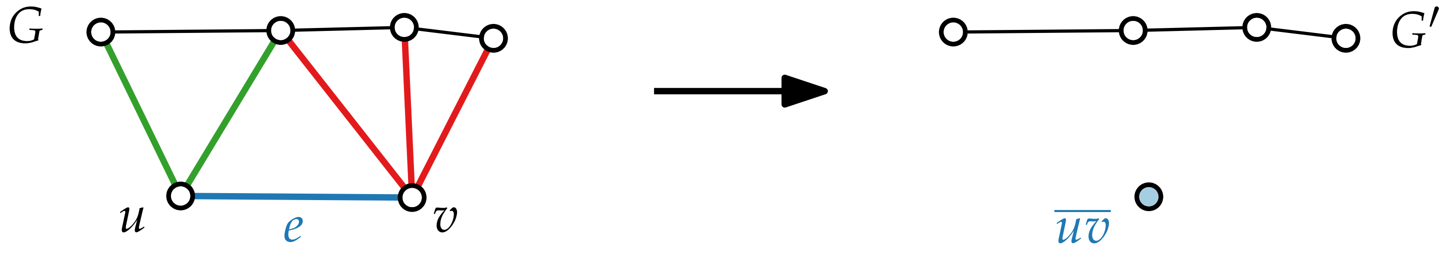
Contracting e gives the graph $G' = (V', E')$
$V' = V \setminus \{u, v\} \cup \overline{uv}$



Contractions and Minors

G simple graph and $e = uv \in E$

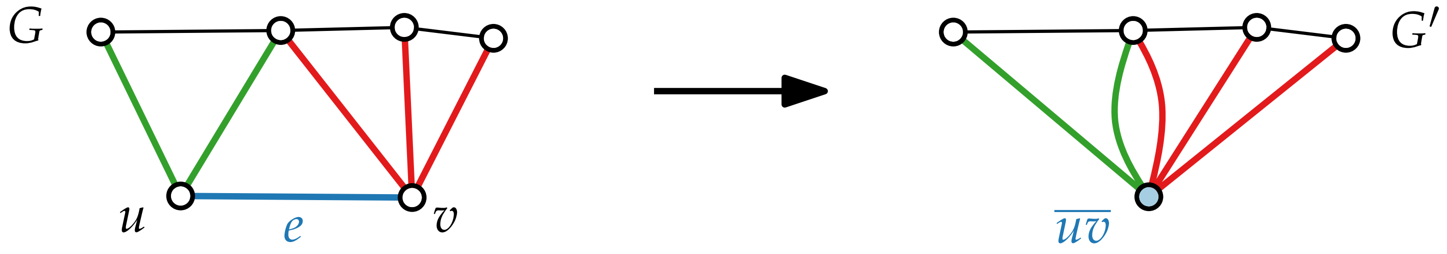
Contracting e gives the graph $G' = (V', E')$
$V' = V \setminus \{u, v\} \cup \overline{uv}$
$E' = E \setminus (\bigcup_{w \in V} \{uw, vw\})$



Contractions and Minors

G simple graph and $e = uv \in E$

Contracting e gives the graph $G' = (V', E')$
$V' = V \setminus \{u, v\} \cup \overline{uv}$
$E' = E \setminus (\bigcup_{w \in V} \{uw, vw\}) \cup \bigcup_{x \in \mathrm{Adj}(u) \cup \mathrm{Adj}(v)} \overline{uv}x$

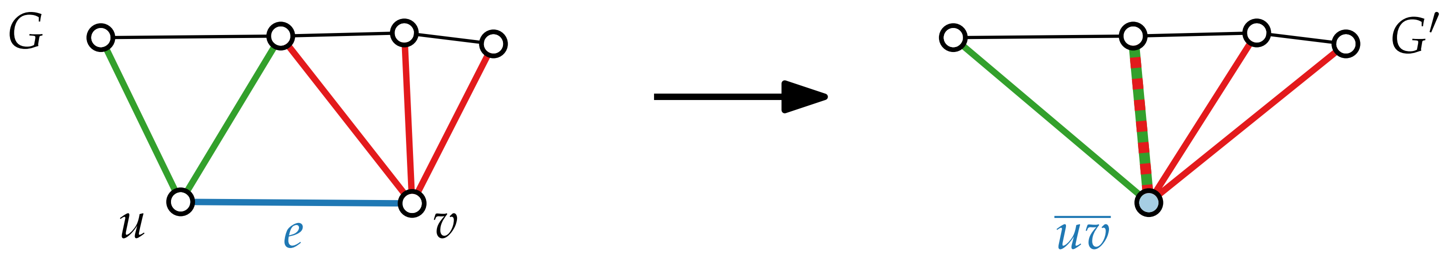


Contractions and Minors

G simple graph and $e = uv \in E$

Contracting e gives the graph $G' = (V', E')$
$V' = V \setminus \{u, v\} \cup \overline{uv}$
$E' = E \setminus (\bigcup_{w \in V} \{uw, vw\}) \cup \bigcup_{x \in \text{Adj}(u) \cup \text{Adj}(v)} \overline{uv}x$

(multi-edges are merged)

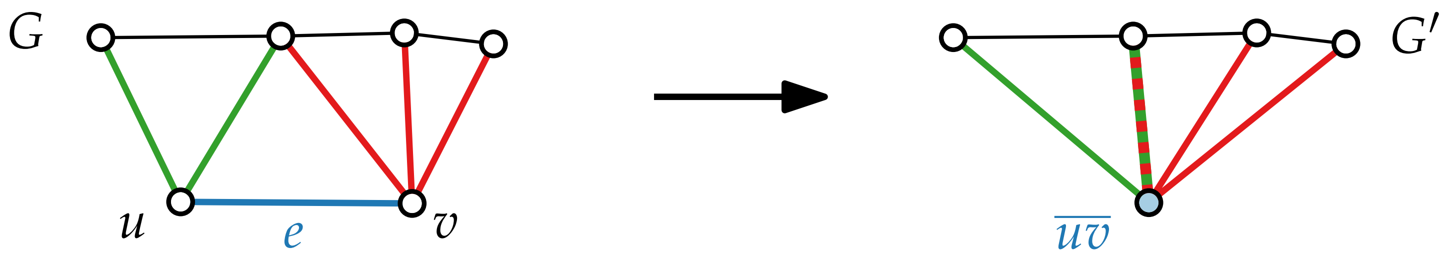


Contractions and Minors

G simple graph and $e = uv \in E$

Contracting e gives the graph $G' = (V', E')$
$V' = V \setminus \{u, v\} \cup \overline{uv}$
$E' = E \setminus (\bigcup_{w \in V} \{uw, vw\}) \cup \bigcup_{x \in \mathrm{Adj}(u) \cup \mathrm{Adj}(v)} \overline{uv}x$

(multi-edges are merged)



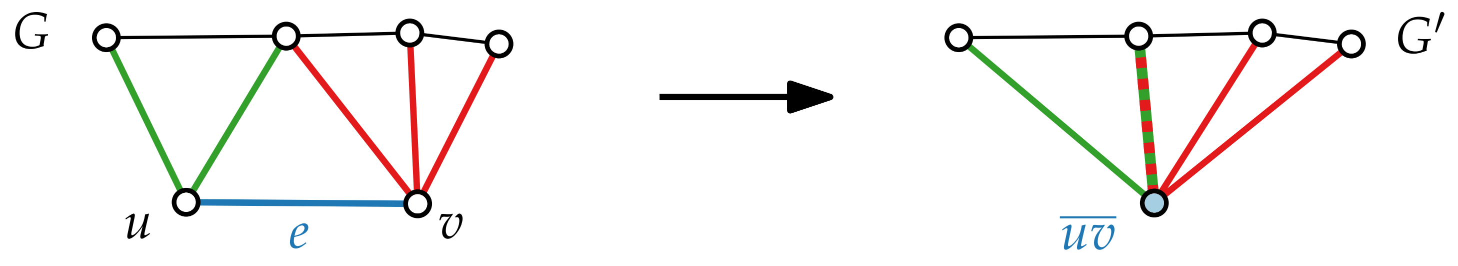
A graph H is a **minor** of G (write $H \leq G$)
if it is obtained by a set of contractions
from a subgraph of G.

Contractions and Minors

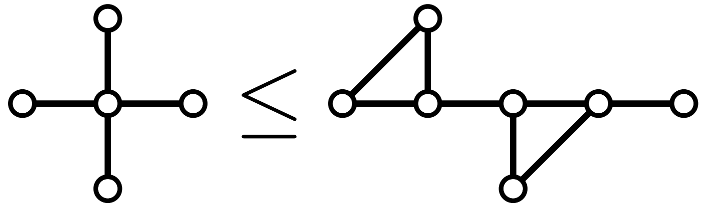
G simple graph and $e = uv \in E$

Contracting e gives the graph $G' = (V', E')$
$V' = V \setminus \{u, v\} \cup \overline{uv}$
$E' = E \setminus (\bigcup_{w \in V} \{uw, vw\}) \cup \bigcup_{x \in \mathrm{Adj}(u) \cup \mathrm{Adj}(v)} \overline{uv}x$

(multi-edges are merged)



A graph H is a **minor** of G (write $H \leq G$) if it is obtained by a set of contractions from a subgraph of G.

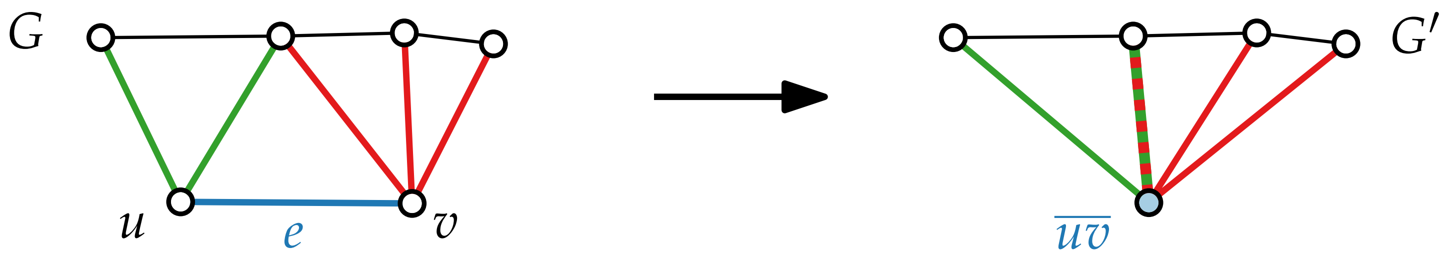


Contractions and Minors

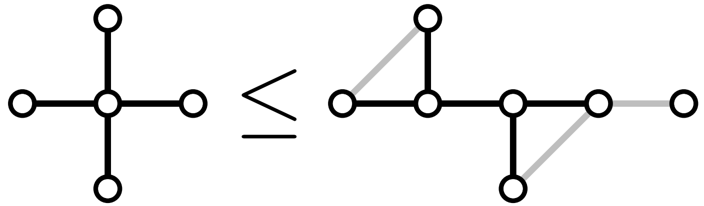
G simple graph and $e = uv \in E$

Contracting e gives the graph $G' = (V', E')$
$V' = V \setminus \{u, v\} \cup \overline{uv}$
$E' = E \setminus (\bigcup_{w \in V} \{uw, vw\}) \cup \bigcup_{x \in \mathrm{Adj}(u) \cup \mathrm{Adj}(v)} \overline{uv}x$

(multi-edges are merged)



A graph H is a **minor** of G (write $H \leq G$) if it is obtained by a set of contractions from a subgraph of G.

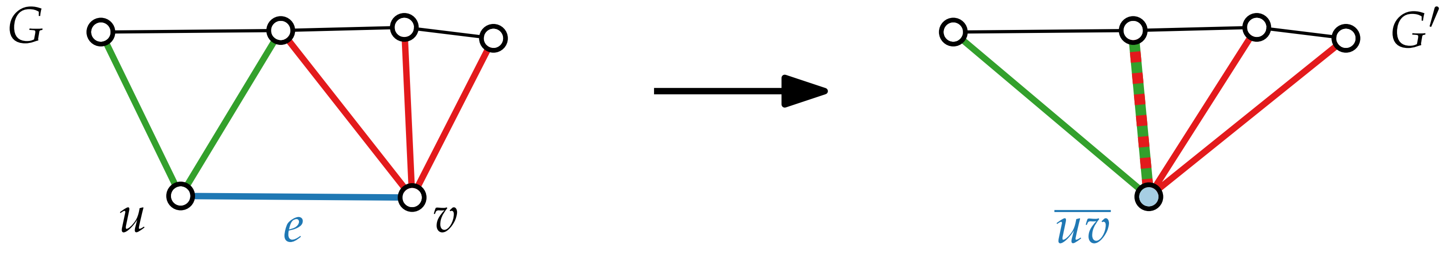


Contractions and Minors

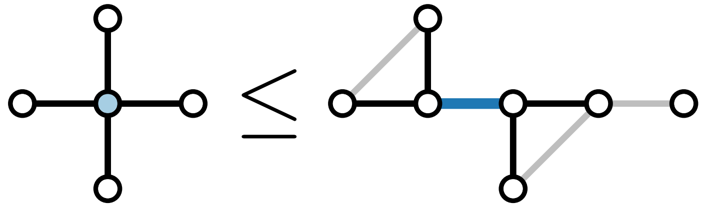
G simple graph and $e = uv \in E$

Contracting e gives the graph $G' = (V', E')$
$V' = V \setminus \{u, v\} \cup \overline{uv}$
$E' = E \setminus (\bigcup_{w \in V} \{uw, vw\}) \cup \bigcup_{x \in \text{Adj}(u) \cup \text{Adj}(v)} \overline{uv}x$

(multi-edges are merged)



A graph H is a **minor** of G (write $H \leq G$) if it is obtained by a set of contractions from a subgraph of G.

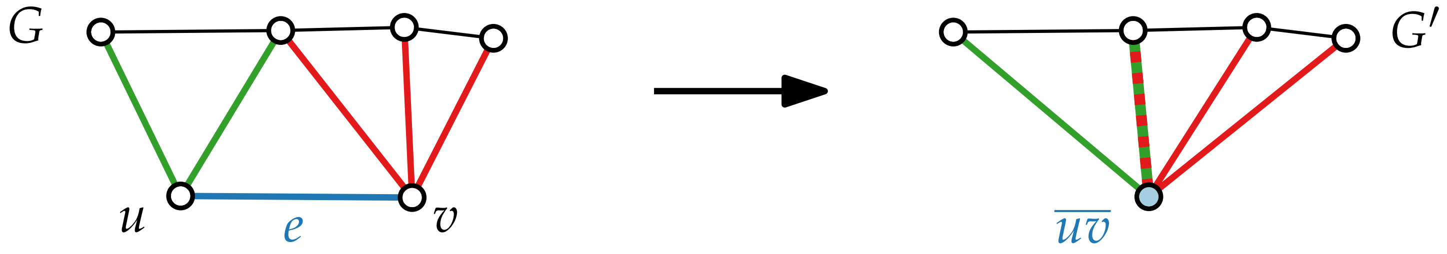


Contractions and Minors

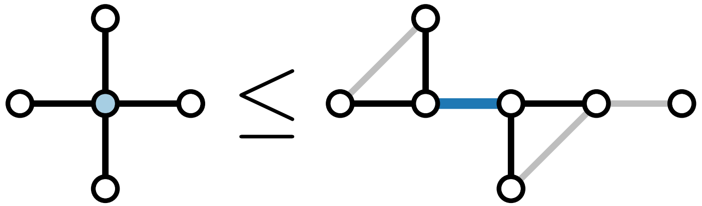
G simple graph and $e = uv \in E$

Contracting e gives the graph $G' = (V', E')$
$V' = V \setminus \{u, v\} \cup \overline{uv}$
$E' = E \setminus (\bigcup_{w \in V} \{uw, vw\}) \cup \bigcup_{x \in \mathrm{Adj}(u) \cup \mathrm{Adj}(v)} \overline{uv}x$

(multi-edges are merged)



A graph H is a **minor** of G (write $H \leq G$) if it is obtained by a set of contractions from a subgraph of G.



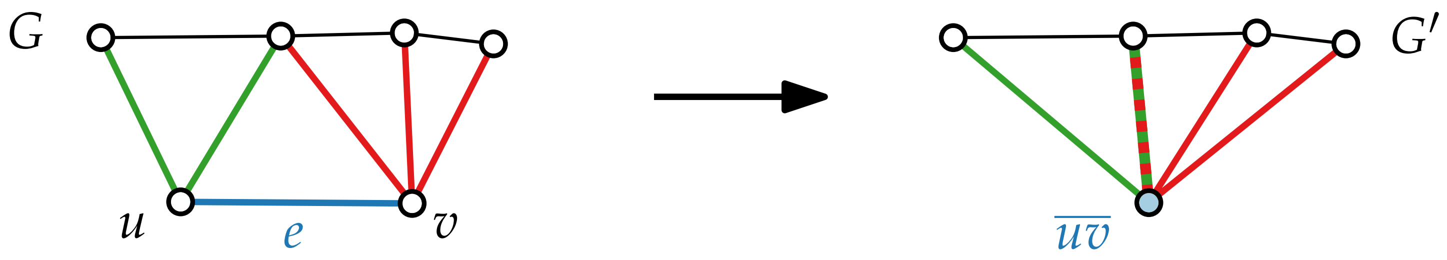
Observation.
G planar, $H \leq G \Rightarrow H$ planar

Contractions and Minors

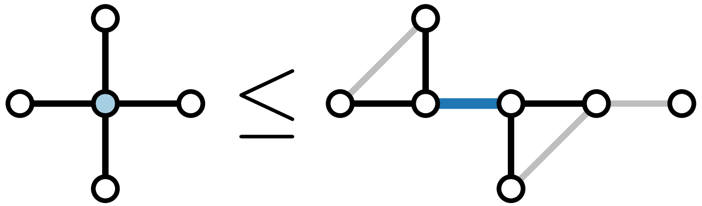
G simple graph and $e = uv \in E$

Contracting e gives the graph $G' = (V', E')$
$V' = V \setminus \{u, v\} \cup \overline{uv}$
$E' = E \setminus (\bigcup_{w \in V} \{uw, vw\}) \cup \bigcup_{x \in \mathrm{Adj}(u) \cup \mathrm{Adj}(v)} \overline{uv}x$

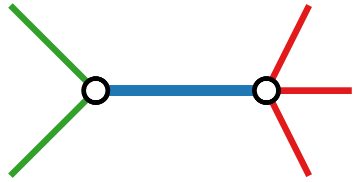
(multi-edges are merged)



A graph H is a **minor** of G (write $H \leq G$)
if it is obtained by a set of contractions
from a subgraph of G.



Observation.
G planar, $H \leq G \Rightarrow H$ planar

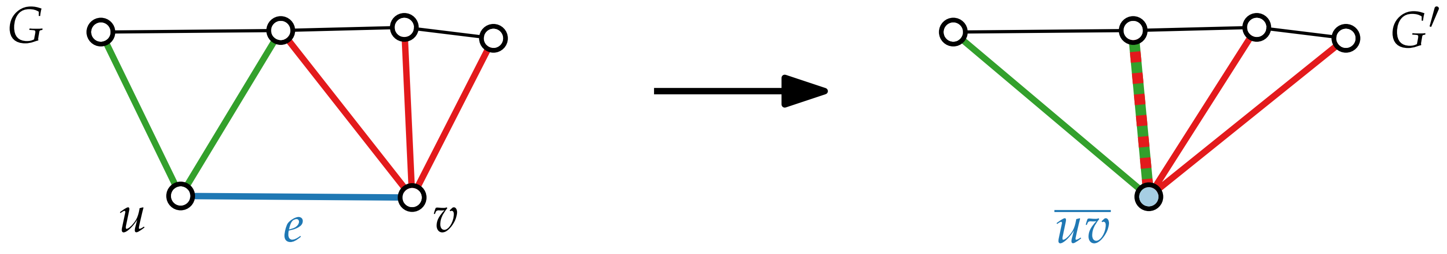


Contractions and Minors

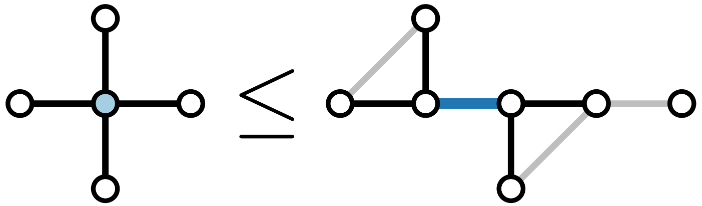
G simple graph and $e = uv \in E$

Contracting e gives the graph $G' = (V', E')$
$V' = V \setminus \{u, v\} \cup \overline{uv}$
$E' = E \setminus (\bigcup_{w \in V} \{uw, vw\}) \cup \bigcup_{x \in \mathrm{Adj}(u) \cup \mathrm{Adj}(v)} \overline{uv}x$

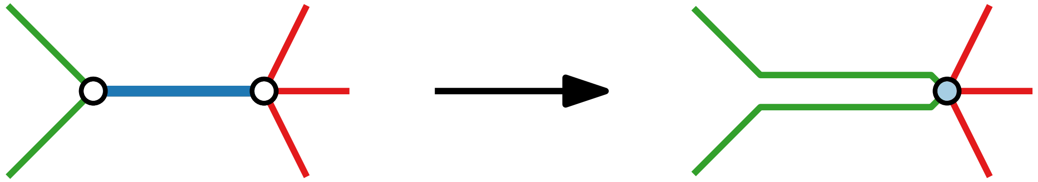
(multi-edges are merged)



A graph H is a **minor** of G (write $H \leq G$) if it is obtained by a set of contractions from a subgraph of G.



Observation.
G planar, $H \leq G \Rightarrow H$ planar

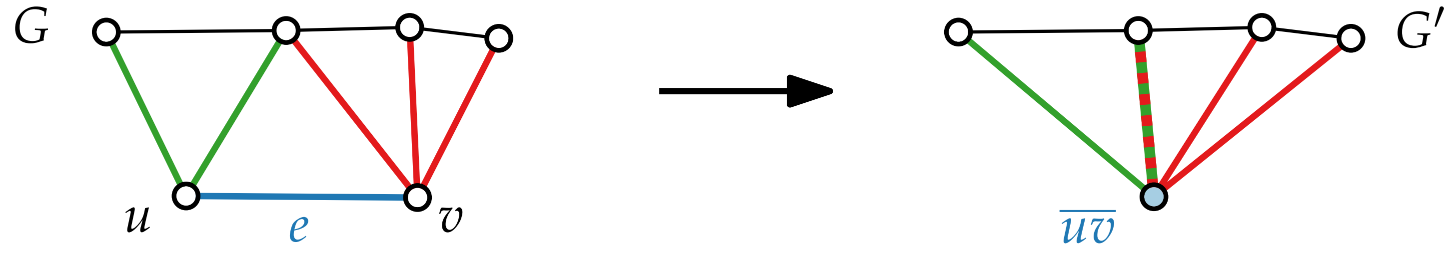


Contractions and Minors

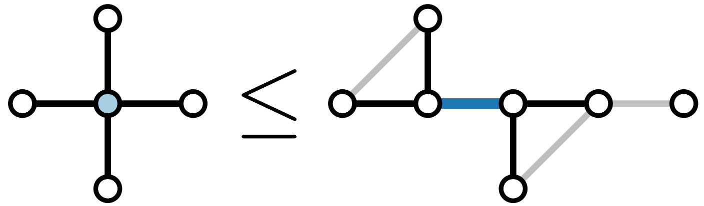
G simple graph and $e = uv \in E$

Contracting e gives the graph $G' = (V', E')$
$V' = V \setminus \{u, v\} \cup \overline{uv}$
$E' = E \setminus (\bigcup_{w \in V} \{uw, vw\}) \cup \bigcup_{x \in \mathrm{Adj}(u) \cup \mathrm{Adj}(v)} \overline{uv}x$

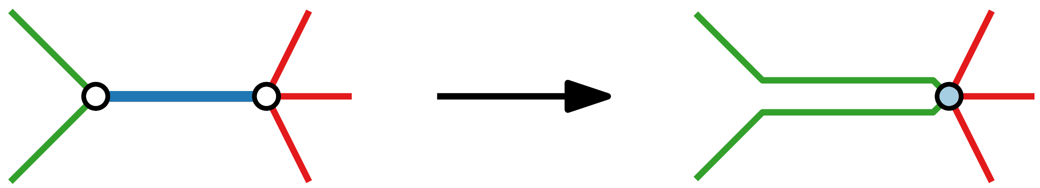
(multi-edges are merged)



A graph H is a **minor** of G (write $H \leq G$)
if it is obtained by a set of contractions
from a subgraph of G.



Observation.
G planar, $H \leq G \Rightarrow H$ planar

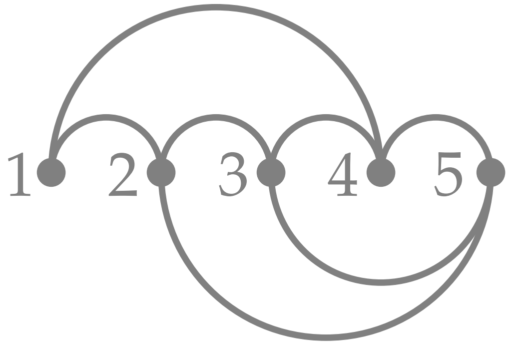


Theorem. [Kuratowski 1930]
G planar $\Leftrightarrow$
neither K_5 nor $K_{3,3}$ minor of G

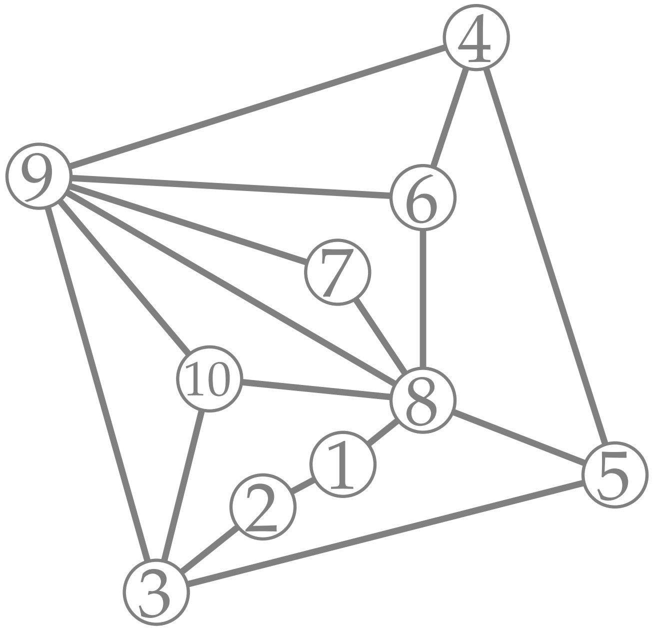
Kazimierz Kuratowski
Warschau 1896–1980 Warschau

Universität Trier

Visualization of Graphs



Lecture 1:
The Graph Visualization Problem

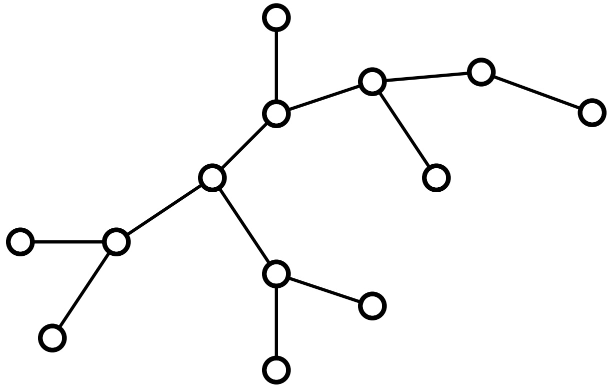


Part V:
Binary Search Trees

Philipp Kindermann
Summer Semester 2021

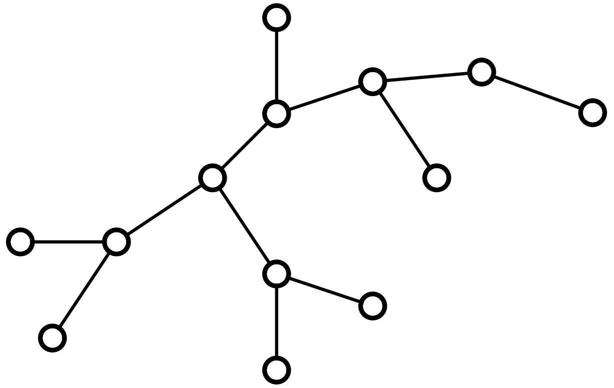
(Rooted) Trees

G is a **tree** if the following equivalent conditions hold:



(Rooted) Trees

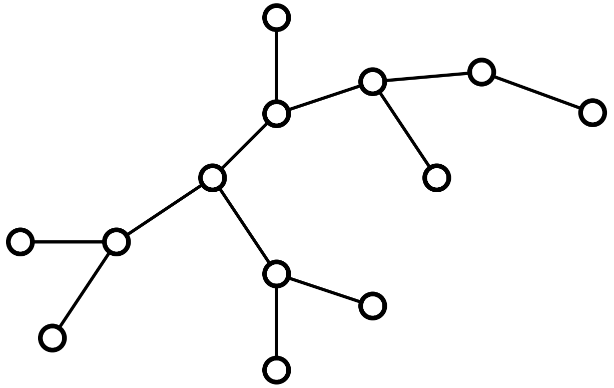
G is a **tree** if the following equivalent conditions hold:
1. there is exactly one v-w-path between any $v, w \in V$



(Rooted) Trees

G is a **tree** if the following equivalent conditions hold:

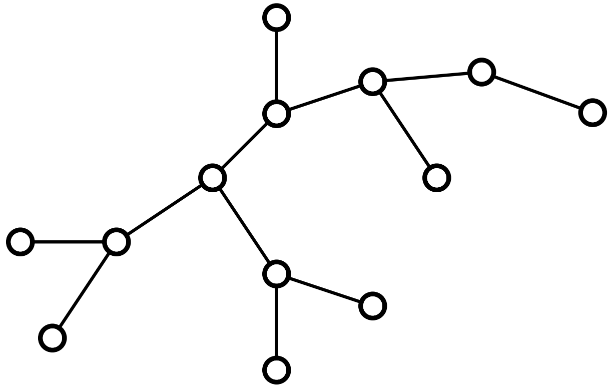
1. there is exactly one v-w-path between any $v, w \in V$
2. G cycle-free and connected



(Rooted) Trees

G is a **tree** if the following equivalent conditions hold:

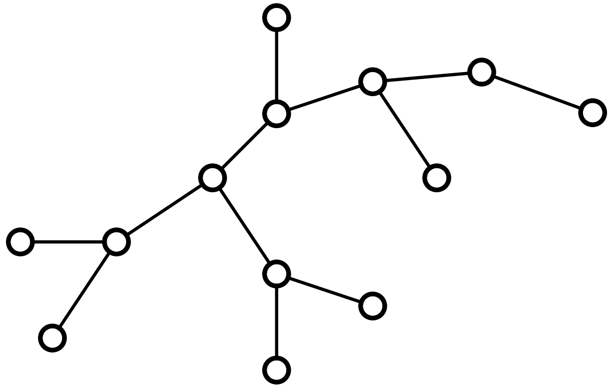
1. there is exactly one v-w-path between any $v, w \in V$
2. G cycle-free and connected
3. G cycle-free and $m = n - 1$



(Rooted) Trees

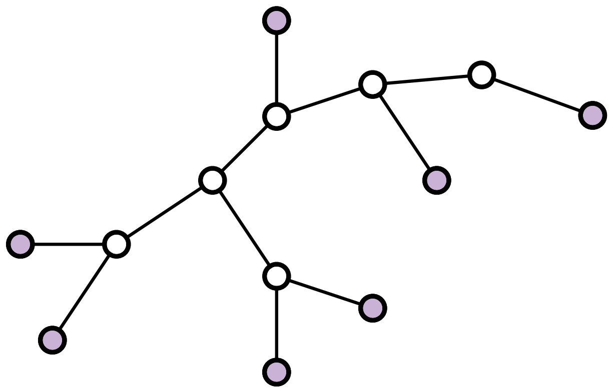
G is a **tree** if the following equivalent conditions hold:

1. there is exactly one v-w-path between any $v, w \in V$
2. G cycle-free and connected
3. G cycle-free and $m = n - 1$
4. G connected and $m = n - 1$



(Rooted) Trees

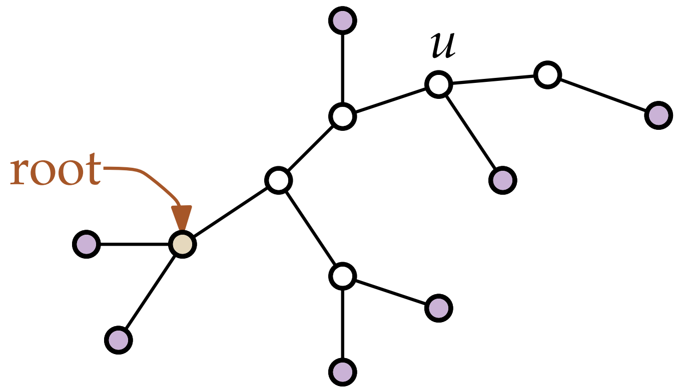
G is a **tree** if the following equivalent conditions hold:
1. there is exactly one v-w-path between any $v, w \in V$
2. G cycle-free and connected
3. G cycle-free and $m = n - 1$
4. G connected and $m = n - 1$



Leaf: Vertex of degree 1

(Rooted) Trees

G is a **tree** if the following equivalent conditions hold:
1. there is exactly one v-w-path between any $v, w \in V$
2. G cycle-free and connected
3. G cycle-free and $m = n - 1$
4. G connected and $m = n - 1$



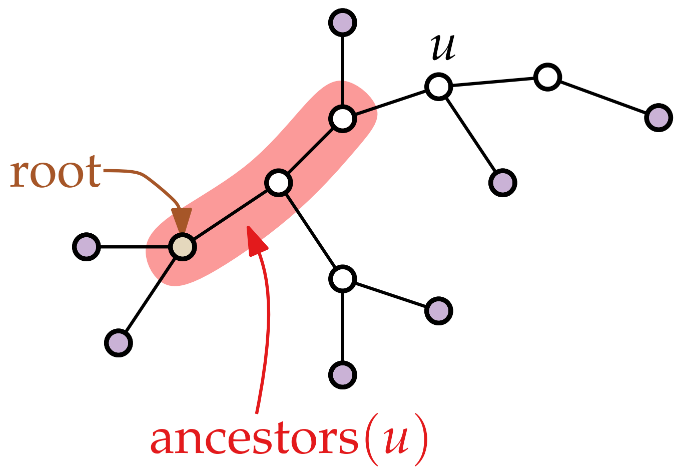
Leaf: Vertex of degree 1

Rooted tree: tree with designated **root**

Parent: Neighbor on path to root

(Rooted) Trees

G is a **tree** if the following equivalent conditions hold:
1. there is exactly one v-w-path between any $v, w \in V$
2. G cycle-free and connected
3. G cycle-free and $m = n - 1$
4. G connected and $m = n - 1$

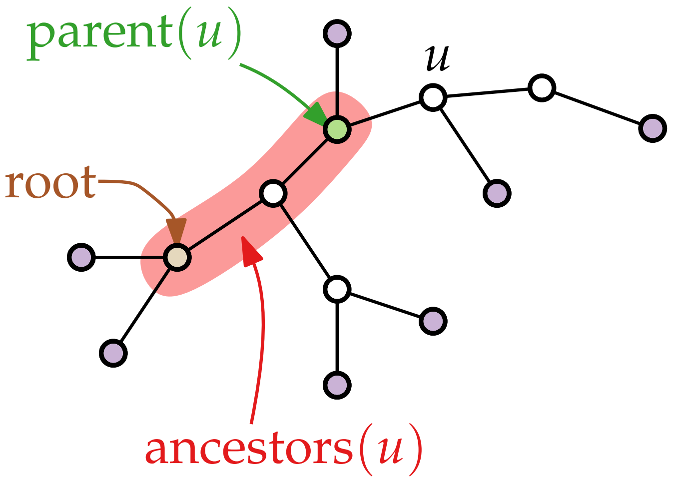


Leaf: Vertex of degree 1
Rooted tree: tree with designated **root**
Ancestor: Vertex on path to root
Parent: Neighbor on path to root

(Rooted) Trees

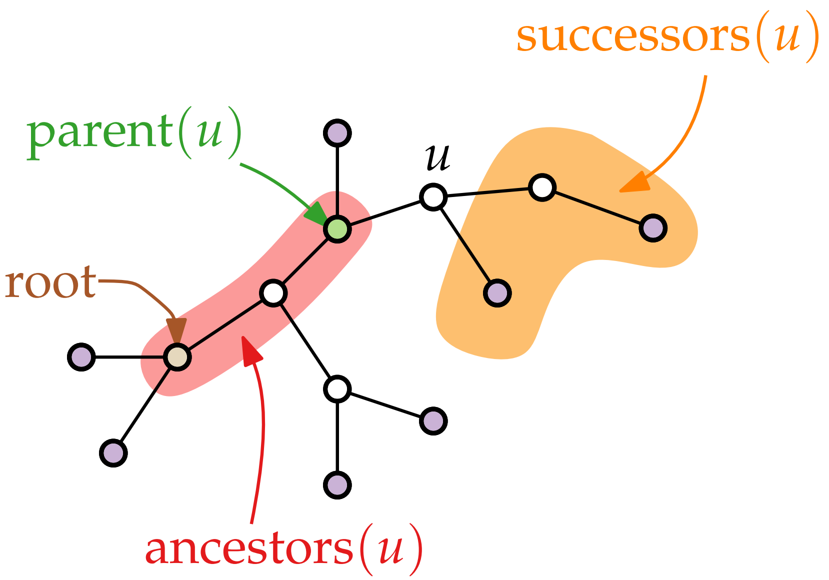
G is a **tree** if the following equivalent conditions hold:
1. there is exactly one v-w-path between any $v, w \in V$
2. G cycle-free and connected
3. G cycle-free and $m = n - 1$
4. G connected and $m = n - 1$

Leaf: Vertex of degree 1
Rooted tree: tree with designated **root**
Ancestor: Vertex on path to root
Parent: Neighbor on path to root



(Rooted) Trees

G is a **tree** if the following equivalent conditions hold:
1. there is exactly one v-w-path between any $v, w \in V$
2. G cycle-free and connected
3. G cycle-free and $m = n - 1$
4. G connected and $m = n - 1$

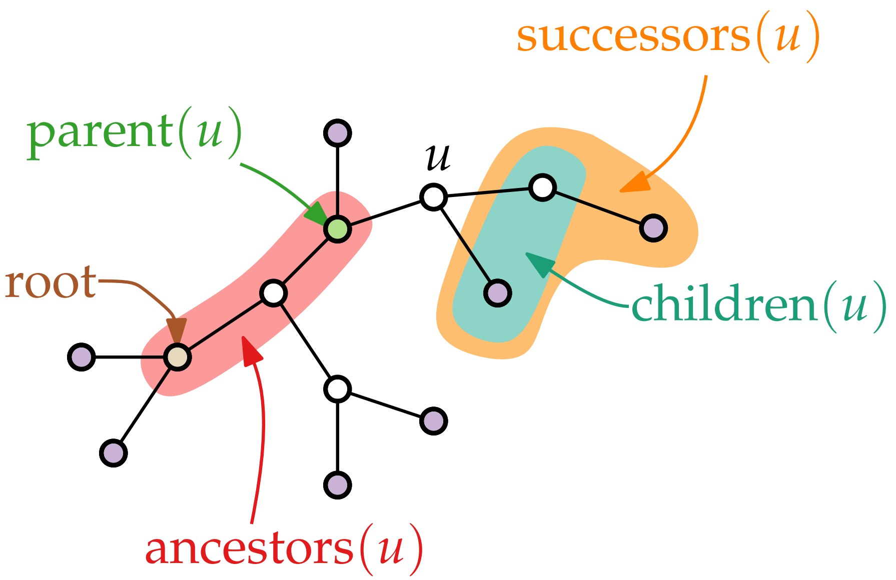


Leaf: Vertex of degree 1
Rooted tree: tree with designated **root**
Ancestor: Vertex on path to root
Parent: Neighbor on path to root
Successor: Vertex not on path to root

(Rooted) Trees

G is a **tree** if the following equivalent conditions hold:
1. there is exactly one v-w-path between any $v, w \in V$
2. G cycle-free and connected
3. G cycle-free and $m = n - 1$
4. G connected and $m = n - 1$

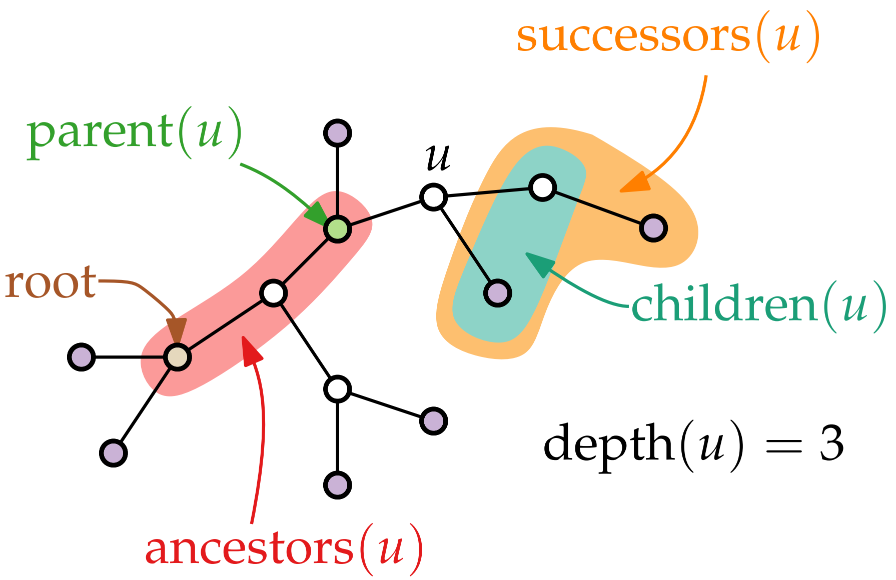
Leaf: Vertex of degree 1
Rooted tree: tree with designated **root**
Ancestor: Vertex on path to root
Parent: Neighbor on path to root
Successor: Vertex not on path to root
Child: Neighbor not on path to root



(Rooted) Trees

G is a **tree** if the following equivalent conditions hold:
1. there is exactly one v-w-path between any $v, w \in V$
2. G cycle-free and connected
3. G cycle-free and $m = n - 1$
4. G connected and $m = n - 1$

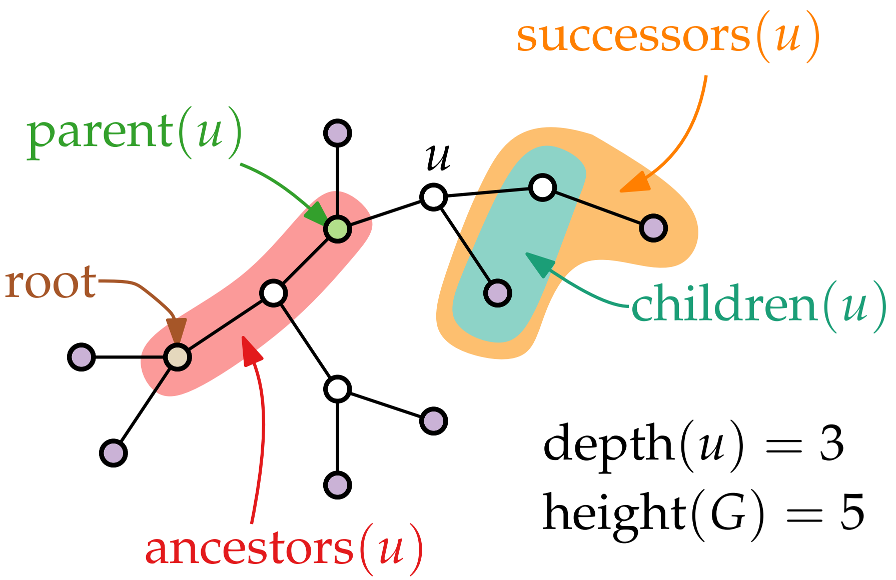
Leaf: Vertex of degree 1
Rooted tree: tree with designated **root**
Ancestor: Vertex on path to root
Parent: Neighbor on path to root
Successor: Vertex not on path to root
Child: Neighbor not on path to root
Depth: Length of path to root



(Rooted) Trees

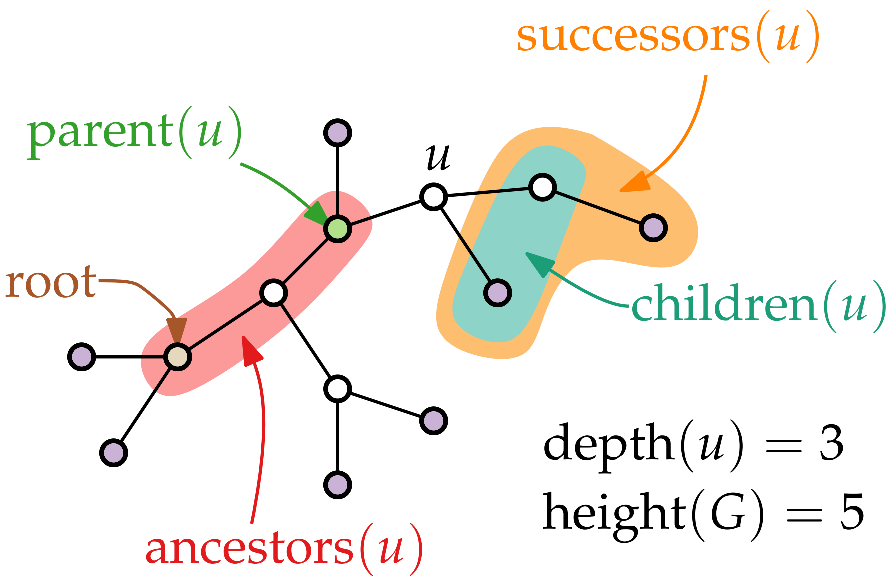
G is a **tree** if the following equivalent conditions hold:
1. there is exactly one v-w-path between any $v, w \in V$
2. G cycle-free and connected
3. G cycle-free and $m = n - 1$
4. G connected and $m = n - 1$

Leaf: Vertex of degree 1
Rooted tree: tree with designated **root**
Ancestor: Vertex on path to root
Parent: Neighbor on path to root
Successor: Vertex not on path to root
Child: Neighbor not on path to root
Depth: Length of path to root
Height: Maximum depth of a leaf



(Rooted) Trees

G is a **tree** if the following equivalent conditions hold:
1. there is exactly one v-w-path between any $v, w \in V$
2. G cycle-free and connected
3. G cycle-free and $m = n - 1$
4. G connected and $m = n - 1$

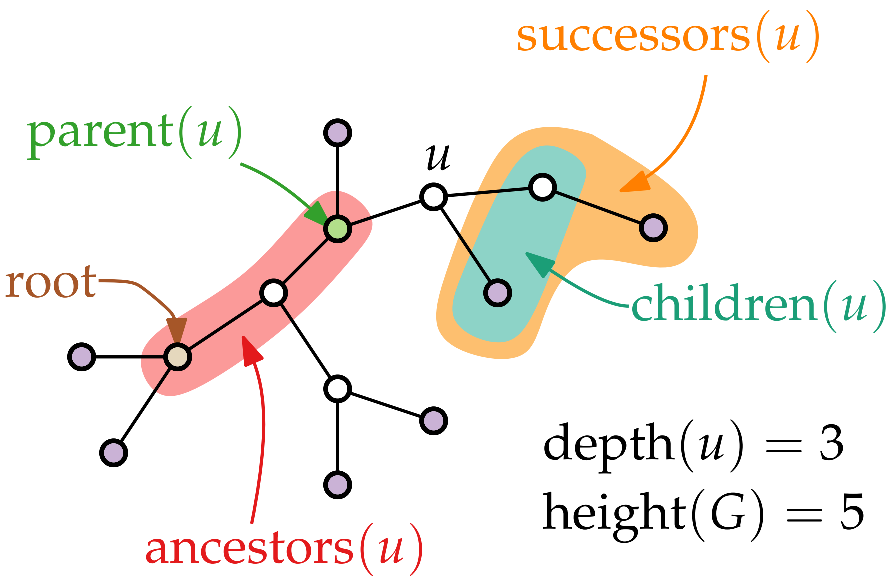


Leaf: Vertex of degree 1
Rooted tree: tree with designated **root**
Ancestor: Vertex on path to root
Parent: Neighbor on path to root
Successor: Vertex not on path to root
Child: Neighbor not on path to root
Depth: Length of path to root
Height: Maximum depth of a leaf

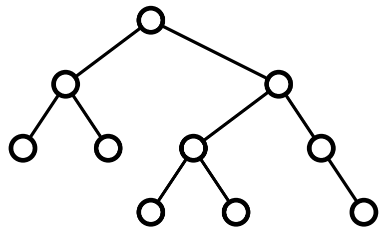
Binary Tree: At most two children per vertex (left / right child)

(Rooted) Trees

G is a **tree** if the following equivalent conditions hold:
1. there is exactly one v-w-path between any $v, w \in V$
2. G cycle-free and connected
3. G cycle-free and $m = n - 1$
4. G connected and $m = n - 1$



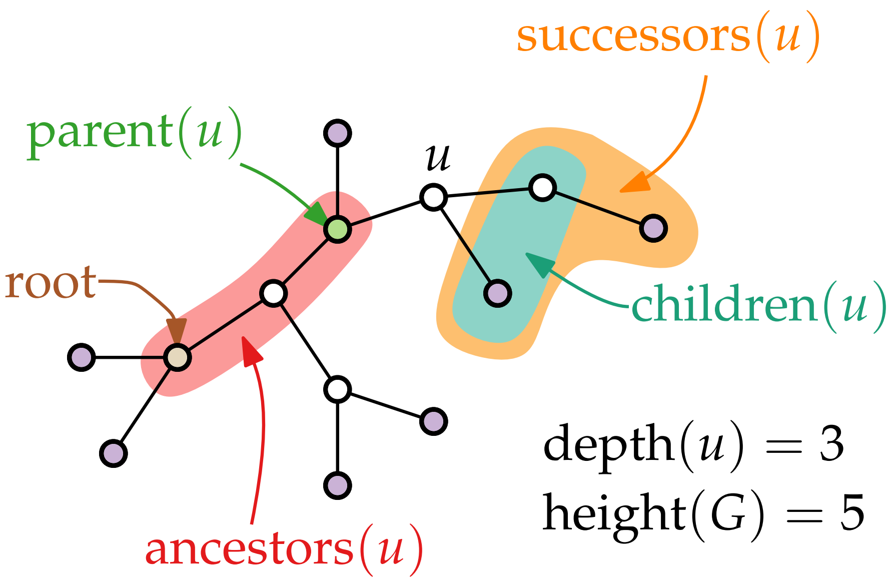
Leaf: Vertex of degree 1
Rooted tree: tree with designated **root**
Ancestor: Vertex on path to root
Parent: Neighbor on path to root
Successor: Vertex not on path to root
Child: Neighbor not on path to root
Depth: Length of path to root
Height: Maximum depth of a leaf



Binary Tree: At most two children per vertex (left / right child)

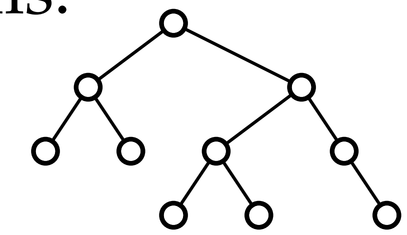
(Rooted) Trees

G is a **tree** if the following equivalent conditions hold:
1. there is exactly one v-w-path between any $v, w \in V$
2. G cycle-free and connected
3. G cycle-free and $m = n - 1$
4. G connected and $m = n - 1$



Leaf: Vertex of degree 1
Rooted tree: tree with designated **root**
Ancestor: Vertex on path to root
Parent: Neighbor on path to root
Successor: Vertex not on path to root
Child: Neighbor not on path to root
Depth: Length of path to root
Height: Maximum depth of a leaf

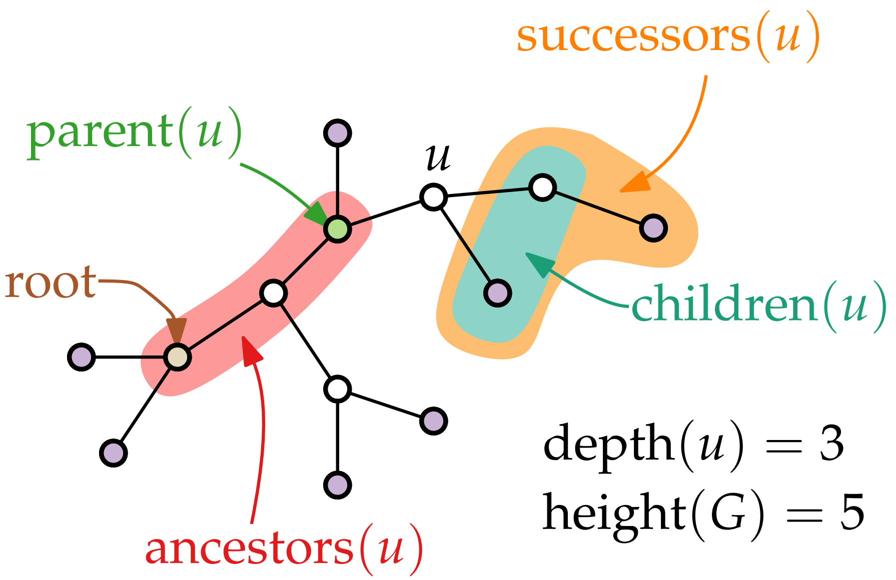
3 traversals:



Binary Tree: At most two children per vertex (left / right child)

(Rooted) Trees

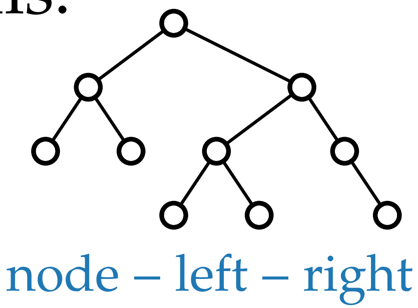
G is a **tree** if the following equivalent conditions hold:
1. there is exactly one v-w-path between any $v, w \in V$
2. G cycle-free and connected
3. G cycle-free and $m = n - 1$
4. G connected and $m = n - 1$



Leaf: Vertex of degree 1
Rooted tree: tree with designated **root**
Ancestor: Vertex on path to root
Parent: Neighbor on path to root
Successor: Vertex not on path to root
Child: Neighbor not on path to root
Depth: Length of path to root
Height: Maximum depth of a leaf

3 traversals:

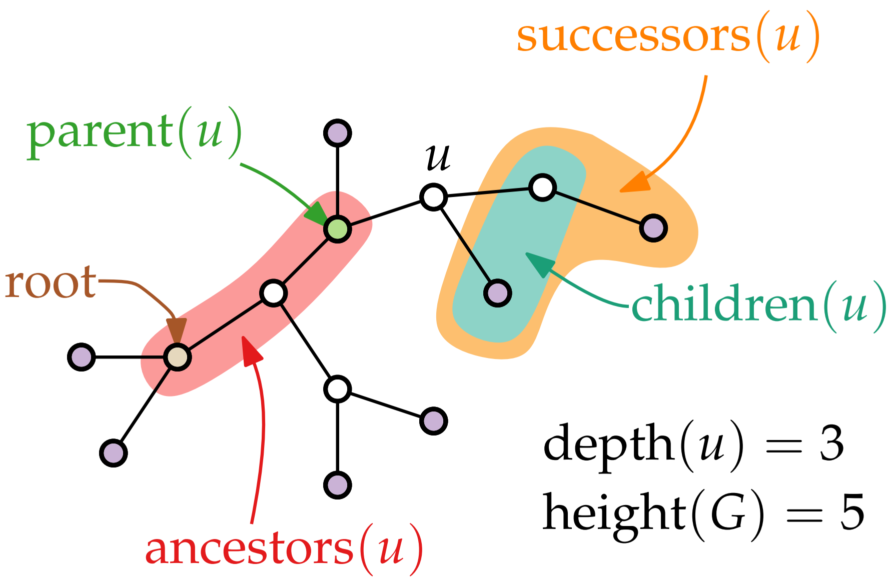
preorder



Binary Tree: At most two children per vertex (left / right child)

(Rooted) Trees

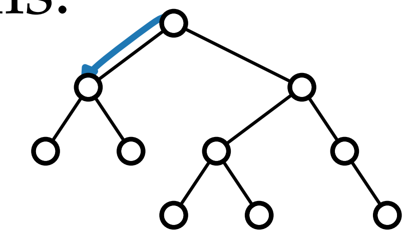
G is a **tree** if the following equivalent conditions hold:
1. there is exactly one v-w-path between any $v, w \in V$
2. G cycle-free and connected
3. G cycle-free and $m = n - 1$
4. G connected and $m = n - 1$



Leaf: Vertex of degree 1
Rooted tree: tree with designated **root**
Ancestor: Vertex on path to root
Parent: Neighbor on path to root
Successor: Vertex not on path to root
Child: Neighbor not on path to root
Depth: Length of path to root
Height: Maximum depth of a leaf

3 traversals:

preorder

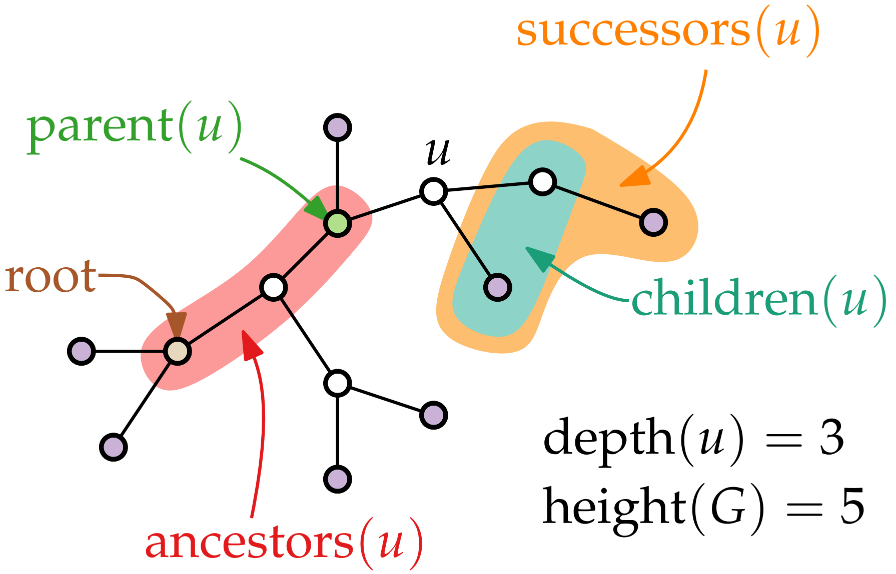


node – left – right

Binary Tree: At most two children per vertex (left / right child)

(Rooted) Trees

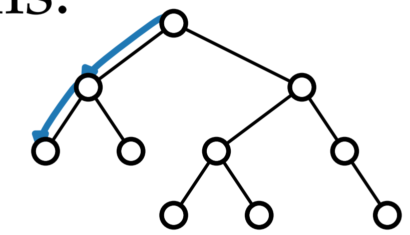
G is a **tree** if the following equivalent conditions hold:
1. there is exactly one v-w-path between any $v, w \in V$
2. G cycle-free and connected
3. G cycle-free and $m = n - 1$
4. G connected and $m = n - 1$



Leaf: Vertex of degree 1
Rooted tree: tree with designated **root**
Ancestor: Vertex on path to root
Parent: Neighbor on path to root
Successor: Vertex not on path to root
Child: Neighbor not on path to root
Depth: Length of path to root
Height: Maximum depth of a leaf

3 traversals:

preorder

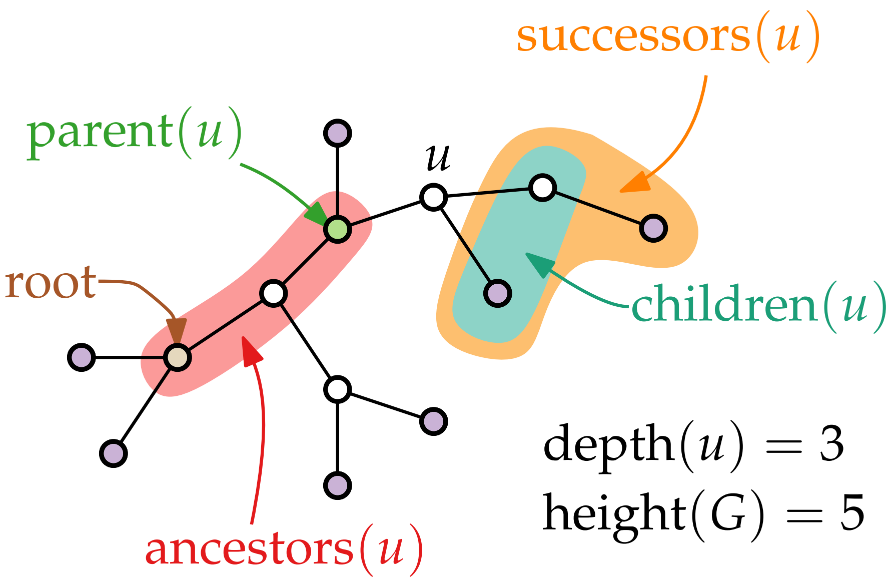


node – left – right

Binary Tree: At most two children per vertex (left / right child)

(Rooted) Trees

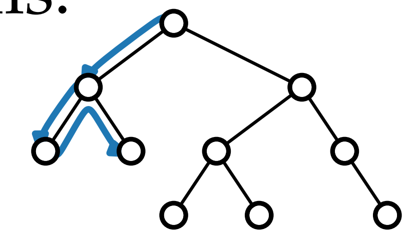
G is a **tree** if the following equivalent conditions hold:
1. there is exactly one v-w-path between any $v, w \in V$
2. G cycle-free and connected
3. G cycle-free and $m = n - 1$
4. G connected and $m = n - 1$



Leaf: Vertex of degree 1
Rooted tree: tree with designated **root**
Ancestor: Vertex on path to root
Parent: Neighbor on path to root
Successor: Vertex not on path to root
Child: Neighbor not on path to root
Depth: Length of path to root
Height: Maximum depth of a leaf

3 traversals:

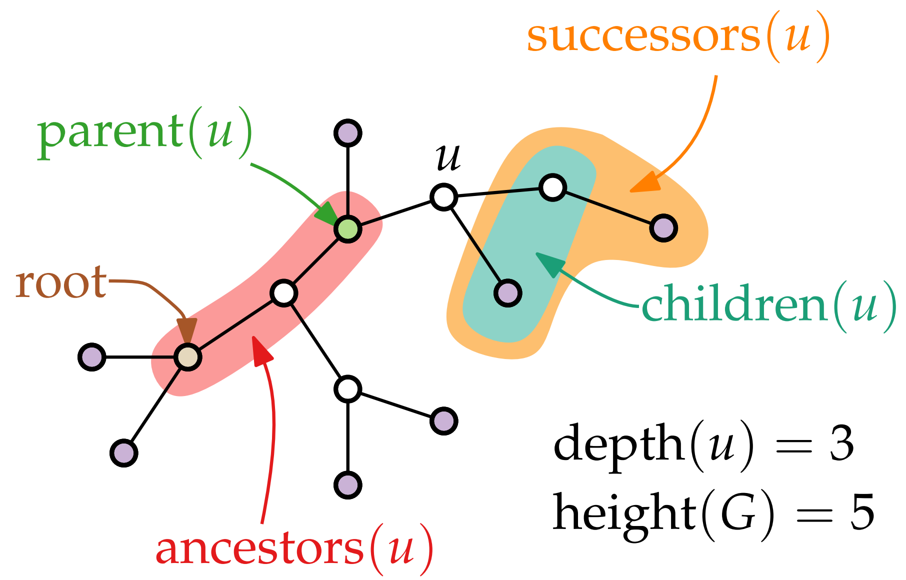
preorder



Binary Tree: At most two children per vertex (left / right child)

(Rooted) Trees

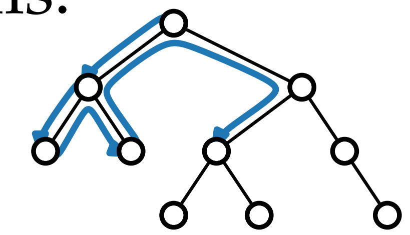
G is a **tree** if the following equivalent conditions hold:
1. there is exactly one v-w-path between any $v, w \in V$
2. G cycle-free and connected
3. G cycle-free and $m = n - 1$
4. G connected and $m = n - 1$



Leaf: Vertex of degree 1
Rooted tree: tree with designated **root**
Ancestor: Vertex on path to root
Parent: Neighbor on path to root
Successor: Vertex not on path to root
Child: Neighbor not on path to root
Depth: Length of path to root
Height: Maximum depth of a leaf

3 traversals:

preorder

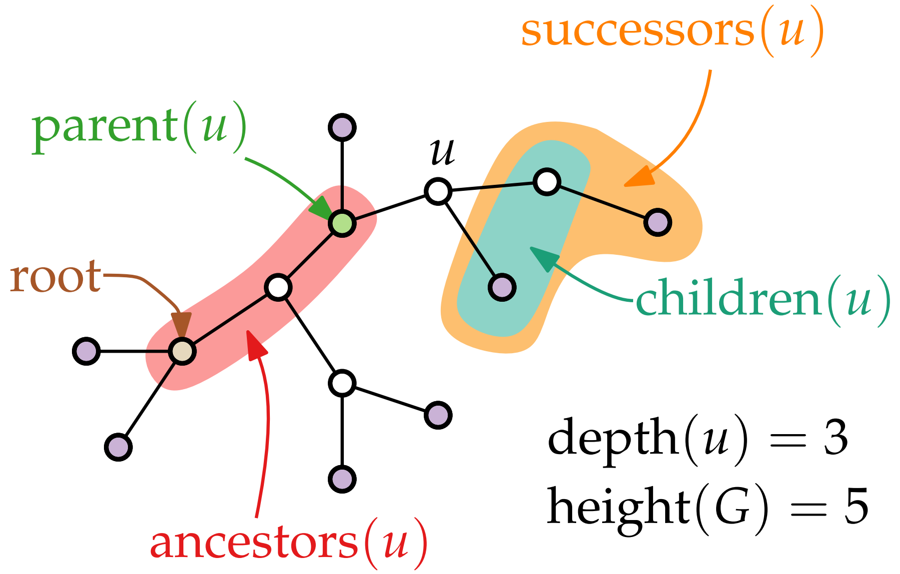


node – left – right

Binary Tree: At most two children per vertex (left / right child)

(Rooted) Trees

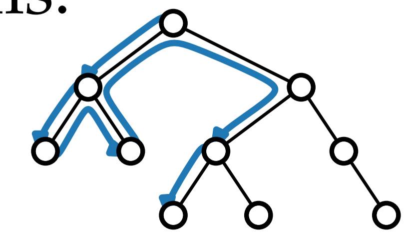
G is a **tree** if the following equivalent conditions hold:
1. there is exactly one v-w-path between any $v, w \in V$
2. G cycle-free and connected
3. G cycle-free and $m = n - 1$
4. G connected and $m = n - 1$



Leaf: Vertex of degree 1
Rooted tree: tree with designated **root**
Ancestor: Vertex on path to root
Parent: Neighbor on path to root
Successor: Vertex not on path to root
Child: Neighbor not on path to root
Depth: Length of path to root
Height: Maximum depth of a leaf

3 traversals:

preorder

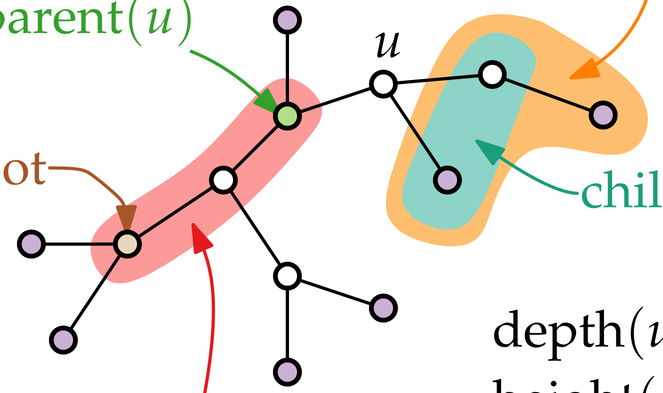


node – left – right

Binary Tree: At most two children per vertex (left / right child)

(Rooted) Trees

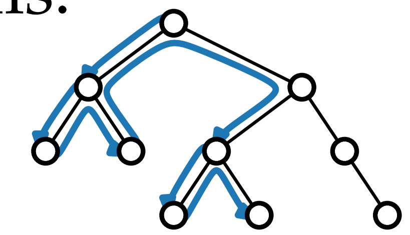
G is a **tree** if the following equivalent conditions hold:
1. there is exactly one v-w-path between any $v, w \in V$
2. G cycle-free and connected
3. G cycle-free and $m = n - 1$
4. G connected and $m = n - 1$



Leaf: Vertex of degree 1
Rooted tree: tree with designated **root**
Ancestor: Vertex on path to root
Parent: Neighbor on path to root
Successor: Vertex not on path to root
Child: Neighbor not on path to root
Depth: Length of path to root
Height: Maximum depth of a leaf

3 traversals:

preorder

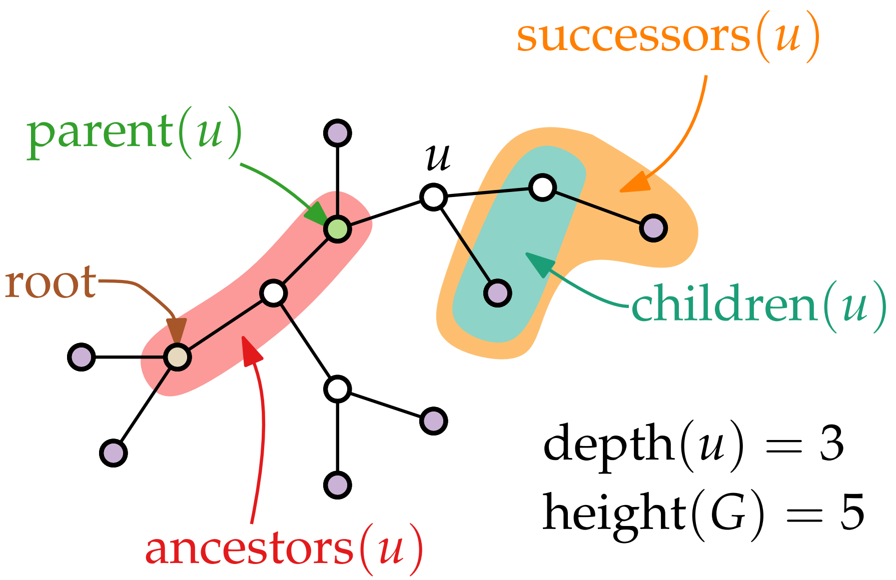


node – left – right

Binary Tree: At most two children per vertex (left / right child)

(Rooted) Trees

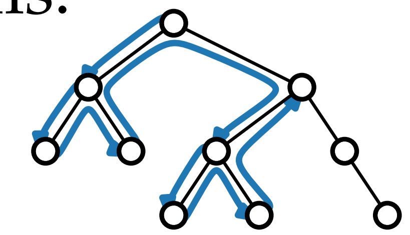
G is a **tree** if the following equivalent conditions hold:
1. there is exactly one v-w-path between any $v, w \in V$
2. G cycle-free and connected
3. G cycle-free and $m = n - 1$
4. G connected and $m = n - 1$



Leaf: Vertex of degree 1
Rooted tree: tree with designated **root**
Ancestor: Vertex on path to root
Parent: Neighbor on path to root
Successor: Vertex not on path to root
Child: Neighbor not on path to root
Depth: Length of path to root
Height: Maximum depth of a leaf

3 traversals:

preorder

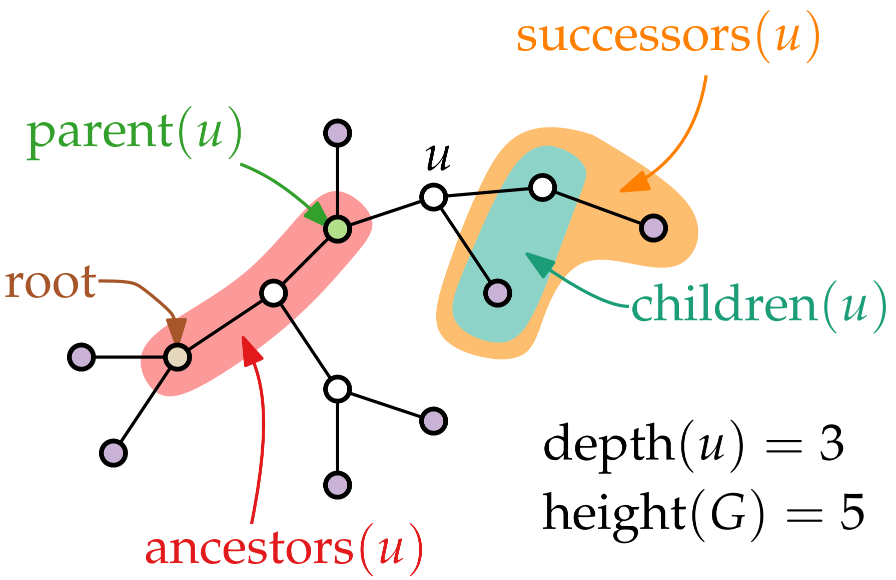


node – left – right

Binary Tree: At most two children per vertex (left / right child)

(Rooted) Trees

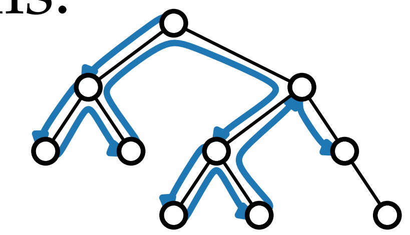
G is a **tree** if the following equivalent conditions hold:
1. there is exactly one v-w-path between any $v, w \in V$
2. G cycle-free and connected
3. G cycle-free and $m = n - 1$
4. G connected and $m = n - 1$



Leaf: Vertex of degree 1
Rooted tree: tree with designated **root**
Ancestor: Vertex on path to root
Parent: Neighbor on path to root
Successor: Vertex not on path to root
Child: Neighbor not on path to root
Depth: Length of path to root
Height: Maximum depth of a leaf

3 traversals:

preorder

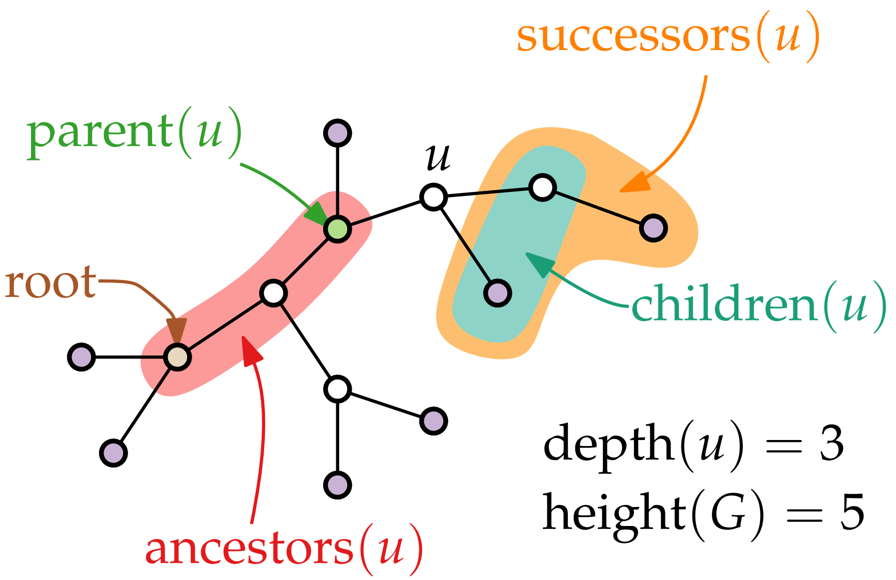


node – left – right

Binary Tree: At most two children per vertex (left / right child)

(Rooted) Trees

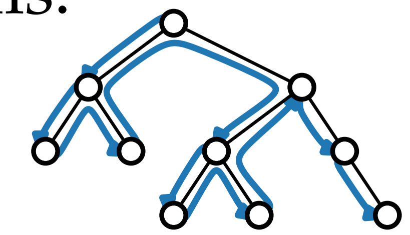
G is a **tree** if the following equivalent conditions hold:
1. there is exactly one v-w-path between any $v, w \in V$
2. G cycle-free and connected
3. G cycle-free and $m = n - 1$
4. G connected and $m = n - 1$



Leaf: Vertex of degree 1
Rooted tree: tree with designated **root**
Ancestor: Vertex on path to root
Parent: Neighbor on path to root
Successor: Vertex not on path to root
Child: Neighbor not on path to root
Depth: Length of path to root
Height: Maximum depth of a leaf

3 traversals:

preorder

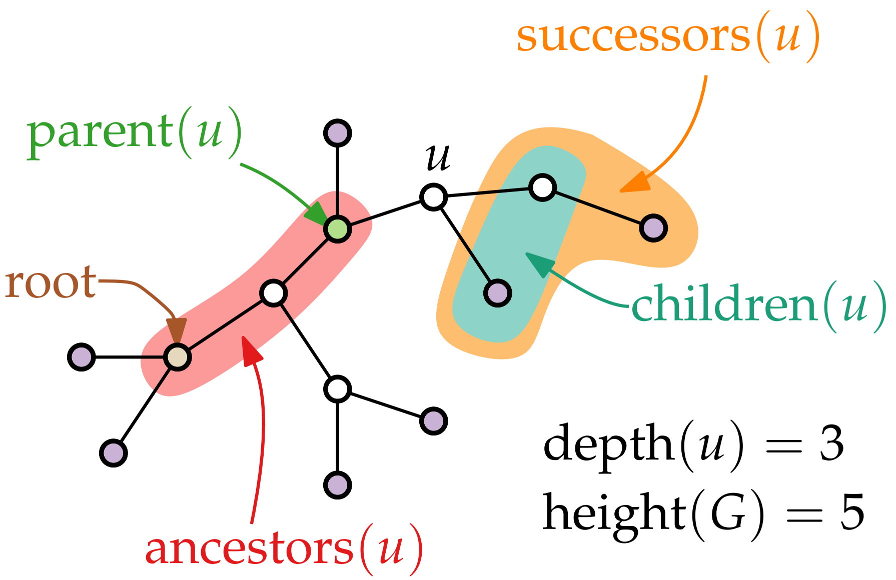


node – left – right

Binary Tree: At most two children per vertex (left / right child)

(Rooted) Trees

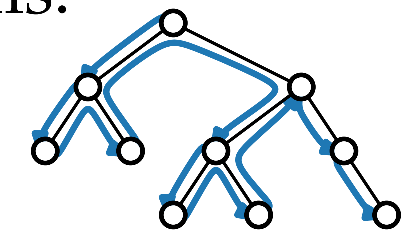
G is a **tree** if the following equivalent conditions hold:
1. there is exactly one v-w-path between any $v, w \in V$
2. G cycle-free and connected
3. G cycle-free and $m = n - 1$
4. G connected and $m = n - 1$



Leaf: Vertex of degree 1
Rooted tree: tree with designated **root**
Ancestor: Vertex on path to root
Parent: Neighbor on path to root
Successor: Vertex not on path to root
Child: Neighbor not on path to root
Depth: Length of path to root
Height: Maximum depth of a leaf

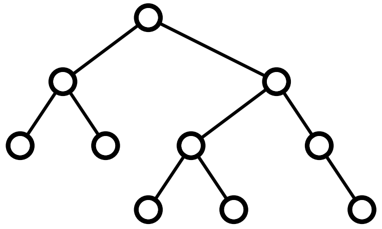
3 traversals:

preorder



node – left – right

inorder

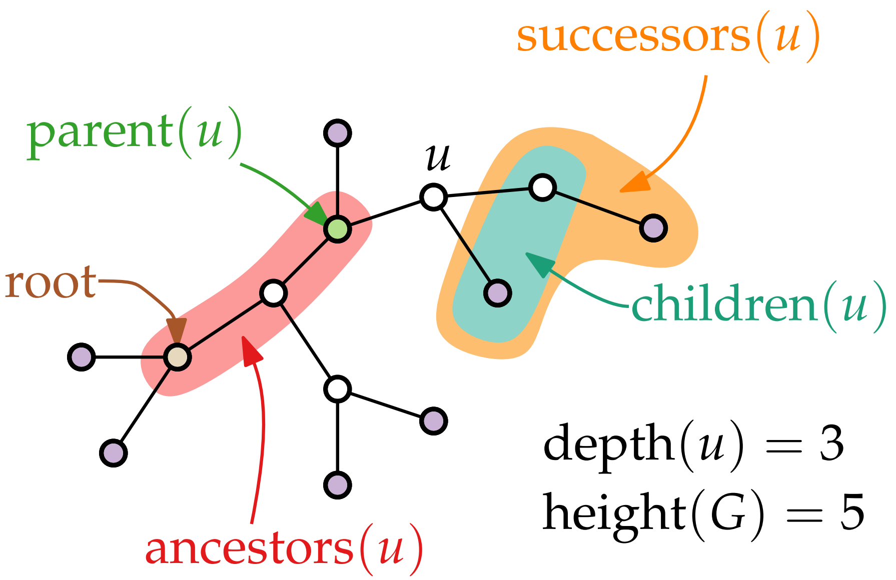


left – node – right

Binary Tree: At most two children per vertex (left / right child)

(Rooted) Trees

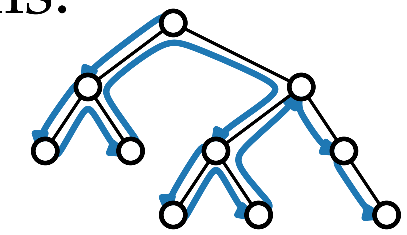
G is a **tree** if the following equivalent conditions hold:
1. there is exactly one v-w-path between any $v, w \in V$
2. G cycle-free and connected
3. G cycle-free and $m = n - 1$
4. G connected and $m = n - 1$



Leaf: Vertex of degree 1
Rooted tree: tree with designated **root**
Ancestor: Vertex on path to root
Parent: Neighbor on path to root
Successor: Vertex not on path to root
Child: Neighbor not on path to root
Depth: Length of path to root
Height: Maximum depth of a leaf

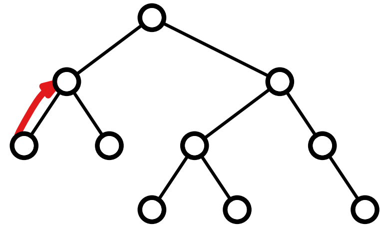
3 traversals:

preorder



node – left – right

inorder

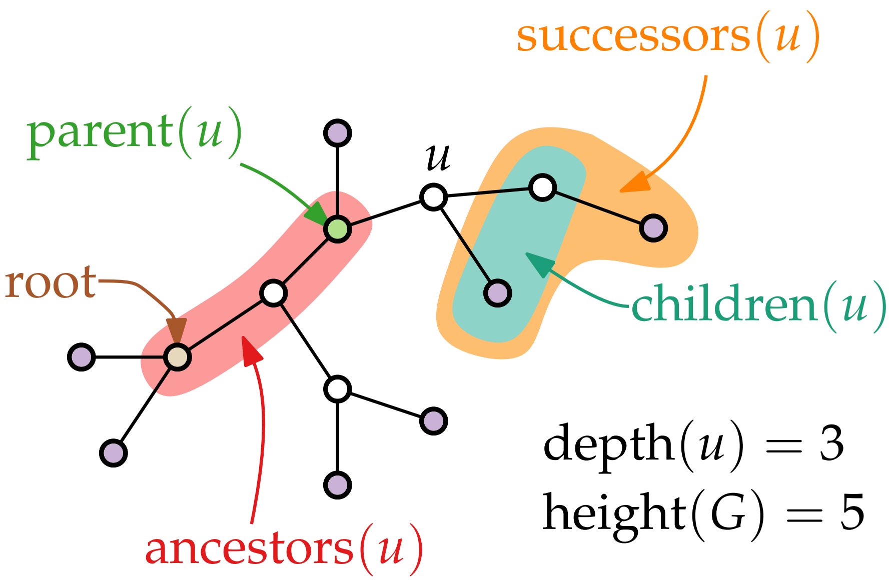


left – node – right

Binary Tree: At most two children per vertex (left / right child)

(Rooted) Trees

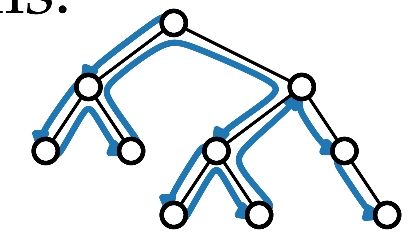
G is a **tree** if the following equivalent conditions hold:
1. there is exactly one v-w-path between any $v, w \in V$
2. G cycle-free and connected
3. G cycle-free and $m = n - 1$
4. G connected and $m = n - 1$



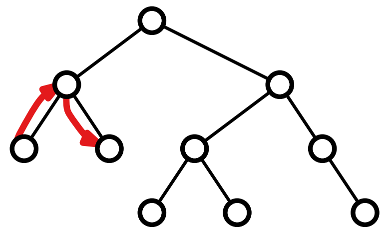
Leaf: Vertex of degree 1
Rooted tree: tree with designated **root**
Ancestor: Vertex on path to root
Parent: Neighbor on path to root
Successor: Vertex not on path to root
Child: Neighbor not on path to root
Depth: Length of path to root
Height: Maximum depth of a leaf

3 traversals:

preorder



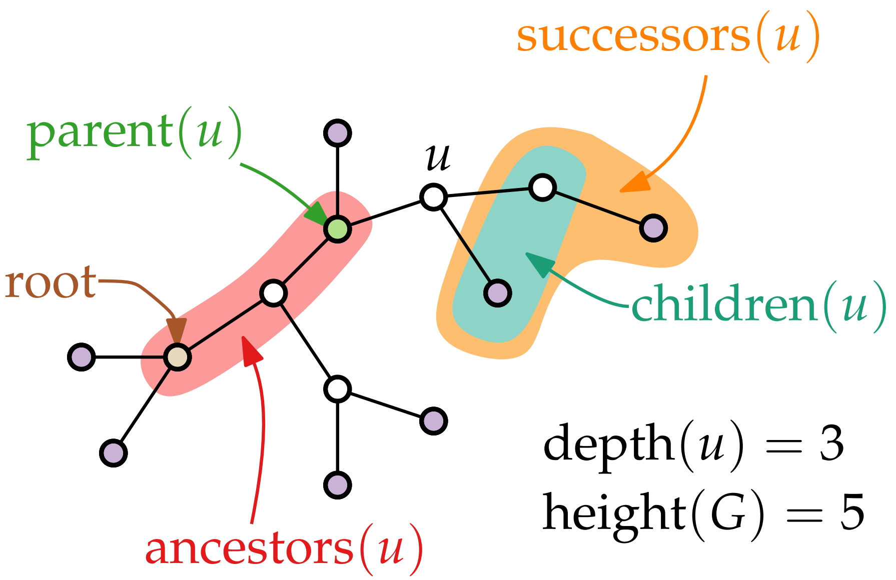
inorder



Binary Tree: At most two children per vertex (left / right child)

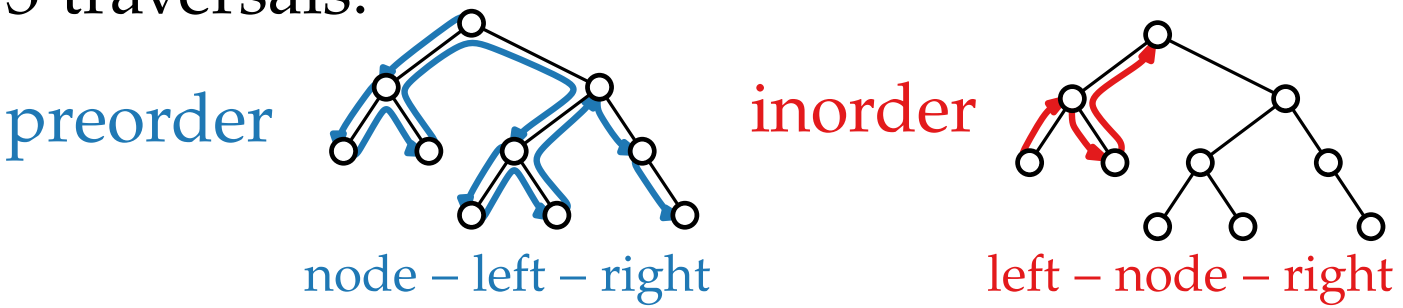
(Rooted) Trees

G is a **tree** if the following equivalent conditions hold:
1. there is exactly one v-w-path between any $v, w \in V$
2. G cycle-free and connected
3. G cycle-free and $m = n - 1$
4. G connected and $m = n - 1$



Leaf: Vertex of degree 1
Rooted tree: tree with designated **root**
Ancestor: Vertex on path to root
Parent: Neighbor on path to root
Successor: Vertex not on path to root
Child: Neighbor not on path to root
Depth: Length of path to root
Height: Maximum depth of a leaf

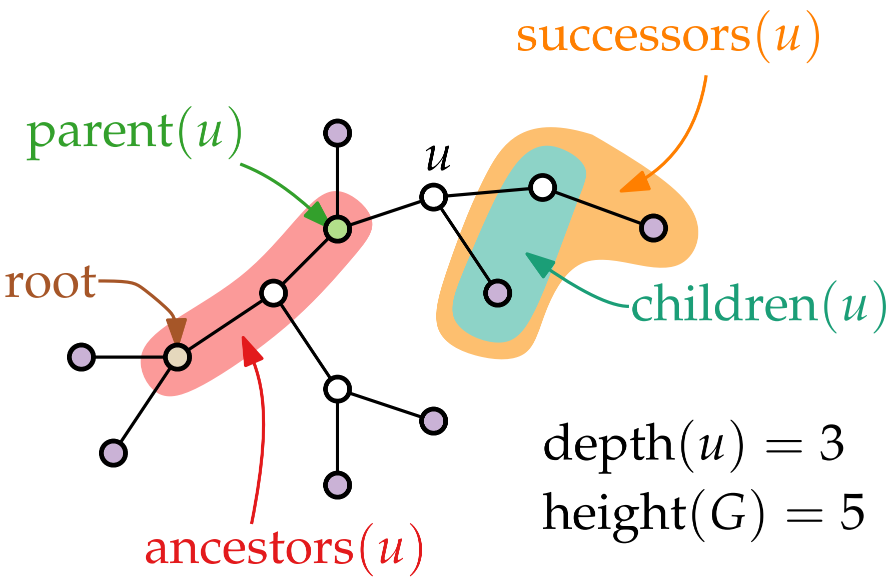
3 traversals:



Binary Tree: At most two children per vertex (left / right child)

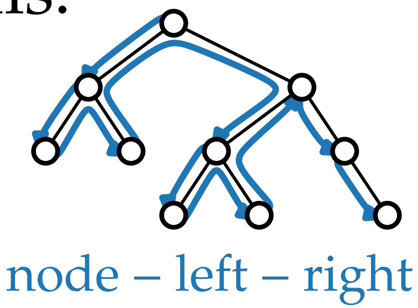
(Rooted) Trees

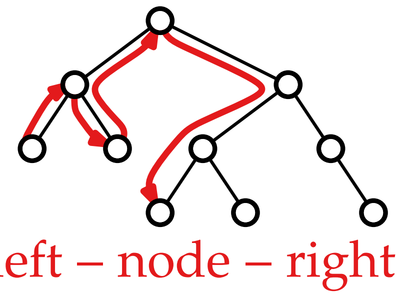
G is a **tree** if the following equivalent conditions hold:
1. there is exactly one v-w-path between any $v, w \in V$
2. G cycle-free and connected
3. G cycle-free and $m = n - 1$
4. G connected and $m = n - 1$



Leaf: Vertex of degree 1
Rooted tree: tree with designated **root**
Ancestor: Vertex on path to root
Parent: Neighbor on path to root
Successor: Vertex not on path to root
Child: Neighbor not on path to root
Depth: Length of path to root
Height: Maximum depth of a leaf

3 traversals:

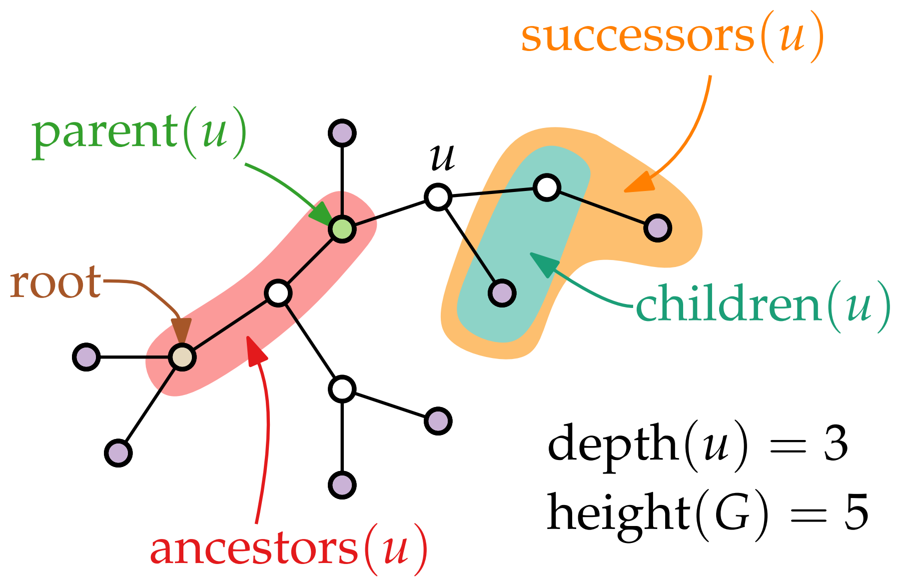




Binary Tree: At most two children per vertex (left / right child)

(Rooted) Trees

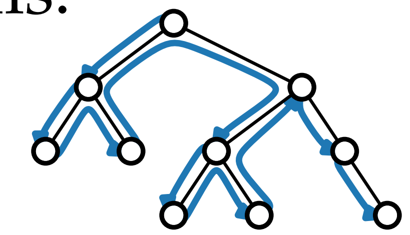
G is a **tree** if the following equivalent conditions hold:
1. there is exactly one v-w-path between any $v, w \in V$
2. G cycle-free and connected
3. G cycle-free and $m = n - 1$
4. G connected and $m = n - 1$



Leaf: Vertex of degree 1
Rooted tree: tree with designated **root**
Ancestor: Vertex on path to root
Parent: Neighbor on path to root
Successor: Vertex not on path to root
Child: Neighbor not on path to root
Depth: Length of path to root
Height: Maximum depth of a leaf

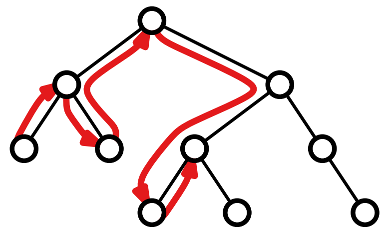
3 traversals:

preorder



node – left – right

inorder

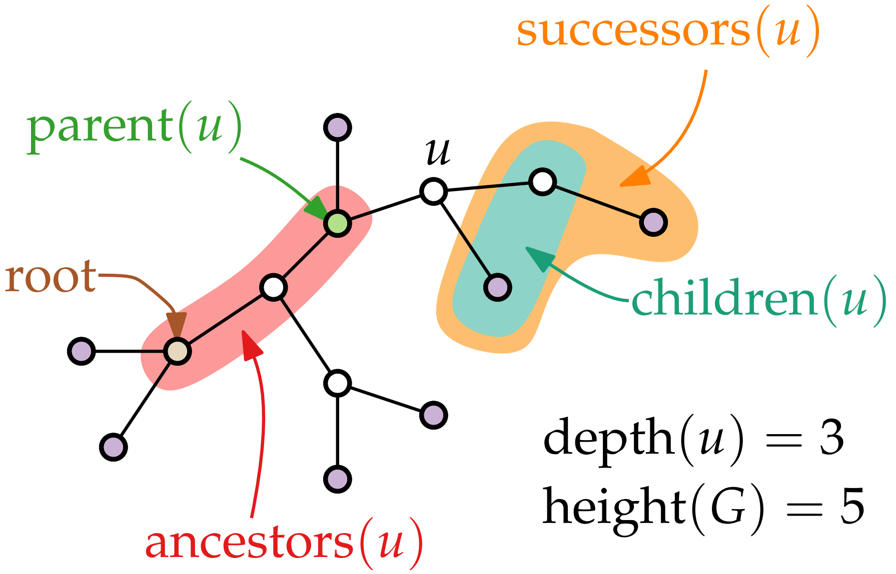


left – node – right

Binary Tree: At most two children per vertex (left / right child)

(Rooted) Trees

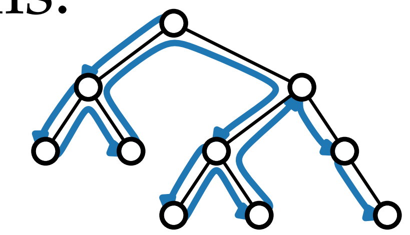
G is a **tree** if the following equivalent conditions hold:
1. there is exactly one v-w-path between any $v, w \in V$
2. G cycle-free and connected
3. G cycle-free and $m = n - 1$
4. G connected and $m = n - 1$



Leaf: Vertex of degree 1
Rooted tree: tree with designated **root**
Ancestor: Vertex on path to root
Parent: Neighbor on path to root
Successor: Vertex not on path to root
Child: Neighbor not on path to root
Depth: Length of path to root
Height: Maximum depth of a leaf

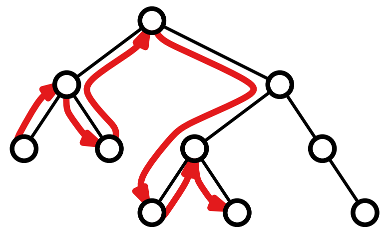
3 traversals:

preorder



node – left – right

inorder

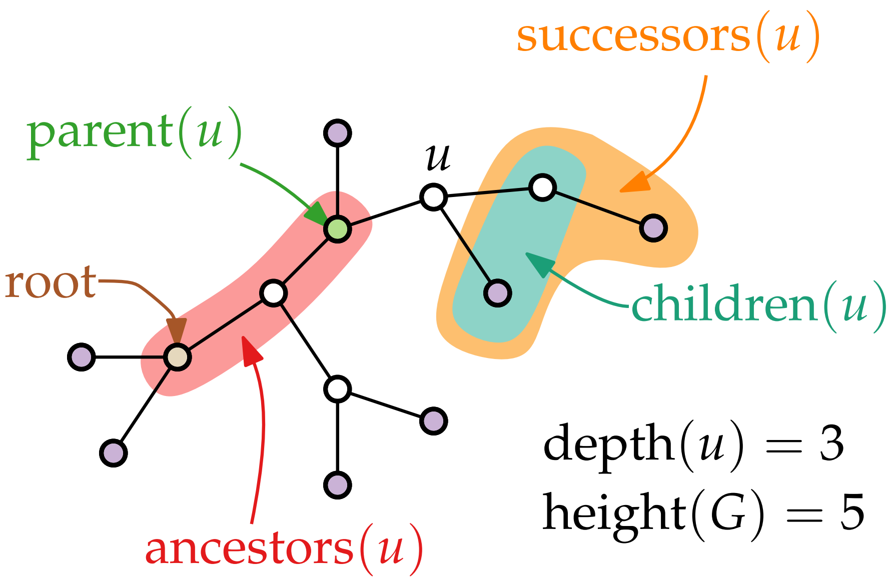


left – node – right

Binary Tree: At most two children per vertex (left / right child)

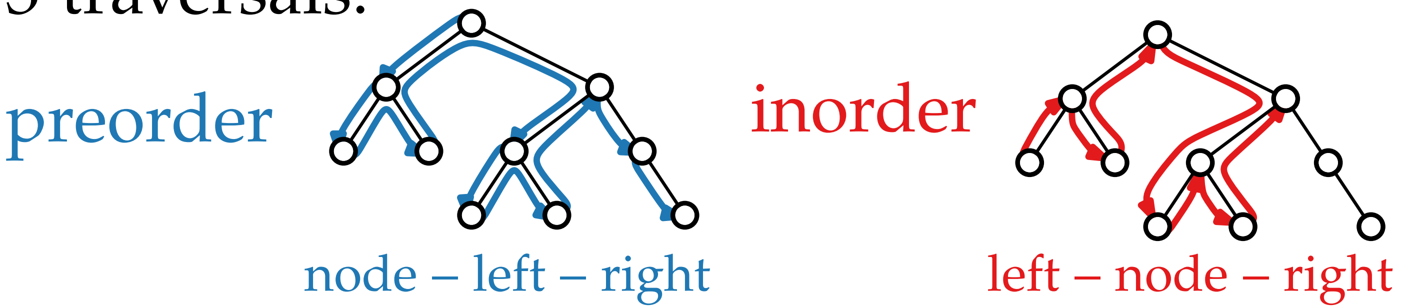
(Rooted) Trees

G is a **tree** if the following equivalent conditions hold:
1. there is exactly one v-w-path between any $v, w \in V$
2. G cycle-free and connected
3. G cycle-free and $m = n - 1$
4. G connected and $m = n - 1$



Leaf: Vertex of degree 1
Rooted tree: tree with designated **root**
Ancestor: Vertex on path to root
Parent: Neighbor on path to root
Successor: Vertex not on path to root
Child: Neighbor not on path to root
Depth: Length of path to root
Height: Maximum depth of a leaf

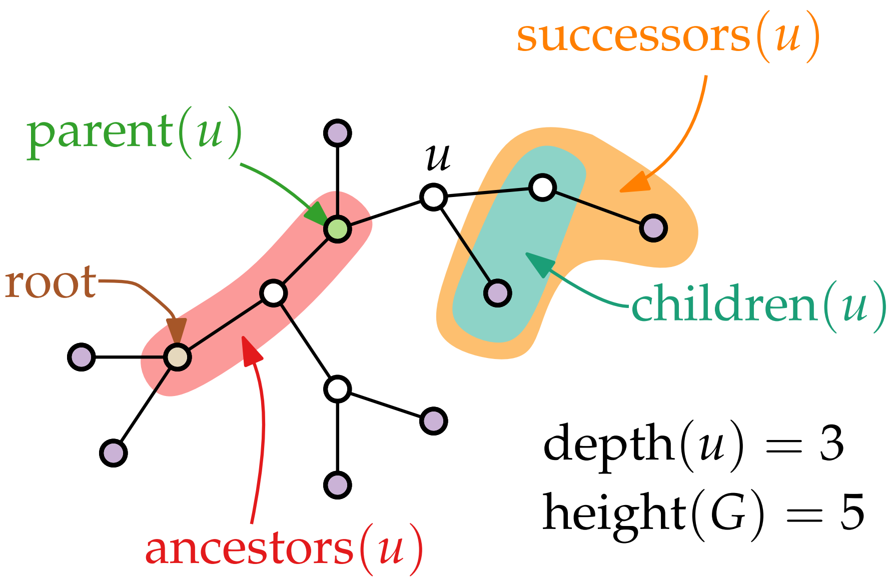
3 traversals:



Binary Tree: At most two children per vertex (left / right child)

(Rooted) Trees

G is a **tree** if the following equivalent conditions hold:
1. there is exactly one v-w-path between any $v, w \in V$
2. G cycle-free and connected
3. G cycle-free and $m = n - 1$
4. G connected and $m = n - 1$

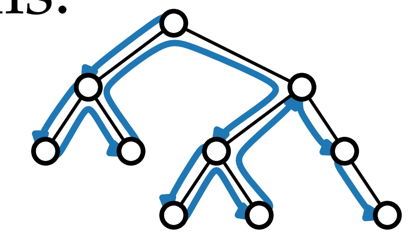


depth(u) $= 3$
height(G) $= 5$

Leaf: Vertex of degree 1
Rooted tree: tree with designated **root**
Ancestor: Vertex on path to root
Parent: Neighbor on path to root
Successor: Vertex not on path to root
Child: Neighbor not on path to root
Depth: Length of path to root
Height: Maximum depth of a leaf

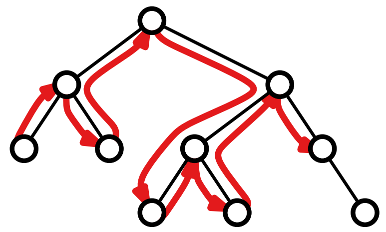
3 traversals:

preorder



node – left – right

inorder

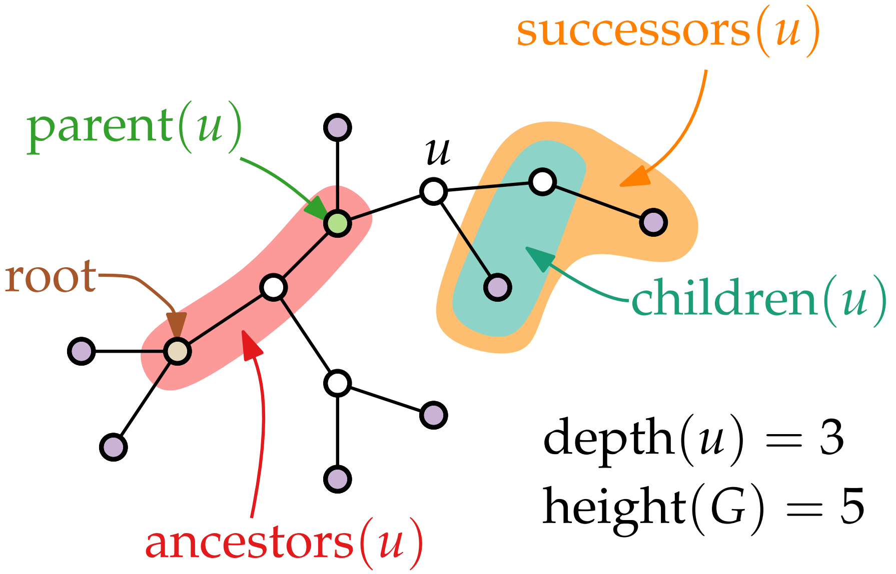


left – node – right

Binary Tree: At most two children per vertex (left / right child)

(Rooted) Trees

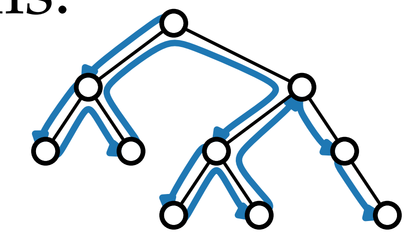
G is a **tree** if the following equivalent conditions hold:
1. there is exactly one v-w-path between any $v, w \in V$
2. G cycle-free and connected
3. G cycle-free and $m = n - 1$
4. G connected and $m = n - 1$



Leaf: Vertex of degree 1
Rooted tree: tree with designated **root**
Ancestor: Vertex on path to root
Parent: Neighbor on path to root
Successor: Vertex not on path to root
Child: Neighbor not on path to root
Depth: Length of path to root
Height: Maximum depth of a leaf

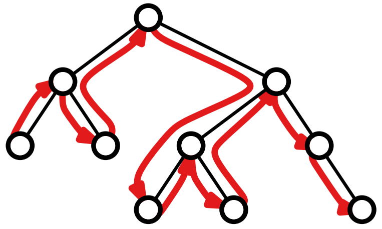
3 traversals:

preorder



node – left – right

inorder

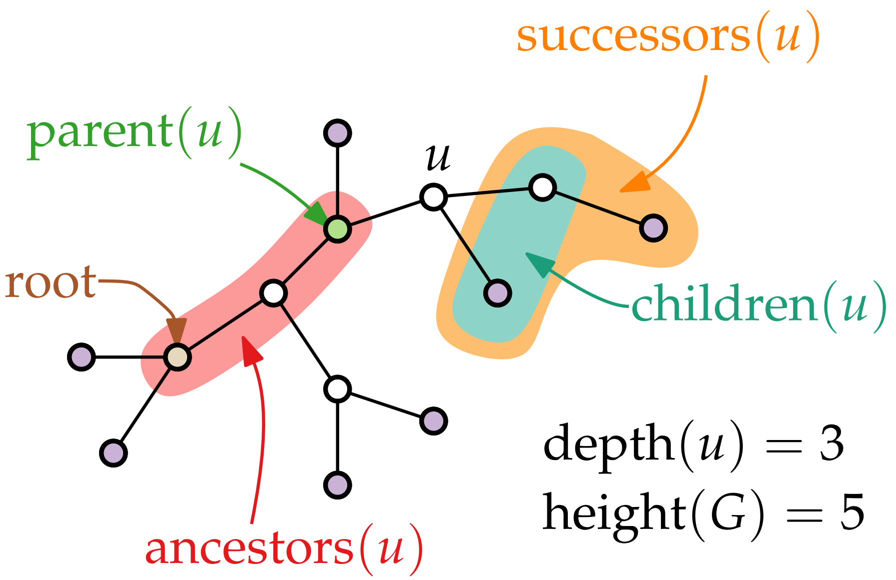


left – node – right

Binary Tree: At most two children per vertex (left / right child)

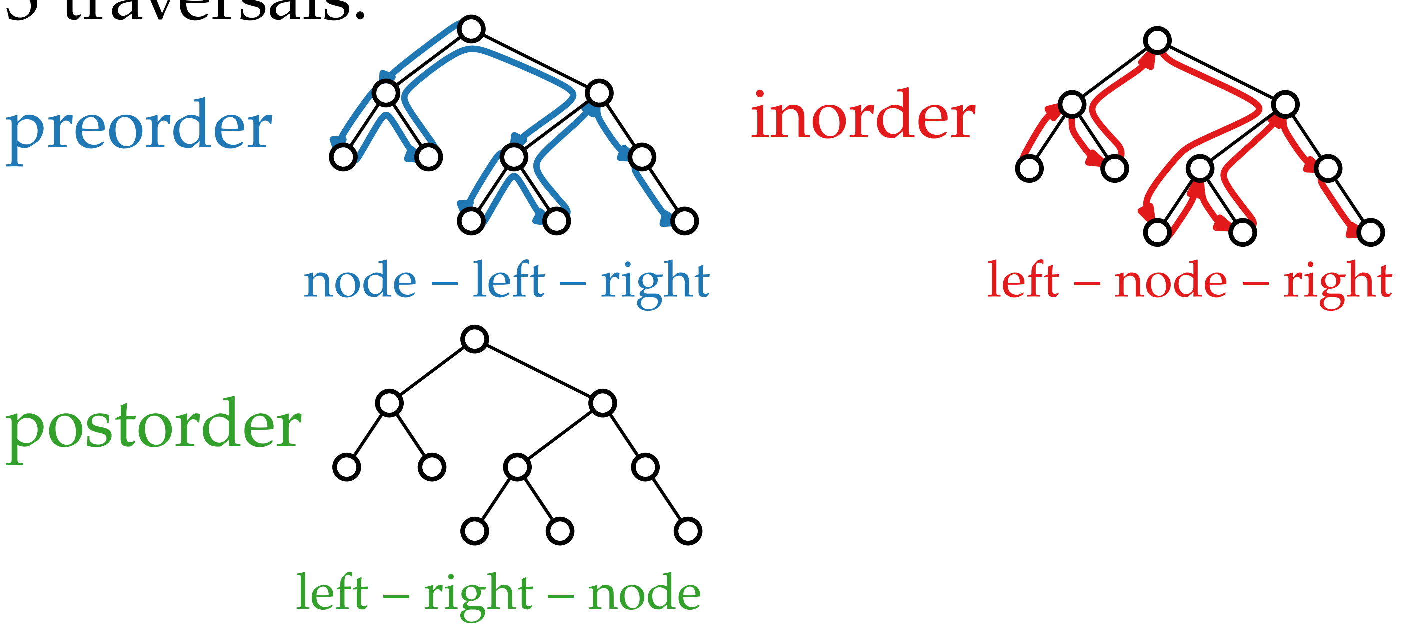
(Rooted) Trees

G is a **tree** if the following equivalent conditions hold:
1. there is exactly one v-w-path between any $v, w \in V$
2. G cycle-free and connected
3. G cycle-free and $m = n - 1$
4. G connected and $m = n - 1$



Leaf: Vertex of degree 1
Rooted tree: tree with designated **root**
Ancestor: Vertex on path to root
Parent: Neighbor on path to root
Successor: Vertex not on path to root
Child: Neighbor not on path to root
Depth: Length of path to root
Height: Maximum depth of a leaf

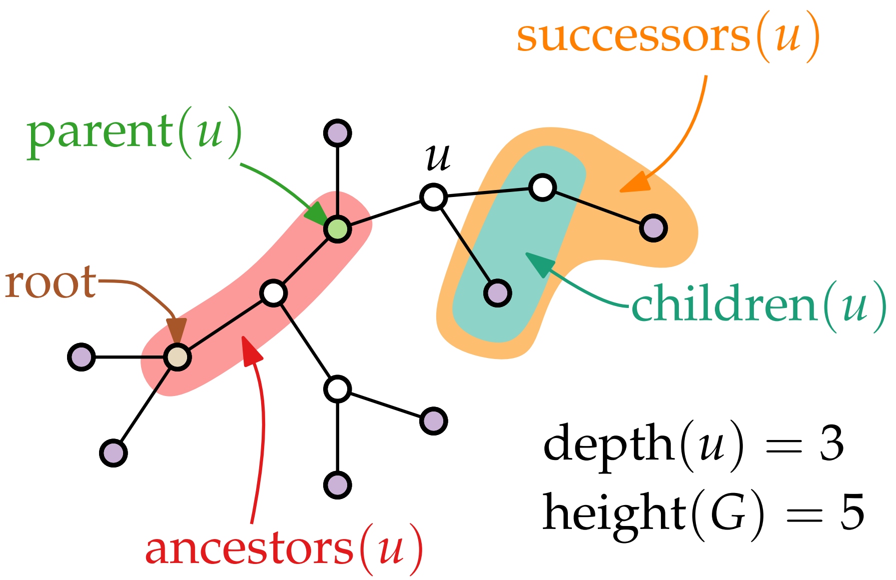
3 traversals:



Binary Tree: At most two children per vertex (left / right child)

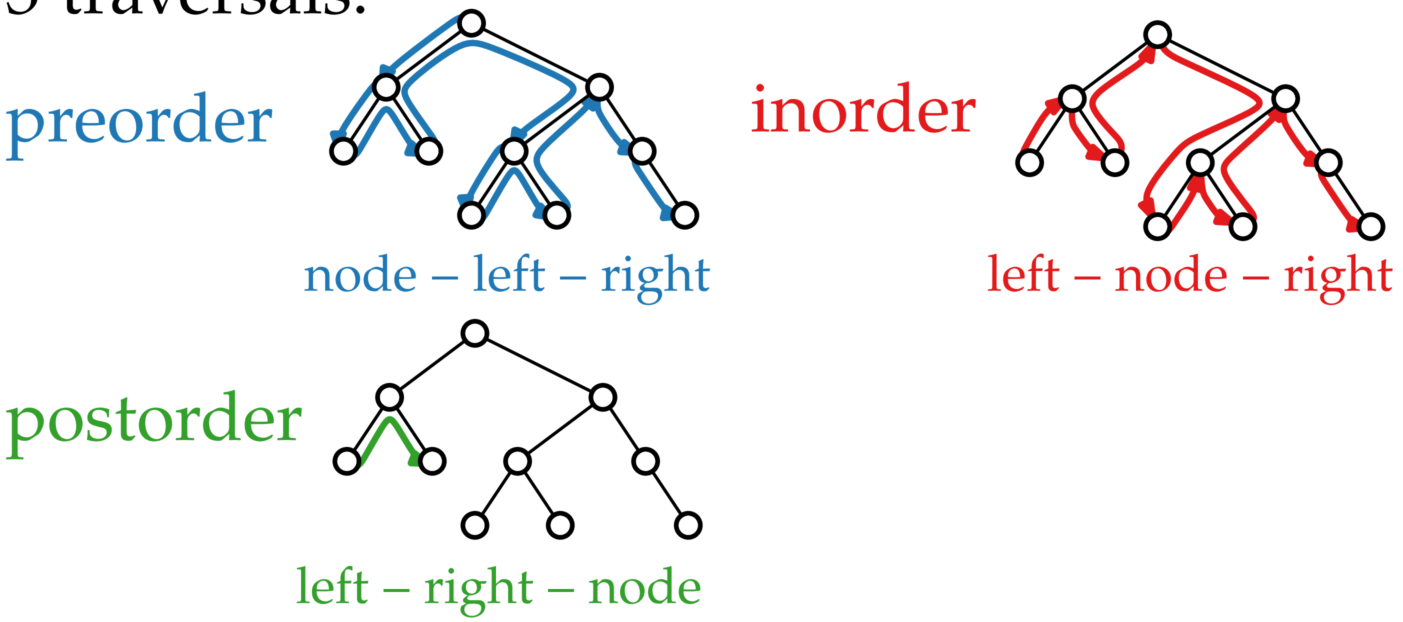
(Rooted) Trees

G is a **tree** if the following equivalent conditions hold:
1. there is exactly one v-w-path between any $v, w \in V$
2. G cycle-free and connected
3. G cycle-free and $m = n - 1$
4. G connected and $m = n - 1$



Leaf: Vertex of degree 1
Rooted tree: tree with designated **root**
Ancestor: Vertex on path to root
Parent: Neighbor on path to root
Successor: Vertex not on path to root
Child: Neighbor not on path to root
Depth: Length of path to root
Height: Maximum depth of a leaf

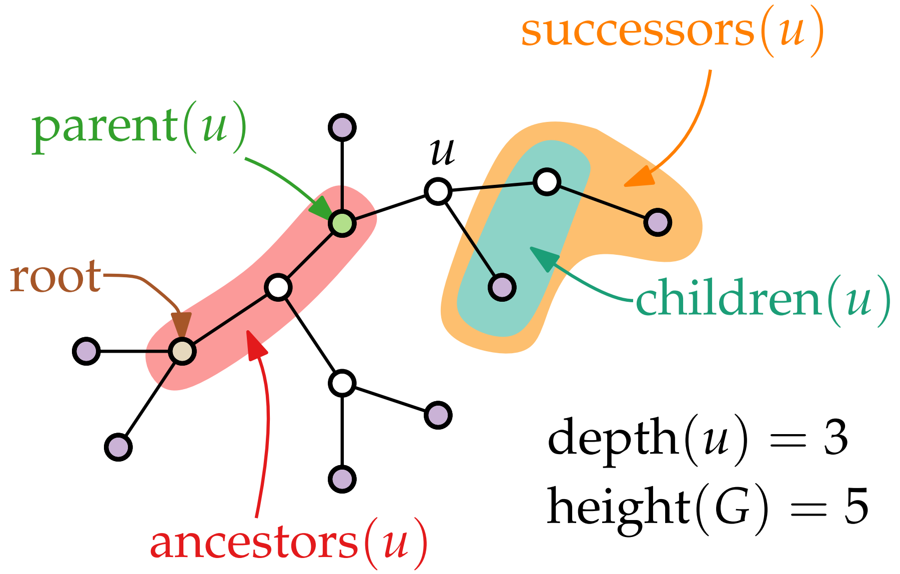
3 traversals:



Binary Tree: At most two children per vertex (left / right child)

(Rooted) Trees

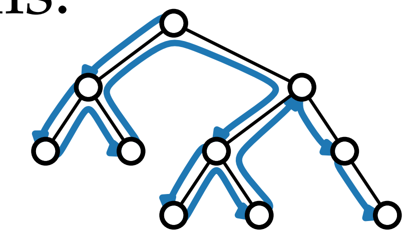
G is a **tree** if the following equivalent conditions hold:
1. there is exactly one v-w-path between any $v, w \in V$
2. G cycle-free and connected
3. G cycle-free and $m = n - 1$
4. G connected and $m = n - 1$



Leaf: Vertex of degree 1
Rooted tree: tree with designated **root**
Ancestor: Vertex on path to root
Parent: Neighbor on path to root
Successor: Vertex not on path to root
Child: Neighbor not on path to root
Depth: Length of path to root
Height: Maximum depth of a leaf

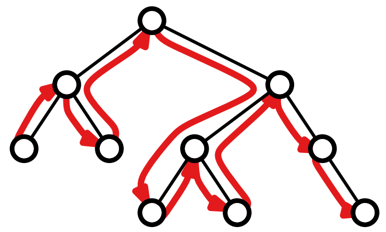
3 traversals:

preorder



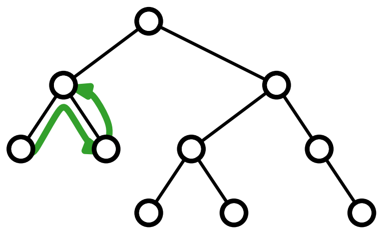
node – left – right

inorder



left – node – right

postorder

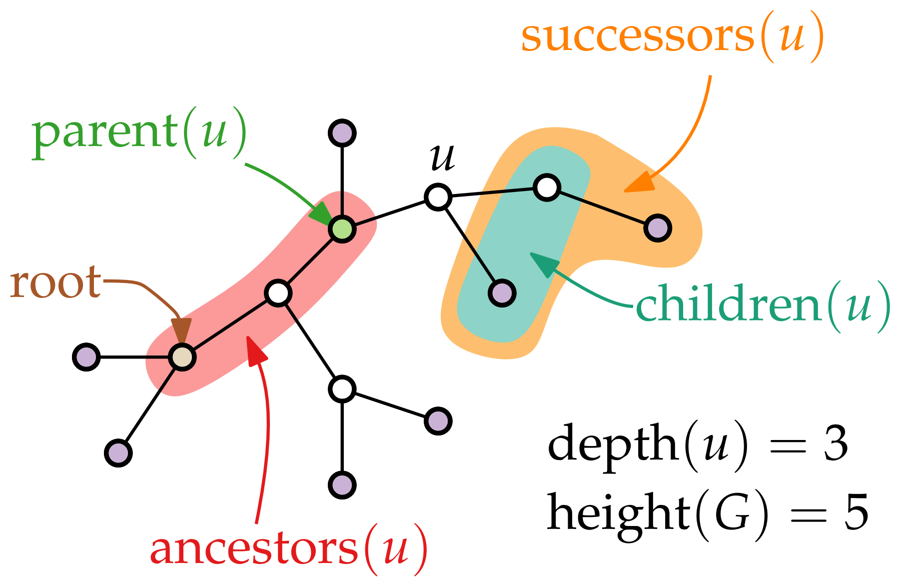


left – right – node

Binary Tree: At most two children per vertex (left / right child)

(Rooted) Trees

G is a **tree** if the following equivalent conditions hold:
1. there is exactly one v-w-path between any $v, w \in V$
2. G cycle-free and connected
3. G cycle-free and $m = n - 1$
4. G connected and $m = n - 1$

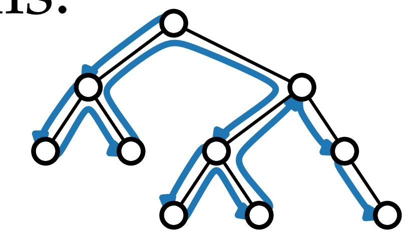


depth(u) $= 3$
height(G) $= 5$

Leaf: Vertex of degree 1
Rooted tree: tree with designated **root**
Ancestor: Vertex on path to root
Parent: Neighbor on path to root
Successor: Vertex not on path to root
Child: Neighbor not on path to root
Depth: Length of path to root
Height: Maximum depth of a leaf

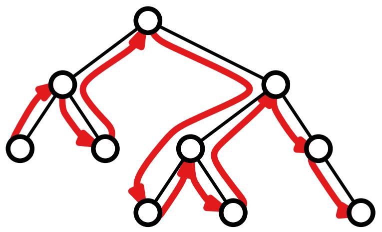
3 traversals:

preorder



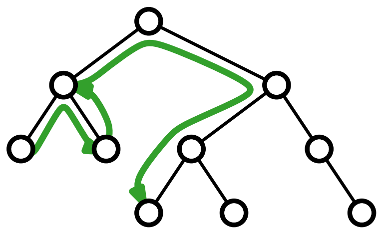
node – left – right

inorder



left – node – right

postorder

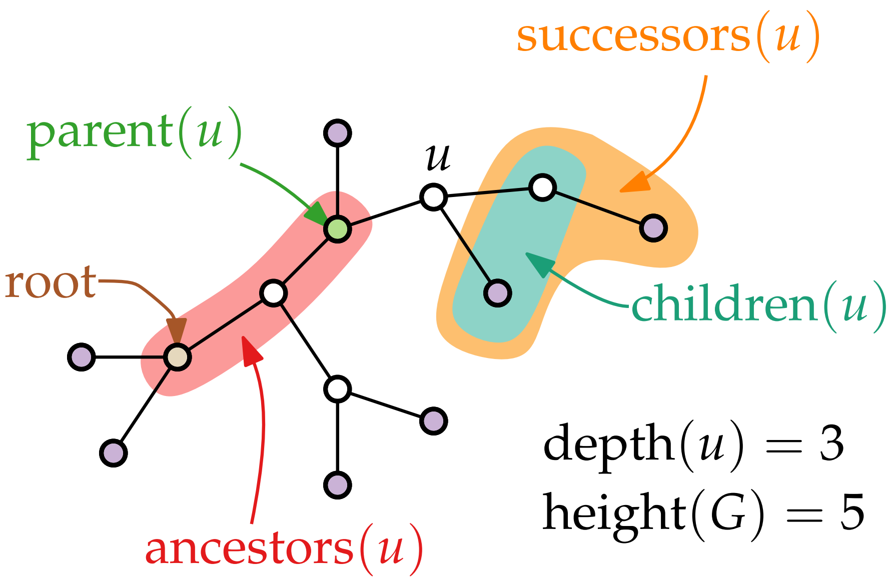


left – right – node

Binary Tree: At most two children per vertex (left / right child)

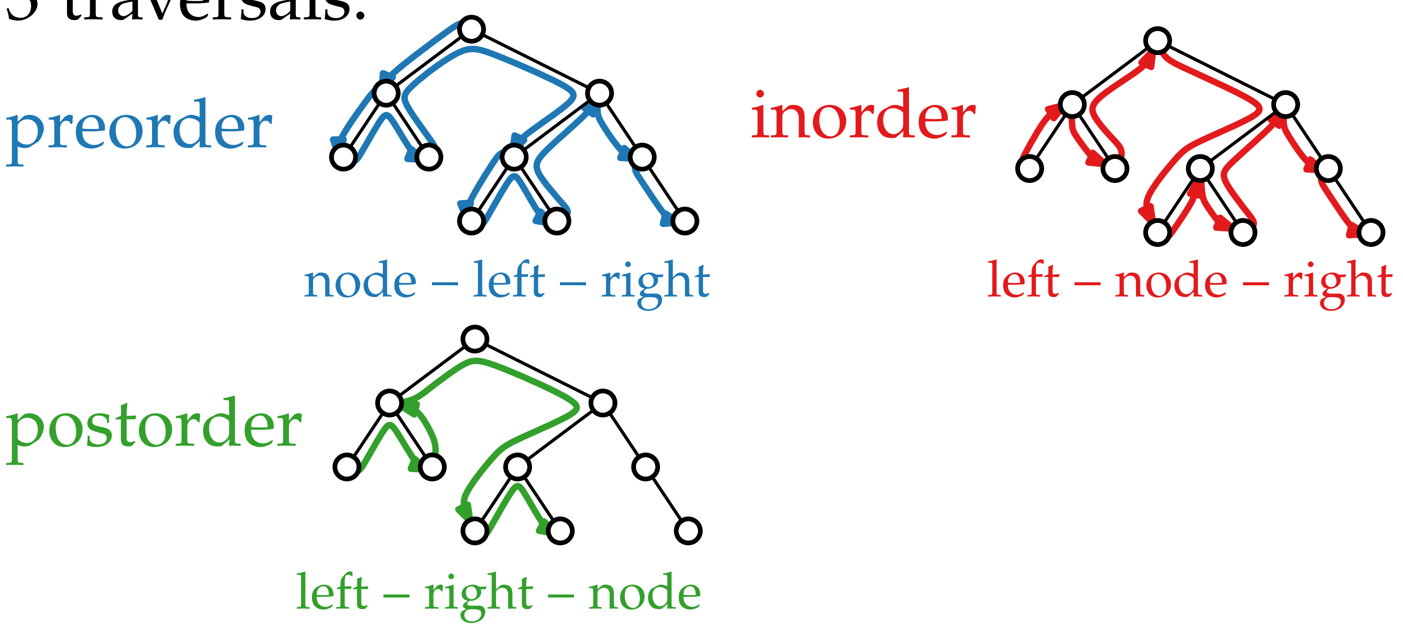
(Rooted) Trees

G is a **tree** if the following equivalent conditions hold:
1. there is exactly one v-w-path between any $v, w \in V$
2. G cycle-free and connected
3. G cycle-free and $m = n - 1$
4. G connected and $m = n - 1$



Leaf: Vertex of degree 1
Rooted tree: tree with designated **root**
Ancestor: Vertex on path to root
Parent: Neighbor on path to root
Successor: Vertex not on path to root
Child: Neighbor not on path to root
Depth: Length of path to root
Height: Maximum depth of a leaf

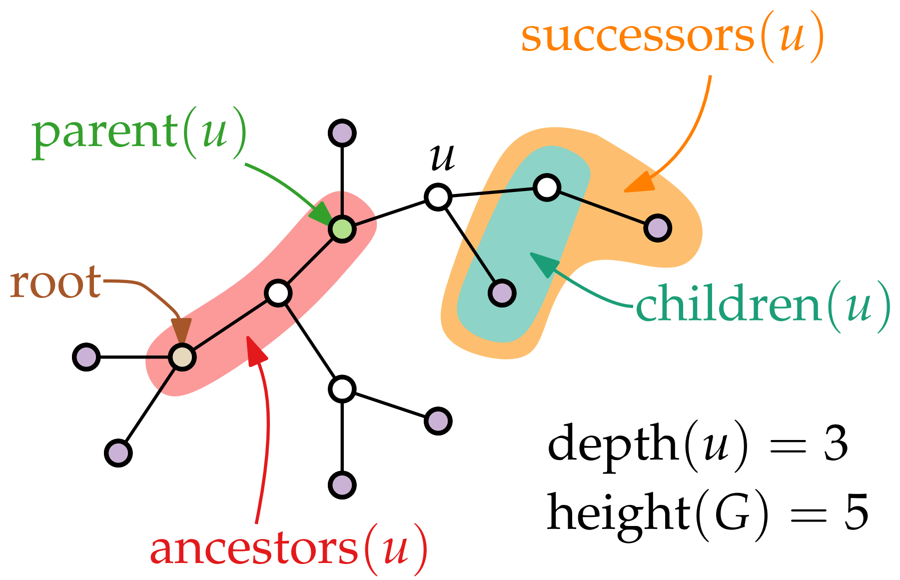
3 traversals:



Binary Tree: At most two children per vertex (left / right child)

(Rooted) Trees

G is a **tree** if the following equivalent conditions hold:
1. there is exactly one v-w-path between any $v, w \in V$
2. G cycle-free and connected
3. G cycle-free and $m = n - 1$
4. G connected and $m = n - 1$

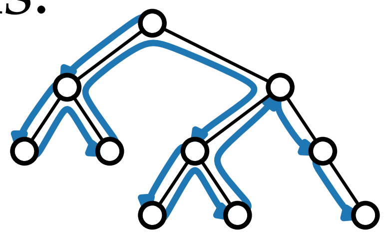


depth(u) $= 3$
height(G) $= 5$

Leaf: Vertex of degree 1
Rooted tree: tree with designated **root**
Ancestor: Vertex on path to root
Parent: Neighbor on path to root
Successor: Vertex not on path to root
Child: Neighbor not on path to root
Depth: Length of path to root
Height: Maximum depth of a leaf

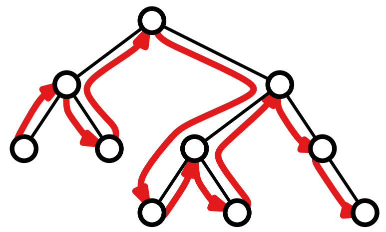
3 traversals:

preorder



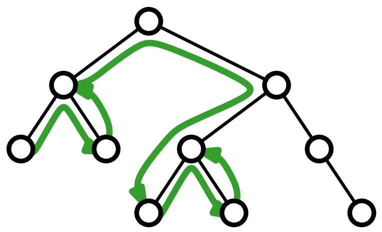
node – left – right

inorder



left – node – right

postorder

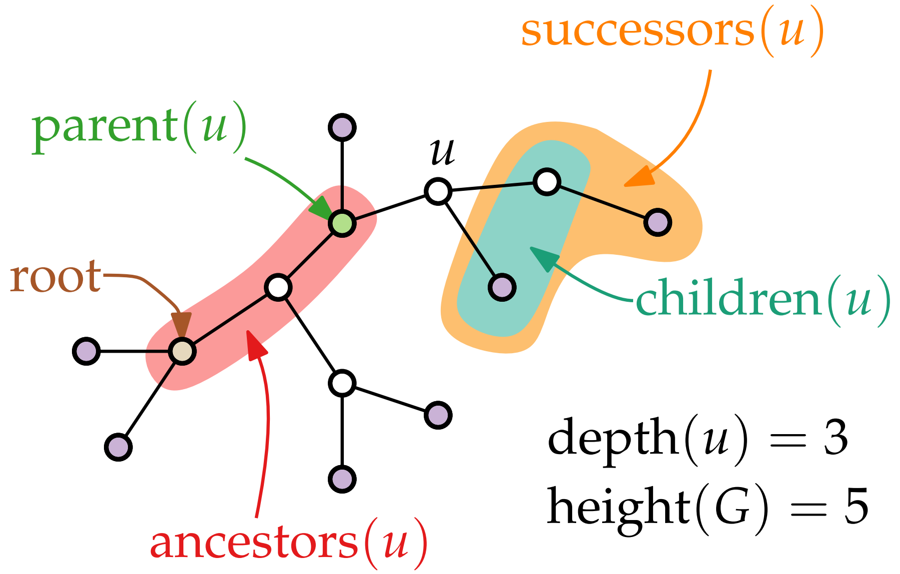


left – right – node

Binary Tree: At most two children per vertex (left / right child)

(Rooted) Trees

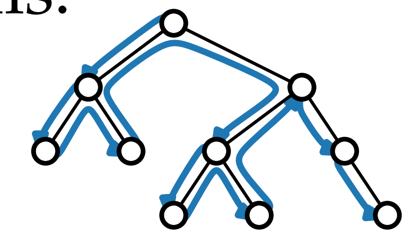
G is a **tree** if the following equivalent conditions hold:
1. there is exactly one v-w-path between any $v, w \in V$
2. G cycle-free and connected
3. G cycle-free and $m = n - 1$
4. G connected and $m = n - 1$



Leaf: Vertex of degree 1
Rooted tree: tree with designated **root**
Ancestor: Vertex on path to root
Parent: Neighbor on path to root
Successor: Vertex not on path to root
Child: Neighbor not on path to root
Depth: Length of path to root
Height: Maximum depth of a leaf

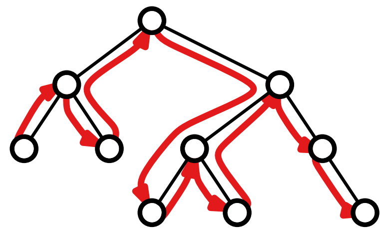
3 traversals:

preorder



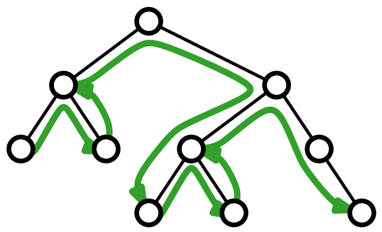
node – left – right

inorder



left – node – right

postorder

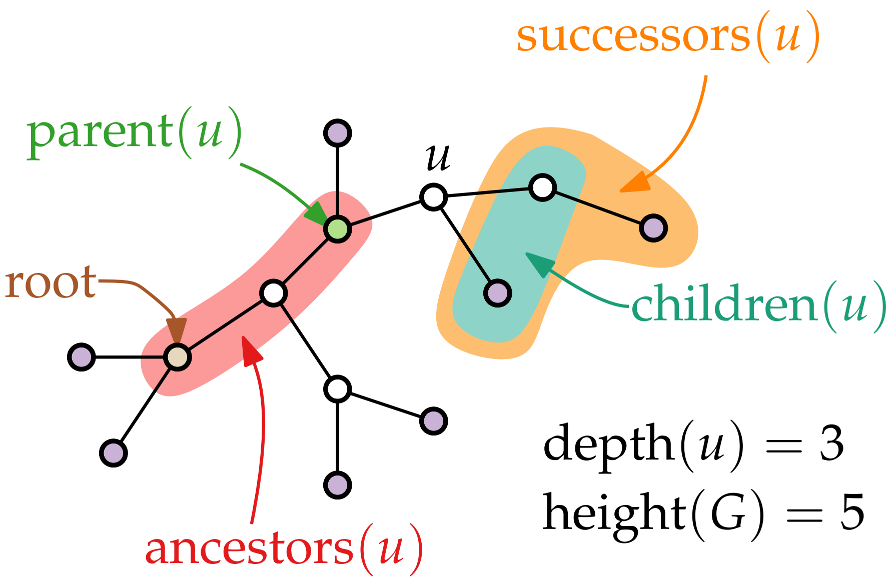


left – right – node

Binary Tree: At most two children per vertex (left / right child)

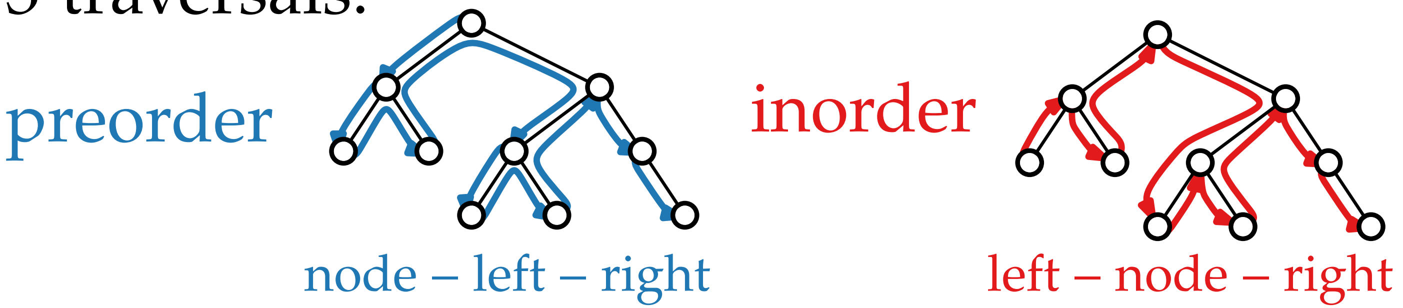
(Rooted) Trees

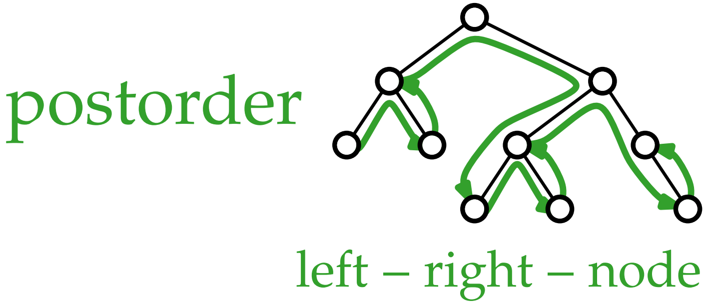
G is a **tree** if the following equivalent conditions hold:
1. there is exactly one v-w-path between any $v, w \in V$
2. G cycle-free and connected
3. G cycle-free and $m = n - 1$
4. G connected and $m = n - 1$



Leaf: Vertex of degree 1
Rooted tree: tree with designated **root**
Ancestor: Vertex on path to root
Parent: Neighbor on path to root
Successor: Vertex not on path to root
Child: Neighbor not on path to root
Depth: Length of path to root
Height: Maximum depth of a leaf

3 traversals:

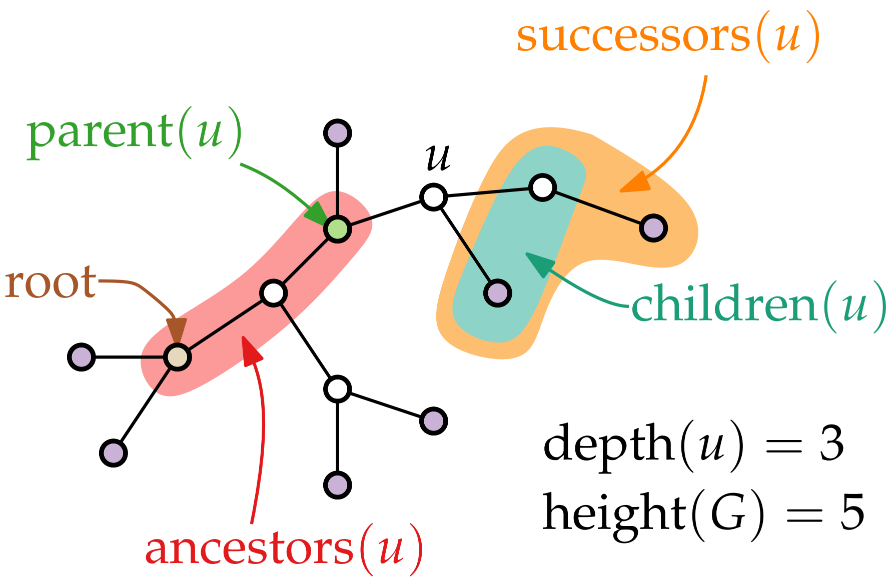




Binary Tree: At most two children per vertex (left / right child)

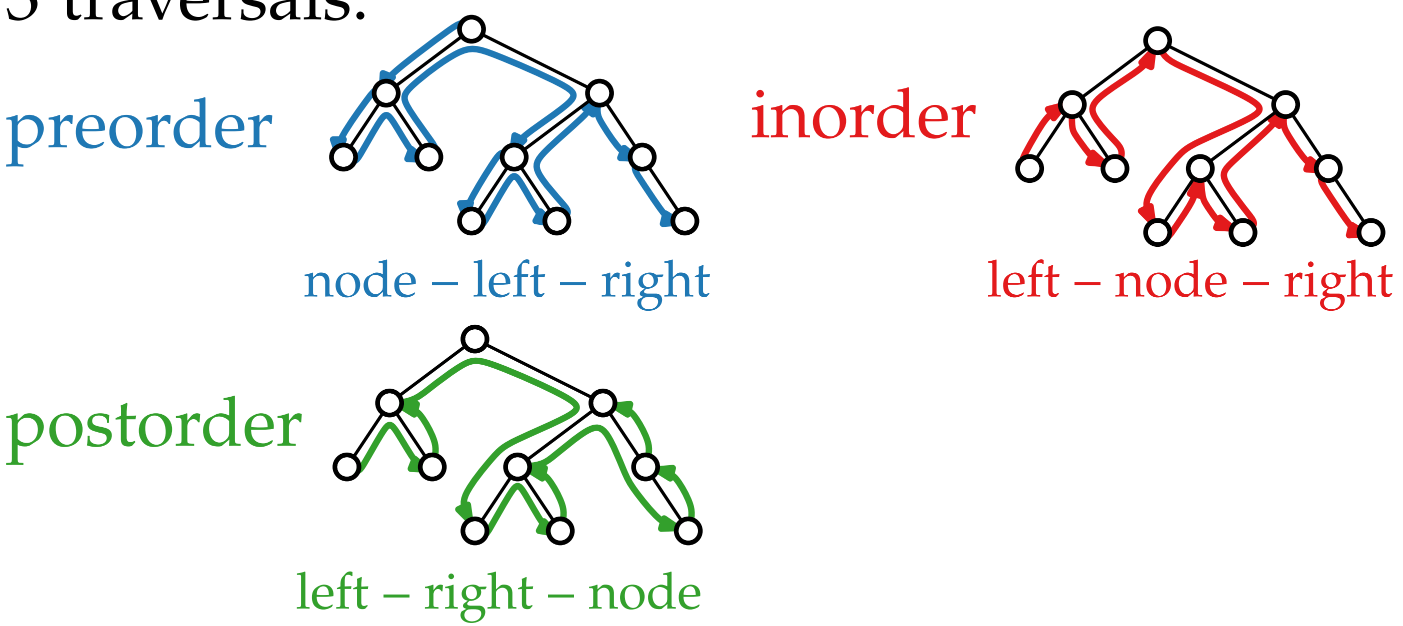
(Rooted) Trees

G is a **tree** if the following equivalent conditions hold:
1. there is exactly one v-w-path between any $v, w \in V$
2. G cycle-free and connected
3. G cycle-free and $m = n - 1$
4. G connected and $m = n - 1$



Leaf: Vertex of degree 1
Rooted tree: tree with designated **root**
Ancestor: Vertex on path to root
Parent: Neighbor on path to root
Successor: Vertex not on path to root
Child: Neighbor not on path to root
Depth: Length of path to root
Height: Maximum depth of a leaf

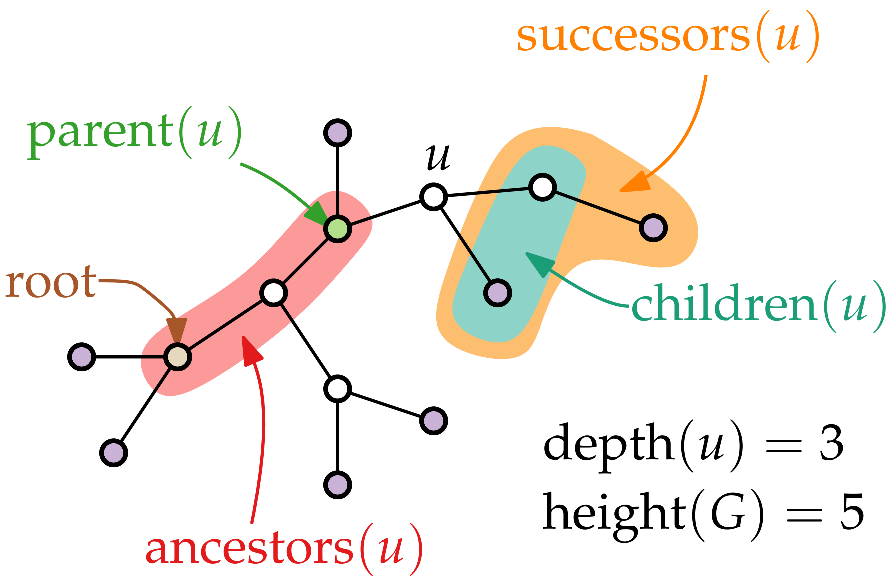
3 traversals:



Binary Tree: At most two children per vertex (left / right child)

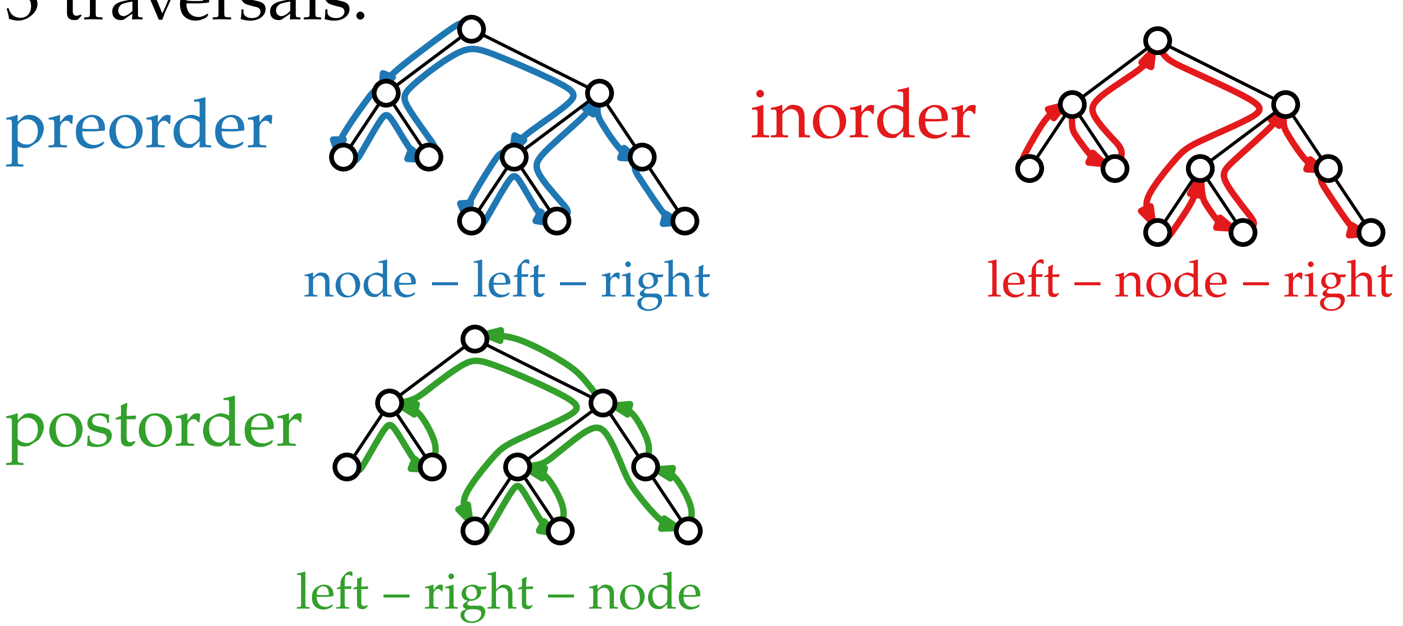
(Rooted) Trees

G is a **tree** if the following equivalent conditions hold:
1. there is exactly one v-w-path between any $v, w \in V$
2. G cycle-free and connected
3. G cycle-free and $m = n - 1$
4. G connected and $m = n - 1$



Leaf: Vertex of degree 1
Rooted tree: tree with designated **root**
Ancestor: Vertex on path to root
Parent: Neighbor on path to root
Successor: Vertex not on path to root
Child: Neighbor not on path to root
Depth: Length of path to root
Height: Maximum depth of a leaf

3 traversals:



Binary Tree: At most two children per vertex (left / right child)

First Grid Layout of Binary Trees

1. Choose y-coordinates:

First Grid Layout of Binary Trees

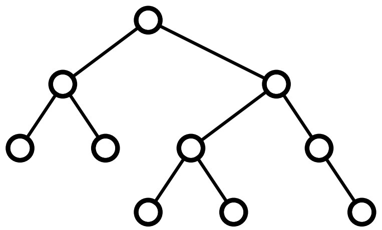
1. Choose y-coordinates:

First Grid Layout of Binary Trees

1. Choose y-coordinates: $y(u) = \text{depth}(u)$

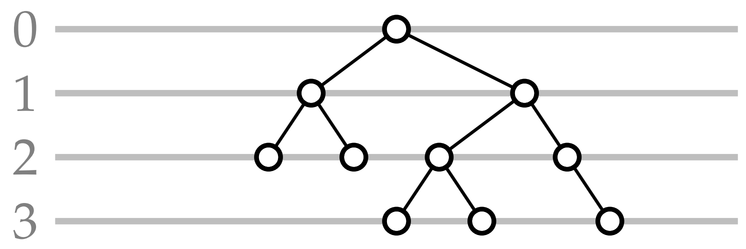
First Grid Layout of Binary Trees

1. Choose y-coordinates: $y(u) = \text{depth}(u)$



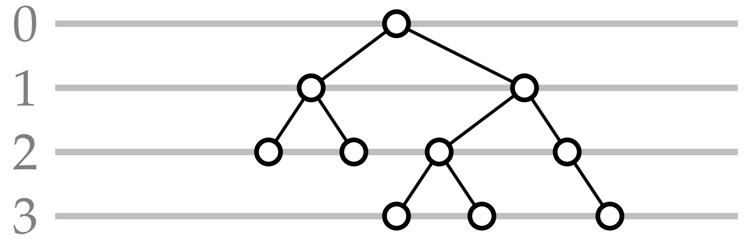
First Grid Layout of Binary Trees

1. Choose y-coordinates: $y(u) = \text{depth}(u)$

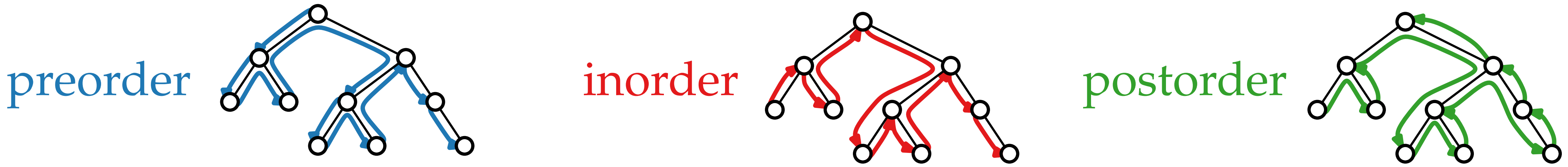


First Grid Layout of Binary Trees

1. Choose y-coordinates: $y(u) =$depth(u)

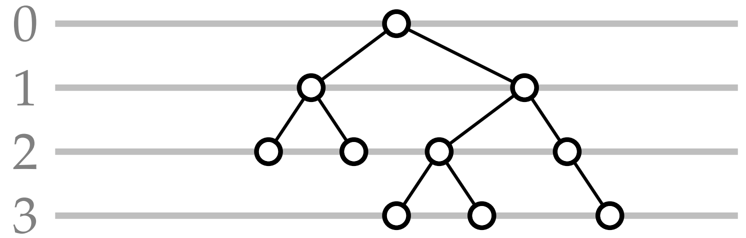


2. Choose x-coordinates:

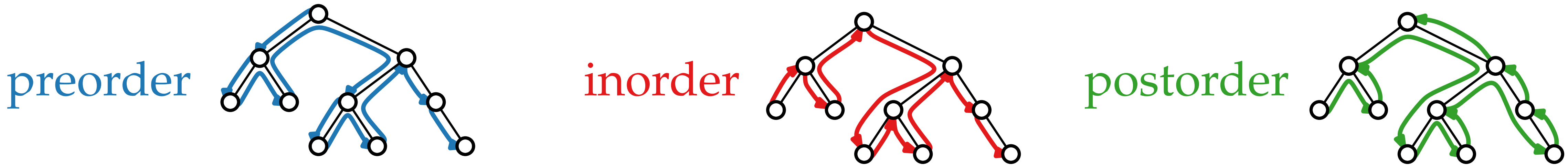


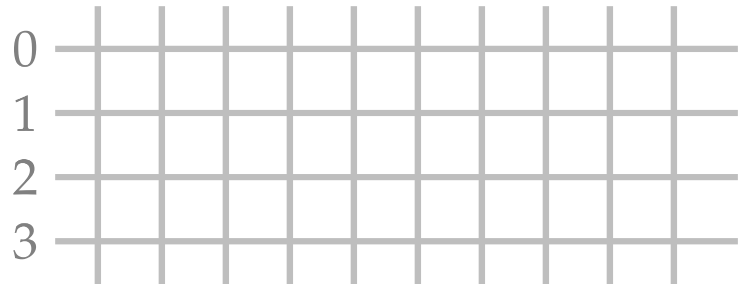
First Grid Layout of Binary Trees

1. Choose y-coordinates: $y(u) = \text{depth}(u)$



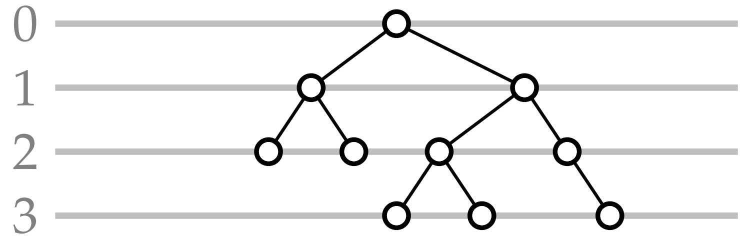
2. Choose x-coordinates:



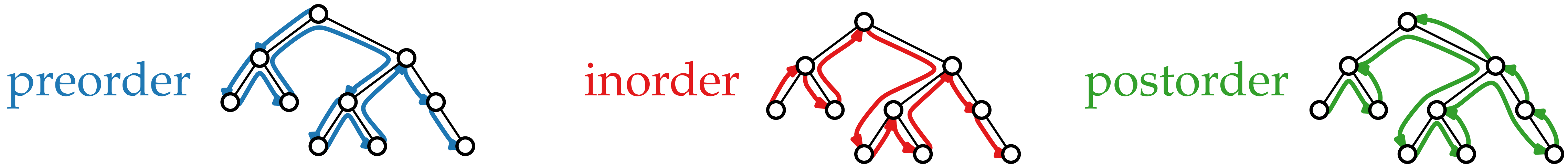


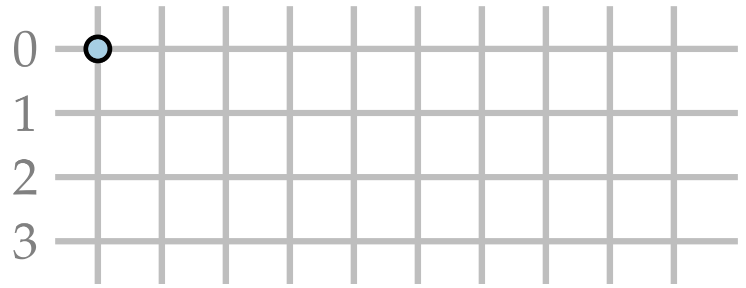
First Grid Layout of Binary Trees

1. Choose y-coordinates: $y(u) = \text{depth}(u)$



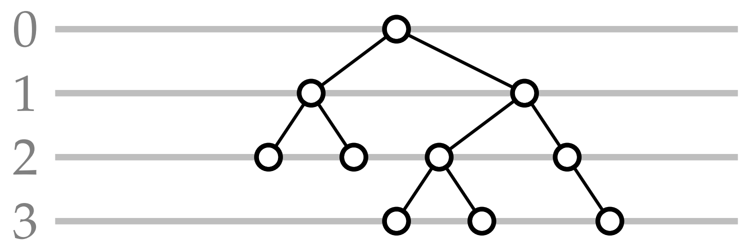
2. Choose x-coordinates:



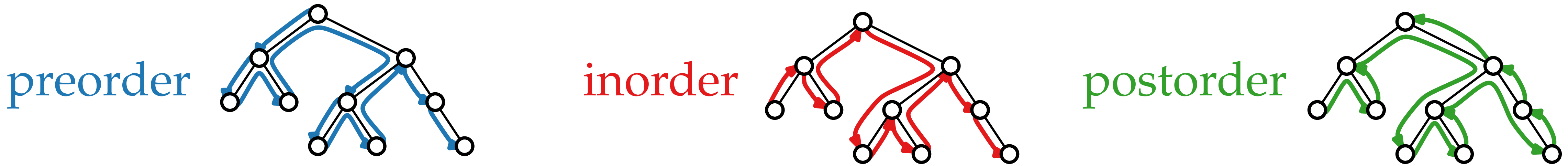


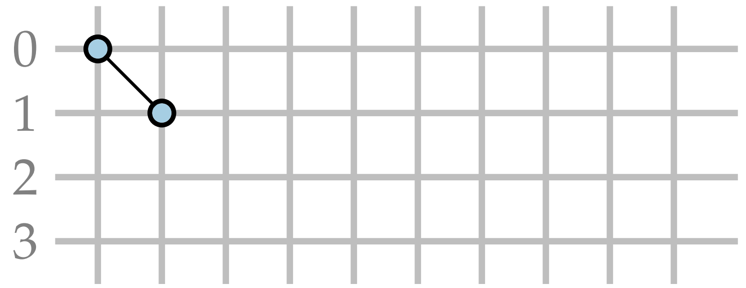
First Grid Layout of Binary Trees

1. Choose y-coordinates: $y(u) = \text{depth}(u)$



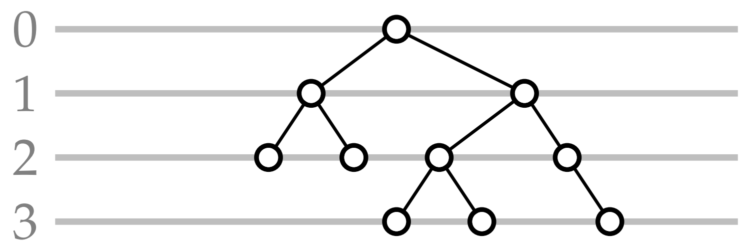
2. Choose x-coordinates:



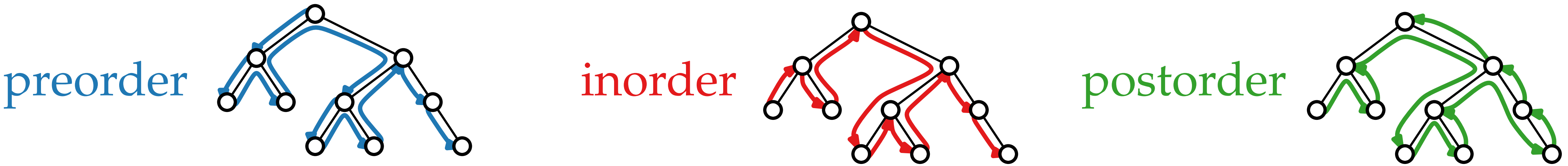


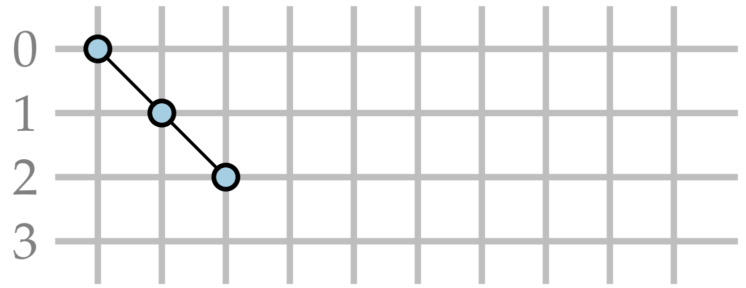
First Grid Layout of Binary Trees

1. Choose y-coordinates: $y(u) = \text{depth}(u)$



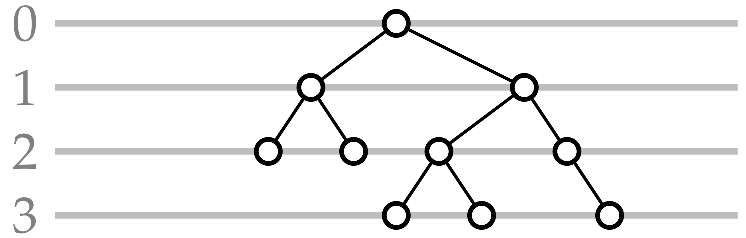
2. Choose x-coordinates:



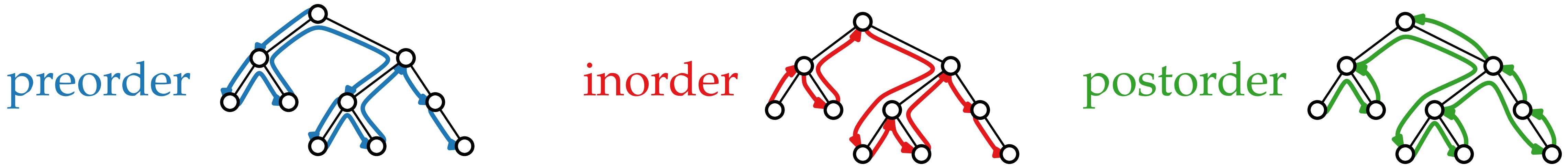


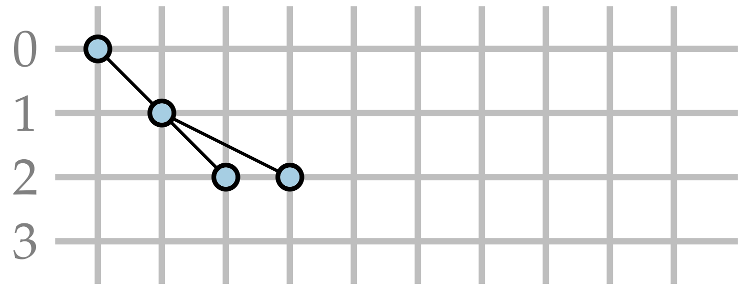
First Grid Layout of Binary Trees

1. Choose y-coordinates: $y(u) = \text{depth}(u)$



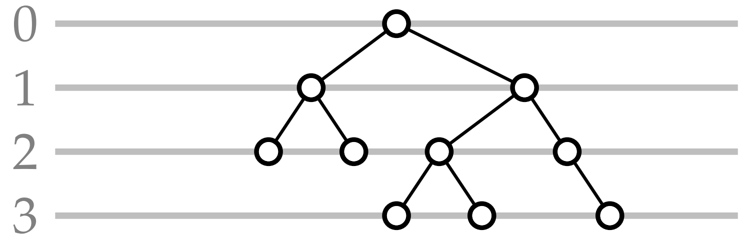
2. Choose x-coordinates:



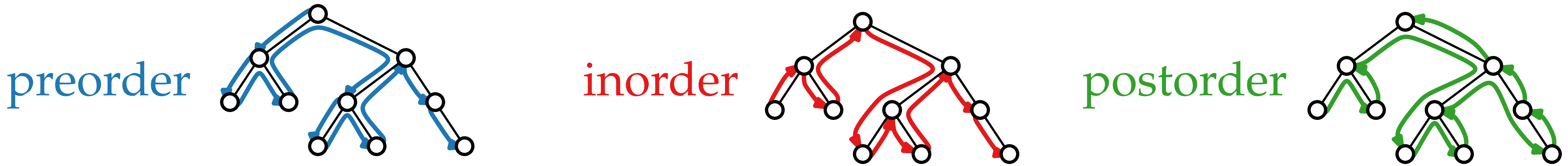


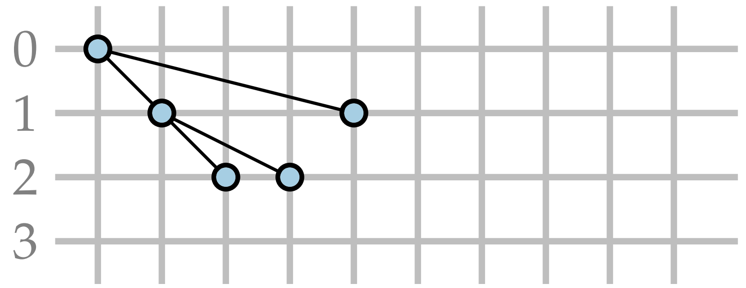
First Grid Layout of Binary Trees

1. Choose y-coordinates: $y(u) =$depth(u)



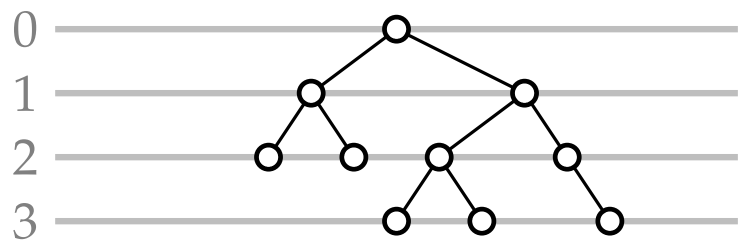
2. Choose x-coordinates:



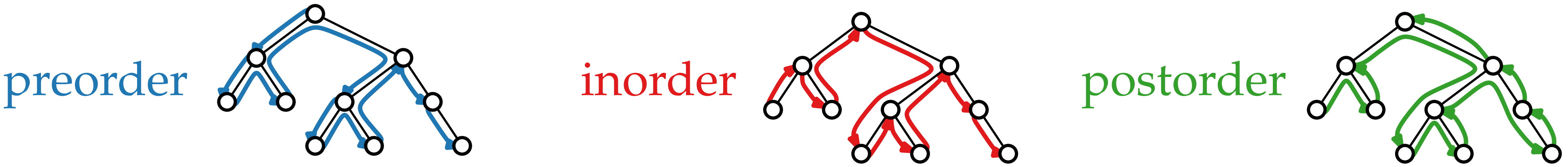


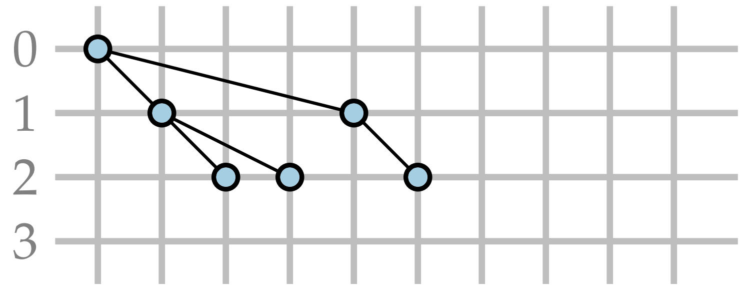
First Grid Layout of Binary Trees

1. Choose y-coordinates: $y(u)$ =depth(u)



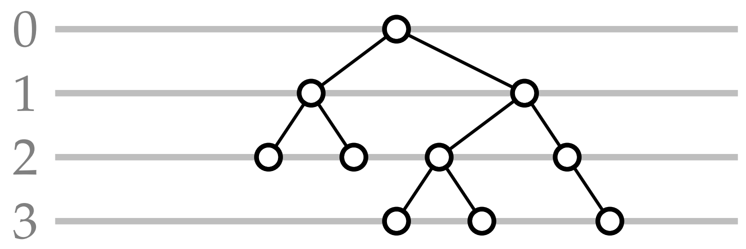
2. Choose x-coordinates:



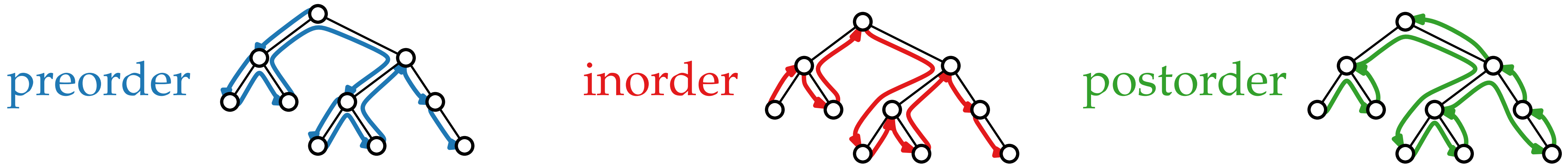


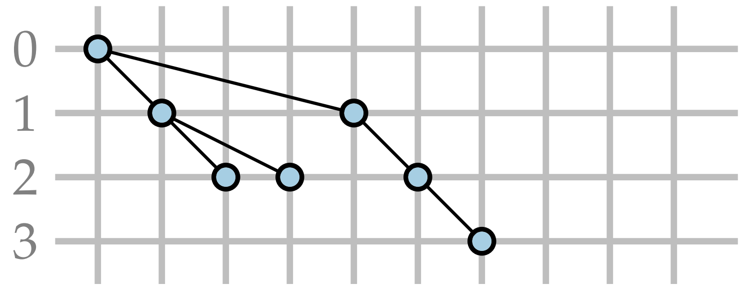
First Grid Layout of Binary Trees

1. Choose y-coordinates: $y(u) = \text{depth}(u)$



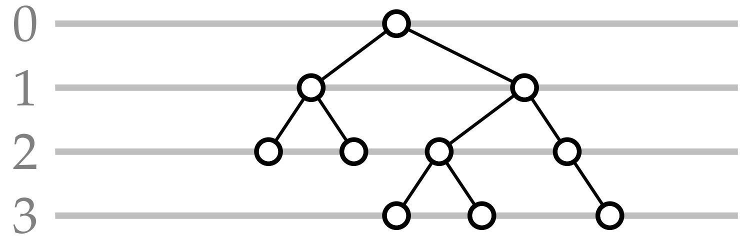
2. Choose x-coordinates:



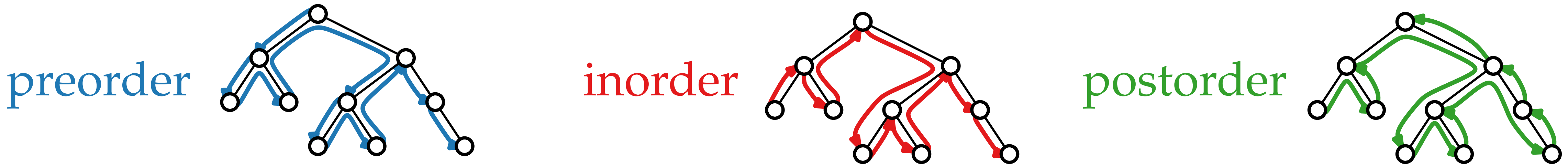


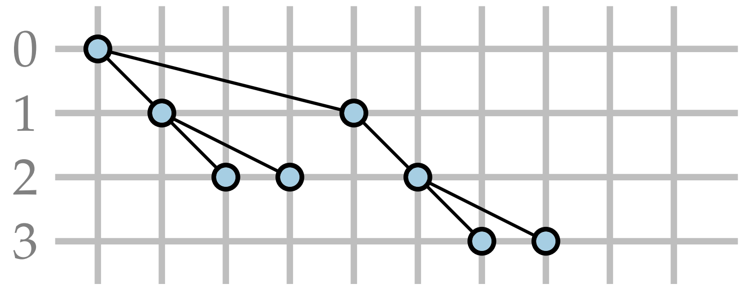
First Grid Layout of Binary Trees

1. Choose y-coordinates: $y(u) =$depth(u)



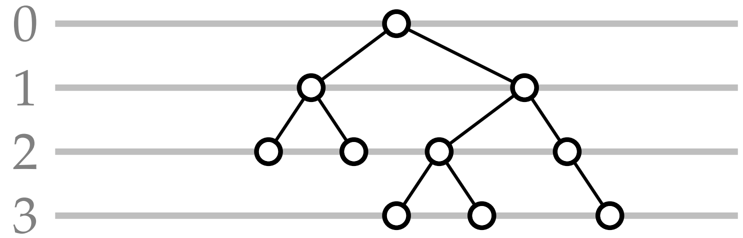
2. Choose x-coordinates:



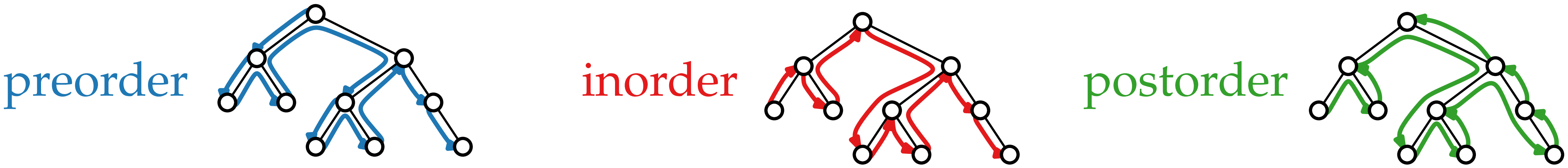


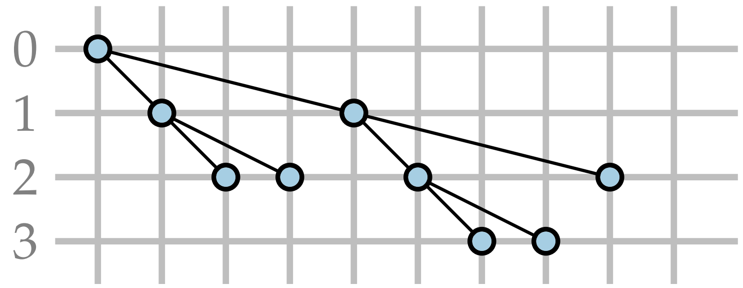
First Grid Layout of Binary Trees

1. Choose y-coordinates: $y(u) =$depth(u)



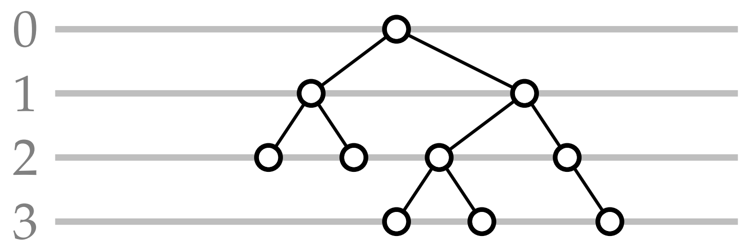
2. Choose x-coordinates:





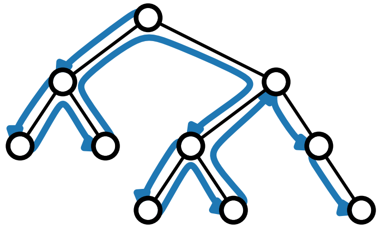
First Grid Layout of Binary Trees

1. Choose y-coordinates: $y(u) = \text{depth}(u)$

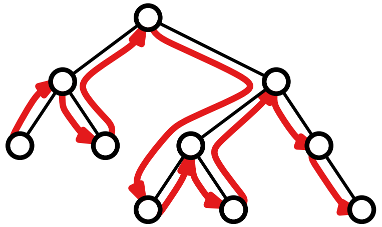


2. Choose x-coordinates:

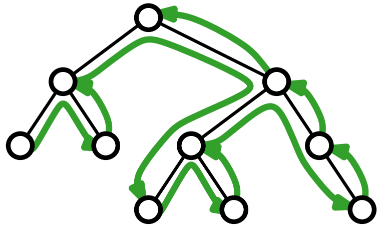
preorder

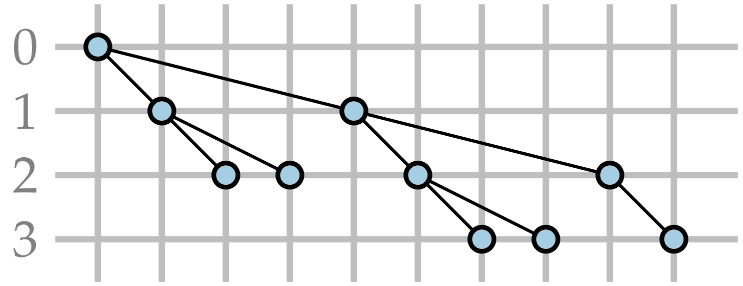


inorder



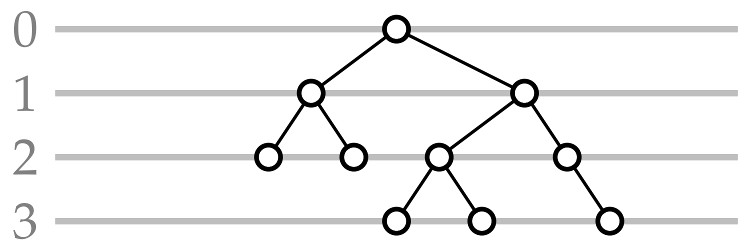
postorder



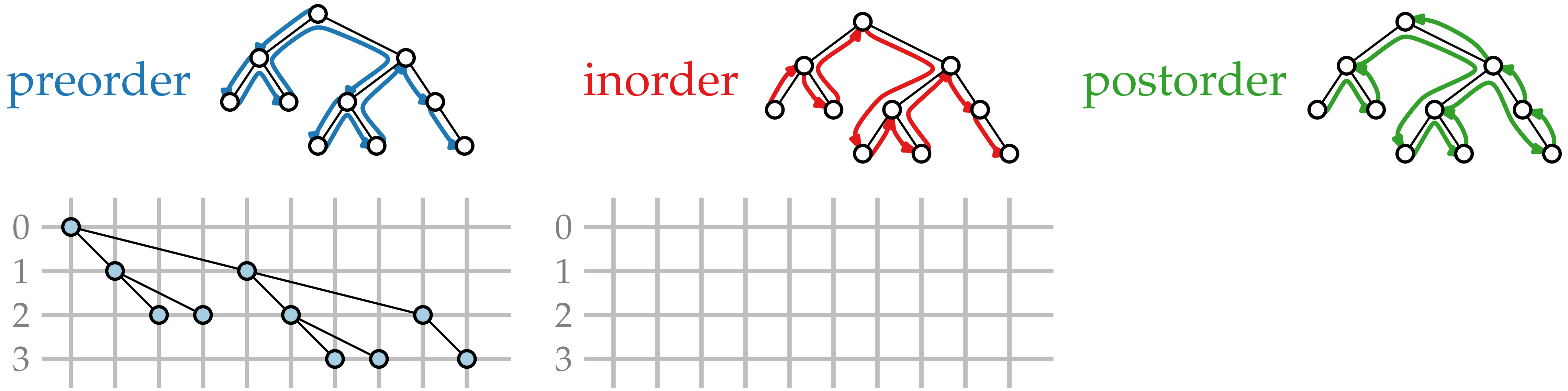


First Grid Layout of Binary Trees

1. Choose y-coordinates: $y(u) =$depth(u)

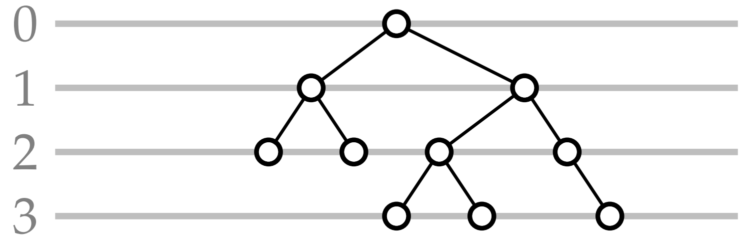


2. Choose x-coordinates:

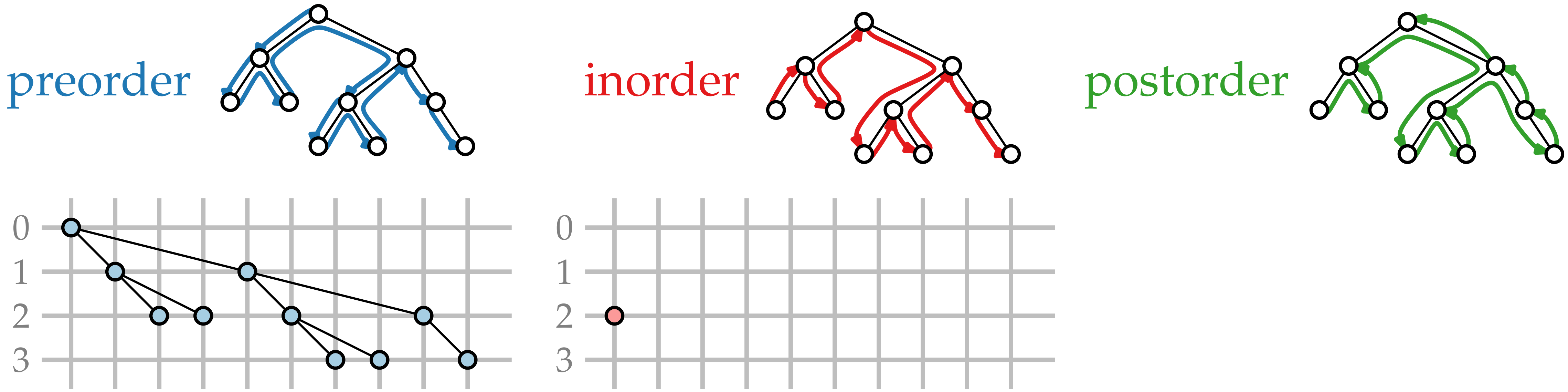


First Grid Layout of Binary Trees

1. Choose y-coordinates: $y(u) = \text{depth}(u)$

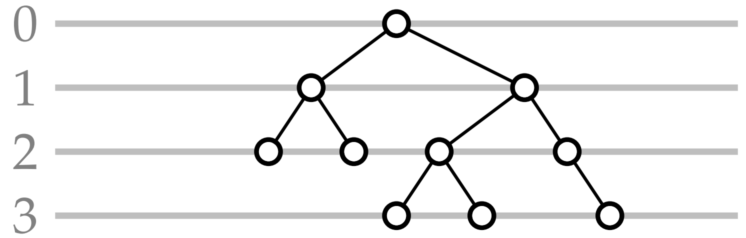


2. Choose x-coordinates:

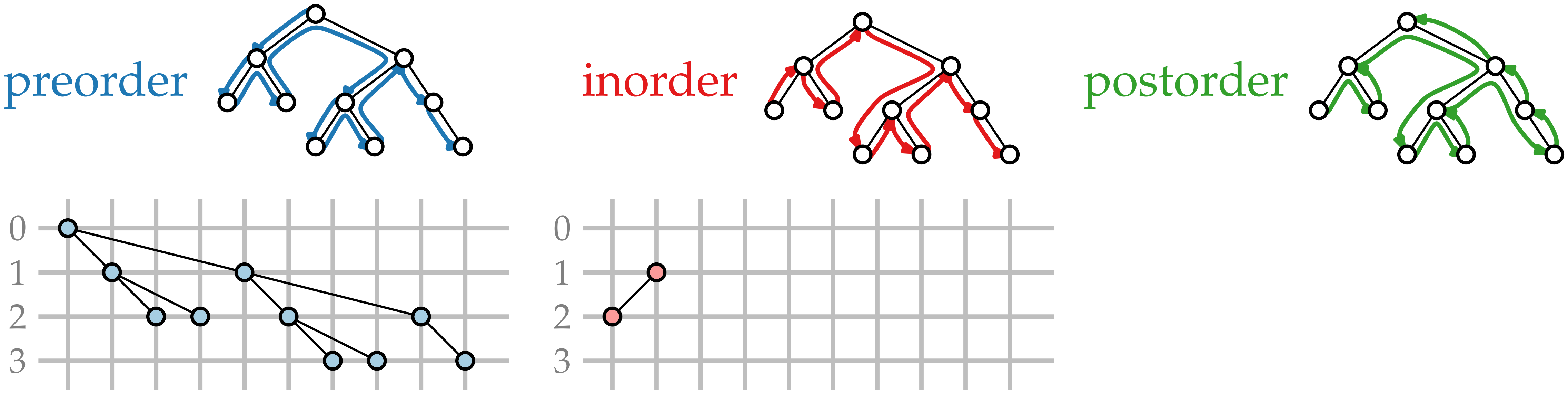


First Grid Layout of Binary Trees

1. Choose y-coordinates: $y(u)$ =depth(u)

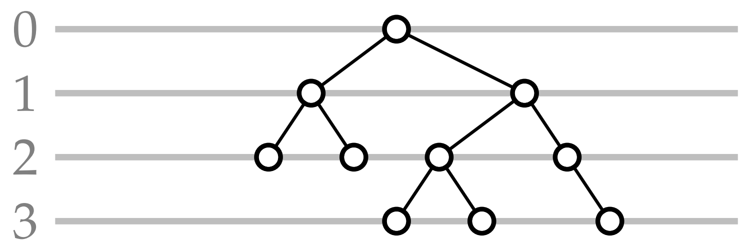


2. Choose x-coordinates:

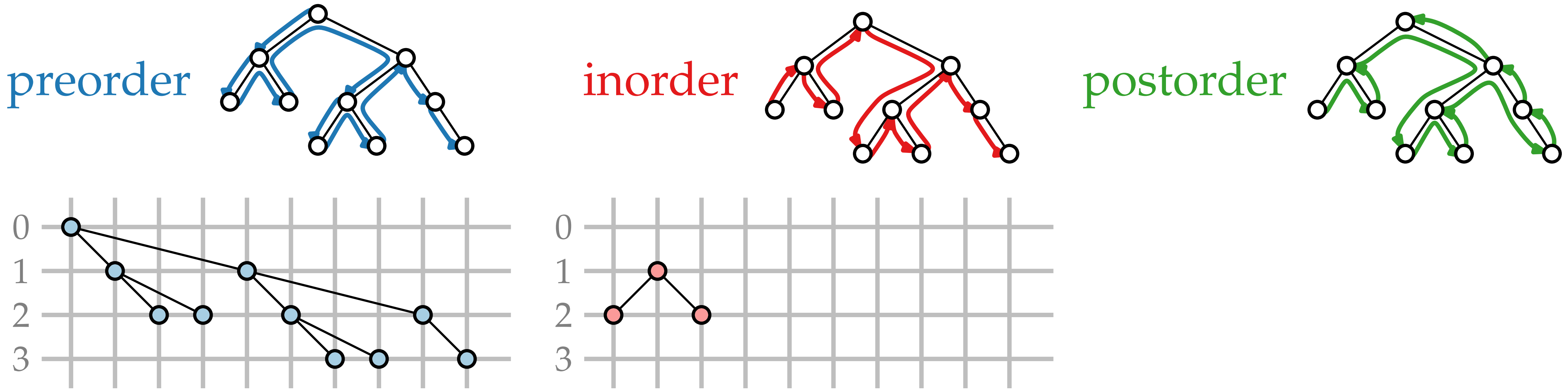


First Grid Layout of Binary Trees

1. Choose y-coordinates: $y(u) =$depth(u)

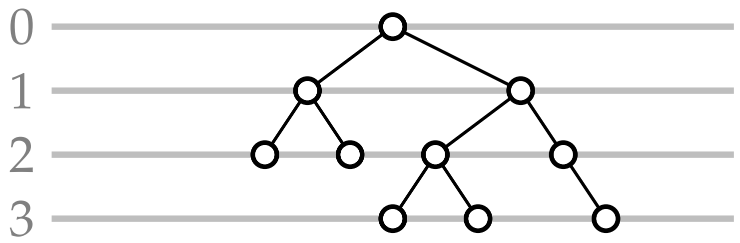


2. Choose x-coordinates:



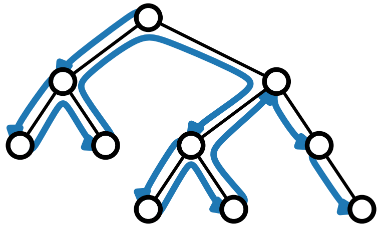
First Grid Layout of Binary Trees

1. Choose y-coordinates: $y(u) = \text{depth}(u)$

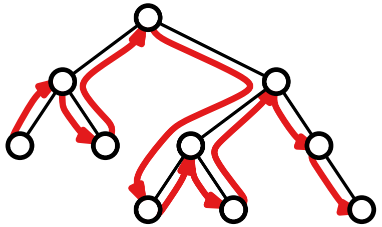


2. Choose x-coordinates:

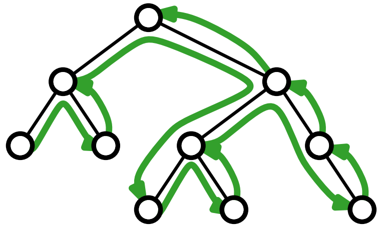
preorder

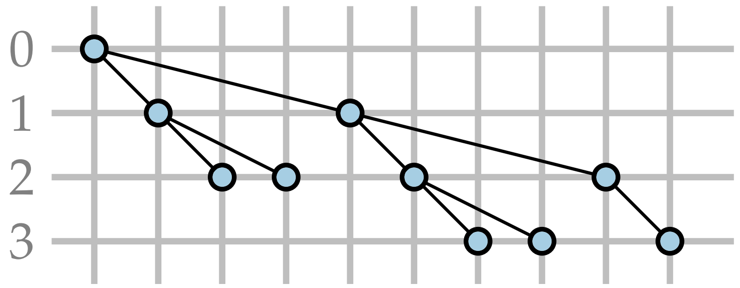


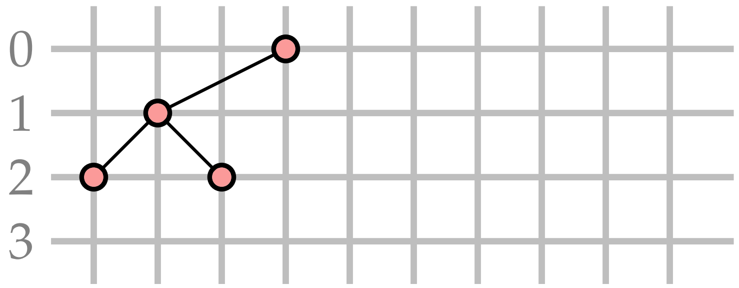
inorder



postorder

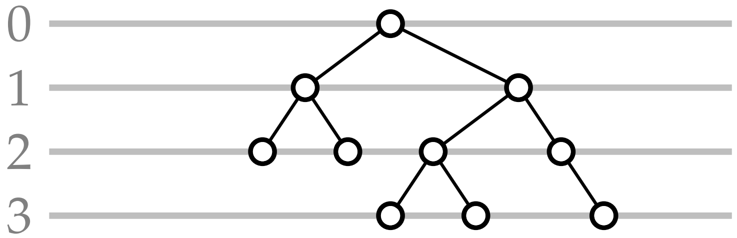




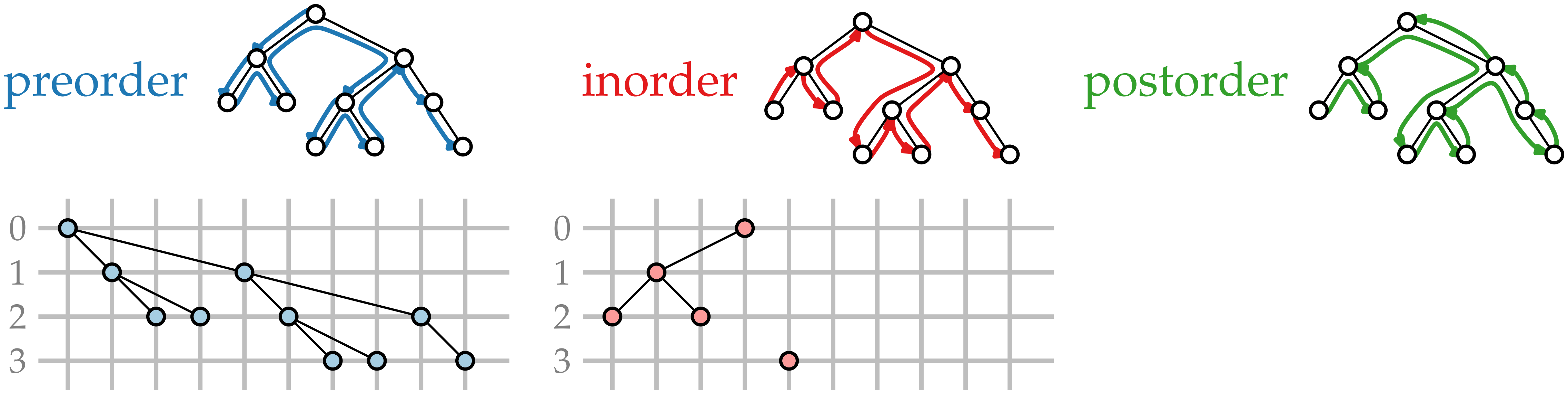


First Grid Layout of Binary Trees

1. Choose y-coordinates: $y(u) = \text{depth}(u)$

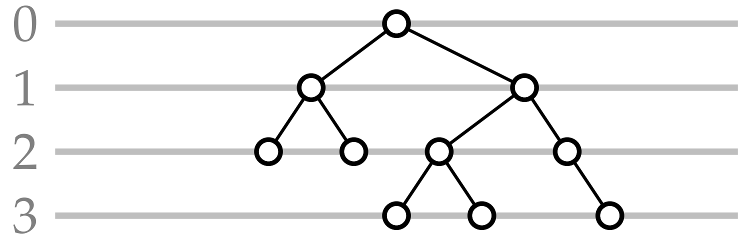


2. Choose x-coordinates:



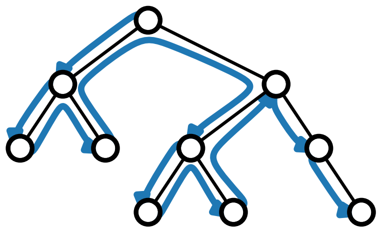
First Grid Layout of Binary Trees

1. Choose y-coordinates: $y(u)$ =depth(u)

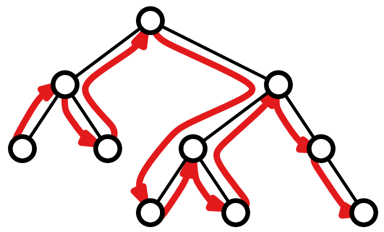


2. Choose x-coordinates:

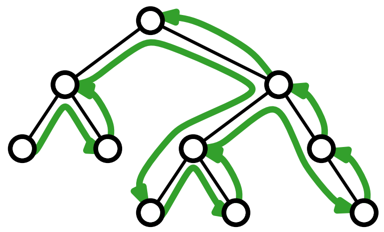
preorder

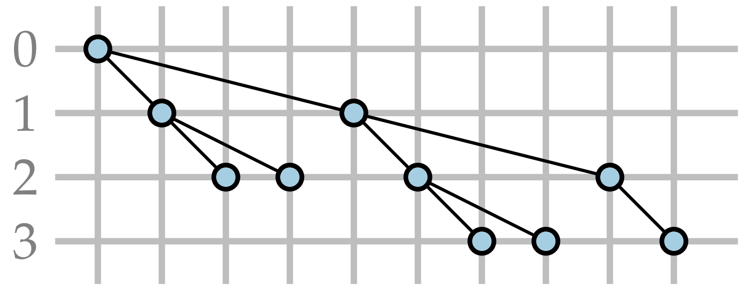


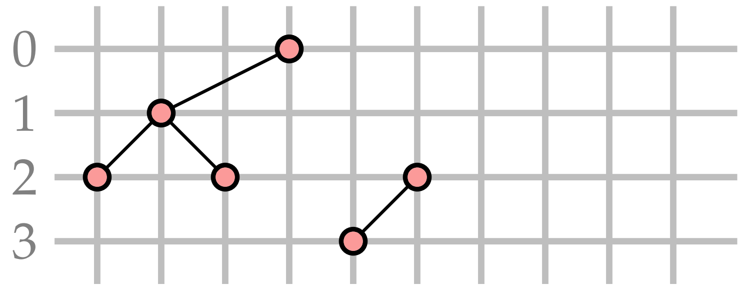
inorder



postorder

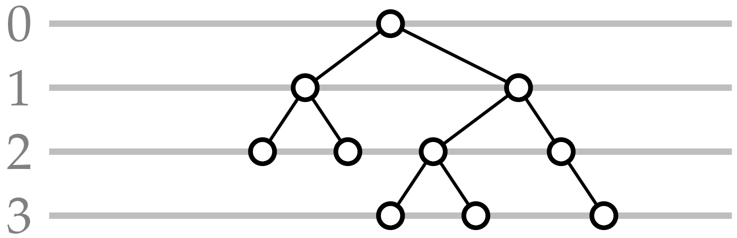






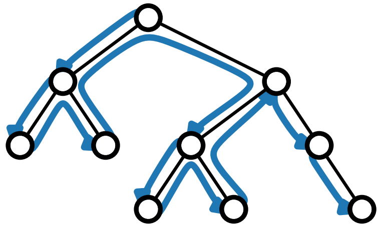
First Grid Layout of Binary Trees

1. Choose y-coordinates: $y(u) = \text{depth}(u)$

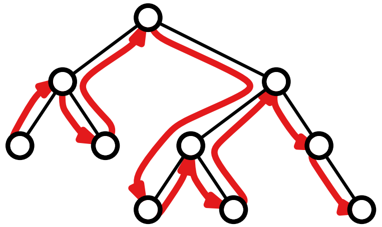


2. Choose x-coordinates:

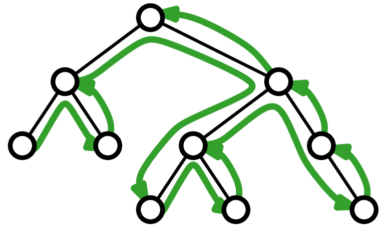
preorder

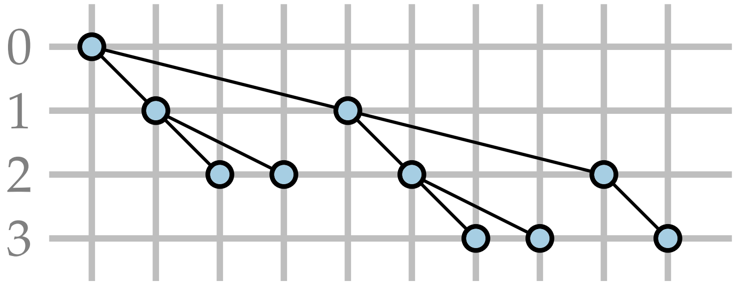


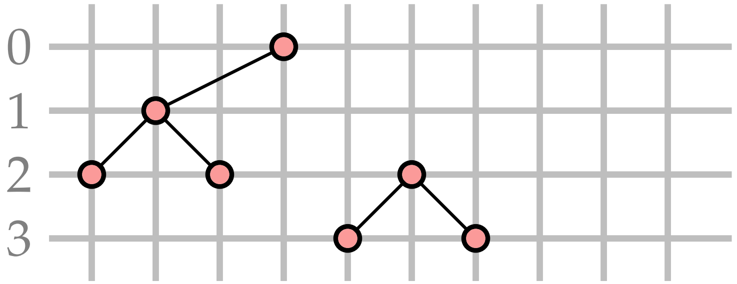
inorder



postorder

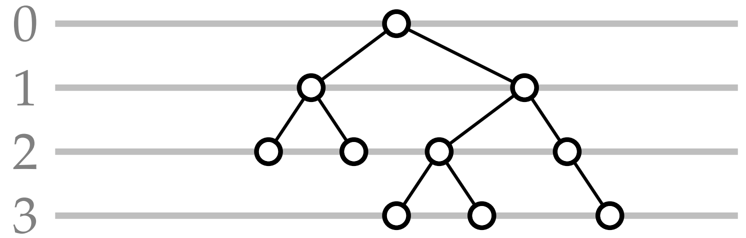




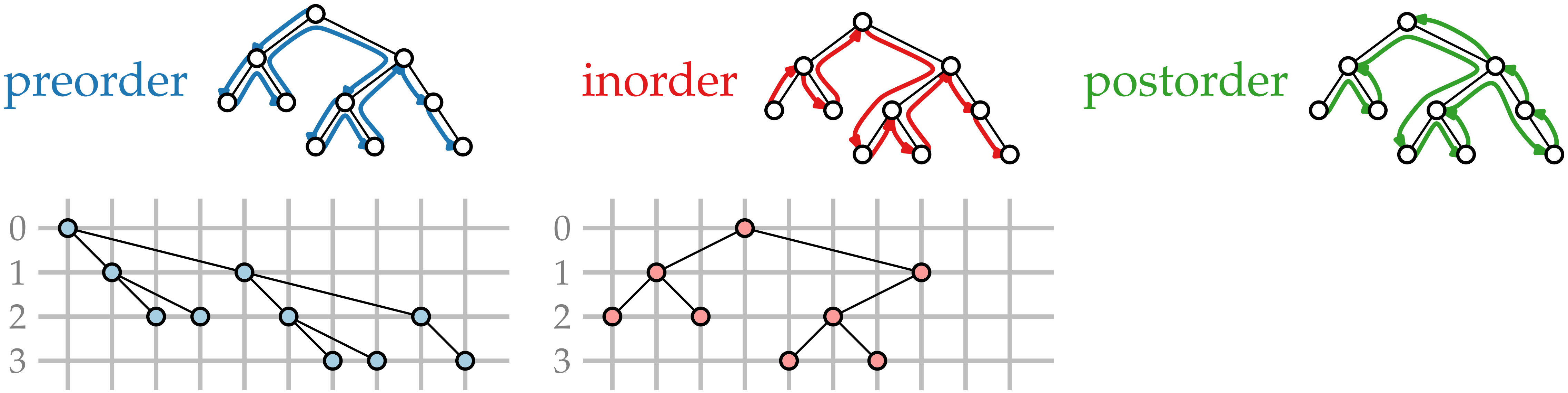


First Grid Layout of Binary Trees

1. Choose y-coordinates: $y(u) =$depth(u)

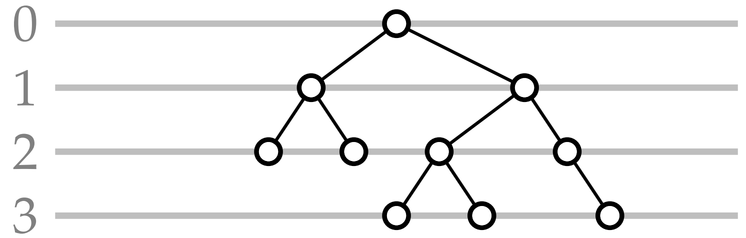


2. Choose x-coordinates:

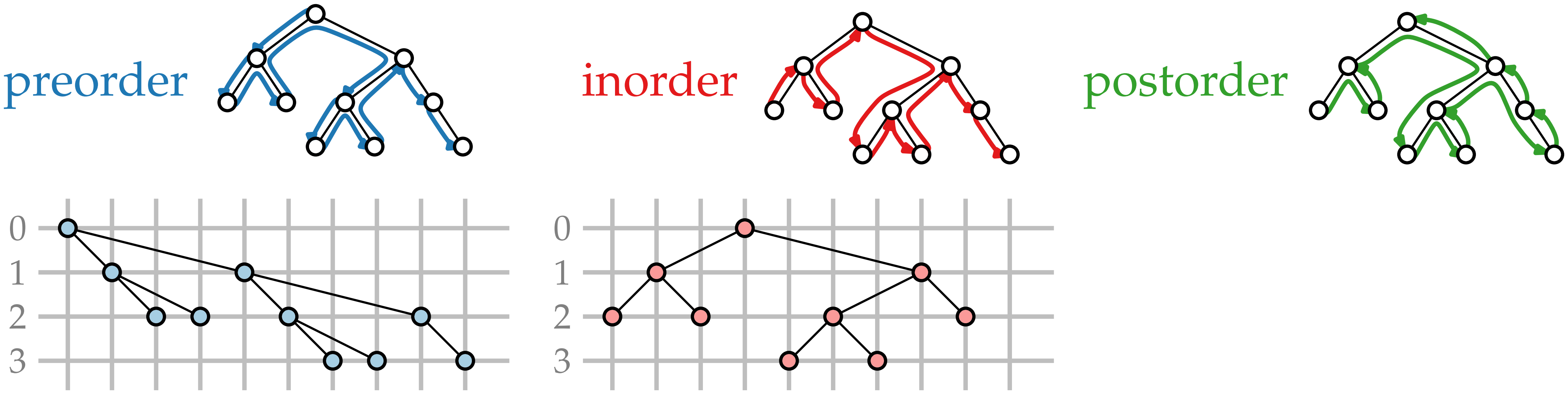


First Grid Layout of Binary Trees

1. Choose y-coordinates: $y(u) = \text{depth}(u)$

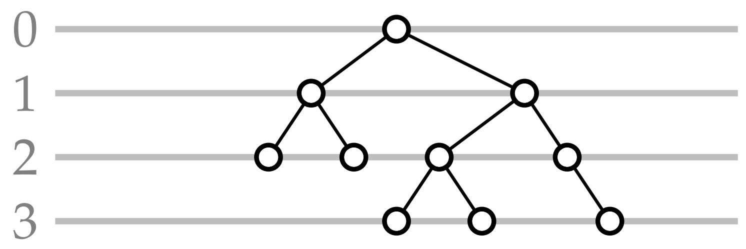


2. Choose x-coordinates:

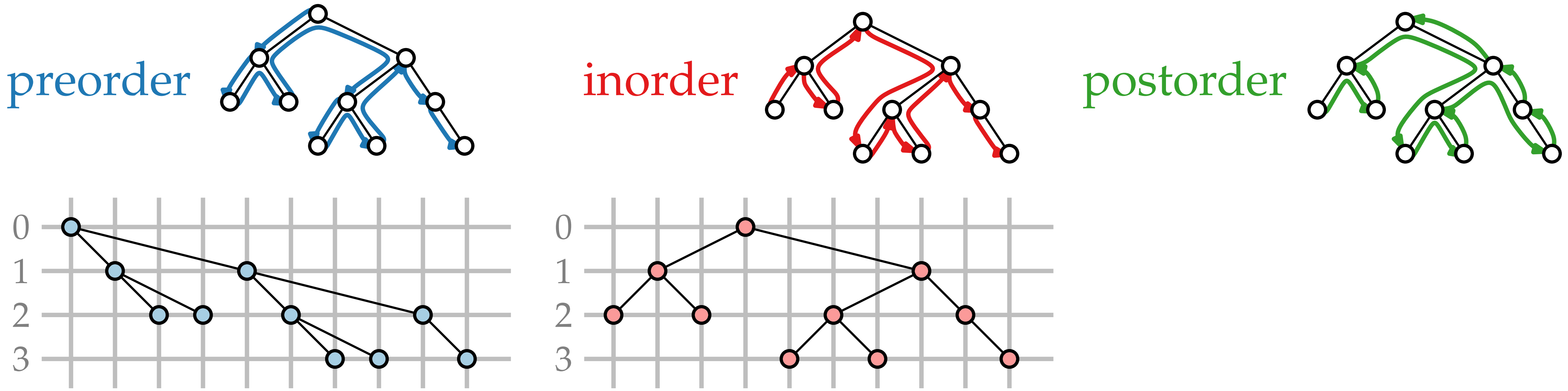


First Grid Layout of Binary Trees

1. Choose y-coordinates: $y(u) =$depth(u)

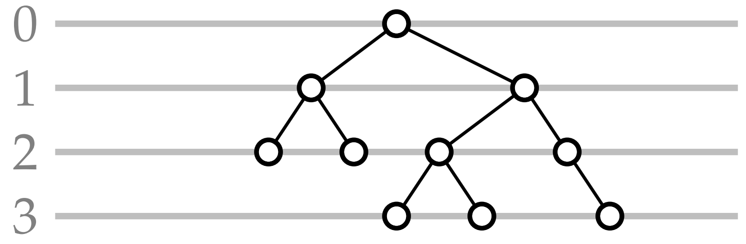


2. Choose x-coordinates:

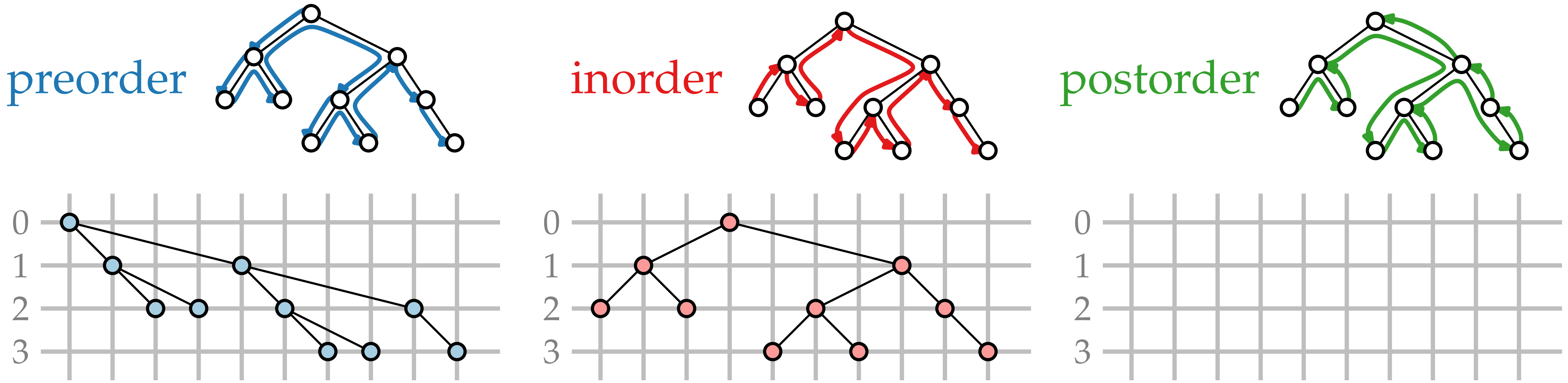


First Grid Layout of Binary Trees

1. Choose y-coordinates: $y(u) =$depth(u)

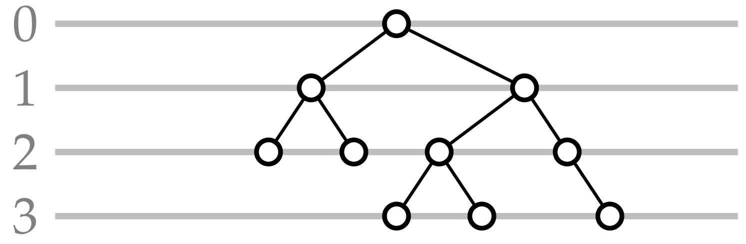


2. Choose x-coordinates:

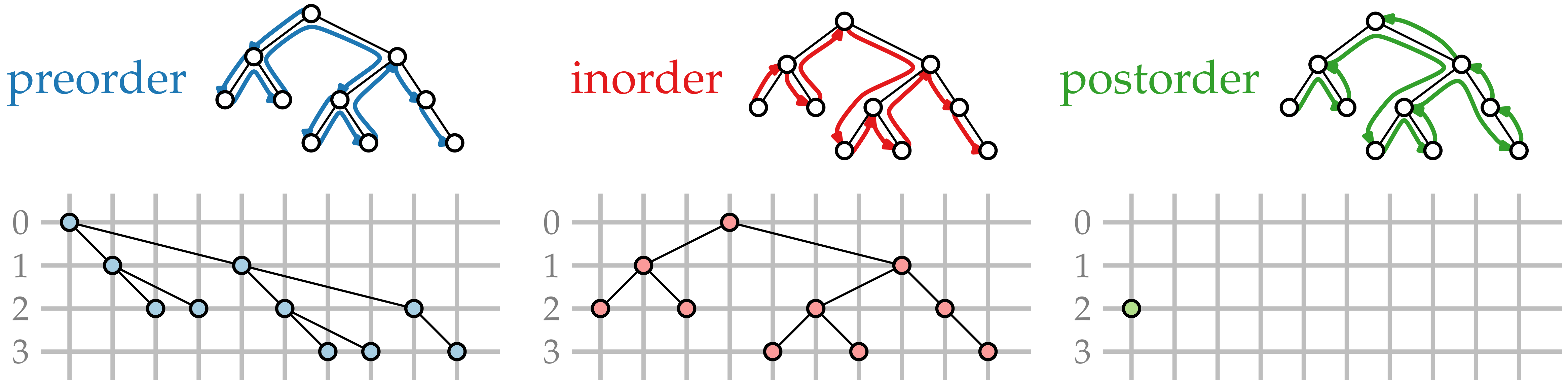


First Grid Layout of Binary Trees

1. Choose y-coordinates: $y(u) = \text{depth}(u)$

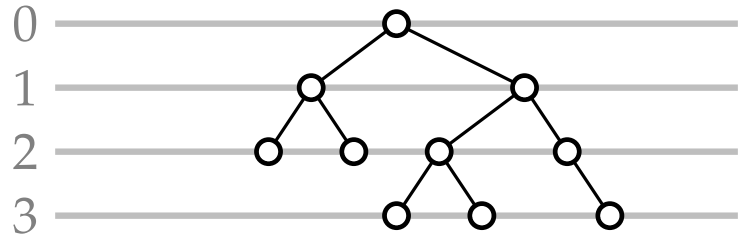


2. Choose x-coordinates:

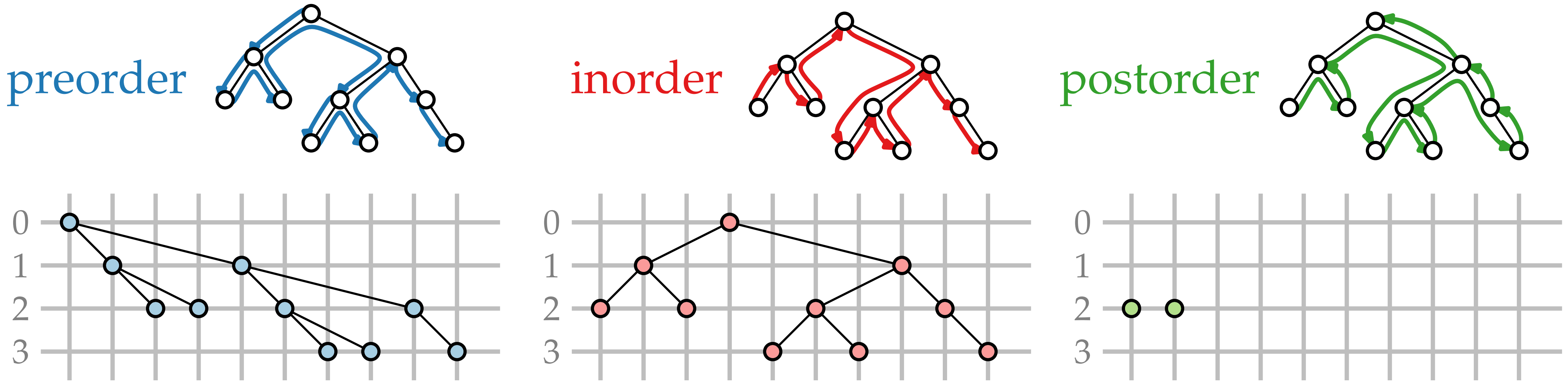


First Grid Layout of Binary Trees

1. Choose y-coordinates: $y(u) = \text{depth}(u)$

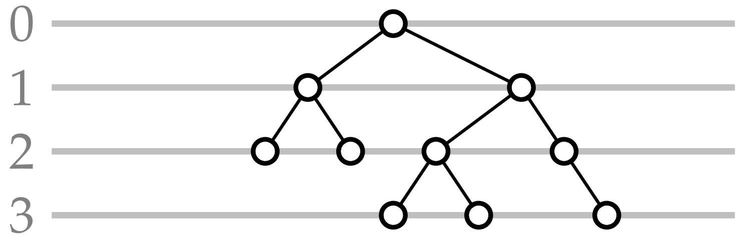


2. Choose x-coordinates:



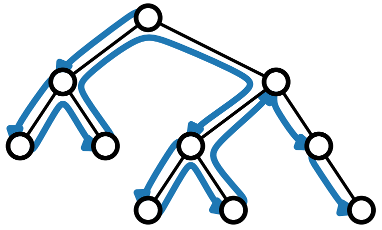
First Grid Layout of Binary Trees

1. Choose y-coordinates: $y(u) = \text{depth}(u)$

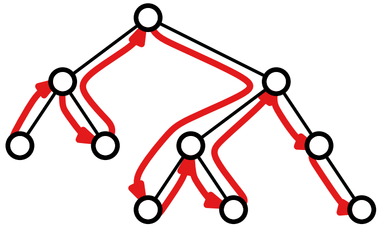


2. Choose x-coordinates:

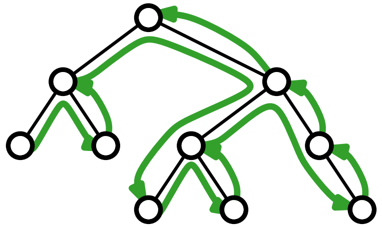
preorder

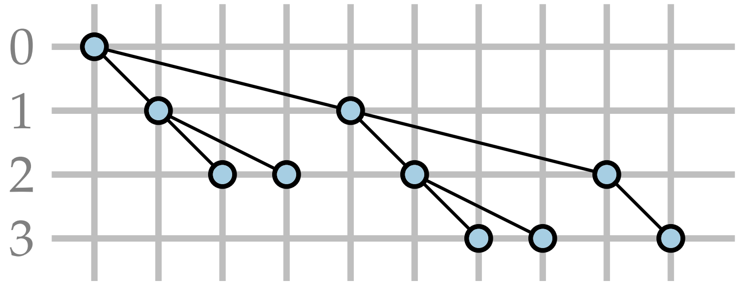


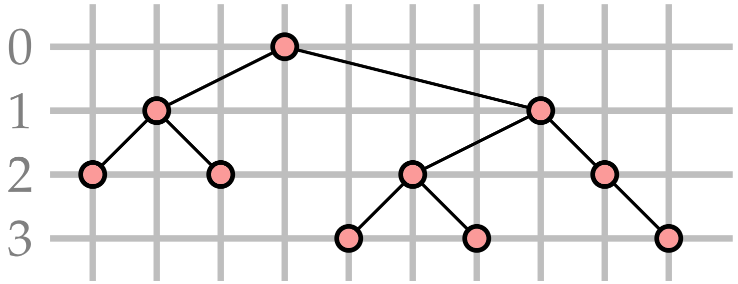
inorder

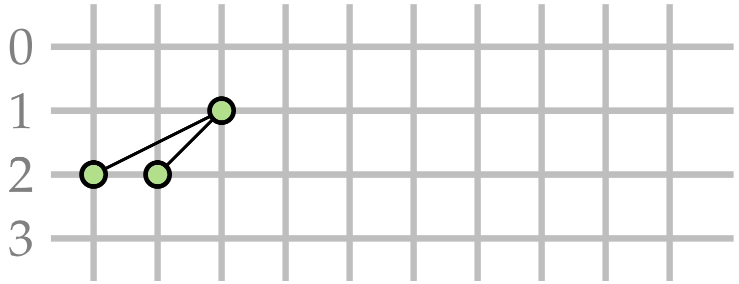


postorder



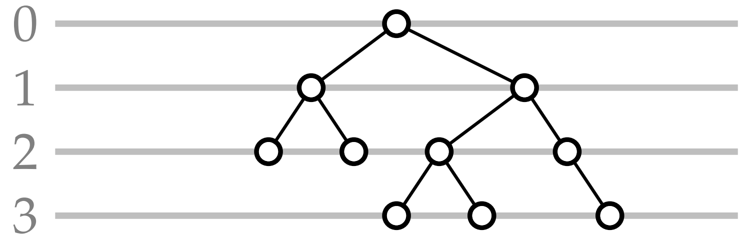




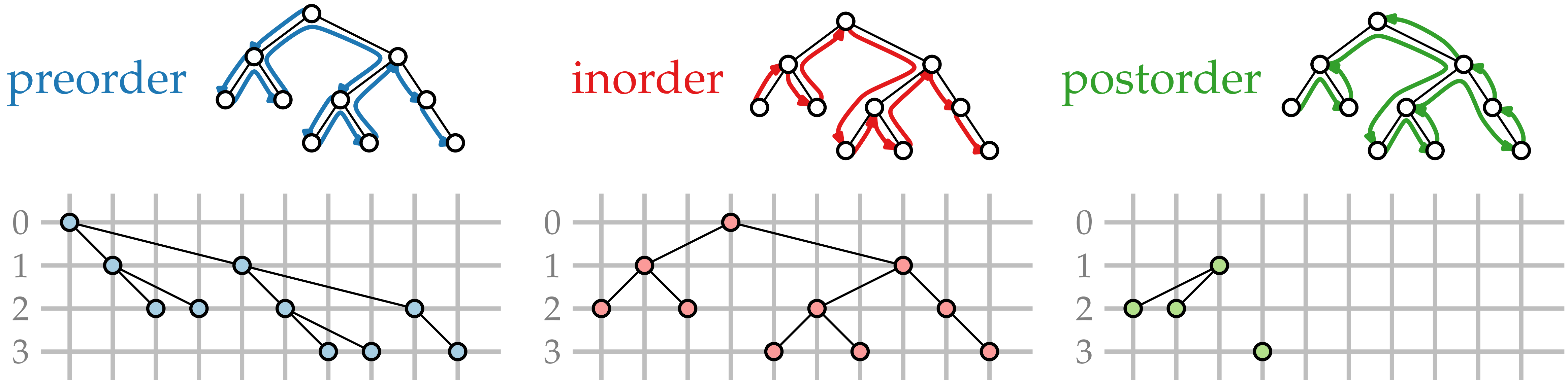


First Grid Layout of Binary Trees

1. Choose y-coordinates: $y(u) =$depth(u)

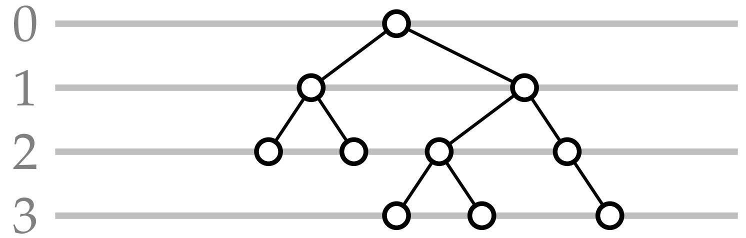


2. Choose x-coordinates:

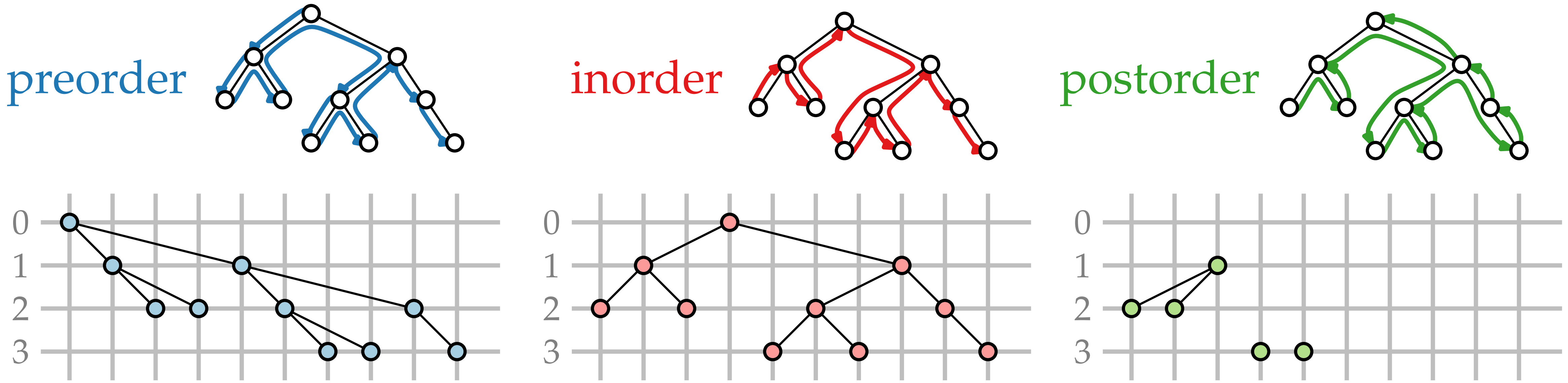


First Grid Layout of Binary Trees

1. Choose y-coordinates: $y(u) =$depth(u)

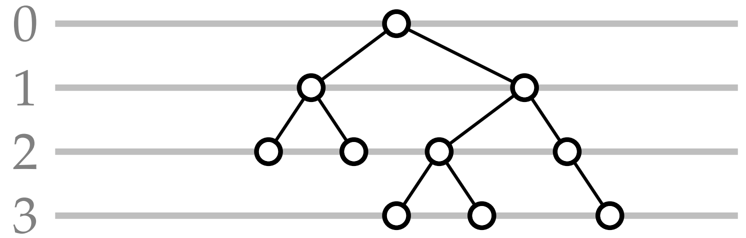


2. Choose x-coordinates:

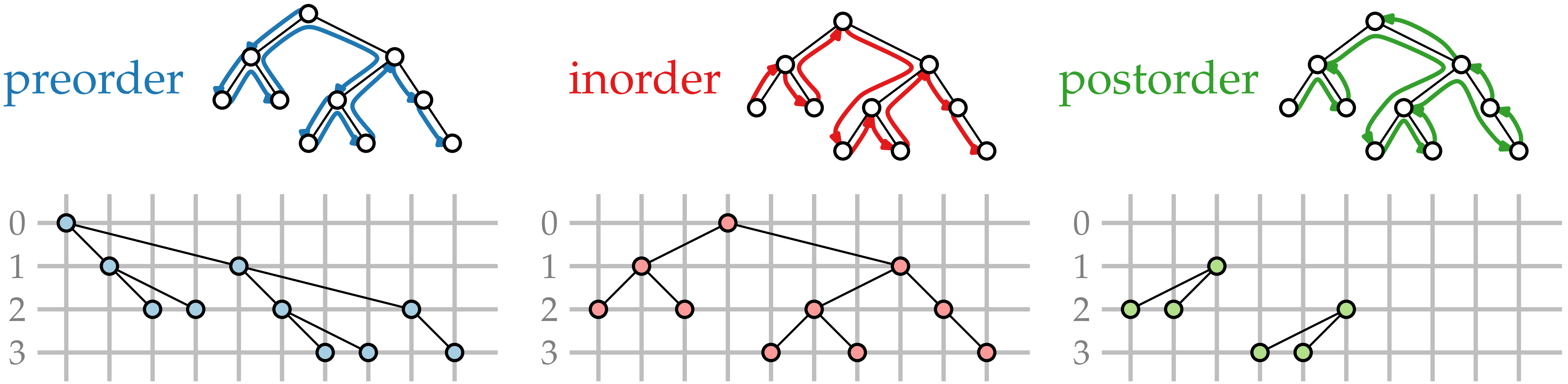


First Grid Layout of Binary Trees

1. Choose y-coordinates: $y(u) = \text{depth}(u)$

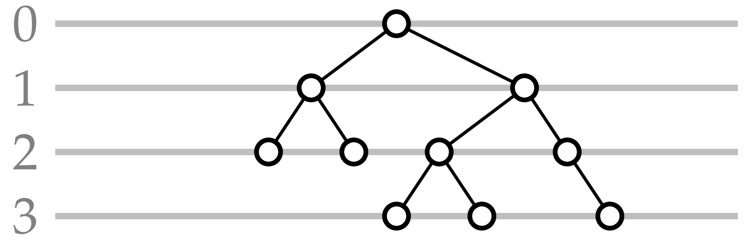


2. Choose x-coordinates:



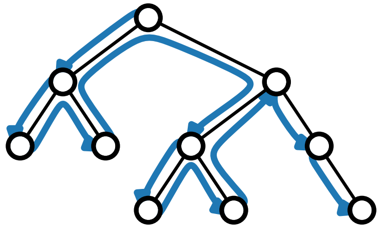
First Grid Layout of Binary Trees

1. Choose y-coordinates: $y(u) = \text{depth}(u)$

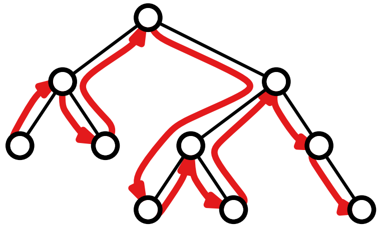


2. Choose x-coordinates:

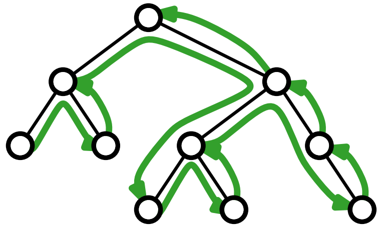
preorder

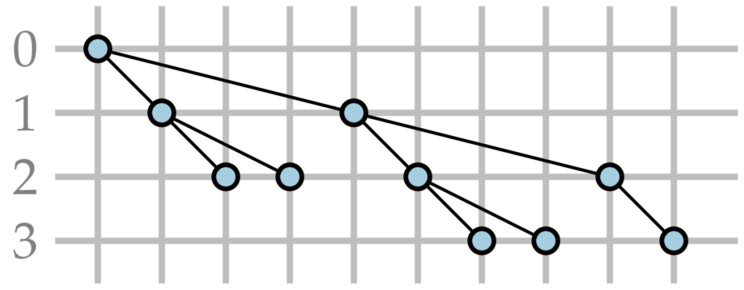


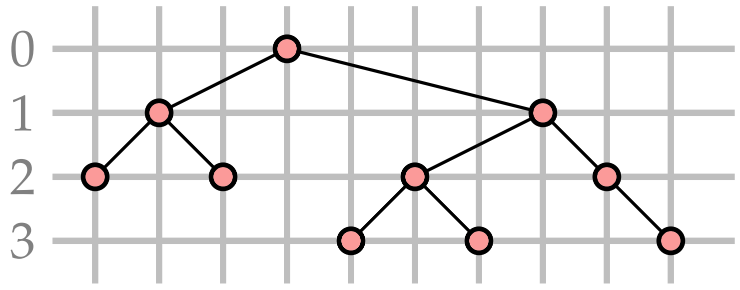
inorder

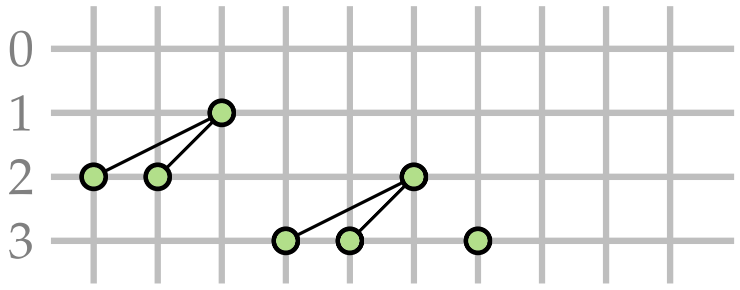


postorder



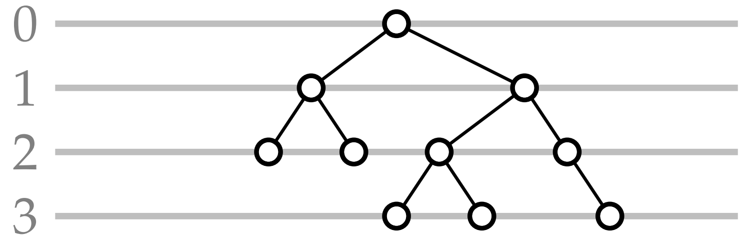




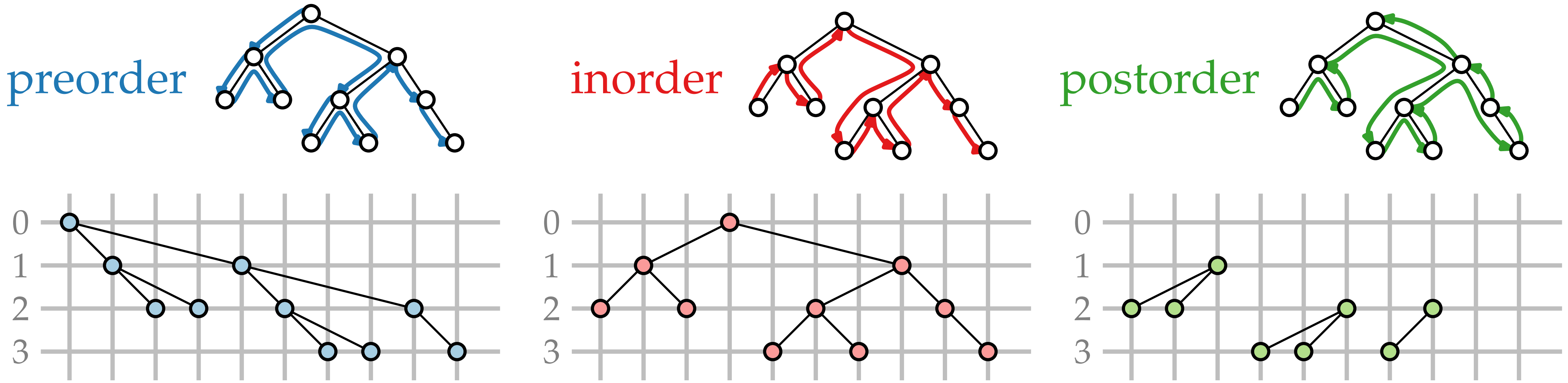


First Grid Layout of Binary Trees

1. Choose y-coordinates: $y(u) =$depth(u)

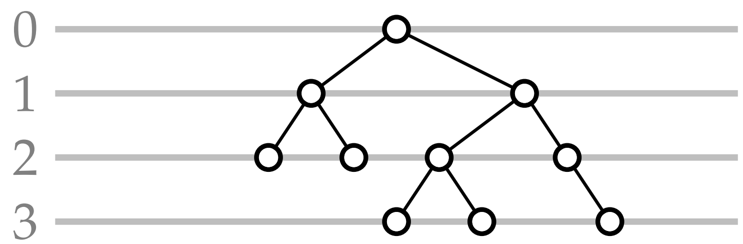


2. Choose x-coordinates:



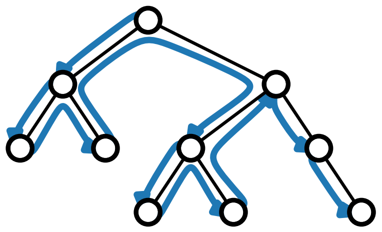
First Grid Layout of Binary Trees

1. Choose y-coordinates: $y(u)$ =depth(u)

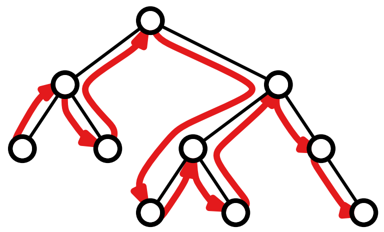


2. Choose x-coordinates:

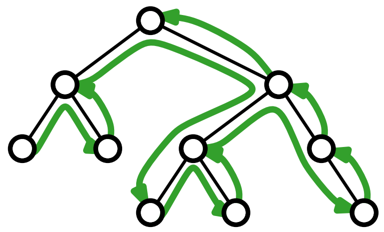
preorder

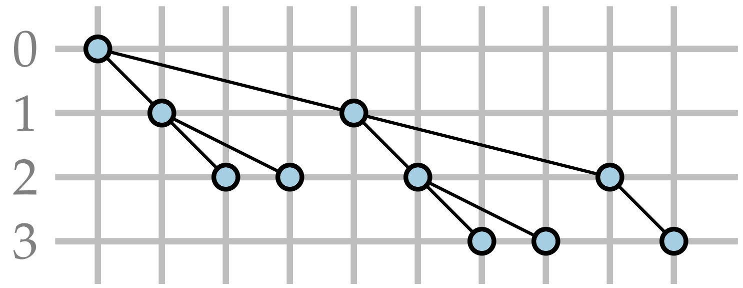


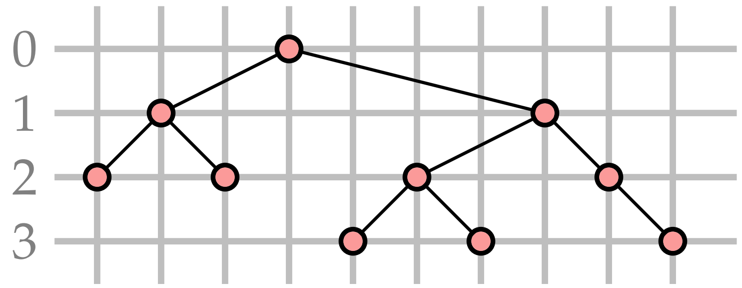
inorder

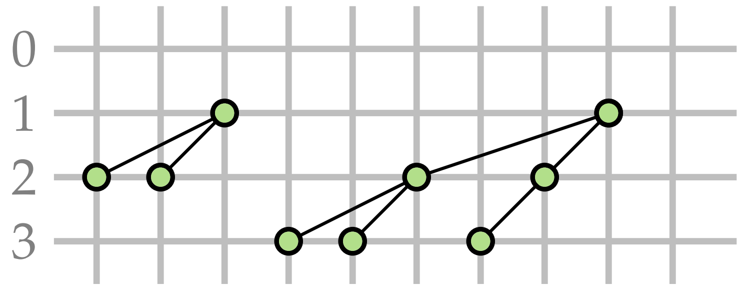


postorder



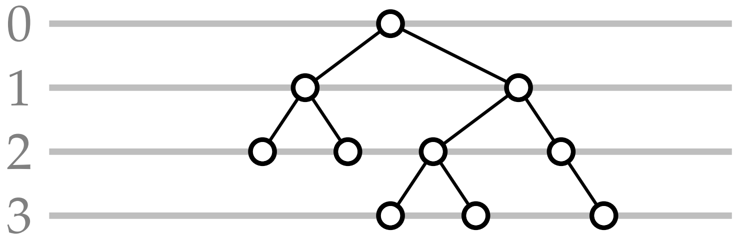






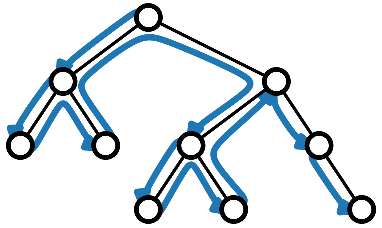
First Grid Layout of Binary Trees

1. Choose y-coordinates: $y(u) = \text{depth}(u)$

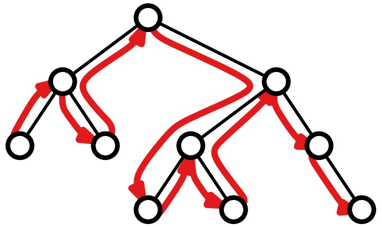


2. Choose x-coordinates:

preorder



inorder



postorder