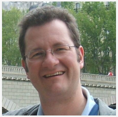
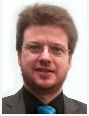
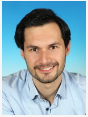
Enumeration of Compacted Binary Trees with Bounded Right Height

AofA 06/2019

Antoine Genitrini, Bernhard Gittenberger, Manuel Kauers, and Michael Wallner

Erwin Schrödinger-Fellow (Austrian Science Fund (FWF): J 4162)

Laboratoire Bordelais de Recherche en Informatique, Université de Bordeaux, France

June 27$^{\text{th}}$, 2019

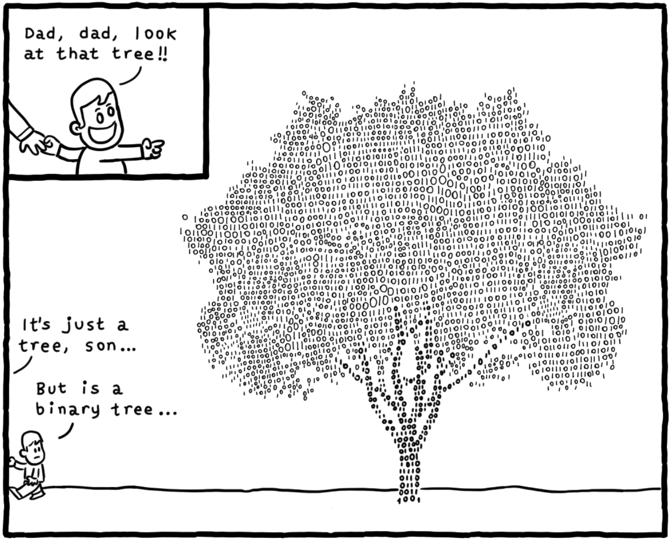
Based on the paper:
Asymptotic Enumeration of Compacted Binary Trees of Bounded Right Height,
to appear in Journal of Combinatorial Theory, Series A.
ArXiv:1703.10031

Creating a compacted binary tree

But first, what is a binary tree?

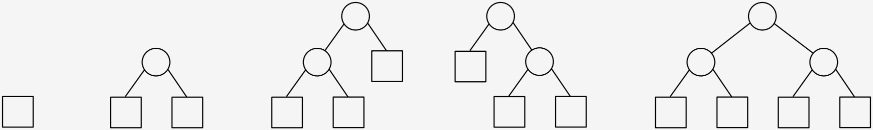
But first, what is a binary tree?

But first, what is a binary tree?

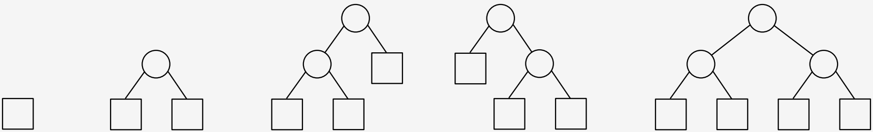


Daniel Stori {turnoff.us}

But first, what is a binary tree?

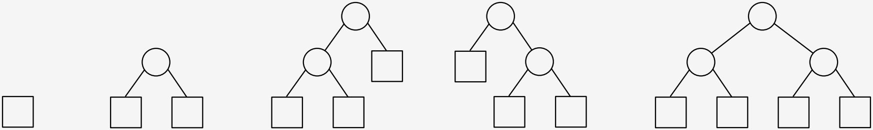


But first, what is a binary tree?



- *Internal node*: Node of out-degree 2 (circle)
- *Leave*: Node of out-degree 0 (square)
- *Root*: Distinguished node (top node)
- Order of children important

But first, what is a binary tree?



- *Internal node*: Node of out-degree 2 (circle)
- *Leave*: Node of out-degree 0 (square)
- *Root*: Distinguished node (top node)
- Order of children important

A recursive construction

- A binary tree is either a leaf,
- or it consists of a (root) node and a left and right binary tree.

Motivation – Efficiently store redundant information

Example

Consider the labeled tree necessary to store the arithmetic expression

```
(* (- (* x x) (* y y)) (+ (* x x) (* y y)))
```

which represents $(x^2 - y^2)(x^2 + y^2)$.

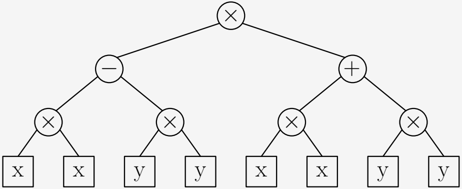
Motivation – Efficiently store redundant information

Example

Consider the labeled tree necessary to store the arithmetic expression

```
(* (- (* x x) (* y y)) (+ (* x x) (* y y)))
```

which represents $(x^2 - y^2)(x^2 + y^2)$.



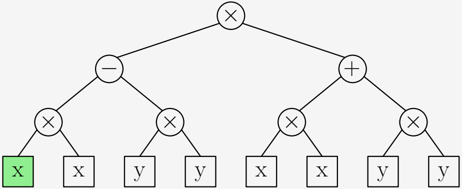
Motivation – Efficiently store redundant information

Example

Consider the labeled tree necessary to store the arithmetic expression

```
(* (- (* x x) (* y y)) (+ (* x x) (* y y)))
```

which represents $(x^2 - y^2)(x^2 + y^2)$.



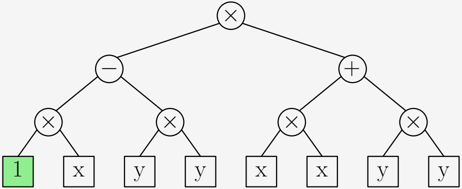
Motivation – Efficiently store redundant information

Example

Consider the labeled tree necessary to store the arithmetic expression

```
(* (- (* x x) (* y y)) (+ (* x x) (* y y)))
```

which represents $(x^2 - y^2)(x^2 + y^2)$.



$(1, (x, 0, 0))$

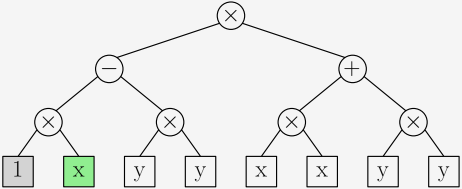
Motivation – Efficiently store redundant information

Example

Consider the labeled tree necessary to store the arithmetic expression

```
(* (- (* x x) (* y y)) (+ (* x x) (* y y)))
```

which represents $(x^2 - y^2)(x^2 + y^2)$.



$(1, (x, 0, 0))$

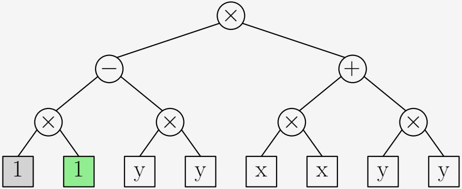
Motivation – Efficiently store redundant information

Example

Consider the labeled tree necessary to store the arithmetic expression

```
(* (- (* x x) (* y y)) (+ (* x x) (* y y)))
```

which represents $(x^2 - y^2)(x^2 + y^2)$.



$(1, (x, 0, 0))$

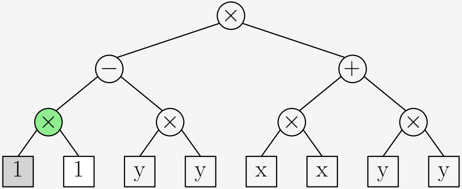
Motivation – Efficiently store redundant information

Example

Consider the labeled tree necessary to store the arithmetic expression

```
(* (- (* x x) (* y y)) (+ (* x x) (* y y)))
```

which represents $(x^2 - y^2)(x^2 + y^2)$.



$(1, (x, 0, 0))$

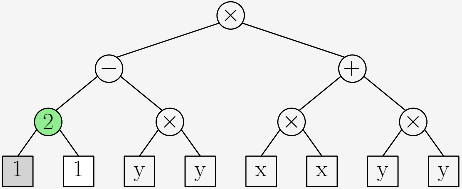
Motivation – Efficiently store redundant information

Example

Consider the labeled tree necessary to store the arithmetic expression

```
(* (- (* x x) (* y y)) (+ (* x x) (* y y)))
```

which represents $(x^2 - y^2)(x^2 + y^2)$.



$(1, (x, 0, 0)), \quad (2, (\times, 1, 1))$

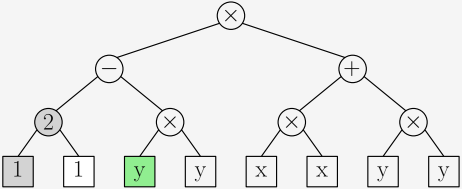
Motivation – Efficiently store redundant information

Example

Consider the labeled tree necessary to store the arithmetic expression

```
(* (- (* x x) (* y y)) (+ (* x x) (* y y)))
```

which represents $(x^2 - y^2)(x^2 + y^2)$.



$(1, (x, 0, 0)), \quad (2, (\times, 1, 1))$

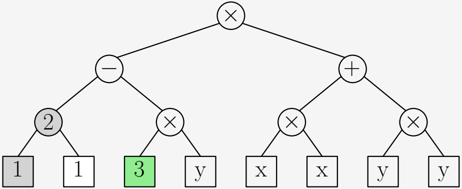
Motivation – Efficiently store redundant information

Example

Consider the labeled tree necessary to store the arithmetic expression

```
(* (- (* x x) (* y y)) (+ (* x x) (* y y)))
```

which represents $(x^2 - y^2)(x^2 + y^2)$.



$(1, (x, 0, 0)), \quad (2, (\times, 1, 1)), \quad (3, (y, 0, 0))$

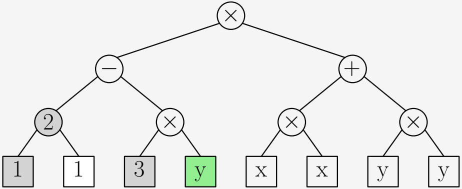
Motivation – Efficiently store redundant information

Example

Consider the labeled tree necessary to store the arithmetic expression

```
(* (- (* x x) (* y y)) (+ (* x x) (* y y)))
```

which represents $(x^2 - y^2)(x^2 + y^2)$.



$(1, (x, 0, 0)), \quad (2, (\times, 1, 1)), \quad (3, (y, 0, 0))$

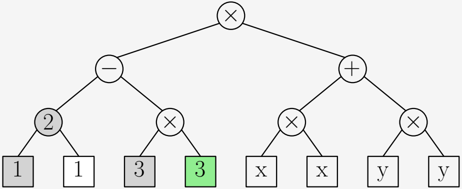
Motivation – Efficiently store redundant information

Example

Consider the labeled tree necessary to store the arithmetic expression

```
(* (- (* x x) (* y y)) (+ (* x x) (* y y)))
```

which represents $(x^2 - y^2)(x^2 + y^2)$.



$$(1, (x, 0, 0)), \quad (2, (\times, 1, 1)), \quad (3, (y, 0, 0))$$

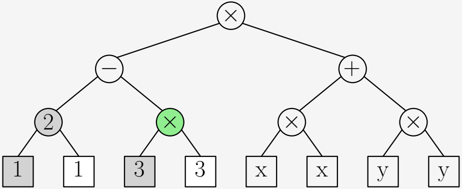
Motivation – Efficiently store redundant information

Example

Consider the labeled tree necessary to store the arithmetic expression

```
(* (- (* x x) (* y y)) (+ (* x x) (* y y)))
```

which represents $(x^2 - y^2)(x^2 + y^2)$.



$(1,(x,0,0)),\quad (2,(\times,1,1)),\quad (3,(y,0,0))$

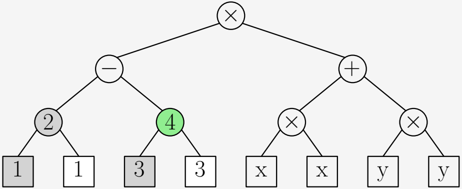
Motivation – Efficiently store redundant information

Example

Consider the labeled tree necessary to store the arithmetic expression

```
(* (- (* x x) (* y y)) (+ (* x x) (* y y)))
```

which represents $(x^2 - y^2)(x^2 + y^2)$.



$(1, (x, 0, 0)), \quad (2, (\times, 1, 1)), \quad (3, (y, 0, 0)), \quad (4, (\times, 3, 3))$

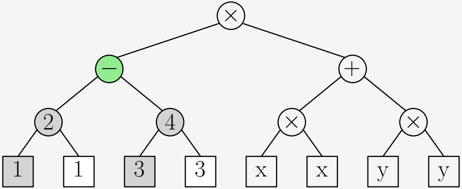
Motivation – Efficiently store redundant information

Example

Consider the labeled tree necessary to store the arithmetic expression

```
(* (- (* x x) (* y y)) (+ (* x x) (* y y)))
```

which represents $(x^2 - y^2)(x^2 + y^2)$.



$(1, (x, 0, 0)), \quad (2, (\times, 1, 1)), \quad (3, (y, 0, 0)), \quad (4, (\times, 3, 3))$

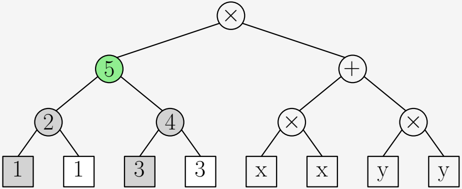
Motivation – Efficiently store redundant information

Example

Consider the labeled tree necessary to store the arithmetic expression

```
(* (- (* x x) (* y y)) (+ (* x x) (* y y)))
```

which represents $(x^2 - y^2)(x^2 + y^2)$.



$(1, (x, 0, 0)), \quad (2, (\times, 1, 1)), \quad (3, (y, 0, 0)), \quad (4, (\times, 3, 3)), \quad (5, (-, 2, 4))$

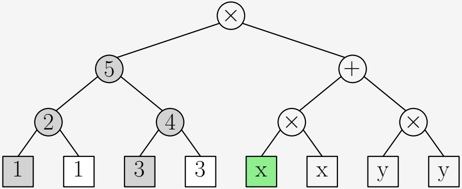
Motivation – Efficiently store redundant information

Example

Consider the labeled tree necessary to store the arithmetic expression

```
(* (- (* x x) (* y y)) (+ (* x x) (* y y)))
```

which represents $(x^2 - y^2)(x^2 + y^2)$.



$(1,(x,0,0)), \quad (2,(\times,1,1)), \quad (3,(y,0,0)), \quad (4,(\times,3,3)), \quad (5,(-,2,4))$

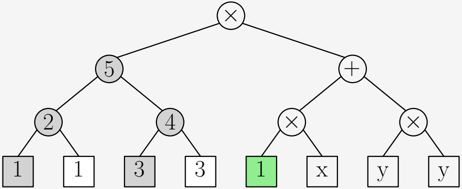
Motivation – Efficiently store redundant information

Example

Consider the labeled tree necessary to store the arithmetic expression

```
(* (- (* x x) (* y y)) (+ (* x x) (* y y)))
```

which represents $(x^2 - y^2)(x^2 + y^2)$.



$(1, (x, 0, 0))$, $(2, (\times, 1, 1))$, $(3, (y, 0, 0))$, $(4, (\times, 3, 3))$, $(5, (-, 2, 4))$

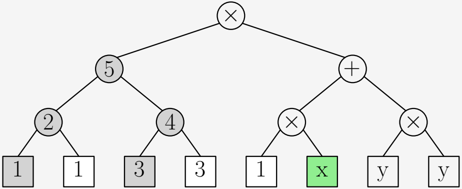
Motivation – Efficiently store redundant information

Example

Consider the labeled tree necessary to store the arithmetic expression

```
(* (- (* x x) (* y y)) (+ (* x x) (* y y)))
```

which represents $(x^2 - y^2)(x^2 + y^2)$.



$(1, (x, 0, 0))$, $(2, (\times, 1, 1))$, $(3, (y, 0, 0))$, $(4, (\times, 3, 3))$, $(5, (-, 2, 4))$

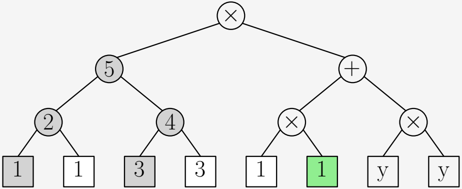
Motivation – Efficiently store redundant information

Example

Consider the labeled tree necessary to store the arithmetic expression

```
(* (- (* x x) (* y y)) (+ (* x x) (* y y)))
```

which represents $(x^2 - y^2)(x^2 + y^2)$.



$(1,(x,0,0)),\quad (2,(\times,1,1)),\quad (3,(y,0,0)),\quad (4,(\times,3,3)),\quad (5,(-,2,4))$

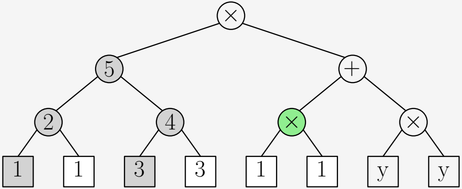
Motivation – Efficiently store redundant information

Example

Consider the labeled tree necessary to store the arithmetic expression

```
(* (- (* x x) (* y y)) (+ (* x x) (* y y)))
```

which represents $(x^2 - y^2)(x^2 + y^2)$.



$(1, (x, 0, 0))$, $(2, (\times, 1, 1))$, $(3, (y, 0, 0))$, $(4, (\times, 3, 3))$, $(5, (-, 2, 4))$

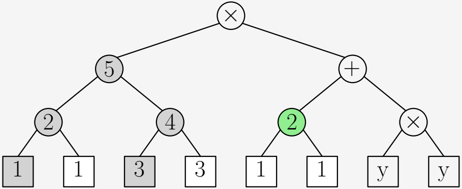
Motivation – Efficiently store redundant information

Example

Consider the labeled tree necessary to store the arithmetic expression

```
(* (- (* x x) (* y y)) (+ (* x x) (* y y)))
```

which represents $(x^2 - y^2)(x^2 + y^2)$.



$(1, (x, 0, 0))$, $(2, (\times, 1, 1))$, $(3, (y, 0, 0))$, $(4, (\times, 3, 3))$, $(5, (-, 2, 4))$

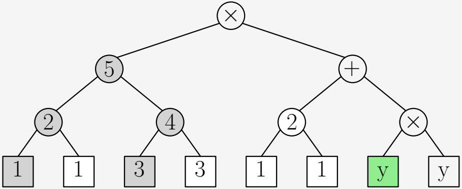
Motivation – Efficiently store redundant information

Example

Consider the labeled tree necessary to store the arithmetic expression

```
(* (- (* x x) (* y y)) (+ (* x x) (* y y)))
```

which represents $(x^2 - y^2)(x^2 + y^2)$.



$(1, (x, 0, 0))$, $(2, (\times, 1, 1))$, $(3, (y, 0, 0))$, $(4, (\times, 3, 3))$, $(5, (-, 2, 4))$

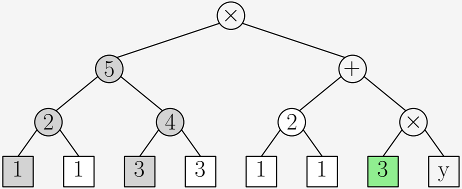
Motivation – Efficiently store redundant information

Example

Consider the labeled tree necessary to store the arithmetic expression

```
(* (- (* x x) (* y y)) (+ (* x x) (* y y)))
```

which represents $(x^2 - y^2)(x^2 + y^2)$.



$(1,(x,0,0))$, $(2,(\times,1,1))$, $(3,(y,0,0))$, $(4,(\times,3,3))$, $(5,(-,2,4))$

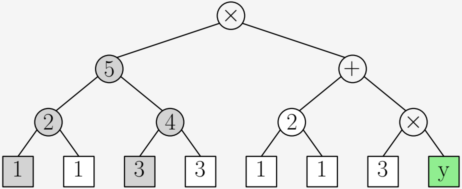
Motivation – Efficiently store redundant information

Example

Consider the labeled tree necessary to store the arithmetic expression

```
(* (- (* x x) (* y y)) (+ (* x x) (* y y)))
```

which represents $(x^2 - y^2)(x^2 + y^2)$.



$(1, (x, 0, 0))$, $(2, (\times, 1, 1))$, $(3, (y, 0, 0))$, $(4, (\times, 3, 3))$, $(5, (-, 2, 4))$

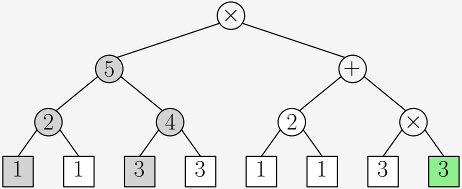
Motivation – Efficiently store redundant information

Example

Consider the labeled tree necessary to store the arithmetic expression

```
(* (- (* x x) (* y y)) (+ (* x x) (* y y)))
```

which represents $(x^2 - y^2)(x^2 + y^2)$.



$(1, (x, 0, 0))$, $(2, (\times, 1, 1))$, $(3, (y, 0, 0))$, $(4, (\times, 3, 3))$, $(5, (-, 2, 4))$

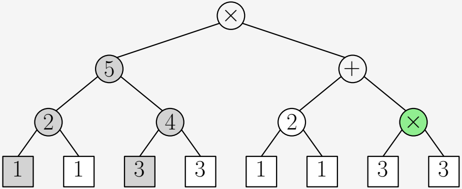
Motivation – Efficiently store redundant information

Example

Consider the labeled tree necessary to store the arithmetic expression

```
(* (- (* x x) (* y y)) (+ (* x x) (* y y)))
```

which represents $(x^2 - y^2)(x^2 + y^2)$.



$(1, (x, 0, 0)), \quad (2, (\times, 1, 1)), \quad (3, (y, 0, 0)), \quad (4, (\times, 3, 3)), \quad (5, (-, 2, 4))$

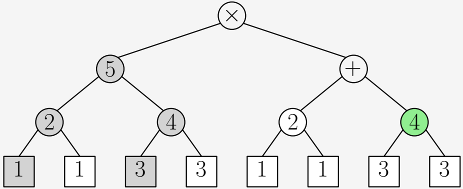
Motivation – Efficiently store redundant information

Example

Consider the labeled tree necessary to store the arithmetic expression

```
(* (- (* x x) (* y y)) (+ (* x x) (* y y)))
```

which represents $(x^2 - y^2)(x^2 + y^2)$.



$(1, (x, 0, 0)), \quad (2, (\times, 1, 1)), \quad (3, (y, 0, 0)), \quad (4, (\times, 3, 3)), \quad (5, (-, 2, 4))$

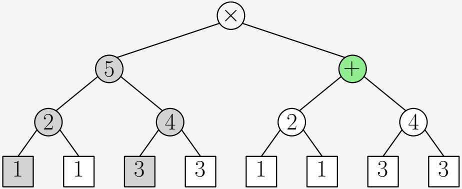
Motivation – Efficiently store redundant information

Example

Consider the labeled tree necessary to store the arithmetic expression

```
(* (- (* x x) (* y y)) (+ (* x x) (* y y)))
```

which represents $(x^2 - y^2)(x^2 + y^2)$.



$$(1,(x,0,0)), \quad (2,(\times,1,1)), \quad (3,(y,0,0)), \quad (4,(\times,3,3)), \quad (5,(-,2,4))$$

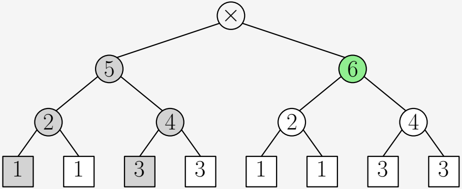
Motivation – Efficiently store redundant information

Example

Consider the labeled tree necessary to store the arithmetic expression

```
(* (- (* x x) (* y y)) (+ (* x x) (* y y)))
```

which represents $(x^2 - y^2)(x^2 + y^2)$.



$(1, (x, 0, 0))$, $(2, (\times, 1, 1))$, $(3, (y, 0, 0))$, $(4, (\times, 3, 3))$, $(5, (-, 2, 4))$, $(6, (+, 2, 4))$

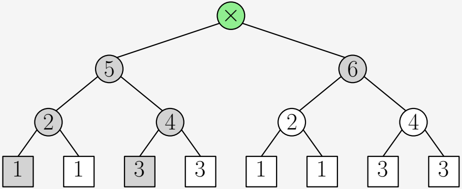
Motivation – Efficiently store redundant information

Example

Consider the labeled tree necessary to store the arithmetic expression

```
(* (- (* x x) (* y y)) (+ (* x x) (* y y)))
```

which represents $(x^2 - y^2)(x^2 + y^2)$.



$(1, (x, 0, 0))$, $(2, (\times, 1, 1))$, $(3, (y, 0, 0))$, $(4, (\times, 3, 3))$, $(5, (-, 2, 4))$, $(6, (+, 2, 4))$

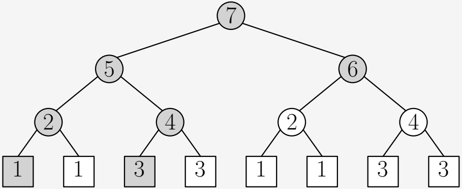
Motivation – Efficiently store redundant information

Example

Consider the labeled tree necessary to store the arithmetic expression

```
(* (- (* x x) (* y y)) (+ (* x x) (* y y)))
```

which represents $(x^2 - y^2)(x^2 + y^2)$.



$(1,(x,0,0))$, $(2,(\times,1,1))$, $(3,(y,0,0))$, $(4,(\times,3,3))$, $(5,(-,2,4))$, $(6,(+,2,4))$, $(7,(\times,5,6))$

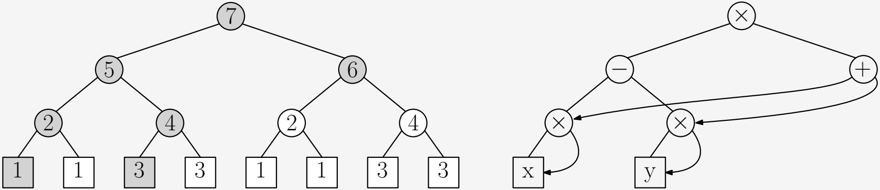
Motivation – Efficiently store redundant information

Example

Consider the labeled tree necessary to store the arithmetic expression

```
(* (- (* x x) (* y y)) (+ (* x x) (* y y)))
```

which represents $(x^2 - y^2)(x^2 + y^2)$.



$(1, (x, 0, 0))$, $(2, (\times, 1, 1))$, $(3, (y, 0, 0))$, $(4, (\times, 3, 3))$, $(5, (-, 2, 4))$, $(6, (+, 2, 4))$, $(7, (\times, 5, 6))$

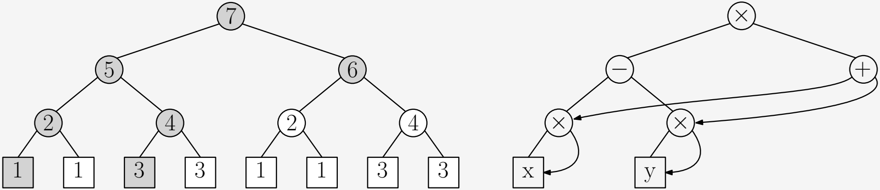
Motivation – Efficiently store redundant information

Example

Consider the labeled tree necessary to store the arithmetic expression

```
(* (- (* x x) (* y y)) (+ (* x x) (* y y)))
```

which represents $(x^2 - y^2)(x^2 + y^2)$.



$(1,(x,0,0))$, $(2,(\times,1,1))$, $(3,(y,0,0))$, $(4,(\times,3,3))$, $(5,(-,2,4))$, $(6,(+,2,4))$, $(7,(\times,5,6))$

Definition

Compacted tree is the directed acyclic graph computed by this procedure.

Compacted trees

- Important property: **Subtrees are unique**
- Efficient algorithm to compute compacted tree: expected time $\mathcal{O}(n)$
- Analyzed by [Flajolet, Sipala, Steyaert 1990] (Flajolet Collected Works, Volume 4): A tree of size n has a compacted form of expected size that is asymptotically equal to

$$C \frac{n}{\sqrt{\log n}},$$

  where C is explicit related to the type of trees and the statistical model.
- Applications: **XML-Compression** [Bousquet-Mélou, Lohrey, Maneth, Noeth 2015], **Compilers** [Aho, Sethi, Ullman 1986], **LISP** [Goto 1974], **Data storage** [Meinel, Theobald 1998], [Knuth 1968], etc.
- Restrict to unlabeled binary trees

Compacted trees

- Important property: **Subtrees are unique**
- Efficient algorithm to compute compacted tree: expected time $\mathcal{O}(n)$
- Analyzed by [Flajolet, Sipala, Steyaert 1990] (Flajolet Collected Works, Volume 4): A tree of size n has a compacted form of expected size that is asymptotically equal to

$$C \frac{n}{\sqrt{\log n}},$$

where C is explicit related to the type of trees and the statistical model.
- Applications: **XML-Compression** [Bousquet-Mélou, Lohrey, Maneth, Noeth 2015], **Compilers** [Aho, Sethi, Ullman 1986], **LISP** [Goto 1974], **Data storage** [Meinel, Theobald 1998], [Knuth 1968], etc.
- Restrict to unlabeled binary trees

Compacted trees

- Important property: **Subtrees are unique**
- Efficient algorithm to compute compacted tree: expected time $\mathcal{O}(n)$
- Analyzed by [Flajolet, Sipala, Steyaert 1990] (Flajolet Collected Works, Volume 4): A tree of size n has a compacted form of expected size that is asymptotically equal to

$$C \frac{n}{\sqrt{\log n}},$$

  where C is explicit related to the type of trees and the statistical model.
- Applications: **XML-Compression** [Bousquet-Mélou, Lohrey, Maneth, Noeth 2015], **Compilers** [Aho, Sethi, Ullman 1986], **LISP** [Goto 1974], **Data storage** [Meinel, Theobald 1998], [Knuth 1968], etc.
- Restrict to unlabeled binary trees

Compacted trees

- Important property: **Subtrees are unique**
- Efficient algorithm to compute compacted tree: expected time $\mathcal{O}(n)$
- Analyzed by [Flajolet, Sipala, Steyaert 1990] (Flajolet Collected Works, Volume 4): A tree of size n has a compacted form of expected size that is asymptotically equal to

$$C \frac{n}{\sqrt{\log n}},$$

  where C is explicit related to the type of trees and the statistical model.
- Applications: **XML-Compression** [Bousquet-Mélou, Lohrey, Maneth, Noeth 2015], **Compilers** [Aho, Sethi, Ullman 1986], **LISP** [Goto 1974], **Data storage** [Meinel, Theobald 1998], [Knuth 1968], etc.
- Restrict to unlabeled binary trees

Compacted trees

- Important property: **Subtrees are unique**
- Efficient algorithm to compute compacted tree: expected time $\mathcal{O}(n)$
- Analyzed by [Flajolet, Sipala, Steyaert 1990] (Flajolet Collected Works, Volume 4): A tree of size n has a compacted form of expected size that is asymptotically equal to

$$C \frac{n}{\sqrt{\log n}},$$

  where C is explicit related to the type of trees and the statistical model.
- Applications: **XML-Compression** [Bousquet-Mélou, Lohrey, Maneth, Noeth 2015], **Compilers** [Aho, Sethi, Ullman 1986], **LISP** [Goto 1974], **Data storage** [Meinel, Theobald 1998], [Knuth 1968], etc.
- Restrict to unlabeled binary trees

Compacted trees

- Important property: **Subtrees are unique**
- Efficient algorithm to compute compacted tree: expected time $\mathcal{O}(n)$
- Analyzed by [Flajolet, Sipala, Steyaert 1990] (Flajolet Collected Works, Volume 4): A tree of size n has a compacted form of expected size that is asymptotically equal to
$$C \frac{n}{\sqrt{\log n}},$$
where C is explicit related to the type of trees and the statistical model.
- Applications: **XML-Compression** [Bousquet-Mélou, Lohrey, Maneth, Noeth 2015], **Compilers** [Aho, Sethi, Ullman 1986], **LISP** [Goto 1974], **Data storage** [Meinel, Theobald 1998], [Knuth 1968], etc.
- Restrict to unlabeled binary trees

Reverse question

How many compacted trees of (compacted) size n exist?

Compacted (unlabeled binary) trees

- **Size:** number of internal nodes
- Number of compacted trees of size n: c_n

Compacted (unlabeled binary) trees

- **Size:** number of internal nodes
- Number of compacted trees of size n: c_n

Compacted (unlabeled binary) trees

- **Size:** number of internal nodes
- Number of compacted trees of size n: c_n

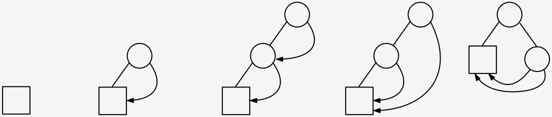


Figure: All compacted binary trees of size $n = 0, 1, 2$.

Compacted (unlabeled binary) trees

- **Size:** number of internal nodes
- Number of compacted trees of size n: c_n

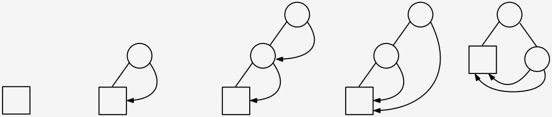


Figure: All compacted binary trees of size $n = 0, 1, 2$.

Simple bounds

size	$n=0$	$n=1$	$n=2$	$n=3$	$n=4$	$n=5$	$n=6$
c_n	1	1	3	15	111	1119	14487

$$n! \leq c_n \leq \frac{1}{n+1}\binom{2n}{n} \cdot n!$$

Compacted (unlabeled binary) trees

- **Size:** number of internal nodes
- Number of compacted trees of size n: c_n

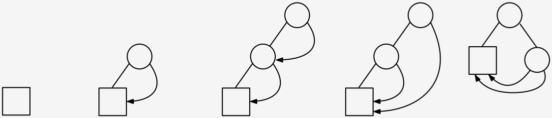


Figure: All compacted binary trees of size $n = 0, 1, 2$.

Simple bounds

size	$n=0$	$n=1$	$n=2$	$n=3$	$n=4$	$n=5$	$n=6$
c_n	1	1	3	15	111	1119	14487

$$n! \leq c_n \leq \frac{1}{n+1}\binom{2n}{n} \cdot n!$$

Lower bound:

Compacted (unlabeled binary) trees

- **Size:** number of internal nodes
- Number of compacted trees of size n: c_n

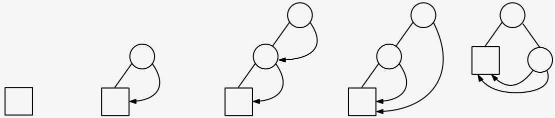


Figure: All compacted binary trees of size $n = 0, 1, 2$.

Simple bounds

size	$n=0$	$n=1$	$n=2$	$n=3$	$n=4$	$n=5$	$n=6$
c_n	1	1	3	15	111	1119	14487

$$n! \leq c_n \leq \frac{1}{n+1}\binom{2n}{n} \cdot n!$$

Lower bound:

Building a compacted binary tree

Idea

Every compacted tree can be build from a binary tree by adding pointers.

Building a compacted binary tree

Idea

Every compacted tree can be build from a binary tree by adding pointers.

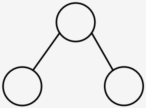
- Pointers may only point to previously seen parts in **post-order**
- Pointers are not allowed to violate uniqueness

Building a compacted binary tree

Idea

Every compacted tree can be build from a binary tree by adding pointers.

- Pointers may only point to previously seen parts in **post-order**
- Pointers are not allowed to violate uniqueness

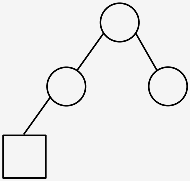


Building a compacted binary tree

Idea

Every compacted tree can be build from a binary tree by adding pointers.

- Pointers may only point to previously seen parts in **post-order**
- Pointers are not allowed to violate uniqueness

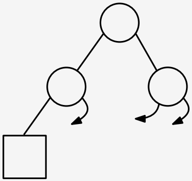


Building a compacted binary tree

Idea

Every compacted tree can be build from a binary tree by adding pointers.

- Pointers may only point to previously seen parts in **post-order**
- Pointers are not allowed to violate uniqueness

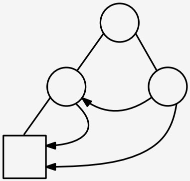


Building a compacted binary tree

Idea

Every compacted tree can be build from a binary tree by adding pointers.

- Pointers may only point to previously seen parts in **post-order**
- Pointers are not allowed to violate uniqueness

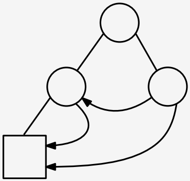


Building a compacted binary tree

Idea

Every compacted tree can be build from a binary tree by adding pointers.

- Pointers may only point to previously seen parts in **post-order**
- Pointers are not allowed to violate uniqueness



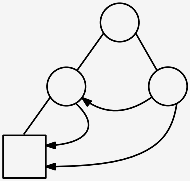
Valid compacted tree

Building a compacted binary tree

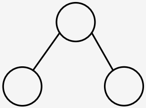
Idea

Every compacted tree can be build from a binary tree by adding pointers.

- Pointers may only point to previously seen parts in **post-order**
- Pointers are not allowed to violate uniqueness



Valid compacted tree

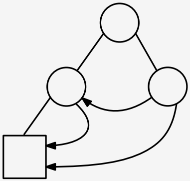


Building a compacted binary tree

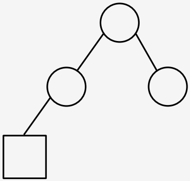
Idea

Every compacted tree can be build from a binary tree by adding pointers.

- Pointers may only point to previously seen parts in **post-order**
- Pointers are not allowed to violate uniqueness



Valid compacted tree

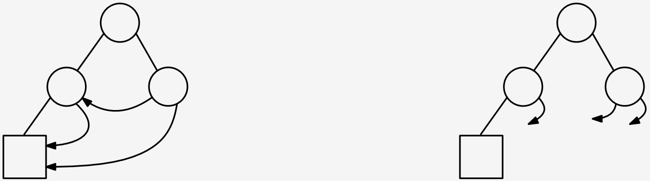


Building a compacted binary tree

Idea

Every compacted tree can be build from a binary tree by adding pointers.

- Pointers may only point to previously seen parts in **post-order**
- Pointers are not allowed to violate uniqueness



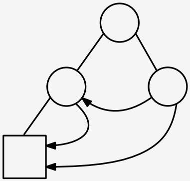
Valid compacted tree

Building a compacted binary tree

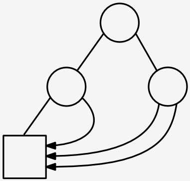
Idea

Every compacted tree can be build from a binary tree by adding pointers.

- Pointers may only point to previously seen parts in **post-order**
- Pointers are not allowed to violate uniqueness



Valid compacted tree

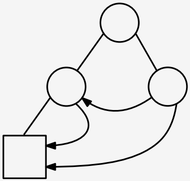


Building a compacted binary tree

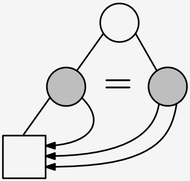
Idea

Every compacted tree can be build from a binary tree by adding pointers.

- Pointers may only point to previously seen parts in **post-order**
- Pointers are not allowed to violate uniqueness



Valid compacted tree

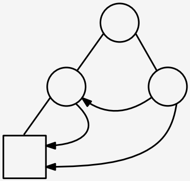


Building a compacted binary tree

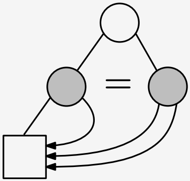
Idea

Every compacted tree can be build from a binary tree by adding pointers.

- Pointers may only point to previously seen parts in **post-order**
- Pointers are not allowed to violate uniqueness



Valid compacted tree



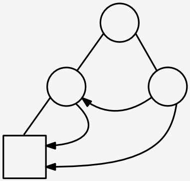
Invalid compacted tree

Building a compacted binary tree

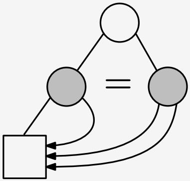
Idea

Every compacted tree can be build from a binary tree by adding pointers.

- Pointers may only point to previously seen parts in **post-order**
- Pointers are not allowed to violate uniqueness



Valid compacted tree



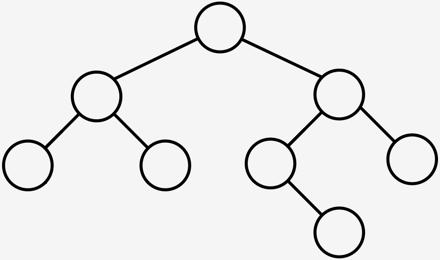
Invalid compacted tree

Observation

Only cherries (nodes with 2 pointers) might violate uniqueness.

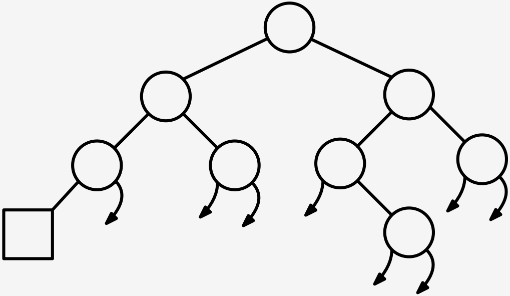
Counting compacted binary trees

Take a binary tree of size 8.



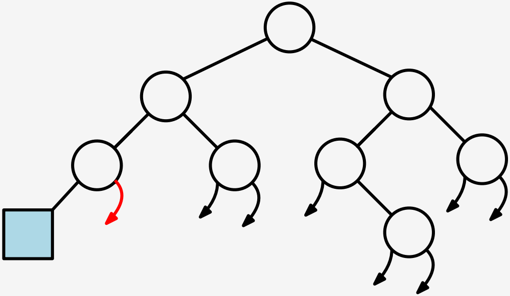
Counting compacted binary trees

Take a binary tree of size 8.



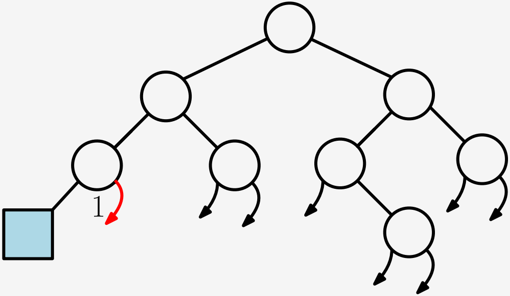
Counting compacted binary trees

Take a binary tree of size 8.



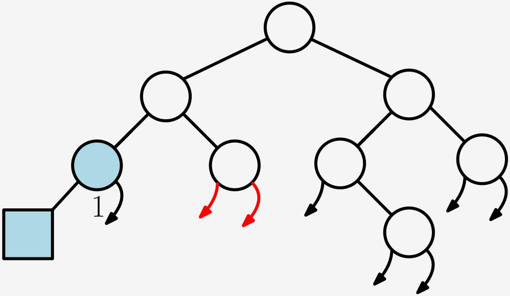
Counting compacted binary trees

Take a binary tree of size 8.



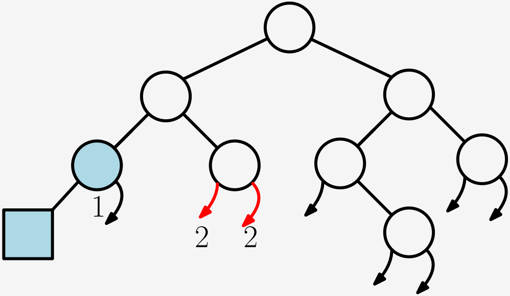
Counting compacted binary trees

Take a binary tree of size 8.



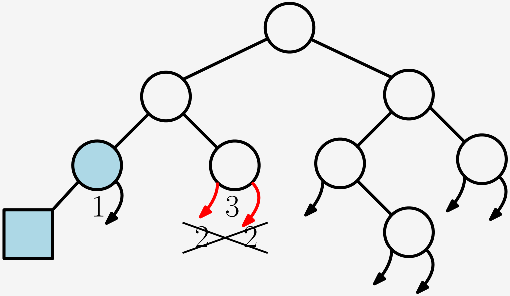
Counting compacted binary trees

Take a binary tree of size 8.



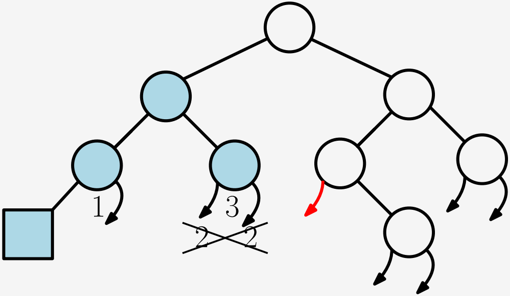
Counting compacted binary trees

Take a binary tree of size 8.



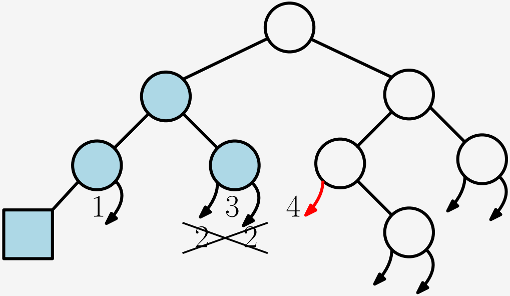
Counting compacted binary trees

Take a binary tree of size 8.



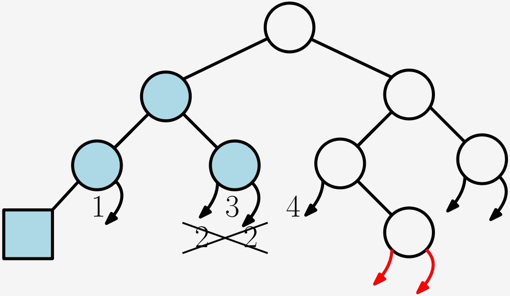
Counting compacted binary trees

Take a binary tree of size 8.



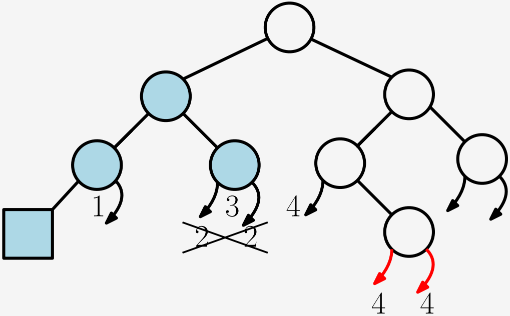
Counting compacted binary trees

Take a binary tree of size 8.



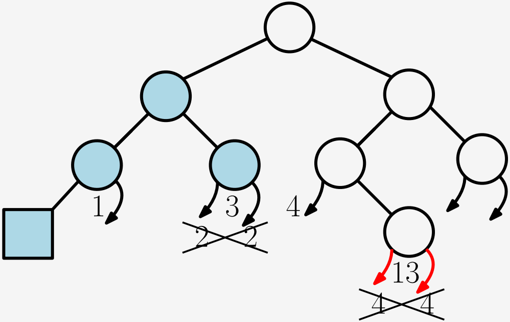
Counting compacted binary trees

Take a binary tree of size 8.



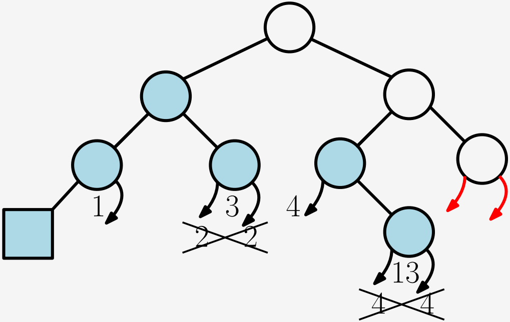
Counting compacted binary trees

Take a binary tree of size 8.



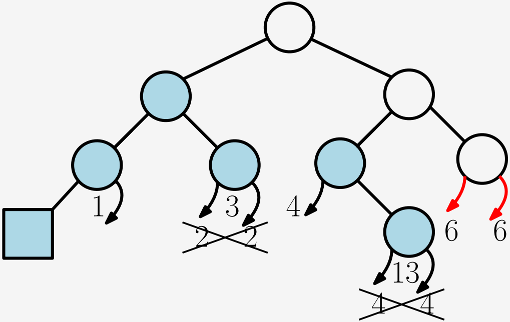
Counting compacted binary trees

Take a binary tree of size 8.



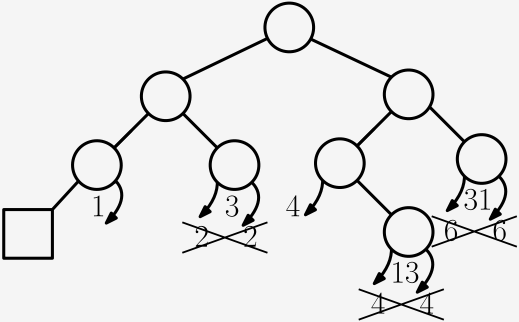
Counting compacted binary trees

Take a binary tree of size 8.



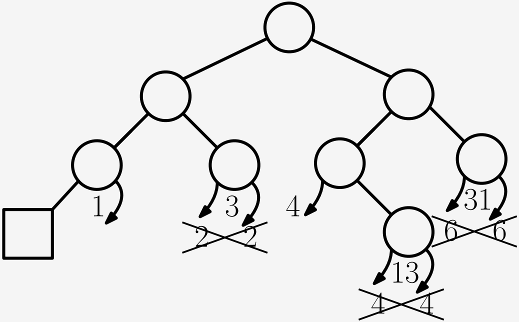
Counting compacted binary trees

Take a binary tree of size 8.



Counting compacted binary trees

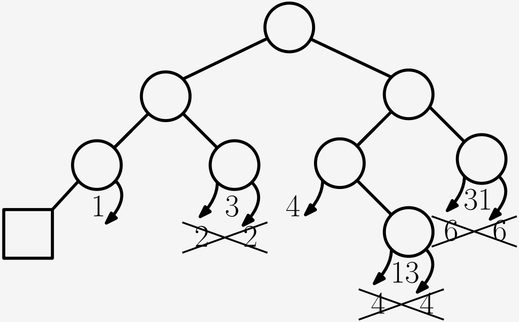
Take a binary tree of size 8.



In total, we can construct $1 \cdot 3 \cdot 4 \cdot 13 \cdot 31 = 4836$ compacted trees.

Counting compacted binary trees

Take a binary tree of size 8.



In total, we can construct $1 \cdot 3 \cdot 4 \cdot 13 \cdot 31 = 4836$ compacted trees.

Recurrence relation

This construction leads to a (complicated) recurrence relation of complexity $\mathcal{O}(n^3)$ to compute $c_1, c_2, \ldots, c_n$.

Relaxed compacted binary trees

Relaxed compacted binary trees

Drop the condition of uniqueness of the subtrees

Relaxed compacted binary trees

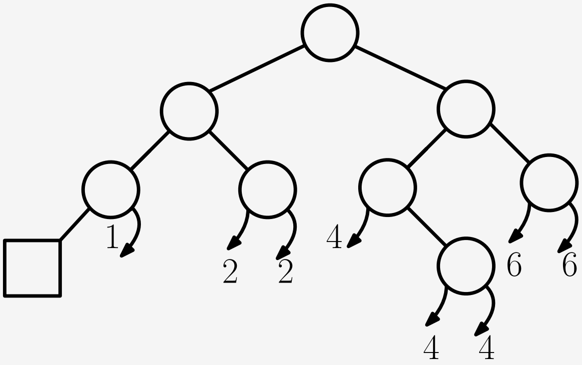
Drop the condition of uniqueness of the subtrees

Let r_n be the number of relaxed binary trees of size n: $c_n \leq r_n$.

Relaxed compacted binary trees

Drop the condition of uniqueness of the subtrees

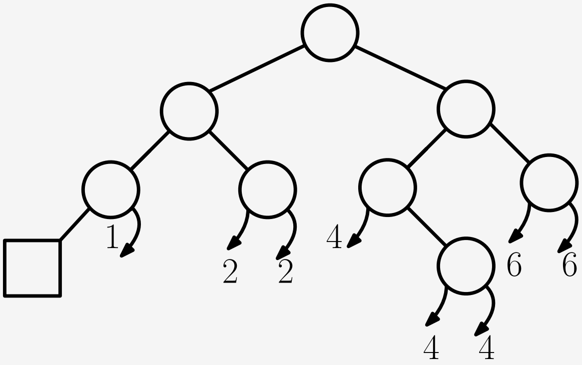
Let r_n be the number of relaxed binary trees of size n: $c_n \leq r_n$.



Relaxed compacted binary trees

Drop the condition of uniqueness of the subtrees

Let r_n be the number of relaxed binary trees of size n: $c_n \leq r_n$.



In total, this gives $1 \cdot 3 \cdot 4 \cdot 4^2 \cdot 6^2 = 6912$ relaxed trees and we get a similar recurrence relation.

(Before, 4836 compacted trees.)

Bounded right height

Bounded right height

We restrict to a subclass of relaxed binary trees: **bounded right height**.

Bounded right height

We restrict to a subclass of relaxed binary trees: **bounded right height**.

Right height

The right height of a binary tree is the maximal number of **right children on any path from the root to a leaf** (not going through pointers).

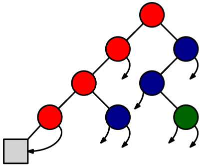
Bounded right height

We restrict to a subclass of relaxed binary trees: **bounded right height**.

Right height

The right height of a binary tree is the maximal number of **right children on any path from the root to a leaf** (not going through pointers).

Example



A binary tree with right height 2. Nodes of level 0 are colored in red, nodes of level 1 in blue, and the node of level 3 in green.

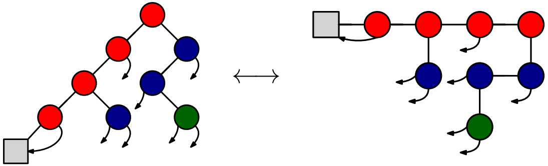
Bounded right height

We restrict to a subclass of relaxed binary trees: **bounded right height**.

Right height

The right height of a binary tree is the maximal number of **right children on any path from the root to a leaf** (not going through pointers).

Example



A binary tree with right height 2. Nodes of level 0 are colored in red, nodes of level 1 in blue, and the node of level 3 in green.

Compacted trees of right height $\leq k$

Figure: Right height ≤ 0.

Compacted trees of right height $\leq k$

Figure: Right height ≤ 0.

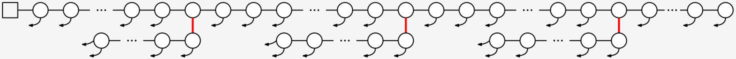


Figure: Right height ≤ 1.

Compacted trees of right height $\leq k$

Figure: Right height ≤ 0.

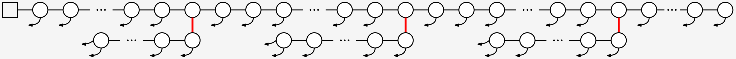


Figure: Right height ≤ 1.

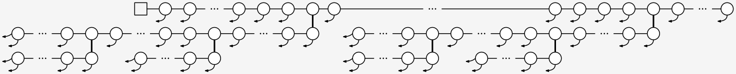


Figure: Right height ≤ 2.

Compacted trees of right height $\leq k$

Figure: Right height ≤ 0.

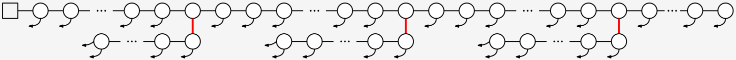


Figure: Right height ≤ 1.

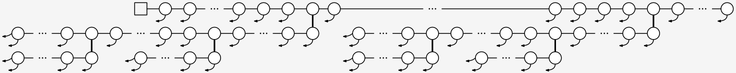


Figure: Right height ≤ 2.

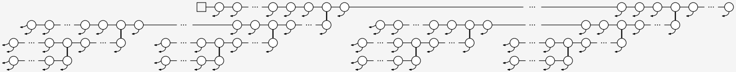


Figure: Right height ≤ 3.

Main results

Main results

Theorem (Relaxed)

The number $r_{k,n}$ of relaxed trees with right height at most k is for $n \to \infty$ asymptotically equivalent to

$$r_{k,n} \sim \gamma_k n! \left(4 \cos\left(\frac{\pi}{k+3}\right)^2\right)^n n^{-\frac{k}{2}},$$

where $\gamma_k \in \mathbb{R} \setminus \{0\}$ is independent of n.

Main results

Theorem (Relaxed)

The number $r_{k,n}$ of relaxed trees with right height at most k is for $n \to \infty$ asymptotically equivalent to

$$r_{k,n} \sim \gamma_k n! \left(4\cos\left(\frac{\pi}{k+3}\right)^2\right)^n n^{-\frac{k}{2}},$$

where $\gamma_k \in \mathbb{R} \setminus \{0\}$ is independent of n.

Theorem (Compacted)

The number $c_{k,n}$ of compacted trees with right height at most k is asymptotically equal to

$$c_{k,n} \sim \kappa_k n! \left(4\cos\left(\frac{\pi}{k+3}\right)^2\right)^n n^{-\frac{k}{2}-\frac{1}{k+3}-\left(\frac{1}{4}-\frac{1}{k+3}\right)\cos\left(\frac{\pi}{k+3}\right)^{-2}},$$

where $\kappa_k \in \mathbb{R} \setminus \{0\}$ is independent of n.

Proof idea

Methods

1. Recurrence relations
2. Bijections
3. Generating functions
4. Symbolic method
5. Differential equations
6. Singularity analysis
7. Chebyshev polynomials
8. Guess and prove

Proof idea

Methods

1. Recurrence relations
2. Bijections
3. Generating functions
4. Symbolic method
5. Differential equations
6. Singularity analysis
7. Chebyshev polynomials
8. Guess and prove

Main idea: Exponential generating functions

Let c_n be the number of compacted trees of size n. Then, we define

$$C(z) = \sum_{n\geq 0} c_n \frac{z^n}{n!}.$$

Upper bound $c_n \leq \frac{n!}{n+1}\binom{2n}{n}$ guarantees positive radius of convergence.

Main idea: Exponential generating functions

- Problem: unlabeled structures!
- Idea: **derive a symbolic method for compacted trees**

Main idea: Exponential generating functions

- Problem: unlabeled structures!
- Idea: **derive a symbolic method for compacted trees**

Let $T(z) = \sum_{n\geq 0} t_n \frac{z^n}{n!}$ be an EGF of the class $\mathcal{T}$.

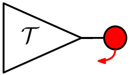
Main idea: Exponential generating functions

- Problem: unlabeled structures!
- Idea: **derive a symbolic method for compacted trees**

Let $T(z) = \sum_{n\geq 0} t_n \frac{z^n}{n!}$ be an EGF of the class $\mathcal{T}$.

$T(z) \mapsto zT(z)$

Append a new node with a pointer to the class $\mathcal{T}$.



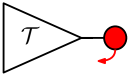
Main idea: Exponential generating functions

- Problem: unlabeled structures!
- Idea: **derive a symbolic method for compacted trees**

Let $T(z) = \sum_{n \geq 0} t_n \frac{z^n}{n!}$ be an EGF of the class $\mathcal{T}$.

$T(z) \mapsto zT(z)$

Append a new node with a pointer to the class $\mathcal{T}$.



Proof:

$$t_k = k![z^k]zT(z) = \quad k \quad \cdot \quad t_{k-1} \qquad \square$$

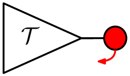
Main idea: Exponential generating functions

- Problem: unlabeled structures!
- Idea: **derive a symbolic method for compacted trees**

Let $T(z) = \sum_{n\geq 0} t_n \frac{z^n}{n!}$ be an EGF of the class $\mathcal{T}$.

$T(z) \mapsto zT(z)$

Append a new node with a pointer to the class $\mathcal{T}$.



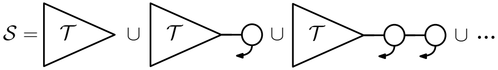
Proof:

$$t_k = k![z^k]zT(z) = \underbrace{k}_{\substack{k \text{ possible} \\ \text{pointers}}} \cdot \underbrace{t_{k-1}}_{\substack{k-1 \text{ internal} \\ \text{nodes}}} \qquad \square$$

Further constructions

$S : T(z) \mapsto \frac{1}{1-z} T(z)$

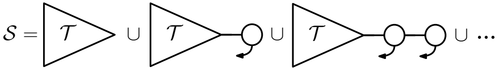
Append a (possibly empty) sequence at the root.



Further constructions

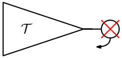
$S : T(z) \mapsto \frac{1}{1-z} T(z)$

Append a (possibly empty) sequence at the root.



$D : T(z) \mapsto \frac{d}{dz} T(z)$

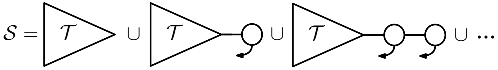
Delete top node but preserve its pointers.



Further constructions

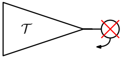
$S : T(z) \mapsto \frac{1}{1-z} T(z)$

Append a (possibly empty) sequence at the root.



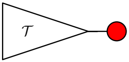
$D : T(z) \mapsto \frac{d}{dz} T(z)$

Delete top node but preserve its pointers.



$I : T(z) \mapsto \int T(z)$

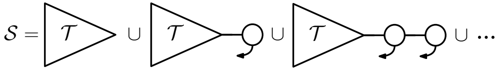
Add top node without pointers.



Further constructions

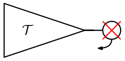
$S : T(z) \mapsto \frac{1}{1-z} T(z)$

Append a (possibly empty) sequence at the root.



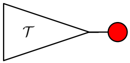
$D : T(z) \mapsto \frac{d}{dz} T(z)$

Delete top node but preserve its pointers.



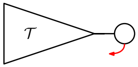
$I : T(z) \mapsto \int T(z)$

Add top node without pointers.



$P : T(z) \mapsto z \frac{d}{dz} T(z)$

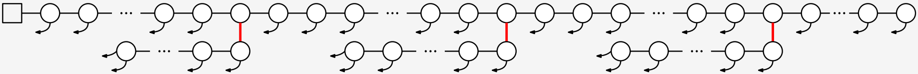
Add a new pointer to the top node.



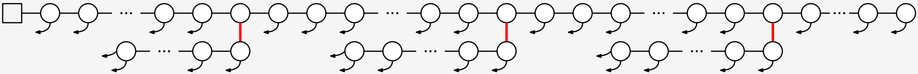
Relaxed binary trees

Highlights

Bounded right height ≤ 1: $R_1(z)$



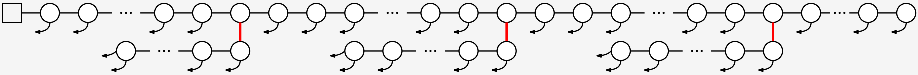
Bounded right height ≤ 1: $R_1(z)$



Symbolic construction

$$(1-2z)\,R_1'(z) - R_1(z) = 0,$$
$$R_1(0) = 1,$$

Bounded right height ≤ 1: $R_1(z)$



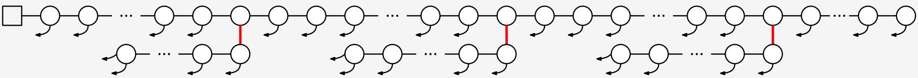
Symbolic construction

$$(1-2z)\,R_1'(z) - R_1(z) = 0,$$
$$R_1(0) = 1,$$

then we get the closed form

$$R_1(z) = \frac{1}{\sqrt{1-2z}},$$

Bounded right height ≤ 1: $R_1(z)$



Symbolic construction

$$(1-2z)\, R_1'(z) - R_1(z) = 0,$$
$$R_1(0) = 1,$$

then we get the closed form

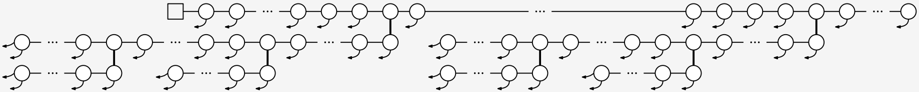
$$R_1(z) = \frac{1}{\sqrt{1-2z}},$$

and the coefficients

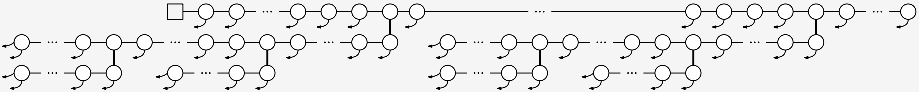
$$r_{1,n} = \frac{n!}{2^n}\binom{2n}{n} = (2n-1)\cdot(2n-3)\cdots 3\cdot 1.$$

[W 2019, "A bijection of plane increasing trees with relaxed binary trees of right height at most one"]. (TCS 2019, Vol. 755, p. 1–12; ArXiv:1706.07163)

Bounded right height ≤ 2: $R_2(z)$



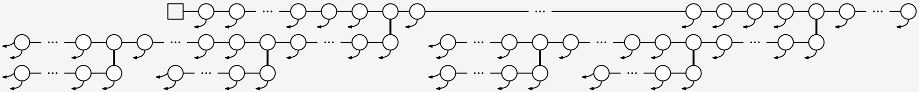
Bounded right height ≤ 2: $R_2(z)$



Symbolic construction

$$\left(1 - 3z + z^2\right) R_2''(z) + (2z - 3)\, R_2'(z) = 0,$$
$$R_2(0) = 1, \;\; R_2'(0) = 1,$$

Bounded right height ≤ 2: $R_2(z)$



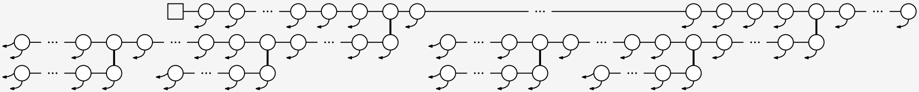
Symbolic construction

$$\left(1 - 3z + z^2\right) R_2''(z) + (2z - 3)\, R_2'(z) = 0,$$
$$R_2(0) = 1, \;\; R_2'(0) = 1,$$

then we get the closed form

$$R_2'(z) = \frac{1}{1 - 3z + z^2},$$

Bounded right height ≤ 2: $R_2(z)$



Symbolic construction

$$\left(1-3z+z^2\right) R_2''(z) + (2z-3)\, R_2'(z) = 0,$$
$$R_2(0) = 1,\ \ R_2'(0) = 1,$$

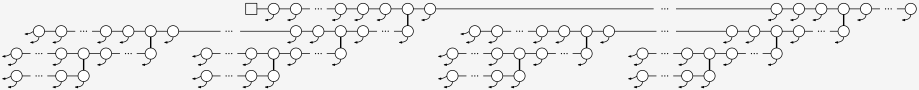
then we get the closed form

$$R_2'(z) = \frac{1}{1-3z+z^2},$$

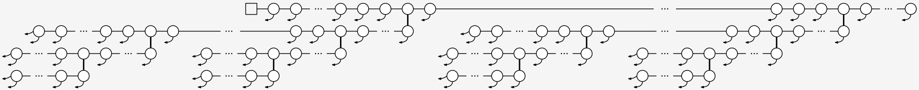
and the coefficients

$$r_{2,n} = \frac{(n-1)!}{\sqrt{5}} \left(\left(\frac{1+\sqrt{5}}{2} \right)^{2n} - \left(\frac{1-\sqrt{5}}{2} \right)^{2n} \right).$$

Bounded right height ≤ 3: $R_3(z)$



Bounded right height ≤ 3: $R_3(z)$

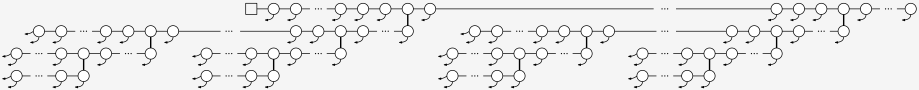


Symbolic construction

$$\left(1-4z+3z^2\right)R_3'''(z)+(9z-6)\,R_3''(z)+2R_3'(z)=0,$$

$$R_3(0)=1,\ R_3'(0)=1,\ R_3''(0)=\frac{3}{2},$$

Bounded right height ≤ 3: $R_3(z)$



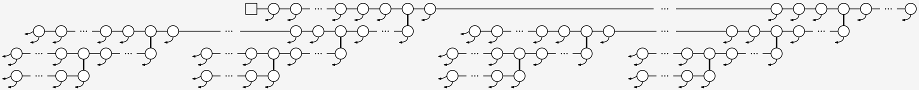
Symbolic construction

$$\left(1-4z+3z^2\right) R_3'''(z) + (9z-6)\, R_3''(z) + 2R_3'(z) = 0,$$
$$R_3(0)=1,\ R_3'(0)=1,\ R_3''(0)=\frac{3}{2},$$

then we get the closed form

$$R_3(z) = \left(\frac{3z-2+\sqrt{3}\sqrt{1-4z+3z^2}}{\sqrt{3}-2} \right)^{1/\sqrt{3}},$$

Bounded right height ≤ 3: $R_3(z)$



Symbolic construction

$$\left(1-4z+3z^2\right) R_3'''(z) + (9z-6)\, R_3''(z) + 2R_3'(z) = 0,$$

$$R_3(0) = 1,\ R_3'(0) = 1,\ R_3''(0) = \frac{3}{2},$$

then we get the closed form

$$R_3(z) = \left(\frac{3z - 2 + \sqrt{3}\sqrt{1-4z+3z^2}}{\sqrt{3}-2} \right)^{1/\sqrt{3}},$$

and the asymptotics of the coefficients

$$r_{3,n} = n![z^n]R_3(z) = \frac{n!}{\sqrt{6}\,(2-\sqrt{3})^{1/\sqrt{3}}} \frac{3^n}{n^{3/2}\sqrt{\pi}} \left(1 + \mathcal{O}\left(\frac{1}{n}\right)\right).$$

Sneak Preview

Enumeration of compacted binary trees WITHOUT height restrictions

(Joint work with Andrew Elvey Price and Wenjie Fang)

A stretched exponential appears

Theorem

The number of compacted and relaxed binary trees satisfy for $n \to \infty$

$$r_n = \Theta\left(n!\, 4^n e^{3a_1 n^{1/3}} n\right),$$

$$c_n = \Theta\left(n!\, 4^n e^{3a_1 n^{1/3}} n^{3/4}\right),$$

where $a_1 \approx -2.3381$ *is the largest root of the Airy function* $\mathrm{Ai}(x) = \frac{1}{\pi}\int_0^\infty \cos\left(\frac{t^3}{3} + xt\right) dt.$

A stretched exponential appears

Theorem

The number of compacted and relaxed binary trees satisfy for $n \to \infty$

$$r_n = \Theta\left(n!\, 4^n e^{3a_1 n^{1/3}} n\right),$$

$$c_n = \Theta\left(n!\, 4^n e^{3a_1 n^{1/3}} n^{3/4}\right),$$

where $a_1 \approx -2.3381$ is the largest root of the Airy function $\mathrm{Ai}(x) = \frac{1}{\pi}\int_0^\infty \cos\left(\frac{t^3}{3} + xt\right) dt.$

Corollary (Proportion of compacted among relaxed trees)

$$\frac{c_n}{r_n} = \Theta(n^{-1/4}),$$

A stretched exponential appears

Theorem

The number of compacted and relaxed binary trees satisfy for $n \to \infty$

$$r_n = \Theta\left(n!\, 4^n e^{3a_1 n^{1/3}} n\right),$$

$$c_n = \Theta\left(n!\, 4^n e^{3a_1 n^{1/3}} n^{3/4}\right),$$

where $a_1 \approx -2.3381$ is the largest root of the Airy function $\mathrm{Ai}(x) = \frac{1}{\pi}\int_0^\infty \cos\left(\frac{t^3}{3} + xt\right) dt.$

Corollary (Proportion of compacted among relaxed trees)

$$\frac{c_n}{r_n} = \Theta(n^{-1/4}),$$

$$\frac{c_{k,n}}{r_{k,n}} \sim \lambda_k n^{-\frac{1}{k+3} - \left(\frac{1}{4} - \frac{1}{k+3}\right)\frac{1}{\cos^2\left(\frac{\pi}{k+3}\right)}} = o\left(n^{-1/4}\right),$$

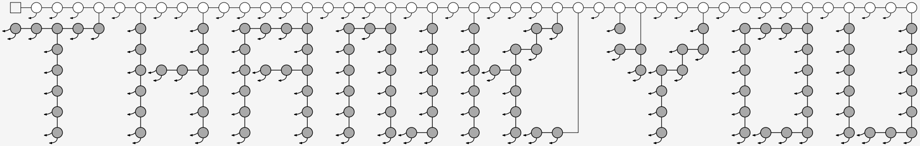
for a constant λ_k independent of n.

Next steps

- Different tree structures, like e.g. ternary trees
- Analyze shape parameters, like height, width, profile, ...

Next steps

- Different tree structures, like e.g. ternary trees
- Analyze shape parameters, like height, width, profile, ...



Backup

A recurrence for compacted binary trees

Counting formula

Let $n, p \in \mathbb{N}$, then

$$\gamma_{n+1,p} = \sum_{i=0}^{n} \gamma_{i,p}\gamma_{n-i,p+i}, \qquad \text{for } n \geq 1,$$

- Helps us to efficiently compute c_n
- Asymptotic analysis failed (so far)
  One reason: asymptotically every summand matters
- Summands possess 3 (!) dependencies on i

A recurrence for compacted binary trees

Counting formula

Let $n, p \in \mathbb{N}$, then

$$\gamma_{n+1,p} = \sum_{i=0}^{n} \gamma_{i,p}\gamma_{n-i,p+i}, \qquad \text{for } n \geq 1,$$
$$\gamma_{0,p} = p + 1,$$

- Helps us to efficiently compute c_n
- Asymptotic analysis failed (so far)
  One reason: asymptotically every summand matters
- Summands possess 3 (!) dependencies on i

A recurrence for compacted binary trees

Counting formula

Let $n, p \in \mathbb{N}$, then

$$\gamma_{n+1,p} = \sum_{i=0}^{n} \gamma_{i,p}\gamma_{n-i,p+i}, \qquad \text{for } n \geq 1,$$
$$\gamma_{0,p} = p+1,$$
$$\gamma_{1,p} = p^2 + p + 1.$$

- Helps us to efficiently compute c_n
- Asymptotic analysis failed (so far)
  One reason: asymptotically every summand matters
- Summands possess 3 (!) dependencies on i

A recurrence for compacted binary trees

Counting formula

Let $n, p \in \mathbb{N}$, then

$$\gamma_{n+1,p} = \sum_{i=0}^{n} \gamma_{i,p}\gamma_{n-i,p+i}, \qquad \text{for } n \geq 1,$$
$$\gamma_{0,p} = p + 1,$$
$$\gamma_{1,p} = p^2 + p + 1.$$

We are interested in $c_n = \gamma_{n,0}$.

- Helps us to efficiently compute c_n
- Asymptotic analysis failed (so far)
  One reason: asymptotically every summand matters
- Summands possess 3 (!) dependencies on i

A recurrence for compacted binary trees

Counting formula

Let $n, p \in \mathbb{N}$, then

$$\gamma_{n+1,p} = \sum_{i=0}^{n} \gamma_{i,p}\gamma_{n-i,p+i}, \qquad \text{for } n \geq 1,$$
$$\gamma_{0,p} = p+1,$$
$$\gamma_{1,p} = p^2 + p + 1.$$

We are interested in $c_n = \gamma_{n,0}$.

- Helps us to efficiently compute c_n
- Asymptotic analysis failed (so far)
  One reason: asymptotically every summand matters
- Summands possess 3 (!) dependencies on i

A recurrence for compacted binary trees

Counting formula

Let $n, p \in \mathbb{N}$, then

$$\begin{aligned} \gamma_{n+1,p} &= \sum_{i=0}^{n} \gamma_{i,p}\gamma_{n-i,p+i}, \qquad \text{for } n \geq 1, \\ \gamma_{0,p} &= p+1, \\ \gamma_{1,p} &= p^2+p+1. \end{aligned}$$

We are interested in $c_n = \gamma_{n,0}$.

- Helps us to efficiently compute c_n
- Asymptotic analysis failed (so far)
  One reason: asymptotically every summand matters
- Summands possess 3 (!) dependencies on i

A recurrence for compacted binary trees

Counting formula

Let $n, p \in \mathbb{N}$, then

$$\gamma_{n+1,p} = \sum_{i=0}^{n} \gamma_{i,p}\gamma_{n-i,p+i}, \qquad \text{for } n \geq 1,$$
$$\gamma_{0,p} = p + 1,$$
$$\gamma_{1,p} = p^2 + p + 1.$$

We are interested in $c_n = \gamma_{n,0}$.

- Helps us to efficiently compute c_n
- Asymptotic analysis failed (so far)
  One reason: asymptotically every summand matters
- Summands possess 3 (!) dependencies on i

A recurrence for relaxed compacted binary trees

Counting formula

Let $n, p \in \mathbb{N}$, then

$$\delta_{n+1,p} = \sum_{i=0}^{n} \delta_{i,p}\delta_{n-i,p+i}, \qquad \text{for } n \geq 0,$$

$$\delta_{0,p} = p + 1, \qquad \xcancel{\delta_{1,p} = p^2 + p + 1}.$$

We are interested in $r_n = \delta_{n,0}$.

A recurrence for relaxed compacted binary trees

Counting formula

Let $n, p \in \mathbb{N}$, then

$$\delta_{n+1,p} = \sum_{i=0}^{n} \delta_{i,p}\delta_{n-i,p+i}, \qquad \text{for } n \geq 0,$$

$$\delta_{0,p} = p + 1, \qquad \cancel{\delta_{1,p} = p^2 + p + 1}.$$

We are interested in $r_n = \delta_{n,0}$.

Recursion still too complicated.

A recurrence for relaxed compacted binary trees

Counting formula

Let $n, p \in \mathbb{N}$, then

$$\delta_{n+1,p} = \sum_{i=0}^{n} \delta_{i,p}\delta_{n-i,p+i}, \qquad \text{for } n \geq 0,$$

$$\delta_{0,p} = p+1, \qquad \cancel{\delta_{1,p} = p^2 + p + 1}.$$

We are interested in $r_n = \delta_{n,0}$.

Recursion still too complicated.

Example (Relaxed binary trees)

size	$n=0$	$n=1$	$n=2$	$n=3$	$n=4$	$n=5$	$n=6$
c_n	1	1	3	15	111	1119	14487
r_n	1	1	3	16	127	1363	18628

Comparing compacted and relaxed trees

Asymptotics of compacted and relaxed trees

$$c_{k,n} \sim \kappa_k n! r_k^n n^{\alpha_k} \qquad \text{and} \qquad r_{k,n} \sim \gamma_k n! r_k^n n^{\beta_k}.$$

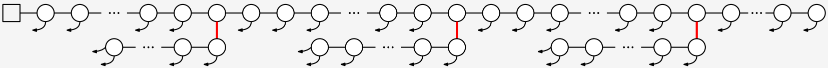
Comparing compacted and relaxed trees

Asymptotics of compacted and relaxed trees

$$c_{k,n} \sim \kappa_k n! r_k^n n^{\alpha_k} \qquad \text{and} \qquad r_{k,n} \sim \gamma_k n! r_k^n n^{\beta_k}.$$

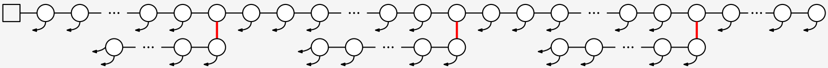
k	r_k	$r_k \approx$	$\kappa_k \approx$	α_k	$\alpha_k \approx$	$\gamma_k \approx$	β_k	$\beta_k \approx$
1	2	2.000	0.708	$-\frac{3}{4}$	-0.750	0.564	$-\frac{1}{2}$	-0.5
2	$4\cos(\frac{\pi}{5})^2$	2.618	0.561	$-\frac{6}{5} - \frac{1}{20\cos(\frac{\pi}{5})^2}$	-1.276	0.447	-1	-1.0
3	3	3.000	0.605	$-\frac{16}{9}$	-1.778	0.493	$-\frac{3}{2}$	-1.5
4	$4\cos(\frac{\pi}{7})^2$	3.246	0.873	$-\frac{15}{7} - \frac{3}{28\cos(\frac{\pi}{7})^2}$	-2.275	0.726	-2	-2.0
5	$4\cos(\frac{\pi}{8})^2$	3.414	1.625	$-\frac{21}{8} - \frac{1}{8\cos(\frac{\pi}{8})^2}$	-2.772	1.379	$-\frac{5}{2}$	-2.5
6	$4\cos(\frac{\pi}{9})^2$	3.532	3.782	$-\frac{28}{9} - \frac{5}{36\cos(\frac{\pi}{9})^2}$	-3.268	3.260	-3	-3.0
7	$4\cos(\frac{\pi}{10})^2$	3.618	10.708	$-\frac{18}{5} - \frac{3}{20\cos(\frac{\pi}{10})^2}$	-3.766	9.350	$-\frac{7}{2}$	-3.5

Construction of $R_1(z)$



Let $R_1(z) = \sum_{\ell \geq 0} r_{1,n} \frac{z^n}{n!}$ be the EGF of relaxed binary trees with bounded right height ≤ 1.

Construction of $R_1(z)$



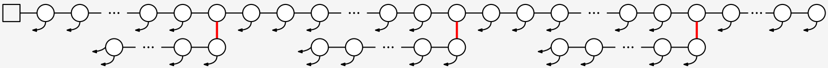
Let $R_1(z) = \sum_{\ell \geq 0} r_{1,n} \frac{z^n}{n!}$ be the EGF of relaxed binary trees with bounded right height ≤ 1.

Decomposition of $R_1(z)$

$$R_1(z) = \sum_{n \geq 0} R_{1,\ell}(z)$$

where $R_{1,\ell}(z)$ is the EGF for relaxed binary trees with exactly ℓ left-subtrees, i.e. ℓ left-edges from level 0 to level 1.

Construction of $R_1(z)$



Let $R_1(z) = \sum_{\ell \geq 0} r_{1,n} \frac{z^n}{n!}$ be the EGF of relaxed binary trees with bounded right height ≤ 1.

Decomposition of $R_1(z)$

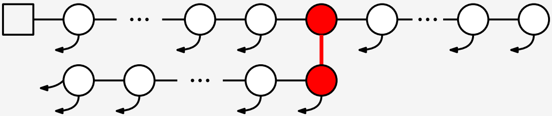
$$R_1(z) = \sum_{n \geq 0} R_{1,\ell}(z)$$

where $R_{1,\ell}(z)$ is the EGF for relaxed binary trees with exactly ℓ left-subtrees, i.e. ℓ left-edges from level 0 to level 1.

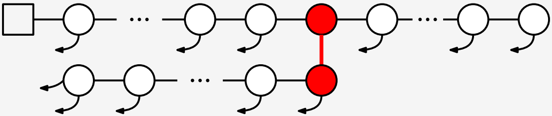
$$R_{1,0}(z) = R_0(z) = \frac{1}{1-z}$$

$$R_{1,1}(z) = \ ?$$

Construction of $R_{1,1}(z)$

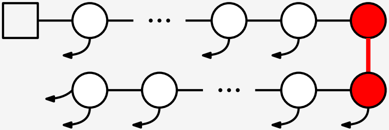


Construction of $R_{1,1}(z)$

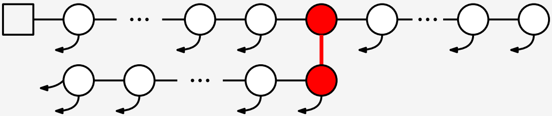


Symbolic specification

1 delete initial sequence

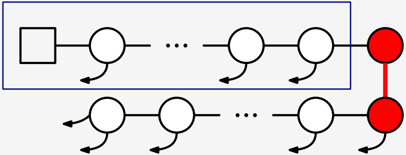


Construction of $R_{1,1}(z)$

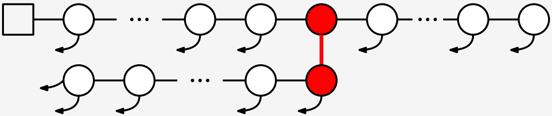


Symbolic specification

1. delete initial sequence
2. decompose

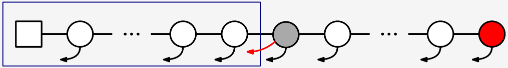


Construction of $R_{1,1}(z)$

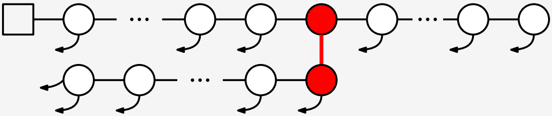


Symbolic specification

1. delete initial sequence
2. decompose
3. append and add pointer

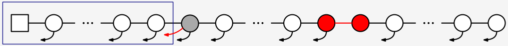


Construction of $R_{1,1}(z)$



Symbolic specification

1. delete initial sequence
2. decompose
3. append and add pointer
4. add initial sequence

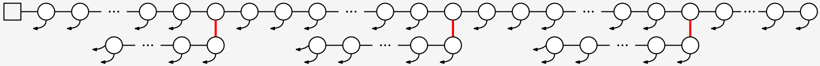


$R_{1,1}(z)$

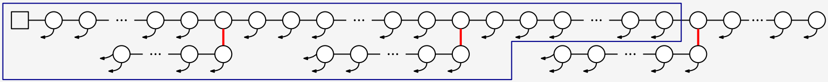
$$R_{1,1}(z) = \underbrace{S}_{\substack{\text{init.}\\\text{seq.}}} \circ \underbrace{I}_{\substack{\text{lvl 0}\\\text{node}}} \circ \underbrace{S \circ P}_{\substack{\text{red pointer}\\\text{and seq.}}} \left(\underbrace{zR_{1,0}(z)}_{\text{non empty}} \right)$$

$$R_{1,1}(z) = \frac{1}{1-z} \int \frac{1}{1-z} z \left(zR_{1,0}(z) \right)' \, dz$$

Construction of $R_{1,\ell}(z)$

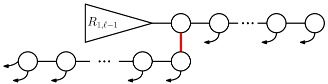


Construction of $R_{1,\ell}(z)$



Observation

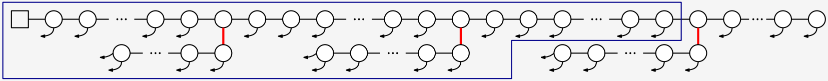
Same structure as for $R_{1,1}(z)$



$$R_{1,\ell}(z) = \frac{1}{1-z} \int \frac{1}{1-z} z \left(z R_{1,\ell-1}(z) \right)' \, dz, \qquad \ell \geq 1,$$

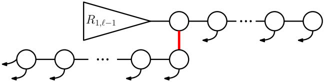
$$R_{1,0}(z) = R_0(z) = \frac{1}{1-z}.$$

Construction of $R_{1,\ell}(z)$



Observation

Same structure as for $R_{1,1}(z)$



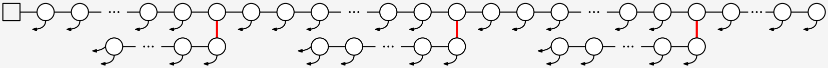
$$R_{1,\ell}(z) = \frac{1}{1-z} \int \frac{1}{1-z} z \left(z R_{1,\ell-1}(z)\right)' \, dz, \qquad \ell \geq 1,$$

$$R_{1,0}(z) = R_0(z) = \frac{1}{1-z}.$$

Recall that $R_1(z) = \sum_{\ell \geq 0} R_{1,\ell}(z)$. Summing the previous equation (formally) for $\ell \geq 1$ gives

$$\frac{1-2z}{1-z} R_1'(z) - \frac{1}{1-z} R_1(z) - \left((1-z) R_{1,0}(z)\right)' = 0.$$

Closed form of $R_1(z)$

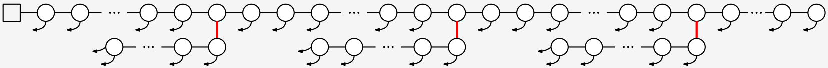


$$\frac{1-2z}{1-z} R_1'(z) - \frac{1}{1-z} R_1(z) - ((1-z)R_{1,0}(z))' = 0.$$

We know that $R_{1,0}(z) = \frac{1}{1-z}$ and get

$$(1-2z)\, R_1'(z) - R_1(z) = 0, \qquad \text{with} \qquad R_1(0) = 1.$$

Closed form of $R_1(z)$



$$\frac{1-2z}{1-z}R_1'(z) - \frac{1}{1-z}R_1(z) - ((1-z)R_{1,0}(z))' = 0.$$

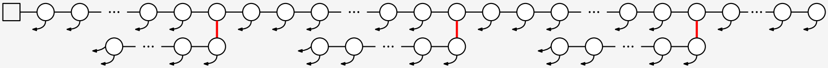
We know that $R_{1,0}(z) = \frac{1}{1-z}$ and get

$$(1-2z)\,R_1'(z) - R_1(z) = 0, \qquad \text{with} \qquad R_1(0) = 1.$$

This directly yields

$$R_1(z) = \frac{1}{\sqrt{1-2z}}.$$

Closed form of $R_1(z)$



$$\frac{1-2z}{1-z}R_1'(z) - \frac{1}{1-z}R_1(z) - ((1-z)R_{1,0}(z))' = 0.$$

We know that $R_{1,0}(z) = \frac{1}{1-z}$ and get

$$(1-2z)\,R_1'(z) - R_1(z) = 0, \qquad \text{with} \qquad R_1(0) = 1.$$

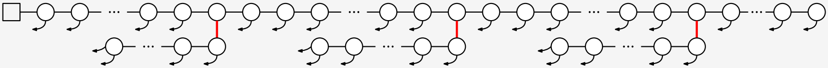
This directly yields

$$R_1(z) = \frac{1}{\sqrt{1-2z}}.$$

Therefore we get

$$r_{1,n} = n![z^n]R_1(z) = \frac{n!}{2^n}\binom{2n}{n} = (2n-1)!!.$$

Closed form of $R_1(z)$



$$\frac{1-2z}{1-z}R_1'(z) - \frac{1}{1-z}R_1(z) - ((1-z)R_{1,0}(z))' = 0.$$

We know that $R_{1,0}(z) = \frac{1}{1-z}$ and get

$$(1-2z)\,R_1'(z) - R_1(z) = 0, \qquad \text{with} \qquad R_1(0) = 1.$$

This directly yields

$$R_1(z) = \frac{1}{\sqrt{1-2z}}.$$

Therefore we get

$$r_{1,n} = n![z^n]R_1(z) = \frac{n!}{2^n}\binom{2n}{n} = (2n-1)!!.$$

Preprint (ArXiv:1706.07163): [W, 2017, "A bijection of plane increasing trees with relaxed binary trees of right height at most one"].

Sketch of proof for relaxed trees

1. Let $\ell_{k,i} \in \mathbb{C}[z]$ be such that
$$L_k = \ell_{k,k}(z)D^k + \ell_{k,k-1}(z)D^{k-1} + \ldots + \ell_{k,0}(z).$$
Find recurrences for $\ell_{k,i}(z)$ using Guess'n'Prove techniques.
2. Use singularity analysis directly on differential equation:
3. Exponential growth ρ_k: Roots of coefficient of leading polynomial $\ell_{k,k}(z)$ are candidates.
4. $\ell_{k,k}(z)$ is a transformed Chebyshev polynomial of the second kind. Hence,
$$\rho_k = \frac{1}{4\cos\left(\frac{\pi}{k+3}\right)^2}.$$
5. Subexponential growth: Use the indicial polynomial derived from the $\ell_{k,i}(z)$.
6. Find a basis of solutions for differential equation:
Only one is singular at ρ_k!
7. Prove that other coefficients $\ell_{k,i}(z)$ are nice.

Sketch of proof for relaxed trees

1. Let $\ell_{k,i} \in \mathbb{C}[z]$ be such that
$$L_k = \ell_{k,k}(z)D^k + \ell_{k,k-1}(z)D^{k-1} + \ldots + \ell_{k,0}(z).$$
Find recurrences for $\ell_{k,i}(z)$ using Guess'n'Prove techniques.
2. Use singularity analysis directly on differential equation:
3. Exponential growth ρ_k: Roots of coefficient of leading polynomial $\ell_{k,k}(z)$ are candidates.
4. $\ell_{k,k}(z)$ is a transformed Chebyshev polynomial of the second kind. Hence,
$$\rho_k = \frac{1}{4\cos\left(\frac{\pi}{k+3}\right)^2}.$$
5. Subexponential growth: Use the indicial polynomial derived from the $\ell_{k,i}(z)$.
6. Find a basis of solutions for differential equation:
Only one is singular at ρ_k!
7. Prove that other coefficients $\ell_{k,i}(z)$ are nice.

Sketch of proof for relaxed trees

1. Let $\ell_{k,i} \in \mathbb{C}[z]$ be such that
$$L_k = \ell_{k,k}(z)D^k + \ell_{k,k-1}(z)D^{k-1} + \ldots + \ell_{k,0}(z).$$
Find recurrences for $\ell_{k,i}(z)$ using Guess'n'Prove techniques.
2. Use singularity analysis directly on differential equation:
3. Exponential growth ρ_k: Roots of coefficient of leading polynomial $\ell_{k,k}(z)$ are candidates.
4. $\ell_{k,k}(z)$ is a transformed Chebyshev polynomial of the second kind. Hence,
$$\rho_k = \frac{1}{4\cos\left(\frac{\pi}{k+3}\right)^2}.$$
5. Subexponential growth: Use the indicial polynomial derived from the $\ell_{k,i}(z)$.
6. Find a basis of solutions for differential equation:
Only one is singular at ρ_k!
7. Prove that other coefficients $\ell_{k,i}(z)$ are nice.

Sketch of proof for relaxed trees

1. Let $\ell_{k,i} \in \mathbb{C}[z]$ be such that
$$L_k = \ell_{k,k}(z)D^k + \ell_{k,k-1}(z)D^{k-1} + \ldots + \ell_{k,0}(z).$$
Find recurrences for $\ell_{k,i}(z)$ using Guess'n'Prove techniques.
2. Use singularity analysis directly on differential equation:
3. Exponential growth ρ_k: Roots of coefficient of leading polynomial $\ell_{k,k}(z)$ are candidates.
4. $\ell_{k,k}(z)$ is a transformed Chebyshev polynomial of the second kind. Hence,
$$\rho_k = \frac{1}{4\cos\left(\frac{\pi}{k+3}\right)^2}.$$
5. Subexponential growth: Use the indicial polynomial derived from the $\ell_{k,i}(z)$.
6. Find a basis of solutions for differential equation:
Only one is singular at ρ_k!
7. Prove that other coefficients $\ell_{k,i}(z)$ are nice.

Sketch of proof for relaxed trees

1. Let $\ell_{k,i} \in \mathbb{C}[z]$ be such that
$$L_k = \ell_{k,k}(z)D^k + \ell_{k,k-1}(z)D^{k-1} + \ldots + \ell_{k,0}(z).$$
Find recurrences for $\ell_{k,i}(z)$ using Guess'n'Prove techniques.
2. Use singularity analysis directly on differential equation:
3. Exponential growth ρ_k: Roots of coefficient of leading polynomial $\ell_{k,k}(z)$ are candidates.
4. $\ell_{k,k}(z)$ is a transformed Chebyshev polynomial of the second kind. Hence,
$$\rho_k = \frac{1}{4\cos\left(\frac{\pi}{k+3}\right)^2}.$$
5. Subexponential growth: Use the indicial polynomial derived from the $\ell_{k,i}(z)$.
6. Find a basis of solutions for differential equation:
Only one is singular at ρ_k!
7. Prove that other coefficients $\ell_{k,i}(z)$ are nice.

Sketch of proof for relaxed trees

1. Let $\ell_{k,i} \in \mathbb{C}[z]$ be such that
$$L_k = \ell_{k,k}(z)D^k + \ell_{k,k-1}(z)D^{k-1} + \ldots + \ell_{k,0}(z).$$
Find recurrences for $\ell_{k,i}(z)$ using Guess'n'Prove techniques.
2. Use singularity analysis directly on differential equation:
3. Exponential growth ρ_k: Roots of coefficient of leading polynomial $\ell_{k,k}(z)$ are candidates.
4. $\ell_{k,k}(z)$ is a transformed Chebyshev polynomial of the second kind. Hence,
$$\rho_k = \frac{1}{4\cos\left(\frac{\pi}{k+3}\right)^2}.$$
5. Subexponential growth: Use the indicial polynomial derived from the $\ell_{k,i}(z)$.
6. Find a basis of solutions for differential equation:
Only one is singular at ρ_k!
7. Prove that other coefficients $\ell_{k,i}(z)$ are nice.

Sketch of proof for relaxed trees

1. Let $\ell_{k,i} \in \mathbb{C}[z]$ be such that
$$L_k = \ell_{k,k}(z)D^k + \ell_{k,k-1}(z)D^{k-1} + \ldots + \ell_{k,0}(z).$$
Find recurrences for $\ell_{k,i}(z)$ using Guess'n'Prove techniques.
2. Use singularity analysis directly on differential equation:
3. Exponential growth ρ_k: Roots of coefficient of leading polynomial $\ell_{k,k}(z)$ are candidates.
4. $\ell_{k,k}(z)$ is a transformed Chebyshev polynomial of the second kind. Hence,
$$\rho_k = \frac{1}{4\cos\left(\frac{\pi}{k+3}\right)^2}.$$
5. Subexponential growth: Use the indicial polynomial derived from the $\ell_{k,i}(z)$.
6. Find a basis of solutions for differential equation:
Only one is singular at ρ_k!
7. Prove that other coefficients $\ell_{k,i}(z)$ are nice.

Differential operators

Theorem

Let $(L_k)_{k\geq 0}$ *be a family of differential operators given by*

$$\begin{aligned} L_0 &= (1-z), \\ L_1 &= (1-2z)D - 1, \\ L_k &= L_{k-1}\cdot D - L_{k-2}\cdot D^2 \cdot z, \qquad k \geq 2. \end{aligned}$$

Then the exponential generating function $R_k(z)$ *for relaxed trees with right height* $\leq k$ *satisfies*

$$L_k \cdot R_k = 0.$$

Differential operators

Theorem

Let $(L_k)_{k\geq 0}$ be a family of differential operators given by

$$\begin{aligned} L_0 &= (1-z), \\ L_1 &= (1-2z)D - 1, \\ L_k &= L_{k-1}\cdot D - L_{k-2}\cdot D^2 \cdot z, \qquad k \geq 2. \end{aligned}$$

Then the exponential generating function $R_k(z)$ for relaxed trees with right height $\leq k$ satisfies

$$L_k \cdot R_k = 0.$$

$$(1-2z)\frac{d}{dz}R_1(z) - R_1(z) = 0$$

Differential operators

Theorem

Let $(L_k)_{k\geq 0}$ be a family of differential operators given by

$$\begin{aligned} L_0 &= (1-z), \\ L_1 &= (1-2z)D - 1, \\ L_k &= L_{k-1} \cdot D - L_{k-2} \cdot D^2 \cdot z, \qquad k \geq 2. \end{aligned}$$

Then the exponential generating function $R_k(z)$ for relaxed trees with right height $\leq k$ satisfies

$$L_k \cdot R_k = 0.$$

$$(1-2z)\frac{d}{dz}R_1(z) - R_1(z) = 0$$

$$(z^2 - 3z + 1)\frac{d^2}{dz^2}R_2(z) + (2z-3)\frac{d}{dz}R_2(z) = 0$$

Differential operators

Theorem

Let $(L_k)_{k\geq 0}$ be a family of differential operators given by

$$\begin{aligned} L_0 &= (1-z), \\ L_1 &= (1-2z)D - 1, \\ L_k &= L_{k-1}\cdot D - L_{k-2}\cdot D^2 \cdot z, \qquad k \geq 2. \end{aligned}$$

Then the exponential generating function $R_k(z)$ for relaxed trees with right height $\leq k$ satisfies

$$L_k \cdot R_k = 0.$$

$$(1-2z)\frac{d}{dz}R_1(z) - R_1(z) = 0$$

$$(z^2-3z+1)\frac{d^2}{dz^2}R_2(z) + (2z-3)\frac{d}{dz}R_2(z) = 0$$

$$(3z^2-4z+1)\frac{d^3}{dz^3}R_3(z) + (9z-6)\frac{d^2}{dz^2}R_3(z) + 2\frac{d}{dz}R_3(z) = 0$$