A New Notion of Noncontiguous Containment for Ordered, Rooted Trees

Eric S. Egge

Carleton College

October 4, 2015

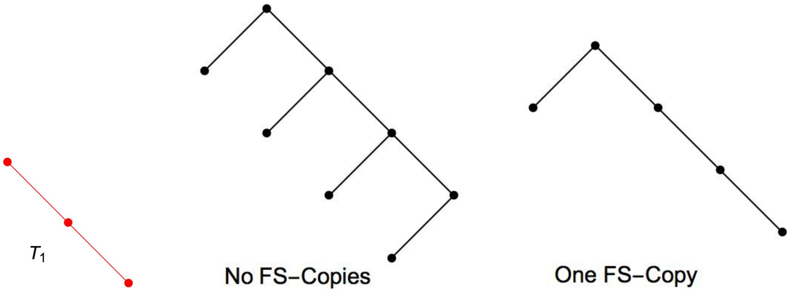
Outline

1. Containment Notions for Binary Trees
2. Enumerations Involving C-Containment and C-Avoidance
3. Connections with Pattern Avoidance in Permutations

FS-Containment

Flajolet and Sedgewick (2009):

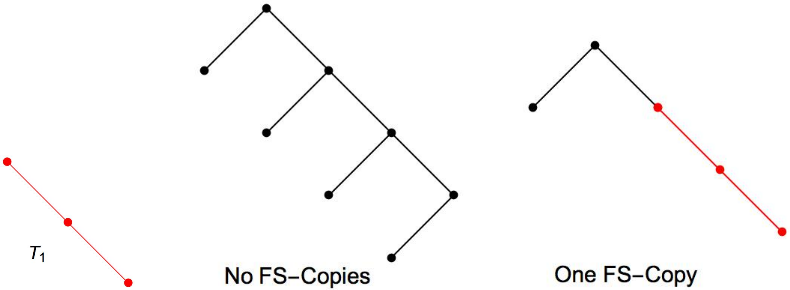
An FS-copy of T_1 in T_2 is a node in T_1 whose dangling subtree (including all of its children) is isomorphic to T_1.



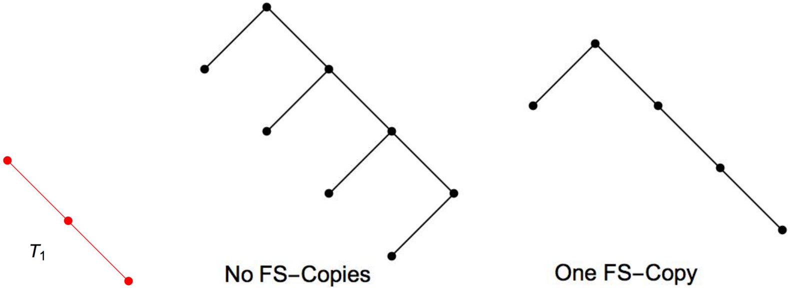
FS-Containment

Flajolet and Sedgewick (2009):

An FS-copy of T_1 in T_2 is a node in T_1 whose dangling subtree (including all of its children) is isomorphic to T_1.



FS-Containment



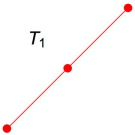
Entire subtree of chosen root must match.

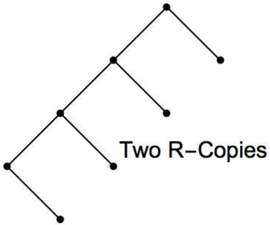
Removing vertices can create a copy.

R-Containment

Rowland (2010):

An R-copy of T_1 in T_2 is a connected copy of T_1 in T_2.



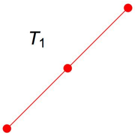


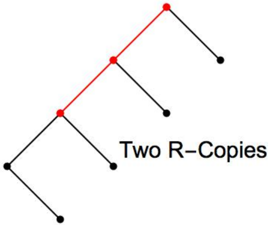
Adjacencies must match but the entire subtree might not.

R-Containment

Rowland (2010):

An R-copy of T_1 in T_2 is a connected copy of T_1 in T_2.



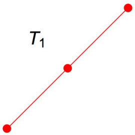


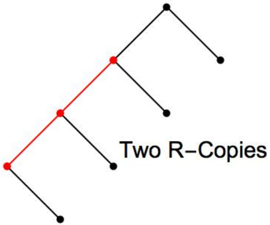
Adjacencies must match but the entire subtree might not.

R-Containment

Rowland (2010):

An R-copy of T_1 in T_2 is a connected copy of T_1 in T_2.





Adjacencies must match but the entire subtree might not.

P-Containment

Dairyko, Pudwell, Tyner, Wynn (2012) and Pudwell, Scholten, Schrock, and Serrato (2014):

For full trees (no node has exactly 1 child), a P-copy of T_1 in T_2 is a set E of edges in T_2 such that

- for each node v in T_2,
  both edges from v are in E or neither is,
  and

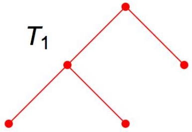
P-Containment

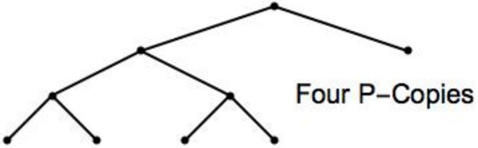
Dairyko, Pudwell, Tyner, Wynn (2012) and Pudwell, Scholten, Schrock, and Serrato (2014):

For full trees (no node has exactly 1 child), a P-copy of T_1 in T_2 is a set E of edges in T_2 such that

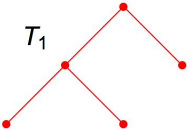
- for each node v in T_2,
  both edges from v are in E or neither is,
  and
- the tree left after contracting all edges not in E is isomorphic to T_1.

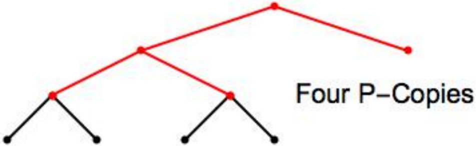
A P-Containment Example



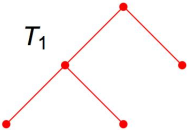


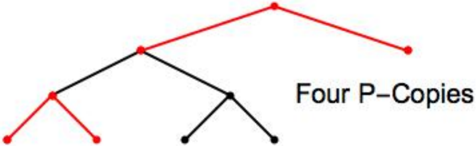
A P-Containment Example



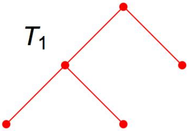


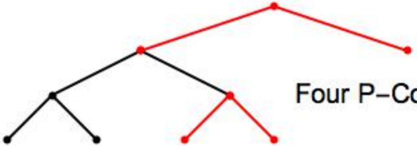
A P-Containment Example



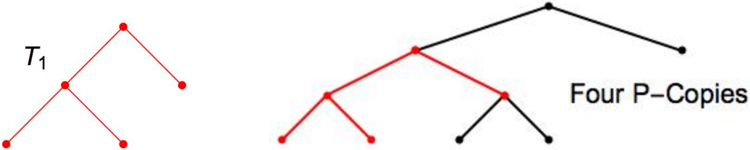


A P-Containment Example

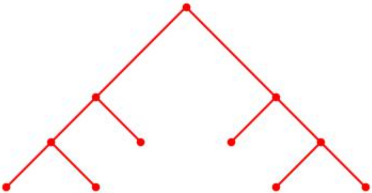




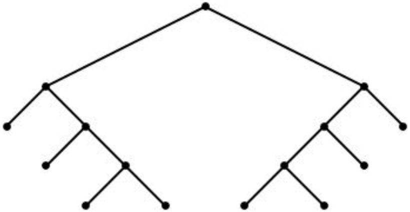
A P-Containment Example



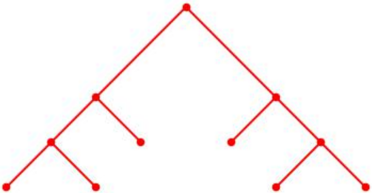
A P-Containment Nonexample



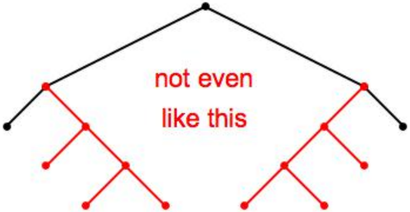
is not P-contained in



A P-Containment Nonexample



is not P-contained in



Definition

A binary tree is a finite set (of vertices or nodes) T together with relations $<_L$ (is a left descendant of) and $<_R$ (is a right descendant of) such that

C-Containment, Part I

Definition

A binary tree is a finite set (of vertices or nodes) T together with relations $<_L$ (is a left descendant of) and $<_R$ (is a right descendant of) such that

1. For all $t \in T$ we have $t \not< t$. ($<$ means $<_L$ or $<_R$.)

Definition

A binary tree is a finite set (of vertices or nodes) T together with relations $<_L$ (is a left descendant of) and $<_R$ (is a right descendant of) such that

1. For all $t \in T$ we have $t \not< t$. ($<$ means $<_L$ or $<_R$.)
2. There exists a unique $r \in T$ such that if $s \in T$ and $s \neq r$, then $s < r$. We call r the *root* of T.

Definition

A binary tree is a finite set (of vertices or nodes) T together with relations $<_L$ (is a left descendant of) and $<_R$ (is a right descendant of) such that

1. For all $t \in T$ we have $t \not< t$. ($<$ means $<_L$ or $<_R$.)
2. There exists a unique $r \in T$ such that if $s \in T$ and $s \neq r$, then $s < r$. We call r the *root* of T.
3. etcetera.

Definition

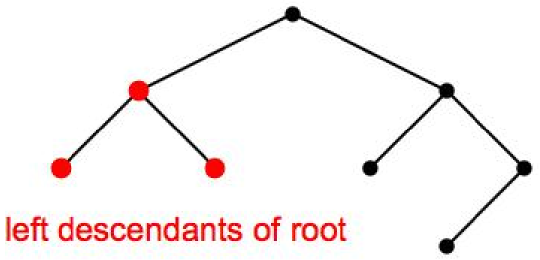
A binary tree is a finite set (of vertices or nodes) T together with relations $<_L$ (is a left descendant of) and $<_R$ (is a right descendant of) such that

1. For all $t \in T$ we have $t \not< t$. ($<$ means $<_L$ or $<_R$.)
2. There exists a unique $r \in T$ such that if $s \in T$ and $s \neq r$, then $s < r$. We call r the *root* of T.
3. For all $s, t, u \in T$, if $s < t$ and $t <_L u$ then $s <_L u$. Similarly, if $s < t$ and $t <_R u$ then $s <_R u$.
4. For all $s, t \in T$, at most one of the following holds: $s <_L t$, $s <_R t$, $t <_L s$, and $t <_R s$.
5. For all $s \in T$, if the set of all $t \in T$ with $t <_L s$ is nonempty, then there is a unique $u \in T$ such that $u <_L s$ and if $t \neq u$ has $t <_L s$ then $t < u$.
6. For all $s \in T$, if the set of all $t \in T$ with $t <_R s$ is nonempty, then there is a unique $u \in T$ such that $u <_R s$ and if $t \neq u$ has $t <_R s$ then $t < u$.

Definition

A binary tree is a finite set (of vertices or nodes) T together with relations $<_L$ (is a left descendant of) and $<_R$ (is a right descendant of) such that

1. For all $t \in T$ we have $t \not< t$. ($<$ means $<_L$ or $<_R$.)
2. There exists a unique $r \in T$ such that if $s \in T$ and $s \neq r$, then $s < r$. We call r the *root* of T.
3. etcetera.



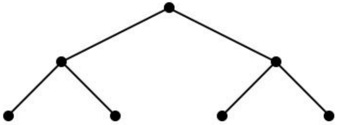
C-Containment, Part II

A C-copy of T_1 in T_2 is a set T of vertices in T_2 for which

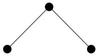
the restriction of $<_L$ and $<_R$ to T

is a binary tree isomorphic to T_1.

A C-copy of T_1 in T_2 is a set T of vertices in T_2 for which
the restriction of $<_L$ and $<_R$ to T
is a binary tree isomorphic to T_1.

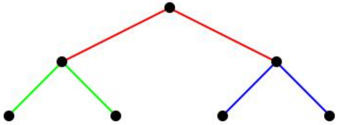


has ? P-copies of

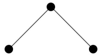


C-Containment, Part II

A C-copy of T_1 in T_2 is a set T of vertices in T_2 for which the restriction of $<_L$ and $<_R$ to T is a binary tree isomorphic to T_1.

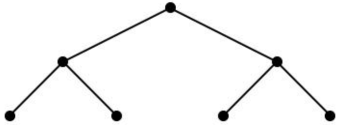


has 3 P-copies of

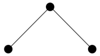


C-Containment, Part II

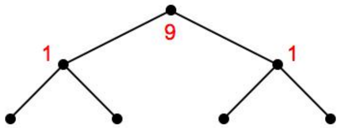
A C-copy of T_1 in T_2 is a set T of vertices in T_2 for which the restriction of $<_L$ and $<_R$ to T is a binary tree isomorphic to T_1.



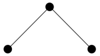
has ?? C-copies of



A C-copy of T_1 in T_2 is a set T of vertices in T_2 for which the restriction of $<_L$ and $<_R$ to T is a binary tree isomorphic to T_1.



has 11 C-copies of



C-Avoidance and P-Avoidance

avoids := contains no copy of

Theorem (Egge)

For any full binary trees T_1 and T_2, T_2 C-avoids T_1 if and only if T_2 P-avoids T_1.

C-Avoidance and P-Avoidance

avoids := contains no copy of

Theorem (Egge)

For any full binary trees T_1 and T_2, T_2 C-avoids T_1 if and only if T_2 P-avoids T_1.

Proof of contrapositive: In a P-copy, take the vertices with two kept edges going down and the vertices with no kept edges anywhere below.

C-Avoidance and P-Avoidance

avoids := contains no copy of

Theorem (Egge)

For any full binary trees T_1 and T_2, T_2 C-avoids T_1 if and only if T_2 P-avoids T_1.

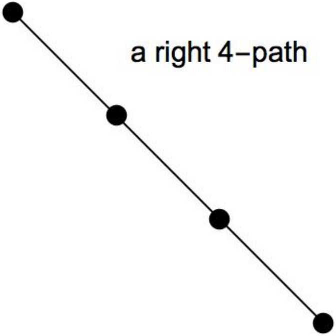
Proof of contrapositive: In a P-copy, take the vertices with two kept edges going down and the vertices with no kept edges anywhere below.

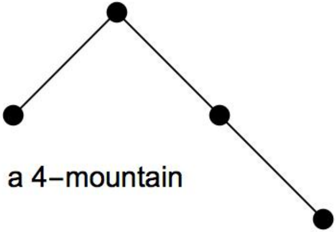
In a C-copy, move chosen non-root vertices with unchosen partners up until siblings are chosen in pairs and keep all edges up from nonroot chosen vertices.

Outline

1. Containment Notions for Binary Trees
2. Enumerations Involving C-Containment and C-Avoidance
3. Connections with Pattern Avoidance in Permutations

Paths and Mountains





C-Avoiding Paths

Theorem (Egge)

If $F_k(x)$ is the ogf for binary trees with n vertices C-avoiding a right k-path then

$$F_k(x) = \frac{U_{k-1}\left(\frac{1}{2\sqrt{x}}\right)}{\sqrt{x}U_k\left(\frac{1}{2\sqrt{x}}\right)},$$

where

$$U_k(x) = \sum_{k=0}^{\lfloor \frac{n}{2} \rfloor} (-1)^k \binom{n-k}{k} (2x)^{n-2k}$$

is the kth Chebyshev polynomial of the second kind.

C-Avoiding Mountains

Theorem (Egge)

For any k-mountain M, the number of binary trees with n vertices C-avoiding M is equal to the number of binary trees with n vertices C-avoiding a right k-path.

C-Avoiding Mountains

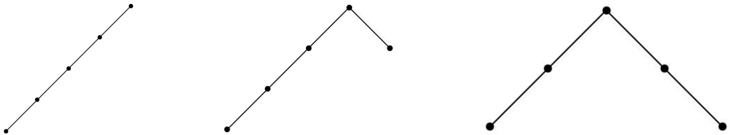
Theorem (Egge)

For any k-mountain M, the number of binary trees with n vertices C-avoiding M is equal to the number of binary trees with n vertices C-avoiding a right k-path.

Sketch of proof: compute the generating function recursively and use identities for the Chebyshev polynomials.

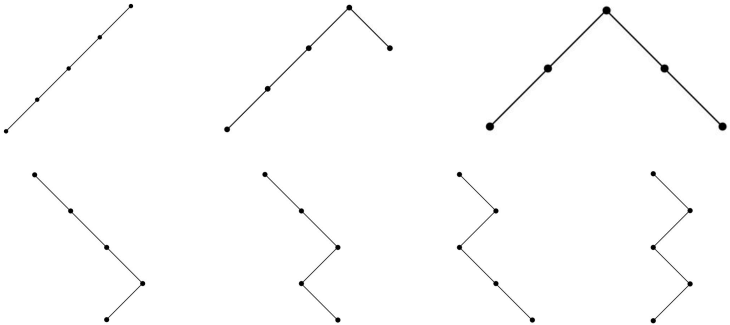
Every Tree is (Equivalent to) a Mountain

The same number of trees of each size avoid each of



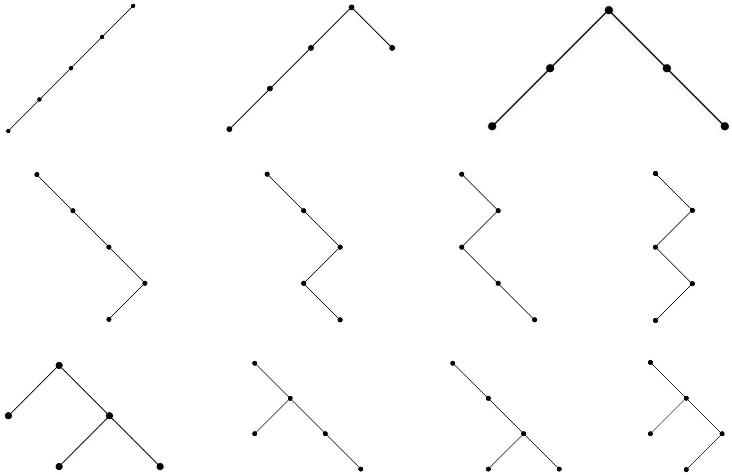
Every Tree is (Equivalent to) a Mountain

The same number of trees of each size avoid each of



Every Tree is (Equivalent to) a Mountain

The same number of trees of each size avoid each of



Every Tree is (Equivalent to) a Mountain

$f(T)$ is the foliation (fulliation?) of T

Every Tree is (Equivalent to) a Mountain

$f(T)$ is the foliation (fulliation?) of T

Theorem (Egge)

T_1 C-avoids T_2 if and only if $f(T_1)$ P-avoids $f(T_2)$.

Every Tree is (Equivalent to) a Mountain

$f(T)$ is the foliation (fulliation?) of T

Theorem (Egge)

T_1 C-avoids T_2 if and only if $f(T_1)$ P-avoids $f(T_2)$.

Theorem (Dairyko, Pudwell, Tyner, and Wynn)

The number of full binary trees with $2n+1$ vertices which P-avoid T depends only on the number of vertices in T.

Counting C-Copies of Complete Trees

Problem: Given a binary tree, how many C-copies of the complete binary tree with k levels does it contain?

Counting C-Copies of Complete Trees

Problem: Given a binary tree, how many C-copies of the complete binary tree with k levels does it contain?

Theorem (Egge)

There are exactly

$$2 \cdot 4^k - (2k+1)2^k - 1$$

copies of the complete binary tree with two levels (and three vertices) in the complete binary tree with k levels.

C-Copies and Continued Fractions

$\tau_k(T)$ is the number of C-copies of the right k-path in T

C-Copies and Continued Fractions

$\tau_k(T)$ is the number of C-copies of the right k-path in T

Theorem (Egge)

$$\sum_{T \text{ binary tree}} \prod_{k \geq 1} x_k^{\tau_k(T)} =$$

C-Copies and Continued Fractions

$\tau_k(T)$ is the number of C-copies of the right k-path in T

Theorem (Egge)

$$\sum_{T \text{ binary tree}} \prod_{k \geq 1} x_k^{\tau_k(T)} = \cfrac{1}{1 - \cfrac{x_1}{1 - \cfrac{x_1 x_2}{1 - \cfrac{x_1 x_2^2 x_3}{1 - \cfrac{x_1 x_2^3 x_3^3 x_4}{1 - \cfrac{x_1 x_2^4 x_3^6 x_4^4 x_5}{1 - \cdots}}}}}}.$$

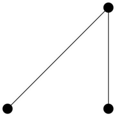
Observation: C-avoidance generalizes to ternary trees, ordered trees, etc.

Ternary Trees and C-Avoidance

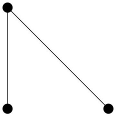
Observation: C-avoidance generalizes to ternary trees, ordered trees, etc.

Theorem (Egge)

The number of ternary trees with n vertices C-avoiding both



and



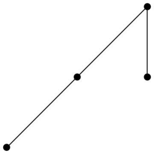
is the number of Motzkin paths with $n-1$ steps in which each level step is one of three colors.

Ternary Trees and C-Avoidance

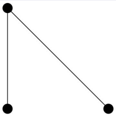
Observation: C-avoidance generalizes to ternary trees, ordered trees, etc.

Theorem (Egge)

The number of ternary trees with n vertices C-avoiding both



and



is

$$\frac{1}{n}\sum_{k=0}^{n}\binom{2k+1}{k}\binom{2n}{n-k}\frac{k}{2k+1}.$$

Outline

1. Containment Notions for Binary Trees
2. Enumerations Involving C-Containment and C-Avoidance
3. Connections with Pattern Avoidance in Permutations

Binary Trees and 231-Avoiding Permutations

Fact: There is a bijection between binary trees with n vertices and 231-avoiding permutations on $1, 2, \ldots, n$.

Starting with a tree,

Binary Trees and 231-Avoiding Permutations

Fact: There is a bijection between binary trees with n vertices and 231-avoiding permutations on $1, 2, \ldots, n$.

Starting with a tree,

- label the (current) root with the largest available number

Binary Trees and 231-Avoiding Permutations

Fact: There is a bijection between binary trees with n vertices and 231-avoiding permutations on $1, 2, \ldots, n$.

Starting with a tree,

- label the (current) root with the largest available number
- send smaller numbers to the left, larger numbers to the right, and label recursively

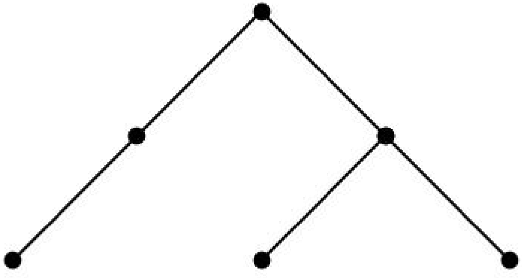
Binary Trees and 231-Avoiding Permutations

Fact: There is a bijection between binary trees with n vertices and 231-avoiding permutations on $1, 2, \ldots, n$.

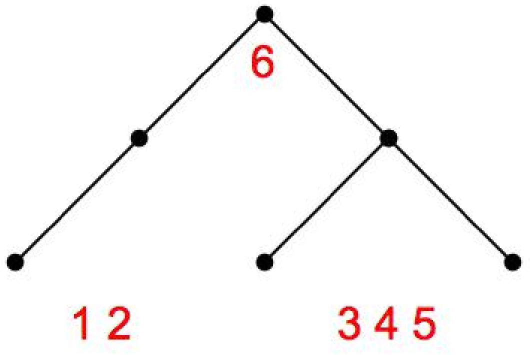
Starting with a tree,

- label the (current) root with the largest available number
- send smaller numbers to the left, larger numbers to the right, and label recursively
- once every vertex is labelled
    - list the labels in the left subtree recursively
    - list the root's label
    - list the labels in the right subtree recursively.

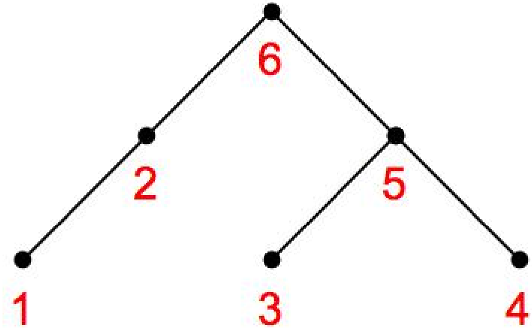
The Bijection in Action



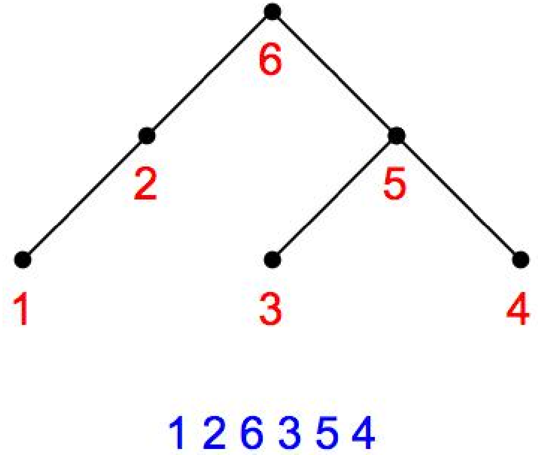
The Bijection in Action



The Bijection in Action



The Bijection in Action



The Permutations for Full Trees

Theorem (Egge)

Under our bijection the full binary trees are in bijection with 231-avoiding permutations which

- *start with an ascent*
- *end with a descent and*
- *have alternating ascents and descents.*

C-Avoidance in Trees and Pattern Avoidance in Permutations

Theorem (Egge)

T_1 and T_2 are binary trees with associated permutations π_1 and π_2. If π_1 avoids π_2 then T_1 C-avoids T_2.

C-Avoidance in Trees and Pattern Avoidance in Permutations

Theorem (Egge)

T_1 and T_2 are binary trees with associated permutations π_1 and π_2. If π_1 avoids π_2 then T_1 C-avoids T_2.

The converse is false.

C-Avoidance in Trees and Pattern Avoidance in Permutations

Theorem (Egge)

T_1 and T_2 are binary trees with associated permutations π_1 and π_2. If π_1 avoids π_2 then T_1 C-avoids T_2.

The converse is false.

Problem: find conditions under which the converse holds.

The Last Slide

Thank you!