Asymptotic Enumeration of Compacted Binary Trees with Height Restrictions

CLA 05/2018

Michael Wallner

joint work with Antoine Genitrini, Bernhard Gittenberger and Manuel Kauers

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

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

May 24th, 2018

Based on the paper:
Asymptotic Enumeration of Compacted Binary Trees,
submitted to a journal.
ArXiv:1703.10031

Creating a compacted 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 $(x^2 - y^2)(x^2 + y^2)$.

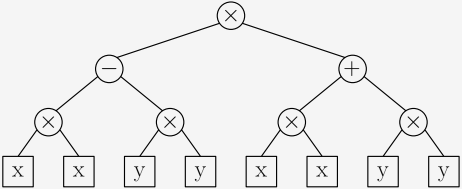
Motivation – Efficiently store redundant information

Example

Consider the labeled tree necessary to store the arithmetic expression

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

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



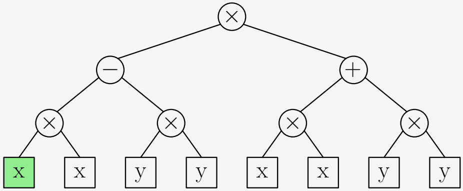
Motivation – Efficiently store redundant information

Example

Consider the labeled tree necessary to store the arithmetic expression

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

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



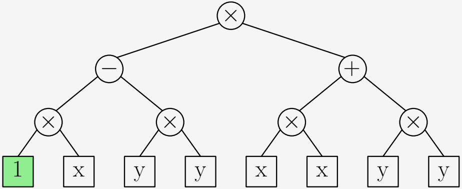
Motivation – Efficiently store redundant information

Example

Consider the labeled tree necessary to store the arithmetic expression

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

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



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

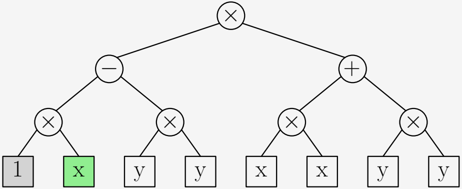
Motivation – Efficiently store redundant information

Example

Consider the labeled tree necessary to store the arithmetic expression

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

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



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

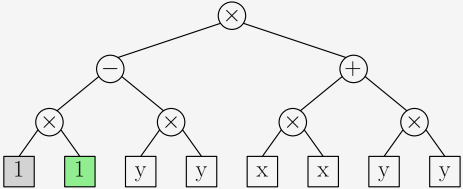
Motivation – Efficiently store redundant information

Example

Consider the labeled tree necessary to store the arithmetic expression

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

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



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

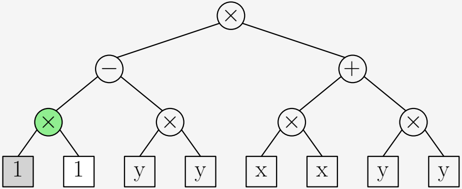
Motivation – Efficiently store redundant information

Example

Consider the labeled tree necessary to store the arithmetic expression

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

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



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

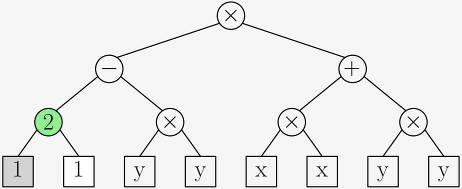
Motivation – Efficiently store redundant information

Example

Consider the labeled tree necessary to store the arithmetic expression

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

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



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

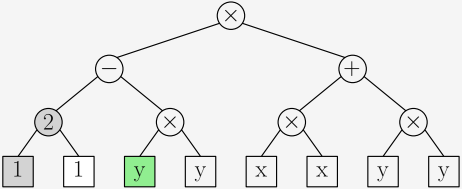
Motivation – Efficiently store redundant information

Example

Consider the labeled tree necessary to store the arithmetic expression

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

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



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

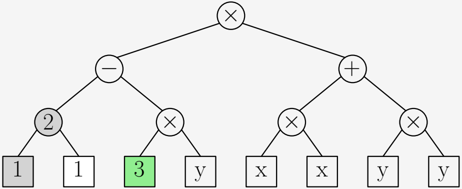
Motivation – Efficiently store redundant information

Example

Consider the labeled tree necessary to store the arithmetic expression

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

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



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

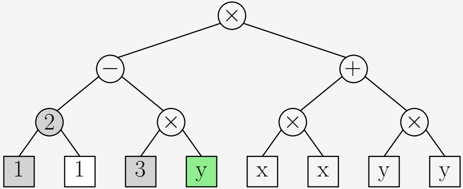
Motivation – Efficiently store redundant information

Example

Consider the labeled tree necessary to store the arithmetic expression

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

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



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

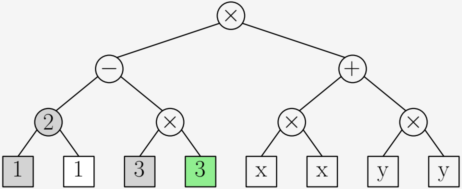
Motivation – Efficiently store redundant information

Example

Consider the labeled tree necessary to store the arithmetic expression

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

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



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

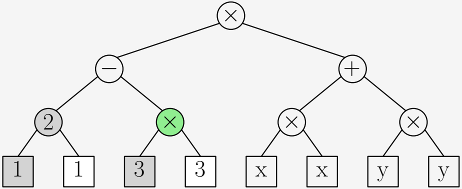
Motivation – Efficiently store redundant information

Example

Consider the labeled tree necessary to store the arithmetic expression

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

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



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

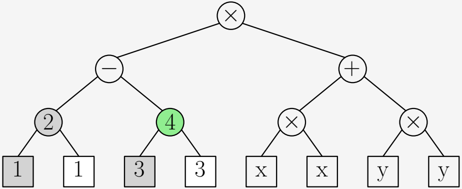
Motivation – Efficiently store redundant information

Example

Consider the labeled tree necessary to store the arithmetic expression

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

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



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

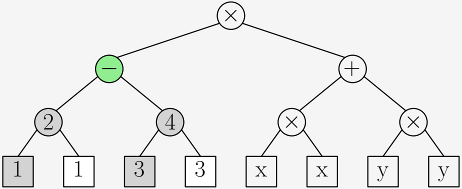
Motivation – Efficiently store redundant information

Example

Consider the labeled tree necessary to store the arithmetic expression

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

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



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

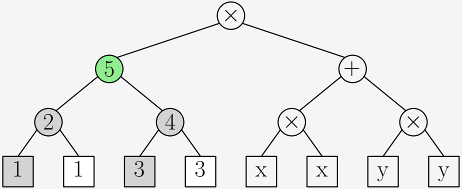
Motivation – Efficiently store redundant information

Example

Consider the labeled tree necessary to store the arithmetic expression

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

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



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

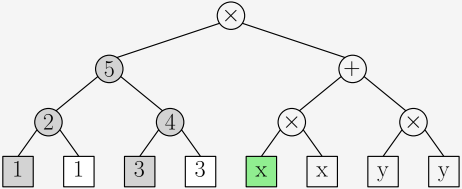
Motivation – Efficiently store redundant information

Example

Consider the labeled tree necessary to store the arithmetic expression

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

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



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

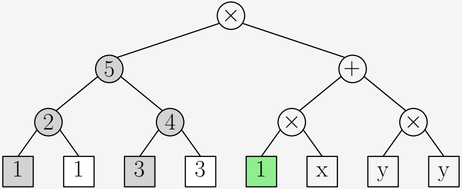
Motivation – Efficiently store redundant information

Example

Consider the labeled tree necessary to store the arithmetic expression

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

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



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

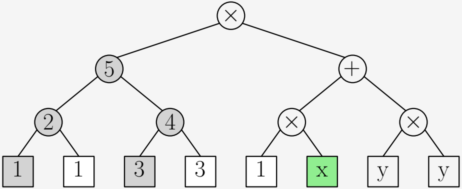
Motivation – Efficiently store redundant information

Example

Consider the labeled tree necessary to store the arithmetic expression

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

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



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

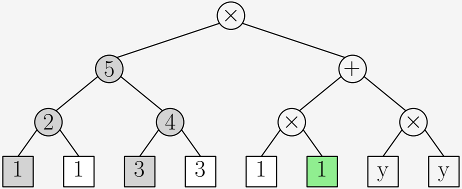
Motivation – Efficiently store redundant information

Example

Consider the labeled tree necessary to store the arithmetic expression

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

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



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

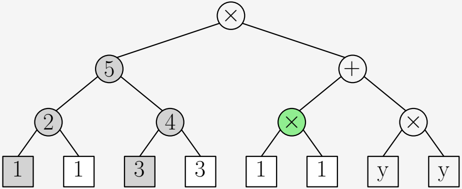
Motivation – Efficiently store redundant information

Example

Consider the labeled tree necessary to store the arithmetic expression

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

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



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

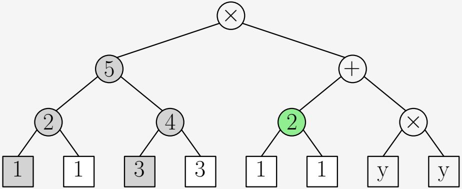
Motivation – Efficiently store redundant information

Example

Consider the labeled tree necessary to store the arithmetic expression

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

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



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

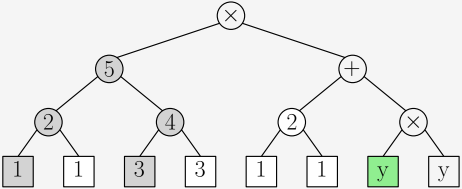
Motivation – Efficiently store redundant information

Example

Consider the labeled tree necessary to store the arithmetic expression

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

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



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

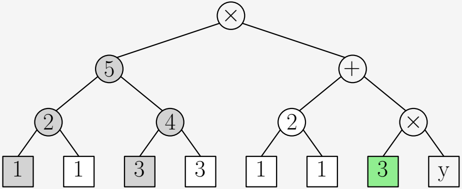
Motivation – Efficiently store redundant information

Example

Consider the labeled tree necessary to store the arithmetic expression

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

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



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

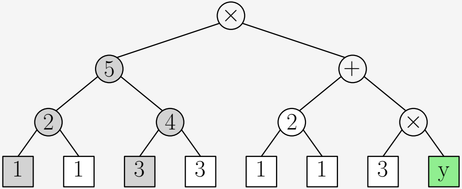
Motivation – Efficiently store redundant information

Example

Consider the labeled tree necessary to store the arithmetic expression

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

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



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

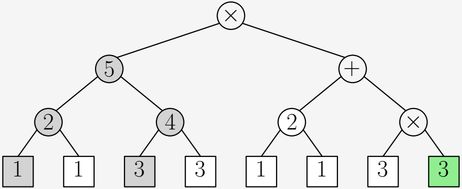
Motivation – Efficiently store redundant information

Example

Consider the labeled tree necessary to store the arithmetic expression

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

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



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

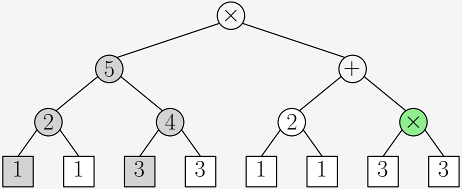
Motivation – Efficiently store redundant information

Example

Consider the labeled tree necessary to store the arithmetic expression

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

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



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

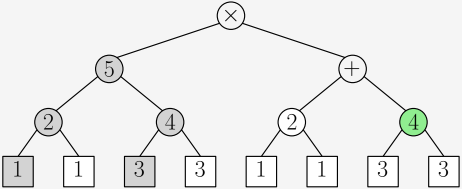
Motivation – Efficiently store redundant information

Example

Consider the labeled tree necessary to store the arithmetic expression

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

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



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

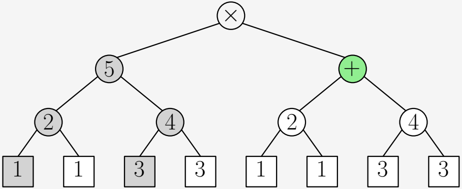
Motivation – Efficiently store redundant information

Example

Consider the labeled tree necessary to store the arithmetic expression

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

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



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

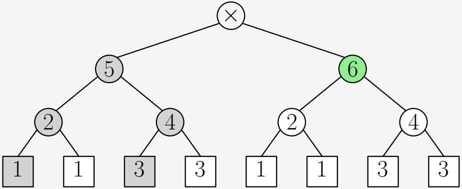
Motivation – Efficiently store redundant information

Example

Consider the labeled tree necessary to store the arithmetic expression

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

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



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

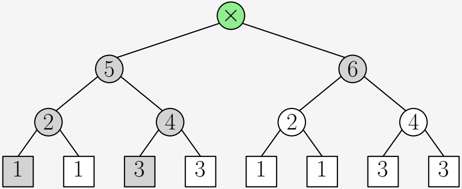
Motivation – Efficiently store redundant information

Example

Consider the labeled tree necessary to store the arithmetic expression

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

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



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

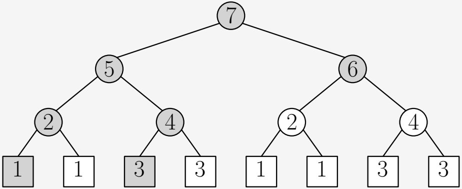
Motivation – Efficiently store redundant information

Example

Consider the labeled tree necessary to store the arithmetic expression

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

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



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

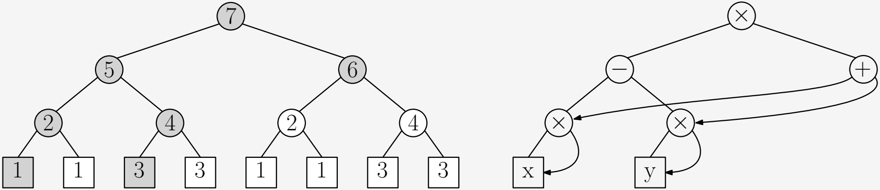
Motivation – Efficiently store redundant information

Example

Consider the labeled tree necessary to store the arithmetic expression

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

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



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

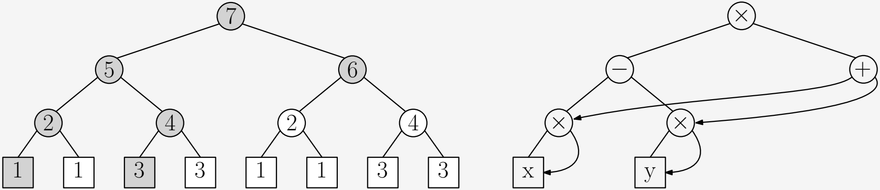
Motivation – Efficiently store redundant information

Example

Consider the labeled tree necessary to store the arithmetic expression

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

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



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

Definition

Compacted tree is the DAG computed by this procedure.

Compacted trees

- Important property: **Subtrees are unique**
- Efficient algorithm to compute compacted tree
  - Traverse tree post-order
  - If subtree appears twice, delete second one and replace by pointer
    $\rightarrow$ *directed acyclic graph* (DAG)
- Analyzed by [Flajolet, Sipala, Steyaert 1990]: 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
  - Traverse tree post-order
  - If subtree appears twice, delete second one and replace by pointer
    $\rightarrow$ *directed acyclic graph* (DAG)
- Analyzed by [Flajolet, Sipala, Steyaert 1990]: 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
  - Traverse tree post-order
  - If subtree appears twice, delete second one and replace by pointer
    $\rightarrow$ *directed acyclic graph* (DAG)
- Analyzed by [Flajolet, Sipala, Steyaert 1990]: 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
  - Traverse tree post-order
  - If subtree appears twice, delete second one and replace by pointer
    $\rightarrow$ *directed acyclic graph* (DAG)
- Analyzed by [Flajolet, Sipala, Steyaert 1990]: 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
  - Traverse tree post-order
  - If subtree appears twice, delete second one and replace by pointer
    $\rightarrow$ *directed acyclic graph* (DAG)
- Analyzed by [Flajolet, Sipala, Steyaert 1990]: 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
  - Traverse tree post-order
  - If subtree appears twice, delete second one and replace by pointer
    $\rightarrow$ *directed acyclic graph* (DAG)
- Analyzed by [Flajolet, Sipala, Steyaert 1990]: 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 trees

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

Compacted trees

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

Compacted trees

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

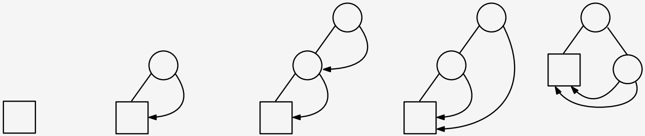


Figure: All compacted binary trees of size $n = 0, 1, 2$.

Compacted trees

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

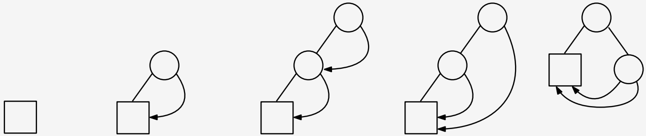


Figure: All compacted binary trees of size $n = 0, 1, 2$.

Example (Compacted 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

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

Hence, $c_n = \mathcal{O}(n! 4^n n^{-1/2})$.

Goals of this talk

Goals

1. Understand compacted trees
2. Find a recurrence relation for compacted trees
3. Use exponential generating functions to count DAGs
4. Solve the (simplified) problem(s)

Goals of this talk

Goals

1. Understand compacted trees
2. Find a recurrence relation for compacted trees
3. Use exponential generating functions to count DAGs
4. Solve the (simplified) problem(s)

Goals of this talk

Goals

1. Understand compacted trees
2. Find a recurrence relation for compacted trees
3. Use exponential generating functions to count DAGs
4. Solve the (simplified) problem(s)

Goals of this talk

Goals

1. Understand compacted trees
2. Find a recurrence relation for compacted trees
3. Use exponential generating functions to count DAGs
4. Solve the (simplified) problem(s)

Goals of this talk

Goals

1. Understand compacted trees
2. Find a recurrence relation for compacted trees
3. Use exponential generating functions to count DAGs
4. Solve the (simplified) problem(s)

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

Building a compacted tree from a binary tree

Idea

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

Building a compacted tree from a binary tree

Idea

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

- Attention: Pointers are not allowed to violate uniqueness
- Observation: Only cherries (nodes with 2 pointers) might violate uniqueness

Building a compacted tree from a binary tree

Idea

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

- Attention: Pointers are not allowed to violate uniqueness
- Observation: Only cherries (nodes with 2 pointers) might violate uniqueness

Building a compacted tree from a binary tree

Idea

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

- Attention: Pointers are not allowed to violate uniqueness
- Observation: Only cherries (nodes with 2 pointers) might violate uniqueness

Procedure

1. Take a binary tree of size n
2. Add leaf as left child on first free spot in post-order traversal
3. Add pointers such that out-degree of all internal nodes is 2
4. Connect pointers to leaf or to internal nodes before the root in post-order NOT violating uniqueness

Building a compacted tree from a binary tree

Idea

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

- Attention: Pointers are not allowed to violate uniqueness
- Observation: Only cherries (nodes with 2 pointers) might violate uniqueness

Procedure

1. Take a binary tree of size n
2. Add leaf as left child on first free spot in post-order traversal
3. Add pointers such that out-degree of all internal nodes is 2
4. Connect pointers to leaf or to internal nodes before the root in post-order NOT violating uniqueness

Building a compacted tree from a binary tree

Idea

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

- Attention: Pointers are not allowed to violate uniqueness
- Observation: Only cherries (nodes with 2 pointers) might violate uniqueness

Procedure

1. Take a binary tree of size n
2. Add leaf as left child on first free spot in post-order traversal
3. Add pointers such that out-degree of all internal nodes is 2
4. Connect pointers to leaf or to internal nodes before the root in post-order NOT violating uniqueness

Building a compacted tree from a binary tree

Idea

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

- Attention: Pointers are not allowed to violate uniqueness
- Observation: Only cherries (nodes with 2 pointers) might violate uniqueness

Procedure

1. Take a binary tree of size n
2. Add leaf as left child on first free spot in post-order traversal
3. Add pointers such that out-degree of all internal nodes is 2
4. Connect pointers to leaf or to internal nodes before the root in post-order NOT violating uniqueness

Building a compacted tree from a binary tree – Example

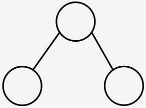
Procedure

1. Take a binary tree of size n (called *spine*)
2. Add leaf as left child on first free spot in post-order traversal
3. Add pointers such that out-degree of all internal nodes is 2
4. Connect pointers to leaf or to internal nodes before the root in post-order NOT violating uniqueness

Building a compacted tree from a binary tree – Example

Procedure

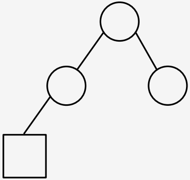
1. Take a binary tree of size n (called *spine*)
2. Add leaf as left child on first free spot in post-order traversal
3. Add pointers such that out-degree of all internal nodes is 2
4. Connect pointers to leaf or to internal nodes before the root in post-order NOT violating uniqueness



Building a compacted tree from a binary tree – Example

Procedure

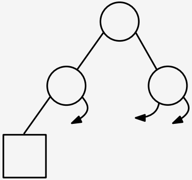
1. Take a binary tree of size n (called *spine*)
2. Add leaf as left child on first free spot in post-order traversal
3. Add pointers such that out-degree of all internal nodes is 2
4. Connect pointers to leaf or to internal nodes before the root in post-order NOT violating uniqueness



Building a compacted tree from a binary tree – Example

Procedure

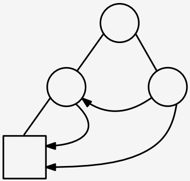
1. Take a binary tree of size n (called *spine*)
2. Add leaf as left child on first free spot in post-order traversal
3. Add pointers such that out-degree of all internal nodes is 2
4. Connect pointers to leaf or to internal nodes before the root in post-order NOT violating uniqueness



Building a compacted tree from a binary tree – Example

Procedure

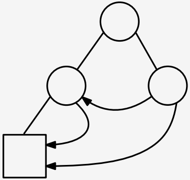
1. Take a binary tree of size n (called *spine*)
2. Add leaf as left child on first free spot in post-order traversal
3. Add pointers such that out-degree of all internal nodes is 2
4. Connect pointers to leaf or to internal nodes before the root in post-order NOT violating uniqueness



Building a compacted tree from a binary tree – Example

Procedure

1. Take a binary tree of size n (called *spine*)
2. Add leaf as left child on first free spot in post-order traversal
3. Add pointers such that out-degree of all internal nodes is 2
4. Connect pointers to leaf or to internal nodes before the root in post-order NOT violating uniqueness

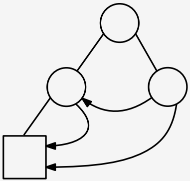


Valid compacted tree

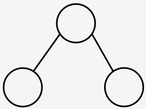
Building a compacted tree from a binary tree – Example

Procedure

1. Take a binary tree of size n (called *spine*)
2. Add leaf as left child on first free spot in post-order traversal
3. Add pointers such that out-degree of all internal nodes is 2
4. Connect pointers to leaf or to internal nodes before the root in post-order NOT violating uniqueness



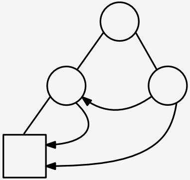
Valid compacted tree



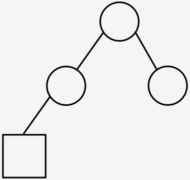
Building a compacted tree from a binary tree – Example

Procedure

1. Take a binary tree of size n (called *spine*)
2. Add leaf as left child on first free spot in post-order traversal
3. Add pointers such that out-degree of all internal nodes is 2
4. Connect pointers to leaf or to internal nodes before the root in post-order NOT violating uniqueness



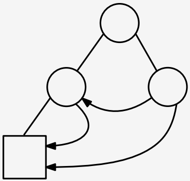
Valid compacted tree



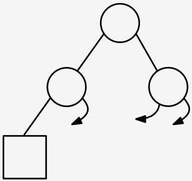
Building a compacted tree from a binary tree – Example

Procedure

1. Take a binary tree of size n (called *spine*)
2. Add leaf as left child on first free spot in post-order traversal
3. Add pointers such that out-degree of all internal nodes is 2
4. Connect pointers to leaf or to internal nodes before the root in post-order NOT violating uniqueness



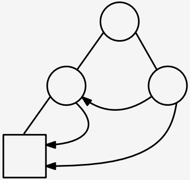
Valid compacted tree



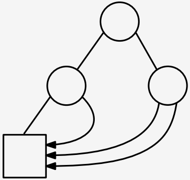
Building a compacted tree from a binary tree – Example

Procedure

1. Take a binary tree of size n (called *spine*)
2. Add leaf as left child on first free spot in post-order traversal
3. Add pointers such that out-degree of all internal nodes is 2
4. Connect pointers to leaf or to internal nodes before the root in post-order NOT violating uniqueness



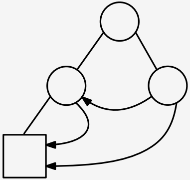
Valid compacted tree



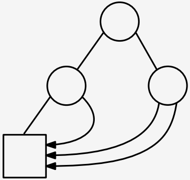
Building a compacted tree from a binary tree – Example

Procedure

1. Take a binary tree of size n (called *spine*)
2. Add leaf as left child on first free spot in post-order traversal
3. Add pointers such that out-degree of all internal nodes is 2
4. Connect pointers to leaf or to internal nodes before the root in post-order NOT violating uniqueness



Valid compacted tree

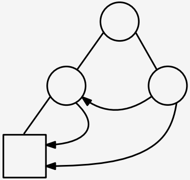


Invalid compacted tree

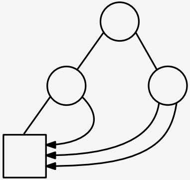
Building a compacted tree from a binary tree – Example

Procedure

1. Take a binary tree of size n (called *spine*)
2. Add leaf as left child on first free spot in post-order traversal
3. Add pointers such that out-degree of all internal nodes is 2
4. Connect pointers to leaf or to internal nodes before the root in post-order NOT violating uniqueness



Valid compacted tree



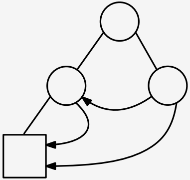
Invalid compacted tree

We call the underlying binary tree from step 1 the *spine*.

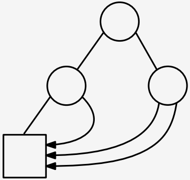
Building a compacted tree from a binary tree – Example

Procedure

1. Take a binary tree of size n (called *spine*)
2. Add leaf as left child on first free spot in post-order traversal
3. Add pointers such that out-degree of all internal nodes is 2
4. Connect pointers to leaf or to internal nodes before the root in post-order NOT violating uniqueness



Valid compacted tree



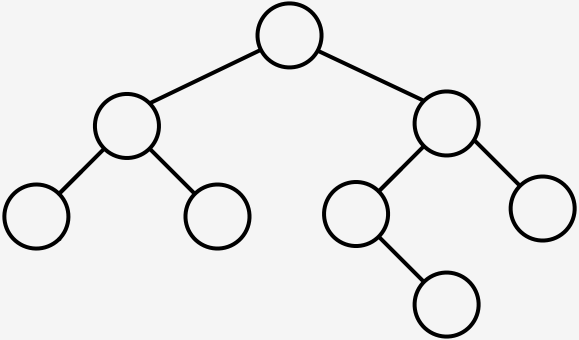
Invalid compacted tree

We call the underlying binary tree from step 1 the *spine*.

This spine is associated to 3 valid compacted trees.

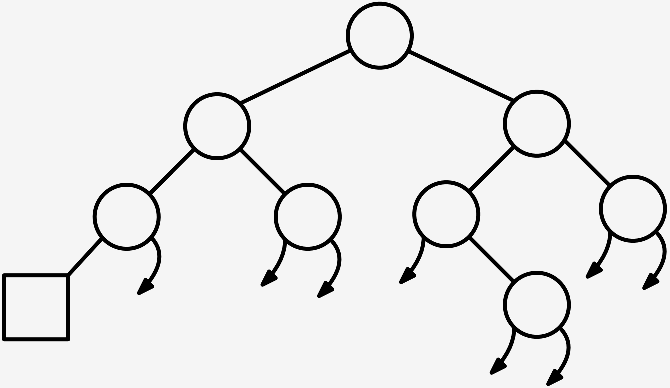
A bigger example

We take a binary tree of size 8.



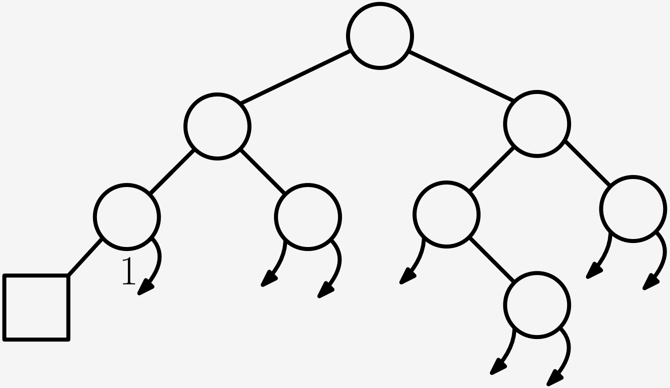
A bigger example

We take a binary tree of size 8.



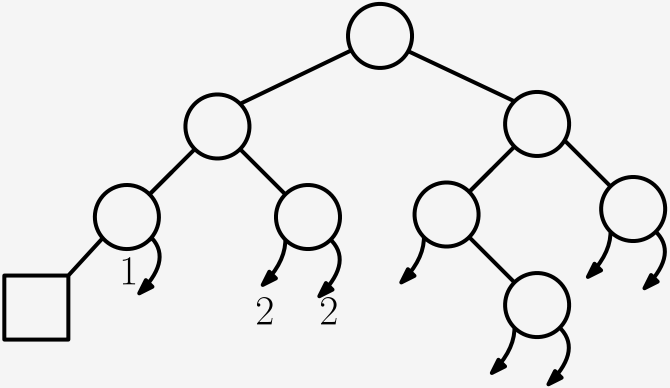
A bigger example

We take a binary tree of size 8.



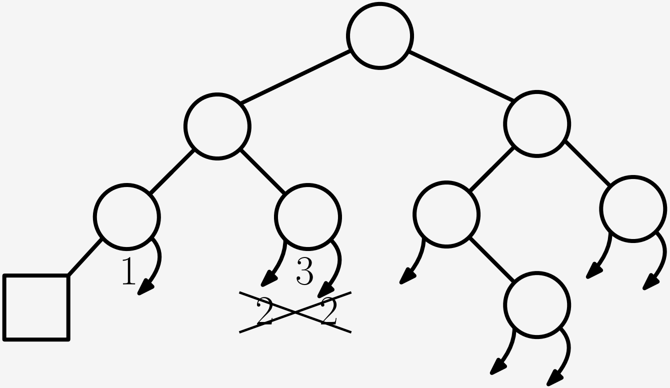
A bigger example

We take a binary tree of size 8.



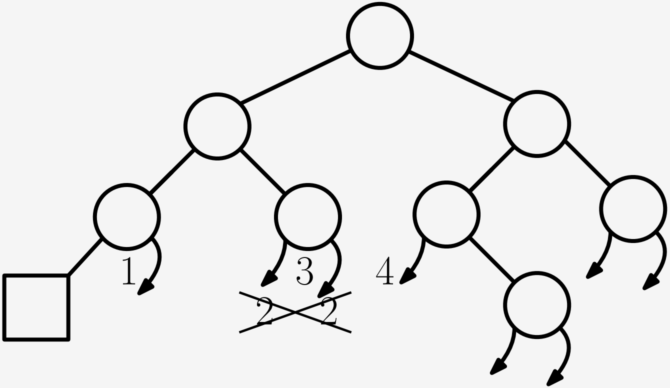
A bigger example

We take a binary tree of size 8.



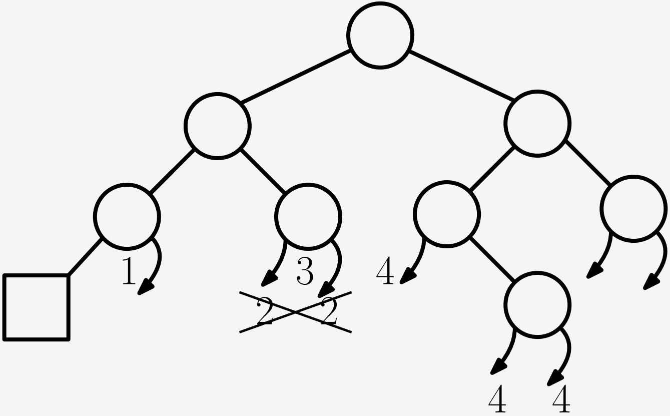
A bigger example

We take a binary tree of size 8.



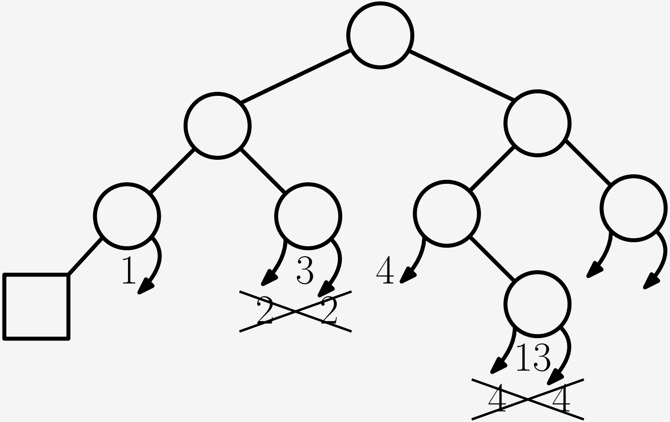
A bigger example

We take a binary tree of size 8.



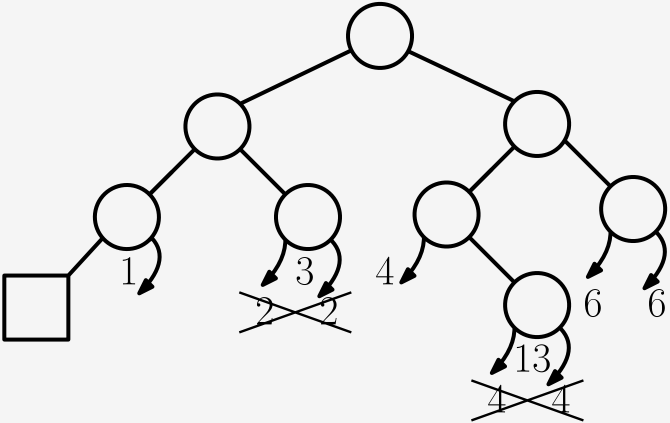
A bigger example

We take a binary tree of size 8.



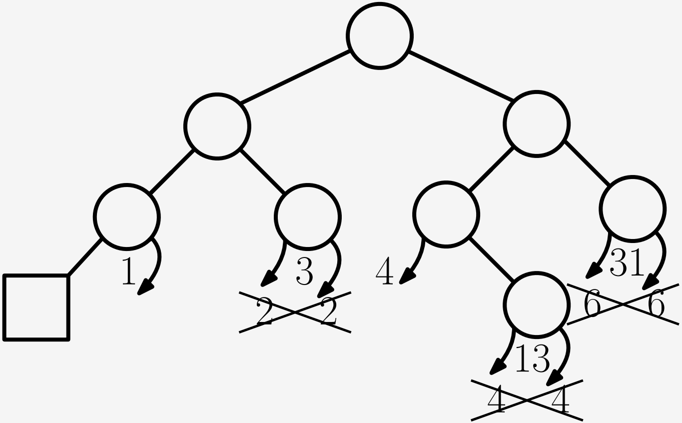
A bigger example

We take a binary tree of size 8.



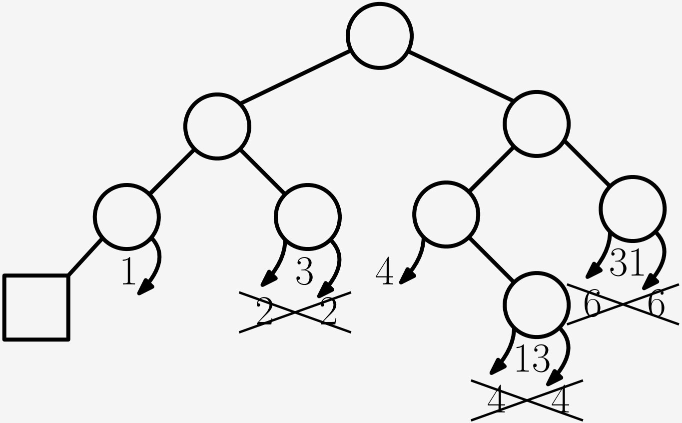
A bigger example

We take a binary tree of size 8.



A bigger example

We take a binary tree of size 8.



In total, this spine corresponds to $1 \cdot 3 \cdot 4 \cdot 13 \cdot 31 = 4836$ compacted trees.

A recurrence relation

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

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

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 $c_n = \gamma_{n,0}$.

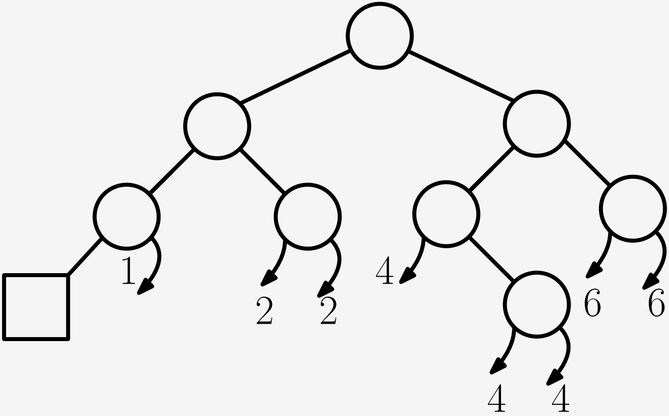
- Helps us to efficiently compute c_n
- Asymptotic analysis failed (so far)
  One reason: asymptotically every summand matters
- Summands possess 3 (!) dependencies on i

Relaxed compacted binary trees

Drop the condition of uniqueness of the subtrees, i.e. $c_n \leq r_n$.

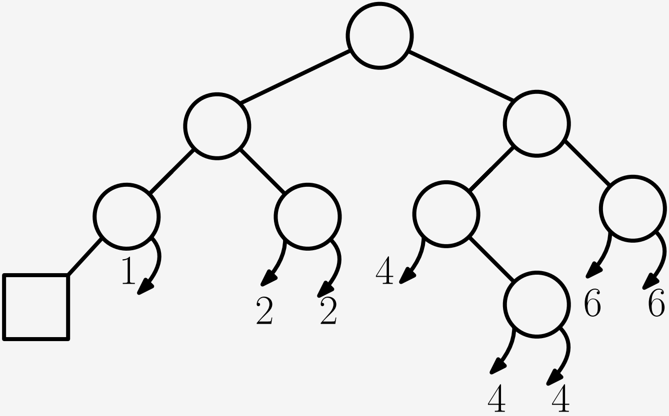
Relaxed compacted binary trees

Drop the condition of uniqueness of the subtrees, i.e. $c_n \leq r_n$.



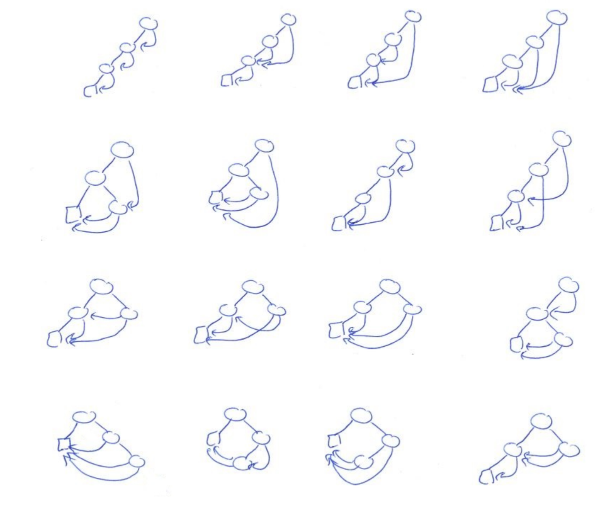
Relaxed compacted binary trees

Drop the condition of uniqueness of the subtrees, i.e. $c_n \leq r_n$.



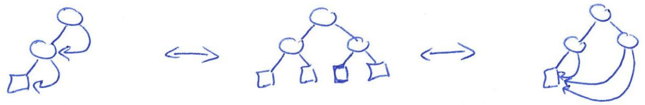
In total, this spine corresponds to $1 \cdot 3 \cdot 4 \cdot 4^2 \cdot 6^2 = 6912$ relaxed trees.
(Recall, that the same spine corresponds to 4836 compacted trees.)

Relaxed compacted binary trees of size 3



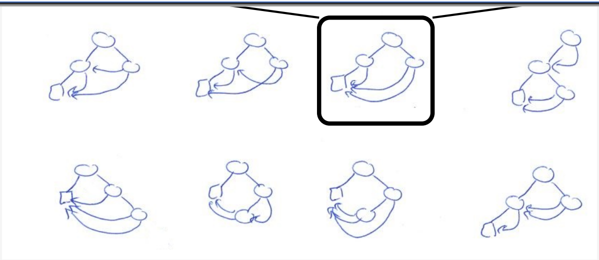
Relaxed compacted binary trees of size 3

The relaxed tree of size 3 which is not a compacted tree



compacted tree   binary tree   relaxed tree

Reason: subtrees not unique



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 $r_n = \delta_{n,0}$.

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

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

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

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

Operations on trees

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**.

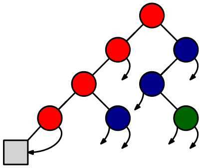
Bounded right height

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**.

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.

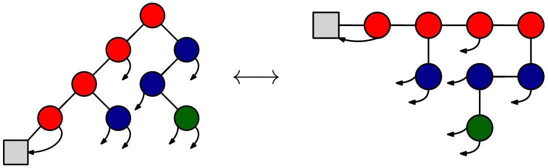
Bounded right height

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**.

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.

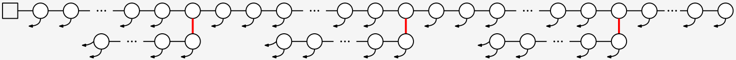


Figure: Right height ≤ 1.

Compacted trees of right height $\leq k$

Figure: Right height ≤ 0.

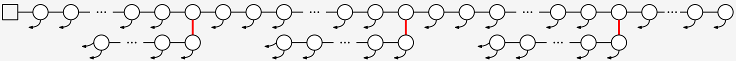


Figure: Right height ≤ 1.

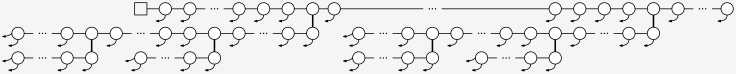


Figure: Right height ≤ 2.

Compacted trees of right height $\leq k$

Figure: Right height ≤ 0.

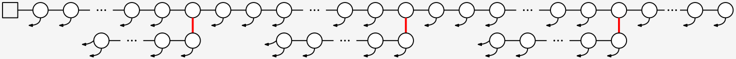


Figure: Right height ≤ 1.

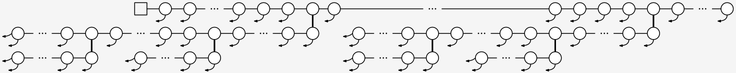


Figure: Right height ≤ 2.

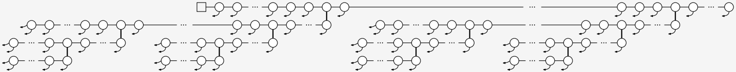


Figure: Right height ≤ 3.

Motivational outlook

Theorem

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

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

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

Motivational outlook

Theorem

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

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

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

Theorem (Main result)

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.

Main idea: Exponential generating functions

- Asymptotic growth: $n!\rho^n \Rightarrow$ exponential generating functions (EGF)
- Upper bound guarantees positive radius of convergence
- Problem: unlabeled structures!

Main idea: Exponential generating functions

- Asymptotic growth: $n!\rho^n \Rightarrow$ exponential generating functions (EGF)
- Upper bound guarantees positive radius of convergence
- Problem: unlabeled structures!

Main idea: Exponential generating functions

- Asymptotic growth: $n!\rho^n \Rightarrow$ exponential generating functions (EGF)
- Upper bound guarantees positive radius of convergence
- Problem: unlabeled structures!

Main idea: Exponential generating functions

- Asymptotic growth: $n!\rho^n \Rightarrow$ exponential generating functions (EGF)
- Upper bound guarantees positive radius of convergence
- Problem: unlabeled structures!
  Idea: **derive symbolic method for compacted trees**

Main idea: Exponential generating functions

- Asymptotic growth: $n!\rho^n \Rightarrow$ exponential generating functions (EGF)
- Upper bound guarantees positive radius of convergence
- Problem: unlabeled structures!
  Idea: **derive symbolic method for compacted trees**

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

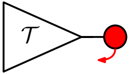
Main idea: Exponential generating functions

- Asymptotic growth: $n!\rho^n \Rightarrow$ exponential generating functions (EGF)
- Upper bound guarantees positive radius of convergence
- Problem: unlabeled structures!
  Idea: **derive 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 $\mathcal{T}$.



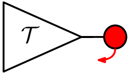
Main idea: Exponential generating functions

- Asymptotic growth: $n!\rho^n \Rightarrow$ exponential generating functions (EGF)
- Upper bound guarantees positive radius of convergence
- Problem: unlabeled structures!
  Idea: **derive 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:

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

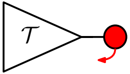
Main idea: Exponential generating functions

- Asymptotic growth: $n!\rho^n \Rightarrow$ exponential generating functions (EGF)
- Upper bound guarantees positive radius of convergence
- Problem: unlabeled structures!
  Idea: **derive 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:

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

Construction of $R_0(z)$

Let $R_0(z) = \sum_{n\geq 0} r_{0,n} \frac{z^n}{n!}$ be the EGF of relaxed binary trees with bounded right height ≤ 0.

Construction of $R_0(z)$

Let $R_0(z) = \sum_{n\geq 0} r_{0,n}\frac{z^n}{n!}$ be the EGF of relaxed binary trees with bounded right height ≤ 0.

$$\mathcal{R}_0 = \underbrace{\{\Box\}}_{\text{Tree of size 0}} \cup \underbrace{\{\circ\} \times \mathcal{R}_0}_{\substack{\text{append new root}\\ \text{and new pointer}}}$$

Construction of $R_0(z)$

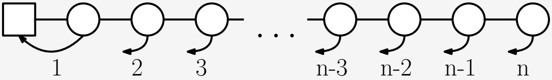
Let $R_0(z) = \sum_{n \geq 0} r_{0,n} \frac{z^n}{n!}$ be the EGF of relaxed binary trees with bounded right height ≤ 0.

$$\mathcal{R}_0 = \underbrace{\{\square\}}_{\text{Tree of size 0}} \cup \underbrace{\{\circ\} \times \mathcal{R}_0}_{\substack{\text{append new root} \\ \text{and new pointer}}}$$

$$R_0(z) = \frac{1}{1-z} = \sum_{n \geq 0} n! \frac{z^n}{n!}$$

Construction of $R_0(z)$

Let $R_0(z) = \sum_{n\geq 0} r_{0,n} \frac{z^n}{n!}$ be the EGF of relaxed binary trees with bounded right height ≤ 0.



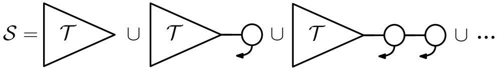
$$\mathcal{R}_0 = \underbrace{\{\square\}}_{\text{Tree of size 0}} \cup \underbrace{\{\circ\} \times \mathcal{R}_0}_{\substack{\text{append new root} \\ \text{and new pointer}}}$$

$$R_0(z) = \frac{1}{1-z} = \sum_{n\geq 0} n! \frac{z^n}{n!}$$

Further constructions

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

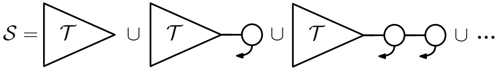
Append a (possibly empty) sequence at the root.



Further constructions

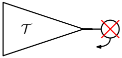
$S : T(z) \mapsto \frac{1}{1-z} T(z)$

Append a (possibly empty) sequence at the root.



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

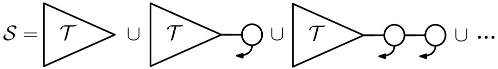
Delete top node but preserve its pointers.



Further constructions

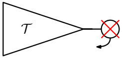
$S : T(z) \mapsto \frac{1}{1-z} T(z)$

Append a (possibly empty) sequence at the root.



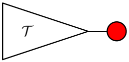
$D : T(z) \mapsto \frac{d}{dz} T(z)$

Delete top node but preserve its pointers.



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

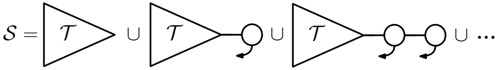
Add top node without pointers.



Further constructions

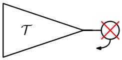
$S : T(z) \mapsto \frac{1}{1-z} T(z)$

Append a (possibly empty) sequence at the root.



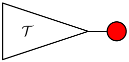
$D : T(z) \mapsto \frac{d}{dz} T(z)$

Delete top node but preserve its pointers.



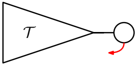
$I : T(z) \mapsto \int T(z)$

Add top node without pointers.



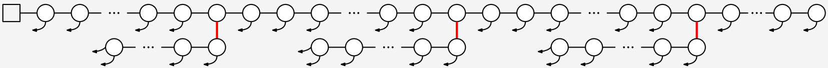
$P : T(z) \mapsto z \frac{d}{dz} T(z)$

Add a new pointer to the top node.



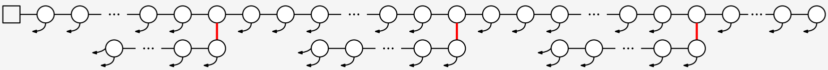
Relaxed binary trees

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 $R_1(z)$



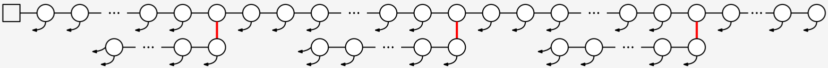
Let $R_1(z) = \sum_{\ell \geq 0} r_{1,n} \frac{z^n}{n!}$ be the EGF of relaxed binary trees with bounded right height ≤ 1.

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 $R_1(z)$

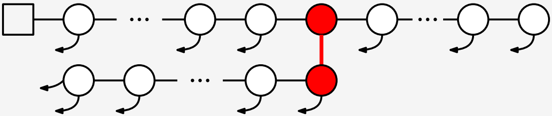
$$R_1(z) = \sum_{n \geq 0} R_{1,\ell}(z)$$

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.

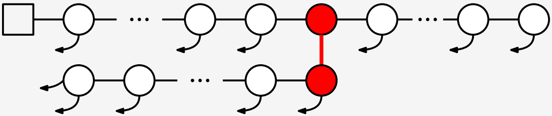
$$R_{1,0}(z) = R_0(z) = \frac{1}{1-z}$$

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

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

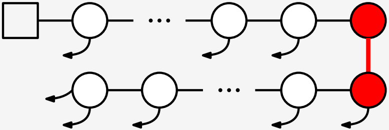


Construction of $R_{1,1}(z)$

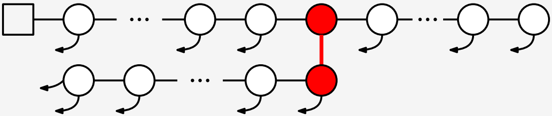


Symbolic specification

1 delete initial sequence

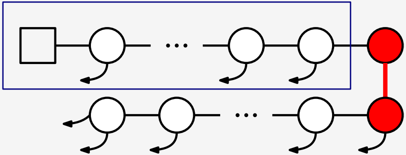


Construction of $R_{1,1}(z)$

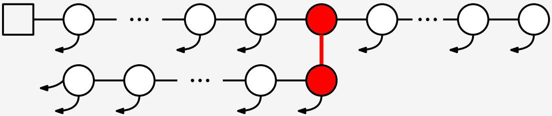


Symbolic specification

1. delete initial sequence
2. decompose

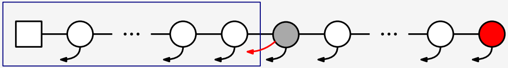


Construction of $R_{1,1}(z)$

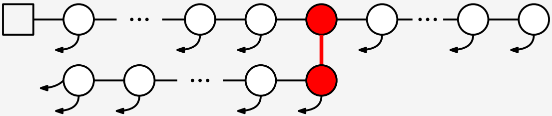


Symbolic specification

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

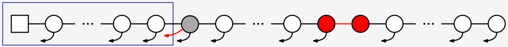


Construction of $R_{1,1}(z)$



Symbolic specification

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

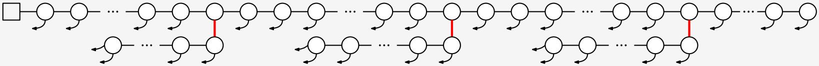


$R_{1,1}(z)$

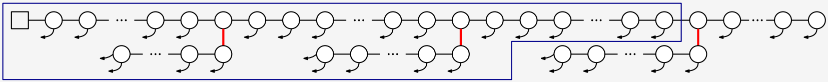
$$R_{1,1}(z) = \underbrace{S}_{\text{init. seq.}} \circ \underbrace{I}_{\text{lvl 0 node}} \circ \underbrace{S \circ P}_{\text{red pointer and seq.}} \left(\underbrace{zR_{1,0}(z)}_{\text{non empty}} \right)$$

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

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

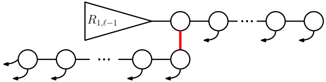


Construction of $R_{1,\ell}(z)$



Observation

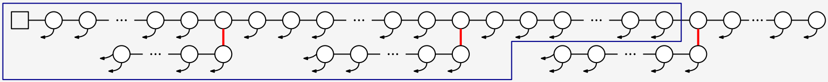
Same structure as for $R_{1,1}(z)$



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

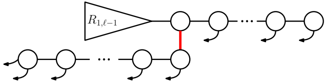
$$R_{1,0}(z) = R_0(z) = \frac{1}{1-z}.$$

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



Observation

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



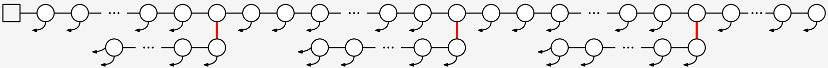
$$R_{1,\ell}(z) = \frac{1}{1-z} \int \frac{1}{1-z} z \left(z R_{1,\ell-1}(z)\right)' \, dz, \qquad \ell \geq 1,$$

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

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 $R_1(z)$

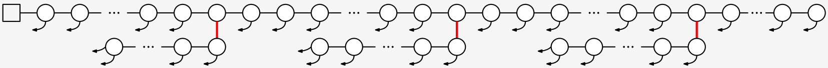


$$\frac{1-2z}{1-z} R_1'(z) - \frac{1}{1-z} R_1(z) - ((1-z)R_{1,0}(z))' = 0.$$

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 $R_1(z)$



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

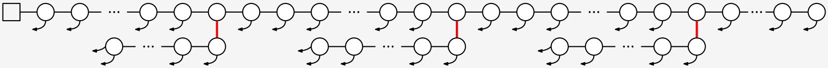
We know that $R_{1,0}(z) = \frac{1}{1-z}$ and get

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

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

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

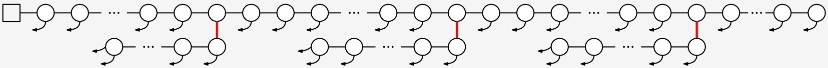
This directly yields

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

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

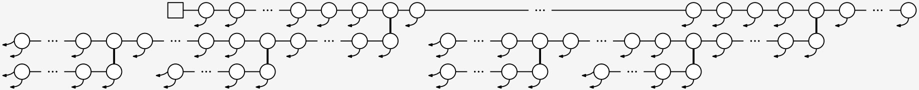
$$R_1(z) = \frac{1}{\sqrt{1-2z}}.$$

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"].

Bounded right height ≤ 2: $R_2(z)$



Symbolic construction

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

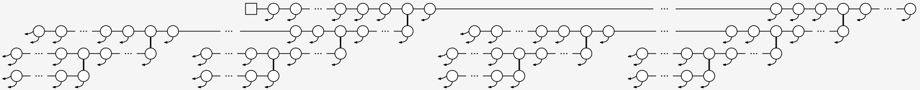
then we get the closed form

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

and the coefficients

$$r_{2,n} = n![z^n]R_2(z) = \frac{(n-1)!}{\sqrt{5}} \left(\left(\frac{3+\sqrt{5}}{2} \right)^n - \left(\frac{3-\sqrt{5}}{2} \right)^n \right).$$

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

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

Differential operators

Theorem

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

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

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

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

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

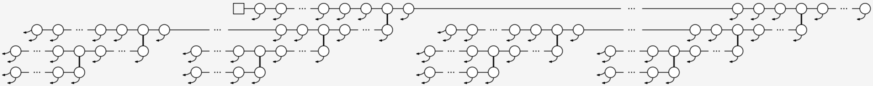
$$L_k \cdot R_k = 0.$$

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

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

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

Asymptotics of relaxed trees with bounded right height



Theorem

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

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

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

Sketch of proof

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 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

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 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

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 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

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 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

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 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

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 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.

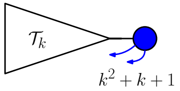
Sketch of proof

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 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.

Compacted binary trees

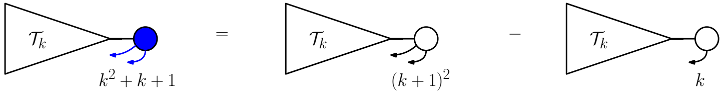
Compacted binary trees

Uniqueness of subtrees



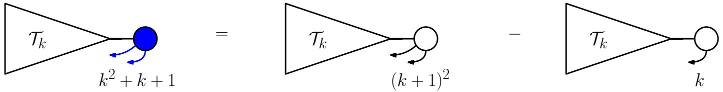
Compacted binary trees

Uniqueness of subtrees



Compacted binary trees

Uniqueness of subtrees

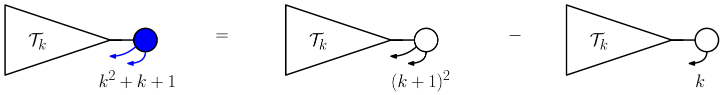


Let $(M_k)_{k\geq 0}$ be a family of differential operators such that the EGF $C_k(z)$ for compacted binary trees with right height $\leq k$ satisfies

$$M_k \cdot C_k = 0.$$

Compacted binary trees

Uniqueness of subtrees



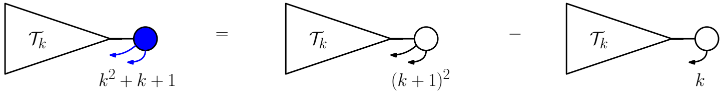
Let $(M_k)_{k\geq 0}$ be a family of differential operators such that the EGF $C_k(z)$ for compacted binary trees with right height $\leq k$ satisfies

$$M_k \cdot C_k = 0.$$

$$(1-2z)\frac{d^2}{dz^2}C_1(z) + (z-3)\frac{d}{dz}C_1(z) = 0,$$

Compacted binary trees

Uniqueness of subtrees



Let $(M_k)_{k\geq 0}$ be a family of differential operators such that the EGF $C_k(z)$ for compacted binary trees with right height $\leq k$ satisfies

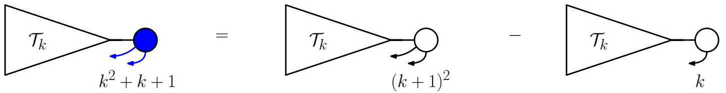
$$M_k \cdot C_k = 0.$$

$$(1-2z)\frac{d^2}{dz^2}C_1(z) + (z-3)\frac{d}{dz}C_1(z) = 0,$$

$$(z^2-3z+1)\frac{d^3}{dz^3}C_2(z) - (z^2-6z+6)\frac{d^2}{dz^2}C_2(z) - (2z-3)\frac{d}{dz}C_2(z) = 0,$$

Compacted binary trees

Uniqueness of subtrees



Let $(M_k)_{k\geq 0}$ be a family of differential operators such that the EGF $C_k(z)$ for compacted binary trees with right height $\leq k$ satisfies

$$M_k \cdot C_k = 0.$$

$$(1-2z)\frac{d^2}{dz^2}C_1(z) + (z-3)\frac{d}{dz}C_1(z) = 0,$$

$$(z^2-3z+1)\frac{d^3}{dz^3}C_2(z) - (z^2-6z+6)\frac{d^2}{dz^2}C_2(z) - (2z-3)\frac{d}{dz}C_2(z) = 0,$$

$$(3z^2-4z+1)\frac{d^4}{dz^4}C_3(z) - (4z^2-18z+10)\frac{d^3}{dz^3}C_3(z) + \cdots$$

$$\cdots + (z^2-12z+14)\frac{d^2}{dz^2}C_3(z) + (z-3)\frac{d}{dz}C_3(z) = 0.$$

Asymptotics of compacted trees with bounded right height

Theorem (Main result)

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.

Asymptotics of compacted trees with bounded right height

Theorem (Main result)

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:

- We derived a **symbolic method** on exponential generating functions,
- leading to **ordinary differential equations**, and
- analyzed them by **singularity analysis** (recurrence relations on polynomial coefficients, indicial polynomial, transfer theorems). □

Asymptotics of compacted trees with bounded right height

Theorem (Main result)

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:

- We derived a **symbolic method** on exponential generating functions,
- leading to **ordinary differential equations**, and
- analyzed them by **singularity analysis** (recurrence relations on polynomial coefficients, indicial polynomial, transfer theorems). □

Asymptotics of compacted trees with bounded right height

Theorem (Main result)

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:

- We derived a **symbolic method** on exponential generating functions,
- leading to **ordinary differential equations**, and
- analyzed them by **singularity analysis** (recurrence relations on polynomial coefficients, indicial polynomial, transfer theorems). □

Asymptotics of compacted trees with bounded right height

Theorem (Main result)

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:

- We derived a **symbolic method** on exponential generating functions,
- leading to **ordinary differential equations**, and
- analyzed them by **singularity analysis** (recurrence relations on polynomial coefficients, indicial polynomial, transfer theorems). □

Comparing compacted and relaxed trees

Compacted trees with right height at most k

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

Comparing compacted and relaxed trees

Compacted trees with right height at most k

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

Relaxed trees with right height at most k

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

Comparing compacted and relaxed trees

Compacted trees with right height at most k

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

Relaxed trees with right height at most k

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

Corollary (Proportion of compacted among relaxed trees)

$$\frac{c_{k,n}}{r_{k,n}}$$

Comparing compacted and relaxed trees

Compacted trees with right height at most k

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

Relaxed trees with right height at most k

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

Corollary (Proportion of compacted among relaxed trees)

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

Comparing compacted and relaxed trees

Compacted trees with right height at most k

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

Relaxed trees with right height at most k

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

Corollary (Proportion of compacted among relaxed trees)

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

Next steps

- Enumeration of compacted trees *without* height restrictions
- Different tree structures, like e.g. ternary trees
- Analyze shape parameters, like height, width, profile, ...

Next steps

- Enumeration of compacted trees *without* height restrictions
- Different tree structures, like e.g. ternary trees
- Analyze shape parameters, like height, width, profile, ...