Local limit of directed animals on the square lattice

Arvind Singh
CNRS & Université Paris-Saclay

[joint work with **O. Hénard** et **E. Maurel-Ségala**]

GrHyDy2024: modèles spatiaux aléatoires
Lille, 23 octobre 2024

Directed animals on the square lattice

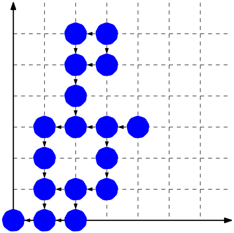
- **[directed] animal**: finite connected set of an [oriented] graph G.

But here we will only consider $G = \mathbb{N}^2 \ldots$

Directed animal (pyramid)

Finite set $\mathbf{A} \subset \mathbb{N}^2$ such that:

1. $0 \in \mathbf{A}$ (the source).
2. Any other site of $\mathbf{A}$ has a neighbor directly on its left or directly below it.



This is a directed animal

Directed animals on the square lattice

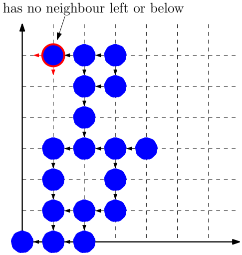
- **[directed] animal**: finite connected set of an [oriented] graph G.

But here we will only consider $G = \mathbb{N}^2 \ldots$

Directed animal (pyramid)

Finite set $\mathbf{A} \subset \mathbb{N}^2$ such that:

1. $0 \in \mathbf{A}$ (the source).
2. Any other site of $\mathbf{A}$ has a neighbor directly on its left or directly below it.



This is not a directed animal

Directed animals on the square lattice

Why directed animals ?

- Appears in the physics literature: *undirected* animals.
- Links with classical percolation.
- Directed animal: “partly exactly solvable model”.

(Very incomplete) biography

1. **Study of D.A. via hard sphere gaz model**

   ▶ Dhar (1982); Dhar Farni Barna (1982); Nadal (1982); Derrida Nadal Vannimenus (1982), Hakim Nadal (1983), Dhar (1983); Bousquet-Mélou, Conway (1996); Bousquet-Melou (1998); Le Borgne, Marckert (2007); Albenque (2009)

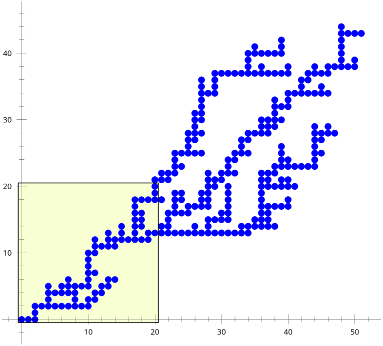
2. **Study of D.A. via des heap of pieces $\rightarrow$ bijections with trees**

   ▶ Viennot (1986); Betrema, Penaud (1993); Corteel, Denise et Gouyou-Beauchamps (2000); Bacher (2009)

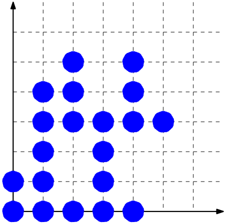
Goal of the talk

What does a large uniformly sampled random directed animal look like around the origin ?

- **We study the local limit of D.A. rooted at the origin.**
  (probabilistic vs combinatorics approach)

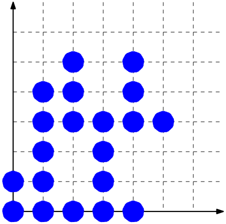


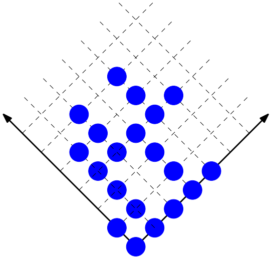
Viennot's heap of pieces



Viennot's heap of pieces

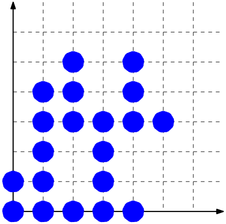
- Rotate $\mathbb{N}^2$ by 45 degrees.

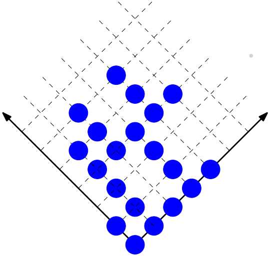


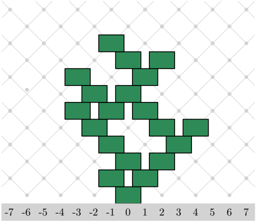


Viennot's heap of pieces

- Rotate $\mathbb{N}^2$ by 45 degrees.
- Replace each vertex by a domino (dimer) of height 1 and width $2 - \varepsilon$.

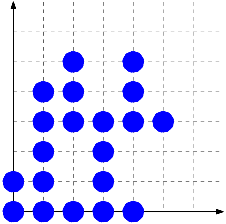


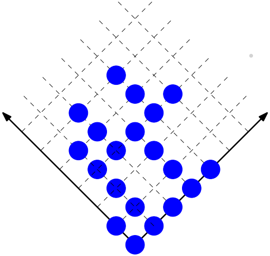


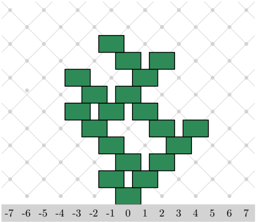


Viennot's heap of pieces

- Rotate $\mathbb{N}^2$ by 45 degrees.
- Replace each vertex by a domino (dimer) of height 1 and width $2-\varepsilon$.



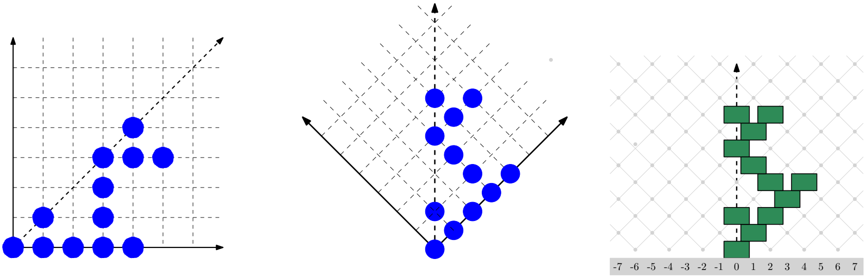




Pyramid $\iff$ Set of dominoes where one domino is on the floor and every other domino is supported by a domino under it

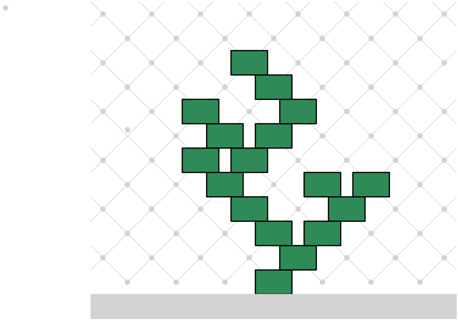
Viennot's heap of pieces

- Rotate $\mathbb{N}^2$ by 45 degrees.
- Replace each vertex by a domino (dimer) of height 1 and width $2 - \varepsilon$.



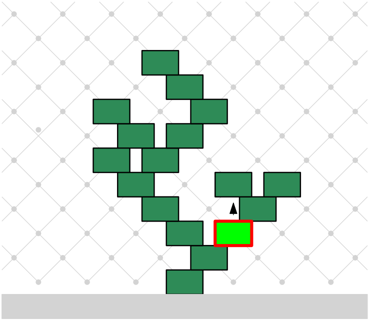
half-pyramid $\iff$ pyramid where all dominoes have non negative x-coordinates

Viennot’s heap of pieces : the “push-up” operation



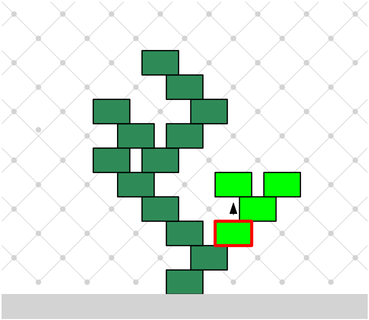
Viennot's heap of pieces : the "push-up" operation

- Lifting up a domino: bring along the pyramid sitting over it.



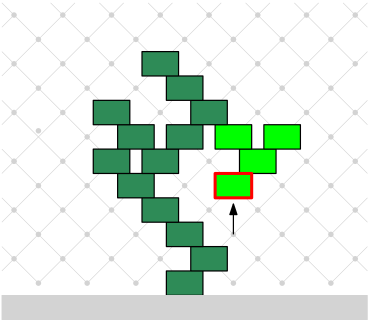
Viennot's heap of pieces : the "push-up" operation

- Lifting up a domino: bring along the pyramid sitting over it.



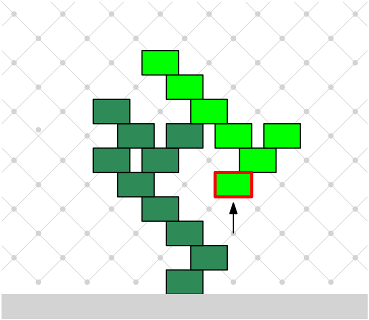
Viennot's heap of pieces : the "push-up" operation

- Lifting up a domino: bring along the pyramid sitting over it.



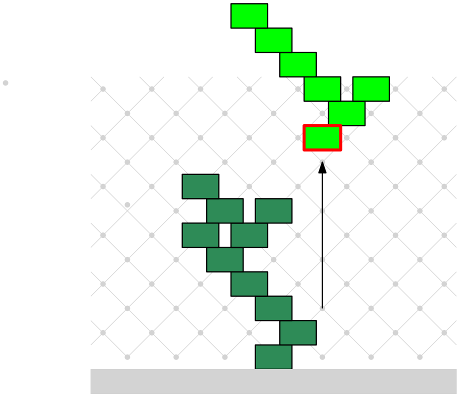
Viennot's heap of pieces : the "push-up" operation

- Lifting up a domino: bring along the pyramid sitting over it.



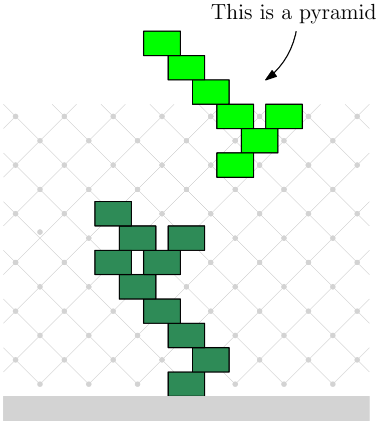
Viennot's heap of pieces : the "push-up" operation

- Lifting up a domino: bring along the pyramid sitting over it.



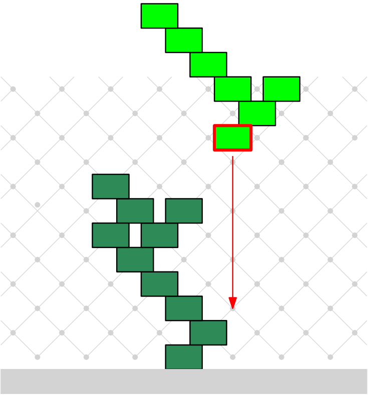
Viennot's heap of pieces : the "push-up" operation

- Lifting up a domino: bring along the pyramid sitting over it.
  ⇒ Generating series via the pyramid/half pyramid decomposition.



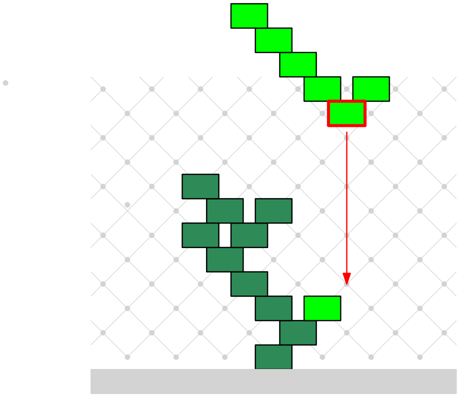
Viennot's heap of pieces : the "push-up" operation

- Lifting up a domino: bring along the pyramid sitting over it.
- Reverse operation: let domino drop "from infinity".



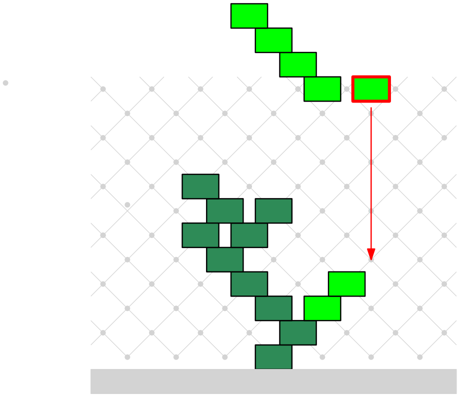
Viennot's heap of pieces : the "push-up" operation

- Lifting up a domino: bring along the pyramid sitting over it.
- Reverse operation: let domino drop "from infinity".



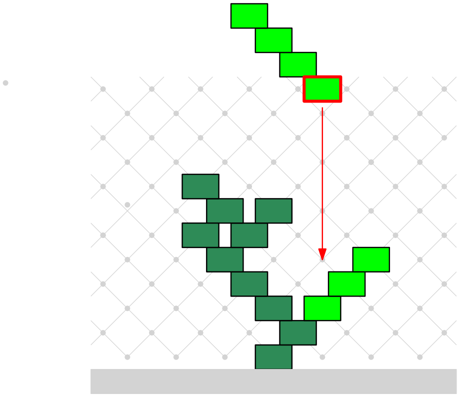
Viennot's heap of pieces : the "push-up" operation

- Lifting up a domino: bring along the pyramid sitting over it.
- Reverse operation: let domino drop "from infinity".



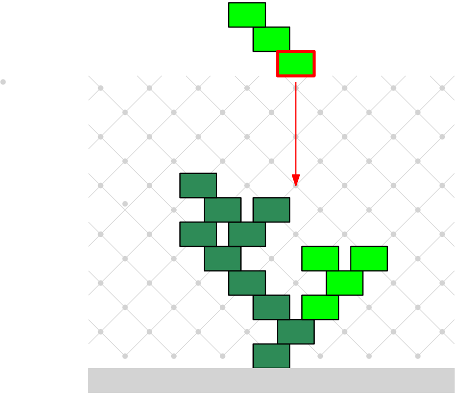
Viennot's heap of pieces : the "push-up" operation

- Lifting up a domino: bring along the pyramid sitting over it.
- Reverse operation: let domino drop "from infinity".



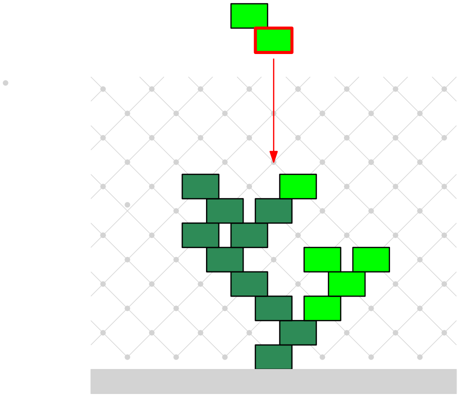
Viennot's heap of pieces : the "push-up" operation

- Lifting up a domino: bring along the pyramid sitting over it.
- Reverse operation: let domino drop "from infinity".



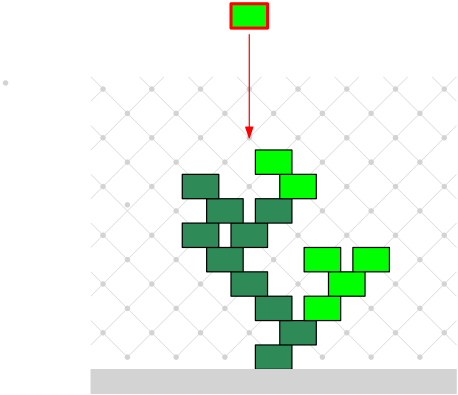
Viennot's heap of pieces : the "push-up" operation

- Lifting up a domino: bring along the pyramid sitting over it.
- Reverse operation: let domino drop "from infinity".



Viennot's heap of pieces : the "push-up" operation

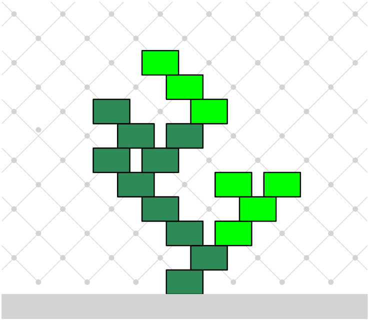
- Lifting up a domino: bring along the pyramid sitting over it.
- Reverse operation: let domino drop "from infinity".



Viennot's heap of pieces : the "push-up" operation

- Lifting up a domino: bring along the pyramid sitting over it.
- Reverse operation: let domino drop "from infinity".

$\Rightarrow$ construct an infinite pyramid by dropping dominos from ∞...



Local limits

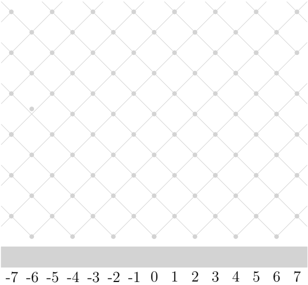
Path encoding of a directed animal

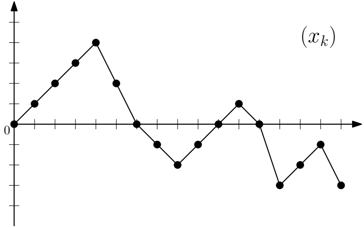
Proposition (Hénard, Maurel-Segala, S. (24))

Directed animals are in bijection with paths (x_k) *such that*

1. $x_{x+1} - x_k \in \mathbb{Z}_-^* \cup \{1\}$.
2. $x_k \geq \min_{i<k} x_i - 1$ *i.e.* x *never beats its current infimum by more than* 1.

Example.





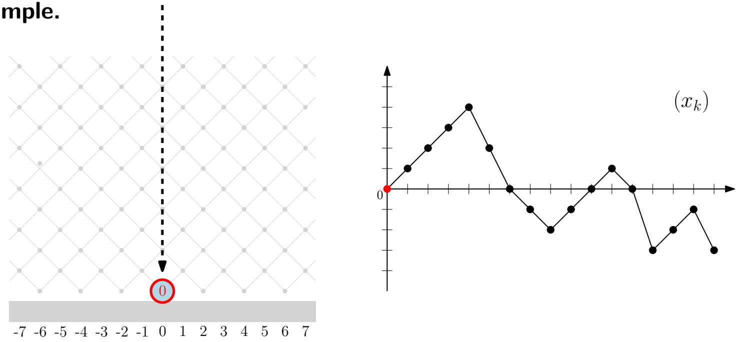
Path encoding of a directed animal

Proposition (Hénard, Maurel-Segala, S. (24))

Directed animals are in bijection with paths (x_k) *such that*

1. $x_{x+1} - x_k \in \mathbb{Z}_-^* \cup \{1\}$.
2. $x_k \geq \min_{i<k} x_i - 1$ *i.e.* x *never beats its current infimum by more than* 1.

Example.



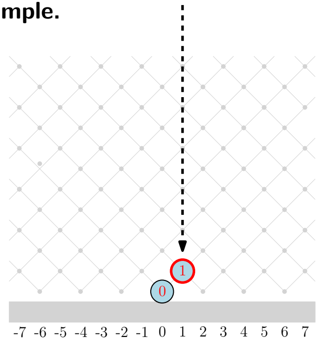
Path encoding of a directed animal

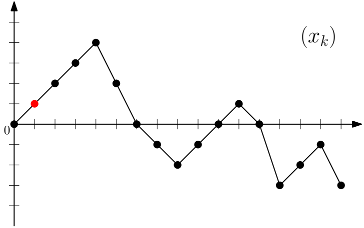
Proposition (Hénard, Maurel-Segala, S. (24))

Directed animals are in bijection with paths (x_k) such that

1. $x_{x+1} - x_k \in \mathbb{Z}_-^* \cup \{1\}$.
2. $x_k \geq \min_{i<k} x_i - 1$ *i.e. x never beats its current infimum by more than* 1.

Example.





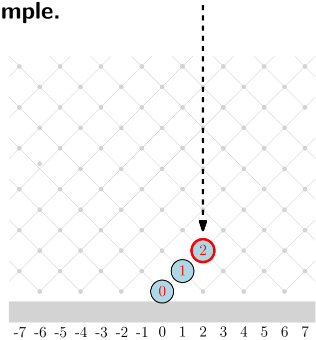
Path encoding of a directed animal

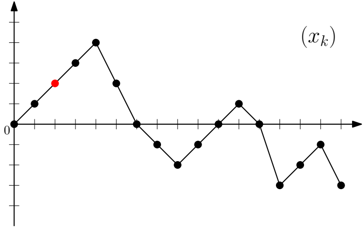
Proposition (Hénard, Maurel-Segala, S. (24))

Directed animals are in bijection with paths (x_k) *such that*

1. $x_{x+1} - x_k \in \mathbb{Z}_-^* \cup \{1\}$.
2. $x_k \geq \min_{i<k} x_i - 1$ *i.e. x never beats its current infimum by more than* 1.

Example.





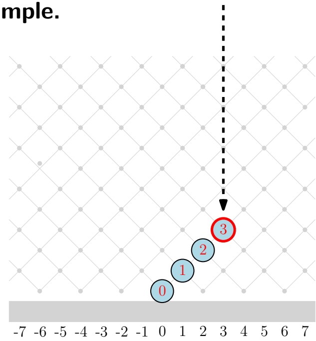
Path encoding of a directed animal

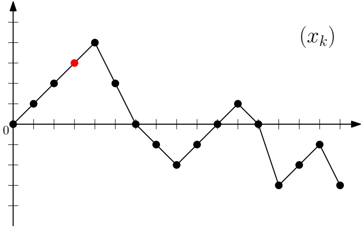
Proposition (Hénard, Maurel-Segala, S. (24))

Directed animals are in bijection with paths (x_k) *such that*

1. $x_{x+1} - x_k \in \mathbb{Z}_-^* \cup \{1\}$.
2. $x_k \geq \min_{i<k} x_i - 1$ *i.e.* x *never beats its current infimum by more than* 1.

Example.





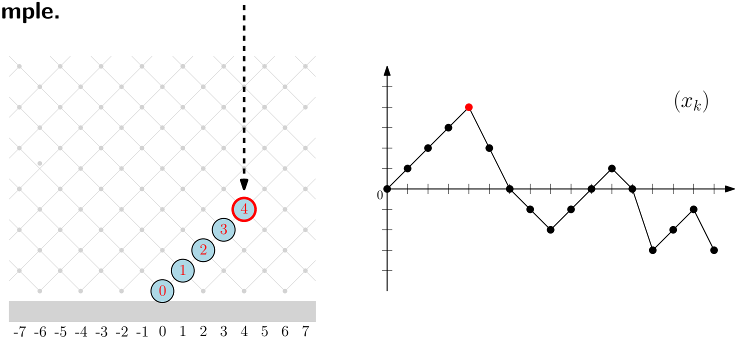
Path encoding of a directed animal

Proposition (Hénard, Maurel-Segala, S. (24))

Directed animals are in bijection with paths (x_k) *such that*

1. $x_{x+1} - x_k \in \mathbb{Z}_-^* \cup \{1\}$.
2. $x_k \geq \min_{i<k} x_i - 1$ *i.e. x never beats its current infimum by more than* 1.

Example.



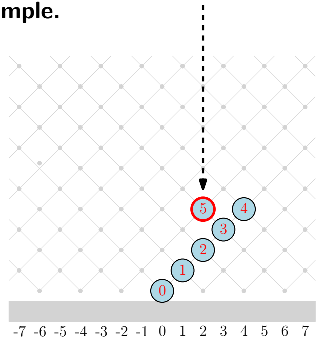
Path encoding of a directed animal

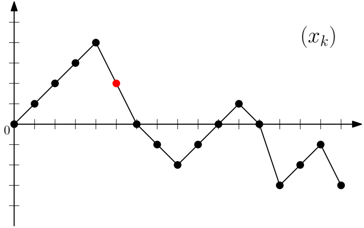
Proposition (Hénard, Maurel-Segala, S. (24))

Directed animals are in bijection with paths (x_k) *such that*

1. $x_{x+1} - x_k \in \mathbb{Z}_-^* \cup \{1\}$.
2. $x_k \geq \min_{i<k} x_i - 1$ *i.e.* x *never beats its current infimum by more than* 1.

Example.





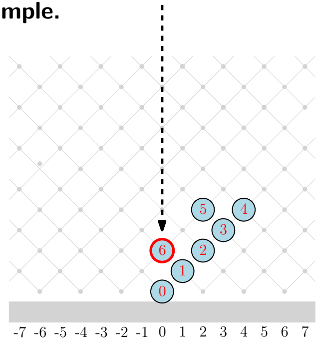
Path encoding of a directed animal

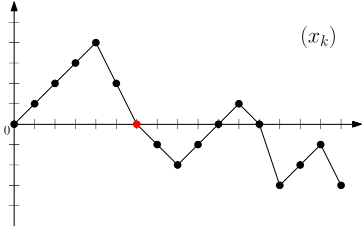
Proposition (Hénard, Maurel-Segala, S. (24))

Directed animals are in bijection with paths (x_k) *such that*

1. $x_{x+1} - x_k \in \mathbb{Z}_-^* \cup \{1\}$.
2. $x_k \geq \min_{i<k} x_i - 1$ *i.e.* x *never beats its current infimum by more than* 1.

Example.





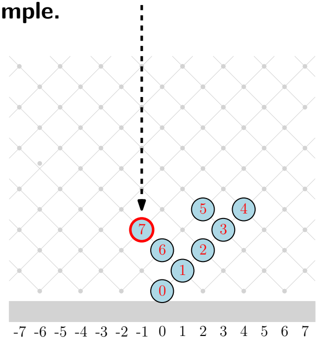
Path encoding of a directed animal

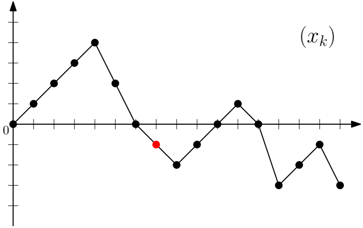
Proposition (Hénard, Maurel-Segala, S. (24))

Directed animals are in bijection with paths (x_k) *such that*

1. $x_{x+1} - x_k \in \mathbb{Z}_-^* \cup \{1\}$.
2. $x_k \geq \min_{i<k} x_i - 1$ *i.e.* x *never beats its current infimum by more than* 1.

Example.





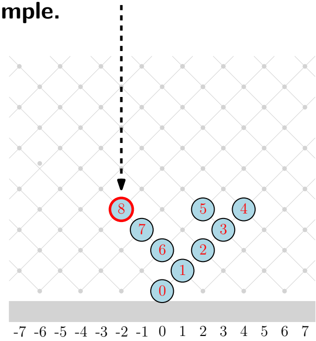
Path encoding of a directed animal

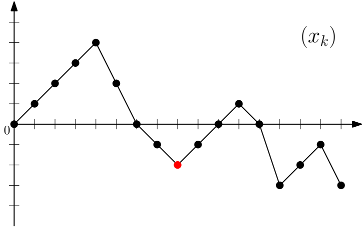
Proposition (Hénard, Maurel-Segala, S. (24))

Directed animals are in bijection with paths (x_k) *such that*

1. $x_{x+1} - x_k \in \mathbb{Z}_-^* \cup \{1\}$.
2. $x_k \geq \min_{i<k} x_i - 1$ *i.e. x never beats its current infimum by more than* 1.

Example.





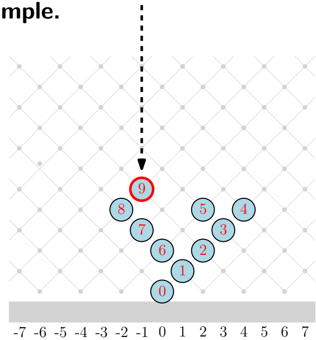
Path encoding of a directed animal

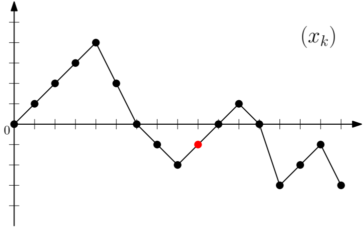
Proposition (Hénard, Maurel-Segala, S. (24))

Directed animals are in bijection with paths (x_k) *such that*

1. $x_{x+1} - x_k \in \mathbb{Z}_-^* \cup \{1\}$.
2. $x_k \geq \min_{i<k} x_i - 1$ *i.e.* x *never beats its current infimum by more than* 1.

Example.





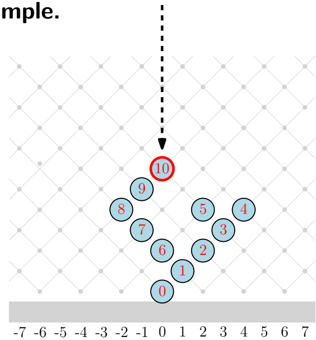
Path encoding of a directed animal

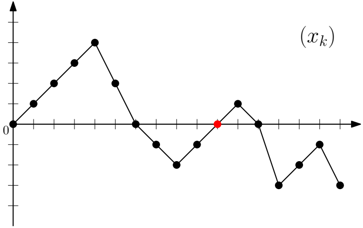
Proposition (Hénard, Maurel-Segala, S. (24))

Directed animals are in bijection with paths (x_k) *such that*

1. $x_{x+1} - x_k \in \mathbb{Z}_-^* \cup \{1\}$.
2. $x_k \geq \min_{i<k} x_i - 1$ *i.e.* x *never beats its current infimum by more than* 1.

Example.





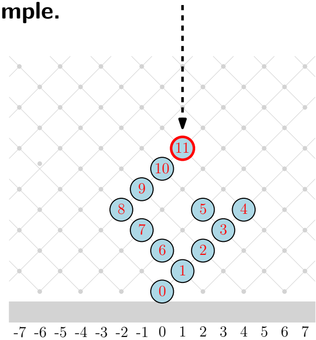
Path encoding of a directed animal

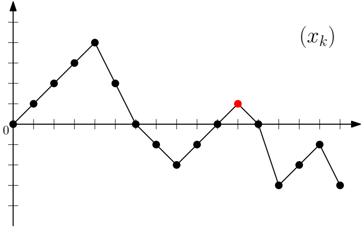
Proposition (Hénard, Maurel-Segala, S. (24))

Directed animals are in bijection with paths (x_k) *such that*

1. $x_{x+1} - x_k \in \mathbb{Z}_-^* \cup \{1\}$.
2. $x_k \geq \min_{i<k} x_i - 1$ *i.e. x never beats its current infimum by more than* 1.

Example.





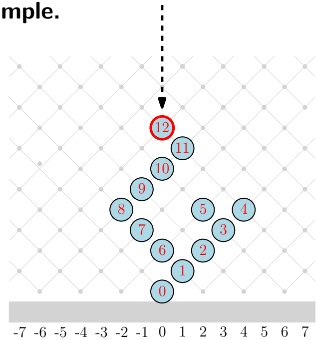
Path encoding of a directed animal

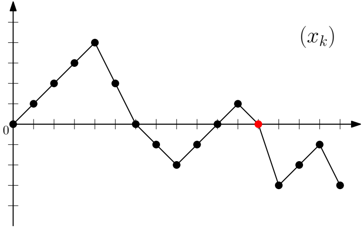
Proposition (Hénard, Maurel-Segala, S. (24))

Directed animals are in bijection with paths (x_k) *such that*

1. $x_{x+1} - x_k \in \mathbb{Z}_-^* \cup \{1\}$.
2. $x_k \geq \min_{i<k} x_i - 1$ *i.e.* x *never beats its current infimum by more than* 1.

Example.





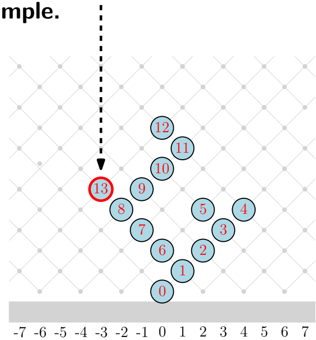
Path encoding of a directed animal

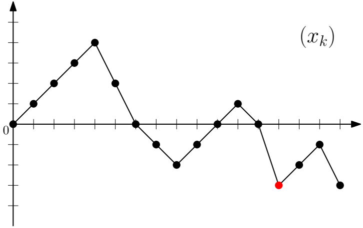
Proposition (Hénard, Maurel-Segala, S. (24))

Directed animals are in bijection with paths (x_k) *such that*

1. $x_{x+1} - x_k \in \mathbb{Z}_-^* \cup \{1\}$.
2. $x_k \geq \min_{i<k} x_i - 1$ *i.e.* x *never beats its current infimum by more than* 1.

Example.





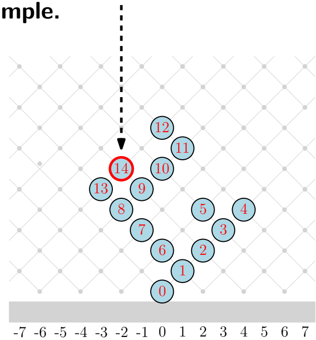
Path encoding of a directed animal

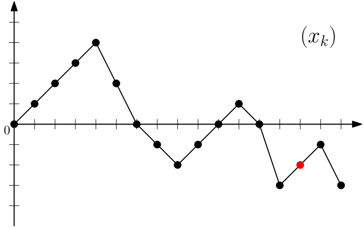
Proposition (Hénard, Maurel-Segala, S. (24))

Directed animals are in bijection with paths (x_k) *such that*

1. $x_{x+1} - x_k \in \mathbb{Z}_-^* \cup \{1\}$.
2. $x_k \geq \min_{i<k} x_i - 1$ *i.e.* x *never beats its current infimum by more than* 1.

Example.





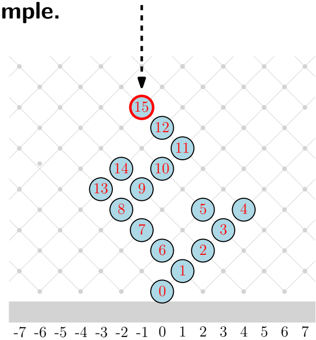
Path encoding of a directed animal

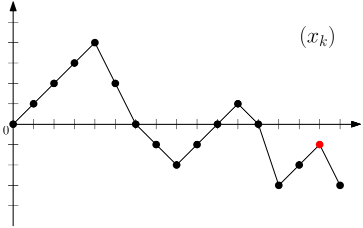
Proposition (Hénard, Maurel-Segala, S. (24))

Directed animals are in bijection with paths (x_k) such that

1. $x_{x+1} - x_k \in \mathbb{Z}_-^* \cup \{1\}$.
2. $x_k \geq \min_{i<k} x_i - 1$ *i.e. x never beats its current infimum by more than* 1.

Example.





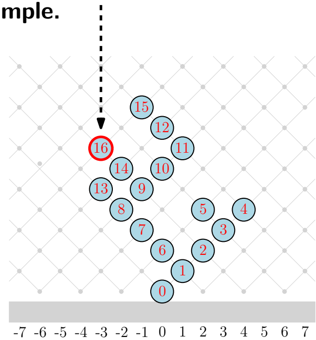
Path encoding of a directed animal

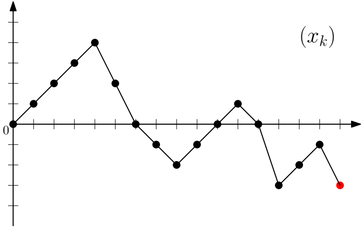
Proposition (Hénard, Maurel-Segala, S. (24))

Directed animals are in bijection with paths (x_k) *such that*

1. $x_{x+1} - x_k \in \mathbb{Z}_-^* \cup \{1\}$.
2. $x_k \geq \min_{i<k} x_i - 1$ *i.e. x never beats its current infimum by more than* 1.

Example.





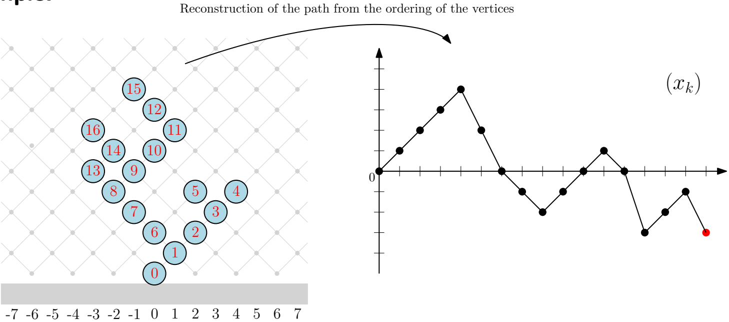
Path encoding of a directed animal

Proposition (Hénard, Maurel-Segala, S. (24))

Directed animals are in bijection with paths (x_k) *such that*

1. $x_{x+1} - x_k \in \mathbb{Z}_-^* \cup \{1\}$.
2. $x_k \geq \min_{i<k} x_i - 1$ *i.e.* x *never beats its current infimum by more than* 1.

Example.



Path encoding of a directed animal

Proposition (Hénard, Maurel-Segala, S. (24))

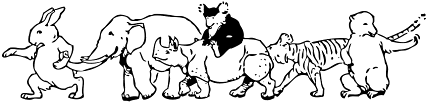
Directed animals are in bijection with paths (x_k) *such that*

1. $x_{x+1} - x_k \in \mathbb{Z}_-^* \cup \{1\}$.
2. $x_k \geq \min_{i<k} x_i - 1$ *i.e.* x *never beats its current infimum by more than* 1.

Remarks

- $|A| = |x|$. Bijections between paths and DA of same size.
- Bijection between half-pyramids and non-negative paths.
- Extends into a bijection between infinite paths and *simple* infinite DA.
- Encoding similar to Lukasiewicz's encoding for trees
  (Gouyou Beauchamps - Viennot $\approx$ Dyck's encoding for trees)

The animal walk



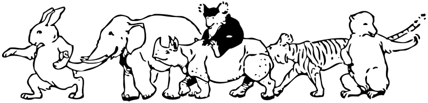
(X_n) sequence of i.i.d random variables with law

$$\mathbb{P}(X_n = k) = \frac{2^k}{3}\mathbf{1}_{k\in\mathbb{Z}_-^*\cup\{1\}}.$$

Remark:

- $X_n = +1$ with probability $2/3$,
- $X_n = -\text{Geom}(1/2)$ with probability $1/3$.

The animal walk



(X_n) sequence of i.i.d random variables with law

$$\mathbb{P}(X_n = k) = \frac{2^k}{3}\mathbf{1}_{k\in\mathbb{Z}^*_-\cup\{1\}}.$$

Remark:

- $X_n = +1$ with probability $2/3$,
- $X_n = -\text{Geom}(1/2)$ with probability $1/3$.

The **animal walk** is the random walk S with $S_0 = 0$ and

$$S_n = X_1 + \ldots + X_n.$$

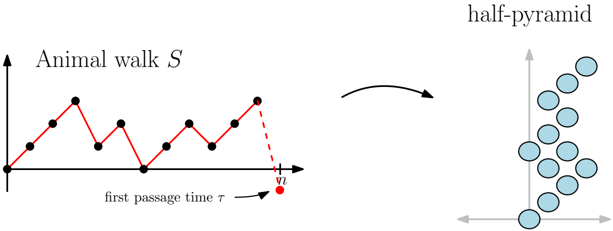
$E[S_n] = 0$, the walk is recurrent and we define

$$\tau = \inf\{n : S_n < 0\}$$

The animal walk

Proposition

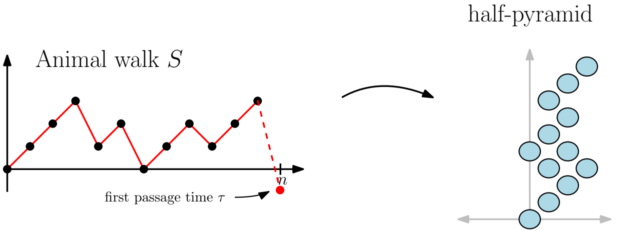
A random uniform half-pyramid with n vertices can be sampled from an excursion of the animal walk $(S_0, \ldots, S_{\tau-1})$ conditioned on $\{\tau = n\}$



The animal walk

Proposition

A random uniform half-pyramid with n vertices can be sampled from an excursion of the animal walk $(S_0, \ldots, S_{\tau-1})$ conditioned on $\{\tau = n\}$



Proof.

Given a path $0 = x_0, x_1, \ldots, x_n = -1$ with $x_{i+1} - x_i \in \mathbb{Z}_-^* \cup \{1\}$,

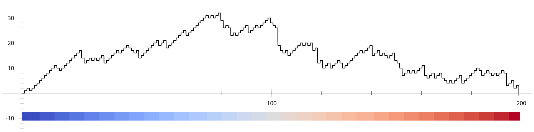
$$\mathbb{P}(S_0 = x_0, S_1 = x_1, \ldots, S_n = x_n) = \prod_{i=1}^{n} \mathbb{P}(X_i = x_i - x_{i-1}) = \prod_{i=1}^{n} \frac{2^{x_i - x_{i-1}}}{3} = \frac{2^{x_n - x_0}}{3^n} = \frac{1}{2.3^n}$$

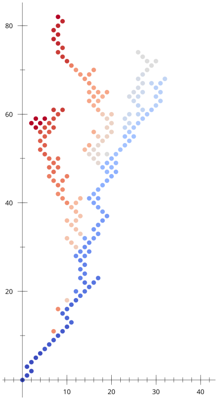
□

The Boltzmann half-pyramid

Définition

A Boltzmann half-pyramid (BHP) is a random directed animal constructed from a "free" positive excursion $(S_0, S_1, \ldots, S_{\tau-1})$ of the animal walk.



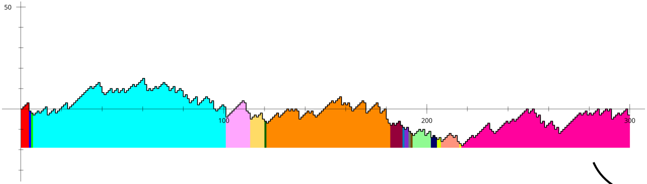


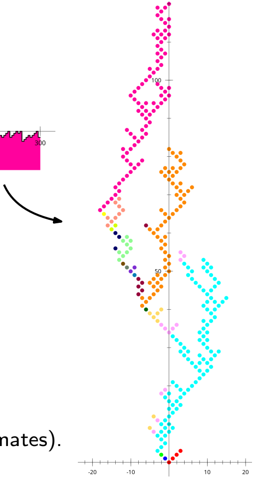
- BHP = critical half-pyramid ($\Longleftrightarrow$ G.W. Geom($\frac{1}{2}$)).
- "building block" to construct local limits.
- There is no Boltzmann pyramid !

Local limit: the uniform infinite pyramid (UIP)

Theorem (Hénard, Maurel-Segala, S. (24))

The local limit (rooted at 0) of a uniformly sampled directed animal as its size goes to infinity exists and is constructed by piling up i.i.d. BHP's.



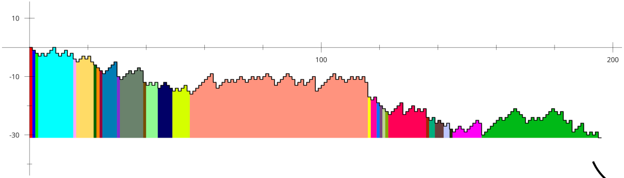


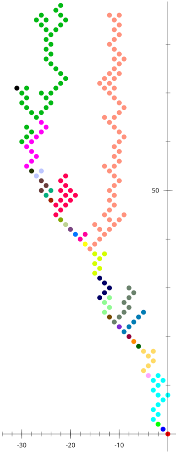
- The limit is non-trivial, random and **simple** a.s.
- "Kesten decomposition" of this critical object:
  - backbone $= 0, -1, -2, \ldots$.
  - BHP's $=$ finite pyramids grafted on the backbone.
- **Proof**: Directly on the animal walk (using hitting time estimates).

Local limit: the uniform infinite half-pyramid (UIHP)

Theorem (Hénard, Maurel-Segala, S. (24))

The local limit of a uniformly sampled **non-positive half-pyramid** *is constructed by piling up i.i.d. BHP's with the n-th BHP conditioned to have width at most n.*





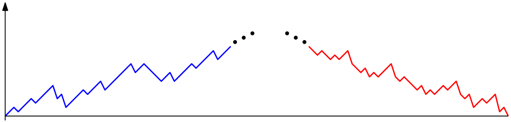
- Doob's conditioning of the animal walk (h-transform).
- "Kesten decomposition":
  - backbone (épine) : $0, -1, -2, \dots$.
  - conditioned BHP's: finite pyramids grafted on the backbone

Local limit: the uniform infinite half-pyramid (UIHP)

Theorem (Hénard, Maurel-Segala, S. (24))

The local limit of a uniformly sampled **non-negative half-pyramid** *is constructed by piling up i.i.d. BHP's on top of the animal obtained from the animal walk conditioned to stay non-negative.*

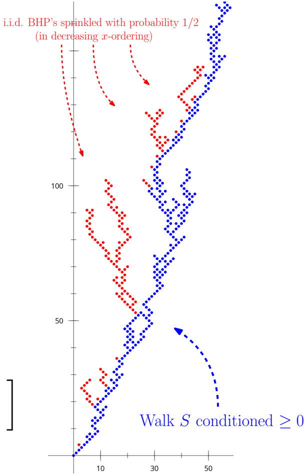
- It is the limit of BHP's when their size grow to ∞.
  - Partie bleu: start of excursion.
  - Partie rouge: end of excursion



“infinite” excursion

- Main tool: **Martingales!**

$\left[\begin{array}{c}\text{Doob's conditionning}\\ \text{on the animal walk } S\end{array}\right] \leftrightarrow \left[\begin{array}{c}\text{Doob's conditionning}\\ \text{on DA kernels}\end{array}\right]$



Spatial Markov property and intertwining

Markov property of the uniform infinite pyramid (UIP)

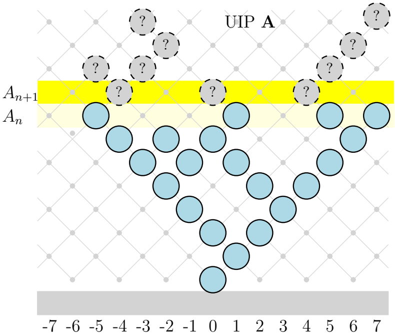
Theorem (Hénard, Maurel-Segala, S. (24))

The UIP is a Markov process when sliced layer by layer.

- *Particle system with product interaction between neighboring vertices:*

$$\eta(x_1, \ldots, x_n) = \prod_{i=1}^{n-1} (x_{i+1} - x_i - 1)$$

$$\mathbb{P}(A_{n+1} | A_n) = \frac{\eta(A_{n+1})}{3^{|A_n|} \eta(A_n)}$$



Markov property of the uniform infinite pyramid (UIP)

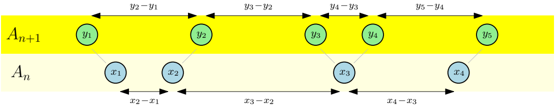
Theorem (Hénard, Maurel-Segala, S. (24))

The UIP is a Markov process when sliced layer by layer.

- *Particle system with product interaction between neighboring vertices:*

$$\eta(x_1, \dots, x_n) = \prod_{i=1}^{n-1} (x_{i+1} - x_i - 1)$$

$$\mathbb{P}(A_{n+1} | A_n) = \frac{\eta(A_{n+1})}{3^{|A_n|}\eta(A_n)}$$



$\eta(A_n) = (x_2 - x_1 - 1)(x_3 - x_2 - 1)(x_4 - x_3 - 1)$

$\eta(A_{n+1}) = (y_2 - y_1 - 1)(y_3 - y_2 - 1)(y_4 - y_3 - 1)(y_5 - y_4 - 1)$

$\longrightarrow \quad \mathbb{P}(A_{n+1} | A_n) = \frac{\eta(A_{n+1})}{3^4 \eta(A_n)}$

Markov property of the uniform infinite pyramid (UIP)

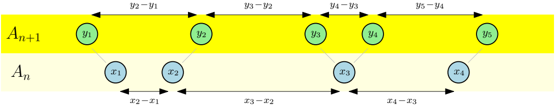
Theorem (Hénard, Maurel-Segala, S. (24))

The UIP is a Markov process when sliced layer by layer.

- *Particle system with product interaction between neighboring vertices:*

$$\eta(x_1, \ldots, x_n) = \prod_{i=1}^{n-1} (x_{i+1} - x_i - 1)$$

$$\mathbb{P}(A_{n+1} | A_n) = \frac{\eta(A_{n+1})}{3^{|A_n|} \eta(A_n)}$$



$$\eta(A_n) = (x_2 - x_1 - 1)(x_3 - x_2 - 1)(x_4 - x_3 - 1)$$

$$\eta(A_{n+1}) = (y_2 - y_1 - 1)(y_3 - y_2 - 1)(y_4 - y_3 - 1)(y_5 - y_4 - 1)$$

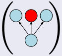
$$\longrightarrow \quad \mathbb{P}(A_{n+1} | A_n) = \frac{\eta(A_{n+1})}{3^4 \eta(A_n)}$$

- Similar results for the Boltzmann pyramid and half-pyramid.
- The kernel identity is non-trivial.
- "Long-range" interaction.

Branching-annihilating particle system

Consider a system with $\bigcirc$ and $\bullet$ particles such that:

1. Each particle (at i) reproduces independently, creating particles at $i-1, i, i+1$ s.t.

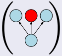
$\mathbb{P}\Big($  $\Big) = \mathbb{P}\Big($  $\Big) = \mathbb{P}\Big($  $\Big) = \frac{1}{3}$ and $\mathbb{P}\Big($  $\Big) = 1$

2. Particles of opposite colors annihilate when they collide.

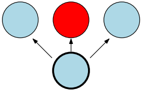
Branching-annihilating particle system

Consider a system with ◯ and ● particles such that:

1. Each particle (at i) reproduces independently, creating particles at $i-1, i, i+1$ s.t.

   $\mathbb{P}\Big($  $\Big) = \mathbb{P}\Big($  $\Big) = \mathbb{P}\Big($  $\Big) = \frac{1}{3}$ and $\mathbb{P}\Big($  $\Big) = 1$

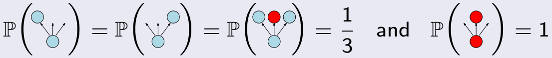
2. Particles of opposite colors annihilate when they collide.



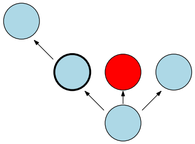
Branching-annihilating particle system

Consider a system with  and  particles such that:

1. Each particle (at i) reproduces independently, creating particles at $i-1, i, i+1$ s.t.



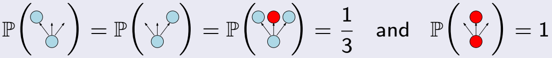
2. Particles of opposite colors annihilate when they collide.



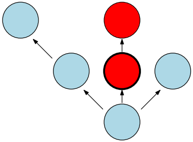
Branching-annihilating particle system

Consider a system with  and  particles such that:

1. Each particle (at i) reproduces independently, creating particles at $i-1, i, i+1$ s.t.



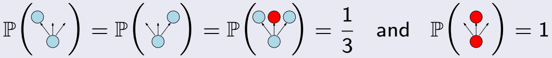
2. Particles of opposite colors annihilate when they collide.



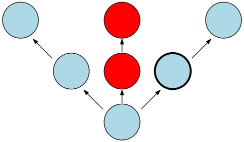
Branching-annihilating particle system

Consider a system with and particles such that:

1. Each particle (at i) reproduces independently, creating particles at $i-1, i, i+1$ s.t.



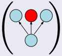
2. Particles of opposite colors annihilate when they collide.



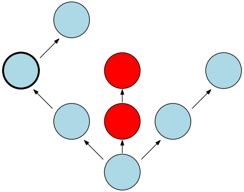
Branching-annihilating particle system

Consider a system with  and  particles such that:

1. Each particle (at i) reproduces independently, creating particles at $i-1, i, i+1$ s.t.

$\mathbb{P}\Big($  $\Big) = \mathbb{P}\Big($  $\Big) = \mathbb{P}\Big($  $\Big) = \frac{1}{3}$ and $\mathbb{P}\Big($  $\Big) = 1$

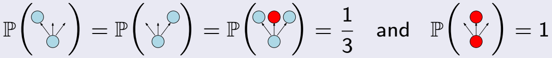
2. Particles of opposite colors annihilate when they collide.



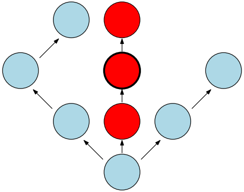
Branching-annihilating particle system

Consider a system with ○ and ● particles such that:

1. Each particle (at i) reproduces independently, creating particles at $i-1, i, i+1$ s.t.



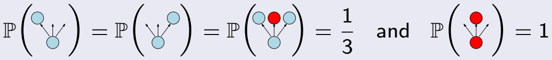
2. Particles of opposite colors annihilate when they collide.



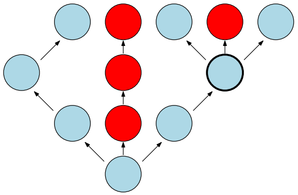
Branching-annihilating particle system

Consider a system with and particles such that:

1. Each particle (at i) reproduces independently, creating particles at $i-1, i, i+1$ s.t.



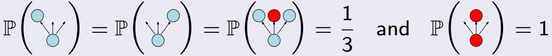
2. Particles of opposite colors annihilate when they collide.



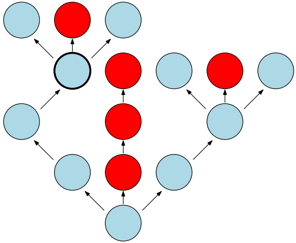
Branching-annihilating particle system

Consider a system with ○ and ● particles such that:

1. Each particle (at i) reproduces independently, creating particles at $i-1, i, i+1$ s.t.



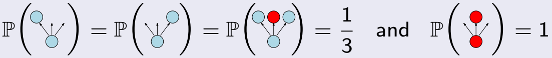
2. Particles of opposite colors annihilate when they collide.



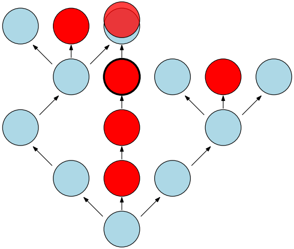
Branching-annihilating particle system

Consider a system with and particles such that:

1. Each particle (at i) reproduces independently, creating particles at $i-1, i, i+1$ s.t.



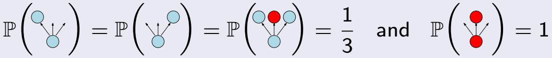
2. Particles of opposite colors annihilate when they collide.



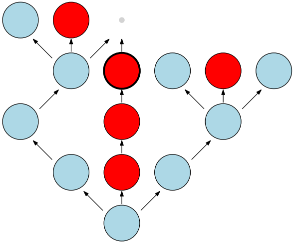
Branching-annihilating particle system

Consider a system with ◯ and ● particles such that:

1. Each particle (at i) reproduces independently, creating particles at $i-1, i, i+1$ s.t.



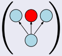
2. Particles of opposite colors annihilate when they collide.



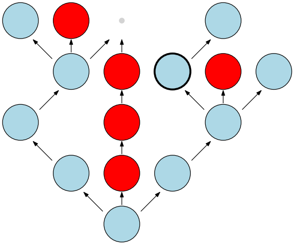
Branching-annihilating particle system

Consider a system with light-blue and red particles such that:

1. Each particle (at i) reproduces independently, creating particles at $i-1, i, i+1$ s.t.

$\mathbb{P}\Big($  $\Big) = \mathbb{P}\Big($  $\Big) = \mathbb{P}\Big($  $\Big) = \frac{1}{3}$ and $\mathbb{P}\Big($  $\Big) = 1$

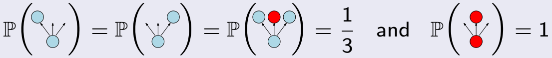
2. Particles of opposite colors annihilate when they collide.



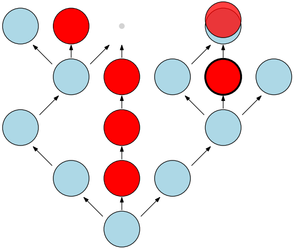
Branching-annihilating particle system

Consider a system with ○ and ● particles such that:

1. Each particle (at i) reproduces independently, creating particles at $i-1, i, i+1$ s.t.



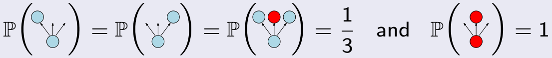
2. Particles of opposite colors annihilate when they collide.



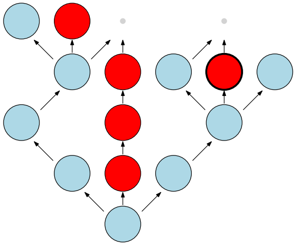
Branching-annihilating particle system

Consider a system with and particles such that:

1. Each particle (at i) reproduces independently, creating particles at $i-1, i, i+1$ s.t.



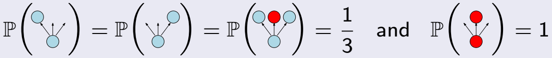
2. Particles of opposite colors annihilate when they collide.



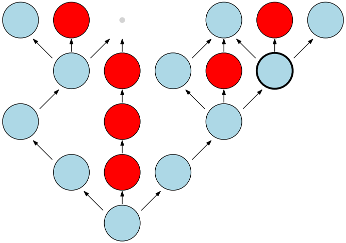
Branching-annihilating particle system

Consider a system with and particles such that:

1. Each particle (at i) reproduces independently, creating particles at $i-1, i, i+1$ s.t.



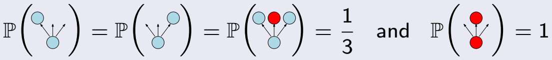
2. Particles of opposite colors annihilate when they collide.



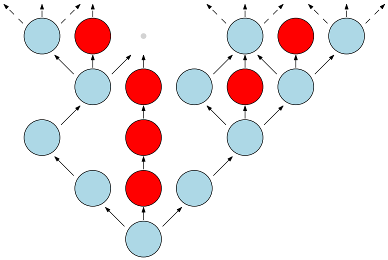
Branching-annihilating particle system

Consider a system with ○ and ● particles such that:

1. Each particle (at i) reproduces independently, creating particles at $i-1, i, i+1$ s.t.



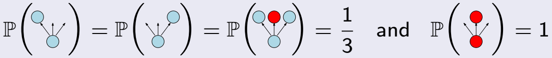
2. Particles of opposite colors annihilate when they collide.



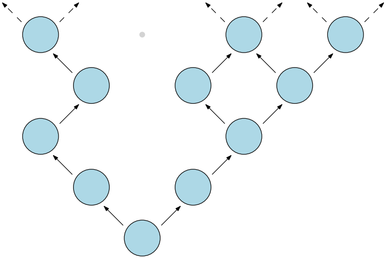
Branching-annihilating particle system

Consider a system with ○ and ● particles such that:

1. Each particle (at i) reproduces independently, creating particles at $i-1, i, i+1$ s.t.



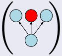
2. Particles of opposite colors annihilate when they collide.



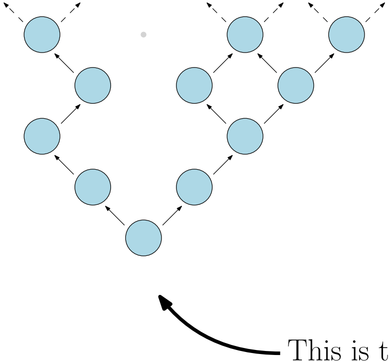
Branching-annihilating particle system

Consider a system with  and  particles such that:

1. Each particle (at i) reproduces independently, creating particles at $i-1, i, i+1$ s.t.

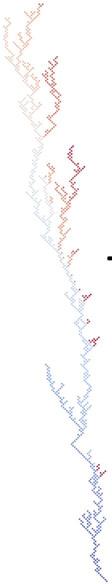
$\mathbb{P}\Big($  $\Big) = \mathbb{P}\Big($  $\Big) = \mathbb{P}\Big($  $\Big) = \frac{1}{3}$ and $\mathbb{P}\Big($  $\Big) = 1$

2. Particles of opposite colors annihilate when they collide.

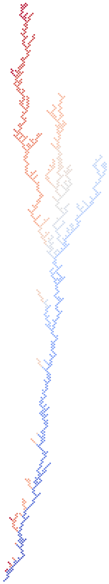


- function of a Markov process is Markov
  $\implies$ Intertwining of kernels.
  $\implies$ Intertwining of red/blue particles.
- The red particle can also move!
- The long-range interaction between blue vertices is mediated by the "invisible red particles".
- Reminiscent of
  - Dyson's Brownian motion
  - Pitman's theorem.

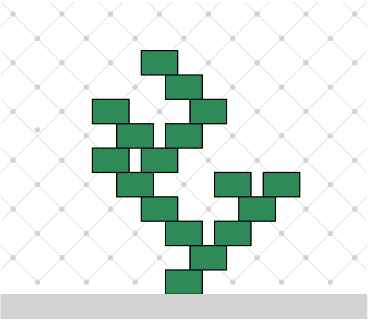
This is the UIP !



Thank you for your attention!

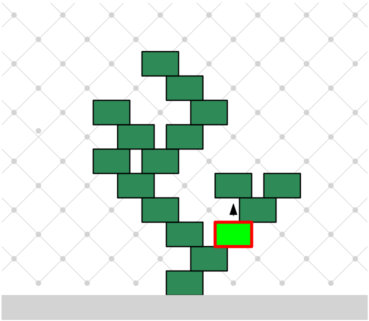


Viennot's heap of pieces : the "push-up" operation



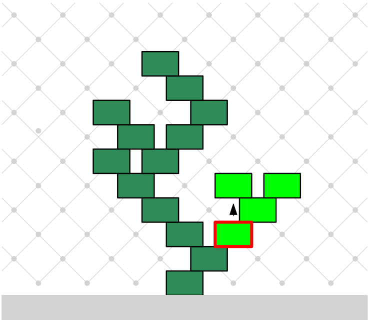
Viennot's heap of pieces : the "push-up" operation

- Lifting up a domino: bring along the pyramid sitting over it.



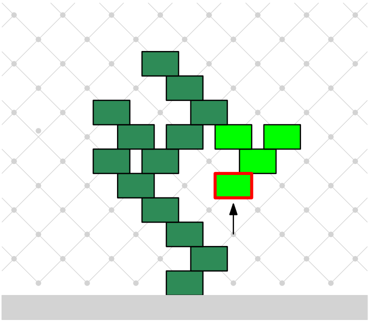
Viennot's heap of pieces : the "push-up" operation

- Lifting up a domino: bring along the pyramid sitting over it.



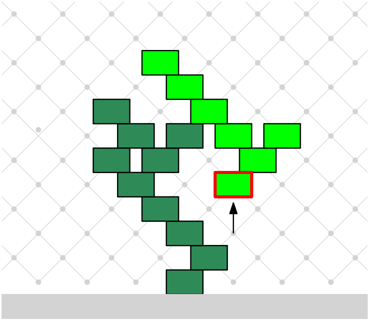
Viennot's heap of pieces : the "push-up" operation

- Lifting up a domino: bring along the pyramid sitting over it.



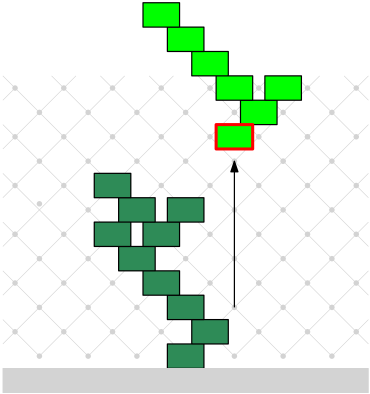
Viennot's heap of pieces : the "push-up" operation

- Lifting up a domino: bring along the pyramid sitting over it.



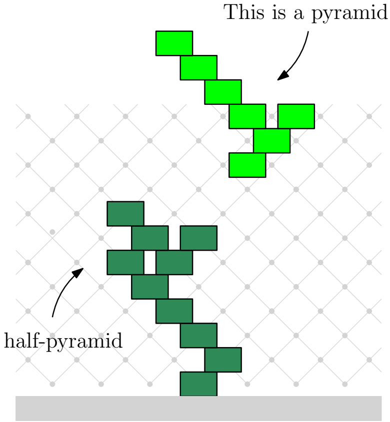
Viennot's heap of pieces : the "push-up" operation

- Lifting up a domino: bring along the pyramid sitting over it.



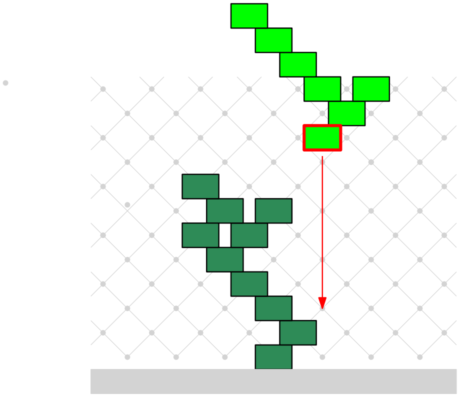
Viennot's heap of pieces : the "push-up" operation

- Lifting up a domino: bring along the pyramid sitting over it.



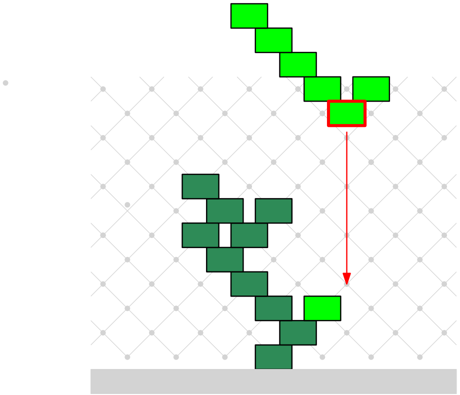
Viennot's heap of pieces : the "push-up" operation

- Lifting up a domino: bring along the pyramid sitting over it.
- Reverse operation: let domino drop "from infinity".



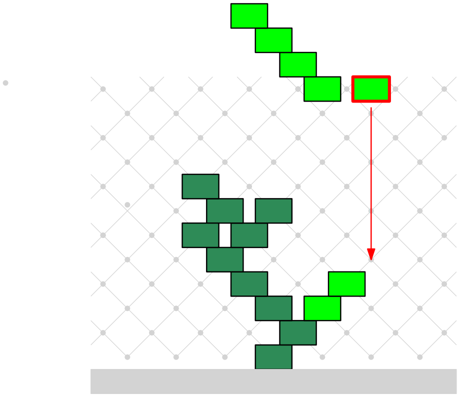
Viennot's heap of pieces : the "push-up" operation

- Lifting up a domino: bring along the pyramid sitting over it.
- Reverse operation: let domino drop "from infinity".



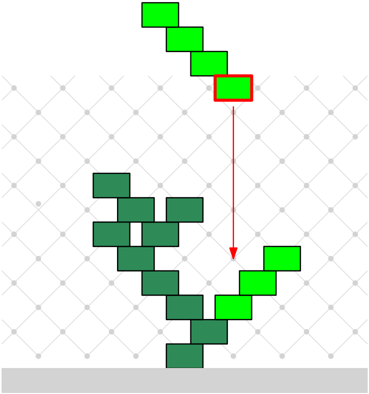
Viennot's heap of pieces : the "push-up" operation

- Lifting up a domino: bring along the pyramid sitting over it.
- Reverse operation: let domino drop "from infinity".



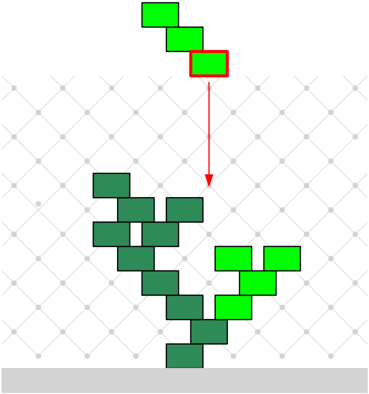
Viennot's heap of pieces : the "push-up" operation

- Lifting up a domino: bring along the pyramid sitting over it.
- Reverse operation: let domino drop "from infinity".



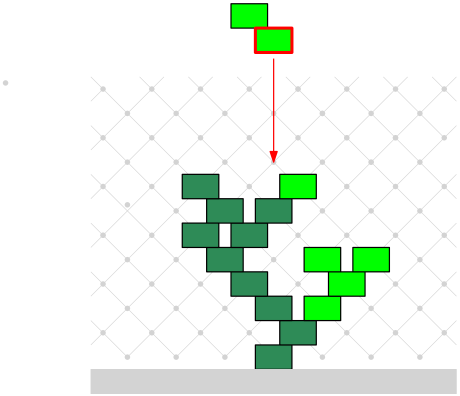
Viennot's heap of pieces : the "push-up" operation

- Lifting up a domino: bring along the pyramid sitting over it.
- Reverse operation: let domino drop "from infinity".



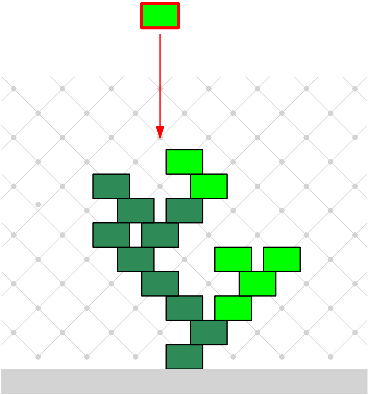
Viennot's heap of pieces : the "push-up" operation

- Lifting up a domino: bring along the pyramid sitting over it.
- Reverse operation: let domino drop "from infinity".



Viennot's heap of pieces : the "push-up" operation

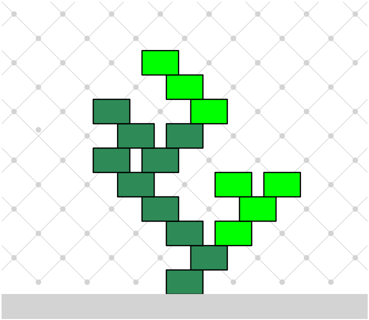
- Lifting up a domino: bring along the pyramid sitting over it.
- Reverse operation: let domino drop "from infinity".



Viennot's heap of pieces : the "push-up" operation

- Lifting up a domino: bring along the pyramid sitting over it.
- Reverse operation: let domino drop "from infinity".

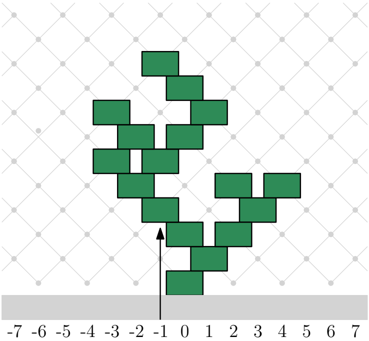
Pyramid / half-pyramid decomposition



Viennot's heap of pieces : the "push-up" operation

- Lifting up a domino: bring along the pyramid sitting over it.
- Reverse operation: let domino drop "from infinity".

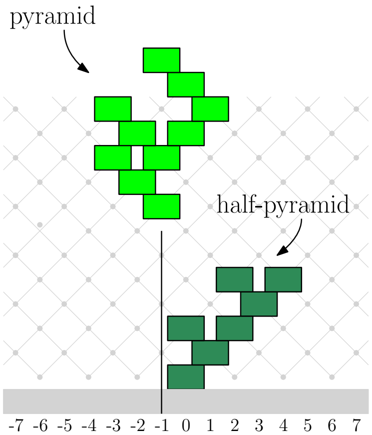
Pyramid / half-pyramid decomposition



Viennot's heap of pieces : the "push-up" operation

- Lifting up a domino: bring along the pyramid sitting over it.
- Reverse operation: let domino drop "from infinity".

Pyramid / half-pyramid decomposition

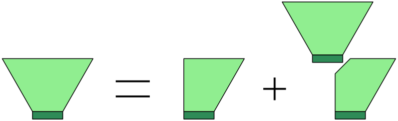


Generating series

$$\mathcal{P}(z) = \sum_{\text{pyramids } P} z^{|P|} \qquad \text{et} \qquad \mathcal{H}(z) = \sum_{\text{half-pyramids } H} z^{|H|}$$

Generating series

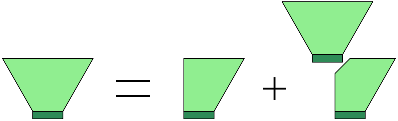
$$\mathcal{P}(z) = \sum_{\text{pyramids } P} z^{|P|} \qquad \text{et} \qquad \mathcal{H}(z) = \sum_{\text{half-pyramids } H} z^{|H|}$$



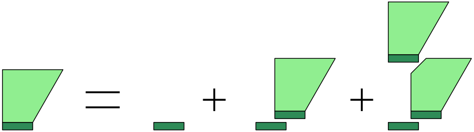
$$\mathcal{P} = \mathcal{H} + \mathcal{P}\mathcal{H}$$

Generating series

$$\mathcal{P}(z) = \sum_{\text{pyramids } P} z^{|P|} \qquad \text{et} \qquad \mathcal{H}(z) = \sum_{\text{half-pyramids } H} z^{|H|}$$



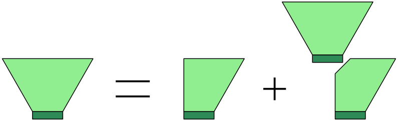
$$\mathcal{P} = \mathcal{H} + \mathcal{P}\mathcal{H}$$



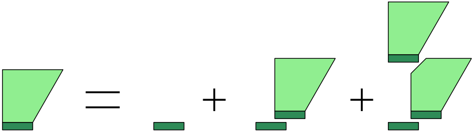
$$\mathcal{H} = z + z\mathcal{H} + z\mathcal{H}^2$$

Generating series

$$\mathcal{P}(z) = \sum_{\text{pyramids } P} z^{|P|} \qquad \text{et} \qquad \mathcal{H}(z) = \sum_{\text{half-pyramids } H} z^{|H|}$$



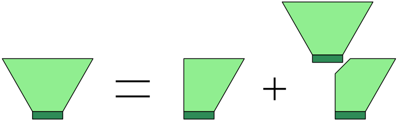
$$\mathcal{P}(z) = \frac{1}{2}\left(\frac{1+z}{1-3z} - 1\right)$$



$$\mathcal{H}(z) = \frac{1-z-\sqrt{(1+z)(1-3z)}}{2z}$$

Generating series

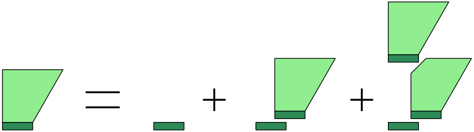
$$\mathcal{P}(z) = \sum_{\text{pyramids } P} z^{|P|} \qquad \text{et} \qquad \mathcal{H}(z) = \sum_{\text{half-pyramids } H} z^{|H|}$$



OEIS A005773

$$\mathcal{P}(z) = \frac{1}{2}\left(\frac{1+z}{1-3z} - 1\right)$$

$$[\mathcal{P}]_n \sim C\frac{3^n}{\sqrt{n}}$$



OEIS A001006 (Motzkin)

$$\mathcal{H}(z) = \frac{1-z-\sqrt{(1+z)(1-3z)}}{2z}$$

$$[\mathcal{H}]_n \sim C'\frac{3^n}{n\sqrt{n}}$$

- We can count directed animals according to their size... **Now we want to consider infinite directed animals !**