Barak-Erdős graphs and the infinite-bin model

Sanjay Ramassamy
Institut de Physique Théorique

Joint work with Bastien Mallein (Université Paris-13)

Séminaire de physique statistique
et physique mathématique
IPhT, June 14 2021

- Barak-Erdős graphs (BEGs) are the directed acyclic version of Erdős-Rényi random graphs and a special case of last passage percolation (LPP).

- The infinite-bin model (IBM) is an interacting particle system, whose Markovian evolution depends on a probability measure μ on the set of positive integers.

- When μ is a geometric distribution, there is a coupling between the IBM and BEGs, relating the speed of the front of the IBM to the length of the longest path of BEGs.

Outline :

1. Barak-Erdős graphs and last passage percolation

2. The infinite-bin model (IBM)

3. Properties of Barak-Erdős graphs via the IBM

4. Perspectives

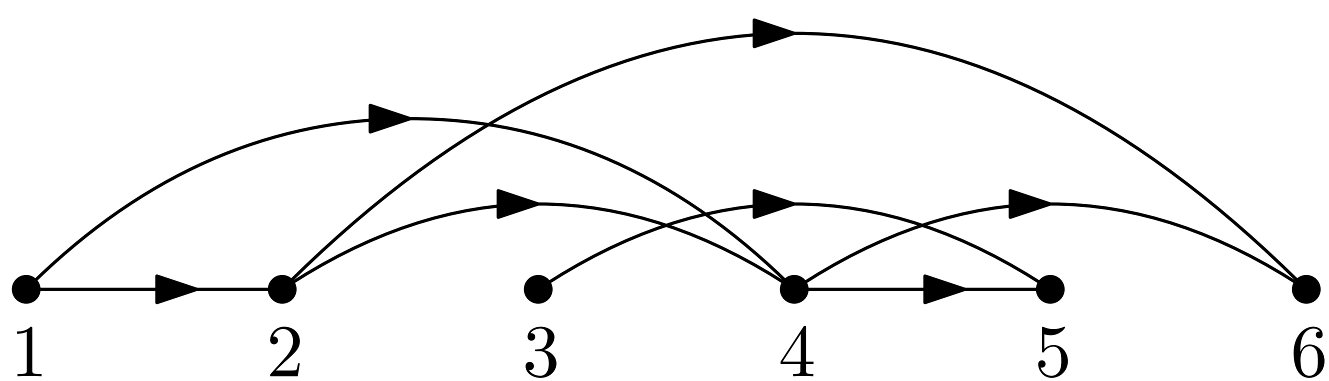
1 Barak-Erdős graphs and last passage percolation

Construction of Barak-Erdős graphs

- Fix $n \geq 1$ integer and $0 \leq p \leq 1$.
- Vertex set is $\{1, 2, \ldots, n\}$.
- For each pair $i < j$, add an edge directed from i to j with probability p, independently for each pair (i, j).

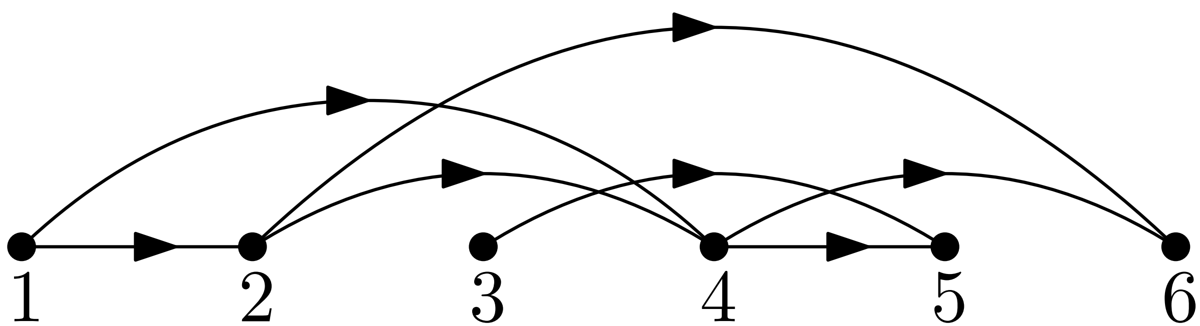
Construction of Barak-Erdős graphs

- Fix $n \geq 1$ integer and $0 \leq p \leq 1$.
- Vertex set is $\{1, 2, \ldots, n\}$.
- For each pair $i < j$, add an edge directed from i to j with probability p, independently for each pair (i, j).



- Introduced by Barak and Erdős in 1984.

- The most studied feature is the length of the longest path $L_n(p)$.



L=3

$1 \to 2 \to 4 \to 5$

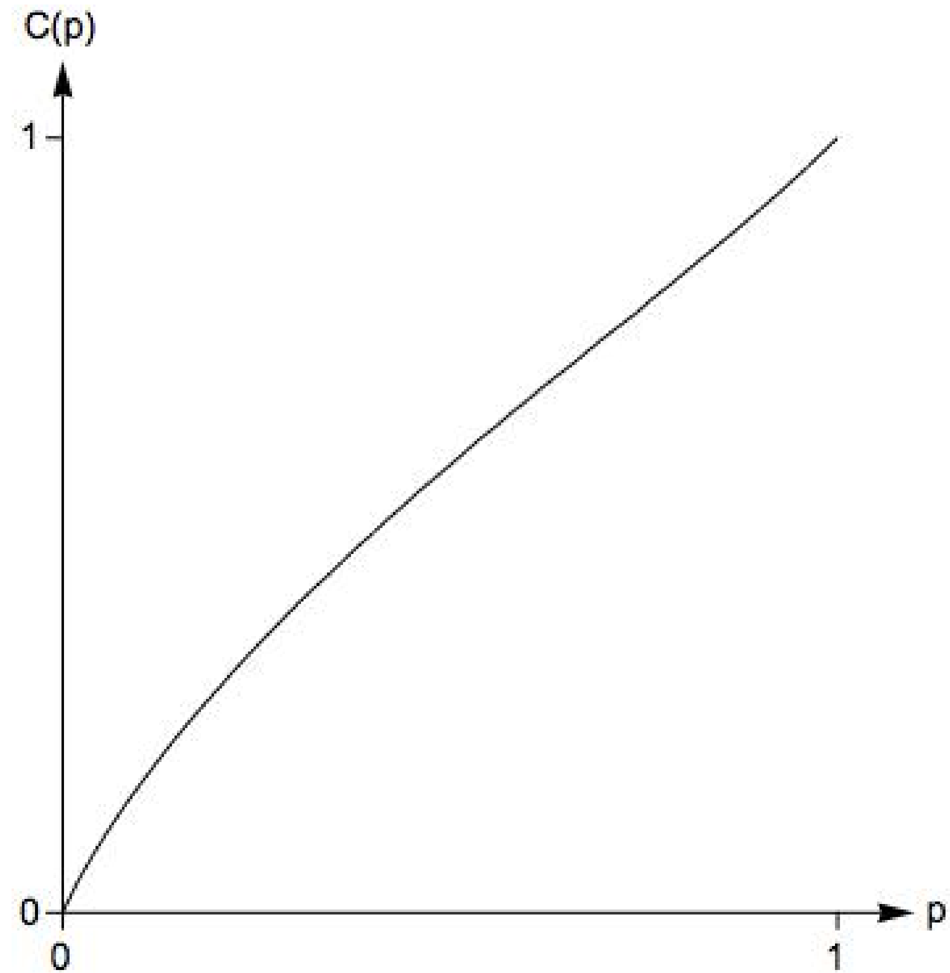
$1 \to 2 \to 4 \to 6$

- Applications to performance evalution of computer systems (Gelenbe-Nelson-Philips-Tantawi, Isopi-Newman), mathematical ecology (Cohen-Newman) and queuing systems (Foss-Konstantopoulos).

- The length of the longest path grows linearly in the number of vertices :

$$\frac{1}{n} L_n(p) \xrightarrow[n\to\infty]{} C(p) \text{ in probability (Newman '92)}$$

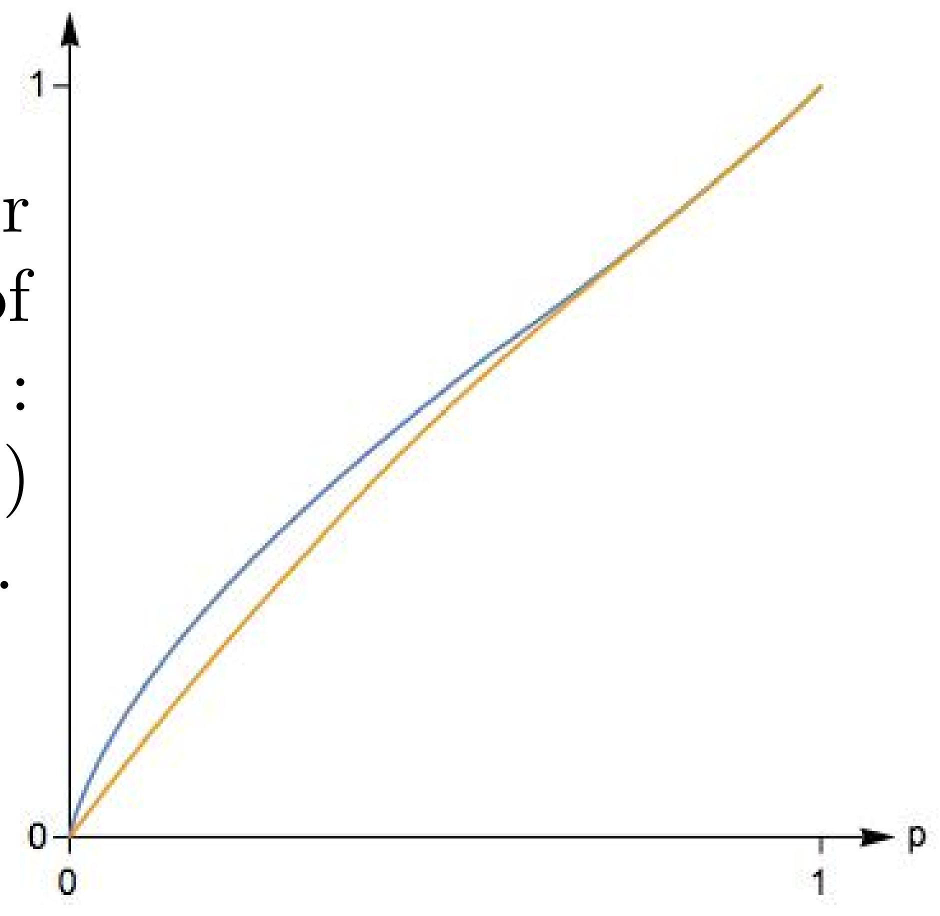
- The growth rate C is a function of p :



Properties of $C(p)$

- $C(p)$ is continuous and $C'(0) = e$ (Newman '92).

- Upper and lower bound for $C(p)$, yielding expansion of $C(1-q)$ for q tending to 0 : $1-q+q^2-3q^3+7q^4+O(q^5)$ (Foss-Konstantopoulos '03).



New results (Mallein-R.)

- For $p > 0$, $C(p)$ is analytic and can be obtained as the sum of a series.

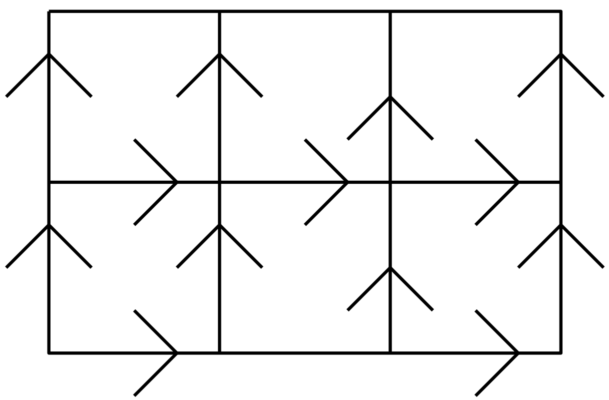
- The power series expansion of $C(p)$ centered at 1 has integer coefficients.

- $C(p)$ has no second derivative at $p = 0$:

$$C(p) = ep - \frac{e\pi^2 p}{2(\log p)^2} + o\left(\frac{p}{(\log p)^2}\right) \text{ when } p \to 0.$$

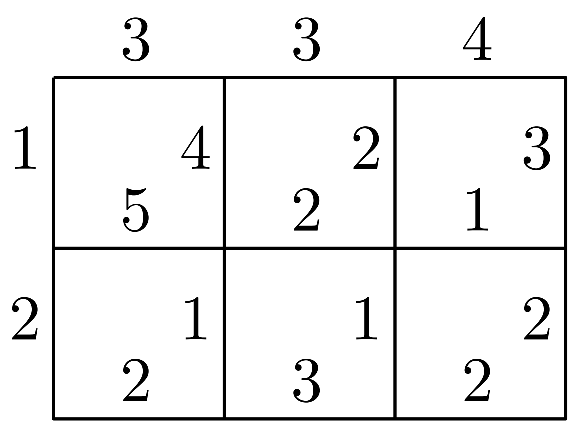
Last passage percolation

- Consider a deterministic directed acyclic graph and attach i.i.d. random lengths to each edge. The length of a path is the sum of the lengths of its edges.



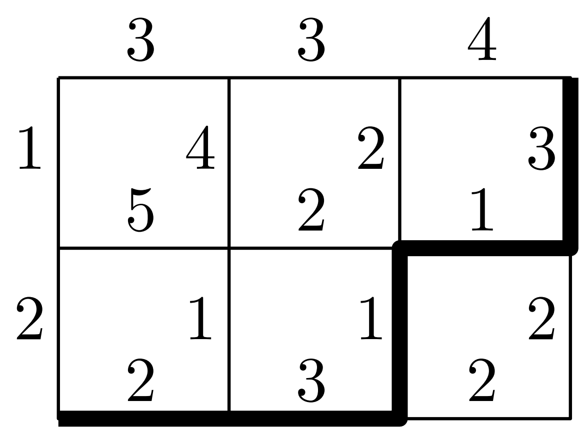
Last passage percolation

- Consider a deterministic directed acyclic graph and attach i.i.d. random lengths to each edge. The length of a path is the sum of the lengths of its edges.



Last passage percolation

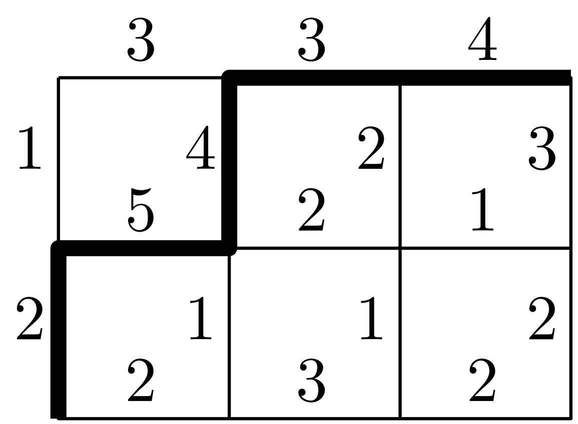
- Consider a deterministic directed acyclic graph and attach i.i.d. random lengths to each edge. The length of a path is the sum of the lengths of its edges.



Path of length 10

Last passage percolation

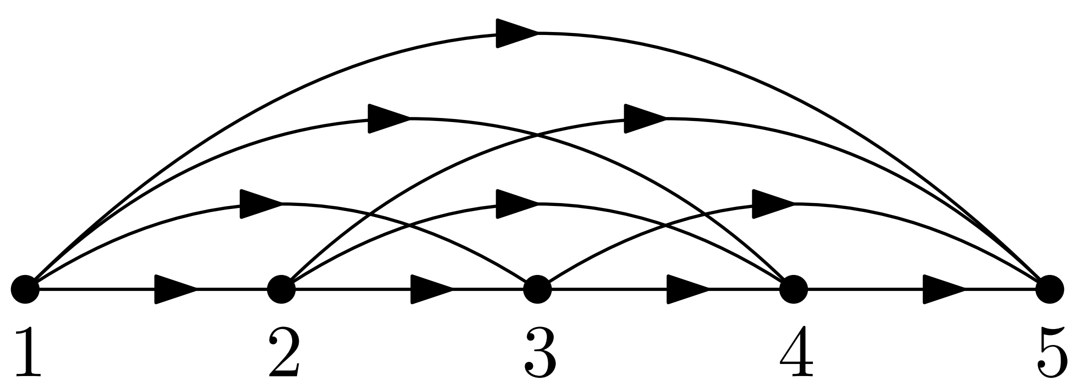
- Consider a deterministic directed acyclic graph and attach i.i.d. random lengths to each edge. The length of a path is the sum of the lengths of its edges.



- How does the length of the longest path grow as the size of the graph grows ?

LPP on the complete graph

- To each edge attach an i.i.d. weight/length, which is 1 with probability p and x with probability $1 - p$.
- The case $x = -\infty$ is the Barak-Erdős case.
- Denote by $C(p, x)$ the linear growth rate of the "length" of the "longest" path.



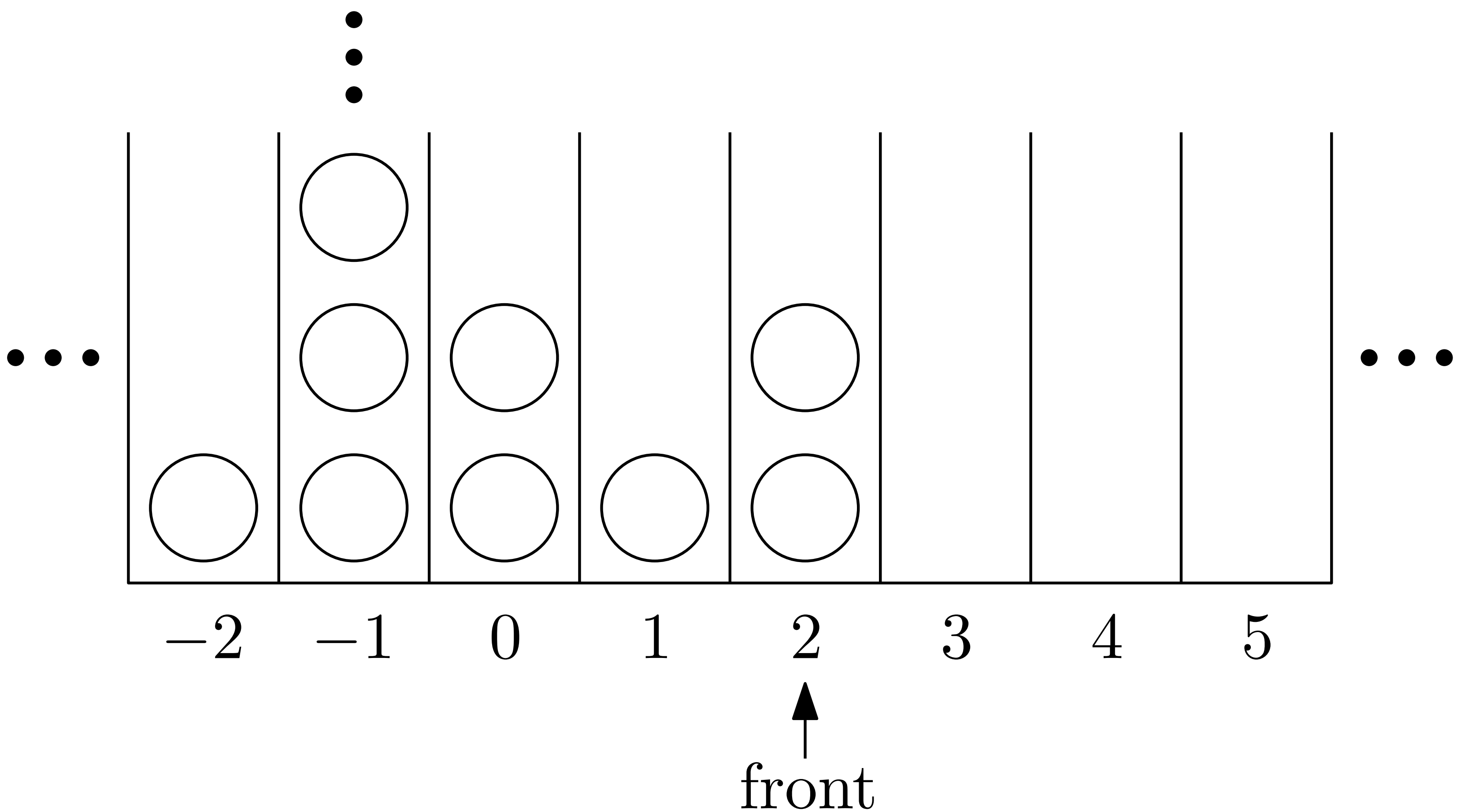
LPP on the complete graph

- To each edge attach an i.i.d. weight/length, which is 1 with probability p and x with probability $1-p$.
- The case $x = -\infty$ is the Barak-Erdős case.
- Denote by $C(p,x)$ the linear growth rate of the "length" of the "longest" path.
- The function $x \mapsto C(p,x)$ is non-differentiable exactly at values of x which are either 0 or a negative rational or of the form n or $1/n$, where n is an integer ≥ 2 (Foss-Konstantopoulos-Pyatkin '20).

2 The infinite-bin model (IBM)

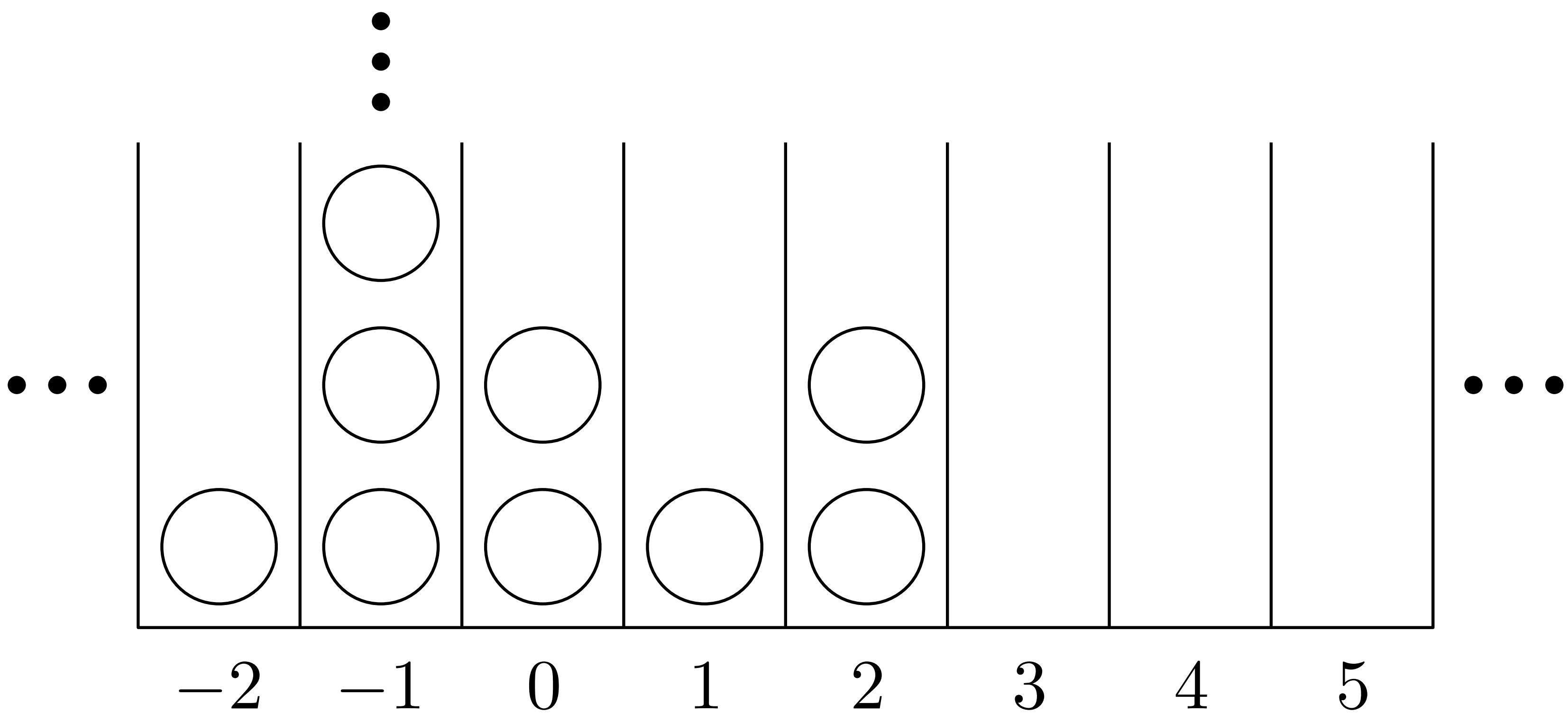
Configurations

Infinitely many balls placed inside bins indexed by $\mathbb{Z}$, such that the set of indices of nonempty bins has a maximal element, the *front*.



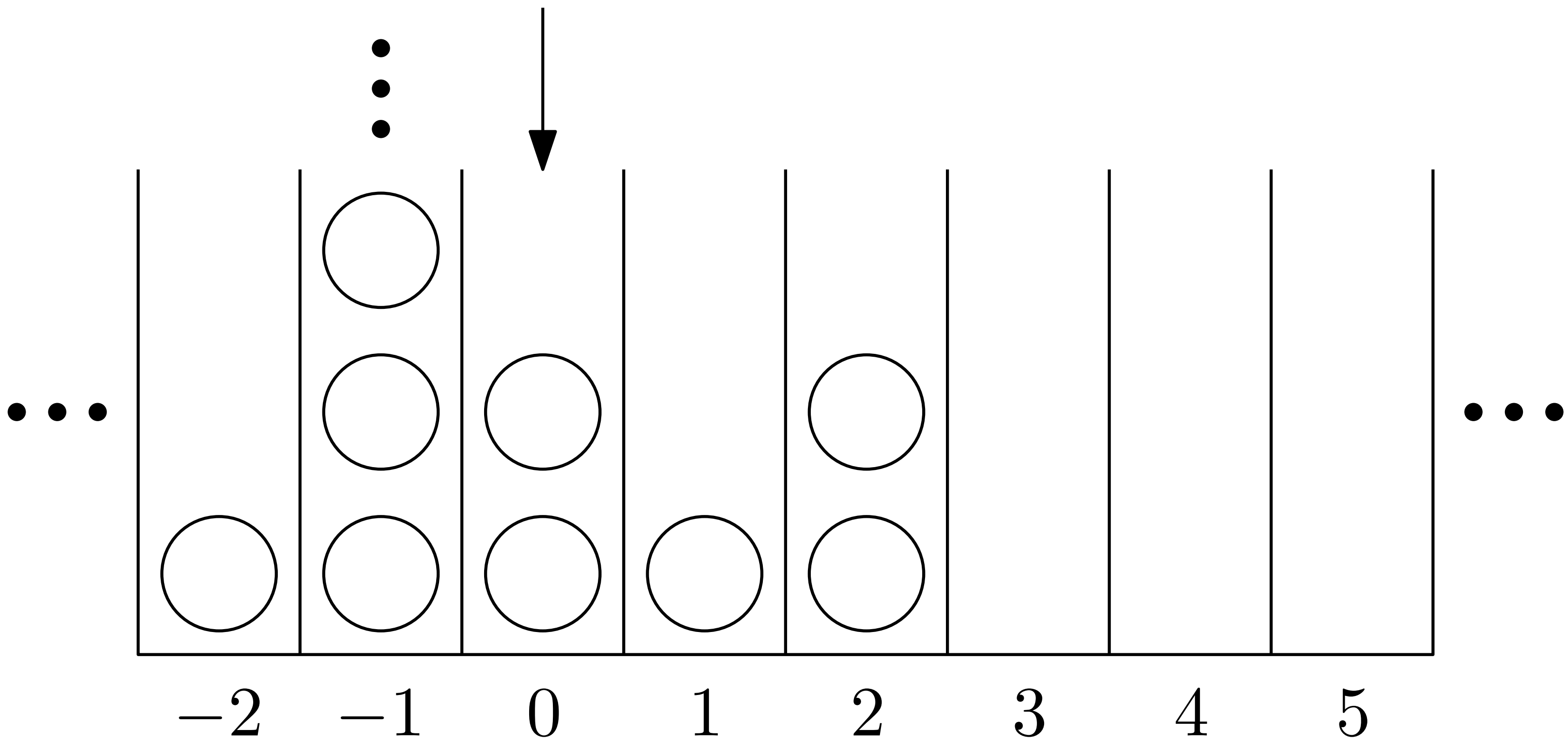
Move of type k

Add a ball in the bin immediately to the right of the bin containing the k-th ball, where balls are counted from right to left.



Move of type k

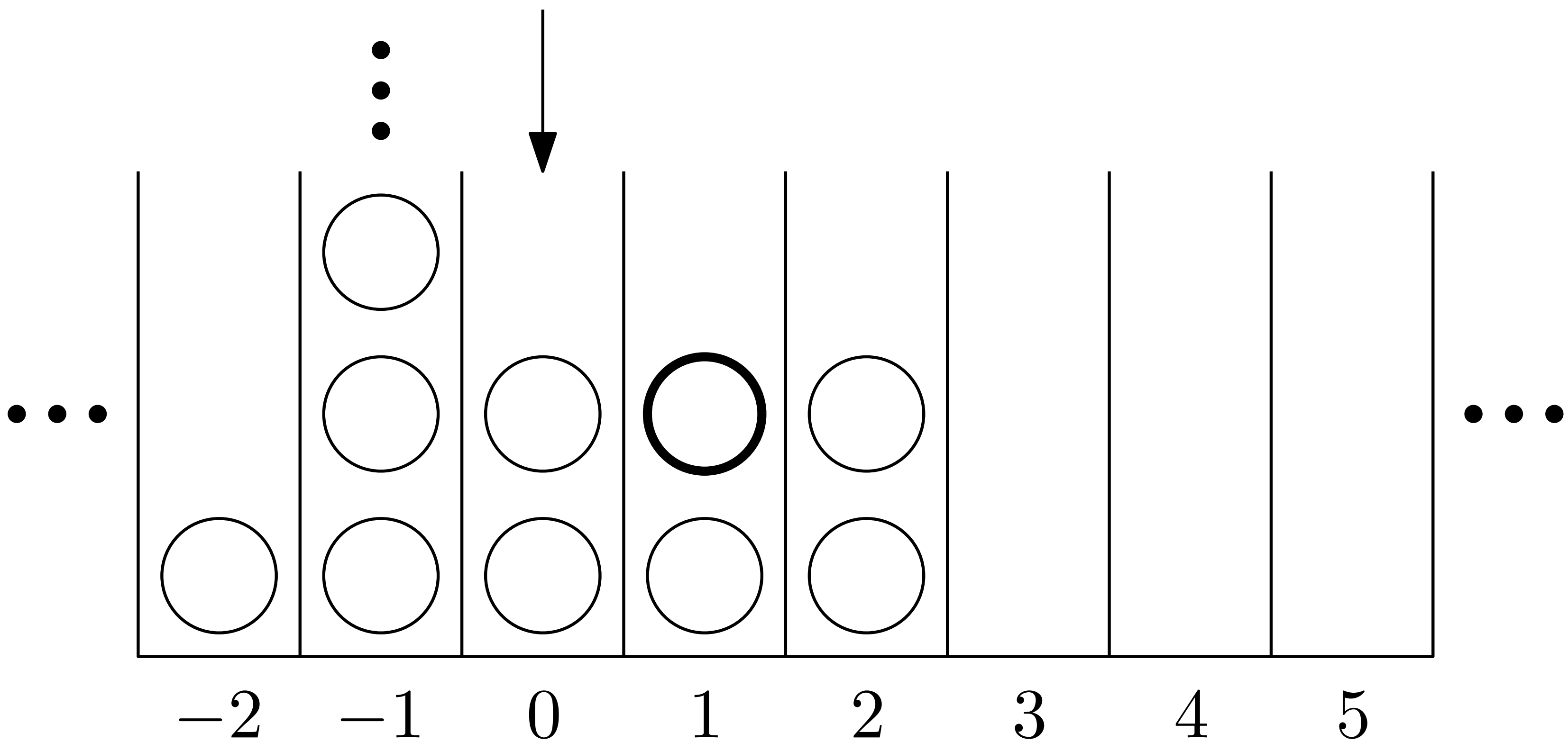
Add a ball in the bin immediately to the right of the bin containing the k-th ball, where balls are counted from right to left.



Move of type 5

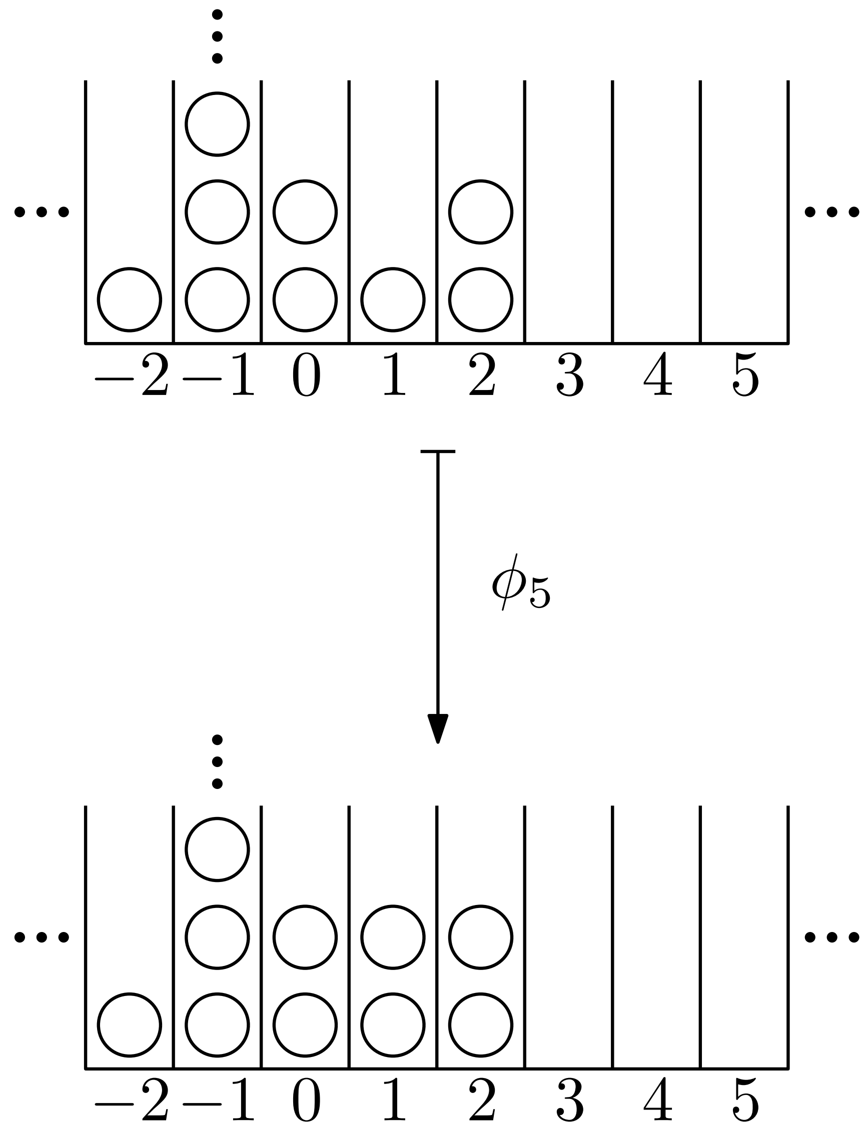
Move of type k

Add a ball in the bin immediately to the right of the bin containing the k-th ball, where balls are counted from right to left.

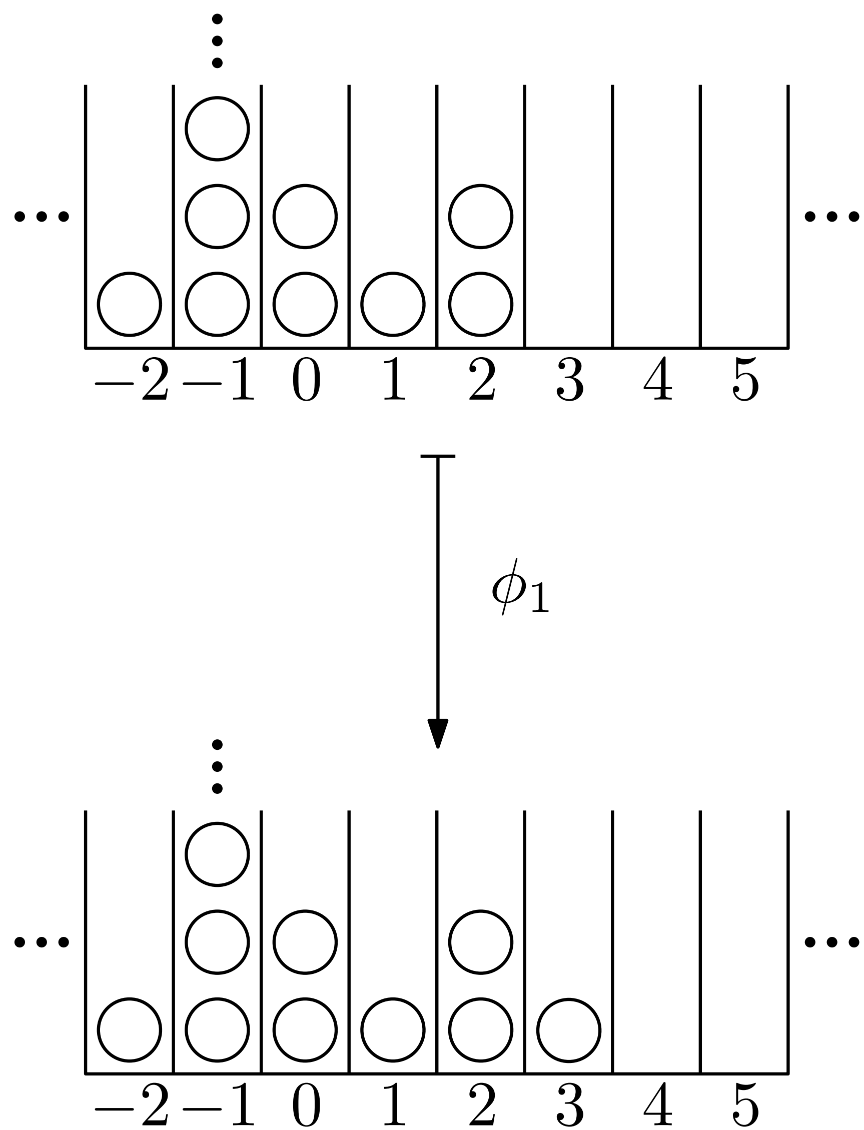


Move of type 5

ϕ_k : move of type k



ϕ_k : move of type k



Markovian evolution

- Fix an initial configuration X_0 and a probability distribution μ on $\mathbb{N} = \{1, 2, 3, \ldots\}$.

- The infinite-bin model with move distribution μ (IBM(μ) for short) and initial configuration X_0 is the Markov chain $(X_n)_{n \geq 0}$ satisfying

$$X_n = \phi_{\xi_n}(X_{n-1}),$$

where the sequence $(\xi_n)_{n \geq 1}$ is i.i.d. distributed like μ.

- Introduced by Foss and Konstantopoulos in 2003 to study the longest paths of Barak-Erdős graphs.

- Special case when μ is the uniform measure on $\{1, \ldots, n\}$ already appeared in Aldous-Pitman '83.

Speed of the front

Consider the IBM(μ) with initial configuration X_0. Denote by F_n the position of the front at time n.

Theorem (Foss-Konstantopoulos, Mallein-R.). *There exists* $v_\mu \in (0, 1]$ *such that*

$$\lim_{n\to\infty} \frac{F_n - F_0}{n} = v_\mu \ a.s.$$

v_μ *(independent of* X_0*) is called the* speed *of the IBM(μ).*

Coupling with Barak-Erdős (Foss-Konstantopoulos '03)

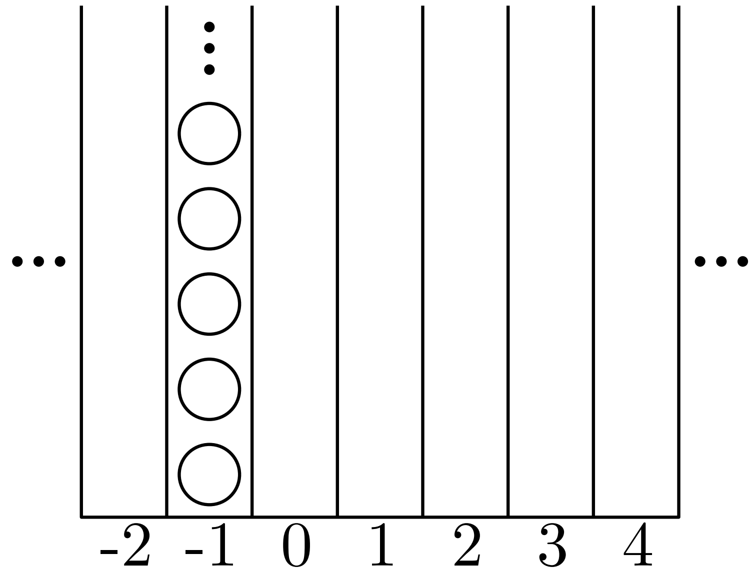
- μ_p : geometric distribution on $\{1, 2, \ldots\}$ with parameter p, *i.e.* $\mu_p(k) = p(1-p)^{k-1}$ for $k \geq 1$.

- The speed of the IBM(μ_p) equals the growth rate of the length of the longest path in Barak-Erdős graphs with edge probability p :

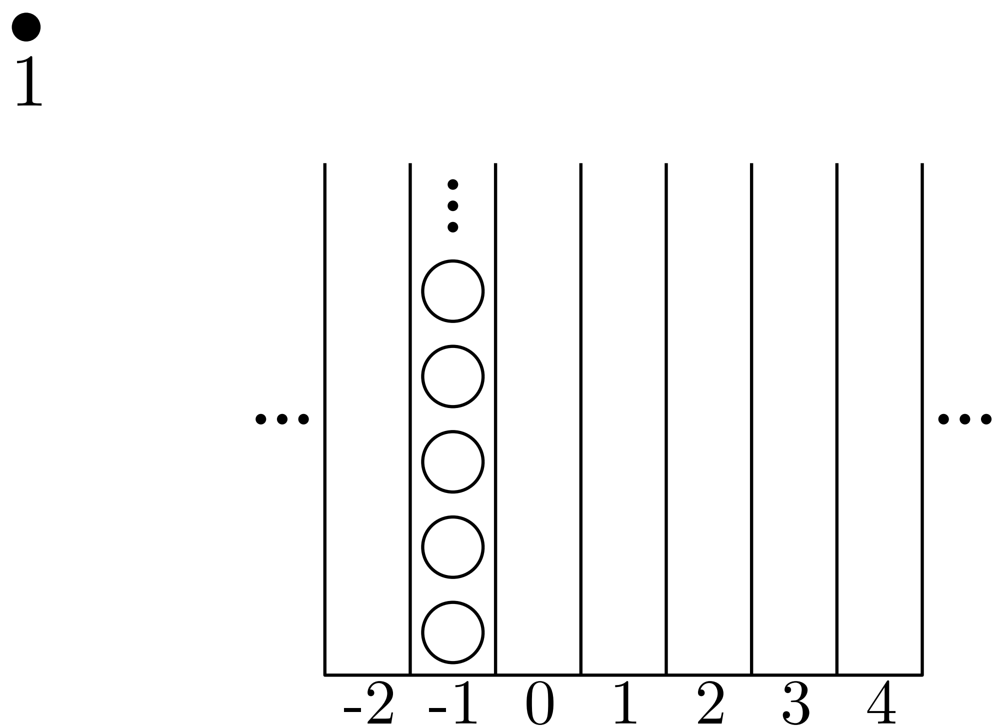
$$v_{\mu_p} = C(p).$$

- Grow Barak-Erdős graph one vertex after the other.

- For each vertex n, call l_n the length of the longest path ending at n. Place a ball with label n in the bin indexed by l_n.

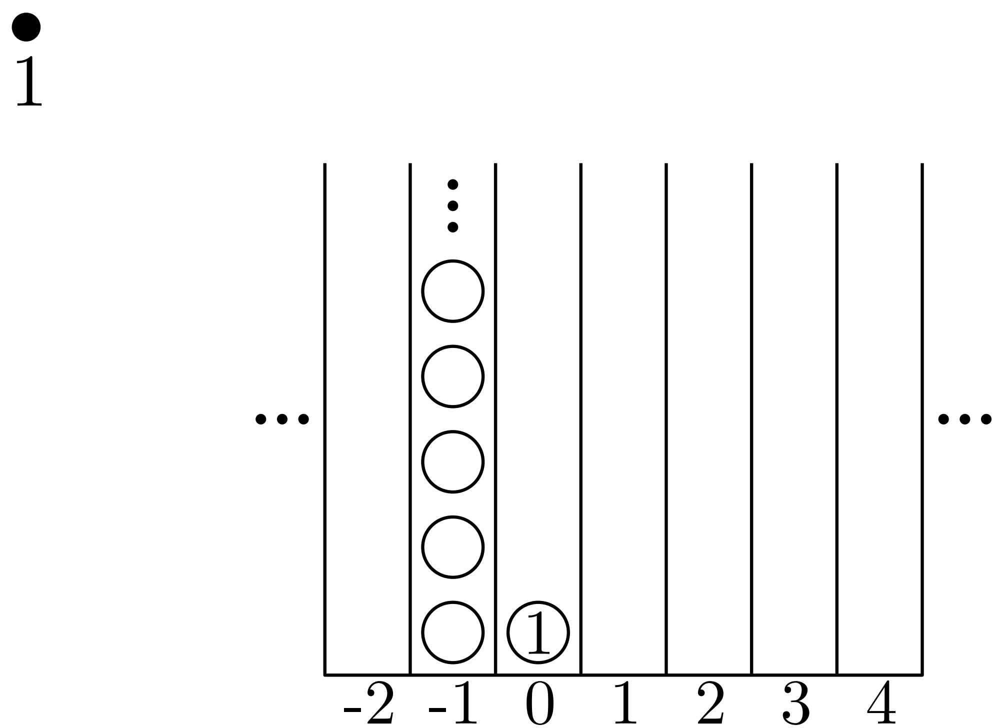


- Grow Barak-Erdős graph one vertex after the other.
- For each vertex n, call l_n the length of the longest path ending at n. Place a ball with label n in the bin indexed by l_n.



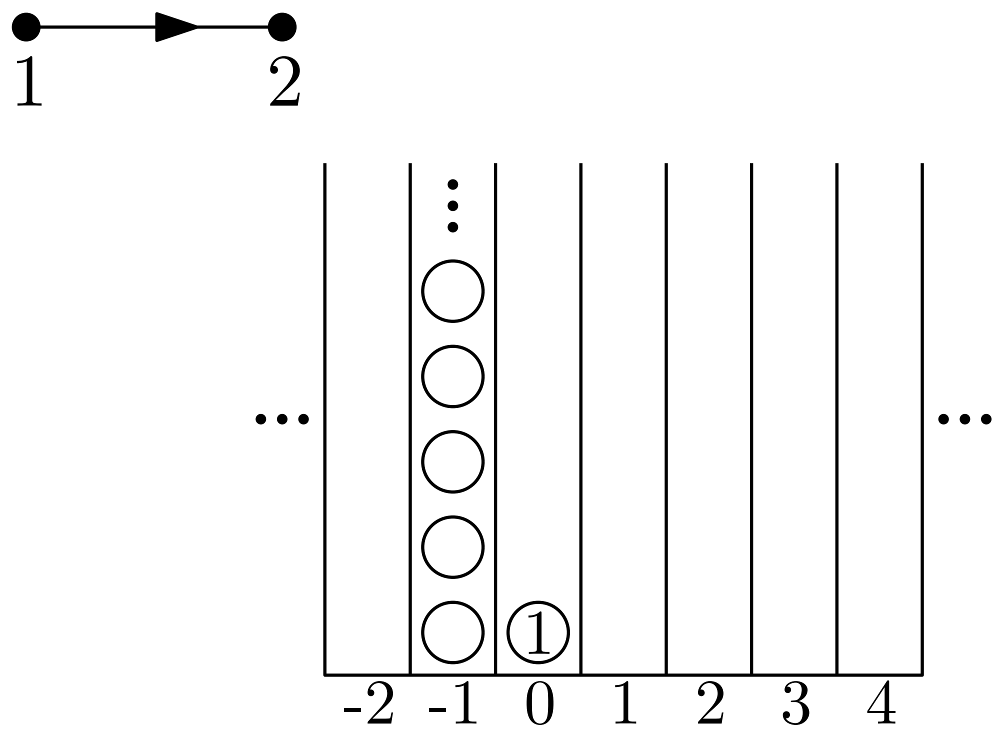
- Grow Barak-Erdős graph one vertex after the other.

- For each vertex n, call l_n the length of the longest path ending at n. Place a ball with label n in the bin indexed by l_n.



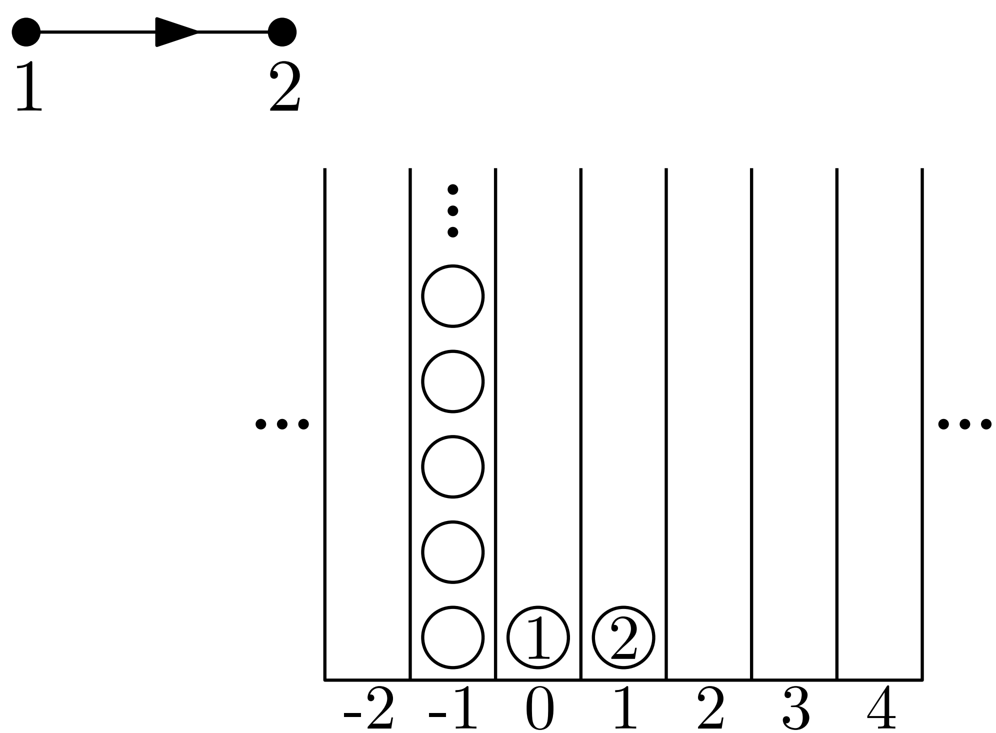
- Grow Barak-Erdős graph one vertex after the other.

- For each vertex n, call l_n the length of the longest path ending at n. Place a ball with label n in the bin indexed by l_n.

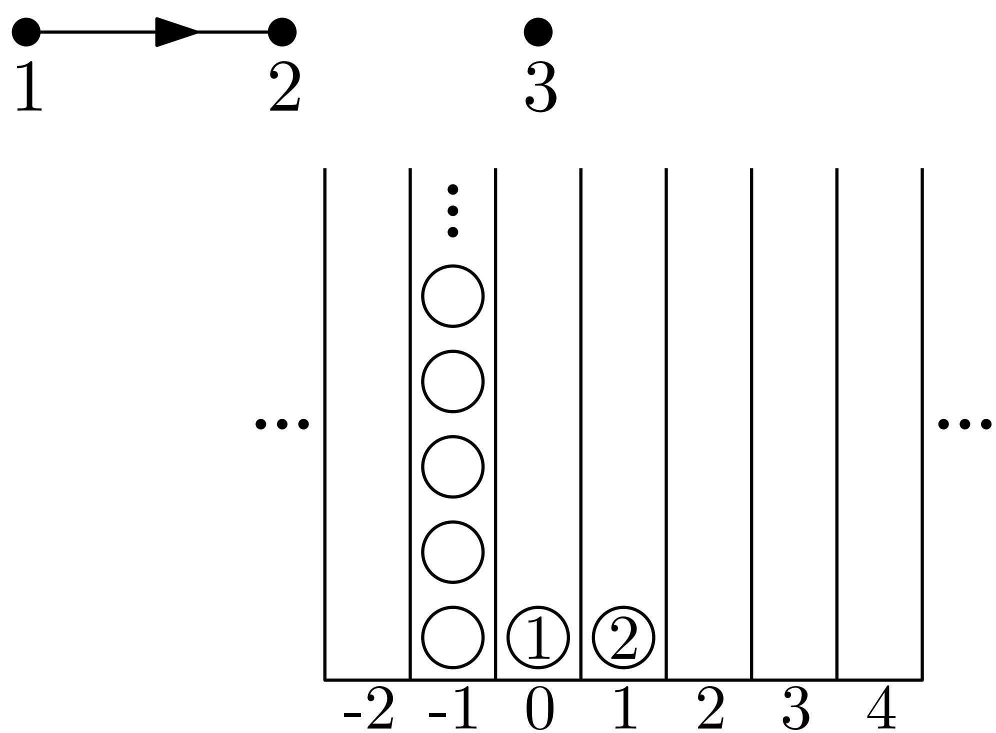


- Grow Barak-Erdős graph one vertex after the other.

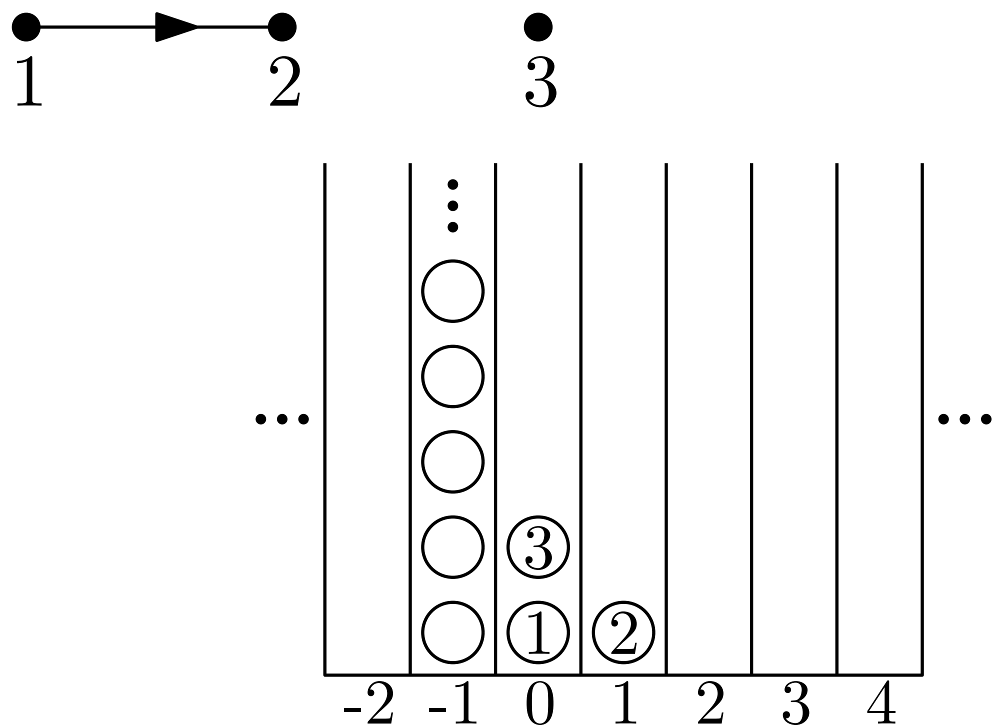
- For each vertex n, call l_n the length of the longest path ending at n. Place a ball with label n in the bin indexed by l_n.



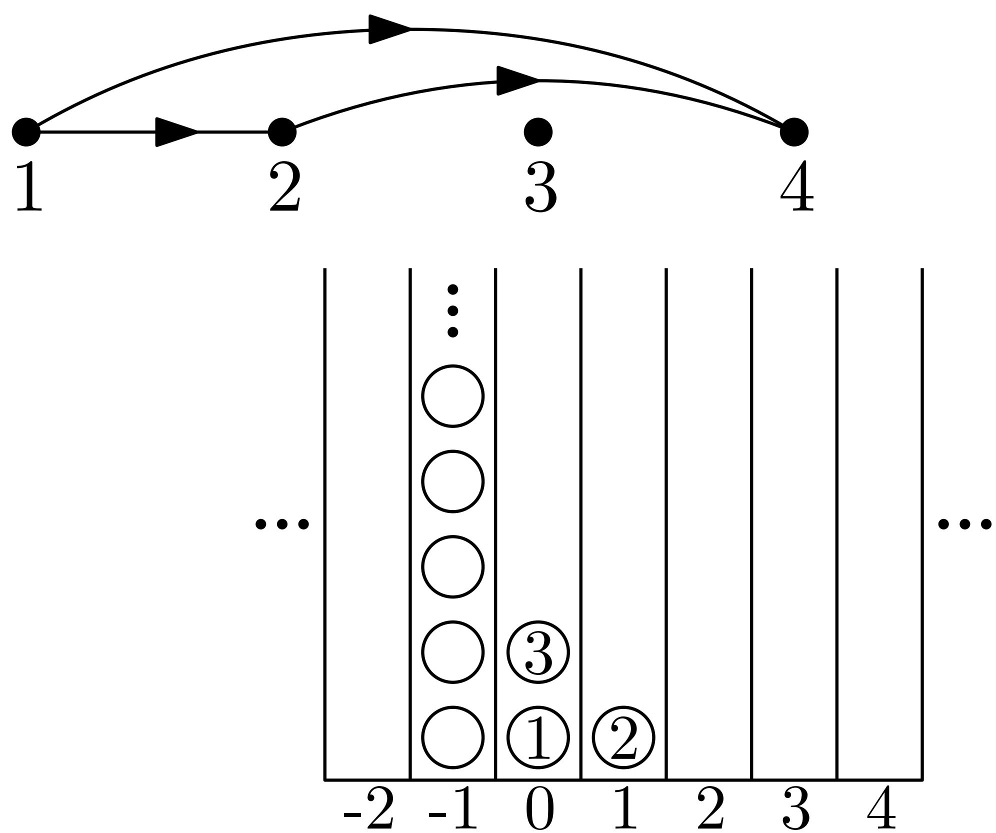
- Grow Barak-Erdős graph one vertex after the other.
- For each vertex n, call l_n the length of the longest path ending at n. Place a ball with label n in the bin indexed by l_n.



- Grow Barak-Erdős graph one vertex after the other.
- For each vertex n, call l_n the length of the longest path ending at n. Place a ball with label n in the bin indexed by l_n.

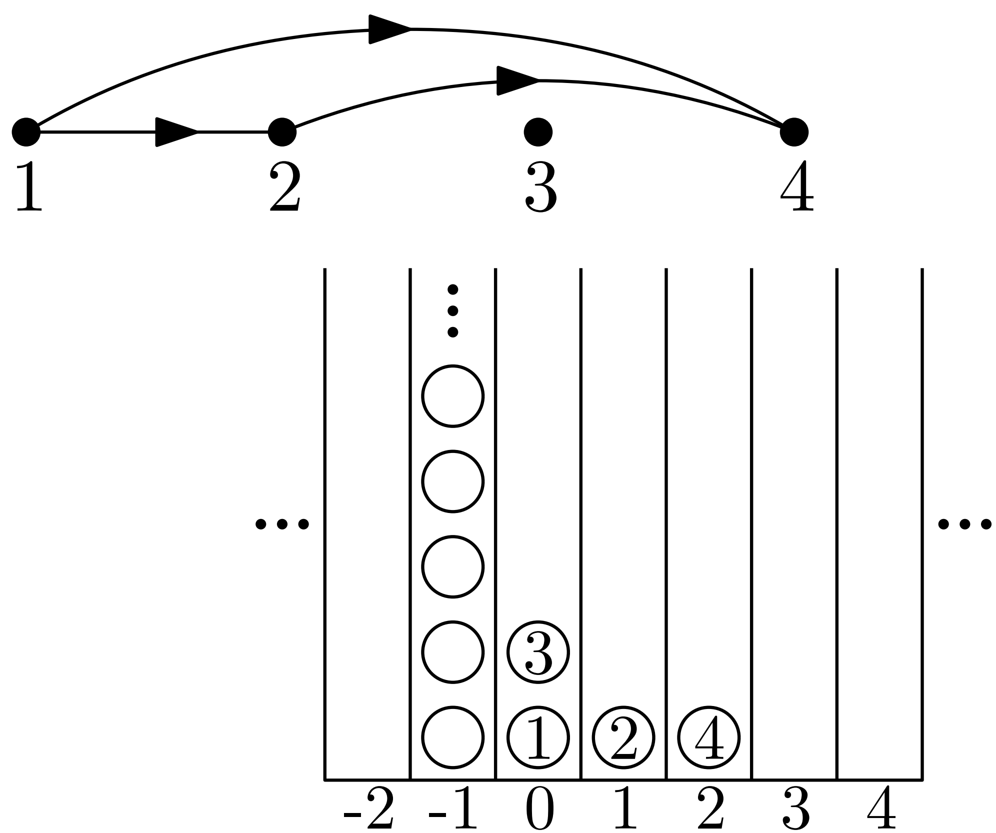


- Grow Barak-Erdős graph one vertex after the other.
- For each vertex n, call l_n the length of the longest path ending at n. Place a ball with label n in the bin indexed by l_n.



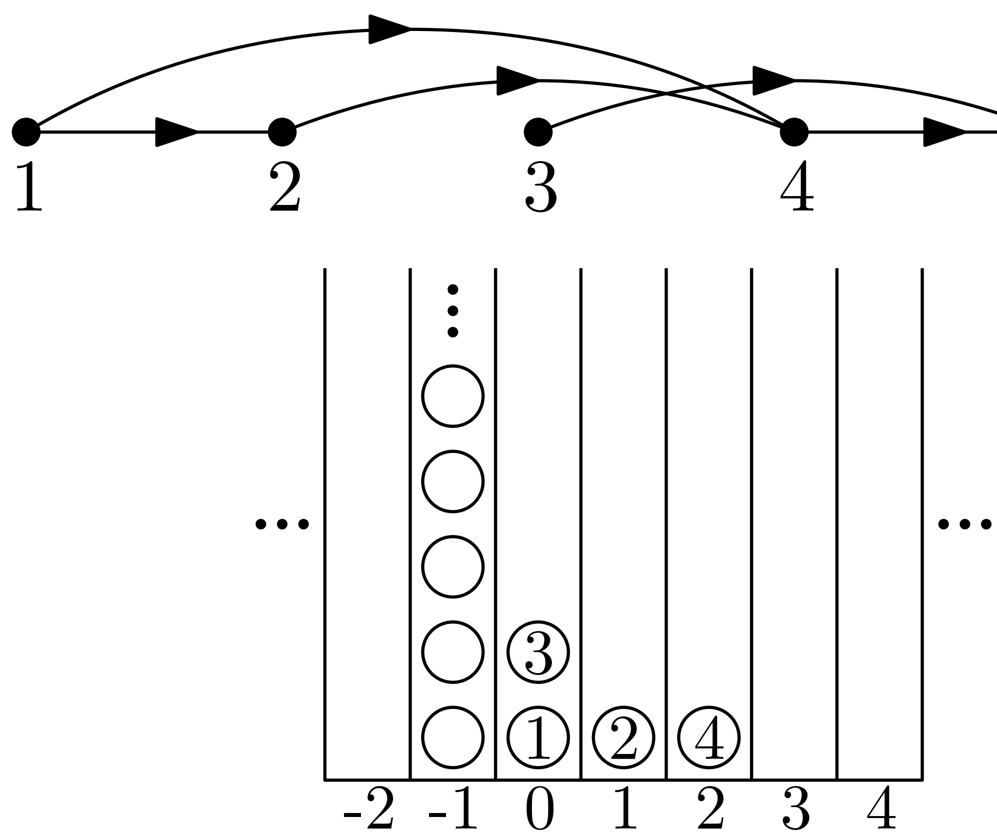
- Grow Barak-Erdős graph one vertex after the other.

- For each vertex n, call l_n the length of the longest path ending at n. Place a ball with label n in the bin indexed by l_n.



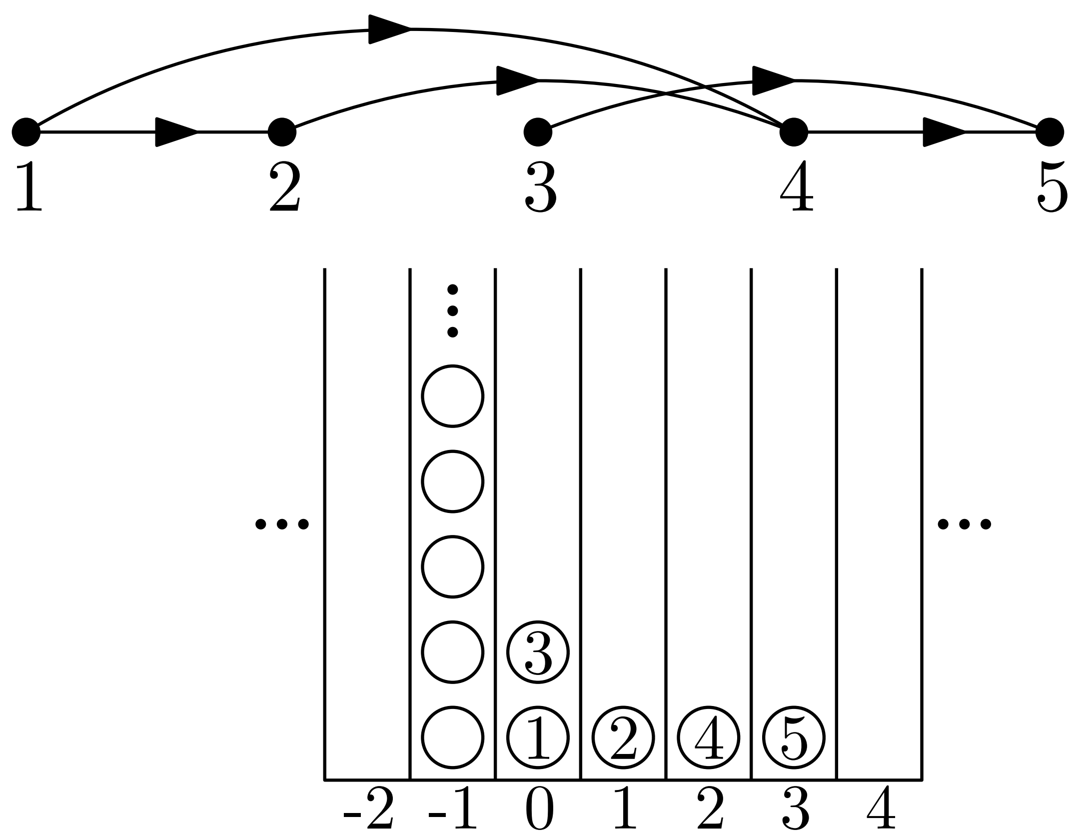
- Grow Barak-Erdős graph one vertex after the other.

- For each vertex n, call l_n the length of the longest path ending at n. Place a ball with label n in the bin indexed by l_n.



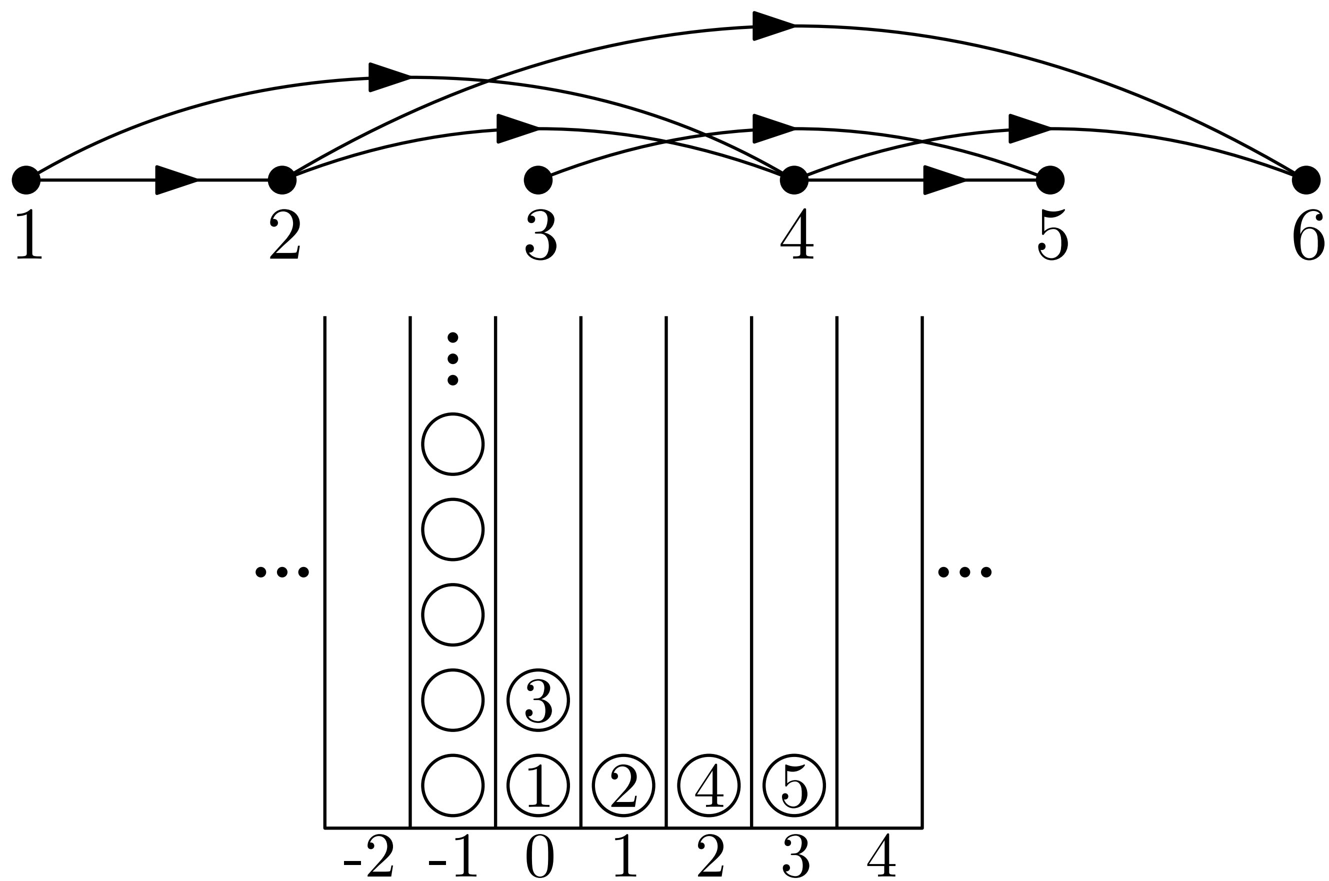
- Grow Barak-Erdős graph one vertex after the other.

- For each vertex n, call l_n the length of the longest path ending at n. Place a ball with label n in the bin indexed by l_n.



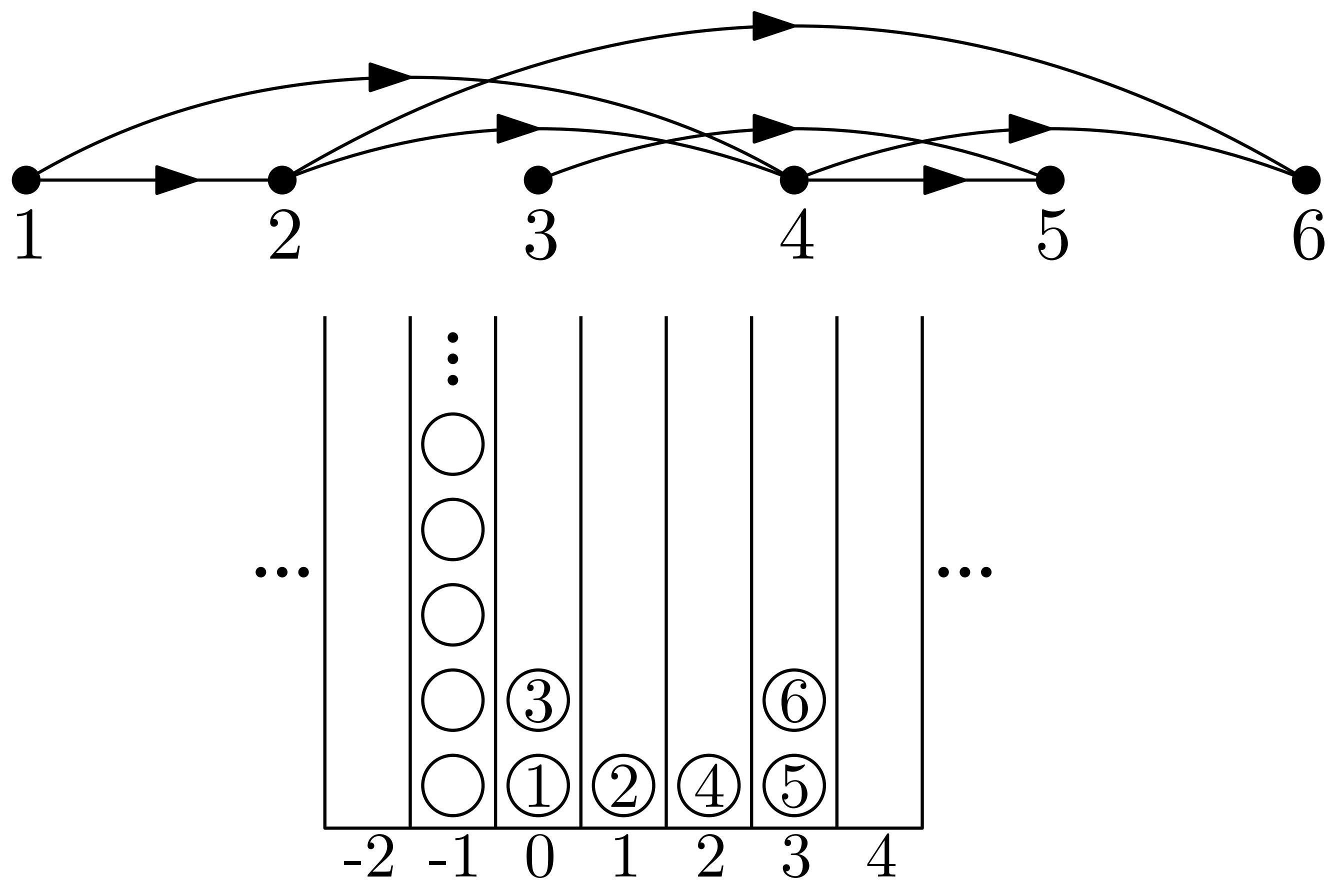
- Grow Barak-Erdős graph one vertex after the other.

- For each vertex n, call l_n the length of the longest path ending at n. Place a ball with label n in the bin indexed by l_n.



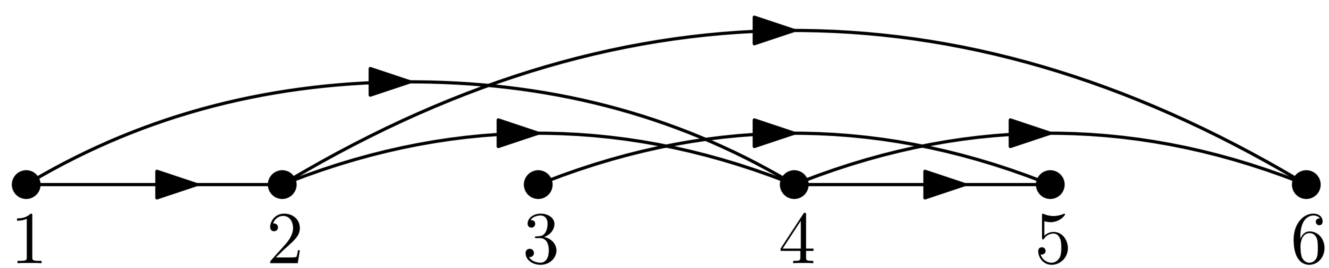
- Grow Barak-Erdős graph one vertex after the other.

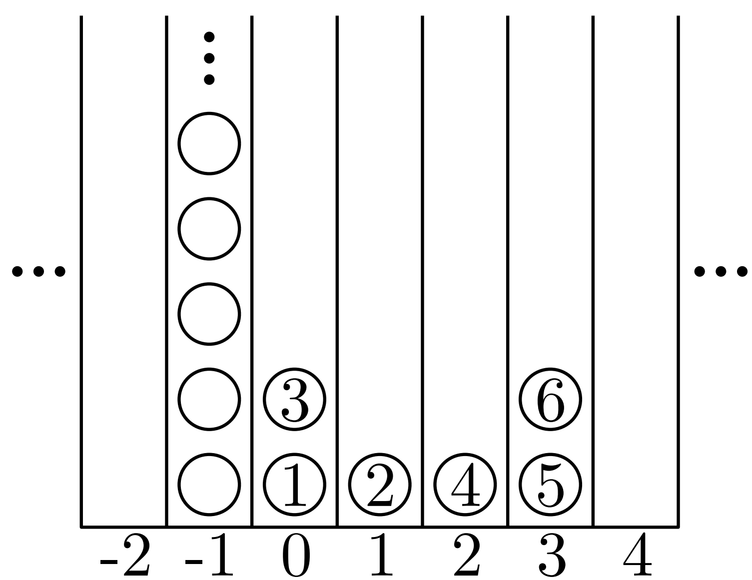
- For each vertex n, call l_n the length of the longest path ending at n. Place a ball with label n in the bin indexed by l_n.



- Grow Barak-Erdős graph one vertex after the other.

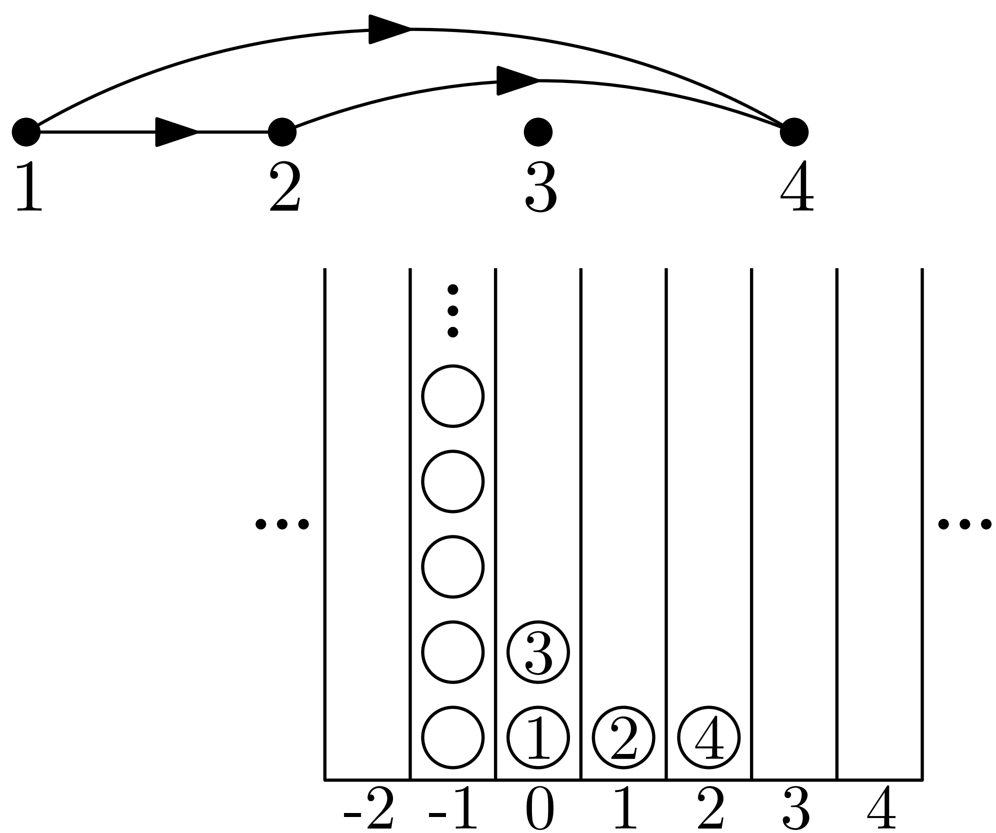
- For each vertex n, call l_n the length of the longest path ending at n. Place a ball with label n in the bin indexed by l_n.



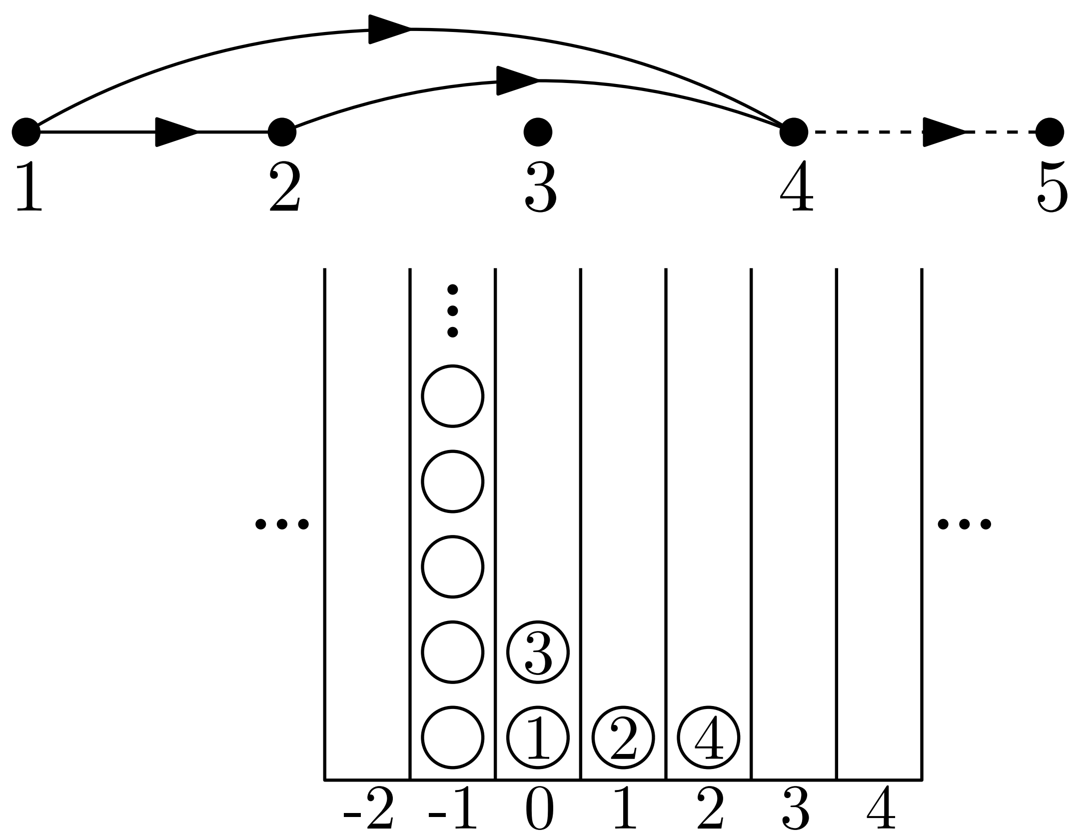


length of longest path in BEG
=
position of front in ball/bin process

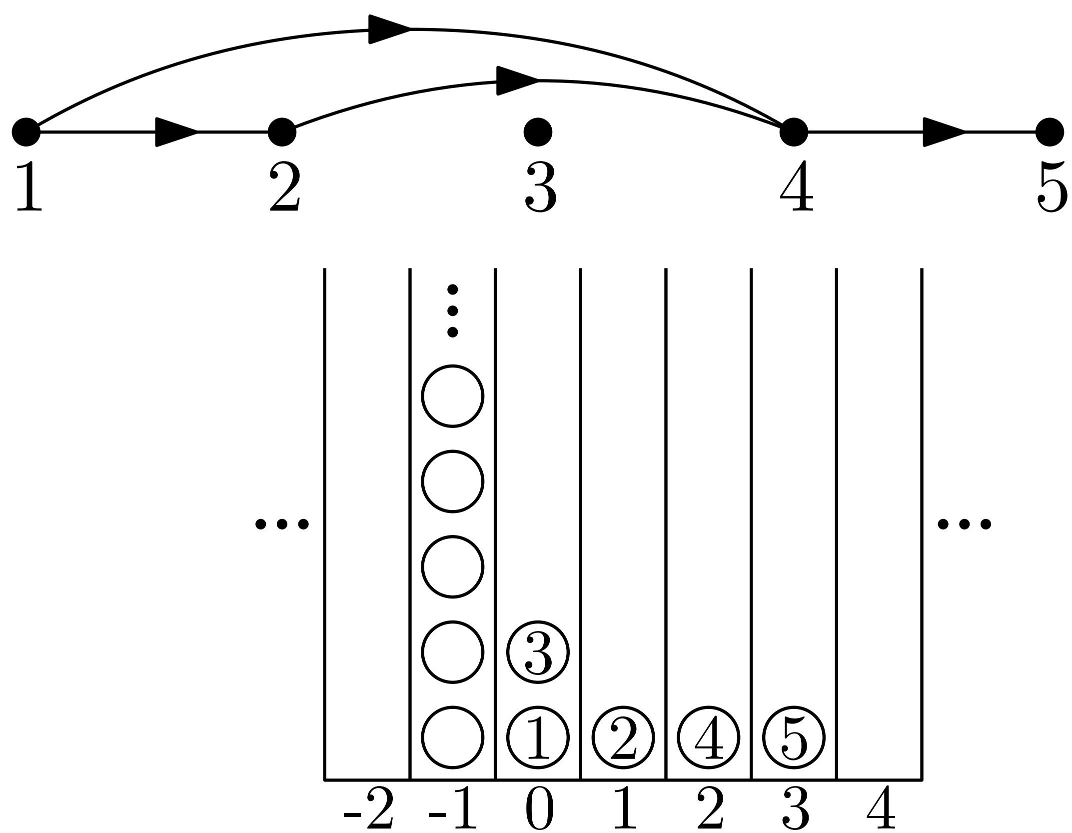
Why is this random evolution of balls in bins an IBM(μ_p) ?



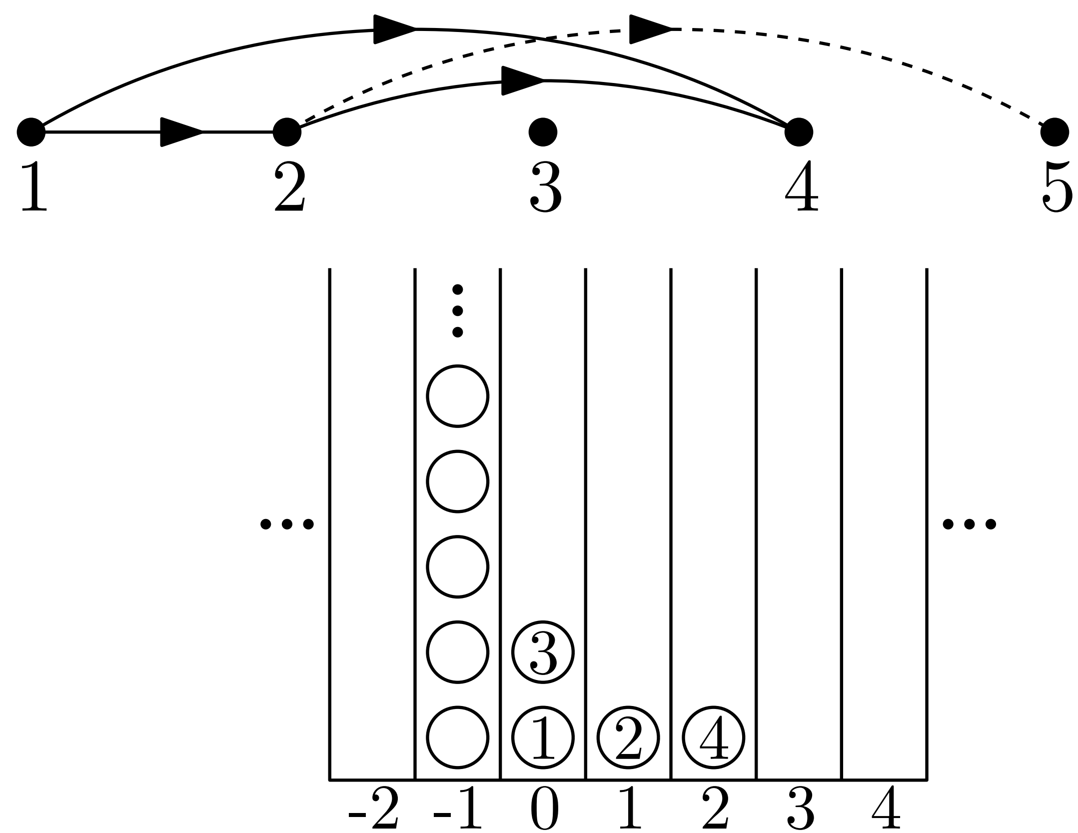
Why is this random evolution of balls in bins an IBM(μ_p) ?



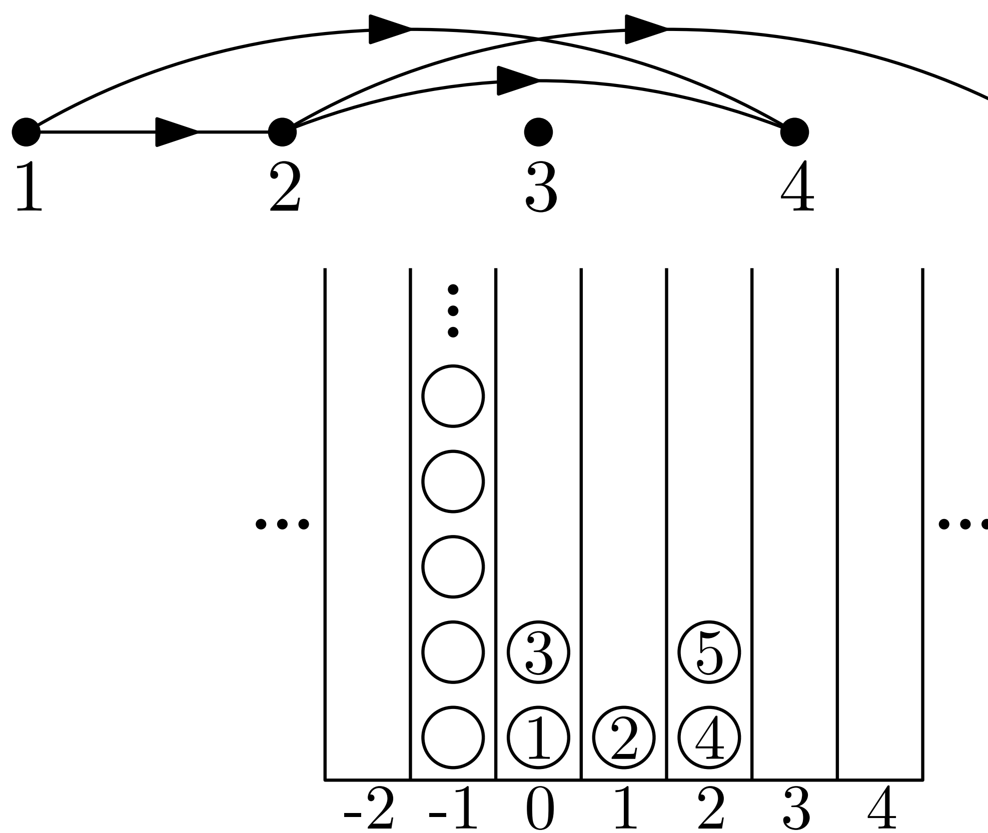
Why is this random evolution of balls in bins an IBM(μ_p) ?



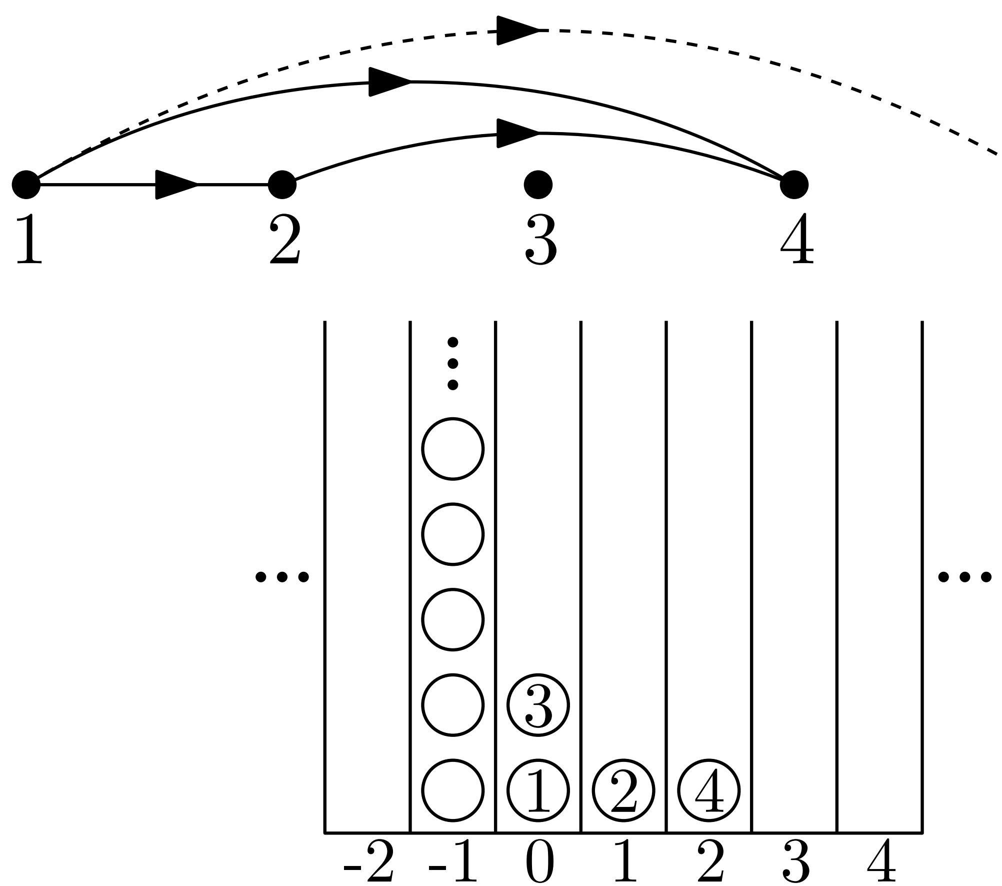
Why is this random evolution of balls in bins an IBM(μ_p) ?



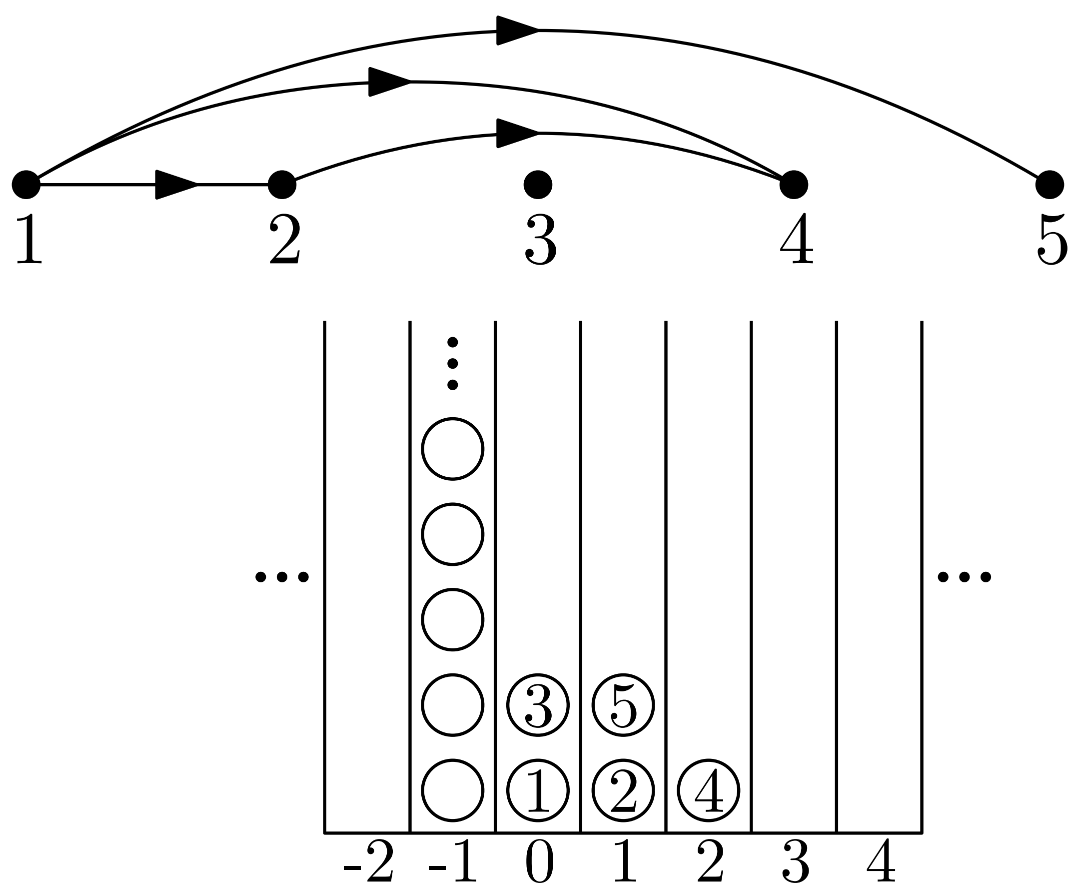
Why is this random evolution of balls in bins an IBM(μ_p) ?



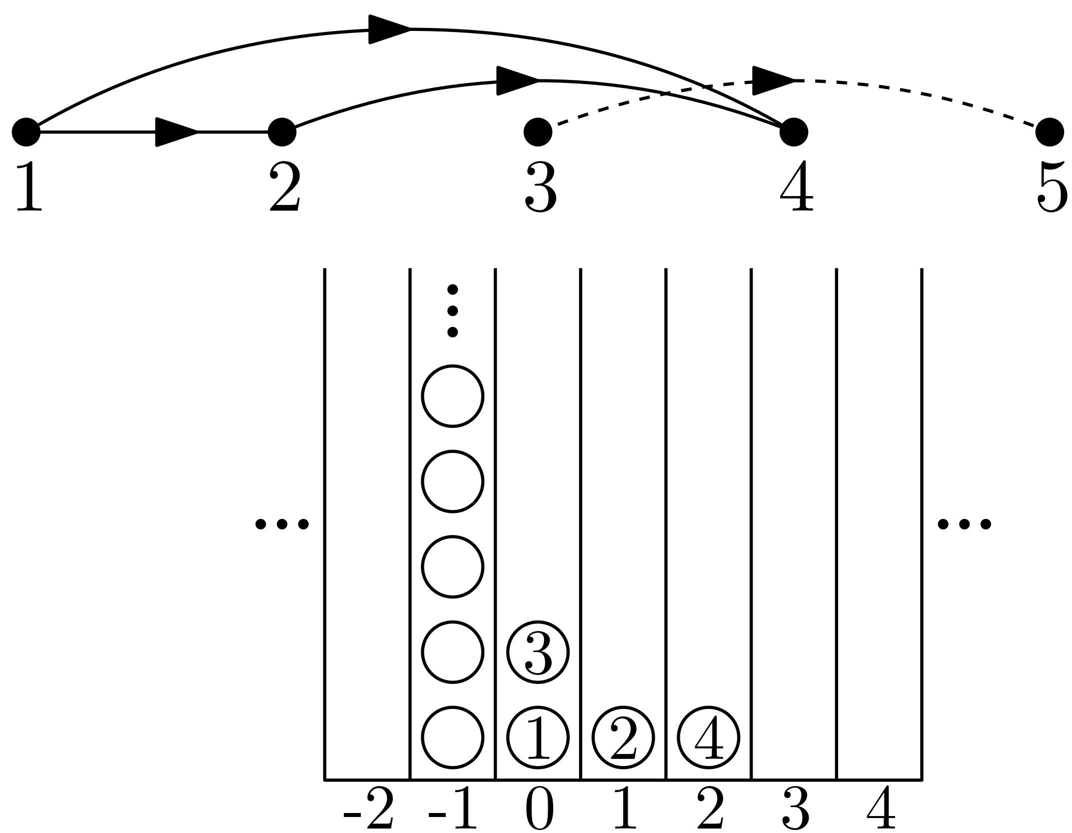
Why is this random evolution of balls in bins an IBM(μ_p) ?



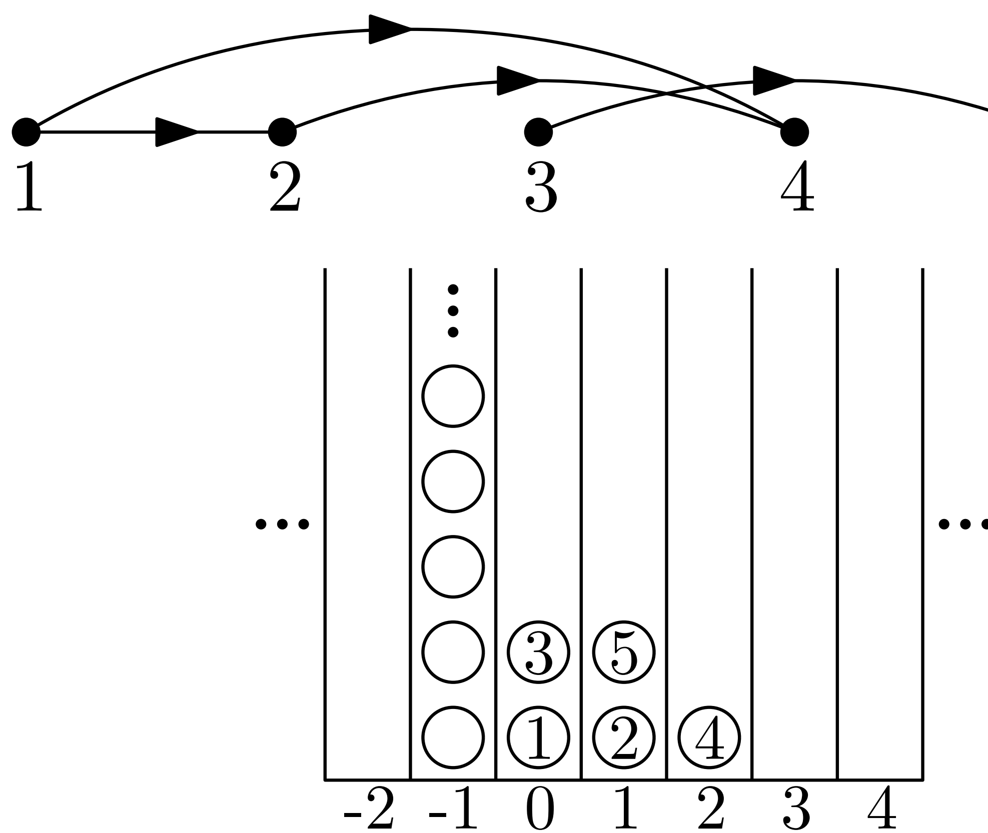
Why is this random evolution of balls in bins an IBM(μ_p) ?



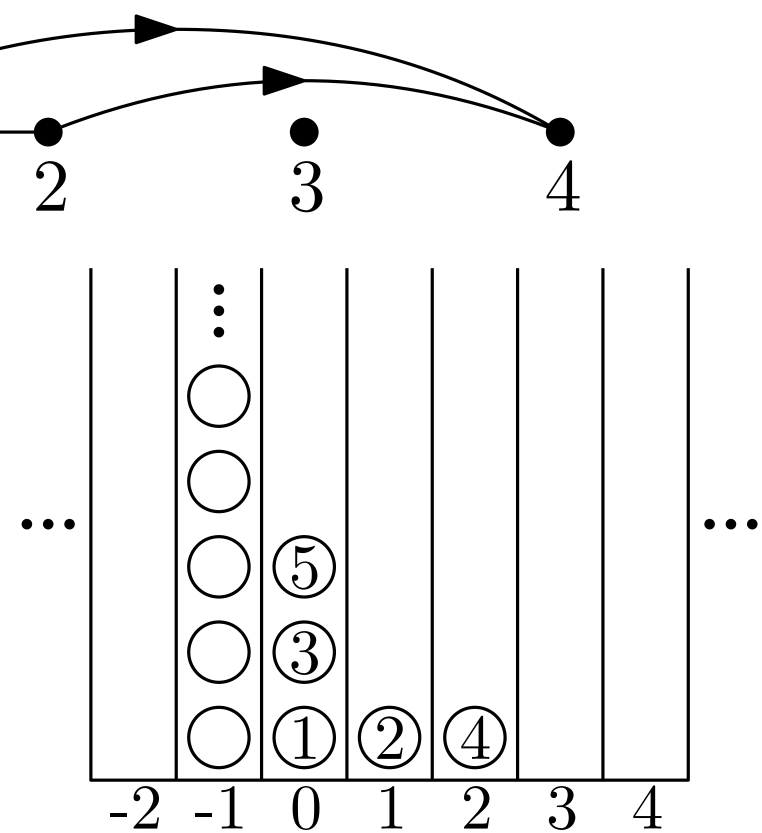
Why is this random evolution of balls in bins an IBM(μ_p) ?



Why is this random evolution of balls in bins an IBM(μ_p) ?



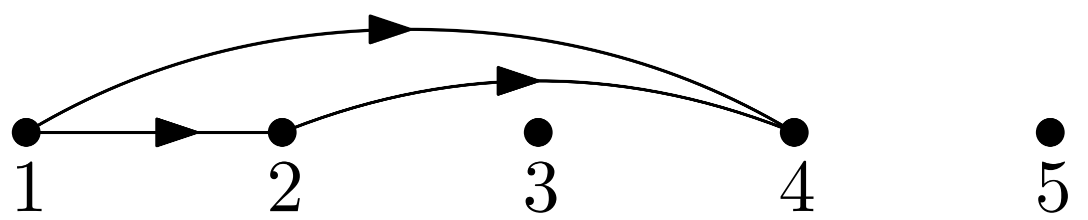
Why is this random evolution of balls in bins an IBM(μ_p) ?

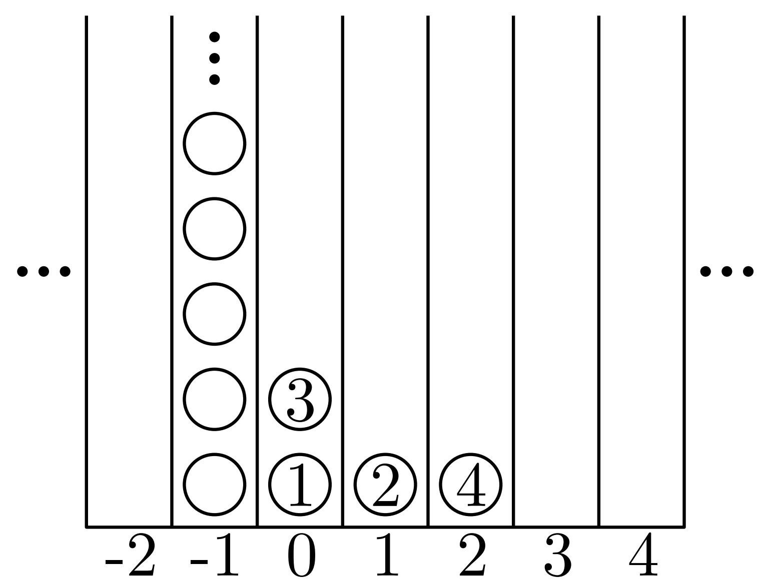


Why is this random evolution of balls in bins an IBM(μ_p) ?

IBM(μ_p): pick a ball whose rank follows the geometric distribution of parameter p and add a ball to its right.

Geometric distribution: number of trials until first success.





Stationary version of the IBM

- How to construct a stationary process, with time indexed by $\mathbb{Z}$ rather than $\mathbb{Z}_+$?

- Given a sequence $(\xi_n)_{n\in\mathbb{Z}}$, we want to construct a process $(X_n)_{n\in\mathbb{Z}}$ satisfying

$$\forall n \in \mathbb{Z}, X_n = \phi_{\xi_n}(X_{n-1}).$$

- Given the value of ξ_n for every $n \leq 0$, one can a.s. reconstruct X_0 up to a global shift (Foss-Konstantopoulos, Mallein-R.).

- Fix $k \geq 1$. A word $(\xi_1, \ldots, \xi_N)$ is called k-coupling if the content of the rightmost k non-empty bins at time N is independent of the configuration at time 0.

- E.g. the word 1 is 1-coupling.

- Fix $k \geq 1$. Looking at the infinite word $(\xi_n)_{n \leq 0}$, go back in time until you find the first suffix which is k-coupling. This will happen a.s. in finite time.

- Define the content of the rightmost k non-empty bins to be that common content. This definition is compatible for different values of k.

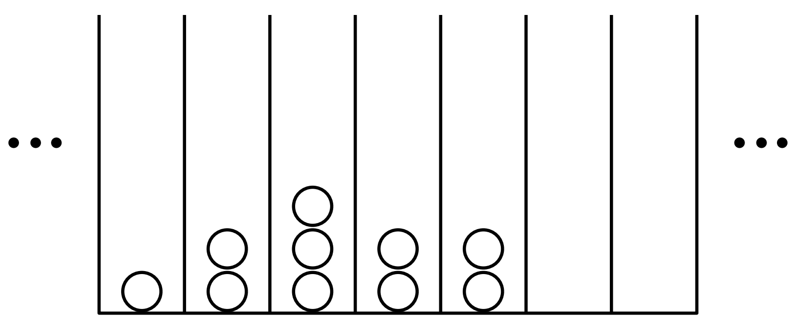
Speed formula for the IBM

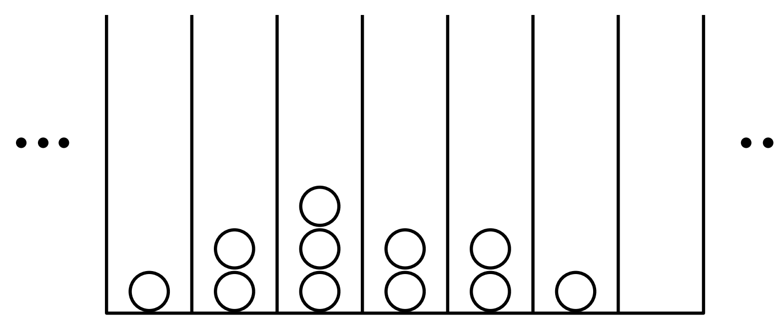
- For the bi-infinite stationary version of the IBM(μ), the speed of the front is the probability that the front advances at time 0.

- Can read this information from the sequence $(\xi_n)_{n \leq 0}$.

- A word $(\xi_1, \ldots, \xi_N)$ is called *good* (resp. *bad*) if it always (resp. never) makes the front advance at time N, regardless of the configuration at time 0.

- 1 is good, 23 is bad, 2 is neither good nor bad.





- A word $(\xi_1, \ldots, \xi_N)$ is called *good* (resp. *bad*) if it always (resp. never) makes the front advance at time N, regardless of the configuration at time 0.

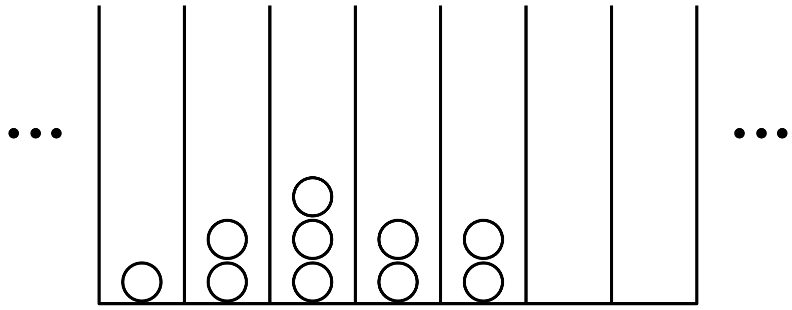
- 1 is good, 23 is bad, 2 is neither good nor bad.

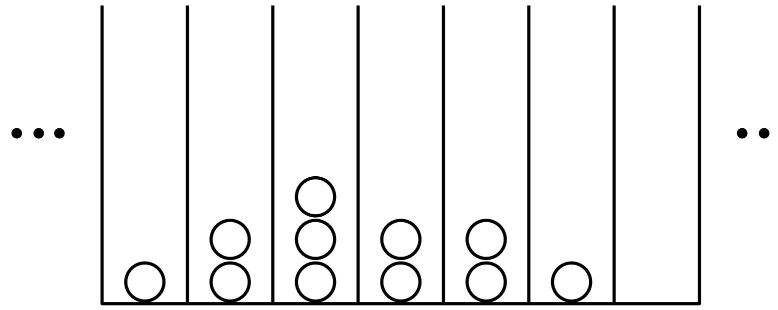
- A word $(\xi_1, \ldots, \xi_N)$ is called *good* (resp. *bad*) if it always (resp. never) makes the front advance at time N, regardless of the configuration at time 0.

- 1 is good, 23 is bad, 2 is neither good nor bad.

- A word $(\xi_1, \ldots, \xi_N)$ is called *good* (resp. *bad*) if it always (resp. never) makes the front advance at time N, regardless of the configuration at time 0.

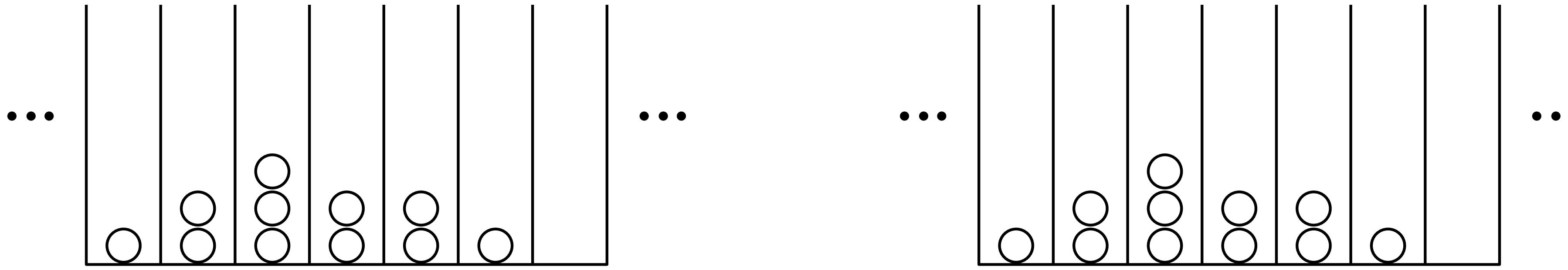
- 1 is good, 23 is bad, 2 is neither good nor bad.





- A word $(\xi_1, \ldots, \xi_N)$ is called *good* (resp. *bad*) if it always (resp. never) makes the front advance at time N, regardless of the configuration at time 0.

- 1 is good, 23 is bad, 2 is neither good nor bad.



- A word $(\xi_1, \ldots, \xi_N)$ is called *good* (resp. *bad*) if it always (resp. never) makes the front advance at time N, regardless of the configuration at time 0.

- 1 is good, 23 is bad, 2 is neither good nor bad.

- A word $(\xi_1, \ldots, \xi_N)$ is called *good* (resp. *bad*) if it always (resp. never) makes the front advance at time N, regardless of the configuration at time 0.

- 1 is good, 23 is bad, 2 is neither good nor bad.

- A word $(\xi_1, \ldots, \xi_N)$ is called *good* (resp. *bad*) if it always (resp. never) makes the front advance at time N, regardless of the configuration at time 0.

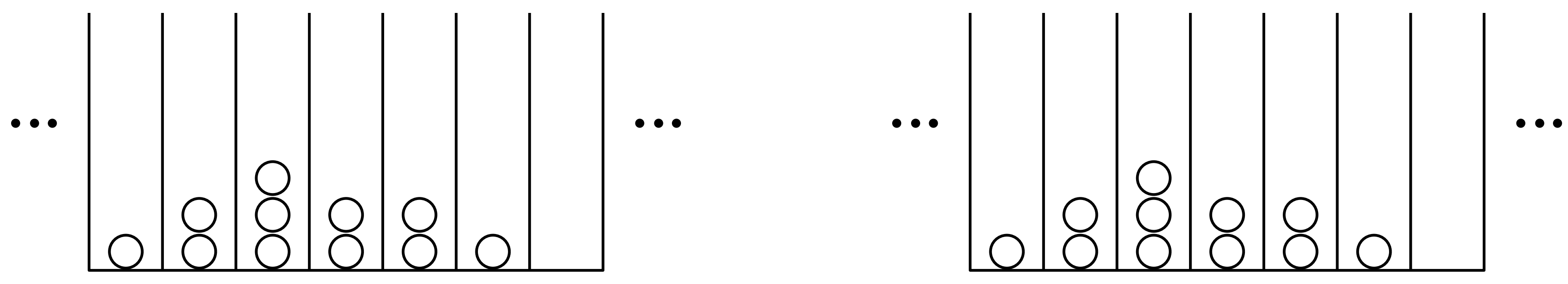
- 1 is good, 23 is bad, 2 is neither good nor bad.

- A word $(\xi_1, \ldots, \xi_N)$ is called *good* (resp. *bad*) if it always (resp. never) makes the front advance at time N, regardless of the configuration at time 0.

- 1 is good, 23 is bad, 2 is neither good nor bad.

- A word $(\xi_1, \ldots, \xi_N)$ is called *good* (resp. *bad*) if it always (resp. never) makes the front advance at time N, regardless of the configuration at time 0.

- 1 is good, 23 is bad, 2 is neither good nor bad.



- A word $(\xi_1, \ldots, \xi_N)$ is called *good* (resp. *bad*) if it always (resp. never) makes the front advance at time N, regardless of the configuration at time 0.

- 1 is good, 23 is bad, 2 is neither good nor bad.

- A word $(\xi_1, \ldots, \xi_N)$ is called *good* (resp. *bad*) if it always (resp. never) makes the front advance at time N, regardless of the configuration at time 0.

- 1 is good, 23 is bad, 2 is neither good nor bad.

- Looking at the infinite word $(\xi_n)_{n \leq 0}$, go back in time until you find the first suffix which is either good or bad. This will happen a.s. in finite time.

- The speed of the front is the probability that this suffix is good.

- Let G_m be the set of good words that have no good strict suffix. 1 is in G_m but 11 is not.

- Given a word α, the weight $w_\mu(\alpha)$ is the product of the probabilities under μ of the letters of α.

  E.g. $w_\mu(2,1,3) = \mu(2) \times \mu(1) \times \mu(3)$.

Theorem (Mallein-R.). *For any μ which is not a Dirac mass, the speed of the IBM(μ) is*

$$v_\mu = \sum_{\alpha \in G_m} w_\mu(\alpha)$$

Perfect simulation

- Sampling exactly from the stationary distribution of a Markov process, unlike MCMC methods which sample from a distribution close to the stationary distribution.

- Perfect simulation is possible for any finite-dimensional marginal of the IBM (Foss-Konstantopoulos, Mallein-R.).

- Work in progress with Foss-Konstantopoulos-Mallein: perfect simulation for LPP on complete graphs with weights 1 and x.

3 Properties of Barak-Erdős graphs via the IBM

Analyticity of $C(p)$

- The special case when μ is μ_p, the geometric distribution of parameter p, gives a formula for $v_{\mu_p} = C(p)$:

$$C(p) = \sum_{\alpha \in G_m} p^{L(\alpha)}(1-p)^{H(\alpha)},$$

  where the height $H(\alpha)$ (resp. length $L(\alpha)$) of a word $\alpha = (\alpha_1, \ldots, \alpha_n)$ is defined to be $\alpha_1 + \cdots + \alpha_n - n$ (resp. n).

- Proving the existence of finite exponential moments for the time one has to wait before discovering a good or bad word implies that $C(p)$ is analytic for $p > 0$.

Power series expansion around $p = 1$

Theorem (Mallein-R.). *$C(1-q)$ can be expanded as a power series in q with radius at least $\frac{\sqrt{2}-1}{2}$ and its coefficients are integers.*

Fix a positive integer h. Then

$$
\begin{aligned}
C(1-q) &= \sum_{\alpha \in G_m} (1-q)^{L(\alpha)} q^{H(\alpha)} \\
&= \sum_{\substack{\alpha \in G_m \\ H(\alpha) \leq h}} (1-q)^{L(\alpha)} q^{H(\alpha)} + \sum_{\substack{\alpha \in G_m \\ H(\alpha) > h}} (1-q)^{L(\alpha)} q^{H(\alpha)}
\end{aligned}
$$

$p \to 0$ limit

Theorem (Mallein-R., 2016)**.**

$$C(p) = ep - \frac{e\pi^2 p}{2(\log p)^2} + o\left(\frac{p}{(\log p)^2}\right) \text{ when } p \to 0.$$

In particular, $C(p)$ has no second derivative at $p = 0$.

Proof strategy

1. Compare to an IBM(μ) where μ is the uniform distribution on $\{1, \ldots, n\}$ and n is large.
2. The IBM with uniform distribution is coupled with a branching random walk with selection.
3. Use known estimates on branching random walks.

Step 1 : reduction to uniform case

- Want the speed of the IBM(μ_p), where μ_p is geometric with parameter p small.

- If $p = \frac{1}{n}$:

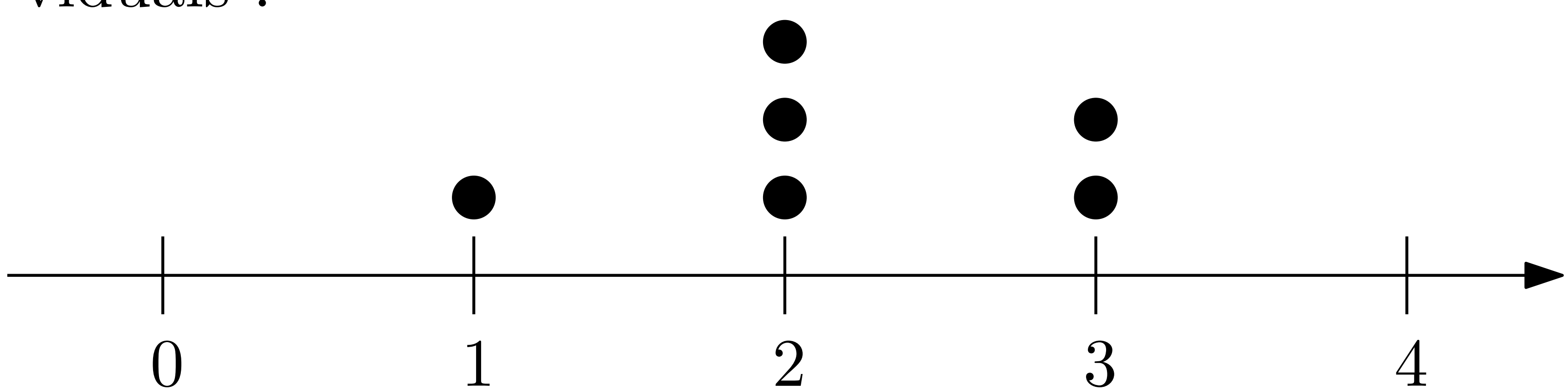
k	1	2	3		n	
$\mu_p(k)$	$\frac{1}{n}$	$\frac{1}{n}\left(1-\frac{1}{n}\right)$	$\frac{1}{n}\left(1-\frac{1}{n}\right)^2$	$\cdots$	$\frac{1}{n}\left(1-\frac{1}{n}\right)^{n-1}$	$\cdots$

- Roughly equal to ν_n, the uniform distribution on $\{1,\ldots,n\}$.

Step 2 : branching random walk analogy

- IBM(ν_n) first studied by Aldous and Pitman in 1983 : speed behaves like $\frac{e}{n}$ when $n \to \infty$.

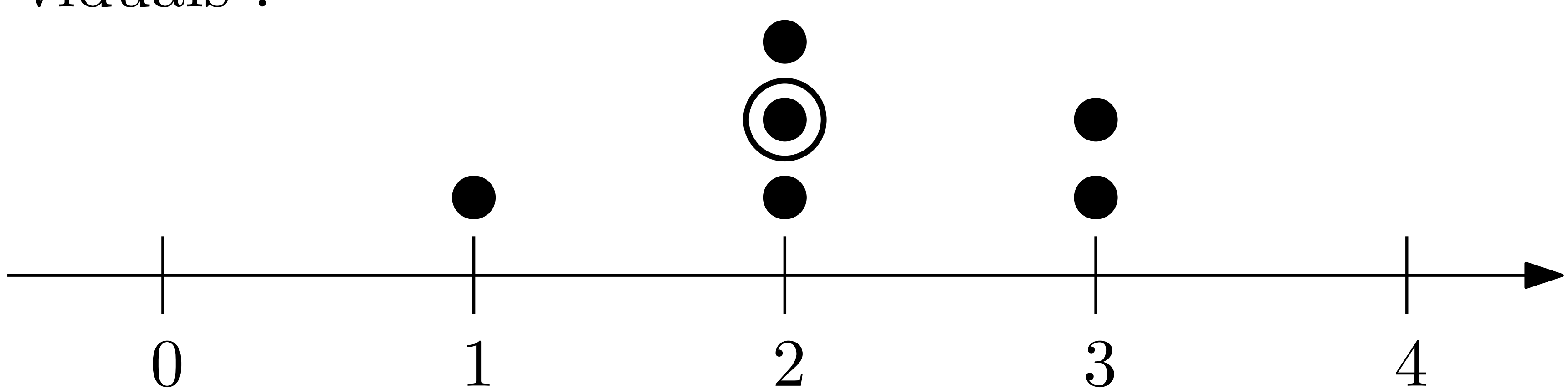
- Use a coupling with a continuous-time branching random walk on $\mathbb{Z}$ with selection of the rightmost n individuals :



Step 2 : branching random walk analogy

- IBM(ν_n) first studied by Aldous and Pitman in 1983 : speed behaves like $\frac{e}{n}$ when $n \to \infty$.

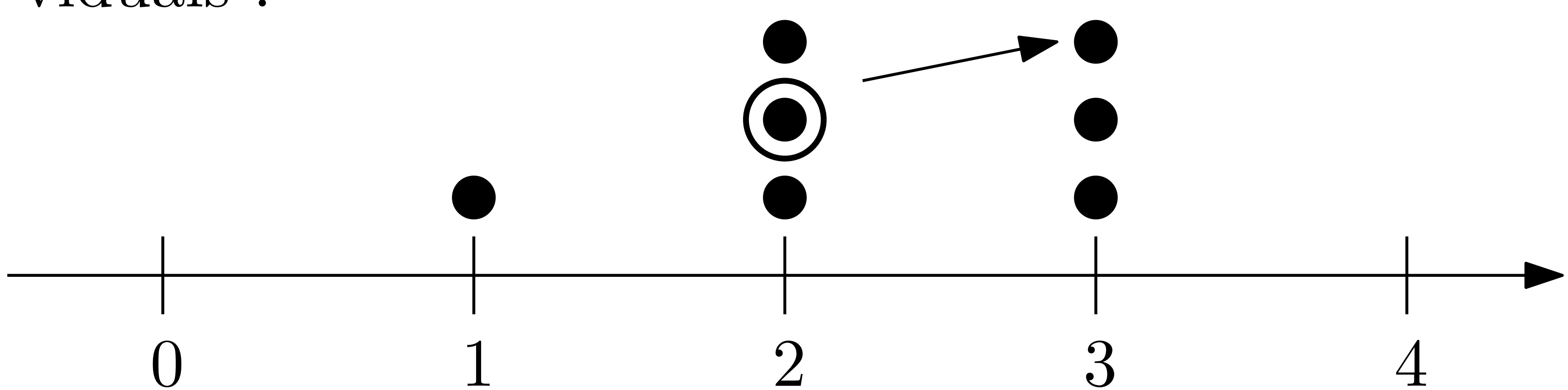
- Use a coupling with a continuous-time branching random walk on $\mathbb{Z}$ with selection of the rightmost n individuals :



Step 2 : branching random walk analogy

- IBM(ν_n) first studied by Aldous and Pitman in 1983 : speed behaves like $\frac{e}{n}$ when $n \to \infty$.

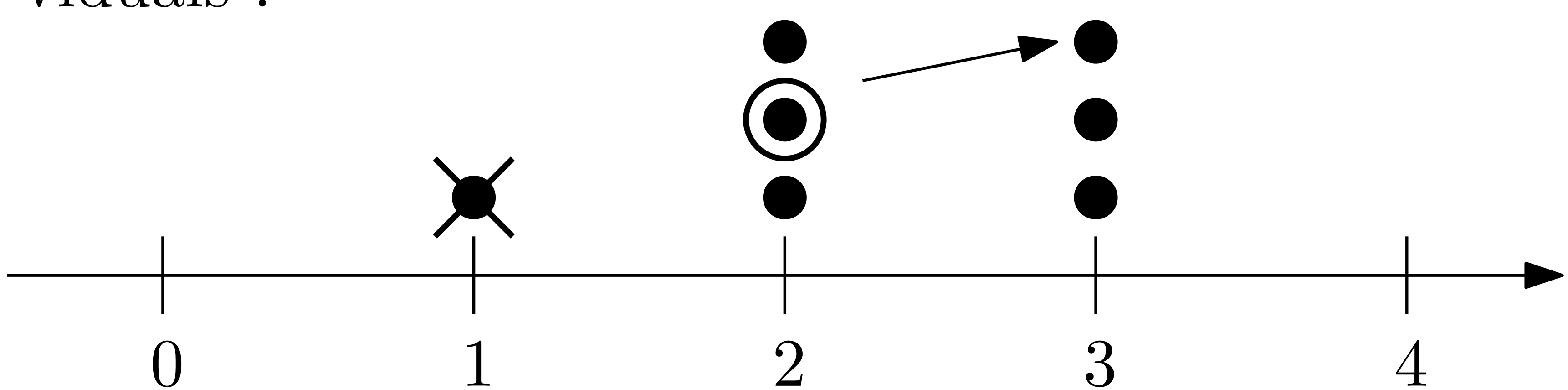
- Use a coupling with a continuous-time branching random walk on $\mathbb{Z}$ with selection of the rightmost n individuals :



Step 2 : branching random walk analogy

- IBM(ν_n) first studied by Aldous and Pitman in 1983 : speed behaves like $\frac{e}{n}$ when $n \to \infty$.

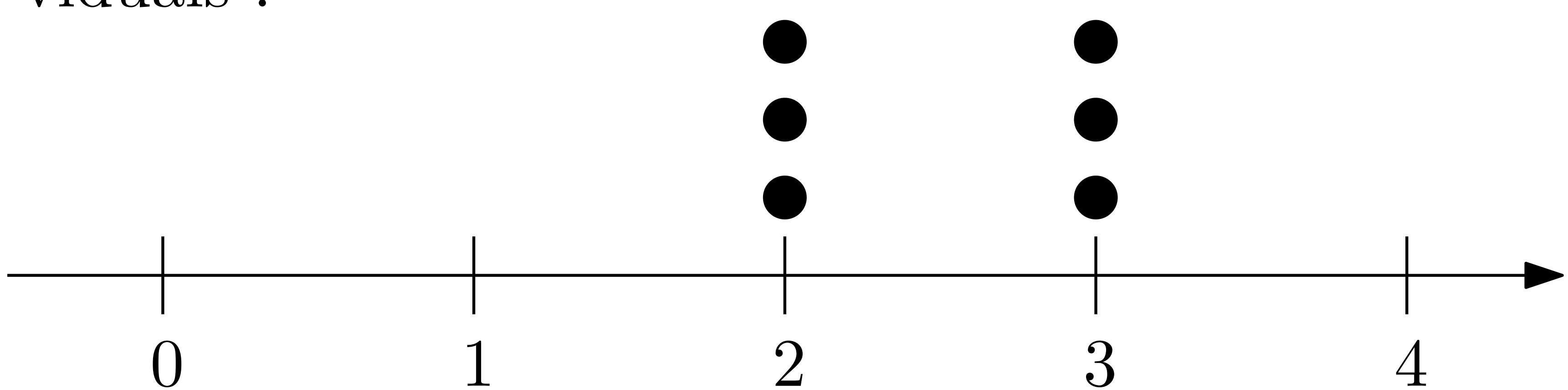
- Use a coupling with a continuous-time branching random walk on $\mathbb{Z}$ with selection of the rightmost n individuals :



Step 2 : branching random walk analogy

- IBM(ν_n) first studied by Aldous and Pitman in 1983 : speed behaves like $\frac{e}{n}$ when $n \to \infty$.

- Use a coupling with a continuous-time branching random walk on $\mathbb{Z}$ with selection of the rightmost n individuals :



Step 3 : use branching random walk estimates

- Discrete-time branching random walk with selection widely studied : Brunet-Derrida '97, Bérard-Gouéré '10, Bérard-Maillard '14, Mallein '15...
- Translate these discrete-time results to the continuous setting.
- Conjecture for next term (after Brunet-Derrida) :

$$\frac{3e\pi^2 p \log(-\log p)}{(-\log p)^3}$$

4 Perspectives

Integer coefficients around $p = 1$

$$\begin{aligned} C(1-q) &= 1 - q + q^2 - 3q^3 + 7q^4 - 15q^5 + \cdots \\ &= \sum_{k \geq 0} (-1)^k a_k q^k. \end{aligned}$$

Foss-Konstantopoulos ('03): first 5 terms
Mallein-R. ('16): first 17 terms
Terlat ('21): first 24 terms

Was not in the Online Encyclopedia of Integer Sequences before I added it (sequence A321309).

- Show that (a_k) forms an increasing sequence of positive integers and find a class of objects counted by them.

- Is there a simple direct proof of all the results on $C(p)$ using just graphs and not the IBM?

- Can one find some more or less explicit formula for $C(p)$?

If u_k is an integer sequence, write its generating function as

$$F(q) = \sum_{k \geq 0} u_k q^k.$$

$F(q)$ is *rational* if it is a quotient of two polynomials in q.

$F(q)$ is *algebraic* if it satisfies $G(q, F(q)) = 0$ for some bivariate polynomial G.

$F(q)$ is *D-finite* if it satisfies $H(q, F(q), F'(q), \cdots, F^{(m)}(q)) = 0$ for some H which is polynomial in its first variable and linear in its other variables.

$F(q)$ is *D-algebraic* if it satisfies a similar equation with H polynomial in every variable.

- Because of the singularity in $p/(\log p)^2$ we found around $p = 0$, $C(p)$ cannot be neither rational, nor algebraic, nor D-finite.

- Is $C(p)$ D-algebraic ?

- Maple package called *gfun* developped by Salvy: if you enter “enough” terms of the sequence u_k, it will guess the generating function if it falls in one of the four categories.

THANK YOU !