

Cayley trees and increasing 1,2-trees: let's twist!

Julien Courtiel

Matthieu Dien

Paul Dorbec

Submitted: Mar 18, 2025; Accepted: Dec 22, 2025; Published: Jun 5, 2026

© The authors. Released under the CC BY-ND license (International 4.0).

Abstract

An increasing 1,2-tree is a labeled graph formed by starting with a vertex and then repeatedly attaching a leaf to a vertex or a triangle to an edge, the labeling of the vertices corresponding to the order in which the vertices are added. Equivalently, increasing 1,2-trees are connected chordal graphs of treewidth at most 2 labeled with a reversed perfect elimination ordering.

We prove that this family is equinumerous with Cayley trees, which are unconstrained labeled trees. In particular, the number of triangles in an increasing 1,2-tree corresponds to the number of twists. A twist (also called improper edge) is an edge whose endpoint closer to vertex **1** has a greater label than some vertex in the subtree rooted at the other endpoint of the edge.

We provide three proofs of this result, the first being based on similar recursive decompositions, the second on the resolution of generating functions, and the third describing a bijection. Finally, we propose efficient random generators for these two combinatorial families.

Mathematics Subject Classifications: 05A15, 05A19

keywords: chordal graphs, bounded treewidth, Cayley trees, generating function, bijection, random generation.

1 Introduction

1.1 Motivation

Chordal graphs, also known as triangulated graphs, are graphs where every cycle of length at least four admits a chord, i.e. an edge joining two non-consecutive vertices on the cycle.

Université Caen Normandie, ENSICAEN, CNRS, Normandie Univ, GREYC UMR 6072, F-14000 Caen, France. (julien.courtiet@unicaen.fr, matthieu.dien@unicaen.fr, paul.dorbec@unicaen.fr).

They form a well-studied class of graphs appearing in a variety of domains (for example in numerical linear algebra [26], in Bayesian networks [19, 21], in phylogenetics [14]), each benefiting from their unique properties and efficient algorithms for processing. Both these appearance domains and these efficient algorithms generally rely on the fact that chordal graphs admit a special vertex ordering called a *perfect elimination ordering*.

Because of their ubiquity, it seems crucial to design random generators for chordal graphs. This is borne out by the abundant literature on the subject since Wormald's seminal work [32]. Numerous algorithmic approaches have been developed for generating random chordal graphs [20, 30, 13, 22, 27], but these algorithms lack formal guarantees regarding the distribution of their output. More recently, a polynomial-time uniform sampler for chordal graphs (tractable up to 30 vertices) was proposed in [18].

The uniform distribution is not necessarily the most interesting one since chordal graphs of large size almost surely have the same shape [4]: about half the vertices form a clique, the rest form an independent set. This is why it seems worthwhile to restrict random generation to a subclass of chordal graphs. It makes sense to examine chordal graphs with bounded treewidth, as this plays an important role in many applicative contexts.

A well-studied subclass of chordal graphs with bounded treewidth is the family of k -trees [17, 3, 2], which are often described in terms of perfect elimination ordering. A perfect elimination ordering is an ordering of the vertices such that for each vertex v , its neighbors occurring after v in the order form a clique. If those cliques are all of size k (except for the last k vertices), then the chordal graph is called a k -tree. Alternatively, k -trees are also defined as maximal graphs with treewidth k (i.e. adding any edge to a k -tree increases its treewidth). Equivalently, the graphs with treewidth at most k are all the subgraphs of k -trees (called *partial k -trees*).

In the following, we consider $(\leq k)$ -trees, which are graphs admitting a perfect elimination ordering where the neighborhoods are cliques of size at most k (instead of exactly k for k -trees). Equivalently, the $(\leq k)$ -trees are the connected chordal partial k -trees, or our natural family of connected chordal graphs with treewidth at most k . It should be noted that the sets of k -trees do not cover all chordal graphs, whereas $(\leq k)$ -trees do.

Exact and asymptotic counting of $(\leq k)$ -trees have been conducted in [7, 8, 2]. In our work, we consider labeled $(\leq k)$ -trees where the labeling induces a perfect elimination ordering. This seems more meaningful in many contexts. This leads to the notion of *increasing $(\leq k)$ -trees*, defined in the next subsection.

The present paper only focuses on the case $k = 2$, for which we have discovered a surprising result: the number of increasing (≤ 2) -trees of size n is given by n^{n-2} , which is also the number of Cayley trees on n vertices. Our article aims to explain why, in the expectation that some of the results will transfer to increasing $(\leq k)$ -trees, for $k \geq 3$.

1.2 Definitions and results

In this paper, we consider simple undirected graphs whose vertices are labeled from 1 to n , where n is the number of vertices. We denote x the vertex with label x .

Recall that chordal graphs, besides being graphs with no chordless cycle, are graphs admitting a perfect elimination ordering, that is an ordering of the vertices v_1, \dots, v_n such that the neighborhood of v_i in the subgraph of G induced by the vertex set v_{i+1}, \dots, v_n is a complete subgraph.

Definition 1 (Increasing $(\leq k)$ -trees). Given a positive integer k , increasing $(\leq k)$ -trees are labeled graphs inductively defined as follows:

- the only 1-vertex increasing $(\leq k)$ -tree is an isolated vertex **1**.
- given a n vertices increasing $(\leq k)$ -tree T , consider a clique of size at most k and attach vertex $n + 1$ adjacent to all vertices of the clique.

Note that an increasing $(\leq k)$ -tree is a $(\leq k)$ -tree whose labeling is a reverse perfect elimination ordering.

In the following, we mostly study (≤ 2) -trees, that we name 1,2-trees. There are 16 increasing 1,2-trees of size 4, as shown by Figure 1. Observe that the top rows contain 1-trees (or just trees), the bottom row contains 2-trees, and in between, there are (≤ 2) -trees which are not k -trees.

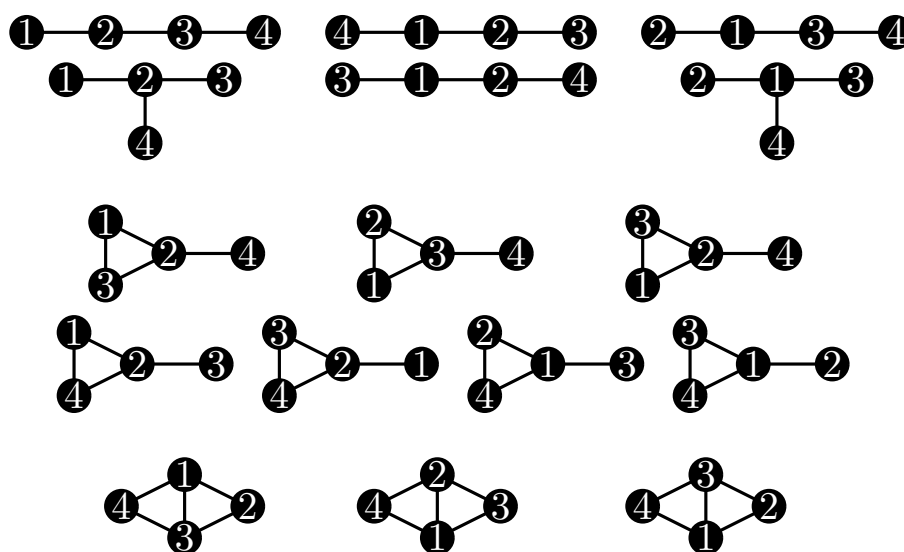


Figure 1: The 16 increasing 1,2-trees with 4 vertices, gathered by their numbers of edges.

The second family of combinatorial objects we consider consists of Cayley trees:

Definition 2 (Cayley trees). Cayley trees are trees with a labeling of the vertices from $\mathbf{1}$ to \mathbf{n} , where n is the number of vertices in the tree. Those trees are naturally rooted in $\mathbf{1}$.

Cayley trees are just trees. The noun adjunct *Cayley* is there to insist that the vertex labeling has no constraint. This name originates from the mathematician Arthur Cayley who found that the number of trees with n labeled vertices is n^{n-2} [9]. Figure 2 lists the 16 Cayley trees with 4 vertices.

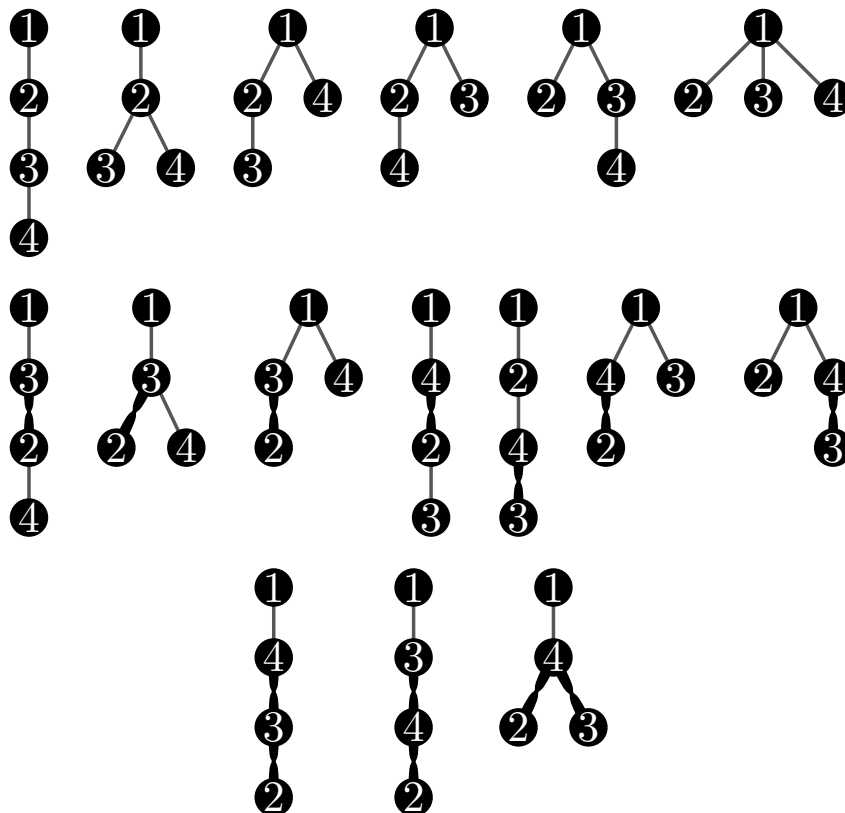


Figure 2: The 16 Cayley trees with 4 vertices. The trees are sorted by the number of twists (0 twist in the first row, 1 twist in the second row, 2 twists in the bottom row). The twists are represented differently (in bold and as if the edges have actually been twisted) than the increasing edges.

By rooting the Cayley trees in $\mathbf{1}$, we induce a notion of lineage (*child*, *parent*). A *descendant* of a vertex is either a child of the vertex or a child of some descendant of the vertex. When we talk about *subtrees*, we imply that they comprise a vertex and all its descendants.

Definition 3 (increasing edge, twist). Let T be a Cayley tree rooted at $\mathbf{1}$, and e be an edge of T from a vertex \mathbf{x} to a subtree S . We say that e is *increasing* if all vertices in S have a label greater than x . Otherwise, e is called a *twist*.

Consider for example the linear tree $1 - 3 - 6 - 2 - 5 - 4$. The edge between 2 and 5 is increasing: it connects 2 to the subtree $5 - 4$. On the contrary, the edge between 3 and 6 is a twist: the subtree rooted at 6 contains 2 whose label is smaller than 3 .

Twists were called *improper edges* in [28].

Definition 4 ($\min(S)$). Let S be a subtree in a Cayley Tree. The *minimum* of S , denoted by $\min(S)$, is the minimal label contained in S .

Note that an edge is increasing if and only if it links a vertex x to a subtree S such that $x < \min(S)$. Conversely, it is a twist if and only if $\min(S) < x$.

In this paper, we prove the following result:

Theorem 5. *Increasing 1,2-trees with n vertices and m edges are in bijection with Cayley trees with n vertices and $m - n + 1$ twists.*

The number of increasing 1,2-trees can thus be straightforwardly deduced from Cayley's formula [9].

Corollary 6. *There are n^{n-2} increasing 1,2-trees with n vertices.*

Note also that the number of triangles in a 1,2-tree is equal to $m - n + 1$ since a n -vertices tree has $n - 1$ edges and in a 1,2-tree, adding a vertex adjacent to an edge (a triangle) adds two edges instead of one. This could also be inferred from Euler's formula, as 1,2-trees are planar and their number of triangles is equal to the number of inner faces, though all faces are not triangles.

This paper provides three different proofs of Theorem 5. In Section 2, we show that the recursive decomposition following from the definition of increasing 1,2-trees is isomorphic to Shor's decomposition [28] of Cayley trees. In Section 3, we find the generating function of increasing 1,2-trees by solving a Partial Differential Equation and show that it can be expressed in terms of the generating function of Cayley trees. In Section 4, we describe an explicit bijection, using as an intermediary object a labeled forest where each edge is a twist. Finally, as an application, we present in Section 5 a linear uniform random generator for increasing 1,2-trees of size n running in $O(n)$ time.

2 Proof by recursive decomposition

This section shows that increasing 1,2-trees and Cayley trees have similar recursive decompositions. This gives a first inductive proof of Theorem 5.

2.1 Recurrence for the number of increasing 1,2-trees

The recurrence for increasing 1,2-trees follows from Definition 1:

Proposition 7. Let $c_{n,m}$ be the number of increasing 1,2-trees with n vertices and m edges. The numbers satisfy the recurrence

$$c_{n,m} = (n - 1)c_{n-1,m-1} + (m - 2)c_{n-1,m-2},$$

with initial condition $c_{1,0} = 1$ and $c_{1,k} = 0$ for $k \neq 0$.

Proof. There are two ways to construct an increasing 1,2-tree with n vertices from an increasing 1,2-tree T with $n - 1$ vertices:

- choose any vertex from T and attach \mathbf{n} to it,
- choose any edge e from T and form a triangle between \mathbf{n} and the endpoints of e .

We cannot obtain two identical 1,2-trees with this process (since the neighbors of \mathbf{n} are always different), and it is possible to recover T by removing \mathbf{n} and all its adjacent edges. The recurrence follows. \square

2.2 Recurrence for the number of Cayley trees

The recursive decomposition for Cayley trees in terms of twists is already known. Originally, Meir noticed in [29] that for every number z , the recursion relation

$$S_z(x + 1, y) = (x + z)S_z(x, y) + (x + y)S_z(x, y - 1)$$

has a solution satisfying

$$\sum_{y=1}^x S_z(x, y) = (x + z)^x.$$

(This is a generalization of a formula found by Ramanujan [25] – even though Meir got inspired not by the Indian mathematician, but from a recreational mathematical problem created by Shor.) With the notational change $c_{n,m} = S_2(n - 2, m - n + 1)$, we recover the recurrence for increasing 1,2-trees (see Proposition 7). So from Meir’s observation, we deduce that the number of increasing 1,2-trees satisfies

$$\sum_{m=n-1}^{2n-3} c_{n,m} = n^{n-2},$$

which is consistent with Corollary 6.

A combinatorial interpretation of this result in terms of Cayley trees has been independently found by Shor [28] on the one hand, and Dumont and Ramamonjisoa [12] on the other hand. It translates as follows.

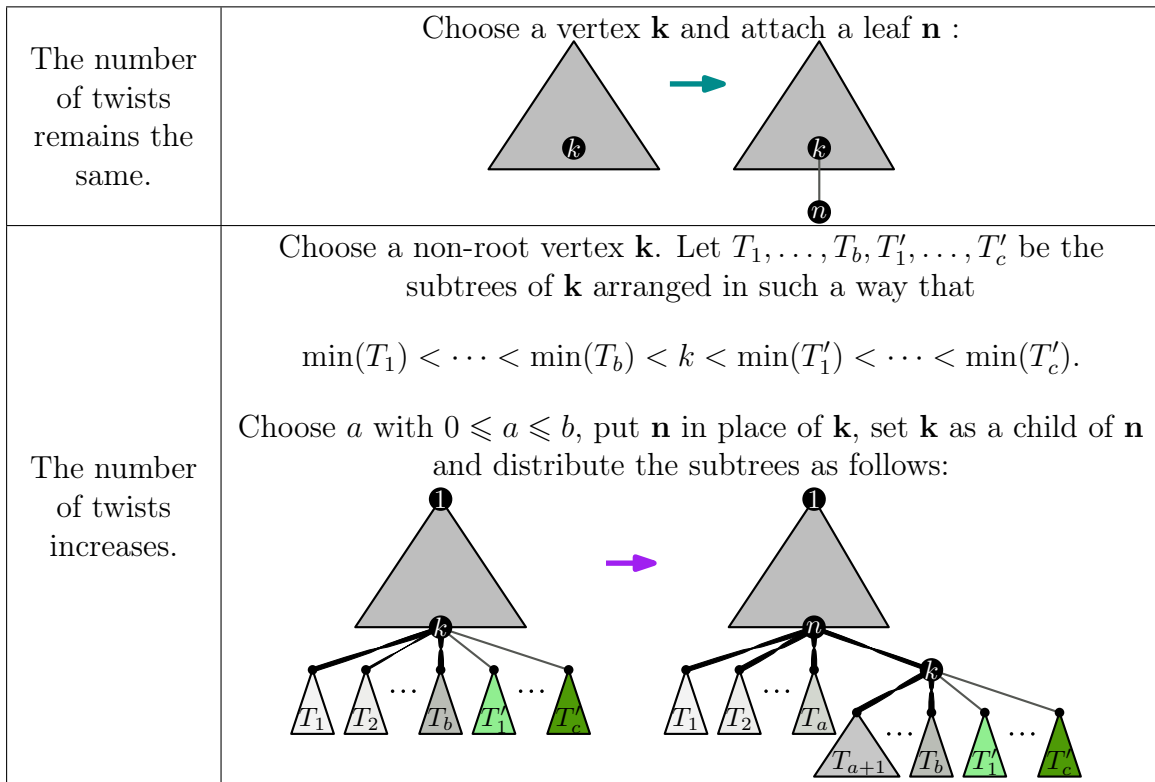


Figure 3: Shor’s idea to enumerate all Cayley trees while controlling the number of twists

Proposition 8. Let $c_{n,m}$ be the number of Cayley trees with n vertices and $m - n + 1$ twists. The numbers satisfy the recurrence

$$c_{n,m} = (n - 1)c_{n-1,m-1} + (m - 2)c_{n-1,m-2},$$

with initial condition $c_{1,0} = 1$ and $c_{1,k} = 0$ for $k \neq 0$.

Proof. Even though it is a well-known result, the authors make the choice of rewriting here Shor’s proof (for the sake of self-completeness). The idea is summarized by Figure 3.

Let T be a Cayley tree with $n - 1$ vertices. We build a new Cayley Tree T' with n vertices in one of the following ways:

- choose any vertex \mathbf{k} and attach \mathbf{n} to it as a leaf. The new edge is obviously increasing.
- choose any vertex \mathbf{k} different from $\mathbf{1}$. We denote by $T_1, \dots, T_b, T'_1, \dots, T'_c$ the subtrees rooted at a child of \mathbf{k} , sorted such that

$$\min(T_1) < \dots < \min(T_b) < k < \min(T'_1) < \dots < \min(T'_c).$$

Thus, b is the number of twists leaving \mathbf{k} and c is the number of increasing edges leaving \mathbf{k} .

First, change \mathbf{k} by \mathbf{n} and add \mathbf{k} as a child of \mathbf{n} .

Then, choose any $a \in \{0, \dots, b\}$. Leave T_1, \dots, T_a attached to \mathbf{n} and move the subtrees $T_{a+1}, \dots, T_b, T'_1, \dots, T'_c$ as children of \mathbf{k} .

By doing so, we create a new twist between \mathbf{n} and \mathbf{k} and since we replace \mathbf{k} by a vertex with a larger label, all twists formerly attached to \mathbf{k} remain twists.

Remark 9. After the addition of \mathbf{n} with the second method, the subtree rooted in \mathbf{k} is the subtree S such that $\min(S)$ is maximal among all subtrees rooted at a child of \mathbf{n} . Indeed, $\min(S)$ is larger than $\min(T_i)$ for $i \leq a$ since $\min(S) = \min(T_{a+1})$ if $a < b$ and $\min(S) = k$ if $a = b$.

We need now to make sure that we generate every Cayley tree of size n exactly once. Consider then a Cayley tree T with n vertices.

If \mathbf{n} is a leaf, then there is a unique way to create T , that is applying the first method in $T - \mathbf{n}$ on the parent of \mathbf{n} .

If \mathbf{n} is not a leaf, then it can only be obtained with the second method. Every edge leaving \mathbf{n} is a twist. Let $a + 1$ be the number of such edges, and S be the subtree attached to \mathbf{n} whose quantity $\min(S)$ is maximal over all these subtrees. We call \mathbf{k} the root of S . Note that by Remark 9, \mathbf{k} is the only possible choice for applying the second method to get T .

To recover the Cayley tree with $n - 1$ edges, we contract the edge between \mathbf{n} and \mathbf{k} (we keep \mathbf{k} for the label of the merged vertex). This corresponds to the reverse direction of the second method. Note that all of the a twists that were leaving \mathbf{n} (excluded the edge we contracted) are still twists in the smaller tree, and the minima of the associated subtrees are smaller than the minima of the other subtrees attached to \mathbf{k} . This is consistent with our choice in the second construction to attach the subtrees T_1, \dots, T_a to vertex \mathbf{n} for some $a \in \{0, \dots, b\}$, and the other subtrees to \mathbf{k} .

Finally, let us count the number of different Cayley trees T' with n vertices and $m - n + 1$ twists we generate:

- The number of trees generated from the first method is $(n - 1) \times c_{n-1, m-1}$, since T must have $n - 1$ vertices and $m - n + 1 = (m - 1) - (n - 1) + 1$ twists.
- The number of ways of choosing a non-root vertex \mathbf{k} and an integer a between 0 and b_k , which is the number of twists adjacent to \mathbf{k} in T , is

$$\sum_{\mathbf{k} \neq \mathbf{1}} (1 + b_k) = (n - 2) + \sum_{\mathbf{k} \neq \mathbf{1}} b_k. \quad (1)$$

Since no twist leaves $\mathbf{1}$, $\sum_{\mathbf{k} \neq \mathbf{1}} b_k$ is the total number of twists in T which is $(m - n + 1) - 1 = m - n$. So the right operand in (1) is equal to $m - 2$. In total, the number of trees generated from the second approach is $(m - 2) \times c_{n-1, m-2}$.

Summing the numbers for the two methods, we obtain the wanted recurrence. □

2.3 Bijection(s) via generating trees

Once a recursive decomposition of a combinatorial family such as above is found, it is easy to generate exhaustively all objects up to a certain size. Formally, we can build a *generating tree*, that is an infinite tree where the root is the only object of size 1 and the children of a node are the objects that can be constructed from the node thanks to the recursive decomposition.

Figure 4 shows the top of the generating trees of increasing 1,2-trees and the generating tree of Cayley trees.

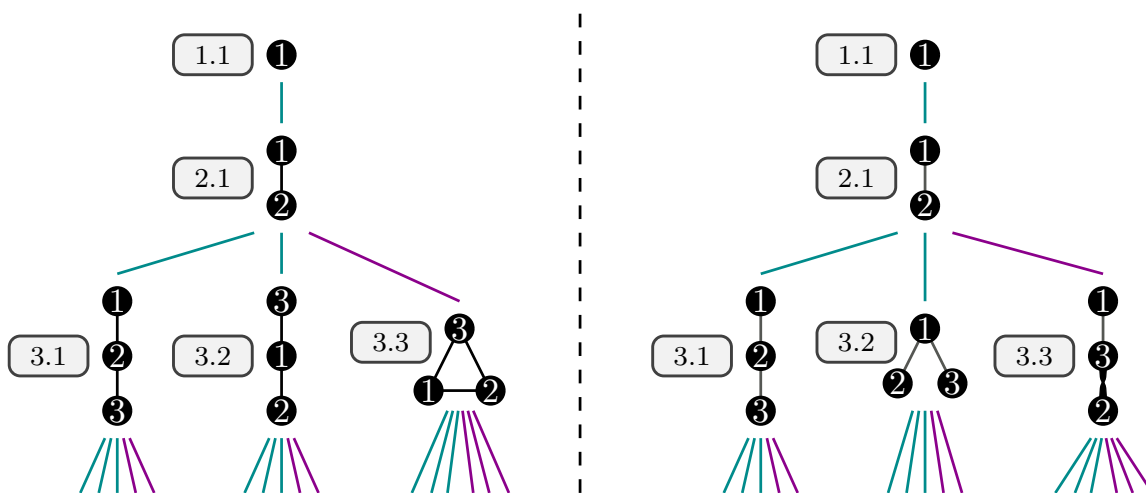


Figure 4: The first three levels of the generating tree of increasing 1,2-trees (*left*) and the generating tree of Cayley trees (*right*).

If the recursive decompositions of two combinatorial families are isomorphic (here like increasing 1,2-trees and Cayley trees – cf Propositions 7 and 8), then the generating trees are identical. A natural bijection between the two families, albeit an artificial one, can be obtained by numbering the nodes of the two generating trees in the same way (for example with a breadth-first traversal). Figure 4 illustrates how it can be done with increasing 1,2-trees and Cayley trees.

Such a bijection may not be very satisfactory, in particular since the ordering of the children in a generating tree is not quite explicit.

In Section 4 we are going to describe a more explicit bijection, which is unambiguous and has nice global properties.

3 Proof by generating functions

In this section, we find a formula for the generating function of 1,2-trees using a more analytic approach. Remark that this method should be generalizable to increasing ($\leq k$)-trees for every integer $k \geq 1$.

More precisely, we prove the following:

Theorem 10. *The exponential generating function of increasing 1,2-trees*

$$C(z, u) := \sum_{n, m \geq 0} c_{n, m} \frac{z^n u^m}{n!}$$

counted by vertices (variable z) and edges (variable u) is given by

$$C(z, u) = \text{Cayley} \left(\frac{uz + u - 1}{u} e^{\frac{1-u}{u}} \right) + \frac{1}{2u^2} - \frac{1}{2} \quad (2)$$

where $\text{Cayley}(z)$ is the exponential generating function of (unrooted) Cayley trees counted by the vertices, that is $\text{Cayley}(z) = \sum_{n \geq 0} n^{n-2} \frac{z^n}{n!}$.

By plugging $u = 1$ in the previous equation, we obtain that $C(z, 1) = \text{Cayley}(z)$ which gives another proof that 1,2-trees and Cayley trees are counted by the same numbers.

Remark 11. If we add a variable t in Cayley to count the number of twists, since $\text{Cayley}(z, t) = tC(\frac{z}{t}, t)$ by Theorem 5, we can express the bivariate generating function $\text{Cayley}(z, t)$ in terms of the univariate generating function $\text{Cayley}(z)$ thanks to the previous theorem:

$$\text{Cayley}(z, t) = t \text{Cayley} \left(\frac{z + t - 1}{t} e^{\frac{1-t}{t}} \right) + \frac{1}{2t} - \frac{t}{2}.$$

It should be interesting to find a combinatorial interpretation to this formula. We leave this as an open question.

In order to prove Theorem 5, let us start by giving an equation satisfied by $C(z, u)$.

Lemma 12. *The exponential generating function of increasing 1,2-trees $C(z, u)$ is solution of the Partial Differential Equation (PDE)*

$$\frac{\partial C}{\partial z} = 1 + zu \frac{\partial C}{\partial z} + u^3 \frac{\partial C}{\partial u} \quad (3)$$

with the initial condition $C(0, u) = 0$.

Proof. This PDE can be derived quite straightforwardly from the recurrence from Proposition 7.

We can also obtain this PDE with the help of symbolic method (see [16]). Indeed, the class \mathcal{C} of increasing 1,2-trees admits the combinatorial specification

$$\mathcal{C} = \mathcal{Z} \uplus \mathcal{Z}\mathcal{U} \times \Theta_{\mathcal{Z}}\mathcal{C} \uplus \mathcal{Z}\mathcal{U}^2 \times \Theta_{\mathcal{U}}\mathcal{C}$$

where \mathcal{Z} and \mathcal{U} are the atoms for the vertices and the edges, \uplus is the disjoint union, and $\Theta_{\mathcal{Z}}$ and $\Theta_{\mathcal{U}}$ are the pointing operators for the vertices and the edges, respectively. This translates in terms of generating function as

$$C_{\text{ord}}(z, u) = z + z^2 u \frac{\partial C_{\text{ord}}}{\partial z}(z, u) + z u^3 \frac{\partial C_{\text{ord}}}{\partial u}(z, u)$$

where $C_{\text{ord}}(z, u)$ is the *ordinary* generating function $\sum_{n,m \geq 1} c_{n,m} z^n u^m$. Then the PDE from the statement can be obtained via a Borel transform¹ and a differentiation with respect to z . \square

We can now tackle the proof of Theorem 10.

Proof of Theorem 10. First, we check that the expression of $C(z, u)$ given by Equation (2) is compatible with the PDE of Lemma 12. Since

$$\frac{\partial C}{\partial z}(z, u) = e^{\frac{1-u}{u}} \text{Cayley}'\left(\frac{uz + u - 1}{u} e^{\frac{1-u}{u}}\right)$$

and

$$\frac{\partial C}{\partial u}(z, u) = \frac{1 - zu}{u^3} e^{\frac{1-u}{u}} \text{Cayley}'\left(\frac{uz + u - 1}{u} e^{\frac{1-u}{u}}\right) - \frac{1}{u^3},$$

we have

$$(1 - zu) \frac{\partial C}{\partial z} - u^3 \frac{\partial C}{\partial u} = 1, \tag{4}$$

which is consistent with PDE (3). Note that we could have replaced Cayley by any other univariate function in Formula (2) and it would still satisfy PDE (3). However, $C(z, u)$ also needs to satisfy the initial condition $C(0, u) = 0$. To check this, we express Cayley in terms of the Lambert W function which is implicitly defined by $W(t) \exp(W(t)) = t$ and satisfies $W(t \exp(t)) = t$. More precisely, formula [10, Eq. (2.1)] translates

$$\text{Cayley}(x) = -W(-x) - \frac{1}{2}W(-x)^2 \tag{5}$$

and hence

$$\begin{aligned} C(0, u) &= \text{Cayley}\left(\frac{u-1}{u} e^{\frac{1-u}{u}}\right) + \frac{1}{2u^2} - \frac{1}{2} \\ &= -W\left(\frac{1-u}{u} e^{\frac{1-u}{u}}\right) - \frac{1}{2}W\left(\frac{1-u}{u} e^{\frac{1-u}{u}}\right)^2 + \frac{1}{2u^2} - \frac{1}{2} \\ &= \frac{u-1}{u} - \frac{(u-1)^2}{2u^2} + \frac{1}{2u^2} - \frac{1}{2} = 0. \end{aligned}$$

¹The *Borel transform* \mathcal{B} maps ordinary generating functions $f(z) = \sum_n f_n z^n$ to exponential generating functions $(\mathcal{B}f)(z) := \sum_n f_n \frac{z^n}{n!}$. This operator is linear and has nice properties like $\mathcal{B}(zf) = \int \mathcal{B}f$. See for example [6].

To conclude, we need to establish the uniqueness of the solution of PDE (3). It can be done by strong analytic theorems like the Cauchy-Kovalevskaya theorem [11]. However, we have chosen to roll out the *method of characteristics* [15] to show the reader how Formula (2) can be found.

Let $C(z, u)$ be any solution of PDE (3), and set two functions $z(t)$ and $u(t)$ such that

$$\begin{cases} z'(t) = 1 - z(t)u(t), & z(0) = 0 \\ u'(t) = -u(t)^3, & u(0) = 1/\sqrt{K}, \end{cases} \quad (6)$$

with K be a real positive parameter. The choice of (6) is justified by the fact that

$$\frac{\partial}{\partial t} (C(z(t), u(t))) = (1 - z(t)u(t)) \frac{\partial C}{\partial z}(z(t), u(t)) - u(t)^3 \frac{\partial C}{\partial u}(z(t), u(t)) = 1$$

because C satisfies PDE (4). Therefore, this equation can be integrated from 0 to t :

$$C(z(t), u(t)) = C(z(0), u(0)) + t = C(0, 1/\sqrt{K}) + t$$

and hence

$$C(z(t), u(t)) = t \quad (7)$$

since $C(0, u) = 0$. We now solve the system of differential equations (6). First, we see that $-\frac{u'}{u^3}$ (which is equal to 1) is the derivative of $\frac{1}{2u^2}$. So by integrating between 0 and t we see that $\frac{1}{2u(t)^2} - \frac{1}{2u(0)^2} = t$, and so

$$u(t) = \frac{1}{\sqrt{K + 2t}}.$$

Plugging this into the differential equation for $z(t)$ (see Equation (6)) yields a linear ordinary differential equation of order 1. In order to solve it, multiply both sides by $\exp(\sqrt{K + 2t})$, and obtain

$$e^{\sqrt{K+2t}} z'(t) + \frac{1}{\sqrt{K + 2t}} e^{\sqrt{K+2t}} z(t) = e^{\sqrt{K+2t}}.$$

The left-side member is at sight the derivative of $e^{\sqrt{K+2t}} z(t)$, while the indefinite integral of $e^{\sqrt{K+2t}}$ is $(\sqrt{K + 2t} - 1) e^{\sqrt{K+2t}}$. This is why by integration from 0 to t we get

$$e^{\sqrt{K+2t}} z(t) - e^{\sqrt{K}} z(0) = \left(\sqrt{K + 2t} - 1\right) e^{\sqrt{K+2t}} - \left(\sqrt{K} - 1\right) e^{\sqrt{K}}$$

and since $z(0) = 0$,

$$z(t) = \sqrt{K + 2t} - 1 - \left(\sqrt{K} - 1\right) e^{\sqrt{K} - \sqrt{K+2t}}.$$

Now that we have the expressions of $z(t)$ and $u(t)$, we want to solve $\begin{cases} z(t) = z \\ u(t) = u \end{cases}$ in terms of K and t . Plugging $\sqrt{K+2t} = u^{-1}$ into the equation $z(t) = z$ gives

$$z = u^{-1} - 1 - (\sqrt{K} - 1) e^{\sqrt{K}-u^{-1}}$$

which can be rewritten as

$$(\sqrt{K} - 1) e^{\sqrt{K}-1} = (-z + u^{-1} - 1) e^{u^{-1}-1}.$$

Using the Lambert W function described above, we then see that

$$\sqrt{K} - 1 = W\left((\sqrt{K} - 1) e^{\sqrt{K}-1}\right) = W\left((-z + u^{-1} - 1) e^{u^{-1}-1}\right),$$

which implies that

$$K = W\left((-z + u^{-1} - 1) e^{u^{-1}-1}\right)^2 + 2W\left((-z + u^{-1} - 1) e^{u^{-1}-1}\right) + 1.$$

Since $K = \frac{1}{u^2} - 2t$, we have

$$t = -\frac{1}{2}W\left((-z + u^{-1} - 1) e^{u^{-1}-1}\right)^2 - W\left((-z + u^{-1} - 1) e^{u^{-1}-1}\right) + \frac{1}{2u^2} - \frac{1}{2}.$$

Using Equation (7) and the expression of the Cayley function in terms of W (see Equation (5)), we obtain a formula which is consistent with Equation (2). \square

4 Proof by bijection

This section establishes a direct bijection between increasing 1,2-trees and Cayley trees. Actually, we propose two descriptions of the mapping from 1,2-trees to Cayley trees (see Figure 5 for an overview). The first one uses three different steps and intermediary structures. It is useful because each step is easily reversible, which sums up into the reciprocal bijection. The second is condensed in a single step, but it is more difficult to reverse. Though, it is much easier to see in this condensed description a generalization of Shor's decomposition of Cayley trees (as in Figure 3).

4.1 From increasing 1,2-trees to Cayley trees

Let G be an increasing 1,2-tree on n vertices, which we are going to transform into a Cayley tree $\tau(G)$. Remarkably, this direction makes no mention of twists. The increasing 1,2-tree from Figure 6 serves as an example for the construction.

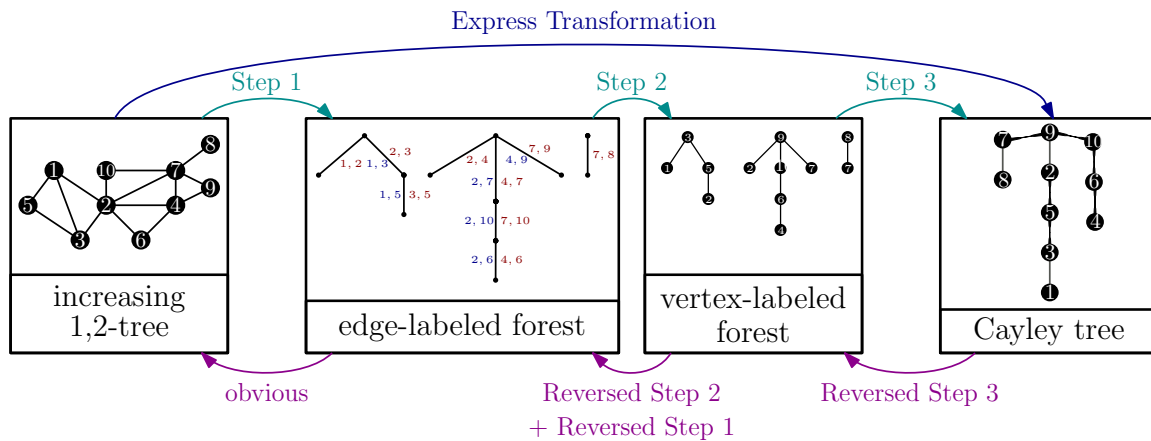


Figure 5: Overview of the bijection from increasing 1,2-trees to Cayley trees.

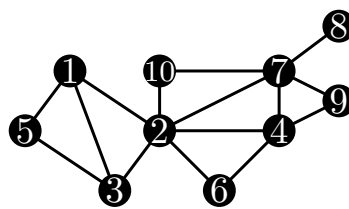


Figure 6: An example of an increasing 1,2-tree

Step 1: transformation into a plane forest F labeled by edges of G . This step is illustrated in Table 1.

Forest F is plane, meaning that for each vertex, its children are ordered from left to right. However, the order between the trees of F does not matter.

Each edge of F will carry one or two labels, that we write along the edge on the left or on the right. When there is only one label, it will always be on the right, while when there are two labels, they will be distributed on both sides. These labels correspond to the edges of G .

Let us initialize F as an empty forest. We successively consider every vertex $v \neq 1$ of G , starting from vertex 2 and ending to vertex n . By construction of an increasing 1,2-tree, vertex v has 1 or 2 neighbors smaller than it.

- **Rule 1.** If v has only one such neighbor x (which means it was attached to x as a leaf during the building of G), then we add a new rooted tree to F , consisting in a sole edge labeled on its right by (x, v) .
- **Rule 2.** If v has two neighbors x and y such that $x < y < v$ (which means it was attached to edge xy as a triangle during the building of G), then we modify the forest depending on whether the label (x, y) is on the left or right of the edge e that

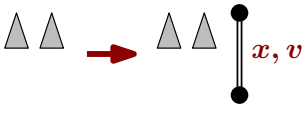
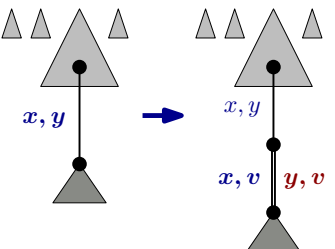
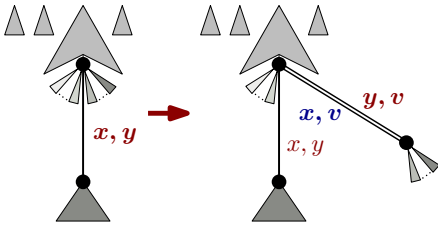
<p>Rule 1. v is attached to a vertex x as a leaf</p>		
<p>Rule 2. v is attached to x and y as a triangle</p>	<p>the label is on the left</p> 	<p>the label is on the right</p> 

Table 1: Three rules to transform an increasing 1,2-tree into a plane forest labeled by edges (Step 1 of the transformation from increasing 1,2-trees to Cayley trees)

carries it.

- **Rule 2 Left.** If (x, y) is on the left of e , then we split e into two edges e and e' , with e above e' (see Table 1 bottom left). Doing so, every subtree incident and below e becomes incident and below e' instead. We put the label (x, v) on the left of e' , and the label (y, v) on the right of e' . The labels on e remain the same.
- **Rule 2 Right.** If (x, y) is on the right of e , then we insert a new edge e' in F as the right sibling of e , pushing below e' every subtree that shares an endpoint with e and is on the right of e (see Table 1 bottom right). We put a label (x, v) on the left of e' , and a label (y, v) on the right of e' .

A full example for the graph of Figure 6 is detailed at Table 2.

Definition 13 (root edge). The *root edge* of a tree is the leftmost edge incident to the root.

Lemma 14. Let t be a tree of F . At the end of Step 1, the root edge of t leads to a leaf and does not carry a left label, but a right label of the form (a, b) , where a is the smallest integer appearing in t .

Proof. In order to create the tree t , Rule 1 must have been applied. At this moment, t consists in a single edge with a right label of the form (a, b) with $a < b$. The property holds at this point and we prove that it continues at each substep of Step 1 by induction.

Let us suppose that an edge e' is added to F . If e' is in a different tree than t , then it is clear that the property is still satisfied.

vertex v	2	3	4	5
forest F				
vertex v	6		7	
forest F				
vertex v	8		9	
forest F				
vertex v	10			
forest F				

Table 2: Construction of the forest F edge by edge, starting from the 1,2-tree of Figure 6

Assume that e' is added to t . It must come with Rule 2 Left or Rule 2 Right. Let e be the edge on which Rule 2 has been applied.

If e is not the root edge of t , then the latter is still a leaf and keeps the same right label.

Let us assume now that e is the root edge of t . Since by induction e does not have a left label, Rule 2 Right must have been applied. A new right sibling is added to e , but nothing below e . The edge e remains a leaf.

Finally, the integers x, y, v appearing in the left label (x, v) and right label (y, v) of e' must be at least equal to a since v is the largest integer in the forest so far, and x and y were already in t before insertion of e' (and so must be at least a by induction). \square

In the following, we use the following definition.

Definition 15 (end label). The *end label* of an edge e with right label (x, y) is y .

Note that if e also has a left label (x', y') , then by construction $y = y'$.

Step 2: label the vertices of F . Let t be any tree of F . We are going to label the vertices of t by integers between 1 and $|G|$.

First, consider the root edge of t and its right label (x, y) . We label the endpoint of the root edge (which is a leaf by Lemma 14) with x .

Next, we proceed to a Depth First Search of t where one favors the rightmost edge at each step. The visit order of this DFS gives an ordering for the vertices: $v_1, \dots, v_{|t|}$, and for the edges: $e_1, \dots, e_{|t|-1}$. An example is shown in Figure 7.

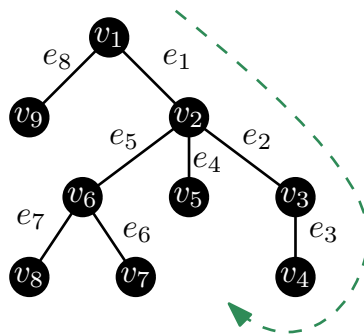


Figure 7: Example of a rightmost DFS of a tree.

Then, for k between 1 and $|t| - 1$, we label v_k by the end label of e_k . Vertex $v_{|t|}$ is the endpoint of the root edge of t , which is already labeled.

An example of this step is shown at Figure 8.

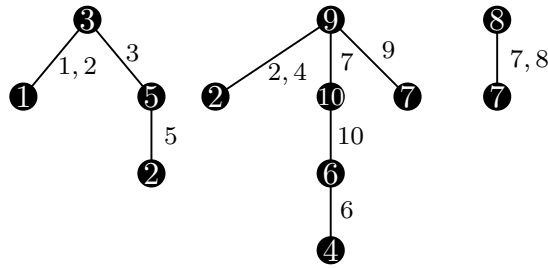


Figure 8: Result of Step 2 for the increasing 1,2-tree of Figure 6. We have kept the end label of all edges and the whole right label of the root edge to illustrate how they are used to label vertices.

Step 3: merge the trees of F . Remove any edge label from F and forget the fact that F is plane (i.e. ordered). Merge every vertex of F that carries the same vertex. There results a Cayley tree, which we denote $\tau(G)$. Following our example, the result is depicted at Figure 9.

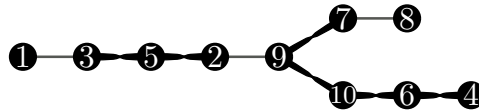


Figure 9: Cayley Tree in bijection with the 1-2 increasing tree of Figure 6.

4.2 Properties of τ

We now prove several properties on the map τ defined in the previous subsection. These properties will be useful to describe the reciprocal of τ . Incidentally we show the following proposition, which induces Theorem 5 provided that τ is a bijection.

Proposition 16. *The map τ sends increasing 1,2-trees with n vertices and m triangles to Cayley trees with n vertices and m twists. More specifically, τ transforms each 2-connected component of an increasing 1,2-tree into a subtree where every edge but one is a twist.*

In particular, increasing trees (i.e. increasing 1,2-trees where we only attach new vertices as leaves) are preserved by τ , and increasing 2-trees (i.e. increasing 1,2-trees where we only attach new vertices as triangles) are sent to Cayley trees where every edge is a twist, except for one which must be the only edge leaving 1.

We need beforehand the following definition.

Definition 17 (EDDY²). Let \mathbf{v} be a node of a plane forest, not incident to the root edge of a tree. Consider \mathbf{a} the youngest ascendant of \mathbf{v} (\mathbf{v} included) having left siblings. We

²The name is a homage to the famous French singer Eddy Mitchell https://en.wikipedia.org/wiki/Eddy_Mitchell.

define the EDDY of \mathbf{v} , denoted by $\text{EDDY}(\mathbf{v})$, as the edge linking the sibling just on the left of \mathbf{a} to its parent. By extension, for an edge \mathbf{uv} where \mathbf{v} is the child of \mathbf{u} , $\text{EDDY}(\mathbf{uv})$ denotes $\text{EDDY}(\mathbf{v})$.

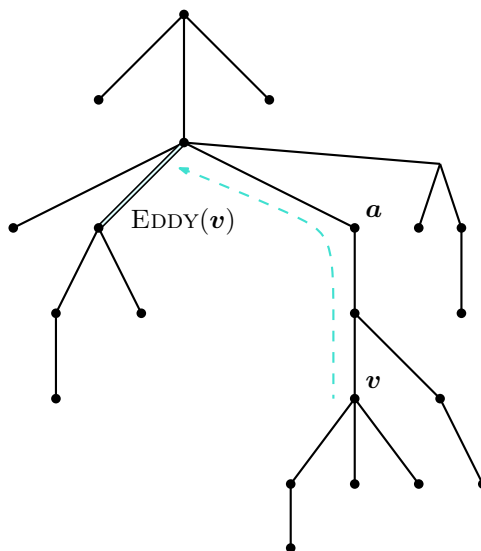


Figure 10: Example of $\text{EDDY}(\mathbf{v})$, given a vertex \mathbf{v} from a tree

This definition is illustrated by Figure 10. Note that all the vertices in the leftmost branch starting at \mathbf{a} share the same EDDY. Note also that during a rightmost DFS, the edge visited immediately after visiting any edge e leading to a leaf is $\text{EDDY}(e)$.

By a careful analysis of the rules of Step 1, we infer the following observation.

Lemma 18. *For every non-root edge e in the forest of Step 1, the labels of $\text{EDDY}(e)$ do not vary during the whole construction of the forest.*

Proof. Let $e = \mathbf{uv}$ be a non-root edge of the forest and let f be an edge added to the forest after using any rule from Step 1. We show that the insertion of f does not change the labels of $\text{EDDY}(e)$.

Let \mathbf{a} be the youngest ascendant of \mathbf{v} with left siblings before insertion of f . In any of the three rules, f has no right sibling. Thus the insertion of f does not add a left sibling to an existing vertex. Therefore the youngest ascendant of \mathbf{v} with left siblings after insertion of f is still \mathbf{a} , or an ascendant of \mathbf{a} .

If it is still \mathbf{a} , then the edge on the left of \mathbf{a} is the same before and after the insertion of f (because f cannot be a left sibling of another edge). So $\text{EDDY}(e)$ remains the same.

If the youngest ascendant of \mathbf{v} with left siblings is not \mathbf{a} anymore, this means that \mathbf{a} has lost its left siblings. So f must be the new edge between \mathbf{a} and its parent. Since the insertion of f can be reverted by contracting f , the new parent of \mathbf{a} must have the same

left sibling than \mathbf{a} before insertion of f . So the youngest ascendant of \mathbf{v} with left siblings is now the parent of \mathbf{a} , and $\text{EDDY}(e)$ is still the same edge. \square

From this, we can deduce the following.

Lemma 19. *For every non-root edge e in the forest of Step 1, the end label of $\text{EDDY}(e)$ is less than the end label of e .*

Proof. For an edge e , $\text{EDDY}(e)$ was already in the forest when e is introduced. So the end label of $\text{EDDY}(e)$ is necessarily less than the end label of e (since the edges are introduced in increasing order of their end label). The property remains true by Lemma 18. \square

Lemma 20. *Given a vertex \mathbf{u} of the forest of Step 1 with children $\mathbf{v}_1, \dots, \mathbf{v}_k$, the end labels of all edges in the subtrees rooted at v_{i+1}, \dots, v_k are greater than the end label of the edge uv_i . Moreover, the end labels of the edges going from u to its children are increasingly ordered from left to right.*

Proof. The property directly comes from one or several uses of Lemma 19. \square

In the following, we use the natural extension of the definition of twists (Definition 3) to rooted subtrees of Cayley trees (i.e. rooted trees whose set of labels is not necessarily $\{1, \dots, n\}$).

Lemma 21. *At the end of Step 2, every edge of every tree is a twist.*

Proof. Before commencing the proof, remark the two following points:

1. By construction (application of Step 2), the label of a leaf ℓ is the end label of $\text{EDDY}(\ell)$.
2. Also by construction, a non-leaf vertex inherits the end label of the edge towards its rightmost child.

Now consider an edge $e = \mathbf{u}\mathbf{v}$ (where \mathbf{v} is the child of \mathbf{u}).

If e is the root edge of its tree, then \mathbf{v} is a leaf (Lemma 14) which inherits (by construction) the smallest label of the tree. Therefore e is a twist.

Now let us assume that e is not a root edge, hence $\text{EDDY}(e)$ is defined. Let ℓ be the leftmost leaf descendant of \mathbf{v} . We are going to prove that the label of \mathbf{u} is greater than the label of ℓ . To do so, we introduce v' , the rightmost child of \mathbf{u} (which may or may not be v).

We have the chain of inequalities:

$$\begin{aligned}
u &= \text{end label}(uv') && \text{(by point 2)} \\
&\geq \text{end label}(e) && \text{(by Lemma 20)} \\
&> \text{end label}(\text{EDDY}(e)) && \text{(by Lemma 19)} \\
&= \text{end label}(\text{EDDY}(\ell)) && \text{(since } \text{EDDY}(e) = \text{EDDY}(\ell) \text{ by definition of } \ell) \\
&= \ell && \text{(by point 1).}
\end{aligned}$$

This implies that e is a twist. □

We can now show the proposition stated at the beginning of this subsection.

Proof of Proposition 16. Let G be an increasing 1,2-tree. We can recover $\tau(G)$ by merging the trees of the forest at the end of Step 2 in a specific order. By iterating over all trees t of F with respect to their minimal label, we amalgamate t with the previously joined trees by merging the smallest vertex of t with the other vertex with the same label. By construction and by Lemma 14, the smallest vertex of t is a leaf incident to the root edge of t . So if e is the root edge of t , then e is increasing in $\tau(G)$. By Lemma 21, all other edges of t are twists.

Thus the final number of twists in $\tau(G)$ is equal to the number of non-root edges at the end of Step 1, which is also the number of times where Rule 2 has been used, and so the number of triangles in G .

To prove the more specific statement of the proposition, note that the 2-connected components of G are in correspondence with trees of F . Indeed, all the edges of a 2-connected component appear as left and right labels of the corresponding tree. This can be proved by induction. Since all the edges of a tree t of F are twists at the end of Step 2 (by Lemma 21), t forms at the end of Step 3 a subtree of $\tau(G)$ where every edge is a twist, except the root edge. □

Before considering the reverse direction, let us establish the following properties that will prove useful.

Lemma 22. *Let T be a tree of F and S be any subtree of T . At the end of Step 2, the vertex with the smallest label in S is the leftmost leaf of S .*

Proof. Let \mathbf{v} be the root of S , and ℓ be the leftmost leaf of S .

If \mathbf{v} is also the root of T , then the lemma obviously holds by Lemma 14 (ℓ is the leftmost child of \mathbf{v} and carries the smallest vertex of T).

So we can suppose that \mathbf{v} has a parent, say \mathbf{u} . As mentioned by point 1 of the previous proof, ℓ is the end label of $\text{EDDY}(\ell)$, which is the same as $\text{EDDY}(\mathbf{u}\mathbf{v})$. By Lemma 20, the end label of $\text{EDDY}(\mathbf{u}\mathbf{v})$ must be smaller than the end label of any edge of S . So ℓ is smaller than the end label of any edge of S .

Let \mathbf{x} be a vertex of S different from ℓ . Since the vertex labels are induced by a rightmost DFS, the label of \mathbf{x} is necessarily an end label of an edge of S . According to what we said above, the label of \mathbf{x} is greater than the label of ℓ . \square

Lemma 23. *Let \mathbf{u} be a vertex of F . At the end of Step 2, the subtrees rooted in \mathbf{u} are sorted from left to right with respect to their minimum.*

Proof. Let $\mathbf{v}_1, \dots, \mathbf{v}_k$ be the children of \mathbf{u} , in this order, and T_1, \dots, T_k the corresponding subtrees. Let us fix i and j such that $1 \leq i < j \leq k$.

If $i = 1$, then by Lemma 22, the minimum of T_1 is the leftmost leaf of T_1 , which is also the leftmost leaf of the subtree rooted at \mathbf{u} , and so the minimum of the whole subtree. In particular, the minimum of T_1 is smaller than the minimum of T_j .

If $i > 1$, then the minimum of T_i and the minimum of T_j are given by the leftmost leaves of T_i and T_j (again by Lemma 22). But these leaves are labeled with the end labels of $\mathbf{u}\mathbf{v}_{i-1}$ and $\mathbf{u}\mathbf{v}_{j-1}$, respectively. By Lemma 20, the former is smaller than the latter, which concludes the proof. \square

4.3 From Cayley trees to increasing 1,2-trees

In this subsection, we describe how to recover the increasing 1,2-tree $\tau^{-1}(T)$, given a Cayley tree T . The process will be done by reversing the 3 steps from Subsection 4.1. Remember that T is rooted at $\mathbf{1}$.

Reversed Step 3: cutting T into a forest and order the children at each node. This step is illustrated by Figure 11.

For each increasing edge $\mathbf{u}\mathbf{v}$ of T where \mathbf{v} is the child of \mathbf{u} , split T in two by detaching edge $\mathbf{u}\mathbf{v}$ from \mathbf{u} (vertex \mathbf{u} is duplicated). We root the new tree at vertex \mathbf{v} . Since $\mathbf{1}$ is only linked to the rest of the tree by increasing edges, the process must induce a tree with a single vertex $\mathbf{1}$. Discard this tree.

Once done, order the children at each node so that the subtrees are ordered with respect to their minimal vertex (see Definition 4). Note that the leftmost child of the root at each tree is a leaf: since we have cut the tree at each increasing edge $\mathbf{u}\mathbf{v}$, the minimal vertex of the subtree is \mathbf{u} , which must be the leftmost child of the root after reordering the children.

Reversed Step 2: computing the end-label of the edges Let t be any tree of F . We are going to add left labels and right labels of the form $(?, v)$ to edges of t , where the question mark $?$ is a temporary symbol which will be replaced by a vertex label in the next (reversed) step, and v is the vertex label.

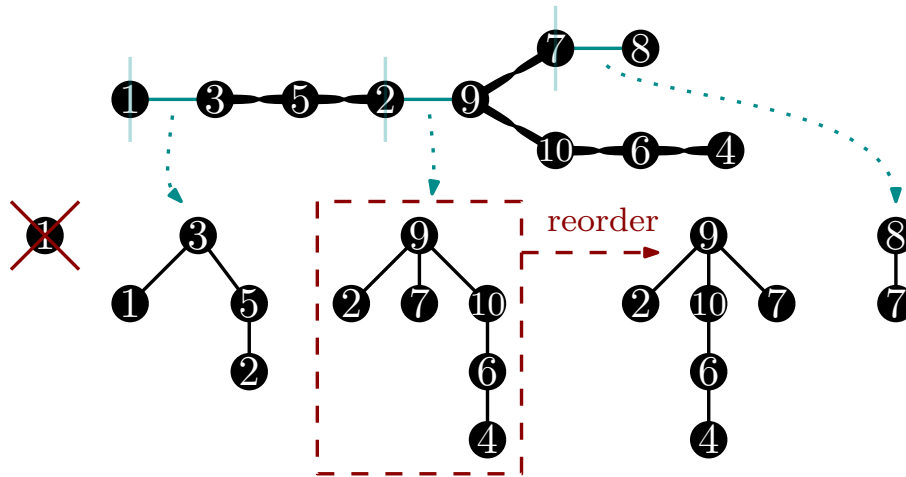


Figure 11: Example of a Reversed Step 3, given the Cayley tree of Figure 9. The ordering of the children has changed in only one tree in the forest, since the other trees were already rightly ordered.

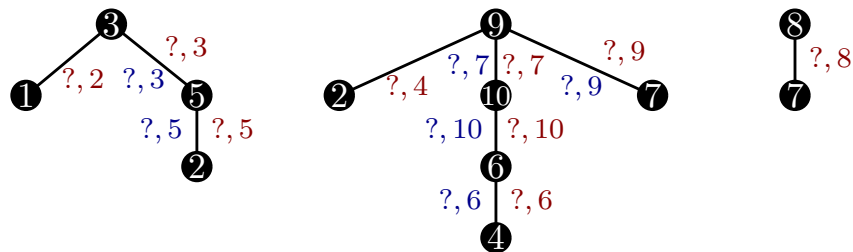


Figure 12: Reversed Step 2 with the example of the Cayley tree of Figure 9.

To do so, proceed to a Depth First Search of t where one favors the rightmost edge at each step, and name vertices $v_1, \dots, v_{|t|}$, and edges $e_1, \dots, e_{|t|-1}$ with respect to the visit order in the DFS (see again Figure 7).

Then, for k between 1 and $|t| - 2$, we add $(?, v_k)$ as the left *and* right labels of e_k . For $k = |t| - 1$, we just add a right label $(?, v_{|t|-1})$ (no left label) at $e_{|t|-1}$.

This step is shown at Figure 12 with the same example as earlier.

Reversed Step 1: fully identifying the 1,2-tree edges. We have now to fill in the question marks of the left and right labels to be consistent with the rules of Step 1 (see Table 1).

Let t be any tree of F . Let ℓ be the leftmost child of the root of t . As we noticed at the end of Reversed Step 3, ℓ is a leaf. Since the edge linking the root of t to ℓ has been visited last in the rightmost DFS, it has only a right label, which is of the form $(?, y)$. Replace this right label by (ℓ, y) .

edge	initial state	9 - 2	9 - 10
tree			
edge	10 - 6	6 - 4	9 - 7
tree			

Table 3: Reversed Step 1, edge by edge, given the second tree of Figure 12.

Then, iterate over all edges e of t via a Depth First Search, which this time favors the leftmost edges. We skip the first edge, which connects the root to ℓ (this edge has already been treated).

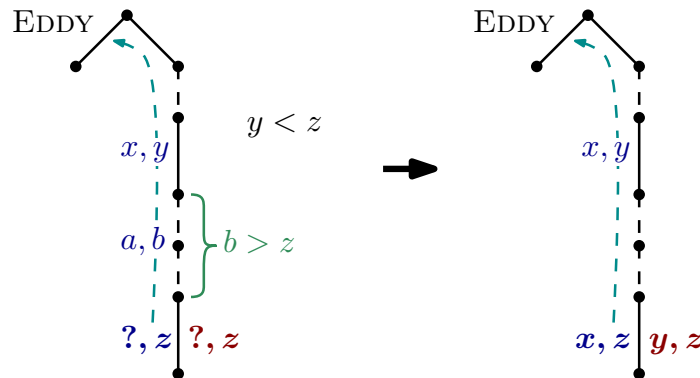


Figure 13: How to recover the vertices behind the question marks.

Let $(?, z)$ be the left label (or right label) of e . By induction, the question marks appearing in the left and right labels of every edge linking e to $\text{EDDY}(e)$ have been replaced. Among all of these edges, $\text{EDDY}(e)$ excluded, we pick the first one, starting from the bottom, which has a left label (x, y) such that $y < z$. If no such edge exists, then we set (x, y) as the right label of $\text{EDDY}(e)$. We then replace the left label of e by (x, z) and the right label of e by (y, z) . This process is schematized by Figure 13.

A full example is given by Table 3.

Proposition 24. *The above transformation describes the inverse of τ . In particular, τ is a bijection.*

Proof. Reversed Step 3 is the inverse of Step 3. It is not complicated to see that after Reversed Step 3, we can recover the original Cayley tree by merging the trees of the forest as in Step 3.

We also need to prove that the decomposition of Reversed Step 3 is unique, i.e. there is no other way to cut a Cayley tree that would form a forest in the image of Step 2.

First, by Lemma 14, the leftmost child of the root of each tree must be a leaf and carry the smallest label in its tree. It implies that when we attach the trees back together, the edges incident to the points of attachment are necessarily increasing, which is the case in Reversed Step 3.

Second, by Lemma 21, every edge of each tree must be a twist. This is checked here since during Reversed Step 3, we have cut all the increasing edges.

Finally, the ordering of the children is uniquely determined by the minima, as stated by Lemma 23.

Reversed Step 2 is the inverse of Step 2. In both Step 2 and Reversed Step 2, we match the edge end labels with the vertex labels by respecting the visit order of a rightmost DFS. During Step 2, we label the vertices in Step 2, while in Reversed Step 2, we label the edges.

Reversed Step 1 is the inverse of Step 1. The property given by Figure 13, namely "*If e is an edge with left label (x, z) and right label (y, z) , then (x, y) is the first label encountered along the path from e (left side) to $EDDY(e)$ (right side), which satisfies $y < z$.*", always holds during Step 1. Indeed, this is true when we insert e in the forest (see Rule 2 Table 1 for evidence). It remains true by induction since every edge we insert afterwards has a left label (a, b) where $b > z$.

Moreover, this property uniquely characterizes the numbers to be used in place of question marks. □

4.4 Express transformation

It is possible to express τ in a more straightforward manner, skipping the Depth First Search of Step 2 by incorporating the vertex labels directly in Step 1. The transformation is summed up in Table 4.

More precisely, given an increasing 1,2-tree G , we construct a plane tree \mathbb{T} where the edges carry left or/and right labels, like in Step 1 in Subsection 4.1, but on top of that, the vertices are labeled by a number between 1 and n .

Let us initialize \mathbb{T} as a tree with only one vertex labeled by 1. Like Step 1 of τ , we successively consider every vertex $\mathbf{v} \neq \mathbf{1}$ of G , starting from vertex $\mathbf{2}$ and ending to vertex \mathbf{n} .

- **Rule I.** If \mathbf{v} has only one neighbor \mathbf{x} such that $x < v$ (which means it was attached

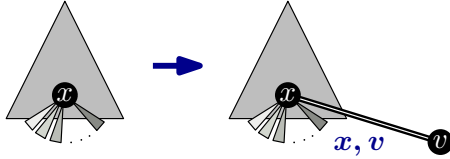
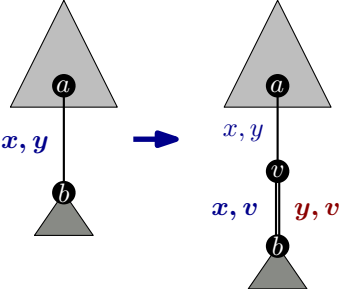
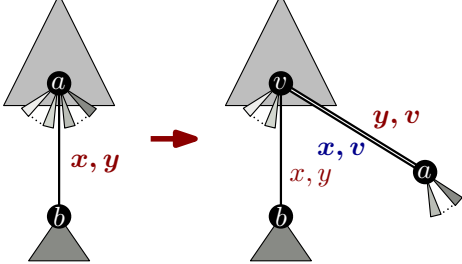
<p>Rule I. v is attached to a vertex x as a leaf</p>		
<p>Rule II. v is attached to x and y as a triangle</p>	<p>the label is on the left</p> 	<p>the label is on the right</p> 

Table 4: Three rules of the express transformation.

to x as a leaf during the building of G), then we add a new leaf v in \mathbb{T} as the rightmost child of vertex x . The edge linking x to v carries a left label (x, v) .

- **Rule II.** If v has two neighbors x and y such that $x < y < v$ (which means it was attached to edge $\{x, y\}$ as a triangle during the building of G), then we modify \mathbb{T} depending on whether the label (x, y) is on the left or right of the edge $e = ab$ that carries it.
 - **Rule II Left.** If (x, y) is on the left of e , then we add a new vertex v in the middle of e (see Table 4 bottom left). The edge av inherits the left label and the right label (if any) of e . We put a label (x, v) on the left of av , and a label (y, v) on the right of av .
 - **Rule II Right.** If (x, y) is on the right of e , then we change the label of a by v and we insert a new edge va in \mathbb{T} as the right sibling of vb , pushing below va every subtree that was incident to e and on the right of e (see Table 4 bottom right). We put a label (x, v) on the left of va , and a label (y, v) on the right of va .

The process is illustrated step by step by Table 5.

Note that this construction is reminiscent of Shor's decomposition of Cayley tree with respect to their twists (cf Figure 3). Rule II Left can be seen as the special case $a = 0$ of Figure 3 bottom, while Rule II Right represents the case $a > 0$.

Proposition 25. *If we remove the edge labels from \mathbb{T} and forget the planarity (i.e how children are ordered), then we recover $\tau(G)$.*

vertex v	1	2	3	4	5	6
tree						
vertex v	7	8	9	10		
tree						

Table 5: Construction of the tree through the express transformation from the increasing 1,2-tree of Figure 6

Proof ideas. Instead of a formal dragging on proof, the authors prefer to explain in rough words why the two transformations are identical. We introduce a new intermediary rule, named Rule 1', to justify why Rule 1 and Rule I induce the same Cayley tree. Rule 1' is a modification of Rule 1 from Table 1 where the new edge carries a left label instead of a right label (cf. Table 6).³

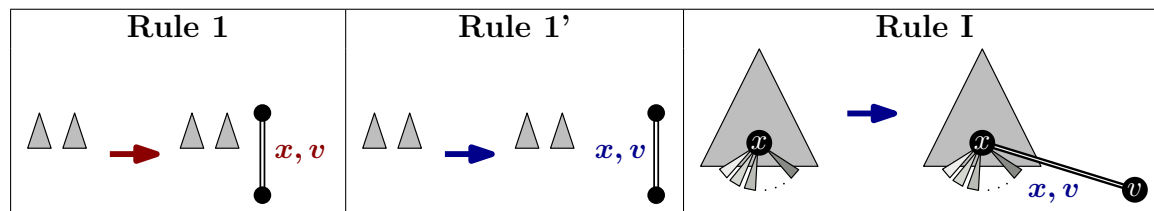


Table 6: Introduction of Rule 1'.

Claim 1. Rule 1 is equivalent to Rule 1'. Let us suppose that we modify Rule 1 in the original transformation τ so that the edge in the new tree carries a left label and no right label. Then, instead of trees such that the leftmost children incident to the root are leaves (see Lemma 14), we obtain trees where the root has only one child (the proof is similar as the one of Lemma 14). These two families of trees are in bijection: to recover the original trees, we have to topple the root edges to the right so that they become leaves at the far left of the new root, and change the left labels into right labels.

Claim 2. Rule I (where we attach a leaf to the same unique tree) is equivalent to Rule 1' (where we add a new one-edge tree to the previous forest). In comparison with τ , the express transformation does not treat the 2-connected components separately. In order to recover the trees of Step 1 of τ , we can cut \mathbb{T} at the root of the edges with no right label.

Note also that the sources⁴ of the edges with no right label keep the same vertex labels throughout the construction of \mathbb{T} . Indeed, we can prove that the edges with no right label are always the rightmost children of their parents, which ensures that we cannot apply Rule II Right to one of their right siblings, which would change the label of the source. Thus, if we cut \mathbb{T} at the root of the edges with no right label at any point during the construction, we can rebuilt τ by merging vertices with the same label, like in Step 3 of τ .

Claim 3. The vertex labeling of \mathbb{T} is consistent with a rightmost Depth First Search (Step 2 of τ). Cut \mathbb{T} at the root of each edge uv with no right label, and

³Even if it does not seem consistent, there are reasons why in τ we use right labels (Rule 1) and for the express transformation we use left labels (Rule I). On one hand, using right labels in τ makes the rightmost Depth First Search of Step 2 more natural (we need to visit the root edge last). On the other hand, if we had used right labels in the express transformation, then it would be possible that a vertex label will be shifted in a wrong place by using a Rule II Right on a root edge.

⁴A *source* of an edge uv is the vertex u .

start a rightmost DFS starting at \mathbf{v} . We claim that for each integer i , the label of the i -th visited vertex is equal to the end label of the i -th visited edge, as in Step 2 of τ .

This can be proved by induction and by thoroughly checking how the vertex labels evolve in Rule II. For example, in Rule II Right (Table 4 bottom right), v is the label of the newly inserted \mathbf{v} and the end label of the rightmost incident edge of \mathbf{v} , which agrees with the visit order of a rightmost DFS. In the same vein, we have to check that y , the end label of the edge \mathbf{vb} , corresponds to the label of the leftmost leaf descendant of \mathbf{a} (which is true by induction).

By combining all three claims, which together factor all the differences between τ and the construction of \mathbb{T} , we infer that the two constructions lead to the same Cayley tree. \square

5 Uniform random sampling in linear time

In this section, we describe two sampling algorithms which given an integer n , output uniformly at random:

- a Cayley tree with n vertices,
- an increasing 1,2-tree with n vertices.

The computation time is guaranteed in $O(n)$ for both algorithms, even in the worst-case scenario.

All the complexities in this section do not include the cost of a Random Number Generator. Indeed, we assume that drawing a random integer between 0 and n is in $O(1)$ and not in $O(\log(n))$.

5.1 Random Cayley trees

Most random samplers for Cayley trees found in the literature or on the Internet are based on the bijection between Cayley trees and Prüfer codes [24]. The idea is to draw a uniform Prüfer code, that is a sequence of $n - 2$ numbers between 0 and $n - 1$, and use the Prüfer transformation to get a uniform random tree.

The book by Nijenhuis and Wilf [23] already introduced this idea by presenting a random sampler running in $O(n^2)$. This complexity can be reduced to $O(n)$ using the algorithm described in [31].

We propose here a new elementary sampling algorithm running in $O(n)$, whose idea is directly inspired from the Aldous-Broder algorithm [1]. The Aldous-Broder algorithm is a random walk-based algorithm that generates a uniform random spanning tree of

any connected undirected graph G . The algorithm starts from an arbitrary vertex and proceeds by randomly traversing the edges of G until all vertices are visited. At each step, if an unvisited neighbor is encountered, the algorithm adds the corresponding edge to the spanning tree. The process continues until all vertices of G are visited, resulting in a uniformly random spanning tree.

In our context, G is the complete graph K_n with n vertices (loops included). If we naively run the Aldous-Broder algorithm on K_n , we obtain a random Cayley tree with n vertices, but the running time is in $O(n \log(n))$ in the average case: this is equivalent to the coupon collector's problem.

It is however possible to fasten the random walk by skipping the part where the random walk goes through vertices which are already visited. Indeed, whenever the walk goes from a vertex \mathbf{u} to an already visited vertex \mathbf{v} , then we can force the random walk to go from \mathbf{v} to an unvisited vertex, chosen at random among the unvisited vertices, without any bias on the spanning tree. This is justified by the following lemma.

Lemma 26. *Consider a portion of a random walk on K_n in the form $\mathbf{u} \rightarrow \mathbf{v}_1 \rightarrow \cdots \rightarrow \mathbf{v}_\ell \rightarrow \mathbf{w}$, where \mathbf{u} and \mathbf{w} are vertices visited for the first time, and $\mathbf{v}_1, \dots, \mathbf{v}_\ell$ are already visited vertices.*

Then, the random variable associated with \mathbf{v}_1 has the same probability distribution as the random variable associated with \mathbf{v}_ℓ .

Proof. The random walk has the same probability to do $\mathbf{u} \rightarrow \mathbf{v}_1 \rightarrow \cdots \rightarrow \mathbf{v}_\ell \rightarrow \mathbf{w}$ than $\mathbf{u} \rightarrow \mathbf{v}_\ell \rightarrow \cdots \rightarrow \mathbf{v}_1 \rightarrow \mathbf{w}$ by symmetry of the complete graph. This explains why \mathbf{v}_1 and \mathbf{v}_ℓ share the same distribution. \square

Using this observation, Algorithm 1 samples Cayley trees of given size according to the uniform distribution. It runs in $O(n)$, where n is the number of vertices, even in the worst-case scenario.

Note that this algorithm performs on average $\frac{3}{2}n$ random integer draws, whereas an algorithm based on Prüfer codes does only n samples (which is optimal).

5.2 Random increasing 1,2-trees

At this point, drawing an increasing 1,2-tree with n vertices uniformly at random is straightforward:

- we start from a random Cayley tree with n vertices output by Algorithm 1,
- we apply the bijection τ of Section 4 – or more precisely its reciprocal τ^{-1} in Subsection 4.3.

Algorithm 1 Uniform random sampler for Cayley trees of size n .

Input: n , a positive integer.

Output: Edges, an array listing the edges of a uniform random Cayley tree of size n whose vertex set is $\{1, \dots, n\}$.

```
Edges  $\leftarrow$  empty array;
Vertices  $\leftarrow [1, 2, \dots, n]$  array;           \ Arrays are indexed starting from 1
for nb_unvisited from  $n - 1$  to 1
  \ the subarray Vertices[nb_unvisited+1..n] lists the encountered vertices
  prev  $\leftarrow$  nb_unvisited+1                   \ index of the current position
  next  $\leftarrow$  random number between 1 and  $n$ ;
  if next > nb_unvisited then                     \ if the next vertex was already visited
    prev  $\leftarrow$  next;                           \ then the random walk moves forward and
    \ at the next step we force it to go to an unvisited vertex :
    next  $\leftarrow$  random number between 1 and nb_unvisited;
  end if
  add (Vertices[prev], Vertices[next]) to Edges;
  swap Vertices[next] and Vertices[nb_unvisited];
end for
return Edges;
```

However, we need to be careful with the implementation of certain steps to ensure that we remain effective. Reversed Step 3 can be performed in linear time with a recursive traversal of the Cayley Tree, remembering the minima of the subtrees. Sorting the children with respect to their minima can be done in $O(n)$ too, with a Bucket Sort for example. Reversed Step 2 is just a rightmost DFS, which also has a linear time complexity. Reversed Step 1 is more delicate, especially the part where we recover the vertices behind the question marks (see Figure 13). This problem is reminiscent of the All Left Nearest Smaller Neighbour problem [5] and can be solved in linear time by gathering all edges which share the same EDDY.

Figure 14 shows some random increasing 1,2-trees drawn uniformly at random.

6 Conclusion

By starting our study of increasing ($\leq k$)-trees with $k = 2$, we discovered that increasing 1,2-trees are equinumerous with Cayley trees. We have given three proofs of this result, each with its own advantages:

1. **Proof by recursive decomposition.** This proof, mainly based on the Cayley tree decomposition of Shor [28], is the simplest of the three.
2. **Proof by generating function.** We found an exact formula for the generating

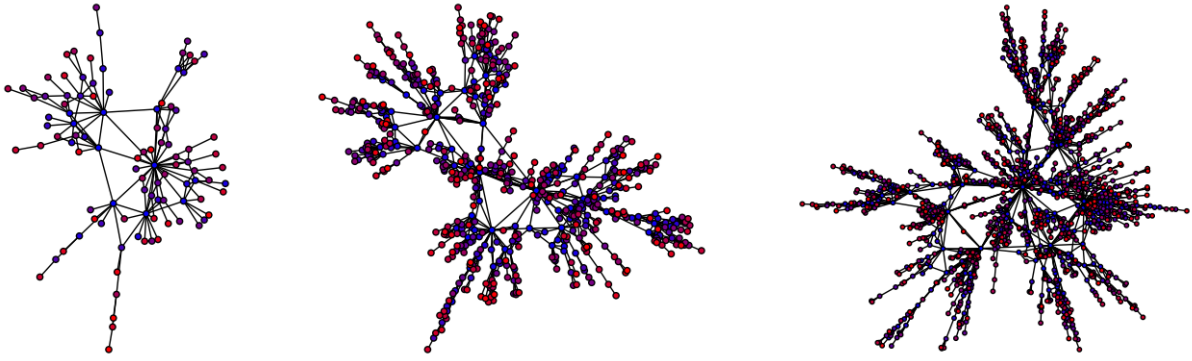


Figure 14: Random increasing 1,2-trees with sizes 100, 600, 1200, respectively. The redder the vertices, the larger the labels.

function of increasing 1,2-trees (see Theorem 10). The method, interesting in itself, involves solving a Partial Differential Equation using the method of characteristics, and should provide a promising approach for the study of $(\leq k)$ -trees, with $k \geq 3$. Besides, given how simple the generating function is, it would be intriguing to investigate a combinatorial interpretation (see Remark 11). Finally, it might be worthwhile to study the asymptotic distribution of the number of triangles in a random increasing 1,2-tree.

3. Proof by bijection. This explanation is the most satisfactory from a combinatorial point of view. It shows that the increasing edges of Cayley trees are preserved via the bijection, while the twists are somehow transformed into triangles. Moreover, we have used this bijection to design an efficient random generator for increasing 1,2-trees.

The investigation of increasing $(\leq k)$ -trees with $k \geq 3$ should be a natural follow-up of this work. In particular, it should be interesting to compare uniform random increasing $(\leq k)$ -trees with other distributions on $(\leq k)$ -trees: uniform random labeled $(\leq k)$ -trees [7], $(\leq k)$ -trees seen as a Pólya urn model⁵. Notably, how does the distribution impact the asymptotic shape of the $(\leq k)$ -tree? How do the numbers of cliques of given size change?

⁵Cliques of size x are represented by balls labeled by x . Start with an urn with one ball labeled by 1. At each step, draw uniformly a ball and put it back in the urn. If it was labeled x , add to the urn $\binom{x}{i-1}$ balls labeled i for each $i \in \{1, \dots, x+1\}$. (For example, if it was labeled 1, you need to add 1 ball labeled 1 and 1 ball labeled 2. If it was labeled 2, you add 1 ball labeled 1, 2 balls labeled 2, and 1 ball labeled 3.)

References

- [1] D. J. Aldous. The random walk construction of uniform spanning trees and uniform labelled trees. *SIAM J. Discrete Math.*, 3(4):450–465, 1990.
- [2] S. Arnborg, D. G. Corneil, and A. Proskurowski. Complexity of finding embeddings in a k -tree. *SIAM J. Algebraic Discrete Methods*, 8(2):277–284, 1987.
- [3] L. W. Beineke and R. E. Pippert. The number of labeled k -dimensional trees. *J. Combinatorial Theory*, 6:200–205, 1969.
- [4] E. A. Bender, L. B. Richmond, and N. C. Wormald. Almost all chordal graphs split. *J. Austral. Math. Soc. Ser. A*, 38(2):214–221, 1985.
- [5] O. Berkman, B. Schieber, and U. Vishkin. Optimal doubly logarithmic parallel algorithms based on finding all nearest smaller values. *J. Algorithms*, 14(3):344–370, 1993.
- [6] O. Bodini, M. Dien, A. Genitrini, and F. Peschanski. The ordered and colored products in analytic combinatorics: application to the quantitative study of synchronizations in concurrent processes. In *2017 Proceedings of the Fourteenth Workshop on Analytic Algorithmics and Combinatorics (ANALCO)*, pages 16–30. SIAM, Philadelphia, PA, 2017.
- [7] J. Castellví, M. Drmota, M. Noy, and C. Requilé. Chordal graphs with bounded tree-width. *Adv. in Appl. Math.*, 157:Paper No. 102700, 27, 2024.
- [8] J. Castellví and B. Stuffer. Limits of chordal graphs with bounded tree-width. *Random Structures Algorithms*, 66(1):Paper No. e21275, 15, 2025.
- [9] A. Cayley. *A theorem on trees*, volume 23. Cambridge University Press, 1889.
- [10] R. M. Corless, G. H. Gonnet, D. E. G. Hare, D. J. Jeffrey, and D. E. Knuth. On the Lambert W function. *Adv. Comput. Math.*, 5(4):329–359, 1996.
- [11] J. Dieudonné. *Éléments d’analyse. Tome IV: Chapitres XVIII à XX*, volume Fasc. XXXIV of *Cahiers Scientifiques [Scientific Reports]*. Gauthier-Villars Éditeur, 1971.
- [12] D. Dumont and A. Ramamonjisoa. Grammaire de Ramanujan et arbres de Cayley. volume 3, pages Research Paper 17, approx. 18. 1996. The Foata Festschrift.
- [13] T. Ekim, M. Shalom, and O. Şeker. The complexity of subtree intersection representation of chordal graphs and linear time chordal graph generation. *J. Comb. Optim.*, 41(3):710–735, 2021.
- [14] J. Enright and G. Kondrak. The application of chordal graphs to inferring phylogenetic trees of languages. In H. Wang and D. Yarowsky, editors, *Proceedings of 5th International Joint Conference on Natural Language Processing*, pages 545–552, Chiang Mai, Thailand, Nov. 2011. Asian Federation of Natural Language Processing.
- [15] L. C. Evans. *Partial differential equations*, volume 19 of *Graduate Studies in Mathematics*. American Mathematical Society, Providence, RI, second edition, 2010.
- [16] P. Flajolet and R. Sedgewick. *Analytic Combinatorics*. Cambridge University Press, 2009.

- [17] F. Harary and E. M. Palmer. On acyclic simplicial complexes. *Mathematika*, 15:115–122, 1968.
- [18] Ú. Hébert-Johnson, D. Lokshantov, and E. Vigoda. Counting and sampling labeled chordal graphs in polynomial time. In *31st annual European Symposium on Algorithms*, volume 274 of *LIPICs. Leibniz Int. Proc. Inform.*, pages Art. No. 58, 17. Schloss Dagstuhl. Leibniz-Zent. Inform., Wadern, 2023.
- [19] I. Hobæk Haff, K. Aas, A. Frigessi, and V. Lacal. Structure learning in Bayesian networks using regular vines. *Comput. Statist. Data Anal.*, 101:186–208, 2016.
- [20] L. Markenzon, O. Vernet, and L. Araujo. Two methods for the generation of chordal graphs. *Annals of Operations Research*, 157(1):47–60, January 2008.
- [21] O. J. Mengshoel. Understanding the scalability of Bayesian network inference using clique tree growth curves. *Artificial Intelligence*, 174(12-13):984–1006, 2010.
- [22] A. Mukhopadhyay and M. Z. Rahman. Algorithms for generating strongly chordal graphs. In *Transactions on computational science XXXVIII*, volume 12620 of *Lecture Notes in Comput. Sci.*, pages 54–75. Springer, Berlin, 2021.
- [23] A. Nijenhuis and H. S. Wilf. *Combinatorial algorithms*. Computer Science and Applied Mathematics. Academic Press, Inc. [Harcourt Brace Jovanovich, Publishers], New York-London, second edition, 1978. For computers and calculators.
- [24] H. Prüfer. Neuer Beweis eines Satzes über Permutationen. *Archiv der Mathematischen Physik*, 27:742–744, 1918.
- [25] S. Ramanujan. *Notebooks. Vols. 1, 2*. Tata Institute of Fundamental Research, Bombay, 1957.
- [26] D. J. Rose. A graph-theoretic study of the numerical solution of sparse positive definite systems of linear equations. In *Graph theory and computing*, pages 183–217. Academic Press, New York-London, 1972.
- [27] O. Şeker, P. Heggernes, T. Ekim, and Z. C. Taşkın. Generation of random chordal graphs using subtrees of a tree. *RAIRO Oper. Res.*, 56(2):565–582, 2022.
- [28] P. Shor. A new proof of Cayley’s formula for counting labeled trees. *J. Combin. Theory Ser. A*, 71(1):154–158, 1995.
- [29] P. Shor, O. G. Ruehr, and A. Meir. A combinatorial identity (Peter Shor). *SIAM Review*, 21(2):258–260, 1979.
- [30] W. Sun and I. Bezáková. Sampling random chordal graphs by MCMC (student abstract). *Proceedings of the AAAI Conference on Artificial Intelligence*, 34:13929–13930, 04 2020.
- [31] X. Wang, L. Wang, Y. Wu, et al. An optimal algorithm for Prüfer codes. *J. Softw. Eng. Appl.*, 2(2):111–115, 2009.
- [32] N. C. Wormald. Counting labelled chordal graphs. *Graphs Combin.*, 1(2):193–200, 1985.