Enumeration of unlabelled chordal graphs with bounded tree-width

Jordi Castellví (CRM)

Work in collaboration with Michael Drmota
and Clément Requilé

CENTRE DE RECERCA MATEMÀTICA

EXCELENCIA MARÍA DE MAEZTU 2022 - 2025

RandNET Workshop - Wien

Introduction

How to build a tree?

Introduction

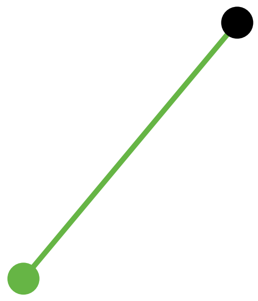
How to build a tree?

Iteratively add a new vertex connected to an existing vertex.

Introduction

How to build a tree?

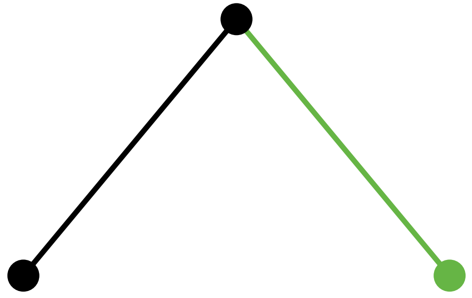
Iteratively add a new vertex connected to an existing vertex.



Introduction

How to build a tree?

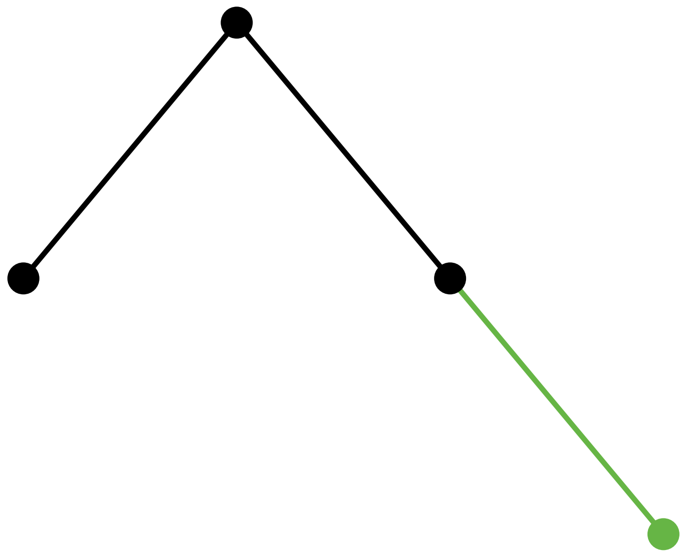
Iteratively add a new vertex connected to an existing vertex.



Introduction

How to build a tree?

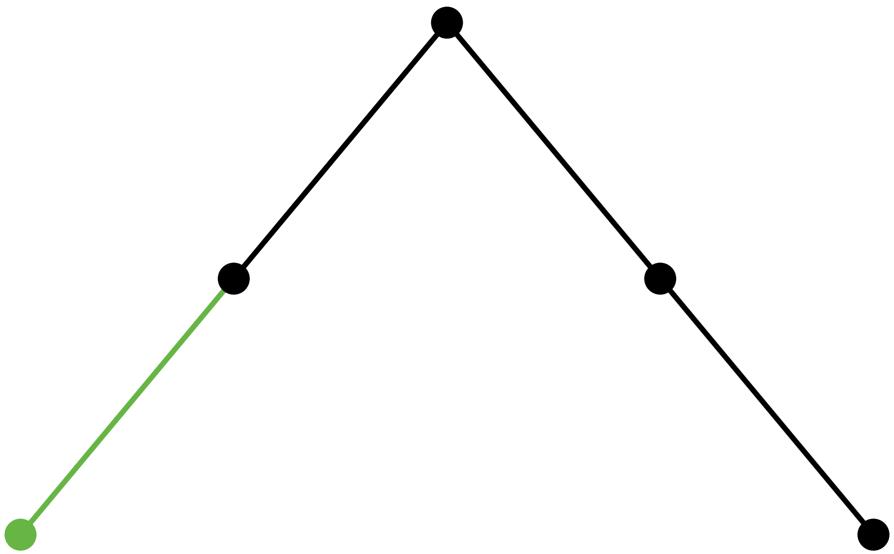
Iteratively add a new vertex connected to an existing vertex.



Introduction

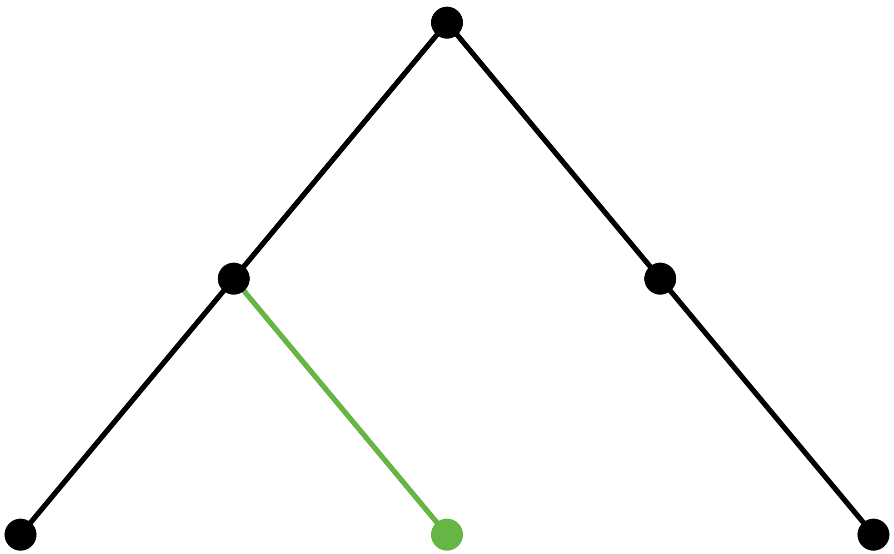
How to build a tree?

Iteratively add a new vertex connected to an existing vertex.



How to build a tree?

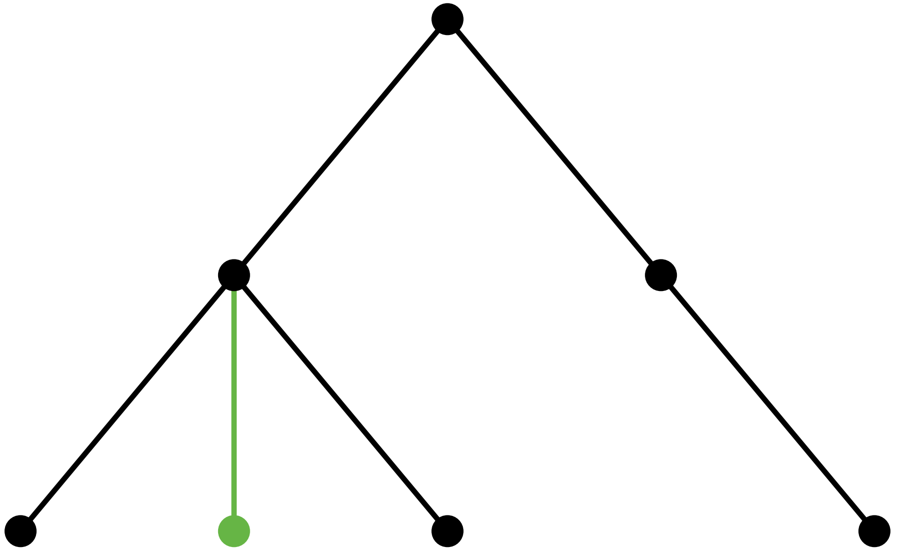
Iteratively add a new vertex connected to an existing vertex.



Introduction

How to build a tree?

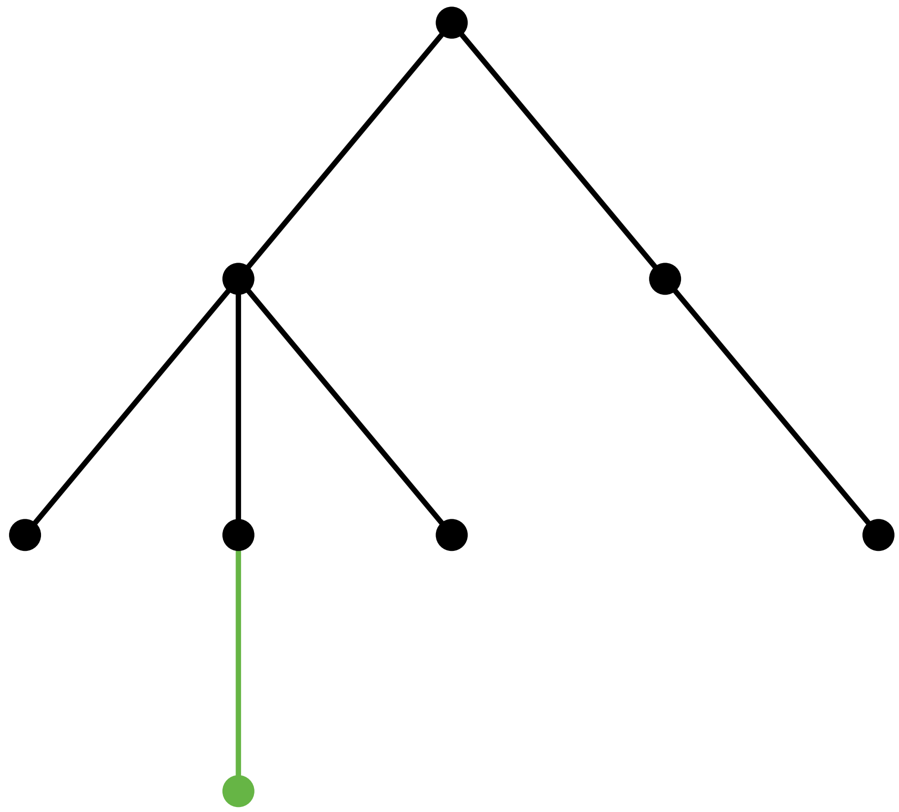
Iteratively add a new vertex connected to an existing vertex.



Introduction

How to build a tree?

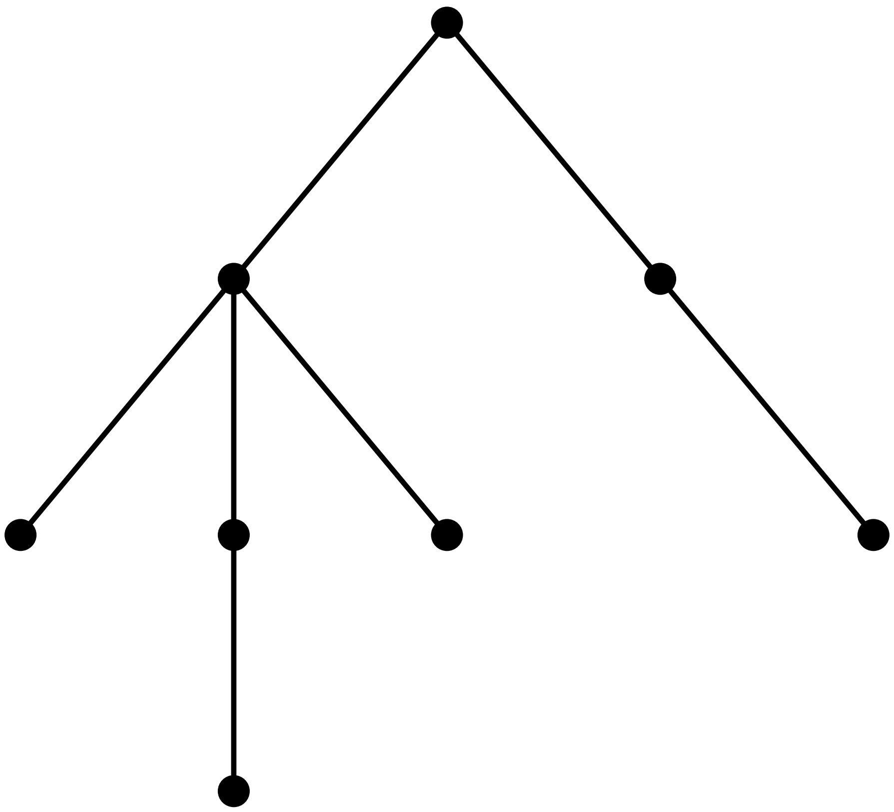
Iteratively add a new vertex connected to an existing vertex.



Introduction

How to build a tree?

Iteratively add a new vertex connected to an existing vertex.



Introduction

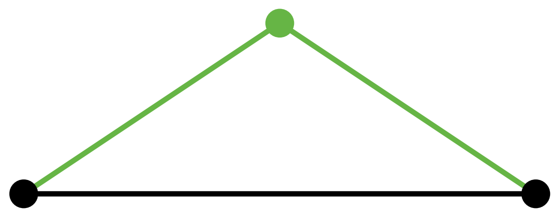
Iteratively add a new vertex connected to the vertices of an existing **edge**.

Introduction

Iteratively add a new vertex connected to the vertices of an existing **edge**.

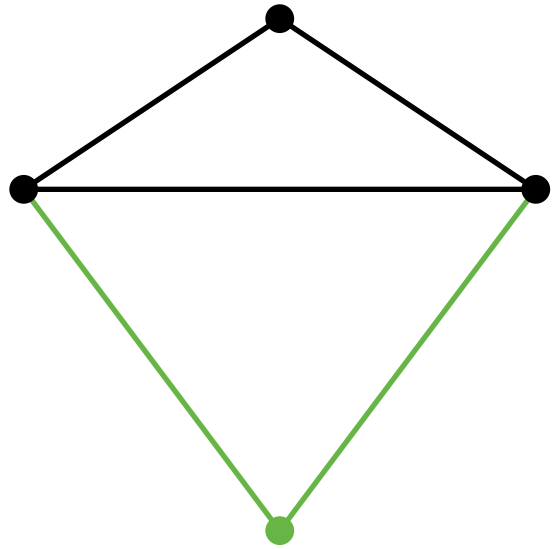
Introduction

Iteratively add a new vertex connected to the vertices of an existing **edge**.



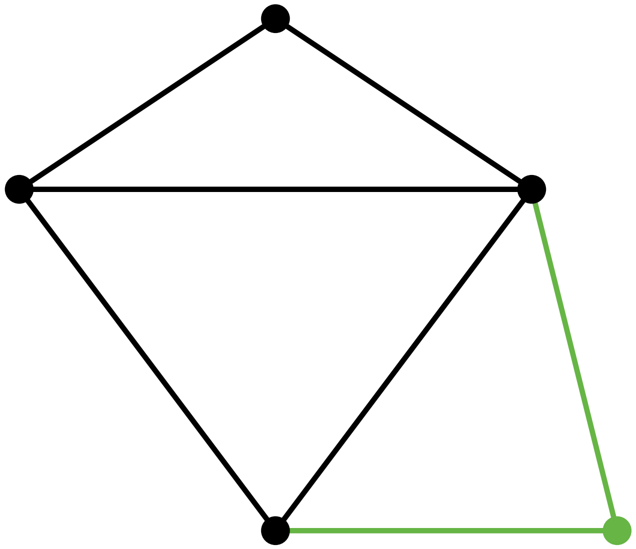
Introduction

Iterately add a new vertex connected to the vertices of an existing **edge**.



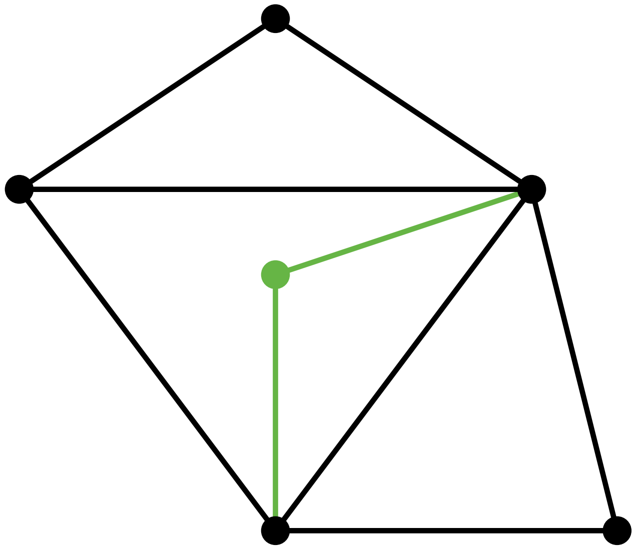
Introduction

Iteratively add a new vertex connected to the vertices of an existing **edge**.



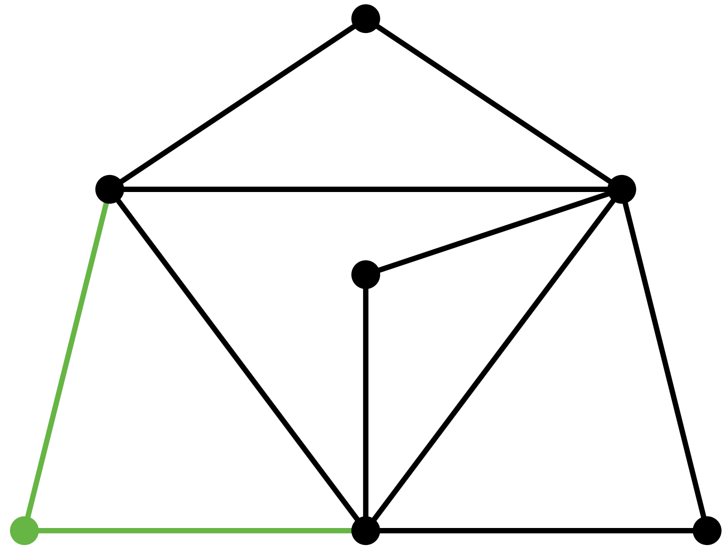
Introduction

Iterativelyadd a new vertex connected to the vertices of an existing **edge**.



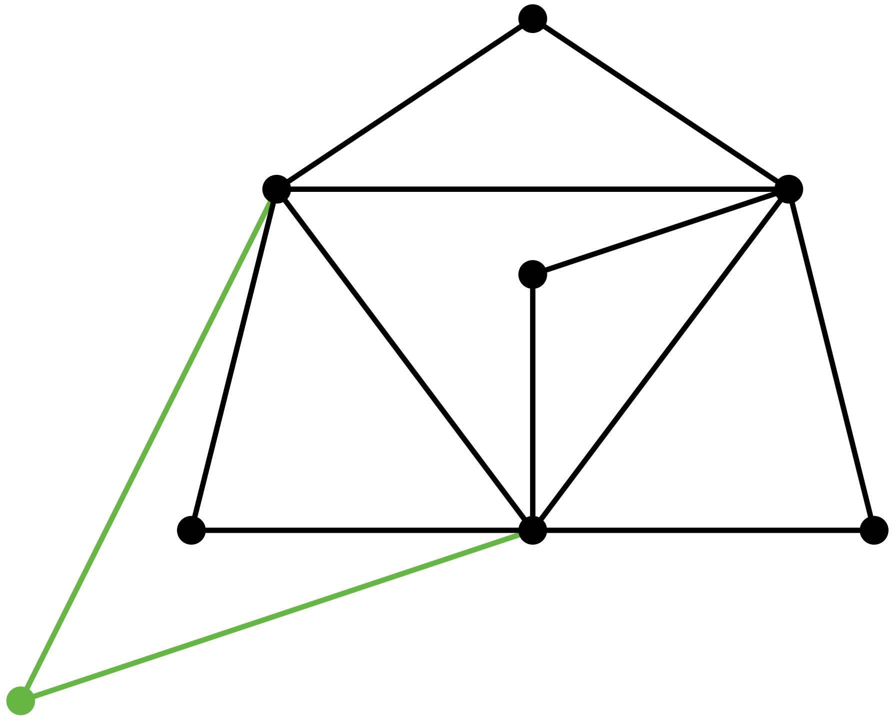
Introduction

Iterately add a new vertex connected to the vertices of an existing **edge**.



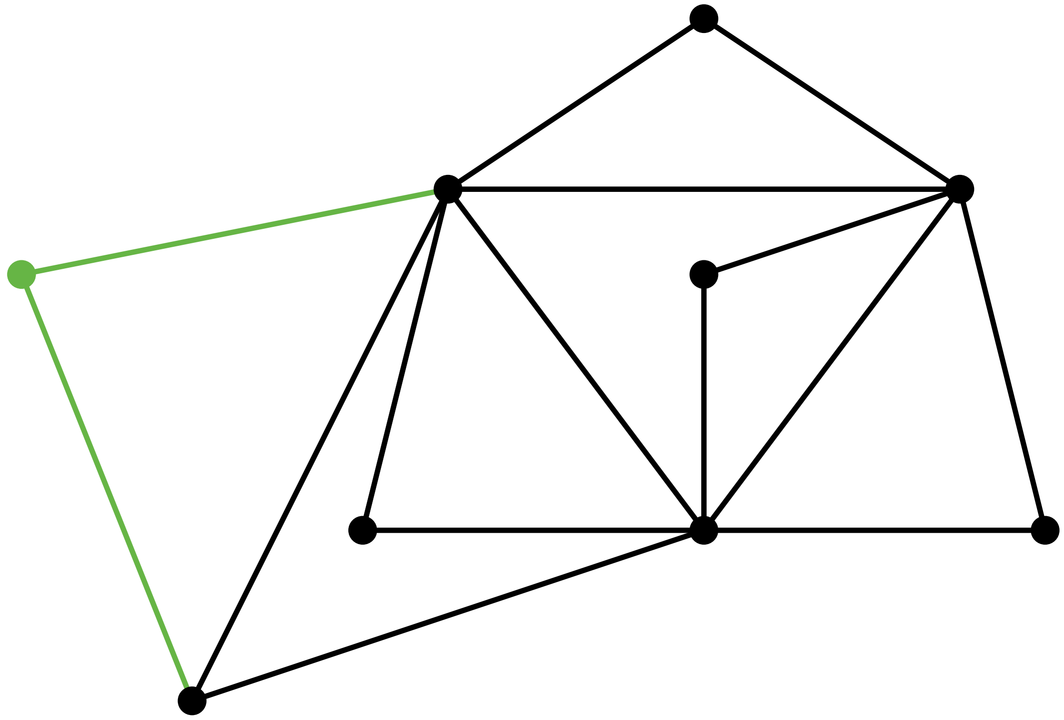
Introduction

Iteratively add a new vertex connected to the vertices of an existing **edge**.



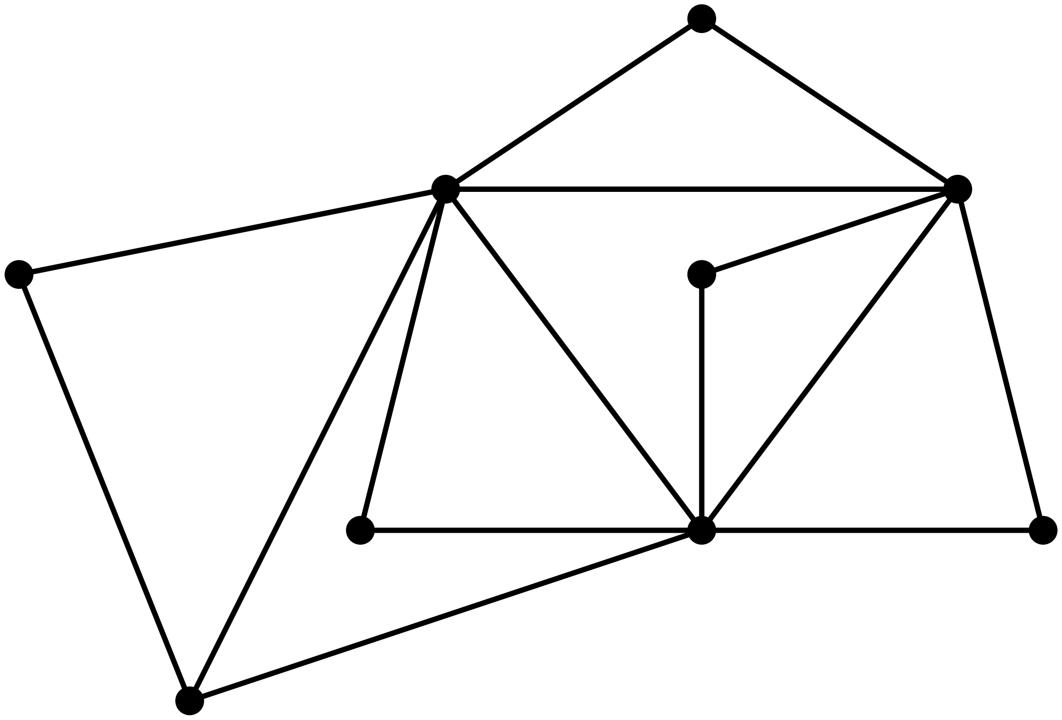
Introduction

Iterately add a new vertex connected to the vertices of an existing **edge**.



Introduction

Iteratively add a new vertex connected to the vertices of an existing **edge**.



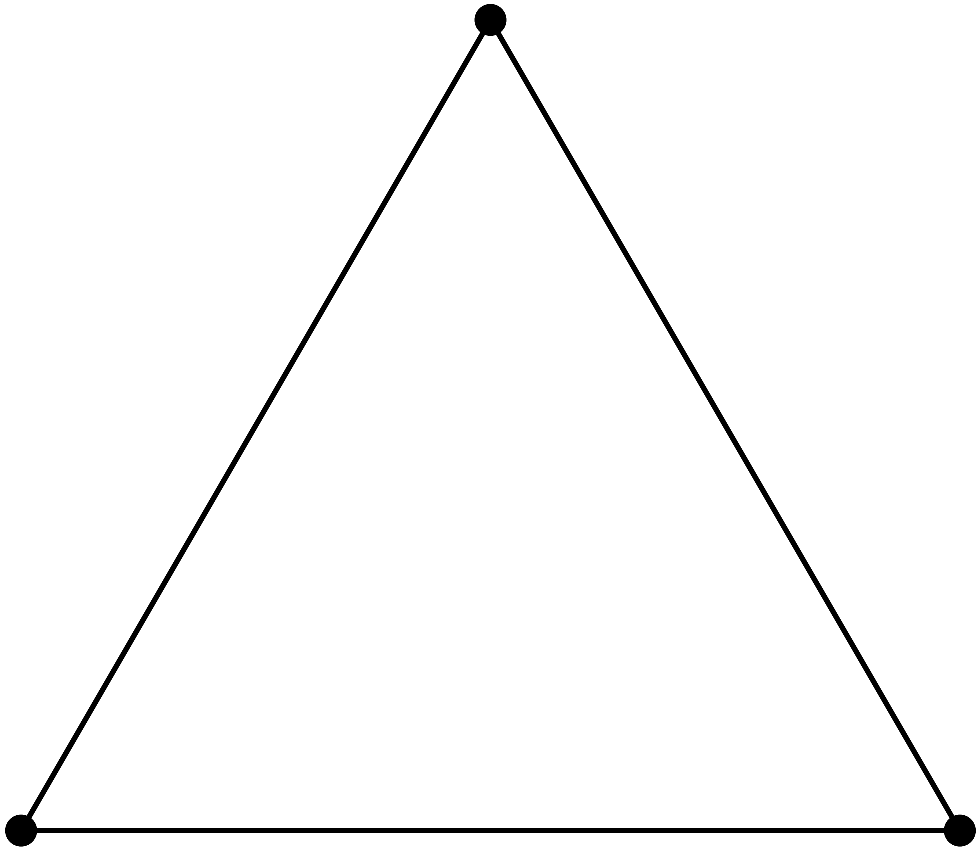
2-trees

Introduction

Iteratively add a new vertex connected to the vertices of an existing **triangle**.

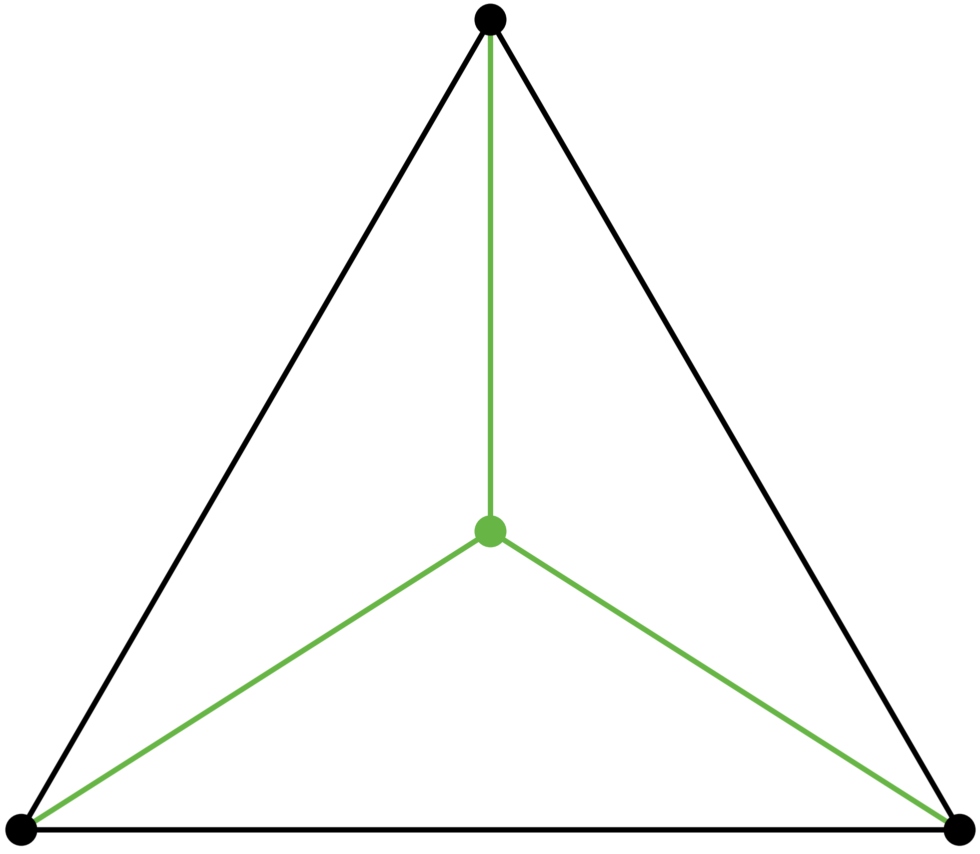
Introduction

Iteratively add a new vertex connected to the vertices of an existing **triangle**.



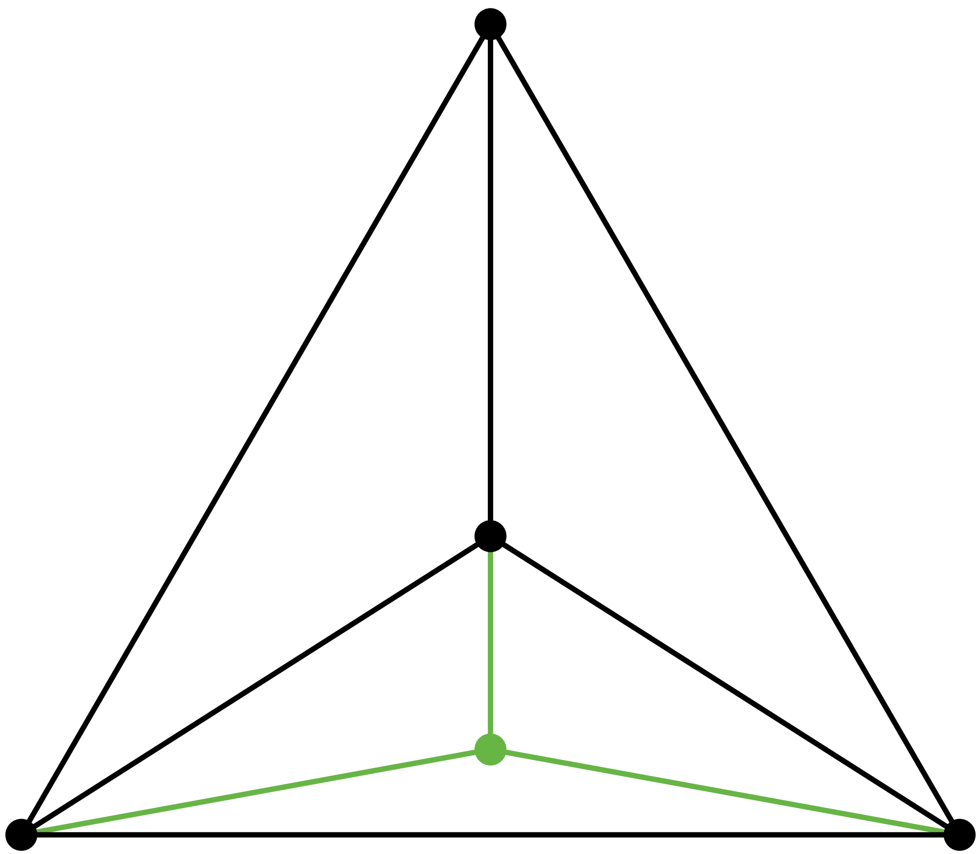
Introduction

Iteratively add a new vertex connected to the vertices of an existing **triangle**.



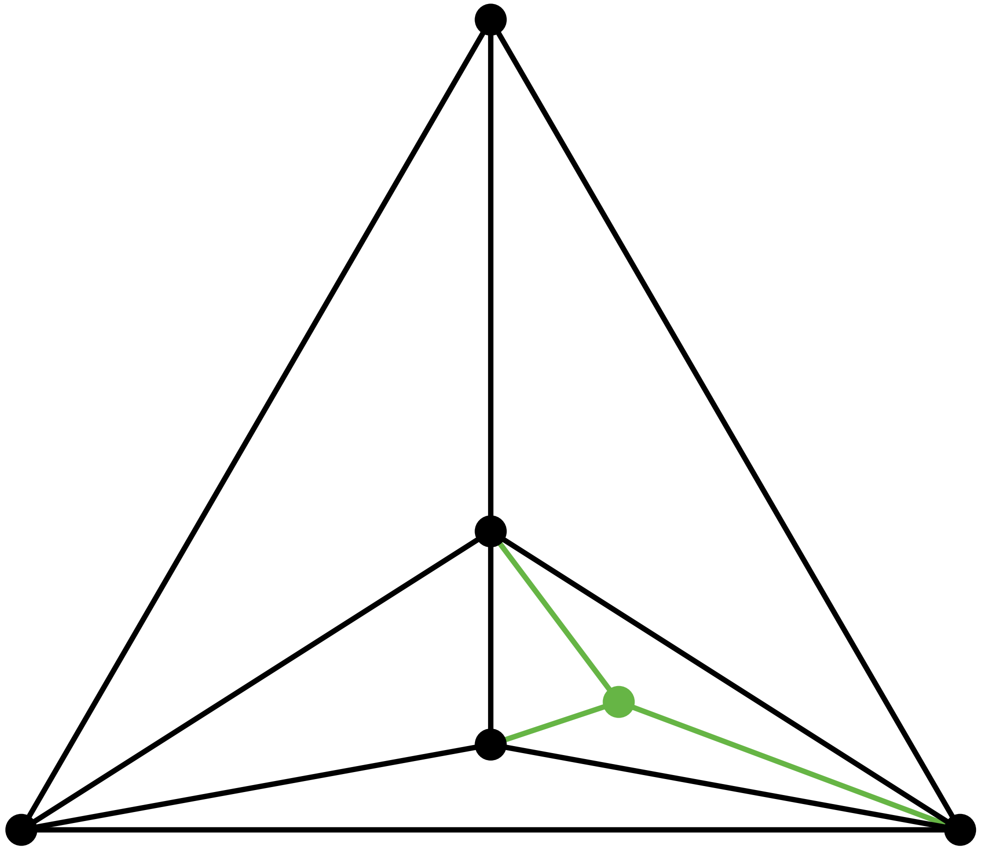
Introduction

Iteratively add a new vertex connected to the vertices of an existing **triangle**.



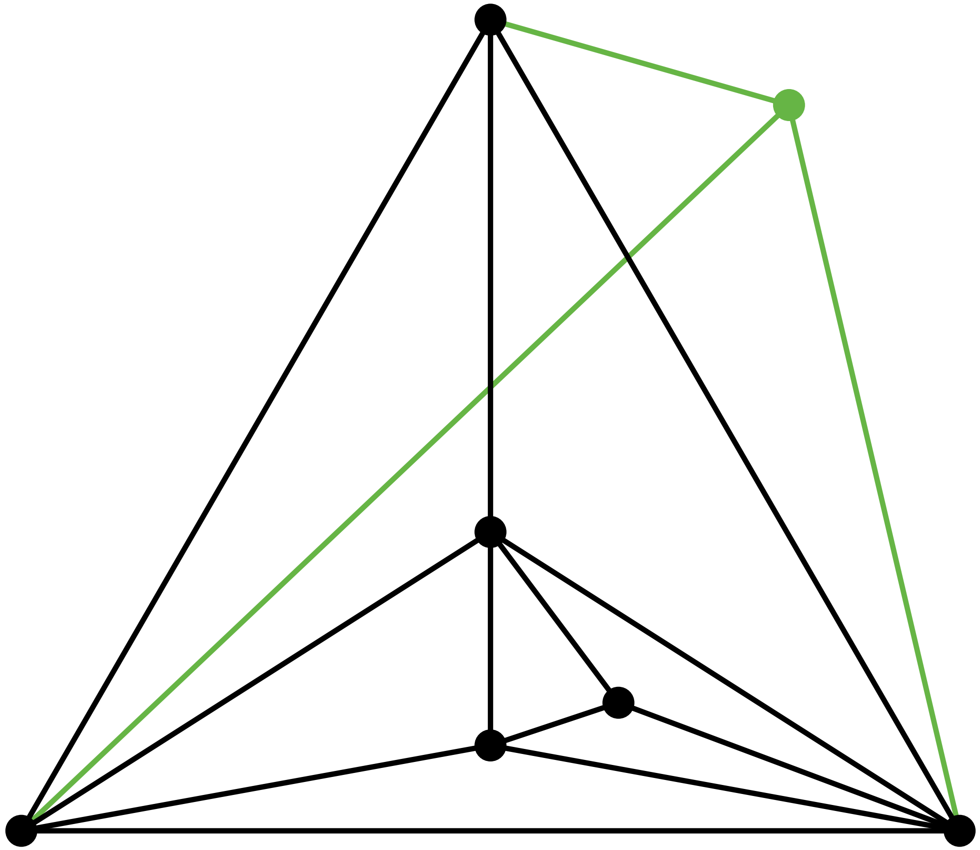
Introduction

Iteratively add a new vertex connected to the vertices of an existing **triangle**.



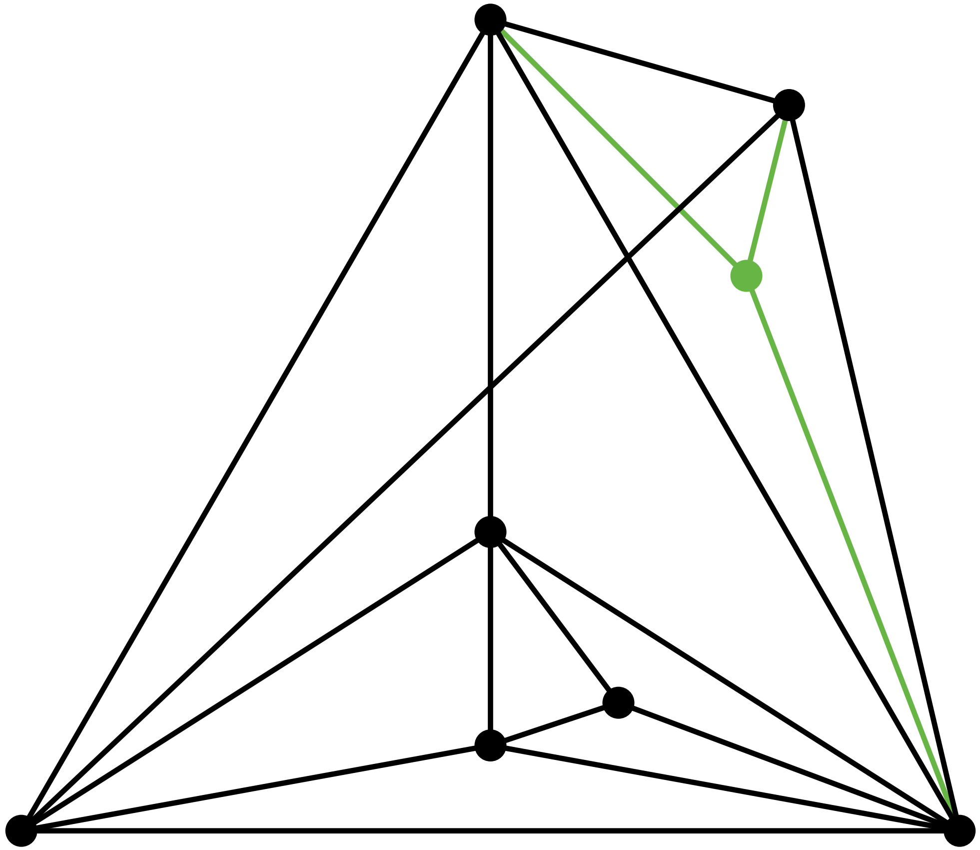
Introduction

Iteratively add a new vertex connected to the vertices of an existing **triangle**.



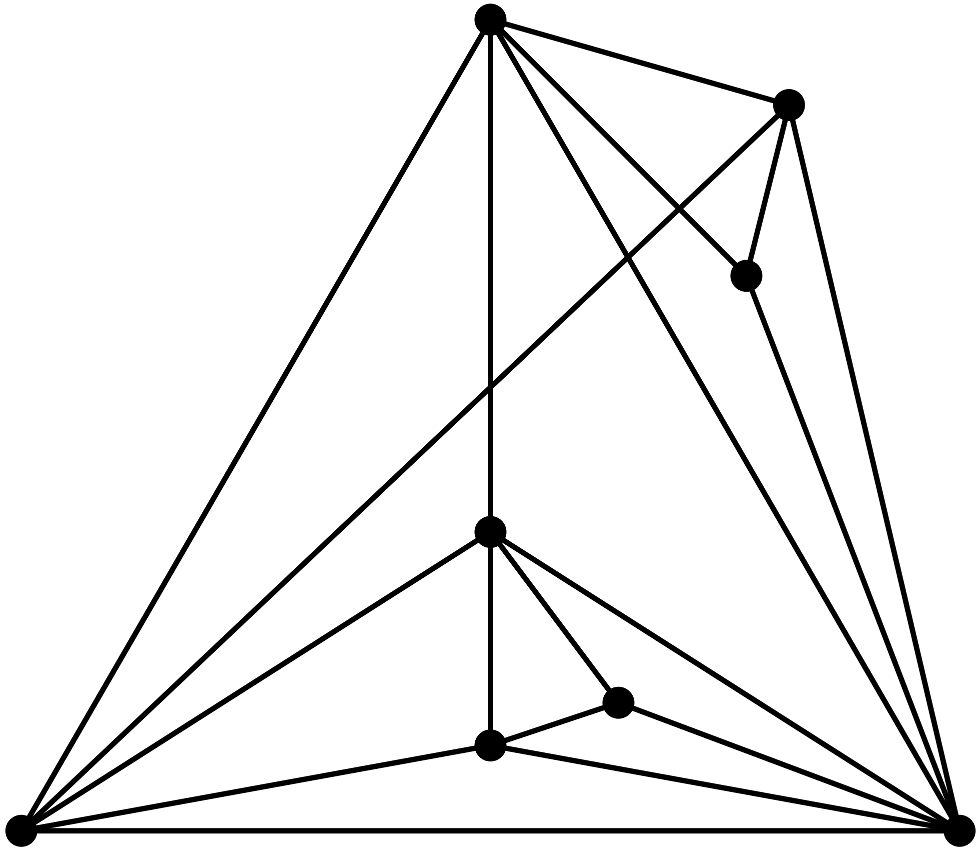
Introduction

Iteratively add a new vertex connected to the vertices of an existing **triangle**.



Introduction

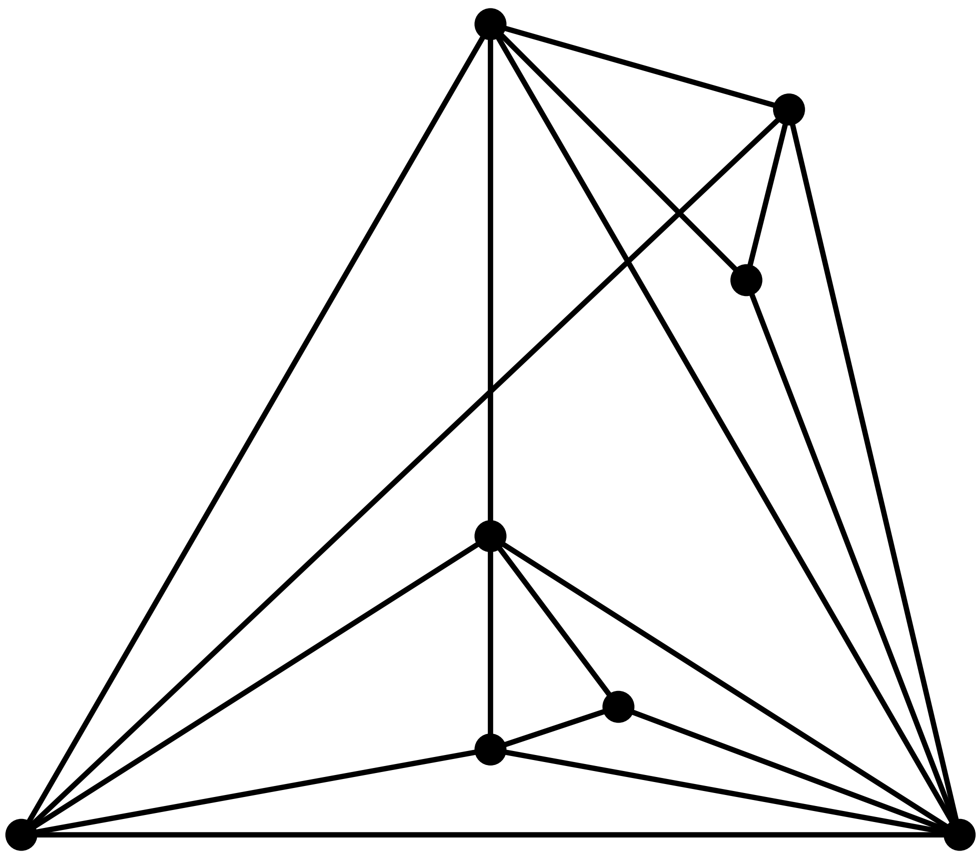
Iteratively add a new vertex connected to the vertices of an existing **triangle**.



3-trees

Introduction

Iteratively add a new vertex connected to the vertices of an existing **triangle**.



3-trees

Definition. A **k-tree** is a graph obtained from a $(k+1)$-clique by successively adding a new vertex connected to all vertices of an existing k-clique.

Introduction

Iteratively add a new vertex connected to the vertices of an existing **clique** (complete subgraph).

Introduction

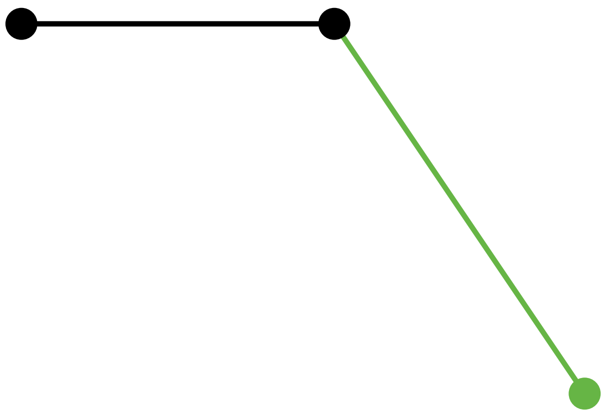
Iteratively add a new vertex connected to the vertices of an existing **clique** (complete subgraph).

Introduction

Iteratively add a new vertex connected to the vertices of an existing **clique** (complete subgraph).

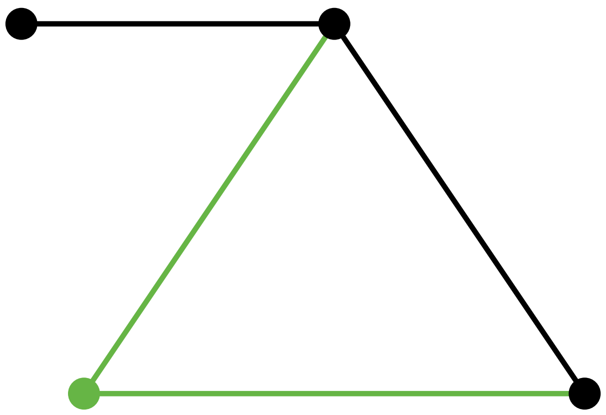
Introduction

Iteratively add a new vertex connected to the vertices of an existing **clique** (complete subgraph).



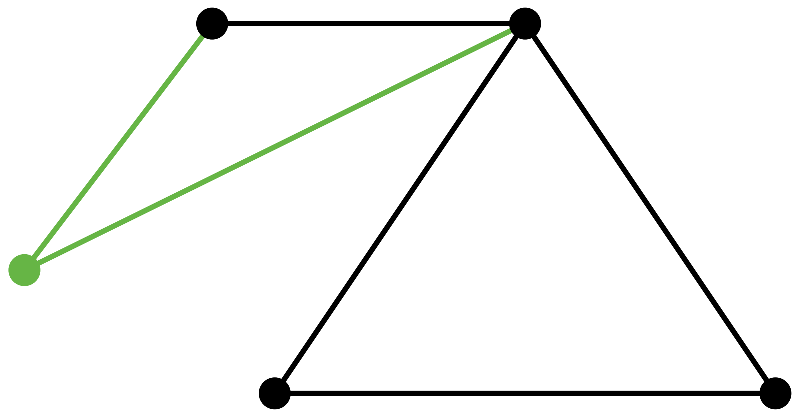
Introduction

Iteratively add a new vertex connected to the vertices of an existing **clique** (complete subgraph).



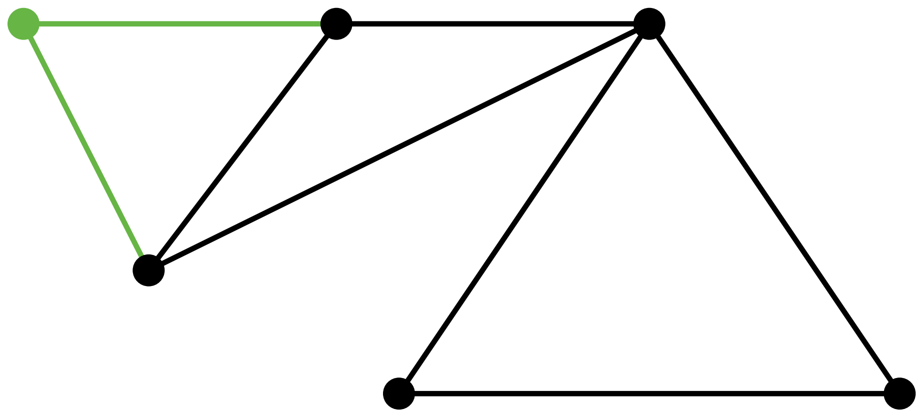
Introduction

Iteratively add a new vertex connected to the vertices of an existing **clique** (complete subgraph).



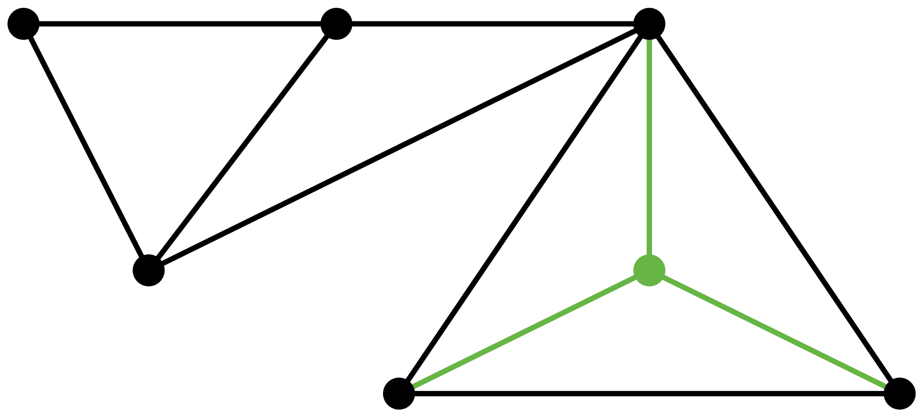
Introduction

Iteratively add a new vertex connected to the vertices of an existing **clique** (complete subgraph).



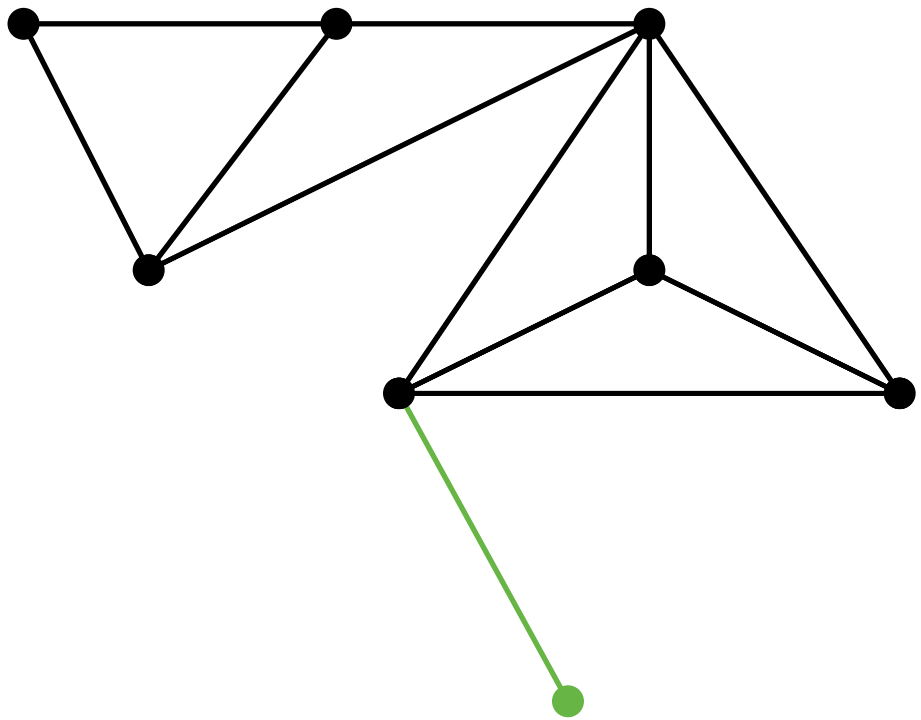
Introduction

Iteratively add a new vertex connected to the vertices of an existing **clique** (complete subgraph).



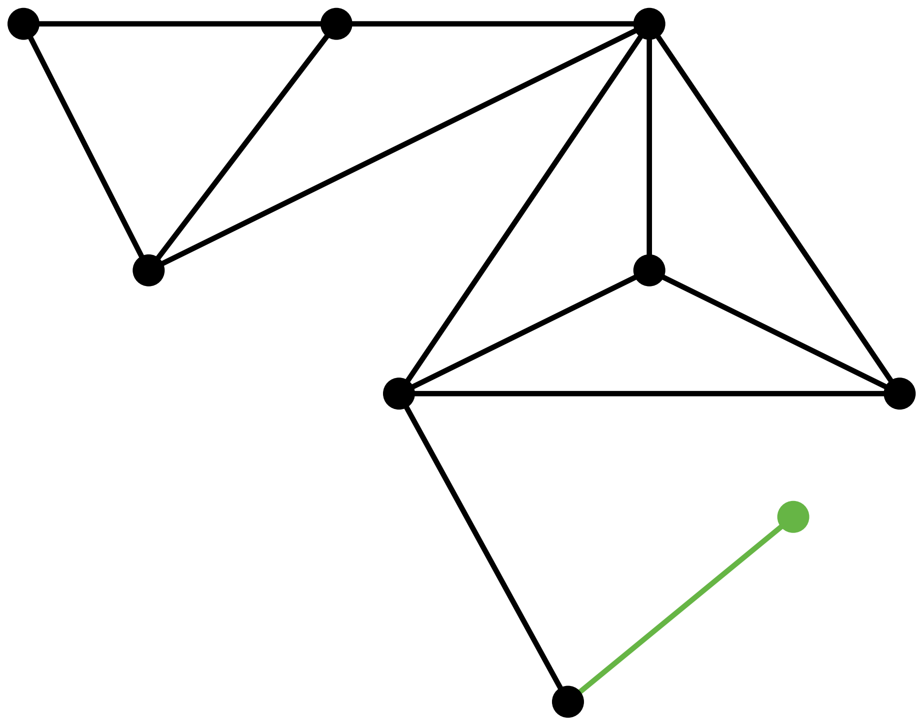
Introduction

Iteratively add a new vertex connected to the vertices of an existing **clique** (complete subgraph).



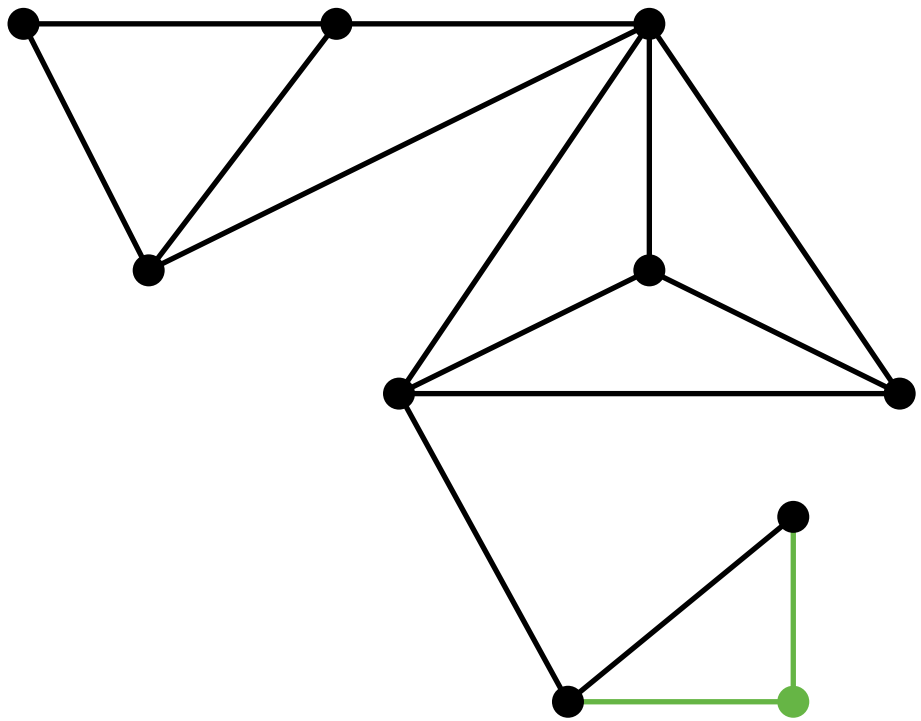
Introduction

Iteratively add a new vertex connected to the vertices of an existing **clique** (complete subgraph).



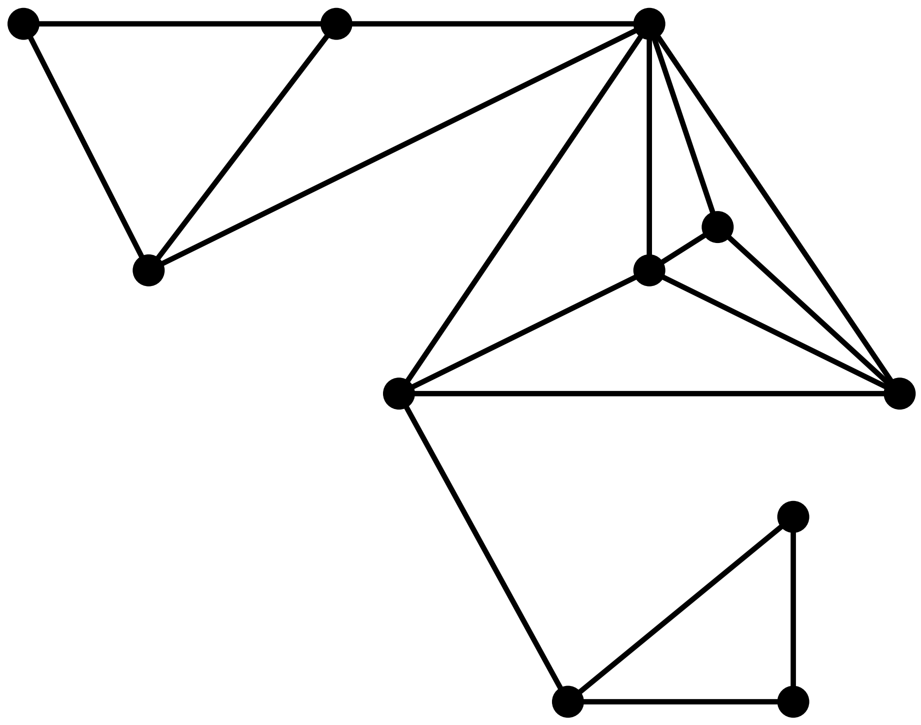
Introduction

Iteratively add a new vertex connected to the vertices of an existing **clique** (complete subgraph).



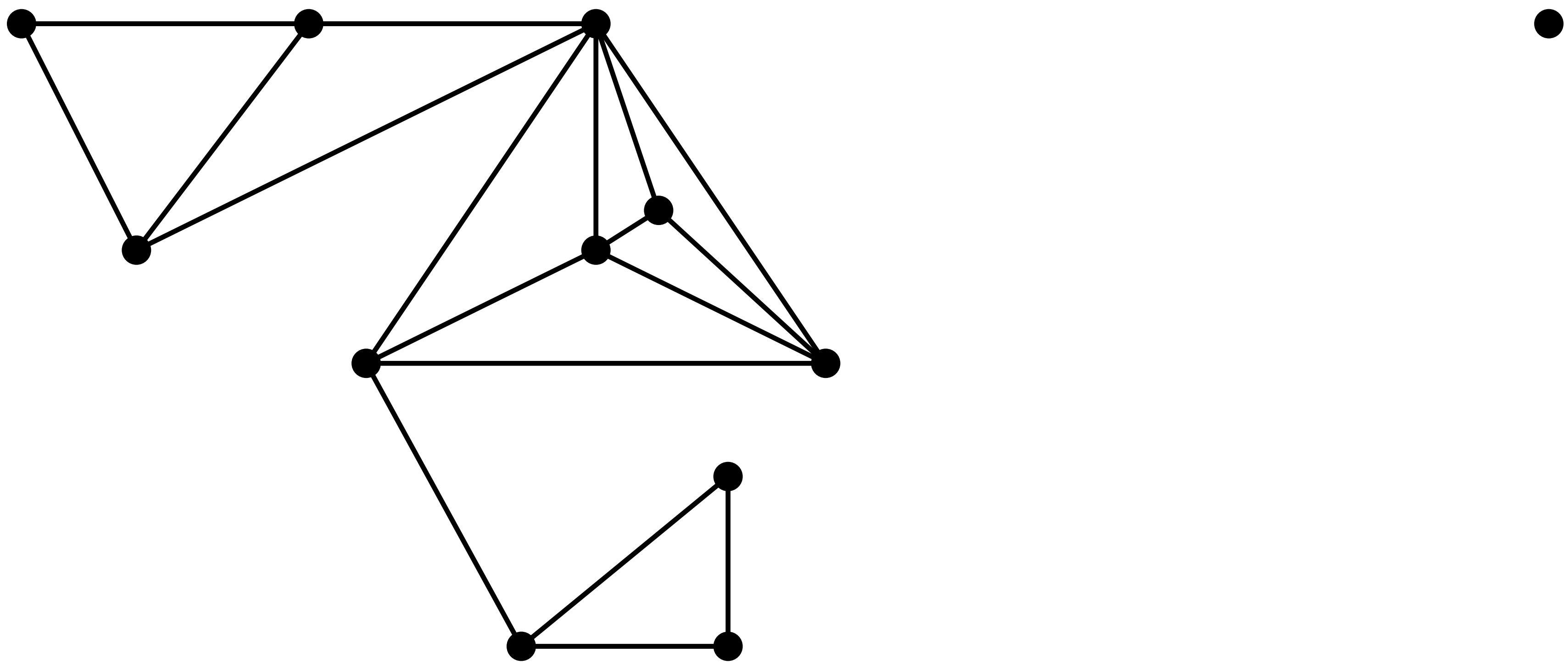
Introduction

Iterative add a new vertex connected to the vertices of an existing **clique** (complete subgraph).



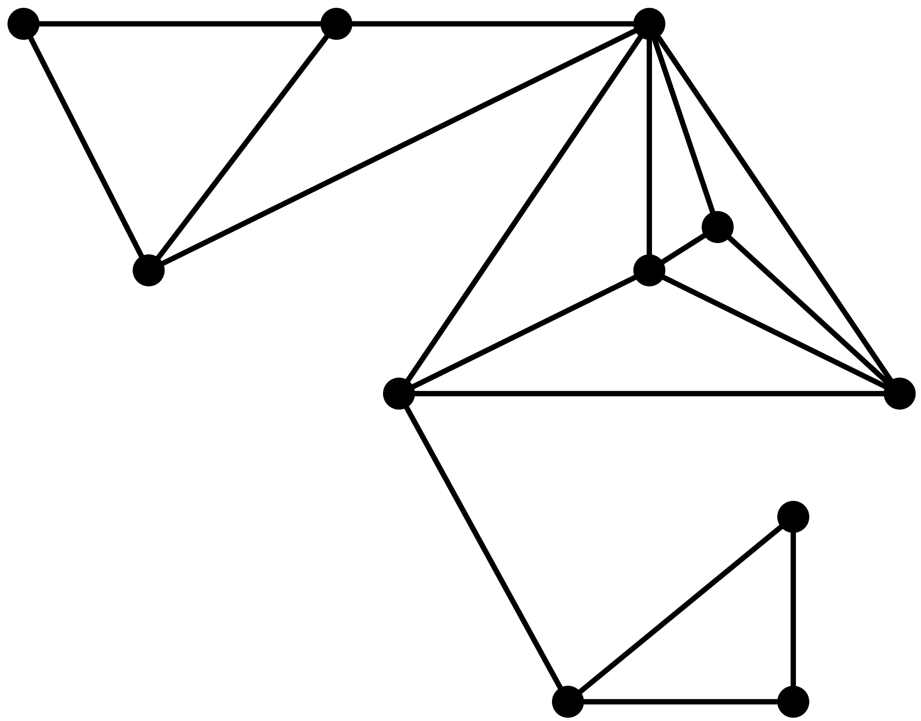
Introduction

Iteratively add a new vertex connected to the vertices of an existing **clique** (complete subgraph).



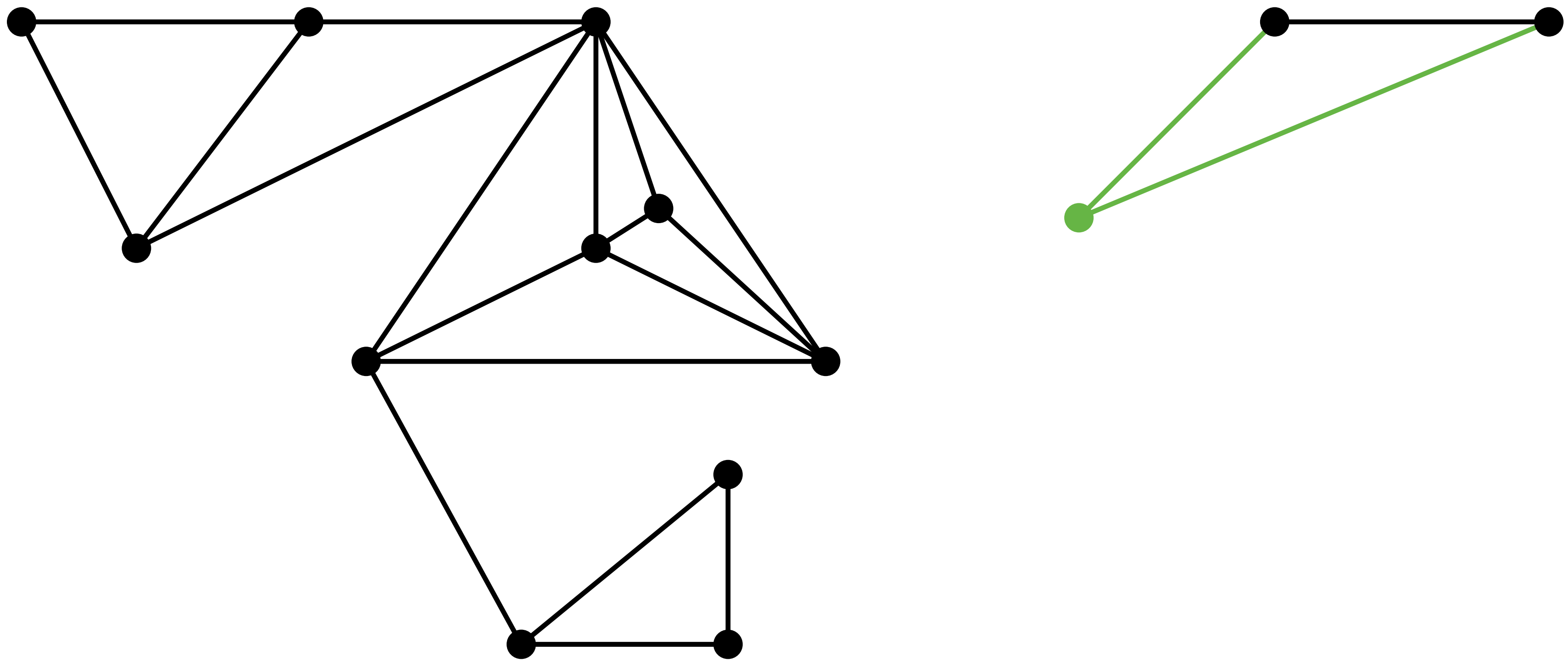
Introduction

Iteratively add a new vertex connected to the vertices of an existing **clique** (complete subgraph).



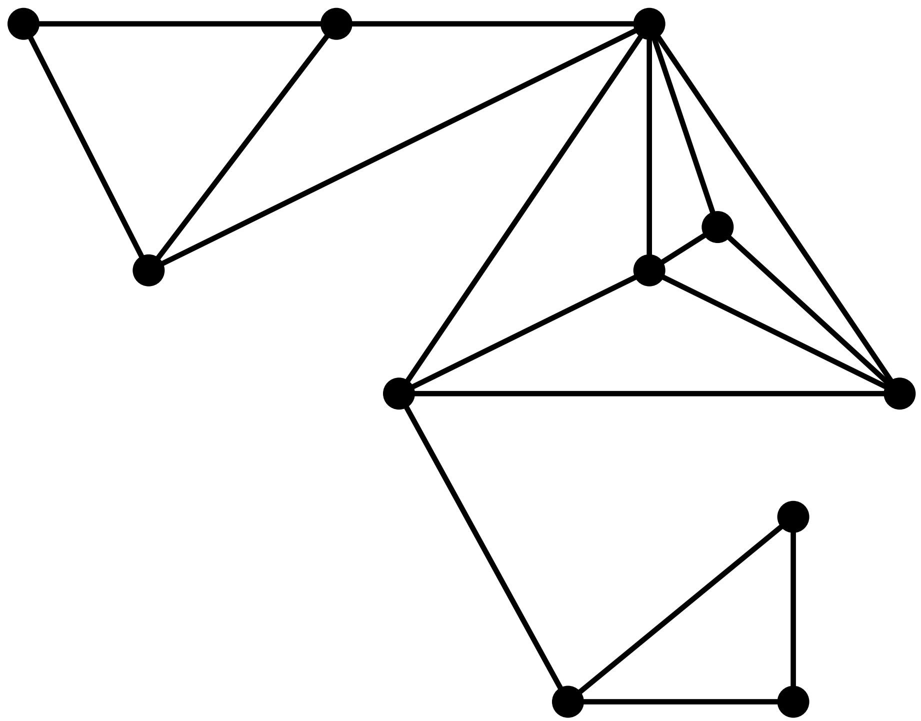
Introduction

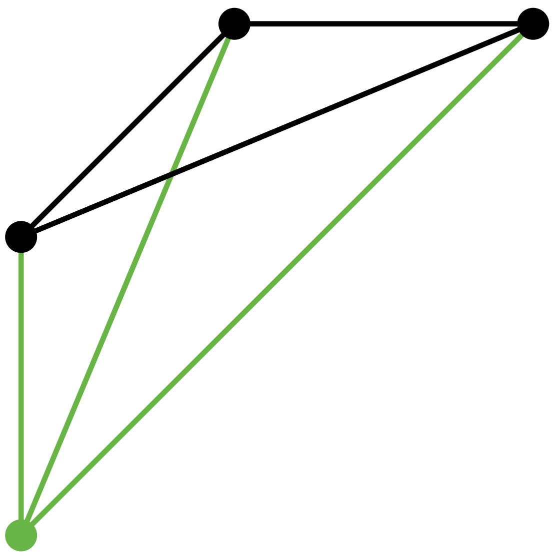
Iteratively add a new vertex connected to the vertices of an existing **clique** (complete subgraph).



Introduction

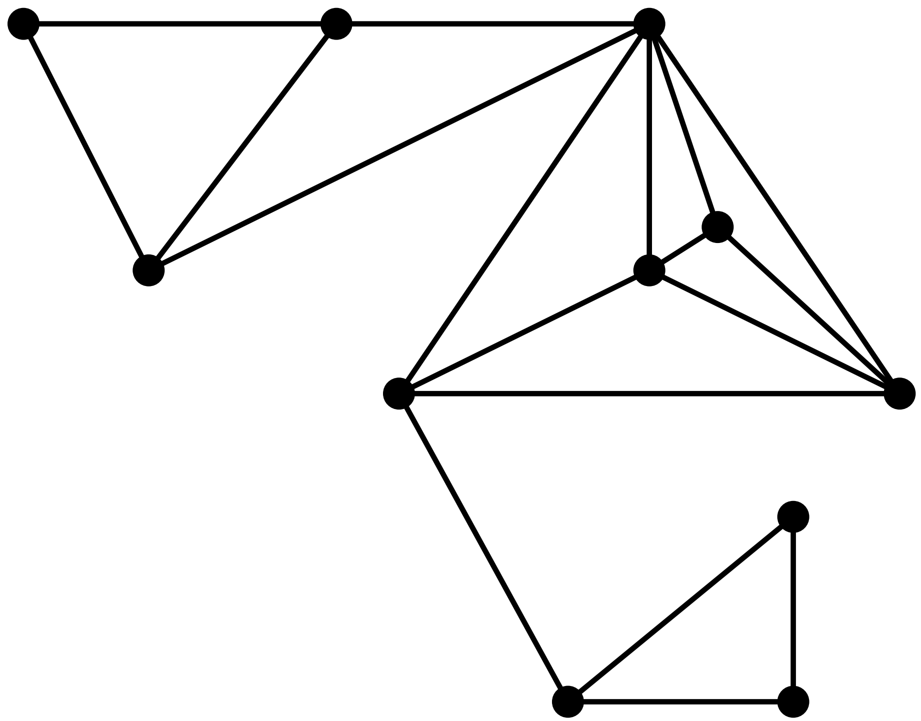
Iterately add a new vertex connected to the vertices of an existing **clique** (complete subgraph).

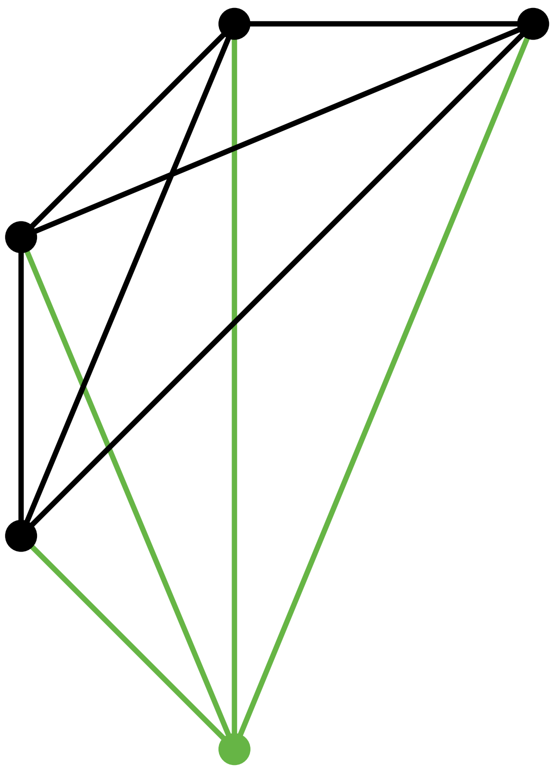




Introduction

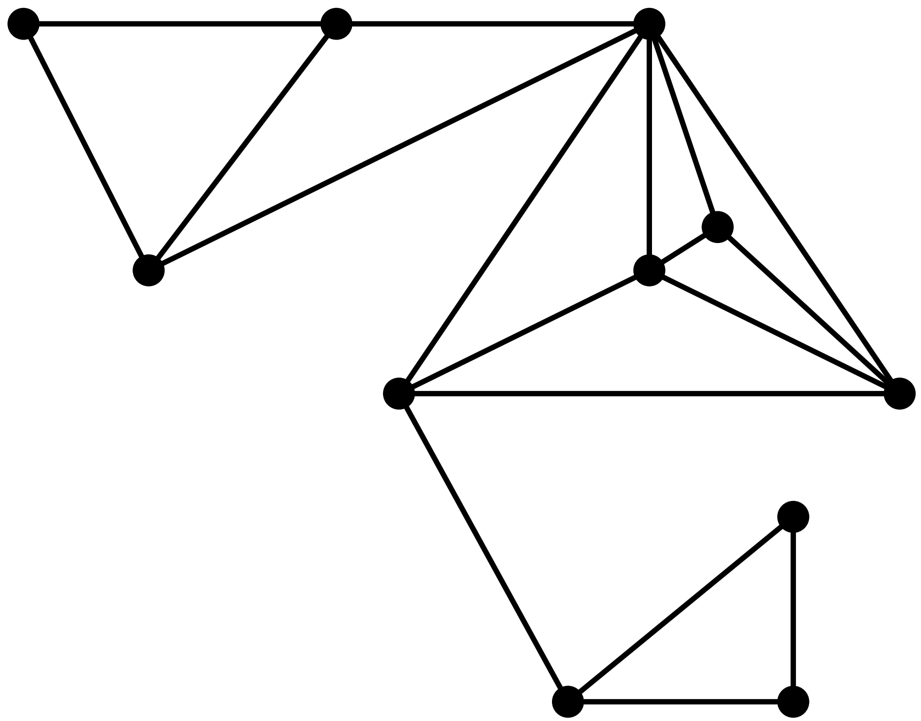
Iteratively add a new vertex connected to the vertices of an existing **clique** (complete subgraph).

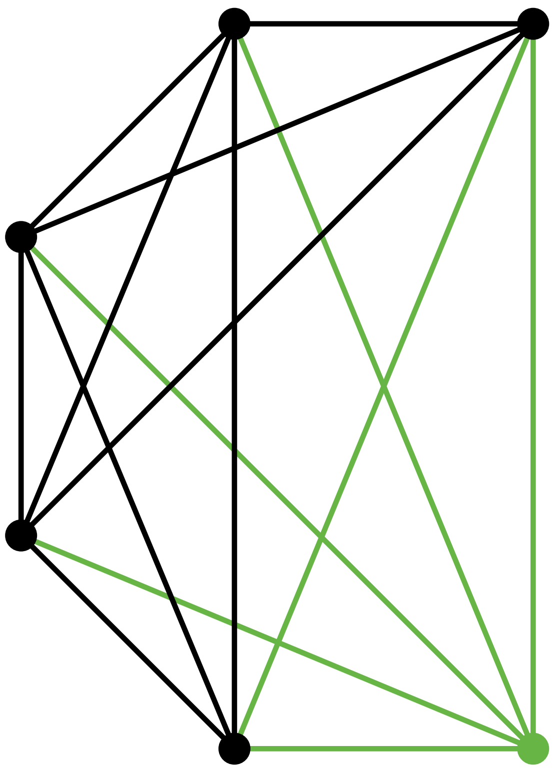




Introduction

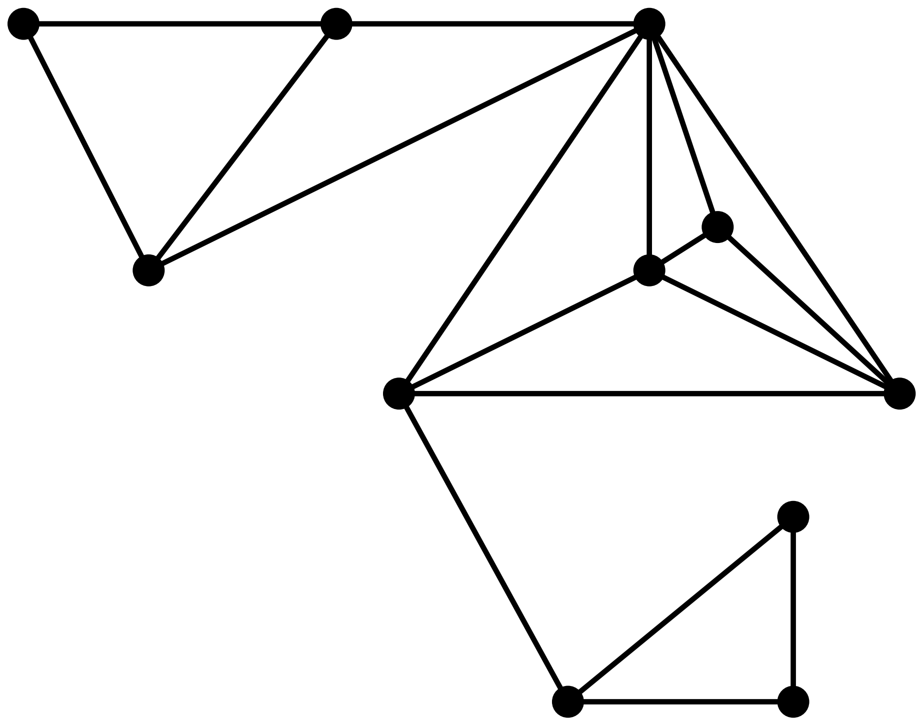
Iteratively add a new vertex connected to the vertices of an existing **clique** (complete subgraph).

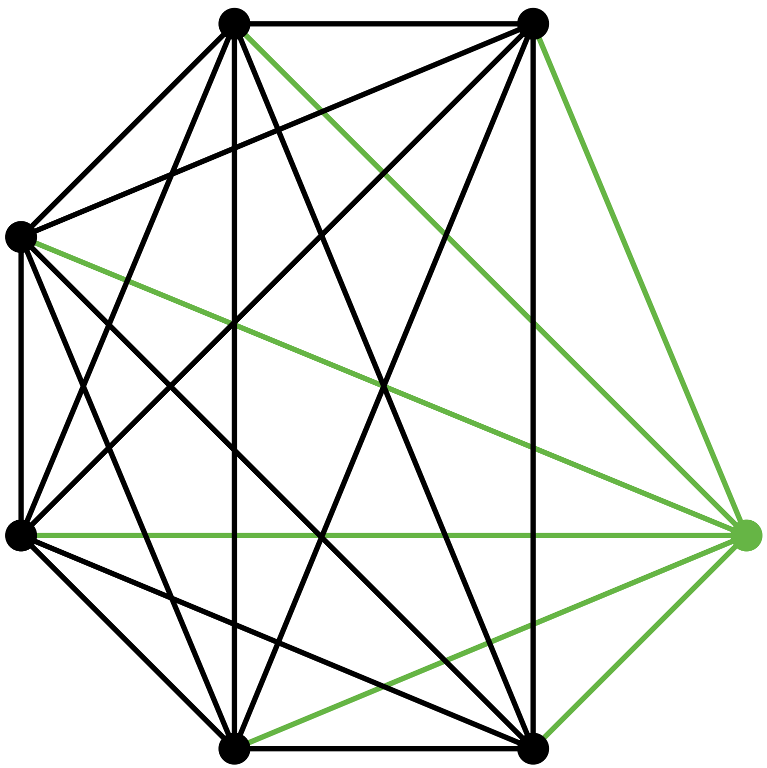




Introduction

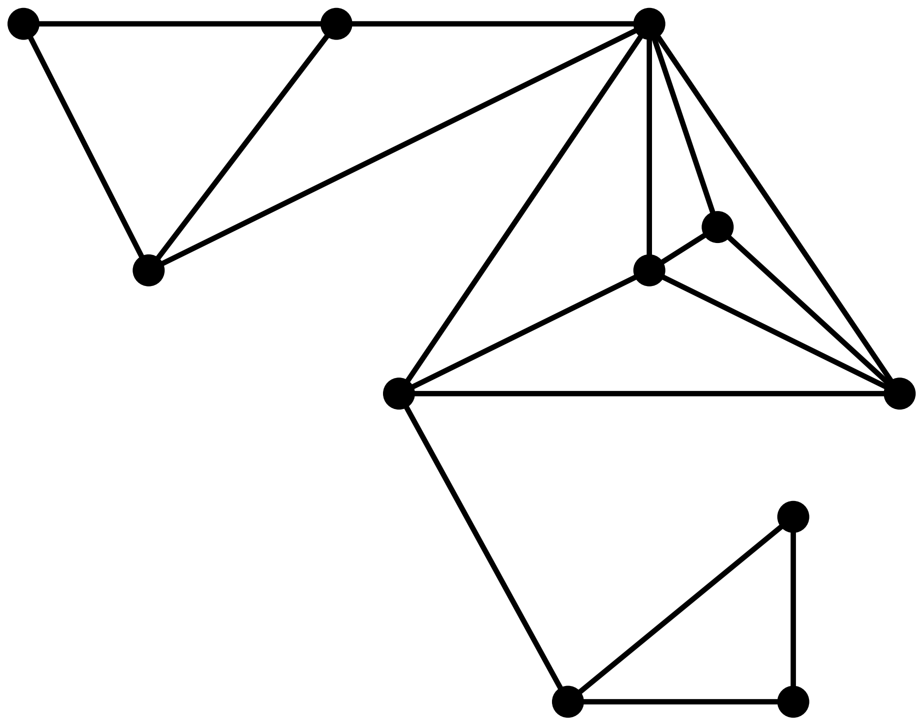
Iteratively add a new vertex connected to the vertices of an existing **clique** (complete subgraph).

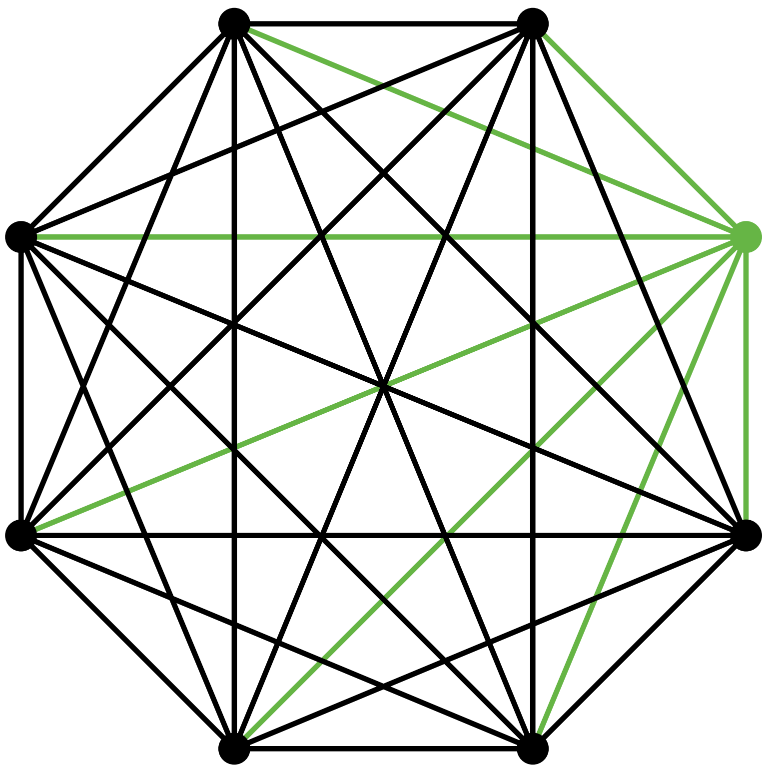




Introduction

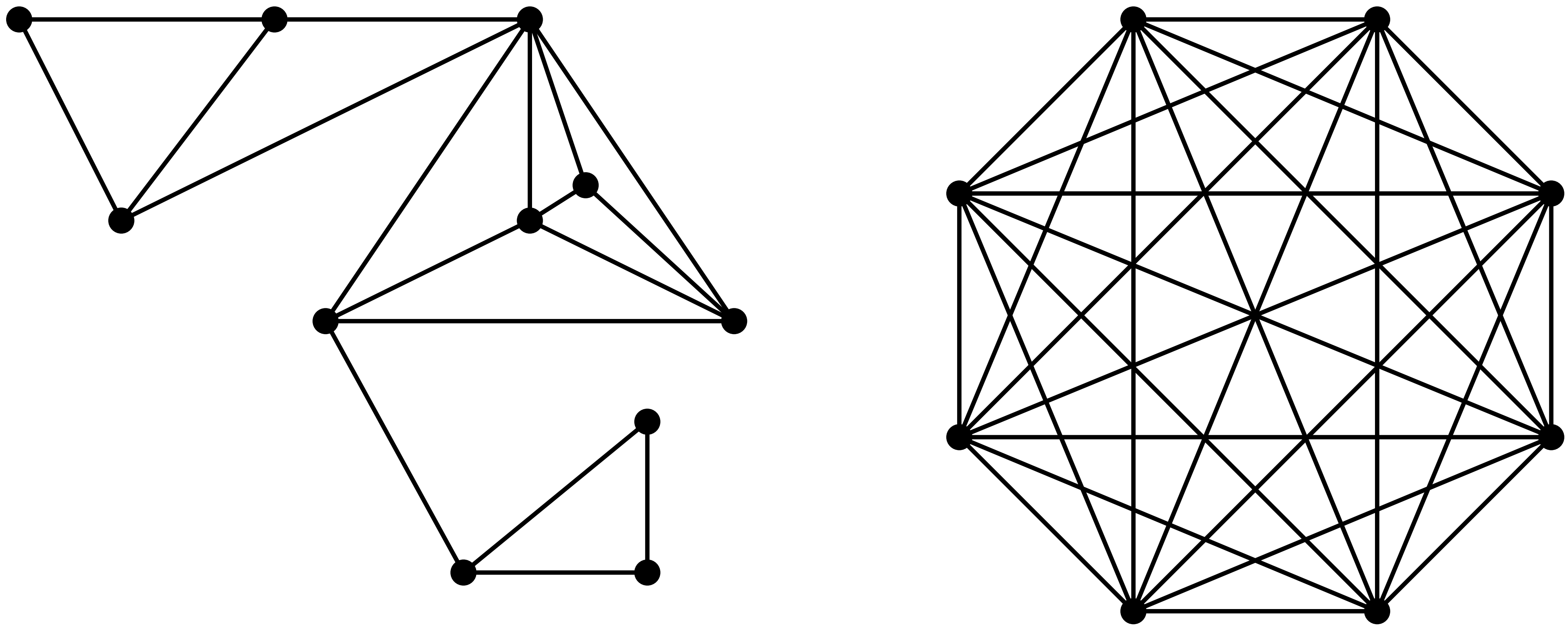
Iteratively add a new vertex connected to the vertices of an existing **clique** (complete subgraph).





Introduction

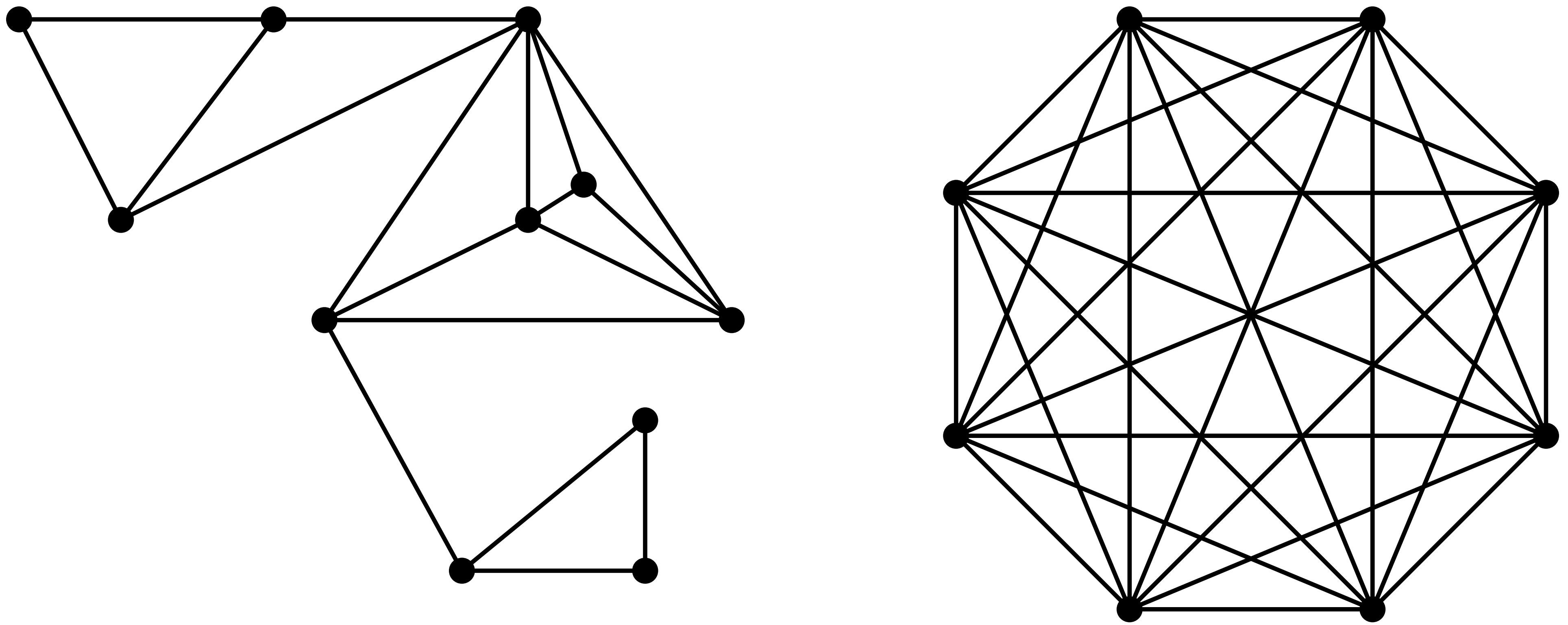
Iteratively add a new vertex connected to the vertices of an existing **clique** (complete subgraph).



Chordal graphs

Introduction

Iteratively add a new vertex connected to the vertices of an existing **clique** (complete subgraph).



Chordal graphs

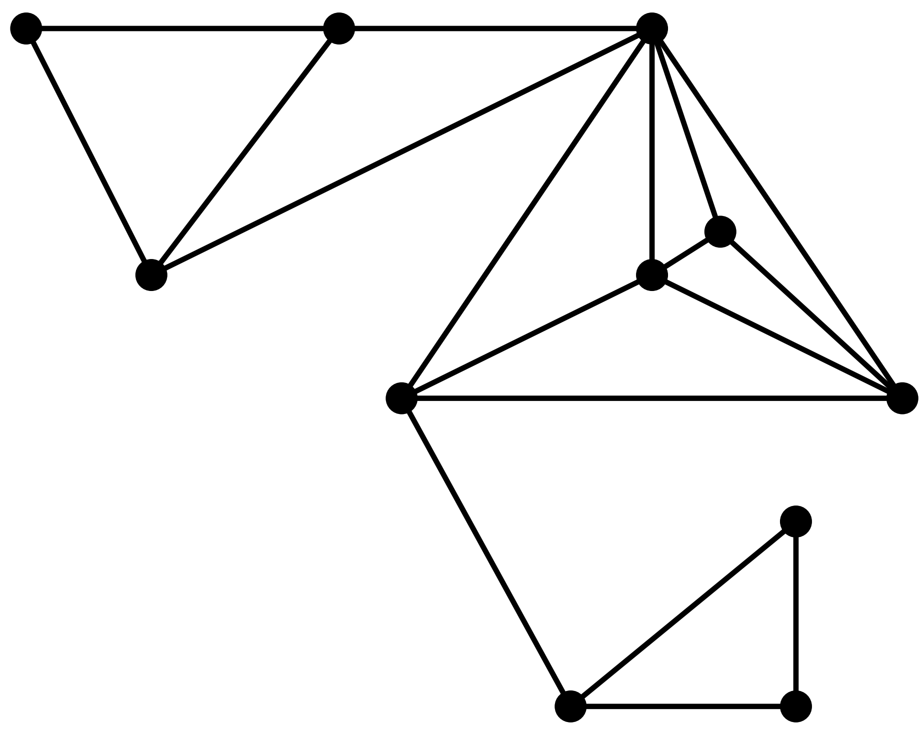
Definition. A graph is **chordal** if it has no induced cycle of lengh greater than 3.

Introduction

Iteratively add a new vertex connected to the vertices of an existing **clique of size at most t**.

Introduction

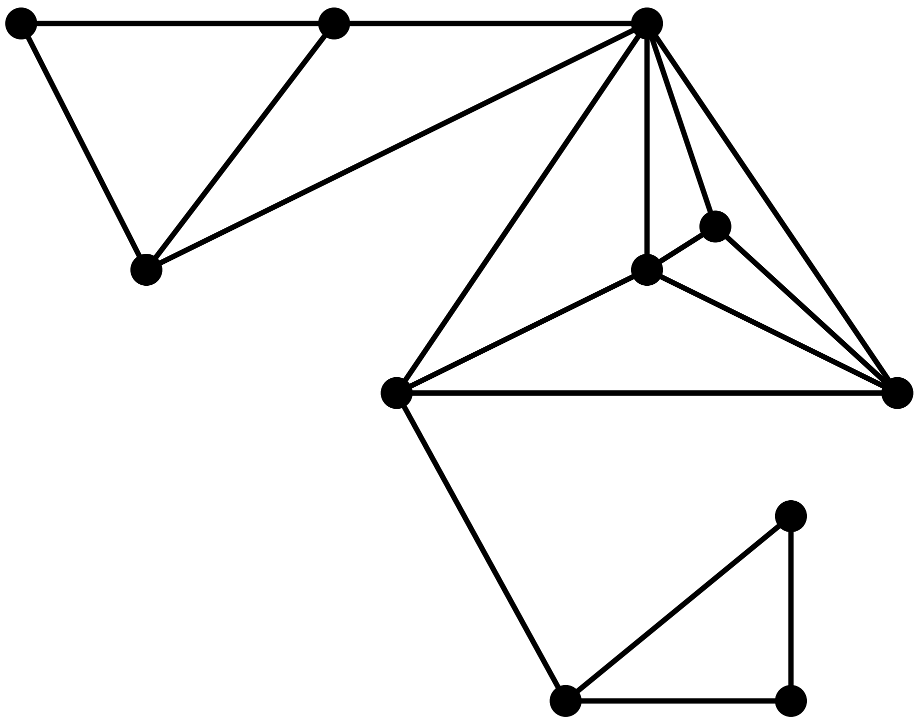
Iteratively add a new vertex connected to the vertices of an existing **clique of size at most t**.



$t = 3$

Introduction

Iteratively add a new vertex connected to the vertices of an existing **clique of size at most t**.

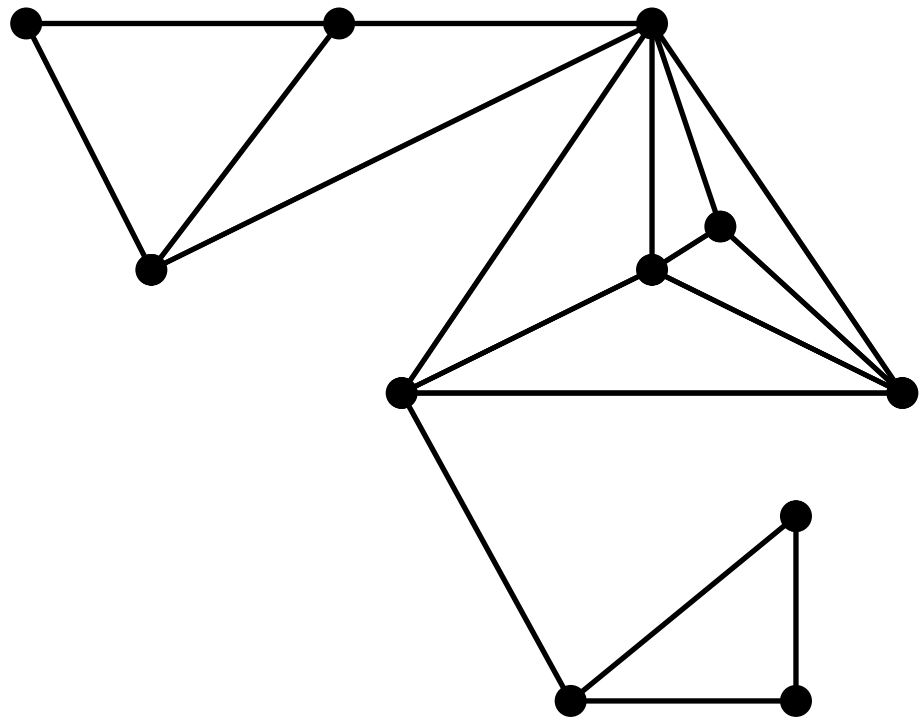


$t = 3$

Chordal graphs with tree-width at most t

Introduction

Iteratively add a new vertex connected to the vertices of an existing **clique of size at most t**.



$t = 3$

Chordal graphs with tree-width at most t

Definition. The **tree-width** of a graph G is the minimum k such that G is the subgraph of a k-tree.

The symbolic method

Our goal is to determine the number of graphs in the family with size n.

The symbolic method

Our goal is to determine the number of graphs in the family with size n.

Definition. A **combinatorial class** is a pair $(\mathcal{A}, |\cdot|)$ where

- $\mathcal{A}$ is a family of combinatorial objects,
- $|\cdot| : \mathcal{A} \to \mathbb{N}$ is a size function,
- The number of objects with size n is $a_n < \infty$.

The symbolic method

Our goal is to determine the number of graphs in the family with size n.

Definition. A **combinatorial class** is a pair $(\mathcal{A}, |\cdot|)$ where

- $\mathcal{A}$ is a family of combinatorial objects,
- $|\cdot| : \mathcal{A} \to \mathbb{N}$ is a size function,
- The number of objects with size n is $a_n < \infty$.

Definition. The **ordinary generating function** (OGF) of $(\mathcal{A}, |\cdot|)$ is the formal power series

$$A(x) = \sum_{n \geq 0} a_n x^n.$$

Suitable for unlabelled classes.

Definition. The **exponential generating function** (EGF) of $(\mathcal{A}, |\cdot|)$ is the formal power series

$$A(x) = \sum_{n \geq 0} \frac{a_n}{n!} x^n.$$

Suitable for labelled classes.

The symbolic method

Our goal is to determine the number of graphs in the family with size n.

Definition. A **combinatorial class** is a pair $(\mathcal{A}, |\cdot|)$ where

- $\mathcal{A}$ is a family of combinatorial objects,
- $|\cdot| : \mathcal{A} \to \mathbb{N}$ is a size function,
- The number of objects with size n is $a_n < \infty$.

Definition. The **ordinary generating function** (OGF) of $(\mathcal{A}, |\cdot|)$ is the formal power series

$$A(x) = \sum_{n \geq 0} a_n x^n.$$

Suitable for unlabelled classes.

Definition. The **exponential generating function** (EGF) of $(\mathcal{A}, |\cdot|)$ is the formal power series

$$A(x) = \sum_{n \geq 0} \frac{a_n}{n!} x^n.$$

Suitable for labelled classes.

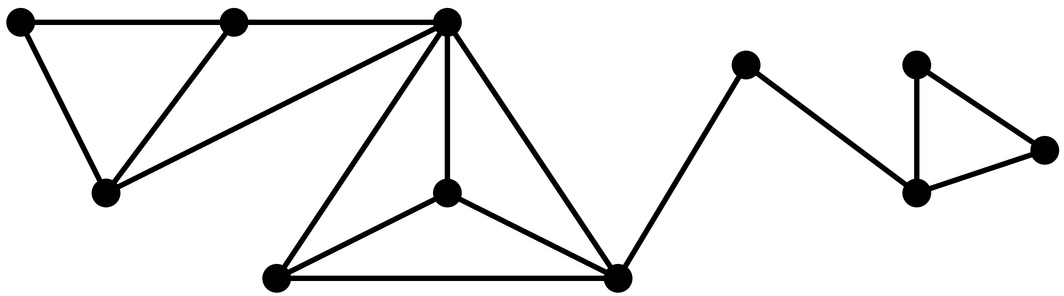
Operations between **classes** translate into relations involving their **generating functions**. **The goal** is to obtain (a system of) equations that determine the GF of our class.

Decomposition of graphs into k-connected components

Definition. The **2-connected components** (or blocks) of a connected graph are its maximal 2-connected subgraphs.

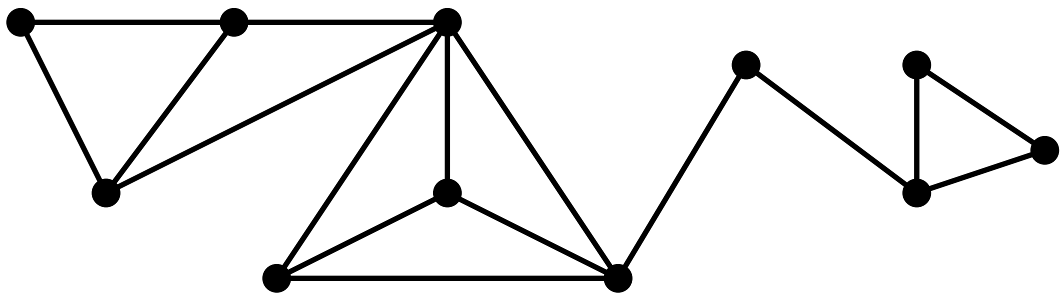
Decomposition of graphs into k-connected components

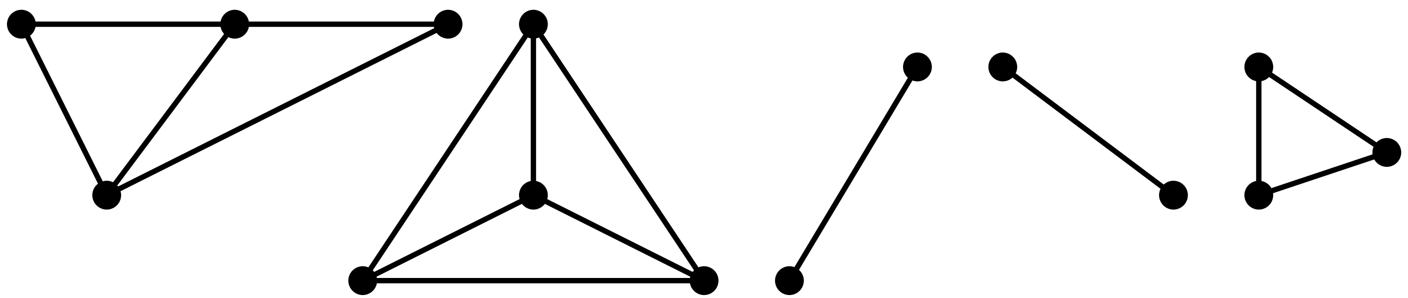
Definition. The **2-connected components** (or blocks) of a connected graph are its maximal 2-connected subgraphs.



Decomposition of graphs into k-connected components

Definition. The **2-connected components** (or blocks) of a connected graph are its maximal 2-connected subgraphs.





Decomposition of graphs into k-connected components

Let $\mathcal{B} \subset \mathcal{C}$ be the class of the 2-connected members of $\mathcal{G}$. Then,

$$C^\bullet(x) = x\exp(B'(C^\bullet(x))), \quad \text{where } C^\bullet(x) = xC'(x),$$

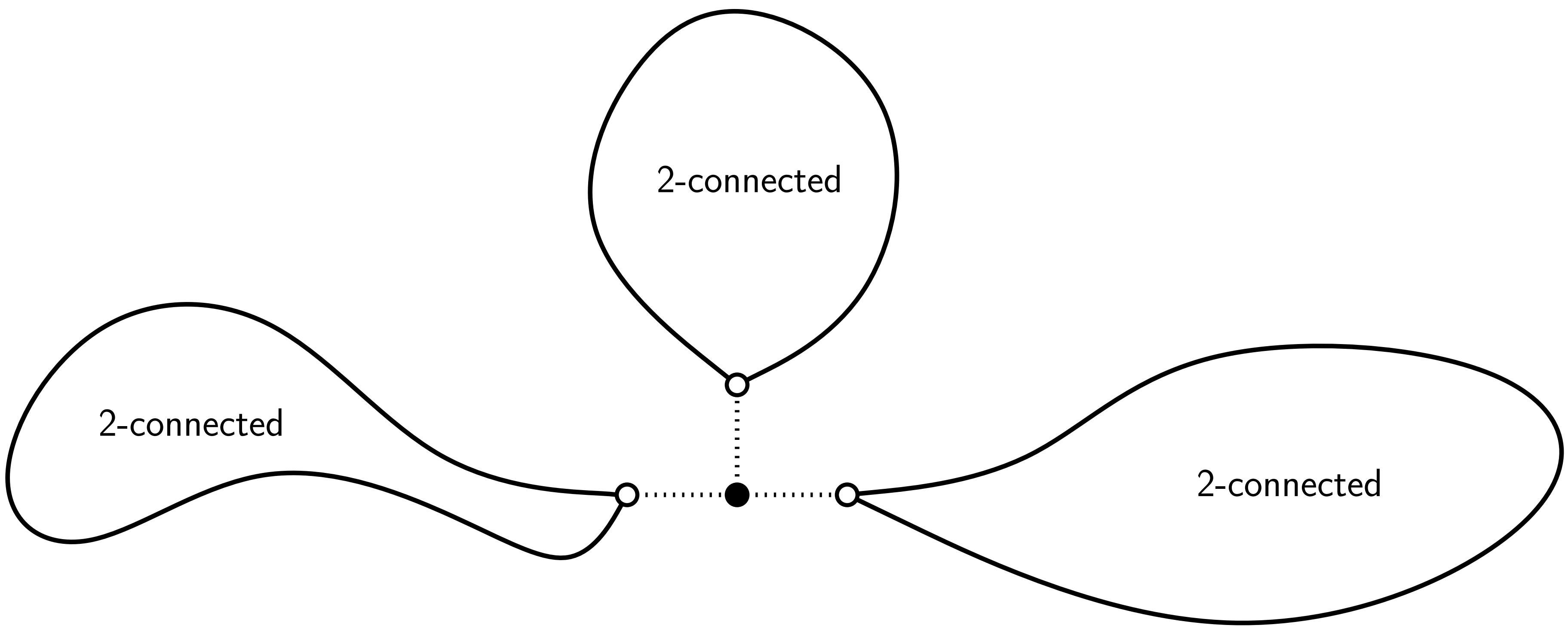
provided that $\mathcal{G}$ is **block-stable**, i.e., that a graph belongs to $\mathcal{C}$ iff its blocks belong to $\mathcal{B}$.

Decomposition of graphs into k-connected components

Let $\mathcal{B} \subset \mathcal{C}$ be the class of the 2-connected members of $\mathcal{G}$. Then,

$$C^\bullet(x) = x\exp(B'(C^\bullet(x))), \quad \text{where } C^\bullet(x) = xC'(x),$$

provided that $\mathcal{G}$ is **block-stable**, i.e., that a graph belongs to $\mathcal{C}$ iff its blocks belong to $\mathcal{B}$.

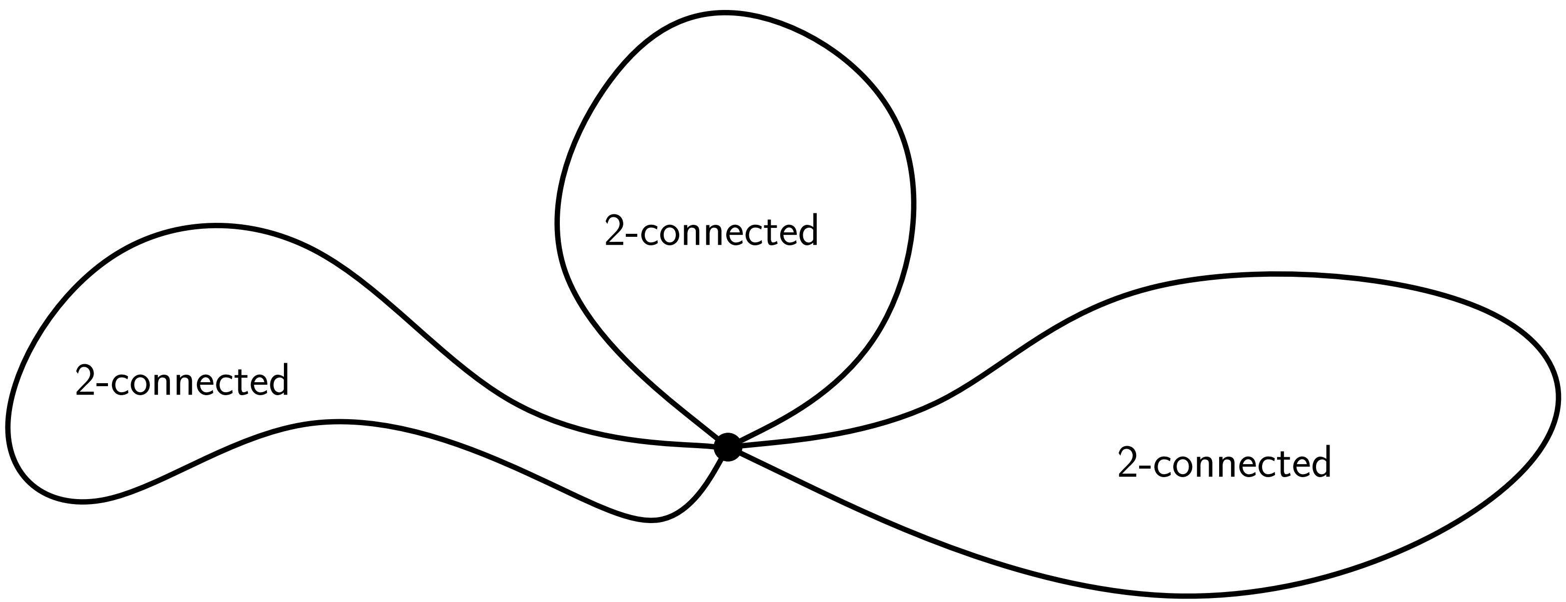


Decomposition of graphs into k-connected components

Let $\mathcal{B} \subset \mathcal{C}$ be the class of the 2-connected members of $\mathcal{G}$. Then,

$$C^\bullet(x) = x \exp(B'(C^\bullet(x))), \quad \text{where } C^\bullet(x) = xC'(x),$$

provided that $\mathcal{G}$ is **block-stable**, i.e., that a graph belongs to $\mathcal{C}$ iff its blocks belong to $\mathcal{B}$.

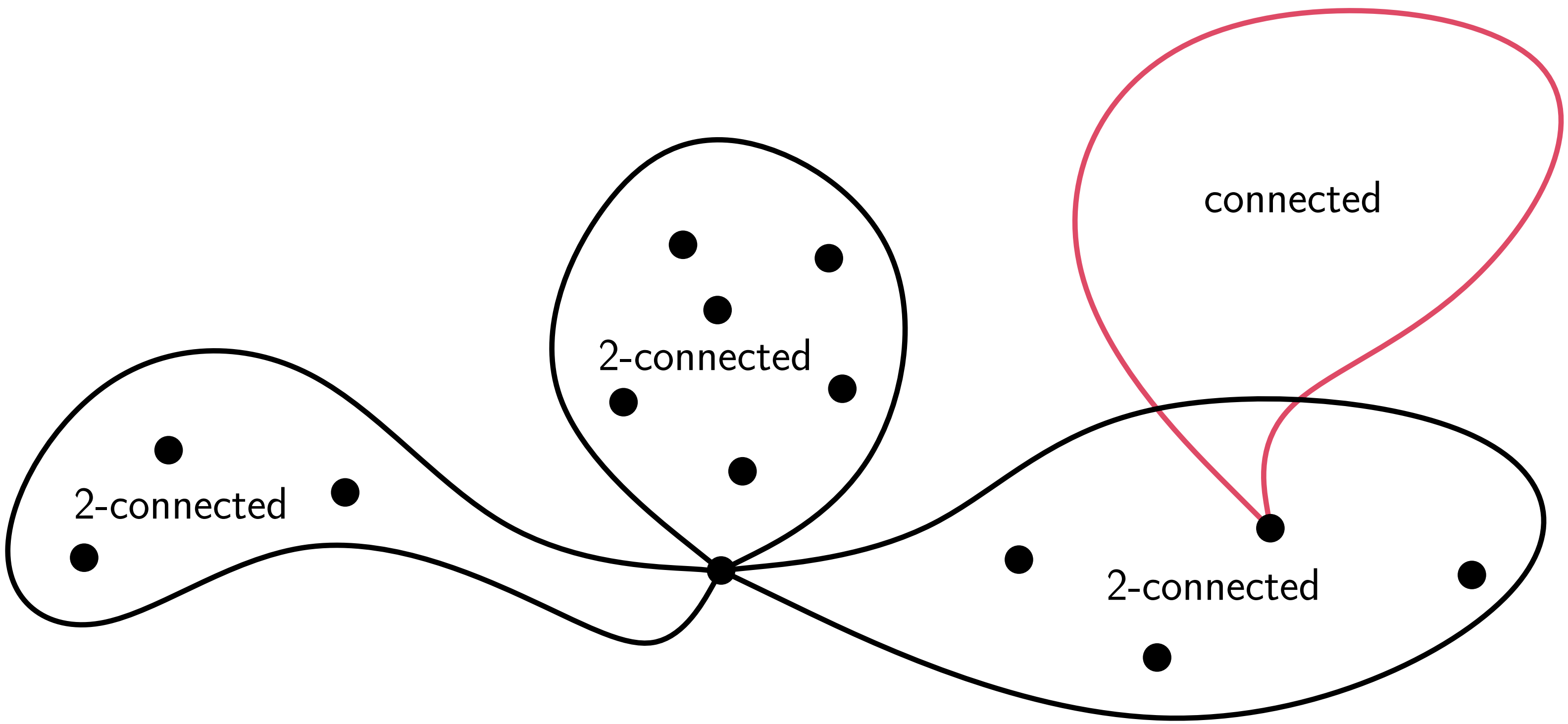


Decomposition of graphs into k-connected components

Let $\mathcal{B} \subset \mathcal{C}$ be the class of the 2-connected members of $\mathcal{G}$. Then,

$$C^\bullet(x) = x\exp(B'(C^\bullet(x))), \quad \text{where } C^\bullet(x) = xC'(x),$$

provided that $\mathcal{G}$ is **block-stable**, i.e., that a graph belongs to $\mathcal{C}$ iff its blocks belong to $\mathcal{B}$.



Decomposition of graphs into k-connected components

The 3-connected components of a 2-connected graph can also be defined, but the details are a little more involved.
There is also a relation between their generating functions.

Decomposition of graphs into k-connected components

The 3-connected components of a 2-connected graph can also be defined, but the details are a little more involved.
There is also a relation between their generating functions.

Unfortunately, the 4-connected components of 3-connected graphs cannot be defined in general (in such a way that the decomposition is unambiguous and well-defined).

Decomposition of graphs into k-connected components

The 3-connected components of a 2-connected graph can also be defined, but the details are a little more involved.
There is also a relation between their generating functions.

Unfortunately, the 4-connected components of 3-connected graphs cannot be defined in general (in such a way that the decomposition is unambiguous and well-defined).

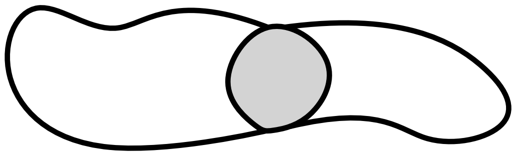
However, any k-connected **chordal** graph admits a decomposition into $(k+1)$-connected components! [Wormald, 1985]

Decomposition of chordal graphs into k-connected components

"Definition". **Slicing** through a k-separator:

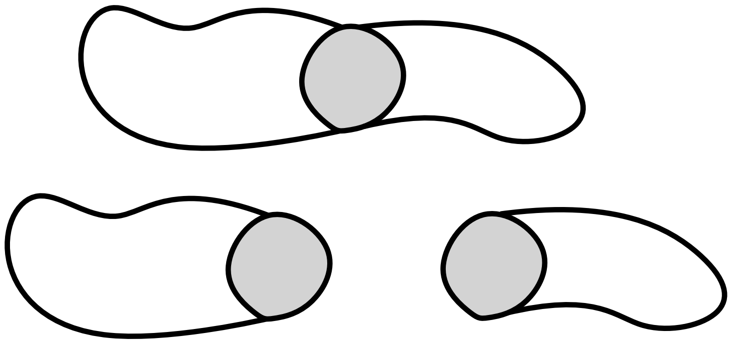
Decomposition of chordal graphs into k-connected components

"Definition". **Slicing** through a k-separator:



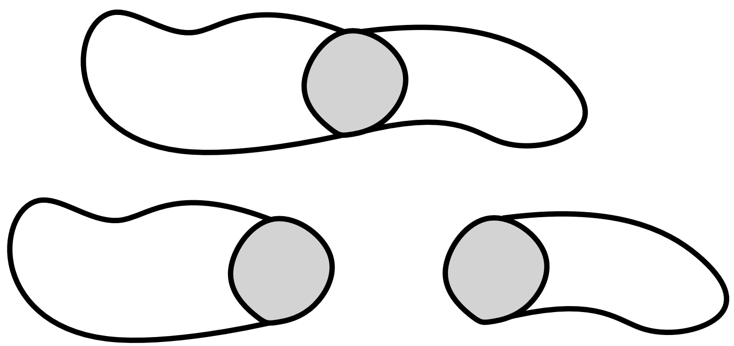
Decomposition of chordal graphs into k-connected components

"Definition". **Slicing** through a k-separator:



Decomposition of chordal graphs into k-connected components

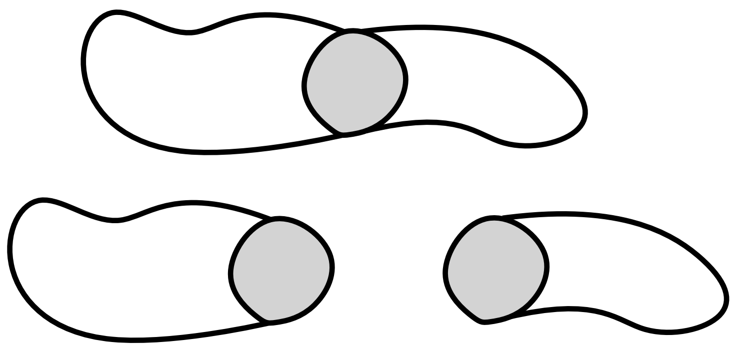
"Definition". **Slicing** through a k-separator:



"Definition". The $(k+1)$-**connected components** of a k-connected chordal graph are obtained by slicing it through all its k-separators (which are k-cliques).

Decomposition of chordal graphs into k-connected components

"Definition". **Slicing** through a k-separator:

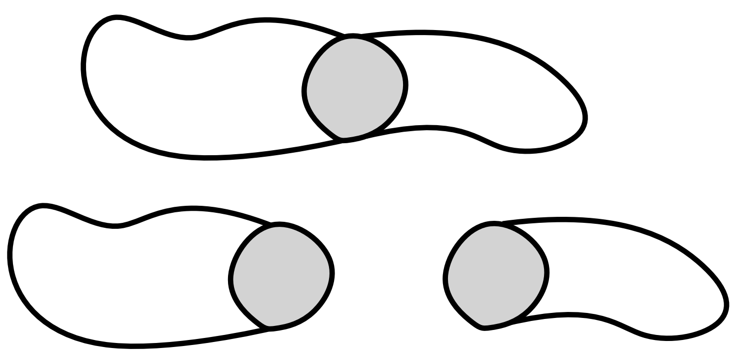


"Definition". The $(k+1)$-**connected components** of a k-connected chordal graph are obtained by slicing it through all its k-separators (which are k-cliques).

Proposition. This is well defined (the order does not matter, no k-separators appear or disappear in the process).

Decomposition of chordal graphs into k-connected components

"Definition". **Slicing** through a k-separator:

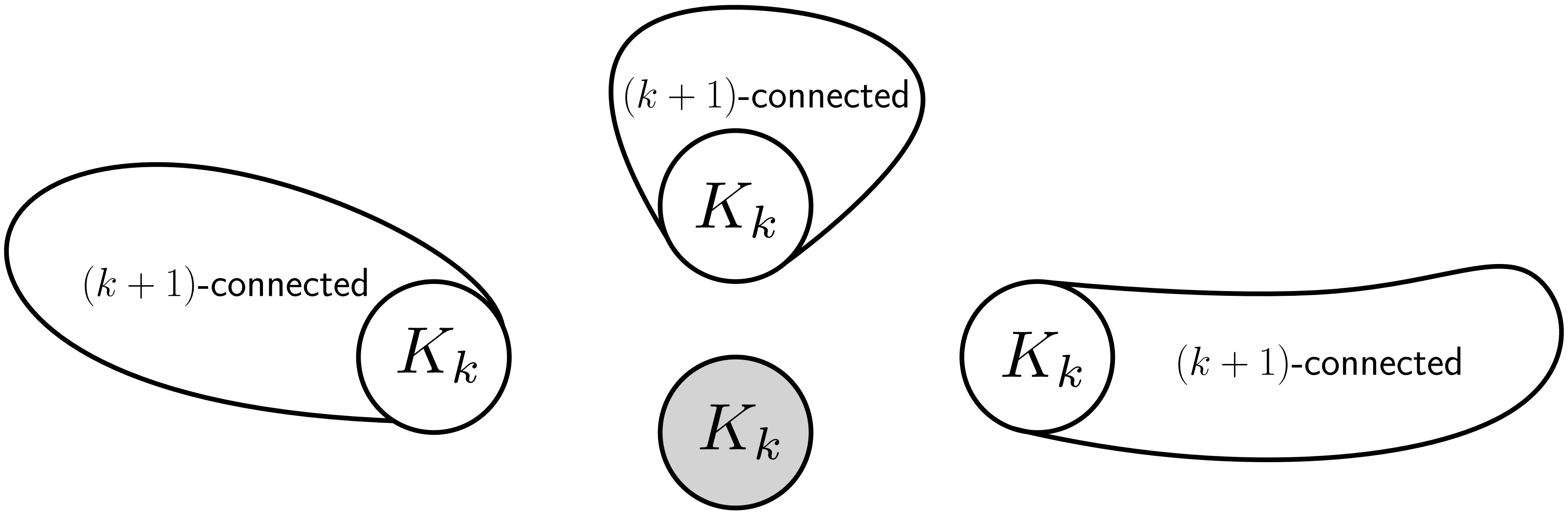


"Definition". The $(k+1)$-**connected components** of a k-connected chordal graph are obtained by slicing it through all its k-separators (which are k-cliques).

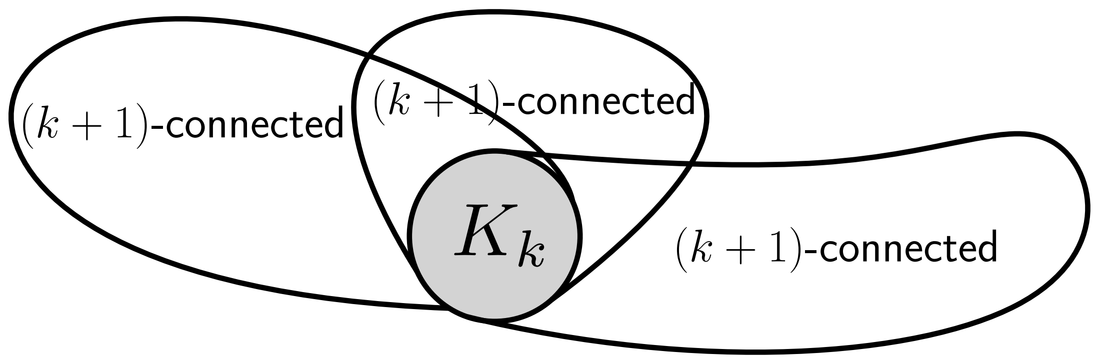
Proposition. This is well defined (the order does not matter, no k-separators appear or disappear in the process).

$\rightarrow$ Note that the $(k+1)$-connected components are the **maximal** $(k+1)$-**connected subgraphs**.

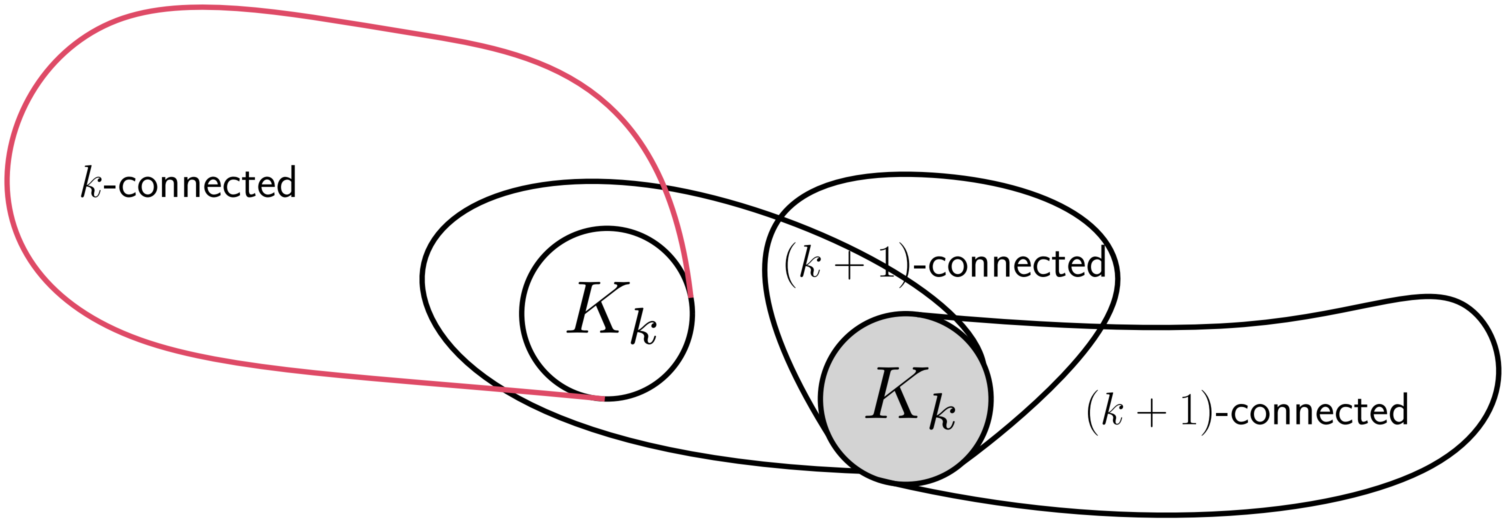
Decomposition of chordal graphs into k-connected components



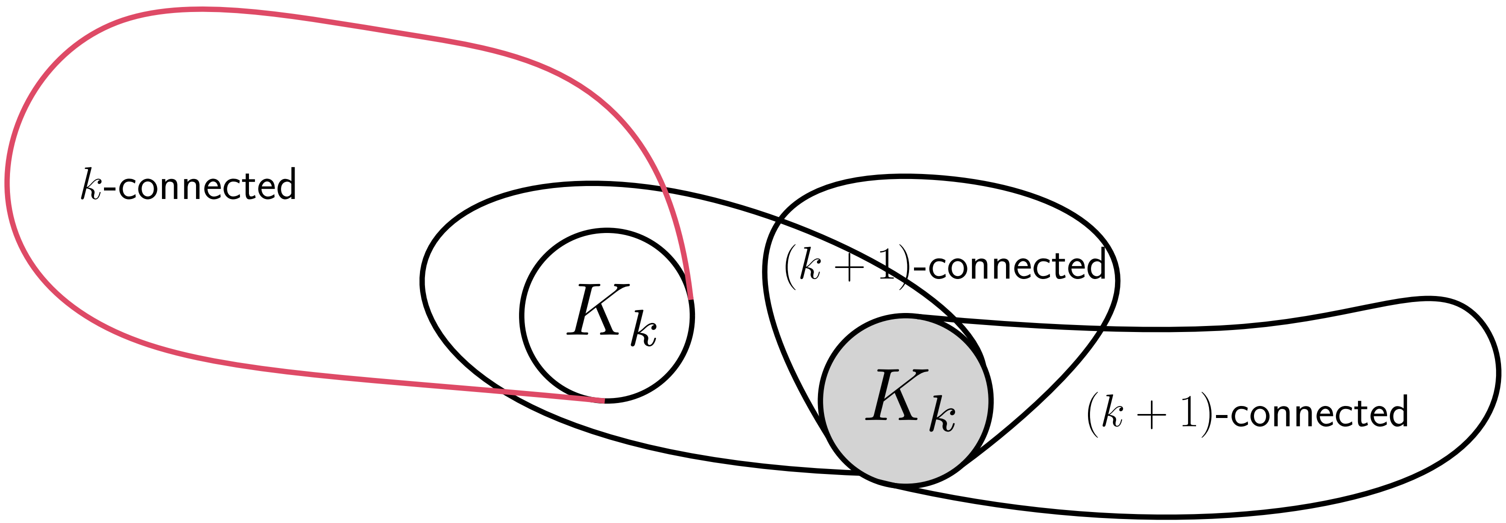
Decomposition of chordal graphs into k-connected components



Decomposition of chordal graphs into k-connected components



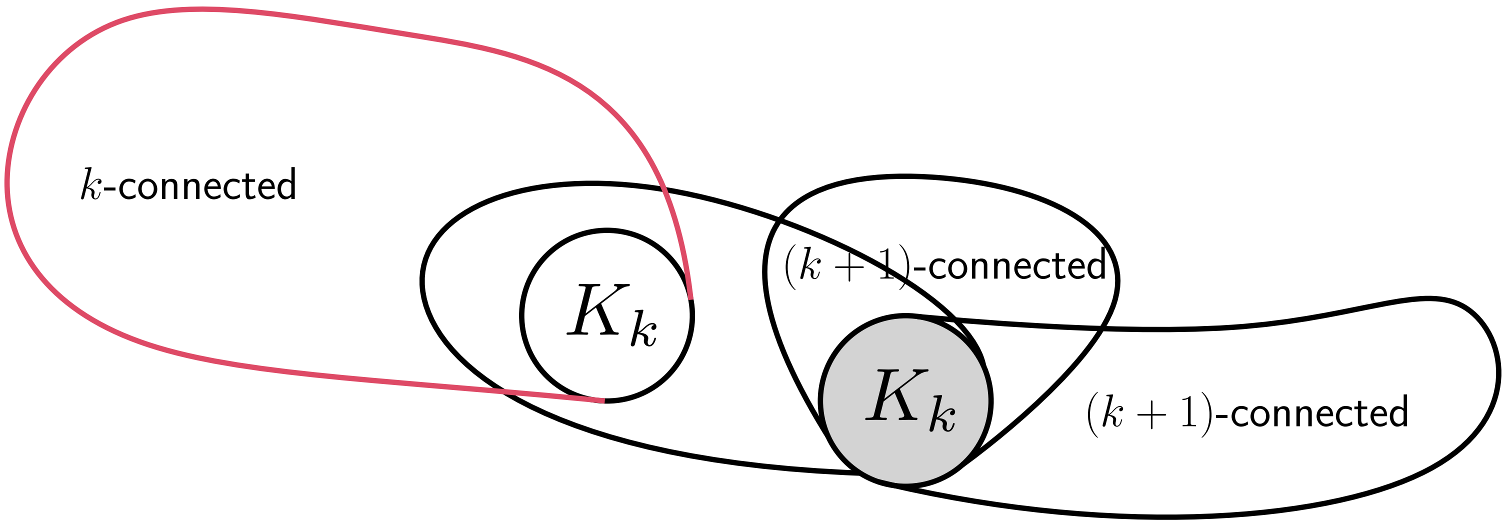
Decomposition of chordal graphs into k-connected components



Let $\mathcal{G}_k^{(i)}$ be the class of k-connected chordal graphs rooted at an unlabelled, ordered i-clique.
Consider its multivariate exponential generating function $G_k^{(j)}(x, x_k)$, where the variable x_k marks the number of k-cliques. Then, we have that

$$G_k^{(k)}(x, x_k) = \exp\left(G_{k+1}^{(k)}(x, x_k G_k^{(k)}(x, x_k))\right).$$

Decomposition of chordal graphs into k-connected components



Let $\mathcal{G}_k^{(i)}$ be the class of k-connected chordal graphs rooted at an unlabelled, ordered i-clique.
Consider its multivariate exponential generating function $G_k^{(j)}(x, x_k)$, where the variable x_k marks the number of k-cliques. Then, we have that

$$G_k^{(k)}(x, x_k) = \exp\left(G_{k+1}^{(k)}(x, x_k G_k^{(k)}(x, x_k))\right).$$

This generalizes the classical decomposition of connected graphs into 2-connected components.

Labelled vs unlabelled

A graph with n vertices is **labelled** if each vertex carries a different label in $\{1, 2, \ldots, n\}$.

Labelled vs unlabelled

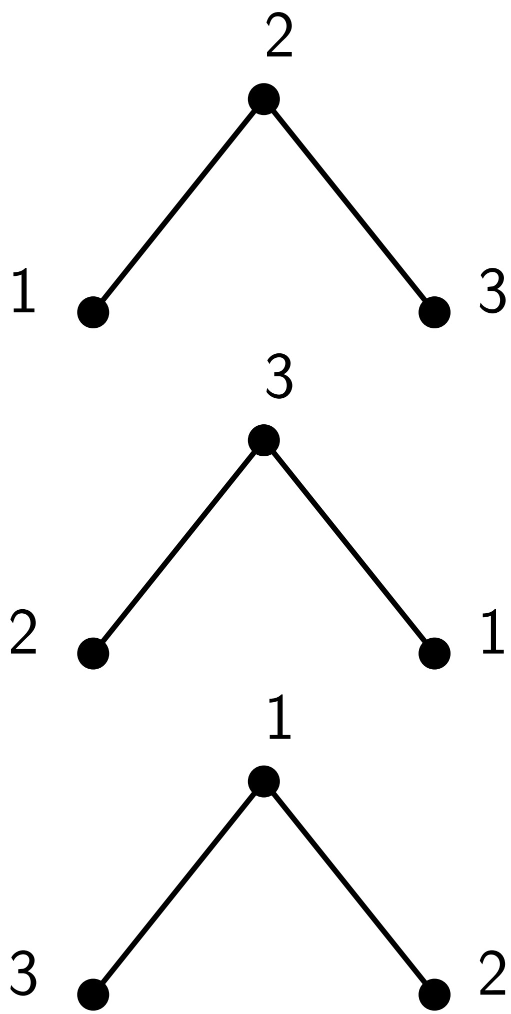
A graph with n vertices is **labelled** if each vertex carries a different label in $\{1, 2, \ldots, n\}$.

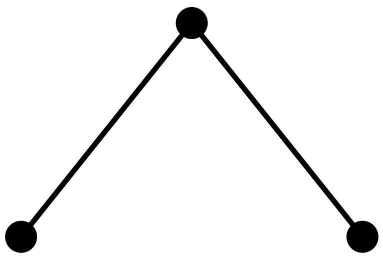
In an **unlabelled** graph, the vertices are undistinguishable.

Labelled vs unlabelled

A graph with n vertices is **labelled** if each vertex carries a different label in $\{1, 2, \ldots, n\}$.

In an **unlabelled** graph, the vertices are undistinguishable.

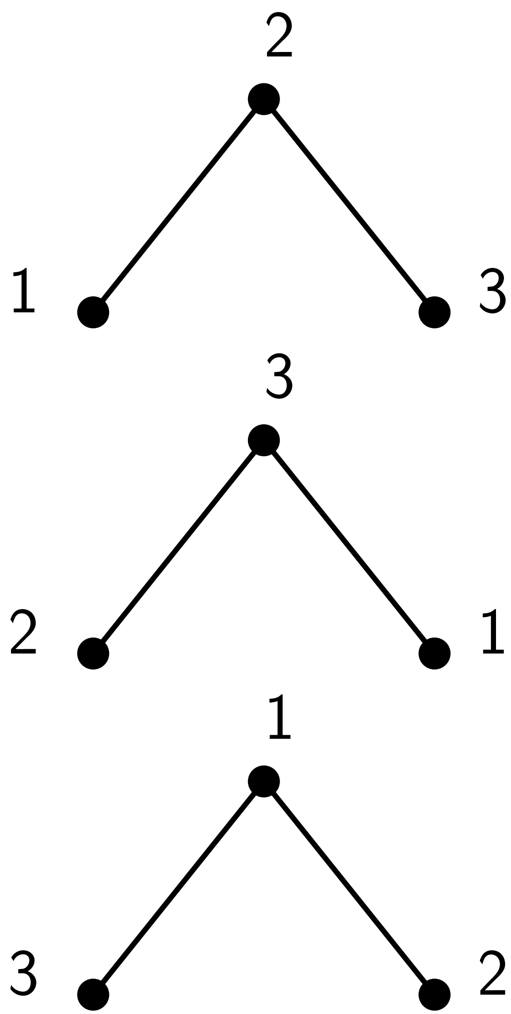


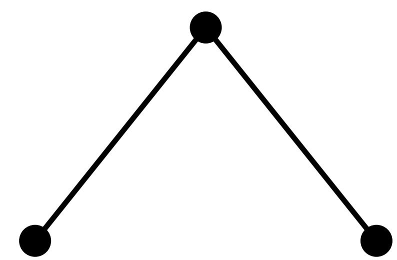


Labelled vs unlabelled

A graph with n vertices is **labelled** if each vertex carries a different label in $\{1, 2, \ldots, n\}$.

In an **unlabelled** graph, the vertices are undistinguishable.

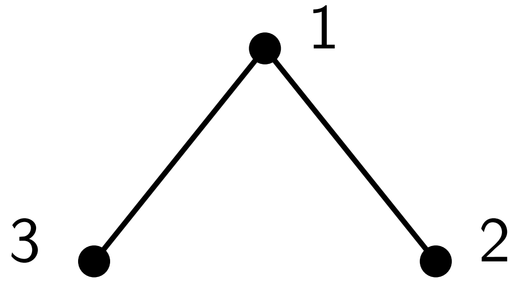




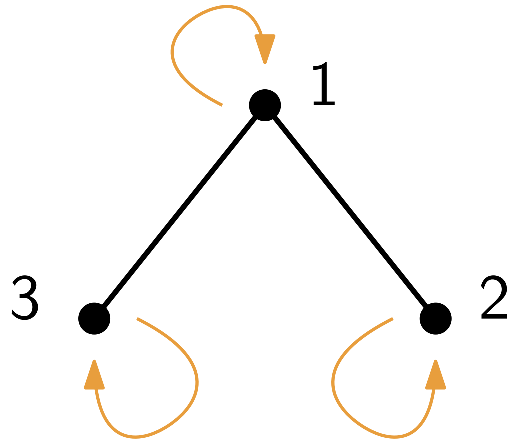
Unlabelled graphs are usually harder to count than labelled graphs.

- Labelled trees (Cayley trees) [Borchardt, 1860]
- Unlabelled trees (free trees) [Otter, 1948]

Counting unlabelled graphs - Pólya theory

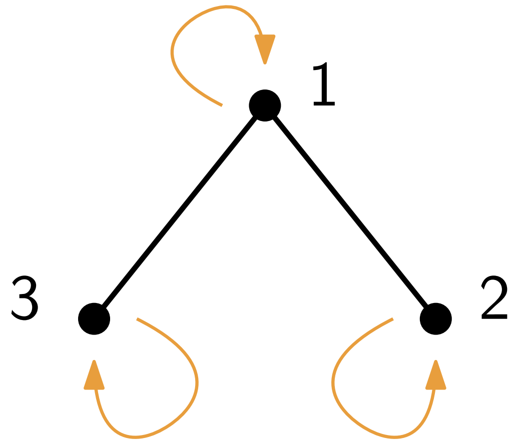


Counting unlabelled graphs - Pólya theory

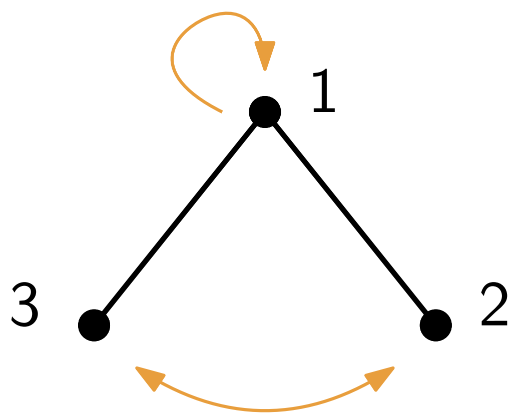


$(1)(2)(3) \longrightarrow s_1^3$

Counting unlabelled graphs - Pólya theory

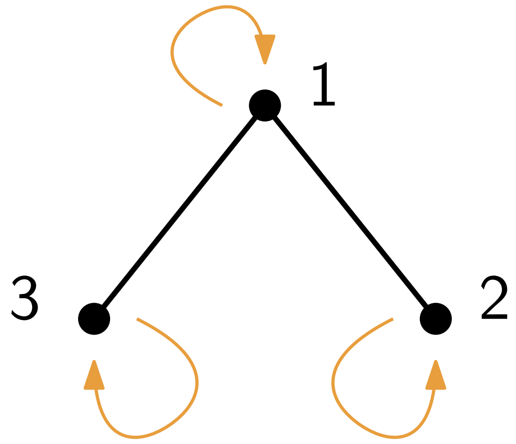


$(1)(2)(3) \longrightarrow s_1^3$

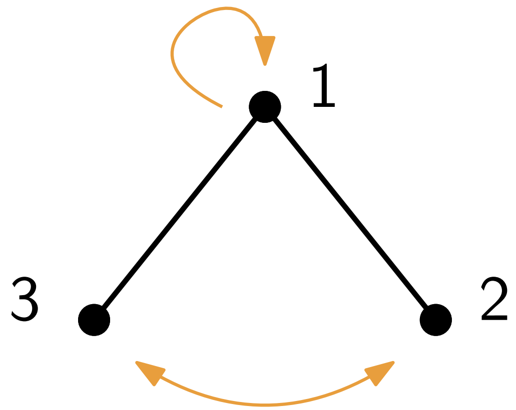


$(1)(23) \longrightarrow s_1 s_2$

Counting unlabelled graphs - Pólya theory



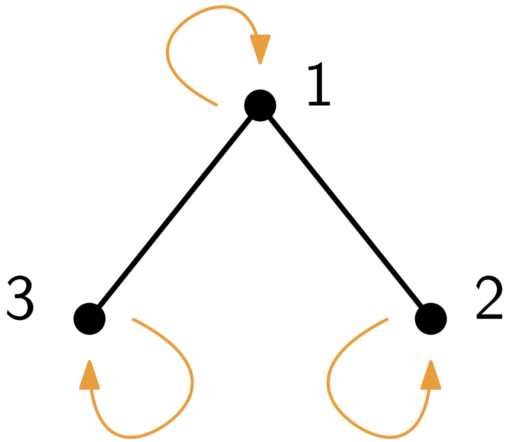
$(1)(2)(3) \longrightarrow s_1^3$



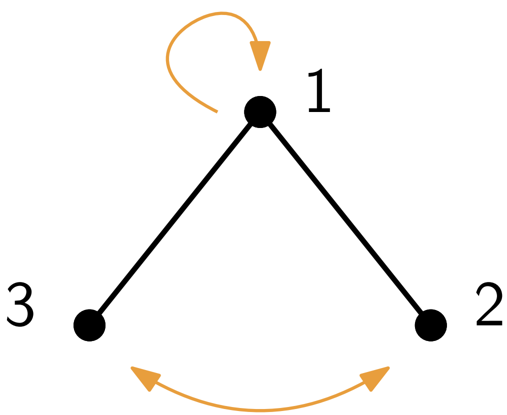
$(1)(23) \longrightarrow s_1 s_2$

$$\frac{1}{3!}(s_1^3 + s_1 s_2)$$

Counting unlabelled graphs - Pólya theory



$\color{orange}{(1)(2)(3)} \longrightarrow s_1^3$



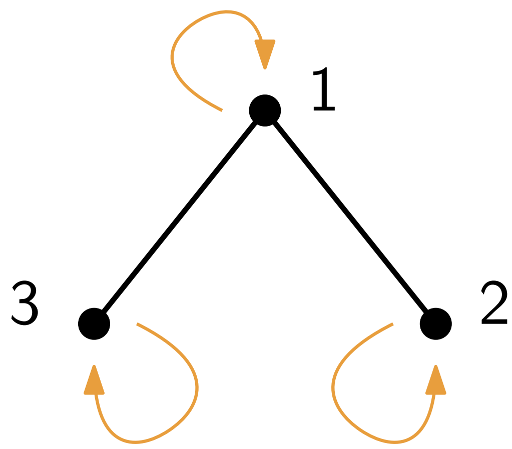
$\color{orange}{(1)(23)} \longrightarrow s_1 s_2$

$$\frac{1}{3!}(s_1^3 + s_1 s_2)$$

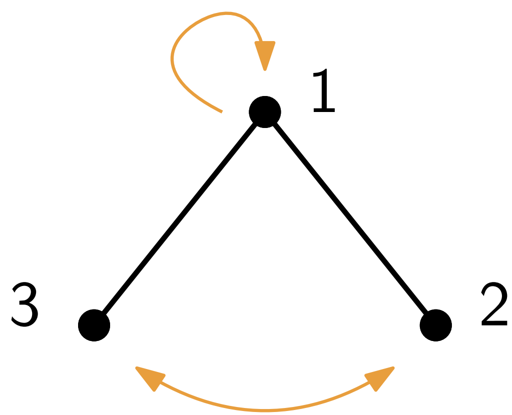
3 labelled graphs in the class

$$\frac{3}{3!}(s_1^3 + s_1 s_2)$$

Counting unlabelled graphs - Pólya theory



$(1)(2)(3) \longrightarrow s_1^3$



$(1)(23) \longrightarrow s_1 s_2$

$$\frac{1}{3!}(s_1^3 + s_1 s_2)$$

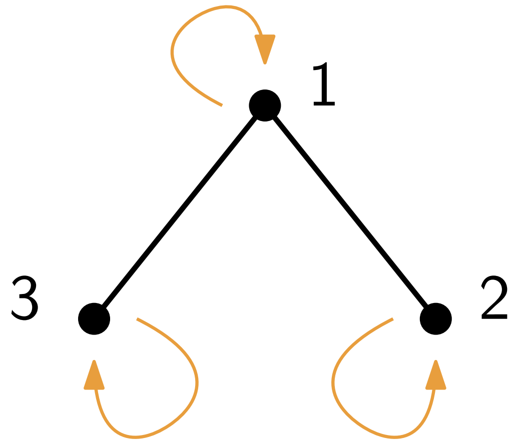
3 labelled graphs in the class

$$\frac{3}{3!}(s_1^3 + s_1 s_2)$$

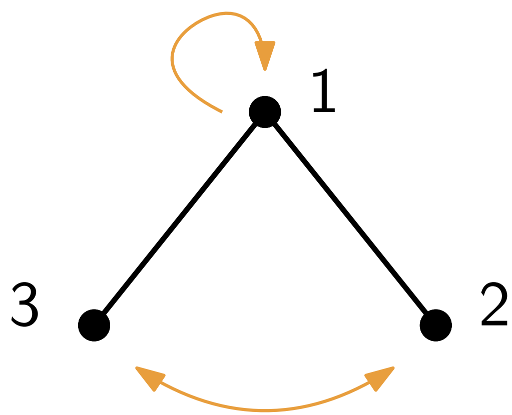
Cycle index sum

$Z_{\mathcal{G}}(s_1, s_2, s_3, \dots)$

Counting unlabelled graphs - Pólya theory



$(1)(2)(3) \longrightarrow s_1^3$



$(1)(23) \longrightarrow s_1 s_2$

$$\frac{1}{3!}(s_1^3 + s_1 s_2)$$

↓ 3 labelled graphs in the class

$$\frac{3}{3!}(s_1^3 + s_1 s_2)$$

Cycle index sum

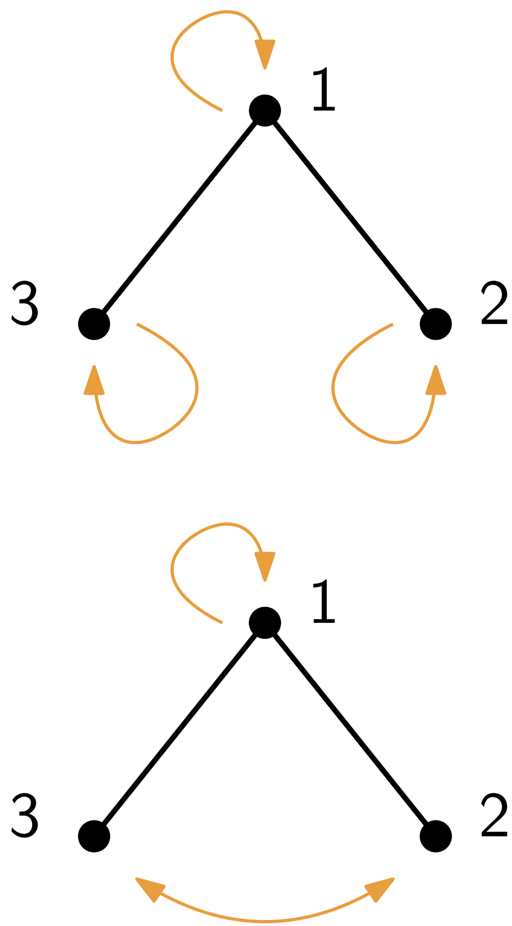
$Z_{\mathcal{G}}(s_1, s_2, s_3, \dots)$

Theorem [Pólya 1937]

The OGF of the unlabelled class $\tilde{\mathcal{G}}$ is given by

$$\tilde{G}(x) = Z_{\mathcal{G}}(x, x^2, x^3, \dots).$$

Counting unlabelled graphs - Pólya theory



$(1)(2)(3) \longrightarrow s_1^3$

$(1)(23) \longrightarrow s_1 s_2$

$$\frac{1}{3!}(s_1^3 + s_1 s_2)$$

3 labelled graphs in the class

$$\frac{3}{3!}(s_1^3 + s_1 s_2)$$

Cycle index sum

$Z_{\mathcal{G}}(s_1, s_2, s_3, \dots)$

Theorem [Pólya 1937]
The OGF of the unlabelled class $\tilde{\mathcal{G}}$ is given by

$$\tilde{G}(x) = Z_{\mathcal{G}}(x, x^2, x^3, \dots).$$

In our case,

$$G(x) = \frac{3}{3!}(x^3 + x \cdot x^2) = x^3$$

Unlabelled trees

Pólya trees: rooted, unlabelled trees.

Unlabelled trees

Pólya trees: rooted, unlabelled trees.

Theorem. [Pólya, 1937]
The OGF $P(x)$ of Pólya trees is given by

$$P(x) = x \exp\left(P(x) + \frac{P(x^2)}{2} + \frac{P(x^3)}{3} + \dots\right).$$

As $n \to \infty$ we have

$$[x^n]P(x) \sim \frac{b\sqrt{\rho}}{2\sqrt{\pi}} \cdot n^{-3/2} \cdot \rho^{-n},$$

with $b \approx 2.681127$ and $\rho \approx 0.338219$.

Unlabelled trees

Pólya trees: rooted, unlabelled trees.

Theorem. [Pólya, 1937]
The OGF $P(x)$ of Pólya trees is given by

$$P(x) = x \exp\left(P(x) + \frac{P(x^2)}{2} + \frac{P(x^3)}{3} + \dots\right).$$

As $n \to \infty$ we have

$$[x^n]P(x) \sim \frac{b\sqrt{\rho}}{2\sqrt{\pi}} \cdot n^{-3/2} \cdot \rho^{-n},$$

with $b \approx 2.681127$ and $\rho \approx 0.338219$.

What about unrooted unlabelled trees?

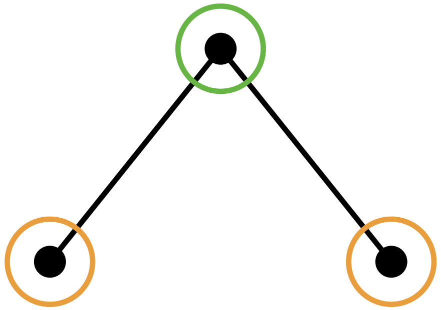
Unlabelled trees

Problem!

Unlabelled trees

Problem!

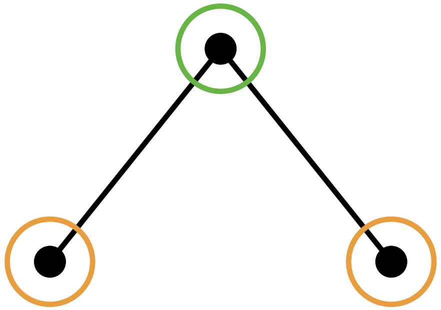
Rooting is biased in unlabelled graphs.
Not every unlabelled graph of size n gives rise to n rooted graphs.



Unlabelled trees

Problem!

Rooting is biased in unlabelled graphs.
Not every unlabelled graph of size n gives rise to n rooted graphs.



Theorem. [Otter, 1948]
The OGF $U(x)$ of unlabelled trees is given by

$$U(x) = P(x) + \frac{1}{2}(P(x^2) - P(x)^2).$$

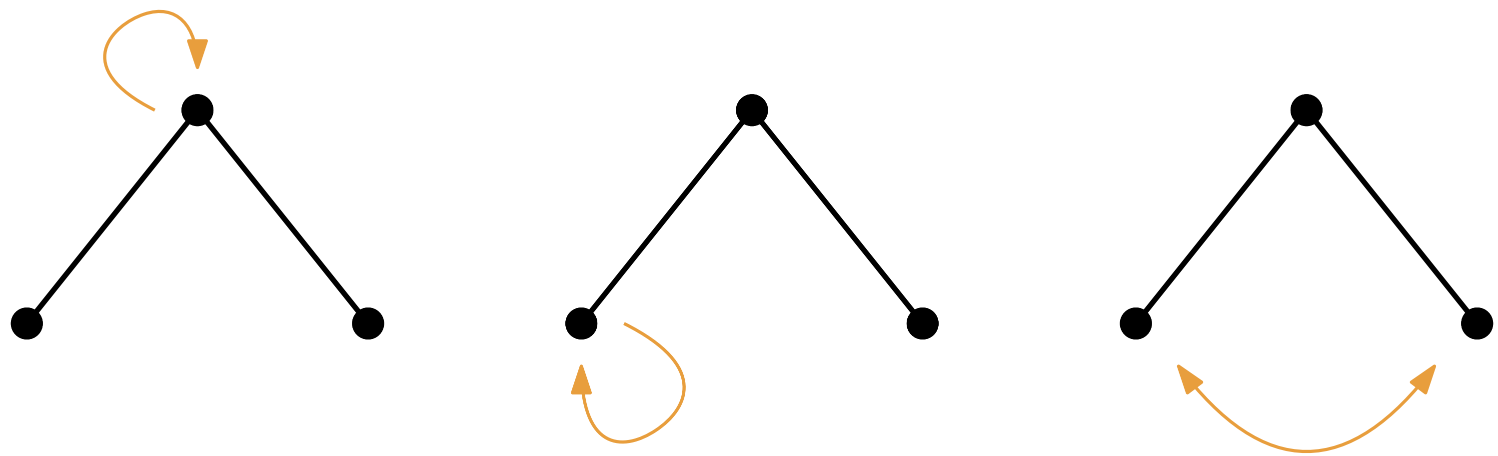
As $n \to \infty$ we have

$$[x^n]P(x) \sim \frac{b^3 \rho^{3/2}}{4\sqrt{\pi}} \cdot n^{-3/2} \cdot \rho^{-n},$$

with $b \approx 2.681127$ and $\rho \approx 0.338219$.

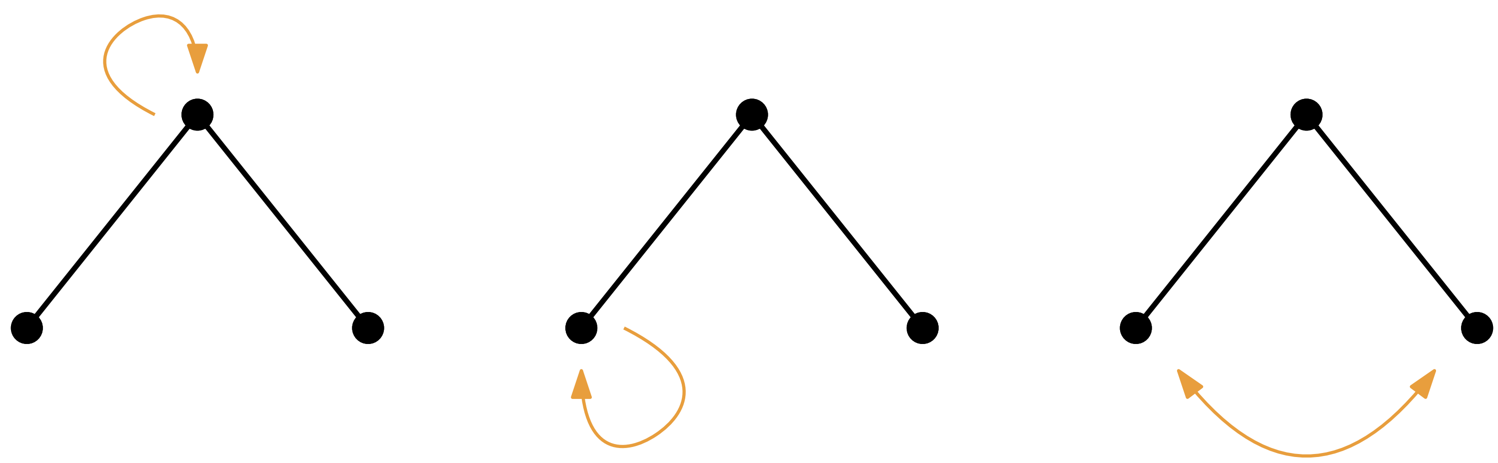
Proof. Using the **dissymmetry theorem**.

Cycle-pointing



Definition. A **cycle-pointed graph** is a pair (G, c) where $G \in \mathcal{G}$ is an unlabelled graph and c is a cycle of some automorphism of G.

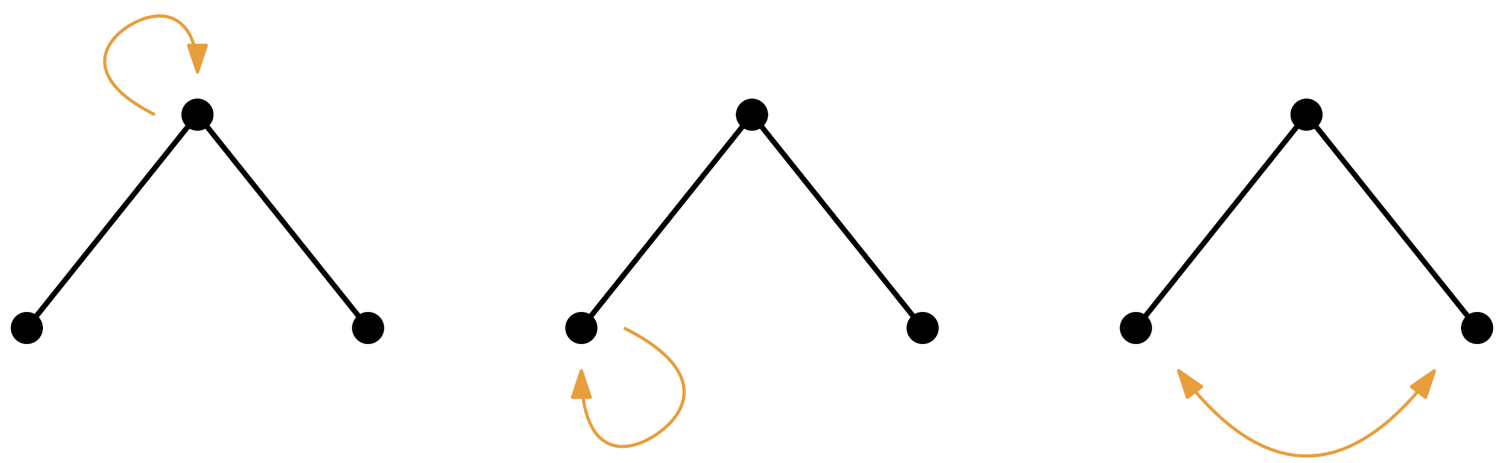
Cycle-pointing



Definition. A **cycle-pointed graph** is a pair (G, c) where $G \in \mathcal{G}$ is an unlabelled graph and c is a cycle of some automorphism of G.

Theorem. [Bodirsky, Fusy, Kang & Vigerske (2007)]
Every unlabelled graph $G \in \mathcal{G}$ of size n admits exactly n cycle-pointings.

Cycle-pointing

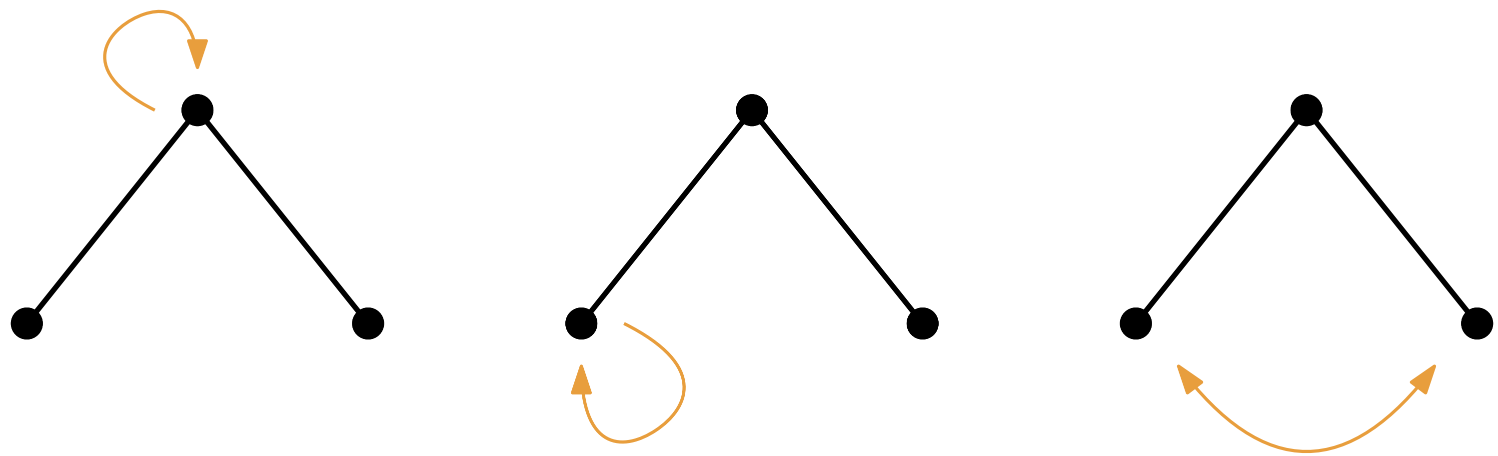


Definition. A **cycle-pointed graph** is a pair (G, c) where $G \in \mathcal{G}$ is an unlabelled graph and c is a cycle of some automorphism of G.

Theorem. [Bodirsky, Fusy, Kang & Vigerske (2007)]
Every unlabelled graph $G \in \mathcal{G}$ of size n admits exactly n cycle-pointings.

An unbiased rooting (pointing) operator!

Cycle-pointing



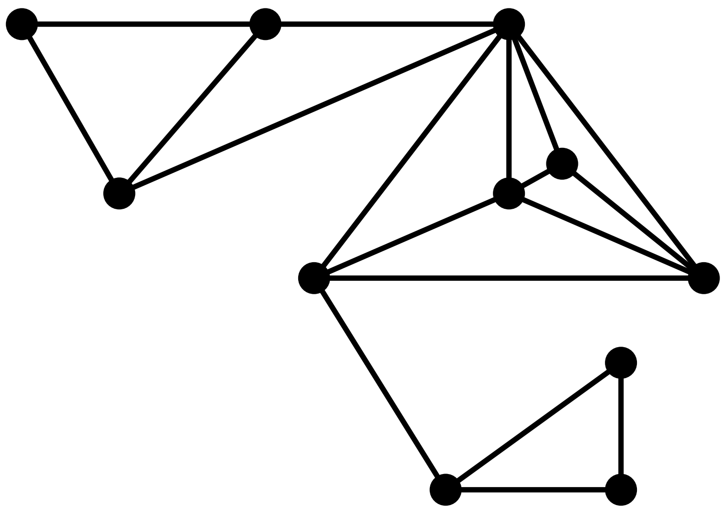
Definition. A **cycle-pointed graph** is a pair (G, c) where $G \in \mathcal{G}$ is an unlabelled graph and c is a cycle of some automorphism of G.

Theorem. [Bodirsky, Fusy, Kang & Vigerske (2007)]
Every unlabelled graph $G \in \mathcal{G}$ of size n admits exactly n cycle-pointings.

An unbiased rooting (pointing) operator!

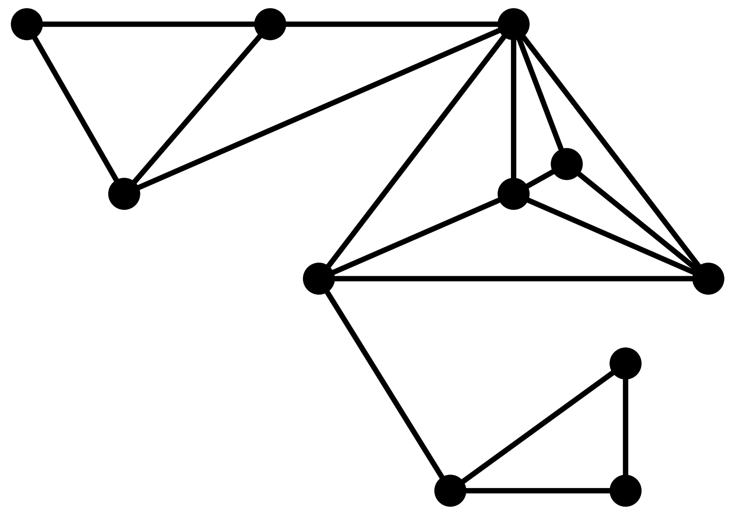
They extend Pólya theory to cycle-pointed graphs. In particular, they manage to unroot Pólya trees via cycle-pointing and they recover Otter's formula.

Our class of graphs



Chordal graphs with tree-width at most t

Our class of graphs



Chordal graphs with tree-width at most t

[C., Drmota, Noy & Requilé, 2023]: assymptotic enumeration of the labelled class.

$$|\mathcal{G}_{t,n}| \sim c_t \cdot n^{-5/2} \cdot \gamma_t^n \cdot n! \qquad \text{as } n \to \infty,$$

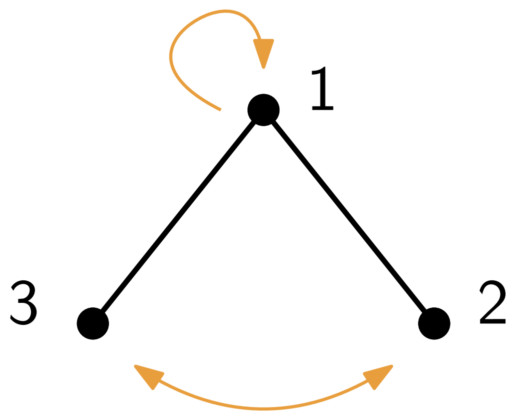
for some $c_t > 0$ and $\gamma_t > 1$

An extension of Pólya theory

We need to take into account **cycles of cliques**, not just vertices.

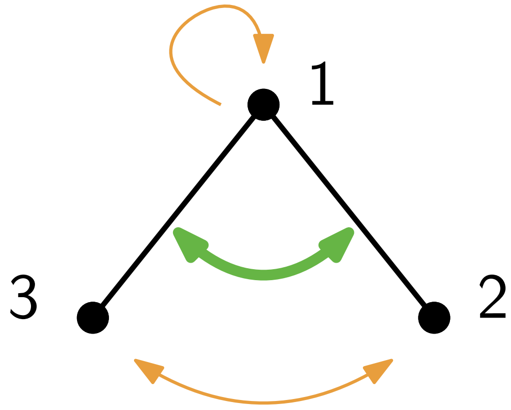
An extension of Pólya theory

We need to take into account **cycles of cliques**, not just vertices.



An extension of Pólya theory

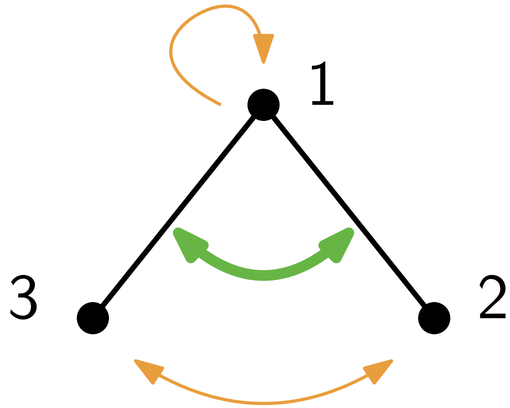
We need to take into account **cycles of cliques**, not just vertices.



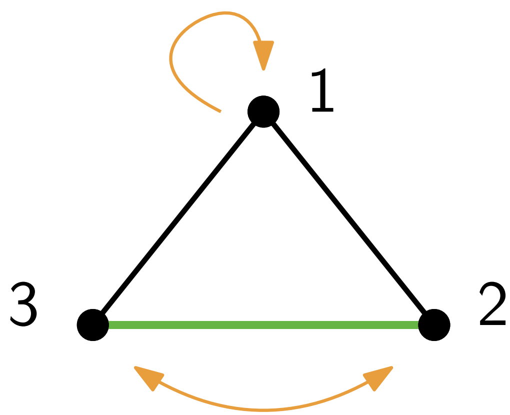
The **edges** are in a cycle of length 2.

An extension of Pólya theory

We need to take into account **cycles of cliques**, not just vertices.



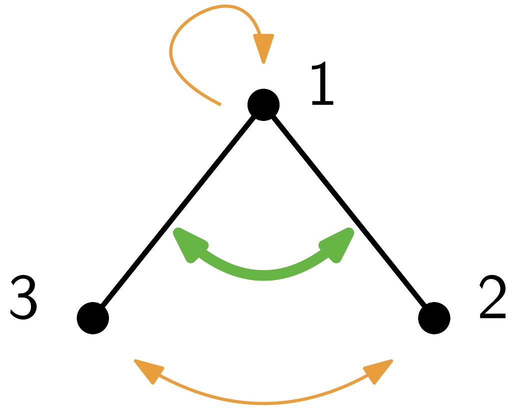
The **edges** are in a cycle of length 2.



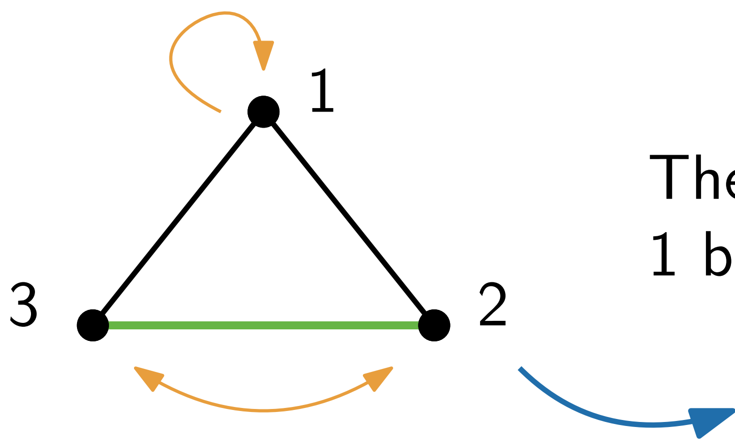
The new edge is in a cycle of length 1 but different type: it flips itself.

An extension of Pólya theory

We need to take into account **cycles of cliques**, not just vertices.



The **edges** are in a cycle of length 2.

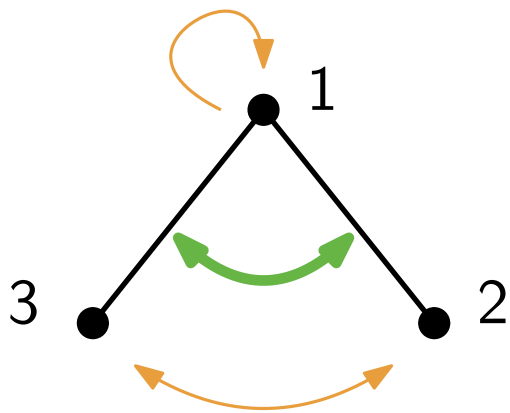


The new edge is in a cycle of length 1 but different type: it flips itself.

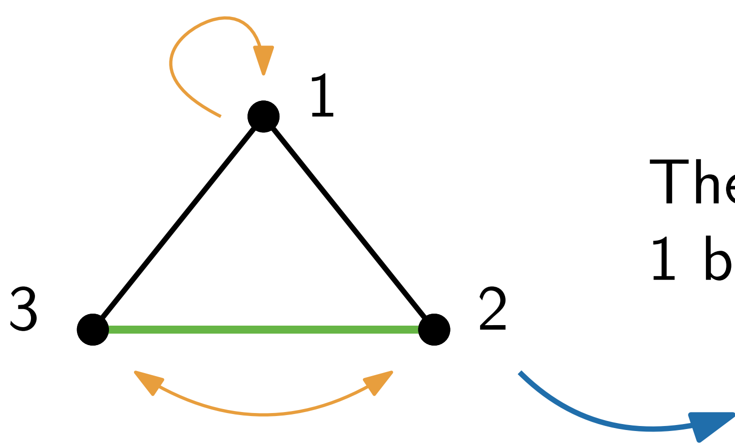
$s_{(1),1}\, s_{(1),2}\, s_{(1,1),2}\, s_{(2),1}$

An extension of Pólya theory

We need to take into account **cycles of cliques**, not just vertices.



The **edges** are in a cycle of length 2.



The new edge is in a cycle of length 1 but different type: it flips itself.

$s_{(1),1} s_{(1),2} s_{(1,1),2} s_{(2),1}$

What we do:

- Refinement of cycle index sums to encode cycles of cliques.
- Extend cycle-pointing to cycles of cliques.

Example: composition

Classic setting: EGF, labelled graphs, substitution of vertices. If $\mathcal{C} = \mathcal{A} \circ \mathcal{B}$

$$C(x) = A(B(X)).$$

Example: composition

Classic setting: EGF, labelled graphs, substitution of vertices. If $\mathcal{C} = \mathcal{A} \circ \mathcal{B}$

$$C(x) = A(B(X)).$$

Pólya setting: Cycle index sum, unlabelled graphs, substitution of vertices. If $\mathcal{C} = \mathcal{A} \circ \mathcal{B}$,

$$\begin{aligned} Z_{\mathcal{C}} &= Z_{\mathcal{A}}(Z_{\mathcal{B}}(s_1, s_2, s_3, \dots), Z_{\mathcal{B}}(s_2, s_4, s_6, \dots), \dots) \\ &= Z_{\mathcal{A}}(s_j \to Z_{\mathcal{B}}(s_j, s_{2j}, s_{3j}, \dots))_{j \geq 1} \\ &= Z_{\mathcal{A}}(s_j \to Z_{\mathcal{B}}^{[j]})_{j \geq 1} \end{aligned}$$

Example: composition

Classic setting: EGF, labelled graphs, substitution of vertices. If $\mathcal{C} = \mathcal{A} \circ \mathcal{B}$

$$C(x) = A(B(X)).$$

Pólya setting: Cycle index sum, unlabelled graphs, substitution of vertices. If $\mathcal{C} = \mathcal{A} \circ \mathcal{B}$,

$$\begin{aligned} Z_{\mathcal{C}} &= Z_{\mathcal{A}}(Z_{\mathcal{B}}(s_1, s_2, s_3, \ldots), Z_{\mathcal{B}}(s_2, s_4, s_6, \ldots), \ldots) \\ &= Z_{\mathcal{A}}(s_j \to Z_{\mathcal{B}}(s_j, s_{2j}, s_{3j}, \ldots))_{j \geq 1} \\ &= Z_{\mathcal{A}}(s_j \to Z_{\mathcal{B}}^{[j]})_{j \geq 1} \end{aligned}$$

Our setting: Extended cycle index sum, unlabelled graphs, substitution of cliques. If $\mathcal{C} = \mathcal{A} \circ_k \mathcal{B}$,

$$X_{\mathcal{C}} = Z_{\mathcal{A}}(s_{\lambda,j} \to (X_{\mathcal{B}}^{\lambda})^{[j]})_{\lambda \vdash k, j \geq 1}$$

Example: composition

Classic setting: EGF, labelled graphs, substitution of vertices. If $\mathcal{C} = \mathcal{A} \circ \mathcal{B}$

$$C(x) = A(B(X)).$$

Pólya setting: Cycle index sum, unlabelled graphs, substitution of vertices. If $\mathcal{C} = \mathcal{A} \circ \mathcal{B}$,

$$\begin{aligned} Z_{\mathcal{C}} &= Z_{\mathcal{A}}(Z_{\mathcal{B}}(s_1, s_2, s_3, \dots), Z_{\mathcal{B}}(s_2, s_4, s_6, \dots), \dots) \\ &= Z_{\mathcal{A}}(s_j \to Z_{\mathcal{B}}(s_j, s_{2j}, s_{3j}, \dots))_{j \geq 1} \\ &= Z_{\mathcal{A}}(s_j \to Z_{\mathcal{B}}^{[j]})_{j \geq 1} \end{aligned}$$

Our setting: Extended cycle index sum, unlabelled graphs, substitution of cliques. If $\mathcal{C} = \mathcal{A} \circ_k \mathcal{B}$,

$$X_{\mathcal{C}} = Z_{\mathcal{A}}(s_{\lambda,j} \to (X_{\mathcal{B}}^{\lambda})^{[j]})_{\lambda \vdash k, j \geq 1}$$

With cycle-pointing:

$$X_{\mathcal{P}} \odot_i (X_{\mathcal{A}}, X_{\mathcal{Q}}) := X_{\mathcal{P}}(s_{\lambda,j} \to (X_{\mathcal{A}}^{\lambda})^{[j]}, t_{\mu,l} \to (X_{\mathcal{Q}}^{\mu})^{[l]})$$

The system

$$
\begin{cases}
X^{\lambda}_{\mathcal{G}^{(k)}_{t,k+1}} = \frac{k!}{\alpha(\lambda)\kappa(\lambda)} \frac{\partial}{\partial s_{\lambda,1}} X_{\mathcal{G}_{t,k+1}}, \\
X^{\lambda}_{\mathcal{G}^{(k)}_{t,k}} = Z_{\mathrm{SET}}(s_j \to (X^{\lambda^j}_{\mathcal{G}^{(k)}_{t,k+1} \circ_k \mathcal{G}^{(k)}_{t,k}})^{[j]})_{j \geq 1}, \\
X^{\lambda}_{\mathcal{G}^{(k)}_{t,k+1} \circ_k \mathcal{G}^{(k)}_{t,k}} = X^{\lambda}_{\mathcal{G}^{(k)}_{t,k+1}} (s_{\mu,j} \to (X^{\mu}_{\mathcal{G}^{(k)}_{t,k}})^{[j]})_{\mu \vdash k, j \geq 1}, \\
X_{\mathcal{G}^{\bullet_k}_{t,k}} = \sum_{\mu \vdash k} \frac{\alpha(\mu)\kappa(\mu)}{k!} t_{\mu,1} X^{\mu}_{\mathcal{G}^{(k)}_{t,k}} + X_{(\mathcal{G}_{t,k})^{\bullet_k}_{\geq 2}}, \\
X_{(\mathcal{G}_{t,k})^{\bullet_k}_{\geq 2}} = X_{(\mathcal{G}_{t,k+1})^{\bullet_k}_{\geq 2}} (s_{\mu,j} \to (X^{\mu}_{\mathcal{G}^{(k)}_{t,k}})^{[j]}, t_{\mu,j} \to (X^{\mu}_{(\mathcal{G}^{(k)}_{t,k})^{\bullet_k}})^{[j]})_{\mu \vdash k, j \geq 1} \\
\qquad + \sum_{\mu \vdash k} \frac{\alpha(\mu)\kappa(\mu)}{k!} s_{\mu,1} Z_{\mathrm{SET}^{\bullet}_{\geq 2}} (s_j \to (X^{\mu^j}_{\mathcal{G}^{(k)}_{t,k+1} \circ_k \mathcal{G}^{(k)}_{t,k}})^{[j]}, \\
\qquad\qquad t_j \to (X^{\mu^j}_{(\mathcal{G}^{(k)}_{t,k+1} \circ_k \mathcal{G}^{(k)}_{t,k})^{\bullet_k}})^{[j]})_{j \geq 1}, \\
X_{\mathcal{G}_{t,k}} = \sum_{\lambda \vdash k} \sum_{1 \leq j \leq \binom{t+1}{k}} \int \frac{1}{j t_{\lambda,j}} X_{\mathcal{G}^{\bullet_k}_{t,k}} (S(\lambda,j) \to 0, T(\lambda,j) \to 0) ds_{\lambda,j}
\end{cases}
$$

Results

- [Harary & Palmer (1968)], [Fowler, Gessel, Labelle & Leroux (2002)]: asymptotic enumeration of unlabelled **2-trees**

Results

- [Harary & Palmer (1968)], [Fowler, Gessel, Labelle & Leroux (2002)]: asymptotic enumeration of unlabelled **2-trees**
- [Gainer-Dewar (2012)], [Gainer-Dewar & Gessel (2014)]: system of equations determining the OGF of unlabelled k-**trees**.

Results

- [Harary & Palmer (1968)], [Fowler, Gessel, Labelle & Leroux (2002)]: asymptotic enumeration of unlabelled **2-trees**
- [Gainer-Dewar (2012)], [Gainer-Dewar & Gessel (2014)]: system of equations determining the OGF of unlabelled k-**trees**.
- [Drmota & Yu Jin (2014)]: asymptotic enumeration of unlabelled k-**trees**.

Results

- [Harary & Palmer (1968)], [Fowler, Gessel, Labelle & Leroux (2002)]: asymptotic enumeration of unlabelled **2-trees**
- [Gainer-Dewar (2012)], [Gainer-Dewar & Gessel (2014)]: system of equations determining the OGF of unlabelled k-**trees**.
- [Drmota & Yu Jin (2014)]: asymptotic enumeration of unlabelled k-**trees**.

This talk: generalisation of previous results.

- [C. & Requilé (2024+)]: system of equations to compute the OGF of unlabelled **chordal graphs with tree-width** $\leq t$.

Results

- [Harary & Palmer (1968)], [Fowler, Gessel, Labelle & Leroux (2002)]: asymptotic enumeration of unlabelled **2-trees**
- [Gainer-Dewar (2012)], [Gainer-Dewar & Gessel (2014)]: system of equations determining the OGF of unlabelled k-**trees**.
- [Drmota & Yu Jin (2014)]: asymptotic enumeration of unlabelled k-**trees**.

This talk: generalisation of previous results.

- [C. & Requilé (2024+)]: system of equations to compute the OGF of unlabelled **chordal graphs with tree-width** $\leq t$.

Future:

- [C.,Drmota & Requilé (soon?)]: asymptotic enumeration of unlabelled chordal graphs with bounded tree-width.

Concluding remarks

What is this machinery useful for?

To enumerate **classes of unlabelled graphs that arise from the identification of cliques**.

Concluding remarks

What is this machinery useful for?
To enumerate **classes of unlabelled graphs that arise from the identification of cliques**.

Other applications:

- Counting **unlabelled chordal planar graphs**.
- The decomposition of 2-connected graphs into 3-connected components. [Walsh, 1978]
  However, Walsh used dissymmetry and with cycle-pointing we could unroot without subtraction.

Concluding remarks

What is this machinery useful for?

To enumerate **classes of unlabelled graphs that arise from the identification of cliques**.

Other applications:

- Counting **unlabelled chordal planar graphs**.
- The decomposition of 2-connected graphs into 3-connected components. [Walsh, 1978]
  However, Walsh used dissymmetry and with cycle-pointing we could unroot without subtraction.

Thank you!