Parking functions and noncrossing partitions of graphs

Josh Hallam
Loyola Marymount University

UNCG ANT-CoG Seminar

March 17, 2023

Outline

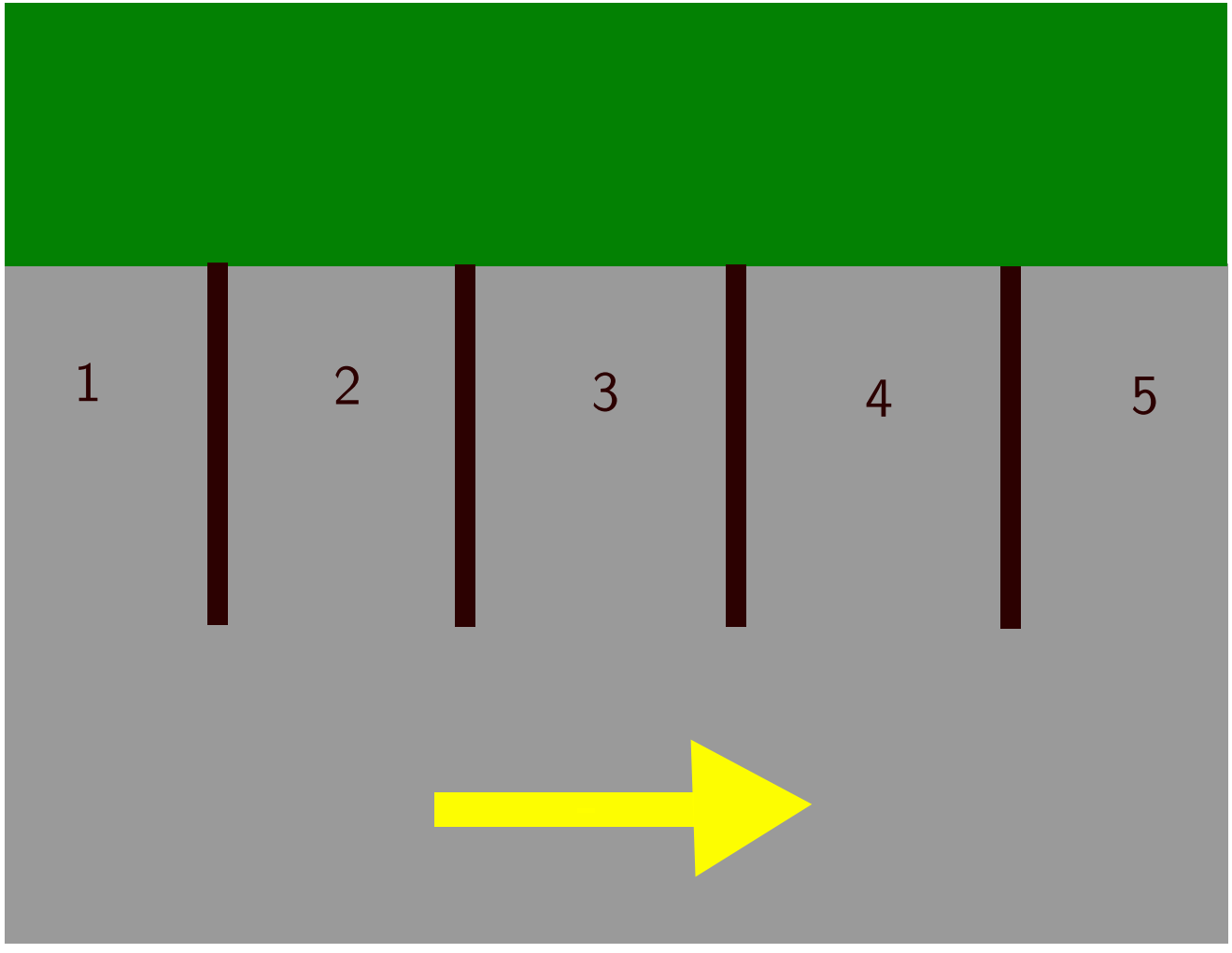
Parking Functions

Noncrossing Partitions

Noncrossing Partitions of Graphs

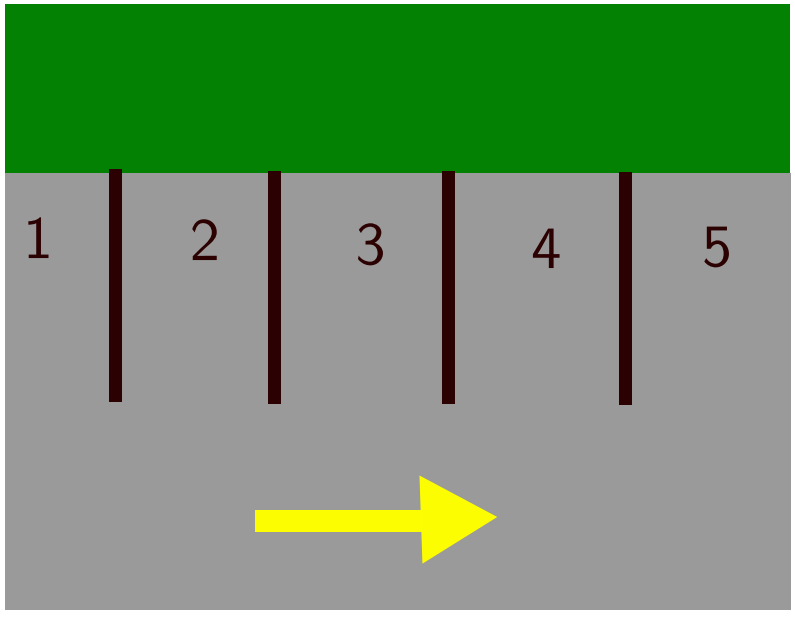
Parking Functions

Imagine you have a one-way street with parking spots labelled by $1, 2, \ldots, n$.



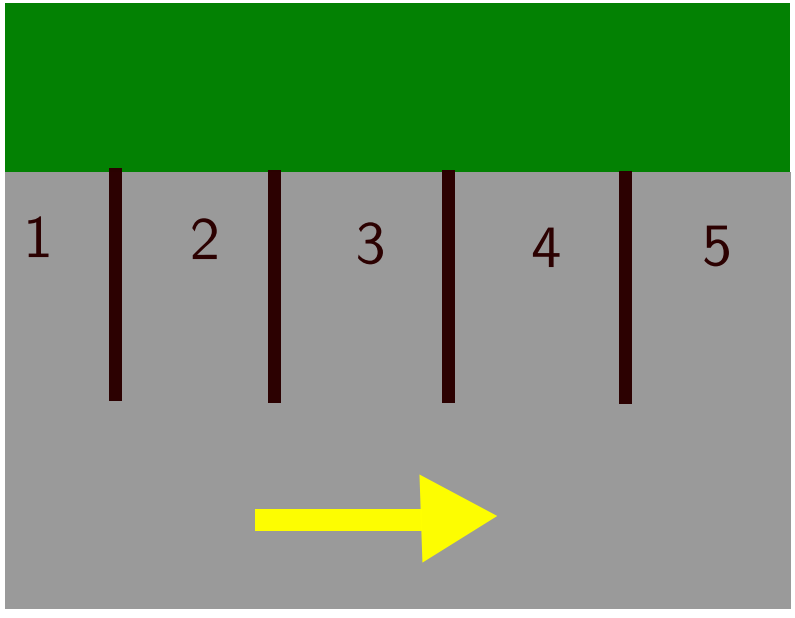
Parking Functions

Imagine you have a one-way street with parking spots labelled by $1, 2, \ldots, n$. Then n cars come to park, each with a preferred spot.



Parking Functions

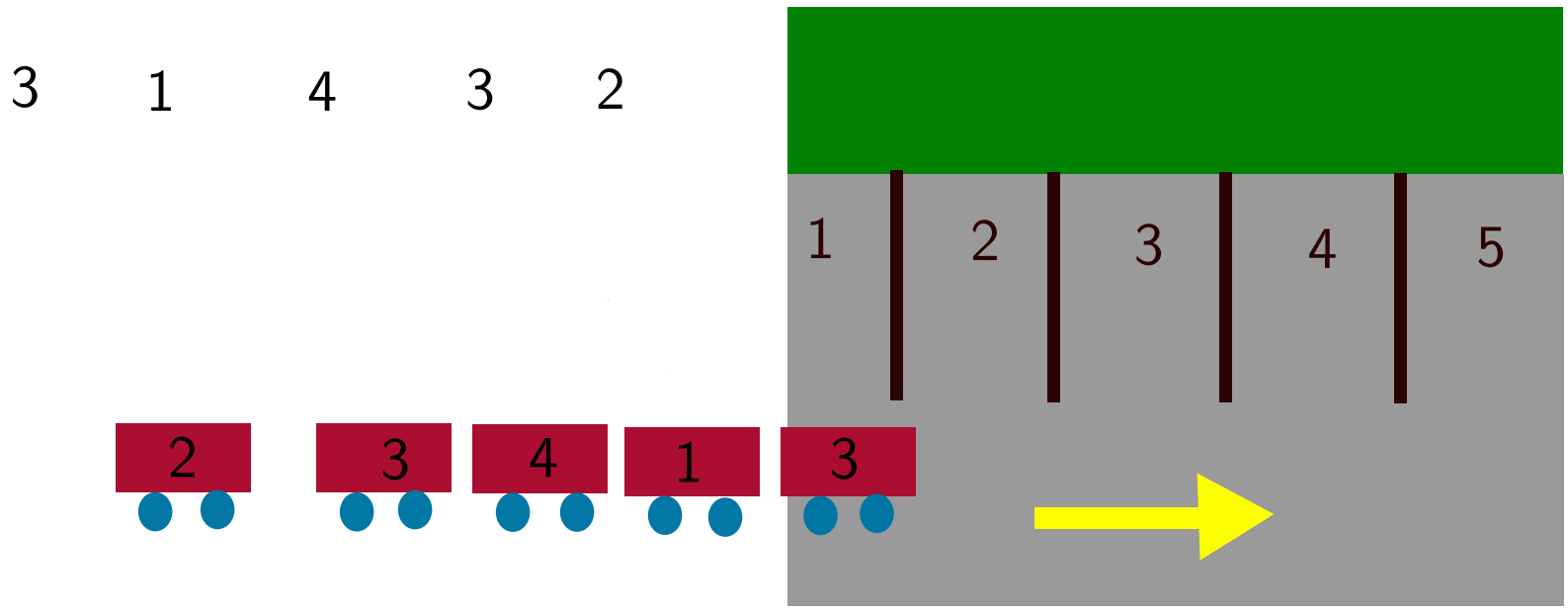
Imagine you have a one-way street with parking spots labelled by $1, 2, \ldots, n$. Then n cars come to park, each with a preferred spot.



Each car enters the parking lot and goes to its preferred spot. If it is empty, it parks. Otherwise it continues to the next available spot.

Parking Functions

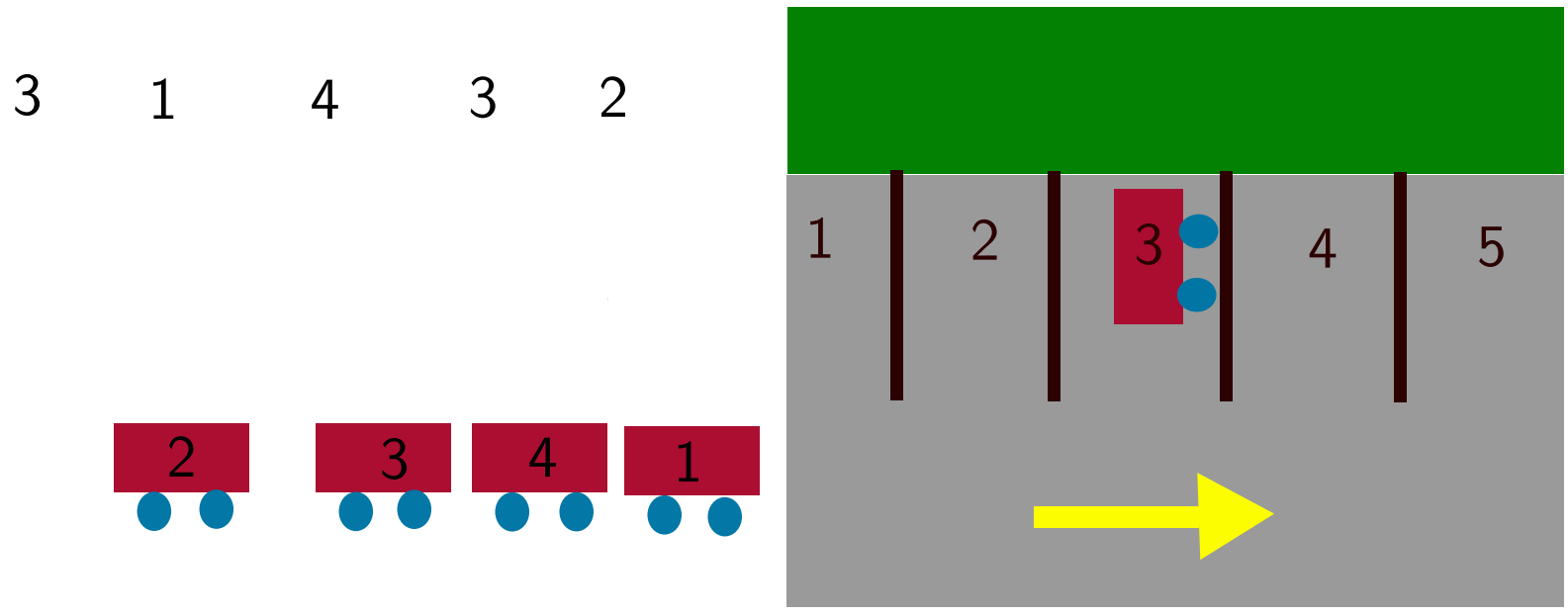
Imagine you have a one-way street with parking spots labelled by $1, 2, \ldots, n$. Then n cars come to park, each with a preferred spot.



Each car enters the parking lot and goes to its preferred spot. If it is empty, it parks. Otherwise it continues to the next available spot.

Parking Functions

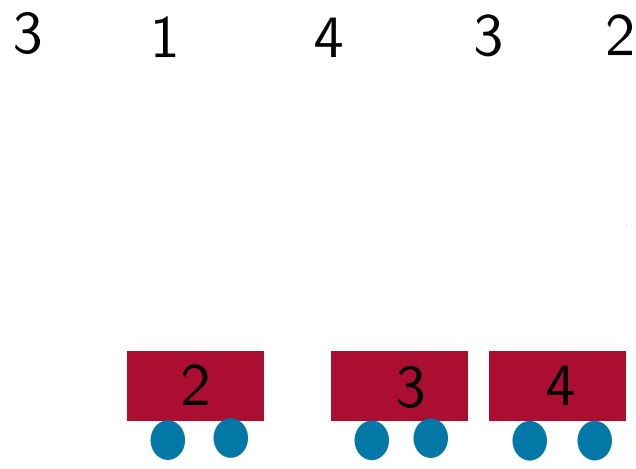
Imagine you have a one-way street with parking spots labelled by $1, 2, \ldots, n$. Then n cars come to park, each with a preferred spot.

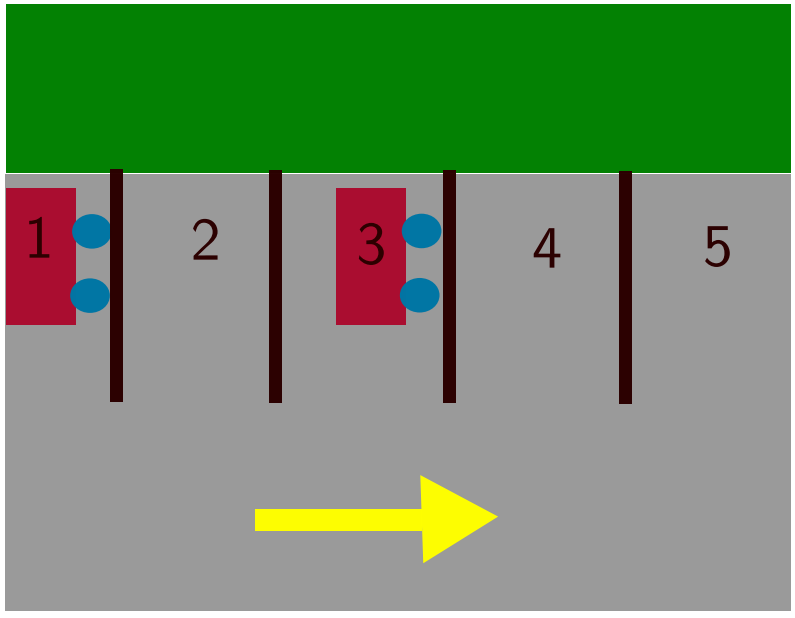


Each car enters the parking lot and goes to its preferred spot. If it is empty, it parks. Otherwise it continues to the next available spot.

Parking Functions

Imagine you have a one-way street with parking spots labelled by $1, 2, \ldots, n$. Then n cars come to park, each with a preferred spot.



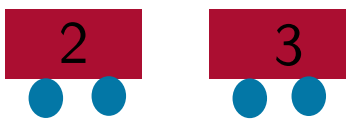


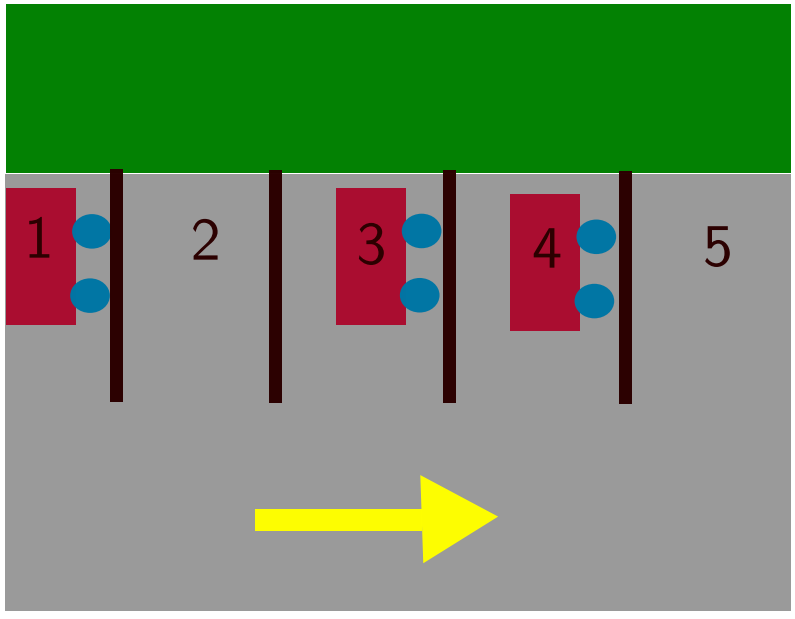
Each car enters the parking lot and goes to its preferred spot. If it is empty, it parks. Otherwise it continues to the next available spot.

Parking Functions

Imagine you have a one-way street with parking spots labelled by $1, 2, \ldots, n$. Then n cars come to park, each with a preferred spot.

3 1 4 3 2



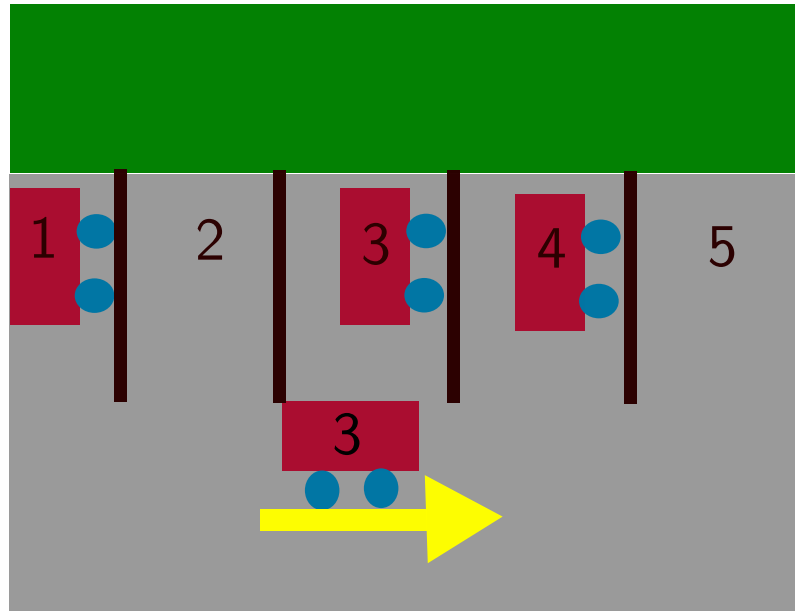


Each car enters the parking lot and goes to its preferred spot. If it is empty, it parks. Otherwise it continues to the next available spot.

Parking Functions

Imagine you have a one-way street with parking spots labelled by $1, 2, \ldots, n$. Then n cars come to park, each with a preferred spot.

3 1 4 3 2

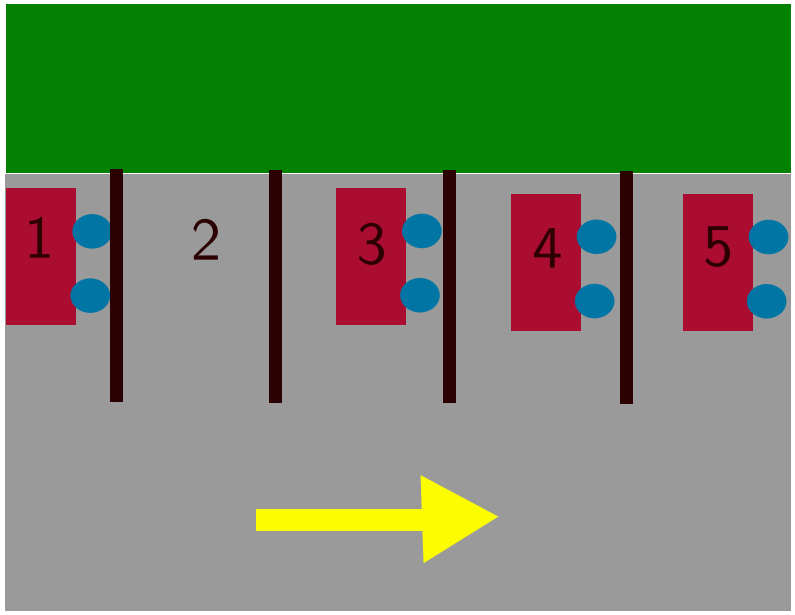


Each car enters the parking lot and goes to its preferred spot. If it is empty, it parks. Otherwise it continues to the next available spot.

Parking Functions

Imagine you have a one-way street with parking spots labelled by $1, 2, \ldots, n$. Then n cars come to park, each with a preferred spot.

3 1 4 3 2

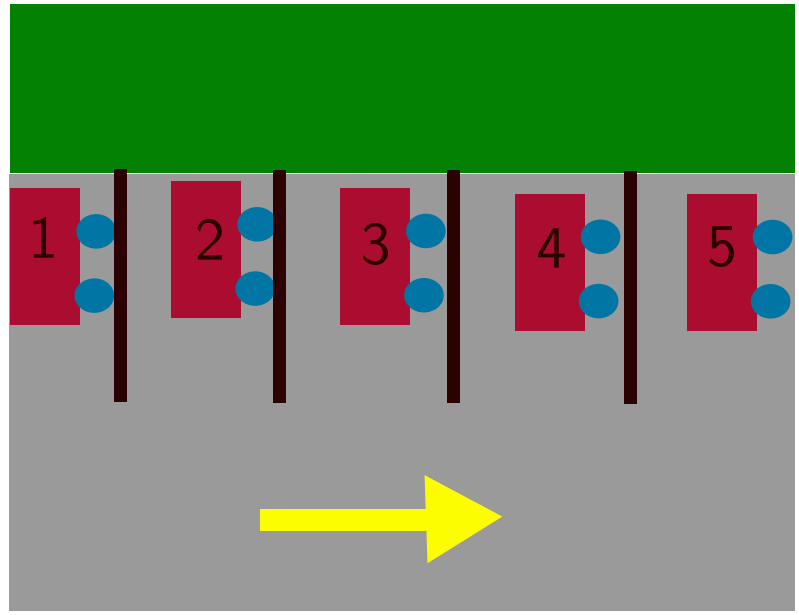


Each car enters the parking lot and goes to its preferred spot. If it is empty, it parks. Otherwise it continues to the next available spot.

Parking Functions

Imagine you have a one-way street with parking spots labelled by $1, 2, \ldots, n$. Then n cars come to park, each with a preferred spot.

3 1 4 3 2

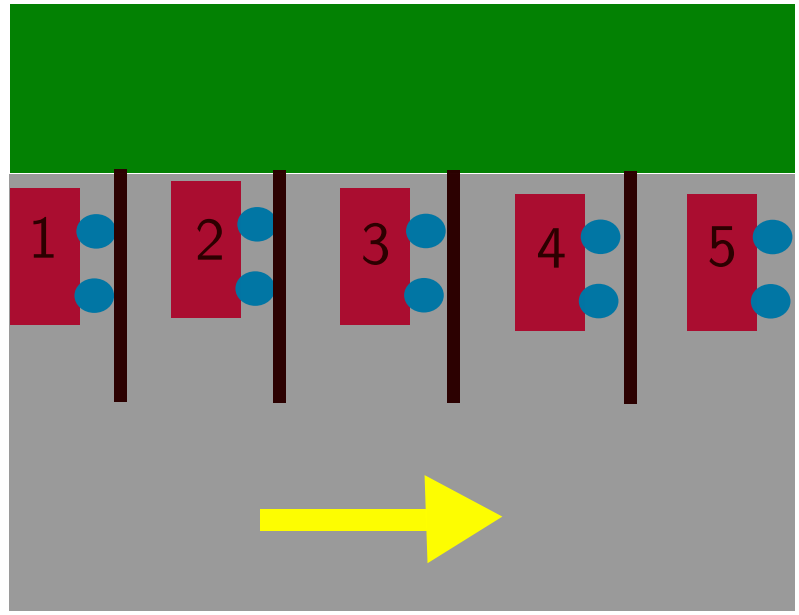


Each car enters the parking lot and goes to its preferred spot. If it is empty, it parks. Otherwise it continues to the next available spot.

Parking Functions

Imagine you have a one-way street with parking spots labelled by $1, 2, \ldots, n$. Then n cars come to park, each with a preferred spot.

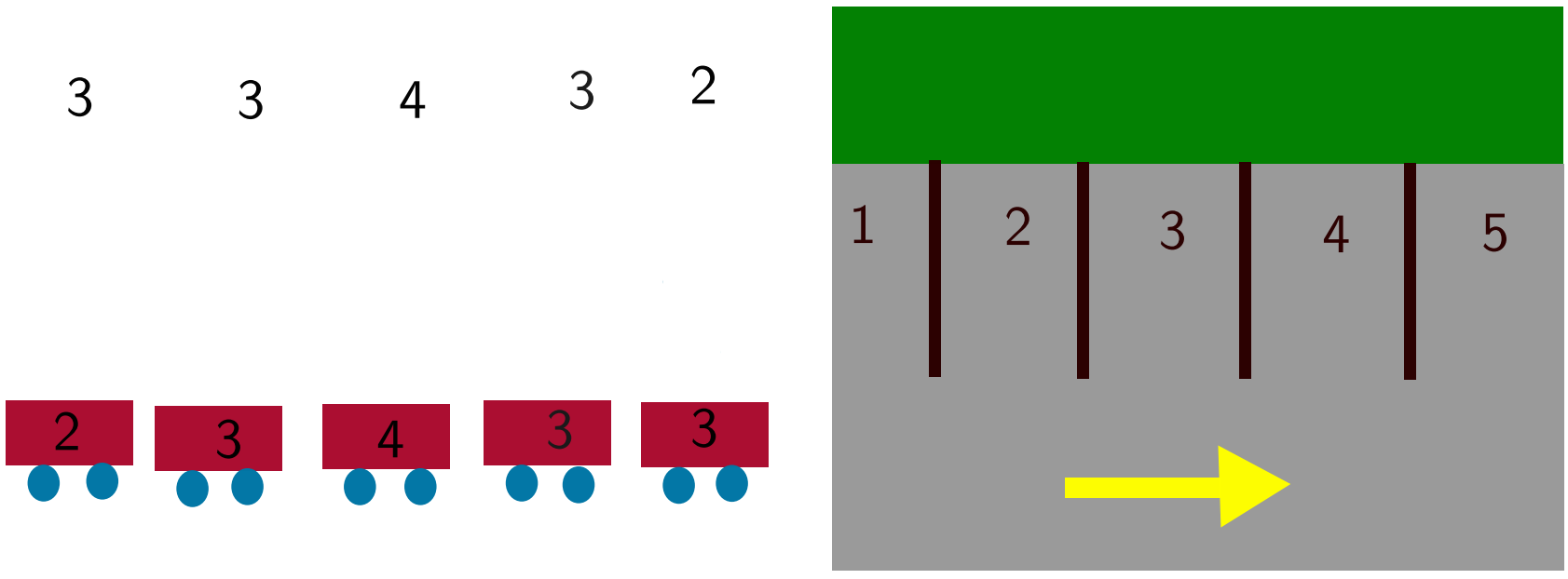
3 1 4 3 2



Each car enters the parking lot and goes to its preferred spot. If it is empty, it parks. Otherwise it continues to the next available spot. So with preferences in the order 31432, all cars could park.

Parking Functions

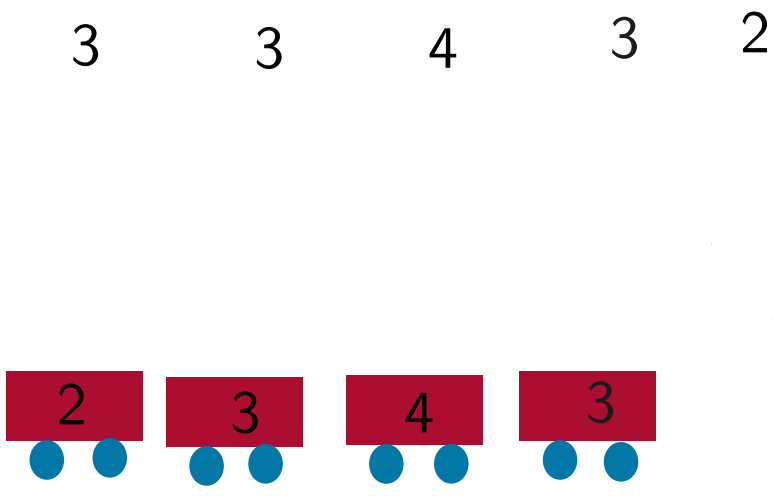
Imagine you have a one-way street with parking spots labelled by $1, 2, \ldots, n$. Then n cars come to park, each with a preferred spot.

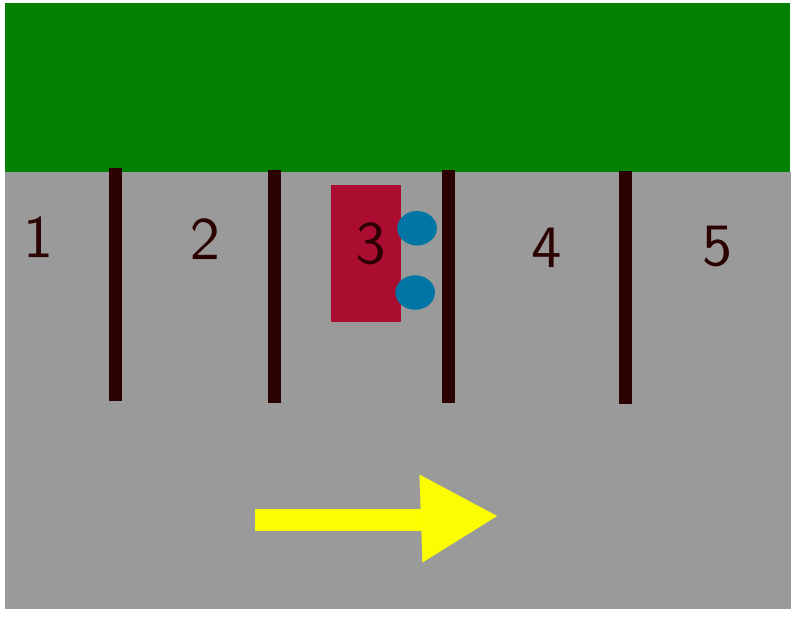


Each car enters the parking lot and goes to its preferred spot. If it is empty, it parks. Otherwise it continues to the next available spot.

Parking Functions

Imagine you have a one-way street with parking spots labelled by $1, 2, \ldots, n$. Then n cars come to park, each with a preferred spot.

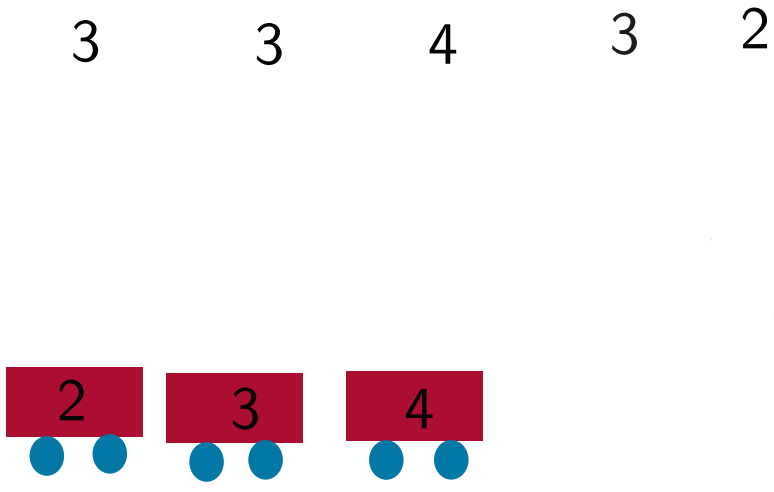


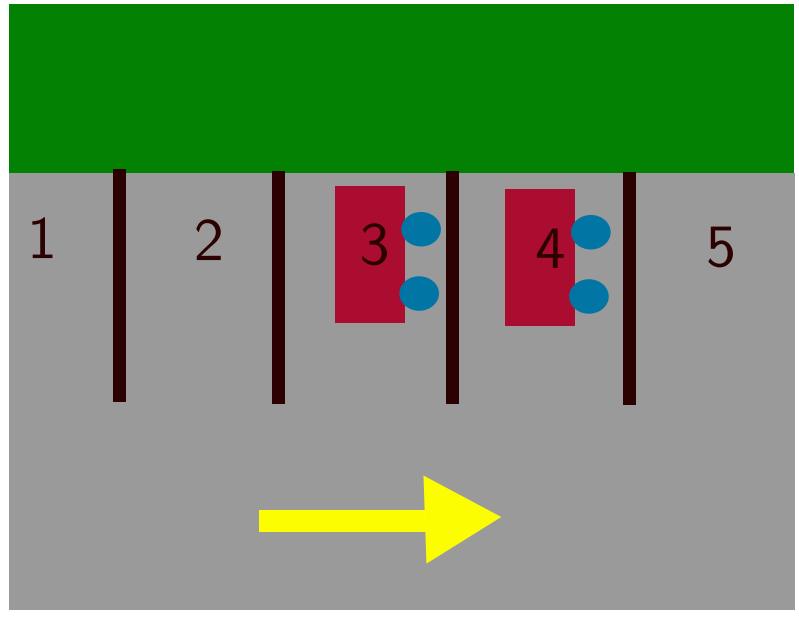


Each car enters the parking lot and goes to its preferred spot. If it is empty, it parks. Otherwise it continues to the next available spot.

Parking Functions

Imagine you have a one-way street with parking spots labelled by $1, 2, \ldots, n$. Then n cars come to park, each with a preferred spot.



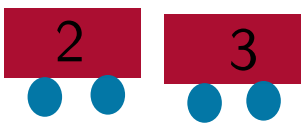


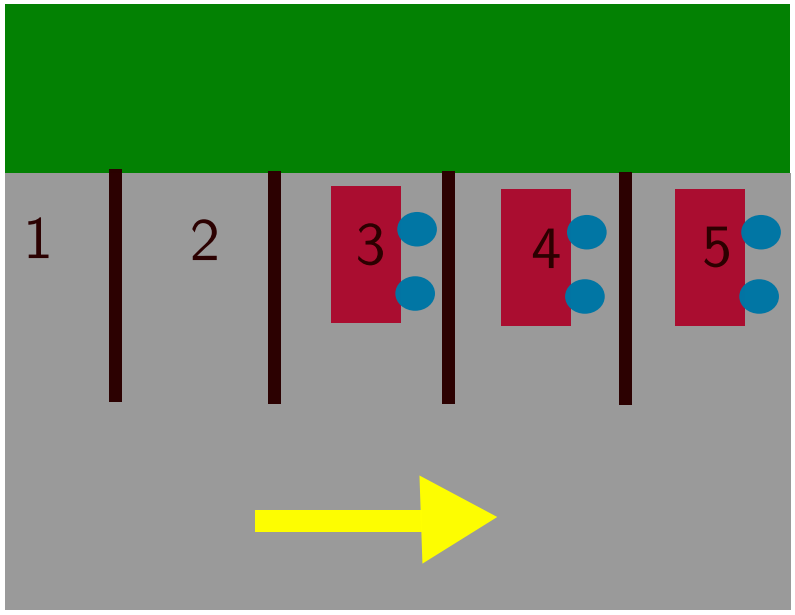
Each car enters the parking lot and goes to its preferred spot. If it is empty, it parks. Otherwise it continues to the next available spot.

Parking Functions

Imagine you have a one-way street with parking spots labelled by $1, 2, \ldots, n$. Then n cars come to park, each with a preferred spot.

3 3 4 3 2



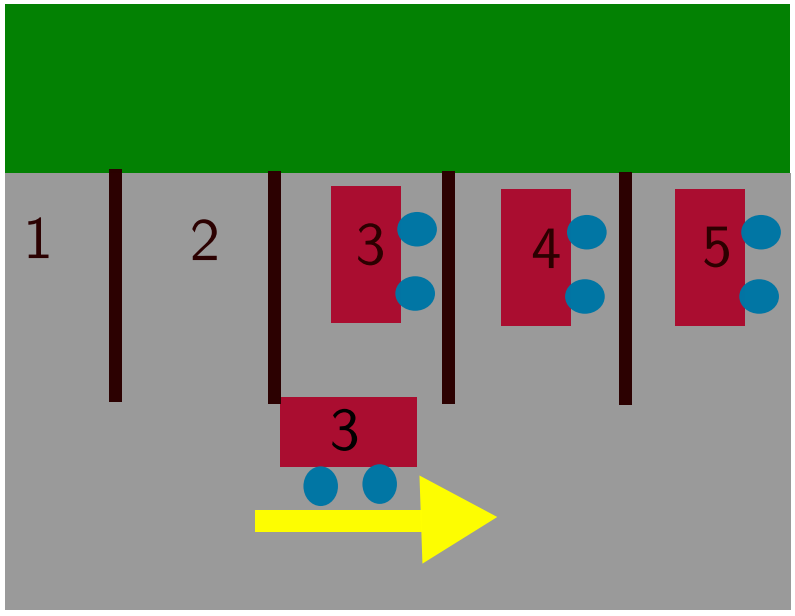


Each car enters the parking lot and goes to its preferred spot. If it is empty, it parks. Otherwise it continues to the next available spot.

Parking Functions

Imagine you have a one-way street with parking spots labelled by $1, 2, \ldots, n$. Then n cars come to park, each with a preferred spot.

3 3 4 3 2

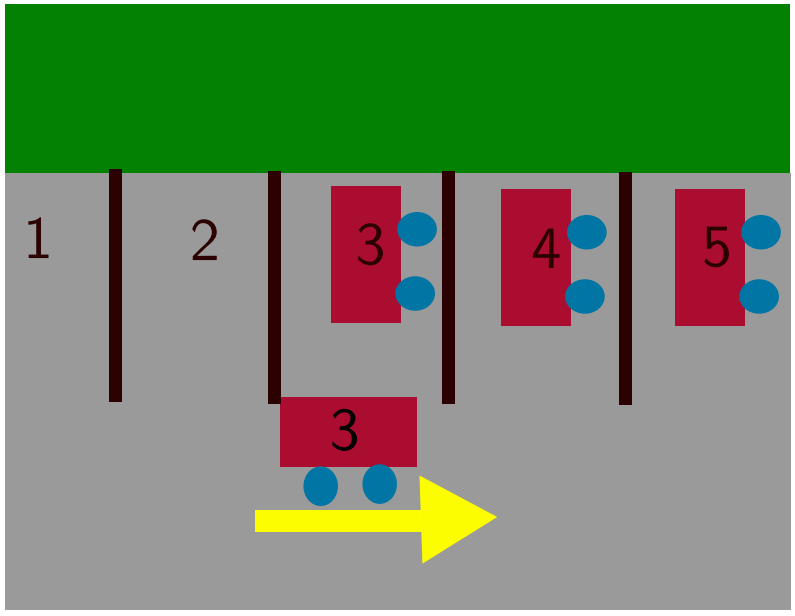


Each car enters the parking lot and goes to its preferred spot. If it is empty, it parks. Otherwise it continues to the next available spot.

Parking Functions

Imagine you have a one-way street with parking spots labelled by $1, 2, \ldots, n$. Then n cars come to park, each with a preferred spot.

3 3 4 3 2



Each car enters the parking lot and goes to its preferred spot. If it is empty, it parks. Otherwise it continues to the next available spot. The fourth car can't park.

Parking Functions

We say a sequence of positive integers of length n is a **parking function** if cars with those parking preferences can park with the rules explained previously.

Parking Functions

We say a sequence of positive integers of length n is a **parking function** if cars with those parking preferences can park with the rules explained previously.

For example, 31432 is a parking function, but 33432 is not.

Parking Functions

We say a sequence of positive integers of length n is a **parking function** if cars with those parking preferences can park with the rules explained previously.

For example, 31432 is a parking function, but 33432 is not.

Parking functions were initially introduced by Konheim and Weiss in 1966 as a way to study storing data on a disk.

Another Definition

Question: In a parking function, at least how many cars have preference 1?

Another Definition

Question: In a parking function, at least how many cars have preference 1?
Answer : At least one car.

Another Definition

Question: In a parking function, at least how many cars have preference 1?
Answer : At least one car.

Question: In a parking function, at least how many cars have preference 1 or 2?

Another Definition

Question: In a parking function, at least how many cars have preference 1?
Answer : At least one car.

Question: In a parking function, at least how many cars have preference 1 or 2?
Answer : At least two cars.

Another Definition

Question: In a parking function, at least how many cars have preference 1?
Answer : At least one car.

Question: In a parking function, at least how many cars have preference 1 or 2?
Answer : At least two cars.

More generally, if $\pi_1\pi_2\cdots\pi_n$ is a parking function, then it must be the case that

$$|\{i \mid \pi_i \leq j\}| \geq j$$

for all $1 \leq j \leq n$.

Another Definition

Question: In a parking function, at least how many cars have preference 1?
Answer : At least one car.

Question: In a parking function, at least how many cars have preference 1 or 2?
Answer : At least two cars.

More generally, if $\pi_1\pi_2\cdots\pi_n$ is a parking function, then it must be the case that

$$|\{i \mid \pi_i \leq j\}| \geq j$$

for all $1 \leq j \leq n$. Maybe somewhat surprisingly, this condition is also sufficient.

Another Definition

So we have the following equivalent definition.

A sequence $\pi_1\pi_2\cdots\pi_n$ is a **parking function** if for all $1 \leq j \leq n$, $|\{i \mid \pi_i \leq j\}| \geq j$.

Another Definition

So we have the following equivalent definition.

A sequence $\pi_1\pi_2\cdots\pi_n$ is a **parking function** if for all $1 \leq j \leq n$, $|\{i \mid \pi_i \leq j\}| \geq j$.

Thus, 31432 is a parking function, but 33432 is not (the $j = 1$ and $j = 2$ cases do not hold).

Counting Parking Functions

A natural question to ask is how many parking functions of length n there are.

Counting Parking Functions

A natural question to ask is how many parking functions of length n there are.

Question: Are permutations also parking functions?

Counting Parking Functions

A natural question to ask is how many parking functions of length n there are.

Question: Are permutations also parking functions?

Answer : Yes! Each car will park in their preferred spot.

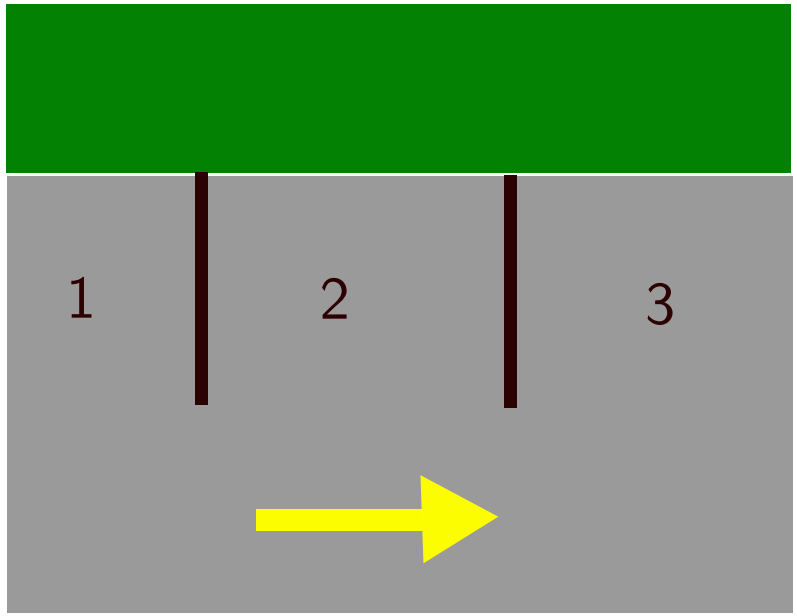
So there are at least $n!$ parking functions of length n. But clearly, there are more.

Counting Parking Functions

The following clever argument is due to Pollak (1974). Suppose you want to count parking functions of length n. Add an extra spot labeled $n+1$. Allow $n+1$ as a preference for the n cars. Also allow the cars to circle back to parking lot if they can't find a spot on their first pass.

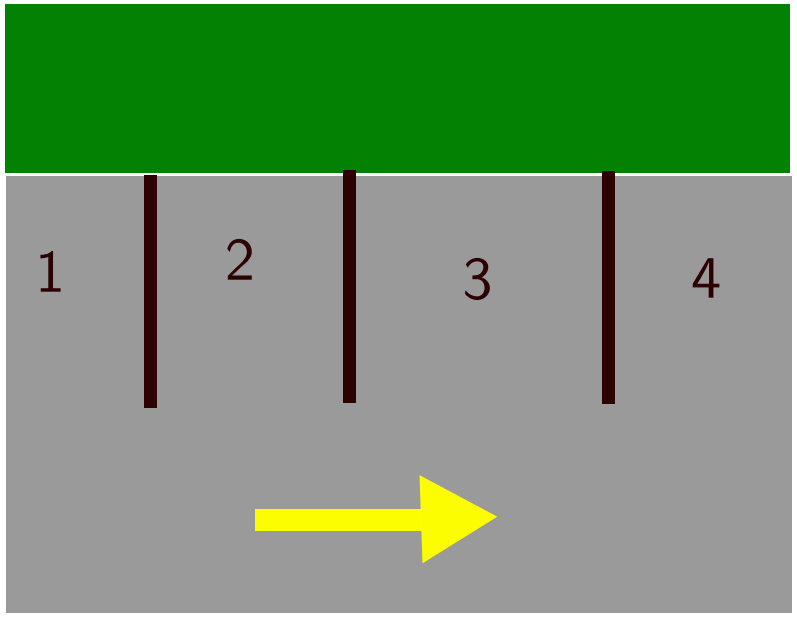
Counting Parking Functions

The following clever argument is due to Pollak (1974). Suppose you want to count parking functions of length n. Add an extra spot labeled $n+1$. Allow $n+1$ as a preference for the n cars. Also allow the cars to circle back to parking lot if they can't find a spot on their first pass.



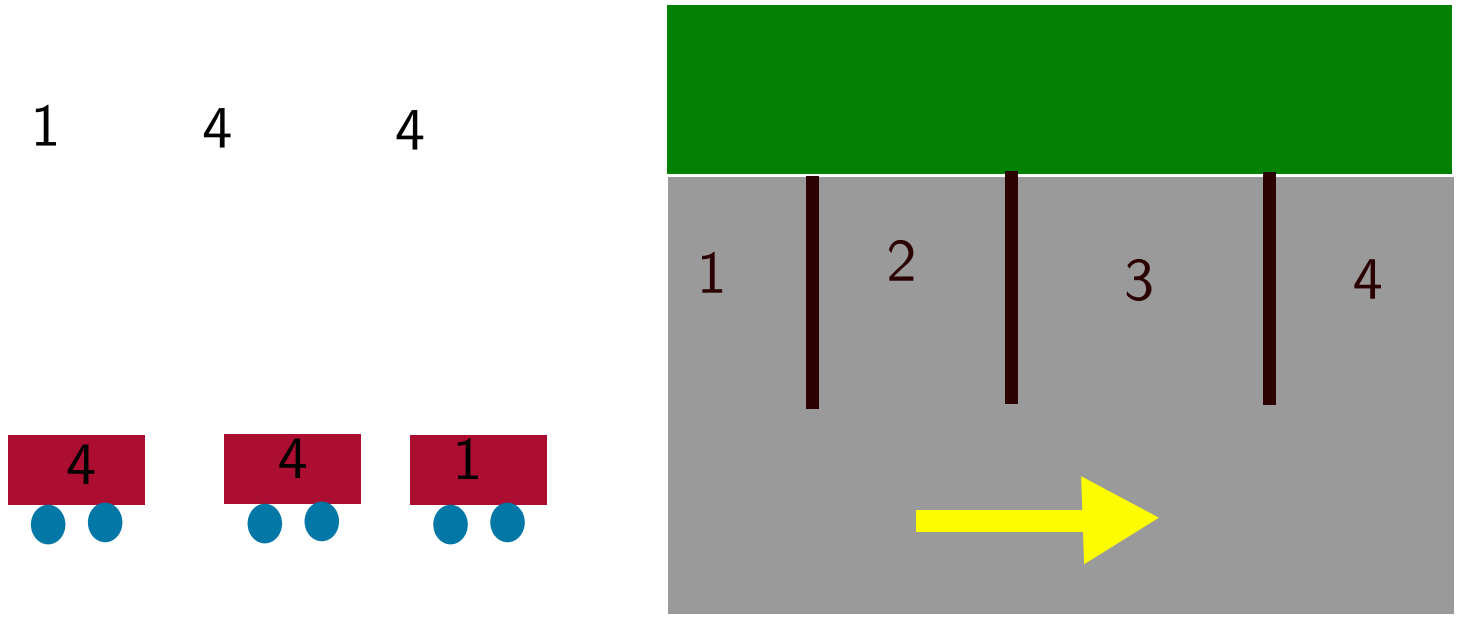
Counting Parking Functions

The following clever argument is due to Pollak (1974). Suppose you want to count parking functions of length n. Add an extra spot labeled $n+1$. Allow $n+1$ as a preference for the n cars. Also allow the cars to circle back to parking lot if they can't find a spot on their first pass.



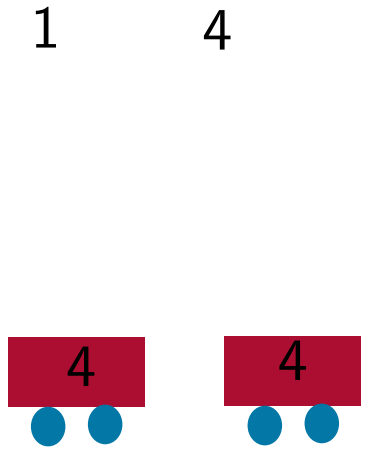
Counting Parking Functions

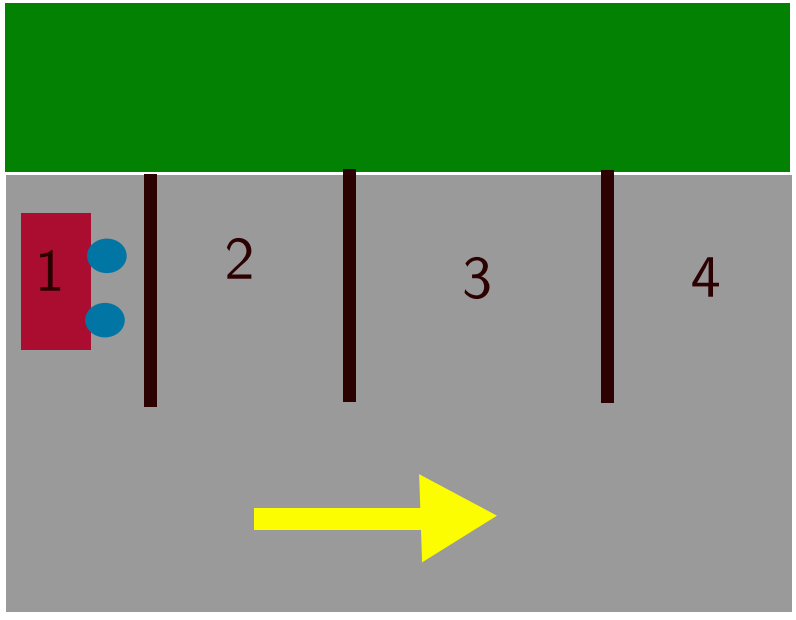
The following clever argument is due to Pollak (1974). Suppose you want to count parking functions of length n. Add an extra spot labeled $n+1$. Allow $n+1$ as a preference for the n cars. Also allow the cars to circle back to parking lot if they can't find a spot on their first pass.



Counting Parking Functions

The following clever argument is due to Pollak (1974). Suppose you want to count parking functions of length n. Add an extra spot labeled $n+1$. Allow $n+1$ as a preference for the n cars. Also allow the cars to circle back to parking lot if they can't find a spot on their first pass.

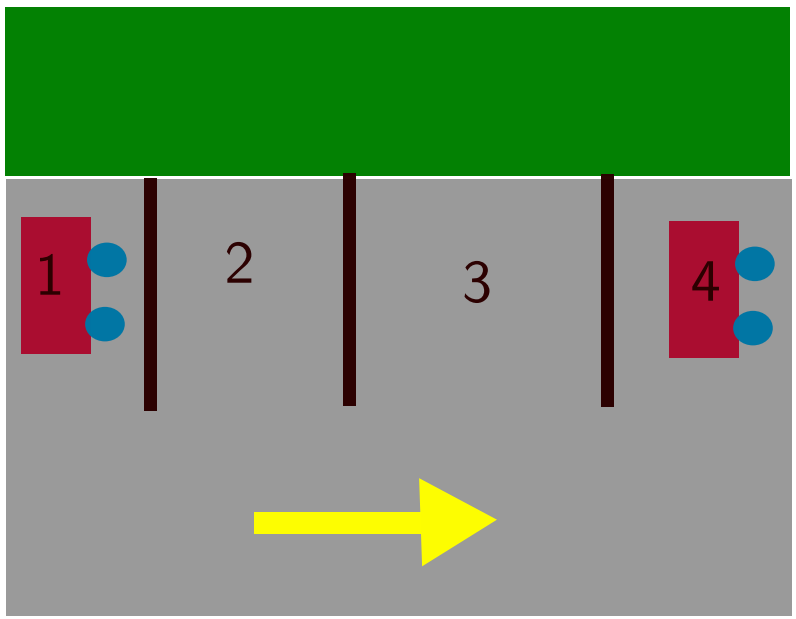




Counting Parking Functions

The following clever argument is due to Pollak (1974). Suppose you want to count parking functions of length n. Add an extra spot labeled $n+1$. Allow $n+1$ as a preference for the n cars. Also allow the cars to circle back to parking lot if they can't find a spot on their first pass.

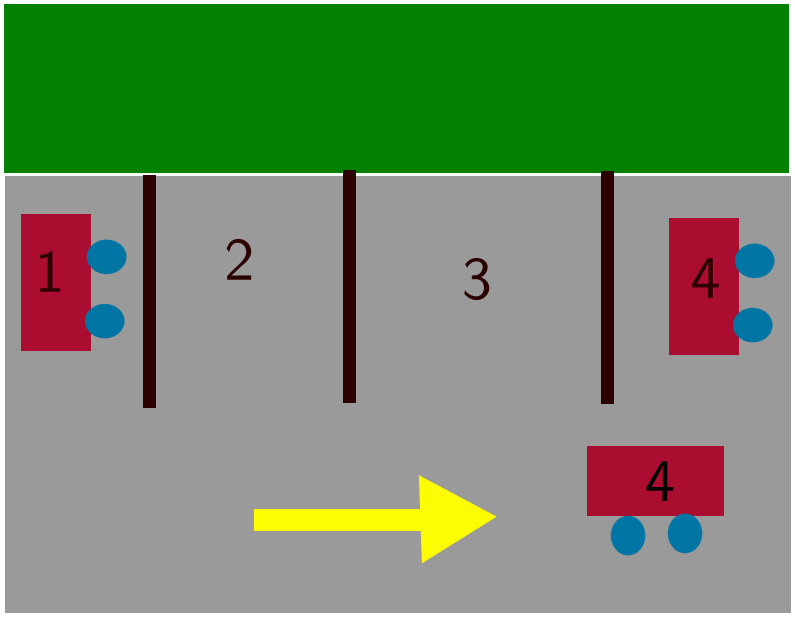
1 4 4



Counting Parking Functions

The following clever argument is due to Pollak (1974). Suppose you want to count parking functions of length n. Add an extra spot labeled $n+1$. Allow $n+1$ as a preference for the n cars. Also allow the cars to circle back to parking lot if they can't find a spot on their first pass.

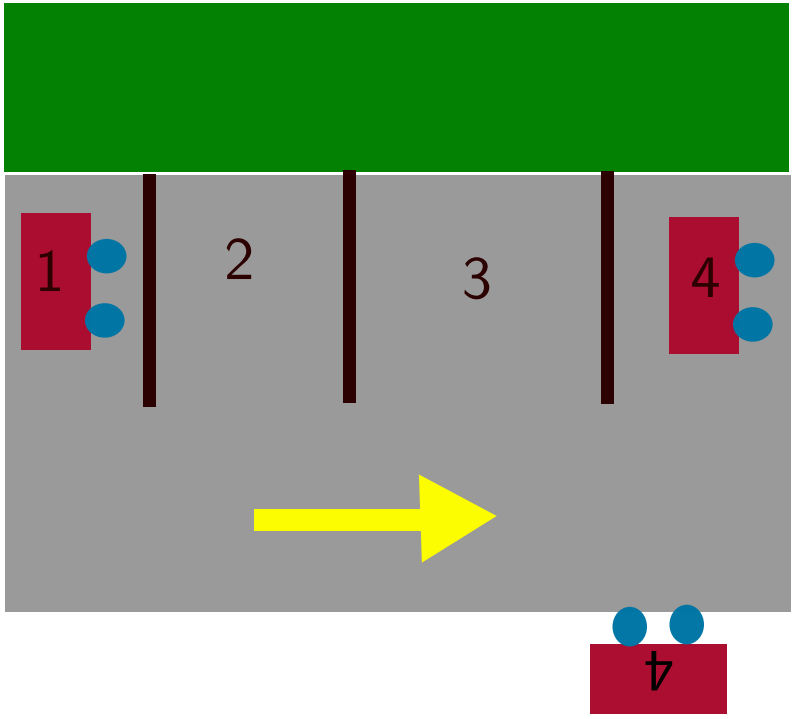
1 4 4



Counting Parking Functions

The following clever argument is due to Pollak (1974). Suppose you want to count parking functions of length n. Add an extra spot labeled $n+1$. Allow $n+1$ as a preference for the n cars. Also allow the cars to circle back to parking lot if they can't find a spot on their first pass.

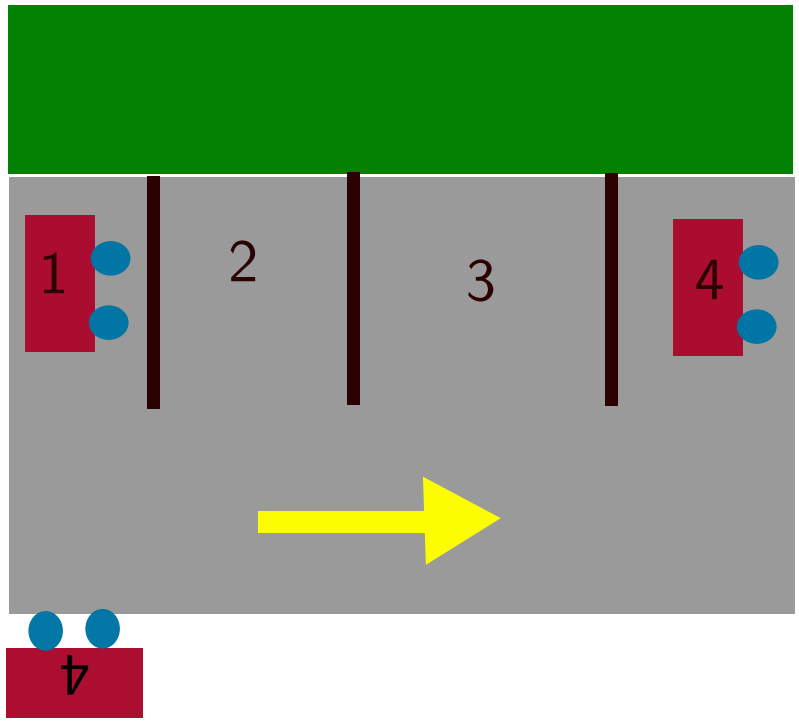
1 4 4



Counting Parking Functions

The following clever argument is due to Pollak (1974). Suppose you want to count parking functions of length n. Add an extra spot labeled $n+1$. Allow $n+1$ as a preference for the n cars. Also allow the cars to circle back to parking lot if they can't find a spot on their first pass.

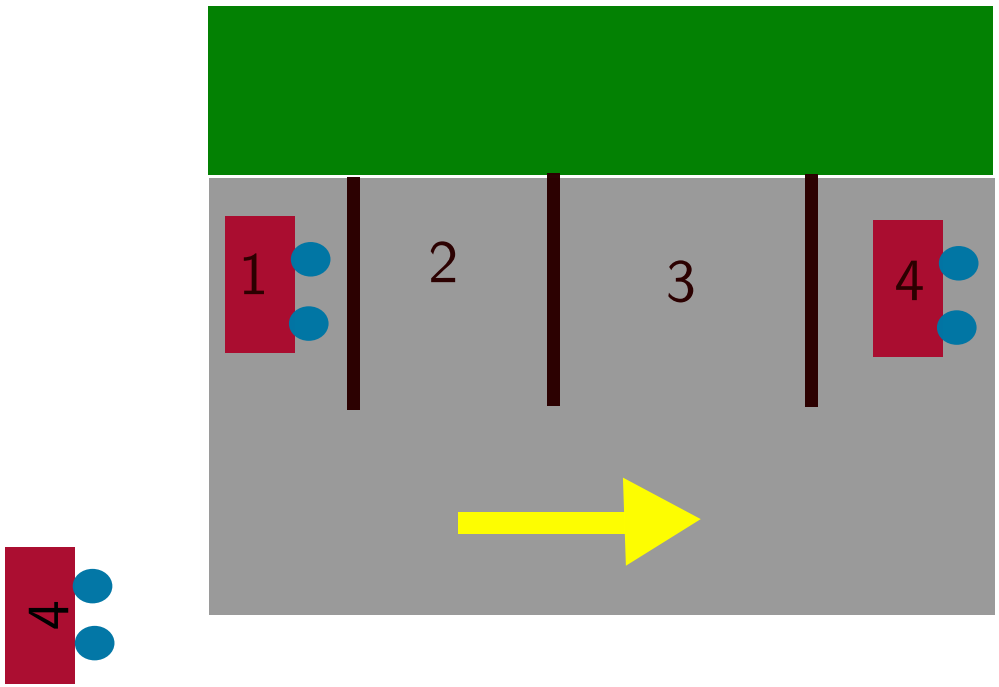
1 4 4



Counting Parking Functions

The following clever argument is due to Pollak (1974). Suppose you want to count parking functions of length n. Add an extra spot labeled $n+1$. Allow $n+1$ as a preference for the n cars. Also allow the cars to circle back to parking lot if they can't find a spot on their first pass.

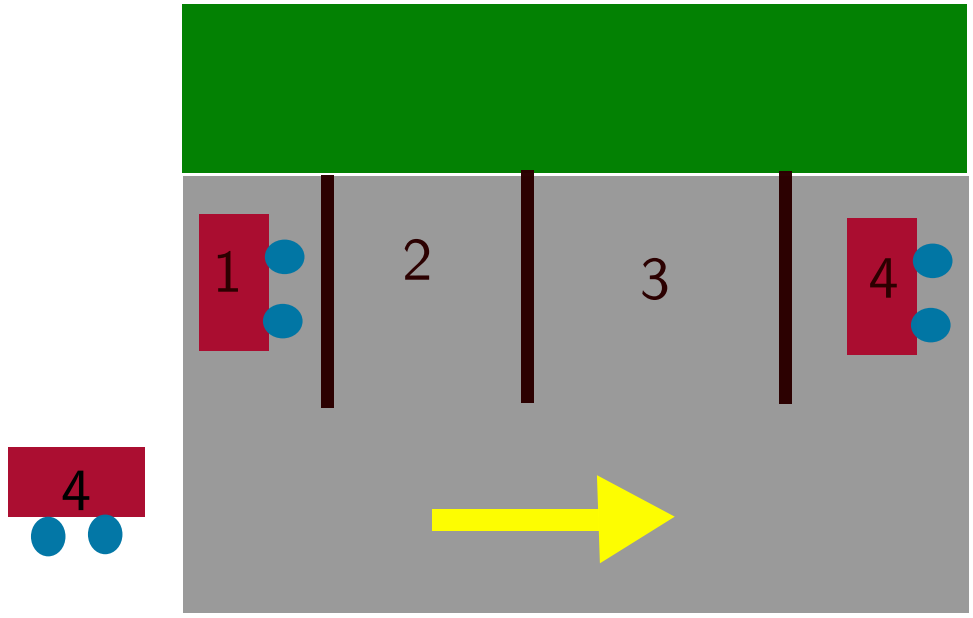
1 4 4



Counting Parking Functions

The following clever argument is due to Pollak (1974). Suppose you want to count parking functions of length n. Add an extra spot labeled $n+1$. Allow $n+1$ as a preference for the n cars. Also allow the cars to circle back to parking lot if they can't find a spot on their first pass.

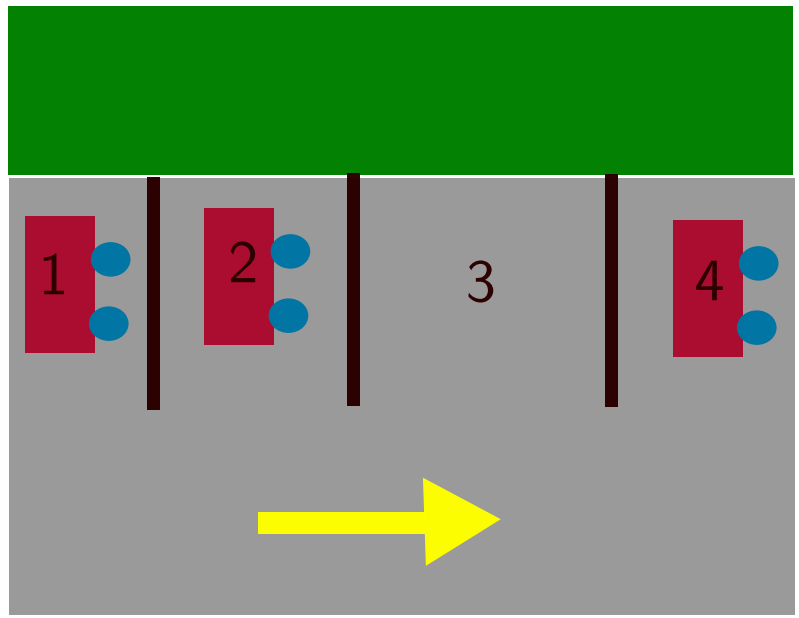
1 4 4



Counting Parking Functions

The following clever argument is due to Pollak (1974). Suppose you want to count parking functions of length n. Add an extra spot labeled $n+1$. Allow $n+1$ as a preference for the n cars. Also allow the cars to circle back to parking lot if they can't find a spot on their first pass.

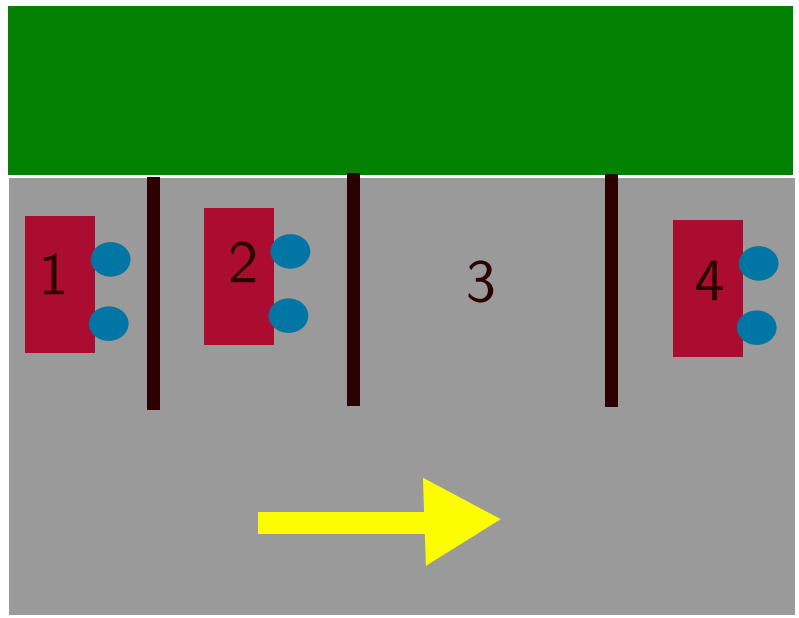
1 4 4



Counting Parking Functions

The following clever argument is due to Pollak (1974). Suppose you want to count parking functions of length n. Add an extra spot labeled $n+1$. Allow $n+1$ as a preference for the n cars. Also allow the cars to circle back to parking lot if they can't find a spot on their first pass.

1 4 4

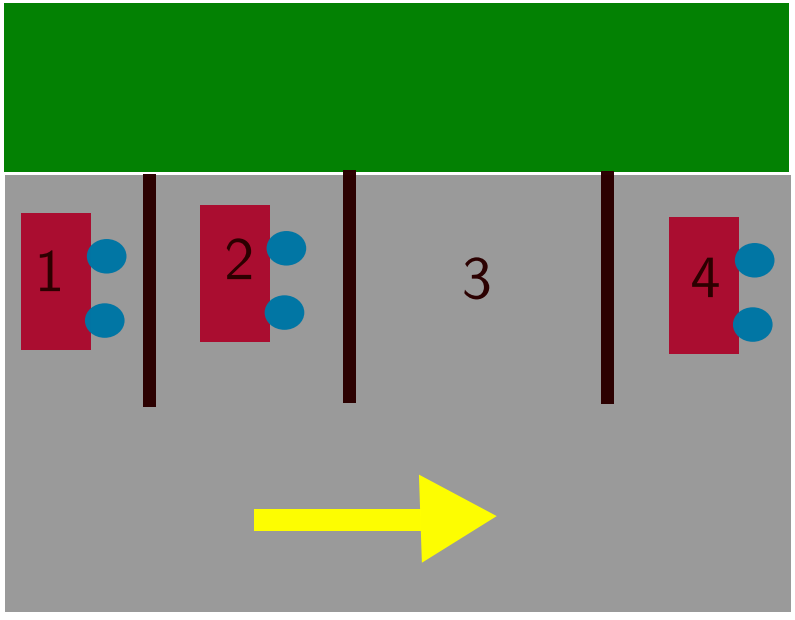


Note that all cars can park and there will be exactly one empty space.

Counting Parking Functions

The following clever argument is due to Pollak (1974). Suppose you want to count parking functions of length n. Add an extra spot labeled $n+1$. Allow $n+1$ as a preference for the n cars. Also allow the cars to circle back to parking lot if they can't find a spot on their first pass.

1 4 4 X

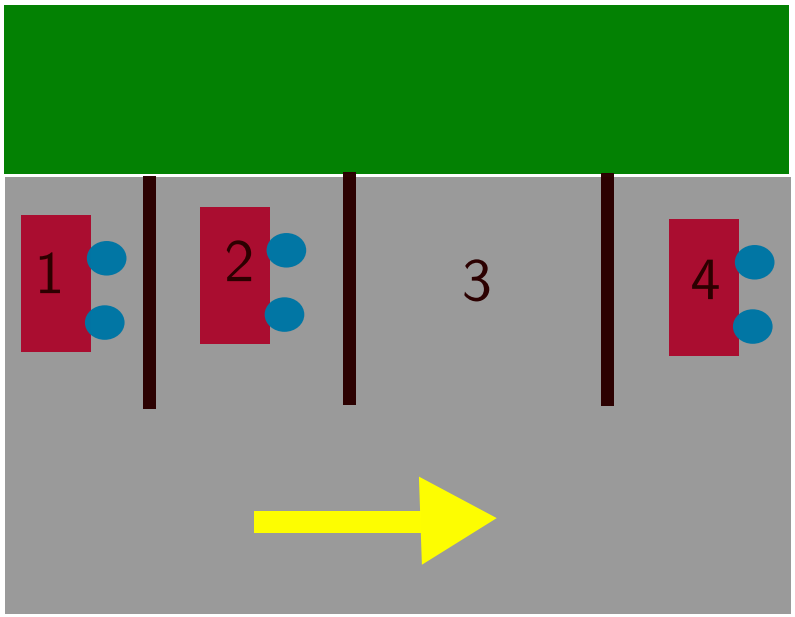


But this is not a valid parking function of length 3.

Counting Parking Functions

The following clever argument is due to Pollak (1974). Suppose you want to count parking functions of length n. Add an extra spot labeled $n+1$. Allow $n+1$ as a preference for the n cars. Also allow the cars to circle back to parking lot if they can't find a spot on their first pass.

1 4 4 X

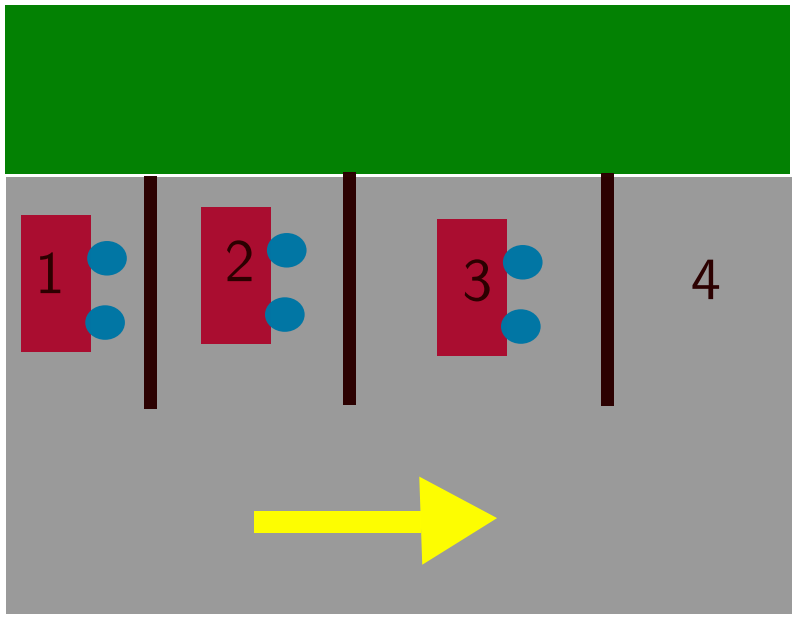


Add 1 to each entry modulo 4.

Counting Parking Functions

The following clever argument is due to Pollak (1974). Suppose you want to count parking functions of length n. Add an extra spot labeled $n+1$. Allow $n+1$ as a preference for the n cars. Also allow the cars to circle back to parking lot if they can't find a spot on their first pass.

1	4	4	X
2	1	1	✓

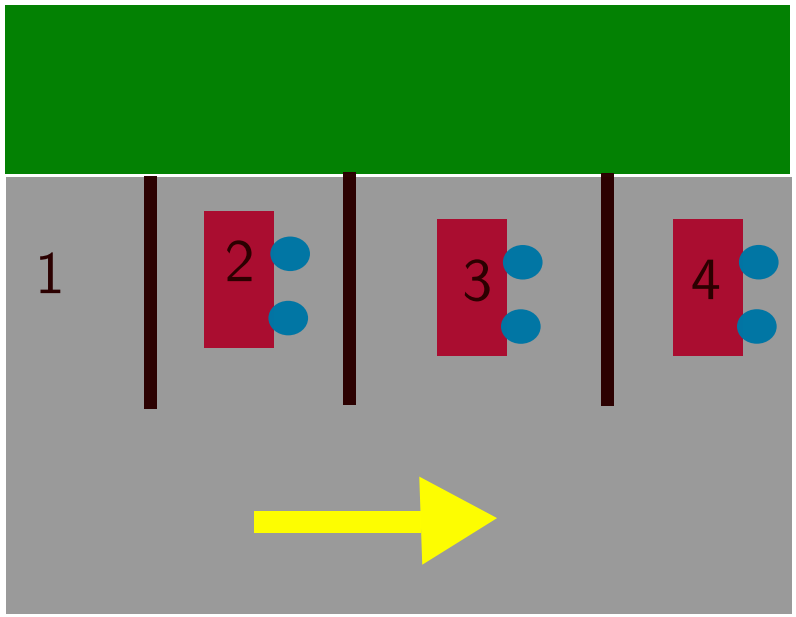


Add 1 to each entry modulo 4.

Counting Parking Functions

The following clever argument is due to Pollak (1974). Suppose you want to count parking functions of length n. Add an extra spot labeled $n+1$. Allow $n+1$ as a preference for the n cars. Also allow the cars to circle back to parking lot if they can't find a spot on their first pass.

1	4	4	X
2	1	1	✓
3	2	2	X

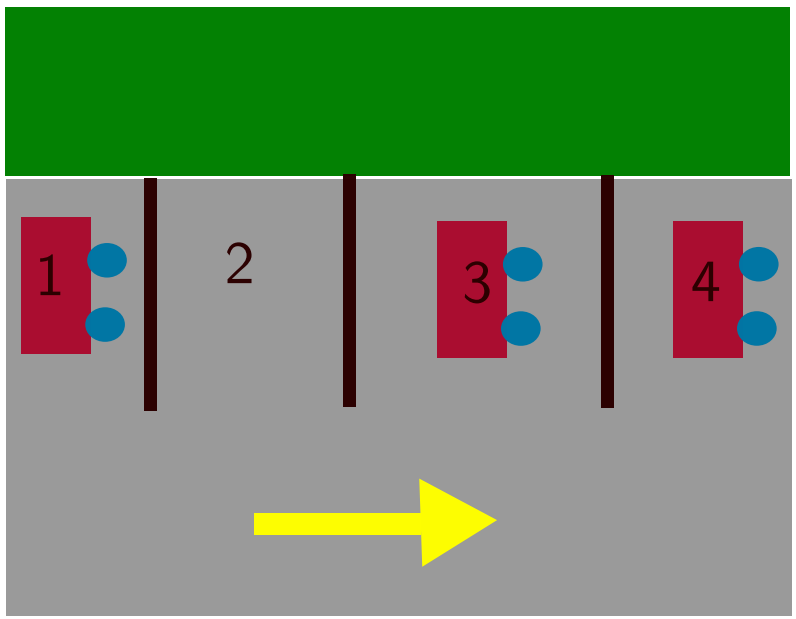


Add 1 to each entry modulo 4.

Counting Parking Functions

The following clever argument is due to Pollak (1974). Suppose you want to count parking functions of length n. Add an extra spot labeled $n+1$. Allow $n+1$ as a preference for the n cars. Also allow the cars to circle back to parking lot if they can't find a spot on their first pass.

1	4	4	X
2	1	1	✓
3	2	2	X
4	3	3	X

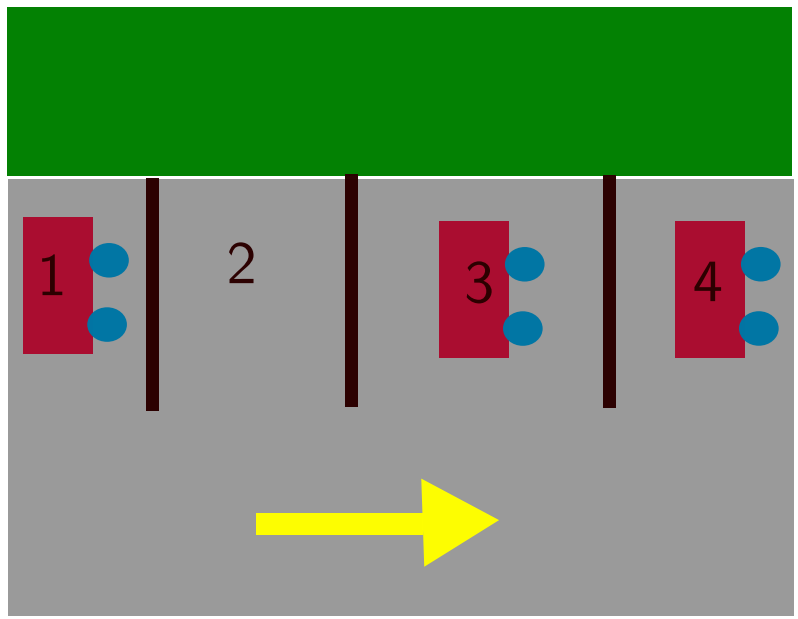


Exactly one of these is a parking function.

Counting Parking Functions

The following clever argument is due to Pollak (1974). Suppose you want to count parking functions of length n. Add an extra spot labeled $n+1$. Allow $n+1$ as a preference for the n cars. Also allow the cars to circle back to parking lot if they can't find a spot on their first pass.

1	4	4	X
2	1	1	✓
3	2	2	X
4	3	3	X



Exactly one of these is a parking function. There are 4^3 words of length 3 using $\{1,2,3,4\}$. Each group has 4 elements and exactly one is a parking function. So there are $4^3/4 = 4^2$ parking functions of length 3.

Counting Parking Functions

More generally, this gives an equivalence relation on words of length n using the characters $\{1, 2, \ldots, n+1\}$. Each equivalence class contains $n+1$ elements of which exactly one is a parking function. There are $(n+1)^n$ words of length n over $\{1, 2, \ldots, n+1\}$. So there are

$$\frac{(n+1)^n}{(n+1)} = (n+1)^{n-1}$$

parking functions of length n.

So the number of parking functions is: $1, 1, 3, 16, 125, \ldots$

Parking Functions Connections

Parking functions have applications in

- Computer Science
- Enumerative Combinatorics
- Algebraic Combinatorics
- Hyperplane Arrangements
- Group Theory

See Catherine Yan's chapter on parking functions in *Handbook of Enumerative Combinatorics* for more examples and generalizations of parking functions.

Noncrossing partitions

A **(set) partition** of $\{1, 2, \ldots, n\}$ is collection of nonempty subsets which are mutually disjoint and whose union is $\{1, 2, \ldots, n\}$.

For example, 145/23/6 and 1/245/36 are both set partitions of $\{1, 2, \ldots, 6\}$.

Noncrossing partitions

A **(set) partition** of $\{1, 2, \ldots, n\}$ is collection of nonempty subsets which are mutually disjoint and whose union is $\{1, 2, \ldots, n\}$.

For example, $145/23/6$ and $1/245/36$ are both set partitions of $\{1, 2, \ldots, 6\}$.

We say partition $B_1/B_2/\cdots/B_k$ is **crossing** if there exists $i \neq j$ and $\mathbf{a} < \mathbf{b} < \mathbf{c} < \mathbf{d}$ with $a, c \in B_i$ and $b, d \in B_j$. We say a partition is **noncrossing** if it is not crossing.

Noncrossing partitions

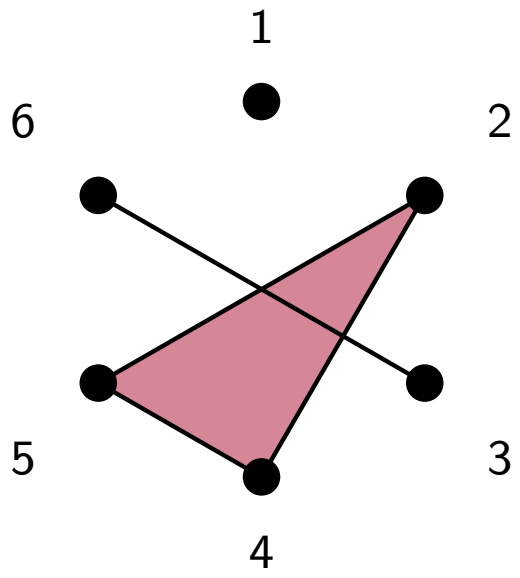
A **(set) partition** of $\{1, 2, \ldots, n\}$ is collection of nonempty subsets which are mutually disjoint and whose union is $\{1, 2, \ldots, n\}$.

For example, $145/23/6$ and $1/245/36$ are both set partitions of $\{1, 2, \ldots, 6\}$.

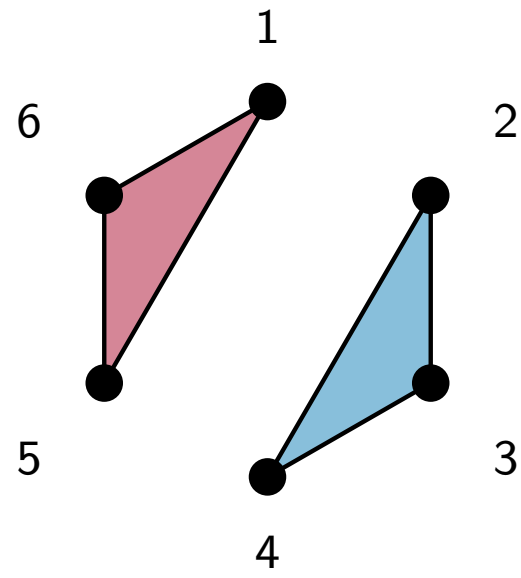
We say partition $B_1/B_2/\cdots/B_k$ is **crossing** if there exists $i \neq j$ and $\mathbf{a} < \mathbf{b} < \mathbf{c} < \mathbf{d}$ with $a, c \in B_i$ and $b, d \in B_j$. We say a partition is **noncrossing** if it is not crossing.

$1/\mathbf{24}5/\mathbf{36}$ is crossing whereas $156/234$ is noncrossing.

Noncrossing partitions



1/245/36



156/234

Noncrossing partitions

Noncrossing partitions play a role in many areas of mathematics including

- Combinatorics
- Group Theory
- Noncommutative Probability

See the article *Noncrossing Partitions in Surprising Locations* by Jon McCammond for more details and examples.

Noncrossing partitions

Noncrossing partitions play a role in many areas of mathematics including

- Combinatorics
- Group Theory
- Noncommutative Probability

See the article *Noncrossing Partitions in Surprising Locations* by Jon McCammond for more details and examples.

Our main interest is in their connection with parking functions. To understand this connection, we need to put an order on noncrossing partitions.

An Ordering on Noncrossing Partitions

Let $B_1/B_2/\cdots/B_k$ and $C_1/C_2/\cdots/C_j$ be noncrossing partitions of $\{1, 2, \ldots, n\}$. We define an order so that

$$B_1/B_2/\cdots/B_k \leq C_1/C_2/\cdots/C_j$$

if and only if each C_i is the union of some blocks of $B_1/B_2/\cdots/B_k$.

An Ordering on Noncrossing Partitions

Let $B_1/B_2/\cdots/B_k$ and $C_1/C_2/\cdots/C_j$ be noncrossing partitions of $\{1, 2, \ldots, n\}$. We define an order so that

$$B_1/B_2/\cdots/B_k \leq C_1/C_2/\cdots/C_j$$

if and only if each C_i is the union of some blocks of $B_1/B_2/\cdots/B_k$.

For example,

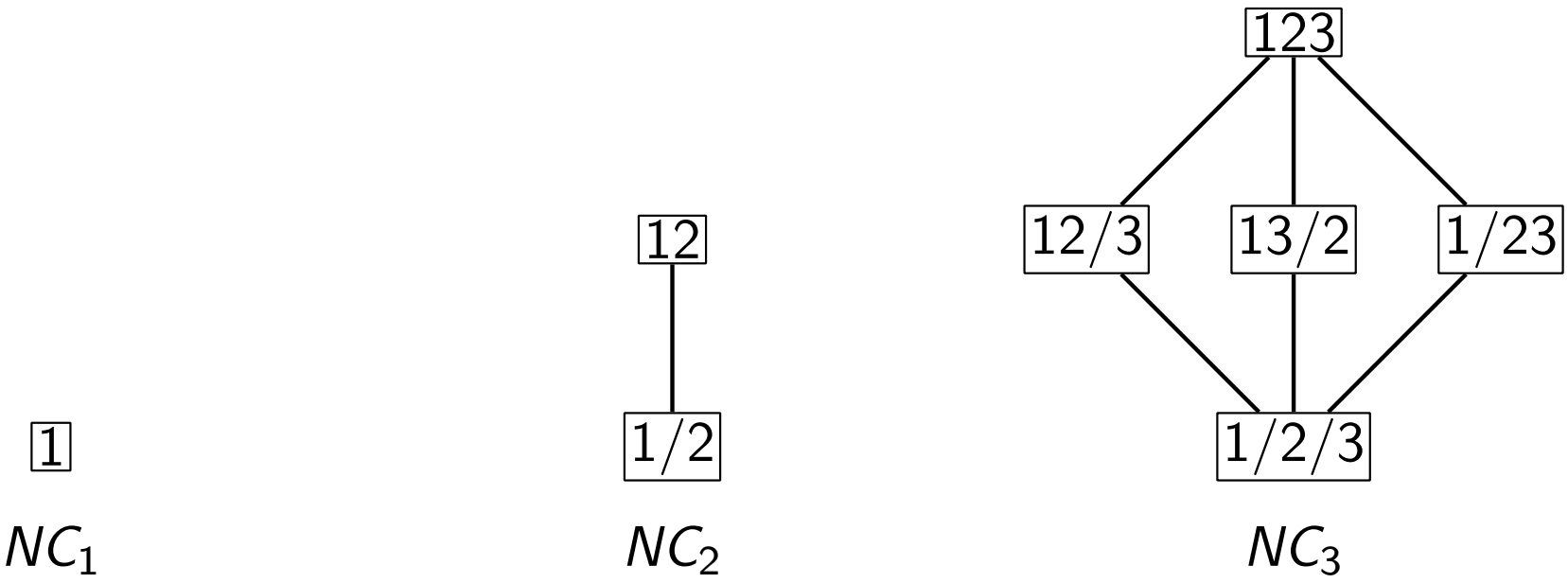
$$\mathbf{134}/\mathbf{2}/\mathbf{56}/\mathbf{78} \leq \mathbf{1234}/\mathbf{5678}$$

whereas

$$\mathbf{134}/\mathbf{2}/\mathbf{56}/\mathbf{78} \not\leq \mathbf{12345}/\mathbf{6}/\mathbf{78}$$

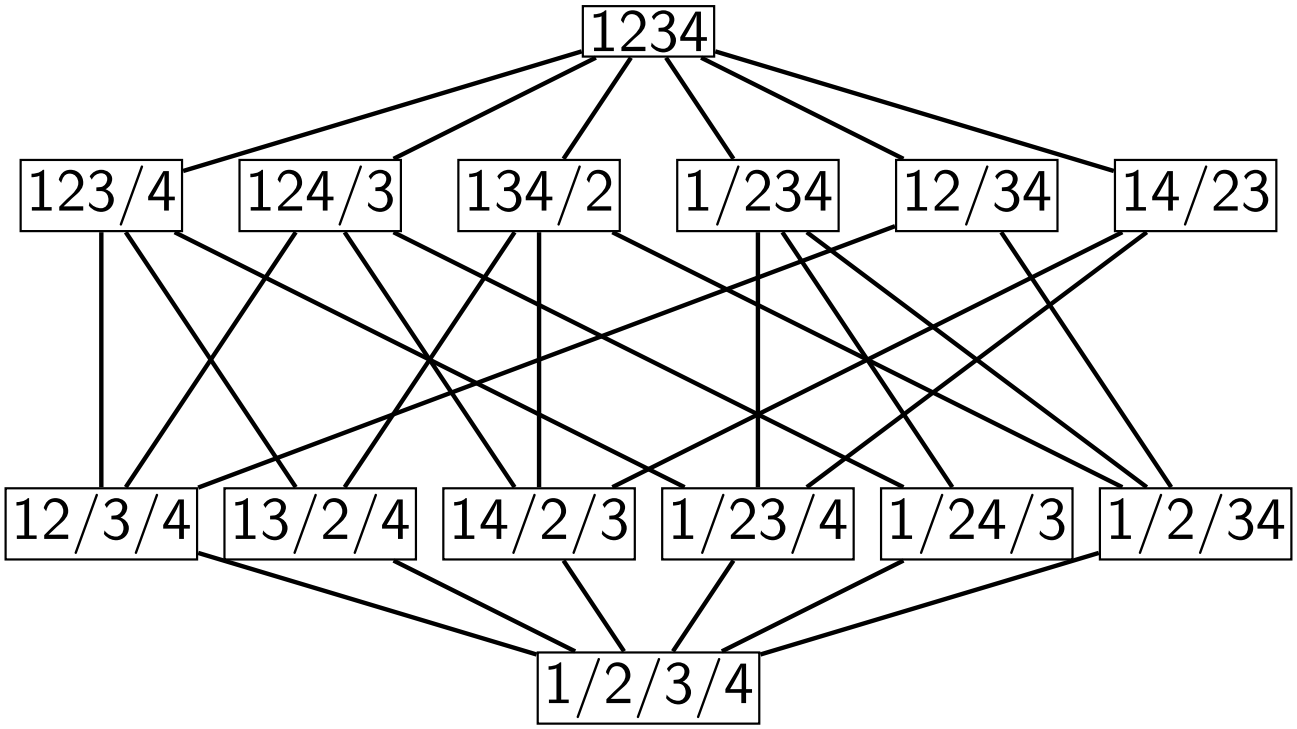
The Noncrossing Partition Lattice

Ordering these the noncrossing partitions of $\{1, 2, \ldots, n\}$ gives us the **noncrossing partition lattice**. It is denoted by NC_n.



NC_1 NC_2 NC_3

Noncrossing Partition Lattice



NC_4

Noncrossing Partitions and Parking Functions

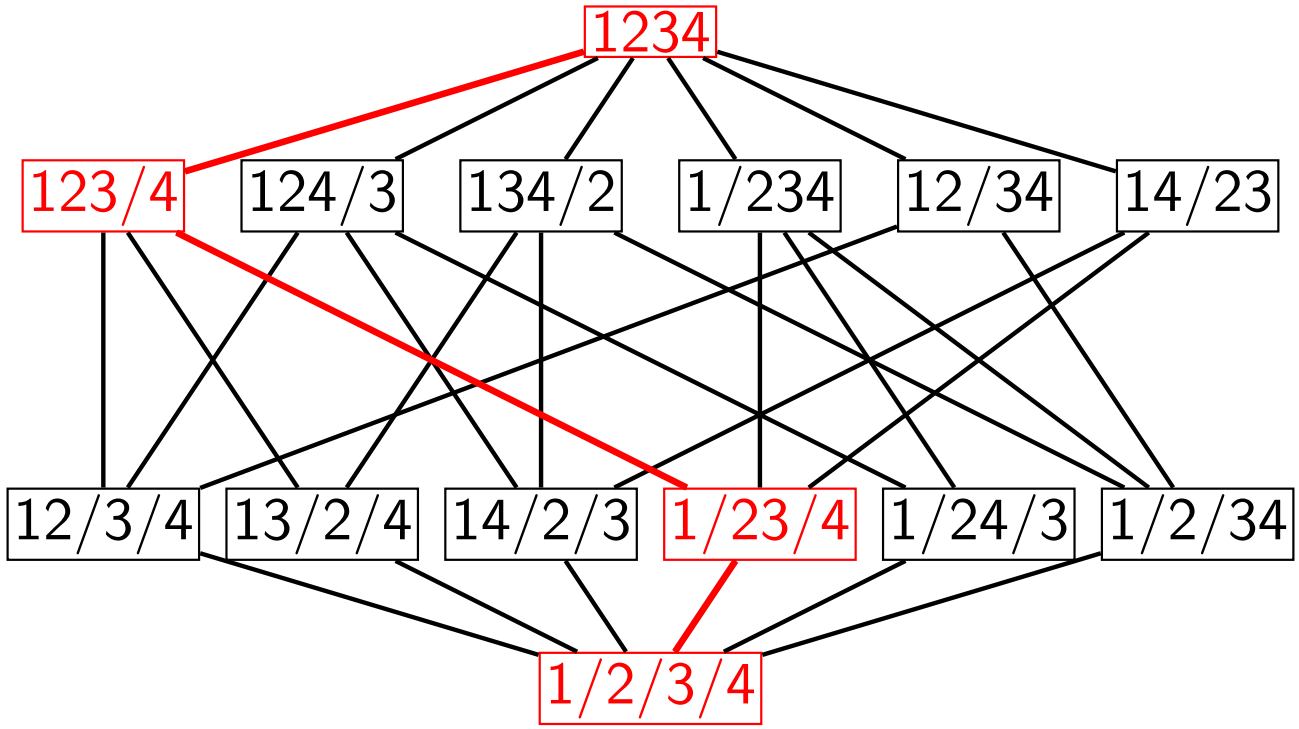
So what do parking functions have to do with noncrossing partitions? The answer has to do with chains.

A **chain** in a poset is a totally ordered subset in the poset. A chain is **maximal** if it is not contained in any other chain.

Noncrossing Partitions and Parking Functions

So what do parking functions have to do with noncrossing partitions? The answer has to do with chains.

A **chain** in a poset is a totally ordered subset in the poset. A chain is **maximal** if it is not contained in any other chain.



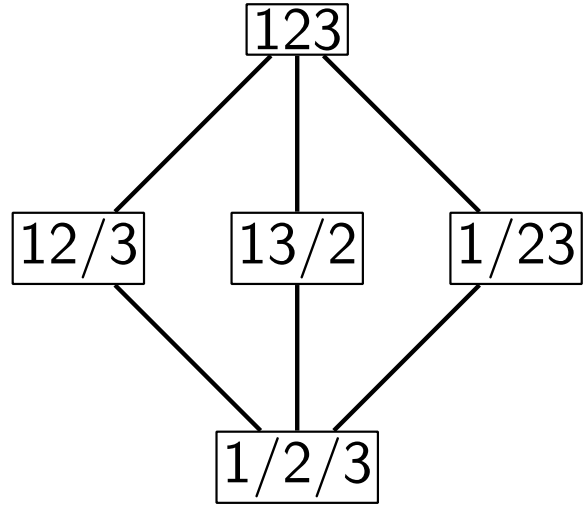
NC_4

Noncrossing Partitions and Parking Functions

Let's count the number of maximal chains in each noncrossing partition lattice.

NC_1

NC_2



NC_3

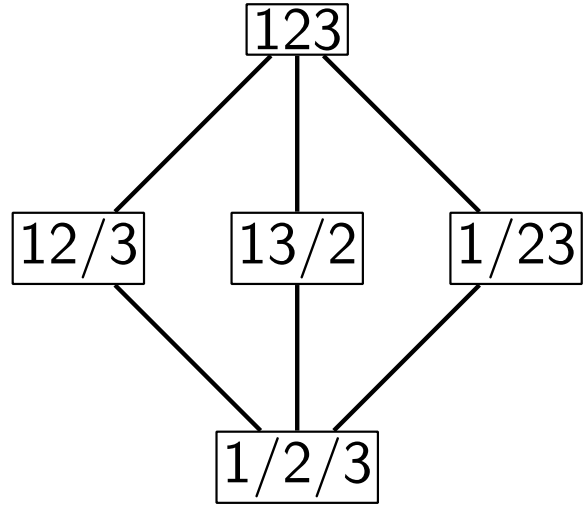
Noncrossing Partitions and Parking Functions

Let's count the number of maximal chains in each noncrossing partition lattice.

NC_1

1 Max Chain

NC_2



NC_3

Noncrossing Partitions and Parking Functions

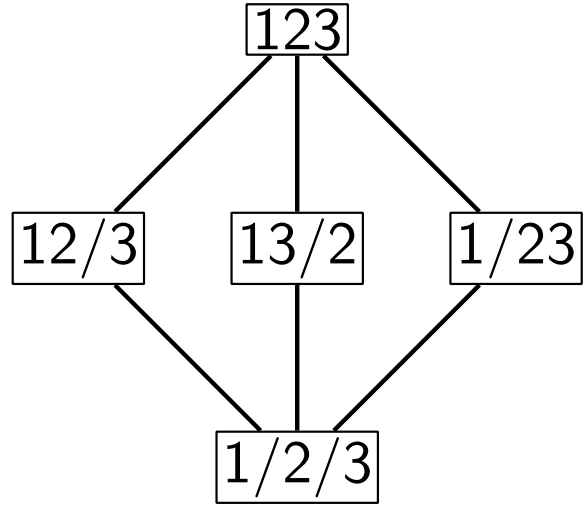
Let's count the number of maximal chains in each noncrossing partition lattice.

NC_1

1 Max Chain

NC_2

1 Max Chain



NC_3

Noncrossing Partitions and Parking Functions

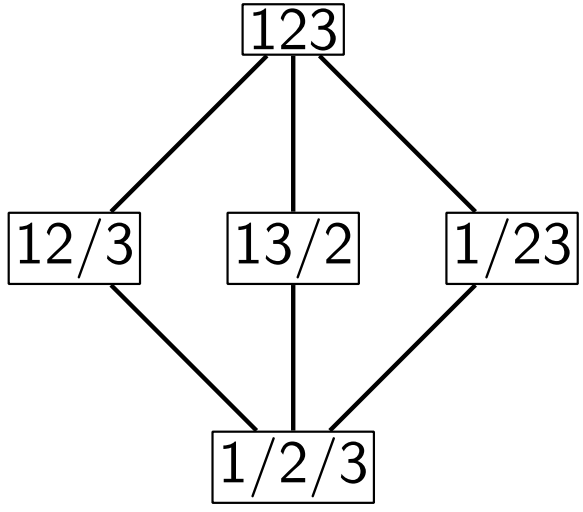
Let's count the number of maximal chains in each noncrossing partition lattice.

NC_1

1 Max Chain

NC_2

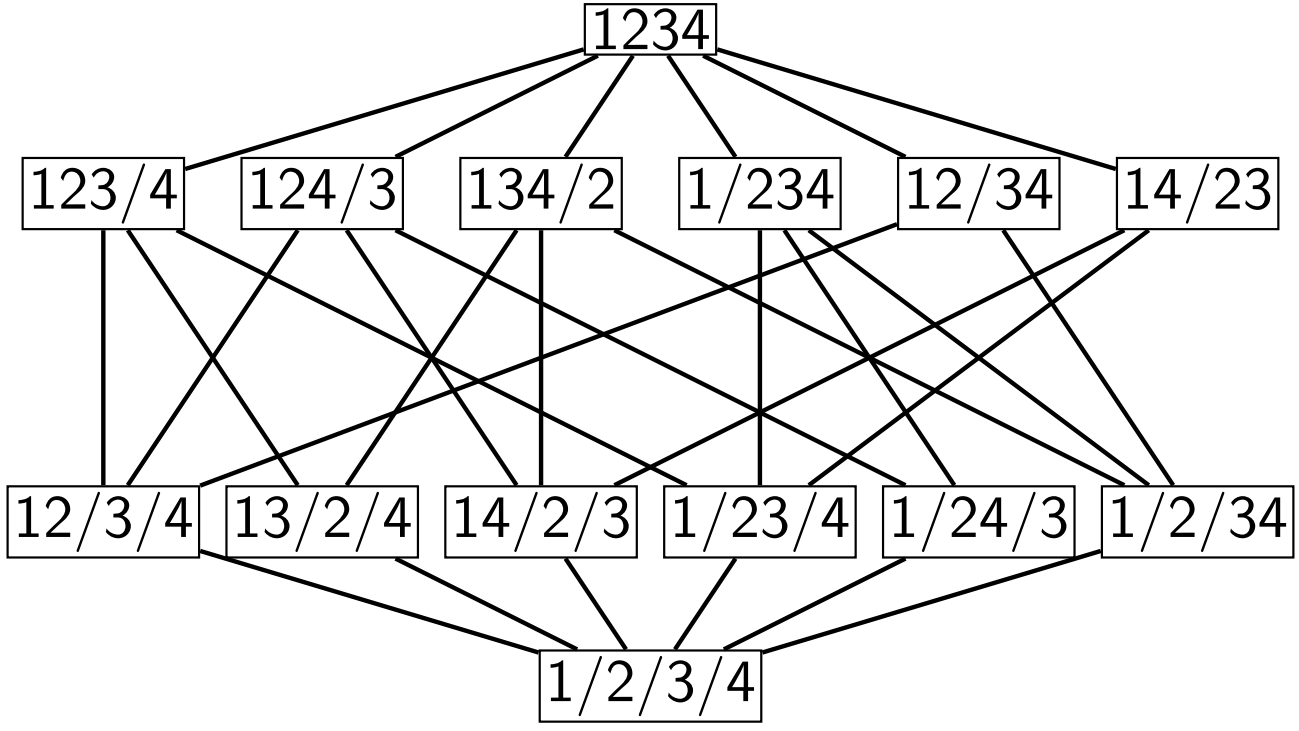
1 Max Chain



NC_3

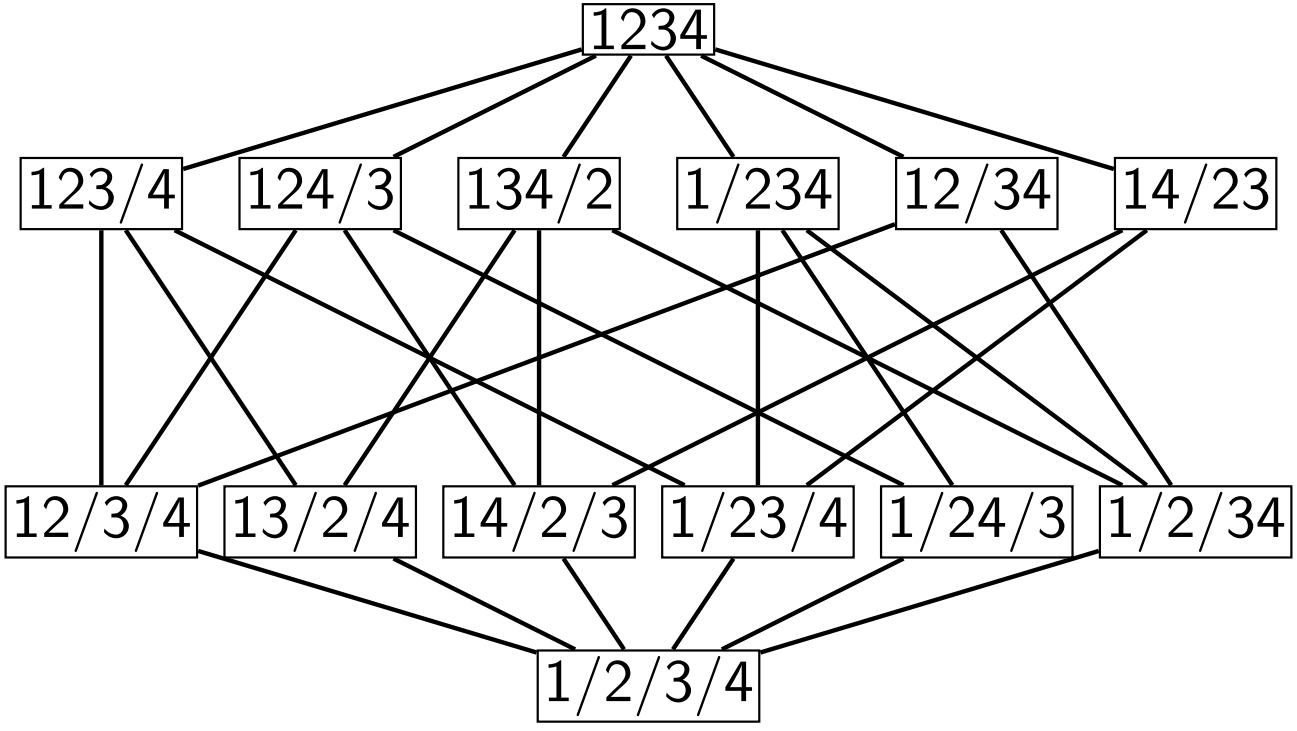
3 Max Chains

Noncrossing Partitions and Parking Functions



NC_4

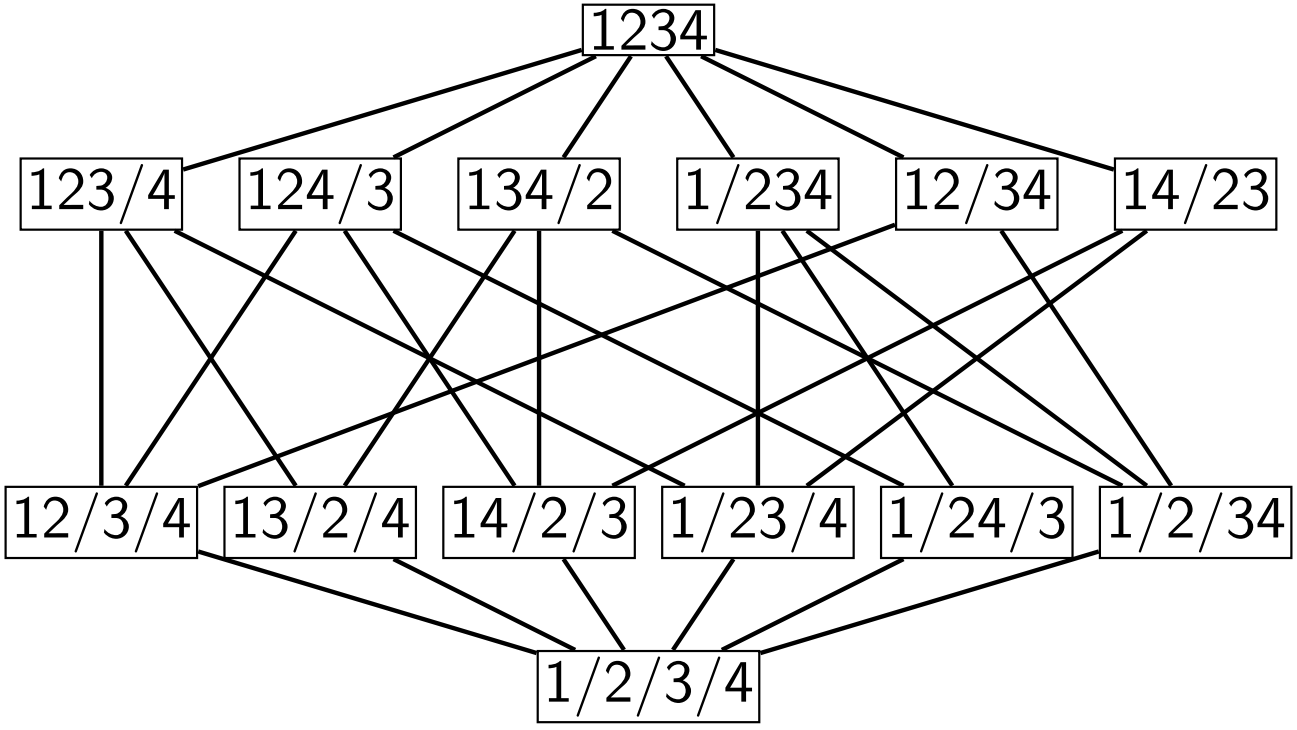
Noncrossing Partitions and Parking Functions



NC_4

16 Max Chains

Noncrossing Partitions and Parking Functions



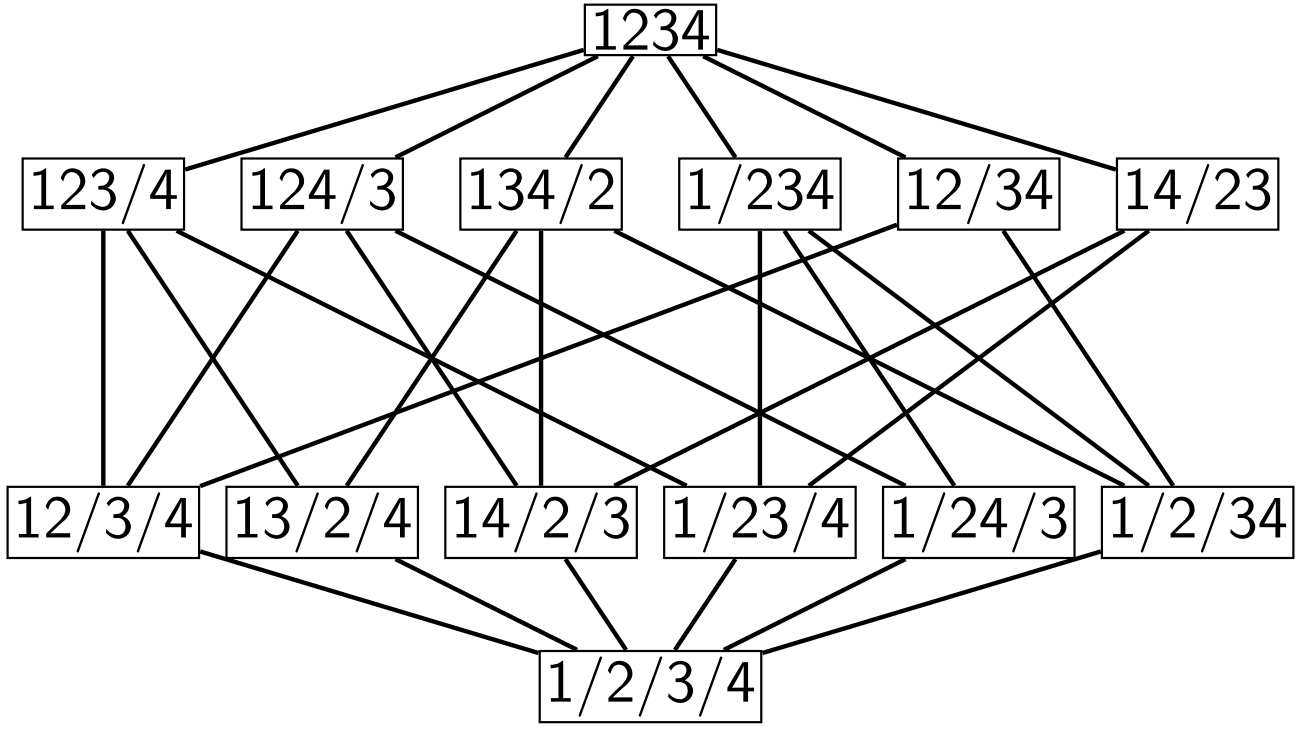
NC_4

16 Max Chains

So the number of maximal chains in NC_n is $1, 1, 3, 16, \ldots$. This is exactly the same count for parking functions.

Noncrossing Partitions and Parking Functions

To prove this (and more), Stanley (1997) introduced a labeling of the edges of the Hasse diagram of NC_n. To explain the labeling, note each edge represents the merging of exactly two blocks.



NC_4

Noncrossing Partitions and Parking Functions

To prove this (and more), Stanley (1997) introduced a labeling of the edges of the Hasse diagram of NC_n. To explain the labeling, note each edge represents the merging of exactly two blocks.

Say we merge together 2 blocks B and B' with $\min B < \min B'$. We label the edge corresponding to this merge as the largest element of B smaller than $\min B'$.

Noncrossing Partitions and Parking Functions

To prove this (and more), Stanley (1997) introduced a labeling of the edges of the Hasse diagram of NC_n. To explain the labeling, note each edge represents the merging of exactly two blocks.

Say we merge together 2 blocks B and B' with $\min B < \min B'$. We label the edge corresponding to this merge as the largest element of B smaller than $\min B'$.

For example the edge in the Hasse diagram of NC_7

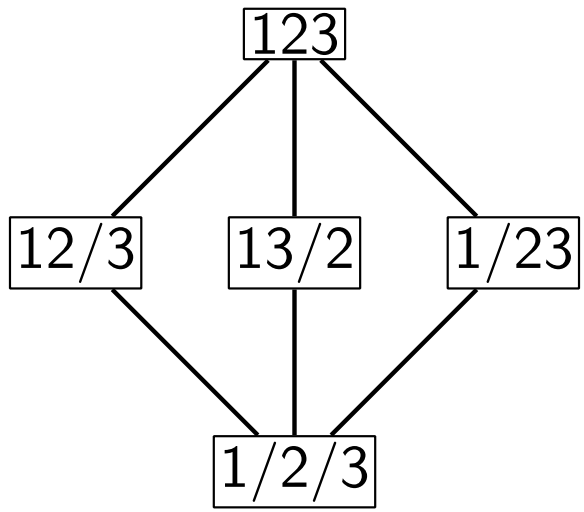
$$1367/45/2 < 134567/2$$

would be labeled by 3 since 3 is the largest element of 1367 smaller than 4.

Noncrossing Partitions and Parking Functions

To prove this (and more), Stanley (1997) introduced a labeling of the edges of the Hasse diagram of NC_n. To explain the labeling, note each edge represents the merging of exactly two blocks.

Say we merge together 2 blocks B and B' with $\min B < \min B'$. We label the edge corresponding to this merge as the largest element of B smaller than $\min B'$.

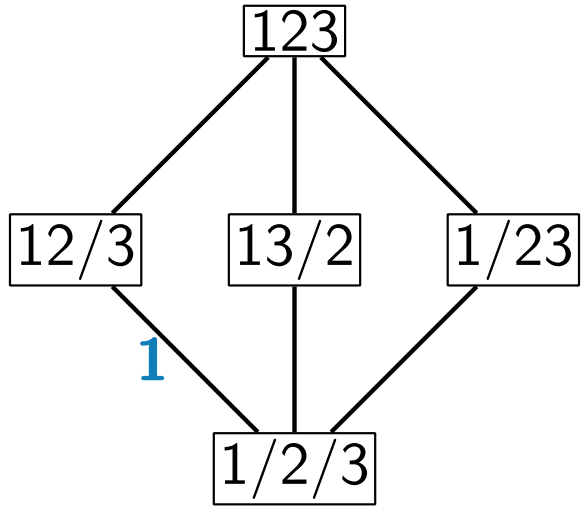


NC_3

Noncrossing Partitions and Parking Functions

To prove this (and more), Stanley (1997) introduced a labeling of the edges of the Hasse diagram of NC_n. To explain the labeling, note each edge represents the merging of exactly two blocks.

Say we merge together 2 blocks B and B' with $\min B < \min B'$. We label the edge corresponding to this merge as the largest element of B smaller than $\min B'$.

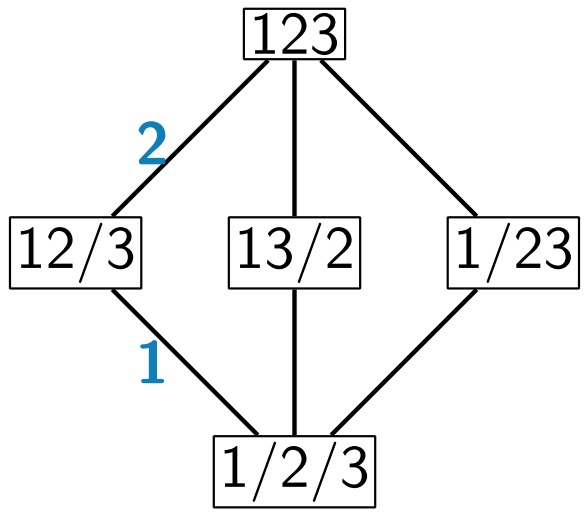


NC_3

Noncrossing Partitions and Parking Functions

To prove this (and more), Stanley (1997) introduced a labeling of the edges of the Hasse diagram of NC_n. To explain the labeling, note each edge represents the merging of exactly two blocks.

Say we merge together 2 blocks B and B' with $\min B < \min B'$. We label the edge corresponding to this merge as the largest element of B smaller than $\min B'$.

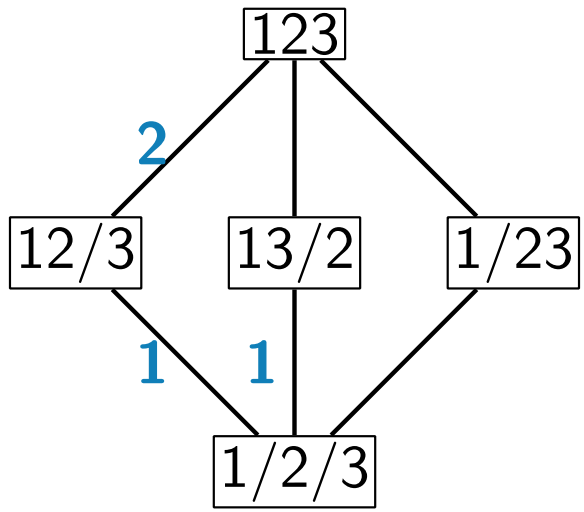


NC_3

Noncrossing Partitions and Parking Functions

To prove this (and more), Stanley (1997) introduced a labeling of the edges of the Hasse diagram of NC_n. To explain the labeling, note each edge represents the merging of exactly two blocks.

Say we merge together 2 blocks B and B' with $\min B < \min B'$. We label the edge corresponding to this merge as the largest element of B smaller than $\min B'$.

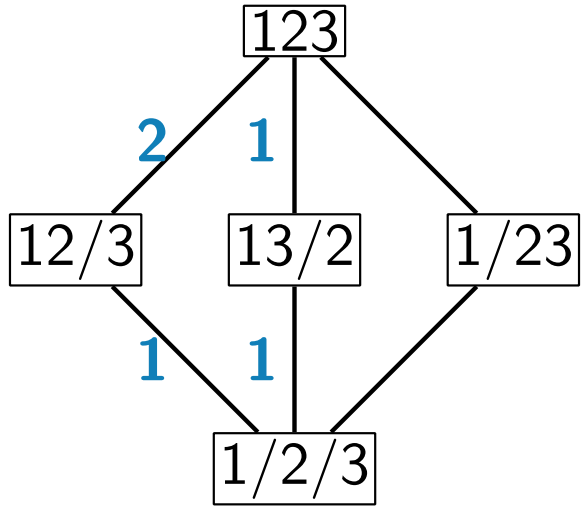


NC_3

Noncrossing Partitions and Parking Functions

To prove this (and more), Stanley (1997) introduced a labeling of the edges of the Hasse diagram of NC_n. To explain the labeling, note each edge represents the merging of exactly two blocks.

Say we merge together 2 blocks B and B' with $\min B < \min B'$. We label the edge corresponding to this merge as the largest element of B smaller than $\min B'$.

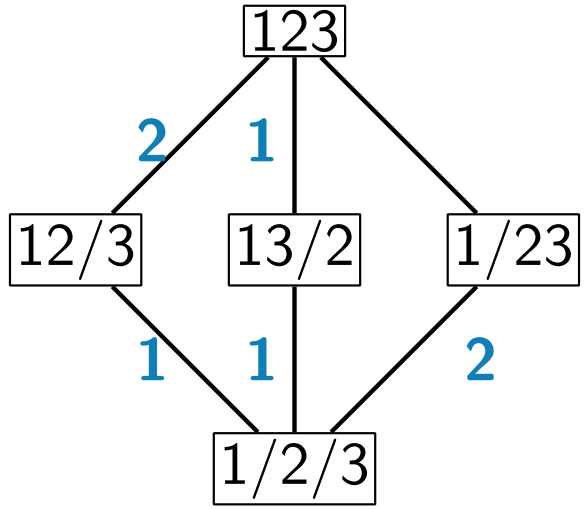


NC_3

Noncrossing Partitions and Parking Functions

To prove this (and more), Stanley (1997) introduced a labeling of the edges of the Hasse diagram of NC_n. To explain the labeling, note each edge represents the merging of exactly two blocks.

Say we merge together 2 blocks B and B' with $\min B < \min B'$. We label the edge corresponding to this merge as the largest element of B smaller than $\min B'$.

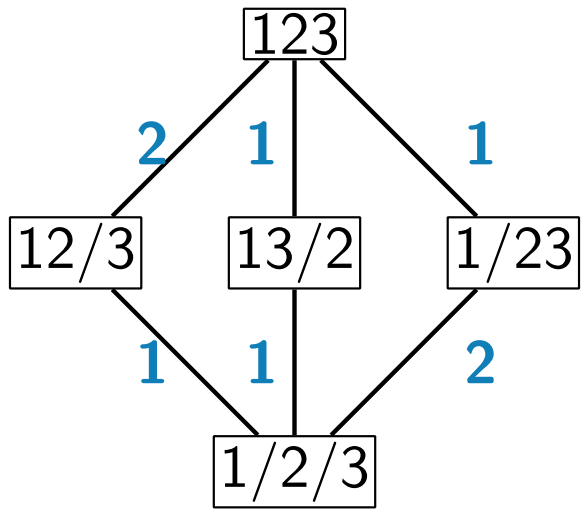


NC_3

Noncrossing Partitions and Parking Functions

To prove this (and more), Stanley (1997) introduced a labeling of the edges of the Hasse diagram of NC_n. To explain the labeling, note each edge represents the merging of exactly two blocks.

Say we merge together 2 blocks B and B' with $\min B < \min B'$. We label the edge corresponding to this merge as the largest element of B smaller than $\min B'$.

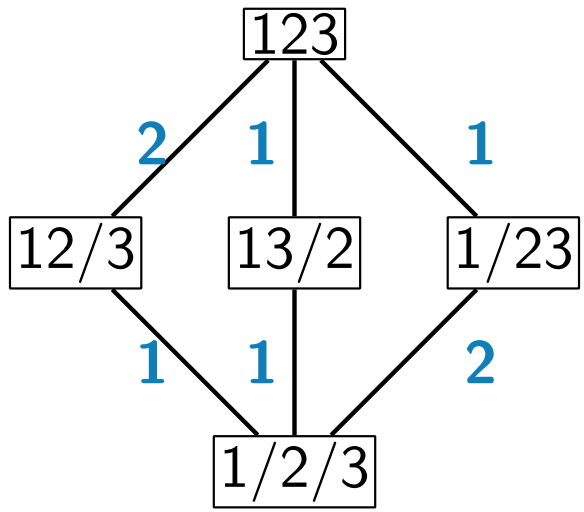


NC_3

Noncrossing Partitions and Parking Functions

To prove this (and more), Stanley (1997) introduced a labeling of the edges of the Hasse diagram of NC_n. To explain the labeling, note each edge represents the merging of exactly two blocks.

Say we merge together 2 blocks B and B' with $\min B < \min B'$. We label the edge corresponding to this merge as the largest element of B smaller than $\min B'$.



NC_3

Note the labels along the maximal chains are $12, 11, 21$ which are exactly the parking functions of length 2.

Noncrossing Partitions and Parking Functions

Theorem (Stanley 1997)

The parking function labeling of NC_n gives a bijection between maximal chains of NC_n and parking functions of length $n-1$.

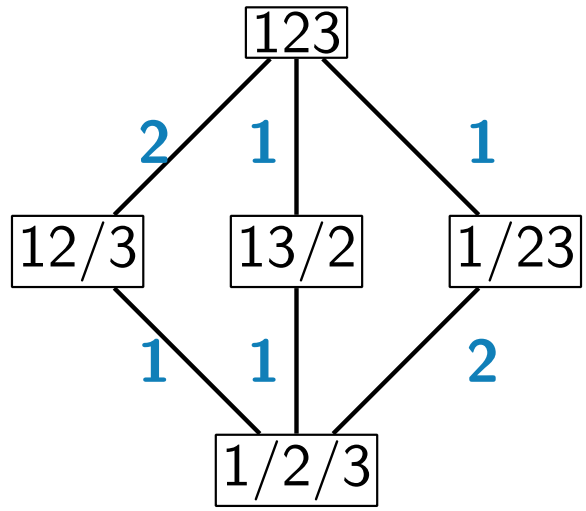
One can interpret this result as saying the parking functions give a "recipe" to build the set $\{1, 2, \ldots, n\}$ from the singletons in a way that never forms crossings.

The Parking Function Labeling

An edge labeling of a poset is a **(strict) ER-labeling** if for every pair x, y with $x < y$, there is a unique maximal sequence of labels from x to y that is strictly increasing.

The Parking Function Labeling

An edge labeling of a poset is a **(strict) ER-labeling** if for every pair x, y with $x < y$, there is a unique maximal sequence of labels from x to y that is strictly increasing.



NC_3

The parking function labeling of NC_3 is a (strict) ER-labeling.

The Parking Function Labeling

ER-labelings tells you some nice information about the poset. For example, you can determine the following:

- The Möbius function. (Using this, one can show the number of parking functions with no strict ascents is a Catalan number)
- The characteristic polynomial.
- The rank generating function.
- The Zeta polynomial.

The Parking Function Labeling

ER-labelings tells you some nice information about the poset. For example, you can determine the following:

- The Möbius function. (Using this, one can show the number of parking functions with no strict ascents is a Catalan number)
- The characteristic polynomial.
- The rank generating function.
- The Zeta polynomial.

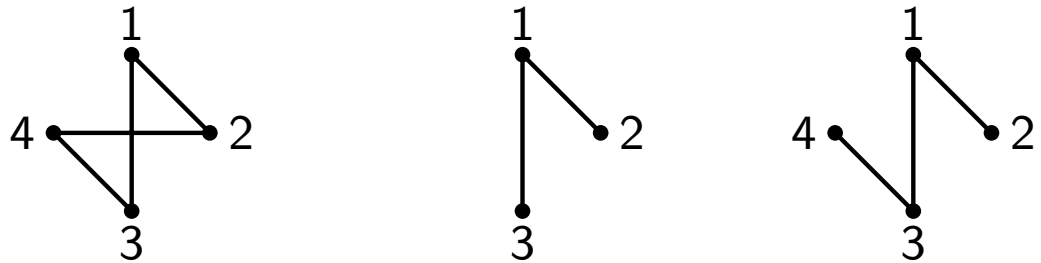
All of these actually come from the flag quasisymmetric function of the poset. The ER-labeling gives combinatorial interpretations for it when expanding in the monomial and fundamental basis.

Stanley (1997) showed that there is an action of the symmetric group on the maximal chains of NC_n whose Frobenius characteristic is the flag quasisymmetric function of NC_n.

Noncrossing Partitions of Graphs

In 2020, Farmer, Hallam, and Smyth introduced a generalization of the noncrossing partition lattice using graphs. First, we need a few definitions.

Let G be a graph and let H be a subgraph. We say H is **induced** if u, v are vertices of H and u and v are adjacent in G, then u and v are adjacent in H.

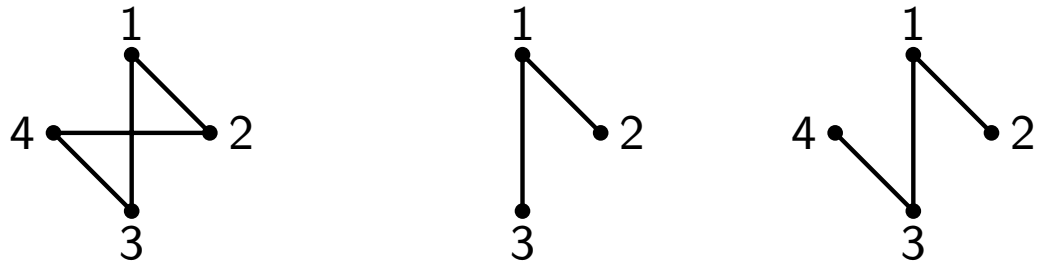


G H H'

Noncrossing Partitions of Graphs

In 2020, Farmer, Hallam, and Smyth introduced a generalization of the noncrossing partition lattice using graphs. First, we need a few definitions.

Let G be a graph and let H be a subgraph. We say H is **induced** if u, v are vertices of H and u and v are adjacent in G, then u and v are adjacent in H.

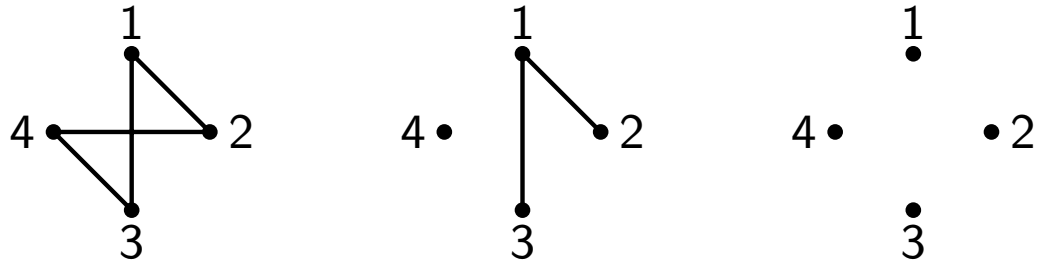


G H H'

Here H is induced, but H' is not induced.

Noncrossing Partitions of Graphs

Let G be a graph with vertex set $\{1, 2, \ldots, n\}$. We say a noncrossing partition $B_1/B_2/\cdots/B_k$ is a **noncrossing bond** of G if for each B_i, the induced subgraph of G on the set B_i is connected.

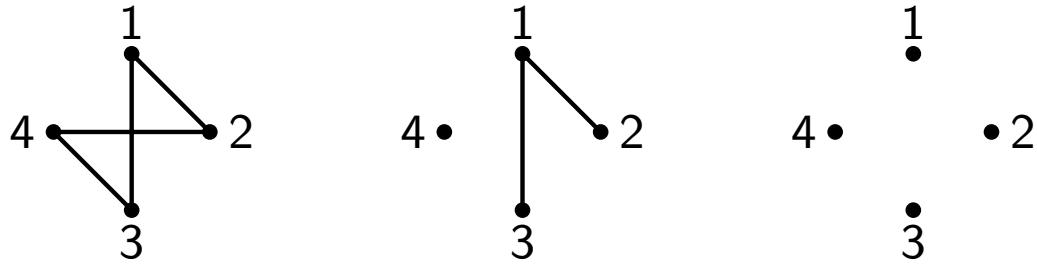


G $\quad$ $123/4$ $\quad$ $14/23$

The noncrossing partition $123/4$ is a noncrossing bond of G since the induced subgraph with 123 is connected and the induced subgraph with 4 is (trivially) connected.

Noncrossing Partitions of Graphs

Let G be a graph with vertex set $\{1, 2, \ldots, n\}$. We say a noncrossing partition $B_1/B_2/\cdots/B_k$ is a **noncrossing bond** of G if for each B_i, the induced subgraph of G on the set B_i is connected.

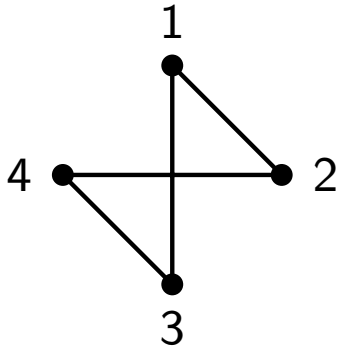


G $\quad$ $123/4$ $\quad$ $14/23$

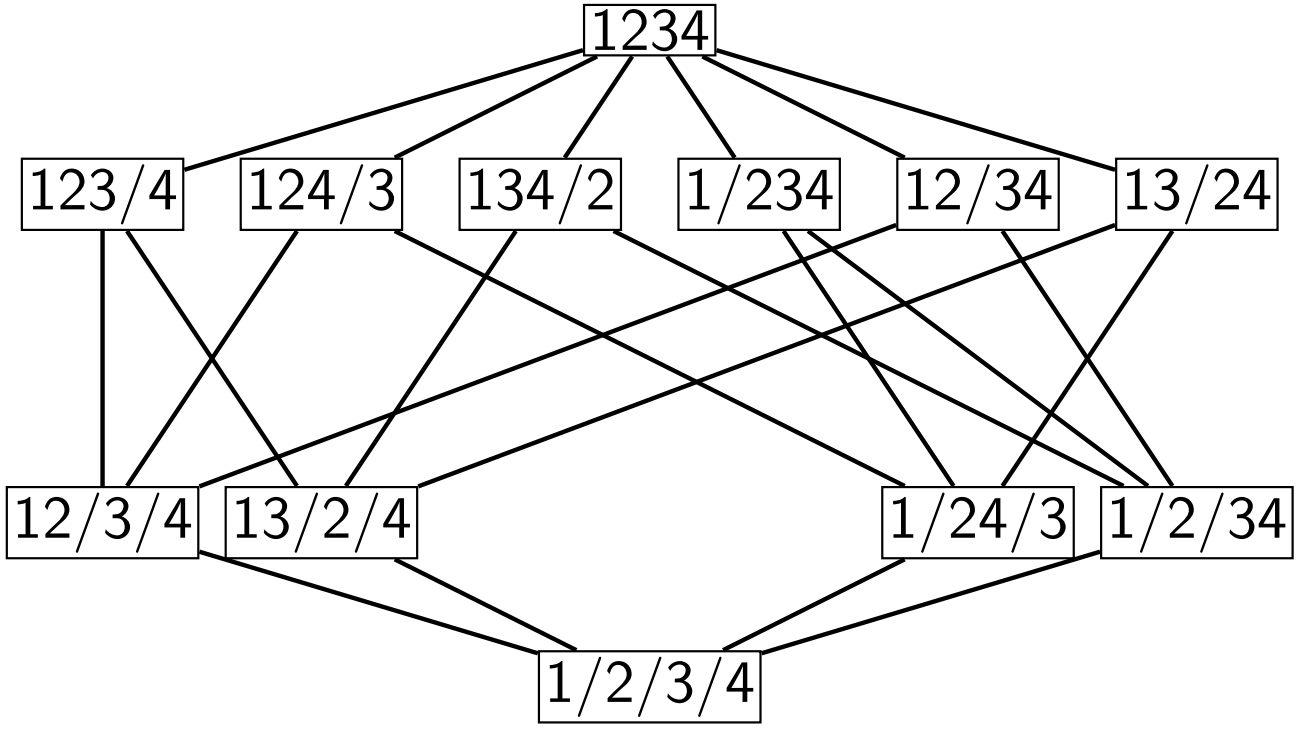
The noncrossing partition $123/4$ is a noncrossing bond of G since the induced subgraph with 123 is connected and the induced subgraph with 4 is (trivially) connected. However, the noncrossing partition $14/23$ is not a noncrossing bond of G since the induced subgraph with 14 is not connected.

Noncrossing Partitions of Graphs

Let G be a graph with vertex set $\{1, 2, \ldots, n\}$. The **noncrossing bond poset** of G, denoted NC_G is the set of noncrossing bonds of G ordered as in NC_n.



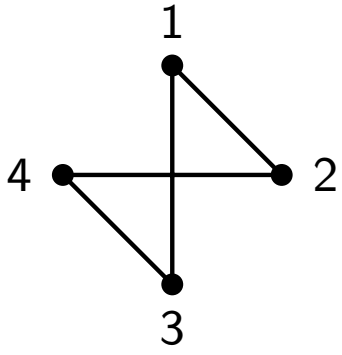
G



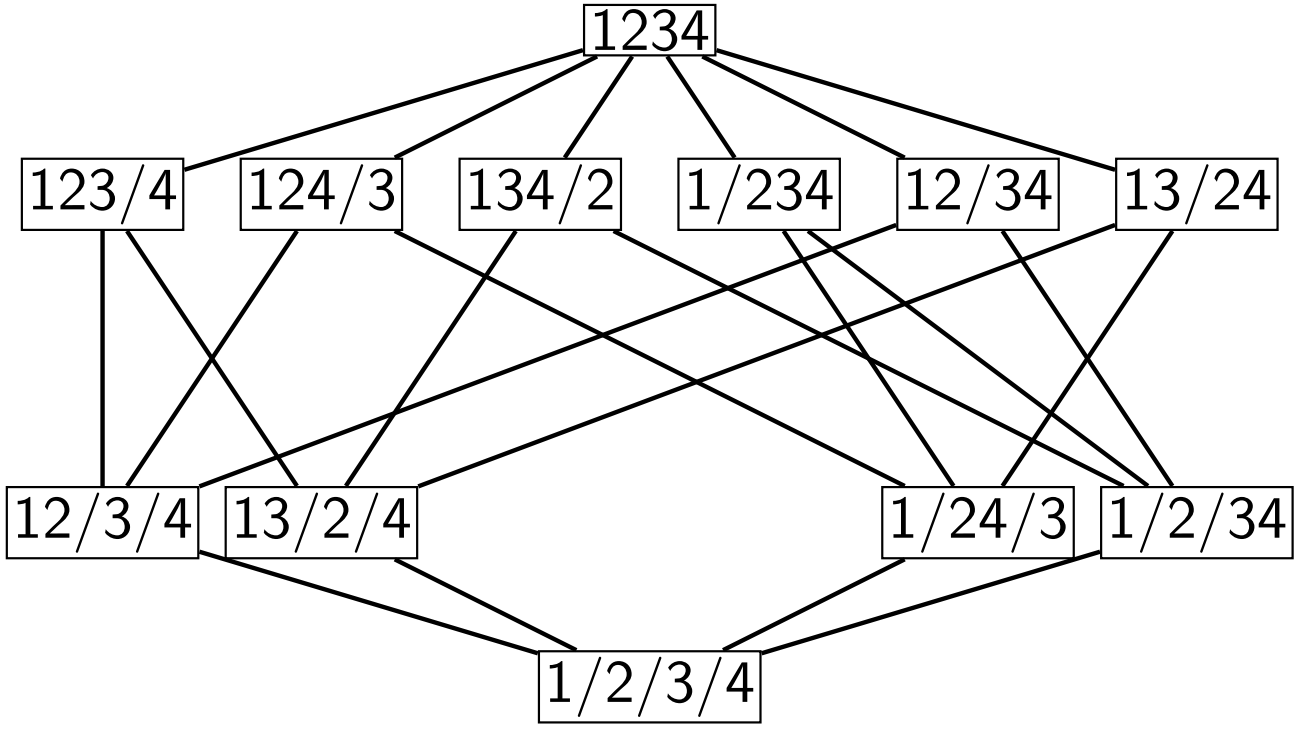
NC_G

Noncrossing Partitions of Graphs

Let G be a graph with vertex set $\{1, 2, \ldots, n\}$. The **noncrossing bond poset** of G, denoted NC_G is the set of noncrossing bonds of G ordered as in NC_n.



G



NC_G

The noncrossing partition lattice is the noncrossing bond poset of the complete graph.

Noncrossing Partitions of Graphs

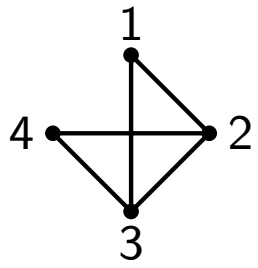
Since the noncrossing partition bond poset is a subposet of the noncrossing partition lattice, it inherits the parking function labeling. So the question arises,

When is the parking function labeling a (strict) ER-labeling of NC_G?

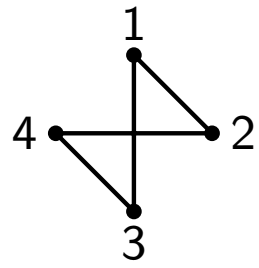
For her senior thesis, Anna Salam answered this question. To explain her result, we need a definition.

Perfect Elimination Ordering

Let G be a graph with vertex set $\{1, 2, \ldots, n\}$. We say G is labeled with **perfect elimination ordering** if whenever $i < j < k$ and ik and jk are edges, ij is an edge.



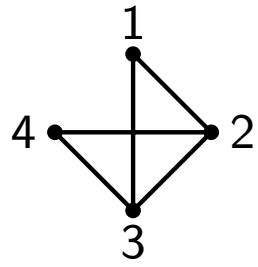
G



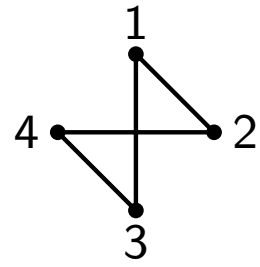
H

Perfect Elimination Ordering

Let G be a graph with vertex set $\{1, 2, \ldots, n\}$. We say G is labeled with **perfect elimination ordering** if whenever $i < j < k$ and ik and jk are edges, ij is an edge.



G



H

Here G is labeled with a perfect elimination ordering, but H is not. The issue is that 24 and 34 are edges of H, but 23 is not an edge of H.

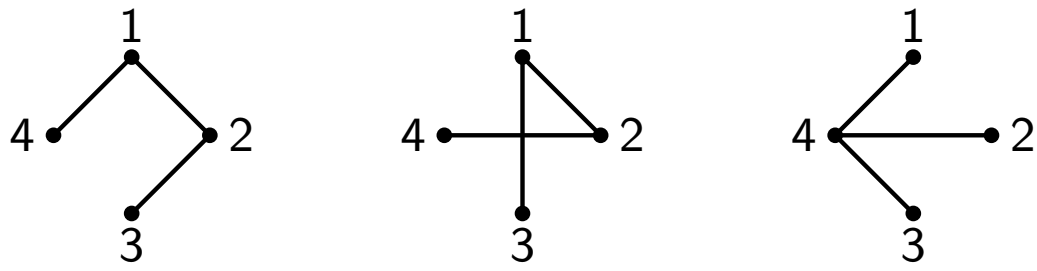
Parking Function Labeling of the Noncrossing Bond Poset

Theorem (H-Salam)

The parking function labeling of NC_G is a strict ER-labeling if and only if G is labeled with a perfect elimination ordering.

Consequences

Let T be a tree on vertex set $\{1, 2, \ldots, n\}$. We say T is **increasing** if the sequence of vertices along any path starting at 1 is increasing. Moreover, we say T is **noncrossing** if, when drawn so that the vertices are placed around a circle in increasing order, none of the edges cross.



T T' T''

T is increasing and noncrossing, T' is increasing, but not noncrossing, and T'' is noncrossing, but not increasing.

Consequences

Let $PF(G)$ denote the set of parking functions along the maximal chains of NC_G.

Corollary

Let T be an increasing noncrossing tree on vertex set $[n+1]$. Then

$$\sum_{\pi \in PF(T)} q^{des(\pi)} = \sum_{\pi \in S_n} q^{des(\pi)}$$

where $des(\pi) = \{i \mid \pi \geq \pi_{i+1}\}$ is the number of weak descents in π. Equivalently, $\sum_{\pi \in PF(T)} q^{des(\pi)}$ is the n^{th} Eulerian polynomial.

Consequences

Let $PF(G)$ denote the set of parking functions along the maximal chains of NC_G.

Corollary

Let T be an increasing noncrossing tree on vertex set $[n+1]$. Then

$$\sum_{\pi \in PF(T)} q^{des(\pi)} = \sum_{\pi \in S_n} q^{des(\pi)}$$

where $des(\pi) = \{i \mid \pi \geq \pi_{i+1}\}$ is the number of weak descents in π. Equivalently, $\sum_{\pi \in PF(T)} q^{des(\pi)}$ is the n^{th} Eulerian polynomial.

In fact, you can replace "descent" with "inversion" or "major index" and still get the result in the displayed equation.

Consequences

Let $PF(G)$ denote the set of parking functions along the maximal chains of NC_G.

Corollary

If G is labeled with a perfect elimination ordering, then the number of ascent-free parking functions in $PF(G)$ is the number of spanning trees of G that are increasing and noncrossing.

Thank You!