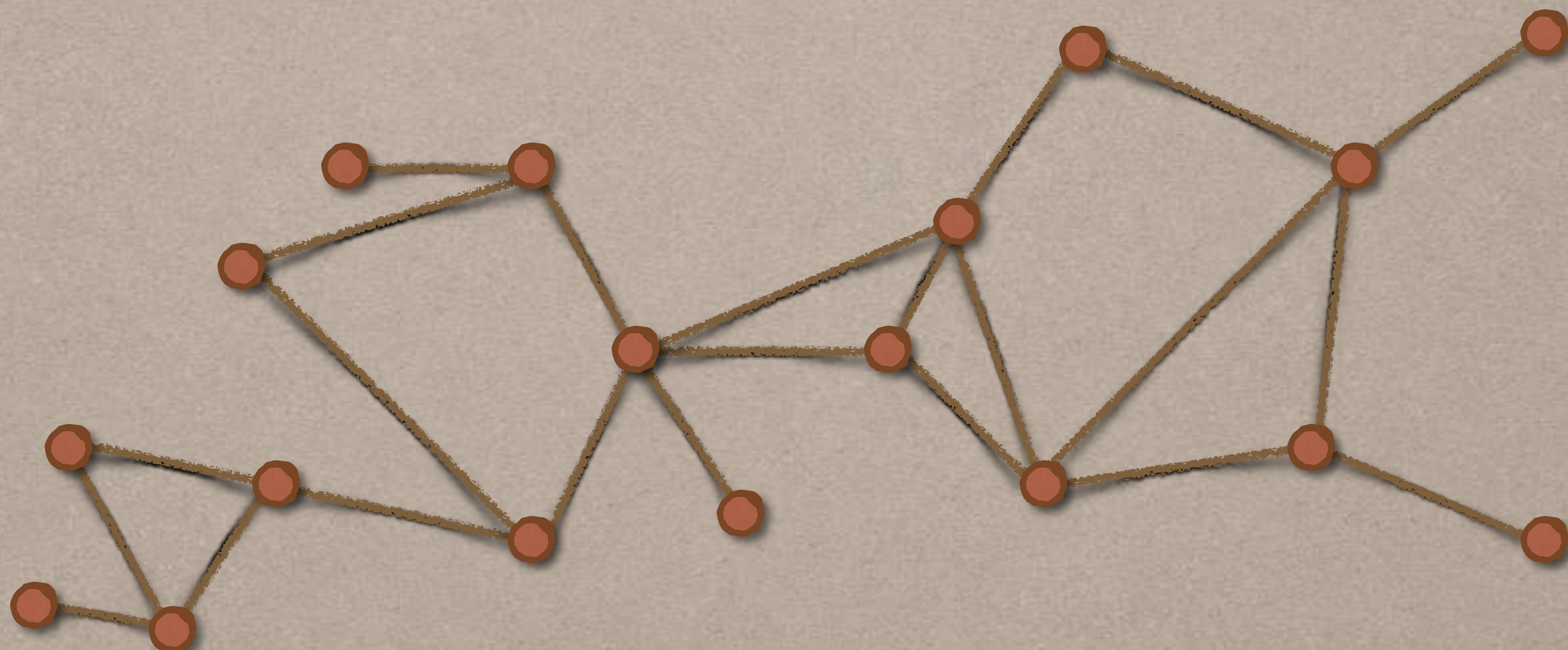
THE SCALING LIMIT OF RANDOM OUTERPLANAR MAPS

ALESSANDRA CARACENI
SCUOLA NORMALE SUPERIORE DI PISA

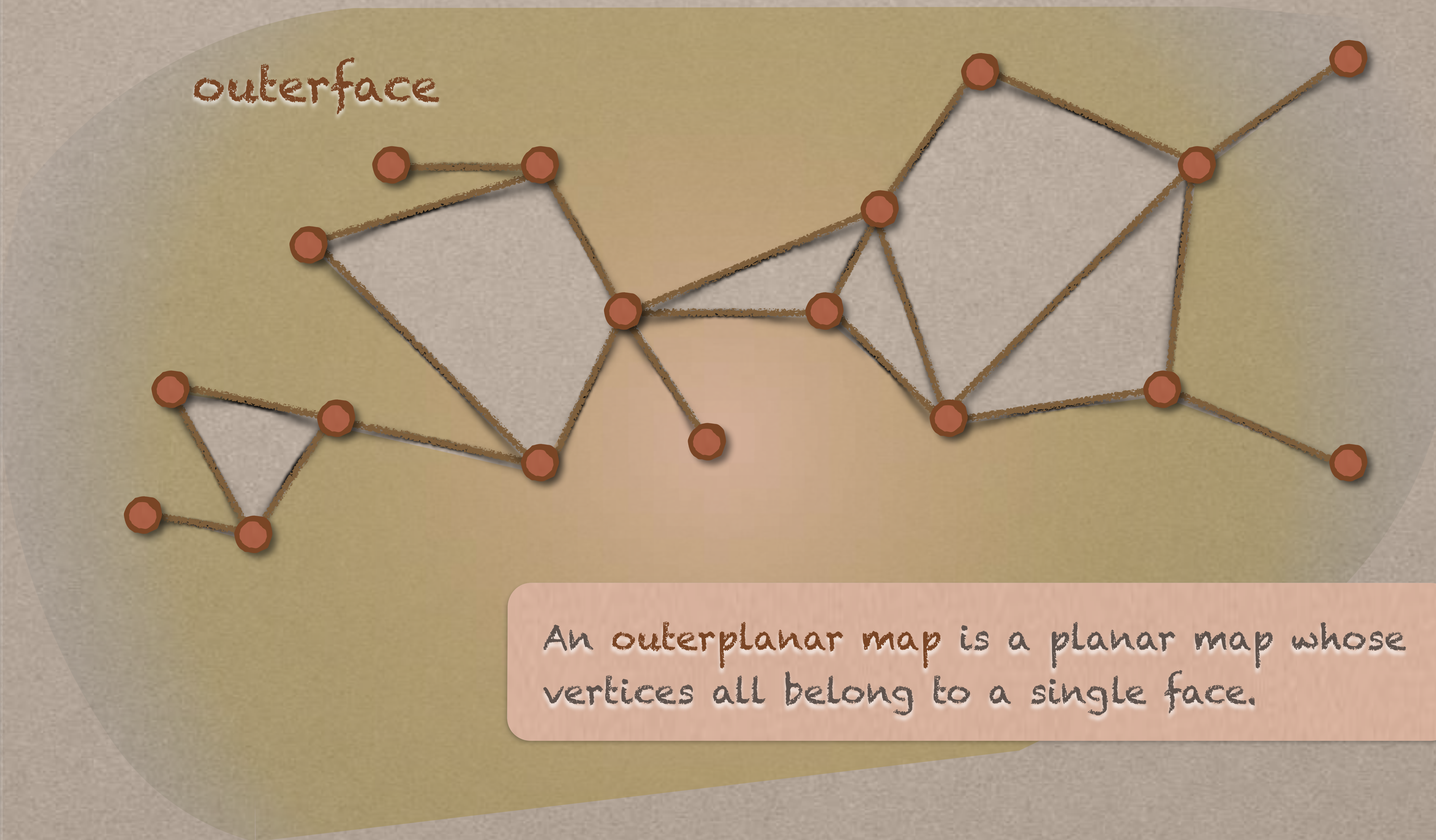
29TH APRIL 2014

OUTERPLANAR MAPS



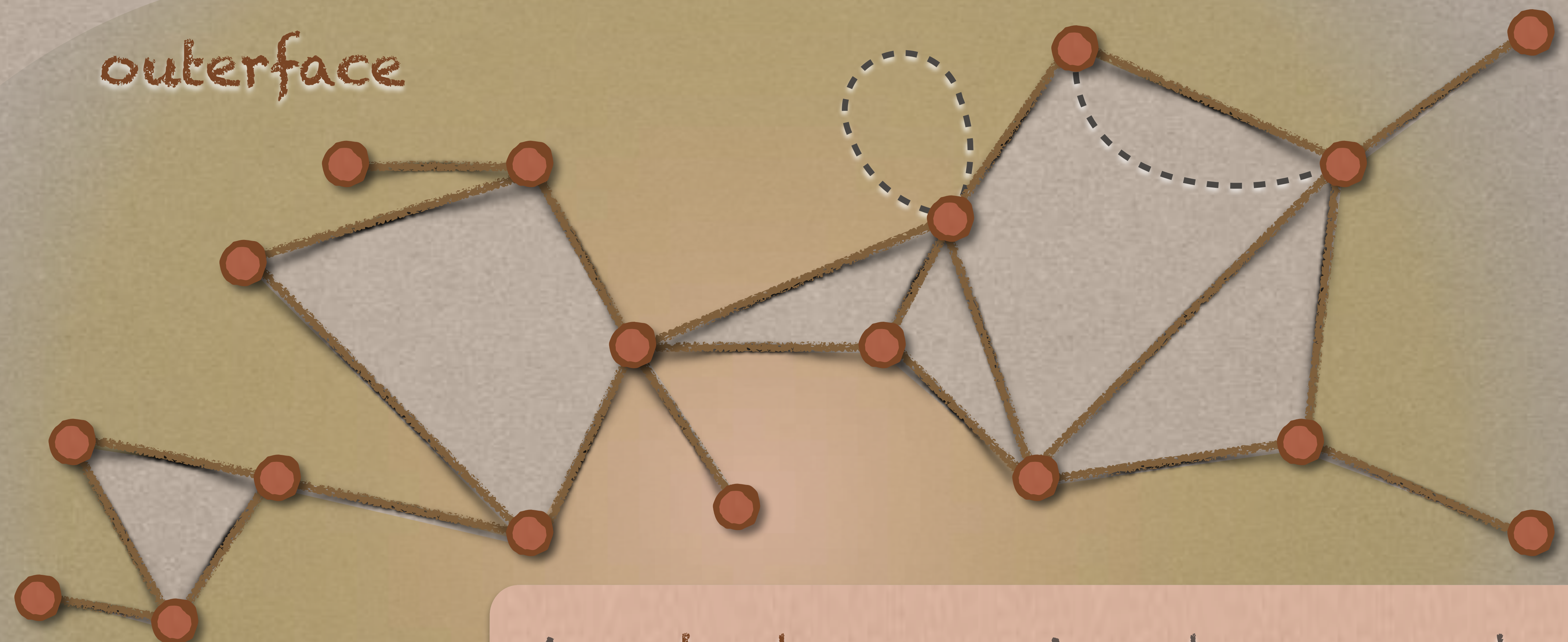
An outerplanar map is a planar map whose vertices all belong to a single face.

OUTERPLANAR MAPS



An outerplanar map is a planar map whose vertices all belong to a single face.

OUTERPLANAR MAPS

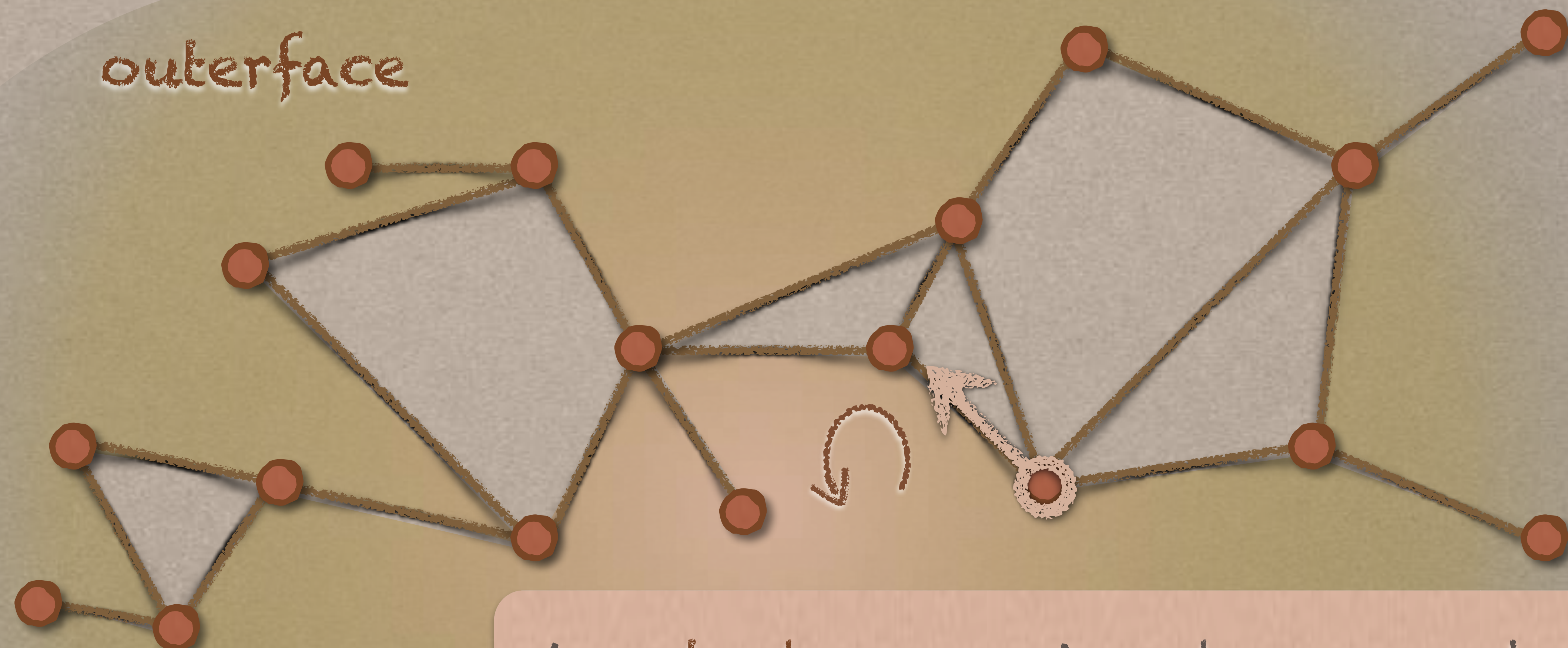


An **outerplanar map** is a planar map whose vertices all belong to a single face.

Outerplanar maps we consider are

- simple

OUTERPLANAR MAPS

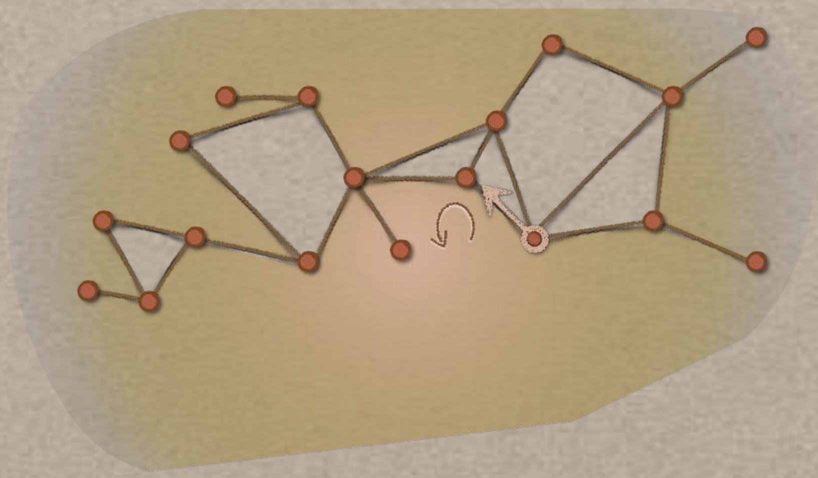


An **outerplanar map** is a planar map whose vertices all belong to a single face.

Outerplanar maps we consider are

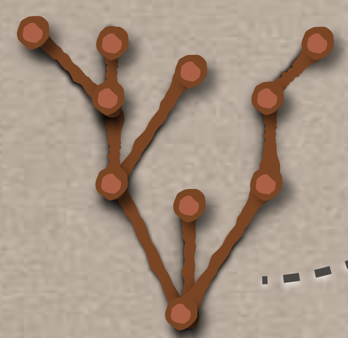
- simple
- rooted

SCALING LIMITS

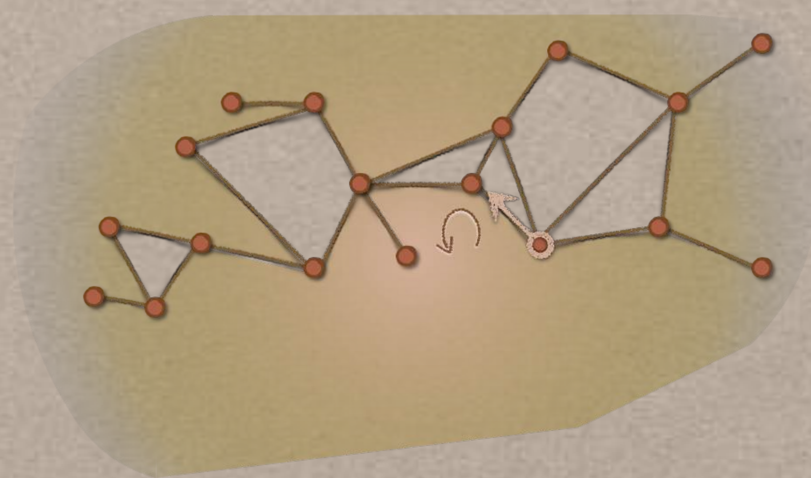


SCALING LIMITS

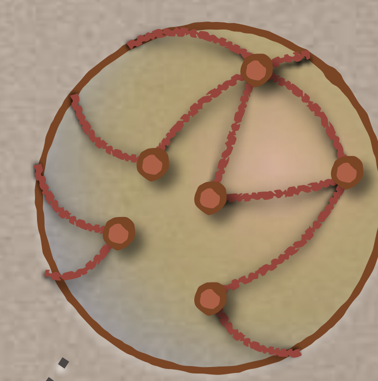
random plane trees
with n vertices



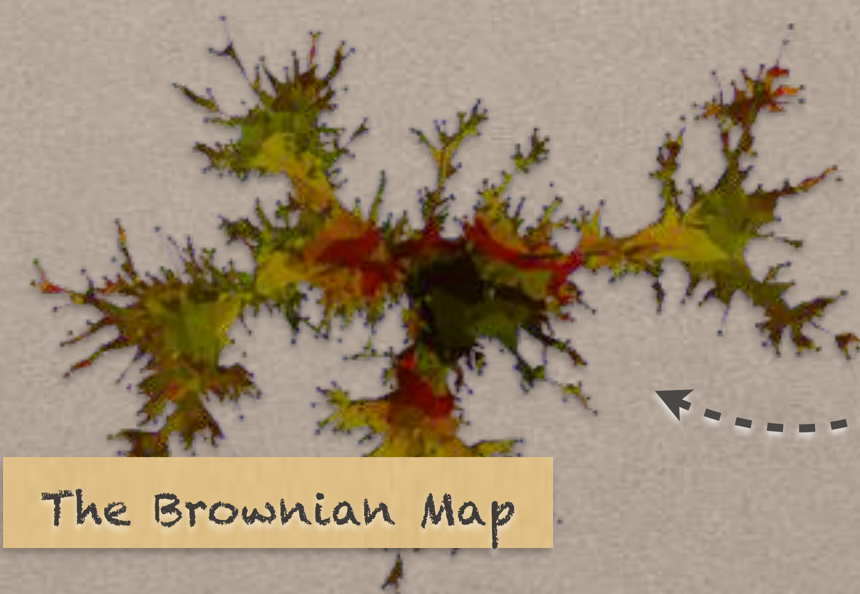
scale by $n^{1/2}$



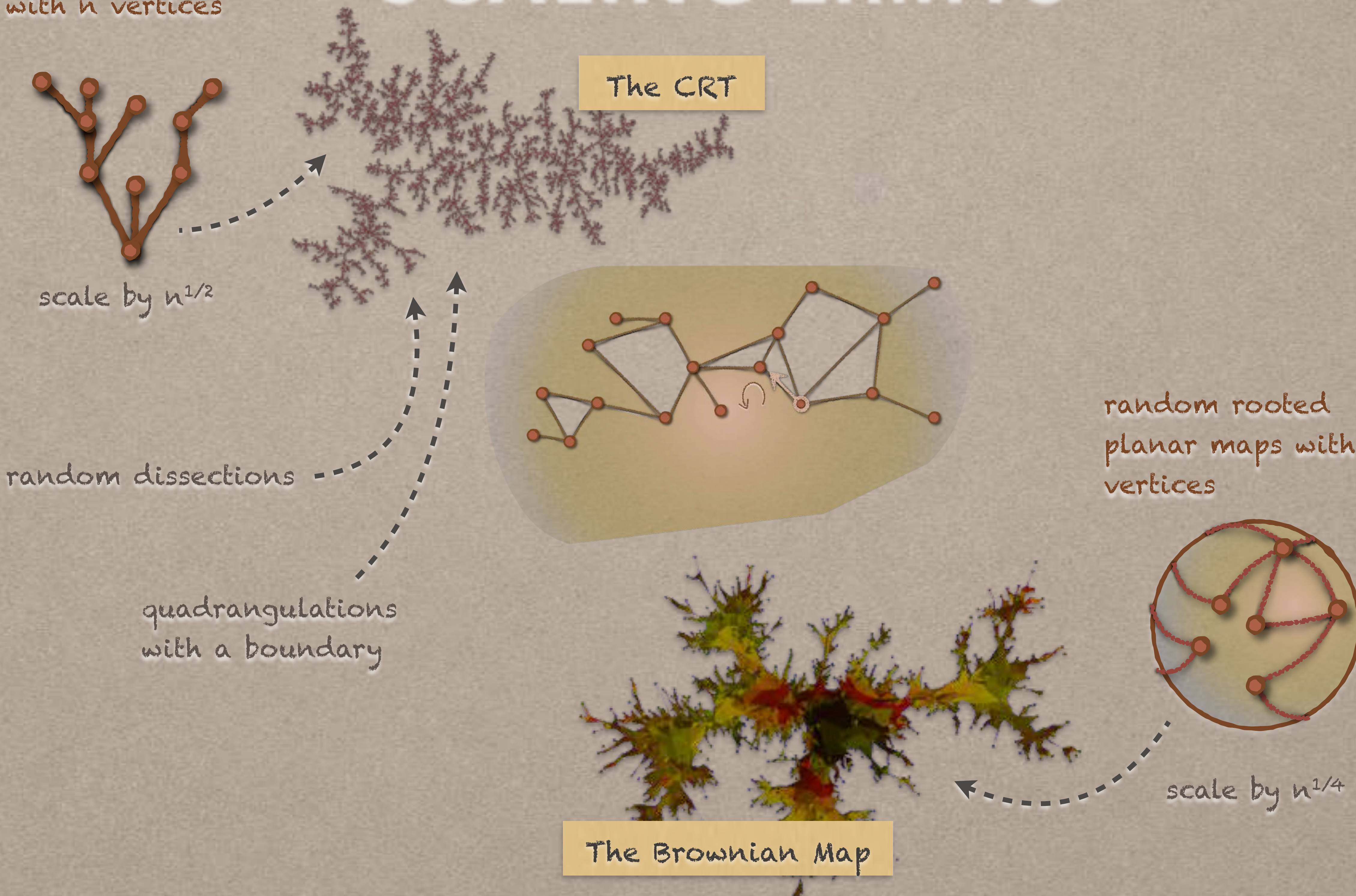
random rooted
planar maps with n
vertices



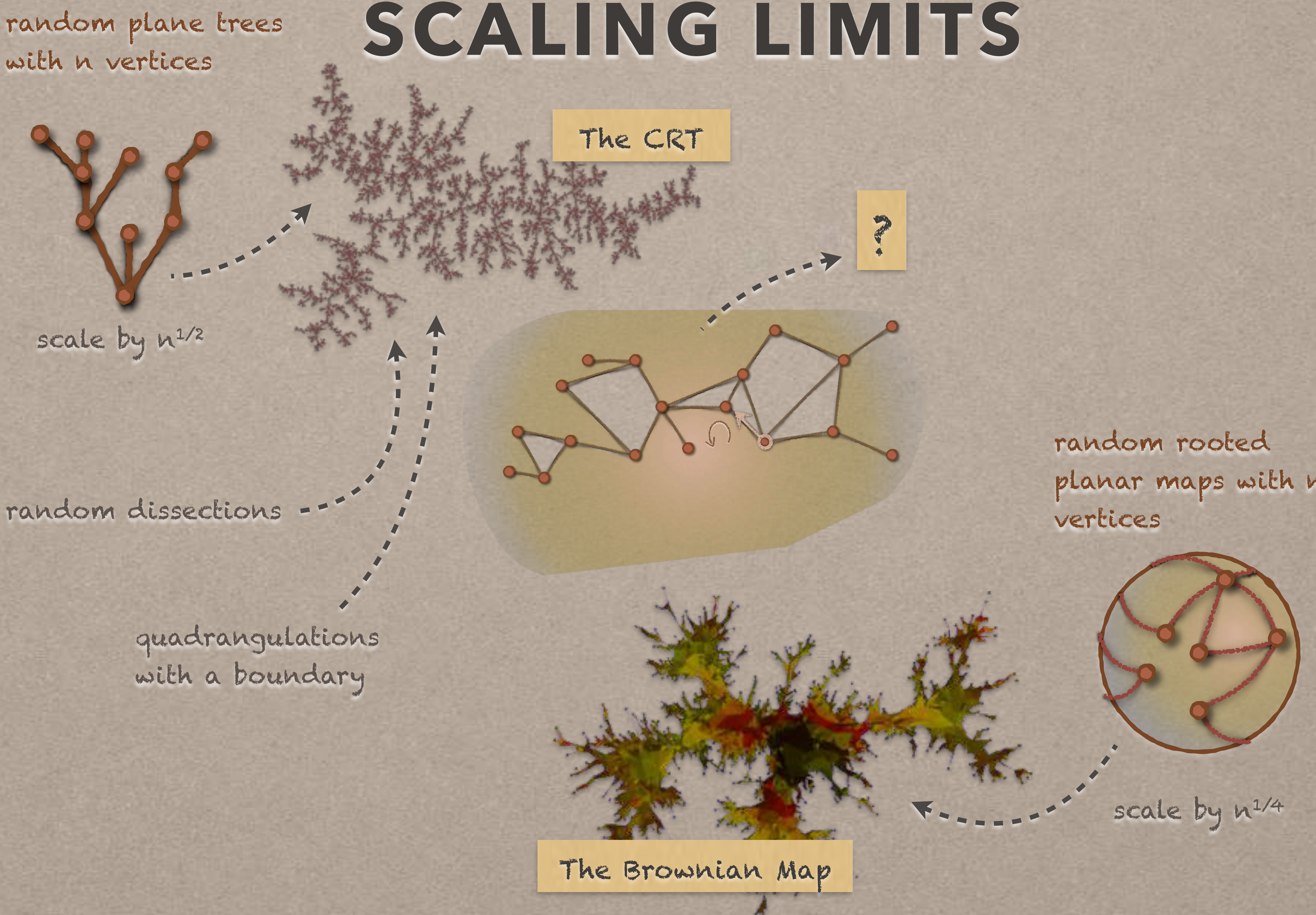
scale by $n^{1/4}$

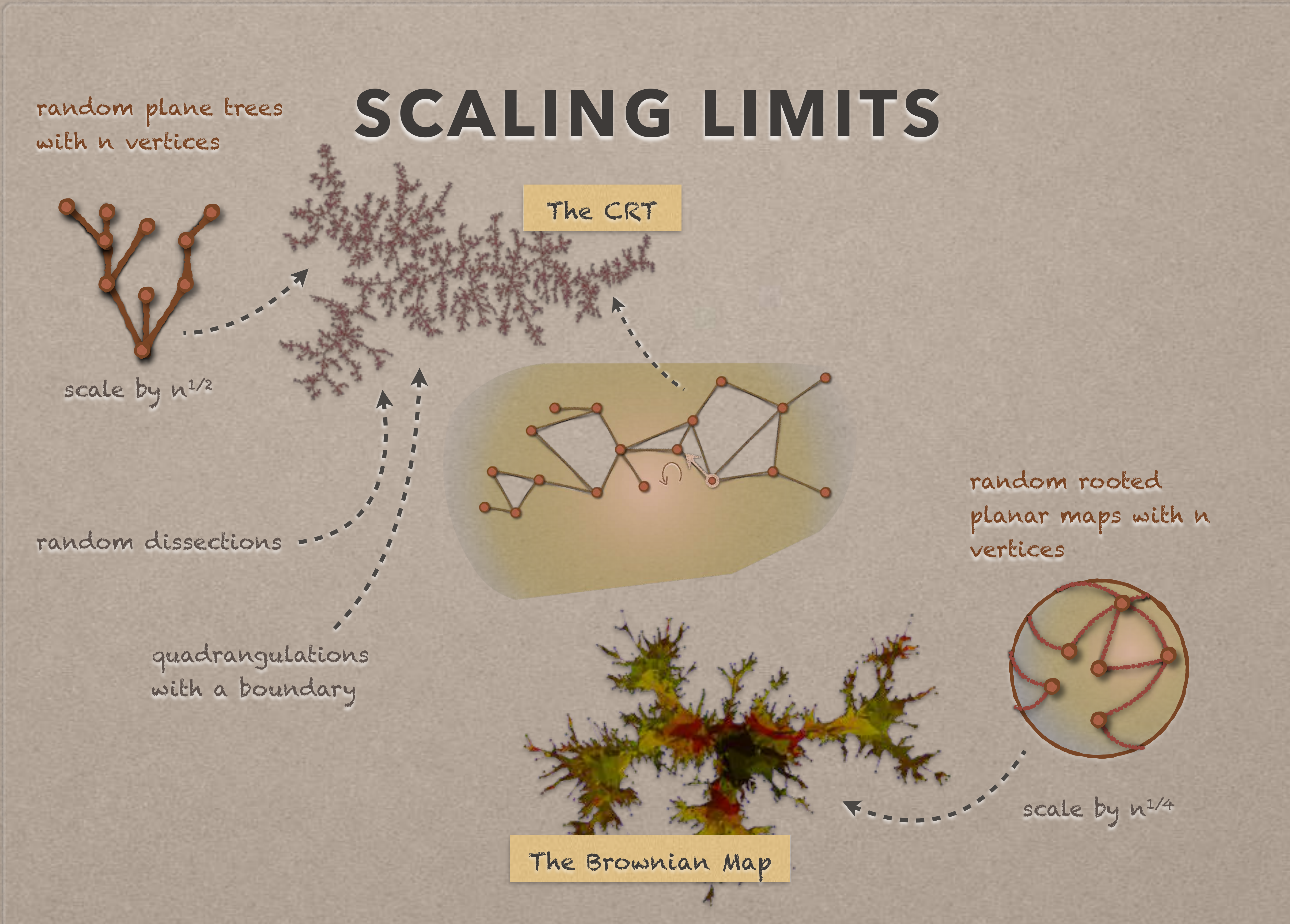


SCALING LIMITS



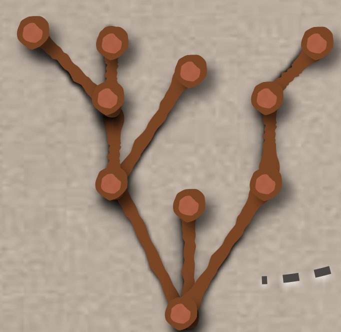
SCALING LIMITS




SCALING LIMITS
random plane trees with n vertices
scale by $n^{1/2}$
The CRT
random dissections
quadrangulations with a boundary
random rooted planar maps with n vertices
scale by $n^{1/4}$
The Brownian Map

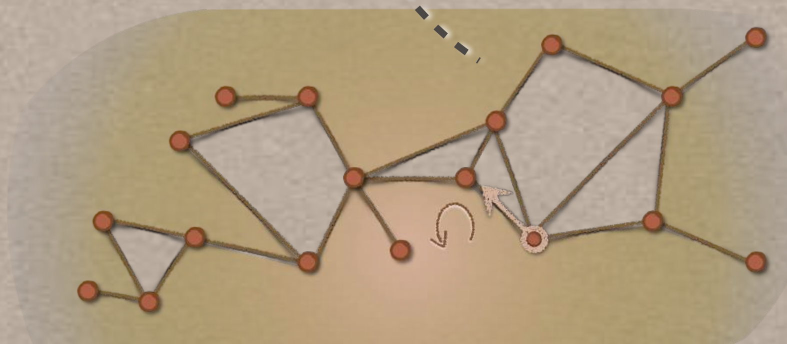
SCALING LIMITS

random plane trees with n vertices



scale by $n^{1/2}$

The CRT



[C. 2014]

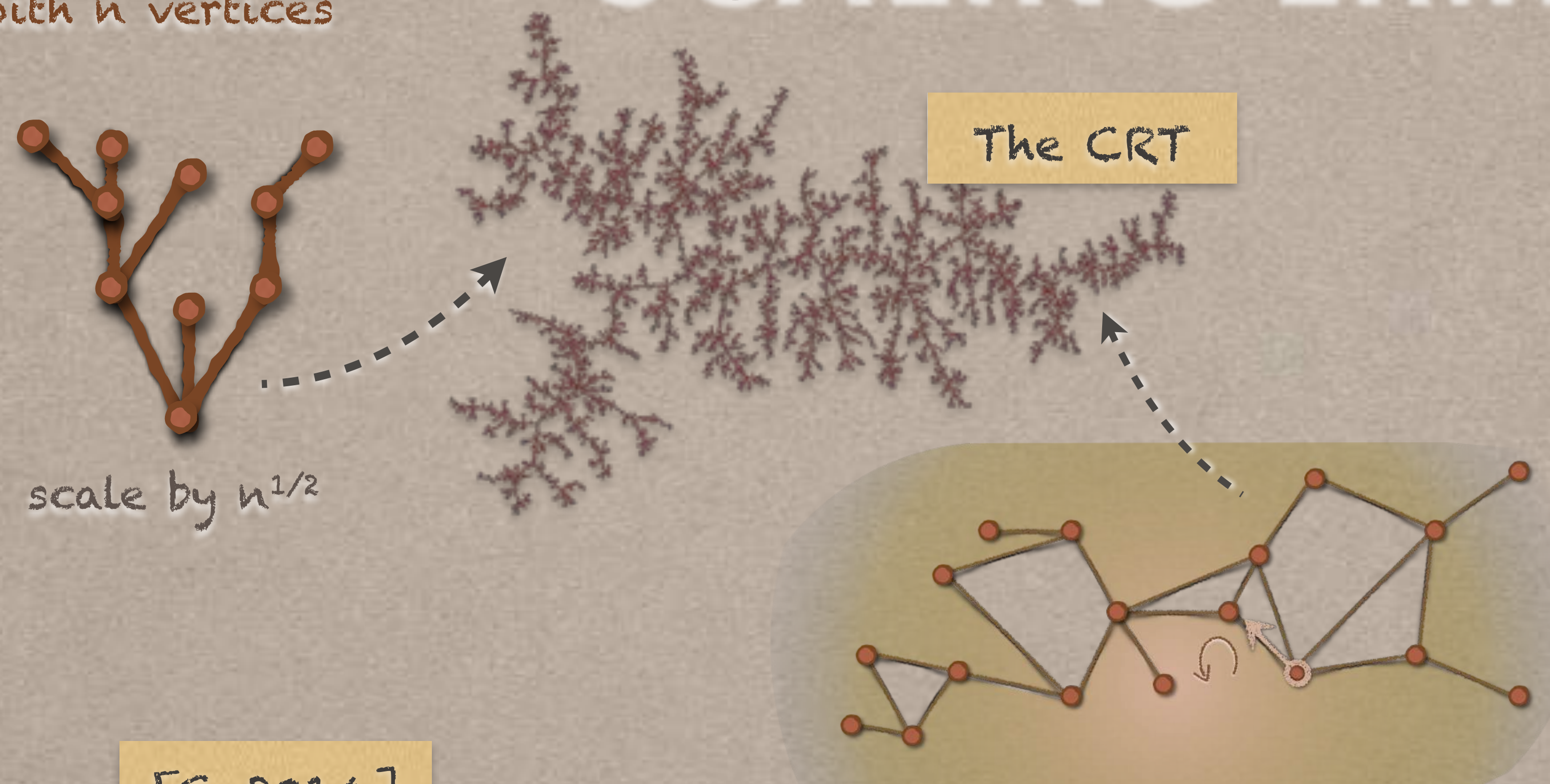
Let $\mathbf{M}_n$ be a random uniform rooted simple outerplanar map with n vertices, and denote by d_{gr} the graph distance on the set of its vertices $V(\mathbf{M}_n)$. We have the following convergence in distribution for the Gromov-Hausdorff topology:

$$\left(V(\mathbf{M}_n), \frac{d_{gr}}{\sqrt{n}}\right) \xrightarrow[n\to\infty]{(d)} \frac{7\sqrt{2}}{9} \cdot (\mathcal{T}_e, d)$$

where $(\mathcal{T}_e, d)$ is the Brownian CRT of Aldous (the normalisation is that of Le Gall: consider $(\mathcal{T}_e, d)$ as constructed from a normalised Brownian excursion).

SCALING LIMITS

random plane trees
with n vertices



?

"Why are outerplanar maps like trees?"

[C. 2014]

Let $\mathbf{M}_n$ be a random uniform rooted simple outerplanar map with n vertices, and denote by d_{gr} the graph distance on the set of its vertices $V(\mathbf{M}_n)$. We have the following convergence in distribution for the Gromov-Hausdorff topology:

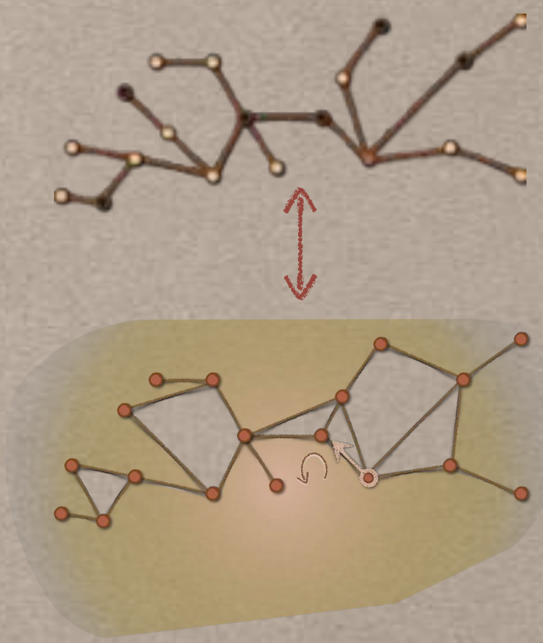
$$\left(V(\mathbf{M}_n), \frac{d_{gr}}{\sqrt{n}} \right) \xrightarrow[n\to\infty]{(d)} \frac{7\sqrt{2}}{9} \cdot (\mathcal{T}_e, d)$$

where $(\mathcal{T}_e, d)$ is the Brownian CRT of Aldous (the normalisation is that of Le Gall: consider $(\mathcal{T}_e, d)$ as constructed from a normalised Brownian excursion).

PLAN OF PROOF

- There is a bijection between outerplanar maps with n vertices and a class of bicoloured plane trees with n vertices. [Bonichon, Gavoille, Hanusse, 05]

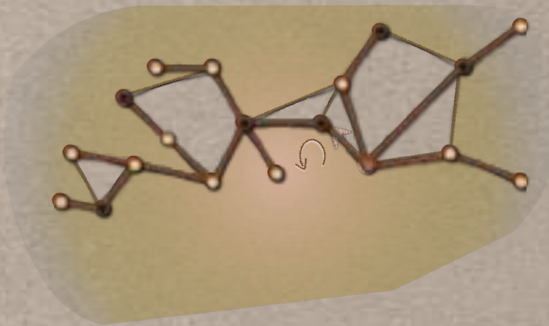
? "Why are outerplanar maps like trees?"



PLAN OF PROOF

- There is a bijection between outerplanar maps with n vertices and a class of bicoloured plane trees with n vertices. [Bonichon, Gavoille, Hanusse, 05]

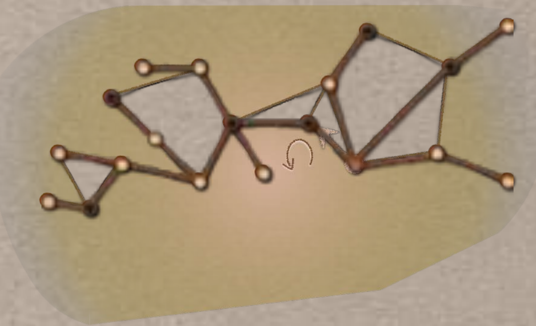
? "Why are outerplanar maps like trees?"



PLAN OF PROOF

?

"Why are outerplanar maps like trees?"



- There is a bijection between outerplanar maps with n vertices and a class of bicoloured plane trees with n vertices. [Bonichon, Gavoille, Hanusse, 05]
- We need a way to relate the "map" metric to the graph distances on the corresponding tree.

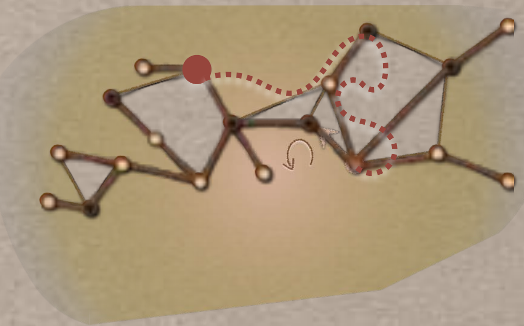
PLAN OF PROOF

- There is a bijection between outerplanar maps with n vertices and a class of bicoloured plane trees with n vertices. [Bonichon, Gavoille, Hanusse, 05]

- We need a way to relate the "map" metric to the graph distances on the corresponding tree.
    - We develop an explicit algorithm that, given a bicoloured plane tree and a vertex, computes the map-distance between said vertex and the root.

?

"Why are outerplanar maps like trees?"

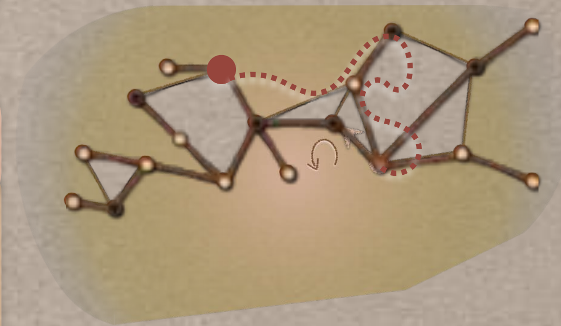


PLAN OF PROOF

- There is a bijection between outerplanar maps with n vertices and a class of bicoloured plane trees with n vertices. [Bonichon, Gavoille, Hanusse, 05]

- We need a way to relate the "map" metric to the graph distances on the corresponding tree.
  - We develop an explicit algorithm that, given a bicoloured plane tree and a vertex, computes the map-distance between said vertex and the root.
  - We apply the algorithm to a random tree and obtain that, for "most" vertices u in the tree, their map-distance from the root is ~c|u|.

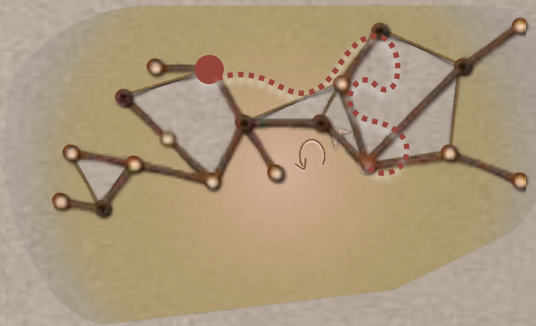
? "Why are outerplanar maps like trees?"



PLAN OF PROOF

?

"Why are outerplanar maps like trees?"



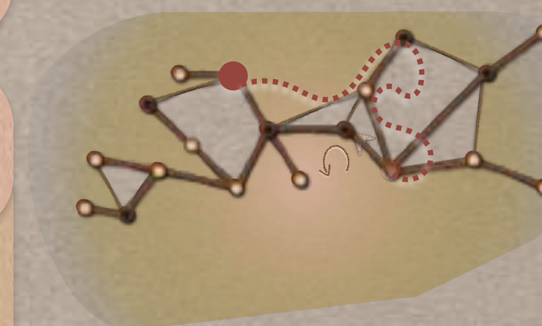
- There is a bijection between outerplanar maps with n vertices and a class of bicoloured plane trees with n vertices. [Bonichon, Gavoille, Hanusse, 05]
- We need a way to relate the "map" metric to the graph distances on the corresponding tree.
    - We develop an explicit algorithm that, given a bicoloured plane tree and a vertex, computes the map-distance between said vertex and the root.
    - We apply the algorithm to a random tree and obtain that, for "most" vertices u in the tree, their map-distance from the root is ~c|u|.
    - We need to extend estimates to all pairs of vertices. We obtain, in fact, that in a random tree with n vertices distances multiplied by c differ from map-distances by more than $\epsilon\sqrt{n}$ with probability that is infinitesimal in n.

PLAN OF PROOF

?

"Why are outerplanar maps like trees?"

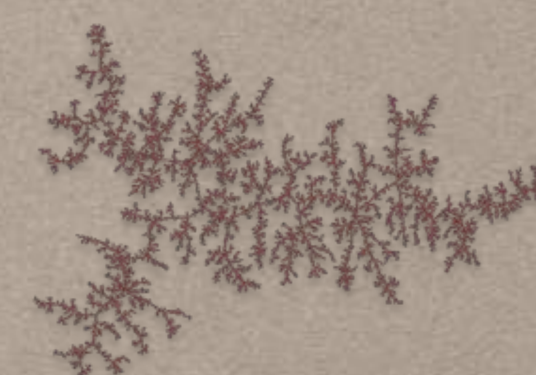
- There is a bijection between outerplanar maps with n vertices and a class of bicoloured plane trees with n vertices. [Bonichon, Gavoille, Hanusse, 05]



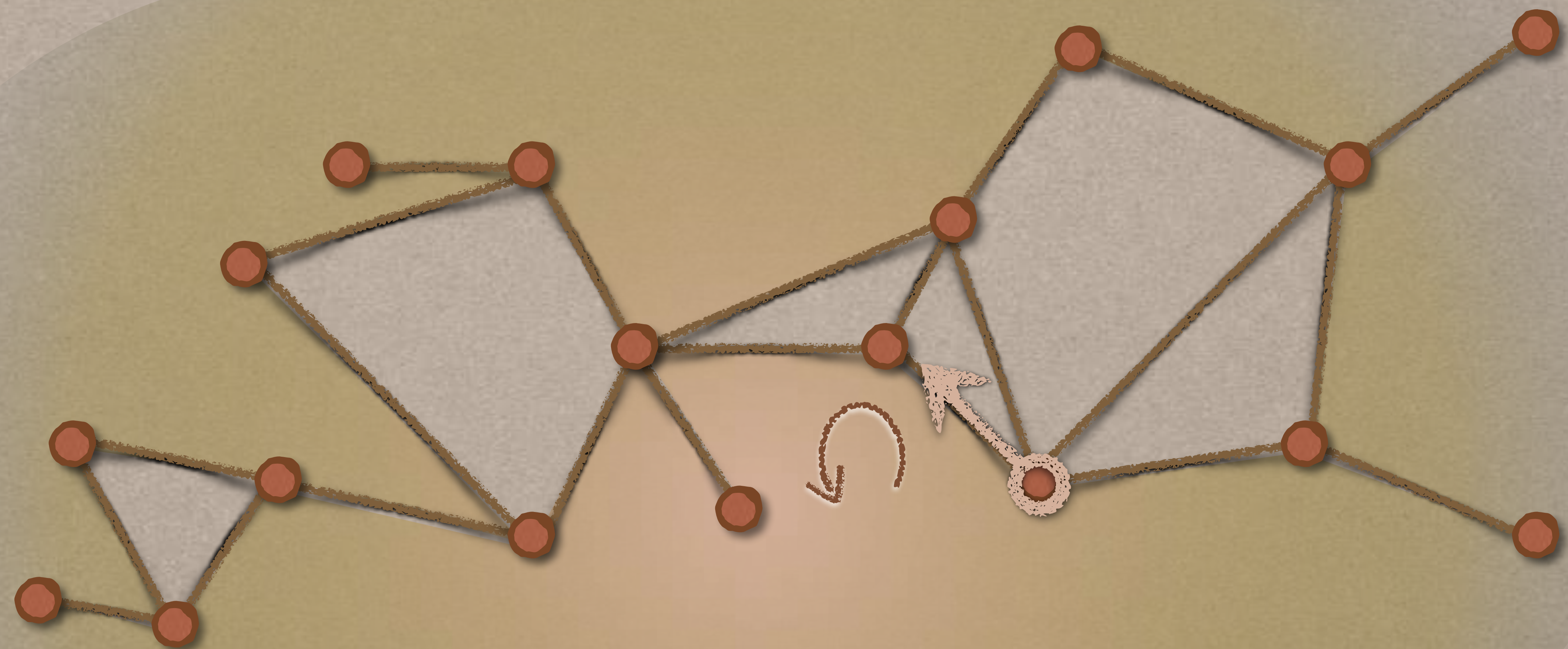
- We need a way to relate the "map" metric to the graph distances on the corresponding tree.
  - We develop an explicit algorithm that, given a bicoloured plane tree and a vertex, computes the map-distance between said vertex and the root.
  - We apply the algorithm to a random tree and obtain that, for "most" vertices u in the tree, their map-distance from the root is ~c|u|.
  - We need to extend estimates to all pairs of vertices. We obtain, in fact, that in a random tree with n vertices distances multiplied by c differ from map-distances by more than $\epsilon\sqrt{n}$ with probability that is infinitesimal in n.

$$\mathbb{P}\left(\sup_{u,v\in\tau_n} |d_M(u,v) - c d_T(u,v)| \geq \epsilon\sqrt{n}\right) \to 0$$

$$\lim_{n\to\infty} d_{GH}\left(\frac{\Psi(\tau_n)}{\sqrt{n}}, \frac{c\tau_n}{\sqrt{n}}\right) = 0$$



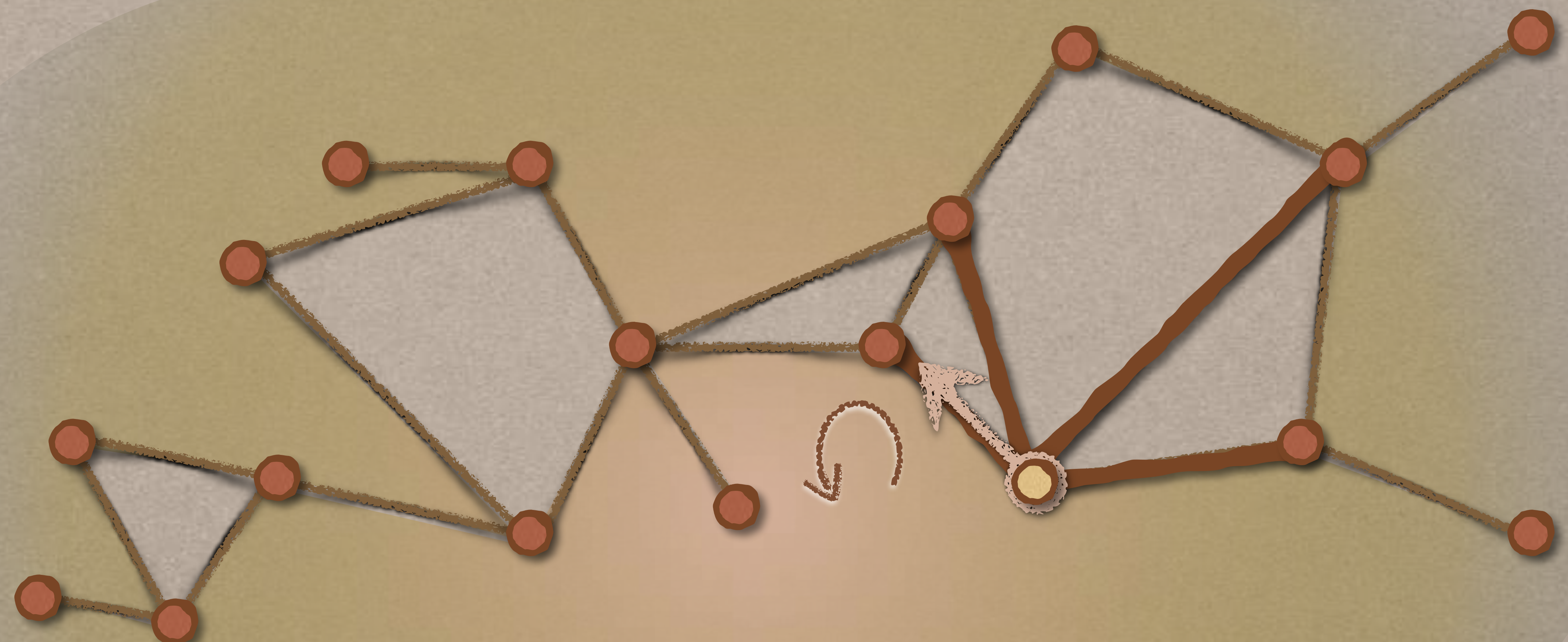
THE BGH BIJECTION



?

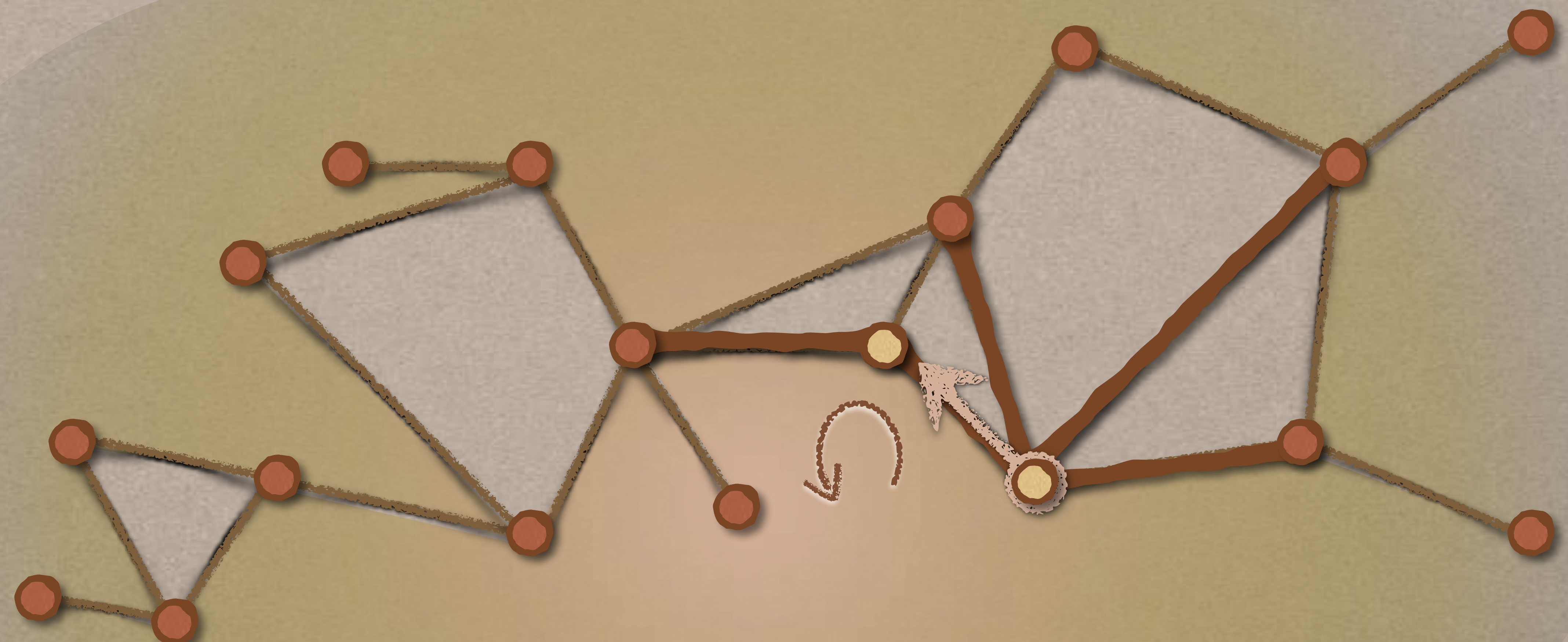
"Why are outerplanar maps like trees?"

THE BGH BIJECTION



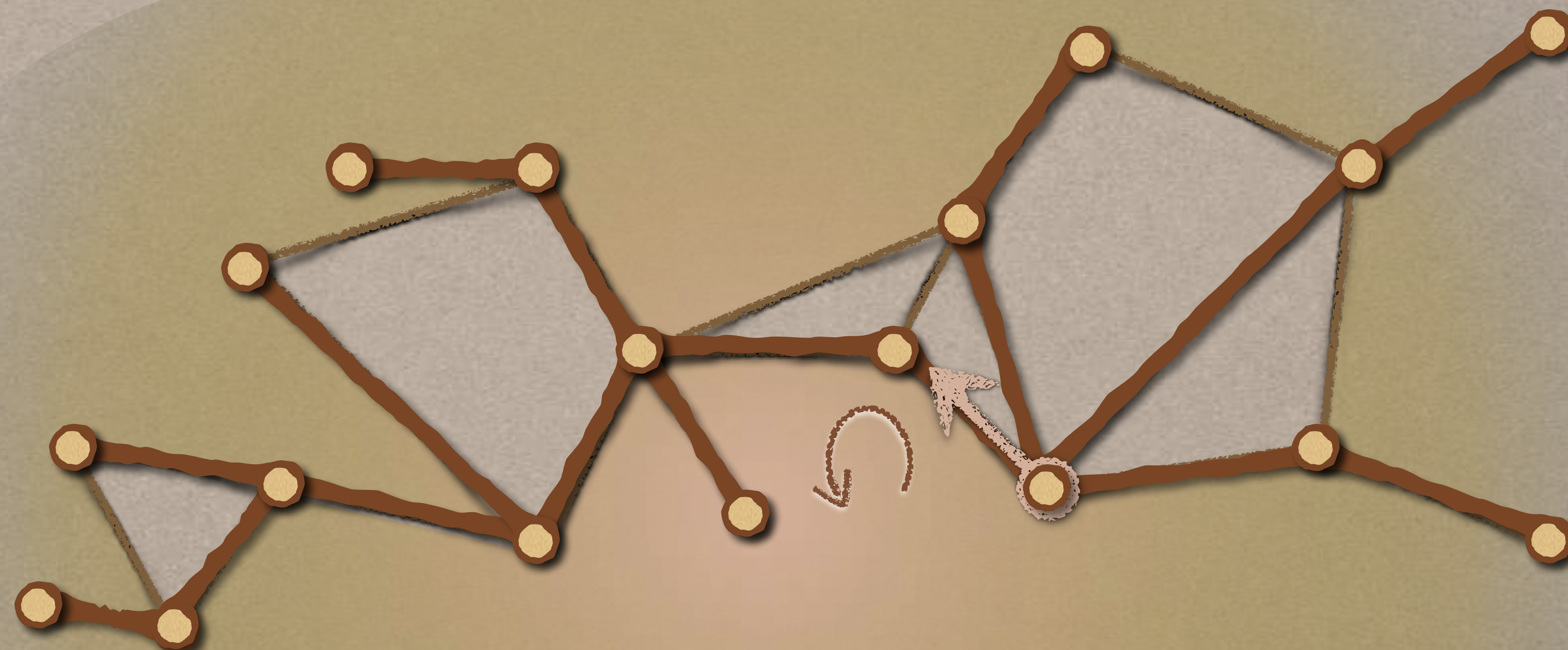
? "Why are outerplanar maps like trees?"

THE BGH BIJECTION



"Why are outerplanar maps like trees?"

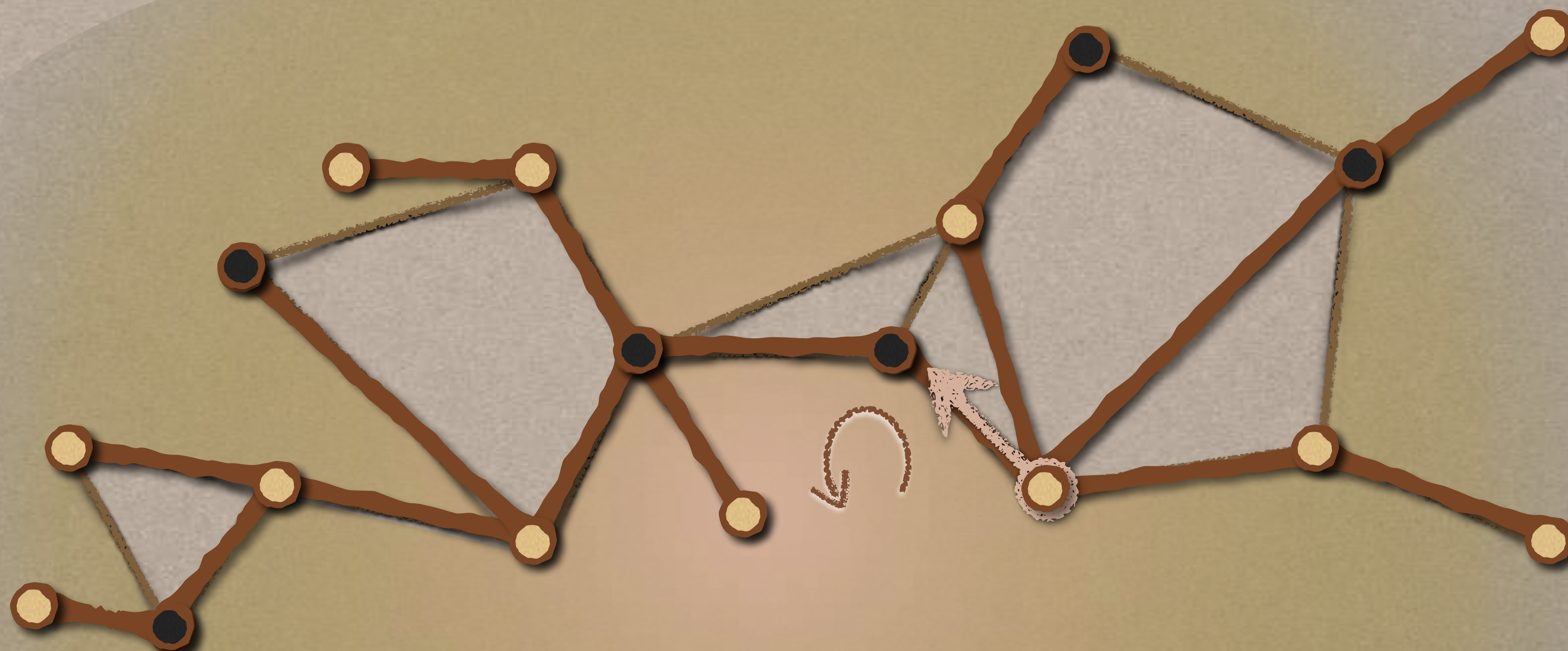
THE BGH BIJECTION



?

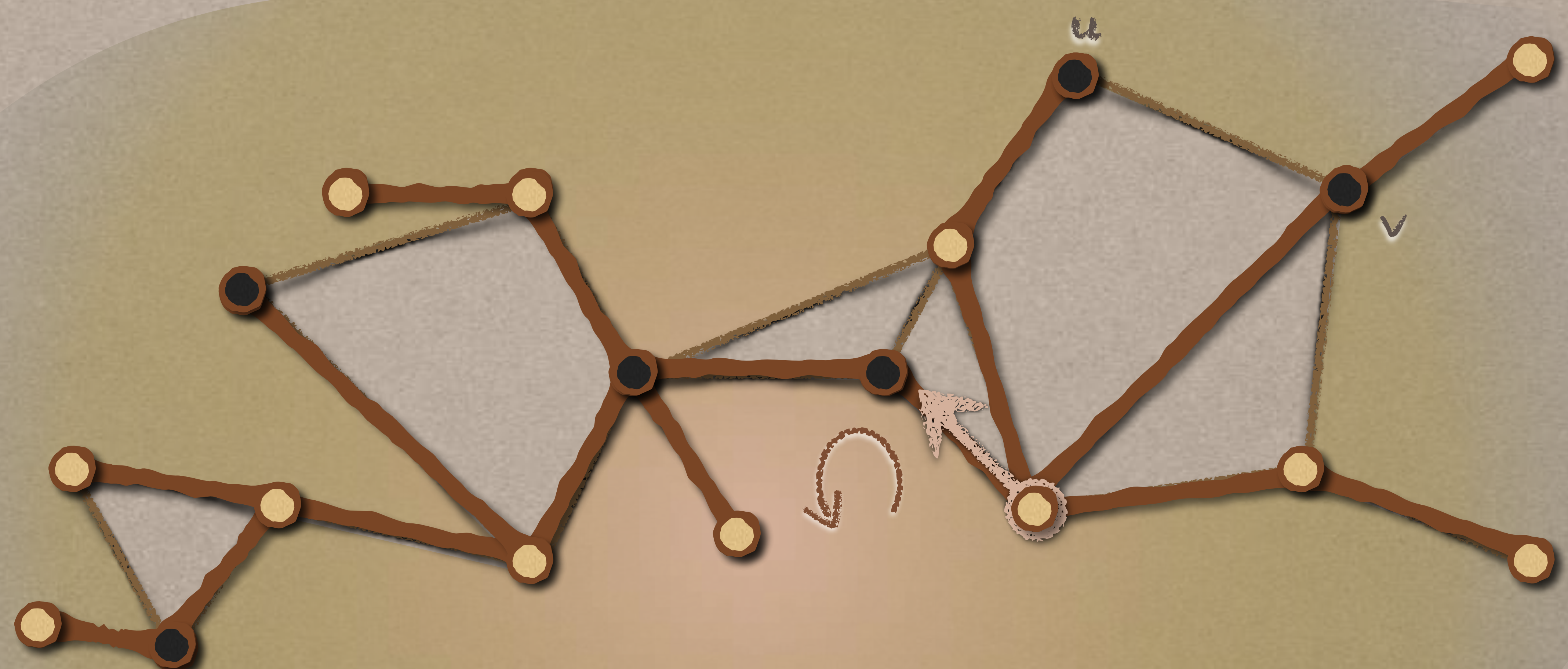
"Why are outerplanar maps like trees?"

THE BGH BIJECTION



"Why are outerplanar maps like trees?"

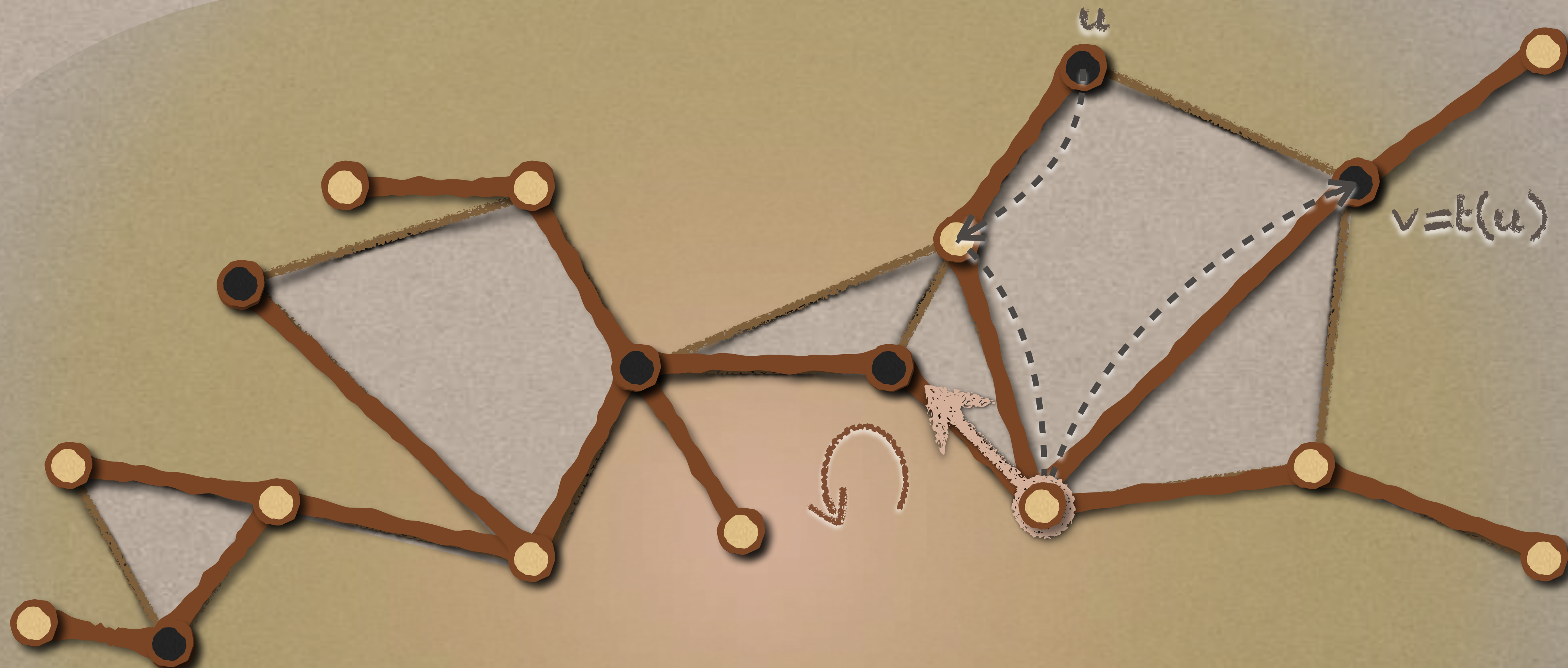
THE BGH BIJECTION



?

"Why are outerplanar maps like trees?"

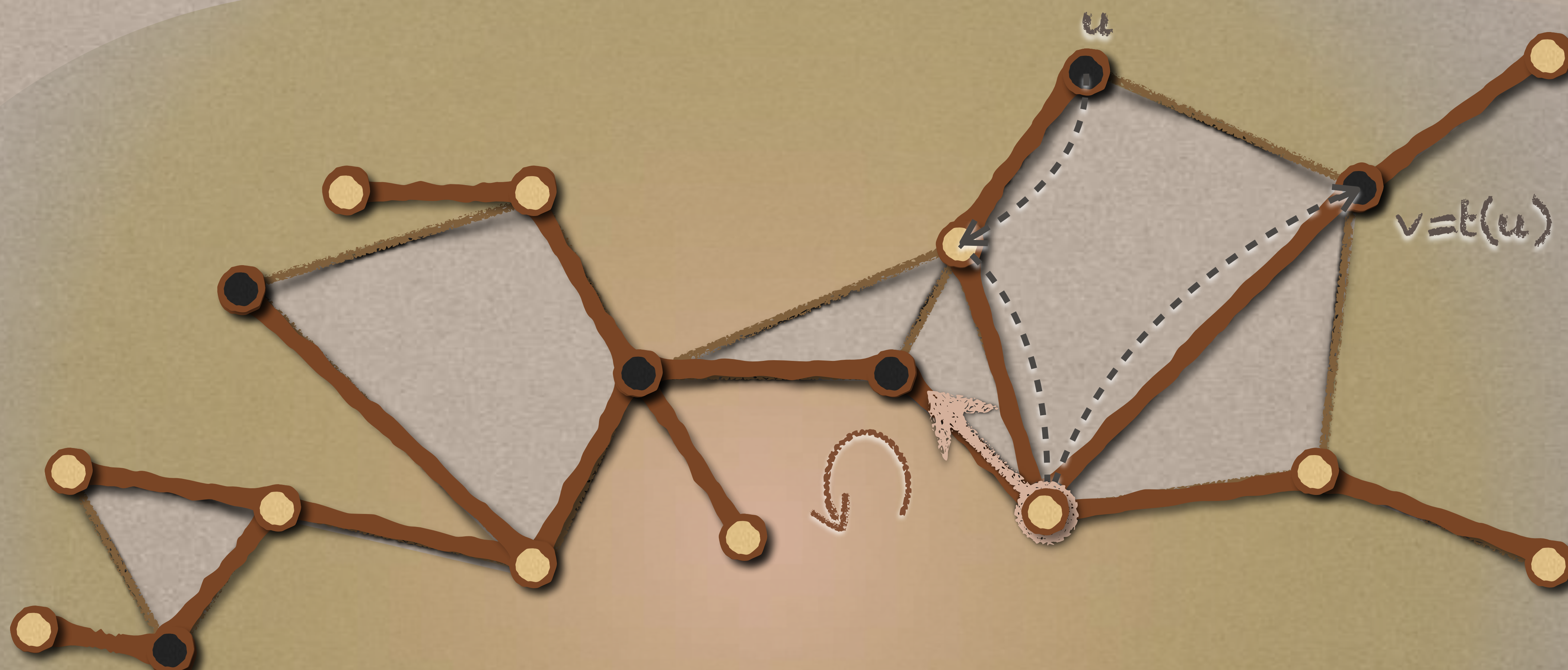
THE BGH BIJECTION



We call v the "target" of u and write v=t(u).
t(u) is the fist vertex unrelated to u to be met after u in a clockwise contour of the tree.

? "Why are outerplanar maps like trees?"

THE BGH BIJECTION



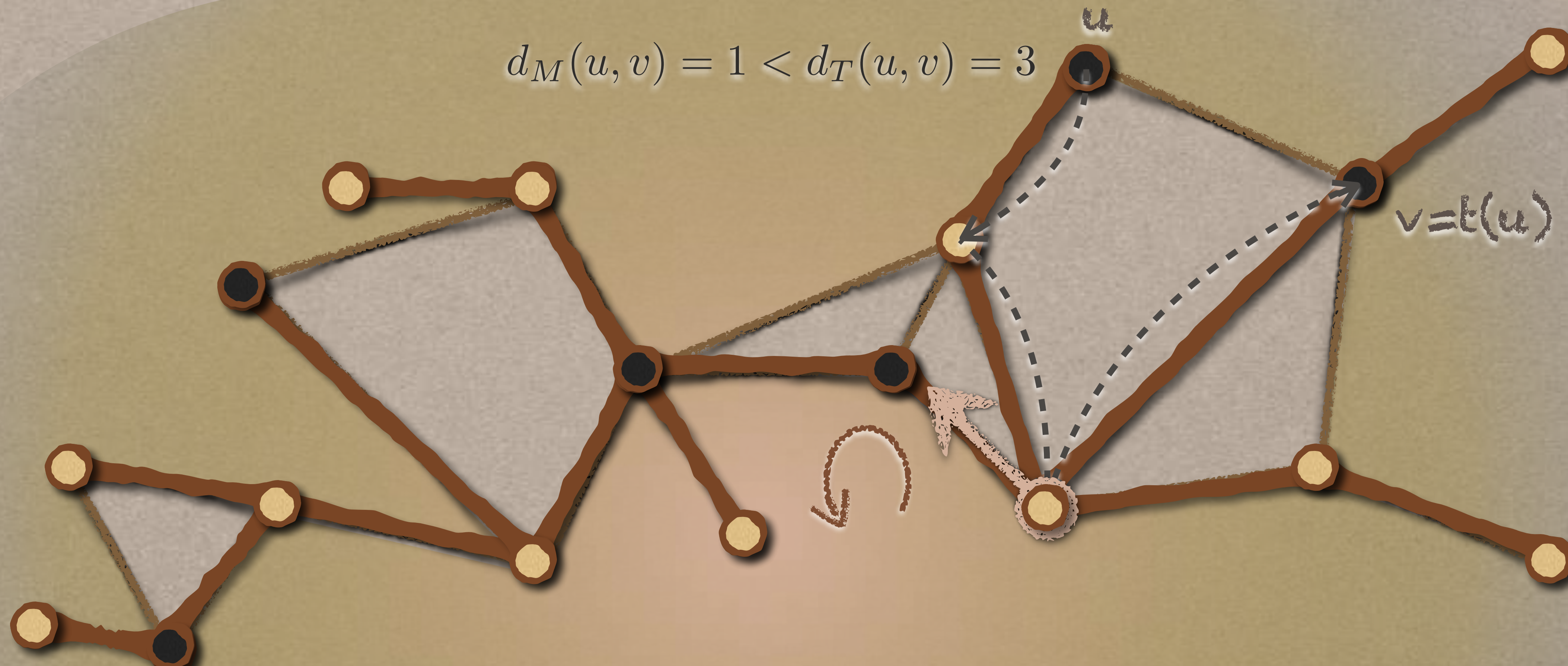
There is a bijection between (simple, rooted) outerplanar maps with n vertices and bicoloured plane trees with n vertices, with a white rightmost branch.

?

"Why are outerplanar maps like trees?"

THE BGH BIJECTION

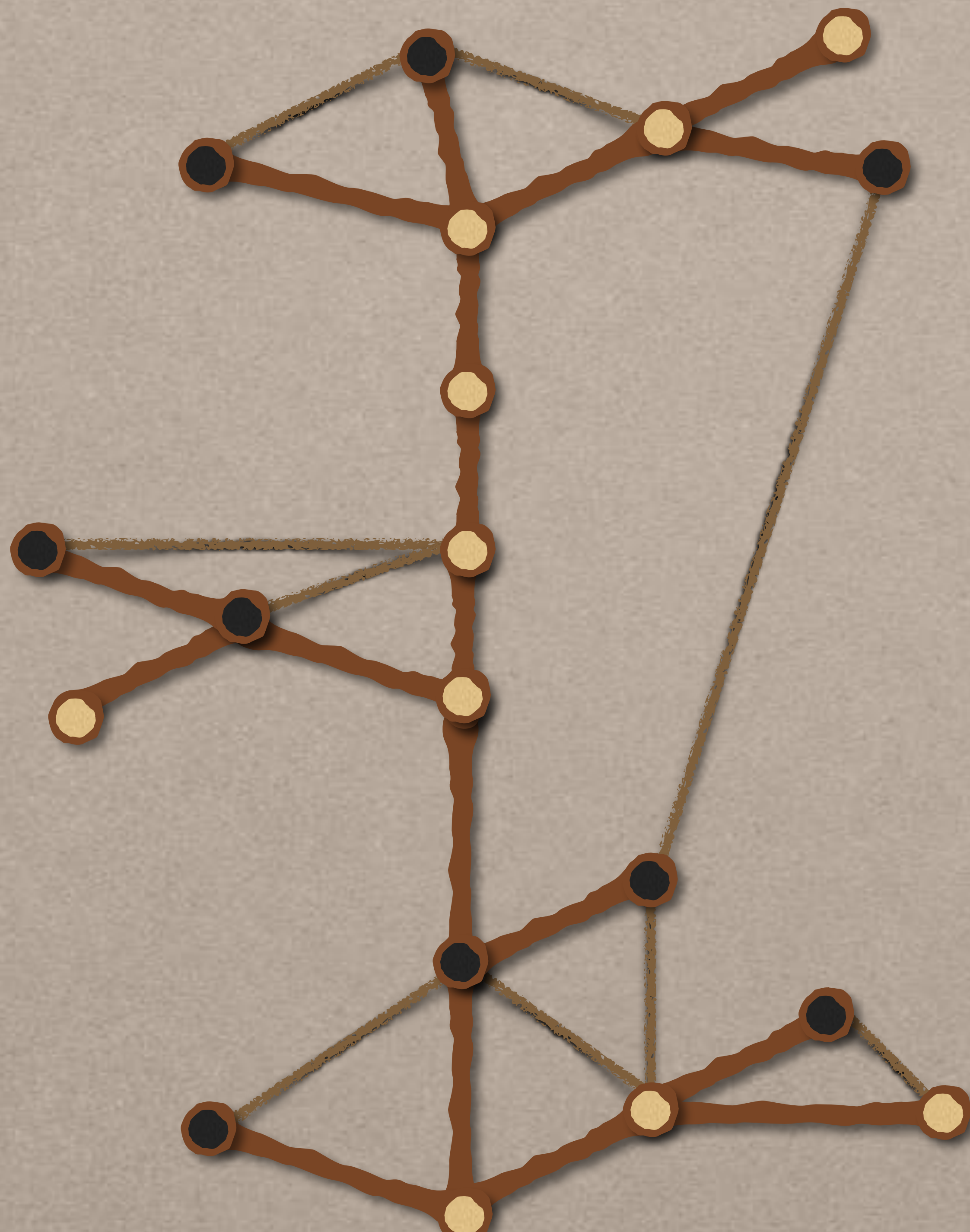
HOW DOES IT AFFECT DISTANCES ?



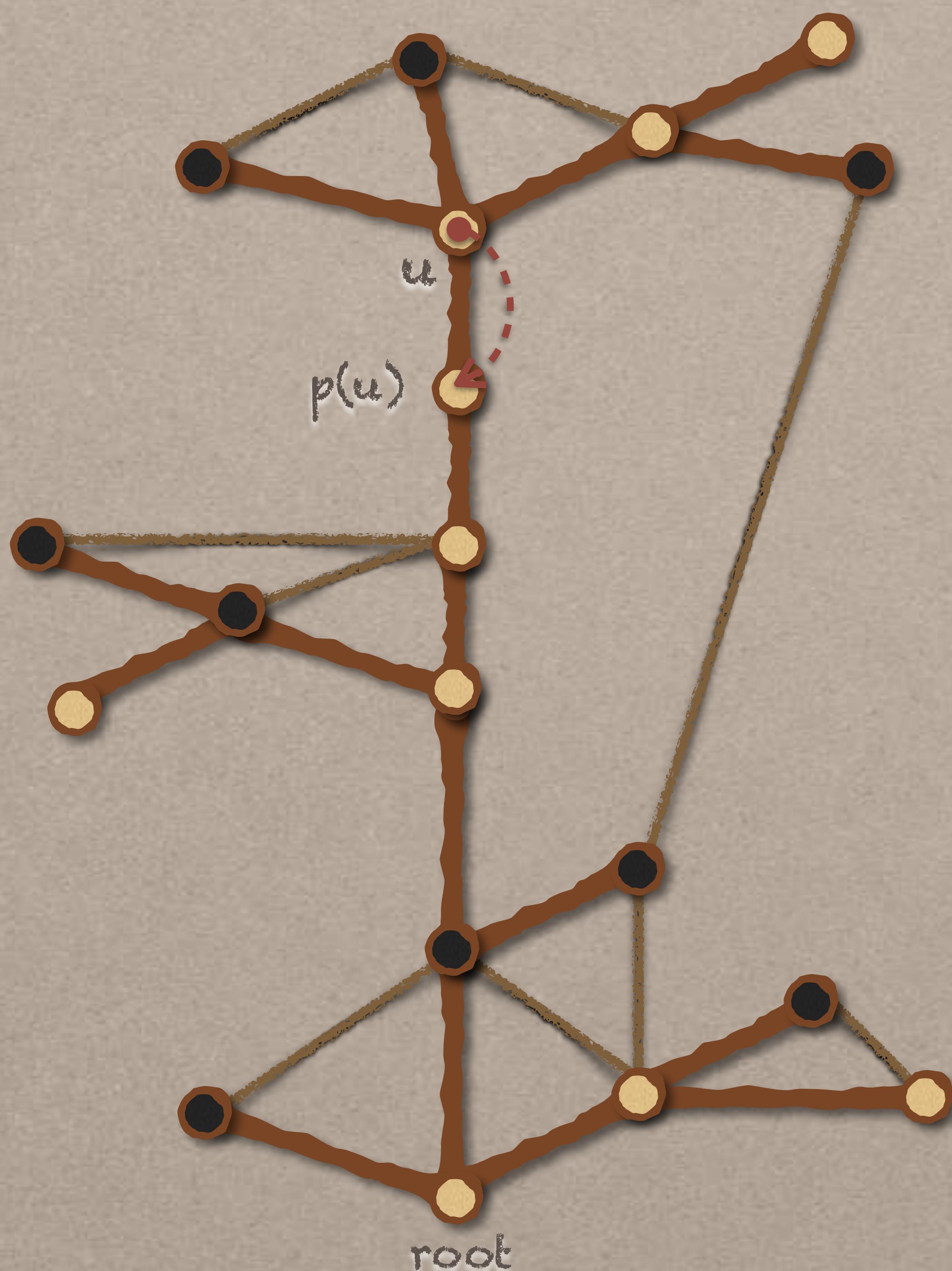
There is a bijection between (simple, rooted) outerplanar maps with n vertices and bicoloured plane trees with n vertices, with a white rightmost branch.

? "Why are outerplanar maps like trees?"

THE CORE ALGORITHM



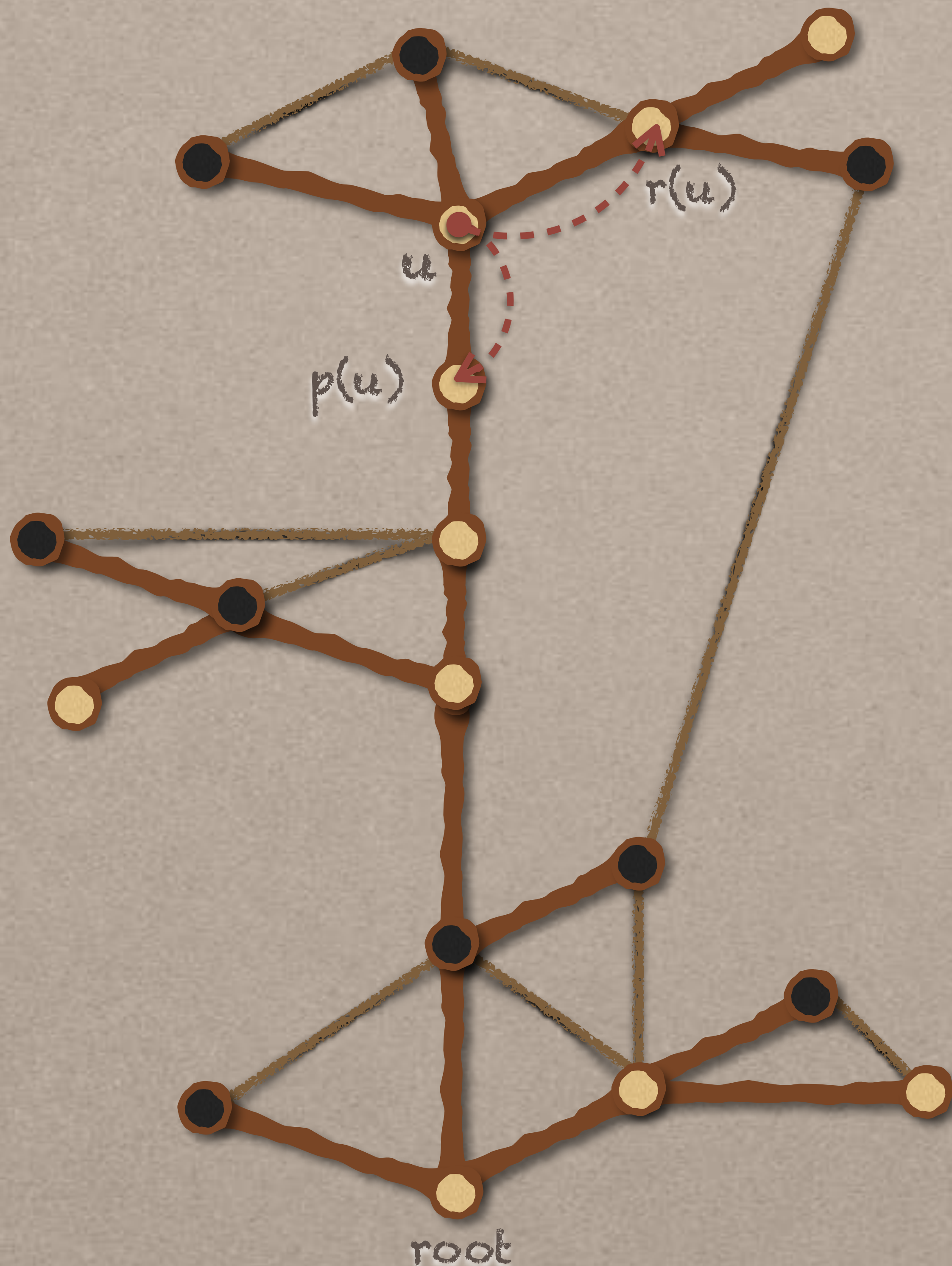
THE CORE ALGORITHM



Let u be a vertex distinct from the root; then a (map-)geodesic ending at the root moves from u to

- p(u),

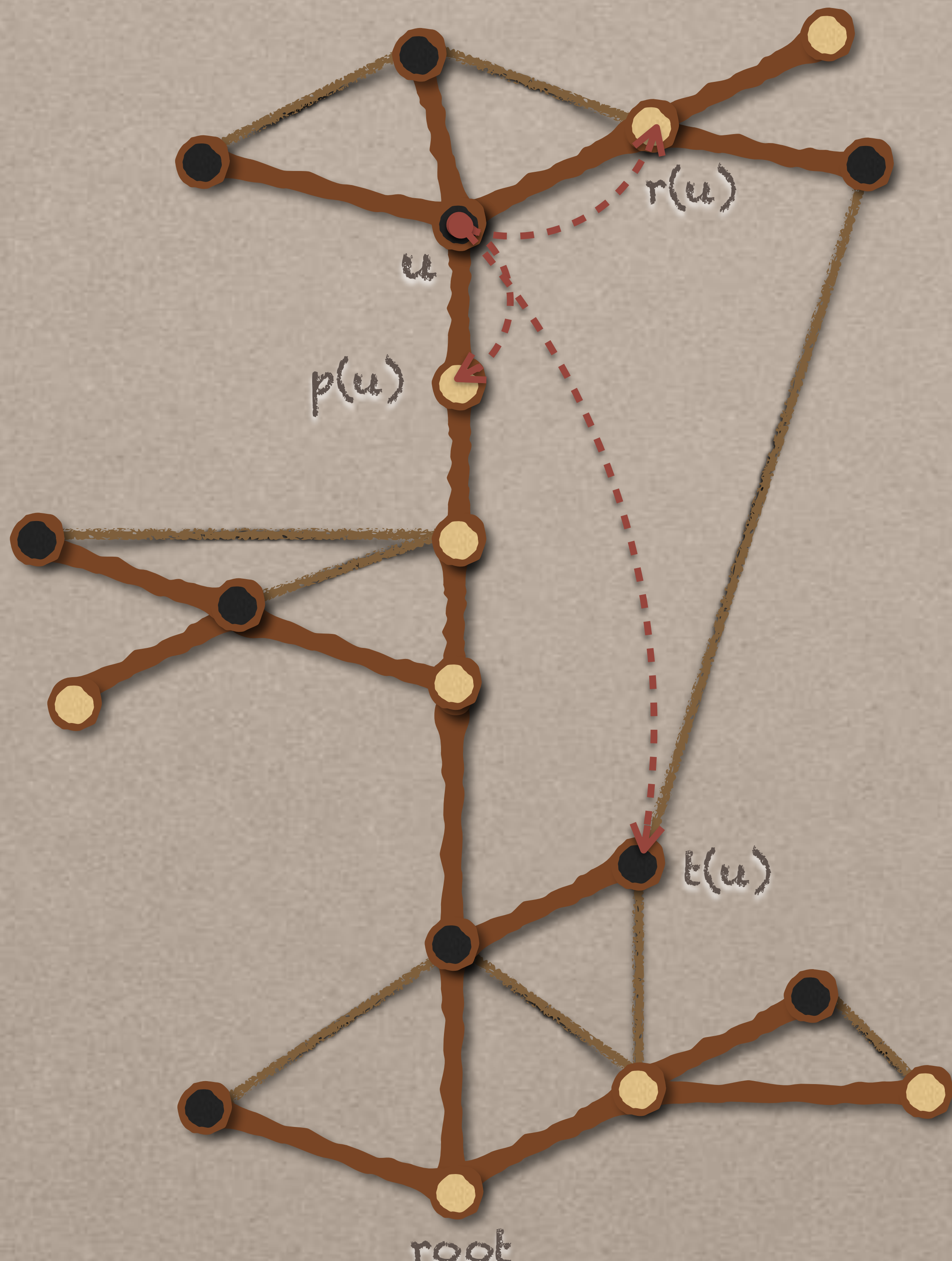
THE CORE ALGORITHM



Let u be a vertex distinct from the root; then a (map-)geodesic ending at the root moves from u to

- p(u),
- or r(u),

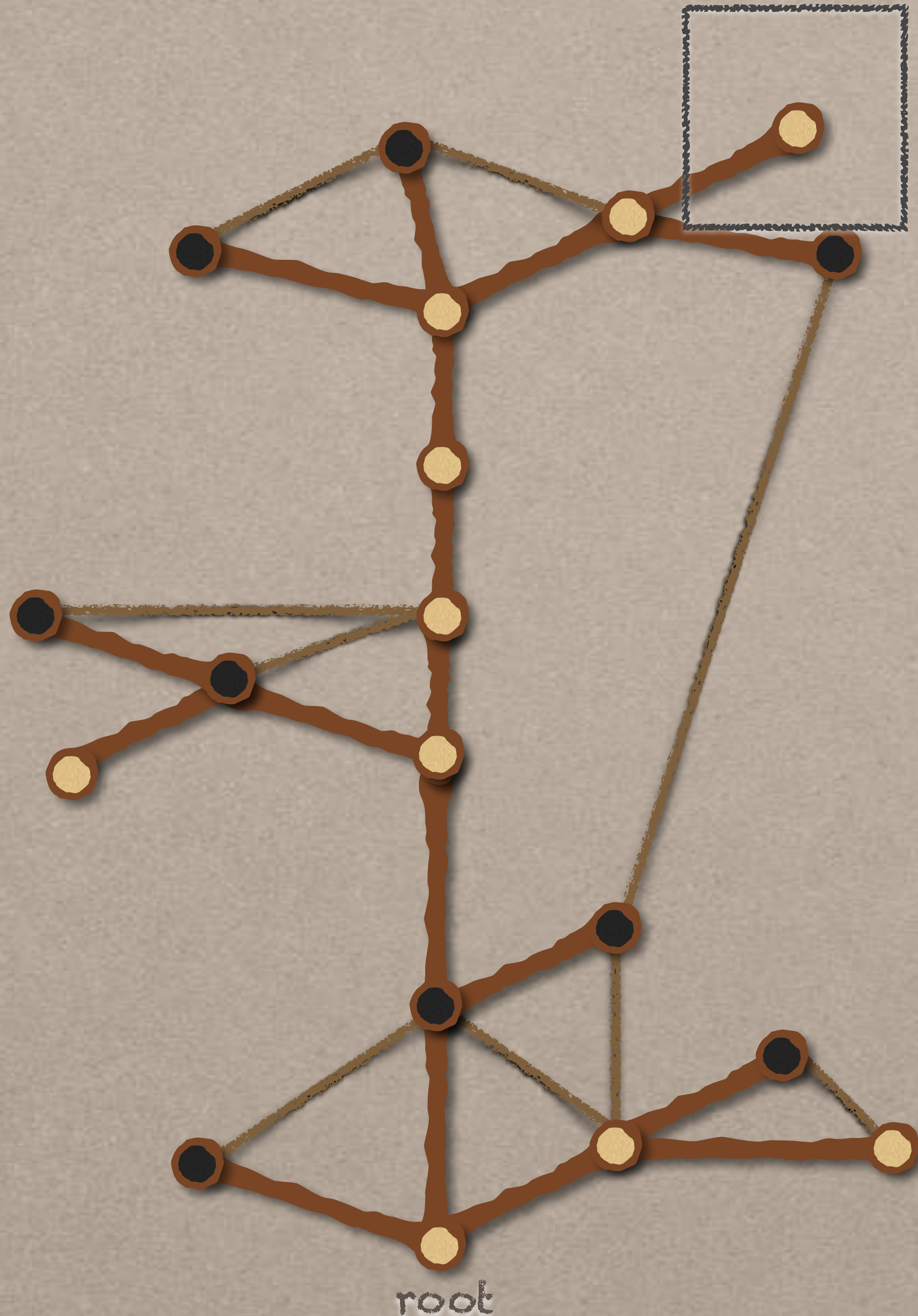
THE CORE ALGORITHM



Let u be a vertex distinct from the root; then a (map-)geodesic ending at the root moves from u to

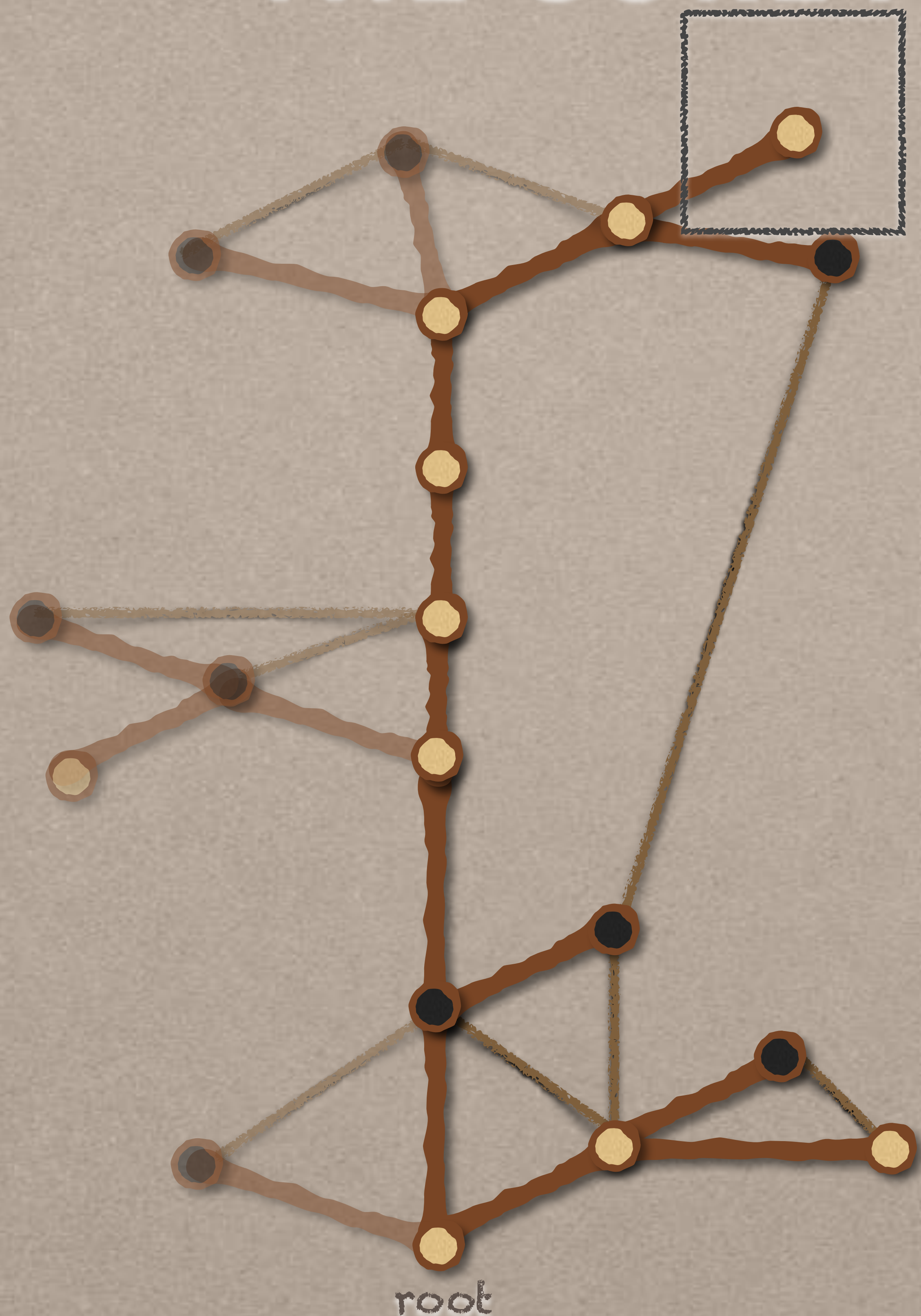
- p(u),
- or r(u),
- or t(u).

THE CORE ALGORITHM



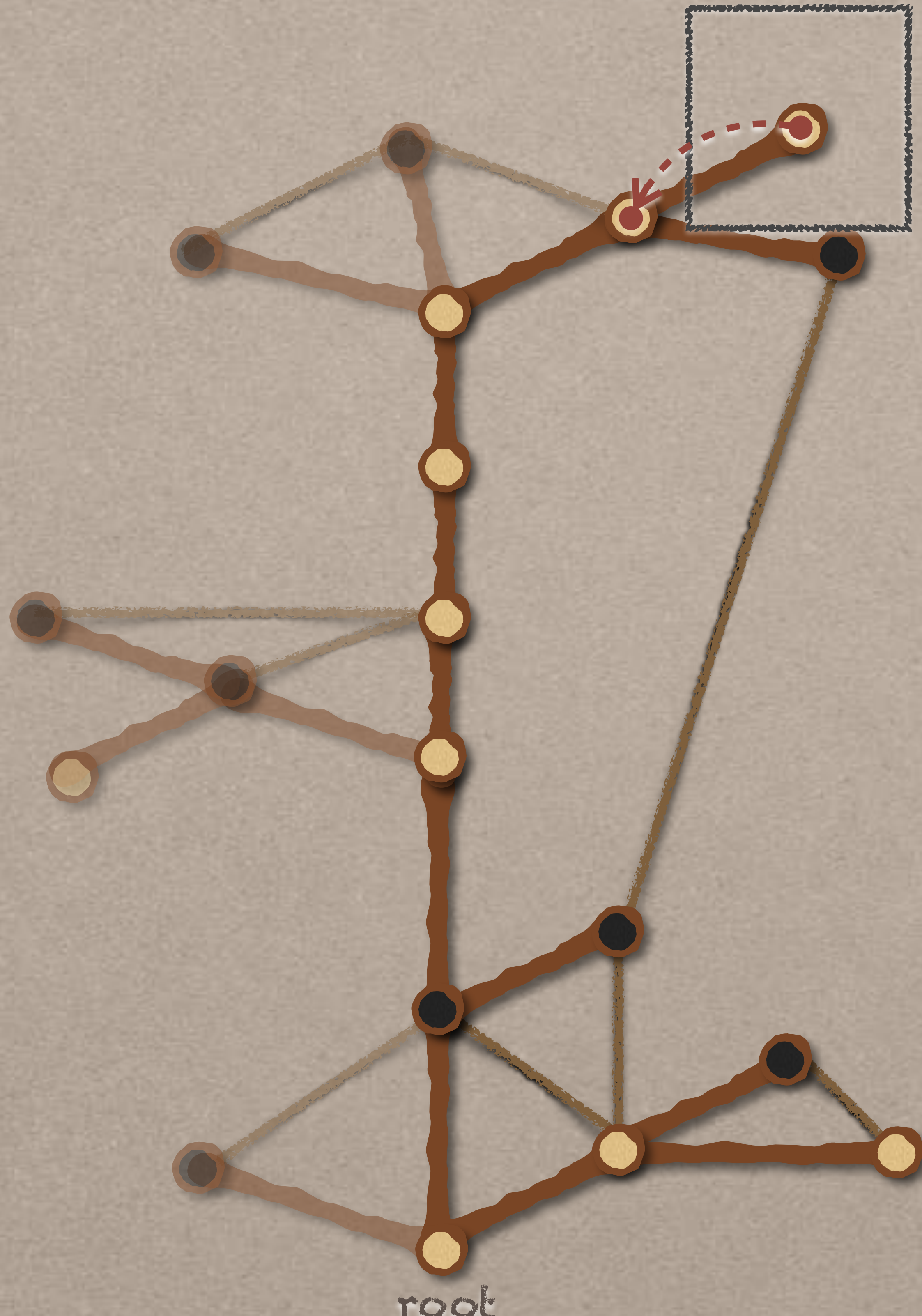
white leaf

THE CORE ALGORITHM



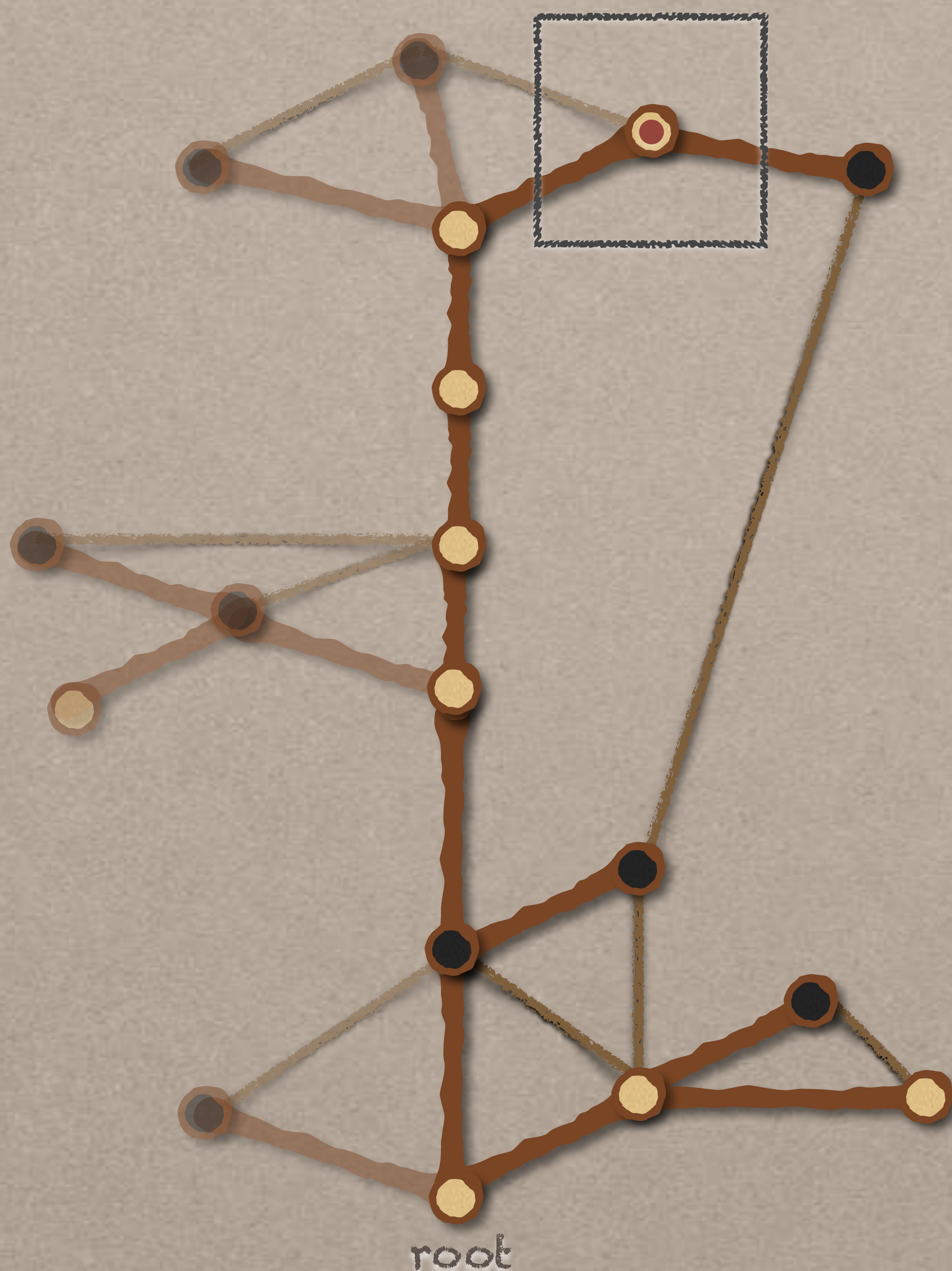
white leaf

THE CORE ALGORITHM



white leaf

THE CORE ALGORITHM

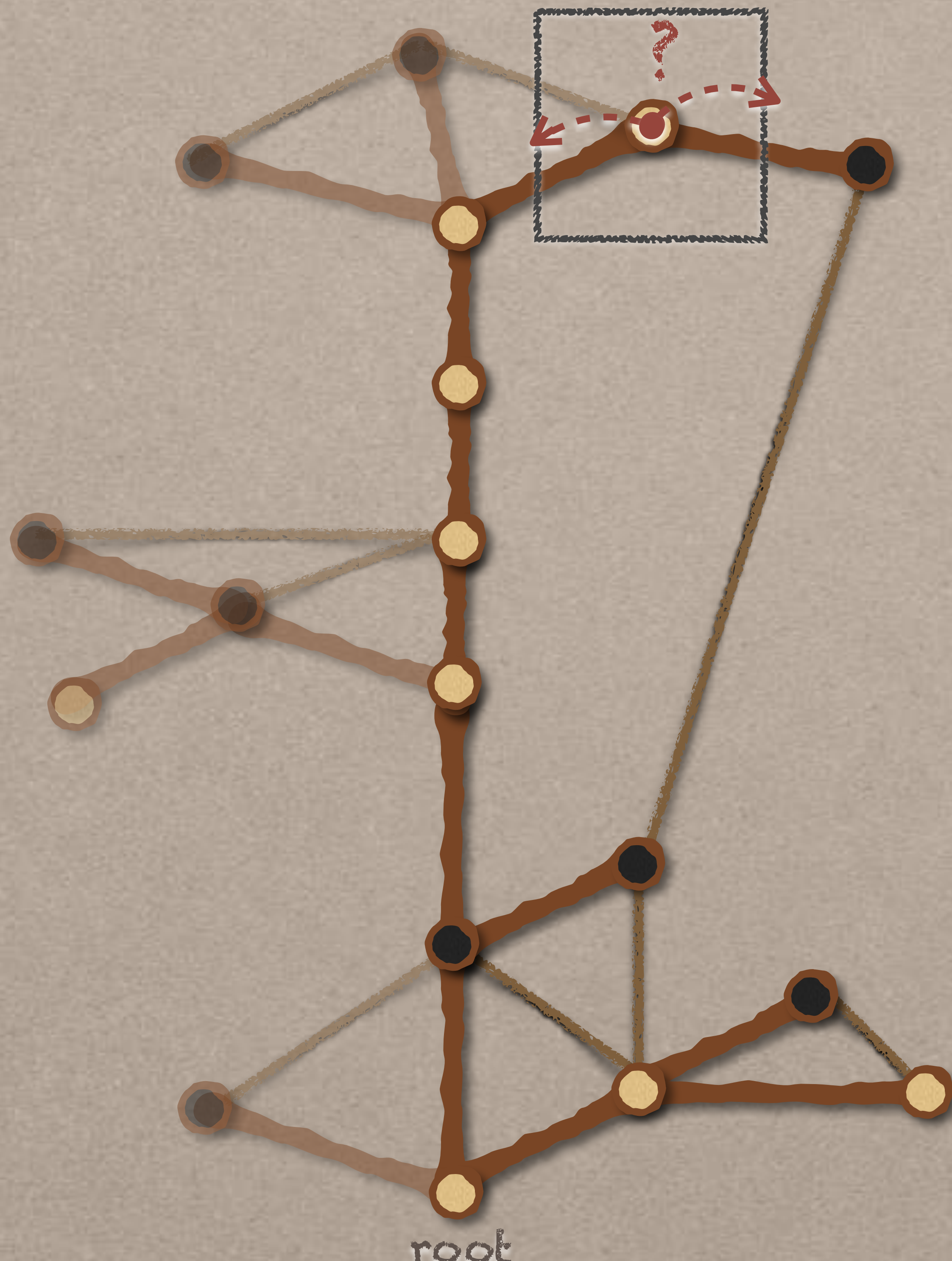


white leaf

`go to p(u), erase u`

white non-leaf

THE CORE ALGORITHM

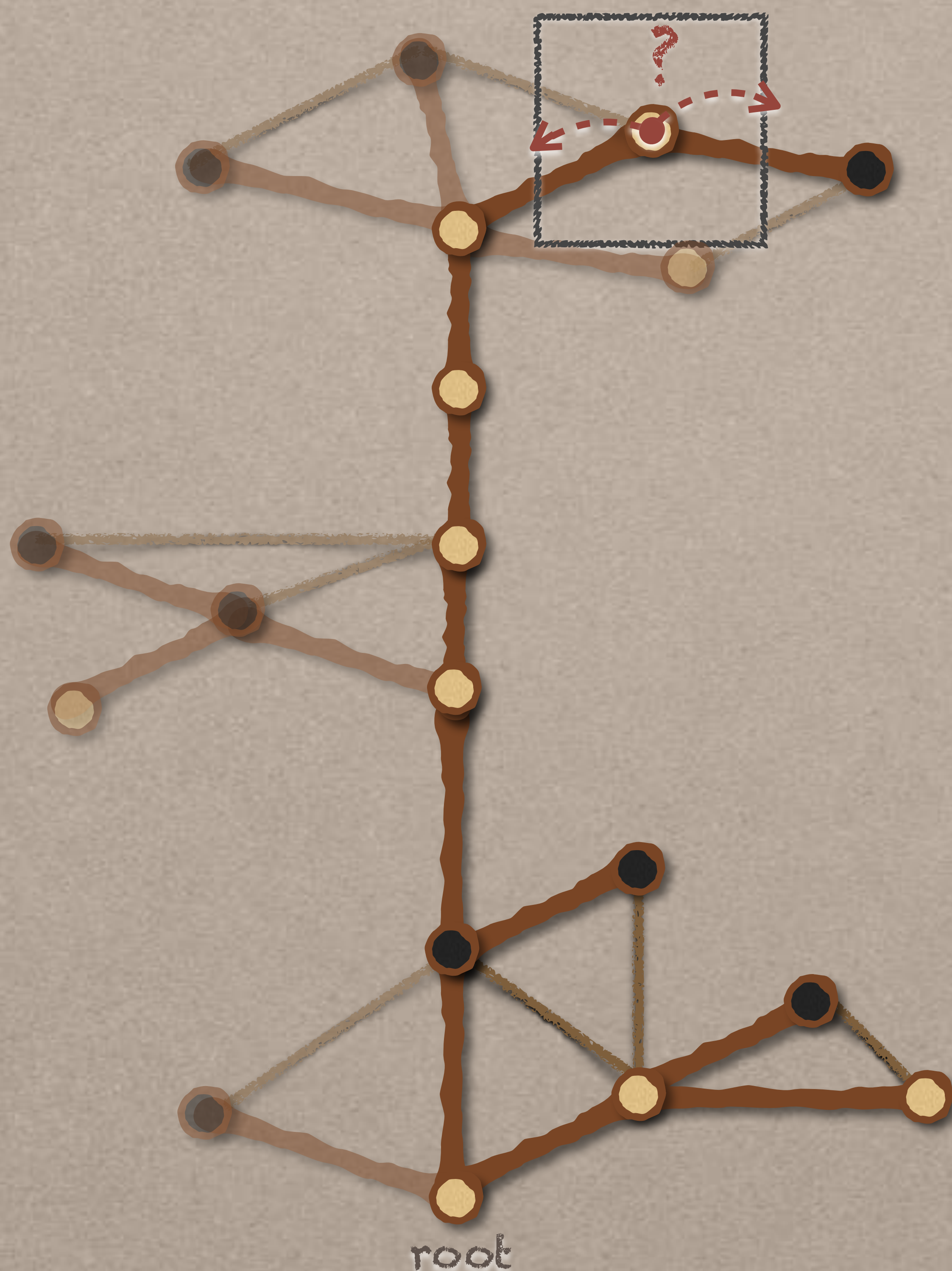


white leaf

go to p(u), erase u

white non-leaf

THE CORE ALGORITHM



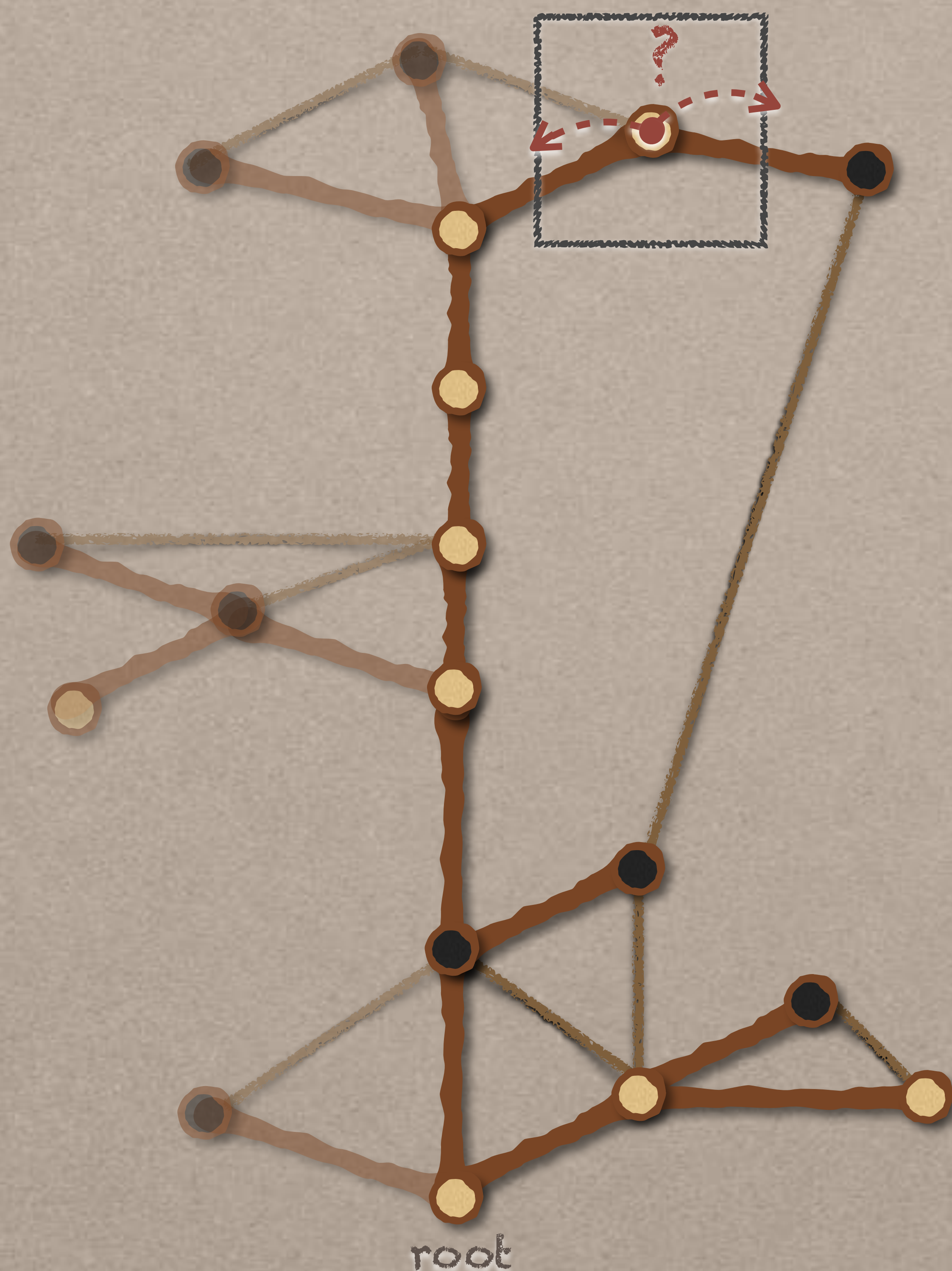
white leaf

go to p(u), erase u

white non-leaf

if u has a right sibling, move to p(u), erase u (and descendants)

THE CORE ALGORITHM



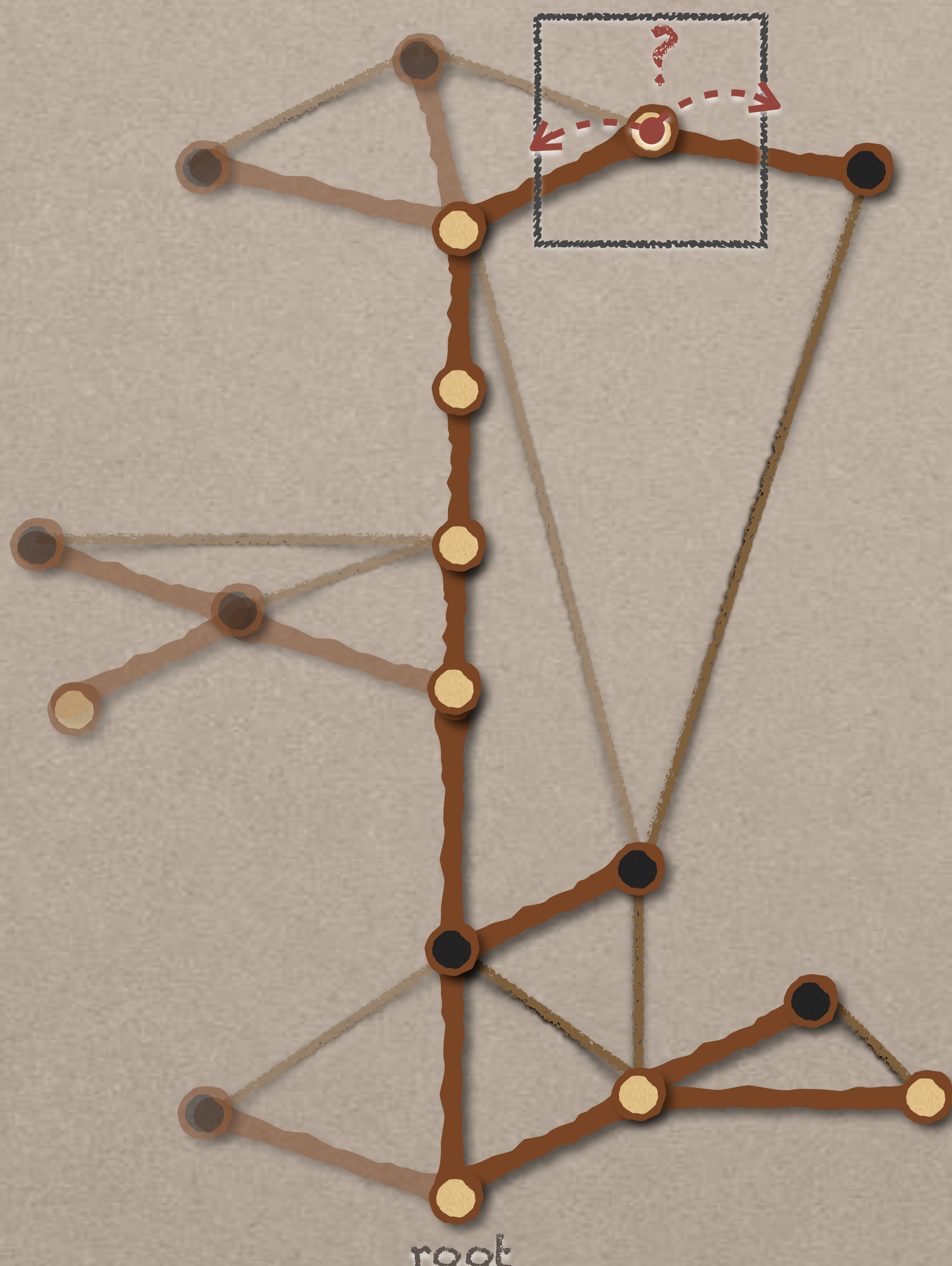
white leaf

go to p(u), erase u

white non-leaf

if u has a right sibling, move to p(u), erase u (and descendants)

THE CORE ALGORITHM



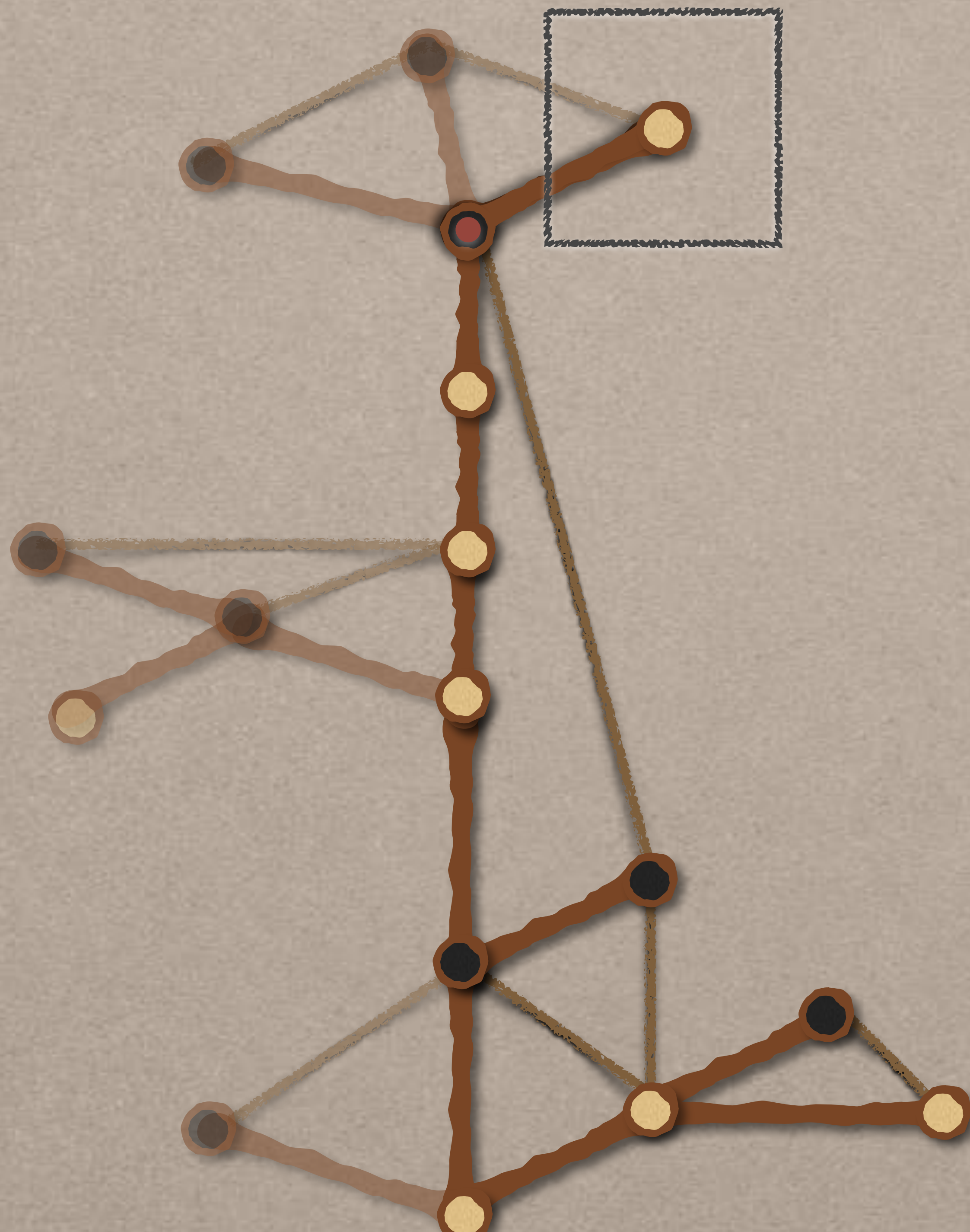
white leaf

go to p(u), erase u

white non-leaf

if u has a right sibling, move to p(u), erase u (and descendants)

THE CORE ALGORITHM



white leaf

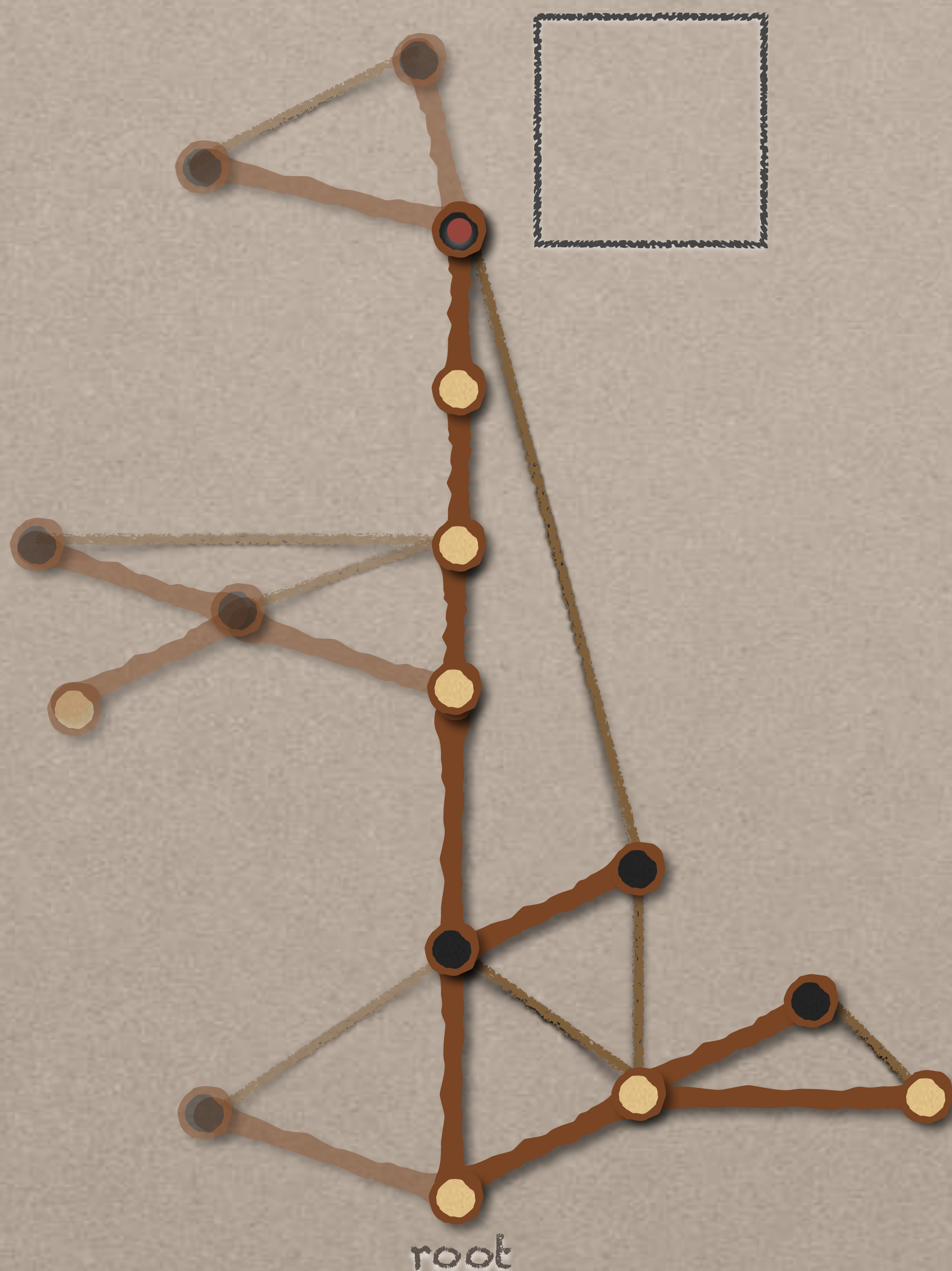
go to p(u), erase u

white non-leaf

if u has a right sibling, move to p(u), erase u (and descendants)

if u has no right siblings, merge p(u) with r(u); go to p(u)

THE CORE ALGORITHM



white leaf

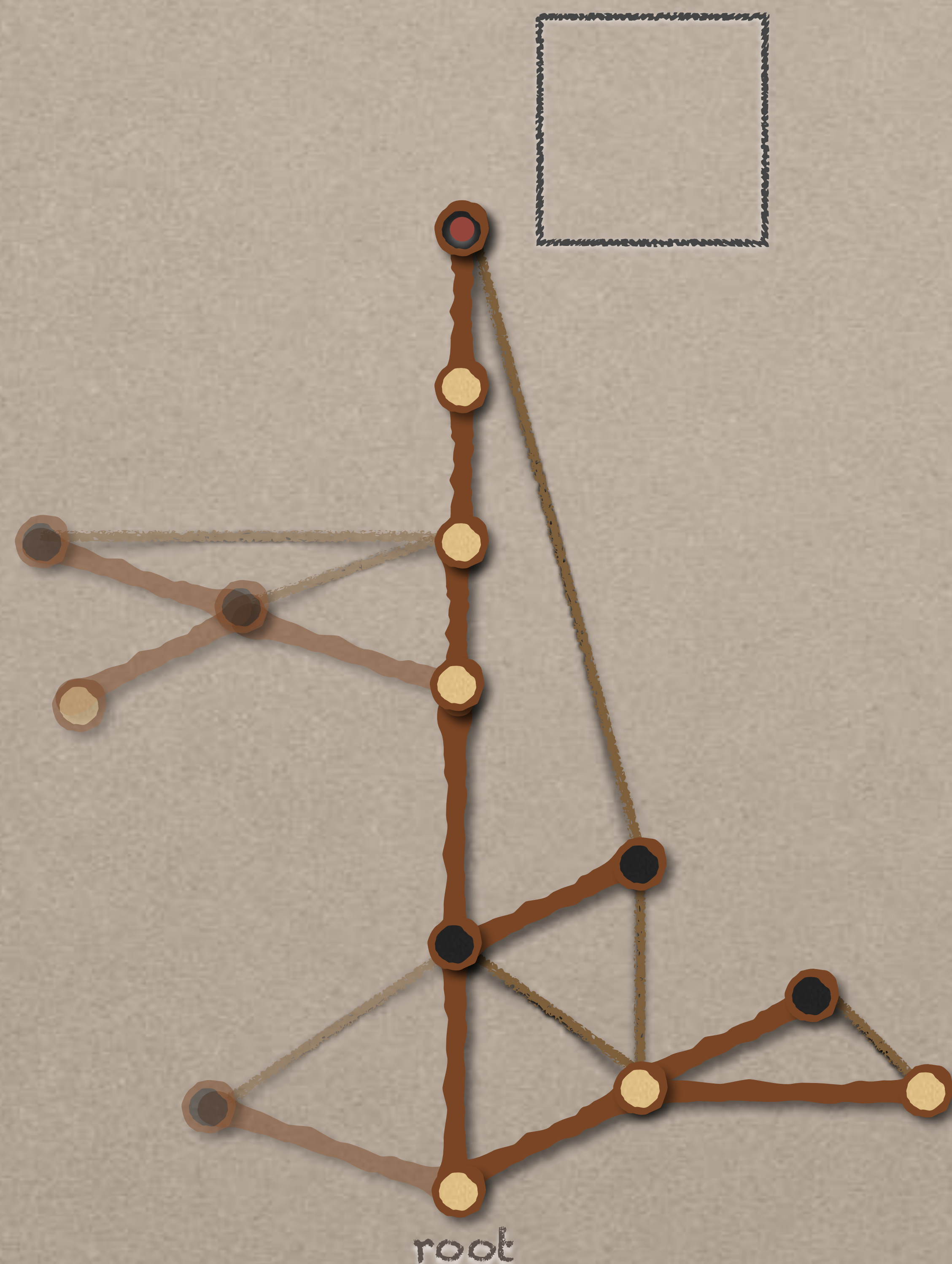
go to p(u), erase u

white non-leaf

if u has a right sibling, move to p(u), erase u (and descendants)

if u has no right siblings, merge p(u) with r(u); go to p(u)

THE CORE ALGORITHM



white leaf

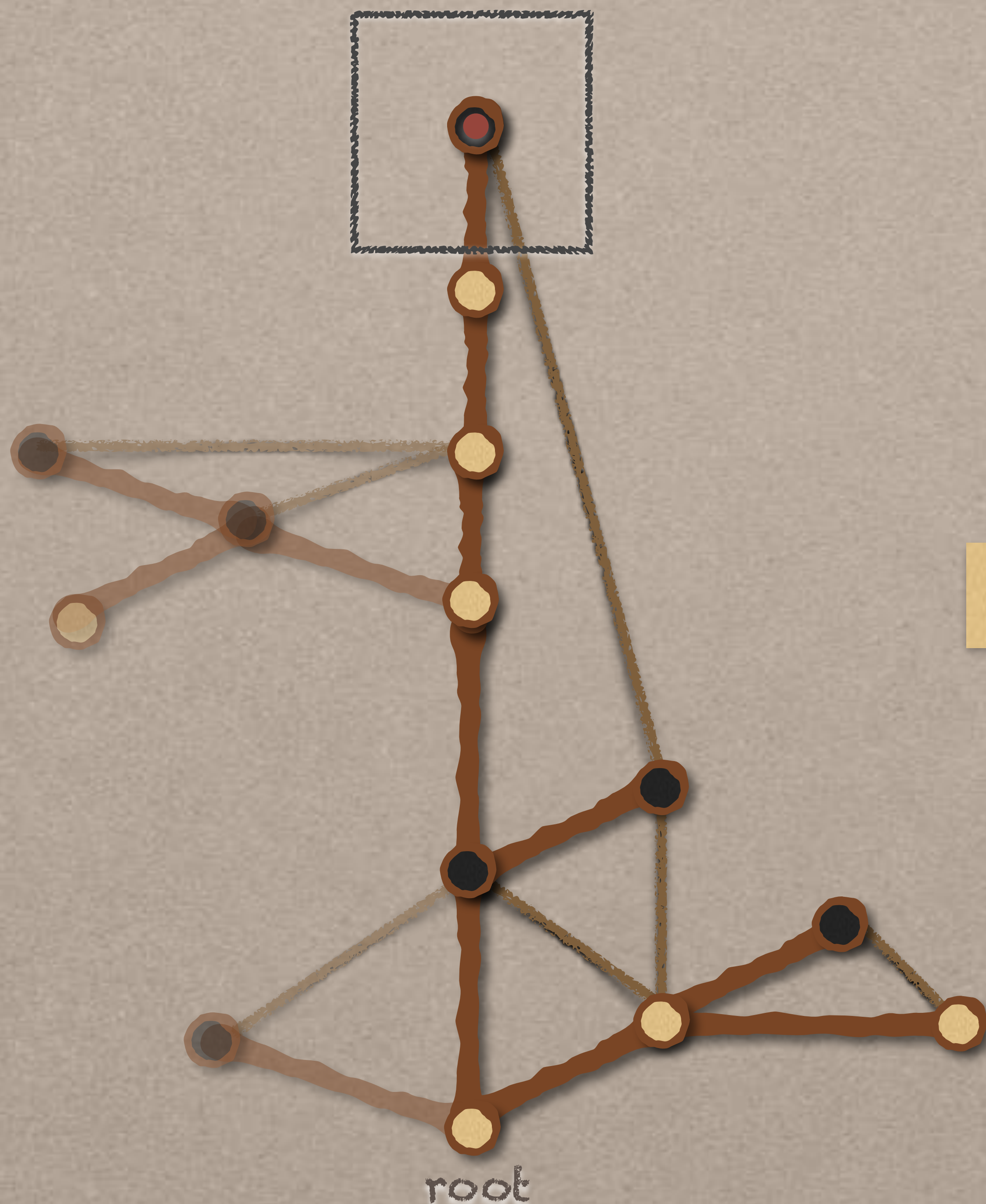
go to p(u), erase u

white non-leaf

if u has a right sibling, move to p(u), erase u (and descendants)

if u has no right siblings, merge p(u) with r(u); go to p(u)

THE CORE ALGORITHM



white leaf

go to p(u), erase u

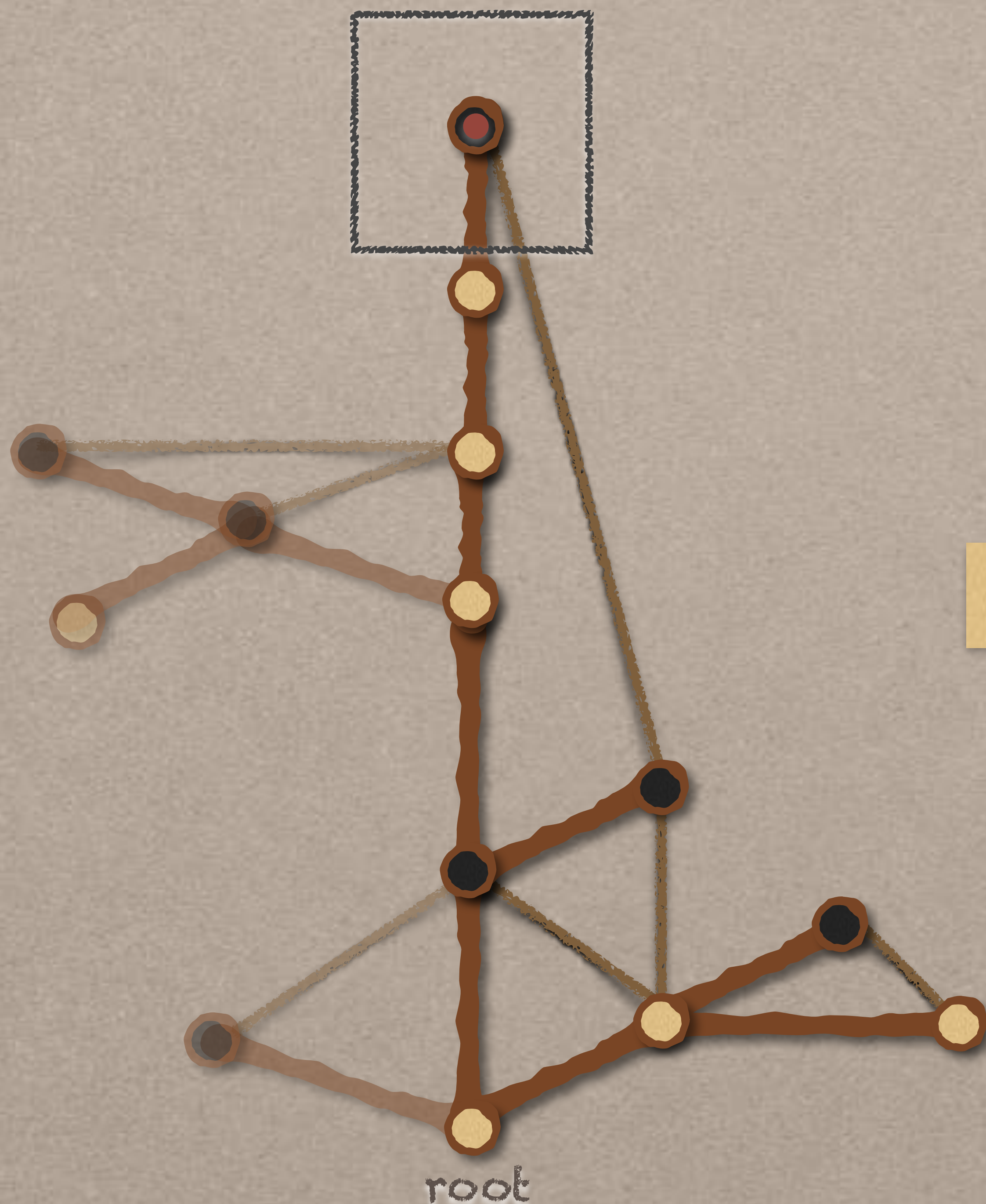
white non-leaf

if u has a right sibling, move to p(u), erase u (and descendants)

if u has no right siblings, merge p(u) with r(u); go to p(u)

black

THE CORE ALGORITHM



white leaf

go to p(u), erase u

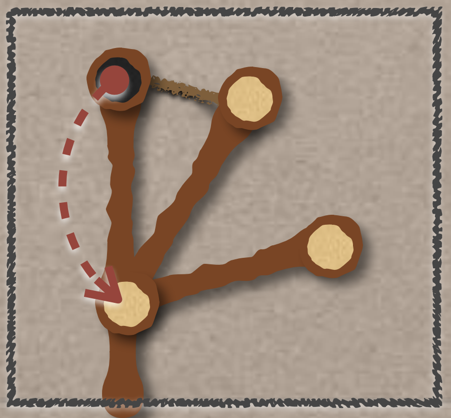
white non-leaf

if u has a right sibling, move to p(u), erase u (and descendants)

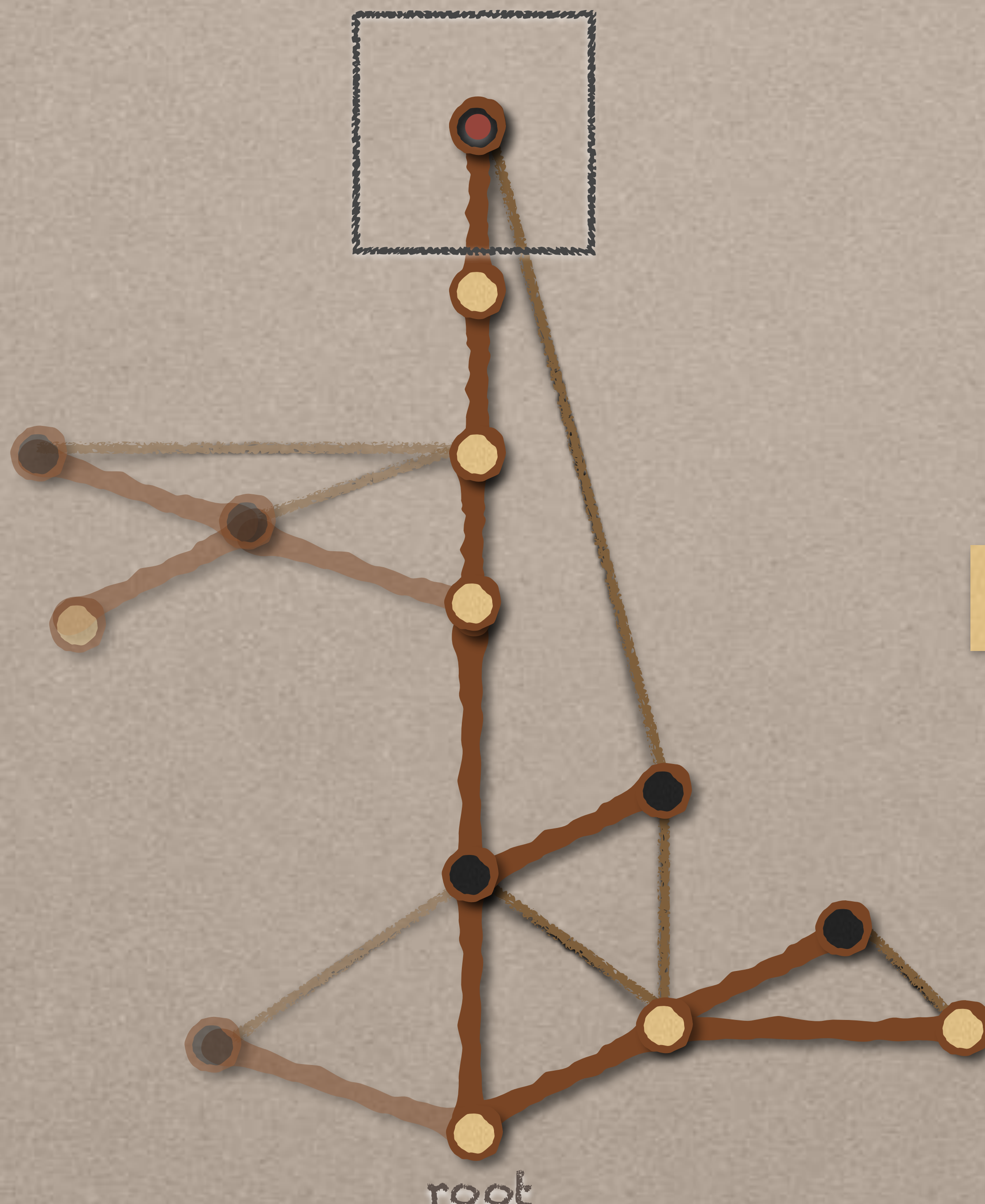
if u has no right siblings, merge p(u) with r(u); go to p(u)

black

if u has at least 2 right siblings, go to p(u)



THE CORE ALGORITHM



white leaf

go to p(u), erase u

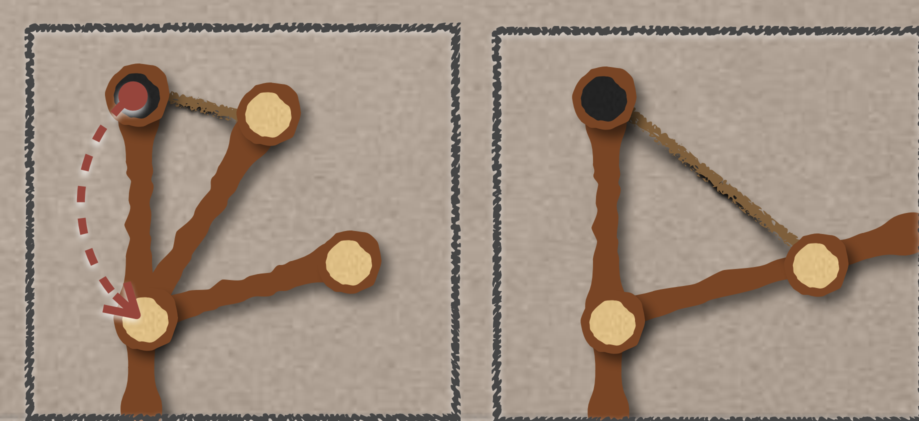
white non-leaf

if u has a right sibling, move to p(u), erase u (and descendants)

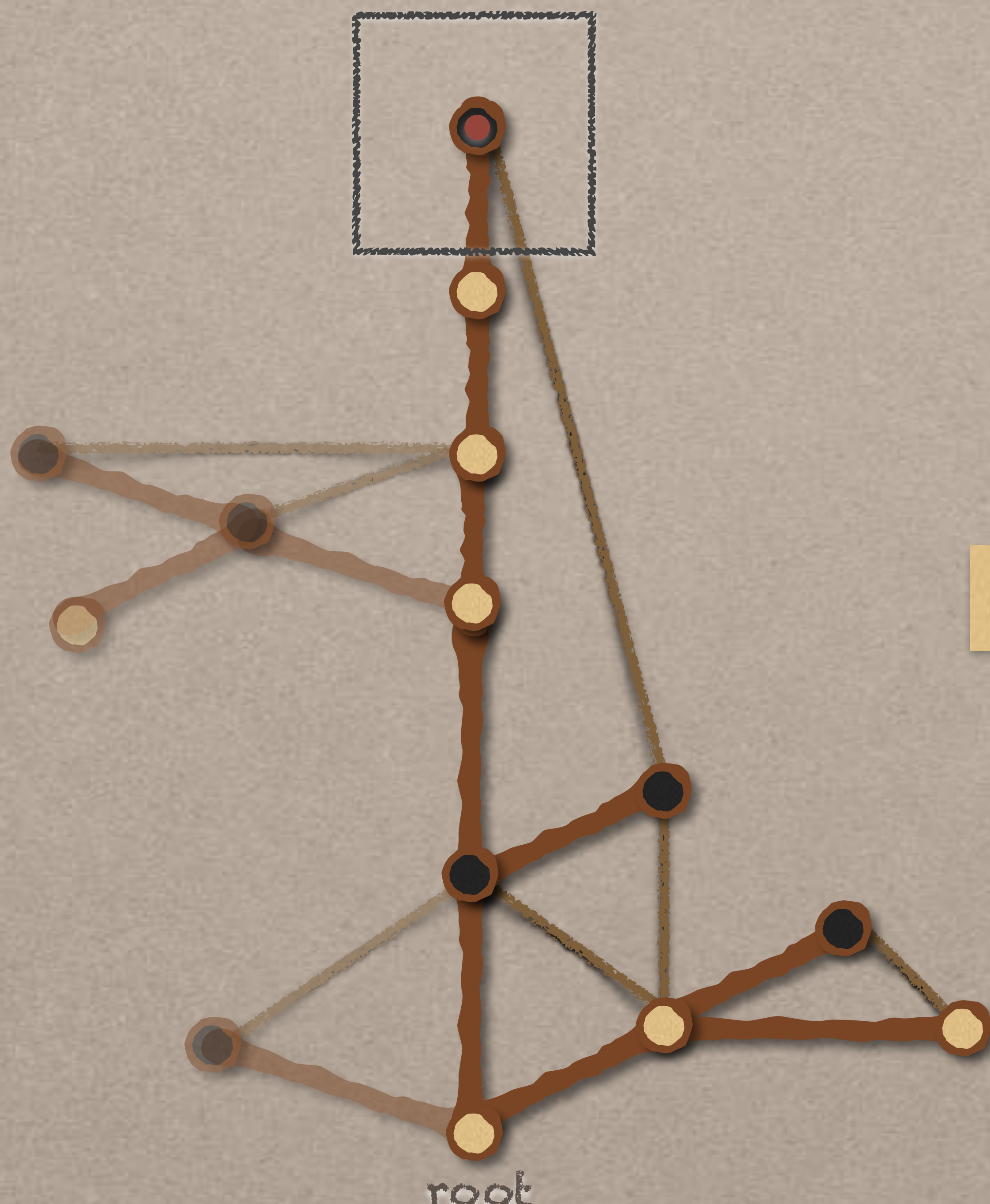
if u has no right siblings, merge p(u) with r(u); go to p(u)

black

if u has at least 2 right siblings, go to p(u)



THE CORE ALGORITHM



white leaf

go to p(u), erase u

white non-leaf

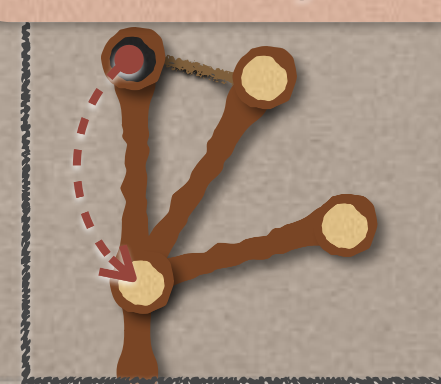
if u has a right sibling, move to p(u), erase u (and descendants)

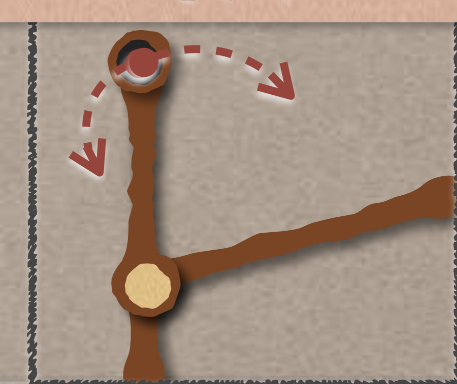
if u has no right siblings, merge p(u) with r(u); go to p(u)

black

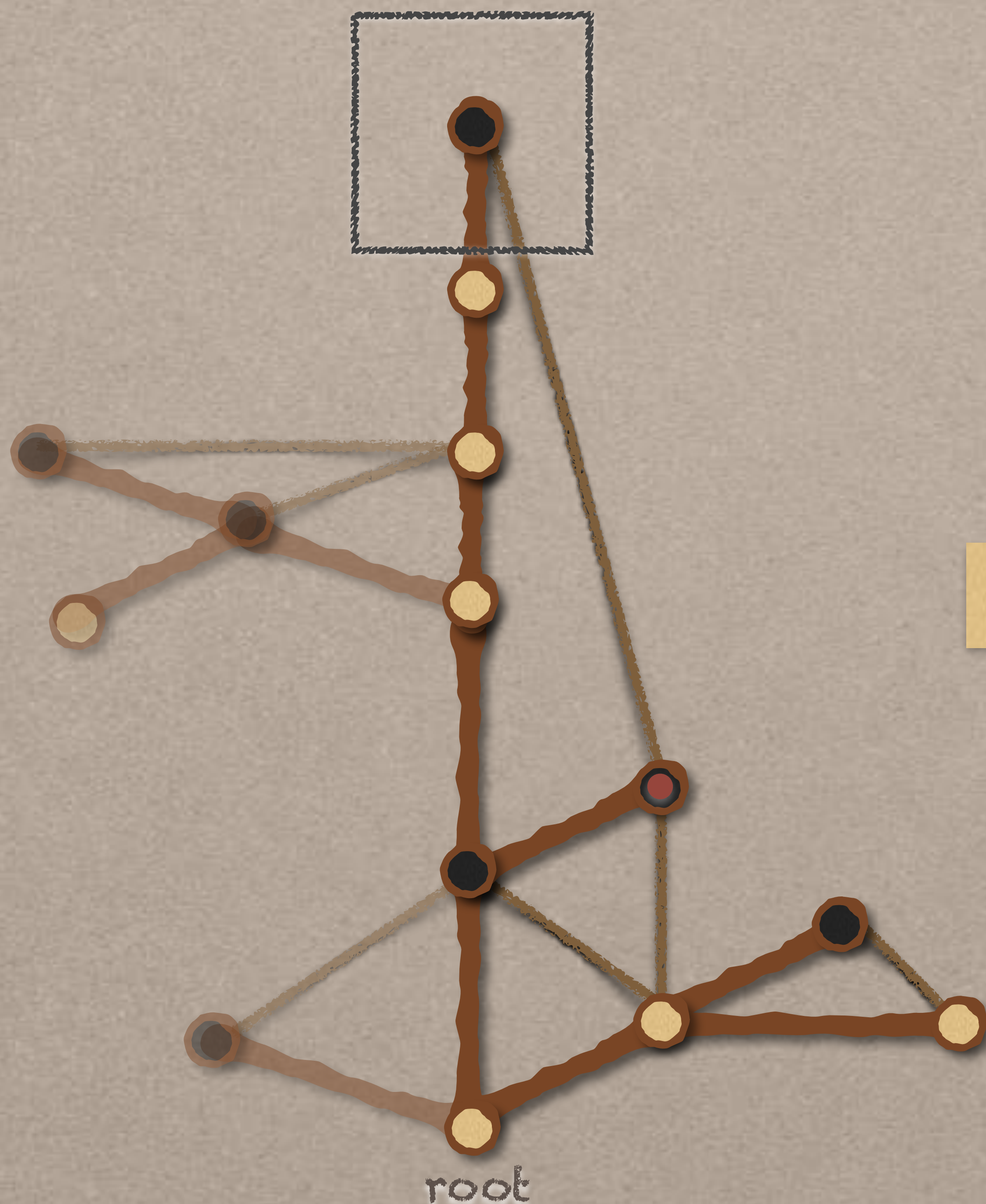
if u has at least 2 right siblings, go to p(u)

if u has 1 right sibling, merge it with p(u)





THE CORE ALGORITHM



white leaf

go to p(u), erase u

white non-leaf

if u has a right sibling, move to p(u), erase u (and descendants)

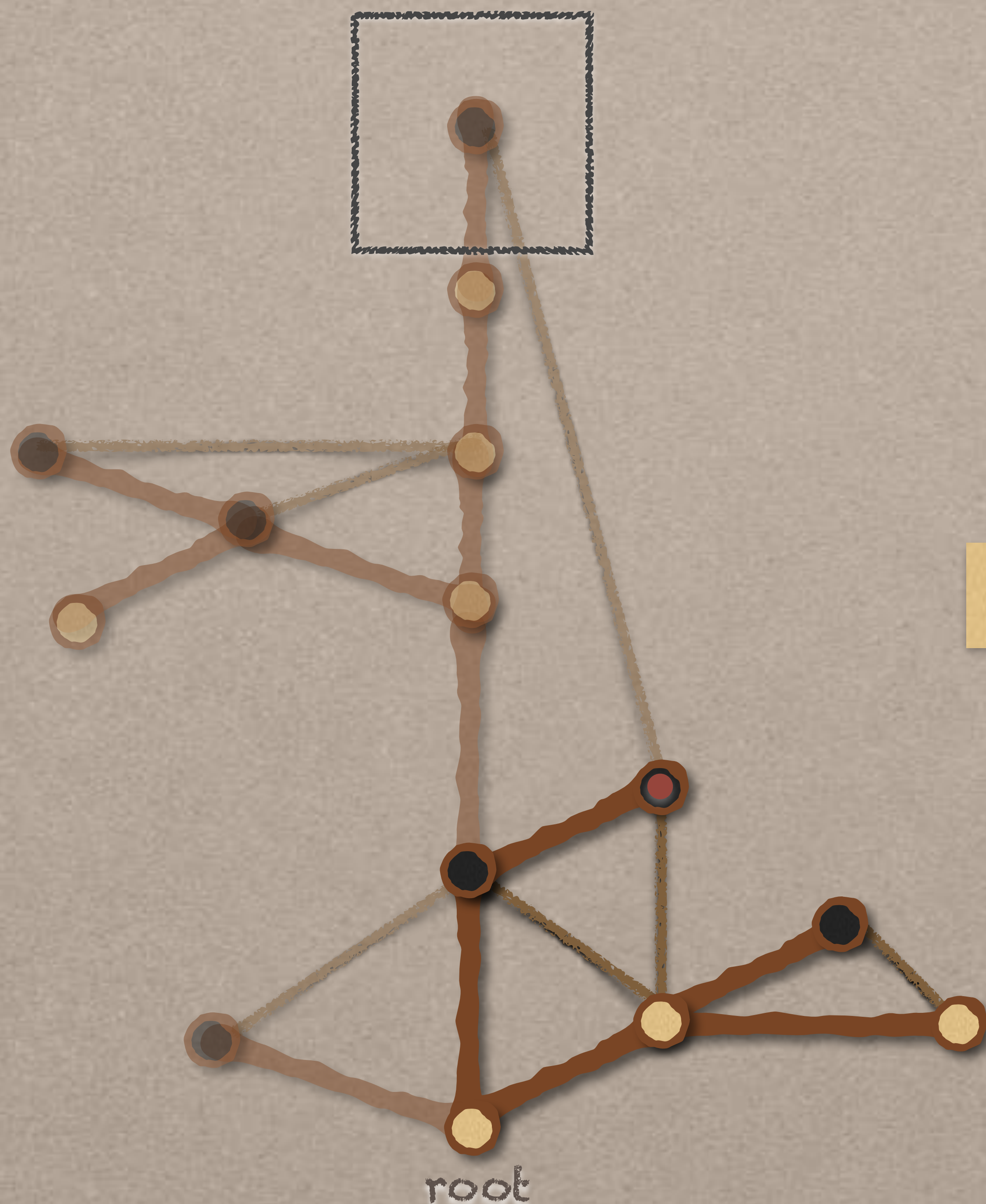
if u has no right siblings, merge p(u) with r(u); go to p(u)

black

if u has at least 2 right siblings, go to p(u)

if u has 1 right sibling, merge it with p(u)

THE CORE ALGORITHM



white leaf

go to p(u), erase u

white non-leaf

if u has a right sibling, move to p(u), erase u (and descendants)

if u has no right siblings, merge p(u) with r(u); go to p(u)

black

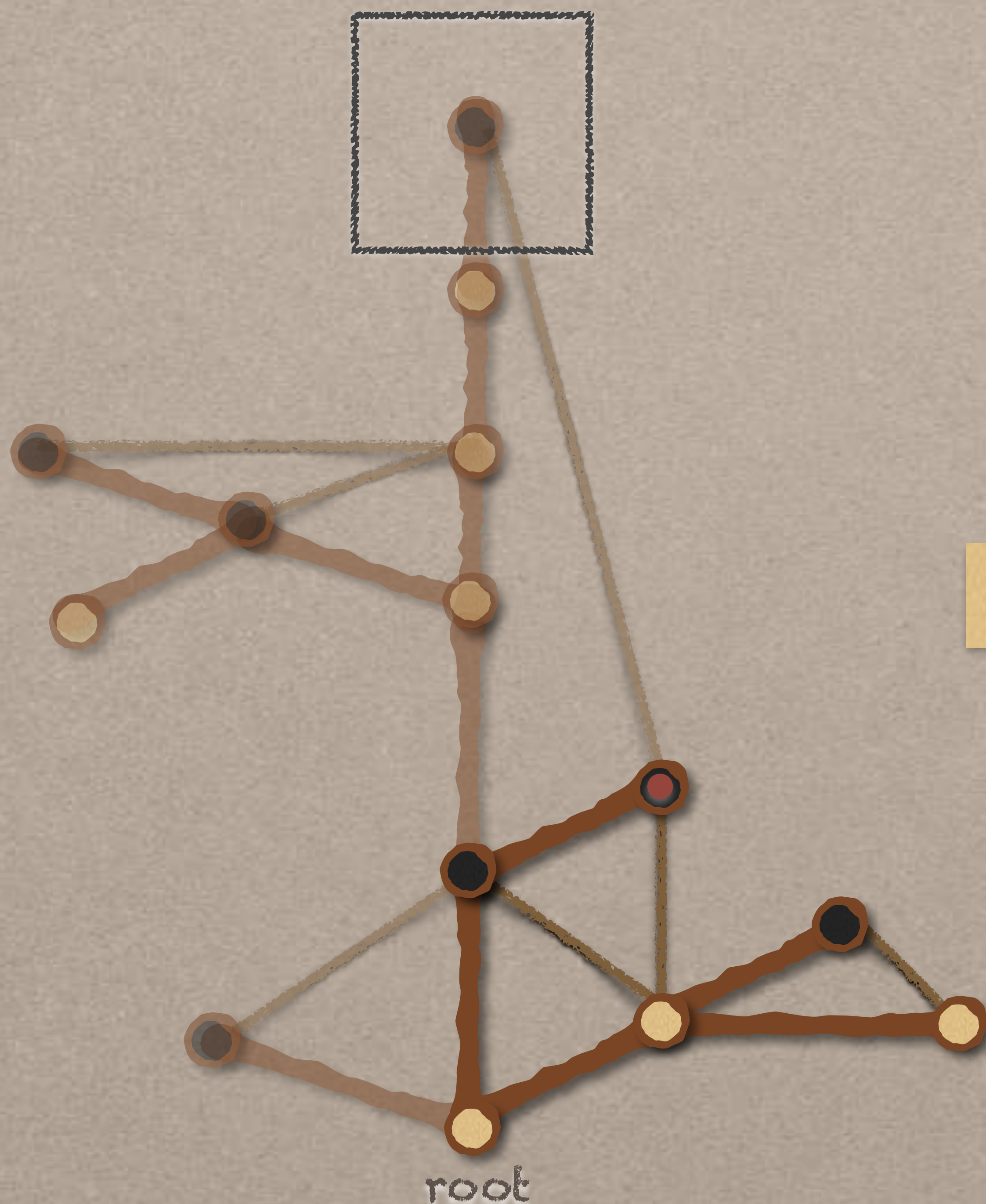
if u has at least 2 right siblings, go to p(u)

if u has 1 right sibling, merge it with p(u)

if u has no right siblings, “jump”!

THE CORE ALGORITHM

? How many steps do we take?



white leaf

go to p(u), erase u

white non-leaf

if u has a right sibling, move to p(u), erase u (and descendants)

if u has no right siblings, merge p(u) with r(u); go to p(u)

black

if u has at least 2 right siblings, go to p(u)

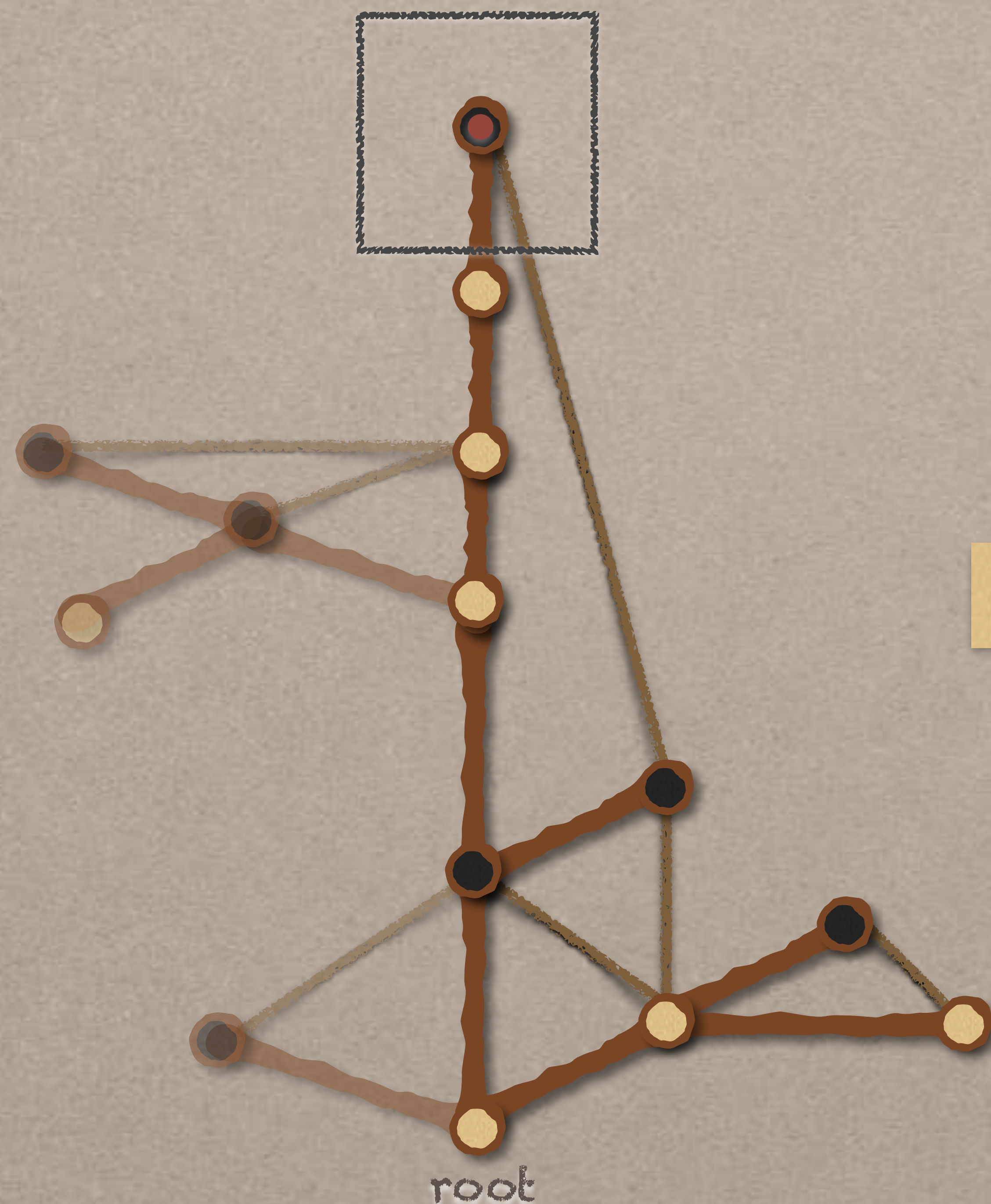
if u has 1 right sibling, merge it with p(u)

if u has no right siblings, "jump"!

THE CORE ALGORITHM

?

How many steps do we take?



white leaf

go to p(u), erase u

white non-leaf

if u has a right sibling, move to p(u), erase u (and descendants)

if u has no right siblings, merge p(u) with r(u); go to p(u)

black

if u has at least 2 right siblings, go to p(u)

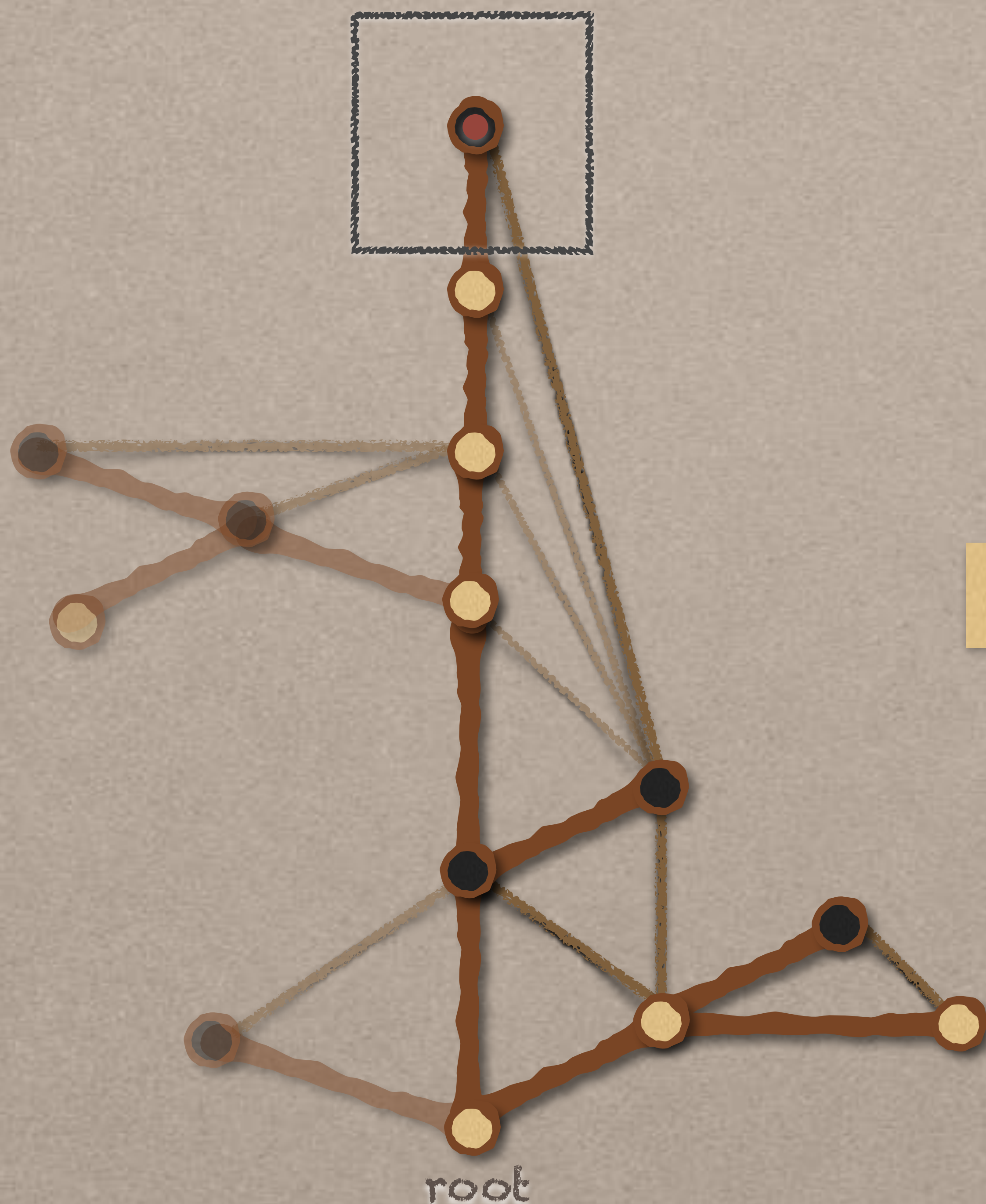
if u has 1 right sibling, merge it with p(u)

if u has no right siblings, "jump"!

THE CORE ALGORITHM

?

How many steps do we take?



white leaf

go to p(u), erase u

white non-leaf

if u has a right sibling, move to p(u), erase u (and descendants)

if u has no right siblings, merge p(u) with r(u); go to p(u)

black

if u has at least 2 right siblings, go to p(u)

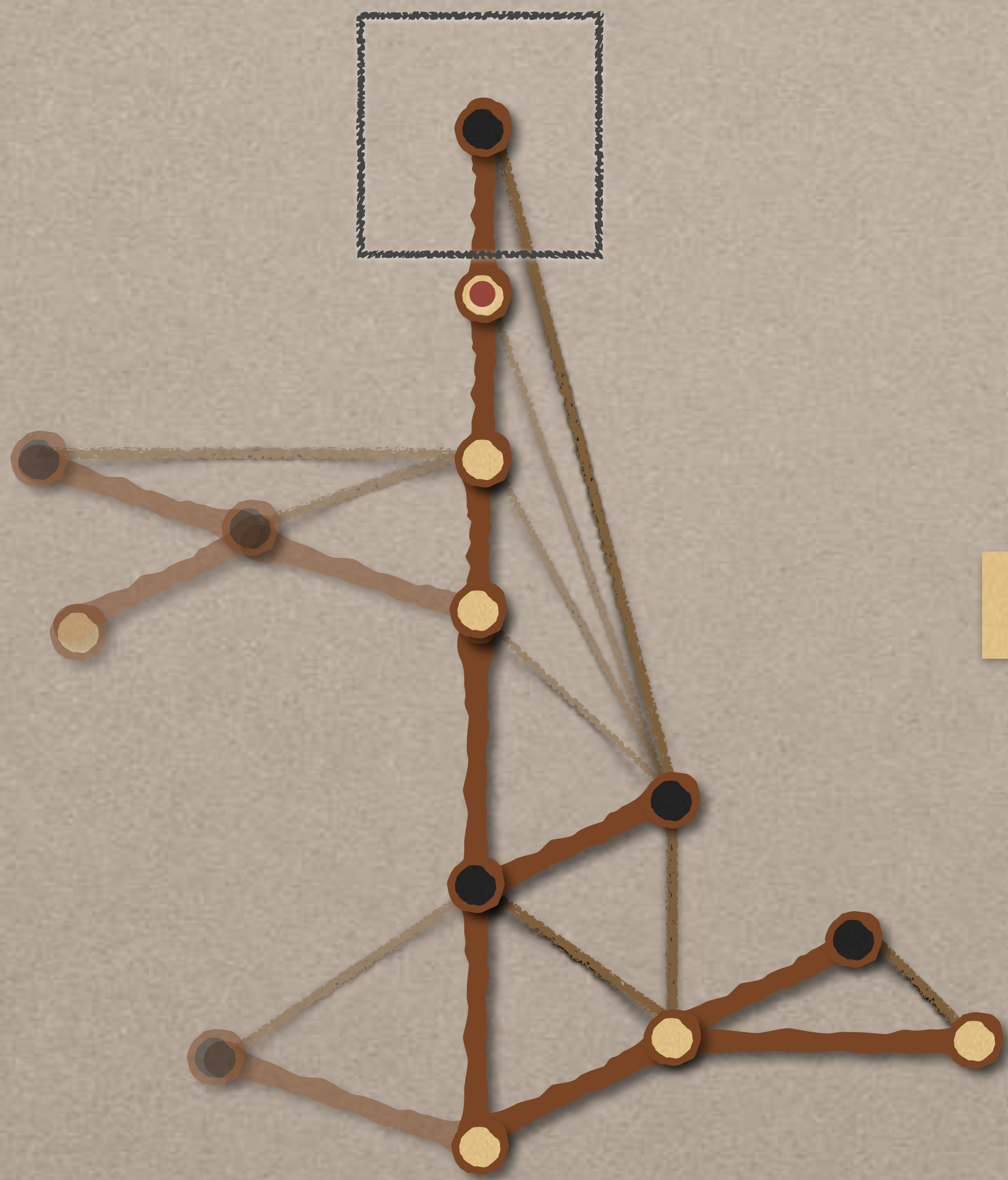
if u has 1 right sibling, merge it with p(u)

if u has no right siblings, “jump”!

THE CORE ALGORITHM

?

How many steps do we take?



white leaf

go to p(u), erase u

white non-leaf

if u has a right sibling, move to p(u), erase u (and descendants)

if u has no right siblings, merge p(u) with r(u); go to p(u)

black

if u has at least 2 right siblings, go to p(u)

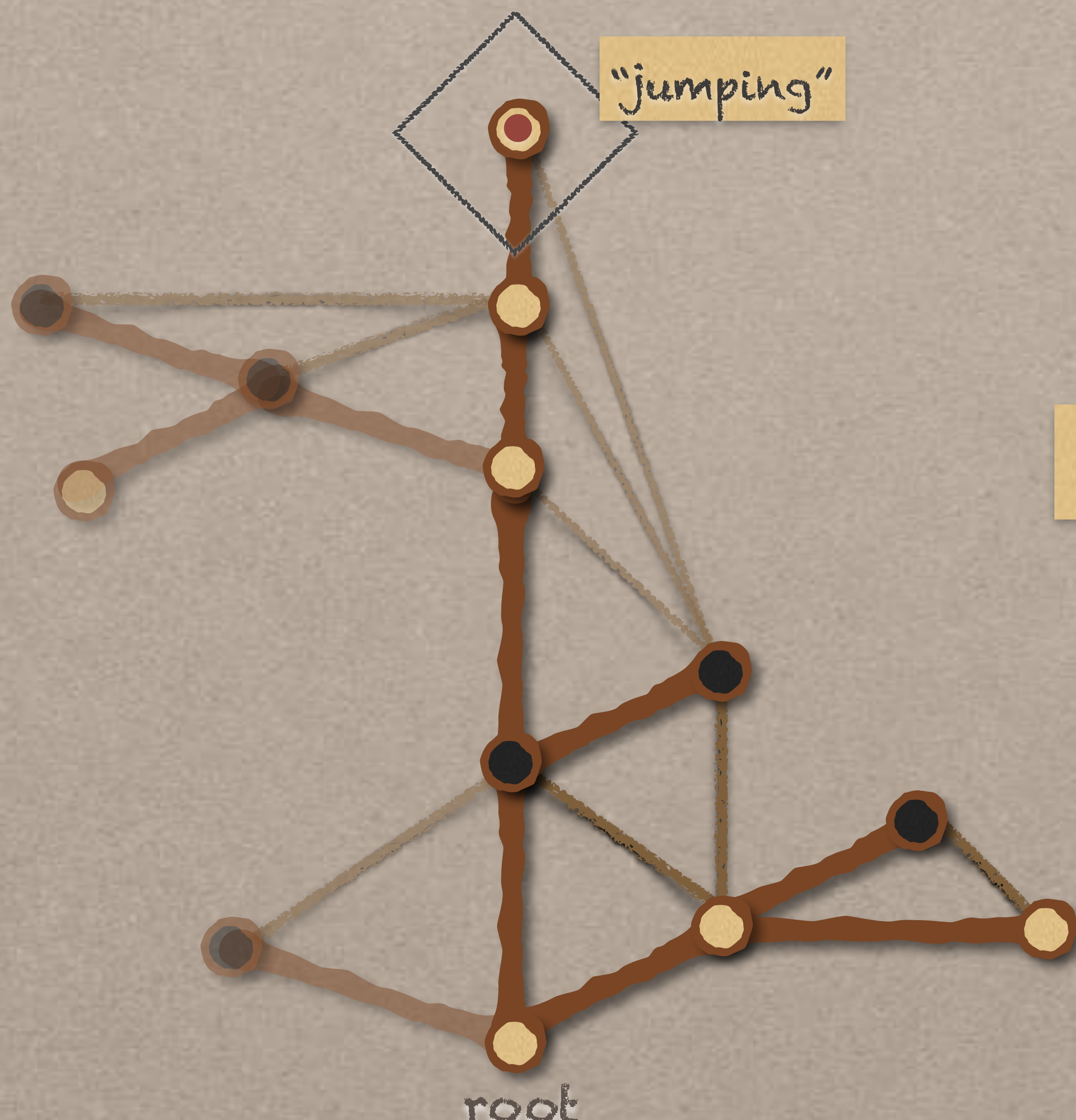
if u has 1 right sibling, merge it with p(u)

if u has no right siblings, "jump"!

THE CORE ALGORITHM

?

How many steps do we take?



white leaf

go to p(u), erase u

white non-leaf

if u has a right sibling, move to p(u), erase u (and descendants)

if u has no right siblings, merge p(u) with r(u); go to p(u)

black

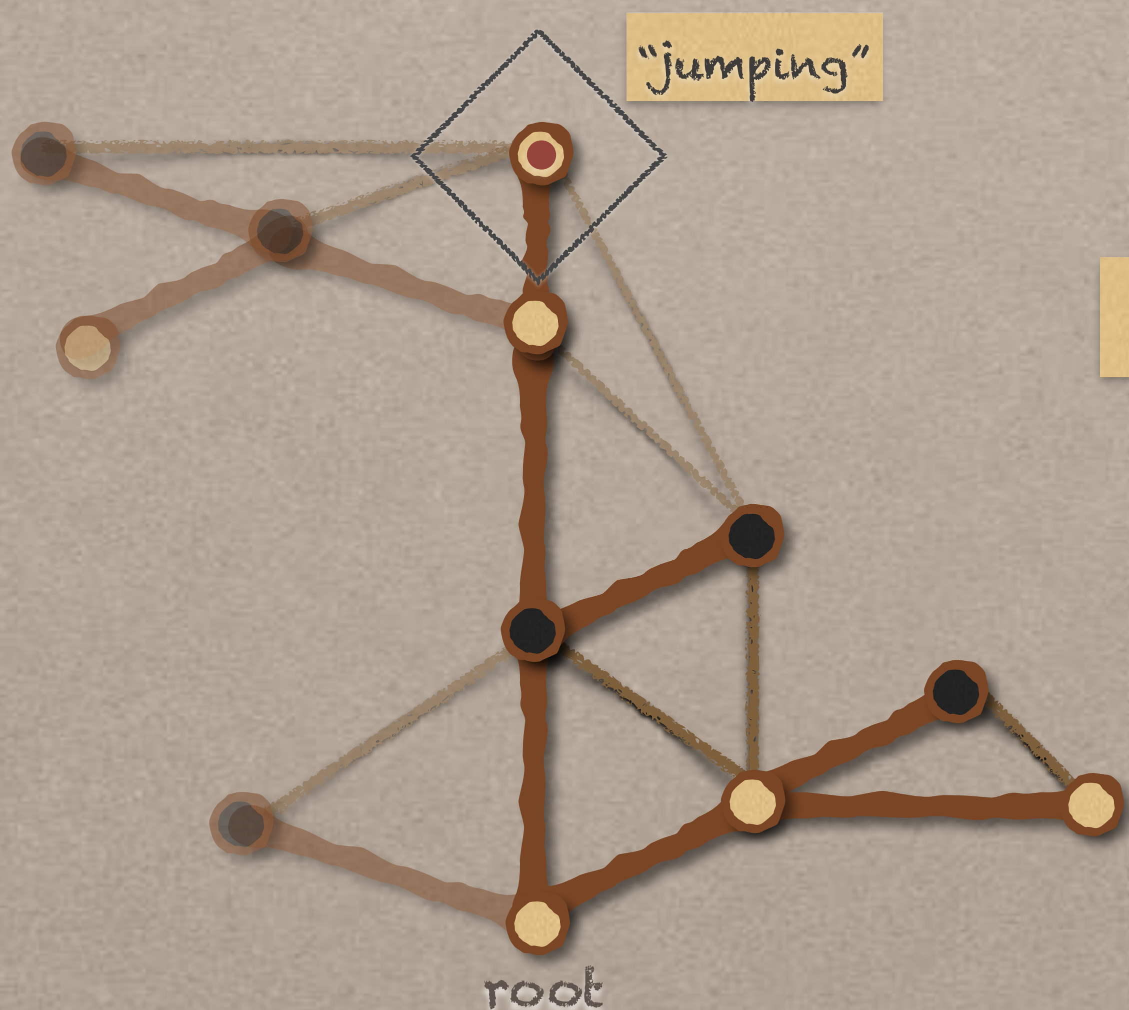
if u has at least 2 right siblings, go to p(u)

if u has 1 right sibling, merge it with p(u)

if u has no right siblings, "jump"!

THE CORE ALGORITHM

? How many steps do we take?



white leaf

go to p(u), erase u

white non-leaf

if u has a right sibling, move to p(u), erase u (and descendants)

if u has no right siblings, merge p(u) with r(u); go to p(u)

black

if u has at least 2 right siblings, go to p(u)

if u has 1 right sibling, merge it with p(u)

if u has no right siblings, "jump"!

THE CORE ALGORITHM

?

How many steps do we take?

white leaf

go to p(u), erase u

white non-leaf

if u has a right sibling, move to p(u), erase u (and descendants)

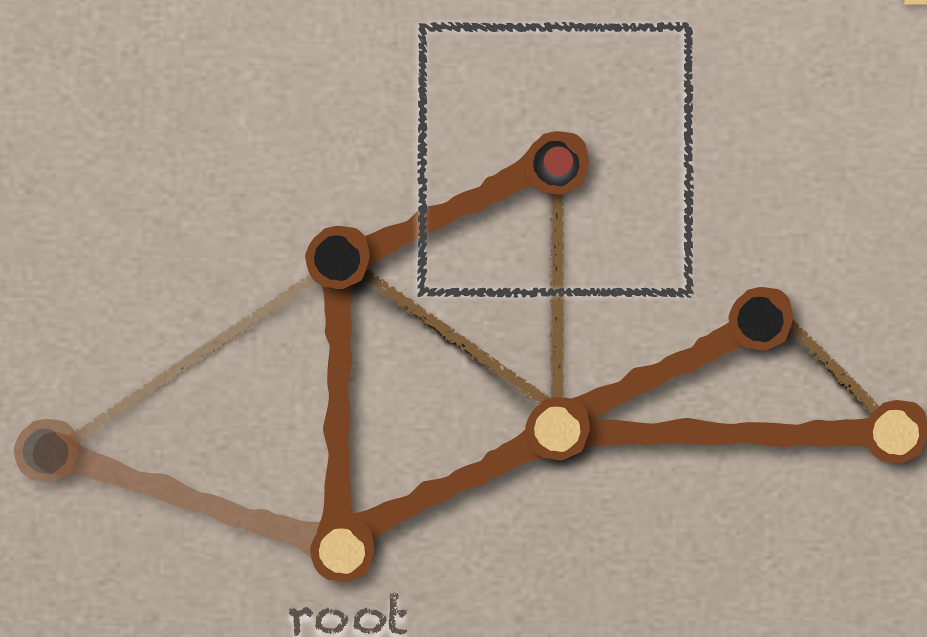
if u has no right siblings, merge p(u) with r(u); go to p(u)

black

if u has at least 2 right siblings, go to p(u)

if u has 1 right sibling, merge it with p(u)

if u has no right siblings, “jump”!



THE CORE ALGORITHM

?

How many steps do we take?

Start the algorithm on u; the map-distance between u and the root is |u| minus the number of steps taken in "jump" state.

white leaf

go to p(u), erase u

white non-leaf

if u has a right sibling, move to p(u), erase u (and descendants)

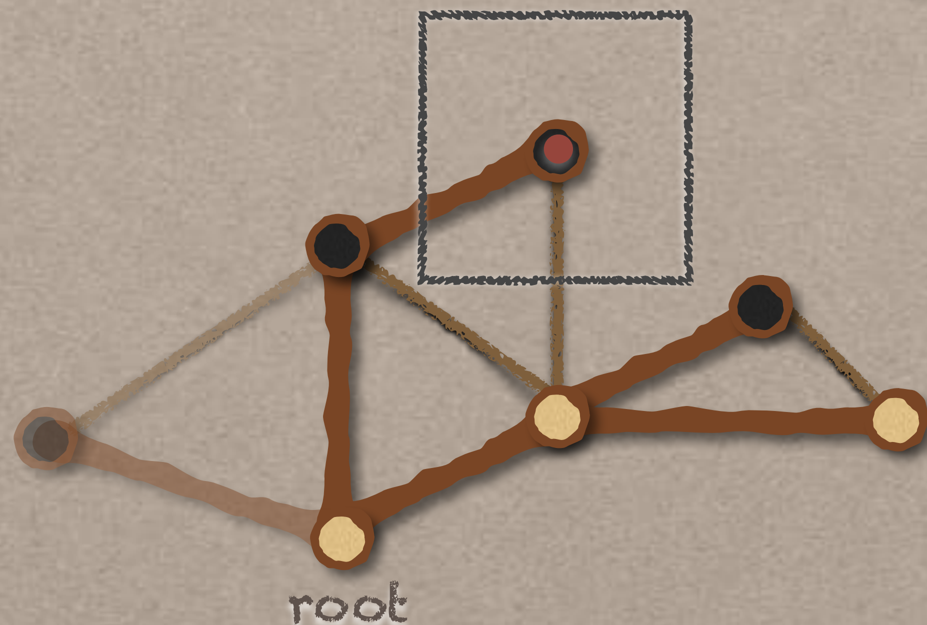
if u has no right siblings, merge p(u) with r(u); go to p(u)

black

if u has at least 2 right siblings, go to p(u)

if u has 1 right sibling, merge it with p(u)

if u has no right siblings, "jump"!



THE CORE ALGORITHM

?

How many steps do we take?

Start the algorithm on u; the map-distance between u and the root is |u| minus the number of steps taken in "jump" state.

white leaf

white non-leaf

jump

black

has height n

```
input: (t, u, s₀)
```

THE CORE ALGORITHM

?

How many steps do we take?

Start the algorithm on u; the map-distance between u and the root is |u| minus the number of steps taken in "jump" state.

white leaf

white non-leaf

jump

black

has height n

input: (t, u, s_0)

(t_1, u_1, s_1)

(t_2, u_2, s_2)

(t_3, u_3, s_3)

(t_4, u_4, s_4)

…

(t_n, root, s_n)

has height 0

THE CORE ALGORITHM

? How many steps do we take?

Start the algorithm on u; the map-distance between u and the root is |u| minus the number of steps taken in "jump" state.

white leaf

white non-leaf

jump

black

has height n

input: (t, u, s_0)

(t_1, u_1, s_1)

(t_2, u_2, s_2)

(t_3, u_3, s_3)

(t_4, u_4, s_4)

…

(t_n, root, s_n)

has height 0

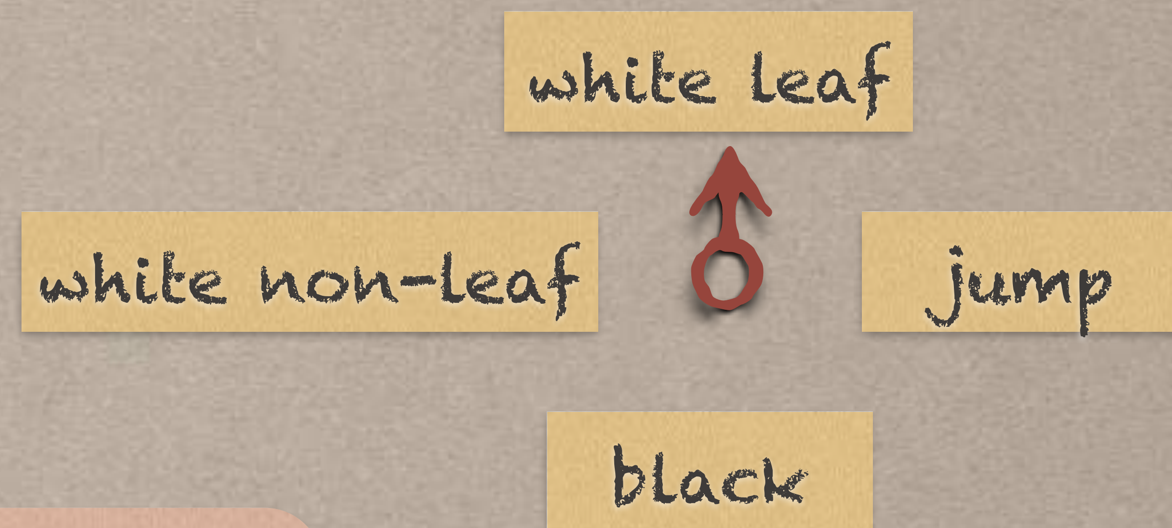
? What if we take a random pair (t,u) as input?

THE CORE ALGORITHM

? How many steps do we take?

Start the algorithm on u; the map-distance between u and the root is n minus the number of steps taken in "jump" state.

? What if we take a random pair (t,u) as input?



has height n

input: (t, u, s_0)

(t_1, u_1, s_1)

(t_2, u_2, s_2)

(t_3, u_3, s_3)

(t_4, u_4, s_4)

…

(t_n, root, s_n)

has height 0

THE CORE ALGORITHM

? How many steps do we take?

Start the algorithm on u; the map-distance between u and the root is n minus the number of steps taken in "jump" state.

? What if we take a random pair (t,u) as input?

The sequence of states $s_1,\dots,s_{n-2}$ is a Markov chain!

white leaf

white non-leaf

jump

black

has height n

input: (t, u, s_0)

(t_1, u_1, s_1)

(t_2, u_2, s_2)

(t_3, u_3, s_3)

(t_4, u_4, s_4)

…

(t_n, root, s_n)

has height 0

THE CORE ALGORITHM

? How many steps do we take?

Start the algorithm on u; the map-distance between u and the root is n minus the number of steps taken in "jump" state.

? What if we take a random pair (t,u) as input?

The sequence of states $s_1,\dots,s_{n-2}$ is a Markov chain!

ACHTUNG!

- One needs to choose the right distribution on pairs (geometric Galton-Watson trees, critical geometric Galton-Watson tree conditioned to survive...)

white leaf

white non-leaf

jump

black

has height n

input: (t, u, s_0)

(t_1, u_1, s_1)

(t_2, u_2, s_2)

(t_3, u_3, s_3)

(t_4, u_4, s_4)

...

$(t_n, \mathrm{root}, s_n)$

has height 0

THE CORE ALGORITHM

? How many steps do we take?

Start the algorithm on u; the map-distance between u and the root is n minus the number of steps taken in "jump" state.

? What if we take a random pair (t,u) as input?

The sequence of states $s_1,\dots,s_{n-2}$ is a Markov chain!

ACHTUNG!

- One needs to choose the right distribution on pairs (geometric Galton-Watson trees, critical geometric Galton-Watson tree conditioned to survive...)
- The white rightmost branch poses some problems (one may as well work with uniformly bicoloured trees, and the conditioning will disappear in the limit)

white leaf

white non-leaf

jump

black

has height n

input: (t, u, s_0)

(t_1, u_1, s_1)

(t_2, u_2, s_2)

(t_3, u_3, s_3)

(t_4, u_4, s_4)

...

(t_n, root, s_n)

has height 0

THE MARKOV CHAIN

?

What if we take a random pair (t,u) as input?

The sequence of states $s_1,\ldots,s_{n-2}$ is a Markov chain!

$$
\begin{array}{c|cccc}
\Rsh & w_0 & w_{>0} & b & j \\
\hline
w_0 & 1/4 & 1/4 & 1/2 & 0 \\
w_{>0} & 1/16 & 5/16 & 5/8 & 0 \\
b & 3/32 & 7/32 & 7/16 & 1/4 \\
j & 1/8 & 1/8 & 1/4 & 1/2
\end{array}
$$

THE MARKOV CHAIN

?

What if we take a random pair (t,u) as input?

The sequence of states $s_1,\dots,s_{n-2}$ is a Markov chain!

$$
\begin{array}{c|c}
\Rsh & \begin{array}{cccc} w_0 & w_{>0} & b & j \end{array} \\
\hline
\begin{array}{c} w_0 \\ w_{>0} \\ b \\ j \end{array} & \begin{pmatrix} 1/4 & 1/4 & 1/2 & 0 \\ 1/16 & 5/16 & 5/8 & 0 \\ 3/32 & 7/32 & 7/16 & 1/4 \\ 1/8 & 1/8 & 1/4 & 1/2 \end{pmatrix}
\end{array}
$$

stationary distribution

$$\pi = \frac{1}{9}\,(1,2,4,2)$$

THE MARKOV CHAIN

?

What if we take a random pair (t,u) as input?

The sequence of states $s_1, \dots, s_{n-2}$ is a Markov chain!

$$\begin{array}{c|cccc} \Rsh & w_0 & w_{>0} & b & j \\ \hline w_0 & 1/4 & 1/4 & 1/2 & 0 \\ w_{>0} & 1/16 & 5/16 & 5/8 & 0 \\ b & 3/32 & 7/32 & 7/16 & 1/4 \\ j & 1/8 & 1/8 & 1/4 & 1/2 \end{array}$$

stationary distribution

$$\pi = \frac{1}{9}(1,2,4,2)$$

Law of Large Numbers

For a certain random variable $X_n=(t_n,u_n)$, we let $d(X_n)$ be the map-distance between u_n and the root of t_n. Then

$$\lim_{n\longrightarrow\infty} \frac{d(X_n)}{n} = 7/9$$

where the convergence is almost sure.

THE MARKOV CHAIN

?

What if we take a random pair (t,u) as input?

The sequence of states $s_1,\dots,s_{n-2}$ is a Markov chain!

$$\left(\begin{array}{c|cccc} \Rsh & w_0 & w_{>0} & b & j \\ \hline w_0 & 1/4 & 1/4 & 1/2 & 0 \\ w_{>0} & 1/16 & 5/16 & 5/8 & 0 \\ b & 3/32 & 7/32 & 7/16 & 1/4 \\ j & 1/8 & 1/8 & 1/4 & 1/2 \end{array}\right)$$

stationary distribution

$$\pi = \frac{1}{9}(1, 2, 4, 2)$$

Large Deviations

For a certain random variable $X_n=(t_n,u_n)$, we let $d(X_n)$ be the map-distance between u_n and the root of t_n. Then
for all $\varepsilon > 0$ there is n_ε such that for all $n \geq n_\varepsilon$

$$\mathbb{P}\left(\left|\frac{d(X_n)}{n} - \frac{7}{9}\right| \geq \varepsilon\right) \leq e^{-Cn}.$$

FINAL PROOFS

WHERE DO WE STAND ?

Let τ_n be a random well bicoloured tree with n vertices, and let $diam(\tau_n)$ be its (random) diameter; then for all $\varepsilon > 0$

$$\mathbb{P}\left(\exists u \in \tau_n \text{ s.t. } |d(\tau_n, u) - c|u|| \geq \varepsilon \max\{diam(\tau_n), \sqrt{n}\}\right) \longrightarrow 0.$$

FINAL PROOFS

WHERE DO WE STAND ?

Let τ_n be a random well bicoloured tree with n vertices, and let $diam(\tau_n)$ be its (random) diameter; then for all $\varepsilon > 0$

$$\mathbb{P}\left(\exists u \in \tau_n \text{ s.t. } |d(\tau_n, u) - c|u|| \geq \varepsilon \max\{diam(\tau_n), \sqrt{n}\}\right) \longrightarrow 0.$$

- Rerooting arguments are difficult to apply, but black vertices behave as the root in some sense;

FINAL PROOFS
WHERE DO WE STAND ?

Let τ_n be a random well bicoloured tree with n vertices, and let $diam(\tau_n)$ be its (random) diameter; then for all $\varepsilon > 0$

$$\mathbb{P}\left(\exists u \in \tau_n \text{ s.t. } |d(\tau_n, u) - c|u|| \geq \varepsilon \max\{diam(\tau_n), \sqrt{n}\}\right) \longrightarrow 0.$$

- Rerooting arguments are difficult to apply, but black vertices behave as the root in some sense;
- Black vertices are "dense" (i.e. there are no extremely large faces, with the exception of the outerface); the same result is true about generic pairs of vertices;

FINAL PROOFS

WHERE DO WE STAND ?

Let τ_n be a random well bicoloured tree with n vertices, and let $diam(\tau_n)$ be its (random) diameter; then for all $\varepsilon > 0$

$$\mathbb{P}\left(\exists u \in \tau_n \text{ s.t. } |d(\tau_n, u) - c|u|| \geq \varepsilon \max\{diam(\tau_n), \sqrt{n}\}\right) \longrightarrow 0.$$

- Rerooting arguments are difficult to apply, but black vertices behave as the root in some sense;
- Black vertices are "dense" (i.e. there are no extremely large faces, with the exception of the outerface); the same result is true about generic pairs of vertices;
- Outerplanar maps have the same scaling limit as random well bicoloured plane trees!

THANK YOU!