Preimages under the Bubblesort operator

Luca Ferrari

Dipartimento di Matematica e Informatica “U. Dini”, Universitá degli Studi di Firenze,
Viale Morgagni 65, 50134 Firenze, Italy
luca.ferrari@unifi.it

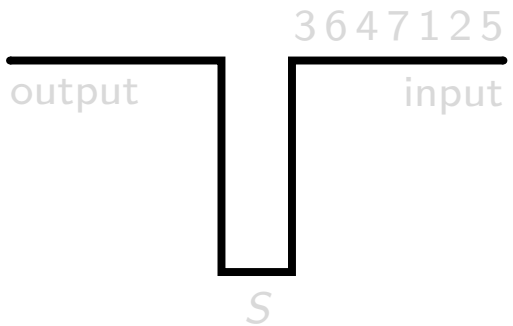
Permutation Patterns 2022, Valparaiso University, 20-24 June 2022.

Joint work with Mathilde Bouvel and Lapo Cioni

Three sorting algorithms

There are several sorting algorithms operating by sorting in phases.

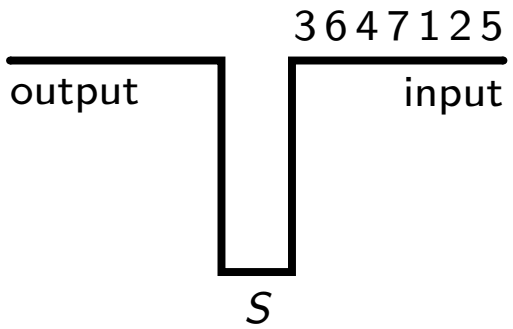
Stacksort



Three sorting algorithms

There are several sorting algorithms operating by sorting in phases.

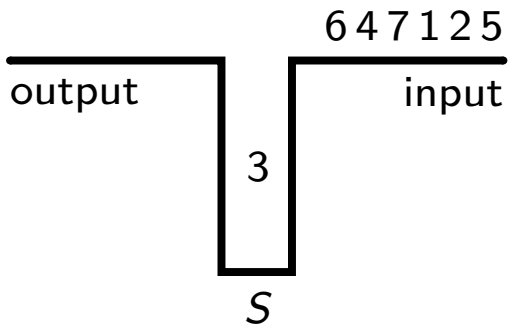
`Stacksort`



Three sorting algorithms

There are several sorting algorithms operating by sorting in phases.

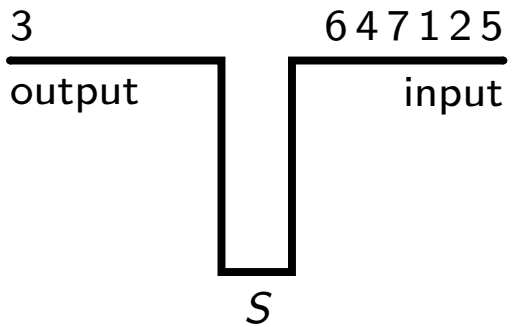
`Stacksort`



Three sorting algorithms

There are several sorting algorithms operating by sorting in phases.

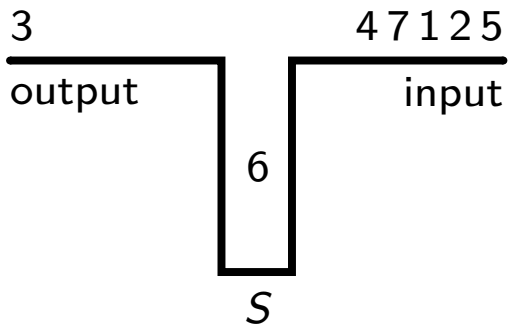
`Stacksort`



Three sorting algorithms

There are several sorting algorithms operating by sorting in phases.

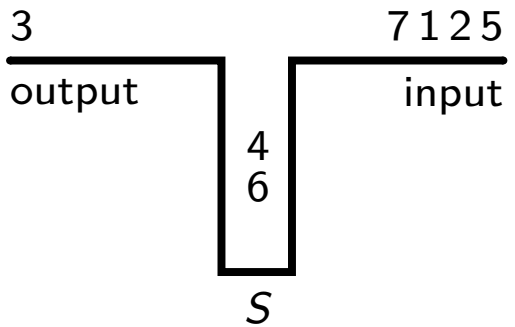
`Stacksort`



Three sorting algorithms

There are several sorting algorithms operating by sorting in phases.

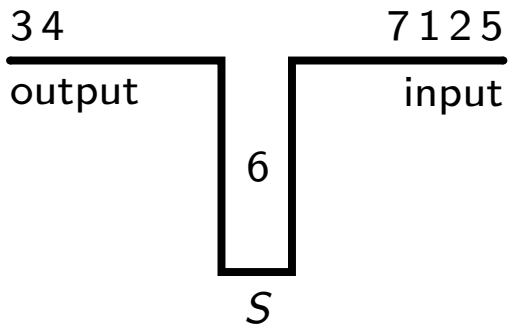
`Stacksort`



Three sorting algorithms

There are several sorting algorithms operating by sorting in phases.

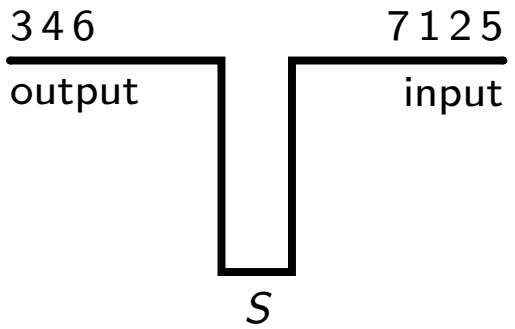
`Stacksort`



Three sorting algorithms

There are several sorting algorithms operating by sorting in phases.

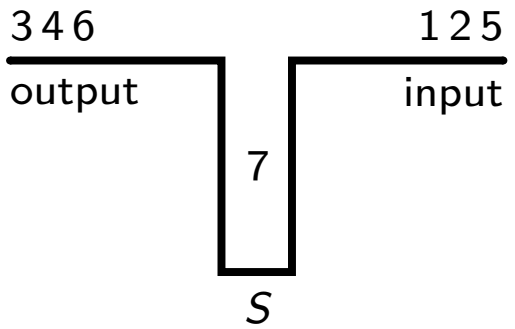
`Stacksort`



Three sorting algorithms

There are several sorting algorithms operating by sorting in phases.

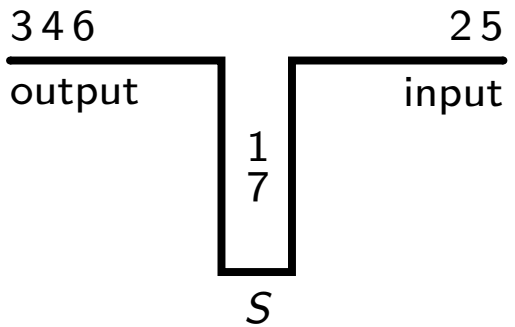
`Stacksort`



Three sorting algorithms

There are several sorting algorithms operating by sorting in phases.

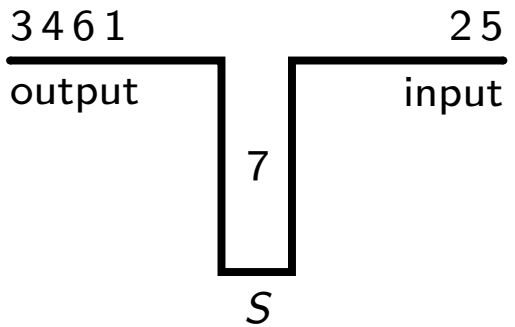
`Stacksort`



Three sorting algorithms

There are several sorting algorithms operating by sorting in phases.

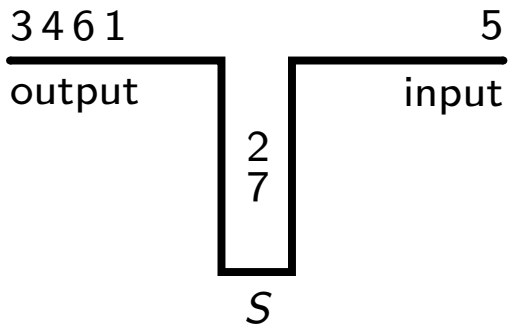
`Stacksort`



Three sorting algorithms

There are several sorting algorithms operating by sorting in phases.

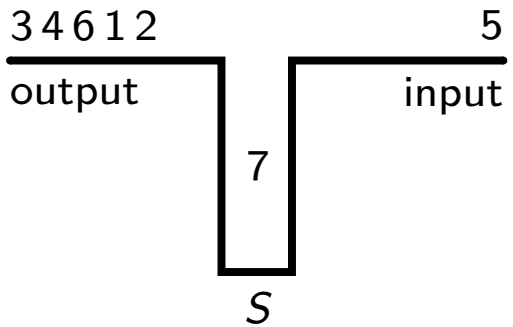
`Stacksort`



Three sorting algorithms

There are several sorting algorithms operating by sorting in phases.

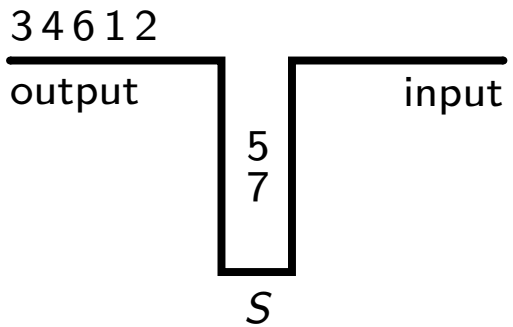
`Stacksort`



Three sorting algorithms

There are several sorting algorithms operating by sorting in phases.

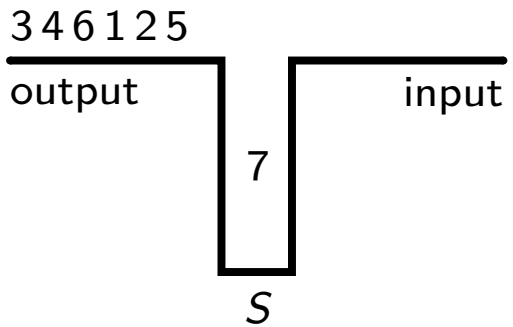
`Stacksort`



Three sorting algorithms

There are several sorting algorithms operating by sorting in phases.

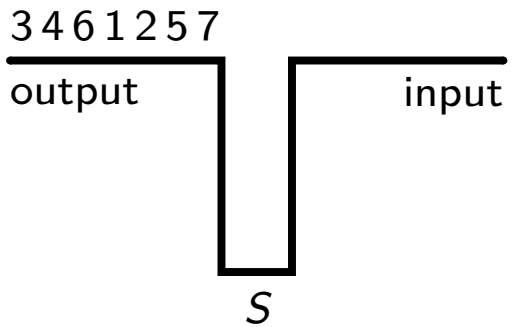
`Stacksort`



Three sorting algorithms

There are several sorting algorithms operating by sorting in phases.

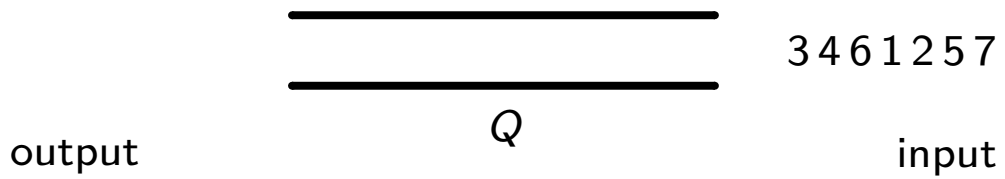
`Stacksort`



Three sorting algorithms

There are several sorting algorithms operating by sorting in phases.

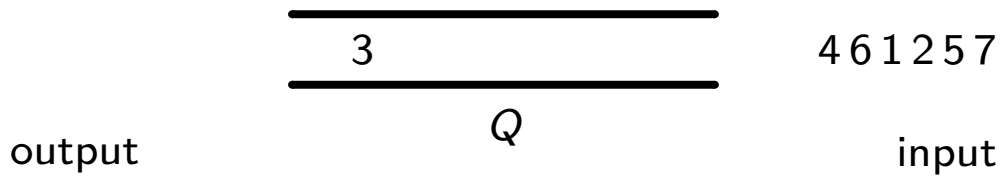
`Queuesort`



Three sorting algorithms

There are several sorting algorithms operating by sorting in phases.

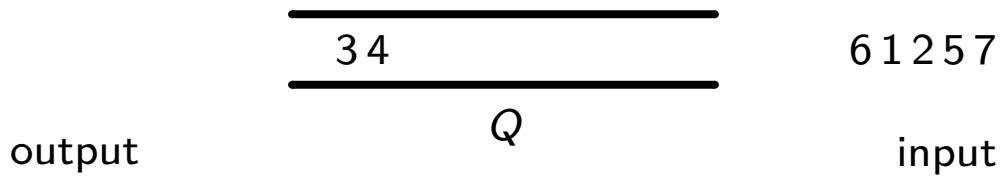
`Queuesort`



Three sorting algorithms

There are several sorting algorithms operating by sorting in phases.

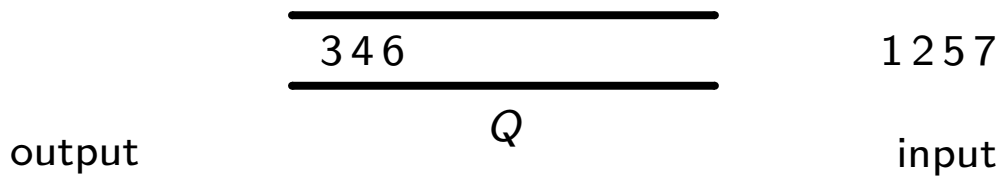
`Queuesort`



Three sorting algorithms

There are several sorting algorithms operating by sorting in phases.

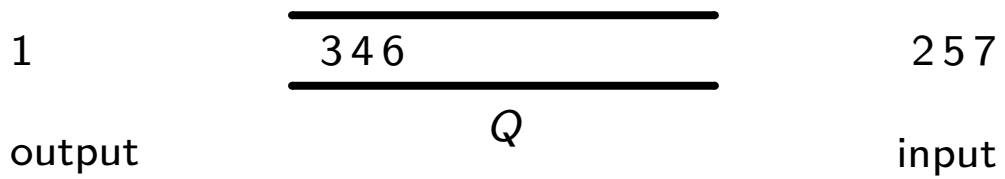
`Queuesort`



Three sorting algorithms

There are several sorting algorithms operating by sorting in phases.

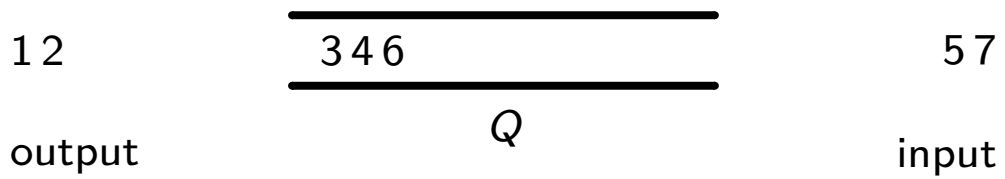
`Queuesort`



Three sorting algorithms

There are several sorting algorithms operating by sorting in phases.

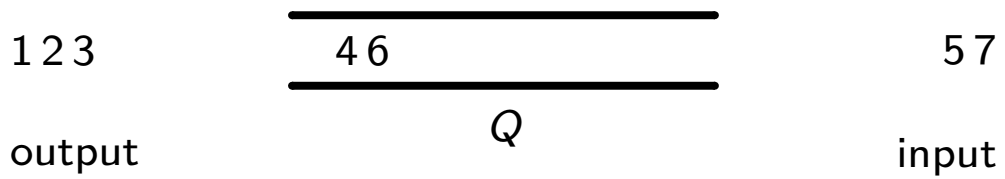
`Queuesort`



Three sorting algorithms

There are several sorting algorithms operating by sorting in phases.

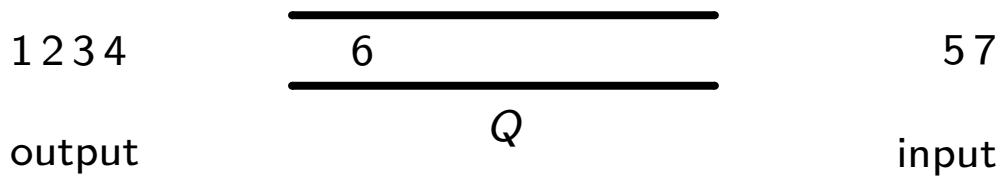
`Queuesort`



Three sorting algorithms

There are several sorting algorithms operating by sorting in phases.

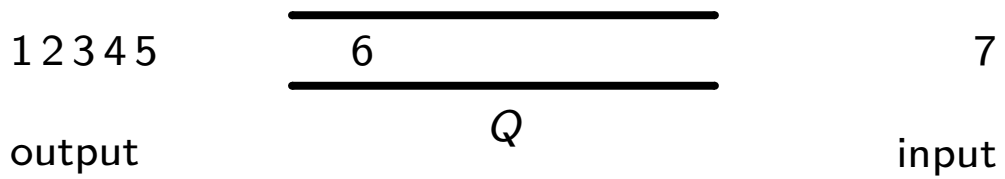
`Queuesort`



Three sorting algorithms

There are several sorting algorithms operating by sorting in phases.

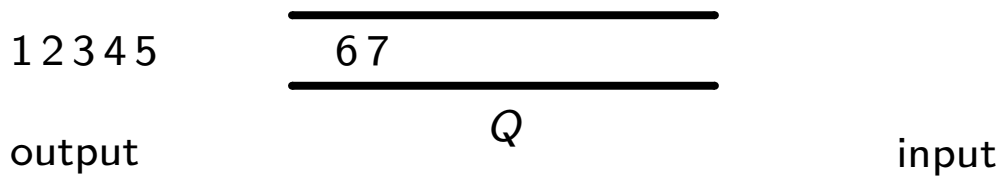
`Queuesort`



Three sorting algorithms

There are several sorting algorithms operating by sorting in phases.

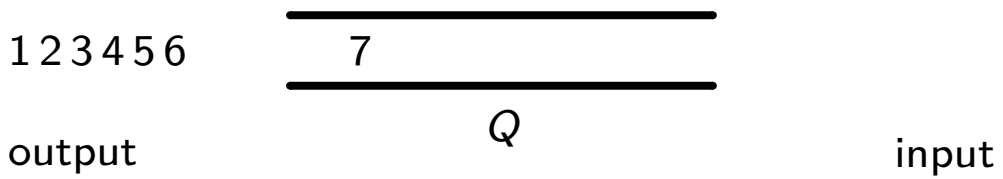
`Queuesort`



Three sorting algorithms

There are several sorting algorithms operating by sorting in phases.

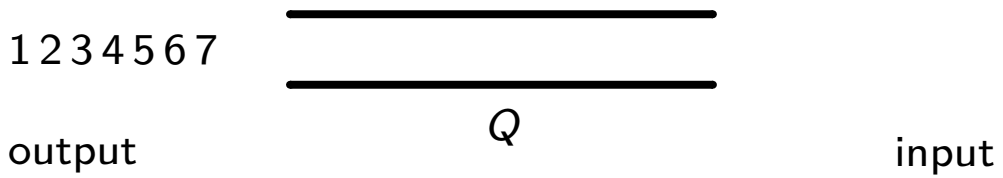
`Queuesort`



Three sorting algorithms

There are several sorting algorithms operating by sorting in phases.

`Queuesort`



Three sorting algorithms

There are several sorting algorithms operating by sorting in phases.

`Bubblesort`

3 4 6 1 2 5 7

Three sorting algorithms

There are several sorting algorithms operating by sorting in phases.

`Bubblesort`

3 4 6 1 2 5 7

Three sorting algorithms

There are several sorting algorithms operating by sorting in phases.

`Bubblesort`

3 4 6 1 2 5 7

Three sorting algorithms

There are several sorting algorithms operating by sorting in phases.

`Bubblesort`

3 4 1 6 2 5 7

Three sorting algorithms

There are several sorting algorithms operating by sorting in phases.

`Bubblesort`

3 4 1 2 6 5 7

Three sorting algorithms

There are several sorting algorithms operating by sorting in phases.

`Bubblesort`

3 4 1 2 5 6 7

Preimages of permutation classes

What can we say about preimages of permutation classes?

- `Stacksort`: West [1990], Claesson and Ulfarsson [2012], Defant [2020]
- `Queuesort`: Magnusson [2013]
- `Bubblesort`: Albert, Atkinson, Bouvel, Claesson, Dukes [2011]

Preimages of permutation classes

What can we say about preimages of permutation classes?

- `Stacksort`: West [1990], Claesson and Ulfarsson [2012], Defant [2020]
- `Queuesort`: Magnusson [2013]
- `Bubblesort`: Albert, Atkinson, Bouvel, Claesson, Dukes [2011]

Preimages of permutation classes

What can we say about preimages of permutation classes?

- `Stacksort`: West [1990], Claesson and Ulfarsson [2012], Defant [2020]
- `Queuesort`: Magnusson [2013]
- `Bubblesort`: Albert, Atkinson, Bouvel, Claesson, Dukes [2011]

Preimages under Stacksort

What can we say about the preimage of a given permutation?

Stacksort:

- first studied by Bousquet-Mélou [2000]
  - algorithm to compute preimages,
  - functional equation for the g.f. of permutations with at least one preimage,
- recent improvements by Defant et al. [2020-2022]
  - enumeration of permutations having exactly one preimage,
  - enumeration of permutation having k preimages is still open for all $k \geq 2$,
  - possible cardinalities of preimages (full characterization of such cardinalities is still open),
  - consequences for the enumeration of t-stack sortable permutations.

Preimages under Stacksort

What can we say about the preimage of a given permutation?

Stacksort:

- first studied by Bousquet-Mélou [2000]
  - algorithm to compute preimages;
  - functional equation for the g.f. of permutations with at least one preimage
- recent improvements by Defant et al. [2020-2022]
  - enumeration of permutations having exactly one preimage (enumeration of permutation having k preimages is still open for all $k \geq 2$);
  - possible cardinalities of preimages (full characterization of such cardinalities is still open);
  - consequences for the enumeration of t-stack sortable permutations.

Preimages under Stacksort

What can we say about the preimage of a given permutation?

Stacksort:

- first studied by Bousquet-Mélou [2000]
  - algorithm to compute preimages;
  - functional equation for the g.f. of permutations with at least one preimage
- recent improvements by Defant et al. [2020-2022]
  - enumeration of permutations having exactly one preimage (enumeration of permutation having k preimages is still open for all $k \geq 2$);
  - possible cardinalities of preimages (full characterization of such cardinalities is still open);
  - consequences for the enumeration of t-stack sortable permutations.

Preimages under Stacksort

What can we say about the preimage of a given permutation?

Stacksort:

- first studied by Bousquet-Mélou [2000]
  - algorithm to compute preimages;
  - functional equation for the g.f. of permutations with at least one preimage
- recent improvements by Defant et al. [2020-2022]
  - enumeration of permutations having exactly one preimage (enumeration of permutation having k preimages is still open for all $k \geq 2$);
  - possible cardinalities of preimages (full characterization of such cardinalities is still open);
  - consequences for the enumeration of t-stack sortable permutations.

Preimages under Stacksort

What can we say about the preimage of a given permutation?

Stacksort:

- first studied by Bousquet-Mélou [2000]
  - algorithm to compute preimages;
  - functional equation for the g.f. of permutations with at least one preimage
- recent improvements by Defant et al. [2020-2022]
  - enumeration of permutations having exactly one preimage (enumeration of permutation having k preimages is still open for all $k \geq 2$);
  - possible cardinalities of preimages (full characterization of such cardinalities is still open);
  - consequences for the enumeration of t-stack sortable permutations.

Preimages under Stacksort

What can we say about the preimage of a given permutation?

Stacksort:

- first studied by Bousquet-Mélou [2000]
  - algorithm to compute preimages;
  - functional equation for the g.f. of permutations with at least one preimage
- recent improvements by Defant et al. [2020-2022]
  - enumeration of permutations having exactly one preimage (enumeration of permutation having k preimages is still open for all $k \geq 2$);
  - possible cardinalities of preimages (full characterization of such cardinalities is still open);
  - consequences for the enumeration of t-stack sortable permutations.

Preimages under Stacksort

What can we say about the preimage of a given permutation?

Stacksort:

- first studied by Bousquet-Mélou [2000]
  - algorithm to compute preimages;
  - functional equation for the g.f. of permutations with at least one preimage
- recent improvements by Defant et al. [2020-2022]
  - enumeration of permutations having exactly one preimage (enumeration of permutation having k preimages is still open for all $k \geq 2$);
  - possible cardinalities of preimages (full characterization of such cardinalities is still open);
  - consequences for the enumeration of t-stack sortable permutations.

Preimages under Stacksort

What can we say about the preimage of a given permutation?

Stacksort:

- first studied by Bousquet-Mélou [2000]
  - algorithm to compute preimages;
  - functional equation for the g.f. of permutations with at least one preimage
- recent improvements by Defant et al. [2020-2022]
  - enumeration of permutations having exactly one preimage (enumeration of permutation having k preimages is still open for all $k \geq 2$);
  - possible cardinalities of preimages (full characterization of such cardinalities is still open);
  - consequences for the enumeration of t-stack sortable permutations.

Preimages under `Queuesort`

What can we say about the preimage of a given permutation?

`Queuesort`:

- studied by Cioni and F. [2021]
  - algorithm to compute preimages;
  - enumeration of permutations having exactly 0,1,and 2 preimages;
  - complete characterization of possible cardinalities of preimages;
  - enumeration of preimages in some special cases.

Preimages under `Queuesort`

What can we say about the preimage of a given permutation?

`Queuesort`:

- studied by Cioni and F. [2021]
  - algorithm to compute preimages;
  - enumeration of permutations having exactly 0,1,and 2 preimages;
  - complete characterization of possible cardinalities of preimages;
  - enumeration of preimages in some special cases.

Preimages under `Queuesort`

What can we say about the preimage of a given permutation?

`Queuesort`:

- studied by Cioni and F. [2021]
  - algorithm to compute preimages;
  - enumeration of permutations having exactly 0,1,and 2 preimages;
  - complete characterization of possible cardinalities of preimages;
  - enumeration of preimages in some special cases.

Preimages under Queuesort

What can we say about the preimage of a given permutation?

Queuesort:

- studied by Cioni and F. [2021]
  - algorithm to compute preimages;
  - enumeration of permutations having exactly 0,1,and 2 preimages;
  - complete characterization of possible cardinalities of preimages;
  - enumeration of preimages in some special cases.

Preimages under `Queuesort`

What can we say about the preimage of a given permutation?

`Queuesort`:

- studied by Cioni and F. [2021]
  - algorithm to compute preimages;
  - enumeration of permutations having exactly 0,1,and 2 preimages;
  - complete characterization of possible cardinalities of preimages;
  - enumeration of preimages in some special cases.

Preimages under Bubblesort

What can we say about the preimage of a given permutation?

Bubblesort:

This talk!

An algorithm to compute preimages

$\sigma = 325146$

$$P = \{325146\}$$

for i from n down to 2:
for each $\pi \in P$:

- if π_{i-1} is not a LTR maximum, then replace π with the permutation obtained by swapping π_i and π_{i-1}
- if π_{i-1} is a LTR maximum, then add to P the permutation obtained by swapping π_i and π_{i-1}

An algorithm to compute preimages

$\sigma = 325146$

$$P = \{3\,2\,5\,1\,4\,6\}$$

for i from n down to 2:
for each $\pi \in P$:

- if π_{i-1} is not a LTR maximum, then replace π with the permutation obtained by swapping π_i and π_{i-1}
- if π_{i-1} is a LTR maximum, then add to P the permutation obtained by swapping π_i and π_{i-1}

An algorithm to compute preimages

$\sigma = 325146$

$$P = \{3\,2\,5\,1\,6\,4\}$$

for i from n down to 2:
for each $\pi \in P$:

- if π_{i-1} is not a LTR maximum, then replace π with the permutation obtained by swapping π_i and π_{i-1}
- if π_{i-1} is a LTR maximum, then add to P the permutation obtained by swapping π_i and π_{i-1}

An algorithm to compute preimages

$\sigma = 325146$

$$P = \{3\,2\,5\,6\,1\,4\}$$

for i from n down to 2:
for each $\pi \in P$:

- if π_{i-1} is not a LTR maximum, then replace π with the permutation obtained by swapping π_i and π_{i-1}
- if π_{i-1} is a LTR maximum, then add to P the permutation obtained by swapping π_i and π_{i-1}

An algorithm to compute preimages

$\sigma = 325146$

$$P = \{325614, 326514\}$$

for i from n down to 2:
for each $\pi \in P$:

- if π_{i-1} is not a LTR maximum, then replace π with the permutation obtained by swapping π_i and π_{i-1}
- if π_{i-1} is a LTR maximum, then add to P the permutation obtained by swapping π_i and π_{i-1}

An algorithm to compute preimages

$\sigma = 325146$

$$P = \{352614, 362514\}$$

for i from n down to 2:
for each $\pi \in P$:

- if π_{i-1} is not a LTR maximum, then replace π with the permutation obtained by swapping π_i and π_{i-1}
- if π_{i-1} is a LTR maximum, then add to P the permutation obtained by swapping π_i and π_{i-1}

An algorithm to compute preimages

$\sigma = 325146$

$$P = \{352614, 532614, 362514, 632514\}$$

for i from n down to 2:
for each $\pi \in P$:

- if π_{i-1} is not a LTR maximum, then replace π with the permutation obtained by swapping π_i and π_{i-1}
- if π_{i-1} is a LTR maximum, then add to P the permutation obtained by swapping π_i and π_{i-1}

Theorem

Let σ be any permutation ending with its maximum. Then $P = \mathbf{B}^{-1}(\sigma)$.

Counting preimages

A preimage of $\sigma \in S_n$ is uniquely determined by a subset of its LTR maxima other than n, that can be interpreted as "guards".

$$3|25|146$$

$$3|52|614$$

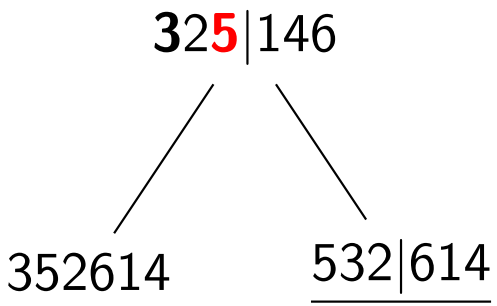
Counting preimages

A preimage of $\sigma \in S_n$ is uniquely determined by a subset of its LTR maxima other than n, that can be interpreted as "guards".

$$\begin{array}{c} \mathbf{3}|2\mathbf{5}|146 \\ \Big| \\ \underline{3|52|614} \end{array}$$

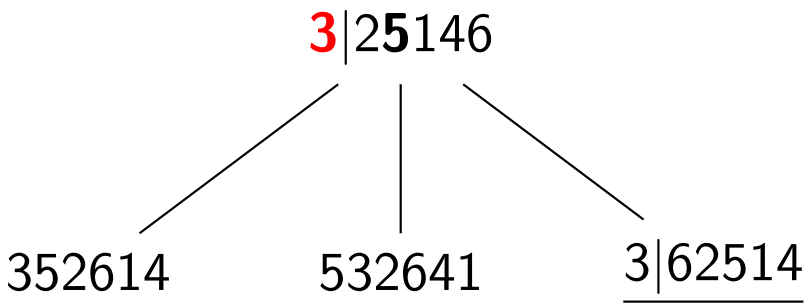
Counting preimages

A preimage of $\sigma \in S_n$ is uniquely determined by a subset of its LTR maxima other than n, that can be interpreted as "guards".



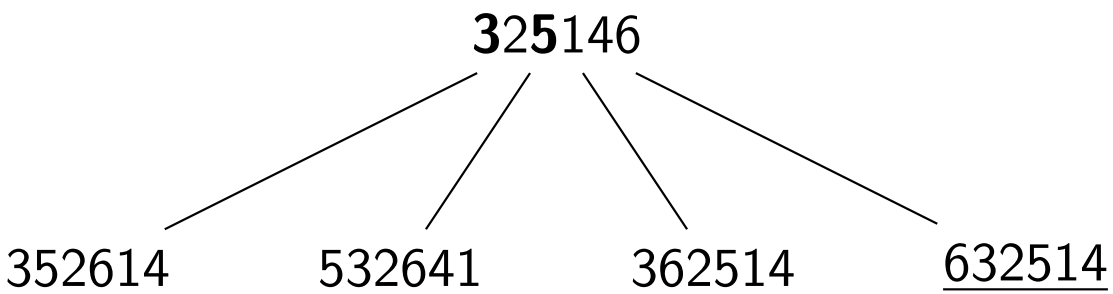
Counting preimages

A preimage of $\sigma \in S_n$ is uniquely determined by a subset of its LTR maxima other than n, that can be interpreted as "guards".



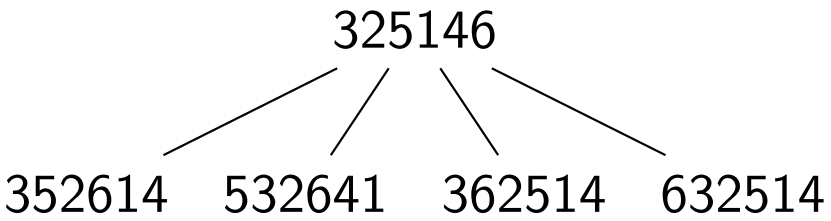
Counting preimages

A preimage of $\sigma \in S_n$ is uniquely determined by a subset of its LTR maxima other than n, that can be interpreted as "guards".



Counting preimages

A preimage of $\sigma \in S_n$ is uniquely determined by a subset of its LTR maxima other than n, that can be interpreted as "guards".



Corollary

$\sigma = \sigma_1 \cdots \sigma_{n-1} n \in S_n$ *with* k *LTR maxima*

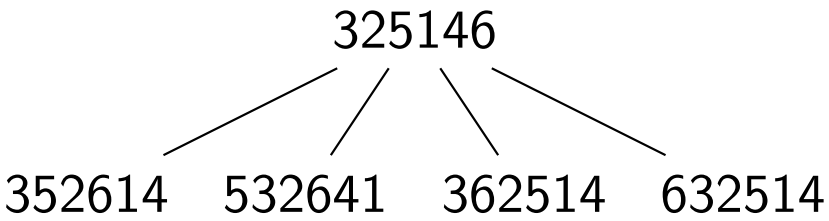
- $|\mathbf{B}^{-1}(\sigma)| = 2^{k-1}$
- $|\mathbf{B}^{-1}(\sigma)$ with j LTR maxima $| = \binom{k-1}{j-1}$

Corollary

$|\{\sigma \in S_n \,|\, |\mathbf{B}^{-1}(\sigma)| = 2^{k-1}\}| = \begin{bmatrix} n-1 \\ k-1 \end{bmatrix}$ (unsigned Stirling numbers of the I kind)

Counting preimages

A preimage of $\sigma \in S_n$ is uniquely determined by a subset of its LTR maxima other than n, that can be interpreted as "guards".



Corollary

$\sigma = \sigma_1 \cdots \sigma_{n-1} n \in S_n$ *with k LTR maxima*

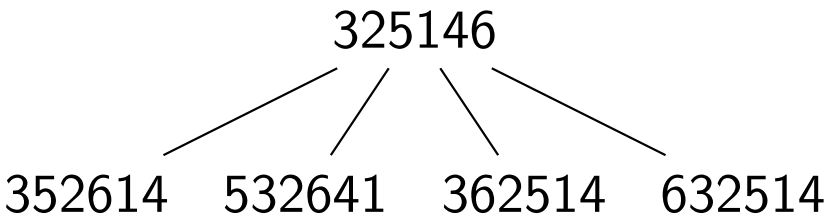
- $|\mathbf{B}^{-1}(\sigma)| = 2^{k-1}$
- $|\mathbf{B}^{-1}(\sigma)$ with j LTR maxima $| = \binom{k-1}{j-1}$

Corollary

$|\{\sigma \in S_n \,|\, |\mathbf{B}^{-1}(\sigma)| = 2^{k-1}\}| = \begin{bmatrix} n-1 \\ k-1 \end{bmatrix}$ (unsigned Stirling numbers of the I kind)

Counting preimages

A preimage of $\sigma \in S_n$ is uniquely determined by a subset of its LTR maxima other than n, that can be interpreted as "guards".



Corollary

$\sigma = \sigma_1 \cdots \sigma_{n-1} n \in S_n$ with k LTR maxima

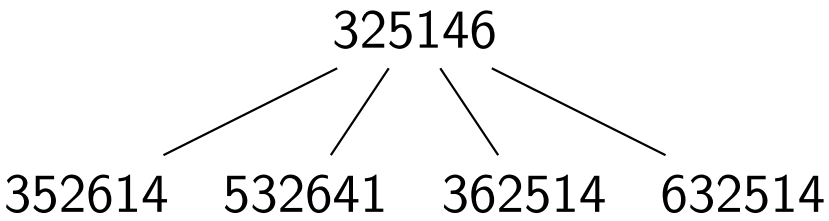
- $|\mathbf{B}^{-1}(\sigma)| = 2^{k-1}$
- $|\mathbf{B}^{-1}(\sigma)$ with j LTR maxima $| = \binom{k-1}{j-1}$

Corollary

$|\{\sigma \in S_n \,|\, |\mathbf{B}^{-1}(\sigma)| = 2^{k-1}\}| = \begin{bmatrix} n-1 \\ k-1 \end{bmatrix}$ (unsigned Stirling numbers of the I kind)

Counting preimages

A preimage of $\sigma \in S_n$ is uniquely determined by a subset of its LTR maxima other than n, that can be interpreted as "guards".



Corollary

$\sigma = \sigma_1 \cdots \sigma_{n-1} n \in S_n$ *with* k *LTR maxima*

- $|\mathbf{B}^{-1}(\sigma)| = 2^{k-1}$
- $|\mathbf{B}^{-1}(\sigma)$ with j LTR maxima $| = \binom{k-1}{j-1}$

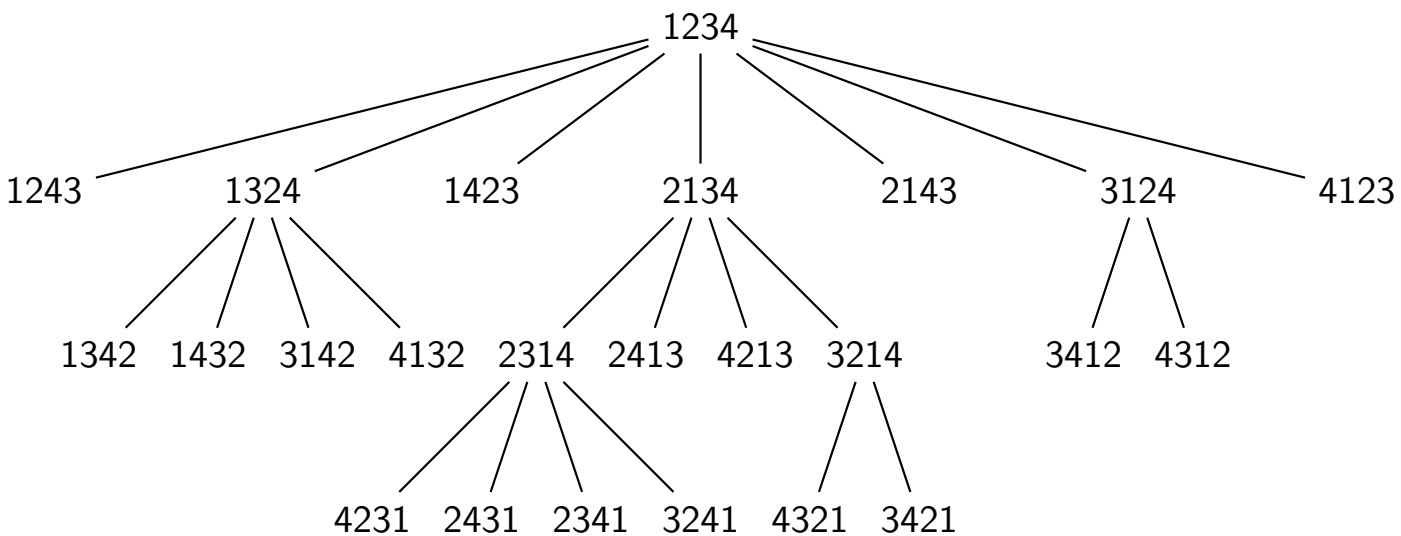
Corollary

$|\{\sigma \in S_n \,|\, |\mathbf{B}^{-1}(\sigma)| = 2^{k-1}\}| = \begin{bmatrix} n-1 \\ k-1 \end{bmatrix}$ (unsigned Stirling numbers of the I kind)

Trees and subtrees

Rooted tree T_n:

- nodes are permutations of size n;
- root is id_n;
- τ is a child of σ whenever $\mathbf{B}(\tau) = \sigma$ (and $\sigma \neq \tau$).

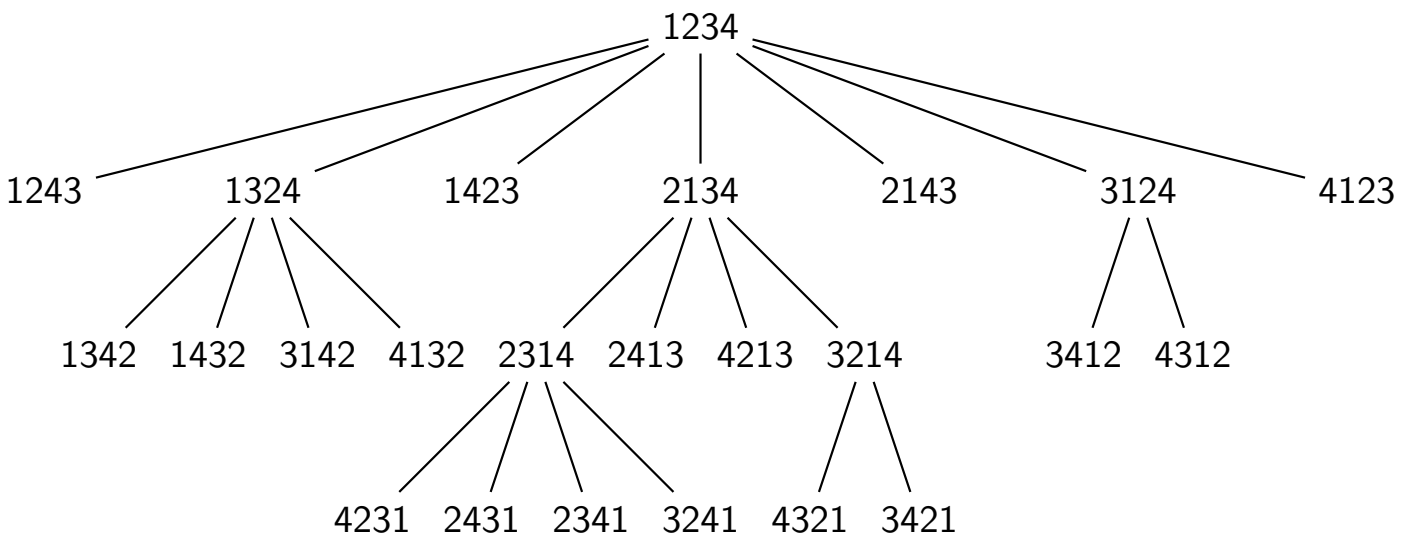


The tree T_4 of iterated preimages.

Trees and subtrees

Rooted tree T_n:

- nodes are permutations of size n;
- root is id_n;
- τ is a child of σ whenever $\mathbf{B}(\tau) = \sigma$ (and $\sigma \neq \tau$).

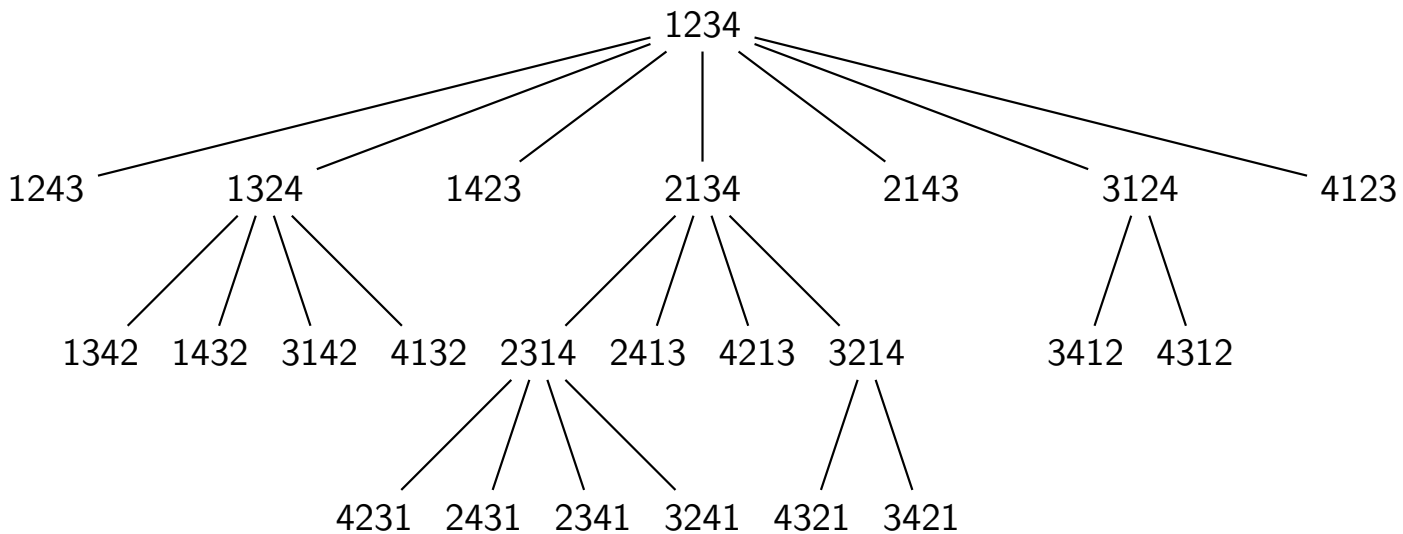


The tree T_4 of iterated preimages.

Trees and subtrees

Rooted tree T_n:

- nodes are permutations of size n;
- root is id_n;
- τ is a child of σ whenever $\mathbf{B}(\tau) = \sigma$ (and $\sigma \neq \tau$).

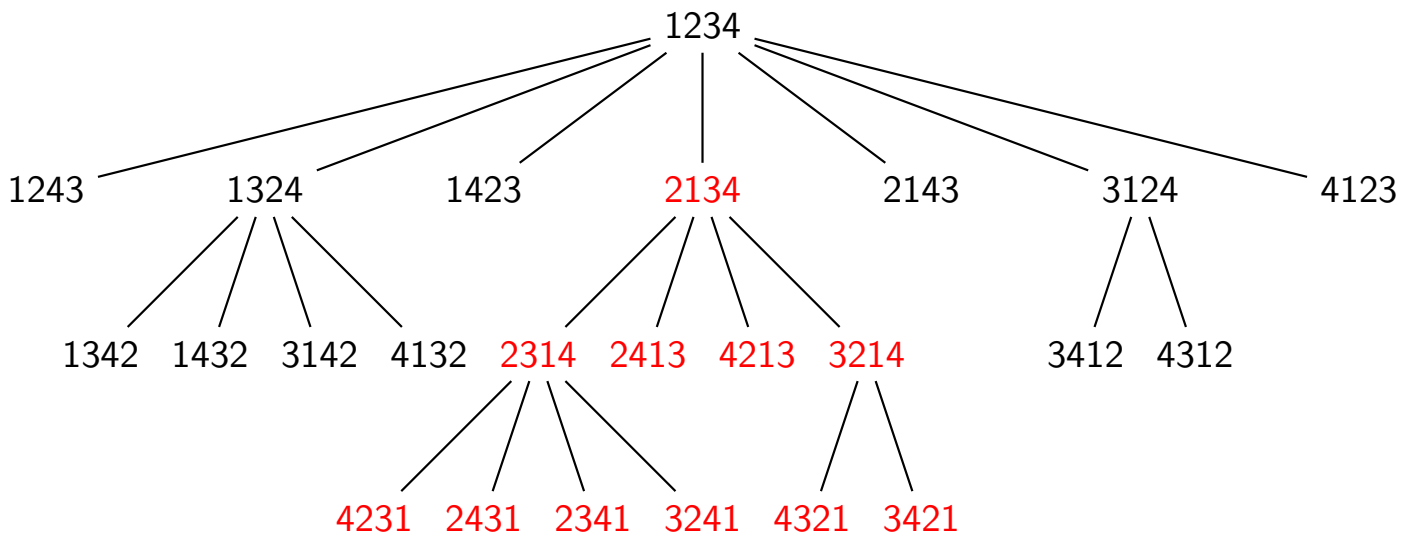


The tree T_4 of iterated preimages.

Trees and subtrees

Rooted tree T_n:

- nodes are permutations of size n;
- root is id_n;
- τ is a child of σ whenever $\mathbf{B}(\tau) = \sigma$ (and $\sigma \neq \tau$).

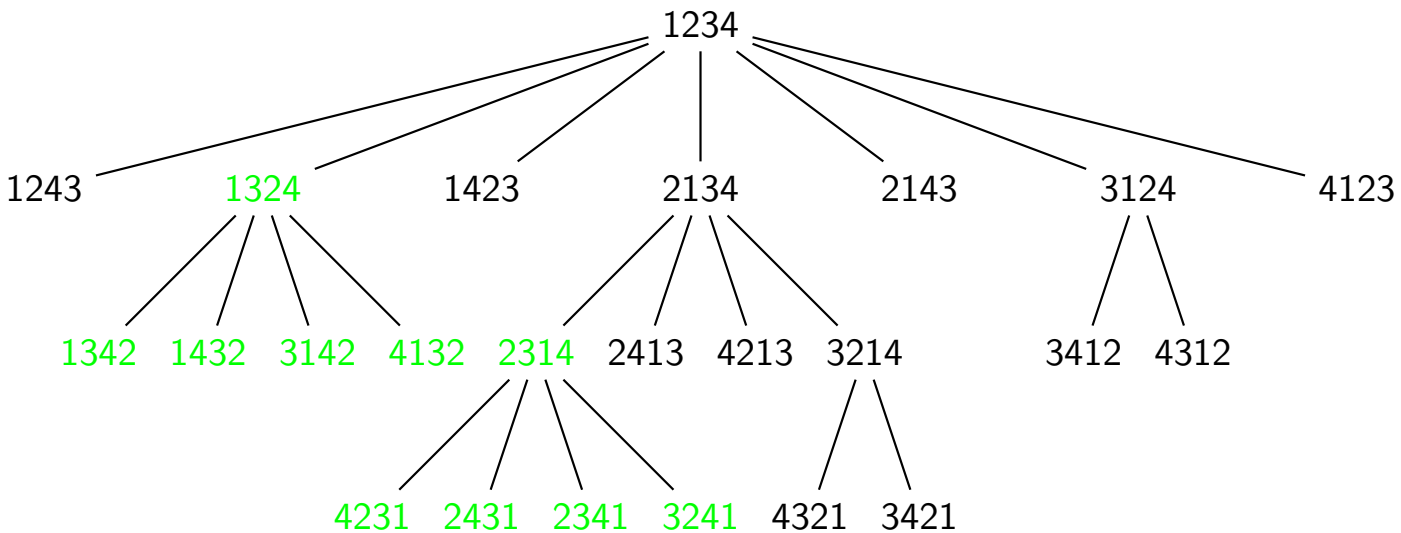


The subtree $T(2134)$.

Isomorphisms between subtrees

Lemma

Let π and τ be two permutations of the same size. If π and τ have their LTR maxima in the same positions, then $T(\pi)$ and $T(\tau)$ are isomorphic.

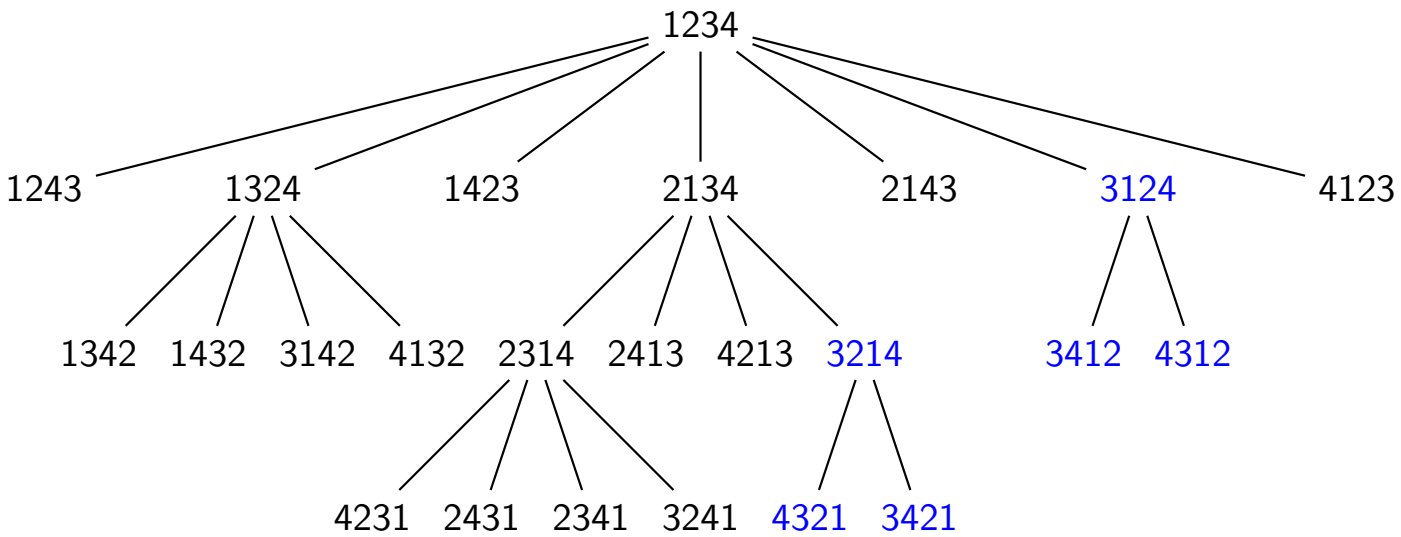


$T(1324)$ and $T(2314)$ are isomorphic.

Isomorphisms between subtrees

Proposition

For every permutation $\pi \in S_n$, $\pi \neq id_n$, there exists a child τ of id_n in T_n such that $T(\pi)$ and $T(\tau)$ are isomorphic.



$T(3214)$ is isomorphic to $T(3124)$.

The skeleton

- $\sigma \in S_n$ having k LTR maxima and whose last m_ℓ elements are LTR maxima
- (k, m_ℓ) is the *label* of σ
- *Skeleton* of $T(\sigma)$: obtained from $T(\sigma)$ by replacing each permutation at a node with its label.

The skeleton

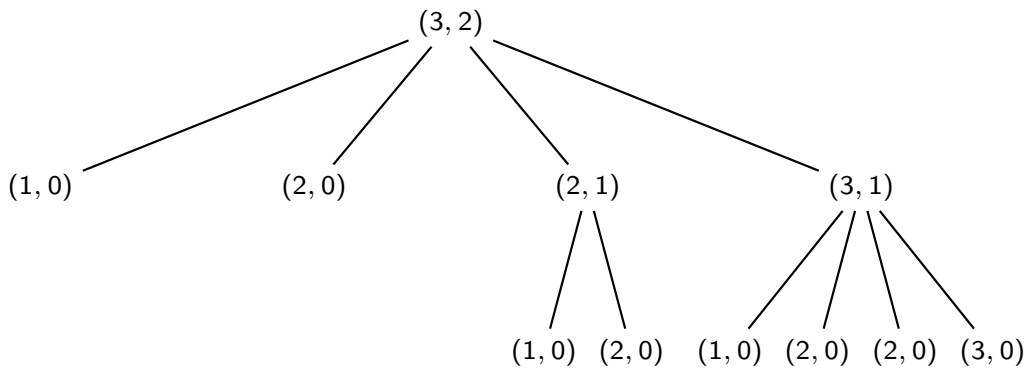
- $\sigma \in S_n$ having k LTR maxima and whose last m_ℓ elements are LTR maxima
- (k, m_ℓ) is the *label* of σ
- *Skeleton* of $T(\sigma)$: obtained from $T(\sigma)$ by replacing each permutation at a node with its label.

The skeleton

- $\sigma \in S_n$ having k LTR maxima and whose last m_ℓ elements are LTR maxima
- (k, m_ℓ) is the *label* of σ
- *Skeleton* of $T(\sigma)$: obtained from $T(\sigma)$ by replacing each permutation at a node with its label.

The skeleton

- $\sigma \in S_n$ having k LTR maxima and whose last m_ℓ elements are LTR maxima
- (k, m_ℓ) is the *label* of σ
- *Skeleton* of $T(\sigma)$: obtained from $T(\sigma)$ by replacing each permutation at a node with its label.



The skeleton of $T(2134)$

The skeleton

The skeleton of $T(\pi)$ is completely determined by the label of its root π.

Proposition

Let $\pi \in S_n$ with label (k, m_ℓ). Let T be the skeleton of $T(\pi)$. Then the root of T has label (k, m_ℓ) and its children have the following labels:

- *for every $h = 0, \ldots, m_\ell - 2$:*
- *if $\pi \neq 10_n$, we also have the case corresponding to $h = m_\ell - 1$.*

The skeleton

The skeleton of $T(\pi)$ is completely determined by the label of its root π.

Proposition

Let $\pi \in S_n$ with label (k, m_ℓ). Let T be the skeleton of $T(\pi)$. Then the root of T has label (k, m_ℓ) and its children have the following labels:

- *for every $h = 0, \ldots, m_\ell - 2$:*
  - *for every $i = 1, \ldots, k-1-h$, there are $\binom{k-2-h}{i-1}$ children with label $(k-i, h)$;*
- *if $\pi \neq id_n$, we also have the case corresponding to $h = m_\ell - 1$:*
  - *for every $i = 0, \ldots, k - m_\ell$, there are $\binom{k-m_\ell}{i}$ children with label $(k-i, m_\ell - 1) = (k-i, h)$.*

The skeleton

The skeleton of $T(\pi)$ is completely determined by the label of its root π.

Proposition

Let $\pi \in S_n$ with label (k, m_ℓ). Let T be the skeleton of $T(\pi)$. Then the root of T has label (k, m_ℓ) and its children have the following labels:

- *for every $h = 0, \ldots, m_\ell - 2$:*
  - *for every $i = 1, \ldots, k - 1 - h$, there are $\binom{k-2-h}{i-1}$ children with label $(k - i, h)$;*
- *if $\pi \neq id_n$, we also have the case corresponding to $h = m_\ell - 1$:*
  - *for every $i = 0, \ldots, k - m_\ell$, there are $\binom{k-m_\ell}{i}$ children with label $(k - i, m_\ell - 1) = (k - i, h)$.*

The skeleton

The skeleton of $T(\pi)$ is completely determined by the label of its root π.

Proposition

Let $\pi \in S_n$ with label (k, m_ℓ). Let T be the skeleton of $T(\pi)$. Then the root of T has label (k, m_ℓ) and its children have the following labels:

- *for every $h = 0, \ldots, m_\ell - 2$:*
  - *for every $i = 1, \ldots, k - 1 - h$, there are $\binom{k-2-h}{i-1}$ children with label $(k - i, h)$;*
- *if $\pi \neq id_n$, we also have the case corresponding to $h = m_\ell - 1$:*
  - *for every $i = 0, \ldots, k - m_\ell$, there are $\binom{k-m_\ell}{i}$ children with label $(k - i, m_\ell - 1) = (k - i, h)$.*

The skeleton

The skeleton of $T(\pi)$ is completely determined by the label of its root π.

Proposition

Let $\pi \in S_n$ with label (k, m_ℓ). Let T be the skeleton of $T(\pi)$. Then the root of T has label (k, m_ℓ) and its children have the following labels:

- *for every $h = 0, \ldots, m_\ell - 2$:*
  - *for every $i = 1, \ldots, k - 1 - h$, there are $\binom{k-2-h}{i-1}$ children with label $(k - i, h)$;*
- *if $\pi \neq id_n$, we also have the case corresponding to $h = m_\ell - 1$:*
  - *for every $i = 0, \ldots, k - m_\ell$, there are $\binom{k-m_\ell}{i}$ children with label $(k - i, m_\ell - 1) = (k - i, h)$.*

The skeleton

The skeleton of $T(\pi)$ is completely determined by the label of its root π.

Proposition

Let $\pi \in S_n$ with label (k, m_ℓ). Let T be the skeleton of $T(\pi)$. Then the root of T has label (k, m_ℓ) and its children have the following labels:

- *for every $h = 0, \ldots, m_\ell - 2$:*
  - *for every $i = 1, \ldots, k - 1 - h$, there are $\binom{k-2-h}{i-1}$ children with label $(k - i, h)$;*
- *if $\pi \neq id_n$, we also have the case corresponding to $h = m_\ell - 1$:*
  - *for every $i = 0, \ldots, k - m_\ell$, there are $\binom{k-m_\ell}{i}$ children with label $(k - i, m_\ell - 1) = (k - i, h)$.*

The skeleton

Some consequences:

Corollary

Let π be a permutation of size n with label (k, m_ℓ) such that $\pi \neq id_n$. Then $T(\pi)$ has height m_ℓ. In addition, for every $n \geq 1$, T_n has height $n - 1$.

Corollary

For any given node $\pi \neq id_n$ in T_n, either half of its children are leaves or all of its children are leaves.

The skeleton

Some consequences:

Corollary

Let π be a permutation of size n with label (k, m_ℓ) such that $\pi \neq id_n$. Then $T(\pi)$ has height m_ℓ. In addition, for every $n \geq 1$, T_n has height $n - 1$.

Corollary

For any given node $\pi \neq id_n$ in T_n, either half of its children are leaves or all of its children are leaves.

Heights of nodes in T_n

Height of a node is equal to the number of passes of Bubblesort needed to sort the permutation.

Proposition (Albert, Atkinson, Bouvel, Claesson, and Dukes, 2011)

The set of permutations of size n sorted by at most k passes of Bubblesort is $\mathrm{Av}_n(\Gamma_{k+2})$, *where* Γ_k *is the set of all permutations of size k whose final element is 1. As a consequence, setting* $\varphi_n^{(k)} = |\mathrm{Av}_n(\Gamma_k)|$, *the number of nodes at height at most k in* T_n *is given by* $\varphi_n^{(k+2)} = (k+1)^{n-k-1}(k+1)!$

Corollary

The number $f_n^{(k)}$ *of nodes at height k in* T_n *is given by*

$$f_n^{(k)} = \varphi_n^{(k+2)} - \varphi_n^{(k+1)} = k! \cdot ((k+1)^{n-k} - k^{n-k}).$$

Heights of nodes in T_n

Height of a node is equal to the number of passes of Bubblesort needed to sort the permutation.

Proposition (Albert, Atkinson, Bouvel, Claesson, and Dukes, 2011)

The set of permutations of size n sorted by at most k passes of Bubblesort is $\mathrm{Av}_n(\Gamma_{k+2})$, *where* Γ_k *is the set of all permutations of size k whose final element is 1. As a consequence, setting* $\varphi_n^{(k)} = |\mathrm{Av}_n(\Gamma_k)|$, *the number of nodes at height at most k in* T_n *is given by* $\varphi_n^{(k+2)} = (k+1)^{n-k-1}(k+1)!$

Corollary

The number $f_n^{(k)}$ *of nodes at height k in* T_n *is given by*

$$f_n^{(k)} = \varphi_n^{(k+2)} - \varphi_n^{(k+1)} = k! \cdot ((k+1)^{n-k} - k^{n-k}).$$

Heights of nodes in T_n

n \ k	0	1	2	3	4	5
1	1					
2	1	1				
3	1	3	2			
4	1	7	10	6		
5	1	15	38	42	24	
6	1	31	130	222	216	120

Sequence A056151: number of nodes in T_n having height k.

Proposition (Flajolet and Sedgewick, 2013)

The average height of a node in T_n is asymptotically equal to $n - \sqrt{\frac{\pi n}{2}} + O(1)$.

Heights of nodes in T_n

n \ k	0	1	2	3	4	5
1	1					
2	1	1				
3	1	3	2			
4	1	7	10	6		
5	1	15	38	42	24	
6	1	31	130	222	216	120

Sequence A056151: number of nodes in T_n having height k.

Proposition (Flajolet and Sedgewick, 2013)

The average height of a node in T_n is asymptotically equal to $n - \sqrt{\frac{\pi n}{2}} + O(1)$.

Height of nodes in $T(\pi)$

Proposition

For a permutation π having label (k, m_ℓ), different from an identity permutation, the number of nodes of $T(\pi)$ is

$$N(k, m_\ell) = \sum_{j=0}^{m_\ell} j!(j+1)^{k-j}. \tag{1}$$

Moreover, each summand records the contribution of each level of $T(\pi)$. In other words, denoting with $N_j(k, m_\ell)$ the number of nodes at height j in $T(\pi)$, we have that $N_j(k, m_\ell) = j!(j+1)^{k-j}$.

Heights of leaves in T_n

- $\mathrm{Av}_n^*(\Gamma_k)$: permutations of size n avoiding Γ_k whose last element is different from n
- $\gamma_n^{(k)} = |\mathrm{Av}_n^*(\Gamma_k)|$
- $\gamma_n^{(k+2)}$ is the number of leaves at height at most k in T_n

Proposition

$$\gamma_n^{(k)} = \begin{cases} (n-1)(n-1)! & n < k, \\ (k-2)(k-1)^{n-k}(k-1)! & n \geq k. \end{cases}$$

Corollary

The number $g_n^{(k)}$ of leaves of T_n at height k is given by

$$g_n^{(k)} = k!(k(k+1)^{n-k-1} - (k-1)k^{n-k-1}).$$

Heights of leaves in T_n

- $\mathrm{Av}_n^*(\Gamma_k)$: permutations of size n avoiding Γ_k whose last element is different from n
- $\gamma_n^{(k)} = |\mathrm{Av}_n^*(\Gamma_k)|$
- $\gamma_n^{(k+2)}$ is the number of leaves at height at most k in T_n

Proposition

$$\gamma_n^{(k)} = \begin{cases} (n-1)(n-1)! & n < k, \\ (k-2)(k-1)^{n-k}(k-1)! & n \geq k. \end{cases}$$

Corollary

The number $g_n^{(k)}$ of leaves of T_n at height k is given by

$$g_n^{(k)} = k!(k(k+1)^{n-k-1} - (k-1)k^{n-k-1}).$$

Heights of leaves in T_n

- $\mathrm{Av}_n^*(\Gamma_k)$: permutations of size n avoiding Γ_k whose last element is different from n
- $\gamma_n^{(k)} = |\mathrm{Av}_n^*(\Gamma_k)|$
- $\gamma_n^{(k+2)}$ is the number of leaves at height at most k in T_n

Proposition

$$\gamma_n^{(k)} = \begin{cases} (n-1)(n-1)! & n < k, \\ (k-2)(k-1)^{n-k}(k-1)! & n \geq k. \end{cases}$$

Corollary

The number $g_n^{(k)}$ of leaves of T_n at height k is given by

$$g_n^{(k)} = k!(k(k+1)^{n-k-1} - (k-1)k^{n-k-1}).$$

Heights of leaves in T_n

- $\mathrm{Av}_n^*(\Gamma_k)$: permutations of size n avoiding Γ_k whose last element is different from n
- $\gamma_n^{(k)} = |\mathrm{Av}_n^*(\Gamma_k)|$
- $\gamma_n^{(k+2)}$ is the number of leaves at height at most k in T_n

Proposition

$$\gamma_n^{(k)} = \begin{cases} (n-1)(n-1)! & n < k, \\ (k-2)(k-1)^{n-k}(k-1)! & n \geq k. \end{cases}$$

Corollary

The number $g_n^{(k)}$ of leaves of T_n at height k is given by

$$g_n^{(k)} = k!(k(k+1)^{n-k-1} - (k-1)k^{n-k-1}).$$

Heights of leaves in T_n

- $\mathrm{Av}_n^*(\Gamma_k)$: permutations of size n avoiding Γ_k whose last element is different from n
- $\gamma_n^{(k)} = |\mathrm{Av}_n^*(\Gamma_k)|$
- $\gamma_n^{(k+2)}$ is the number of leaves at height at most k in T_n

Proposition

$$\gamma_n^{(k)} = \begin{cases} (n-1)(n-1)! & n < k, \\ (k-2)(k-1)^{n-k}(k-1)! & n \geq k. \end{cases}$$

Corollary

The number $g_n^{(k)}$ of leaves of T_n at height k is given by

$$g_n^{(k)} = k!(k(k+1)^{n-k-1} - (k-1)k^{n-k-1}).$$

Heights of leaves in T_n

The argument by Flajolet and Sedgewick to compute the average height of a node in T_n can be adapted to the case of leaves.

Proposition

The average height of a leaf in T_n is asymptotically equal to $n - \sqrt{\frac{\pi n}{2}} + O(1)$.

Height of leaves in $T(\pi)$

Proposition

For a permutation π having label (k, m_ℓ), different from an identity permutation, the number of leaves of $T(\pi)$ is

$$L(k, m_\ell) = \sum_{j=1}^{m_\ell - 1} j!j(j+1)^{k-j-1} + m_\ell!(m_\ell+1)^{k-m_\ell}. \tag{2}$$

Moreover, each summand in (2) records the contribution of each level of $T(\pi)$. In other words, denoting with $L_j(k, m_\ell)$ the number of leaves at height j in $T(\pi)$, we have that $L_j(k, m_\ell) = j!j(j+1)^{k-j-1}$ for $j < m_\ell$, and $L_{m_\ell}(k, m_\ell) = m_\ell!(m_\ell+1)^{k-m_\ell}$.

Further work

- Asymptotics for nodes and leaves in $T(\pi)$.
- Preimages of other sorting operators (Popstacksort, popqueuesort, sorting by reversals/transpositions,...).

Thank you!

Further work

- Asymptotics for nodes and leaves in $T(\pi)$.
- Preimages of other sorting operators (Popstacksort, popqueuesort, sorting by reversals/transpositions,...).

Thank you!

Further work

- Asymptotics for nodes and leaves in $T(\pi)$.
- Preimages of other sorting operators (Popstacksort, popqueuesort, sorting by reversals/transpositions,...).

Thank you!

Further work

- Asymptotics for nodes and leaves in $T(\pi)$.
- Preimages of other sorting operators (Popstacksort, popqueuesort, sorting by reversals/transpositions,...).

Thank you!