Forests and Parking Functions

Heesung Shin

Institut Camille Jordan
Université Claude Bernard Lyon-I, France

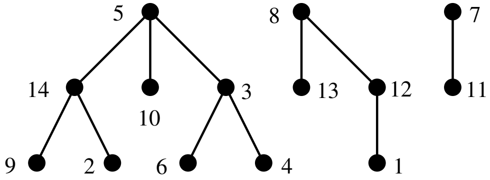
THE 61ST SÉMINAIRE LOTHARINGIEN DE COMBINATOIRE
CURIA, PORTUGAL, SEP. 24, 2008

Outline

1. Forests
2. Parking Functions
3. Statistics
4. The Map $\varphi : F_n \to PF_n$
5. Further Result

Outline

(Rooted Labeled) Forests



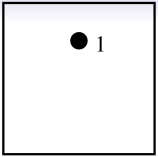


Figure: Forest on [1]

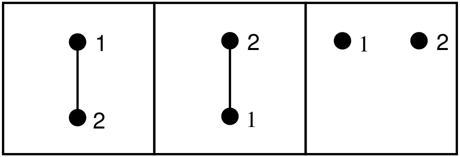


Figure: Forests on [2]

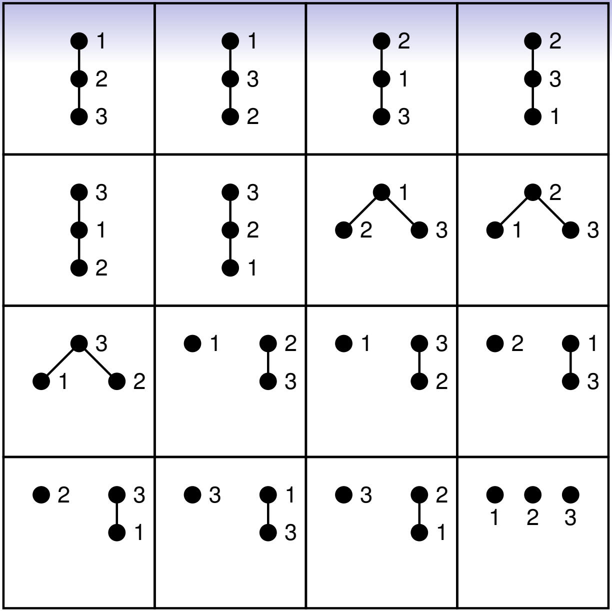


Figure: Forests on [3]

$$
\begin{aligned}
\#\text{ of forests on }[1] &= 1 = 2^0\\
\#\text{ of forests on }[2] &= 3 = 3^1\\
\#\text{ of forests on }[3] &= 16 = 4^2\\
\#\text{ of forests on }[4] &= 125 = 5^3\\
&\vdots\\
\#\text{ of forests on }[n] &= (n+1)^{n-1}
\end{aligned}
$$

Actually,

$$\#\text{ of forests on }[n] = \#\text{ of trees on }[n+1].$$

$$
\begin{aligned}
\#\text{ of forests on } [1] &= 1 = 2^0 \\
\#\text{ of forests on } [2] &= 3 = 3^1 \\
\#\text{ of forests on } [3] &= 16 = 4^2 \\
\#\text{ of forests on } [4] &= 125 = 5^3 \\
&\vdots \\
\#\text{ of forests on } [n] &= (n+1)^{n-1}
\end{aligned}
$$

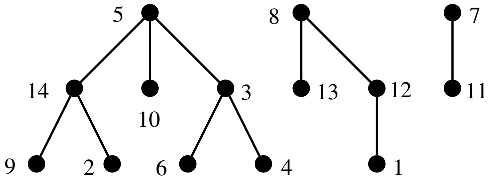
Actually,

$$\#\text{ of forests on } [n] = \#\text{ of trees on } [n+1].$$

Inversion in Forests

$$\begin{aligned} \mathrm{inv}(F;v) &= \#\{(v,u) \mid u \text{ is descendant of } v \text{ and } u < v\} \\ \mathrm{inv}(F) &= \sum_v \mathrm{inv}(F;v) \end{aligned}$$

For example,



$$\begin{aligned} \mathrm{inv}(F;5) &= \#\{(5,3),(5,4),(5,2)\} = 3 \\ \mathrm{inv}(F) &= 7 \end{aligned}$$

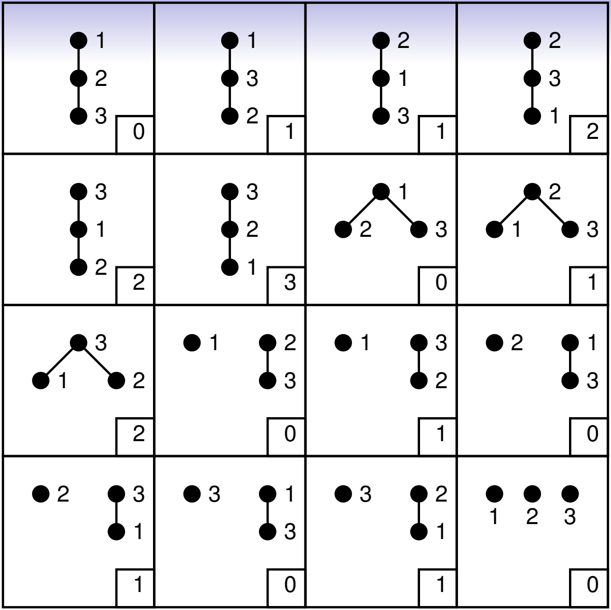
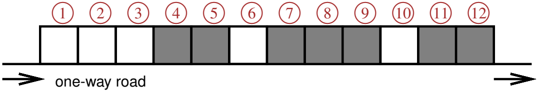


Figure: inv on forests on $[3]$

Outline

Parking Rule



If you like parking at ⑦, then you could park at ⑩.
If you like parking at ⑪, then you could not park.

Parking Function

Suppose $(p_1, \ldots, p_n)$ is a sequence of favorite parking spaces for each cars. If no car failed in parking, $(p_1, \ldots, p_n)$ is called parking function.

For example,

$$(4, 3, 3, 1, 4) \longrightarrow \boxed{4}\boxed{\emptyset}\boxed{2}\boxed{1}\boxed{3}$$

is not a parking function.

$$(4, 3, 3, 1, 1) \longrightarrow \boxed{4}\boxed{5}\boxed{2}\boxed{1}\boxed{3}$$

is a parking function.

Criteria of Parking Function

$$q_i \leqslant i \text{ for all } i$$

where $(q_1, \ldots, q_n)$ is rearrangement of $(p_1, \ldots, p_n)$ by order.

For example,

$$(4, 3, 3, 1, 4) \to (1, 3, 3, 4, 4)$$

is not a parking function.

$$(4, 3, 3, 1, 1) \to (1, 1, 3, 3, 4)$$

is a parking function.

(1)

Figure: Parking Function with length 1

$(1,1)$	$(1,2)$	$(2,1)$

Figure: Parking Functions with length 2

$(1,1,1)$	$(1,1,2)$	$(1,1,3)$	$(1,2,1)$
$(1,2,2)$	$(1,2,3)$	$(1,3,1)$	$(1,3,2)$
$(2,1,1)$	$(2,1,2)$	$(2,1,3)$	$(2,2,1)$
$(2,3,1)$	$(3,1,1)$	$(3,1,2)$	$(3,2,1)$

Figure: Parking Functions with length 3

Jump in Parking Functions

1. $PA(p_1, \ldots, p_n) = (q_1, \ldots, q_n)$
   where q_i is the space parked actually by i-th car. here
2. $\text{jump}(P; i) = q_i - p_i$
3. $\text{jump}(P) = \sum_i \text{jump}(P; i)$

Note that,

$$\begin{aligned} \text{jump}(P) &= \sum_i \text{jump}(P; i) \\ &= (q_1 + \ldots + q_n) - (p_1 + \ldots + p_n) \\ &= \binom{n+1}{2} - (p_1 + \ldots + p_n) \end{aligned}$$

For example,

1. $PA(4,3,3,1,1) = (4,3,5,1,2)$
2. $\mathrm{jump}(4,3,3,1,1;3) = 5 - 3 = 2$
3. $\mathrm{jump}(4,3,3,1,1) = 0 + 0 + 2 + 0 + 1 = 3$

$(1,1,1)$ 3	$(1,1,2)$ 2	$(1,1,3)$ 1	$(1,2,1)$ 2
$(1,2,2)$ 1	$(1,2,3)$ 0	$(1,3,1)$ 1	$(1,3,2)$ 0
$(2,1,1)$ 2	$(2,1,2)$ 1	$(2,1,3)$ 0	$(2,2,1)$ 1
$(2,3,1)$ 0	$(3,1,1)$ 1	$(3,1,2)$ 0	$(3,2,1)$ 0

Figure: jump on Parking Functions with length 3

Outline

Generating Function

- GF for inv on forests

$$\sum_{F \in F_1} q^{\text{inv}(F)} = 1$$

$$\sum_{F \in F_2} q^{\text{inv}(F)} = 2 + q$$

$$\sum_{F \in F_3} q^{\text{inv}(F)} = 6 + 6q + 3q^2 + q^3$$

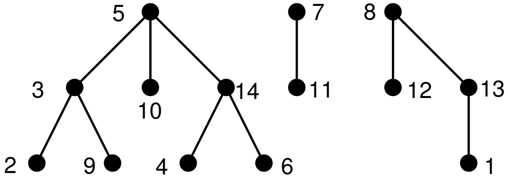
- GF for jump on parking function

$$\sum_{P \in PF_1} q^{\text{jump}(P)} = 1$$

$$\sum_{P \in PF_2} q^{\text{jump}(P)} = 2 + q$$

$$\sum_{P \in PF_3} q^{\text{jump}(P)} = 6 + 6q + 3q^2 + q^3$$

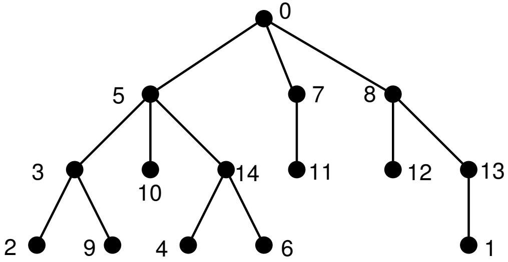
One Map



$$\Downarrow$$

$$(\_,\_,\_,\_,\_,\_,\_,\_,\_,\_,\_,\_,\_,\_)$$

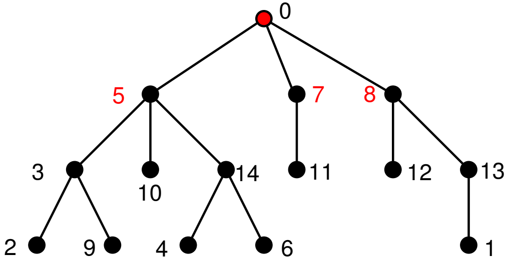
One Map



$$\Downarrow$$

$$(\_, \_, \_, \_, \_, \_, \_, \_, \_, \_, \_, \_, \_, \_)$$

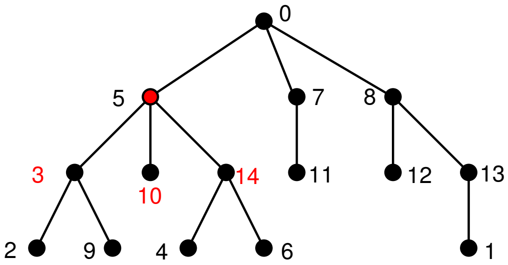
One Map



$$\Downarrow$$

$$(\_, \_, \_, \_, 1, \_, 1, 1, \_, \_, \_, \_, \_, \_)$$

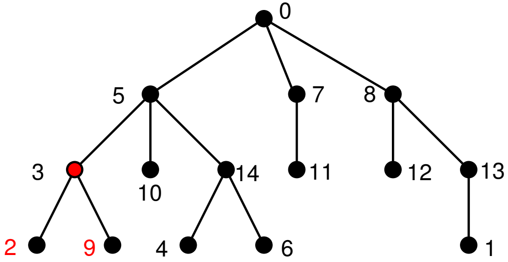
One Map



$\Downarrow$

$(\_,\_,2,\_,1,\_,1,1,\_,2,\_,\_,\_,2)$

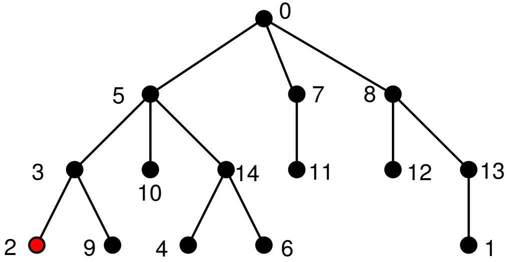
One Map



$\Downarrow$

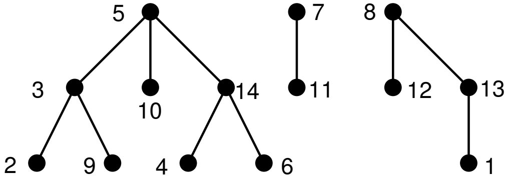
$$(\_, 3, 2, \_, 1, \_, 1, 1, 3, 2, \_, \_, \_, 2)$$

One Map



$$\Downarrow$$

$$(\_, 3, 2, \_, 1, \_, 1, 1, 3, 2, \_, \_, \_, 2)$$



$$\Downarrow$$

$$(14, 3, 2, 7, 1, 7, 1, 1, 3, 2, 10, 12, 12, 2)$$

Theorem (G.Kreweras 1980)

$$\sum_{F \in F_n} q^{\mathrm{inv}(F)} = \sum_{P \in PF_n} q^{\binom{n+1}{2} - |P|} \tag{1}$$

NOTE. In 2004, R. Stanley notices that a nonrecursive bijection between forests and parking functions would be greatly preferred, which yields (1)

Leader in Forests

Definition (Leader in Forests)

1. v = leader in $F \Leftrightarrow \mathrm{inv}(F; v) = 0$.
2. $\mathrm{lead}(F) =$ the # of leaders in F

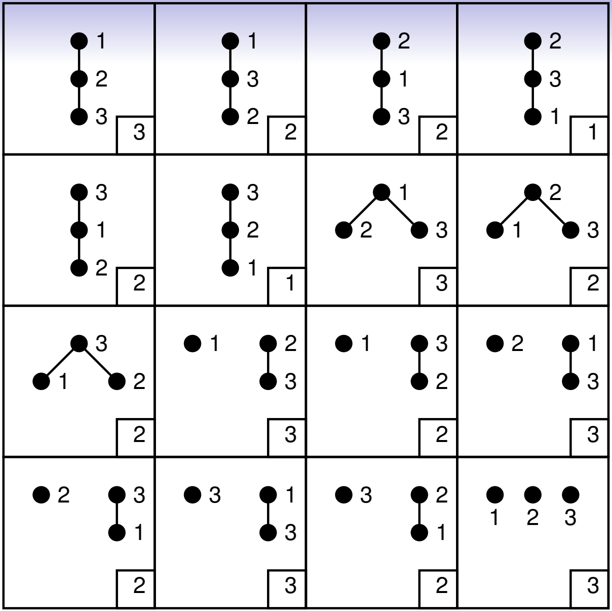


Figure: lead on forests on [3]

Definition (Lucky in Parking Functions)

1. c = lucky in $P \Leftrightarrow \mathrm{jump}(P;c) = 0$.
2. $\mathrm{lucky}(P) =$ the # of luckys in P

$(1,1,1)$ 1	$(1,1,2)$ 1	$(1,1,3)$ 2	$(1,2,1)$ 2
$(1,2,2)$ 2	$(1,2,3)$ 3	$(1,3,1)$ 2	$(1,3,2)$ 3
$(2,1,1)$ 2	$(2,1,2)$ 2	$(2,1,3)$ 3	$(2,2,1)$ 2
$(2,3,1)$ 3	$(3,1,1)$ 2	$(3,1,2)$ 3	$(3,2,1)$ 3

Figure: lucky on Parking Functions with length 3

Theorem (Gessel-Seo 2004)

$$\sum_{F \in F_n} u^{\mathrm{lead}\, F} = \sum_{P \in PF_n} u^{\mathrm{lucky}\, P} \tag{2}$$

NOTE. They do not have a direct proof of (2). They found each of two GFs for lead and lucky, that are neither bijective.

$$u \prod_{i=1}^{n-1} (i + (n - i + 1)u)$$

GS 2004

SS 2007

GS 2004

$$\sum_{F \in F_n} u^{\mathrm{lead}\, F} \overset{?}{=\!=} \sum_{P \in PF_n} u^{\mathrm{lucky}\, P}$$

Objective

We'll construct the nonrecursive bijection between forests and parking functions such that

$$\begin{array}{rcl} \varphi \quad : \quad F_n & \rightarrow & PF_n \\ F & \hookrightarrow & P = \varphi(F) \\ \mathrm{inv}(F) & = & \mathrm{jump}(P) \\ \mathrm{lead}(F) & = & \mathrm{lucky}(P) \end{array}$$

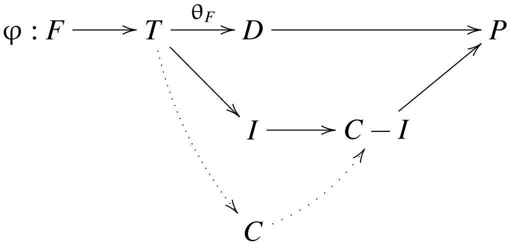
Theorem (S. 2008)

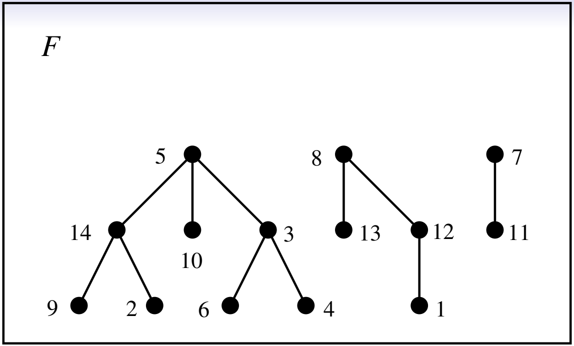
We have

$$\sum_{F \in F_n} q^{\mathrm{inv}(F)} u^{\mathrm{lead}(F)} = \sum_{P \in PF_n} q^{\mathrm{jump}(P)} u^{\mathrm{lucky}(P)}.$$

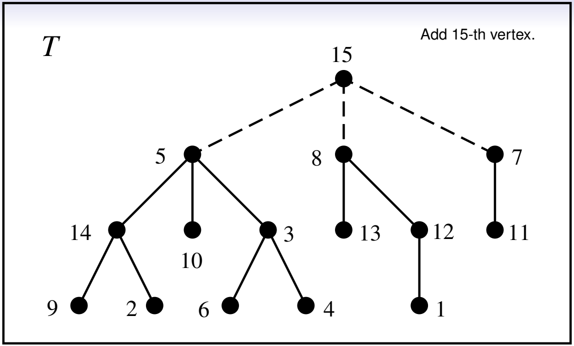
Outline

Diagram of φ

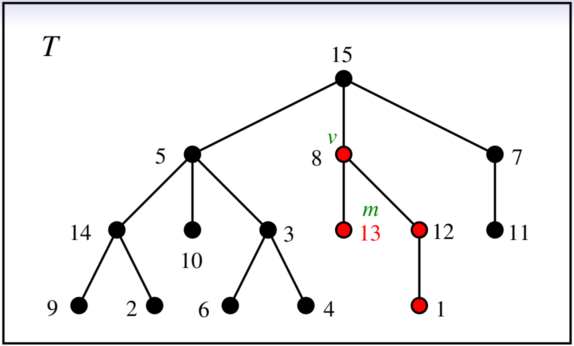




We consider $F \in F_{14}$ for example. Of course, F is drawn in the method we decide.



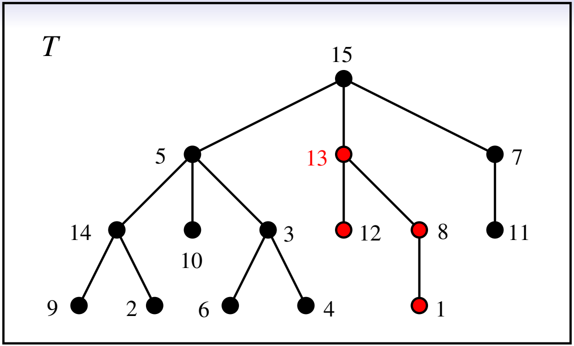
Add the vertex 15 at the top and change the forest F to the tree T.



for all $v \in V$ do

1. find the maximum label m on descendants of v.
2. label m on v.
3. rearrange the other labels in descendants of v by order-preserving.

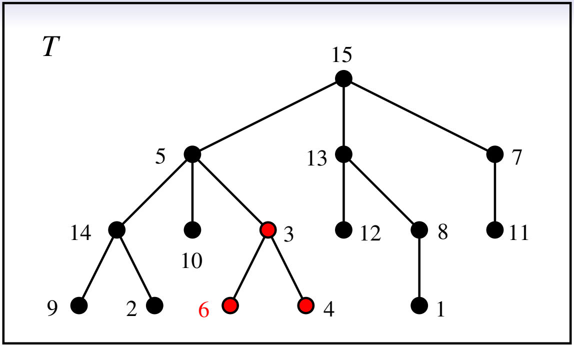
end do



for all $v \in V$ do

1. find the maximum label m on descendants of v.
2. label m on v.
3. rearrange the other labels in descendants of v by order-preserving.

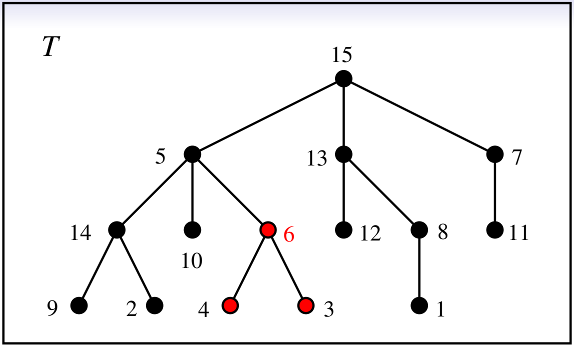
end do



for all $v \in V$ do

1. find the maximum label m on descendants of v.
2. label m on v.
3. rearrange the other labels in descendants of v by order-preserving.

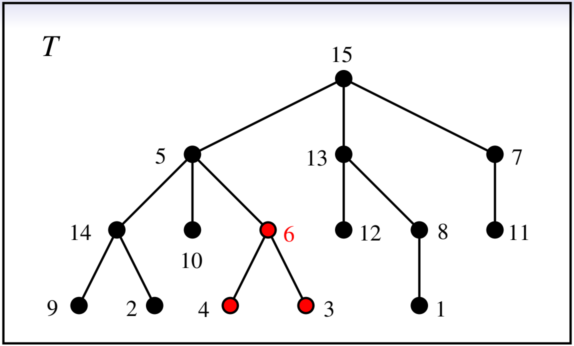
end do



for all $v \in V$ do

1. find the maximum label m on descendants of v.
2. label m on v.
3. rearrange the other labels in descendants of v by order-preserving.

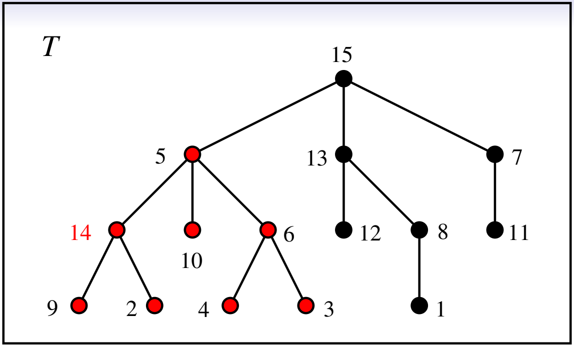
end do



for all $v \in V$ do

1. find the maximum label m on descendants of v.
2. label m on v.
3. rearrange the other labels in descendants of v by order-preserving.

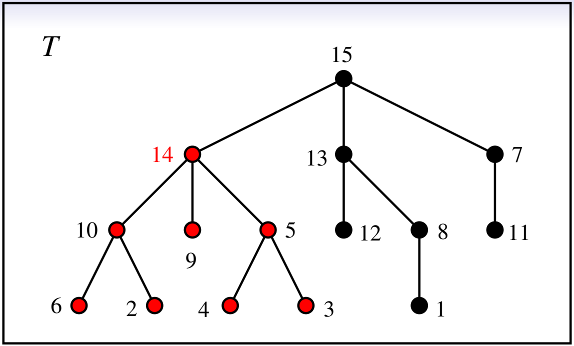
end do



for all $v \in V$ do

1. find the maximum label m on descendants of v.
2. label m on v.
3. rearrange the other labels in descendants of v by order-preserving.

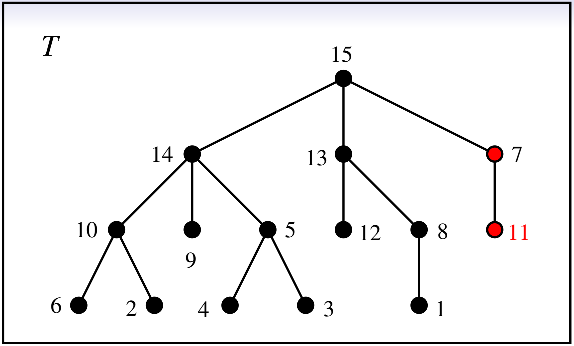
end do



for all $v \in V$ do

1. find the maximum label m on descendants of v.
2. label m on v.
3. rearrange the other labels in descendants of v by order-preserving.

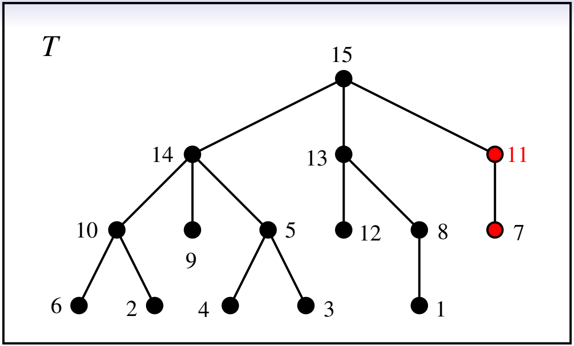
end do



for all $v \in V$ do

1. find the maximum label m on descendants of v.
2. label m on v.
3. rearrange the other labels in descendants of v by order-preserving.

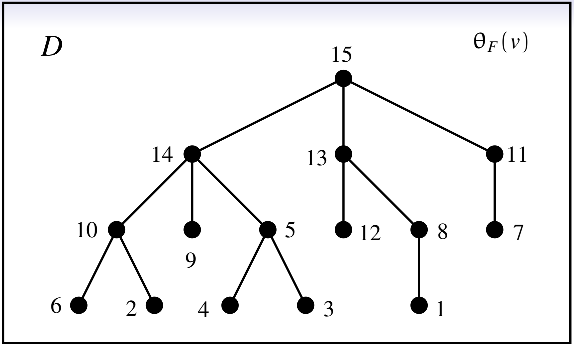
end do



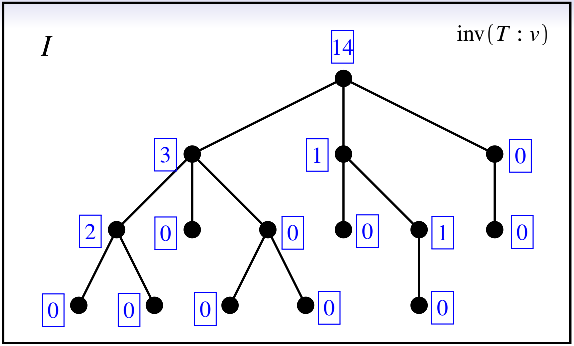
for all $v \in V$ do

1. find the maximum label m on descendants of v.
2. label m on v.
3. rearrange the other labels in descendants of v by order-preserving.

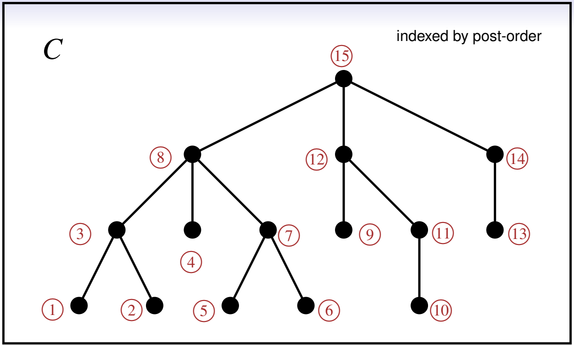
end do



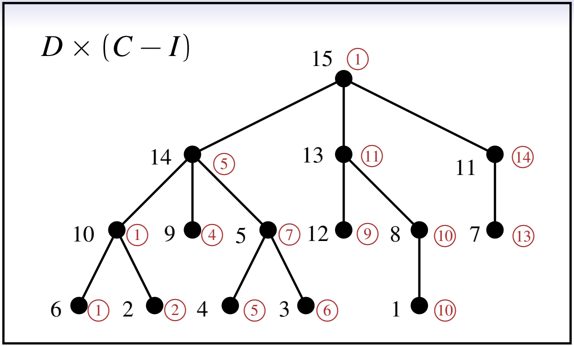
The decreasing tree is made after the process for every vertex. But we cannot remake the original tree T from only tree D. So, we need another tree induced from the unused information of T.



Label $\mathrm{inv}(T : v)$ on vertex v. In order to distinguish it from other labels, we use the box (or blue color). And then, the trees D and I can produce the original tree T.



Label the vertices indexed by post-order which is indicated by circle (or brown color). The tree C is determined by only the *underlying graph*, that is, its tree structure. This is the reason why we define the method we draw the tree.



The plain labels are induced by D.
The circled labels are induced by C subtracted by I.

And then, we delete the tree structure and sort by plain #.

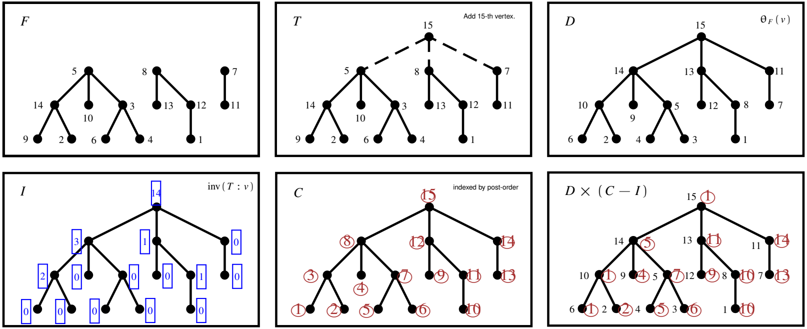
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$\textcircled{10}$	$\textcircled{2}$	$\textcircled{6}$	$\textcircled{5}$	$\textcircled{7}$	$\textcircled{1}$	$\textcircled{13}$	$\textcircled{10}$	$\textcircled{4}$	$\textcircled{1}$	$\textcircled{14}$	$\textcircled{9}$	$\textcircled{11}$	$\textcircled{5}$	$\textcircled{1}$

Below the plain label 15, there is always circle label $\textcircled{1}$. So, we can omit it, and then second row (circle label) becomes a *parking function* P of length 14.

$$P = \textcircled{10}\textcircled{2}\textcircled{6}\textcircled{5}\textcircled{7}\textcircled{1}\textcircled{13}\textcircled{10}\textcircled{4}\textcircled{1}\textcircled{14}\textcircled{9}\textcircled{11}\textcircled{5}$$

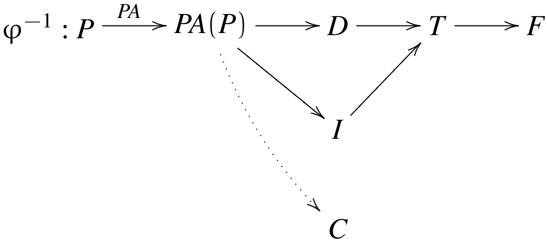
Because all labels of C are distinct in worst case which means every labels of I is all zero. Note that every permutation is a parking function.

Summary of the map φ



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
P=	10	2	6	5	7	1	13	10	4	1	14	9	11	5	1

Diagram of φ^{-1}



Inverse Map of φ

Let $P =$ ⑩②⑥⑤⑦①⑬⑩④①⑭⑨⑪⑤
After adding the ① at the end, 15 cars is parked as following by the parking algorithm.

Parking Space	①	②	③	④	⑤	⑥	⑦	⑧	⑨	⑩	⑪	⑫	⑬	⑭	⑮
Cars' Number c	6	2	10	9	4	3	5	14	12	1	8	13	7	11	15
$\text{jump}(P:c)$	0	0	2	0	0	0	0	3	0	0	1	1	0	0	14

We draw a edge between car c and the closest car on its right which is larger than c. If we consider 15 as a root, we can rebuild the tree structure and find trees C, D and I.

Inverse Map of φ

Let $P =$ ⑩ ② ⑥ ⑤ ⑦ ① ⑬ ⑩ ④ ① ⑭ ⑨ ⑪ ⑤
After adding the ① at the end, 15 cars is parked as following by the parking algorithm.

Parking Space	①	②	③	④	⑤	⑥	⑦	⑧	⑨	⑩	⑪	⑫	⑬	⑭	⑮
Cars' Number c	6	2	10	9	4	3	5	14	12	1	8	13	7	11	15
$\mathrm{jump}(P:c)$	0	0	2	0	0	0	0	3	0	0	1	1	0	0	14

We draw a edge between car c and the closest car on its right which is larger than c. If we consider 15 as a root, we can rebuild the tree structure and find trees C, D and I.

Outline

i-Leader in Forests, i-Lucky in PFs

Definition (i-Leader in Forests)

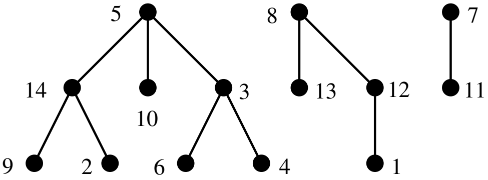
1. v = i-leader in $F \Leftrightarrow \mathrm{inv}(F;v) = i$
2. $\mathrm{lead}_i(F) =$ the # of i-leaders in F

Definition (i-Lucky in Parking Functions)

1. c = i-lucky in $P \Leftrightarrow \mathrm{jump}(P;c) = i$
2. $\mathrm{lucky}_i(P) =$ the # of i-luckys in P

We define

$$\begin{aligned}\mathbf{inv}(F) &= (\mathrm{lead}_0(F), \mathrm{lead}_1(F), \ldots, \mathrm{lead}_n(F)) \\ &= \text{type of inversions of } F \\ \mathbf{jump}(P) &= (\mathrm{lucky}_0(P), \mathrm{lucky}_1(P), \ldots, \mathrm{lucky}_n(P)) \\ &= \text{type of jumps of } P\end{aligned}$$



$$\mathbf{inv}(F) = (10, 2, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)$$

$$\Downarrow \varphi$$

$$(10, 2, 6, 5, 7, 1, 13, 10, 4, 1, 14, 9, 11, 5)$$

$$\mathbf{jump}(P) = (10, 2, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)$$

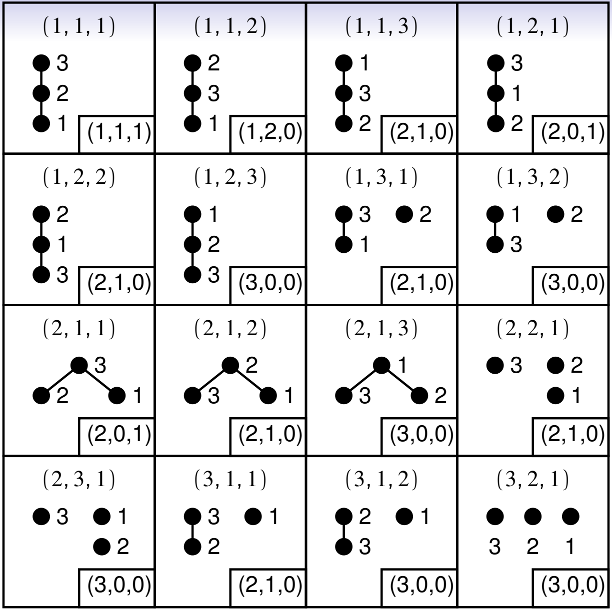
Theorem (S. 2008)

We have

$$\sum_{F \in F_n} \mathbf{q}^{\mathbf{inv}(F)} = \sum_{P \in PF_n} \mathbf{q}^{\mathbf{jump}(P)}$$

where $\mathbf{q}^{\mathbf{v}} = q_0^{v_0} q_1^{v_1} \cdots q_n^{v_n}$.

here


(1, 1, 1)
3 2 1
(1,1,1)
(1, 1, 2)
2 3 1
(1,2,0)
(1, 1, 3)
1 3 2
(2,1,0)
(1, 2, 1)
3 1 2
(2,0,1)
(1, 2, 2)
2 1 3
(2,1,0)
(1, 2, 3)
1 2 3
(3,0,0)
(1, 3, 1)
3 2 1
(2,1,0)
(1, 3, 2)
1 2 3
(3,0,0)
(2, 1, 1)
3 2 1
(2,0,1)
(2, 1, 2)
2 3 1
(2,1,0)
(2, 1, 3)
1 3 2
(3,0,0)
(2, 2, 1)
3 2 1
(2,1,0)
(2, 3, 1)
3 1 2
(3,0,0)
(3, 1, 1)
3 1 2
(2,1,0)
(3, 1, 2)
2 1 3
(3,0,0)
(3, 2, 1)
3 2 1
(3,0,0)

Question

How many forests with a given type of inversions are there?

There are $n!$ forests with type $(n, 0, \ldots, 0)$.
There is only one forest with $(1, \ldots, 1)$.
But I know nothing about general cases yet.

References

- Ira M. Gessel and Seunghyun Seo, *A refinement of Cayley's formula for trees*, Electron. J. Combin. **11** (2004/06), no. 2, Research Paper 27, 23 pp. (electronic). MR MR2224940 (2006m:05010)

- G. Kreweras, *Une famille de polynômes ayant plusieurs propriétés énumeratives*, Period. Math. Hungar. **11** (1980), no. 4, 309–320. MR MR603398 (82f:05007)

- Seunghyun Seo and Heesung Shin, *A generalized enumeration of labeled trees and reverse Prüfer algorithm*, J. Combin. Theory Ser. A **114** (2007), no. 7, 1357–1361. MR MR2353128

- Richard P. Stanley, *An introduction to hyperplane arrangements*, Geometric combinatorics, IAS/Park City Math. Ser., vol. 13, Amer. Math. Soc., Providence, RI, 2007, pp. 389–496. MR MR2383131

Thank you for listening!

hshin@math.univ-lyon1.fr

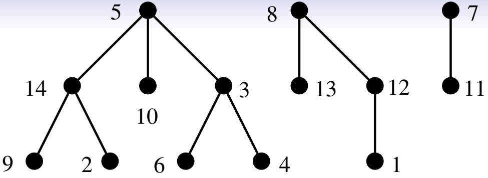
Parking Algorithm

$PA(p_1, \ldots, p_n)$

- $E_1 = \{1, \ldots, n\}$
- for $i = 1, \ldots, n$ do
  - $q_i = \min(E_1 \setminus \{p_i, \ldots, n\})$
  - $E_{i+1} = E_i \setminus \{q_i\}$

  end do
- return $(q_1, \ldots, q_n)$

◂ here



$$\text{tree}(F) = 3$$

$$\Downarrow \varphi$$

$$(10, 2, 6, 5, 7, 1, 13, 10, 4, 1, 14, 9, 11, 5)$$

Parking Space	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Cars' Number c	6	2	10	9	4	3	5	14	12	1	8	13	7	11

$$\text{critical}(P) = 3$$

Theorem (S. 2008)

Moreover, we have

$$\sum_{F \in F_n} \mathbf{q}^{\mathbf{inv}(F)} c^{\mathrm{tree}(F)} = \sum_{P \in PF_n} \mathbf{q}^{\mathbf{jump}(P)} c^{\mathrm{critical}(P)}$$

where $\mathbf{q}^{\mathbf{v}} = q_0^{v_0} q_1^{v_1} \cdots q_n^{v_n}$*.*

◂ here