The four dimensional uniform spanning tree

Perla Sousi [1]

Joint work with Tom Hutchcroft

[1]University of Cambridge

Spanning trees

Let $G = (V, E)$ be a finite connected graph – V = **vertices** and E = **edges**.

Spanning trees

Let $G = (V, E)$ be a finite connected graph – $V =$ **vertices** and $E =$ **edges**.

Spanning trees

Let $G = (V, E)$ be a finite connected graph – $V =$ **vertices** and $E =$ **edges**.

Definition

A tree is a connected graph with no cycles. A spanning tree of G is a subgraph of G which is a tree and has vertex set V.

Random trees

Let $G = (V, E)$ be a finite connected graph.

Random trees

Let $G = (V, E)$ be a finite connected graph.

Let $\mathcal{T}$ be the set of all spanning trees of G.

Random trees

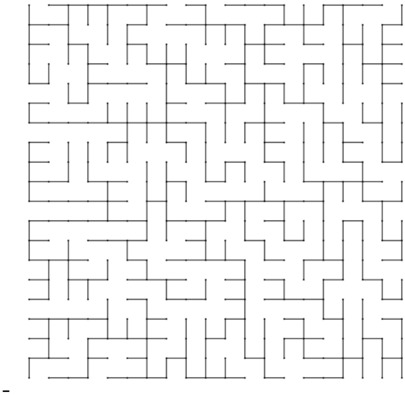
Let $G = (V, E)$ be a finite connected graph.

Let $\mathcal{T}$ be the set of all spanning trees of G. Pick T uniformly at random from $\mathcal{T}$. We call T a **uniform spanning tree – UST**.

Random trees

Let $G = (V, E)$ be a finite connected graph.

Let $\mathcal{T}$ be the set of all spanning trees of G. Pick T uniformly at random from $\mathcal{T}$. We call T a **uniform spanning tree – UST**.



credit: Sam Watson

This is the uniform spanning tree of $\mathbb{Z}^2$

Spanning trees

The study of spanning trees goes back to the work of **Kirchhoff** in **1847**.

1847. ANNALEN *No.* 12

DER PHYSIK UND CHEMIE.

BAND LXXII.

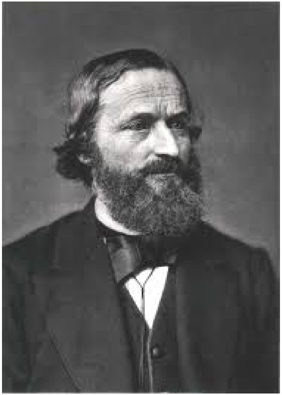
I. *Ueber die Auflösung der Gleichungen, auf welche man bei der Untersuchung der linearen Vertheilung galvanischer Ströme geführt wird;*

von G. Kirchhoff.

Ist ein System von n Drähten: $1, 2 \ldots n$ gegeben, welche auf eine beliebige Weise unter einander verbunden sind, und hat in einem jeden derselben eine beliebige elektromotorische Kraft ihren Sitz, so findet man zur Bestimmung der Intensitäten der Ströme, von welchen die Drähte durchflossen werden, $I_1, I_2 \ldots I_n$, die nöthige Anzahl linearer Gleichungen durch Benutzung der beiden folgenden Sätze [1]):

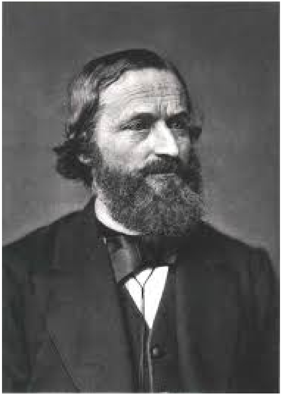
I. Wenn die Drähte $k_1, k_2, \ldots$ eine geschlossene Figur bilden, und w_k bezeichnet den Widerstand des Drahtes k, E_k die elektromotorische Kraft, die in demselben ihren Sitz hat, nach derselben Richtung positiv gerechnet als I_k, so ist, falls $I_{k1}, I_{k2}, \ldots$ alle nach *einer* Richtung als positiv gerechnet werden:

Gustav Kirchhoff



Gustav Kirchhoff

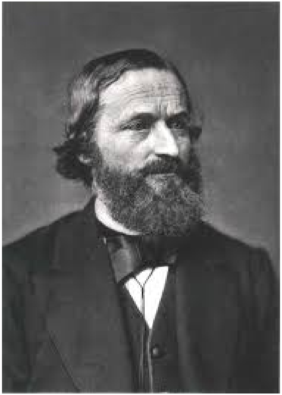
- In his 1847 paper he developed a set of rules that formalise that current and energy are conserved in electrical circuits.



Gustav Kirchhoff

- In his 1847 paper he developed a set of rules that formalise that current and energy are conserved in electrical circuits.
- In his work it is the first time that the connection between **UST** and **electrical networks** was established:

$$R_{\text{eff}}(e) = \frac{\#\text{spanning trees containing } \mathbf{e}}{|\mathcal{T}|}$$

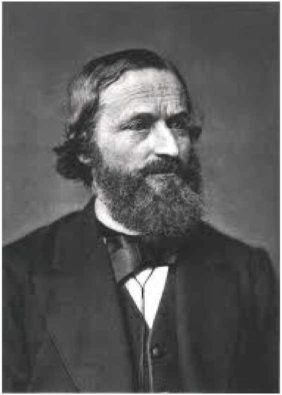


Gustav Kirchhoff

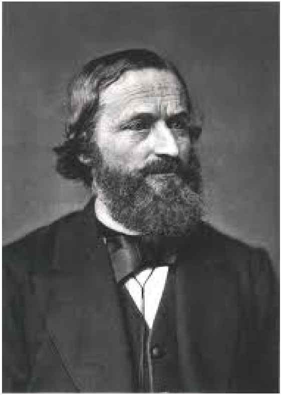
- In his 1847 paper he developed a set of rules that formalise that current and energy are conserved in electrical circuits.
- In his work it is the first time that the connection between **UST** and **electrical networks** was established:

$$R_{\mathrm{eff}}(e) = \frac{\#\text{spanning trees containing } \mathbf{e}}{|\mathcal{T}|}$$

- In his **8** page long paper, he also proved the **Matrix Tree Theorem** – counting the number of spanning trees of a graph.

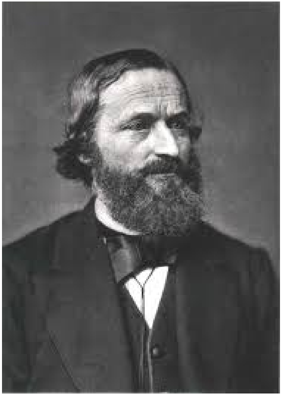


Gustav Kirchhoff



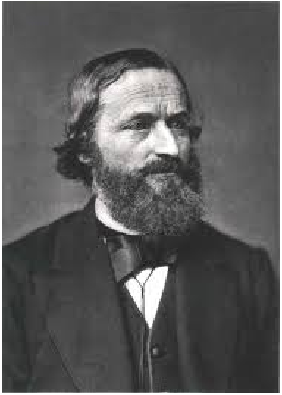
Gustav Kirchhoff

- His motivation was not *probabilistic*.



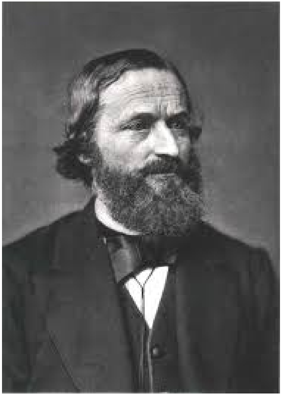
Gustav Kirchhoff

- His motivation was not *probabilistic*.
- Instead to set the foundations for the theory of electrical networks.



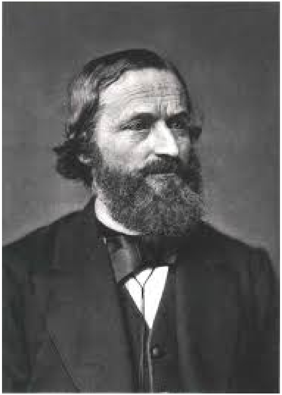
Gustav Kirchhoff

- His motivation was not *probabilistic*.
- Instead to set the foundations for the theory of electrical networks.
- His insight has been fruitful in both directions.



Gustav Kirchhoff

- His motivation was not *probabilistic*.
- Instead to set the foundations for the theory of electrical networks.
- His insight has been fruitful in both directions.
- Electrical networks are an important tool to understand the geometry of **large UST's**.



Why do we study UST?

- Picking a uniform spanning tree is an essential component in many randomised algorithms in computer science.

Why do we study UST?

- Picking a uniform spanning tree is an essential component in many randomised algorithms in computer science.
- It is connected to models in statistical physics, such as the random cluster model.

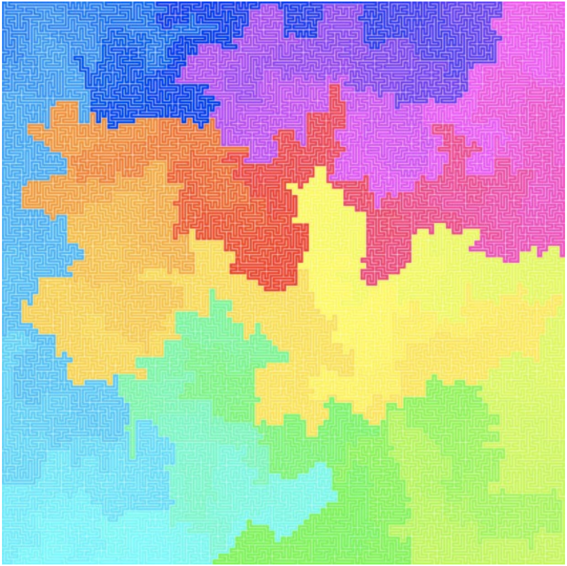
Why do we study UST?

- Picking a uniform spanning tree is an essential component in many randomised algorithms in computer science.
- It is connected to models in statistical physics, such as the random cluster model.
- Connections between UST, electrical networks and random walks.

Why do we study UST?

- Picking a uniform spanning tree is an essential component in many randomised algorithms in computer science.
- It is connected to models in statistical physics, such as the random cluster model.
- Connections between UST, electrical networks and random walks.
- Study of scaling limit of UST led **Oded Schramm** to develop the beautiful theory of **SLE** that describes the scaling limits of conformally invariant processes on the plane.

SLE(8)



credit: Russ Lyons

Sampling algorithms

Let $G = (V, E)$ be a finite connected graph.

Sampling algorithms

Let $G = (V, E)$ be a finite connected graph.

Algorithms for sampling a UST

Sampling algorithms

Let $G = (V, E)$ be a finite connected graph.

Algorithms for sampling a UST

- First sampling algorithm (1847) using **Matrix Tree Theorem** – Kirchhoff
- Wilson's algorithm using **loop erased walks**
- Aldous – Broder (and Diaconis) algorithm

Loop-erased random walk

Let γ be a finite path of vertices in G.

Loop-erased random walk

Let γ be a finite path of vertices in G.

We remove **loops** chronologically as they appear until there are no more loops left

Loop-erased random walk

Let γ be a finite path of vertices in G.

We remove **loops** chronologically as they appear until there are no more loops left

$\Rightarrow$ yields the **loop-erasure** of γ.

Loop-erased random walk

Let γ be a finite path of vertices in G.

We remove **loops** chronologically as they appear until there are no more loops left

$\Rightarrow$ yields the **loop-erasure** of γ.

Loop-erasing simple random walk path yields loop-erased random walk.

Loop-erased random walk

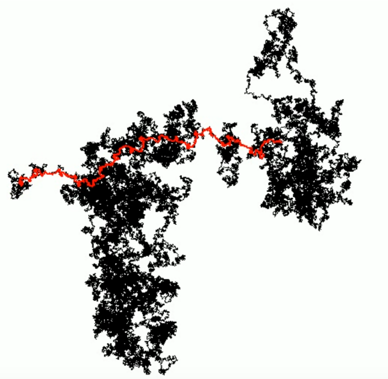
Let γ be a finite path of vertices in G.

We remove **loops** chronologically as they appear until there are no more loops left

$\Rightarrow$ yields the **loop-erasure** of γ.

Loop-erasing simple random walk path yields loop-erased random walk.

Take a loop erased random walk on $\mathbb{Z}^2$ and rescale space $\rightsquigarrow$ SLE(**2**) curve.



Wilson's algorithm for UST

Back to the finite case, $G = (V, E)$.

Wilson's algorithm for UST

Back to the finite case, $G = (V, E)$.

- Designate a root vertex r and order $V \setminus \{r\} = \{v_1, \ldots, v_{n-1}\}$.

Wilson's algorithm for UST

Back to the finite case, $G = (V, E)$.

- Designate a root vertex r and order $V \setminus \{r\} = \{v_1, \ldots, v_{n-1}\}$.
- Start a simple random walk from v_1 and run until it first hits r.

Wilson's algorithm for UST

Back to the finite case, $G = (V, E)$.

- Designate a root vertex r and order $V \setminus \{r\} = \{v_1, \ldots, v_{n-1}\}$.
- Start a simple random walk from v_1 and run until it first hits r.
- Erase loops.

Wilson's algorithm for UST

Back to the finite case, $G = (V, E)$.

- Designate a root vertex r and order $V \setminus \{r\} = \{v_1, \ldots, v_{n-1}\}$.
- Start a simple random walk from v_1 and run until it first hits r.
- Erase loops.
- Start a random walk from v_2 and run until it hits the first path. Erase loops.

Wilson's algorithm for UST

Back to the finite case, $G = (V, E)$.

- Designate a root vertex r and order $V \setminus \{r\} = \{v_1, \ldots, v_{n-1}\}$.
- Start a simple random walk from v_1 and run until it first hits r.
- Erase loops.
- Start a random walk from v_2 and run until it hits the first path. Erase loops.
- Continue until you exhaust all vertices.

Wilson's algorithm for UST

Back to the finite case, $G = (V, E)$.

- Designate a root vertex r and order $V \setminus \{r\} = \{v_1, \ldots, v_{n-1}\}$.
- Start a simple random walk from v_1 and run until it first hits r.
- Erase loops.
- Start a random walk from v_2 and run until it hits the first path. Erase loops.
- Continue until you exhaust all vertices.

We obtained a spanning tree!

Wilson's algorithm for UST

Back to the finite case, $G = (V, E)$.

- Designate a root vertex r and order $V \setminus \{r\} = \{v_1, \ldots, v_{n-1}\}$.
- Start a simple random walk from v_1 and run until it first hits r.
- Erase loops.
- Start a random walk from v_2 and run until it hits the first path. Erase loops.
- Continue until you exhaust all vertices.

We obtained a spanning tree!

Theorem (Wilson)

The tree we obtained has the same distribution as the **UST**.

Large UST's

Generate a "uniform" spanning tree of $\mathbb{Z}^2$

Large UST's

Generate a "uniform" spanning tree of $\mathbb{Z}^2$

- Designate 0 as the root vertex and order the rest.

Generate a "uniform" spanning tree of $\mathbb{Z}^2$

- Designate 0 as the root vertex and order the rest.
- Start a simple random walk from the first vertex in the ordering till it hits 0.

Large UST's

Generate a "uniform" spanning tree of $\mathbb{Z}^2$

- Designate 0 as the root vertex and order the rest.
- Start a simple random walk from the first vertex in the ordering till it hits 0.
- Erase loops.

Large UST's

Generate a "uniform" spanning tree of $\mathbb{Z}^2$

- Designate 0 as the root vertex and order the rest.
- Start a simple random walk from the first vertex in the ordering till it hits 0.
- Erase loops.
- Start a SRW from the next vertex in the ordering till it hits the previous path. Erase loops.

UST on $\mathbb{Z}^2$

SRW on $\mathbb{Z}^2$ is recurrent. It visits every vertex ∞ many times.

UST on $\mathbb{Z}^2$

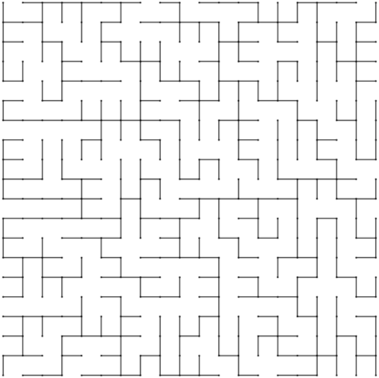
SRW on $\mathbb{Z}^2$ is recurrent. It visits every vertex ∞ many times.

So this algorithm is guaranteed to visit all vertices of $\mathbb{Z}^2$ and produce a connected tree.

UST on $\mathbb{Z}^2$

SRW on $\mathbb{Z}^2$ is recurrent. It visits every vertex ∞ many times.

So this algorithm is guaranteed to visit all vertices of $\mathbb{Z}^2$ and produce a connected tree.

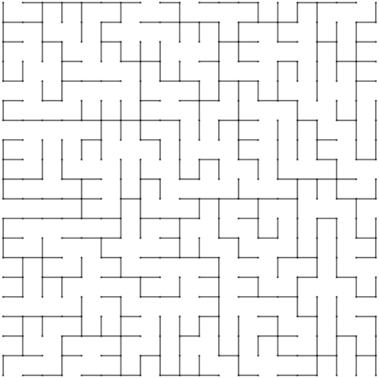


credit: Sam Watson

UST on $\mathbb{Z}^2$

SRW on $\mathbb{Z}^2$ is recurrent. It visits every vertex ∞ many times.

So this algorithm is guaranteed to visit all vertices of $\mathbb{Z}^2$ and produce a connected tree.



credit: Sam Watson

This is the uniform spanning tree of $\mathbb{Z}^2$

Higher dimensions

What is the canonical way to define the **UST** *in* $\mathbb{Z}^d$, $d \geq 3$?

Higher dimensions

What is the canonical way to define the **UST** *in* $\mathbb{Z}^d$, $d \geq 3$*?*

Consider the **UST** measure on finite **exhaustions** of the ∞ graph.

Higher dimensions

What is the canonical way to define the **UST** *in* $\mathbb{Z}^d$, $d \geq 3$?

Consider the **UST** measure on finite **exhaustions** of the ∞ graph.

Take $G_n = [-n, n]^d \cap \mathbb{Z}^d$ and let μ_n be the **UST** measure on G_n.

Higher dimensions

What is the canonical way to define the **UST** *in* $\mathbb{Z}^d$*,* $d \geq 3$*?*

Consider the **UST** measure on finite **exhaustions** of the ∞ graph.

Take $G_n = [-n, n]^d \cap \mathbb{Z}^d$ and let μ_n be the **UST** measure on G_n.

The sequence μ_n converges weakly to a limiting measure μ as $n \to \infty$.

Higher dimensions

What is the canonical way to define the **UST** *in* $\mathbb{Z}^d$, $d \geq 3$?

Consider the **UST** measure on finite **exhaustions** of the ∞ graph.

Take $G_n = [-n, n]^d \cap \mathbb{Z}^d$ and let μ_n be the **UST** measure on G_n.

The sequence μ_n converges weakly to a limiting measure μ as $n \to \infty$.

We call μ the **Uniform Spanning Forest** (**USF**) measure on $\mathbb{Z}^d$.

Higher dimensions

What is the canonical way to define the **UST** *in* $\mathbb{Z}^d$, $d \geq 3$*?*

Consider the **UST** measure on finite **exhaustions** of the ∞ graph.

Take $G_n = [-n, n]^d \cap \mathbb{Z}^d$ and let μ_n be the **UST** measure on G_n.

The sequence μ_n converges weakly to a limiting measure μ as $n \to \infty$.

We call μ the **Uniform Spanning Forest** (**USF**) measure on $\mathbb{Z}^d$.

Indeed, by construction μ is supported on acyclic graphs $\rightsquigarrow$ *forests*.

Higher dimensions

What is the canonical way to define the **UST** *in* $\mathbb{Z}^d$*,* $d \geq 3$*?*

Consider the **UST** measure on finite **exhaustions** of the ∞ graph.

Take $G_n = [-n, n]^d \cap \mathbb{Z}^d$ and let μ_n be the **UST** measure on G_n.

The sequence μ_n converges weakly to a limiting measure μ as $n \to \infty$.

We call μ the **Uniform Spanning Forest** (**USF**) measure on $\mathbb{Z}^d$.

Indeed, by construction μ is supported on acyclic graphs $\rightsquigarrow$ *forests*.

Theorem (Pemantle (1991))

The USF on $\mathbb{Z}^d$ *has one tree with probability* 1 *for* $d \leq 4$ *and infinitely many trees for* $d \geq 5$*.*

One ended trees

An infinite tree is k-**ended** if there exist exactly k distinct infinite simple paths starting at each vertex of the tree.

One ended trees

An infinite tree is **k-ended** if there exist exactly k distinct infinite simple paths starting at each vertex of the tree.

Examples: $\mathbb{Z}$ is 2-**ended**, $\mathbb{N}$ is 1-**ended**.

One ended trees

An infinite tree is k-**ended** if there exist exactly k distinct infinite simple paths starting at each vertex of the tree.

Examples: $\mathbb{Z}$ is 2-**ended**, $\mathbb{N}$ is 1-**ended**.

Theorem (Pemantle (1991); Benjamini, Lyons, Peres and Schramm (2001))

All trees in the USF in $\mathbb{Z}^d$ are one-ended for all $d \geq 2$.

Quantifying one-endedness

past of 0 $= \{0\} \cup$ finite piece disconnected from ∞ by 0

Quantifying one-endedness

past of 0 $= \{0\} \cup$ finite piece disconnected from ∞ by 0

Theorem (Hutchcroft (2017) $d \geq 5$)

$$\mathbb{P}(\textit{past of } 0 \text{ contains a path of length } n) \asymp \frac{1}{n}$$

$$\mathbb{P}(\textit{past of } 0 \cap \partial B(0,n) \neq \emptyset) \asymp \frac{1}{n^2}$$

$$\mathbb{P}(|\textit{past of } 0| \geq n) \asymp \frac{1}{\sqrt{n}}$$

Quantifying one-endedness

past of 0 $= \{0\} \cup$ finite piece disconnected from ∞ by 0

Theorem (Hutchcroft (2017) $d \geq 5$)

$$\mathbb{P}(\textit{past of } 0 \text{ contains a path of length } n) \asymp \frac{1}{n}$$

$$\mathbb{P}(\textit{past of } 0 \cap \partial B(0,n) \neq \emptyset) \asymp \frac{1}{n^2}$$

$$\mathbb{P}(|\textit{past of } 0| \geq n) \asymp \frac{1}{\sqrt{n}}$$

What happens at $\mathbf{d = 4}$?

$d = 4$

Theorem (Hutchcroft and S. (2020) d=4)

$$\mathbb{P}(\textit{past of } 0 \text{ contains a path of length } n) \asymp \frac{(\log n)^{1/3}}{n}$$

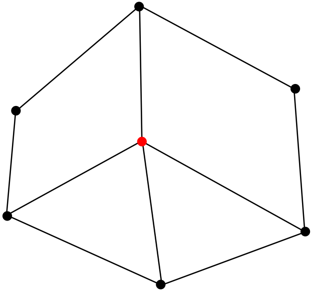
$$\mathbb{P}(\textit{past of } 0 \;\cap \partial B(0,n) \neq \emptyset) \asymp \frac{(\log n)^{2/3+o(1)}}{n^2}$$

$$\mathbb{P}(|\textit{past of } 0| \geq n) \asymp \frac{(\log n)^{1/6}}{\sqrt{n}}$$

Proof ideas

Small detour

Aldous - **Broder** algorithm for generating a **UST** of a finite graph G

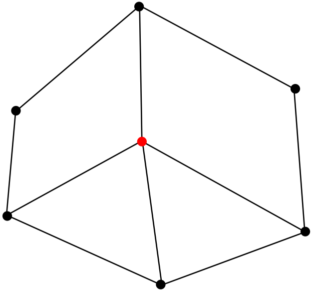


Proof ideas

Small detour

Aldous - **Broder** algorithm for generating a **UST** of a finite graph G

- Let $\boldsymbol{o}$ be a vertex of G.

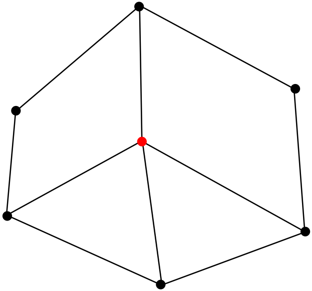


Proof ideas

Small detour

Aldous - **Broder** algorithm for generating a **UST** of a finite graph G

- Let $\boldsymbol{o}$ be a vertex of G.
- Start a random walk from **o** and run until the **cover time** **(1st time walk has visited every vertex)**

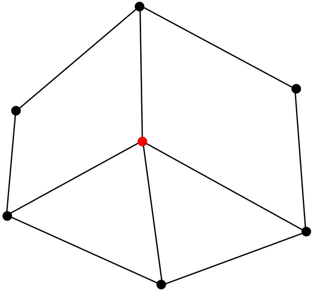


Proof ideas

Small detour

Aldous - **Broder** algorithm for generating a **UST** of a finite graph G

- Let $\boldsymbol{o}$ be a vertex of G.
- Start a random walk from **o** and run until the **cover time (1st time walk has visited every vertex)**
- For every vertex $\boldsymbol{v} \neq \boldsymbol{o}$, keep the **edge** that was used when visiting v for the first time.

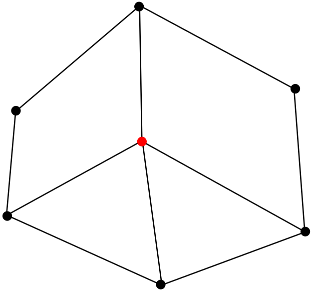


Proof ideas

Small detour

Aldous - **Broder** algorithm for generating a **UST** of a finite graph G

- Let $\boldsymbol{o}$ be a vertex of G.
- Start a random walk from **o** and run until the **cover time** **(1st time walk has visited every vertex)**
- For every vertex $\boldsymbol{v} \neq \boldsymbol{o}$, keep the **edge** that was used when visiting v for the first time.

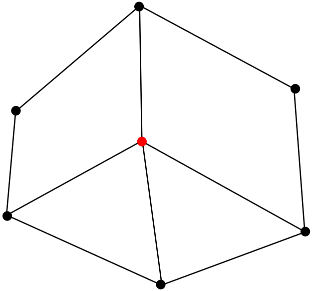


Proof ideas

Small detour

Aldous - **Broder** algorithm for generating a **UST** of a finite graph G

- Let $\boldsymbol{o}$ be a vertex of G.
- Start a random walk from **o** and run until the **cover time** **(1st time walk has visited every vertex)**
- For every vertex $\boldsymbol{v} \neq \boldsymbol{o}$, keep the **edge** that was used when visiting v for the first time.

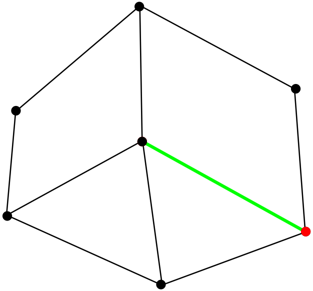


Proof ideas

Small detour

Aldous - **Broder** algorithm for generating a **UST** of a finite graph G

- Let $\boldsymbol{o}$ be a vertex of G.
- Start a random walk from **o** and run until the **cover time** **(1st time walk has visited every vertex)**
- For every vertex $\boldsymbol{v} \neq \boldsymbol{o}$, keep the **edge** that was used when visiting v for the first time.

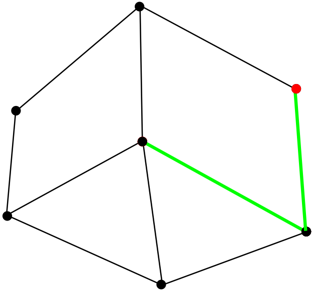


Proof ideas

Small detour

Aldous - **Broder** algorithm for generating a **UST** of a finite graph G

- Let o be a vertex of G.
- Start a random walk from **o** and run until the **cover time** **(1st time walk has visited every vertex)**
- For every vertex $v \neq o$, keep the **edge** that was used when visiting v for the first time.

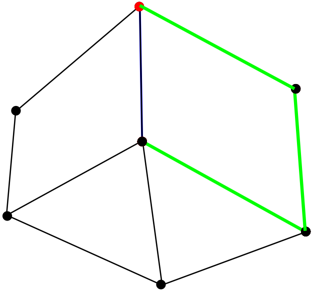


Proof ideas

Small detour

Aldous - **Broder** algorithm for generating a **UST** of a finite graph G

- Let $\boldsymbol{o}$ be a vertex of G.
- Start a random walk from **o** and run until the **cover time** **(1st time walk has visited every vertex)**
- For every vertex $\boldsymbol{v} \neq \boldsymbol{o}$, keep the **edge** that was used when visiting v for the first time.

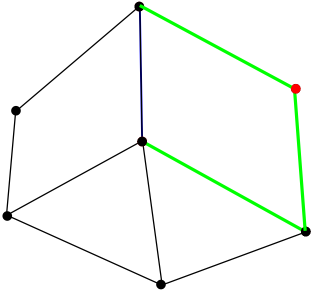


Proof ideas

Small detour

Aldous - **Broder** algorithm for generating a **UST** of a finite graph G

- Let $\boldsymbol{o}$ be a vertex of G.
- Start a random walk from **o** and run until the **cover time** **(1st time walk has visited every vertex)**
- For every vertex $\boldsymbol{v} \neq \boldsymbol{o}$, keep the **edge** that was used when visiting v for the first time.

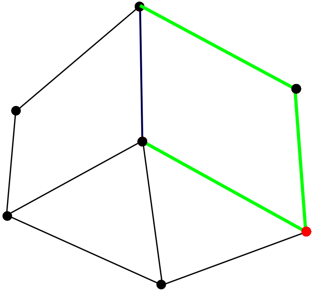


Proof ideas

Small detour

Aldous - **Broder** algorithm for generating a **UST** of a finite graph G

- Let $\boldsymbol{o}$ be a vertex of G.
- Start a random walk from **o** and run until the **cover time (1st time walk has visited every vertex)**
- For every vertex $\boldsymbol{v} \neq \boldsymbol{o}$, keep the **edge** that was used when visiting v for the first time.

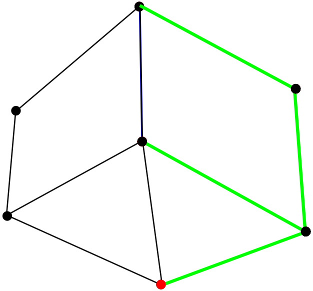


Proof ideas

Small detour

Aldous - **Broder** algorithm for generating a **UST** of a finite graph G

- Let $\boldsymbol{o}$ be a vertex of G.
- Start a random walk from **o** and run until the **cover time** **(1st time walk has visited every vertex)**
- For every vertex $\boldsymbol{v} \neq \boldsymbol{o}$, keep the **edge** that was used when visiting v for the first time.

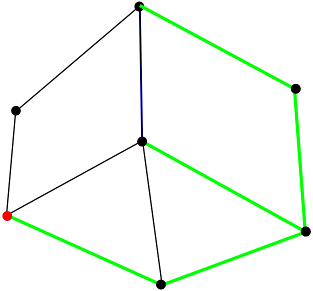


Proof ideas

Small detour

Aldous - **Broder** algorithm for generating a **UST** of a finite graph G

- Let $\boldsymbol{o}$ be a vertex of G.
- Start a random walk from **o** and run until the **cover time** **(1st time walk has visited every vertex)**
- For every vertex $\boldsymbol{v} \neq \boldsymbol{o}$, keep the **edge** that was used when visiting v for the first time.

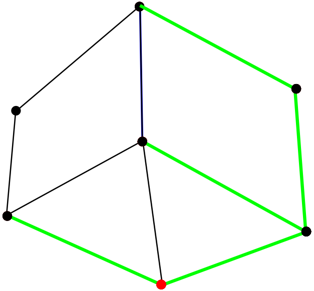


Proof ideas

Small detour

Aldous - **Broder** algorithm for generating a **UST** of a finite graph G

- Let $\boldsymbol{o}$ be a vertex of G.
- Start a random walk from **o** and run until the **cover time** **(1st time walk has visited every vertex)**
- For every vertex $\boldsymbol{v} \neq \boldsymbol{o}$, keep the **edge** that was used when visiting v for the first time.

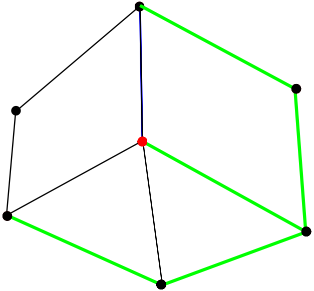


Proof ideas

Small detour

Aldous - **Broder** algorithm for generating a **UST** of a finite graph G

- Let $\boldsymbol{o}$ be a vertex of G.
- Start a random walk from **o** and run until the **cover time** **(1st time walk has visited every vertex)**
- For every vertex $\boldsymbol{v} \neq \boldsymbol{o}$, keep the **edge** that was used when visiting v for the first time.

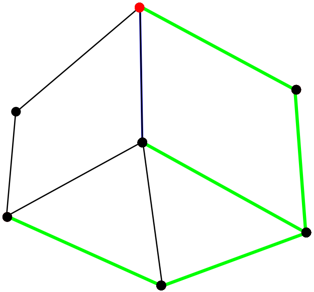


Proof ideas

Small detour

Aldous - **Broder** algorithm for generating a **UST** of a finite graph G

- Let $\boldsymbol{o}$ be a vertex of G.
- Start a random walk from **o** and run until the **cover time (1st time walk has visited every vertex)**
- For every vertex $\boldsymbol{v} \neq \boldsymbol{o}$, keep the **edge** that was used when visiting v for the first time.

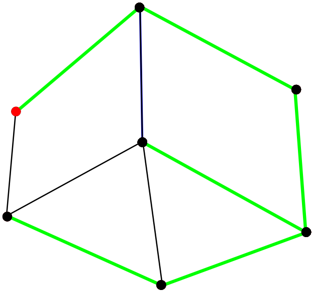


Proof ideas

Small detour

Aldous - **Broder** algorithm for generating a **UST** of a finite graph G

- Let o be a vertex of G.
- Start a random walk from **o** and run until the **cover time (1st time walk has visited every vertex)**
- For every vertex $v \neq o$, keep the **edge** that was used when visiting v for the first time.

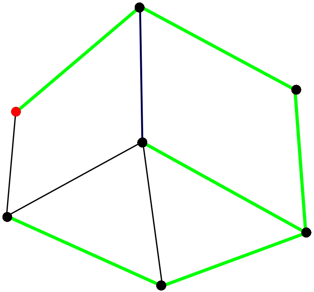


Proof ideas

Small detour

Aldous - **Broder** algorithm for generating a **UST** of a finite graph G

- Let $\boldsymbol{o}$ be a vertex of G.
- Start a random walk from **o** and run until the **cover time (1st time walk has visited every vertex)**
- For every vertex $\boldsymbol{v} \neq \boldsymbol{o}$, keep the **edge** that was used when visiting v for the first time.
- The collection of all these **edges** constitutes a spanning tree.

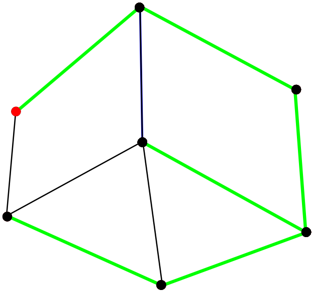


Proof ideas

Small detour

Aldous - **Broder** algorithm for generating a **UST** of a finite graph G

- Let $\boldsymbol{o}$ be a vertex of G.
- Start a random walk from **o** and run until the **cover time (1st time walk has visited every vertex)**
- For every vertex $\boldsymbol{v} \neq \boldsymbol{o}$, keep the **edge** that was used when visiting v for the first time.
- The collection of all these **edges** constitutes a spanning tree.



Theorem (Aldous - Broder (discussions with Diaconis) 1990)

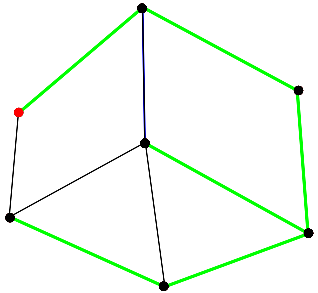
The distribution of the spanning tree generated is that of the **UST**.

Proof ideas

Small detour

Aldous - **Broder** algorithm for generating a **UST** of a finite graph G

- Let $\boldsymbol{o}$ be a vertex of G.
- Start a random walk from **o** and run until the **cover time (1st time walk has visited every vertex)**
- For every vertex $\boldsymbol{v} \neq \boldsymbol{o}$, keep the **edge** that was used when visiting v for the first time.
- The collection of all these **edges** constitutes a spanning tree.



Theorem (Aldous - Broder (discussions with Diaconis) 1990)

The distribution of the spanning tree generated is that of the **UST**.

Clear how to generalise the algorithm for an ∞ *recurrent graph* (walk visits every vertex with probability 1).

Proof ideas

Hutchcroft's generalisation for transient graphs

Proof ideas

Hutchcroft's generalisation for transient graphs

- Replace the walk with Sznitman's **random interlacements (RI)**

Proof ideas

Hutchcroft's generalisation for transient graphs

- Replace the walk with Sznitman's **random interlacements (RI)**
- **RI** = Poisson process of bi-∞ random walk trajectories arriving in time

$$\mathbb{P}(\textbf{RI} \text{ has not hit } K \text{ by time } t) = \exp\left(-t \cdot \mathrm{Cap}(K)\right),$$

Proof ideas

Hutchcroft's generalisation for transient graphs

- Replace the walk with Sznitman's **random interlacements (RI)**
- **RI** $=$ Poisson process of bi-∞ random walk trajectories arriving in time

$$\mathbb{P}(\textbf{RI} \text{ has not hit } K \text{ by time } t) = \exp\left(-t \cdot \mathrm{Cap}(K)\right),$$

$\mathrm{Cap}(K) = \sum_{x \in K} \mathbb{P}_x(\text{never hit } K \text{ again})$ for K finite.

Proof ideas

Hutchcroft's generalisation for transient graphs

- Replace the walk with Sznitman's **random interlacements (RI)**
- **RI** $=$ Poisson process of bi-∞ random walk trajectories arriving in time

$$\mathbb{P}(\textbf{RI} \text{ has not hit } K \text{ by time } t) = \exp\left(-t \cdot \mathrm{Cap}(K)\right),$$

$\mathrm{Cap}(K) = \sum_{x \in K} \mathbb{P}_x(\text{never hit } K \text{ again})$ for K finite.

Discrete analogue of the Newtonian capacity: for $A \subseteq \mathbb{R}^d$ compact

$$\frac{1}{\mathrm{Cap}(A)} = \inf\left\{\int\int G(x,y)d\mu(x)d\mu(y) : \mu \text{ prob. measure on } A\right\}$$

(G is the Green kernel)

Proof ideas

Proof ideas

Proof ideas

This dynamic way of generating the USF is best suited for calculating tail probabilities.

Proof ideas

This dynamic way of generating the USF is best suited for calculating tail probabilities.

Lower bound Let ε to be determined later.

Proof ideas

This dynamic way of generating the USF is best suited for calculating tail probabilities.

Lower bound Let ε to be determined later.

- $A = \{0 \text{ is hit by a unique trajectory } W \text{ of RI in } [0, \varepsilon]\}$.

Proof ideas

This dynamic way of generating the USF is best suited for calculating tail probabilities.

Lower bound Let ε to be determined later.

- $A = \{0$ is hit by a unique trajectory W of RI in $[0, \varepsilon]\}$.
- Apply Aldous Broder to $W|_{[0,\infty)} \rightsquigarrow \eta =$ LERW in $\mathbb{Z}^4$.

Proof ideas

This dynamic way of generating the USF is best suited for calculating tail probabilities.

Lower bound Let ε to be determined later.

- $A = \{0 \text{ is hit by a unique trajectory } W \text{ of RI in } [0, \varepsilon]\}$.
- Apply Aldous Broder to $W|_{[0,\infty)} \rightsquigarrow \eta = \text{LERW in } \mathbb{Z}^4$.
- $B = \{ \ W|_{(-\infty,0)} \cap \eta[0, n] = \emptyset \ \} \rightsquigarrow \mathbb{P}(B) \asymp (\log n)^{-1/3}$ [Lawler]

Proof ideas

This dynamic way of generating the USF is best suited for calculating tail probabilities.

Lower bound Let ε to be determined later.

- $A = \{0 \text{ is hit by a unique trajectory } W \text{ of RI in } [0, \varepsilon]\}$.
- Apply Aldous Broder to $W|_{[0,\infty)} \rightsquigarrow \eta = \text{LERW in } \mathbb{Z}^4$.
- $B = \{ \; W|_{(-\infty,0)} \cap \eta[0, n] = \emptyset \; \} \rightsquigarrow \mathbb{P}(B) \asymp (\log n)^{-1/3}$ [Lawler]
- $C = \{ \text{ no other trajectory of RI hits } \eta[0, n] \text{ in } [0, \varepsilon] \; \}$.

Proof ideas

This dynamic way of generating the USF is best suited for calculating tail probabilities.

Lower bound Let ε to be determined later.

- $A = \{0$ is hit by a unique trajectory W of RI in $[0, \varepsilon]\}$.
- Apply Aldous Broder to $W|_{[0,\infty)} \rightsquigarrow \eta =$ LERW in $\mathbb{Z}^4$.
- $B = \{ W|_{(-\infty,0)} \cap \eta[0, n] = \emptyset \} \rightsquigarrow \mathbb{P}(B) \asymp (\log n)^{-1/3}$ [Lawler]
- $C = \{$ no other trajectory of RI hits $\eta[0, n]$ in $[0, \varepsilon] \}$.

So $\mathbb{P}($past of 0 contains a path of length $n) \geq \mathbb{P}(A \cap B \cap C)$

Proof ideas

This dynamic way of generating the USF is best suited for calculating tail probabilities.

Lower bound Let ε to be determined later.

- $A = \{0 \text{ is hit by a unique trajectory } W \text{ of RI in } [0, \varepsilon]\}$.
- Apply Aldous Broder to $W|_{[0,\infty)} \rightsquigarrow \eta = \text{LERW in } \mathbb{Z}^4$.
- $B = \{ \ W|_{(-\infty,0)} \cap \eta[0,n] = \emptyset \ \} \rightsquigarrow \mathbb{P}(B) \asymp (\log n)^{-1/3}$ [Lawler]
- $C = \{ \text{ no other trajectory of RI hits } \eta[0,n] \text{ in } [0,\varepsilon] \ \}$.

So $\mathbb{P}(\text{past of 0 contains a path of length } n) \geq \mathbb{P}(A \cap B \cap C)$ and

$$\mathbb{P}(A \cap B \cap C) \asymp \varepsilon \cdot \frac{1}{(\log n)^{1/3}} \cdot \mathbb{E}\left[e^{-\varepsilon \mathrm{Cap}(\eta[0,n])}\right].$$

Proof ideas

This dynamic way of generating the USF is best suited for calculating tail probabilities.

Lower bound Let ε to be determined later.

- $A = \{0$ is hit by a unique trajectory W of RI in $[0, \varepsilon]\}$.
- Apply Aldous Broder to $W|_{[0,\infty)} \rightsquigarrow \eta =$ LERW in $\mathbb{Z}^4$.
- $B = \{ W|_{(-\infty,0)} \cap \eta[0,n] = \emptyset \} \rightsquigarrow \mathbb{P}(B) \asymp (\log n)^{-1/3}$ [Lawler]
- $C = \{$ no other trajectory of RI hits $\eta[0,n]$ in $[0,\varepsilon]$ $\}$.

So $\mathbb{P}$(past of 0 contains a path of length n) $\geq \mathbb{P}(A \cap B \cap C)$ and

$$\mathbb{P}(A \cap B \cap C) \asymp \varepsilon \cdot \frac{1}{(\log n)^{1/3}} \cdot \mathbb{E}\left[e^{-\varepsilon \mathrm{Cap}(\eta[0,n])}\right].$$

Need to estimate $\mathbb{E}\left[e^{-\varepsilon \mathrm{Cap}(\eta[0,n])}\right]$, where η LERW in $\mathbb{Z}^4$.

Capacity

Recall $\mathrm{Cap}(K) = \sum_{x \in K} \mathbb{P}_x(\text{never hit } K \text{ again})$.

Capacity

Recall $\mathrm{Cap}(K) = \sum_{x \in K} \mathbb{P}_x(\text{never hit } K \text{ again})$. *Equivalently*

$$\boxed{\mathrm{Cap}(K) = \lim_{x \to \infty} \frac{\mathbb{P}_x(\text{hit } K)}{G(x)}}$$

Capacity

Recall $\mathrm{Cap}(K) = \sum_{x \in K} \mathbb{P}_x(\text{never hit } K \text{ again})$. *Equivalently*

$$\mathrm{Cap}(K) = \lim_{x \to \infty} \frac{\mathbb{P}_x(\text{hit } K)}{G(x)}$$

$\mathrm{Cap}(\eta[0,n])$ $\Longrightarrow$ intersection probabilities between **LERW** and **SRW**

Capacity

Recall $\mathrm{Cap}(K) = \sum_{x \in K} \mathbb{P}_x(\text{never hit } K \text{ again})$. *Equivalently*

$$\mathrm{Cap}(K) = \lim_{x \to \infty} \frac{\mathbb{P}_x(\text{hit } K)}{G(x)}$$

$\mathrm{Cap}(\eta[0, n])$ $\Longrightarrow$ intersection probabilities between **LERW** and **SRW**

Step back: Let X, Y be independent **SRW**'s in $\mathbb{Z}^4$ with $\|X_0 - Y_0\| \asymp \sqrt{n}$.

Capacity

Recall $\mathrm{Cap}(K) = \sum_{x \in K} \mathbb{P}_x(\text{never hit } K \text{ again})$. *Equivalently*

$$\mathrm{Cap}(K) = \lim_{x \to \infty} \frac{\mathbb{P}_x(\text{hit } K)}{G(x)}$$

$\mathrm{Cap}(\eta[0,n])$ $\Longrightarrow$ intersection probabilities between **LERW** and **SRW**

Step back: Let X, Y be independent **SRW**'s in $\mathbb{Z}^4$ with $\|X_0 - Y_0\| \asymp \sqrt{n}$.

$$\mathbb{P}(X[0,n] \cap Y[0,\infty) \neq \emptyset) \asymp \frac{1}{\log n}$$

[Lawler '90's]

Recall $\mathrm{Cap}(K) = \sum_{x \in K} \mathbb{P}_x(\text{never hit } K \text{ again})$. *Equivalently*

$$\mathrm{Cap}(K) = \lim_{x \to \infty} \frac{\mathbb{P}_x(\text{hit } K)}{G(x)}$$

$\mathrm{Cap}(\eta[0, n])$ $\Longrightarrow$ intersection probabilities between **LERW** and **SRW**

Step back: Let X, Y be independent **SRW**'s in $\mathbb{Z}^4$ with $\|X_0 - Y_0\| \asymp \sqrt{n}$.

$$\mathbb{P}(X[0, n] \cap Y[0, \infty) \neq \emptyset) \asymp \frac{1}{\log n}$$ [Lawler '90's]

This easily then yields

$$\mathbb{E}[\mathrm{Cap}(X[0, n])] \asymp \frac{n}{\log n}.$$

Capacity

Theorem (Lyons, Peres and Schramm)

Let X and Y be transient chains. Then

$$\mathbb{P}(\mathrm{LE}(X) \cap Y[0,\infty) \neq \emptyset) \asymp \mathbb{P}(X \cap Y[0,\infty) \neq \emptyset)\,.$$

Capacity

Theorem (Lyons, Peres and Schramm)

Let X and Y be transient chains. Then

$$\mathbb{P}(\mathrm{LE}(X) \cap Y[0,\infty) \neq \emptyset) \asymp \mathbb{P}(X \cap Y[0,\infty) \neq \emptyset)\,.$$

$\rightsquigarrow$ $\mathbb{E}[\mathrm{Cap}(\mathrm{LE}(X[0,n]))] \asymp \mathbb{E}[\mathrm{Cap}(X[0,n])] \asymp \frac{n}{\log n}$.

Capacity

Theorem (Lyons, Peres and Schramm)

Let X and Y be transient chains. Then

$$\mathbb{P}(\mathrm{LE}(X) \cap Y[0,\infty) \neq \emptyset) \asymp \mathbb{P}(X \cap Y[0,\infty) \neq \emptyset).$$

$\rightsquigarrow \quad \mathbb{E}[\mathrm{Cap}(\mathrm{LE}(X[0,n]))] \asymp \mathbb{E}[\mathrm{Cap}(X[0,n])] \asymp \frac{n}{\log n}.$

Theorem (Lawler)

$n(\log n)^{1/3}$ *steps of SRW produce* n *steps of LERW.*

Capacity

Theorem (Lyons, Peres and Schramm)

Let X and Y be transient chains. Then

$$\mathbb{P}(\mathrm{LE}(X) \cap Y[0,\infty) \neq \emptyset) \asymp \mathbb{P}(X \cap Y[0,\infty) \neq \emptyset)\,.$$

$\rightsquigarrow \quad \mathbb{E}[\mathrm{Cap}(\mathrm{LE}(X[0,n]))] \asymp \mathbb{E}[\mathrm{Cap}(X[0,n])] \asymp \frac{n}{\log n}.$

Theorem (Lawler)

$n(\log n)^{1/3}$ steps of SRW produce n steps of LERW.

$\rightsquigarrow \quad \mathbb{E}[\mathrm{Cap}(\eta[0,n])] \asymp \mathbb{E}\left[\mathrm{Cap}(X[0,n(\log n)^{1/3}])\right] \asymp \frac{n}{(\log n)^{2/3}}.$

Back to the lower bound

Recall we showed

$$\mathbb{P}(\text{past of } 0 \text{ contains a path of length } n) \geq \varepsilon \cdot \frac{1}{(\log n)^{1/3}} \cdot \mathbb{E}\left[e^{-\varepsilon \mathrm{Cap}(\eta[0,n])}\right]$$

Back to the lower bound

Recall we showed

$$\mathbb{P}(\text{past of 0 contains a path of length } n) \geq \varepsilon \cdot \frac{1}{(\log n)^{1/3}} \cdot \mathbb{E}\left[e^{-\varepsilon \mathrm{Cap}(\eta[0,n])}\right]$$

Substitute $\mathrm{Cap}(\eta[0,n]) = \frac{n}{(\log n)^{2/3}}$ (typical value)

Back to the lower bound

Recall we showed

$$\mathbb{P}(\text{past of } 0 \text{ contains a path of length } n) \geq \varepsilon \cdot \frac{1}{(\log n)^{1/3}} \cdot \mathbb{E}\left[e^{-\varepsilon \mathrm{Cap}(\eta[0,n])}\right]$$

Substitute $\mathrm{Cap}(\eta[0,n]) = \frac{n}{(\log n)^{2/3}}$ (typical value)and take $\varepsilon = \frac{(\log n)^{2/3}}{n}$

Back to the lower bound

Recall we showed

$$\mathbb{P}(\text{past of } 0 \text{ contains a path of length } n) \geq \varepsilon \cdot \frac{1}{(\log n)^{1/3}} \cdot \mathbb{E}\left[e^{-\varepsilon \mathrm{Cap}(\eta[0,n])}\right]$$

Substitute $\mathrm{Cap}(\eta[0,n]) = \frac{n}{(\log n)^{2/3}}$ (typical value)and take $\varepsilon = \frac{(\log n)^{2/3}}{n}$

$$\mathbb{P}(\text{past of } 0 \text{ contains a path of length } n) \geq \frac{(\log n)^{1/3}}{n}.$$

Upper bound

Upper bound: **more delicate**

Upper bound

Upper bound: **more delicate**

Let $Q(n) = \mathbb{P}($ past of 0 contains a path of length $n)$

Upper bound

Upper bound: **more delicate**

Let $Q(n) = \mathbb{P}($ past of 0 contains a path of length $n)$

Idea prove an inductive inequality

$$Q(2n) \leq \frac{C(\log n)^{1/3}}{n} + \frac{1}{4}Q(n)$$

Upper bound

Upper bound: **more delicate**

Let $Q(n) = \mathbb{P}($ past of 0 contains a path of length $n)$

Idea prove an inductive inequality

$$Q(2n) \leq \frac{C(\log n)^{1/3}}{n} + \frac{1}{4}Q(n)$$

Useful tools developed in work with **Asselah and Schapira** on the capacity of the range of a SRW.

$d = 4$

Theorem (Hutchcroft and S. (2020) d=4)

$$\mathbb{P}(\textit{past of } 0 \text{ contains a path of length } n) \asymp \frac{(\log n)^{1/3}}{n}$$

$$\mathbb{P}(\textit{past of } 0 \cap \partial B(0,n) \neq \emptyset) \asymp \frac{(\log n)^{2/3+o(1)}}{n^2}$$

$$\mathbb{P}(|\textit{past of } 0| \geq n) \asymp \frac{(\log n)^{1/6}}{\sqrt{n}}$$