An automaton describing left burnable configurations in the sandpile model on the ladder graph

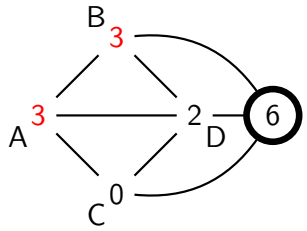
Henri Derycke

LaBRI

ICGT 2018, July 9-13, Lyon

université de BORDEAUX

The sandpile model

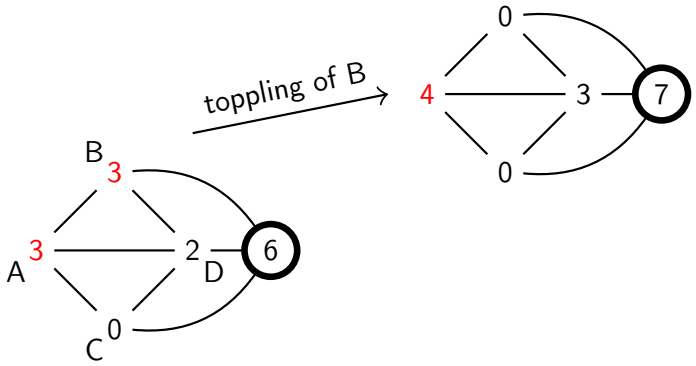


The sandpile model

$G = (V, E)$ simple connected graph with a special vertex $S \in V$ called the *sink*

- Configuration : assignment $u : V \to \mathbf{N}$ of grains on the vertices
- A vertex distinct from the sink is *unstable* if $u(x) \geq \deg(x)$
- Toppling : a vertex sends one grain to each of its neighbors
- Stabilization : While there is an unstable vertex, we topple it. Given u, we note $\mathrm{stab}(u)$ the result.

The sandpile model

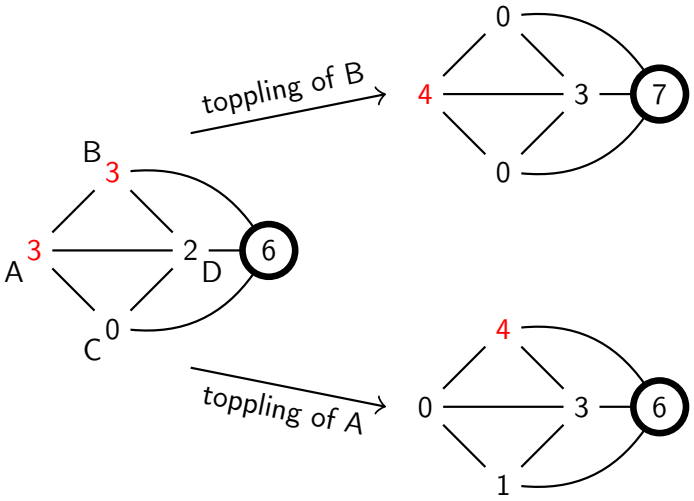


The sandpile model

$G = (V, E)$ simple connected graph with a special vertex $S \in V$ called the *sink*

- Configuration : assignment $u : V \to \mathbf{N}$ of grains on the vertices
- A vertex distinct from the sink is *unstable* if $u(x) \geq \deg(x)$
- Toppling : a vertex sends one grain to each of its neighbors
- Stabilization : While there is an unstable vertex, we topple it. Given u, we note $\mathrm{stab}(u)$ the result.

The sandpile model

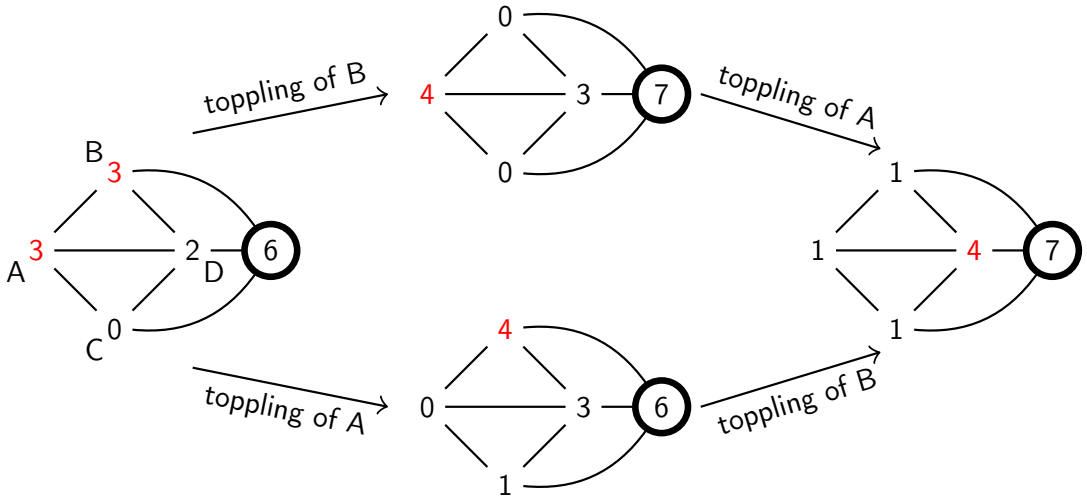


The sandpile model

$G = (V, E)$ simple connected graph with a special vertex $S \in V$ called the *sink*

- Configuration : assignment $u : V \to \mathbf{N}$ of grains on the vertices
- A vertex distinct from the sink is *unstable* if $u(x) \geq \deg(x)$
- Toppling : a vertex sends one grain to each of its neighbors
- Stabilization : While there is an unstable vertex, we topple it. Given u, we note $\mathrm{stab}(u)$ the result.

The sandpile model

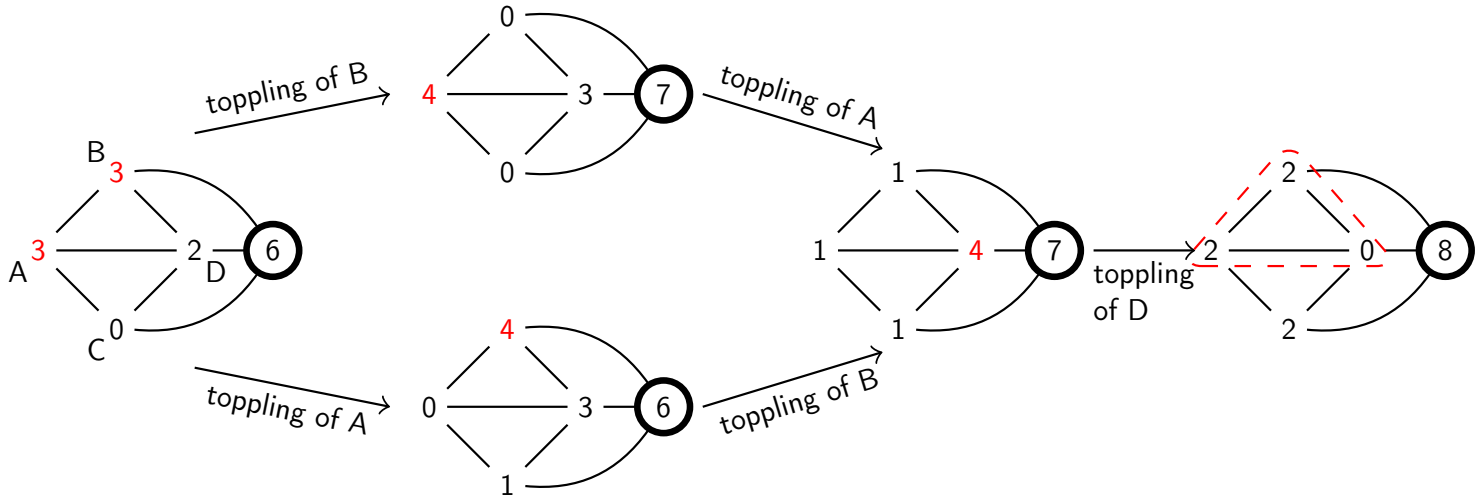


The sandpile model

$G = (V, E)$ simple connected graph with a special vertex $S \in V$ called the *sink*

- Configuration : assignment $u : V \to \mathbf{N}$ of grains on the vertices
- A vertex distinct from the sink is *unstable* if $u(x) \geq \deg(x)$
- Toppling : a vertex sends one grain to each of its neighbors
- Stabilization : While there is an unstable vertex, we topple it. Given u, we note $\mathrm{stab}(u)$ the result.

The sandpile model



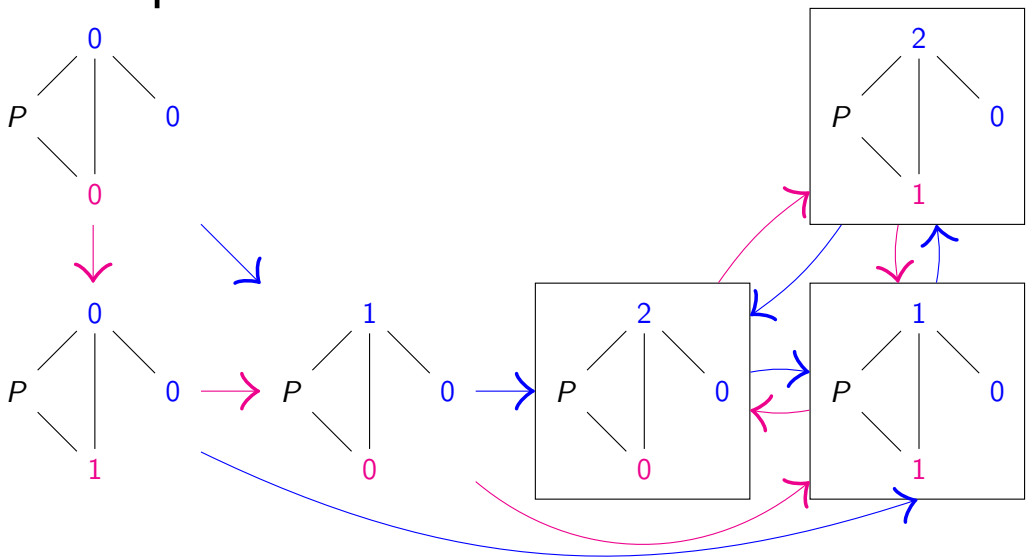
The sandpile model

$G = (V, E)$ simple connected graph with a special vertex $S \in V$ called the *sink*

- Configuration : assignment $u : V \to \mathbf{N}$ of grains on the vertices
- A vertex distinct from the sink is *unstable* if $u(x) \geq \deg(x)$
- Toppling : a vertex sends one grain to each of its neighbors
- Stabilization : While there is an unstable vertex, we topple it. Given u, we note stab(u) the result.

Markov Chain for $G = (V \cup \{S\}, E)$

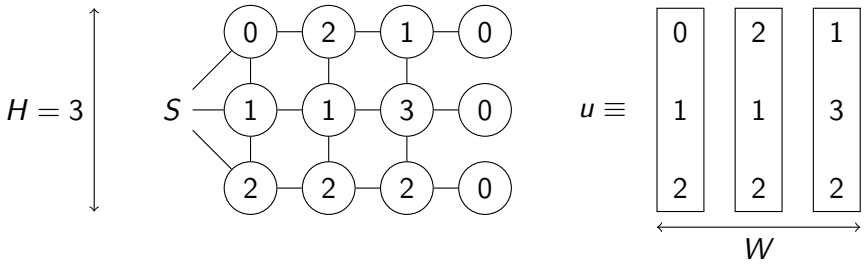
- States : stable configurations on G
- Transition : Add a particle to a vertex chosen uniformly and stabilize



- The recurrent states are called recurrent configurations.
- The stationary distribution is uniform on the recurrent configurations.

Dhar Criterion A stable configuration is recurrent if and only if adding a grain to each neighbor of the sink, and stabilizing result to the same configuration. (fixed point)

Configurations on left rooted square grid graphs

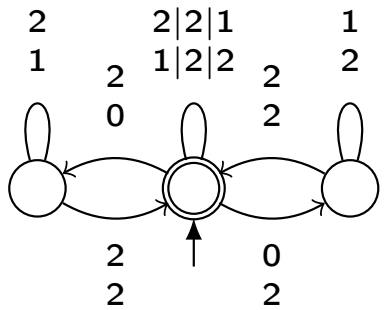


A configuration u may be read as a word of size W on the alphabet of columns.

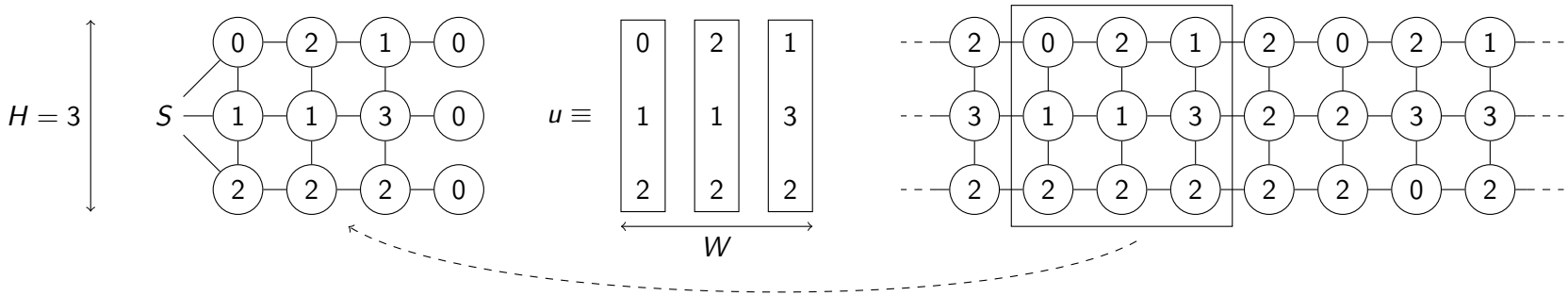
Járai, Lyons 07

The recurrent configurations of the left-rooted square grid graphs are recognized by a finite automaton on the columns.

For height 2, they found the following automaton.



Configurations on left rooted square grid graphs

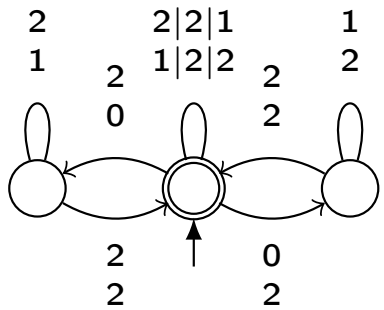


A configuration u may be read as a word of size W on the alphabet of columns.

Járai, Lyons 07

The recurrent configurations of the left-rooted square grid graphs are recognized by a finite automaton on the columns.

For height 2, they found the following automaton.



Some results on these automata of height $H \geq 2$

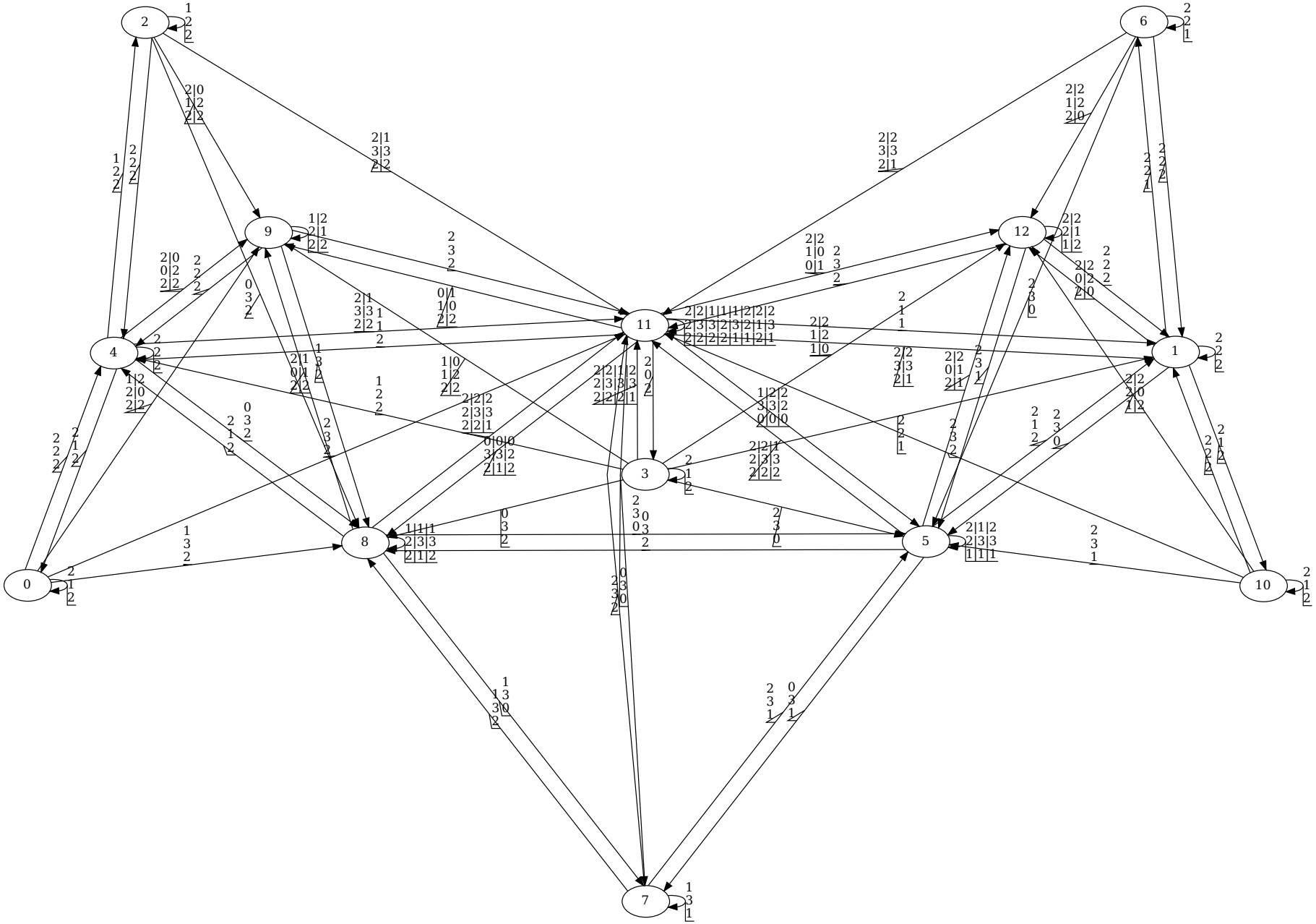
From Járai and Lyons, the number of states is roughly bounded by $(2^H)^{2^H}$.

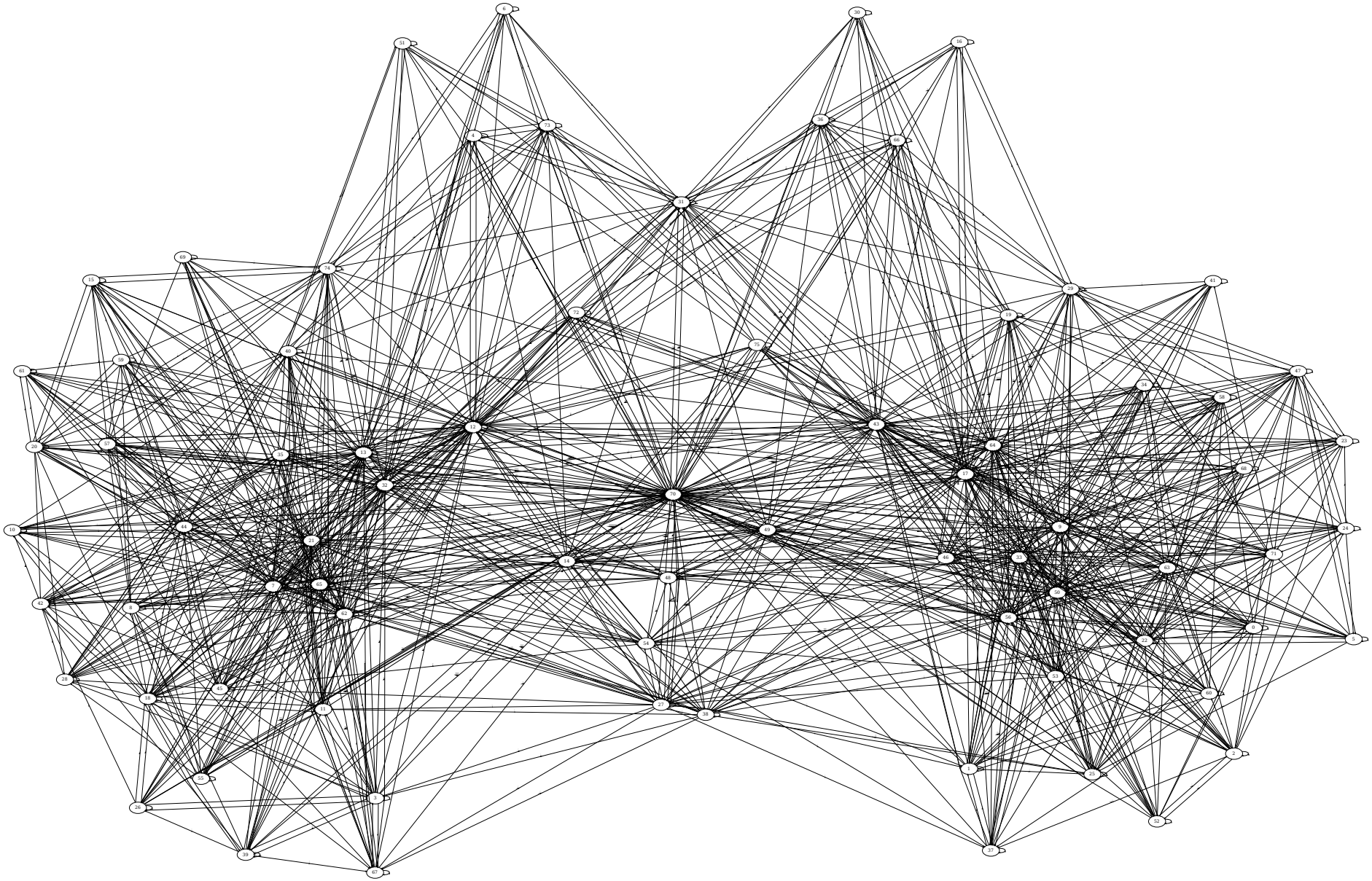
Conjecture from Gamlin (Thesis 2015) $\exp cH$ states may be enough.

Similar to an automaton for spanning trees (eg. dominoes tiling) but optimistic on the non local transformation to recurrent configurations.

My results

- Efficient construction of an automaton.
- From α^H to $\beta^{H \log H}$ states.
- The lower bound is from a bijection between a subset of the states and separable permutations (avoiding patterns 2413 and 3142).
- Explicit automata for heights 3 and 4


2
6
9
12
4
11
1
3
8
5
0
10
7



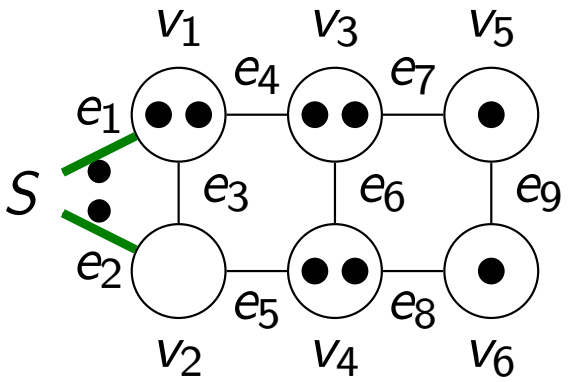
Bijections between recurrent configurations and spanning trees

Some bijections : Dhar/Majumdar 92, Cori/Le Borgne 03, Bernardi 06...

We choose an order on the edges : from left to right and from top to bottom.

Bijection CLB : (from a recurrent configuration)

Decreasing traversals : $e_1, v_1, e_2, e_3, e_4, v_3, e_6, v_4, e_5, v_2, e_7, v_5, e_8, v_6, e_9$



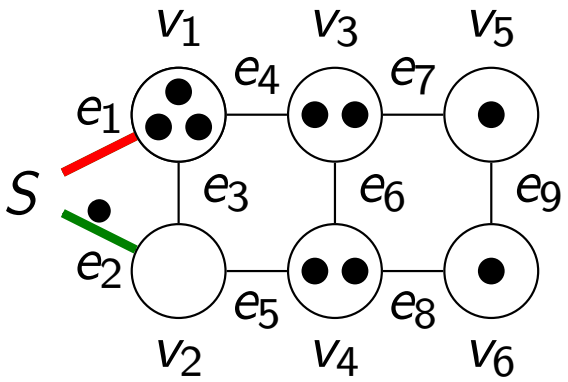

```
Mark edges incident to the sink as pending edges.
While there is a pending edge
  Get the smallest pending edge
  Process the grain(s) on the edge
  If a vertex become unstable, topple it and mark its
    untreated incident edges as pending edges.
```

Bijections between recurrent configurations and spanning trees

Some bijections : Dhar/Majumdar 92, Cori/Le Borgne 03, Bernardi 06...

We choose an order on the edges : from left to right and from top to bottom.

Bijection CLB : (from a recurrent configuration)
Decreasing traversals : e_1,



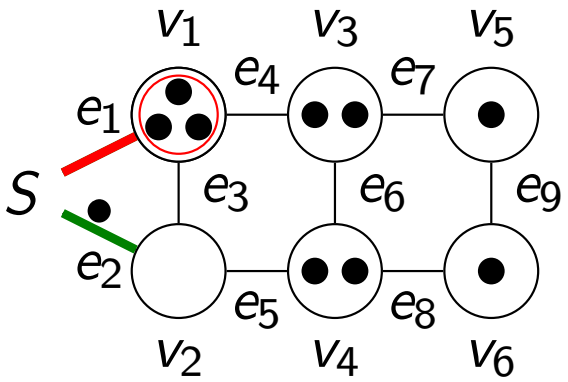

```
Mark edges incident to the sink as pending edges.
While there is a pending edge
  Get the smallest pending edge
  Process the grain(s) on the edge
  If a vertex become unstable, topple it and mark its
    untreated incident edges as pending edges.
```

Bijections between recurrent configurations and spanning trees

Some bijections : Dhar/Majumdar 92, Cori/Le Borgne 03, Bernardi 06...

We choose an order on the edges : from left to right and from top to bottom.

Bijection CLB : (from a recurrent configuration)
Decreasing traversals : e_1,



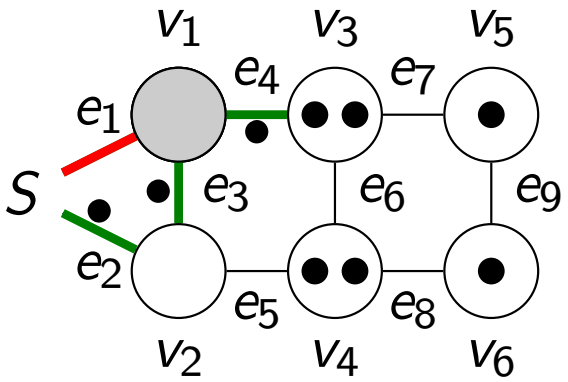

```
Mark edges incident to the sink as pending edges.
While there is a pending edge
  Get the smallest pending edge
  Process the grain(s) on the edge
  If a vertex become unstable, topple it and mark its
    untreated incident edges as pending edges.
```

Bijections between recurrent configurations and spanning trees

Some bijections : Dhar/Majumdar 92, Cori/Le Borgne 03, Bernardi 06...

We choose an order on the edges : from left to right and from top to bottom.

Bijection CLB : (from a recurrent configuration)
Decreasing traversals : $e_1, v_1,$



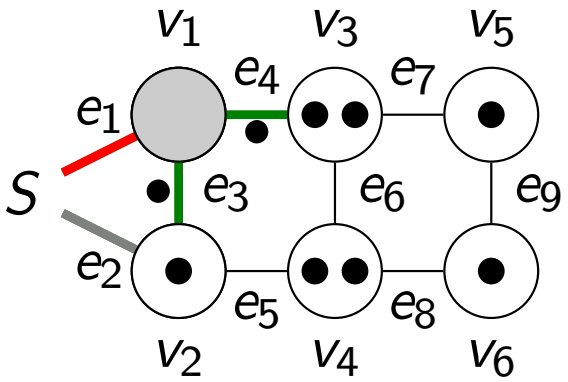

```
Mark edges incident to the sink as pending edges.
While there is a pending edge
  Get the smallest pending edge
  Process the grain(s) on the edge
  If a vertex become unstable, topple it and mark its
    untreated incident edges as pending edges.
```

Bijections between recurrent configurations and spanning trees

Some bijections : Dhar/Majumdar 92, Cori/Le Borgne 03, Bernardi 06...

We choose an order on the edges : from left to right and from top to bottom.

Bijection CLB : (from a recurrent configuration)
Decreasing traversals : $e_1, v_1, e_2,$



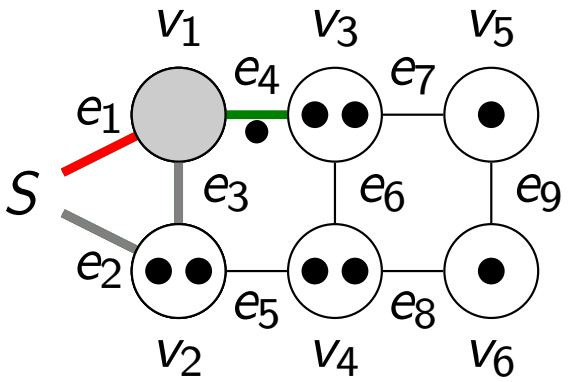

```
Mark edges incident to the sink as pending edges.
While there is a pending edge
  Get the smallest pending edge
  Process the grain(s) on the edge
  If a vertex become unstable, topple it and mark its
    untreated incident edges as pending edges.
```

Bijections between recurrent configurations and spanning trees

Some bijections : Dhar/Majumdar 92, Cori/Le Borgne 03, Bernardi 06...

We choose an order on the edges : from left to right and from top to bottom.

Bijection CLB : (from a recurrent configuration)
Decreasing traversals : $e_1, v_1, e_2, e_3,$



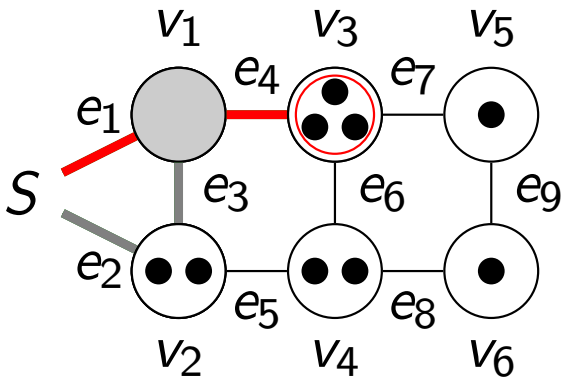

```
Mark edges incident to the sink as pending edges.
While there is a pending edge
  Get the smallest pending edge
  Process the grain(s) on the edge
  If a vertex become unstable, topple it and mark its
    untreated incident edges as pending edges.
```

Bijections between recurrent configurations and spanning trees

Some bijections : Dhar/Majumdar 92, Cori/Le Borgne 03, Bernardi 06...

We choose an order on the edges : from left to right and from top to bottom.

Bijection CLB : (from a recurrent configuration)
Decreasing traversals : $e_1, v_1, e_2, e_3, e_4,$



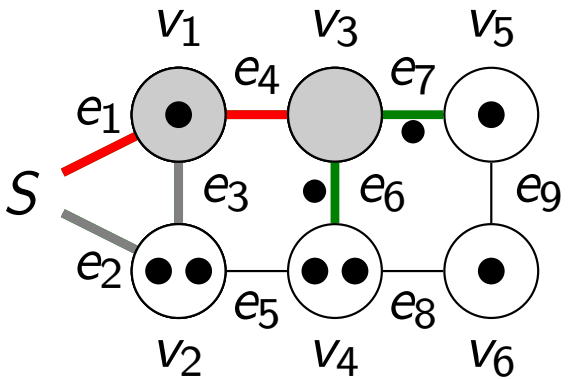

```
Mark edges incident to the sink as pending edges.
While there is a pending edge
  Get the smallest pending edge
  Process the grain(s) on the edge
  If a vertex become unstable, topple it and mark its
    untreated incident edges as pending edges.
```

Bijections between recurrent configurations and spanning trees

Some bijections : Dhar/Majumdar 92, Cori/Le Borgne 03, Bernardi 06...

We choose an order on the edges : from left to right and from top to bottom.

Bijection CLB : (from a recurrent configuration)
Decreasing traversals : $e_1, v_1, e_2, e_3, e_4, v_3,$



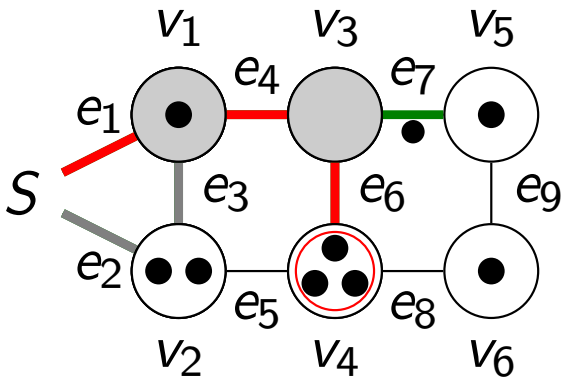

```
Mark edges incident to the sink as pending edges.
While there is a pending edge
  Get the smallest pending edge
  Process the grain(s) on the edge
  If a vertex become unstable, topple it and mark its
    untreated incident edges as pending edges.
```

Bijections between recurrent configurations and spanning trees

Some bijections : Dhar/Majumdar 92, Cori/Le Borgne 03, Bernardi 06...

We choose an order on the edges : from left to right and from top to bottom.

Bijection CLB : (from a recurrent configuration)
Decreasing traversals : $e_1, v_1, e_2, e_3, e_4, v_3, e_6,$



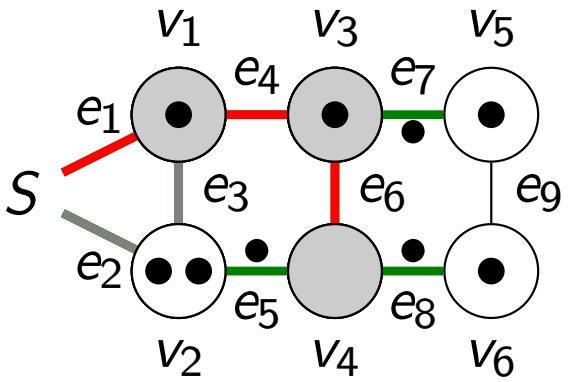

```
Mark edges incident to the sink as pending edges.
While there is a pending edge
  Get the smallest pending edge
  Process the grain(s) on the edge
  If a vertex become unstable, topple it and mark its
    untreated incident edges as pending edges.
```

Bijections between recurrent configurations and spanning trees

Some bijections : Dhar/Majumdar 92, Cori/Le Borgne 03, Bernardi 06...

We choose an order on the edges : from left to right and from top to bottom.

Bijection CLB : (from a recurrent configuration)
Decreasing traversals : $e_1, v_1, e_2, e_3, e_4, v_3, e_6, v_4,$



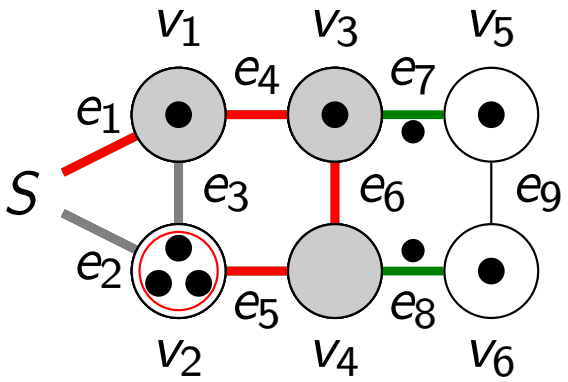

```
Mark edges incident to the sink as pending edges.
While there is a pending edge
  Get the smallest pending edge
  Process the grain(s) on the edge
  If a vertex become unstable, topple it and mark its
    untreated incident edges as pending edges.
```

Bijections between recurrent configurations and spanning trees

Some bijections : Dhar/Majumdar 92, Cori/Le Borgne 03, Bernardi 06...

We choose an order on the edges : from left to right and from top to bottom.

Bijection CLB : (from a recurrent configuration)
Decreasing traversals : $e_1, v_1, e_2, e_3, e_4, v_3, e_6, v_4, e_5,$



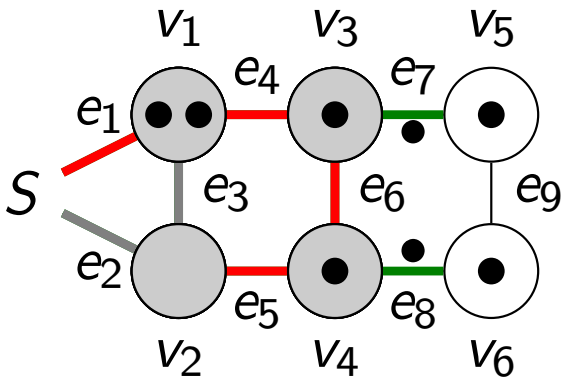

```
Mark edges incident to the sink as pending edges.
While there is a pending edge
  Get the smallest pending edge
  Process the grain(s) on the edge
  If a vertex become unstable, topple it and mark its
    untreated incident edges as pending edges.
```

Bijections between recurrent configurations and spanning trees

Some bijections : Dhar/Majumdar 92, Cori/Le Borgne 03, Bernardi 06...

We choose an order on the edges : from left to right and from top to bottom.

Bijection CLB : (from a recurrent configuration)
Decreasing traversals : $e_1, v_1, e_2, e_3, e_4, v_3, e_6, v_4, e_5, v_2,$




```
Mark edges incident to the sink as pending edges.
While there is a pending edge
  Get the smallest pending edge
  Process the grain(s) on the edge
  If a vertex become unstable, topple it and mark its
    untreated incident edges as pending edges.
```

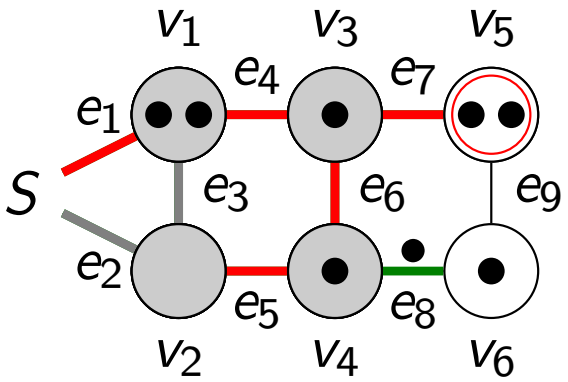
Bijections between recurrent configurations and spanning trees

Some bijections : Dhar/Majumdar 92, Cori/Le Borgne 03, Bernardi 06...

We choose an order on the edges : from left to right and from top to bottom.

Bijection CLB : (from a recurrent configuration)

Decreasing traversals : $e_1, v_1, e_2, e_3, e_4, v_3, e_6, v_4, e_5, v_2, e_7,$




```
Mark edges incident to the sink as pending edges.
While there is a pending edge
  Get the smallest pending edge
  Process the grain(s) on the edge
  If a vertex become unstable, topple it and mark its
    untreated incident edges as pending edges.
```

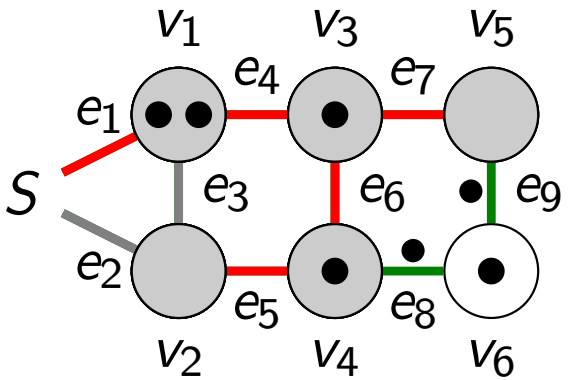
Bijections between recurrent configurations and spanning trees

Some bijections : Dhar/Majumdar 92, Cori/Le Borgne 03, Bernardi 06...

We choose an order on the edges : from left to right and from top to bottom.

Bijection CLB : (from a recurrent configuration)

Decreasing traversals : $e_1, v_1, e_2, e_3, e_4, v_3, e_6, v_4, e_5, v_2, e_7, v_5,$




```
Mark edges incident to the sink as pending edges.
While there is a pending edge
  Get the smallest pending edge
  Process the grain(s) on the edge
  If a vertex become unstable, topple it and mark its
    untreated incident edges as pending edges.
```

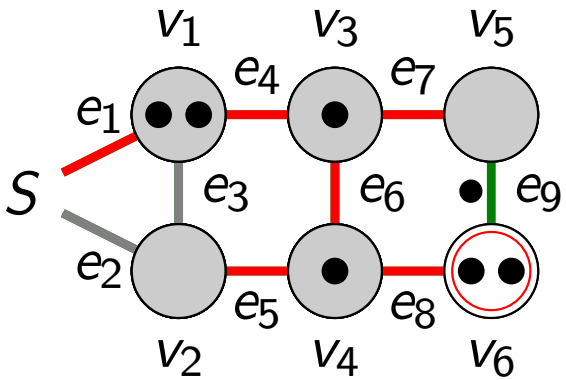
Bijections between recurrent configurations and spanning trees

Some bijections : Dhar/Majumdar 92, Cori/Le Borgne 03, Bernardi 06...

We choose an order on the edges : from left to right and from top to bottom.

Bijection CLB : (from a recurrent configuration)

Decreasing traversals : $e_1, v_1, e_2, e_3, e_4, v_3, e_6, v_4, e_5, v_2, e_7, v_5, e_8$,




```
Mark edges incident to the sink as pending edges.
While there is a pending edge
  Get the smallest pending edge
  Process the grain(s) on the edge
  If a vertex become unstable, topple it and mark its
    untreated incident edges as pending edges.
```

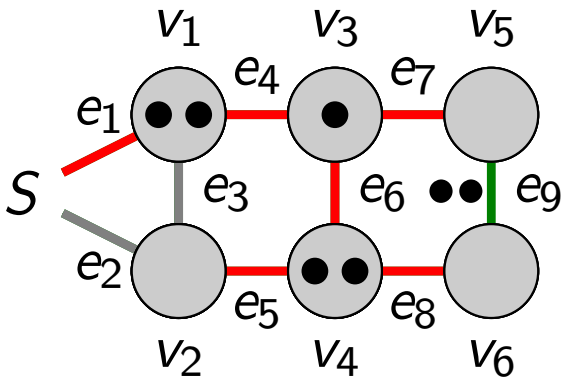
Bijections between recurrent configurations and spanning trees

Some bijections : Dhar/Majumdar 92, Cori/Le Borgne 03, Bernardi 06...

We choose an order on the edges : from left to right and from top to bottom.

Bijection CLB : (from a recurrent configuration)

Decreasing traversals : $e_1, v_1, e_2, e_3, e_4, v_3, e_6, v_4, e_5, v_2, e_7, v_5, e_8, v_6,$




```
Mark edges incident to the sink as pending edges.
While there is a pending edge
  Get the smallest pending edge
  Process the grain(s) on the edge
  If a vertex become unstable, topple it and mark its
    untreated incident edges as pending edges.
```

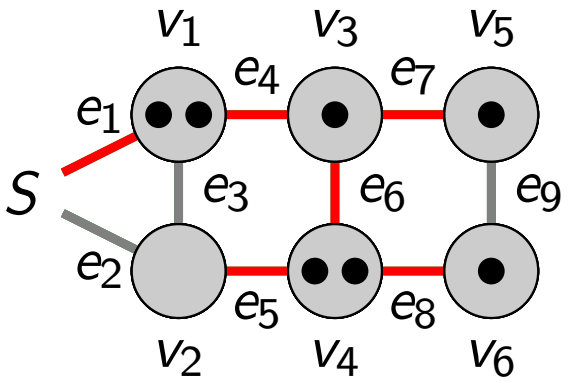
Bijections between recurrent configurations and spanning trees

Some bijections : Dhar/Majumdar 92, Cori/Le Borgne 03, Bernardi 06...

We choose an order on the edges : from left to right and from top to bottom.

Bijection CLB : (from a recurrent configuration)

Decreasing traversals : $e_1, v_1, e_2, e_3, e_4, v_3, e_6, v_4, e_5, v_2, e_7, v_5, e_8, v_6, e_9$




```
Mark edges incident to the sink as pending edges.
While there is a pending edge
  Get the smallest pending edge
  Process the grain(s) on the edge
  If a vertex become unstable, topple it and mark its
    untreated incident edges as pending edges.
```

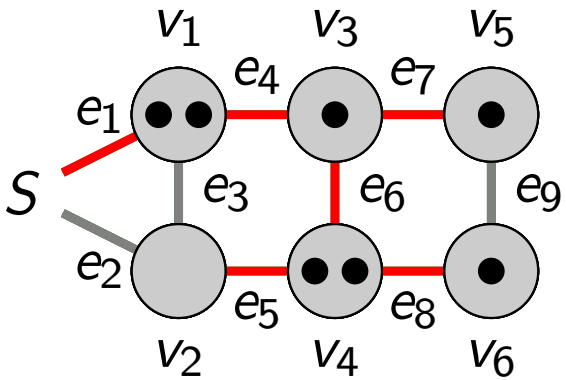
Bijections between recurrent configurations and spanning trees

Some bijections : Dhar/Majumdar 92, Cori/Le Borgne 03, Bernardi 06...

We choose an order on the edges : from left to right and from top to bottom.

Bijection CLB : (from a recurrent configuration)

Decreasing traversals : $e_1, v_1, e_2, e_3, e_4, v_3, e_6, v_4, e_5, v_2, e_7, v_5, e_8, v_6, e_9$




```
Mark edges incident to the sink as pending edges.
While there is a pending edge
  Get the smallest pending edge
  Process the grain(s) on the edge
  If a vertex become unstable, topple it and mark its
    untreated incident edges as pending edges.
```

- The spanning tree is encoded by the red edges that precede vertices in the decreasing edge-vertex traversal.
- For any vertex, the number of incident edges that are treated after the vertex is equal to the number of grains in the recurrent configuration.

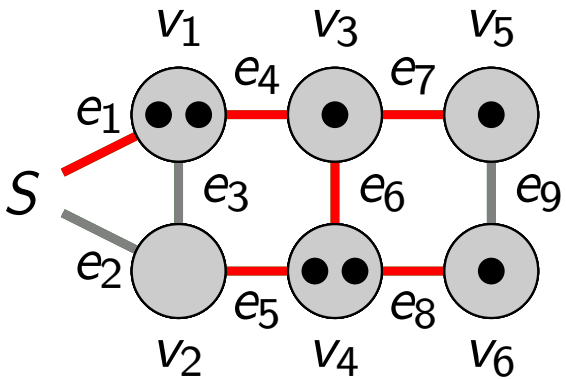
Bijections between recurrent configurations and spanning trees

Some bijections : Dhar/Majumdar 92, Cori/Le Borgne 03, Bernardi 06...

We choose an order on the edges : from left to right and from top to bottom.

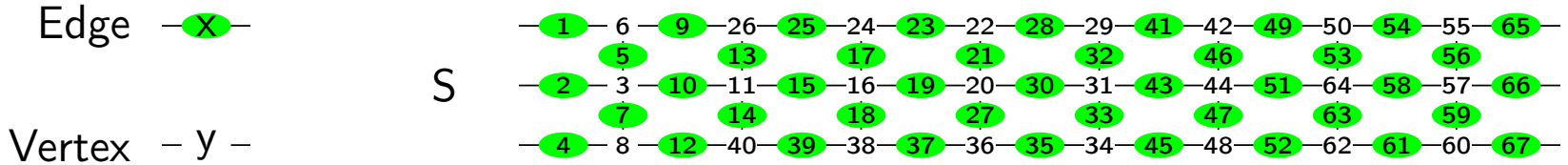
Bijection CLB : (from a recurrent configuration)

Decreasing traversals : $e_1, v_1, e_2, e_3, e_4, v_3, e_6, v_4, e_5, v_2, e_7, e_8, v_6, e_9, v_5$




```
Mark edges incident to the sink as pending edges.
While there is a pending edge
  Get the smallest pending edge
  Process the grain(s) on the edge
  If a vertex become unstable, topple it and mark its
    untreated incident edges as pending edges.
```

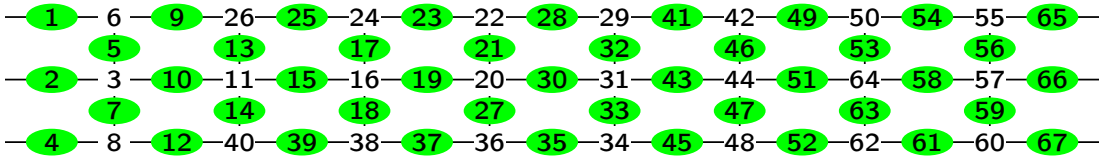
- The spanning tree is encoded by the red edges that precede vertices in the decreasing edge-vertex traversal.
- For any vertex, the number of incident edges that are treated after the vertex is equal to the number of grains in the recurrent configuration.


Edge
X
Vertex
y
S
1 6 9 26 25 24 23 22 28 29 41 42 49 50 54 55 65
5 13 17 21 32 46 53 56
2 3 10 11 15 16 19 20 30 31 43 44 51 64 58 57 66
7 14 18 27 33 47 63 59
4 8 12 40 39 38 37 36 35 34 45 48 52 62 61 60 67

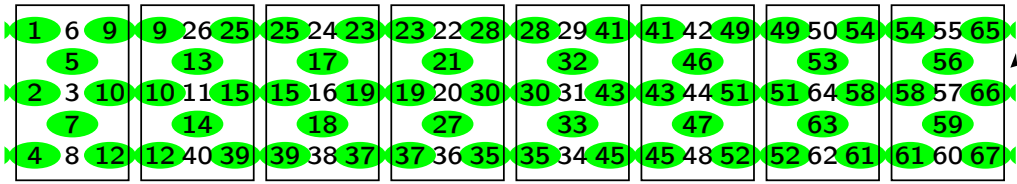
Edge – x –

Vertex – y –

S



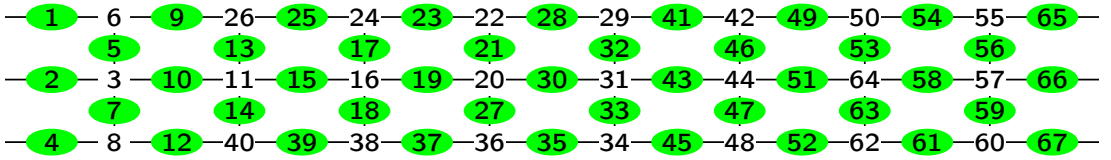
- Decomposition



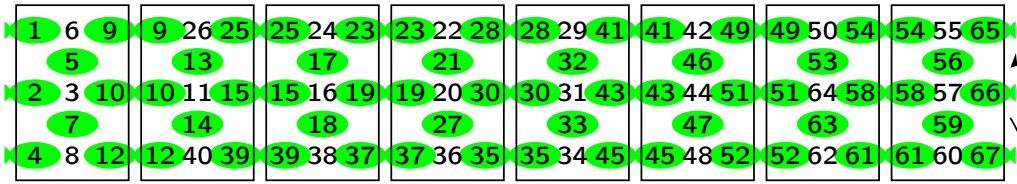
Edge – x –

Vertex – y –

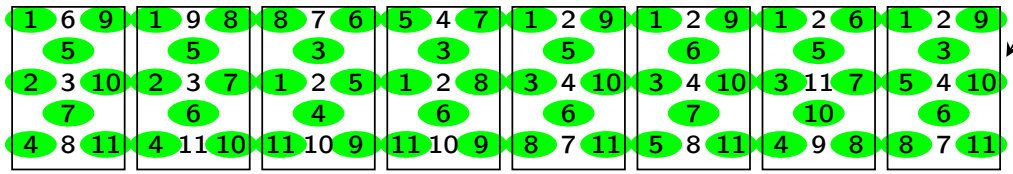
S

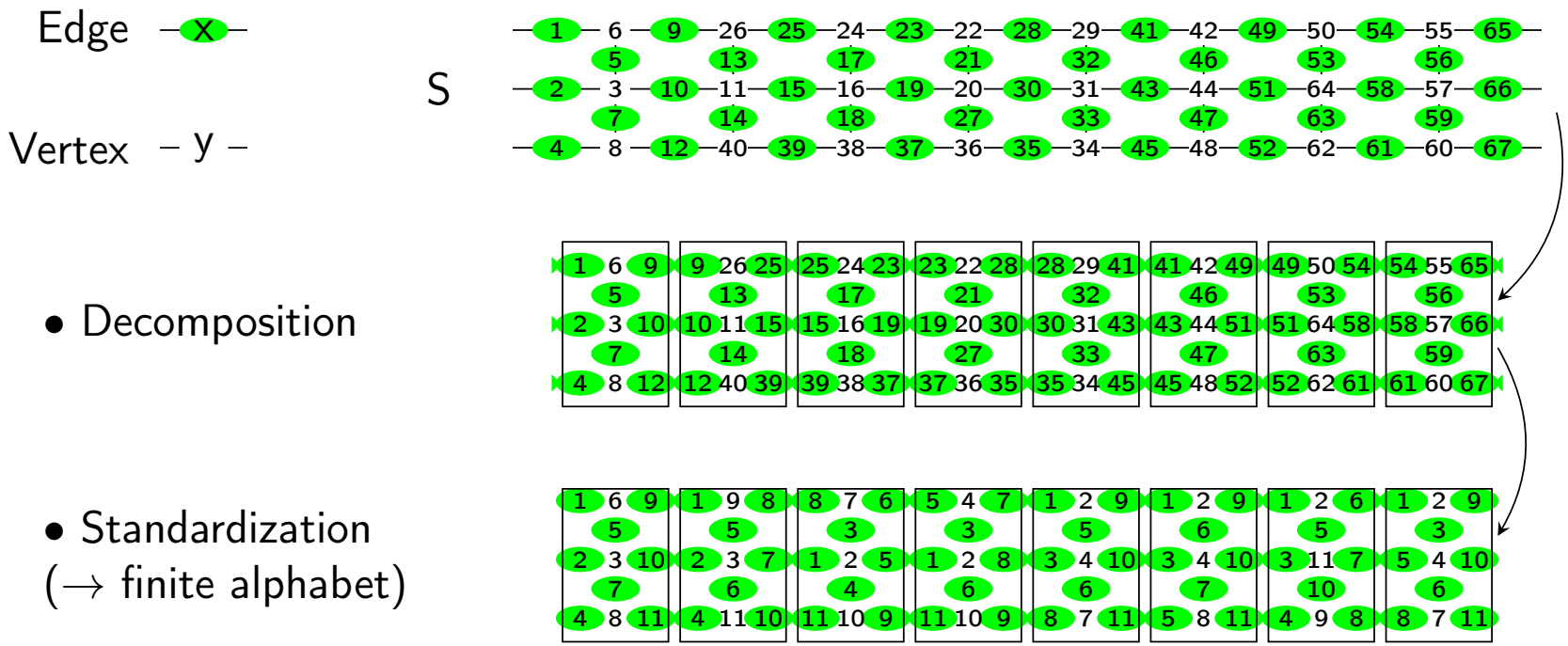


- Decomposition



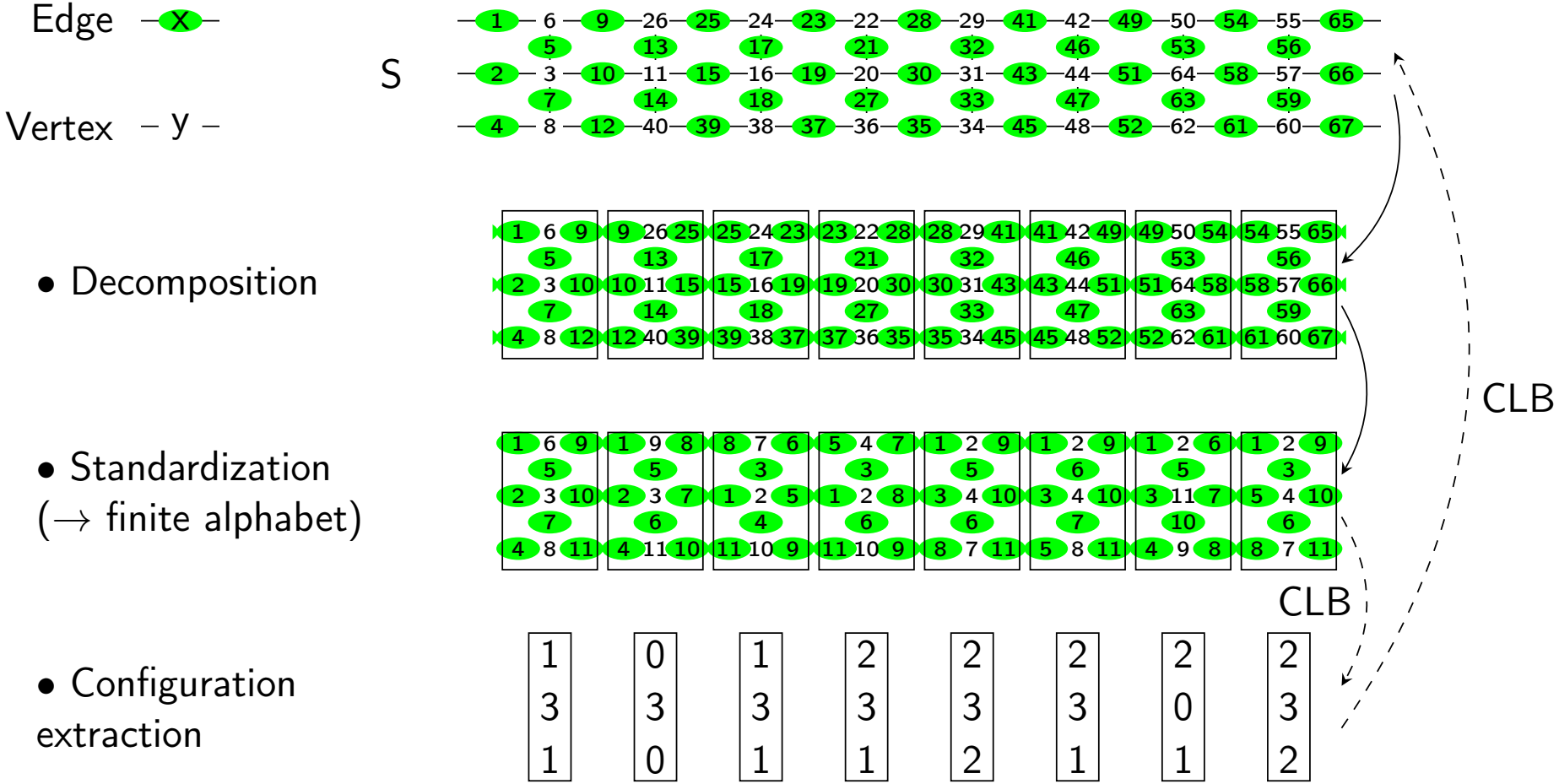
- Standardization
($\rightarrow$ finite alphabet)





Proposition

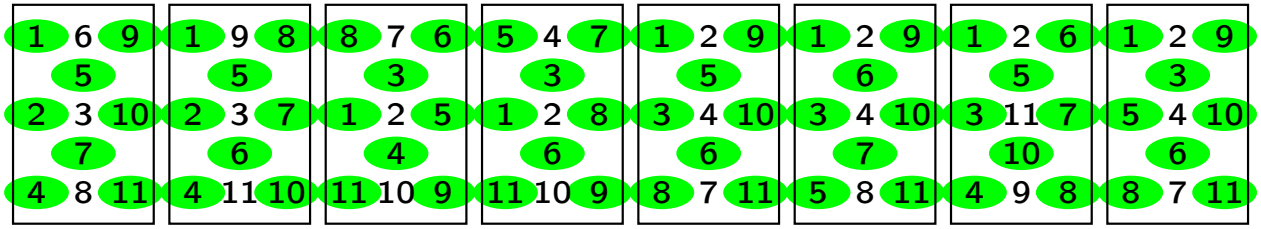
The edge-vertex decreasing traversals are in bijection with their standardized decompositions (in induced orders).



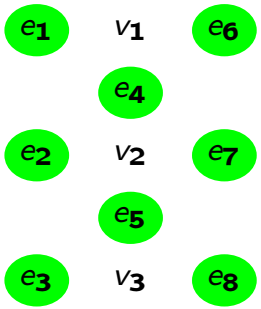
- Decomposition
- Standardization ($\rightarrow$ finite alphabet)
- Configuration extraction

Proposition

The edge-vertex decreasing traversals are in bijection with their standardized decompositions (in induced orders).



A decomposition is a word on the alphabet of induced orders.



- Initial induced order : $e_1(v_1?)e_2(v_2?)e_3(v_3?)\dots$
- Sequence of consecutive pairwise compatible induced orders : (o, o') verify $T(o, o')$
- Final induced order : $\dots e_6 e_7 e_8$

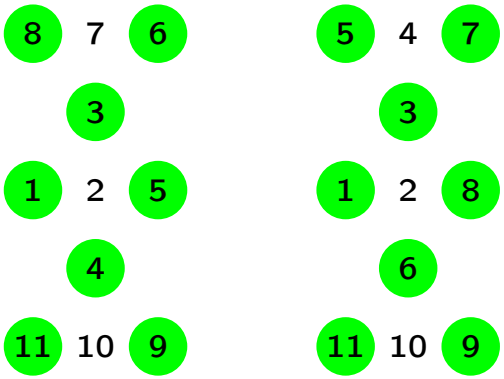
Automaton on induced orders

- States : induced orders
- Alphabet : induced orders
- Transition : $o \xrightarrow{o'} o'$ iff o' is right-compatible to o

Prop $\sharp stats \leq (4H - 1)!$

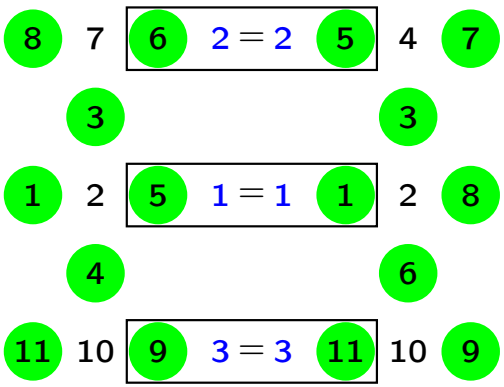
We have an automaton recognizing recurrent configurations

Right compatibility relation



Necessary conditions

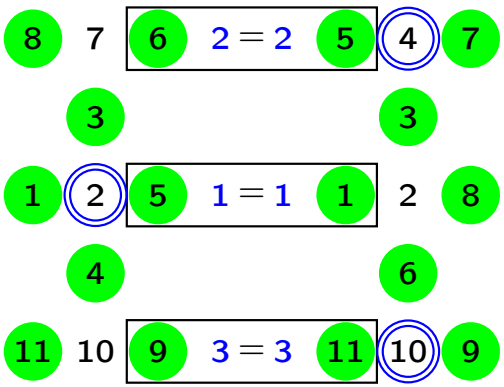
Right compatibility relation



Necessary conditions

- the orders on common edges are the same

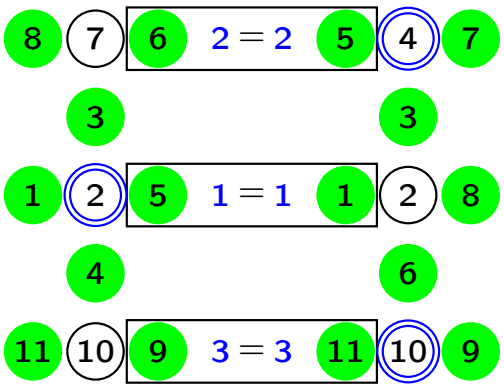
Right compatibility relation



Necessary conditions

- the orders on common edges are the same
- horizontal edges are after at least one of their endpoints

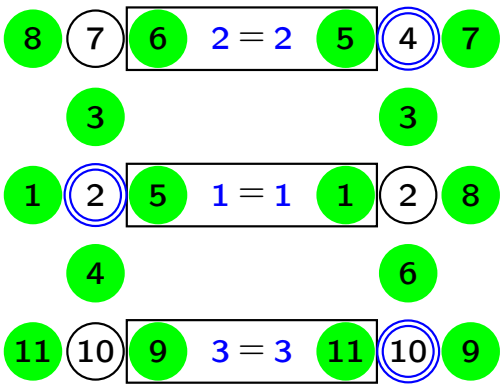
Right compatibility relation



Necessary conditions

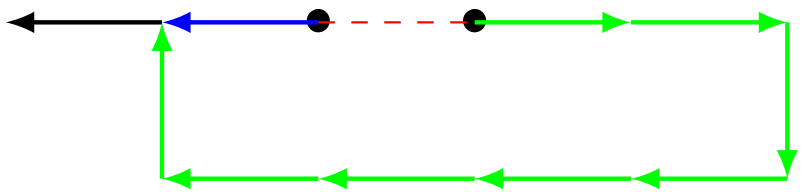
- the orders on common edges are the same
- horizontal edges are after at least one of their endpoints
- horizontal edges are before at least one of their endpoints : (the maximal edge of any cycle is vertical)

Right compatibility relation

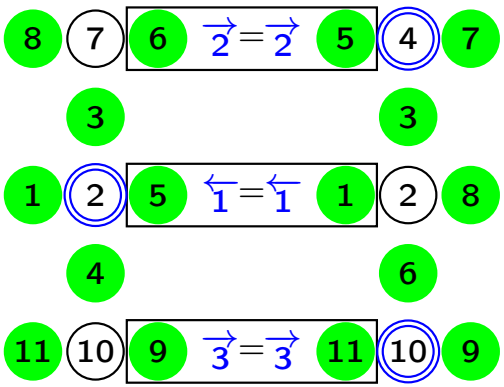


Necessary conditions

- the orders on common edges are the same
- horizontal edges are after at least one of their endpoints
- horizontal edges are before at least one of their endpoints : (the maximal edge of any cycle is vertical)

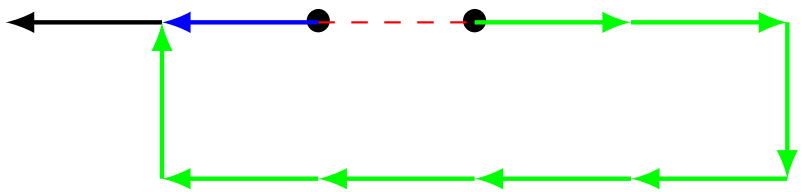


Right compatibility relation

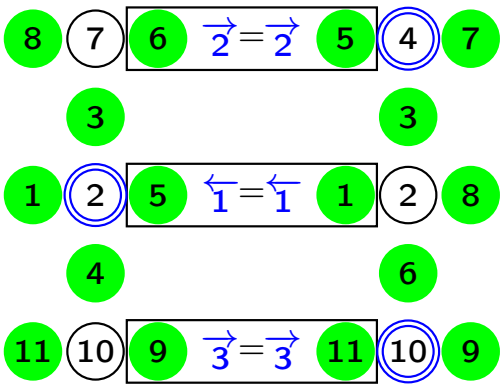


Necessary conditions

- the orders on common edges are the same
- horizontal edges are after at least one of their endpoints
- horizontal edges are before at least one of their endpoints : (the maximal edge of any cycle is vertical)



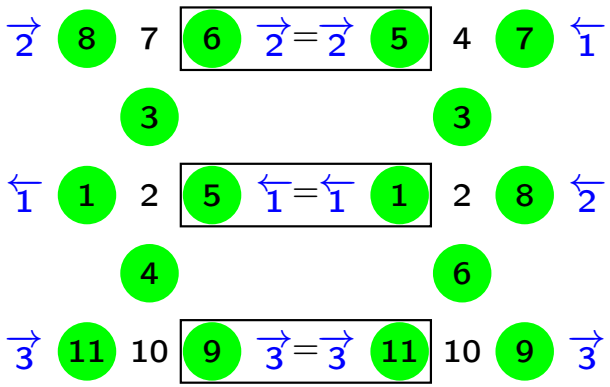
Right compatibility relation



Necessary and sufficient conditions

- the orders on common edges are the same
- horizontal edges are after at least one of their endpoints
- horizontal edges are before at least one of their endpoints : (the maximal edge of any cycle is vertical)

Right compatibility relation



Necessary and sufficient conditions

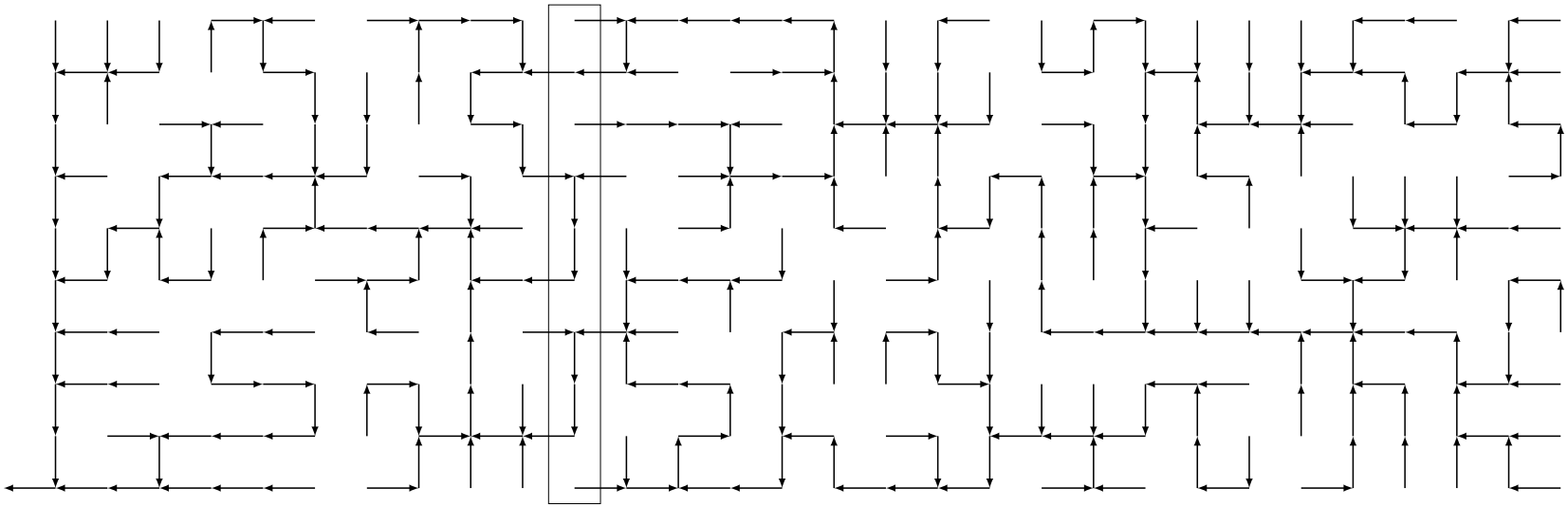
- the orders on common edges are the same
- horizontal edges are after at least one of their endpoints
- horizontal edges are before at least one of their endpoints : (the maximal edge of any cycle is vertical)

Reduction on the number of states : permutations and orientations on the right horizontal edges

- $\sharp states \leq 2^H \cdot H! (\leq (4H-1)!)$
- the transitions and alphabet are still the induced orders

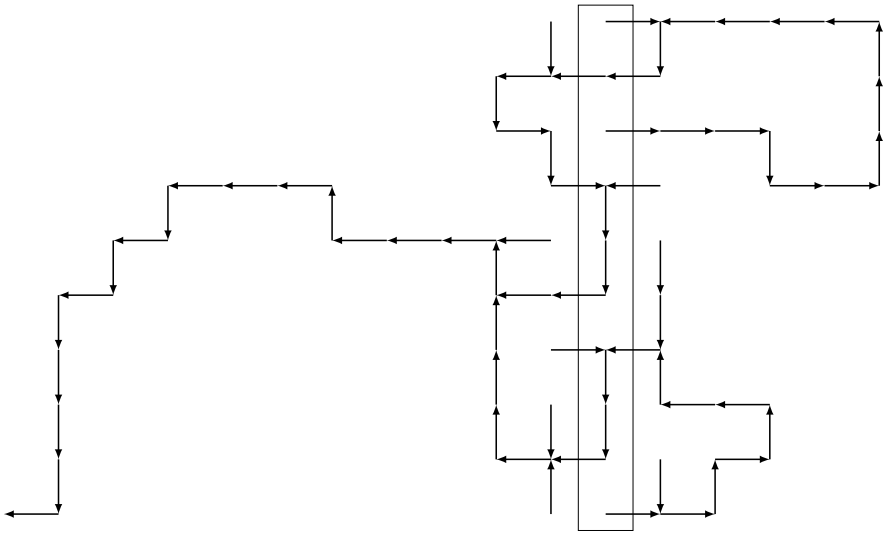
Proposition

Any induced order appears as a letter on some recurrent configuration's standardized decomposition on $[1, W] \times [1, H]$ for $W = 2H + 1$.



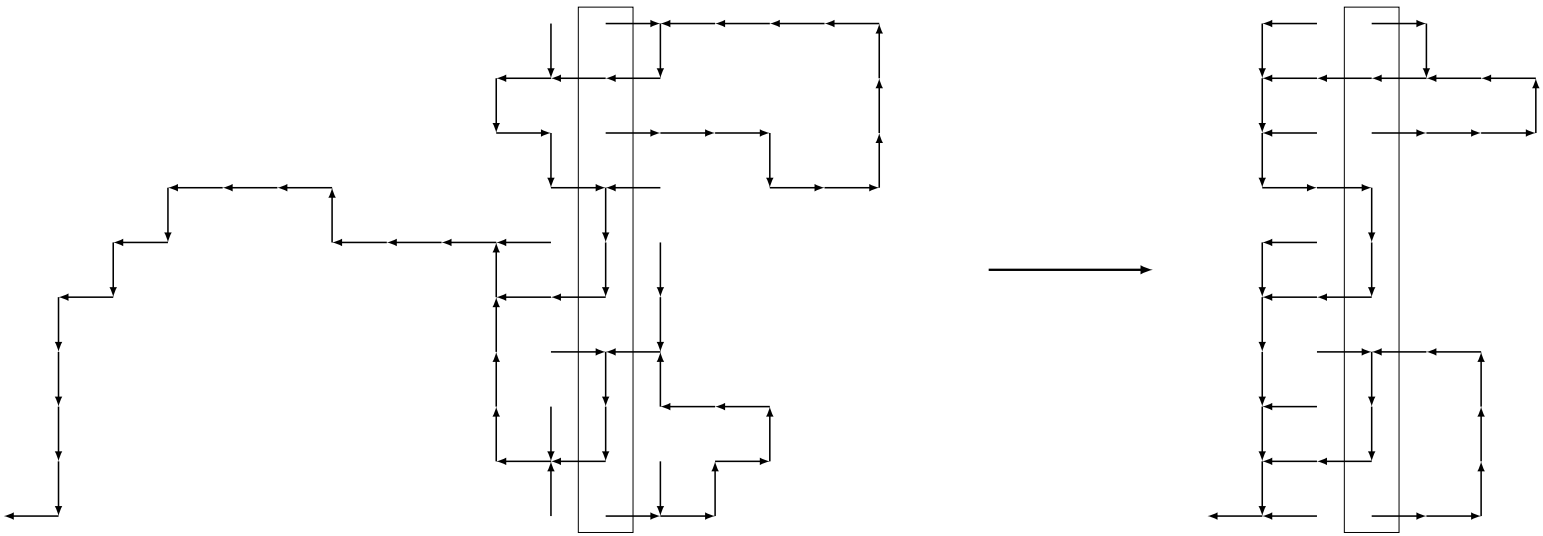
Proposition

Any induced order appears as a letter on some recurrent configuration's standardized decomposition on $[1, W] \times [1, H]$ for $W = 2H + 1$.



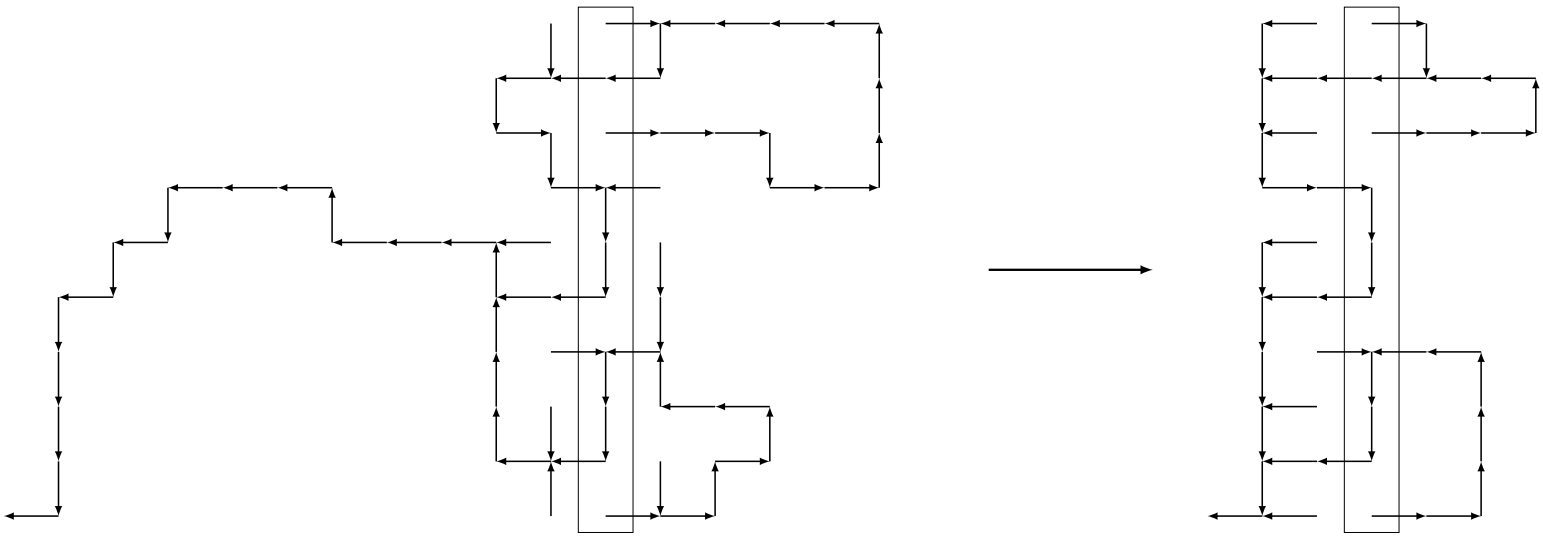
Proposition

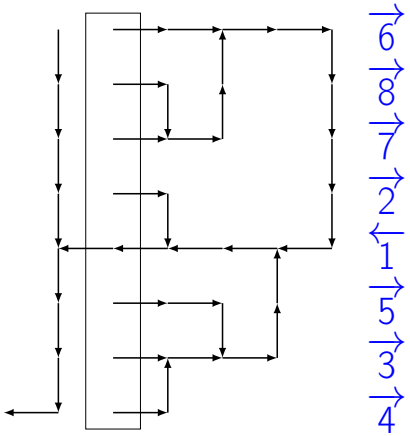
Any induced order appears as a letter on some recurrent configuration's standardized decomposition on $[1, W] \times [1, H]$ for $W = 2H + 1$.



Proposition

Any induced order appears as a letter on some recurrent configuration's standardized decomposition on $[1, W] \times [1, H]$ for $W = 2H + 1$.

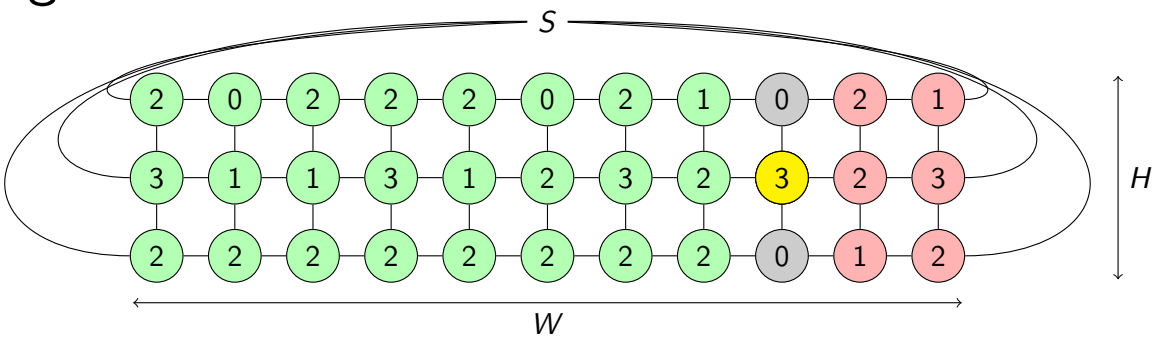




- Order over the right horizontal edges : $5, 4, 7, 8, 6, 1, 3, 2$
- Binary trees $\Rightarrow$ Separable permutations (avoiding pattern 2413 and 3142)
- Exponential lower bound on the number of states

Towards the infinite ladder (Járai, Lyons 07)

Approximation of $\mathbf{Z} \times [1, H]$ by square grid graphs rooted to the sink on the left and right sides.



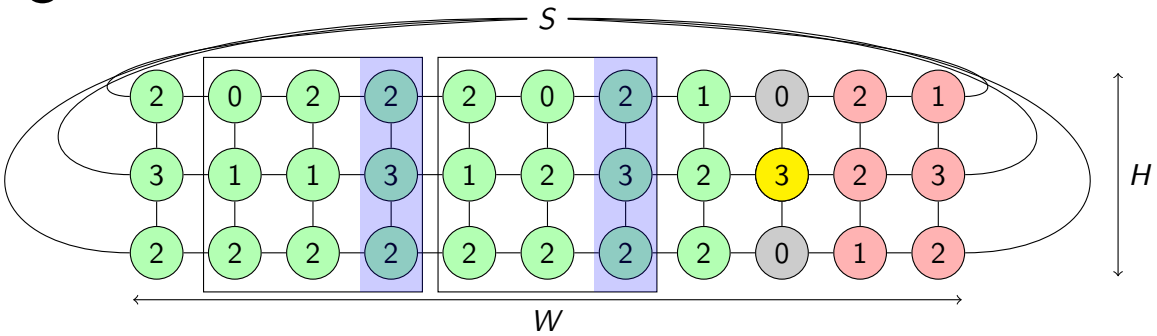
- vertices that topples from the left (green and yellow)
- vertices that topples from the right (red and yellow)
- other vertices (gray)

A *left burnable configuration* is a green/yellow configuration.
Decomposition in left-burnable blocks..

Automaton on a super set of column alphabet $\Rightarrow$ Parry measure $\Rightarrow$ Uniform distribution on left-burnable configurations of $\mathbf{Z} \times [1, H]$

Towards the infinite ladder (Járai, Lyons 07)

Approximation of $\mathbf{Z} \times [1, H]$ by square grid graphs rooted to the sink on the left and right sides.



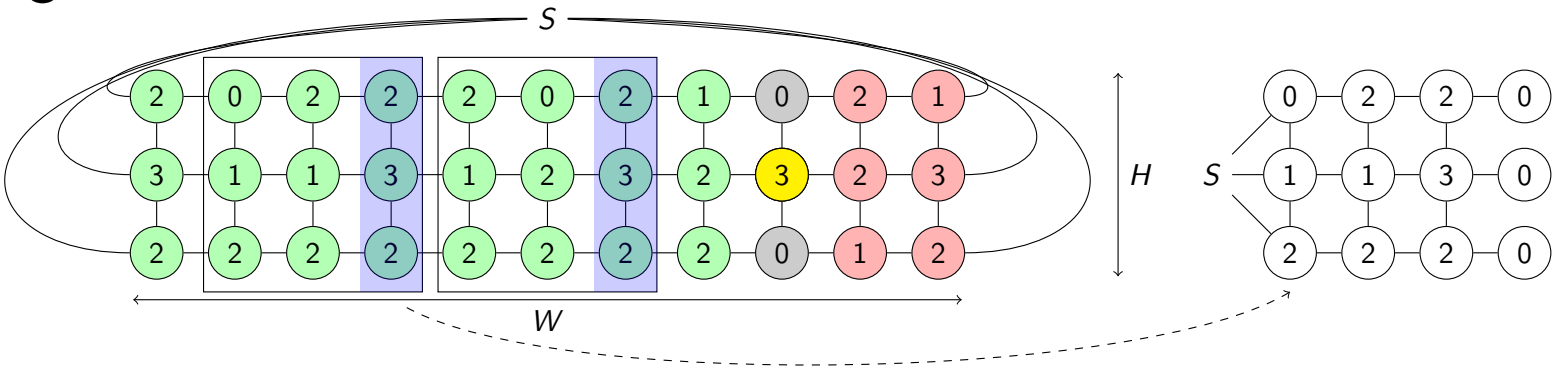
- vertices that topples from the left (green and yellow)
- vertices that topples from the right (red and yellow)
- other vertices (gray)

A *left burnable configuration* is a green/yellow configuration.
Decomposition in left-burnable blocks..

Automaton on a super set of column alphabet $\Rightarrow$ Parry measure $\Rightarrow$ Uniform distribution on left-burnable configurations of $\mathbf{Z} \times [1, H]$

Towards the infinite ladder (Járai, Lyons 07)

Approximation of $\mathbf{Z} \times [1, H]$ by square grid graphs rooted to the sink on the left and right sides.



- vertices that topples from the left (green and yellow)
- vertices that topples from the right (red and yellow)
- other vertices (gray)

A *left burnable configuration* is a green/yellow configuration.
Decomposition in left-burnable blocks..

Automaton on a super set of column alphabet $\Rightarrow$ Parry measure $\Rightarrow$ Uniform distribution on left-burnable configurations of $\mathbf{Z} \times [1, H]$

THANK YOU