What Else Can You Count If You Can Count Trees?

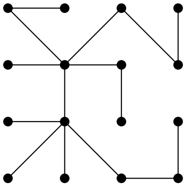
Jeremy L. Martin
Department of Mathematics
University of Kansas

Benedictine College
April 4, 2019

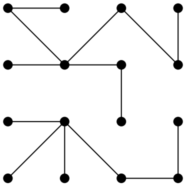
Trees

Tree: a nonempty set of vertices connected by edges, so that]

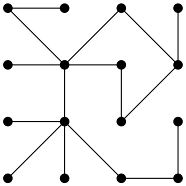
- there is a path between any two vertices (*connectedness*);
- there are no closed loops (*acyclicity*).



Tree



Not connected

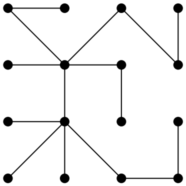


Not acyclic

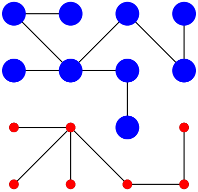
Trees

Tree: a nonempty set of vertices connected by edges, so that]

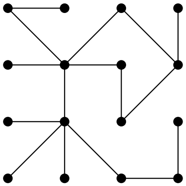
- there is a path between any two vertices (*connectedness*);
- there are no closed loops (*acyclicity*).



Tree



Not connected

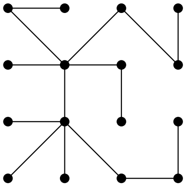


Not acyclic

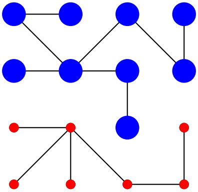
Trees

Tree: a nonempty set of vertices connected by edges, so that]

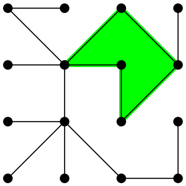
- there is a path between any two vertices (*connectedness*);
- there are no closed loops (*acyclicity*).



Tree



Not connected



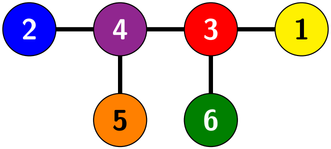
Not acyclic

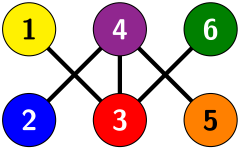
Properties of Trees

1. Every tree with n vertices has **exactly $n - 1$ edges**. (Any fewer and it cannot be connected; any more and it must contain a cycle.)

2. Every tree with at least two vertices has **at least two leaves** (vertices with only one neighbor).

3. We only care about **which vertices are connected**, not how the tree is depicted on the page. These trees are the same:





How Many Trees Are There On *n* Labeled Vertices?

How Many Trees Are There On n Labeled Vertices?

$n = 1$

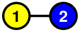
1 tree

How Many Trees Are There On n Labeled Vertices?

$n = 1$

1 tree

$n = 2$



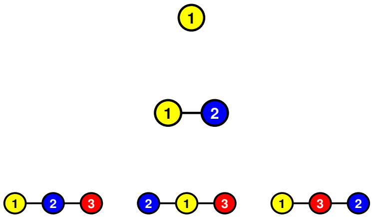
1 tree

How Many Trees Are There On n Labeled Vertices?

$n = 1$ — 1 tree

$n = 2$ — 1 tree

$n = 3$ — 3 trees

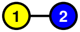


How Many Trees Are There On n Labeled Vertices?

$n = 1$

1 tree

$n = 2$

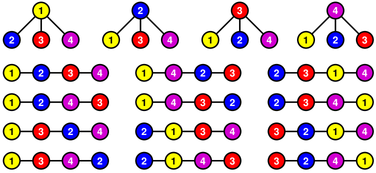


1 tree

$n = 3$

3 trees

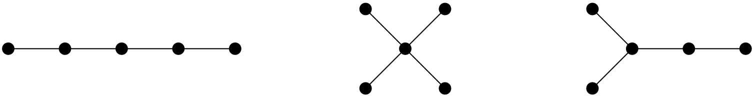
$n = 4$



16 trees

How Many Trees Are There On n Vertices?

For $n = 5$, there are three tree shapes:



$5!/2 = 60$ trees

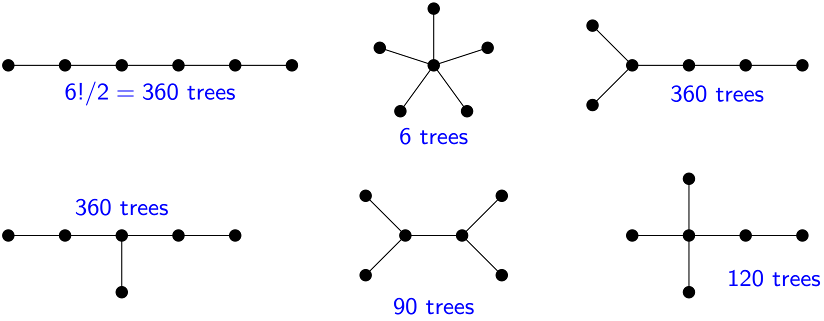
5 trees

$5!/2 = 60$ trees

Total: **125** trees on 5 labeled vertices.

How Many Trees Are There On n Vertices?

For $n = 6$, there are six tree shapes:



Total: **1296** trees on 6 labeled vertices.

How Many Trees Are There On n Vertices?

Let $T(n)$ = number of labeled trees on n vertices.

n	$T(n)$
1	1
2	1
3	3
4	16
5	125
6	1296
7	16807
8	262144

How Many Trees Are There On n Vertices?

Let $T(n)$ = number of labeled trees on n vertices.

n	$T(n)$	
1	1	$= \mathbf{1^{-1}}$
2	1	$= \mathbf{2^{0}}$
3	3	$= \mathbf{3^{1}}$
4	16	$= \mathbf{4^{2}}$
5	125	$= \mathbf{5^{3}}$
6	1296	$= \mathbf{6^{4}}$
7	16807	$= \mathbf{7^{5}}$
8	262144	$= \mathbf{8^{6}}$

How Many Trees Are There On n Vertices?

Let $T(n)$ = number of labeled trees on n vertices.

n	$T(n)$	
1	1	$= 1^{-1}$
2	1	$= 2^{0}$
3	3	$= 3^{1}$
4	16	$= 4^{2}$
5	125	$= 5^{3}$
6	1296	$= 6^{4}$
7	16807	$= 7^{5}$
8	262144	$= 8^{6}$

Theorem 1 $\boldsymbol{T(n) = n^{n-2}}$ for all $\boldsymbol{n}$.

Yeah, But How Do You Prove That? (#1)

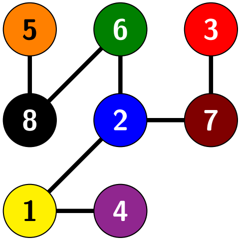
Prüfer code of a tree: the sequence $P(T)$ constructed as follows.

- Find the leaf with the smallest label.
- Write down its neighbor (not the leaf itself!)
- Delete it.
- Repeat until just two vertices are left.

Yeah, But How Do You Prove That? (#1)

Prüfer code of a tree: the sequence $P(T)$ constructed as follows.

- Find the leaf with the smallest label.
- Write down its neighbor (not the leaf itself!)
- Delete it.
- Repeat until just two vertices are left.

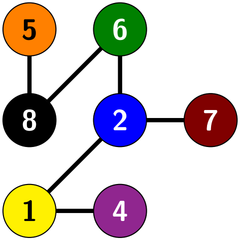


$P(T) =$

Yeah, But How Do You Prove That? (#1)

Prüfer code of a tree: the sequence $P(T)$ constructed as follows.

- Find the leaf with the smallest label.
- Write down its neighbor (not the leaf itself!)
- Delete it.
- Repeat until just two vertices are left.

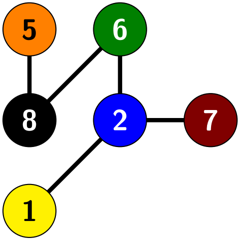


$P(T) = 7$

Yeah, But How Do You Prove That? (#1)

Prüfer code of a tree: the sequence $P(T)$ constructed as follows.

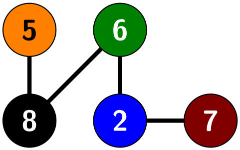
- Find the leaf with the smallest label.
- Write down its neighbor (not the leaf itself!)
- Delete it.
- Repeat until just two vertices are left.



$P(T) = \ 7\,1$

Prüfer code of a tree: the sequence $P(T)$ constructed as follows.

- Find the leaf with the smallest label.
- Write down its neighbor (not the leaf itself!)
- Delete it.
- Repeat until just two vertices are left.

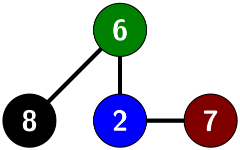


$P(T) = \; 7\,1\,2$

Yeah, But How Do You Prove That? (#1)

Prüfer code of a tree: the sequence $P(T)$ constructed as follows.

- Find the leaf with the smallest label.
- Write down its neighbor (not the leaf itself!)
- Delete it.
- Repeat until just two vertices are left.

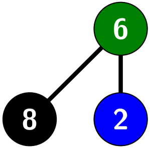


$P(T) = \ 7\,1\,2\,8$

Yeah, But How Do You Prove That? (#1)

Prüfer code of a tree: the sequence $P(T)$ constructed as follows.

- Find the leaf with the smallest label.
- Write down its neighbor (not the leaf itself!)
- Delete it.
- Repeat until just two vertices are left.

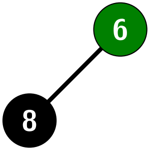


$P(T) = 7\ 1\ 2\ 8\ 2$

Yeah, But How Do You Prove That? (#1)

Prüfer code of a tree: the sequence $P(T)$ constructed as follows.

- Find the leaf with the smallest label.
- Write down its neighbor (not the leaf itself!)
- Delete it.
- Repeat until just two vertices are left.

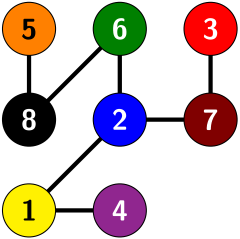


$P(T) = 7\ 1\ 2\ 8\ 2\ 6$

Yeah, But How Do You Prove That? (#1)

Prüfer code of a tree: the sequence $P(T)$ constructed as follows.

- Find the leaf with the smallest label.
- Write down its neighbor (not the leaf itself!)
- Delete it.
- Repeat until just two vertices are left.



$$P(T) = \; 7\,1\,2\,8\,2\,6$$

Yeah, But How Do You Prove That? (#1)

Fact: Every tree can be reconstructed from its Prüfer code, giving a bijection

$$\{\text{trees on } n \text{ vertices}\} \rightarrow \{(p_1, \ldots, p_{n-2}) : \ 1 \leq p_i \leq n\}$$

and the size of the right-hand set is clearly n^{n-2}.

Yeah, But How Do You Prove That? (#1)

Fact: Every tree can be reconstructed from its Prüfer code, giving a bijection

$$\{\text{trees on } n \text{ vertices}\} \;\to\; \{(p_1, \ldots, p_{n-2}) :\; 1 \leq p_i \leq n\}$$

and the size of the right-hand set is clearly n^{n-2}.

Corollary: The number of trees in which vertex i has exactly d_i neighbors is the coefficient of the monomial

$$x_1^{d_1} x_2^{d_2} \cdots x_n^{d_n}$$

in the expansion of $x_1 x_2 \cdots x_n (x_1 + x_2 + \cdots + x_n)^{n-2}$.

Yeah, But How Do You Prove That? (#2)

The **Matrix-Tree Theorem** (which dates back to 1845!) says that trees can be counted using linear algebra.

Long story short: $T(n)$ is the determinant of the $(n-1) \times (n-1)$ matrix

$$\begin{pmatrix} n-1 & -1 & -1 & \cdots & -1 \\ -1 & n-1 & -1 & \cdots & -1 \\ -1 & -1 & n-1 & \cdots & -1 \\ \vdots & \vdots & \vdots & & \vdots \\ -1 & -1 & -1 & \cdots & n-1 \end{pmatrix}$$

Convince yourself that its eigenvalues are n (with multiplicity $n-2$) and 1.

The Sandpile Model

Start with a bucket of sand. Take out m piles. Let s_i be the number of grains of sand in the i^{th} pile.

- When a sandpile gets too big, it topples over.

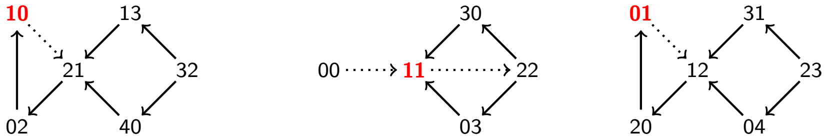
  Specifically, if $s_i \geq m$, then pile i spews sand in all directions, giving one grain of sand to each other pile and putting one grain back in the bucket.

- If no pile is too big, add one grain from the bucket to each pile.

How does the system evolve?

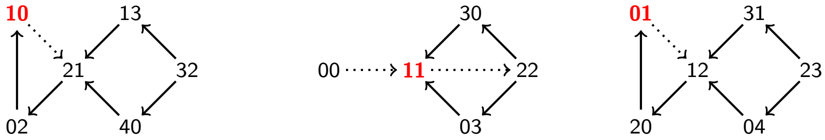
The Sandpile Model

Let $m = 2$. Record the state the model is in by the pair (s_1, s_2).



The Sandpile Model

Let $m = 2$. Record the state the model is in by the pair (s_1, s_2).

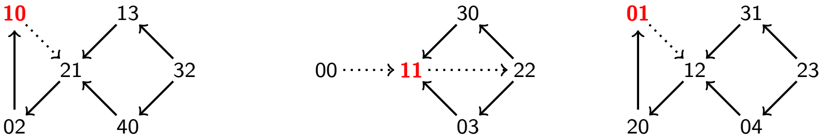


The states $(0,1)$, $(1,0)$, and $(1,1)$ are called **critical**:

- no pile other than the sink can topple ("stability")
- these states appear repeatedly as the model evolves ("recurrence")

The Sandpile Model

Let $m = 2$. Record the state the model is in by the pair (s_1, s_2).



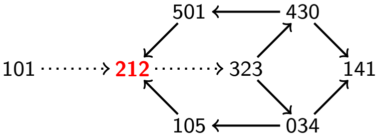
The states $(0, 1)$, $(1, 0)$, and $(1, 1)$ are called **critical**:

- no pile other than the sink can topple ("stability")
- these states appear repeatedly as the model evolves ("recurrence")

Fact: Every initial state evolves to exactly one critical state.

The Sandpile Model

A possible evolution pattern for $m = 3$:



Complete list of critical states for $m = 3$:

222,   221, 212, 122,   211, 121, 112,   220, 202, 022,
012, 021, 102, 120, 201, 210.

Sandpiles and Shopping Sprees

Sandpile model (statistical physics)	**Dollar game** (economics)
Sandpiles	Consumers
Sand grains	Dollars
Big enough	Rich enough
Toppling	Shopping spree
Sink	Bank
Sink topples	Economic stimulus package

Joint Shopping Sprees

Does a shopping spree really require $\$m$?

Joint Shopping Sprees

Does a shopping spree really require $\$m$?

Suppose that Ani and Bob go on shopping sprees at the same time.

Joint Shopping Sprees

Does a shopping spree really require $\$m$?

Suppose that Ani and Bob go on shopping sprees at the same time.

Ani gives a dollar to Bob
Ani gives a dollar to Chris

Joint Shopping Sprees

Does a shopping spree really require \$$m$?

Suppose that Ani and Bob go on shopping sprees at the same time.

Ani gives a dollar to Bob	Bob gives a dollar to Ani
Ani gives a dollar to Chris	Bob gives a dollar to Chris

Joint Shopping Sprees

Does a shopping spree really require \$$m$?

Suppose that Ani and Bob go on shopping sprees at the same time.

~~Ani gives a dollar to Bob~~	~~Bob gives a dollar to Ani~~
Ani gives a dollar to Chris	Bob gives a dollar to Chris

Joint Shopping Sprees

Does a shopping spree really require \$$m$?

Suppose that Ani and Bob go on shopping sprees at the same time.

~~Ani gives a dollar to Bob~~	~~Bob gives a dollar to Ani~~
Ani gives a dollar to Chris	Bob gives a dollar to Chris

Ani and Bob only need \$1 each to go on a shopping spree *together*.

Joint Shopping Sprees

Does a shopping spree really require \$$m$?

Suppose that Ani and Bob go on shopping sprees at the same time.

~~Ani gives a dollar to Bob~~	~~Bob gives a dollar to Ani~~
Ani gives a dollar to Chris	Bob gives a dollar to Chris

Ani and Bob only need \$1 each to go on a shopping spree *together*.

In a **joint shopping spree**, each consumer in a set X (not including the bank) gives \$1 to each consumer not in X (including the bank). This is possible if

$$s_i > m - |X| \qquad \forall x \in X.$$

Superstable States

A state of the dollar game is called **superstable** if no simultaneous shopping sprees are possible.

Theorem 2
$(s_1, \ldots, s_m)$ superstable $\iff$ $(m - s_1, \ldots, m - s_m)$ critical

Theorem 3
There is a bijection

$$\{\text{superstable states}\} \to \{\text{labeled trees on } n \text{ vertices}\}.$$

The proof uses the **Burning Algorithm** [Dhar, 1990].

Dhar's Burning Algorithm (A Sketch)

Let $n = m + 1$. Start with a superstable state $\mathbf{s} = (s_1, \ldots, s_{n-1})$.

- For each $i = 1, \ldots, n-1$, place s_i firefighters at vertex i.
- Set vertex n on fire.
- The fire tries to spread from burned vertices to unburned vertices. Unburned vertices can deploy firefighters to protect themselves. (A firefighter cannot be moved once deployed.)
- Superstability of $\mathbf{s}$ is precisely equivalent to the condition that the fire eventually reaches every vertex!
- The route that the fire takes is a tree!
- Algorithm is reversible: $\mathbf{s}$ can be reconstructed from the output tree!

Parking Functions

- There are n parking spaces on a one-way street, labeled $0, \ldots, n-1$.

Parking Functions

- There are n parking spaces on a one-way street, labeled $0, \ldots, n-1$.
- Along come n cars trying to park. Each car has a preferred spot p_i.

Parking Functions

- There are n parking spaces on a one-way street, labeled $0, \ldots, n-1$.
- Along come n cars trying to park. Each car has a preferred spot p_i.
- Each car drives to its preferred spot and tries to park there.

Parking Functions

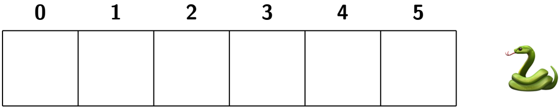
- There are n parking spaces on a one-way street, labeled $0, \ldots, n-1$.
- Along come n cars trying to park. Each car has a preferred spot p_i.
- Each car drives to its preferred spot and tries to park there.
- If a car's preferred spot is occupied, it takes the next available spot.

Parking Functions

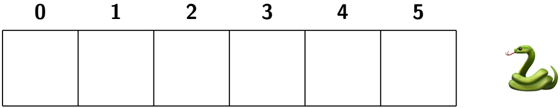
- There are n parking spaces on a one-way street, labeled $0, \ldots, n-1$.
- Along come n cars trying to park. Each car has a preferred spot p_i.
- Each car drives to its preferred spot and tries to park there.
- If a car's preferred spot is occupied, it takes the next available spot.
- Did I mention the pit full of snakes?

Parking Functions

Example #1 $(p_1, \ldots, p_6) = (0, 3, 0, 4, 3, 0)$

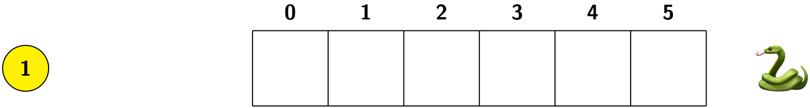


Example #2 $(p_1, \ldots, p_6) = (0, 3, 3, 4, 3, 0)$



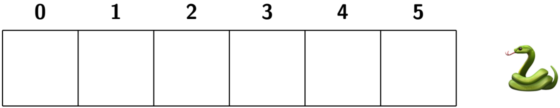
Parking Functions

Example #1 $(p_1, \ldots, p_6) = (0, 3, 0, 4, 3, 0)$



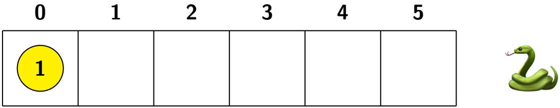
$p_1 = 0$

Example #2 $(p_1, \ldots, p_6) = (0, 3, 3, 4, 3, 0)$

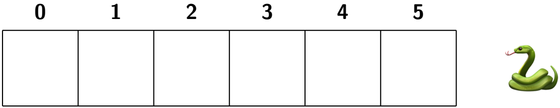


Parking Functions

Example #1 $(p_1, \ldots, p_6) = (0, 3, 0, 4, 3, 0)$

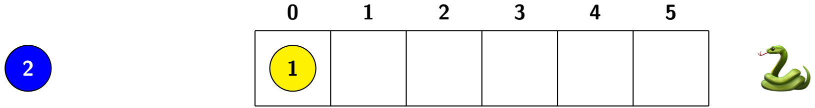


Example #2 $(p_1, \ldots, p_6) = (0, 3, 3, 4, 3, 0)$



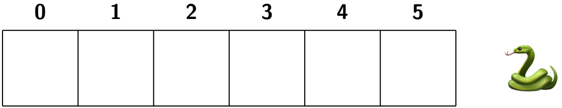
Parking Functions

Example #1 $(p_1, \ldots, p_6) = (0, 3, 0, 4, 3, 0)$



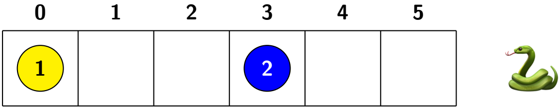
$p_2 = 3$

Example #2 $(p_1, \ldots, p_6) = (0, 3, 3, 4, 3, 0)$

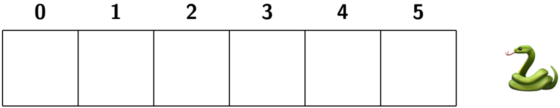


Parking Functions

Example #1 $(p_1, \ldots, p_6) = (0, 3, 0, 4, 3, 0)$

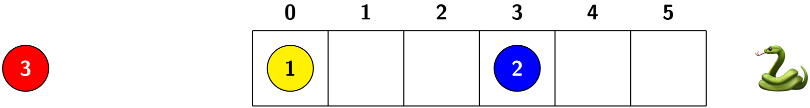


Example #2 $(p_1, \ldots, p_6) = (0, 3, 3, 4, 3, 0)$



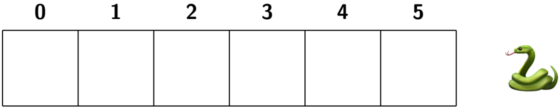
Parking Functions

Example #1 $(p_1, \ldots, p_6) = (0, 3, 0, 4, 3, 0)$



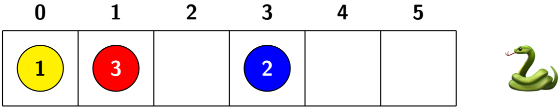
$p_3 = 0$

Example #2 $(p_1, \ldots, p_6) = (0, 3, 3, 4, 3, 0)$

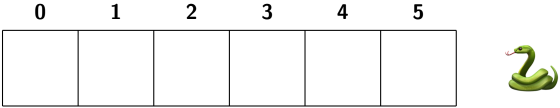


Parking Functions

Example #1 $(p_1, \ldots, p_6) = (0, 3, 0, 4, 3, 0)$

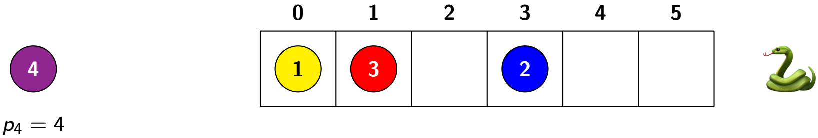


Example #2 $(p_1, \ldots, p_6) = (0, 3, 3, 4, 3, 0)$

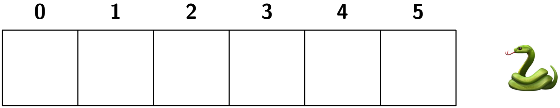


Parking Functions

Example #1 $(p_1, \ldots, p_6) = (0, 3, 0, 4, 3, 0)$

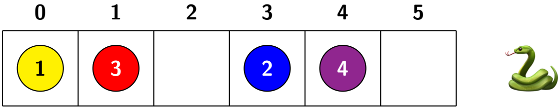


Example #2 $(p_1, \ldots, p_6) = (0, 3, 3, 4, 3, 0)$

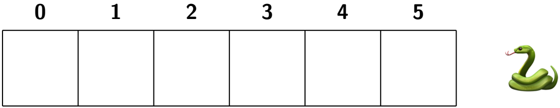


Parking Functions

Example #1 $(p_1, \ldots, p_6) = (0, 3, 0, 4, 3, 0)$

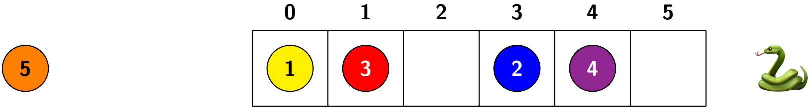


Example #2 $(p_1, \ldots, p_6) = (0, 3, 3, 4, 3, 0)$



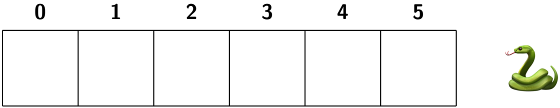
Parking Functions

Example #1 $(p_1, \ldots, p_6) = (0, 3, 0, 4, 3, 0)$



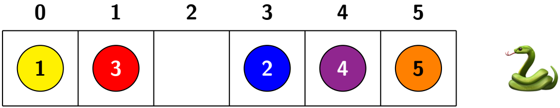
$p_5 = 3$

Example #2 $(p_1, \ldots, p_6) = (0, 3, 3, 4, 3, 0)$

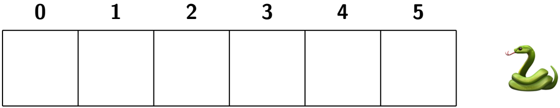


Parking Functions

Example #1 $(p_1, \ldots, p_6) = (0, 3, 0, 4, 3, 0)$

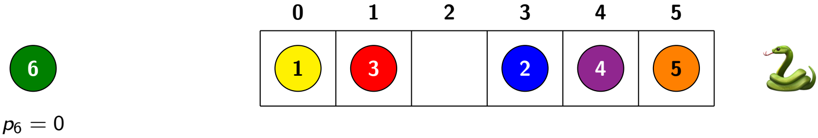


Example #2 $(p_1, \ldots, p_6) = (0, 3, 3, 4, 3, 0)$

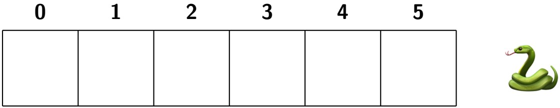


Parking Functions

Example #1 $(p_1, \ldots, p_6) = (0, 3, 0, 4, 3, 0)$



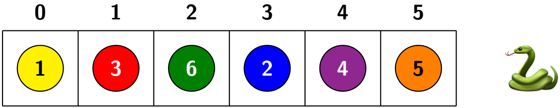
Example #2 $(p_1, \ldots, p_6) = (0, 3, 3, 4, 3, 0)$



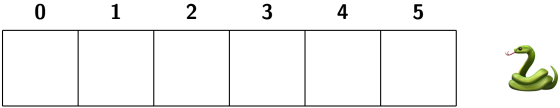
Parking Functions

Example #1 $(p_1, \ldots, p_6) = (0, 3, 0, 4, 3, 0)$

Success!



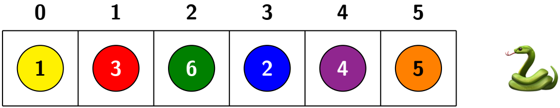
Example #2 $(p_1, \ldots, p_6) = (0, 3, 3, 4, 3, 0)$



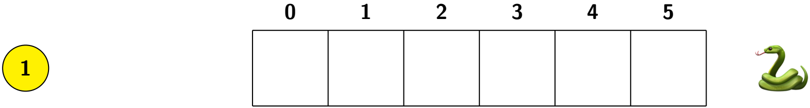
Parking Functions

Example #1 $(p_1, \ldots, p_6) = (0, 3, 0, 4, 3, 0)$

Success!



Example #2 $(p_1, \ldots, p_6) = (0, 3, 3, 4, 3, 0)$

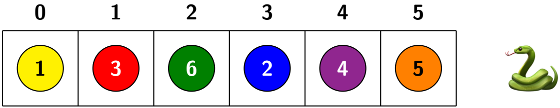


$p_1 = 0$

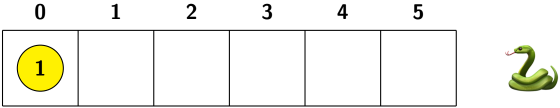
Parking Functions

Example #1 $(p_1, \ldots, p_6) = (0, 3, 0, 4, 3, 0)$

Success!

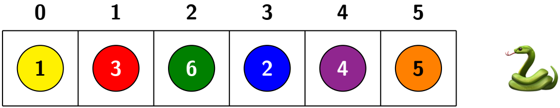


Example #2 $(p_1, \ldots, p_6) = (0, 3, 3, 4, 3, 0)$



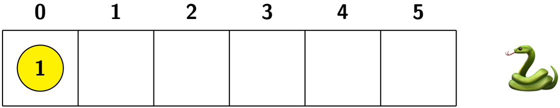
Example #1 $(p_1, \ldots, p_6) = (0, 3, 0, 4, 3, 0)$

Success!



Example #2 $(p_1, \ldots, p_6) = (0, 3, 3, 4, 3, 0)$

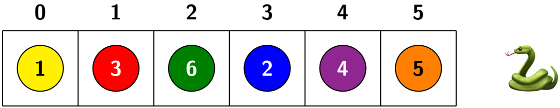
$p_2 = 3$



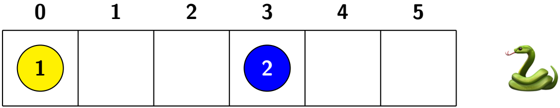
Parking Functions

Example #1 $(p_1, \ldots, p_6) = (0, 3, 0, 4, 3, 0)$

Success!



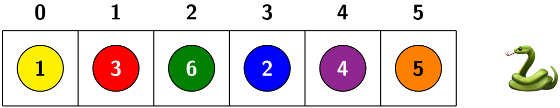
Example #2 $(p_1, \ldots, p_6) = (0, 3, 3, 4, 3, 0)$



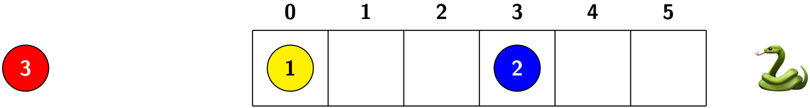
Parking Functions

Example #1 $(p_1, \ldots, p_6) = (0, 3, 0, 4, 3, 0)$

Success!



Example #2 $(p_1, \ldots, p_6) = (0, 3, 3, 4, 3, 0)$

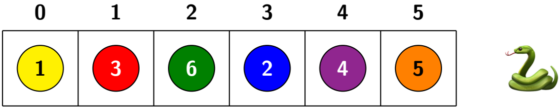


$p_3 = 3$

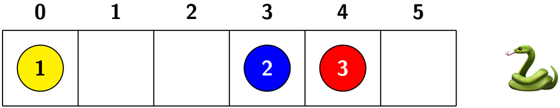
Parking Functions

Example #1 $(p_1, \ldots, p_6) = (0, 3, 0, 4, 3, 0)$

Success!



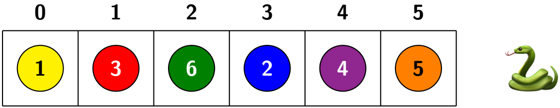
Example #2 $(p_1, \ldots, p_6) = (0, 3, 3, 4, 3, 0)$



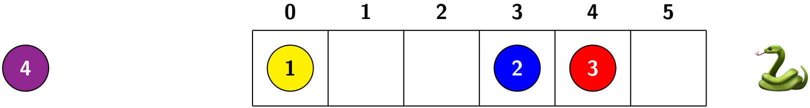
Parking Functions

Example #1 $(p_1, \ldots, p_6) = (0, 3, 0, 4, 3, 0)$

Success!



Example #2 $(p_1, \ldots, p_6) = (0, 3, 3, 4, 3, 0)$

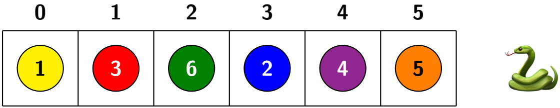


$p_4 = 4$

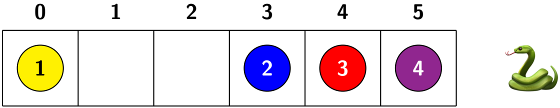
Parking Functions

Example #1 $(p_1, \ldots, p_6) = (0, 3, 0, 4, 3, 0)$

Success!



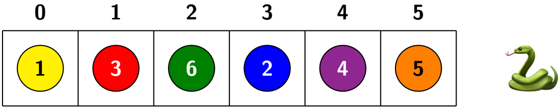
Example #2 $(p_1, \ldots, p_6) = (0, 3, 3, 4, 3, 0)$



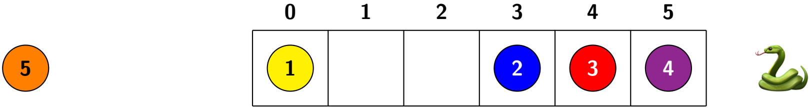
Parking Functions

Example #1 $(p_1, \ldots, p_6) = (0, 3, 0, 4, 3, 0)$

Success!



Example #2 $(p_1, \ldots, p_6) = (0, 3, 3, 4, 3, 0)$

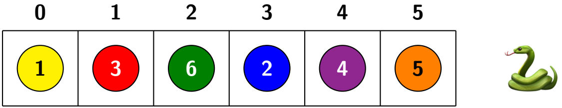


$p_5 = 3$

Parking Functions

Example #1 $(p_1, \ldots, p_6) = (0, 3, 0, 4, 3, 0)$

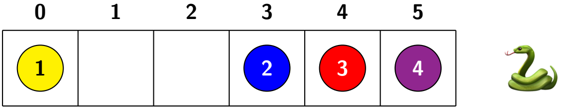
Success!



Example #2 $(p_1, \ldots, p_6) = (0, 3, 3, 4, 3, 0)$

5

Oops.



$p_5 = 3$

Parking Functions

Definition A sequence $\mathbf{p} = (p_1, \ldots, p_n)$ is a **parking function (PF)** if it enables all cars to park without being eaten by snakes.

Parking Functions

Definition A sequence $\mathbf{p} = (p_1, \ldots, p_n)$ is a **parking function (PF)** if it enables all cars to park without being eaten by snakes.

Theorem 4
$\mathbf{p}$ is a PF $\iff$ i^{th} smallest entry is $< i$ (for each i).

Parking Functions

Definition A sequence $\mathbf{p} = (p_1, \ldots, p_n)$ is a **parking function (PF)** if it enables all cars to park without being eaten by snakes.

Theorem 4
$\mathbf{p}$ is a PF $\iff$ i^{th} smallest entry is $< i$ (for each i).

(In particular, shuffling $\mathbf{p}$ does not change whether it is a PF.)

Parking Functions

Definition A sequence $\mathbf{p} = (p_1, \ldots, p_n)$ is a **parking function (PF)** if it enables all cars to park without being eaten by snakes.

Theorem 4
$\mathbf{p}$ is a PF $\iff$ i^{th} smallest entry is $< i$ (for each i).

(In particular, shuffling $\mathbf{p}$ does not change whether it is a PF.)

Theorem 5
The number of PF of length n is $(n+1)^{n-1}$.

Parking Functions

Definition A sequence $\mathbf{p} = (p_1, \ldots, p_n)$ is a **parking function (PF)** if it enables all cars to park without being eaten by snakes.

Theorem 4
$\mathbf{p}$ is a PF $\iff$ i^{th} smallest entry is $< i$ (for each i).

(In particular, shuffling $\mathbf{p}$ does not change whether it is a PF.)

Theorem 5
The number of PF of length n is $(n+1)^{n-1}$.

In fact, parking functions are the same thing as superstable states!

Parking Functions and Superstable States

$n = 1$: 0

$n = 2$:

00	01	
	10	

$n = 3$:

000	001	011	002	012 021
	010	101	020	102 120
	100	110	200	201 210

$n = 4$ (up to shuffling):

0000				
0001	0002	0003		
0011	0012	0013	0022	0023
0111	0112	0113	0122	0123

Number of PFs up to shuffling = **Catalan number $\frac{1}{n+1}\binom{2n}{n}$**

A Rather Slick Way To Count Parking Functions

- Remove the snakepit. Replace it with an extra parking spot ($\#n$) and a return ramp (like an airport terminal).
- Number of preference lists $\mathbf{p}$ is now $(n+1)^n$.
- All cars will be able to park, and one spot $o(\mathbf{p})$ will be left open.
- Cyclically rotating $\mathbf{p}$ also rotates $o(\mathbf{p})$.
- Therefore, all spots are equally likely to be open.
- $\mathbf{p}$ is a parking function $\iff o(\mathbf{p}) = n$.
- Number of parking functions $= (n+1)^n/(n+1) = (n+1)^{n-1}$.

Handicap Scoring

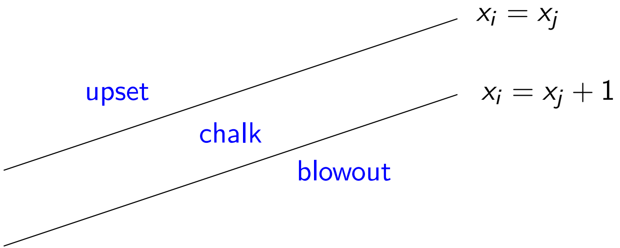
- Competitors in an individual event (e.g., marathon, bowling, pentathlon, Rubik's Cube) are seeded $1, 2, \ldots, n$.
  Lower numbered seed = stronger player.

- Each competitor i achieves a score $x_i \in \mathbb{R}$ (the higher the better).

- We want to level the playing field by comparing each pair of players head-to-head. For each $1 \leq i < j \leq n$:
  - If $x_i < x_j$ ("upset"), then the underdog j scores a point.
  - If $x_j < x_i < x_j + 1$ ("chalk"), then no one scores a point.
  - If $x_j + 1 < x_i$ ("blowout"), then the favorite i scores a point.

Handicap Scoring

- If $x_i < x_j$ ("upset"), then the underdog j scores a point.
- If $x_j < x_i < x_j + 1$ ("chalk"), then no one scores a point.
- If $x_j + 1 < x_i$ ("blowout"), then the favorite i scores a point.



The Shi Arrangement

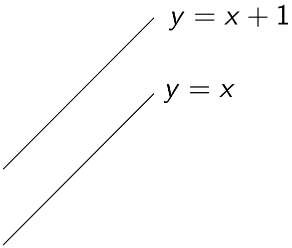
In order to understand the possible score vectors, we want to look at the hyperplanes in $\mathbb{R}^n$ defined by the equations

$$\begin{array}{lllll} x_1 = x_2, & x_1 = x_3, & \dots, & x_i = x_j, & \dots, \quad x_{n-1} = x_n, \\ x_1 = x_2 + 1, & x_1 = x_3 + 1, & \dots, & x_i = x_j + 1, & \dots, \quad x_{n-1} = x_n + 1. \end{array}$$

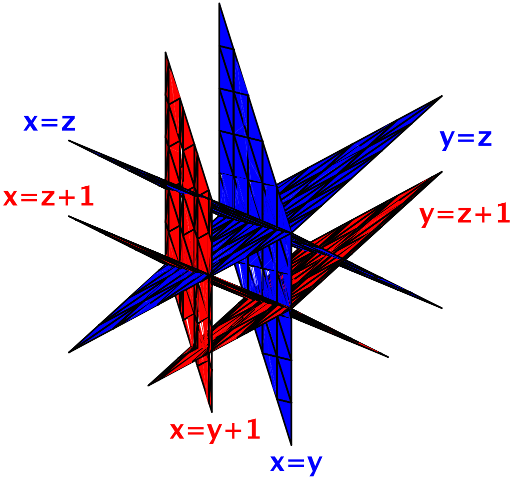
The **Shi arrangement** $Shi(n)$ is the set of all such hyperplanes.

The Shi arrangement separates $\mathbb{R}^n$ into regions that record the possible outcomes from this scoring system.

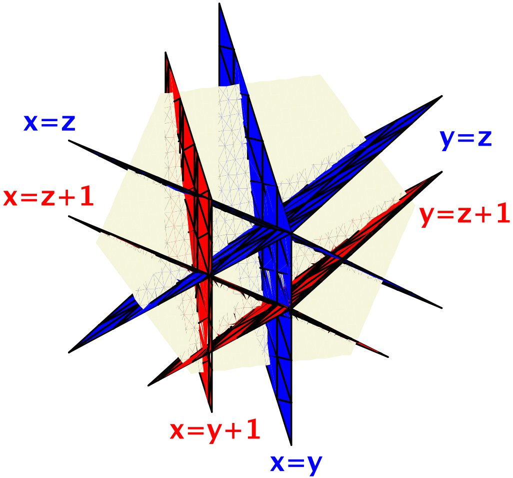
The Arrangement $Shi(2)$

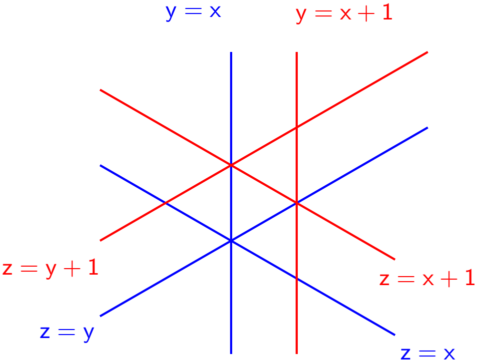


The Arrangement $Shi(3)$



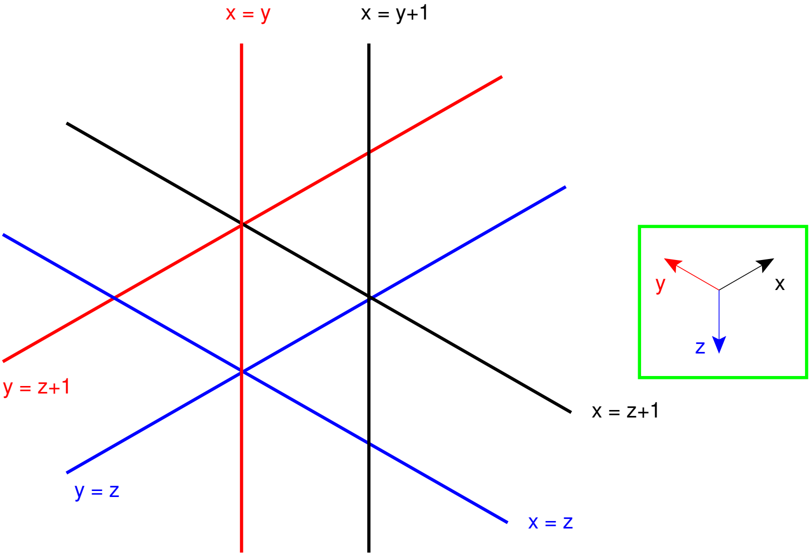
The Arrangement *Shi*(3)



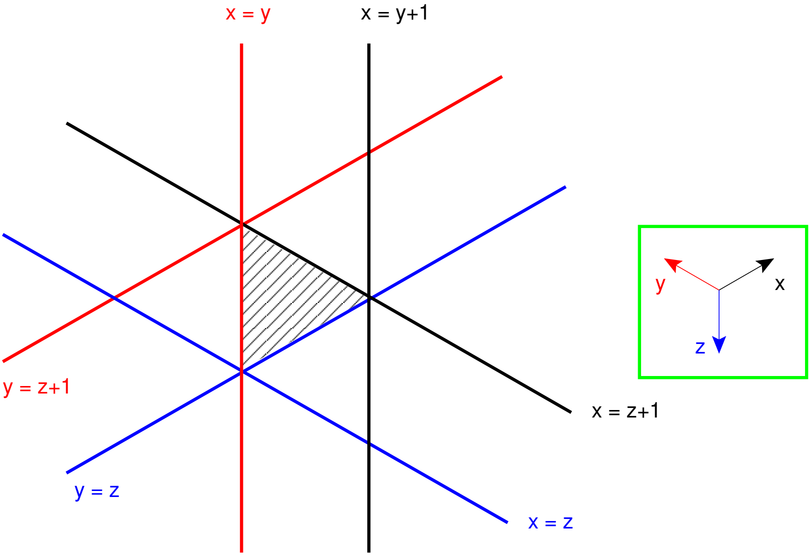


$y = x$
$y = x + 1$
$z = y + 1$
$z = x + 1$
$z = y$
$z = x$

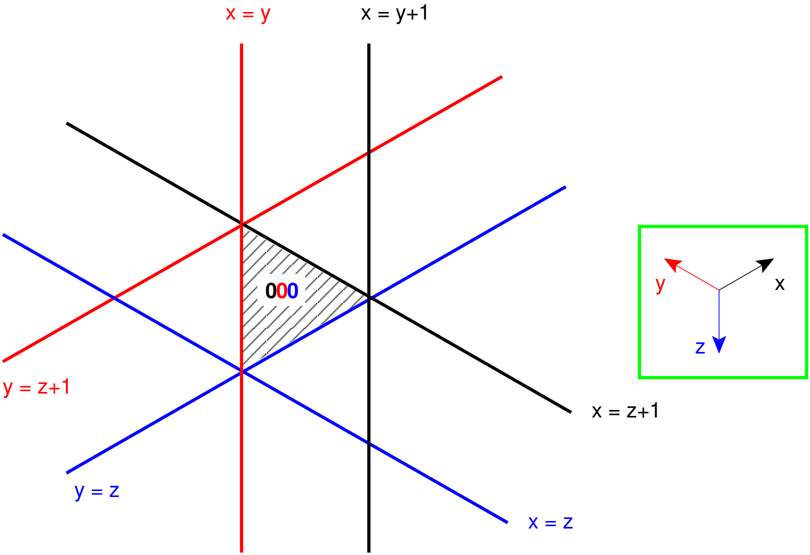
Score Vectors for *Shi*(3)



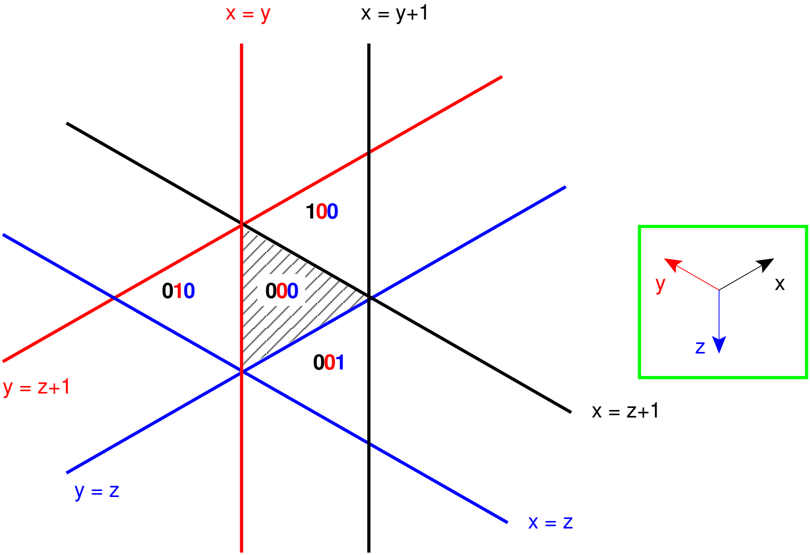
Score Vectors for *Shi*(3)



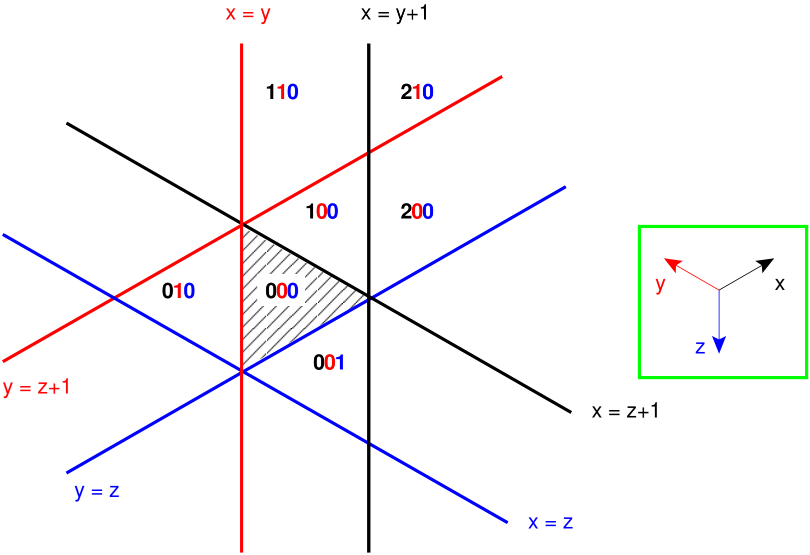
Score Vectors for *Shi*(3)



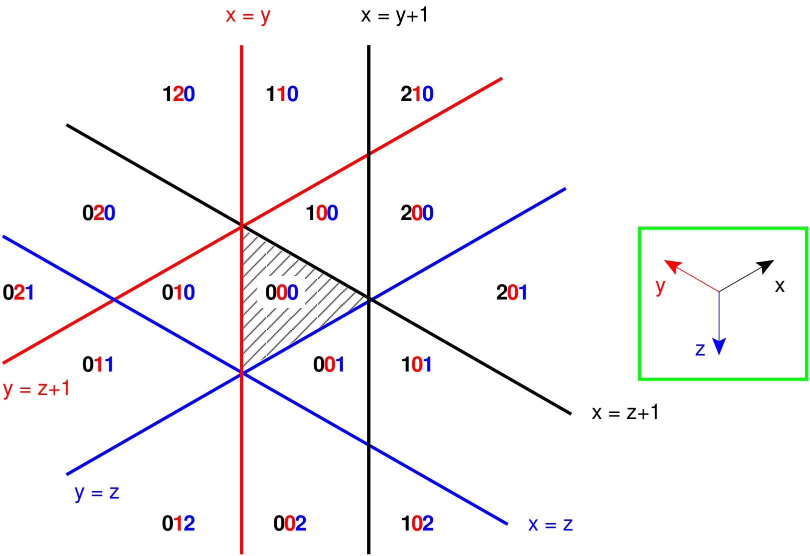
Score Vectors for $Shi(3)$



Score Vectors for *Shi*(3)



Score Vectors for *Shi*(3)



The Shi Arrangement

Theorem 6 [Pak and Stanley]
Labeling with score vectors gives a function

$$\{\text{regions of } \mathit{Shi}(n)\} \;\to\; \{\text{parking functions of length } n\}$$

that is a bijection!

In particular, the number of regions in $\mathit{Shi}(n)$ is $(n+1)^{n-1}$.

Conclusion

The numbers $(n+1)^{n-1}$ count lots of things:

- labeled trees on $n+1$ vertices,
- long-term behaviors of the sandpile model with n vertices plus a sink,
- superstable states of the dollar game with n vertices plus a bank,
- parking functions for n cars,
- regions of the Shi arrangement in $\mathbb{R}^n$,
- and, quite possibly, other beautiful combinatorial structures that you will discover yourself (and please tell me).

Thank you!