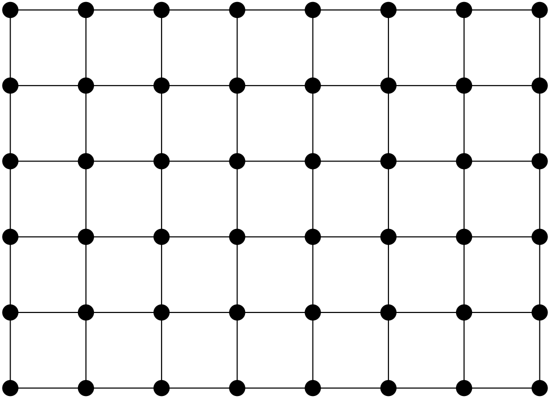
Wilson's algorithm for the Uniform Spanning Tree

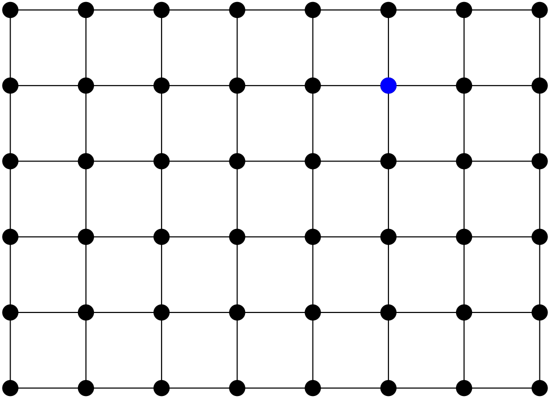
March 15, 2016

Random arrows



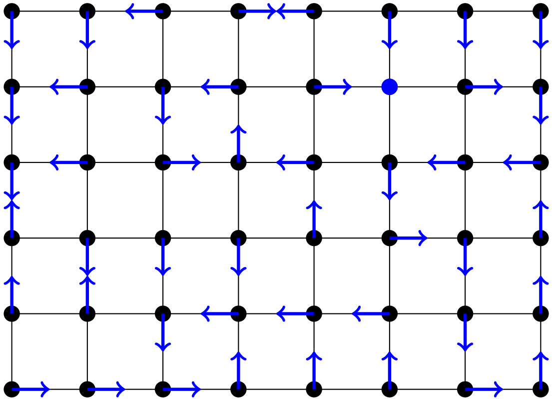
We start with a graph.

Random arrows



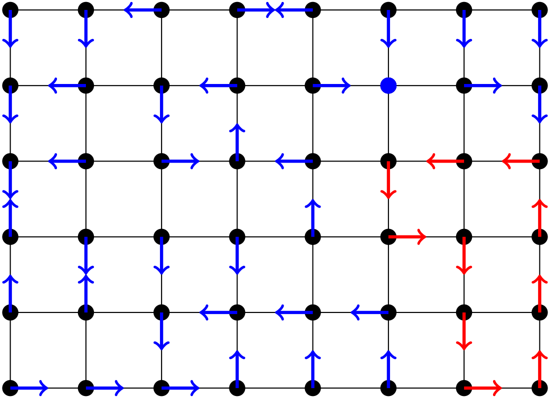
We pick a root.

Random arrows



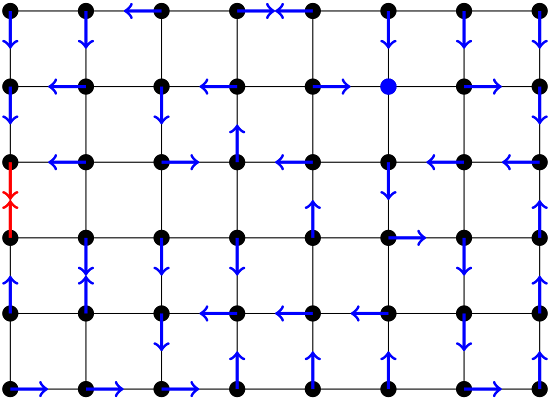
At each vertex other than the root, we draw an outgoing arrow according to an irreducible Markov kernel P.

Random arrows



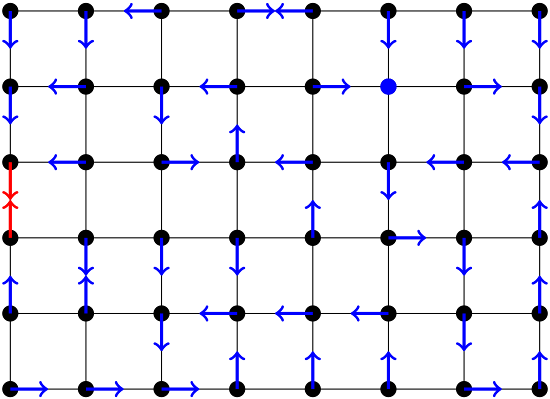
The resulting oriented subgraph may contain cycles.

Random arrows



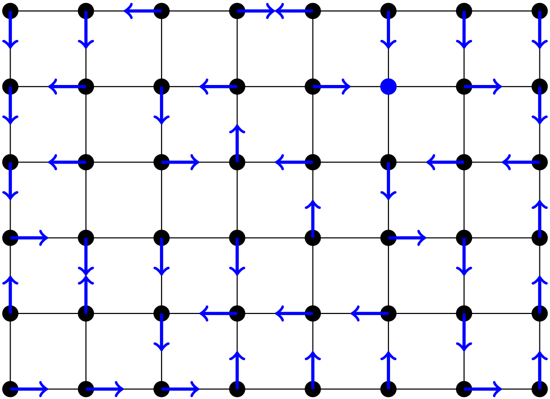
Including very short cycles.

Random arrows



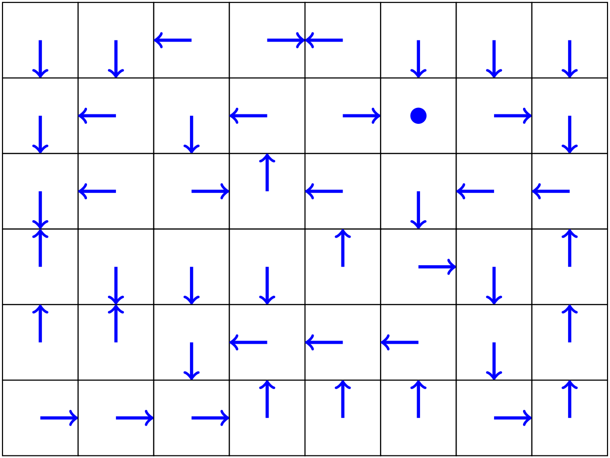
For each cycle, we redraw the arrows.

Random arrows



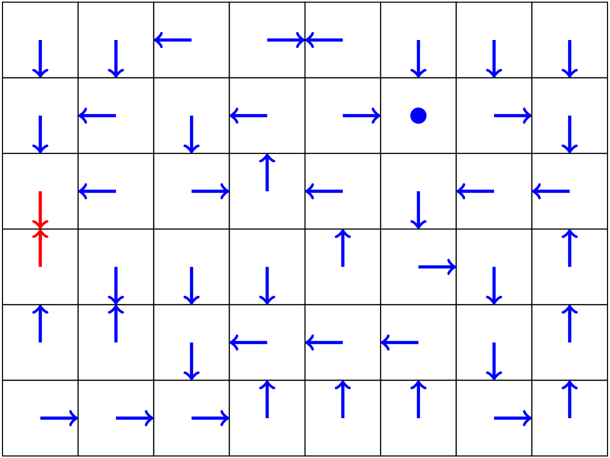
For each cycle, we redraw the arrows.

Random arrows



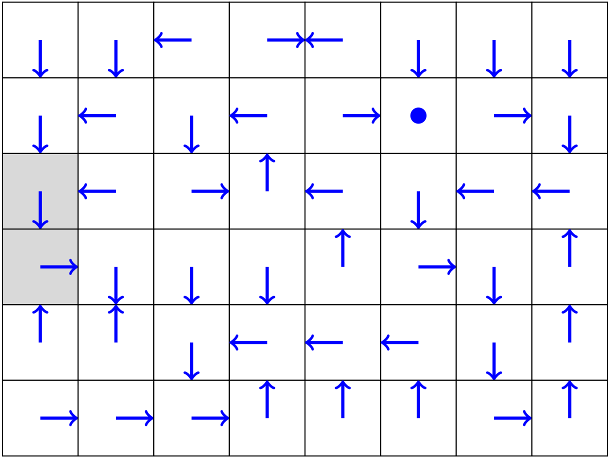
We can imagine that initially, at each vertex, there is an infinite pile of cards.

Random arrows



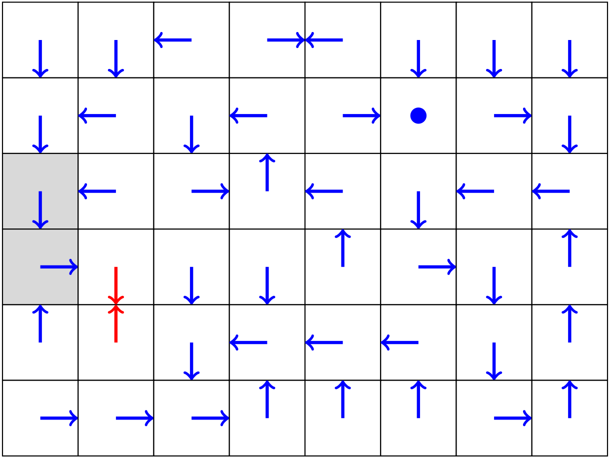
When we redraw a cycle. . .

Random arrows



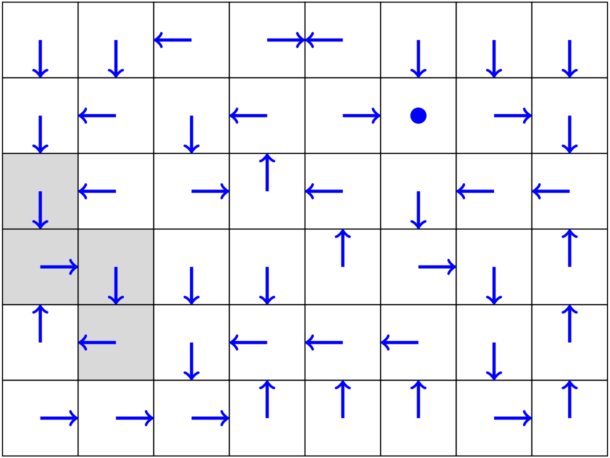
. . . we uncover the cards that lie below it.

Random arrows



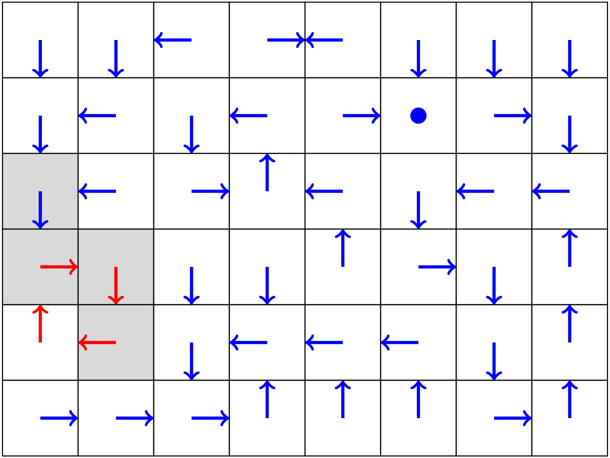
We continue the process of "cycle popping".

Random arrows



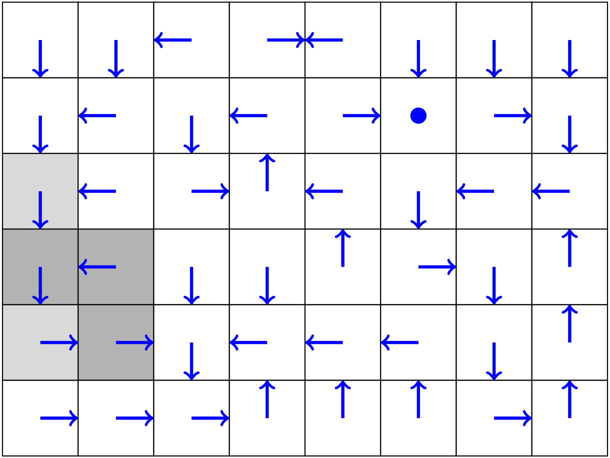
We continue the process of “cycle popping”.

Random arrows



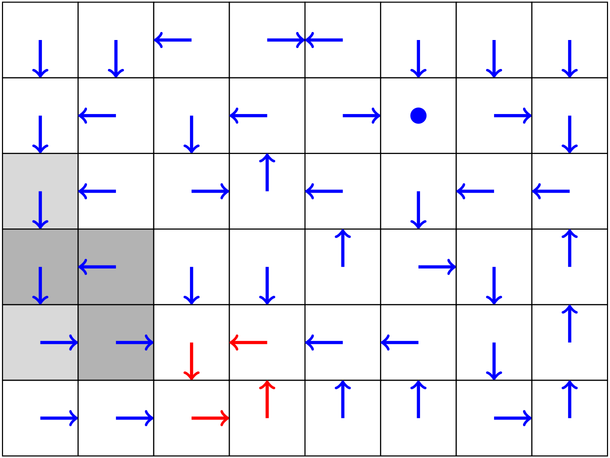
We continue the process of “cycle popping”.

Random arrows



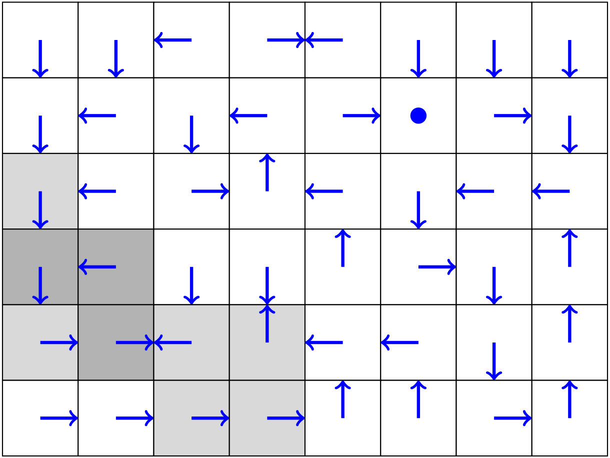
We continue the process of “cycle popping”.

Random arrows

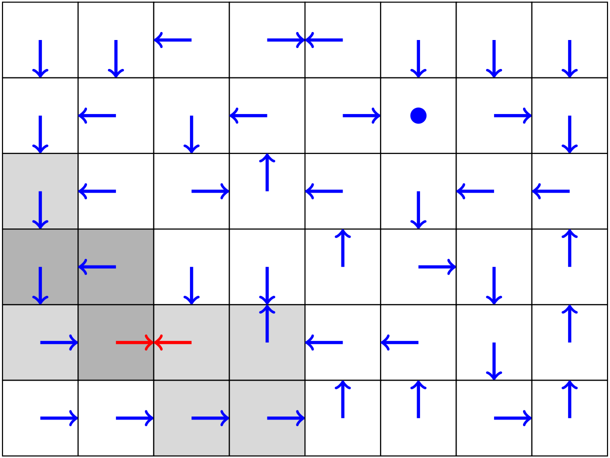


We continue the process of “cycle popping”.

Random arrows

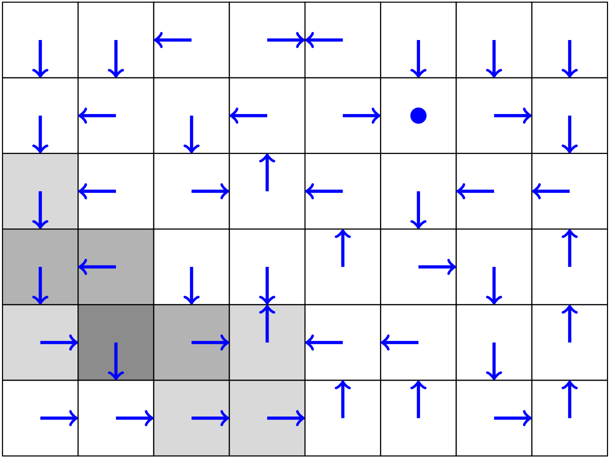


We continue the process of “cycle popping”.



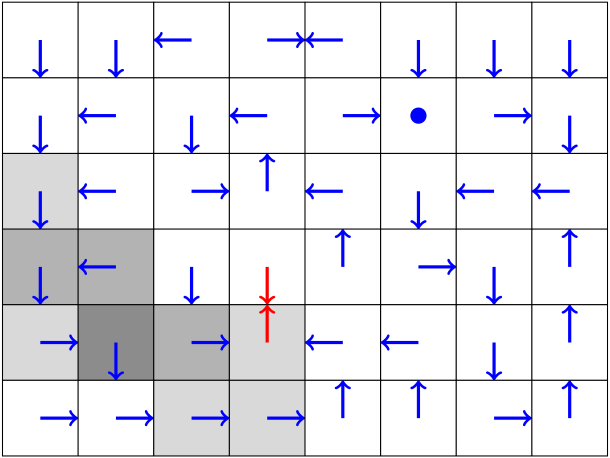
We continue the process of “cycle popping”.

Random arrows



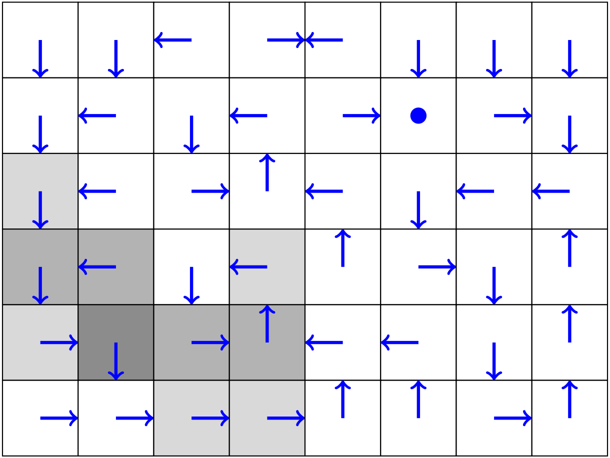
We continue the process of “cycle popping”.

Random arrows



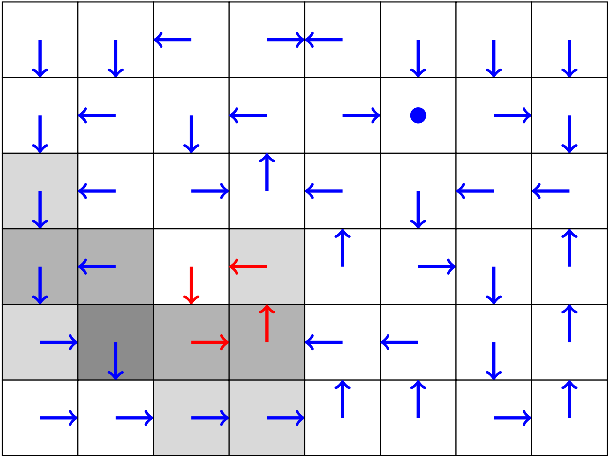
We continue the process of “cycle popping”.

Random arrows



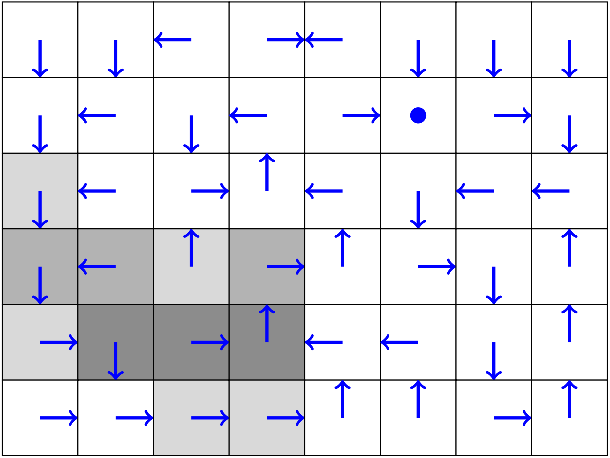
We continue the process of “cycle popping”.

Random arrows



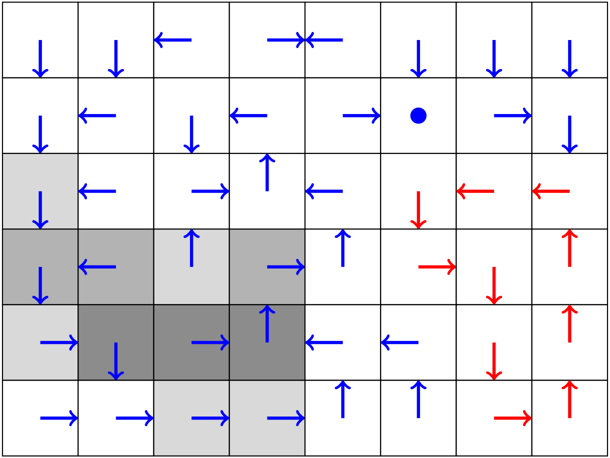
We continue the process of “cycle popping”.

Random arrows



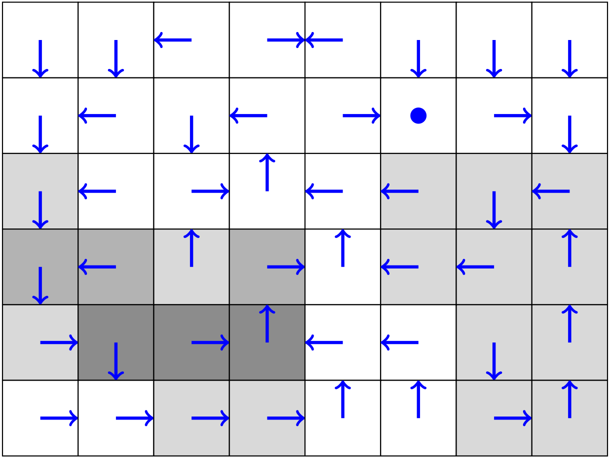
We continue the process of "cycle popping".

Random arrows



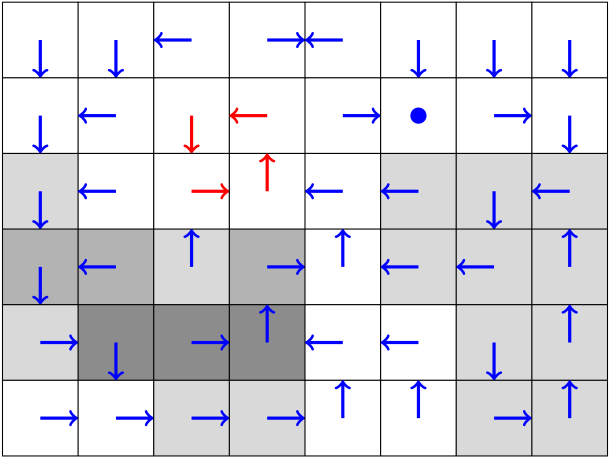
We continue the process of “cycle popping”.

Random arrows



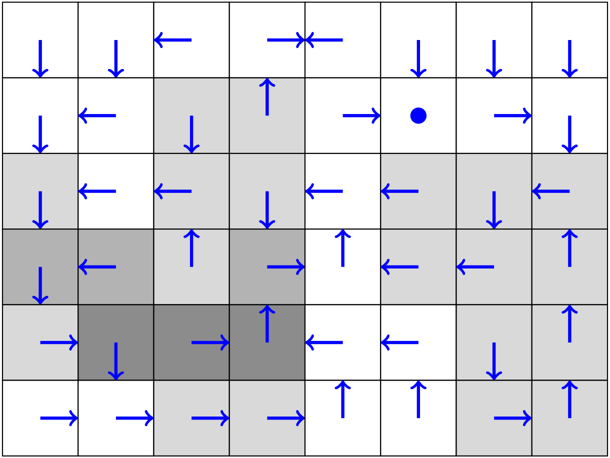
We continue the process of “cycle popping”.

Random arrows



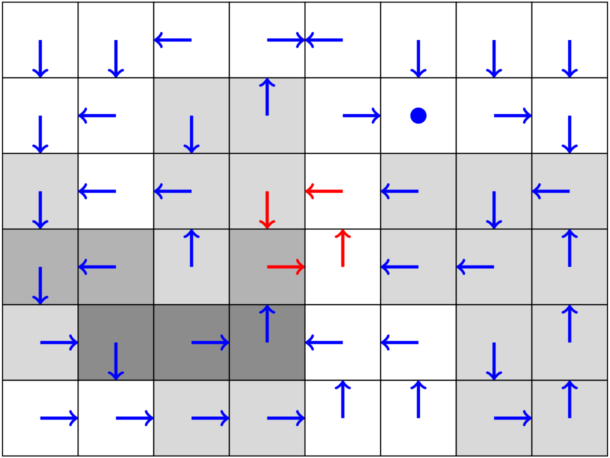
We continue the process of “cycle popping”.

Random arrows



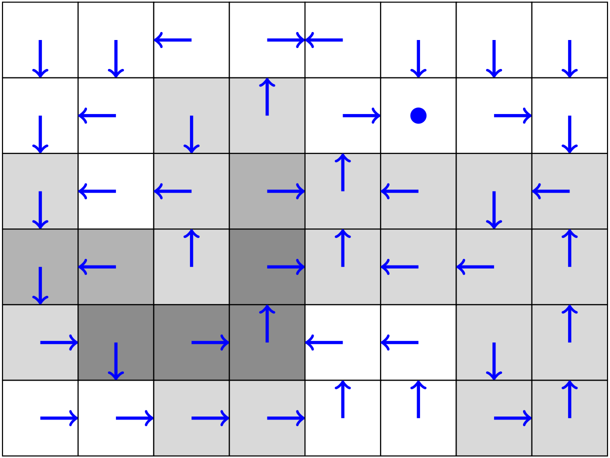
We continue the process of "cycle popping".

Random arrows



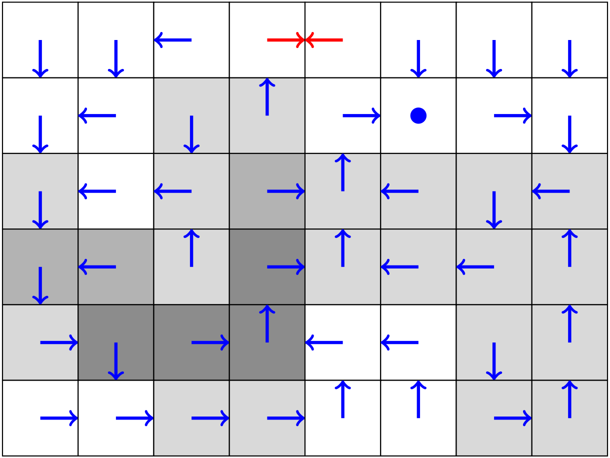
We continue the process of "cycle popping".

Random arrows



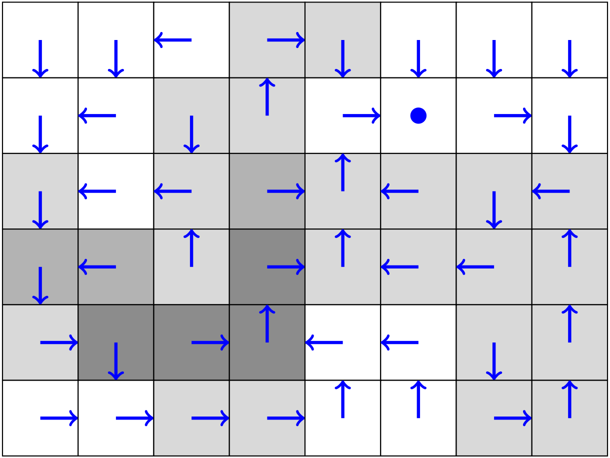
We continue the process of “cycle popping”.

Random arrows



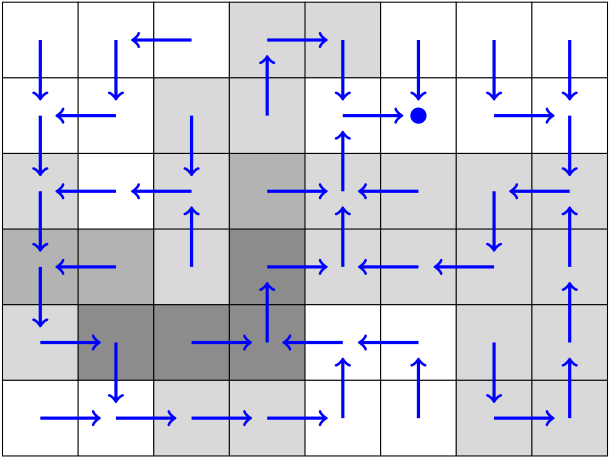
We continue the process of “cycle popping”.

Random arrows



Until there are no cycles left.

Random arrows



The resulting directed tree is called an *arborescence*.

Wilson's theorem

Theorem

1. If the process of cycle popping ends, then at the end, always the same cycles of cards have been popped.

2. The process of cycle popping ends a.s.

3. The resulting dircted graph has the same distribution as the original configuration of arrows conditioned on being an arborescence.

Wilson's theorem

For given piles of cards, call a cycle C of cards *poppable* if it is possible to pop cycles $C_1, \ldots, C_n$ such that $C_n = C$.

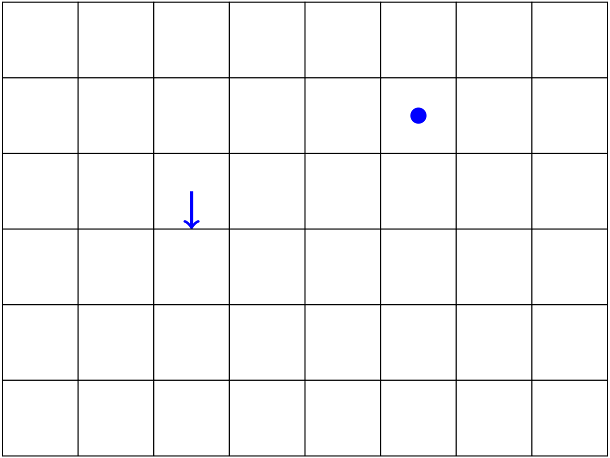
Claim Let C be a poppable cycle and let C' be a cycle that lies on top. Then either $C = C'$, or they are disjoint and C is still poppable after removing C'.

Proof The claim is trivial if C' is disjoint from $C_1, \ldots, C_n$. Otherwise, let C_k be the first cycle that intersects C'. We will prove that $C_k = C'$, so either $k = n$ and $C = C'$ or we can pop the cycles in the order $C_k, C_1, \ldots, C_{k-1}, C_{k+1}, \ldots, C_n$, proving that C is still poppable after removing $C' = C_k$.

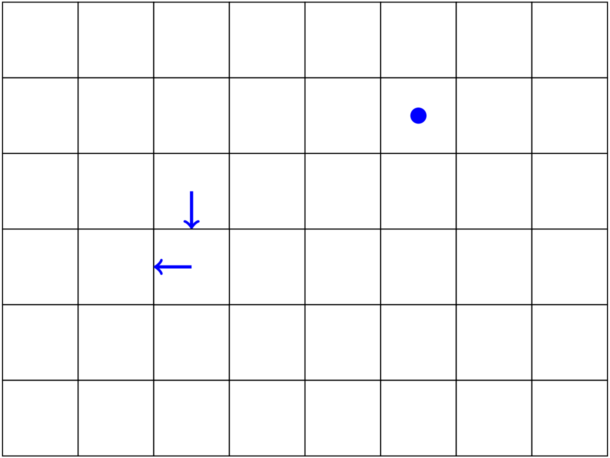
Proof that $C_k = C'$ After popping $C_1, \ldots, C_{k-1}$, the cycles C_k and C' both lie on top. But two cycles on top that intersect somewhere must be equal.

Proof of 1. The process of popping cycles does not end as long as there are still poppable cycles. We have just proved that after popping one cycle, all other poppable cycles are still poppable. So if the process of cycle popping ends, then at the end, always the same cycles of cards have been popped.

Wilson's theorem

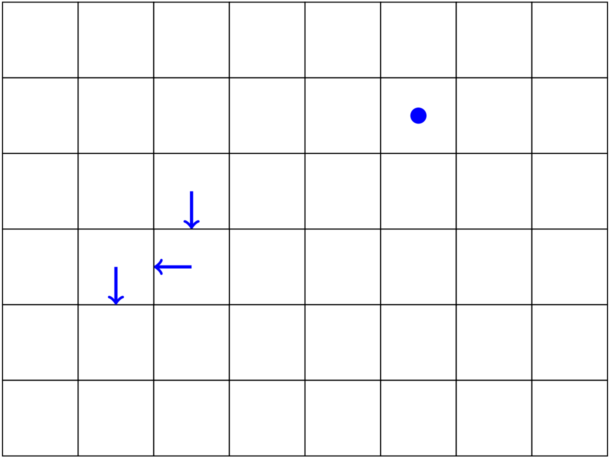


To prove that cycle popping ends a.s., we start a random walk somewhere.



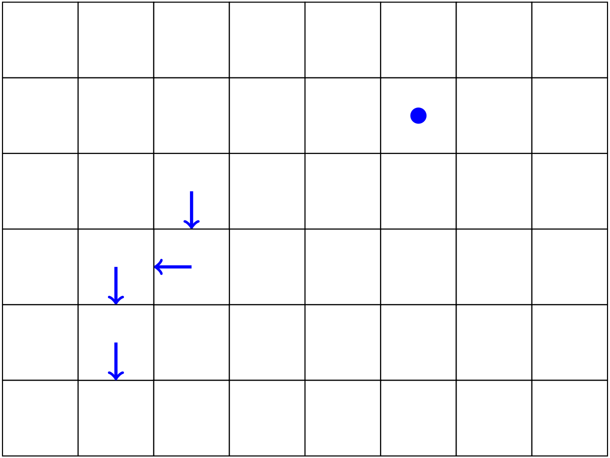
To prove that cycle popping ends a.s., we start a random walk somewhere.

Wilson's theorem



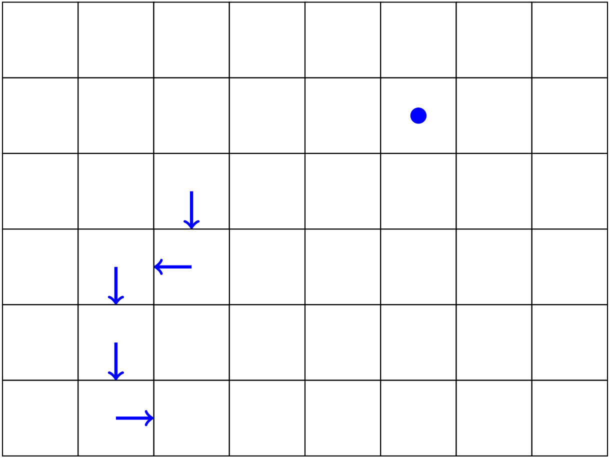
To prove that cycle popping ends a.s., we start a random walk somewhere.

Wilson's theorem



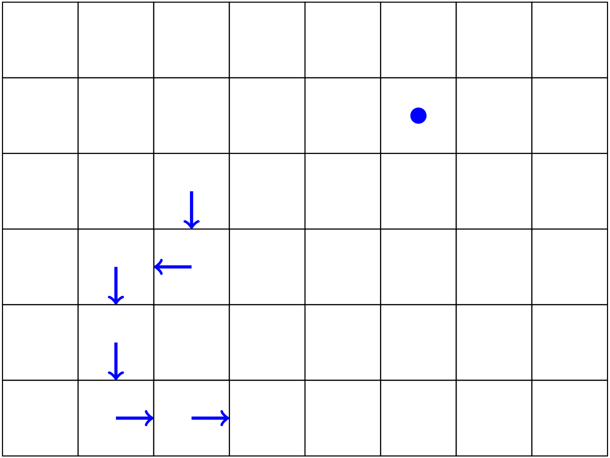
To prove that cycle popping ends a.s., we start a random walk somewhere.

Wilson's theorem



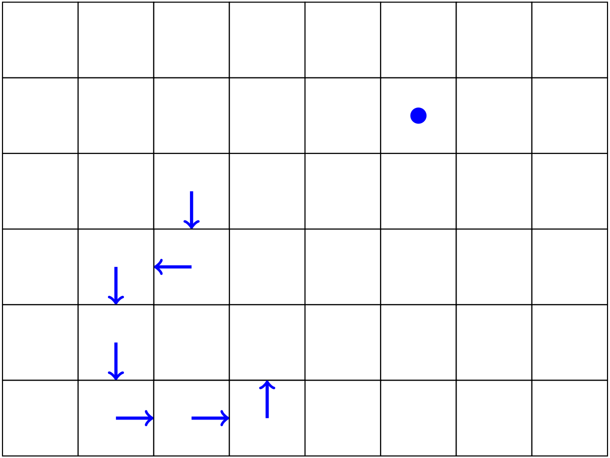
To prove that cycle popping ends a.s., we start a random walk somewhere.

Wilson's theorem

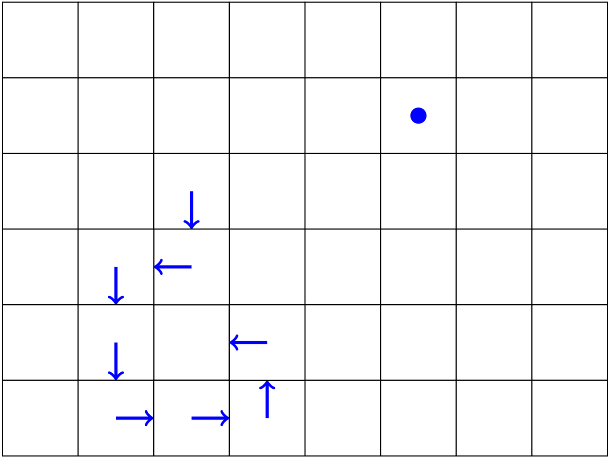


To prove that cycle popping ends a.s., we start a random walk somewhere.

Wilson's theorem

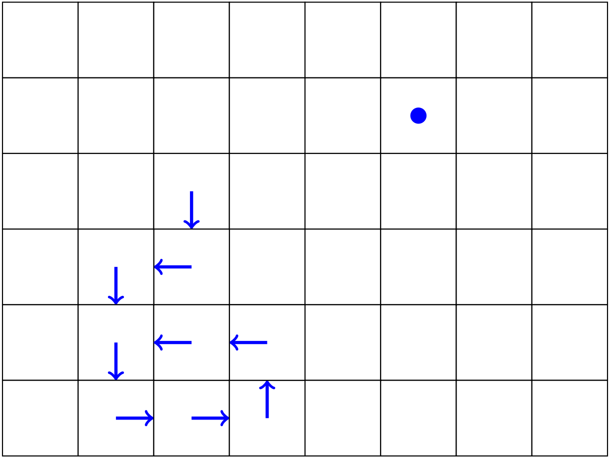


To prove that cycle popping ends a.s., we start a random walk somewhere.



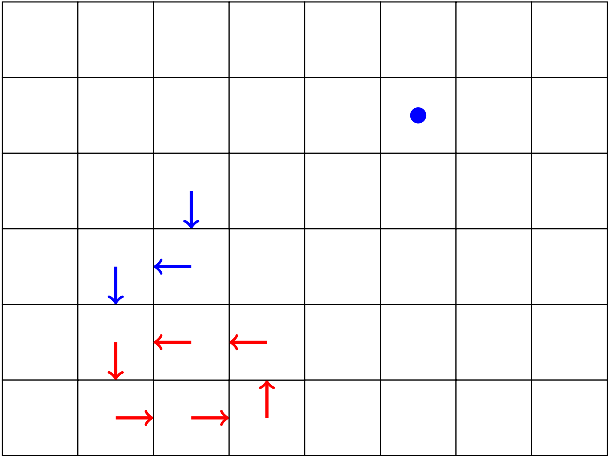
To prove that cycle popping ends a.s., we start a random walk somewhere.

Wilson's theorem



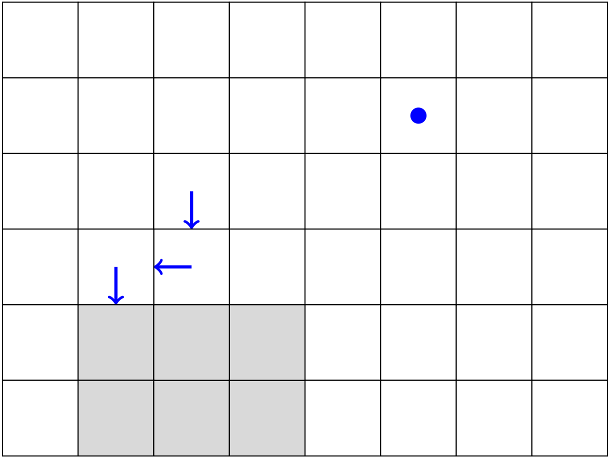
To prove that cycle popping ends a.s., we start a random walk somewhere.

Wilson's theorem



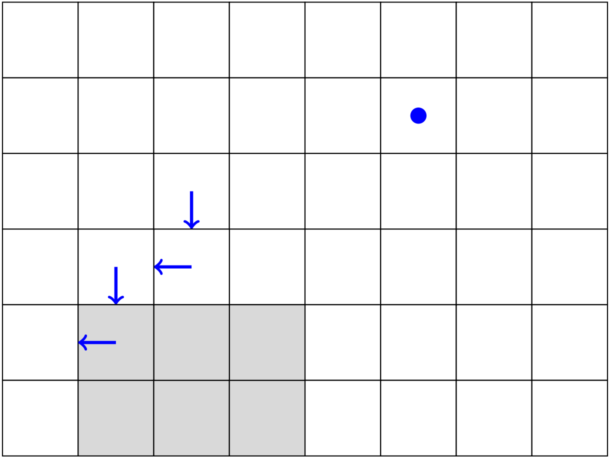
Once a cycle forms...

Wilson's theorem



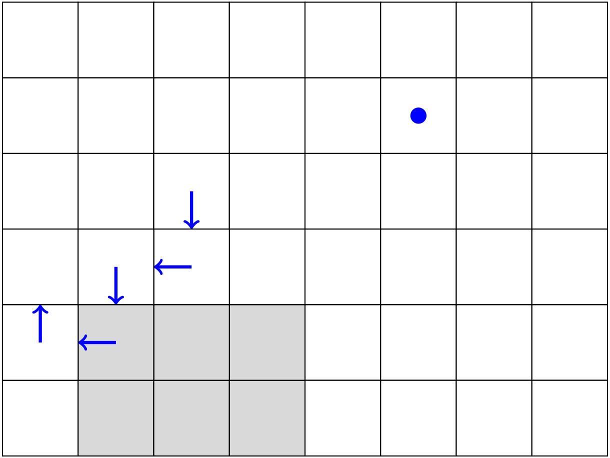
...we remove it...

Wilson's theorem



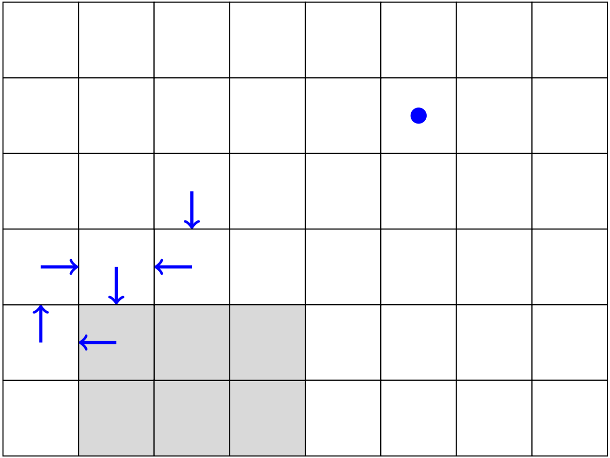
. . . and continue.

Wilson's theorem



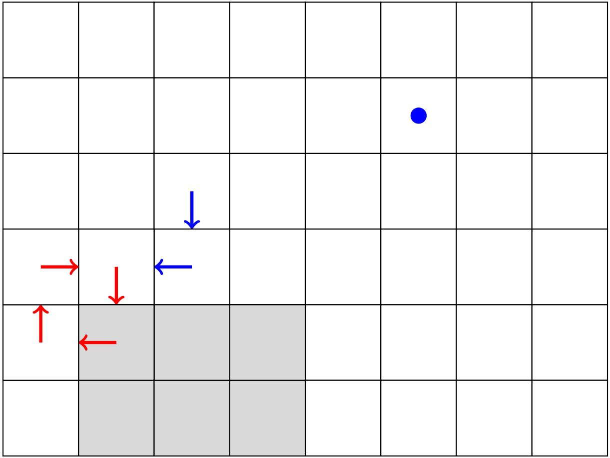
...and continue.

Wilson's theorem



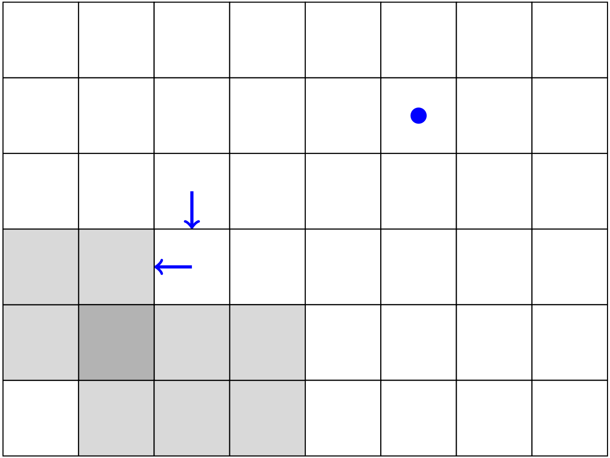
. . . and continue.

Wilson's theorem



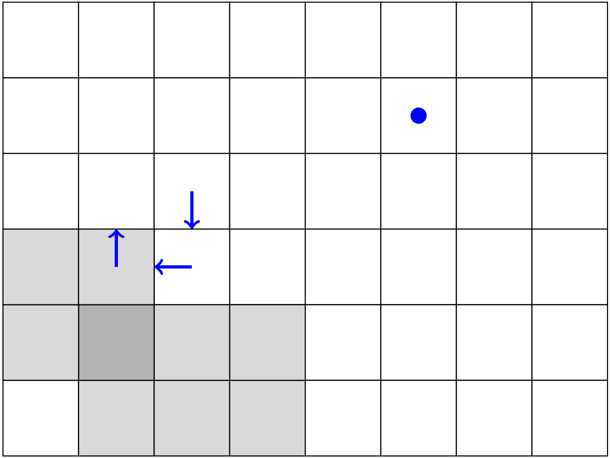
. . . and continue.

Wilson's theorem

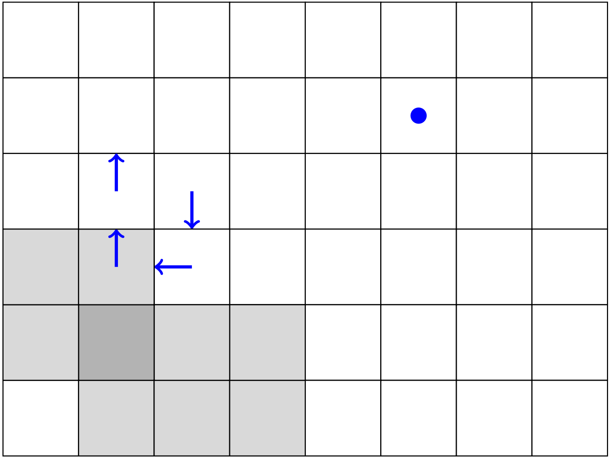


. . . and continue.

Wilson's theorem

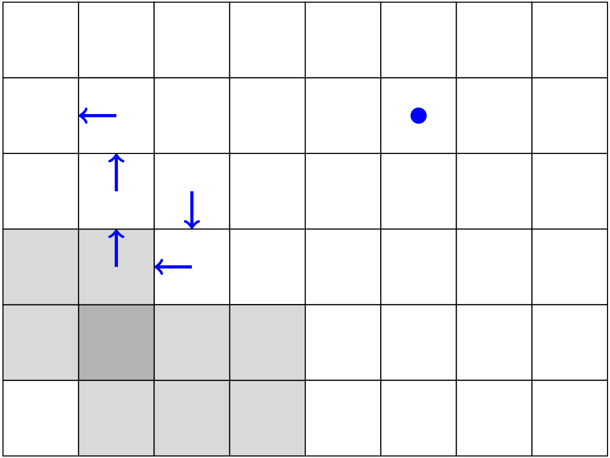


The resulting path is called *loop erased random walk*.



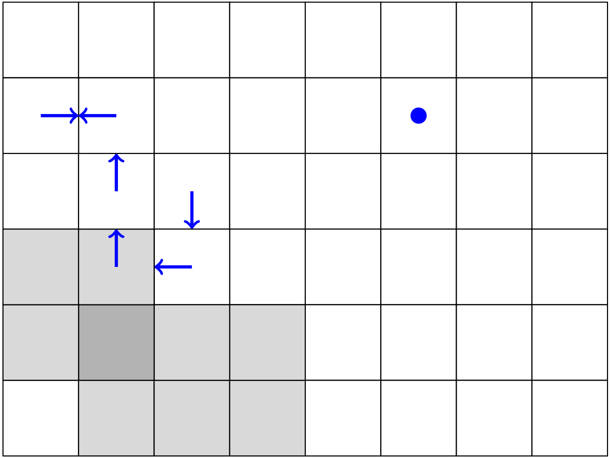
The resulting path is called *loop erased random walk*.

Wilson's theorem



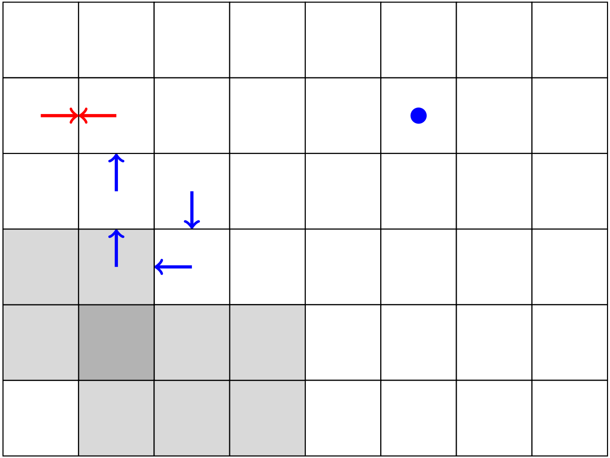
The resulting path is called *loop erased random walk*.

Wilson's theorem



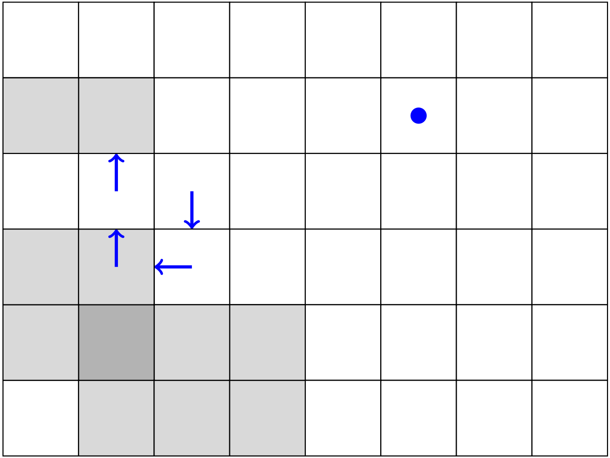
The resulting path is called *loop erased random walk*.

Wilson's theorem



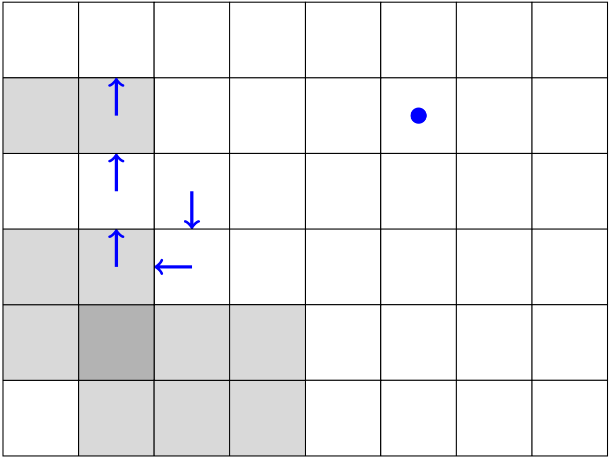
The resulting path is called *loop erased random walk*.

Wilson's theorem



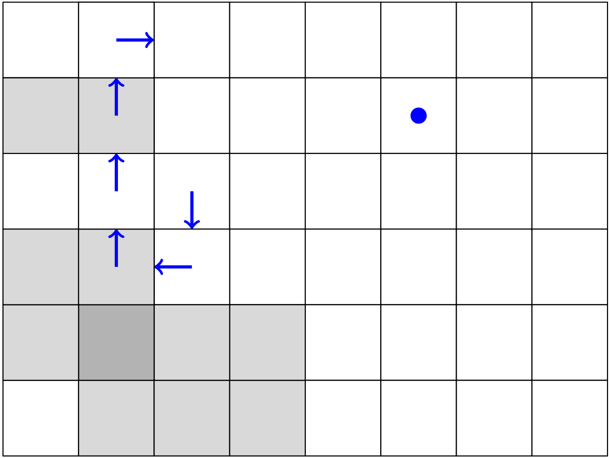
The resulting path is called *loop erased random walk*.

Wilson's theorem



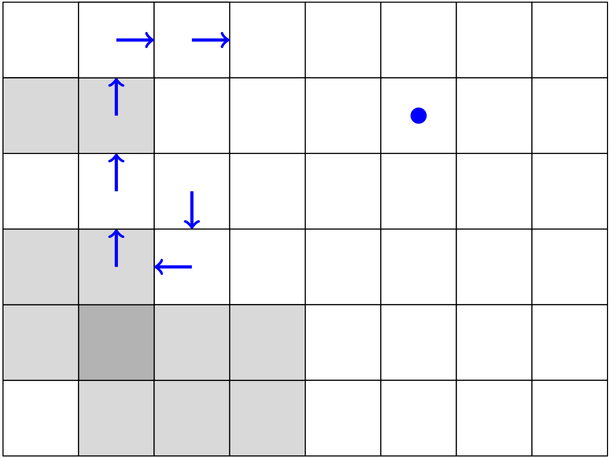
The resulting path is called *loop erased random walk*.

Wilson's theorem



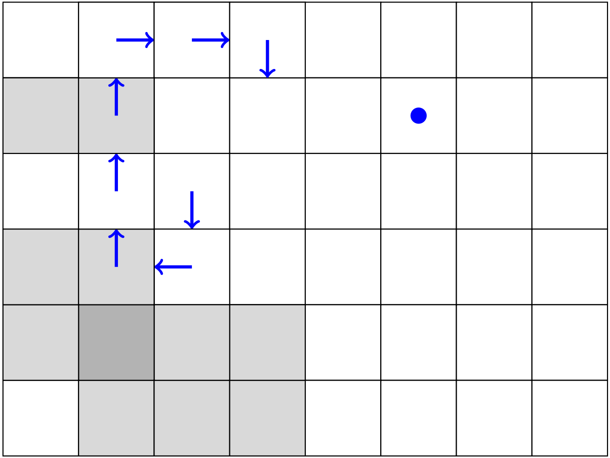
The resulting path is called *loop erased random walk*.

Wilson's theorem



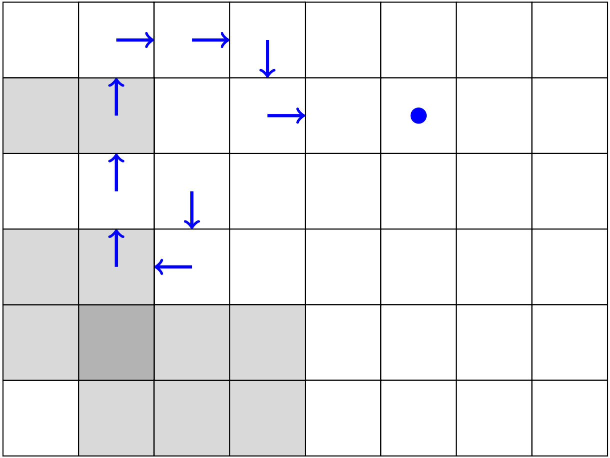
The resulting path is called *loop erased random walk*.

Wilson's theorem



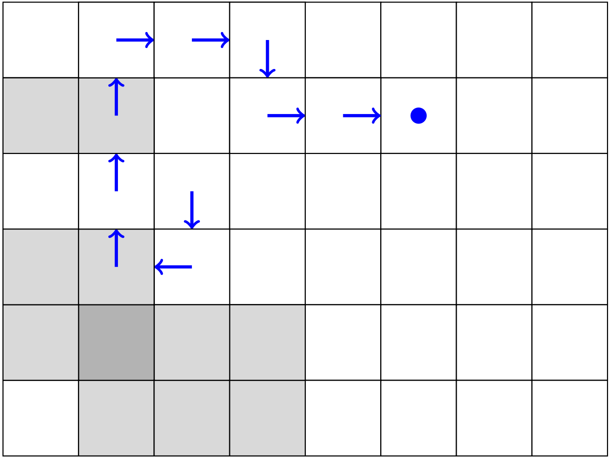
The resulting path is called *loop erased random walk*.

Wilson's theorem



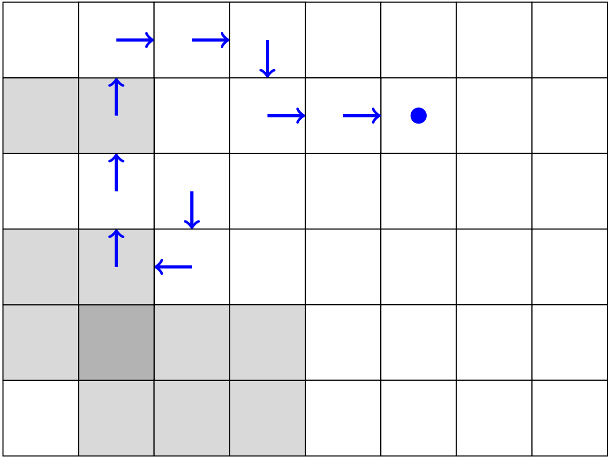
The resulting path is called *loop erased random walk*.

Wilson's theorem



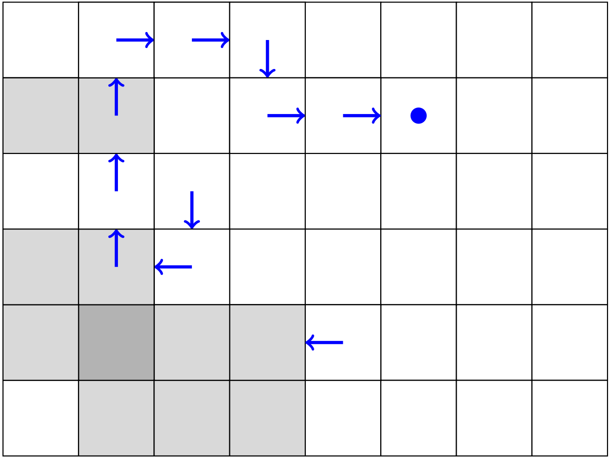
The resulting path is called *loop erased random walk*.

Wilson's theorem



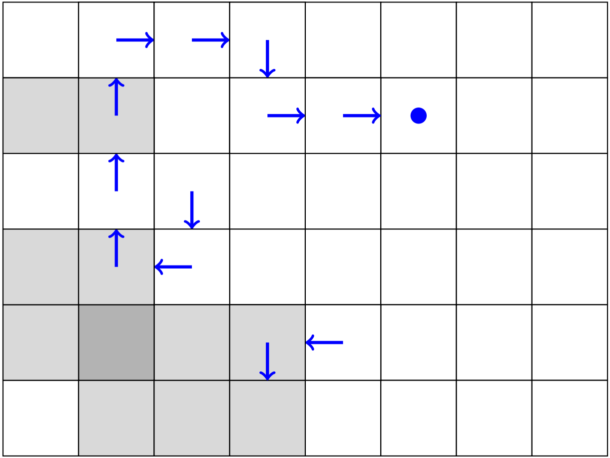
After we have reached the target, we start somewhere else.

Wilson's theorem



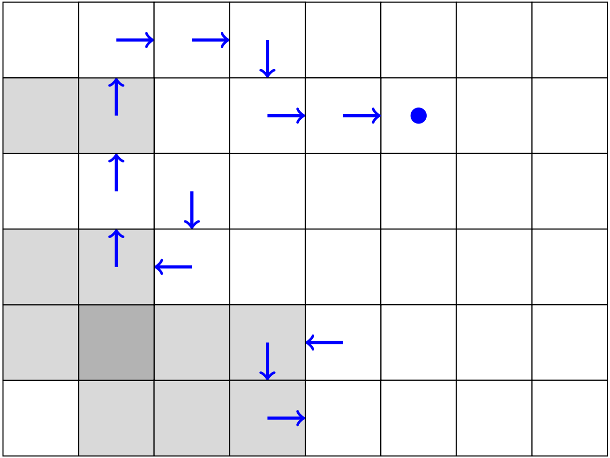
After we have reached the target, we start somewhere else.

Wilson's theorem



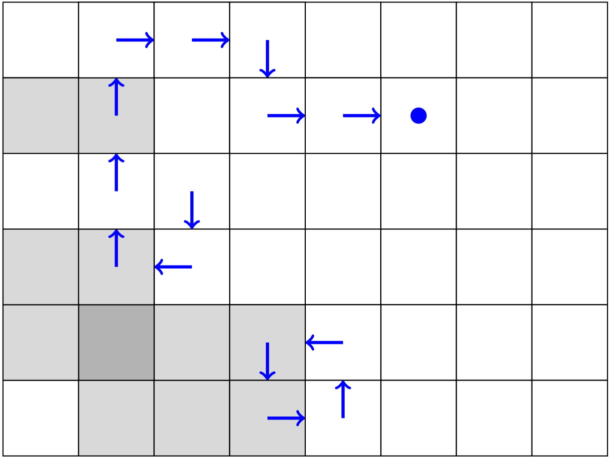
After we have reached the target, we start somewhere else.

Wilson's theorem



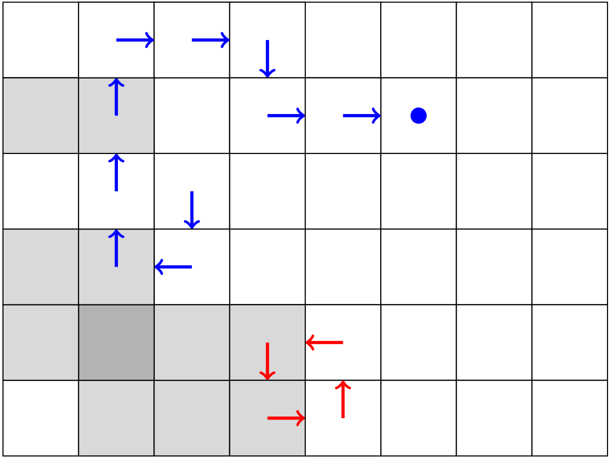
After we have reached the target, we start somewhere else.

Wilson's theorem



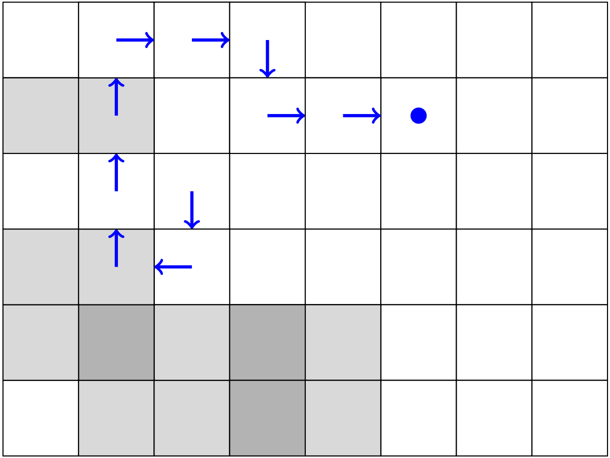
After we have reached the target, we start somewhere else.

Wilson's theorem



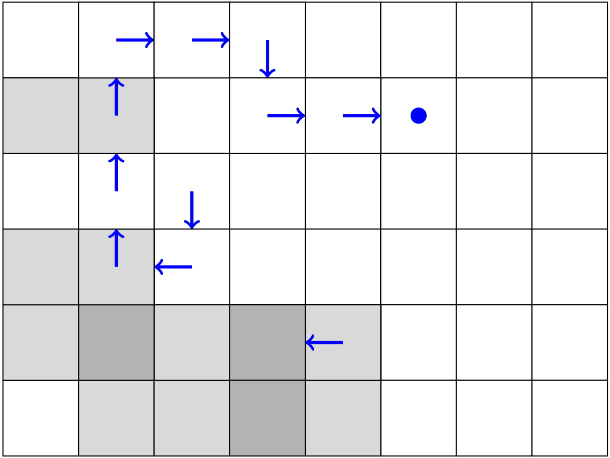
After we have reached the target, we start somewhere else.

Wilson's theorem



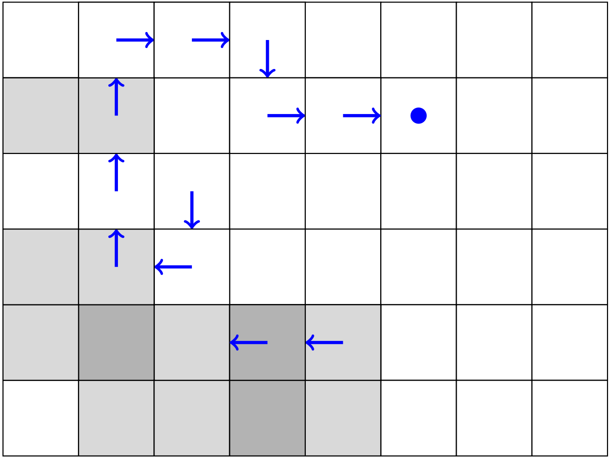
After we have reached the target, we start somewhere else.

Wilson's theorem



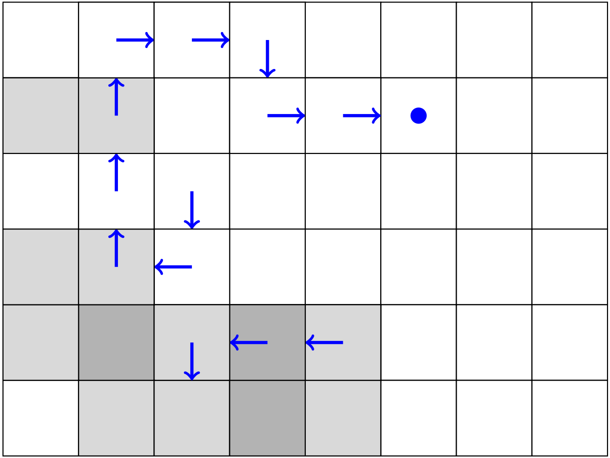
After we have reached the target, we start somewhere else.

Wilson's theorem



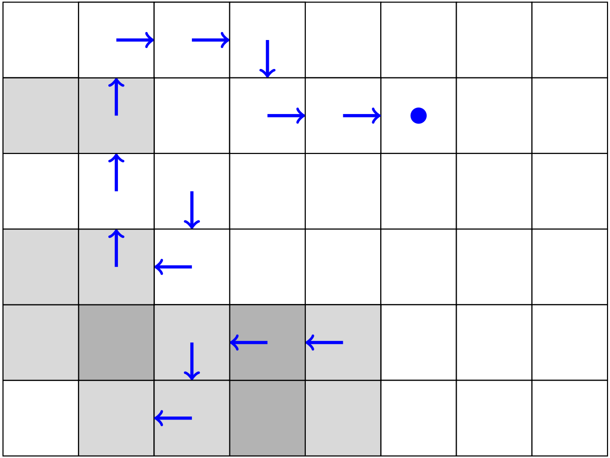
After we have reached the target, we start somewhere else.

Wilson's theorem



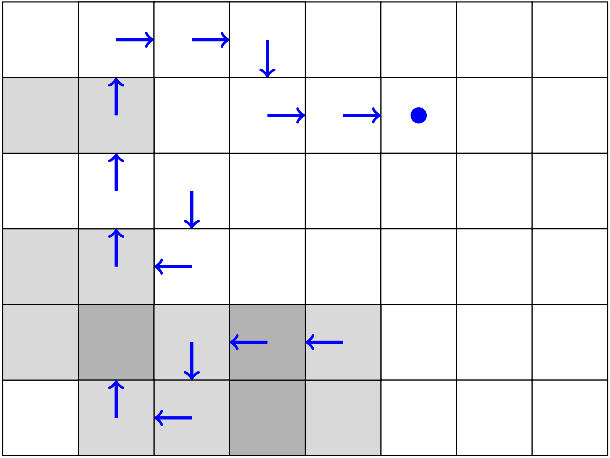
After we have reached the target, we start somewhere else.

Wilson's theorem



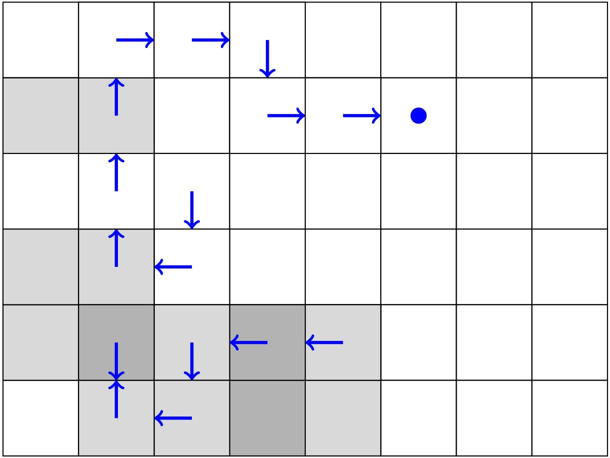
After we have reached the target, we start somewhere else.

Wilson's theorem



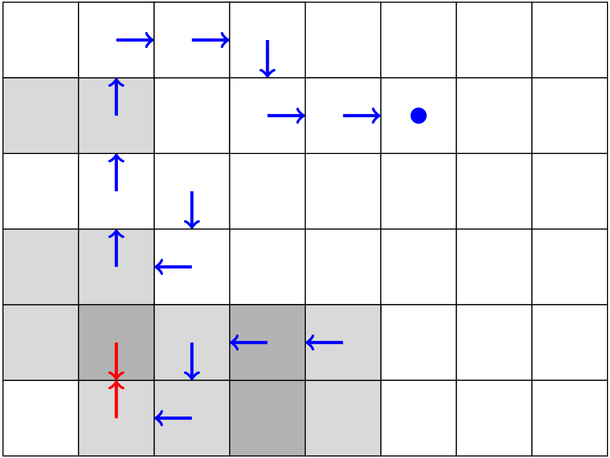
After we have reached the target, we start somewhere else.

Wilson's theorem



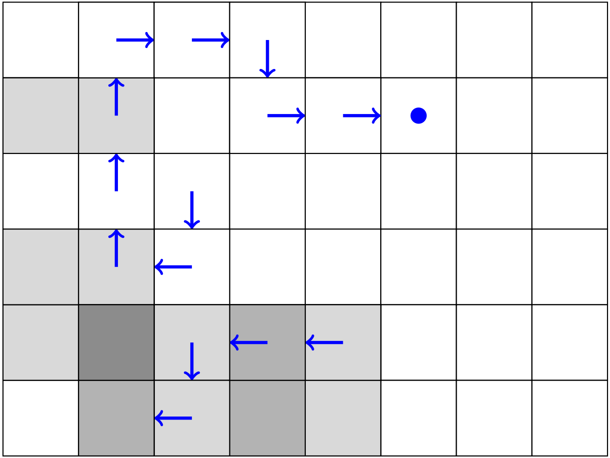
After we have reached the target, we start somewhere else.

Wilson's theorem



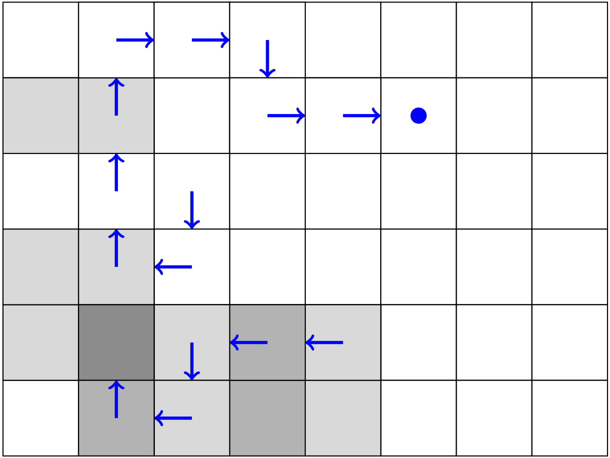
Until we hit on the previous path.

Wilson's theorem



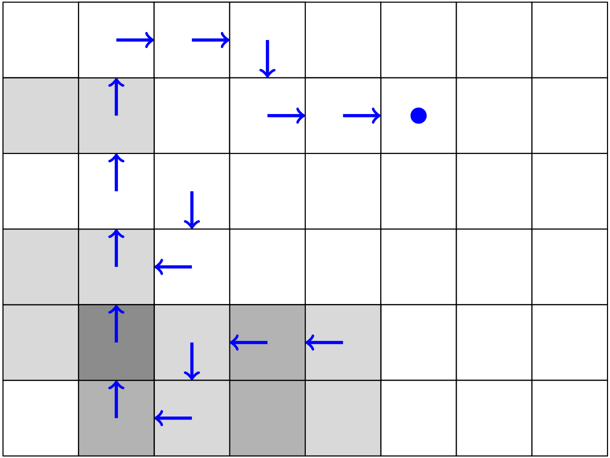
Until we hit on the previous path.

Wilson's theorem



Until we hit on the previous path.

Wilson's theorem



Until we hit on the previous path.

Wilson's theorem

This is *Wilson's algorithm*.

By the recurrence of our Markov chain, this process ends in finite time, proving that a.s., there are only finitely many poppable cycles.

Proof of 3. We can view a card as a directed edge $e = \langle x, y \rangle$ that points from vertex x to vertex y. The probability that a given card at x points twowards y is given by $P(\langle x, y \rangle) := p_{x,y}$, the transition probabilities of our Markov process.

Let S_{a} be the space of all arborescences $\mathcal{A}$ and let S_{c} be the space of all collections $\mathcal{C}$ of cards that can be placed on top of it, so that all cards in $\mathcal{C}$ belong to a poppable cycle.

The probability that Wilson's algorithm uncovers a given arborescence $\mathcal{A} \in S_{\mathrm{a}}$ by removing a given collection of cards $\mathcal{C} \in S_{\mathrm{c}}$ is given by

$$\mathbb{P}(\mathcal{A}, \mathcal{C}) = \Big(\prod_{e \in \mathcal{C}} P(e) \Big) \Big(\prod_{e \in \mathcal{A}} P(e) \Big).$$

Wilson's theorem

It follows that $\mathcal{A}$ and $\mathcal{C}$ are independent and

$$P(\mathcal{A}) = \frac{1}{Z} \prod_{e \in \mathcal{A}} P(e)$$

where Z is a normalization constant. Thus, $\mathcal{A}$ is distributed as the original configuration of arrows conditioned on being an arborescence. $\square$

Reversibility

Proposition Assume that the Markov kernel is reversible with stationary distribution π and let $w_{x,y} := \pi_x p_{x,y} = \pi_y p_{y,x}$ denote the corresponding edge weights. Then, forgetting about the direction of the arrows, the tree T uncovered by Wilson's algorithm has the distribution

$$\mu(T) = \frac{1}{Z'} \prod_{e \in T} w(e),$$

where Z' is a normalization constant. In particular, the law of T does not depend on the choice of the root.

Reversibility

Proof Write $e = \langle e_-, e_+ \rangle$. Then

$$P(\mathcal{A}) = \frac{1}{Z} \frac{\prod_{e \in \mathcal{A}} \pi_{e_-} p_{e_-,e_+}}{\prod_{x \neq r} \pi_x},$$

which equals μ for a suitable choice of Z'. $\square$

In particular, when $w_e \equiv 1$, Wilson't algorithm generates a *Uniform Spanning Tree* (UST).

Proposition For any $e \neq f$, the events $e \in T$ and $f \in T$ are *negatively correlated.*

Negative correlations

Proof (sketch) Fix vertices $s \neq t$ and let

$$I_{x,y} := \begin{cases} +1 & \text{if } s \overset{T}{\rightsquigarrow} x \overset{T}{\rightsquigarrow} y \overset{T}{\rightsquigarrow} t, \\ -1 & \text{if } s \overset{T}{\rightsquigarrow} y \overset{T}{\rightsquigarrow} x \overset{T}{\rightsquigarrow} t, \\ 0 & \text{otherwise,} \end{cases}$$

where T has law μ. Then $i_{x,y} := \mathbb{E}[I_{x,y}]$ is a unit s/t-current satisfying Kirchoff's laws. In particular,

$$\mu\big(\langle s,t\rangle \in T\big) = \mathbb{P}[I_{s,t} = +1] = i_{s,t} = w_{s,t}\big(\phi(t) - \phi(s)\big) = w_{s,t} R_{\text{eff}}(s,t).$$

We claim that conditioning μ on $\langle u,v\rangle \in T$ is the same as giving that edge infinite conductance, so by the Rayleigh principle, the conditional probability that $\langle s,t\rangle \in T$ is $\leq$ the unconditional probability. $\square$

Further topics

There is a close connection between the Uniform Spanning Tree and the so-called Abelian Sandpile Model. In the ASM, stacks of "sandgrains" are toppled in an abelian way that is reminiscent of the cycle popping of Wilson's algorithm. More importantly, Majumdar and Dhar (1992) proved a bijection between so-called "recurrent states" of the AST and spanning trees.

The UST also arises as the $q \to 0$ limit of the Random Cluster Model at zero temperature. Here the RCM is a random graph that includes percolation ($q = 1$) as a special case, and can also be used to construct the Ising model ($q = 2$) and Potts models ($q \geq 3$).

Further topics

In 2000, Oded Schramm studied scaling limits of 2-dimensional loop-erased random walks, which led him to invent SLE (Stochastic Löwner Equation).

This equation contains a parameter κ, where $\kappa = 2$ corresponds to the scaling limit of 2D loop-erased random walk.

It is nowadays believed (and partly proved) that by changing the value of κ, one can also obtain the 2D scaling limits of self-avoiding walks, percolation, the Ising model, and the Gaussian free field.

This led to the development of rigorous conformal field theory, for which Wendelin Werner won the Fields Medal in 2006, the first ever probabilist to do so.

Oded Schramm sadly died in a climbing accident in 2008.