Enumeration of unlabelled chordal graphs with bounded tree-width

Jordi Castellví (CRM)

Work in collaboration with Clément Requilé

CENTRE DE RECERCA MATEMÀTICA

EXCELENCIA MARÍA DE MAEZTU 2022 - 2025

DMD 2024 - Alcalá de Henares

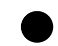
Introduction

How to build a tree?

Introduction

How to build a tree?

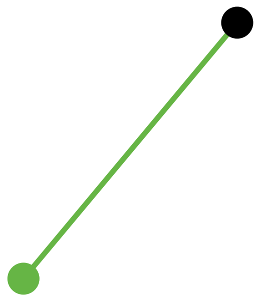
Iteratively add a new vertex connected to an existing vertex.



Introduction

How to build a tree?

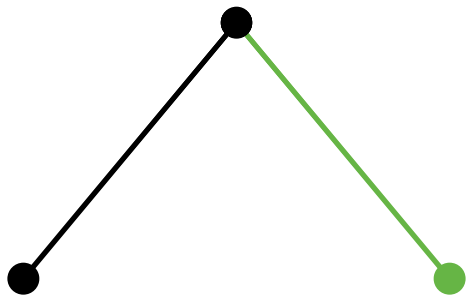
Iteratively add a new vertex connected to an existing vertex.



Introduction

How to build a tree?

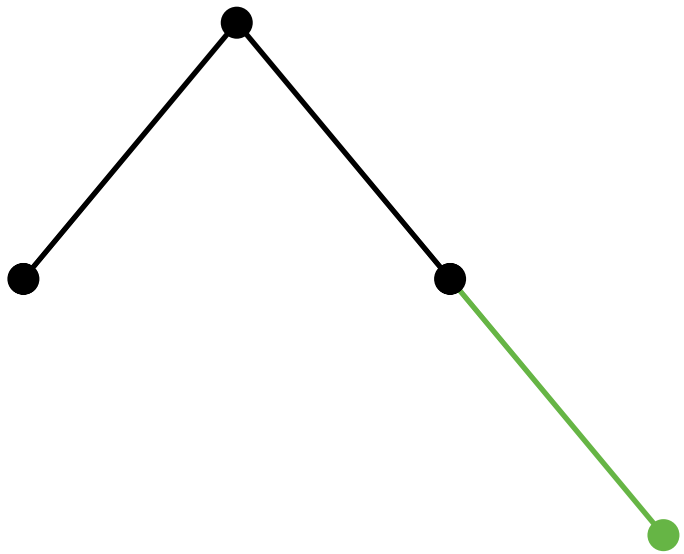
Iteratively add a new vertex connected to an existing vertex.



Introduction

How to build a tree?

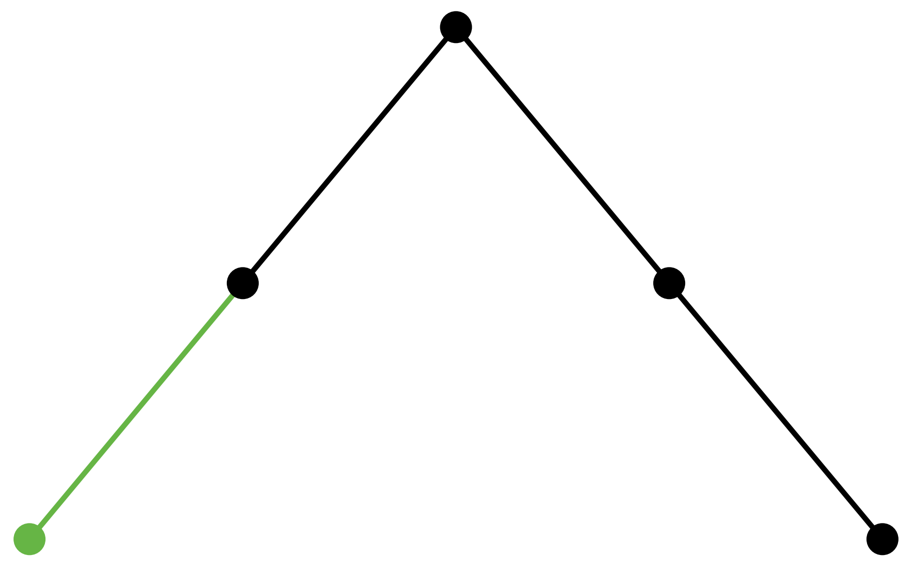
Iteratively add a new vertex connected to an existing vertex.



Introduction

How to build a tree?

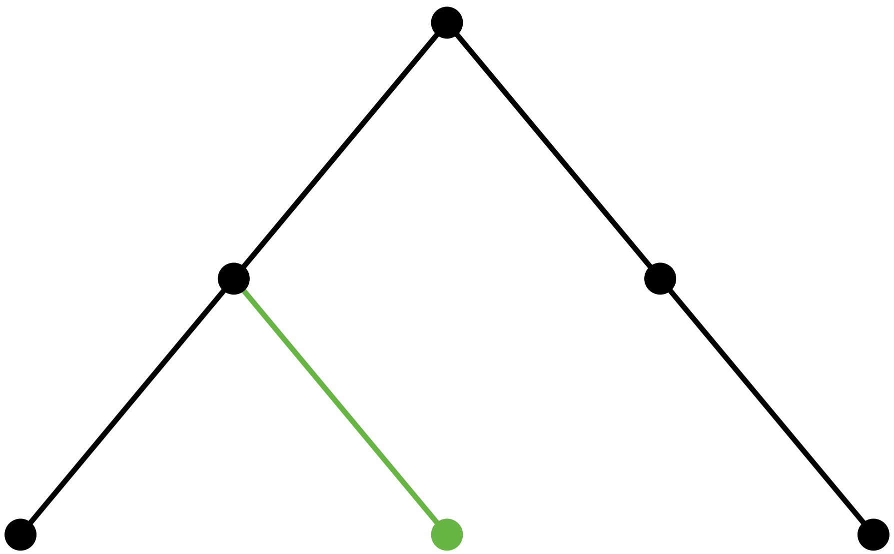
Iteratively add a new vertex connected to an existing vertex.



Introduction

How to build a tree?

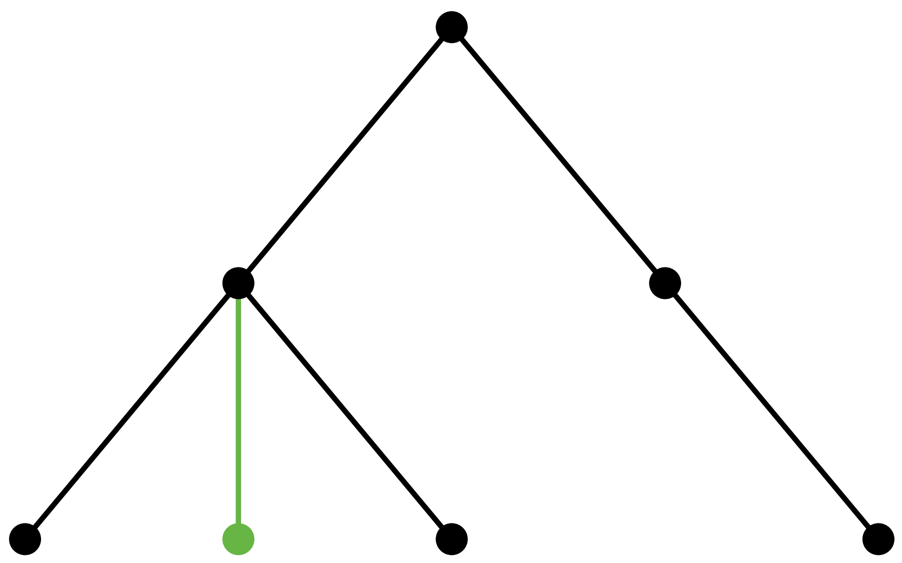
Iteratively add a new vertex connected to an existing vertex.



Introduction

How to build a tree?

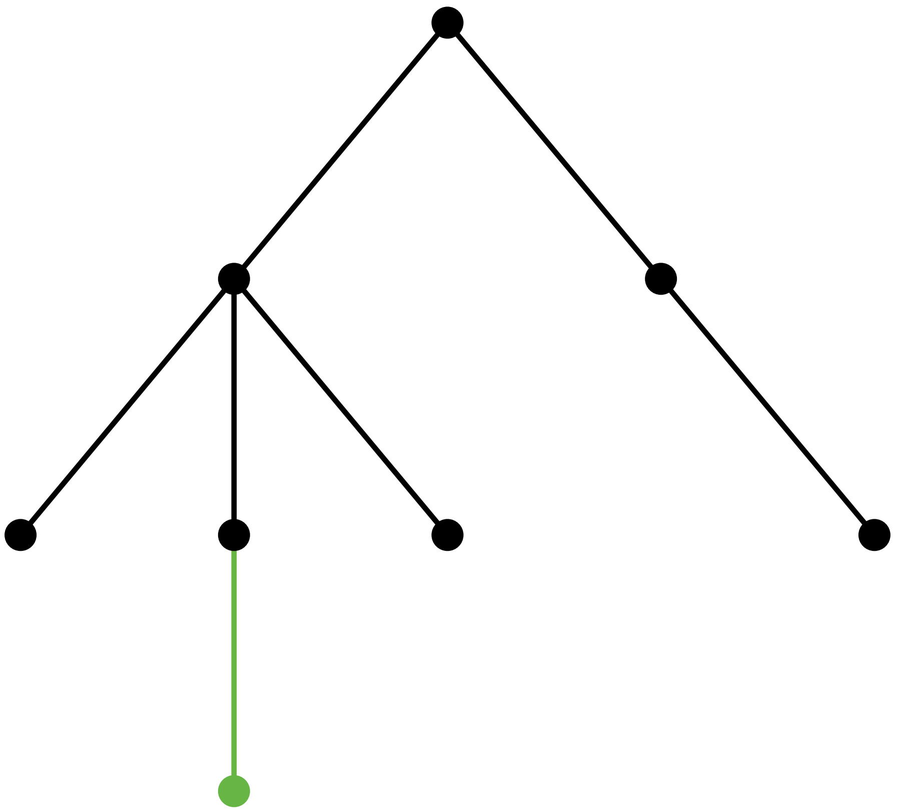
Iteratively add a new vertex connected to an existing vertex.



Introduction

How to build a tree?

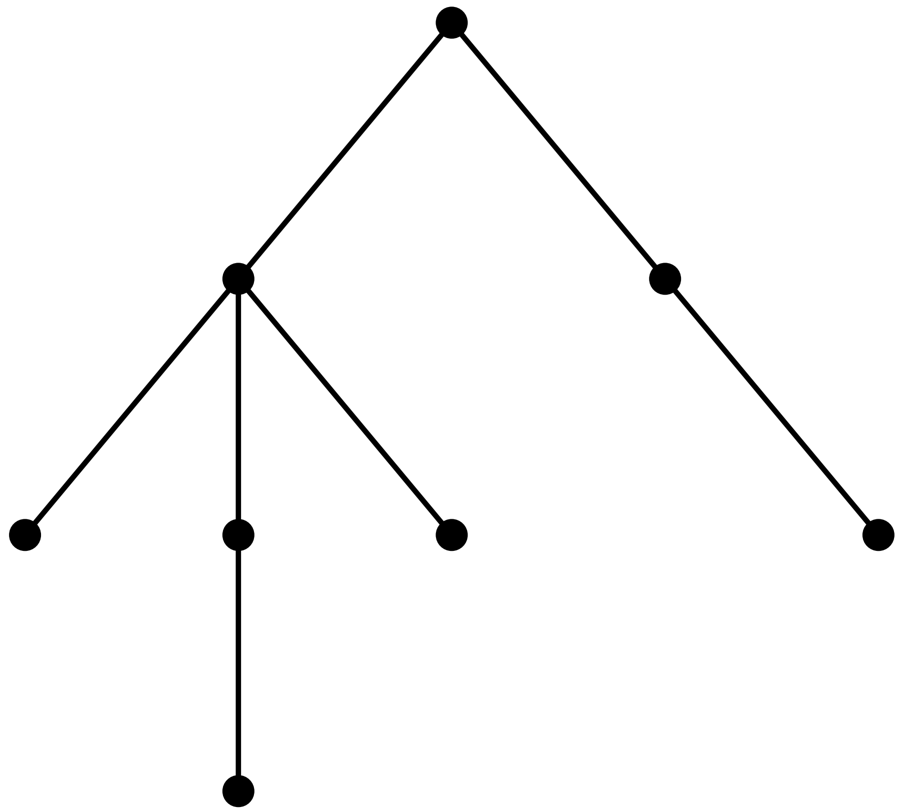
Iteratively add a new vertex connected to an existing vertex.



Introduction

How to build a tree?

Iteratively add a new vertex connected to an existing vertex.



Introduction

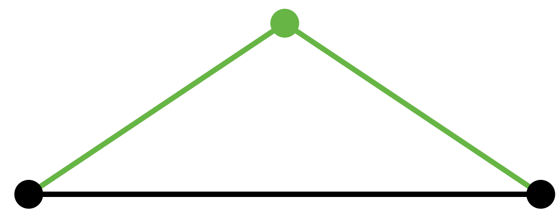
Iteratively add a new vertex connected to the vertices of an existing **edge**.

Introduction

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

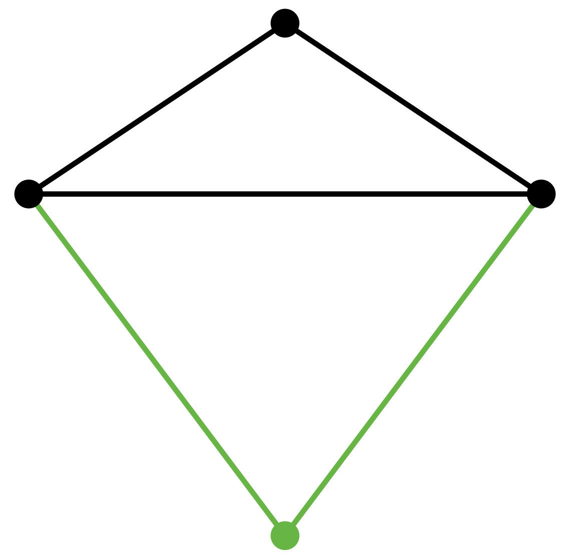
Introduction

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



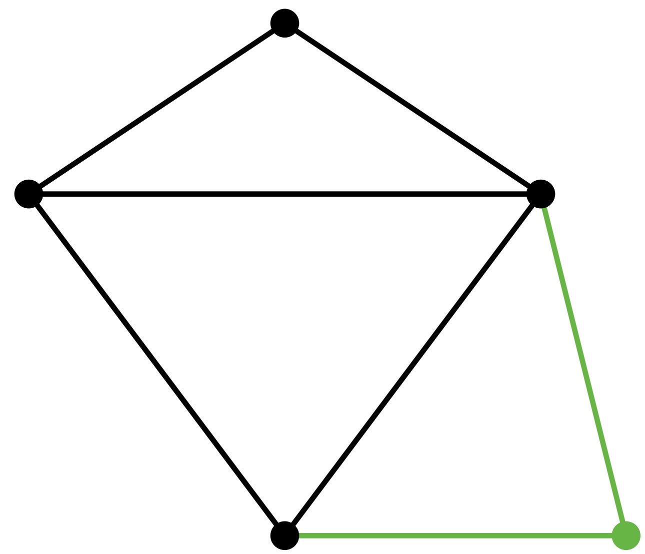
Introduction

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



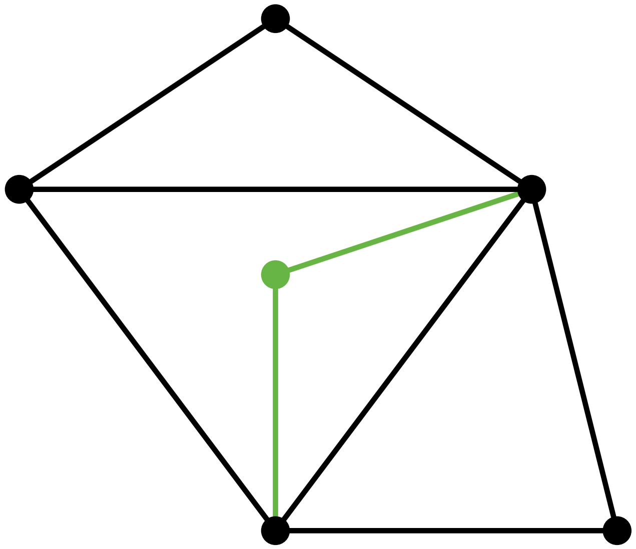
Introduction

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



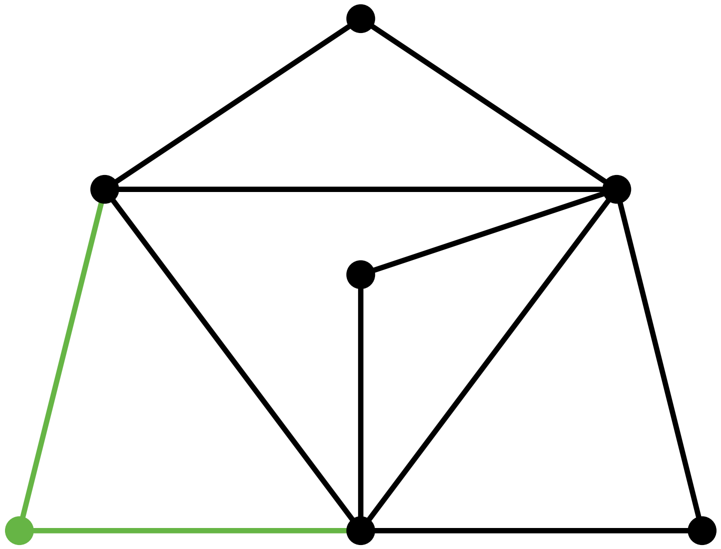
Introduction

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



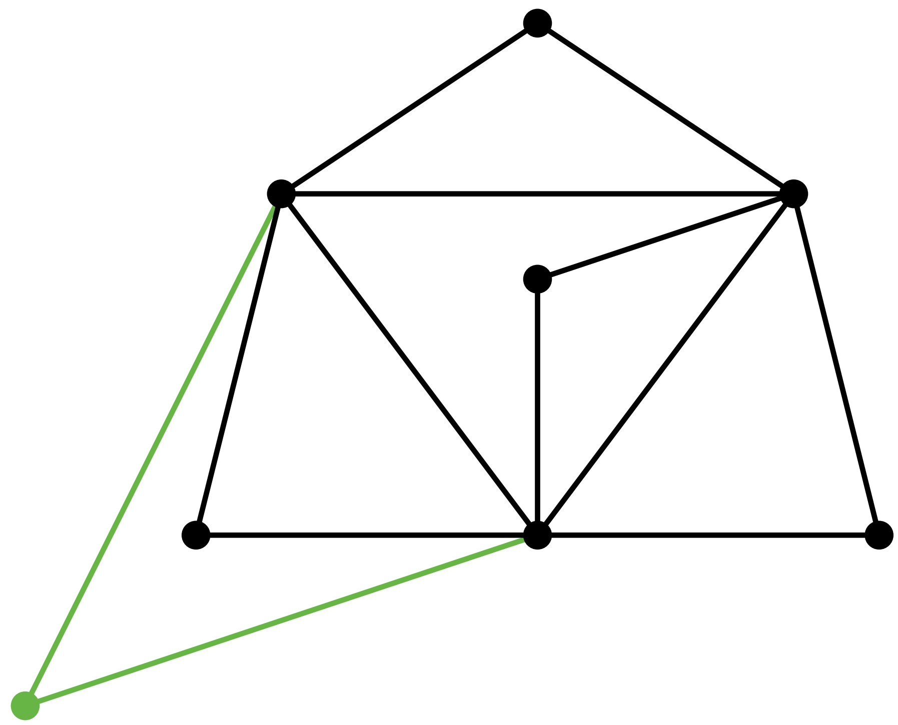
Introduction

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



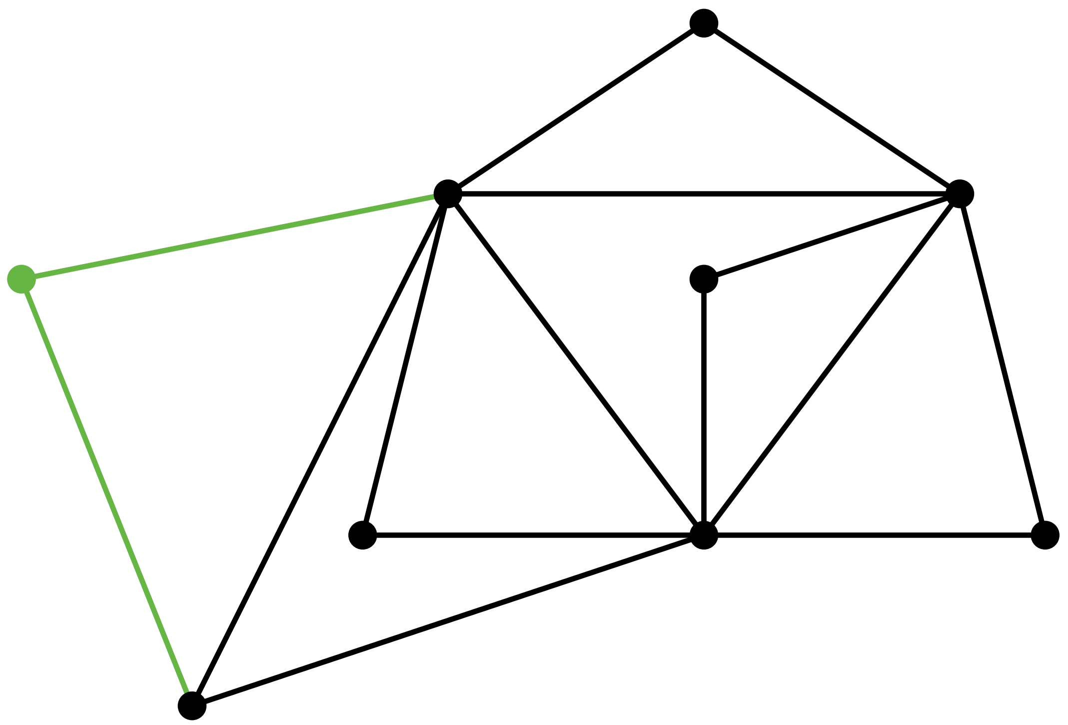
Introduction

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



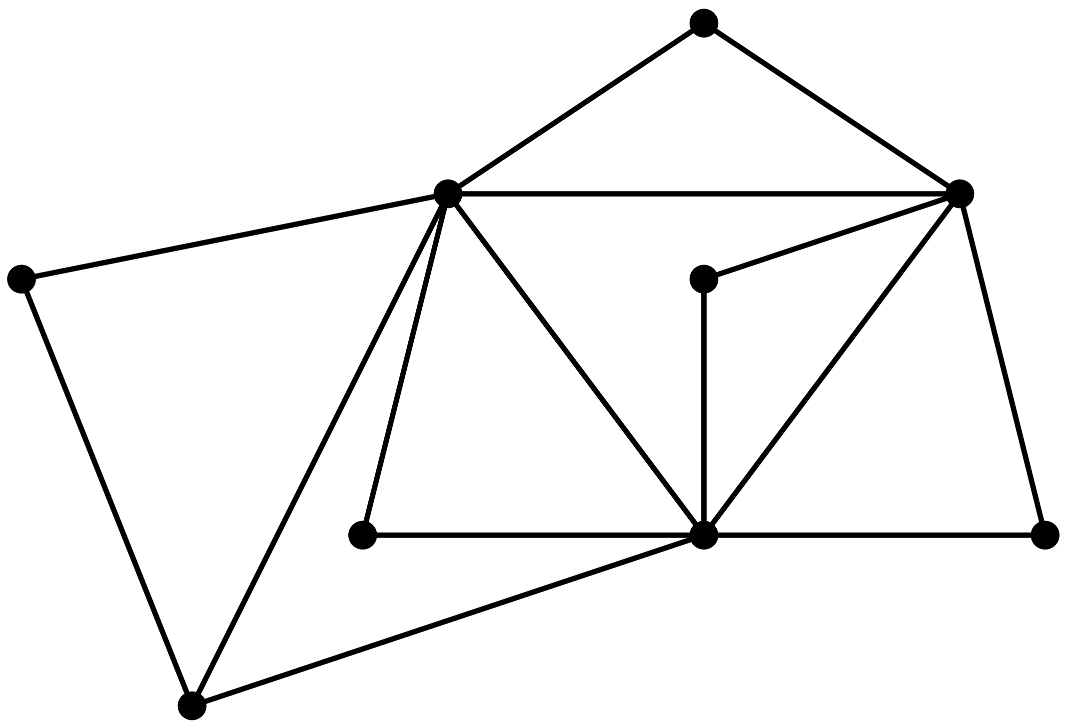
Introduction

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



Introduction

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



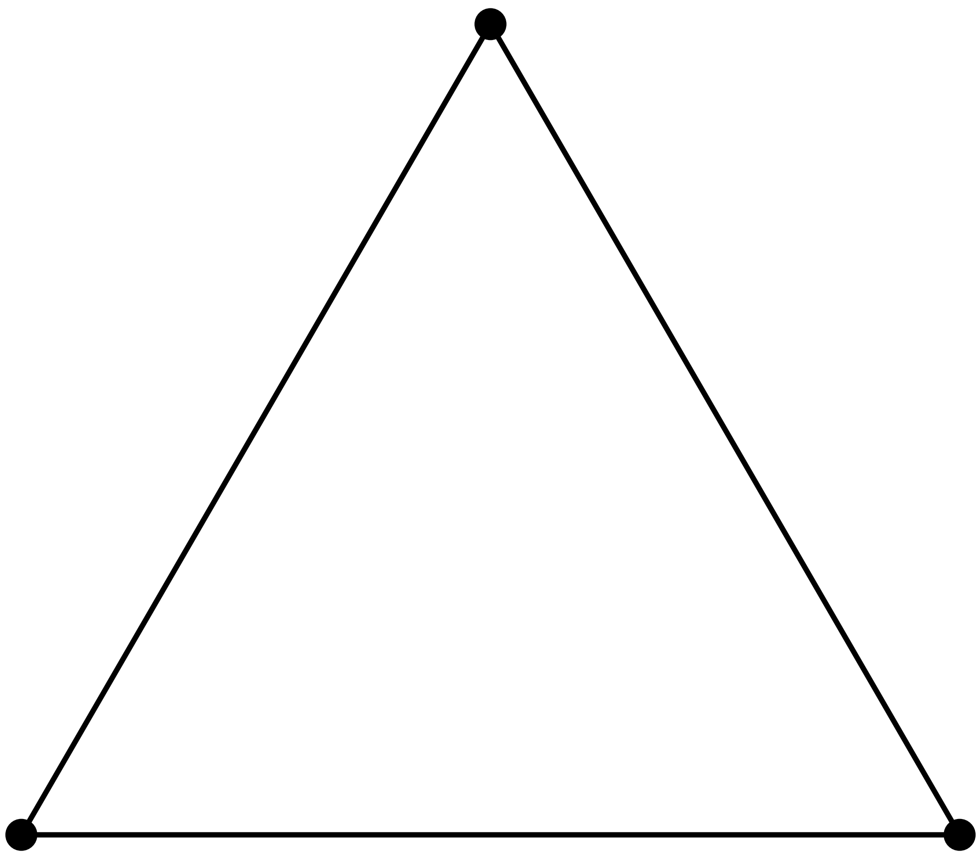
2-trees

Introduction

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

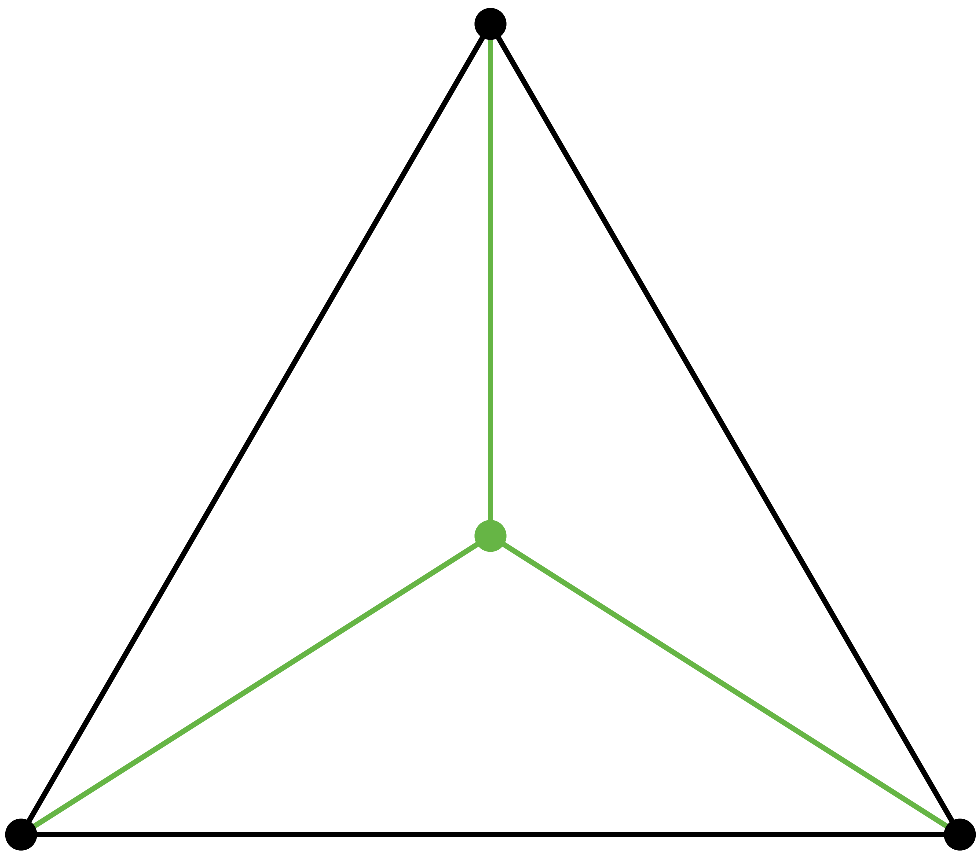
Introduction

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



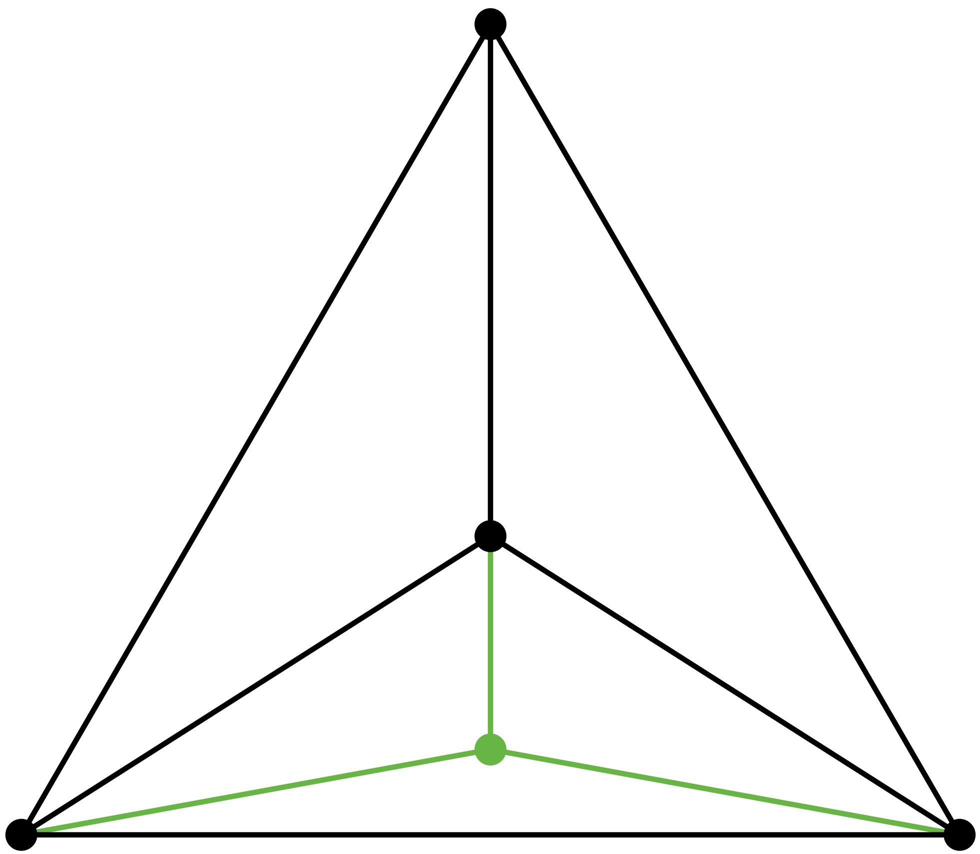
Introduction

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



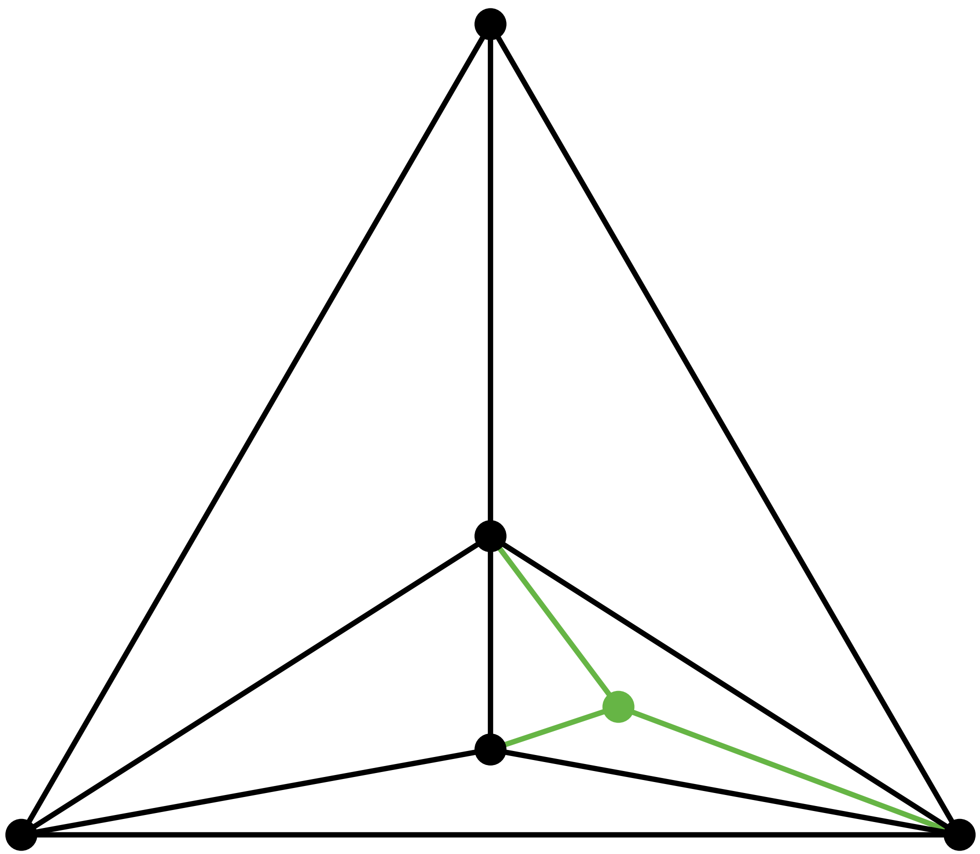
Introduction

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



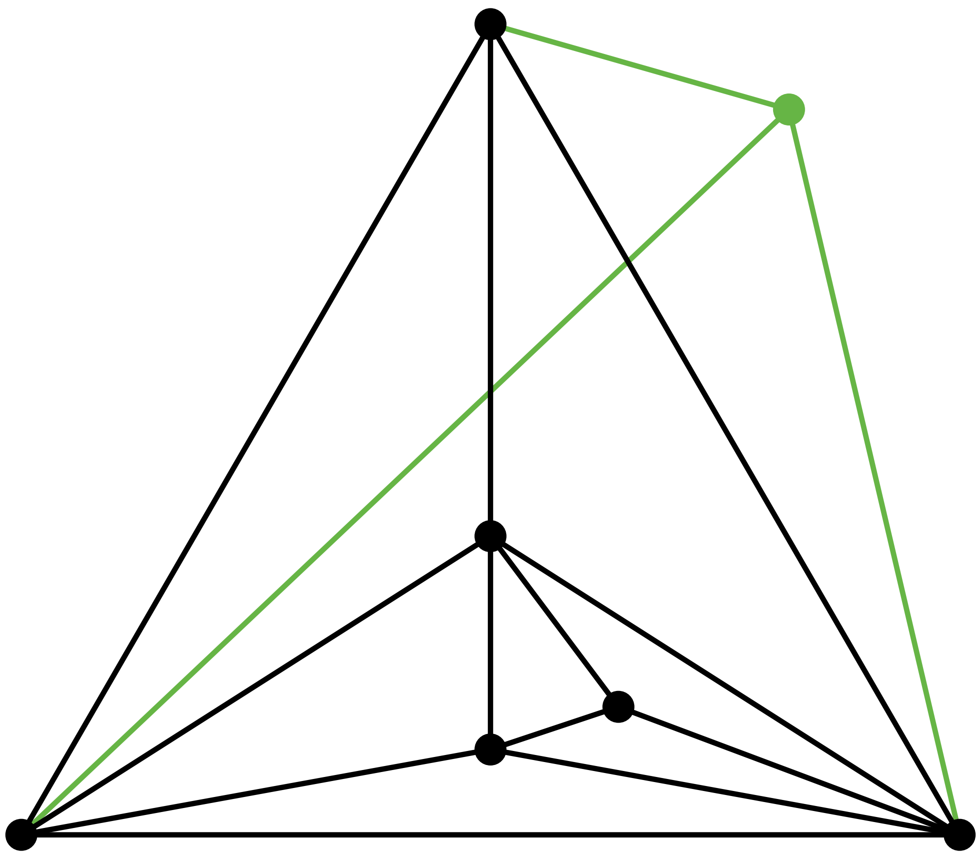
Introduction

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



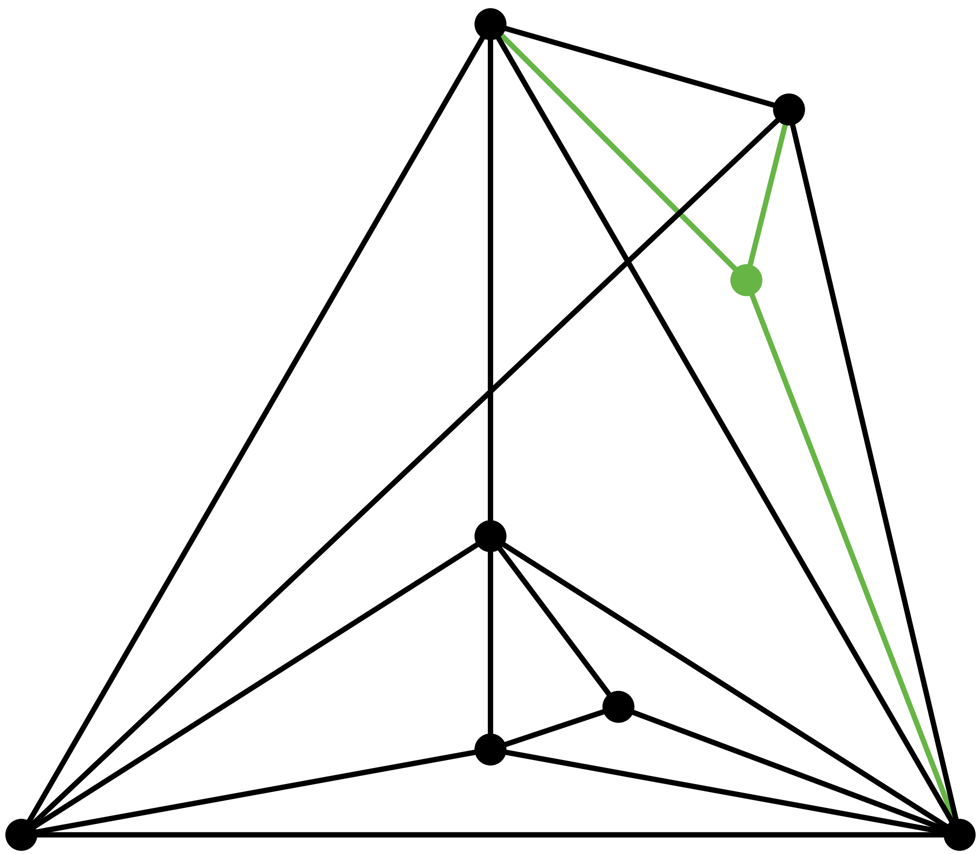
Introduction

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



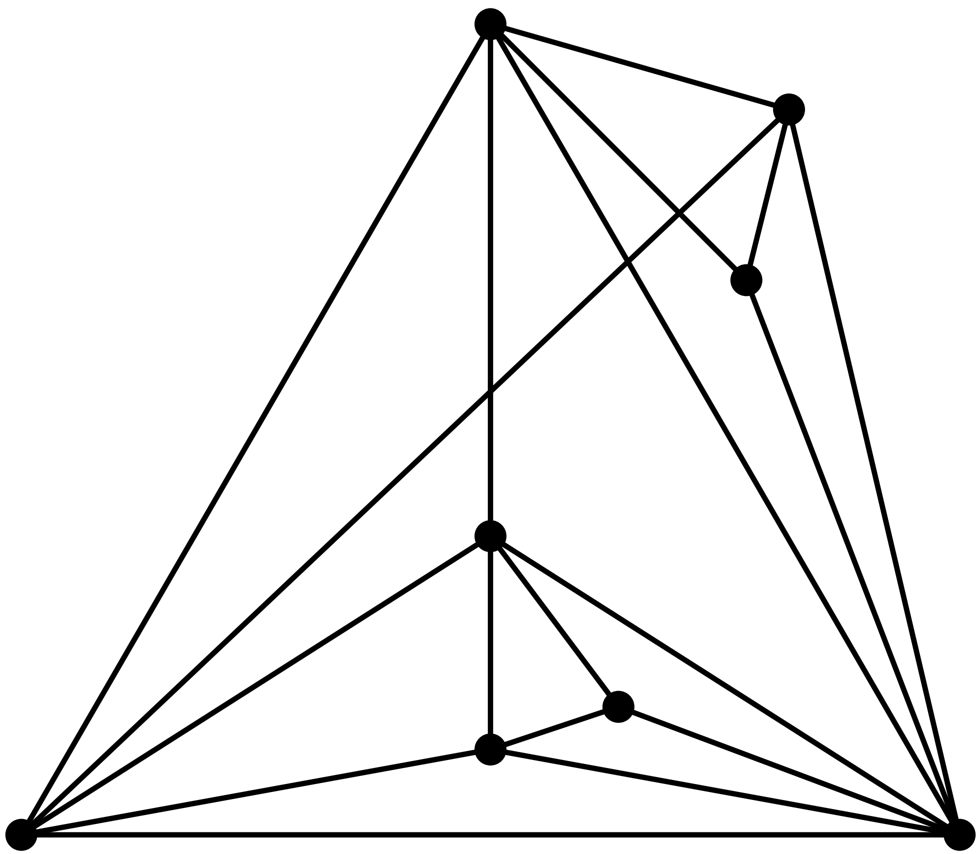
Introduction

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



Introduction

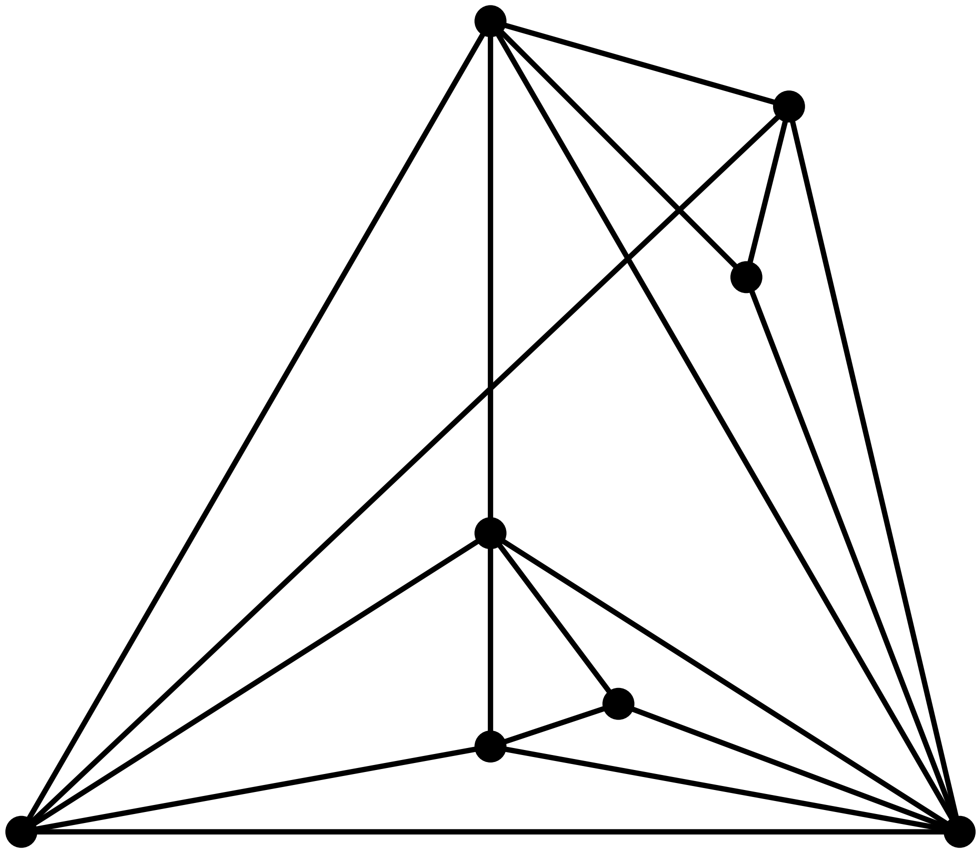
Iteratively add a new vertex connected to the vertices of an existing **triangle**.



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

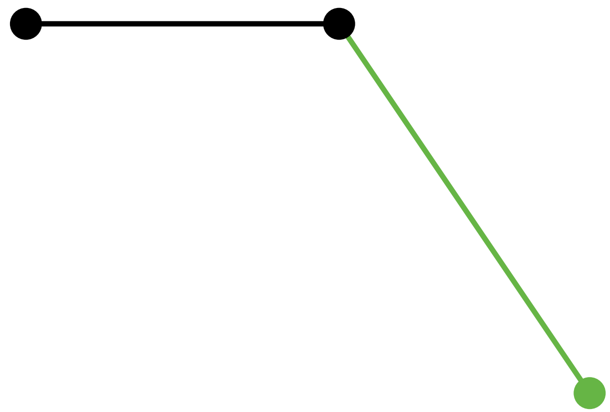
Iteratively add a new vertex connected to the vertices of an existing **clique** (complete subgraph).

Introduction

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

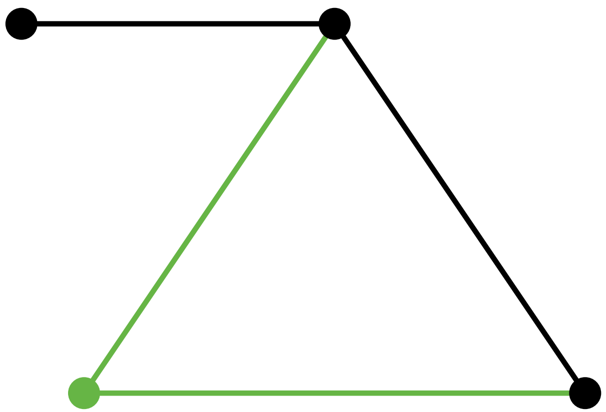
Introduction

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



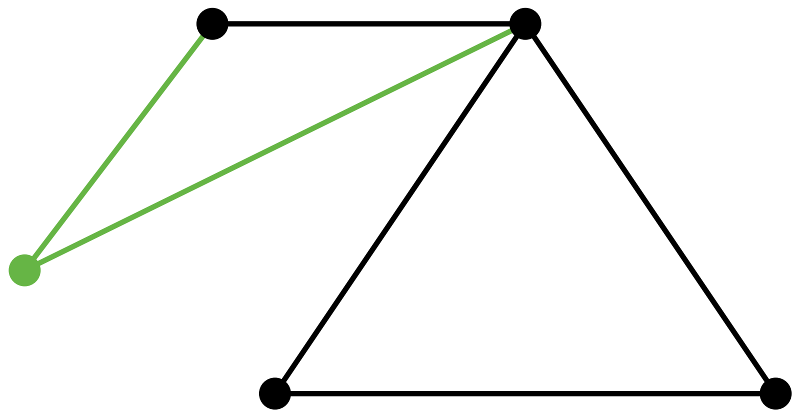
Introduction

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



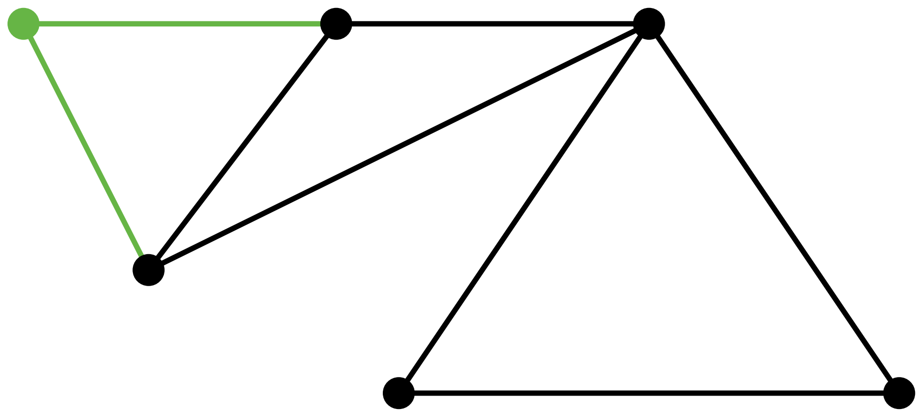
Introduction

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



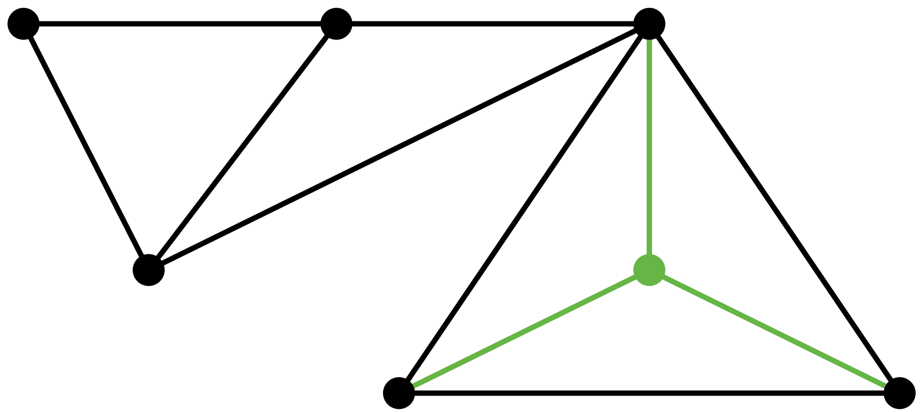
Introduction

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



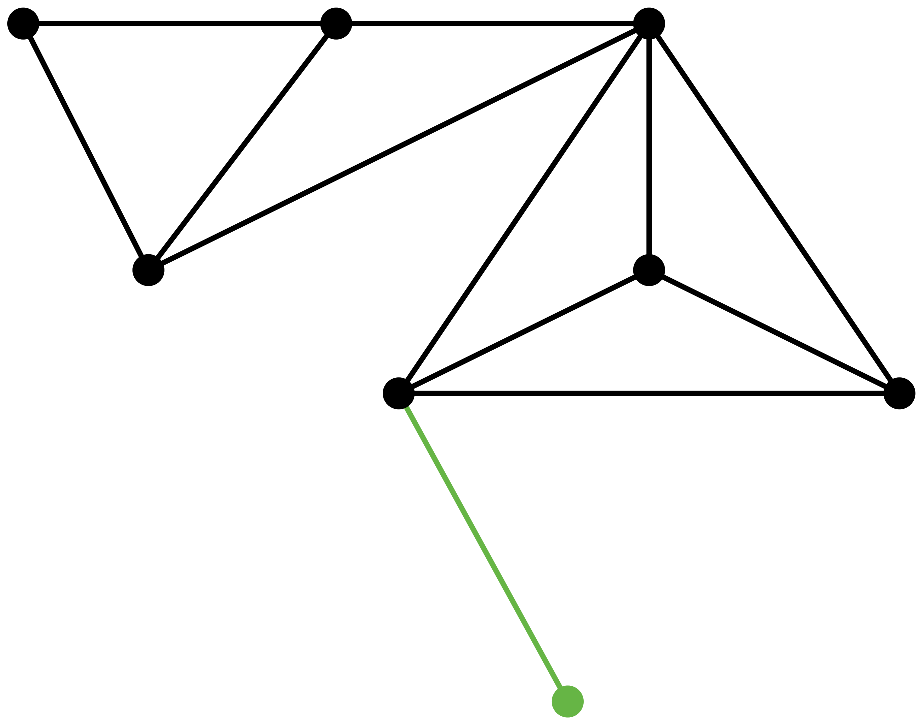
Introduction

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



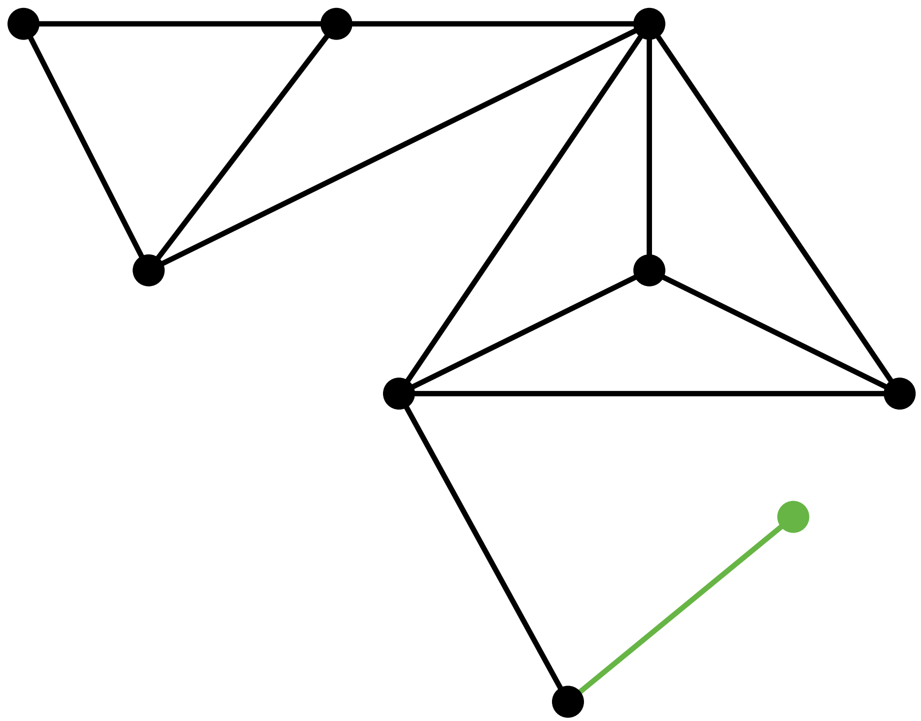
Introduction

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



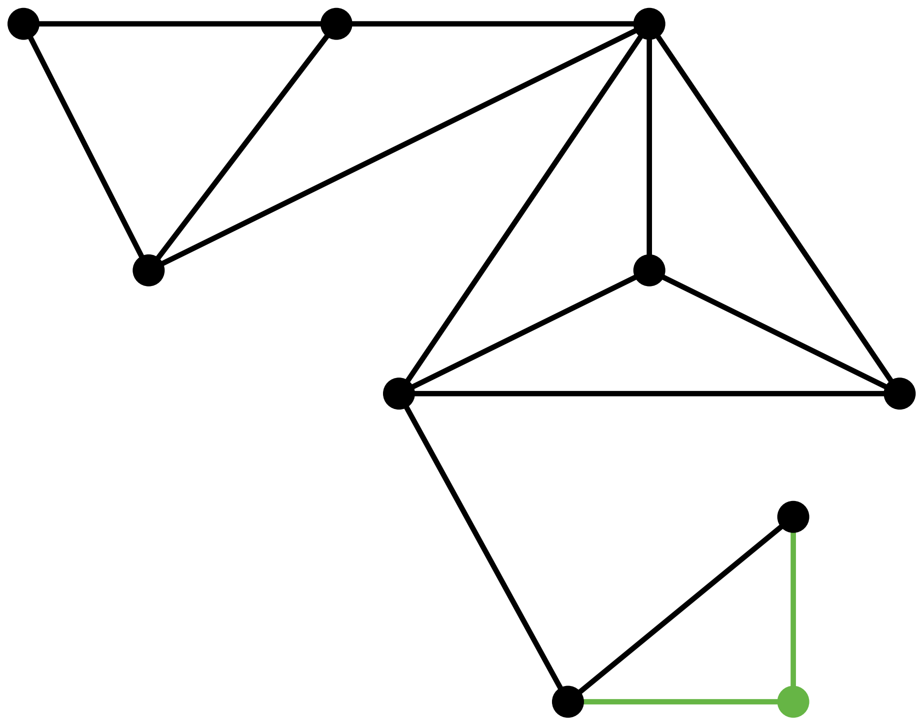
Introduction

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



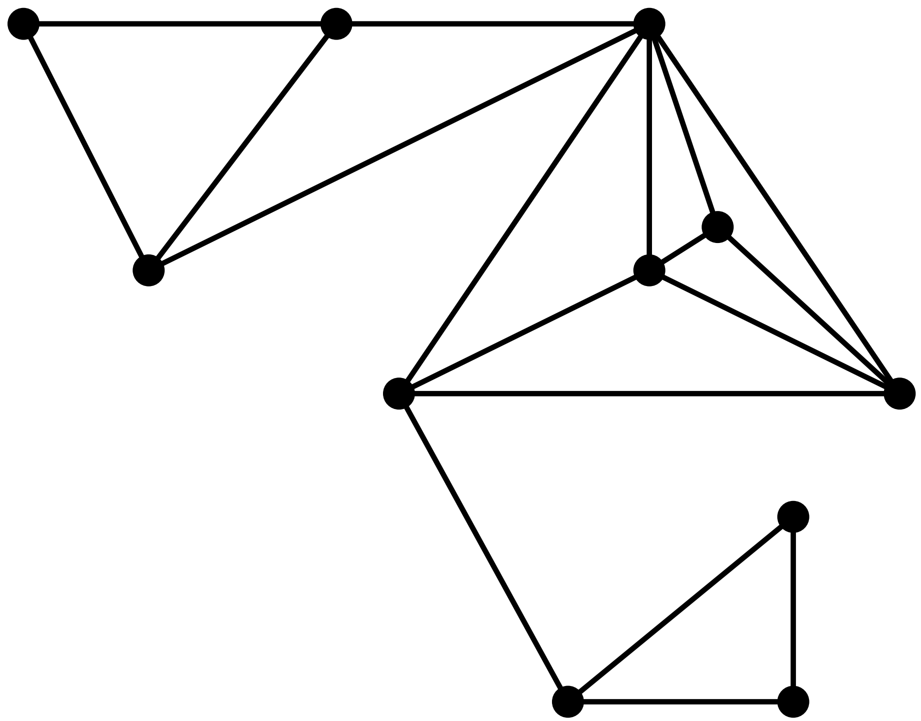
Introduction

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



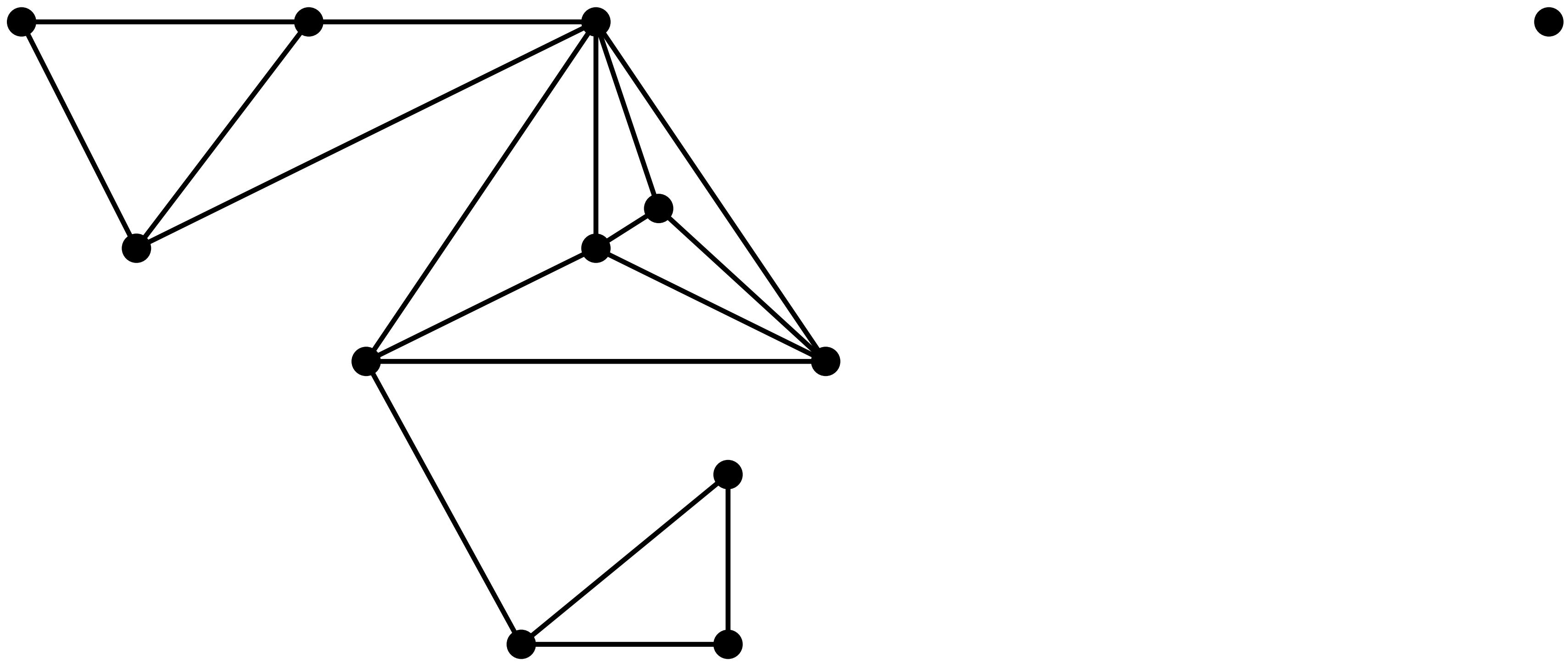
Introduction

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



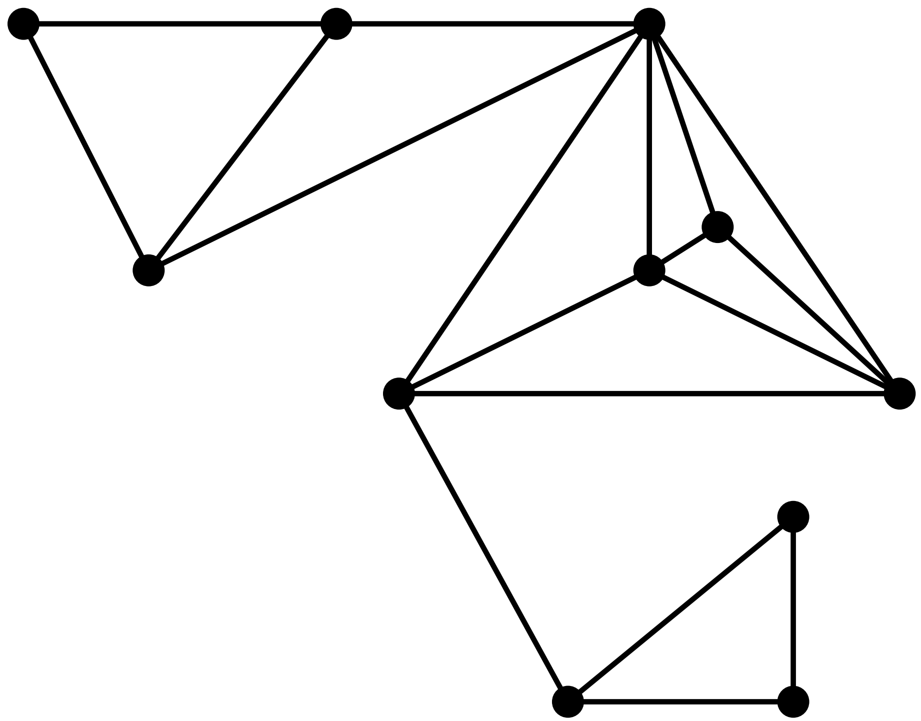
Introduction

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



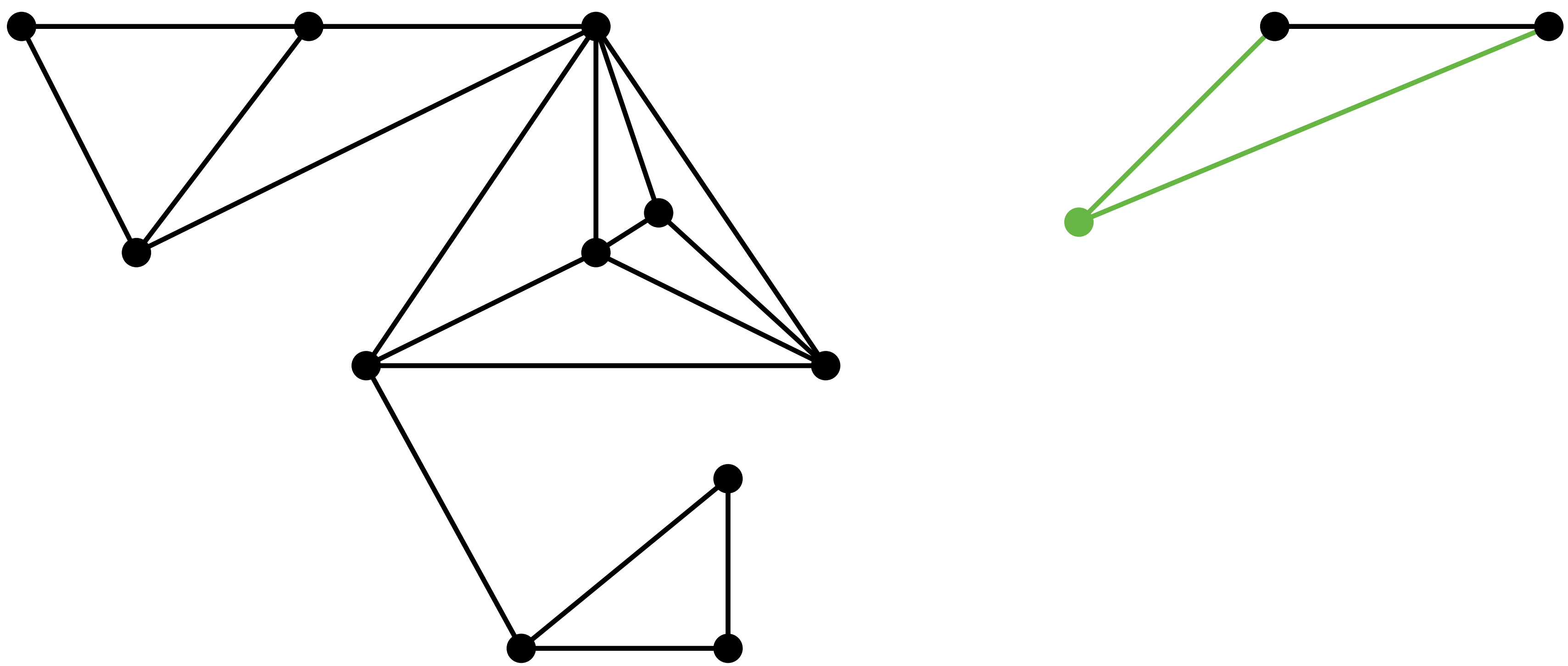
Introduction

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



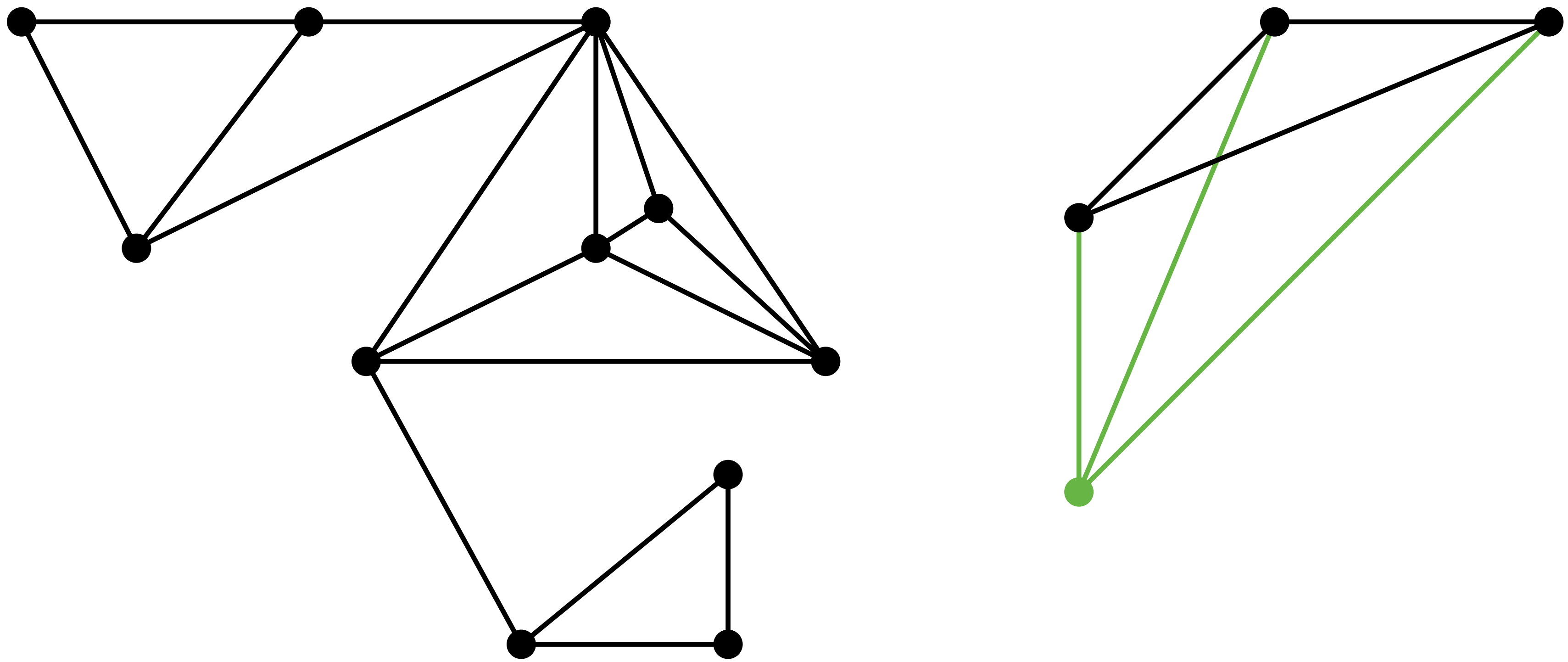
Introduction

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



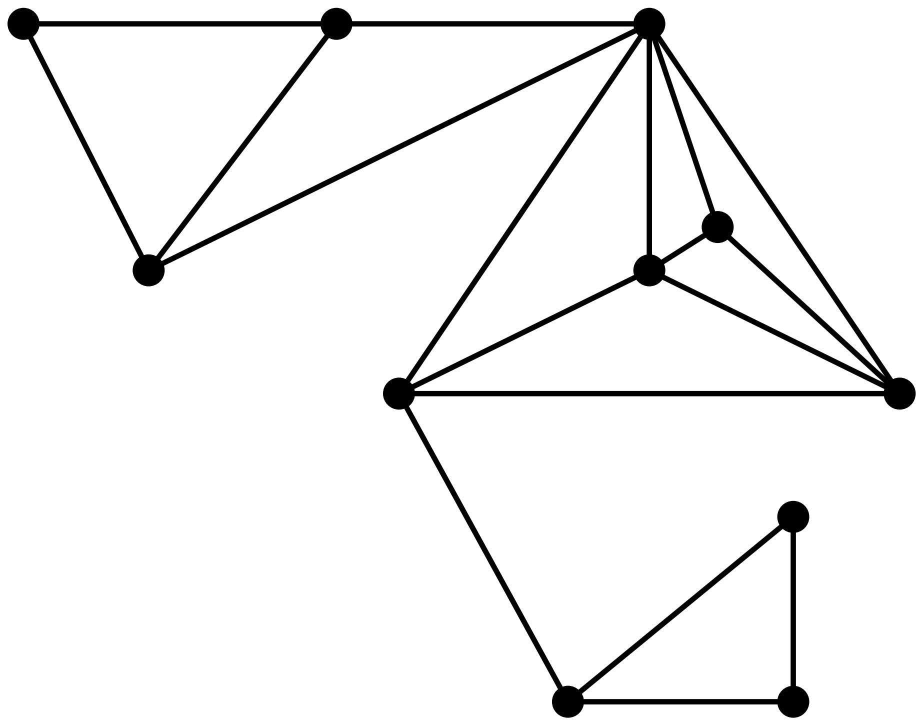
Introduction

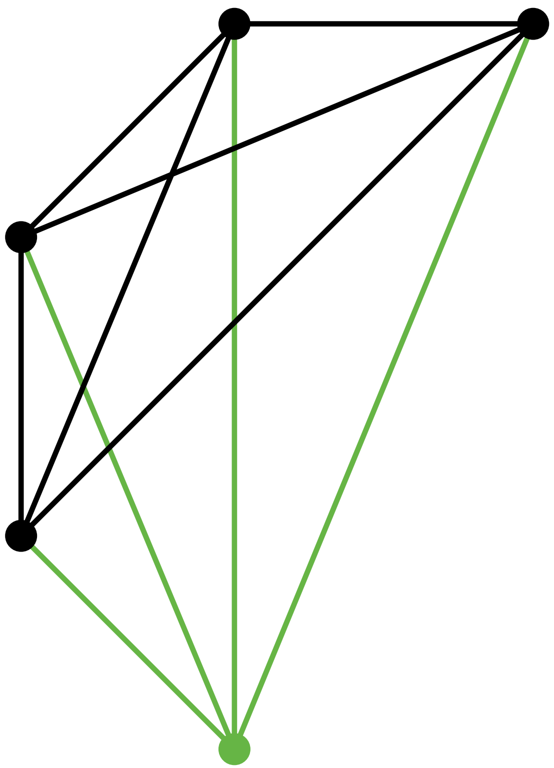
Iteratively add a new vertex connected to the vertices of an existing **clique** (complete subgraph).



Introduction

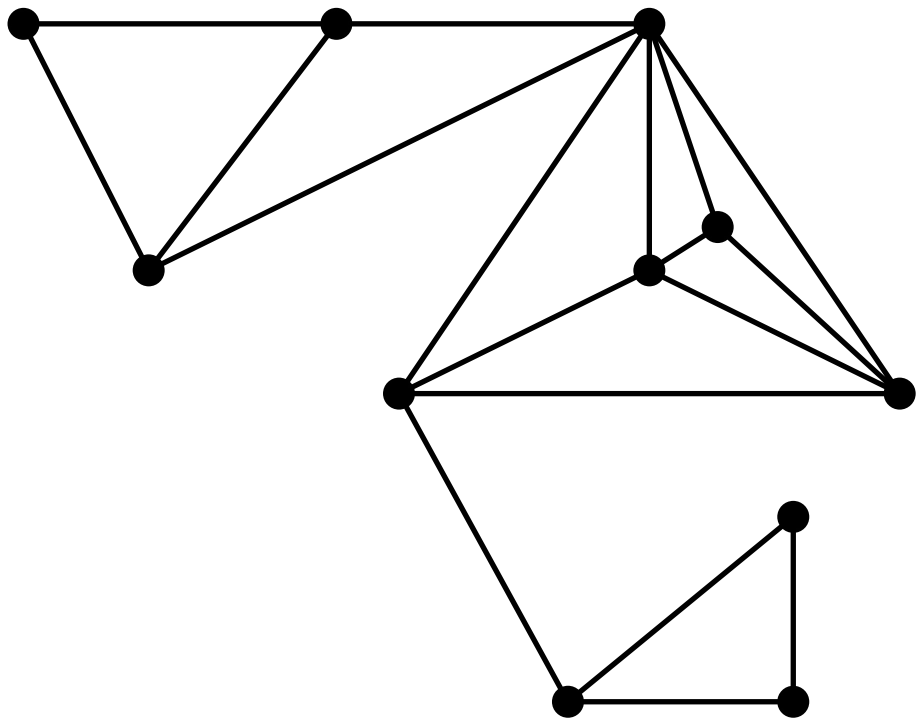
Iteratively add a new vertex connected to the vertices of an existing **clique** (complete subgraph).

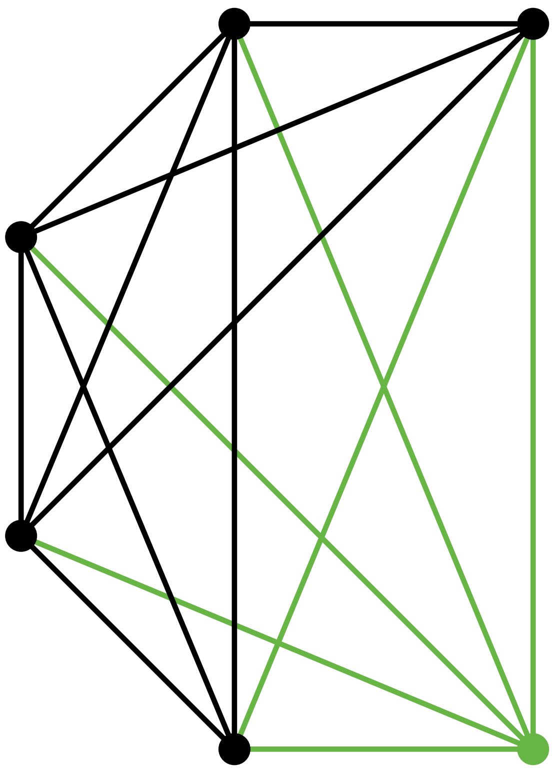




Introduction

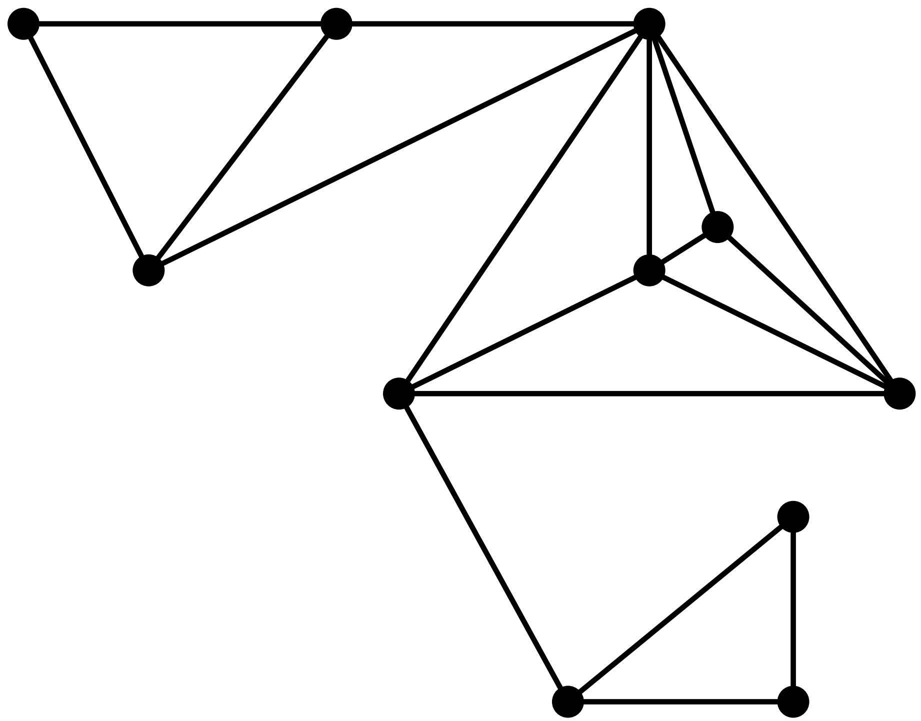
Iteratively add a new vertex connected to the vertices of an existing **clique** (complete subgraph).

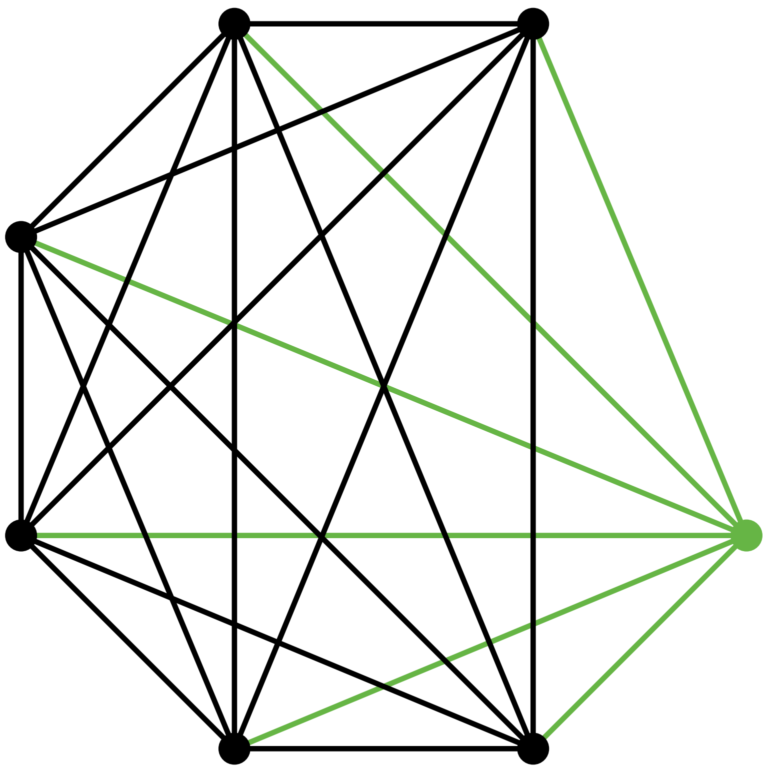




Introduction

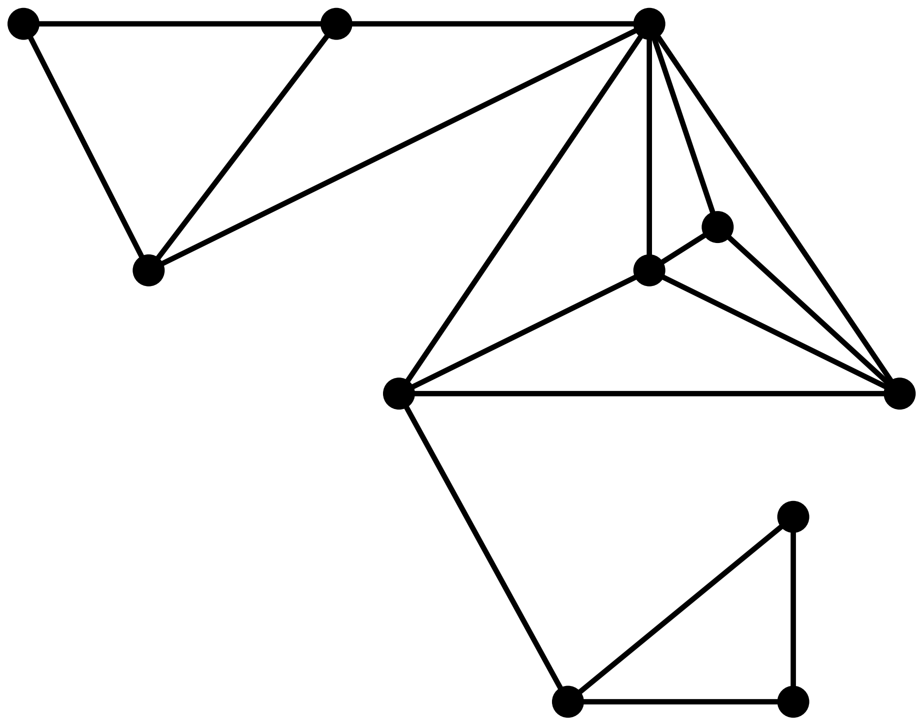
Iteratively add a new vertex connected to the vertices of an existing **clique** (complete subgraph).

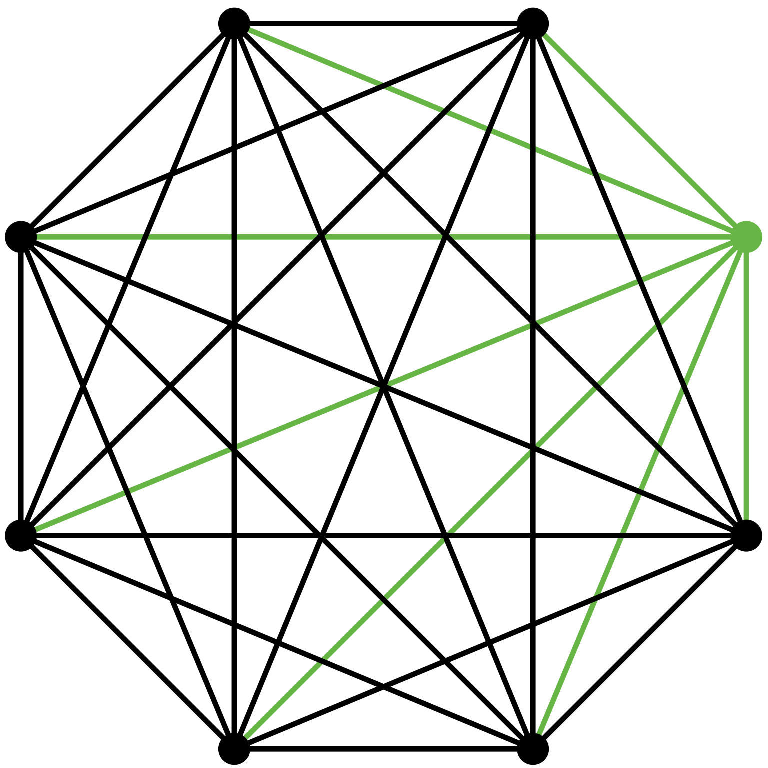




Introduction

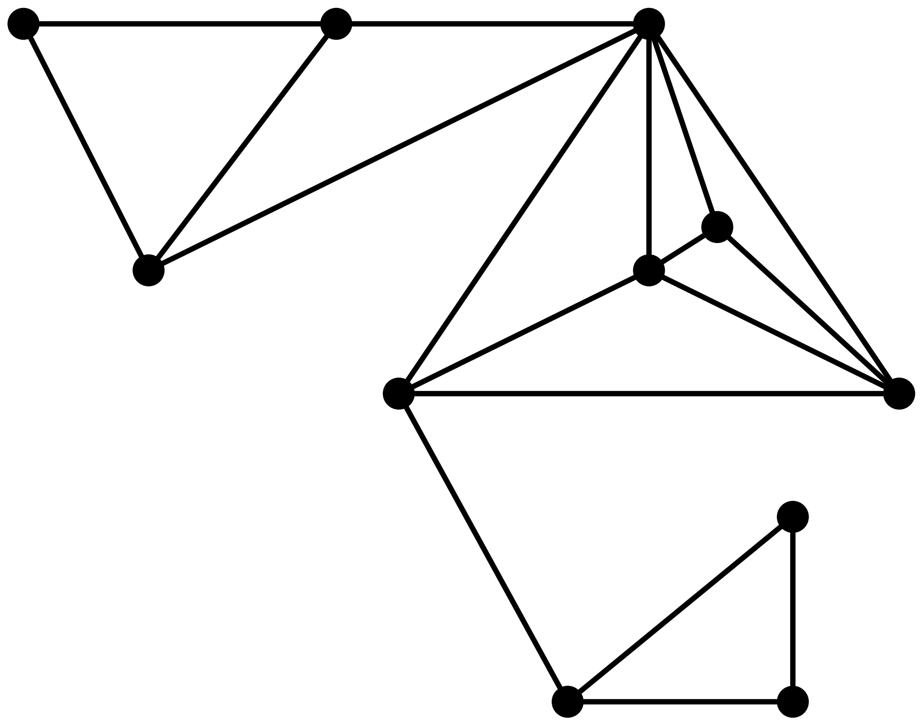
Iteratively add a new vertex connected to the vertices of an existing **clique** (complete subgraph).

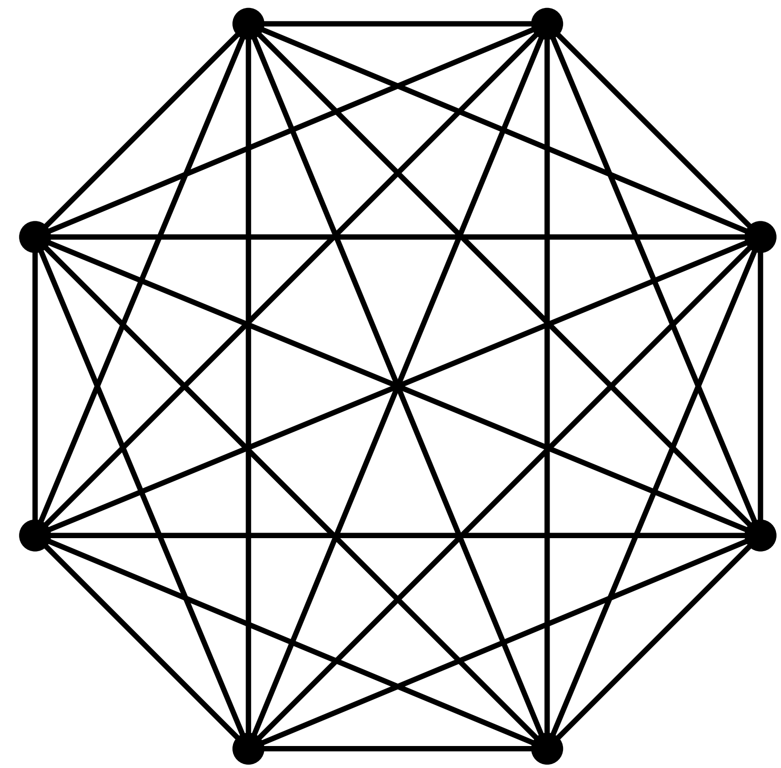




Introduction

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

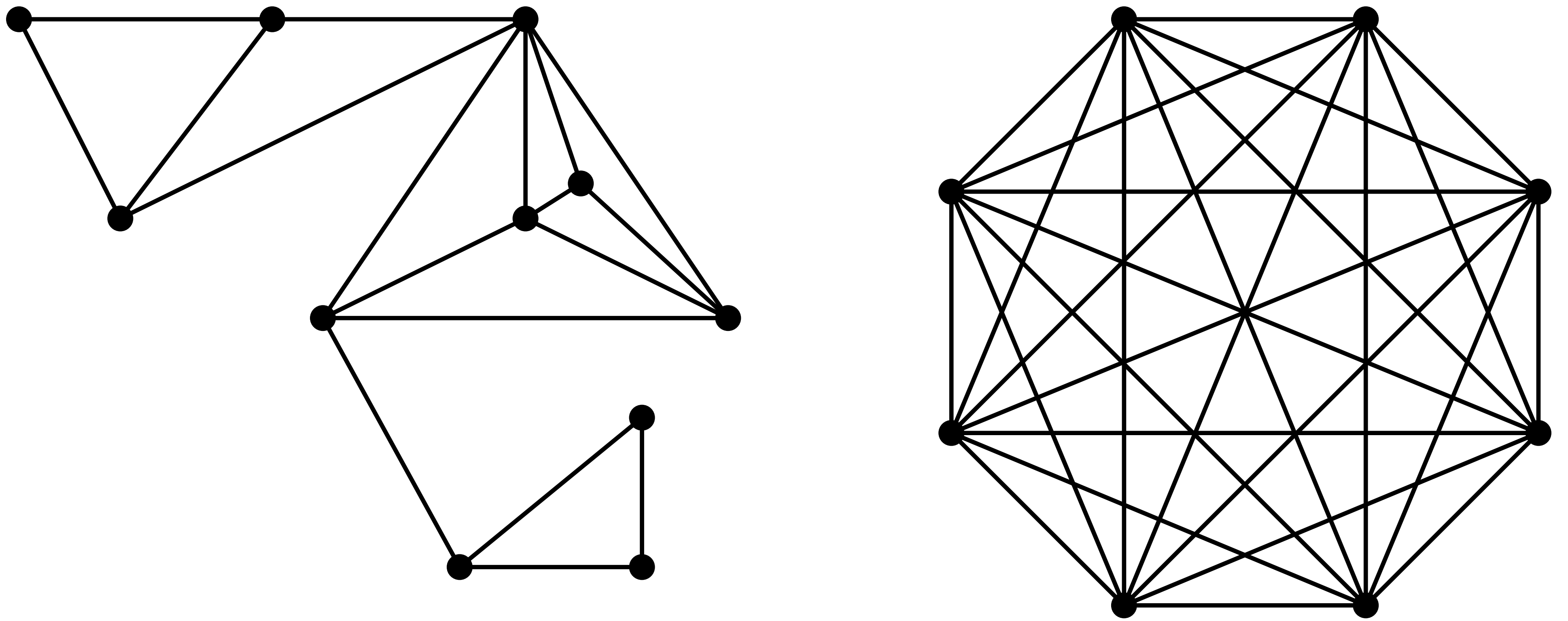




Chordal graphs

Introduction

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



Chordal graphs

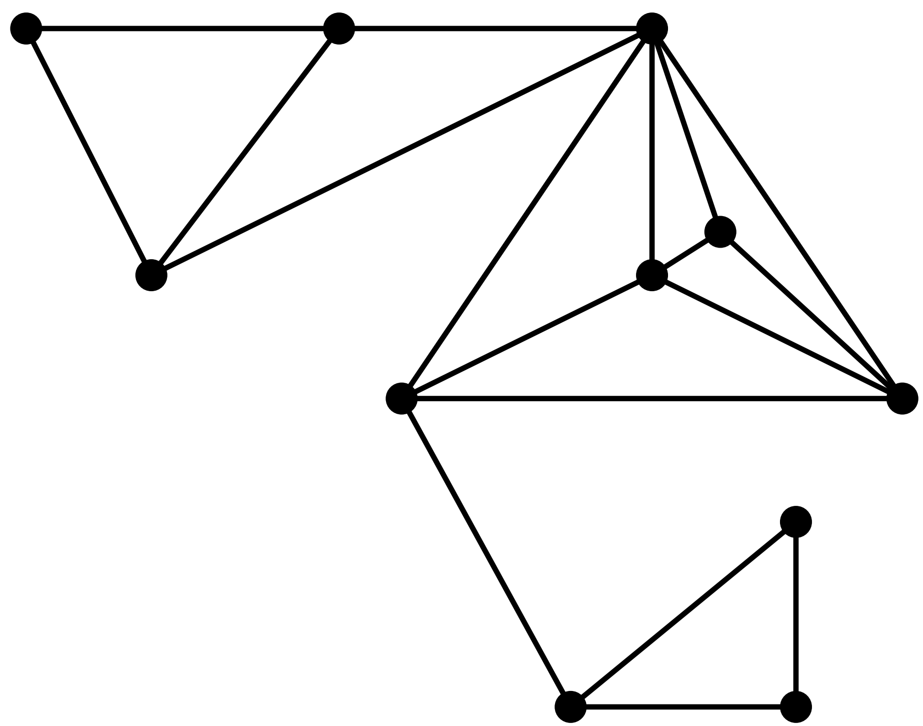
Definition. A graph is **chordal** if it has no induced cycle of lengh greater than 3.

Introduction

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

Introduction

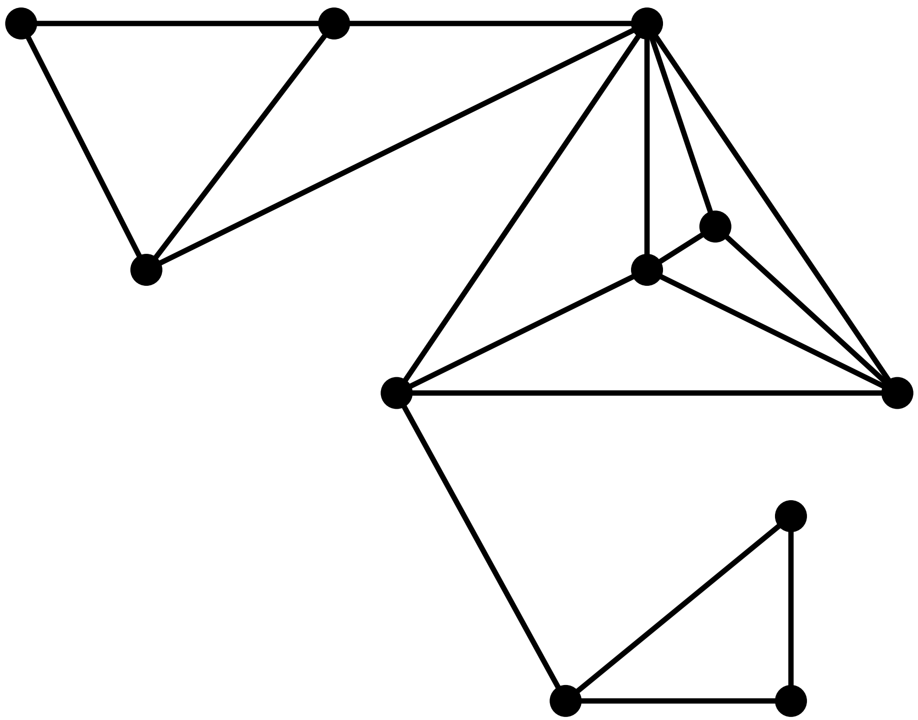
Iteratively add a new vertex connected to the vertices of an existing **clique of size at most t**.



$t = 3$

Introduction

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

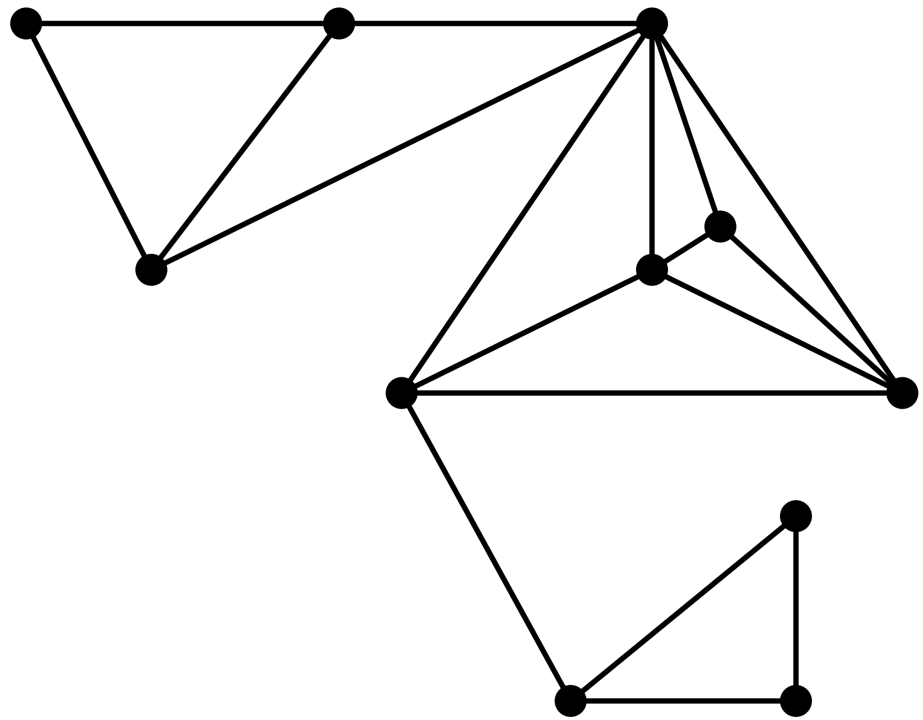


$t = 3$

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.

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

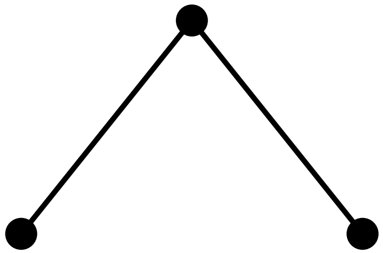
A graph with n vertices is **labelled** if each vertex carries a different label in $\{1, 2, \ldots, n\}$.

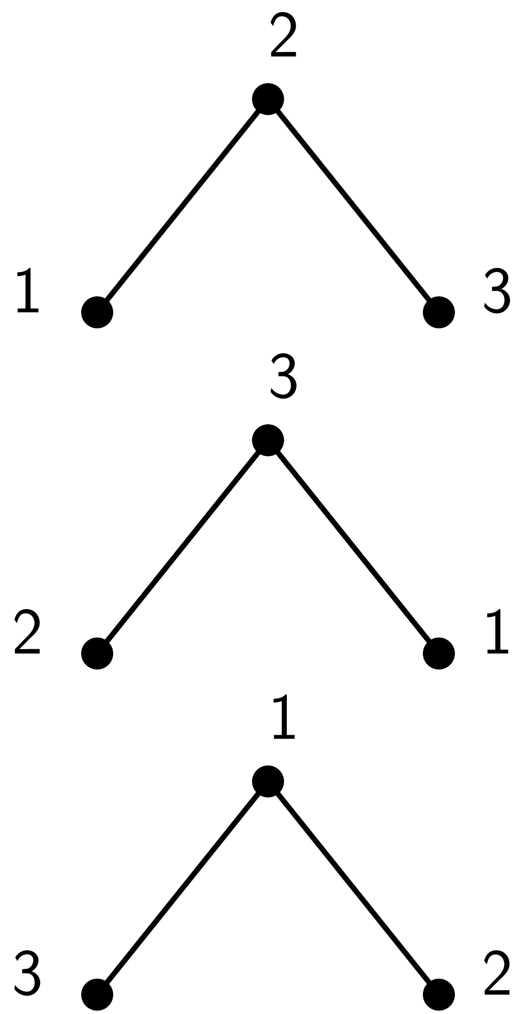
In an **unlabelled** graph, the vertices are undistinguishable.

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.





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.

Labelled trees

Let $\mathcal{T}$ be the class of labelled trees.

Labelled trees

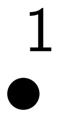
Let $\mathcal{T}$ be the class of labelled trees.

$$T(x) = \frac{1}{1!}x + \frac{1}{2!}x^2 + \frac{3}{3!}x^3 + \frac{16}{4!}x^4 \cdots .$$

Labelled trees

Let $\mathcal{T}$ be the class of labelled trees.

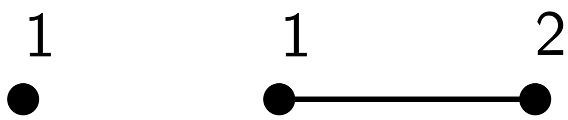
$$T(x) = \frac{1}{1!}x + \frac{1}{2!}x^2 + \frac{3}{3!}x^3 + \frac{16}{4!}x^4 \cdots .$$



Labelled trees

Let $\mathcal{T}$ be the class of labelled trees.

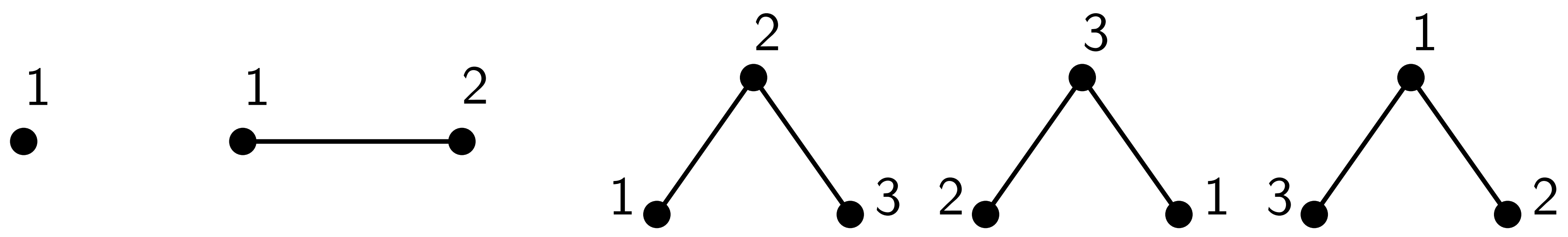
$$T(x) = \frac{1}{1!}x + \frac{1}{2!}x^2 + \frac{3}{3!}x^3 + \frac{16}{4!}x^4 \cdots .$$



Labelled trees

Let $\mathcal{T}$ be the class of labelled trees.

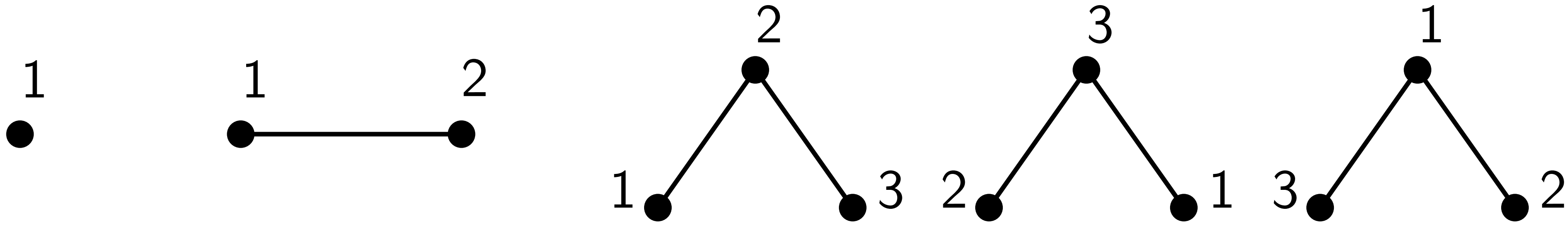
$$T(x) = \frac{1}{1!}x + \frac{1}{2!}x^2 + \frac{3}{3!}x^3 + \frac{16}{4!}x^4 \cdots .$$



Labelled trees

Let $\mathcal{T}$ be the class of labelled trees.

$$T(x) = \frac{1}{1!}x + \frac{1}{2!}x^2 + \frac{3}{3!}x^3 + \frac{16}{4!}x^4 \cdots .$$



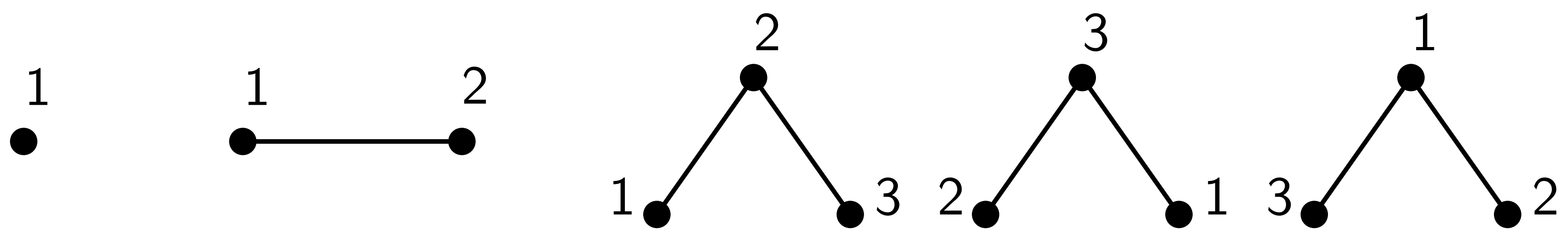
Rooting. Let $\mathcal{T}^\bullet$ be the class of rooted labelled trees.
Since all vertices are distinguishable, there are n ways to root a tree with n vertices. Thus,

$$T^\bullet(x) = \sum_{n\geq 0} n \frac{t_n}{n!} x^n = xT'(x).$$

Labelled trees

Let $\mathcal{T}$ be the class of labelled trees.

$$T(x) = \frac{1}{1!}x + \frac{1}{2!}x^2 + \frac{3}{3!}x^3 + \frac{16}{4!}x^4 \cdots .$$



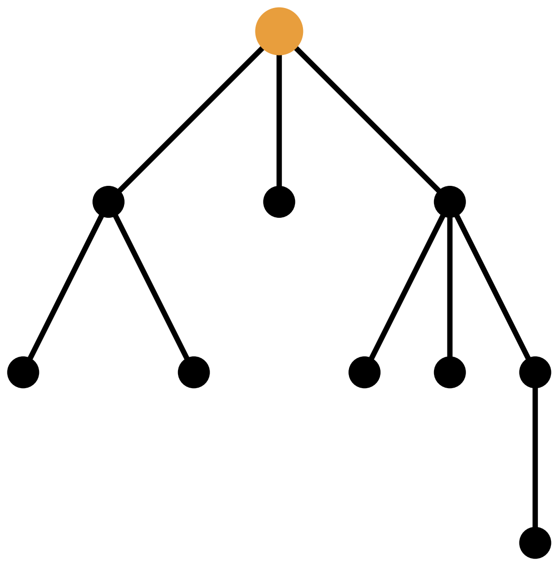
Rooting. Let $\mathcal{T}^\bullet$ be the class of rooted labelled trees.
Since all vertices are distinguishable, there are n ways to root a tree with n vertices. Thus,

$$T^\bullet(x) = \sum_{n\geq 0} n\frac{t_n}{n!}\,x^n = xT'(x).$$

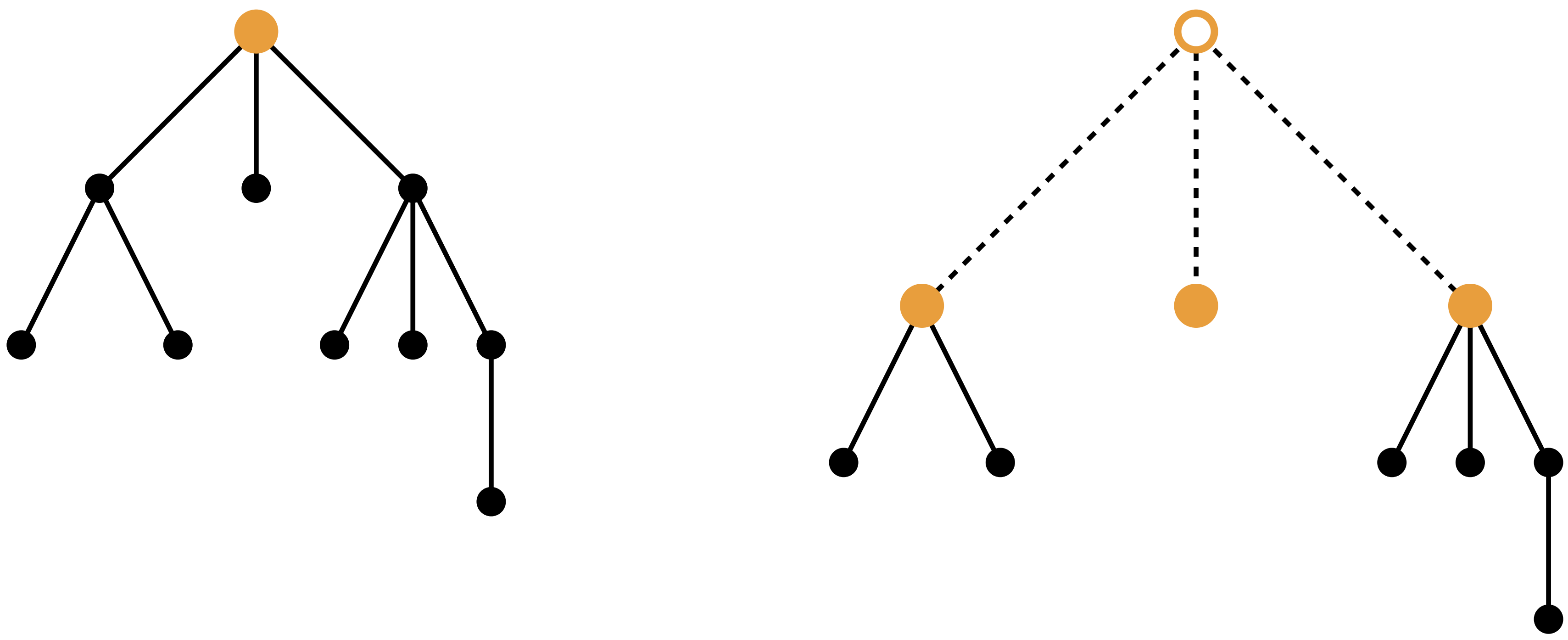
Unrooting. To do the inverse operation, we can simply integrate:

$$T(x) = \int T^\bullet(x)/x\; dx.$$

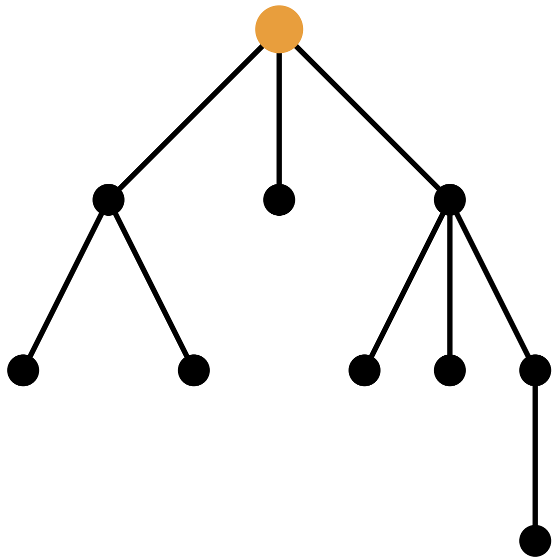
Labelled trees

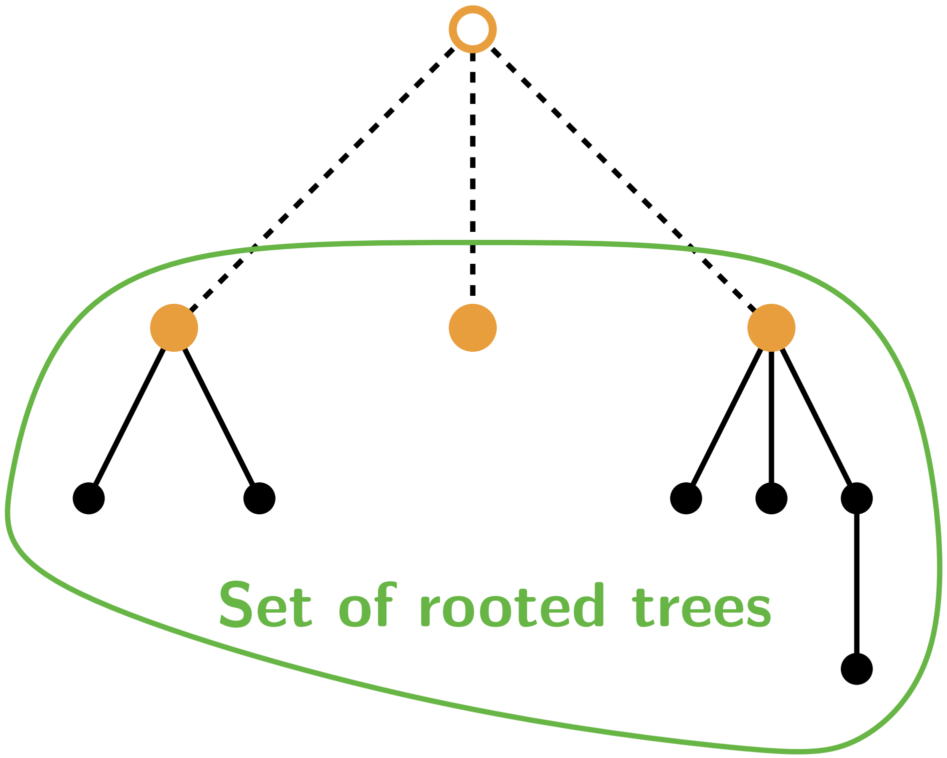


Labelled trees

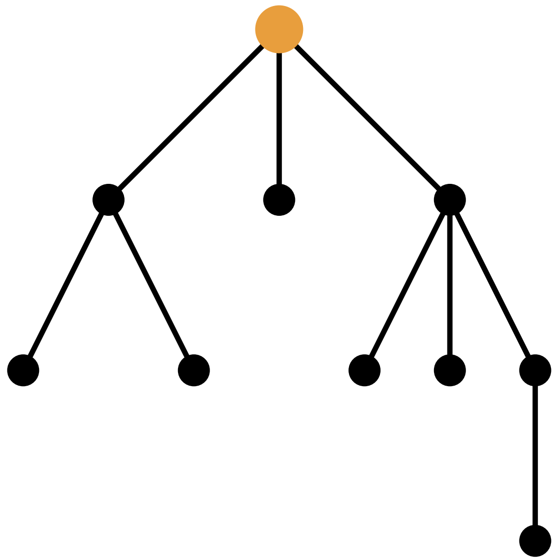


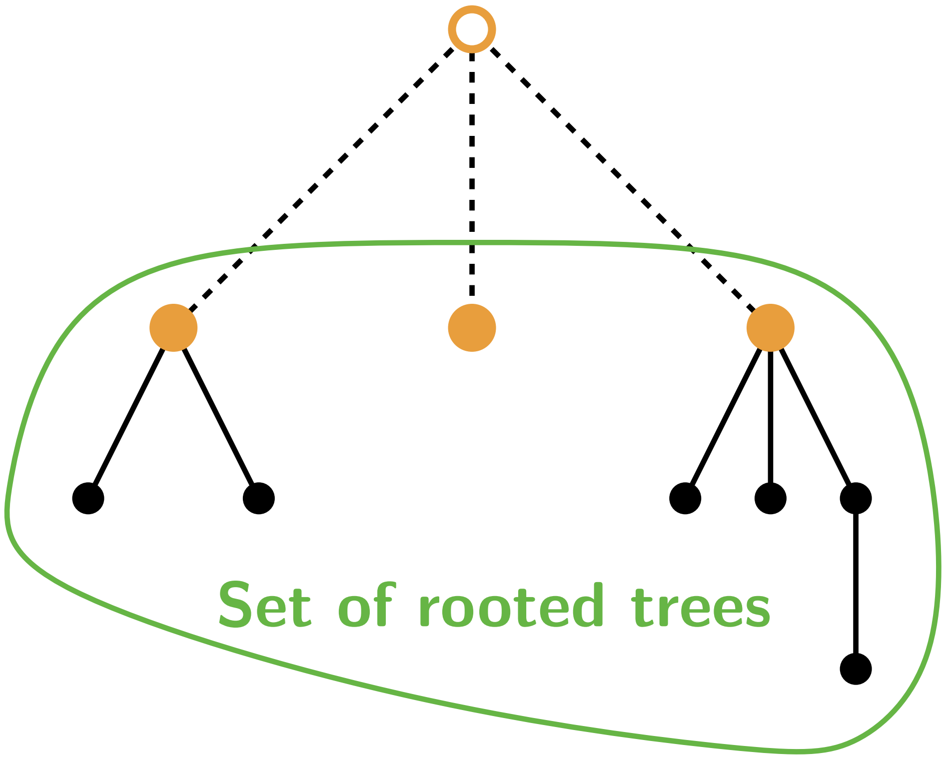
Labelled trees





Labelled trees

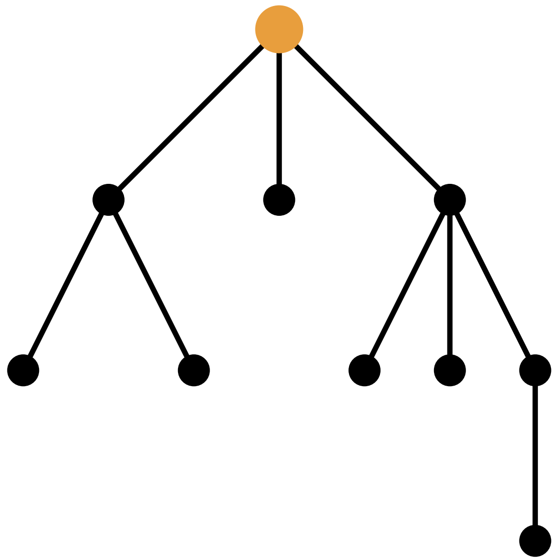


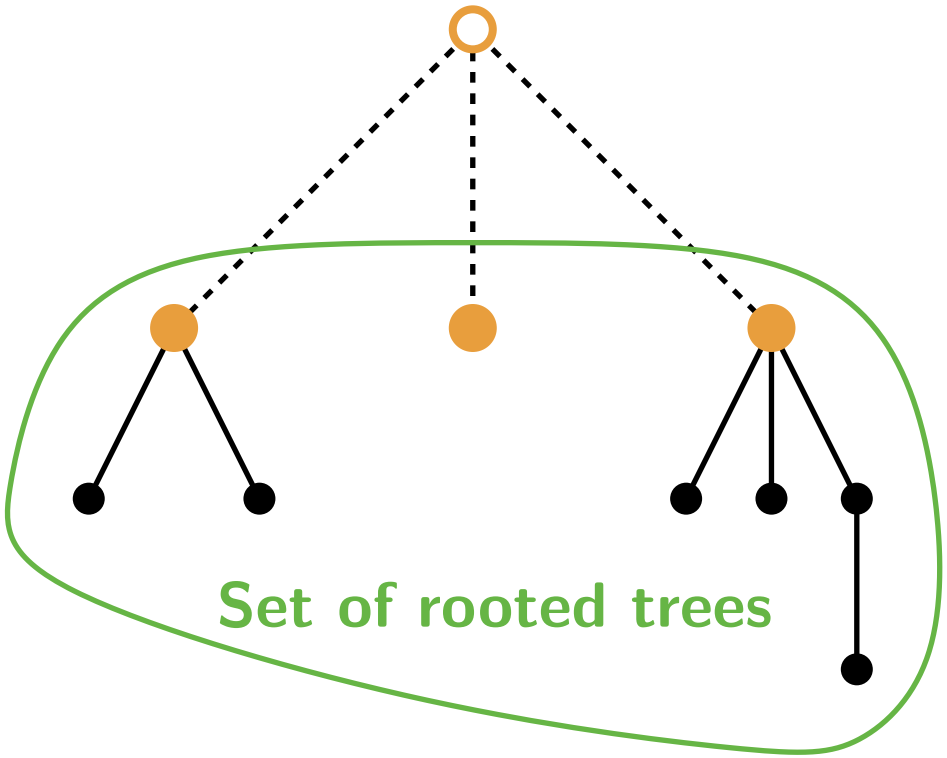


Implicit equation:

$$T^{\bullet}(x) = x \exp\left(T^{\bullet}(x)\right) = x + x^2 + \frac{3x^3}{2} + \cdots . \tag{2}$$

Labelled trees





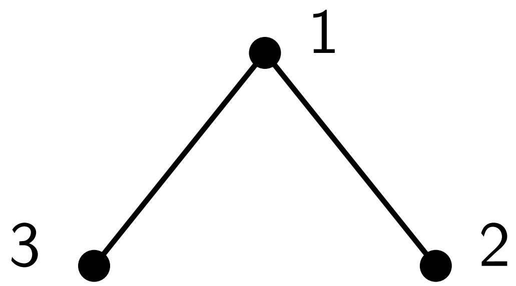
Implicit equation:

$$T^\bullet(x) = x\exp\left(T^\bullet(x)\right) = x + x^2 + \frac{3x^3}{2} + \cdots . \qquad (2)$$

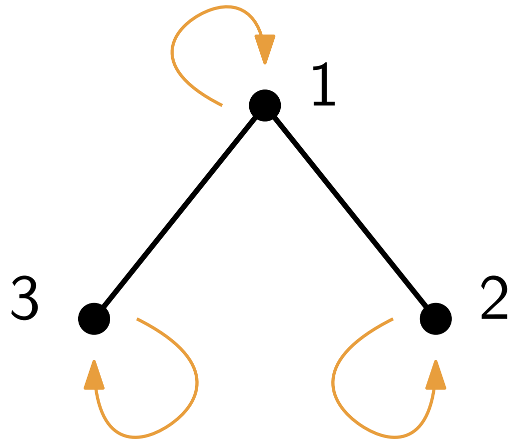
By using the **Lagrange inversion formula** we obtain:

$$|\mathcal{T}_n^\bullet| = n![x^n]T^\bullet(x) = n^{n-1} \implies |\mathcal{T}_n| = |\mathcal{T}_n^\bullet|/n = n^{n-2}.$$

Pólya theory

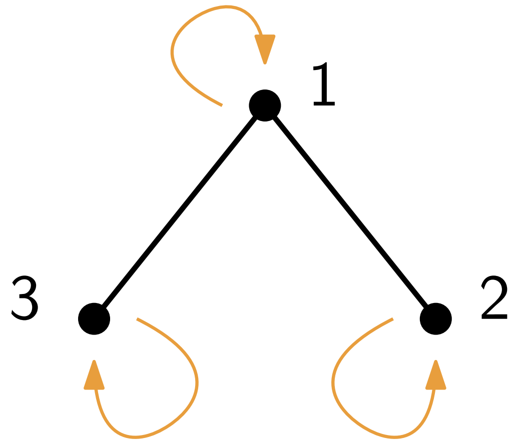


Pólya theory

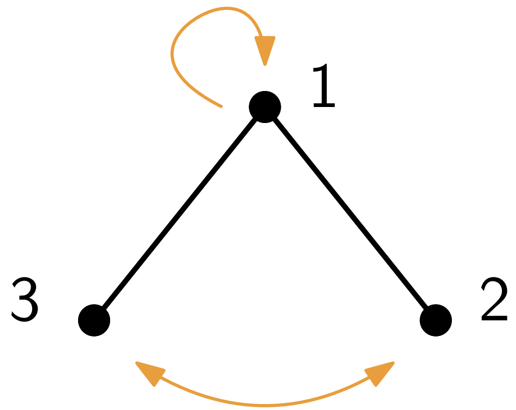


$$(1)(2)(3) \longrightarrow s_1^3$$

Pólya theory

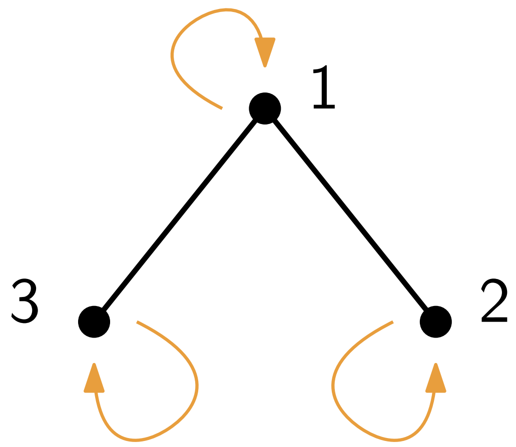


$$(1)(2)(3) \longrightarrow s_1^3$$

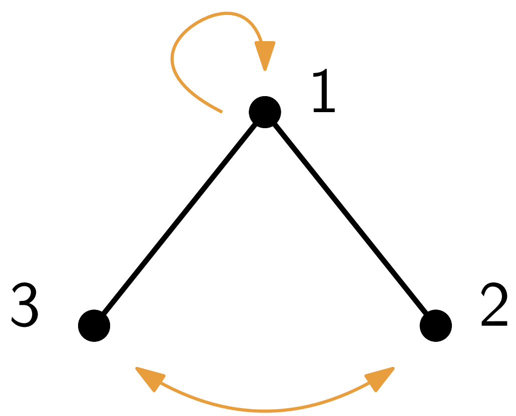


$$(1)(23) \longrightarrow s_1 s_2$$

Pólya theory



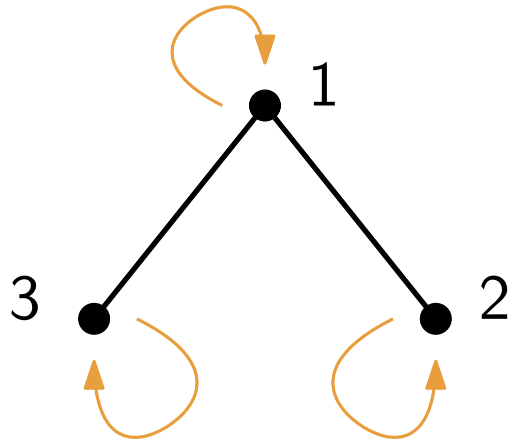
$(1)(2)(3) \longrightarrow s_1^3$



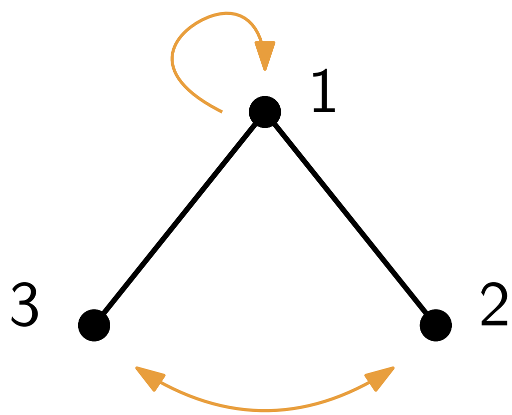
$(1)(23) \longrightarrow s_1 s_2$

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

Pólya theory



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



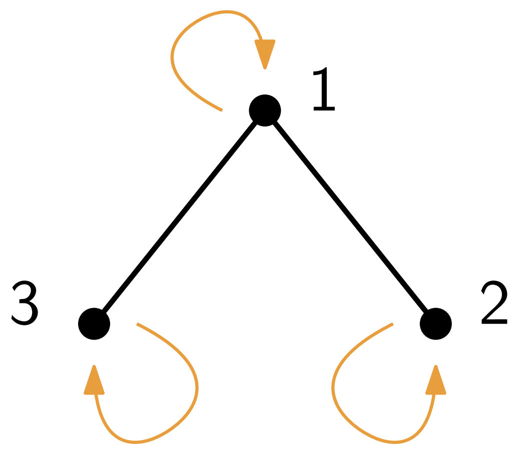
$(1)(23) \longrightarrow s_1 s_2$

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

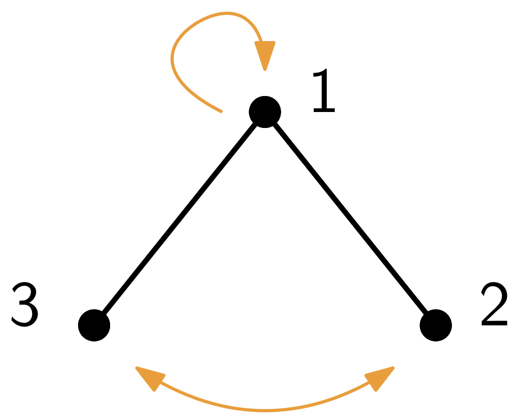
3 labelled graphs in the class

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

Pólya theory



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



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

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

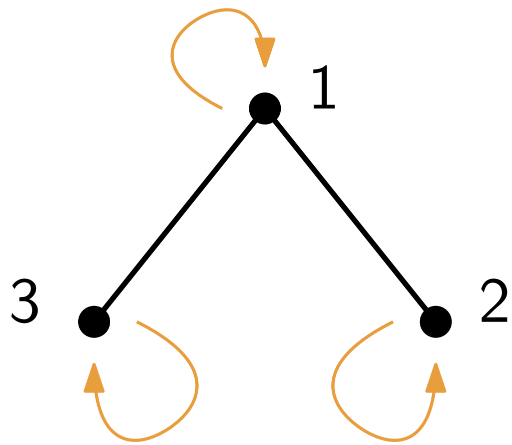
3 labelled graphs in the class

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

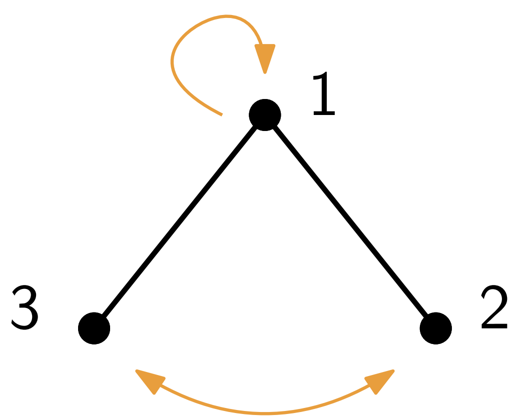
Cycle index sum

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

Pólya theory



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



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

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

$$\Big\downarrow \text{3 labelled graphs in the class}$$

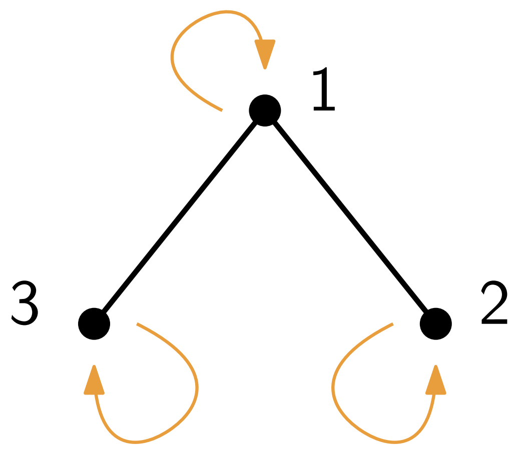
$$\frac{3}{3!}(s_1^3 + s_1 s_2)$$

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

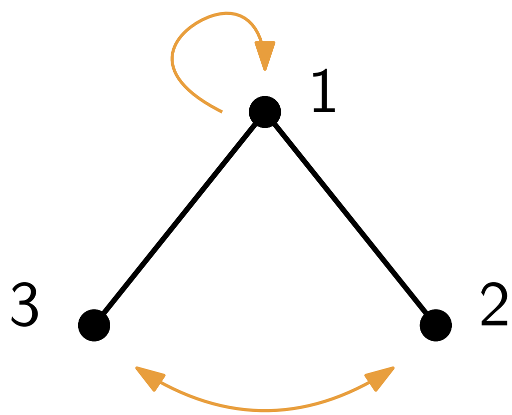
Theorem [Pólya 1937]
The OGF of the unlabelled class $\tilde{\mathcal{G}}$ is given by

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

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

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

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

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

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?

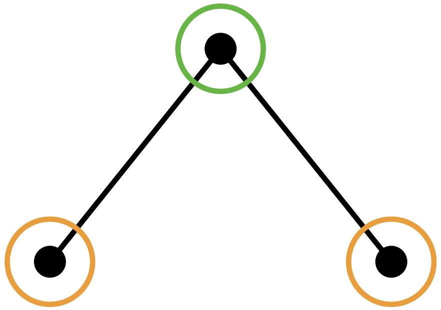
Unlabelled trees

Problem!

Unlabelled trees

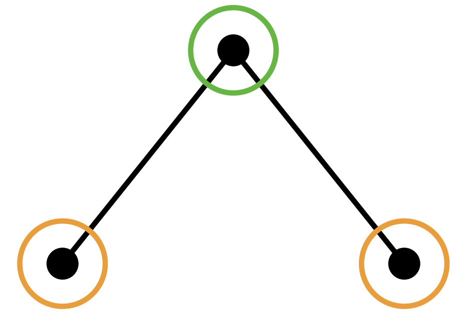
Problem!

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

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

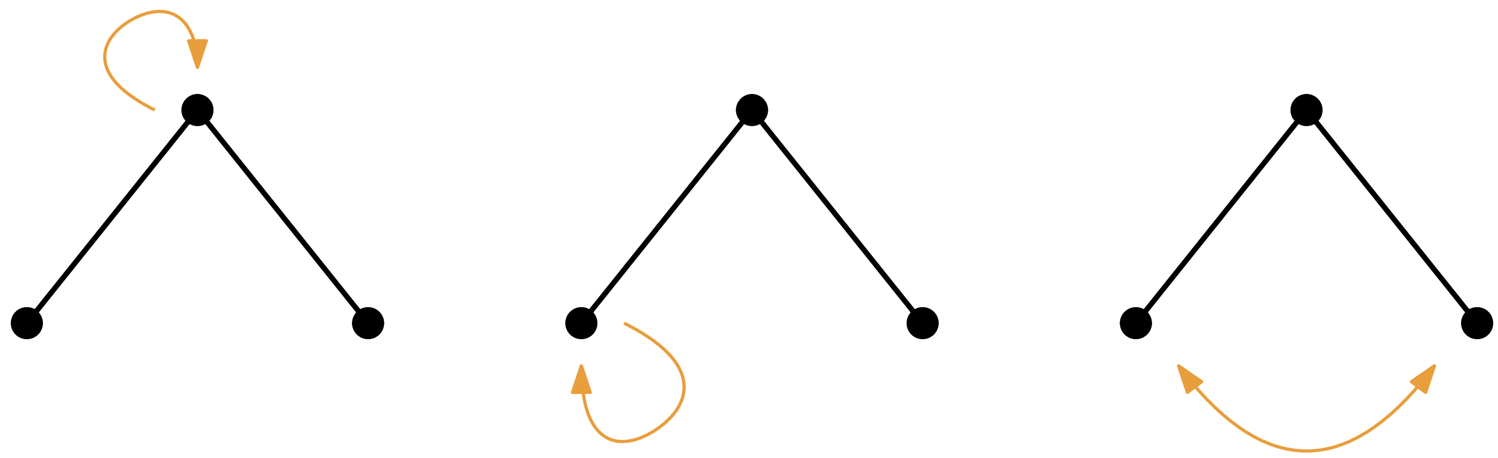
As $n \to \infty$ we have

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

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

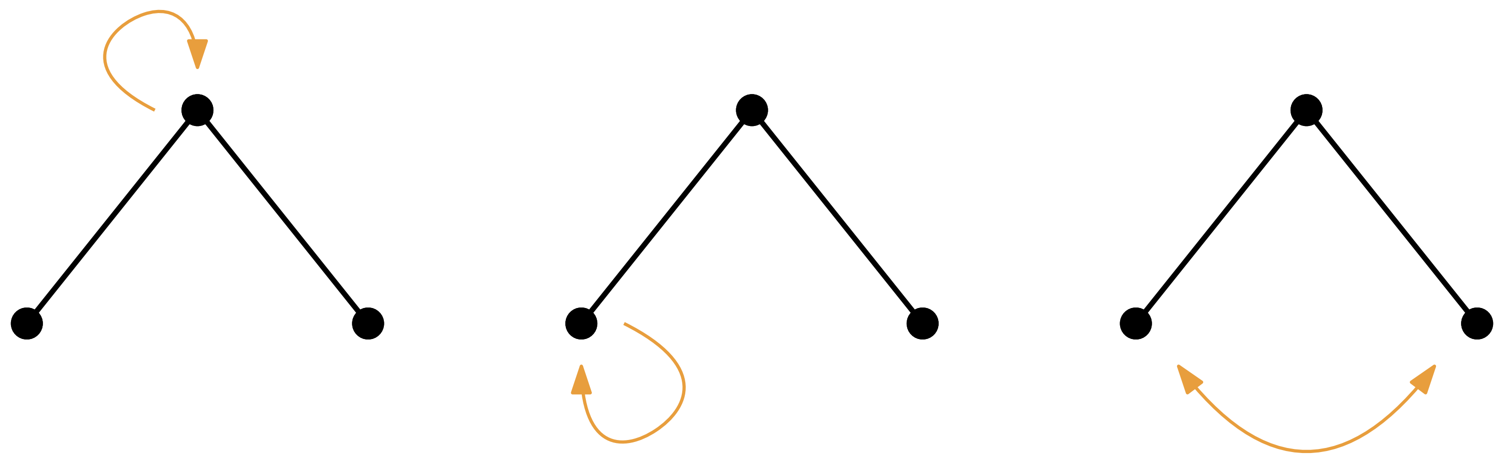
Proof. Using the **dissymmetry theorem**.

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.

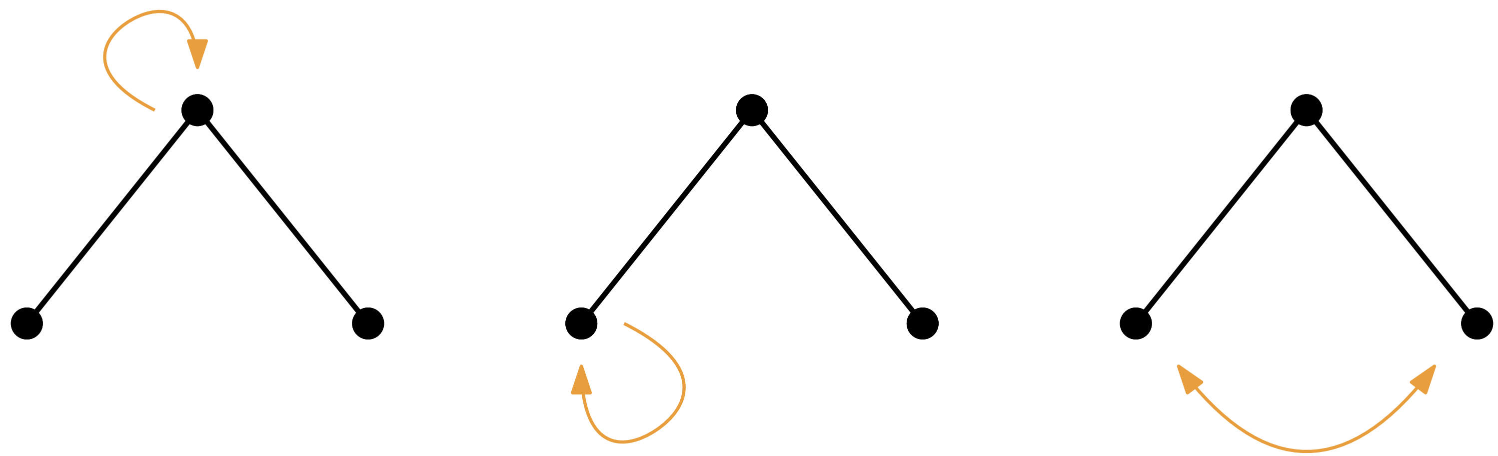
Cycle-pointing



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

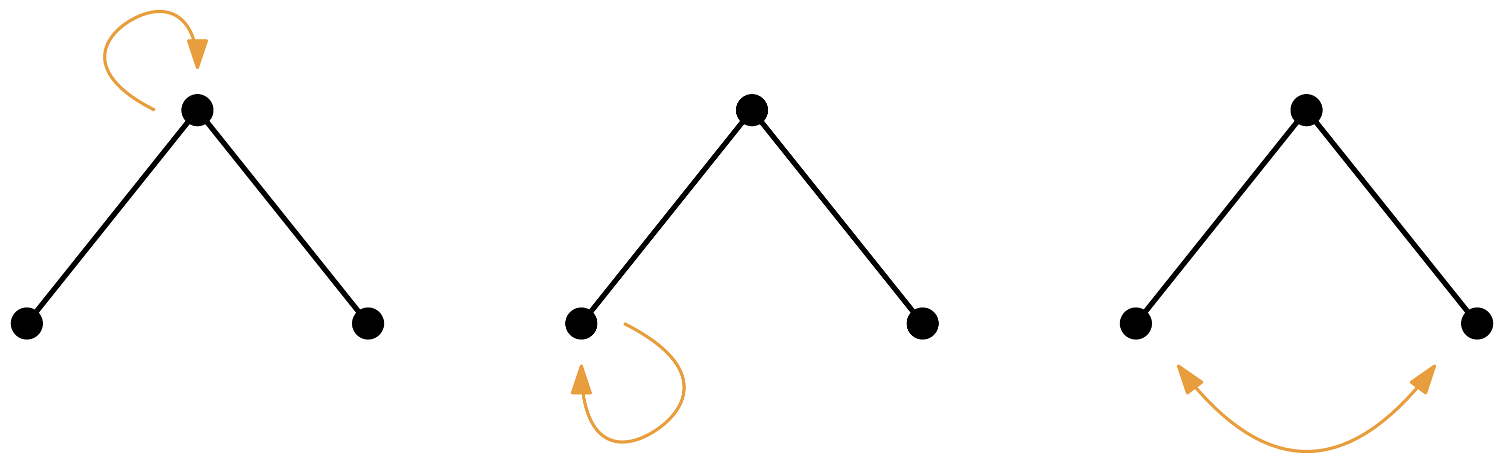


Definition. A **cycle-pointed graph** is a pair (G, c) where $G \in \mathcal{G}$ is an unlabelled graph and c is a cycle of some automorphism of G.

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



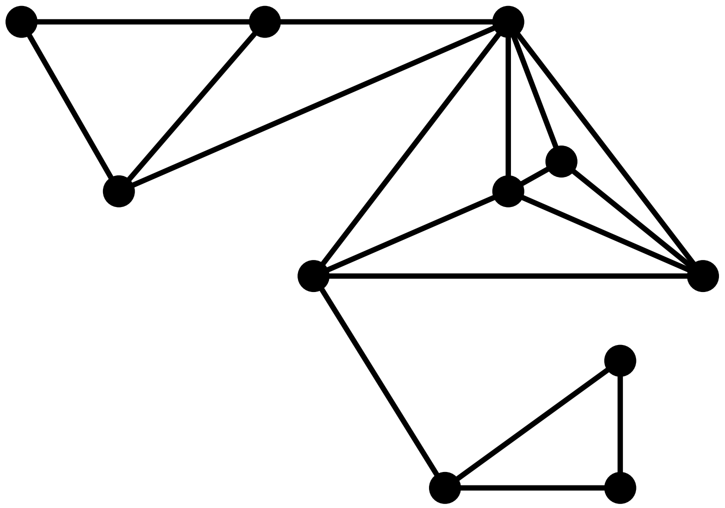
Definition. A **cycle-pointed graph** is a pair (G, c) where $G \in \mathcal{G}$ is an unlabelled graph and c is a cycle of some automorphism of G.

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!

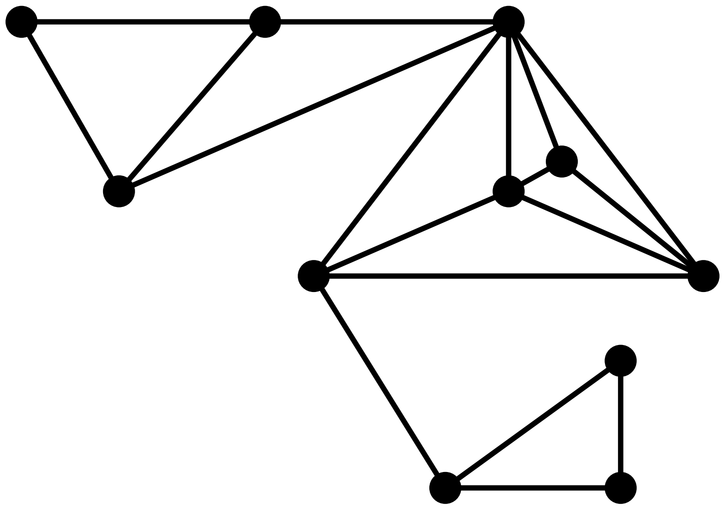
They extend Pólya theory to cycle-pointed graphs. In particular, they manage to unroot Pólya trees via cycle-pointing and they recover Otter's formula.

Our class of graphs



Chordal graphs with tree-width at most t

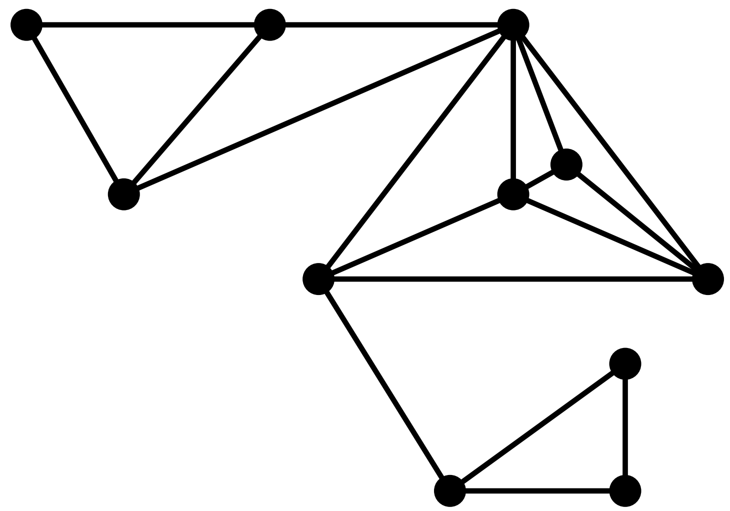
Our class of graphs



Chordal graphs with tree-width at most t

[Wormald, 1985]: they admit a decomposition into k-connected components.

Our class of graphs



Chordal graphs with tree-width at most t

[Wormald, 1985]: they admit a decomposition into k-connected components.

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

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

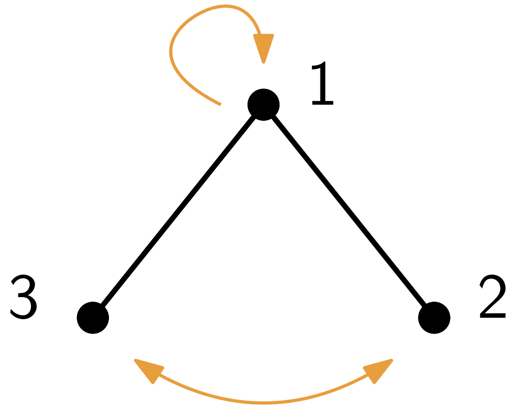
for some $c_t > 0$ and $\gamma_t > 1$

An extension of Pólya theory

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

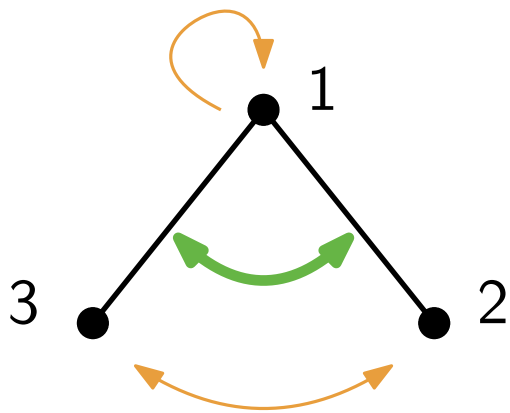
An extension of Pólya theory

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



An extension of Pólya theory

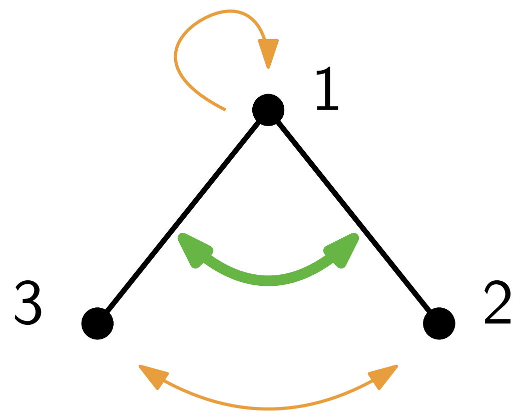
We need to take into account **cycles of cliques**, not just vertices.



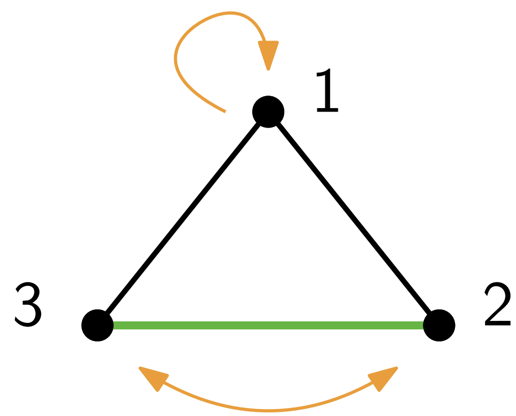
The **edges** are in a cycle of length 2.

An extension of Pólya theory

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



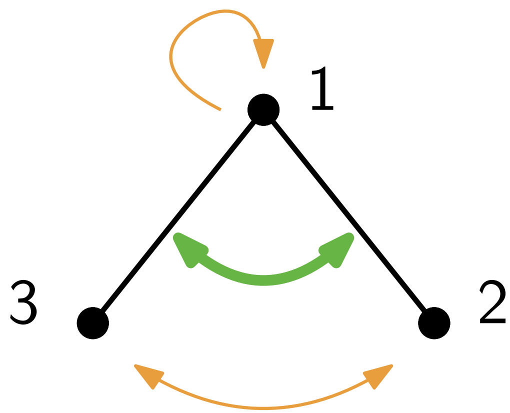
The **edges** are in a cycle of length 2.



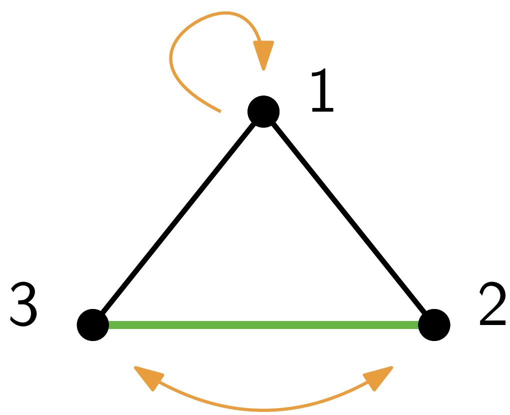
The new edge is in a cycle of length 1 but different type: it flips itself.

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.

What we do:

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

Results

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

Results

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

- [Gainer-Dewar (2012)], [Gainer-Dewar & Gessel (2014)]: system of equations to compute the OGF of unlabelled k-**trees** with n vertices.

Results

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

Results

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

- [Gainer-Dewar (2012)], [Gainer-Dewar & Gessel (2014)]: system of equations to compute the OGF of unlabelled k-**trees** with n vertices.

- [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 (rooted and unrooted) **2-trees**
- [Gainer-Dewar (2012)], [Gainer-Dewar & Gessel (2014)]: system of equations to compute the OGF of unlabelled k-**trees** with n vertices.
- [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.