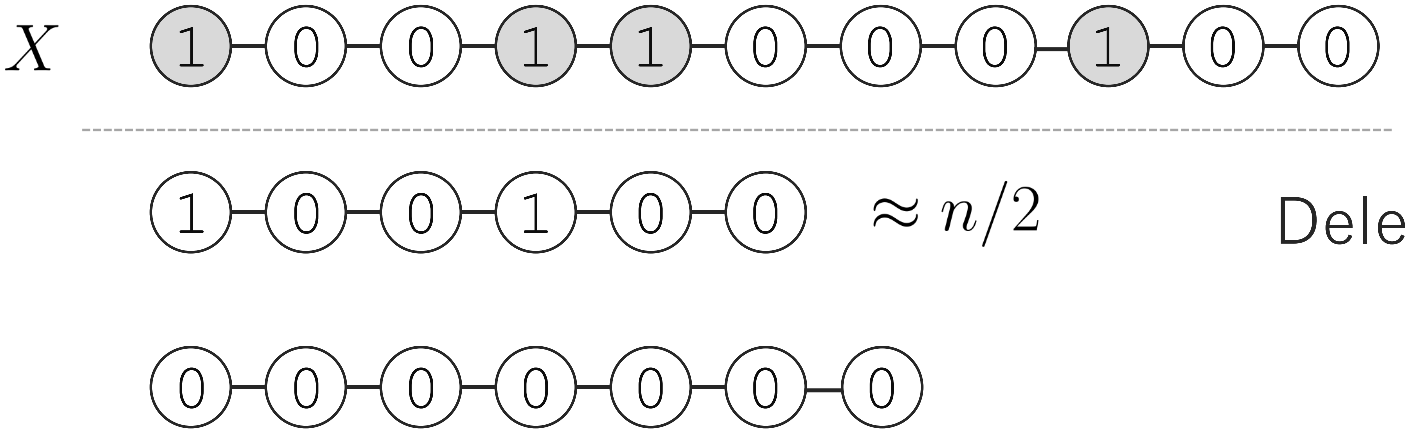
Trace Reconstruction

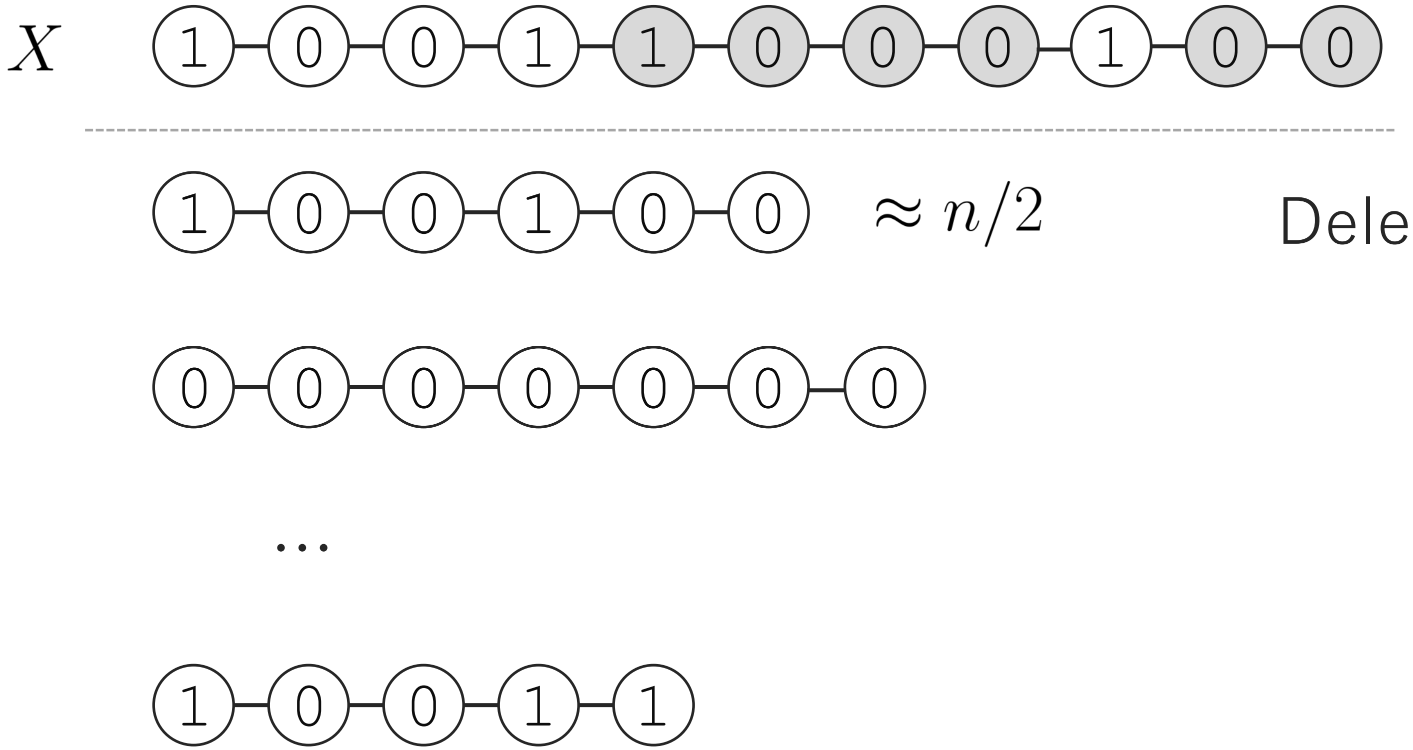
unknown worst-case string n bits



Deletion channel, probability *q=0.5*

Trace Reconstruction

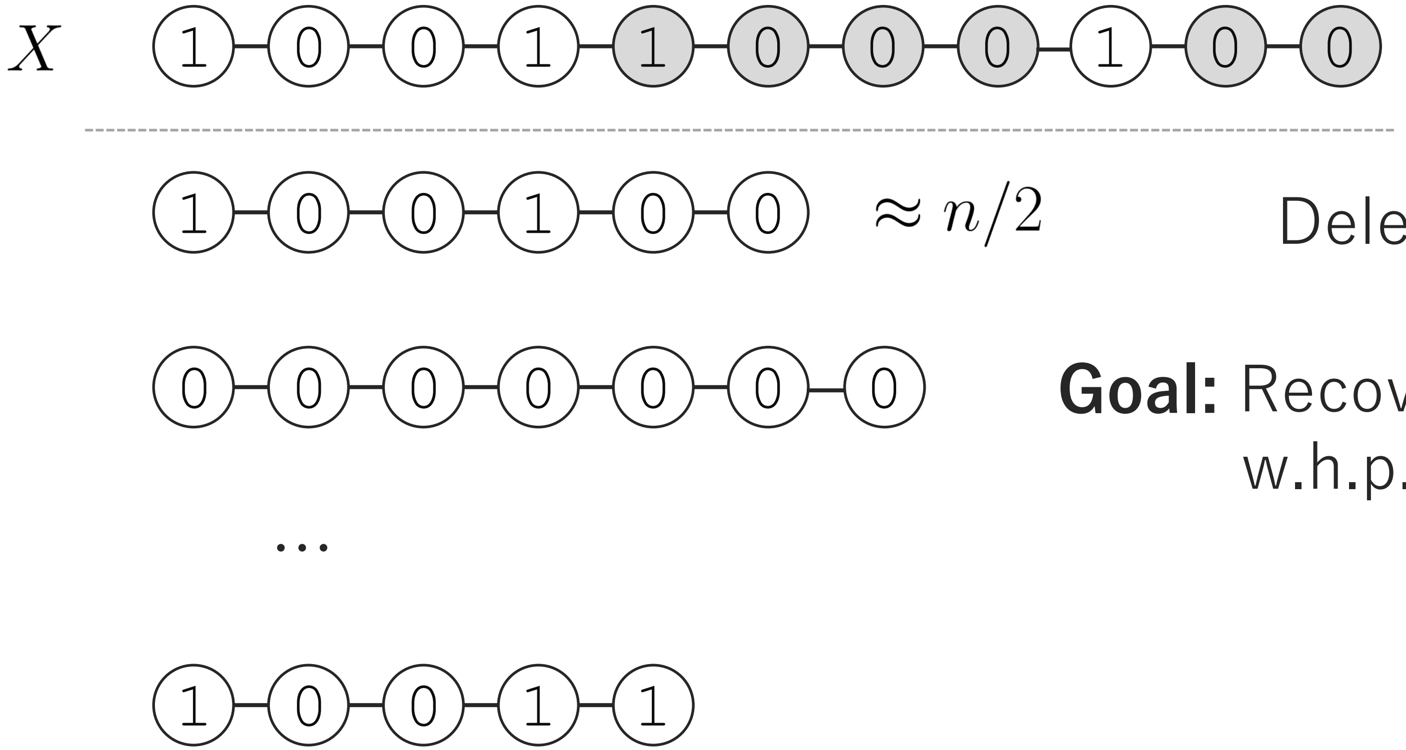
unknown worst-case string n bits



Deletion channel, probability q=0.5

Trace Reconstruction

unknown worst-case string n bits



Deletion channel, probability q=0.5

Goal: Recover ***X*** exactly
w.h.p. using min # traces

Trace Reconstruction

unknown worst-case string n bits

X 1 0 0 1 1 0 0 0 1 0 0

1 0 0 1 0 0

0 0 0 0 0 0 0

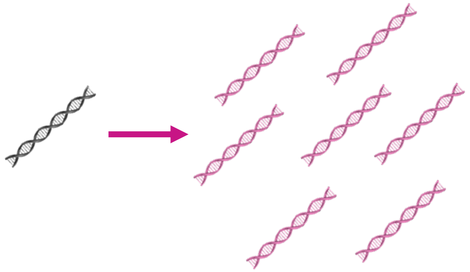
...

1 0 0 1 1

Motivation: Computational Biology

DNA Sequencing

illumina



DNA Data Storage

Past/ current work

Worst case:

- Batu, Kannan, Khanna, McGregor (2004):
  - Bitwise Majority Alignment (BMA) algorithm
  - $\Omega(n)$ traces needed when deletion probability is: $q \leq 1/n^{1/2+\varepsilon}$

Past/ current work

Worst case:

- Batu, Kannan, Khanna, McGregor (2004):
  - Bitwise Majority Alignment (BMA) algorithm
  - $\Omega(n)$ traces needed when deletion probability is: $q \leq 1/n^{1/2+\varepsilon}$
- Holenstein, Mitzenmacher, Panigrahy, Wieder (2008):
  - $\exp(\tilde{O}(n^{1/2}))$ traces suffice

Past/ current work

Worst case:

- Batu, Kannan, Khanna, McGregor (2004):
  - Bitwise Majority Alignment (BMA) algorithm
  - $\Omega(n)$ traces needed when deletion probability is: $q \leq 1/n^{1/2+\varepsilon}$
- Holenstein, Mitzenmacher, Panigrahy, Wieder (2008):
  - $\exp(\tilde{O}(n^{1/2}))$ traces suffice
- De, O'Donnell, Servedio (2017); Nazarov, Peres (2017):
  - $\exp(O(n^{1/3}))$ traces suffice
  - **Mean-based algorithms** using generating functions and complex analysis

$$\mathbb{E}\left[\sum_{j\geq 0} \widetilde{a}_j w^j\right] = p \sum_{k=0}^{n-1} a_k \left(pw + q\right)^k$$

Past/ current work

Worst case:

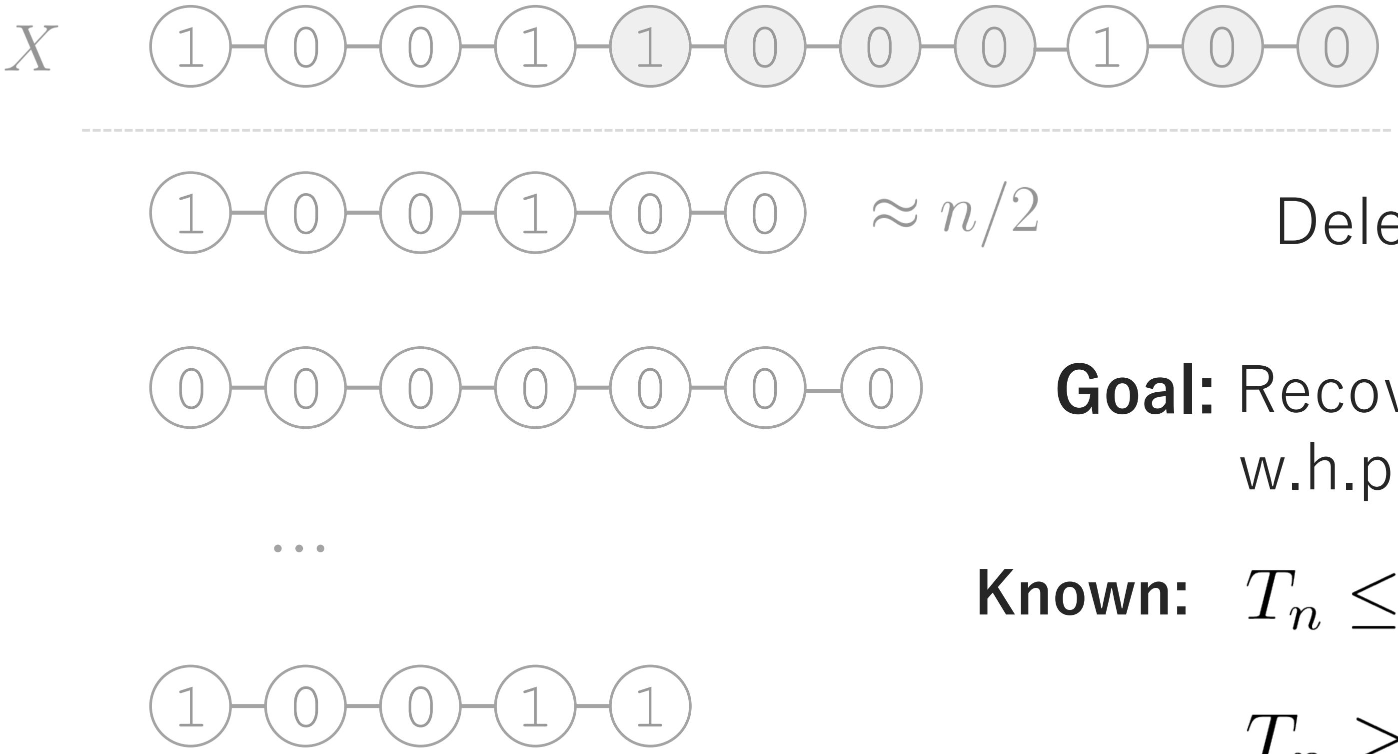
- Batu, Kannan, Khanna, McGregor (2004):
  - Bitwise Majority Alignment (BMA) algorithm
  - $\Omega(n)$ traces needed when deletion probability is: $q \leq 1/n^{1/2+\varepsilon}$
- Holenstein, Mitzenmacher, Panigrahy, Wieder (2008):
  - $\exp(\tilde{O}(n^{1/2}))$ traces suffice
- De, O'Donnell, Servedio (2017); Nazarov, Peres (2017):
  - $\exp(O(n^{1/3}))$ traces suffice
  - **Mean-based algorithms** using generating functions and complex analysis

$$\mathbb{E}\left[\sum_{j\geq 0} \widetilde{a}_j w^j\right] = p\sum_{k=0}^{n-1} a_k \left(pw+q\right)^k$$

- Holden, Lyons (2018); Chase (2019):
  - Improved lower bounds: $\widetilde{\Omega}(n^{3/2})$ traces needed

Trace Reconstruction

unknown worst-case string n bits



Deletion channel, probability q=0.5

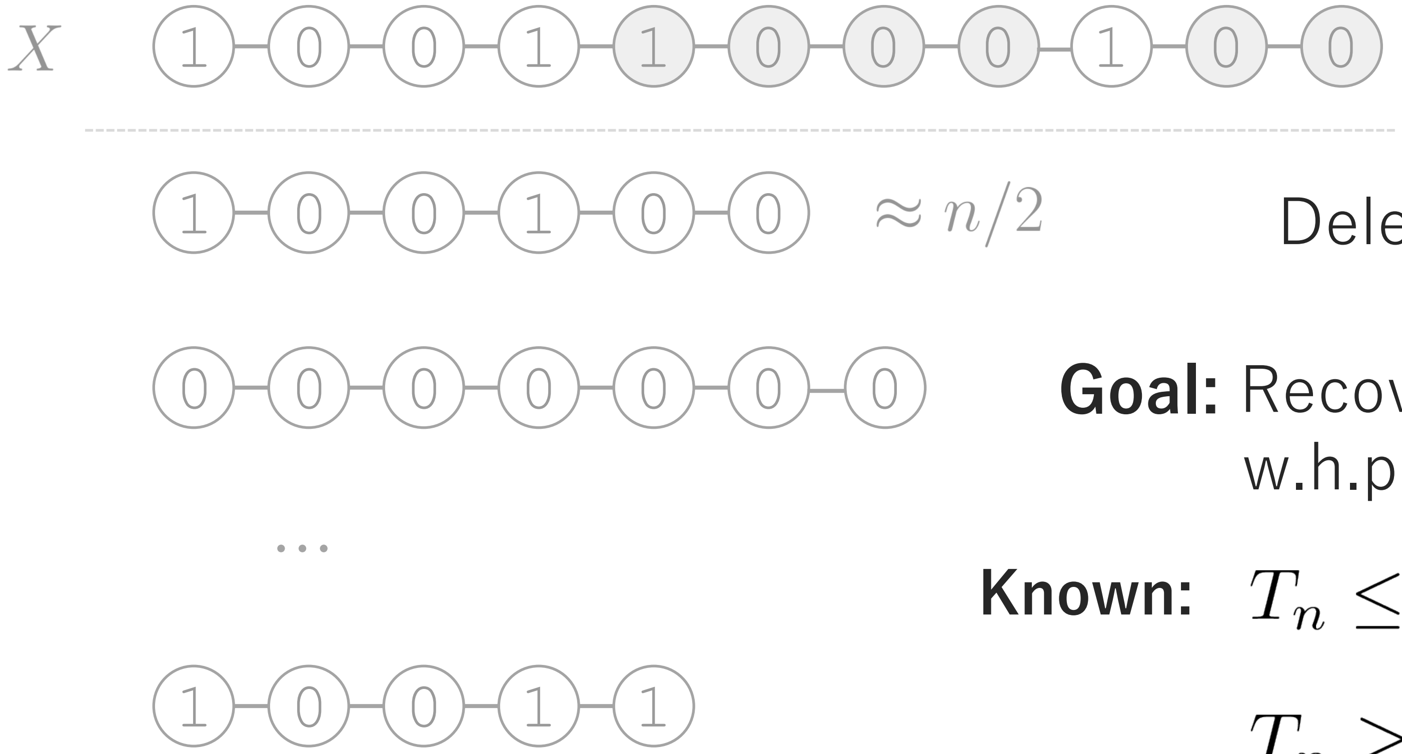
Goal: Recover X **exactly**
w.h.p. using min # traces T_n

Known: $T_n \leq \exp(n^{1/3})$ [Nazarov, Peres '16; De, O'Donnell, Servedio '16]

$T_n \geq \widetilde{\Omega}(n^{3/2})$ [Holden, Lyons '18; Chase '19]

Trace Reconstruction

unknown worst-case string n bits



Deletion channel, probability q=*0.5*

Goal: Recover X **exactly**
w.h.p. using min # traces T_n

Known: $T_n \leq \exp(n^{1/3})$ [Nazarov, Peres '16; De, O'Donnell, Servedio '16]

$T_n \geq \widetilde{\Omega}(n^{3/2})$ [Holden, Lyons '18; Chase '19]

Take away: huge gap between upper and lower bound! New ideas needed to improve upper bound

Trace Reconstruction Variants

- **coded TR:** encoded initial string X [Cheraghchi, Gabrys, Milenkovic, Ribeiro '19; Brakensiek, Li, Spang '19]
- **average-case:** X random → $\exp((\log n)^{1/3})$ [Peres-Zhai '17; Holden, Pemantle, Peres '18]
- **population recovery:** multiple unknown strings [Ban, Chen, Freilich, Servedio, Sinha '19]
- **matrix version:** delete random rows/cols [Krishnamurthy, Mazumdar, McGregor, Pal '19]
- **fixed # deletions:** e.g., 1, 2, 3, ... [Levenshtein '01; Gabrys, Yaakobi '18]
- **Tree TR:** reconstruct labelled trees **[Davies, Racz, Rashtchian '19]**

Deterministic Variants

- **k-deck:** reconstruct from all k-substrings $k \leq O(\sqrt{n})$ [Krasikov, Roditty '97]
- **Graph Reconstruction Conj:** all (n-1)-vertex subgraphs? [Kelly '57; Ulam '60]

Trace Reconstruction Variants

- **coded TR:** encoded initial string X
- **average-case:** X random → $\exp((\log n)^{1/3})$
- **population recovery:** multiple unknown strings
- **matrix version:** delete random rows/cols
- **fixed # deletions:** e.g., 1, 2, 3, ...
- **Tree TR:** reconstruct labelled trees [Davies, Racz, Rashtchian '19]

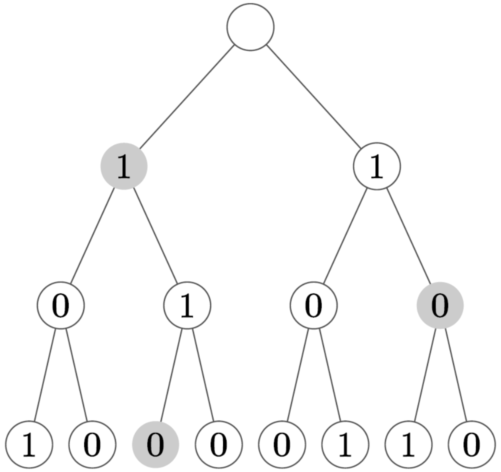
We'll come back to these later!

Deterministic Variants

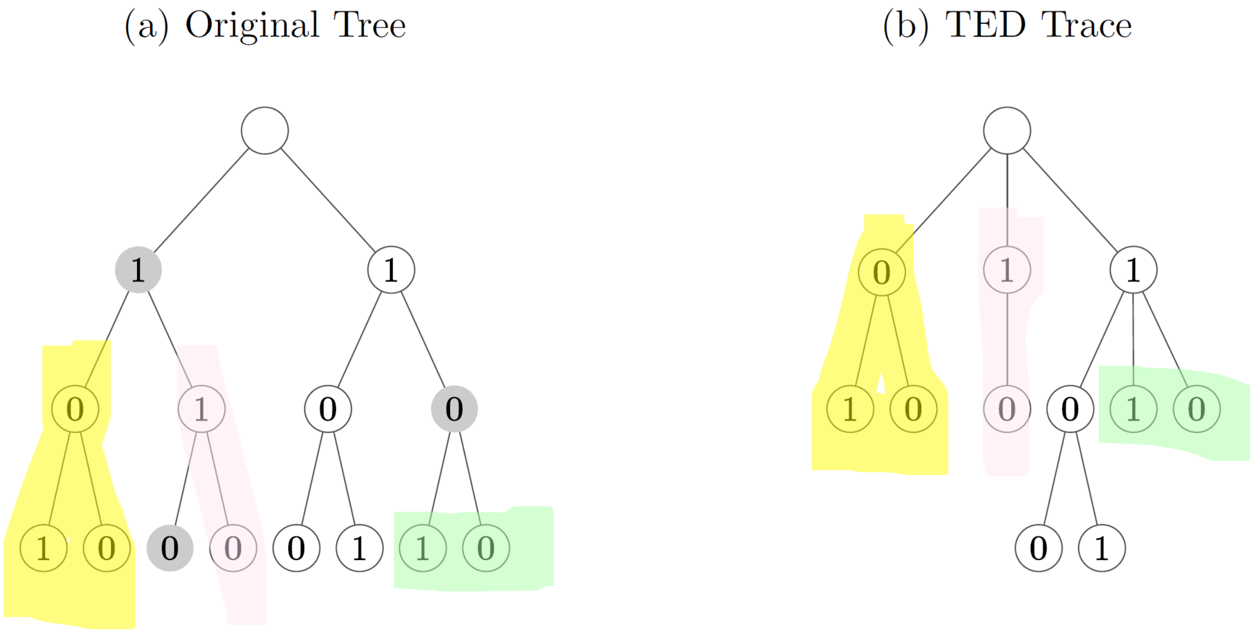
- **k-deck:** reconstruct from all k-substrings $k \leq O(\sqrt{n})$
- **Graph Reconstruction Conj:** all (n-1)-vertex subgraphs?

Generalization to Trees

(a) Original Tree



Generalization to Trees

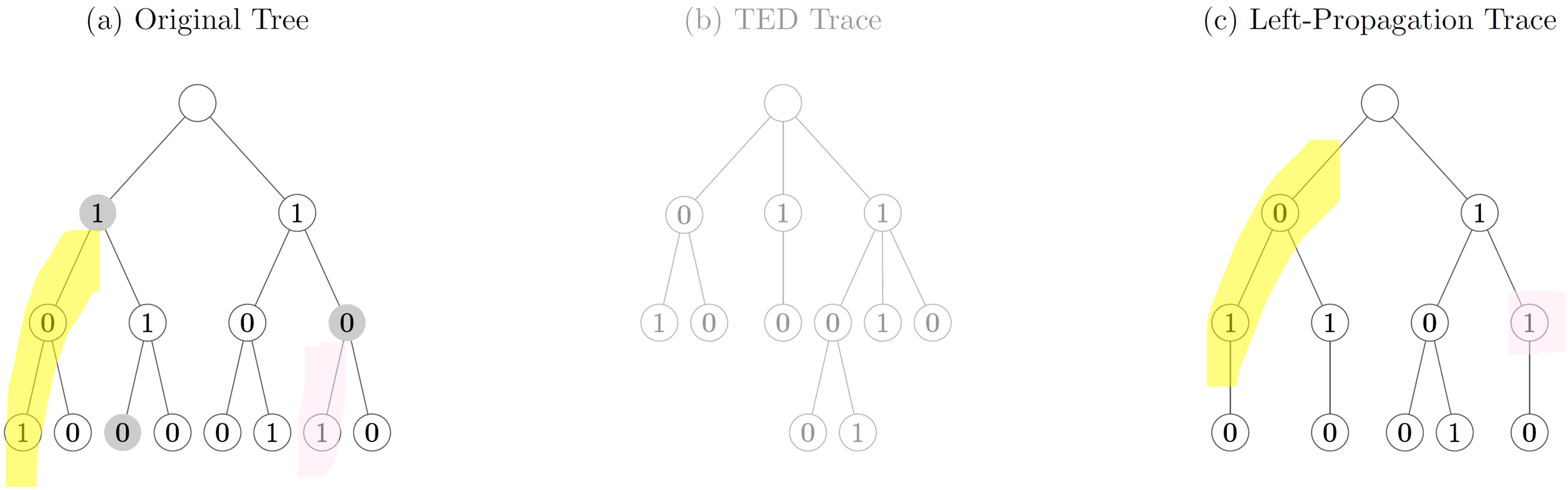


tree edit distance

children move up

degree may increase

Generalization to Trees



tree edit distance

children move up

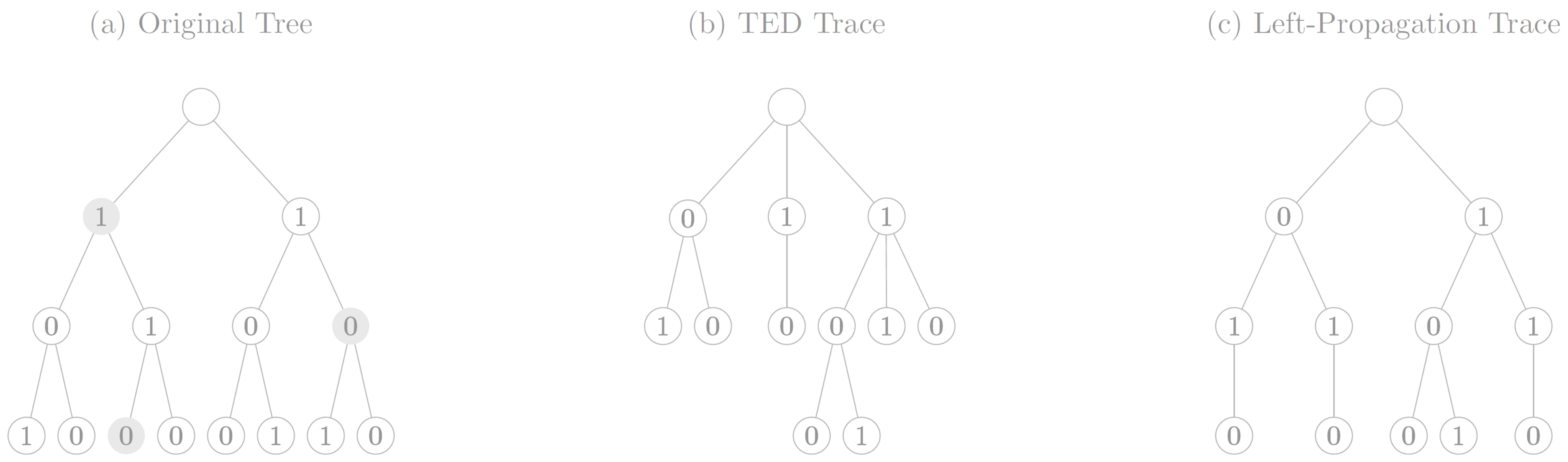
degree may increase

left propagation model

left child moves up

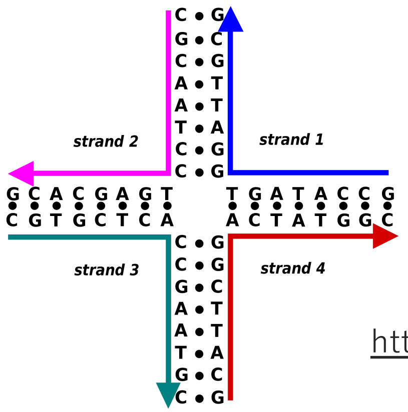
degree never increases

Why trees?



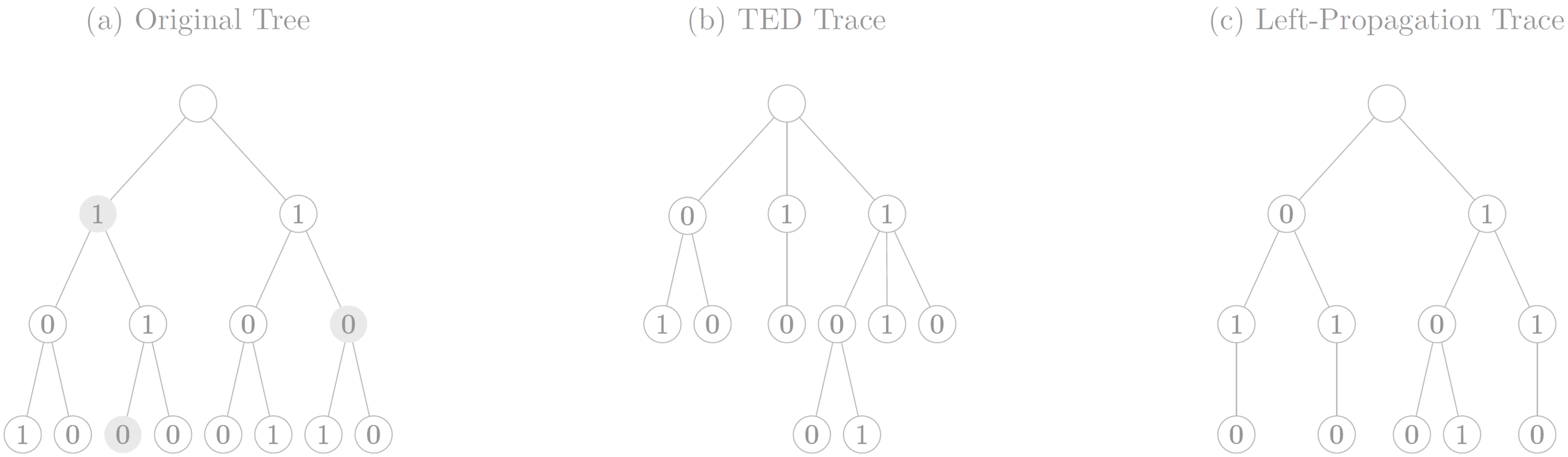
(Kind of, maybe) Practical interest

Tree-structured DNA



https://en.wikipedia.org/wiki/Holliday_junction

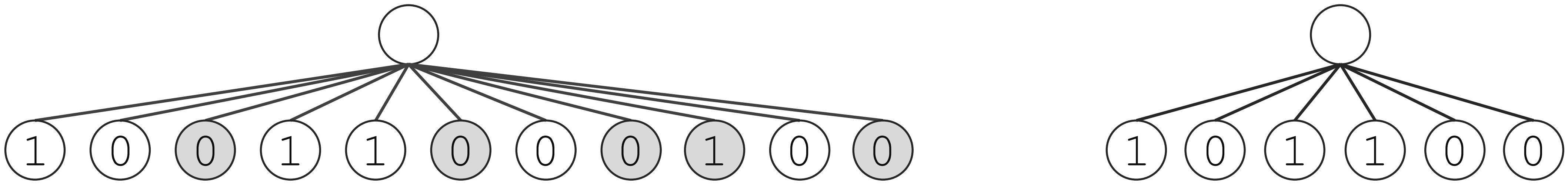
Why trees?



Theoretical interest

How does the addition of combinatorial structure allow us to move away from purely analytic methods (aka mean-based algorithms) in finding upper bounds for TR?

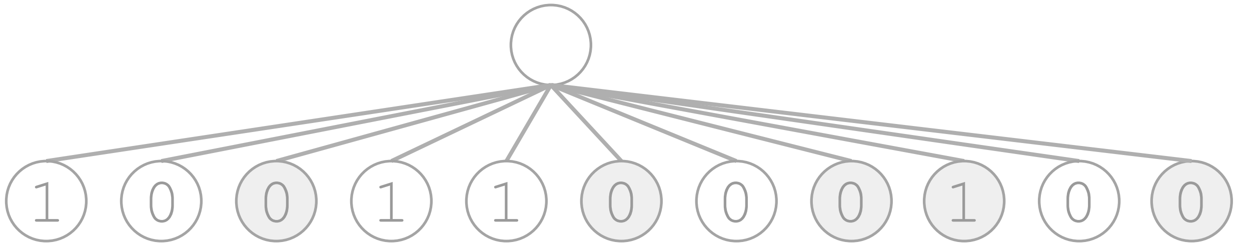
Generalization to Trees

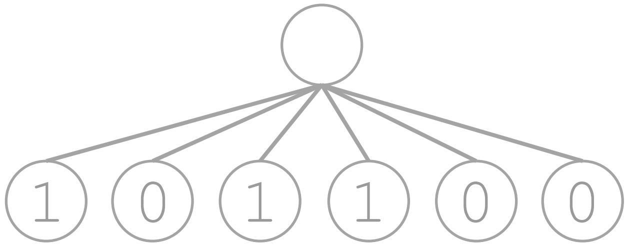


Deletion channel, probability 0.5

Goal: Recover X w.h.p. using min # traces

Generalization to Trees



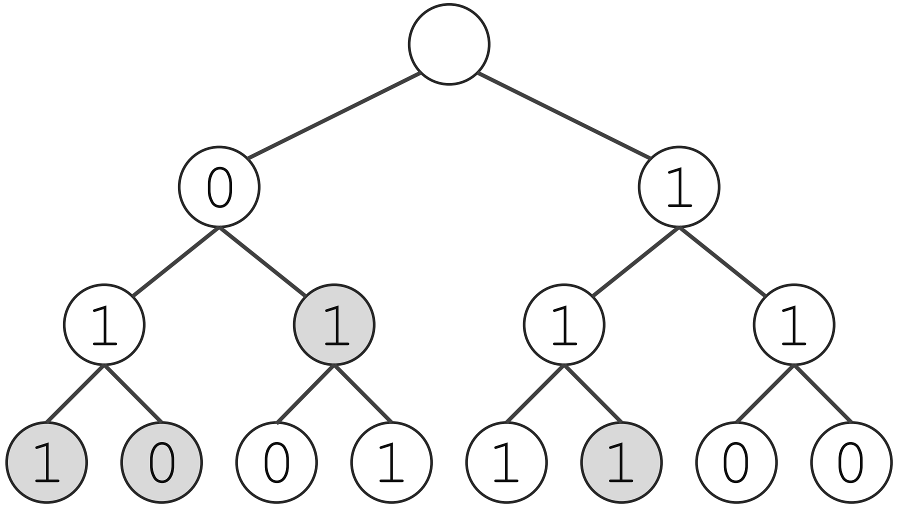


Deletion channel, probability 0.5

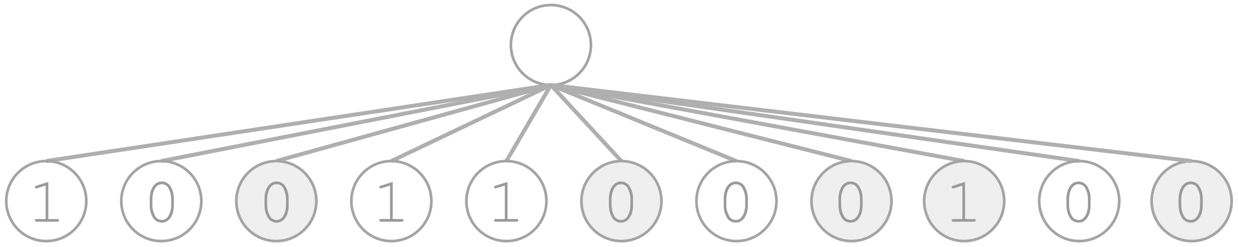
Goal: Recover X w.h.p. using min # traces

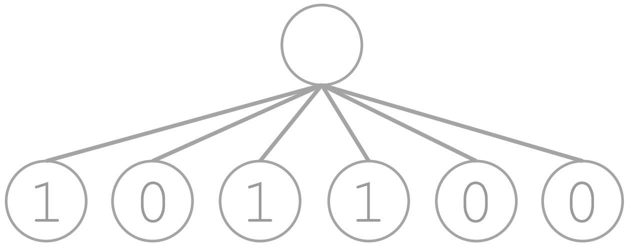
Tree Edit Distance (TED) Model

X



Generalization to Trees

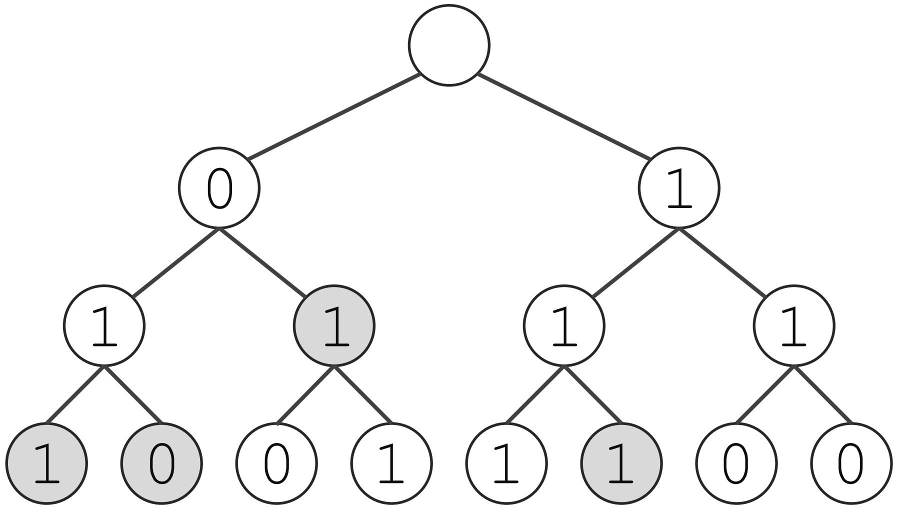


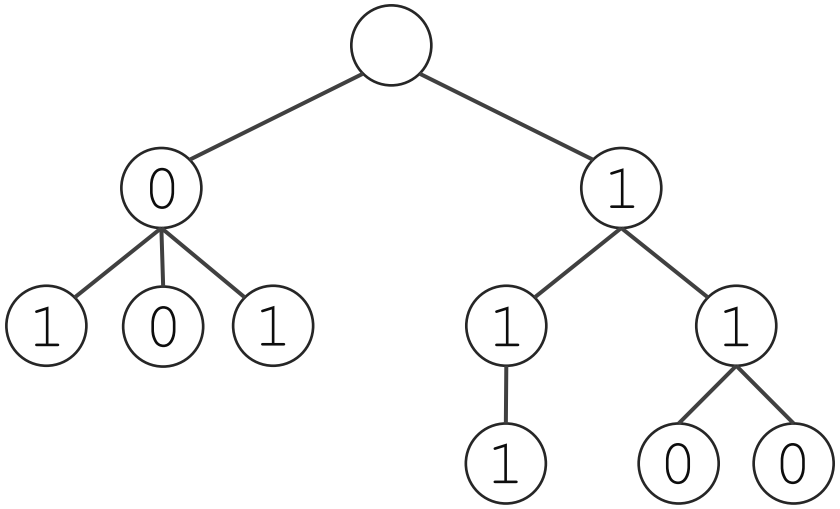


Tree Edit Distance (TED) Model

Vertex Deletion → Children Move Up
(fixed root)

X





Generalization to Trees

Fixed root w.l.o.g. → sample more traces

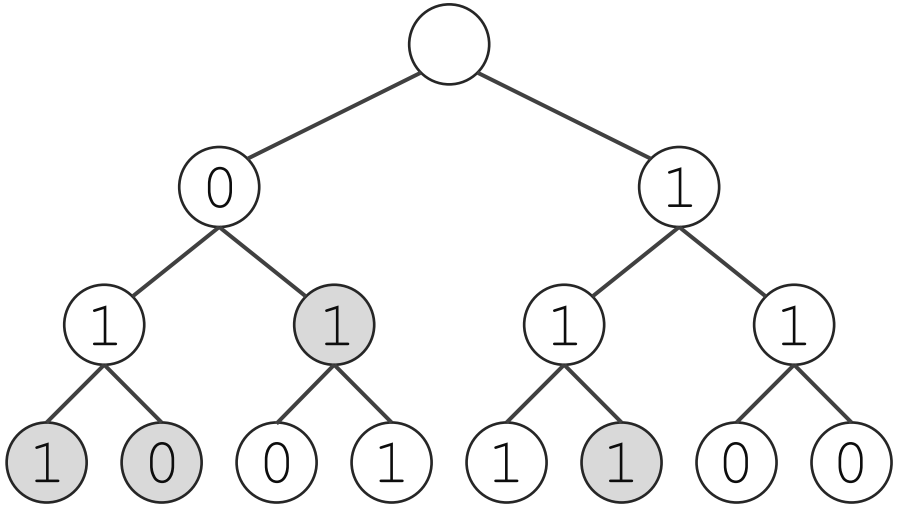
Consistent planar embedding (left-right)

Random tree “close” in Tree Edit Distance

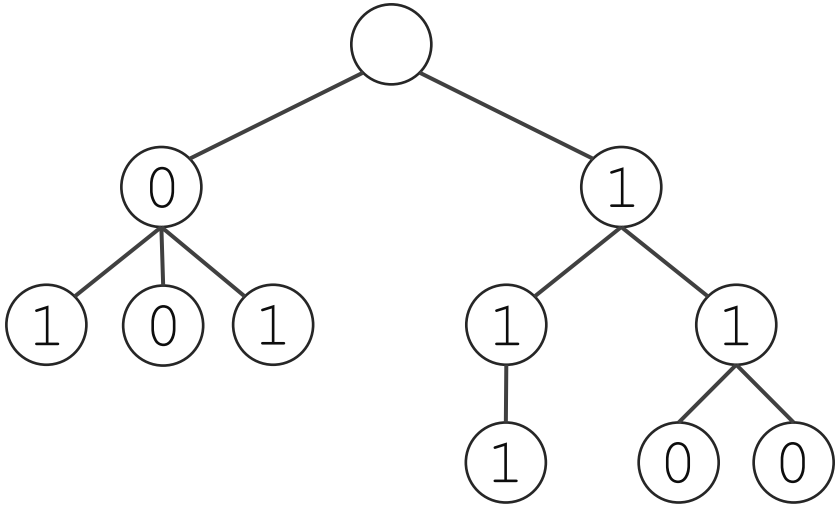
Equivalent: contract edge, keep parent’s label

Tree Edit Distance (TED) Model

X



Vertex Deletion → Children Move Up
(fixed root)



Our Results

unknown worst-case tree X with n vertices
deletion probability $q \in (0,1)$

Our Results

unknown worst-case tree X with n vertices
deletion probability $q \in (0,1)$

Theorem 1 $\exp(k \log_k n)$ traces
to reconstruct complete k-ary trees

Our Results

unknown worst-case tree X with n vertices
deletion probability $q \in (0,1)$

Theorem 1 $\exp\left(k \log_k n\right)$ traces
to reconstruct complete k-ary trees

Theorem 2 $\exp\left(k^{1/3} + \log_k n\right)$ traces
to reconstruct complete k-ary trees
if $k \geq c \log^2(n)$

Our Results

unknown worst-case tree X with n vertices
deletion probability $q \in (0, 1)$

Theorem 1 $\exp(k \log_k n)$ traces
to reconstruct complete k-ary trees

Theorem 2 $\exp\left(k^{1/3} + \log_k n\right)$ traces
to reconstruct complete k-ary trees
if $k \geq c \log^2(n)$

$\mathrm{poly}(n)$ traces for complete trees if

- $k = O(1)$
- $c \log^2 n \leq k \leq O(\log^3 n)$

Our Results

unknown worst-case tree X with n vertices
deletion probability $q \in (0,1)$

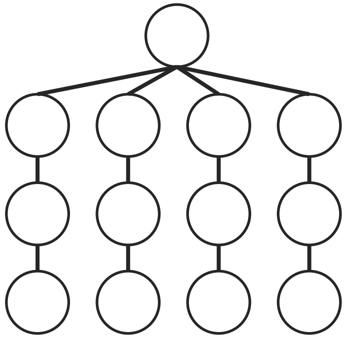
Theorem 1 $\exp(k \log_k n)$ traces
to reconstruct complete k-ary trees

Theorem 2 $\exp\left(k^{1/3} + \log_k n\right)$ traces
to reconstruct complete k-ary trees
if $k \geq c \log^2(n)$

Theorem 3

$\mathrm{poly}(n)$ traces for complete trees if

- $k = O(1)$
- $c \log^2 n \leq k \leq O(\log^3 n)$



(n,d)-spider
n/d paths
depth d

Our Results

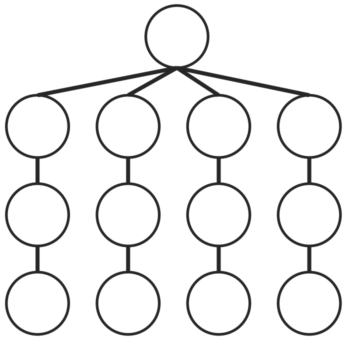
unknown worst-case tree X with n vertices
deletion probability $q \in (0,1)$

Theorem 1 $\exp\left(k \log_k n\right)$ traces
to reconstruct complete k-ary trees

poly(n) traces for complete trees if
- $k = O(1)$
- $c\log^2 n \leq k \leq O(\log^3 n)$

Theorem 2 $\exp\left(k^{1/3} + \log_k n\right)$ traces
to reconstruct complete k-ary trees
if $k \geq c\log^2(n)$

Theorem 3 $\exp\left(\widetilde{O}\left(n^{1/3} q^{d/3}\right)\right)$ traces
to reconstruct (n,d)-spiders $d \leq \log_{1/q}(n)$



(n,d)-spider
n/d paths
depth d

Our Results

unknown worst-case tree X with n vertices
deletion probability $q \in (0,1)$

Theorem 1 $\exp\left(k \log_k n\right)$ traces
to reconstruct complete k-ary trees

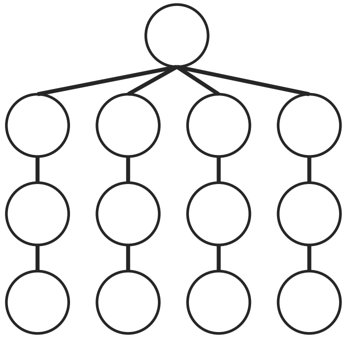
Theorem 2 $\exp\left(k^{1/3} + \log_k n\right)$ traces
to reconstruct complete k-ary trees
if $k \geq c \log^2(n)$

Theorem 3 $\exp\left(\widetilde{O}\left(n^{1/3} q^{d/3}\right)\right)$ traces
to reconstruct (n,d)-spiders $d \leq \log_{1/q}(n)$

traces for **spiders**

$d = \alpha \log_{1/q}(n) \quad \alpha \in (0,1)$

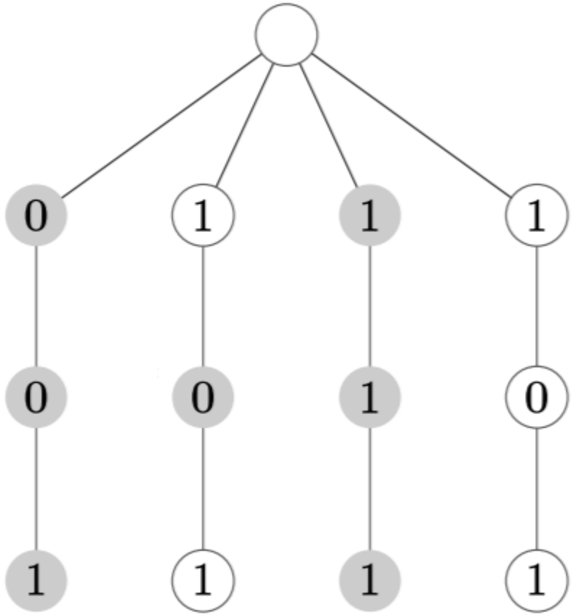
- Previously $\exp\left(\widetilde{O}\left(n^{1-\alpha}\right)\right)$
- Our Result $\exp\left(\widetilde{O}\left(n^{\frac{1-\alpha}{3}}\right)\right)$



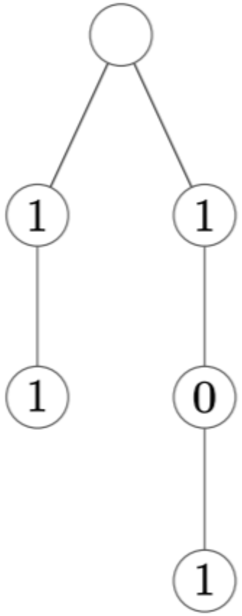
(n,d)-spider
n/d paths
depth d

Spider Trees

original tree

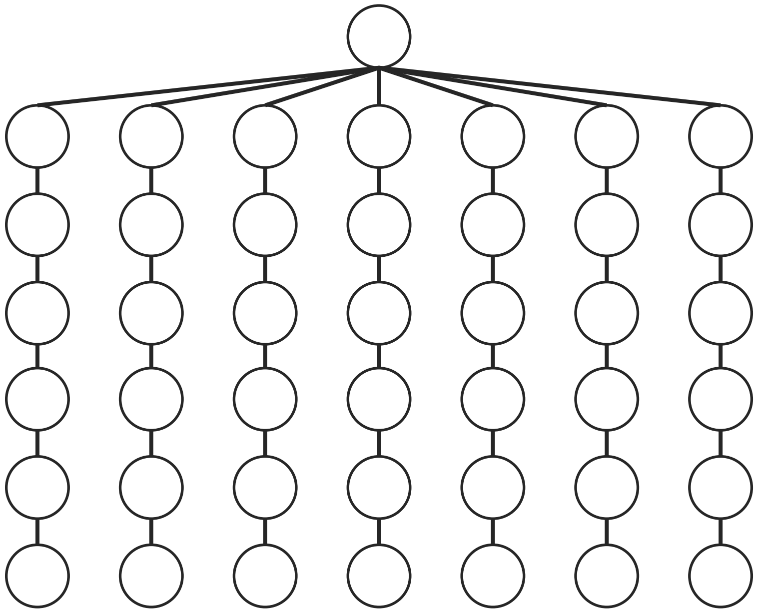


trace



Spider Trees

(n, d)-spider



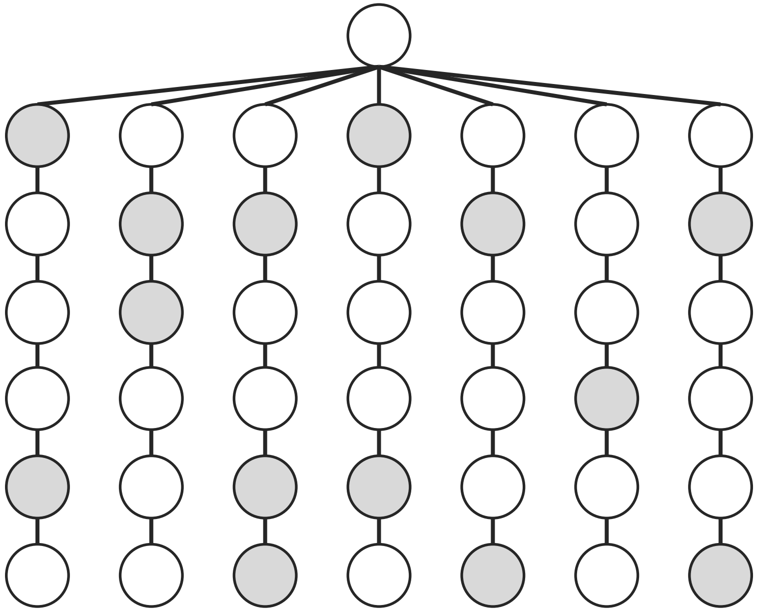
depth d

n/d paths

Spider Trees

(n, d)-spider

depth d



n/d paths

Easy regime: depth $d \geq \log(n)$

only keep traces with n/d paths

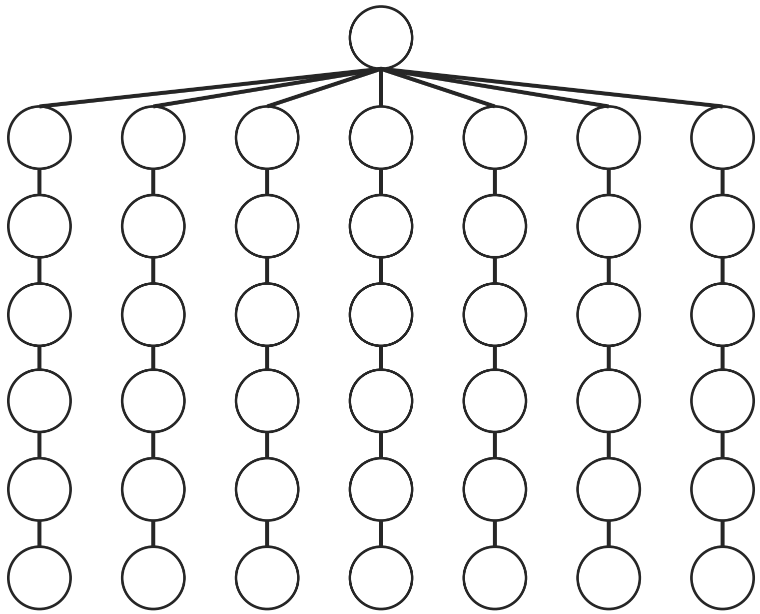
for each, use string TR

$$T_d \leq \exp(d^{1/3})$$

Spider Trees

(n, d)-spider

depth d



n/d paths

Easy regime: depth $d \geq \log(n)$

only keep traces with n/d paths

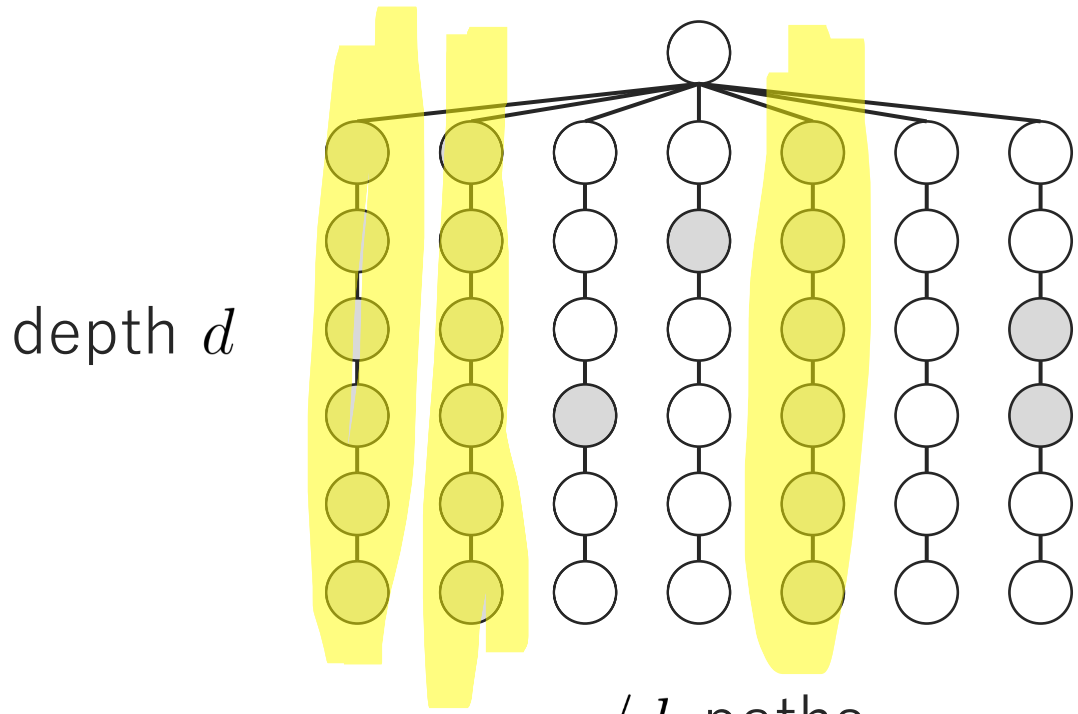
for each, use string TR

$$T_d \leq \exp(d^{1/3})$$

Hard regime: depth $d < \log(n)$

Spider Trees

(n, d)-spider



n/d paths

Deletion prob $0 < q \leq 0.7$

Easy regime: depth $d \geq \log(n)$

only keep traces with n/d paths

for each, use string TR

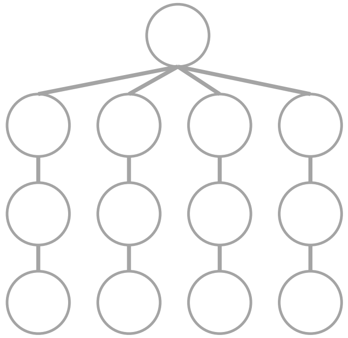
$$T_d \leq \exp(d^{1/3})$$

Hard regime: depth $d < \log(n)$

Full paths deleted

Theorem 3 $\exp\left(\tilde{O}\left(n^{1/3} q^{d/3}\right)\right)$ traces to reconstruct (n, d)-spiders $d \leq \log_{1/q}(n)$

Theorem 3 $\exp\left(\widetilde{O}\left(n^{1/3}q^{d/3}\right)\right)$ traces to reconstruct (n,d)-spiders $d \leq \log_{1/q}(n)$

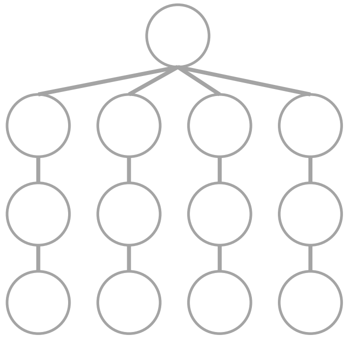


(n,d)-spider

n/d paths

depth d

Theorem 3 $\exp\left(\widetilde{O}\left(n^{1/3}q^{d/3}\right)\right)$ traces to reconstruct (n,d)-spiders $d \leq \log_{1/q}(n)$



(n,d)-spider

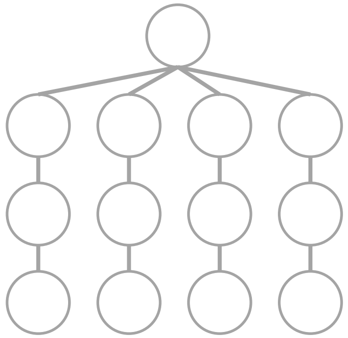
n/d paths

depth d

Mean-based Algorithm [Nazarov-Peres '16; De, O'Donnell, Servedio '16]

X^1 vs. X^2 $\quad$ $\exists\, j$ such that average of j^{th} bit of trace differs by $\exp(-L)$

Theorem 3 $\exp\left(\widetilde{O}\left(n^{1/3}q^{d/3}\right)\right)$ traces to reconstruct (n,d)-spiders $d \le \log_{1/q}(n)$



(n,d)-spider

n/d paths

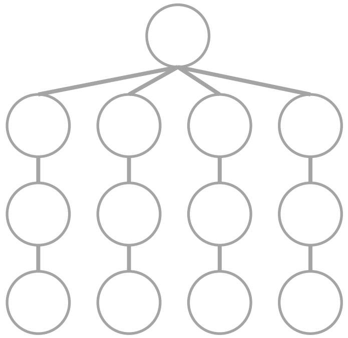
depth d

Mean-based Algorithm [Nazarov-Peres '16; De, O'Donnell, Servedio '16]

X^1 vs. X^2

$\exists\, j$ such that average of j^{th} bit of trace differs by $\exp(-L)$ $\implies$ $\exp(O(L))$ traces suffice

Theorem 3 $\exp\left(\widetilde{O}\left(n^{1/3}q^{d/3}\right)\right)$ traces to reconstruct (n,d)-spiders $d \le \log_{1/q}(n)$



(n,d)-spider
n/d paths
depth d

Mean-based Algorithm [Nazarov-Peres '16; De, O'Donnell, Servedio '16]

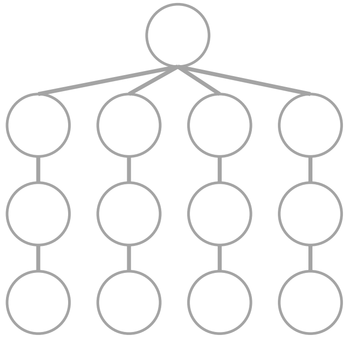
X^1 vs. X^2 $\quad \exists\, j$ such that average of j^{th} bit of trace differs by $\exp(-L)$ $\implies$ $\exp\left(O(L)\right)$ traces suffice

union bound over all pairs of spiders

for every pair of spiders, $\exists$ a coordinate where in expectation traces look different

mean of this coordinate over traces tells us which of pair is more likely to be X

Theorem 3 $\exp\left(\widetilde{O}\left(n^{1/3}q^{d/3}\right)\right)$ traces to reconstruct (n,d)-spiders $d \leq \log_{1/q}(n)$



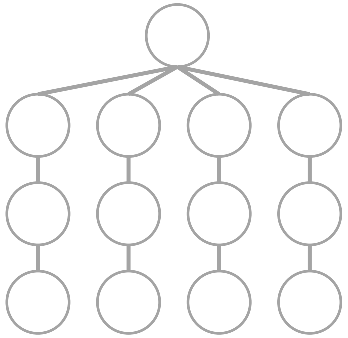
(n,d)-spider
n/d paths
depth d

Mean-based Algorithm [Nazarov-Peres '16; De, O'Donnell, Servedio '16]

X^1 vs. X^2 $\quad \exists\, j$ such that average of j^{th} bit of trace differs by $\exp(-L)$ $\implies$ $\exp\left(O(L)\right)$ traces suffice

Strings: easy to determine how original bits affect trace 😀

Theorem 3 $\exp\left(\widetilde{O}\left(n^{1/3}q^{d/3}\right)\right)$ traces to reconstruct (n,d)-spiders $d \leq \log_{1/q}(n)$



(n,d)-spider

n/d paths

depth d

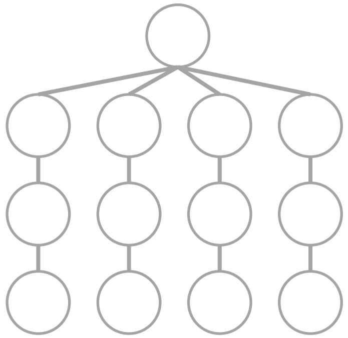
Mean-based Algorithm [Nazarov-Peres '16; De, O'Donnell, Servedio '16]

X^1 vs. X^2 $\exists\, j$ such that average of j^{th} bit of trace differs by $\exp(-L)$ $\implies$ $\exp\left(O(L)\right)$ traces suffice

Strings: easy to determine how original bits affect trace 😀

Spiders: more complicated "two-dimensional" 🤔

Theorem 3 $\exp\left(\widetilde{O}\left(n^{1/3}q^{d/3}\right)\right)$ traces to reconstruct (n,d)-spiders $d \le \log_{1/q}(n)$



(n,d)-spider
n/d paths
depth d

Mean-based Algorithm [Nazarov-Peres '16; De, O'Donnell, Servedio '16]

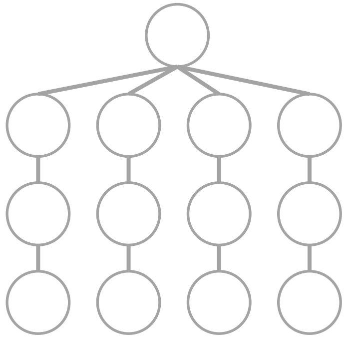
X^1 vs. X^2 $\exists\, j$ such that average of j^{th} bit of trace differs by $\exp(-L)$ $\implies$ $\exp\left(O(L)\right)$ traces suffice

Strings: easy to determine how original bits affect trace 😀

Spiders: more complicated "two-dimensional" 🤔

General trees: no idea... 😔

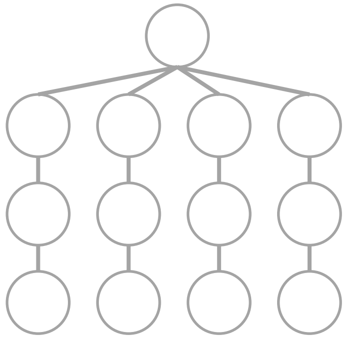
Theorem 3 $\exp\left(\widetilde{O}\left(n^{1/3}q^{d/3}\right)\right)$ traces to reconstruct (n,d)-spiders $d \le \log_{1/q}(n)$



Generating function

$w \in \mathbb{C}$

Theorem 3 $\exp\left(\widetilde{O}\left(n^{1/3}q^{d/3}\right)\right)$ traces to reconstruct (n,d)-spiders $d \le \log_{1/q}(n)$



(n,d)-spider

n/d paths

depth d

DFS indexing

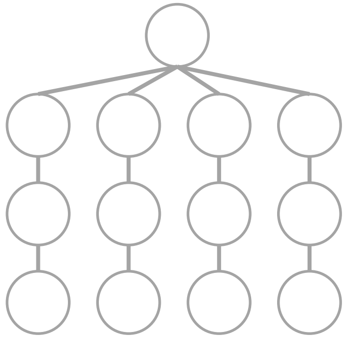
Generating function

original labels $a = a_0, a_1, \dots, a_{n-1}$

$w \in \mathbb{C}$

trace labels $b = b_0, b_1, \dots, b_{n'}, 0, \dots, 0$

Theorem 3 $\exp\left(\tilde{O}\left(n^{1/3}q^{d/3}\right)\right)$ traces to reconstruct (n,d)-spiders $d \le \log_{1/q}(n)$



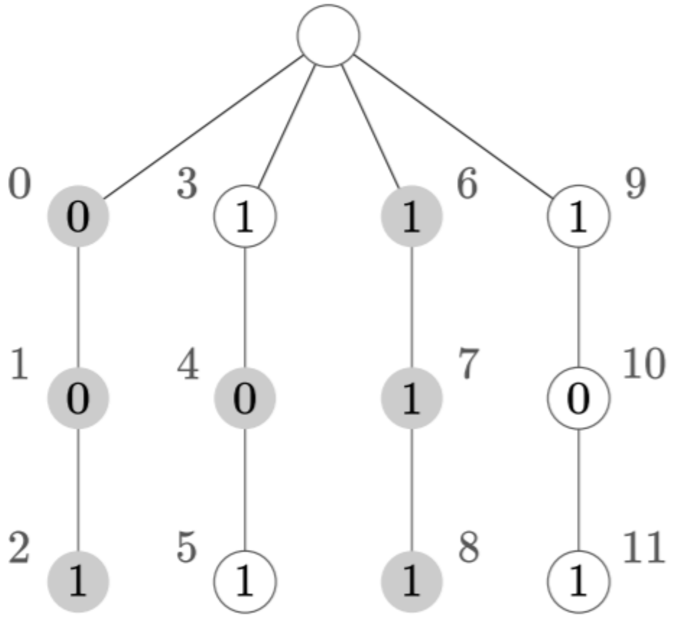
DFS indexing

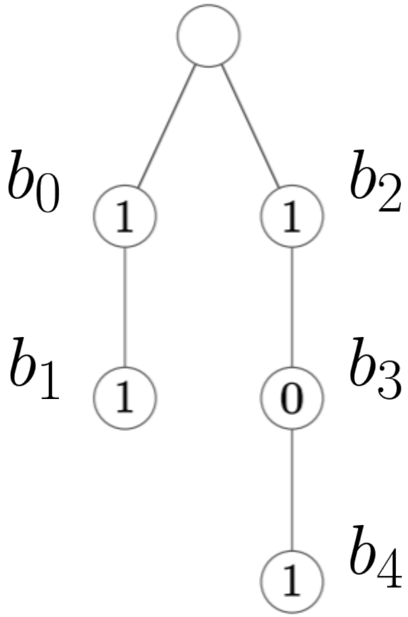
Generating function

original labels $a = a_0, a_1, \ldots, a_{n-1}$

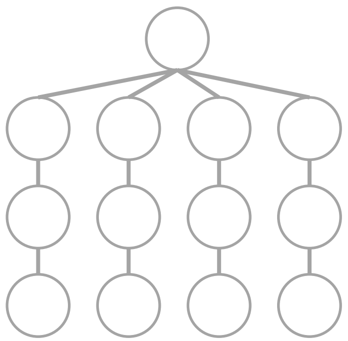
$w \in \mathbb{C}$

trace labels $b = b_0, b_1, \ldots, b_{n'}, 0, \ldots, 0$





Theorem 3 $\exp\left(\widetilde{O}\left(n^{1/3}q^{d/3}\right)\right)$ traces to reconstruct (n, d)-spiders $d \leq \log_{1/q}(n)$



DFS indexing

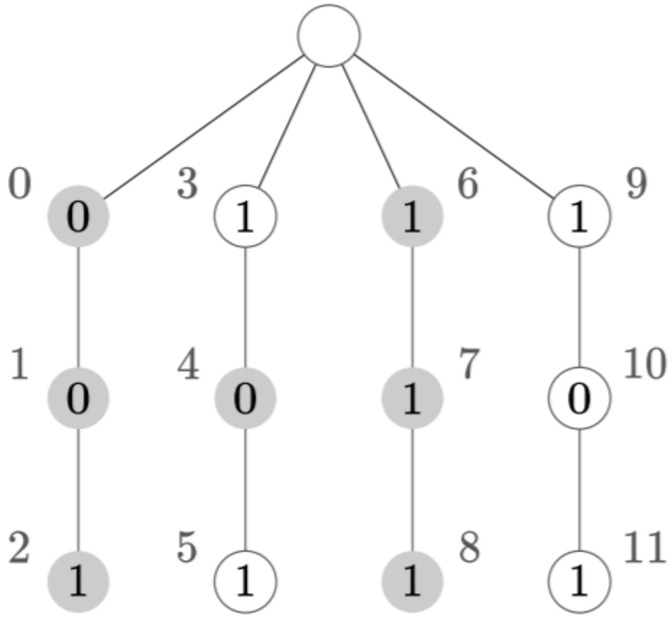
Generating function

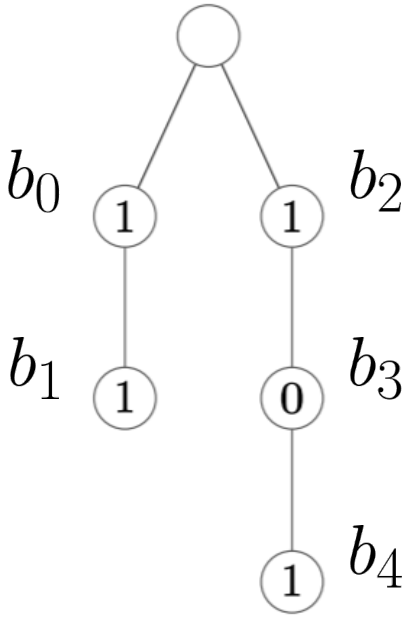
$w \in \mathbb{C}$

$$A(w) := \mathbb{E}\left(\sum_{j=0}^{n-1} b_j w^j\right)$$

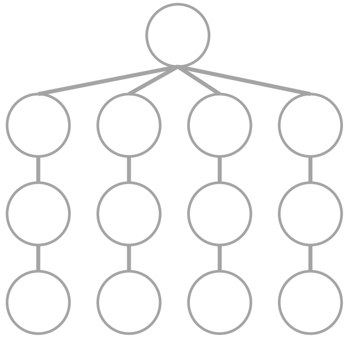
original labels $a = a_0, a_1, \ldots, a_{n-1}$

trace labels $b = b_0, b_1, \ldots, b_{n'}, 0, \ldots, 0$





Theorem 3 $\exp\left(\tilde{O}\left(n^{1/3}q^{d/3}\right)\right)$ traces to reconstruct (n,d)-spiders $d \leq \log_{1/q}(n)$



DFS indexing

Generating function

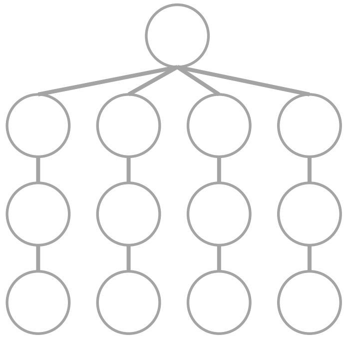
original labels $a = a_0, a_1, \ldots, a_{n-1}$

$w \in \mathbb{C}$

trace labels $b = b_0, b_1, \ldots, b_{n'}, 0, \ldots, 0$

$$A(w) := \mathbb{E}\left(\sum_{j=0}^{n-1} b_j w^j\right) = (1-q)\sum_{\ell=0}^{n-1} a_\ell (q + (1-q)w)^{\ell \pmod d} (q^d + (1-q^d)w^d)^{\lfloor \frac{\ell}{d} \rfloor}$$

Theorem 3 $\exp\left(\tilde{O}\left(n^{1/3}q^{d/3}\right)\right)$ traces to reconstruct (n,d)-spiders $d \le \log_{1/q}(n)$



DFS indexing

Generating function

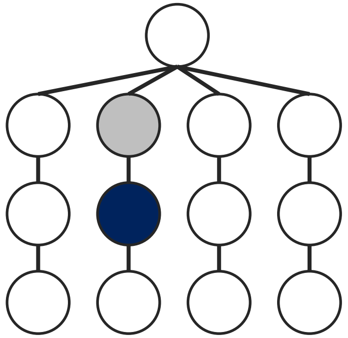
original labels $a = a_0, a_1, \ldots, a_{n-1}$

$w \in \mathbb{C}$

trace labels $b = b_0, b_1, \ldots, b_{n'}, 0, \ldots, 0$

$$A(w) := \mathbb{E}\left(\sum_{j=0}^{n-1} b_j w^j\right) = (1-q)\sum_{\ell=0}^{n-1} a_\ell (q + (1-q)w)^{\ell \pmod d}\,(q^d + (1-q^d)w^d)^{\lfloor \frac{\ell}{d} \rfloor}$$

Theorem 3 $\exp\left(\tilde{O}\left(n^{1/3}q^{d/3}\right)\right)$ traces to reconstruct (n,d)-spiders $d \le \log_{1/q}(n)$



Generating function

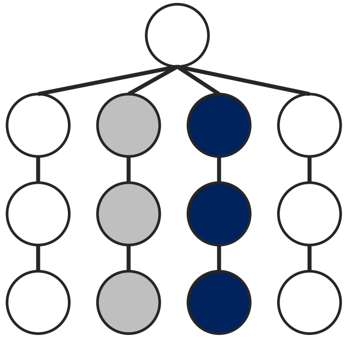
$w \in \mathbb{C}$

original labels $a = a_0, a_1, \ldots, a_{n-1}$

trace labels $b = b_0, b_1, \ldots, b_{n'}, 0, \ldots, 0$

$$A(w) := \mathbb{E}\left(\sum_{j=0}^{n-1} b_j w^j\right) = (1-q)\sum_{\ell=0}^{n-1} a_\ell (q+(1-q)w)^{\ell \pmod d} (q^d+(1-q^d)w^d)^{\lfloor \frac{\ell}{d} \rfloor}$$

Theorem 3 $\exp\left(\tilde{O}\left(n^{1/3}q^{d/3}\right)\right)$ traces to reconstruct (n,d)-spiders $d \leq \log_{1/q}(n)$



(n,d)-spider

n/d paths

depth d

DFS indexing

Generating function

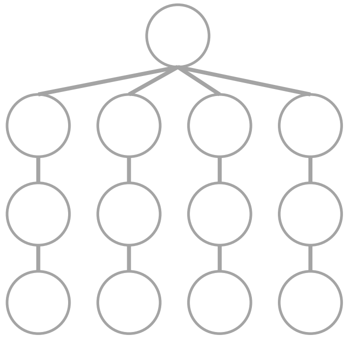
original labels $a = a_0, a_1, \ldots, a_{n-1}$

$w \in \mathbb{C}$

trace labels $b = b_0, b_1, \ldots, b_{n'}, 0, \ldots, 0$

$$A(w) := \mathbb{E}\left(\sum_{j=0}^{n-1} b_j w^j\right) = (1-q)\sum_{\ell=0}^{n-1} a_\ell (q+(1-q)w)^{\ell \pmod d} (q^d + (1-q^d)w^d)^{\lfloor \frac{\ell}{d} \rfloor}$$

Theorem 3 $\exp\left(\tilde{O}\left(n^{1/3}q^{d/3}\right)\right)$ traces to reconstruct (n,d)-spiders $d \le \log_{1/q}(n)$



(n,d)-spider

n/d paths

depth d

DFS indexing

Generating function

original labels $a = a_0, a_1, \dots, a_{n-1}$

$w \in \mathbb{C}$

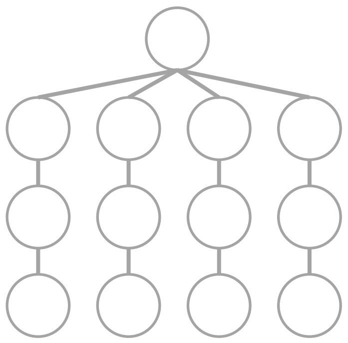
trace labels $b = b_0, b_1, \dots, b_{n'}, 0, \dots, 0$

$$A(w) := \mathbb{E}\left(\sum_{j=0}^{n-1} b_j w^j\right) = (1-q)\sum_{\ell=0}^{n-1} a_\ell (q + (1-q)w)^{\ell \pmod d} (q^d + (1-q^d)w^d)^{\lfloor \frac{\ell}{d} \rfloor}$$

X^1 vs. X^2

- $a = a^1 - a^2$
- $b = b^1 - b^2$

Theorem 3 $\exp\left(\widetilde{O}\left(n^{1/3}q^{d/3}\right)\right)$ traces to reconstruct (n,d)-spiders $d \le \log_{1/q}(n)$



(n,d)-spider

n/d paths

depth d

DFS indexing

Generating function

original labels $a = a_0, a_1, \ldots, a_{n-1}$

$w \in \mathbb{C}$

trace labels $b = b_0, b_1, \ldots, b_{n'}, 0, \ldots, 0$

$$A(w) := \mathbb{E}\left(\sum_{j=0}^{n-1} b_j w^j\right) = (1-q)\sum_{\ell=0}^{n-1} a_\ell (q + (1-q)w)^{\ell \pmod d} (q^d + (1-q^d)w^d)^{\lfloor \frac{\ell}{d} \rfloor}$$

X^1 vs. X^2

- $a = a^1 - a^2$
- $b = b^1 - b^2$

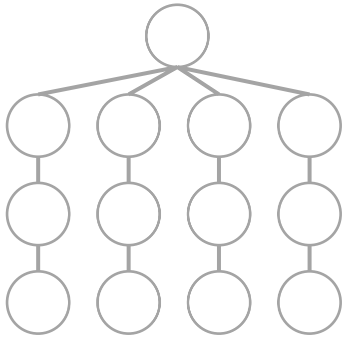
Main Lemma $\exists\, w^*$

$$|A(w^*)| \ge \exp(-L)$$

$$L = \widetilde{O}\left(n^{1/3}q^{d/3}\right)$$

$|w^*| = 1$

Theorem 3 $\exp\left(\widetilde{O}\left(n^{1/3}q^{d/3}\right)\right)$ traces to reconstruct (n,d)-spiders $d \le \log_{1/q}(n)$



(n,d)-spider

n/d paths

depth d

DFS indexing

Generating function

original labels $a = a_0, a_1, \ldots, a_{n-1}$

$w \in \mathbb{C}$

trace labels $b = b_0, b_1, \ldots, b_{n'}, 0, \ldots, 0$

$$A(w) := \mathbb{E}\left(\sum_{j=0}^{n-1} b_j w^j\right) = (1-q)\sum_{\ell=0}^{n-1} a_\ell (q+(1-q)w)^{\ell \pmod d} (q^d + (1-q^d)w^d)^{\lfloor \frac{\ell}{d} \rfloor}$$

X^1 vs. X^2

- $a = a^1 - a^2$
- $b = b^1 - b^2$

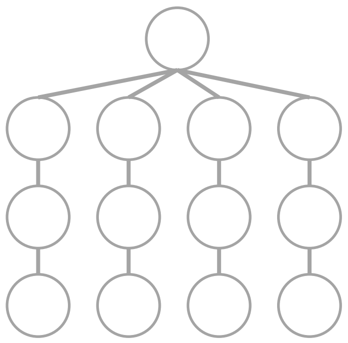
Main Lemma $\exists\, w^*$ $|w^*| = 1$

$|A(w^*)| \ge \exp(-L)$

$L = \widetilde{O}\left(n^{1/3}q^{d/3}\right)$

$\Longrightarrow$ $\exp\left(O(L)\right)$ traces suffice

Theorem 3 $\exp\left(\tilde{O}\left(n^{1/3}q^{d/3}\right)\right)$ traces to reconstruct (n,d)-spiders $d \le \log_{1/q}(n)$



(n,d)-spider

n/d paths

depth d

DFS indexing

Generating function

original labels $a = a_0, a_1, \ldots, a_{n-1}$

$w \in \mathbb{C}$

trace labels $b = b_0, b_1, \ldots, b_{n'}, 0, \ldots, 0$

$$A(w) := \mathbb{E}\left(\sum_{j=0}^{n-1} b_j w^j\right) = (1-q)\sum_{\ell=0}^{n-1} a_\ell (q + (1-q)w)^{\ell \pmod d} (q^d + (1-q^d)w^d)^{\lfloor \frac{\ell}{d} \rfloor}$$

X^1 vs. X^2

- $a = a^1 - a^2$
- $b = b^1 - b^2$

Main Lemma $\exists\, w^*$

$$|A(w^*)| \ge \exp(-L)$$

$$L = \tilde{O}\left(n^{1/3}q^{d/3}\right)$$

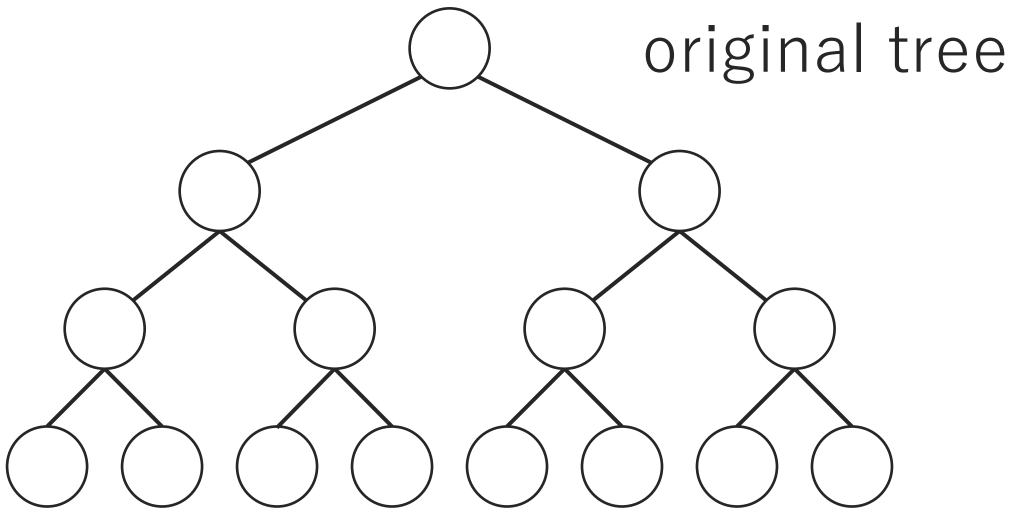
Littlewood-like polynomials

Inspired by

[Borwein and Erdélyi '97]

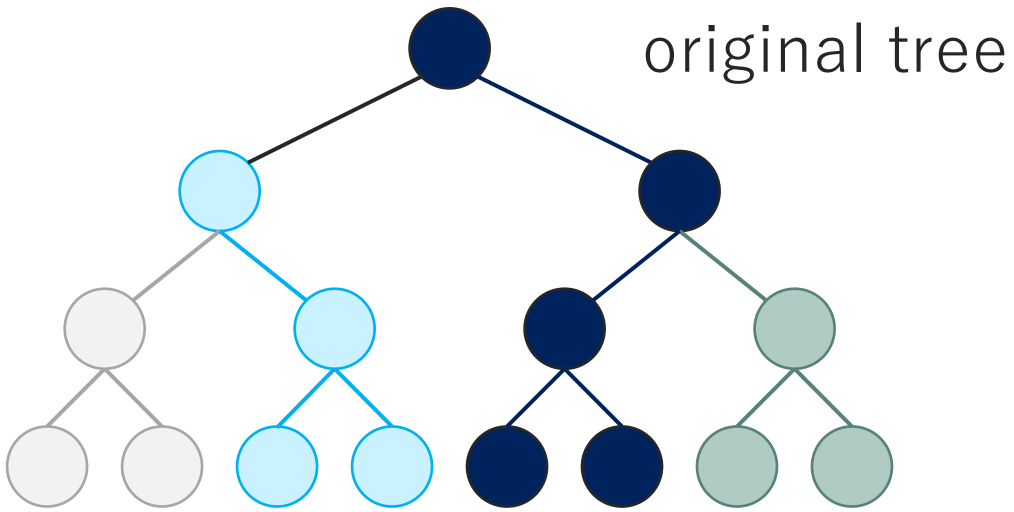
[Hartung, Holden, Peres '18]

Theorem 1 $\exp\left(k \log_k n\right)$ traces to reconstruct complete k-ary trees



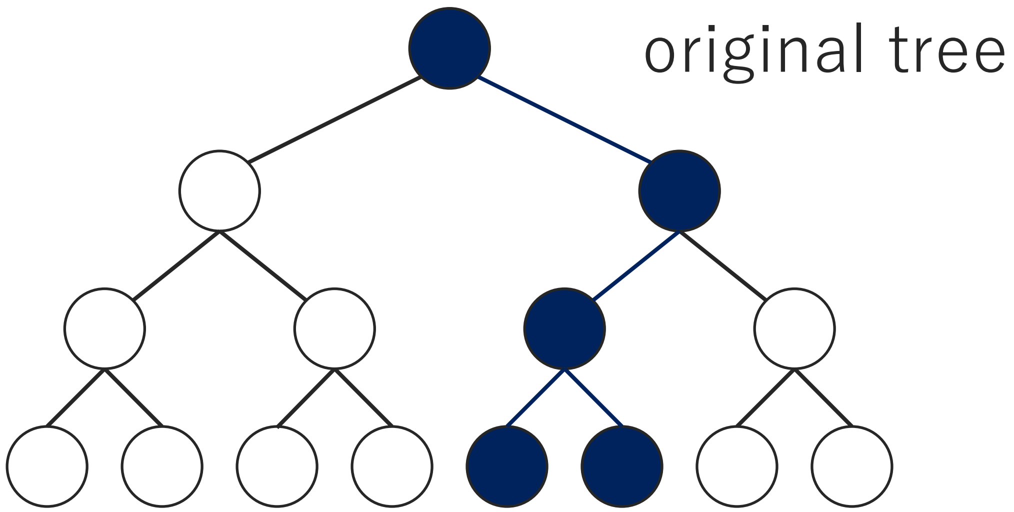
Theorem 1 $\exp(k \log_k n)$ traces to reconstruct complete k-ary trees

Proof: partition X into subtrees



Theorem 1 $\exp(k \log_k n)$ traces
to reconstruct complete k-ary trees

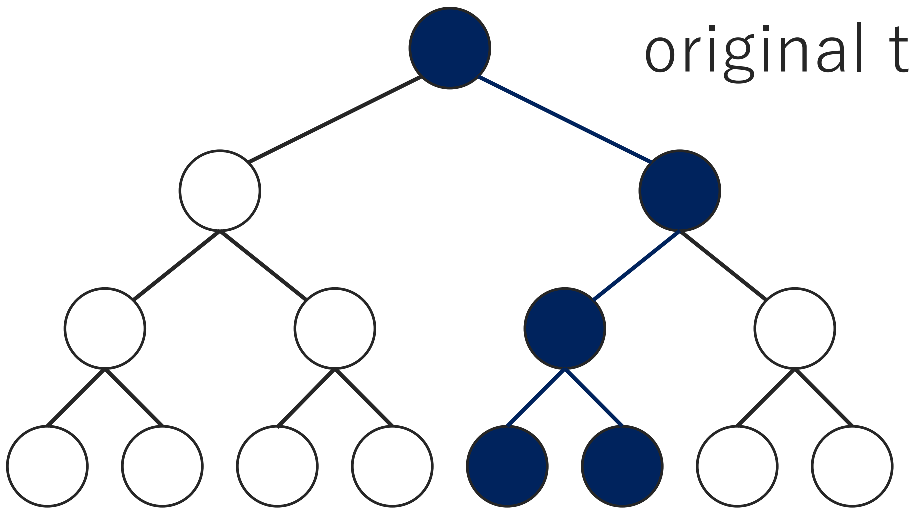
Proof: partition X into subtrees



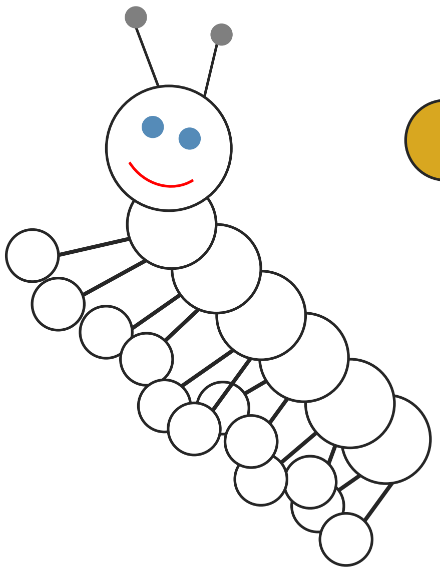
Theorem 1 $\exp\left(k \log_k n\right)$ traces
to reconstruct complete k-ary trees

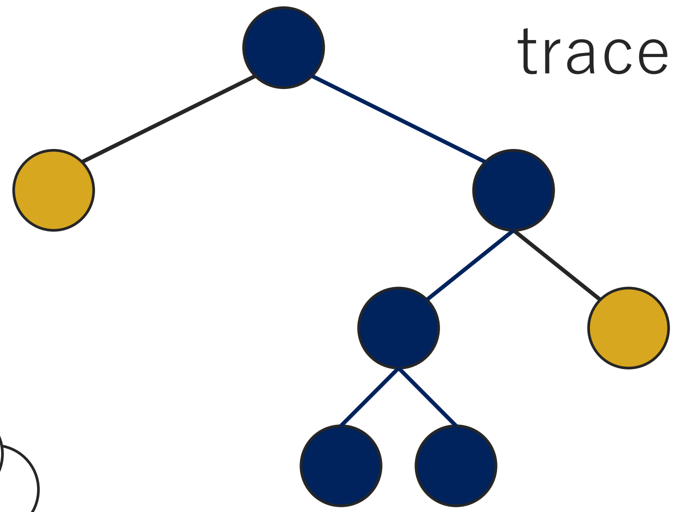
Proof: partition X into subtrees

Lemma
if trace contains caterpillar, then subtree labels are usually correct



original tree

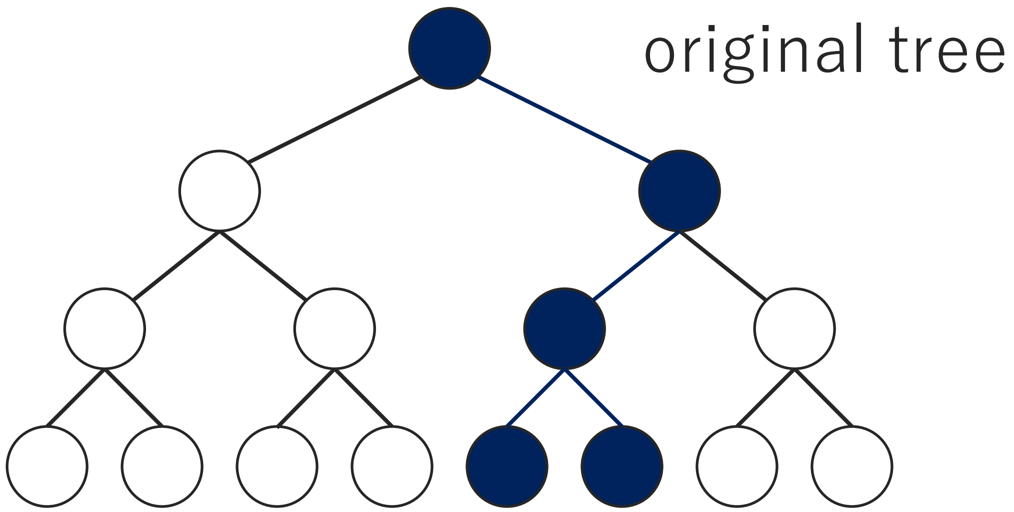




trace

Theorem 1 $\exp(k \log_k n)$ traces to reconstruct complete k-ary trees

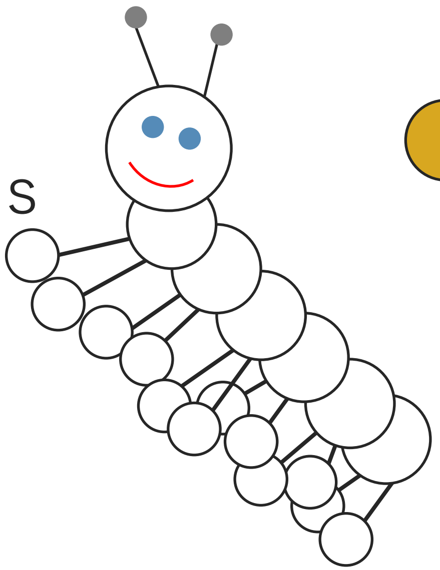
Proof: partition X into subtrees

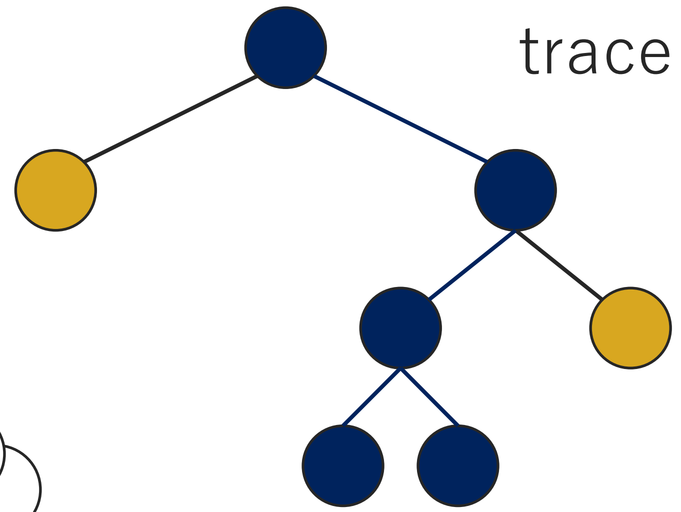


Lemma

if trace contains caterpillar, then subtree labels are usually correct

Extra nodes witness correct positions





Labels correct with prob > 2/3

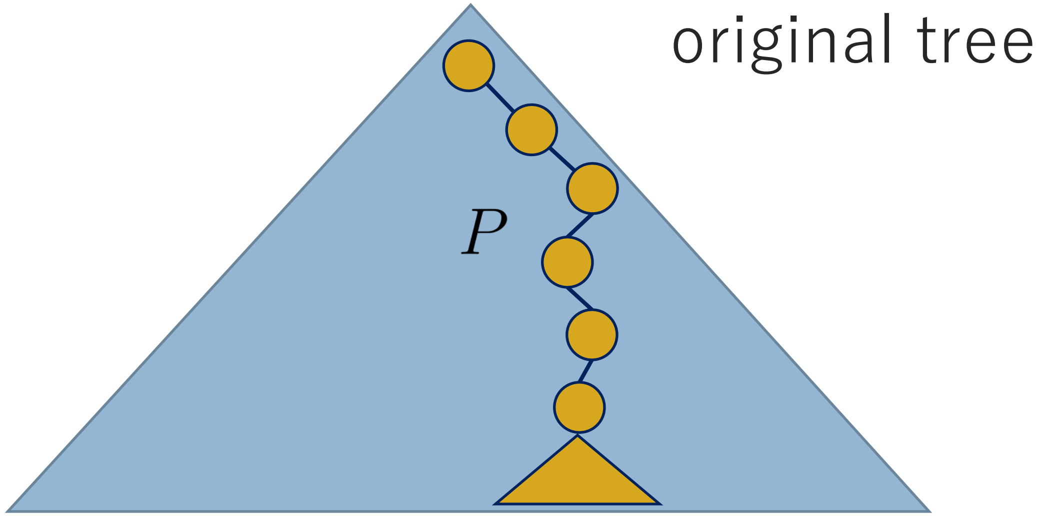
Majority vote O(log n) traces

Theorem 2 $\exp\left(k^{1/3} + \log_k n\right)$ traces to reconstruct complete k-ary trees if $k \geq c \log^2(n)$

When k is large, Theorem 1 is expensive (wait a long time to see a trace preserving a caterpillar)

Theorem 2 $\exp\left(k^{1/3} + \log_k n\right)$ traces to reconstruct complete k-ary trees if $k \geq c\log^2(n)$

Proof: cover X by subtrees

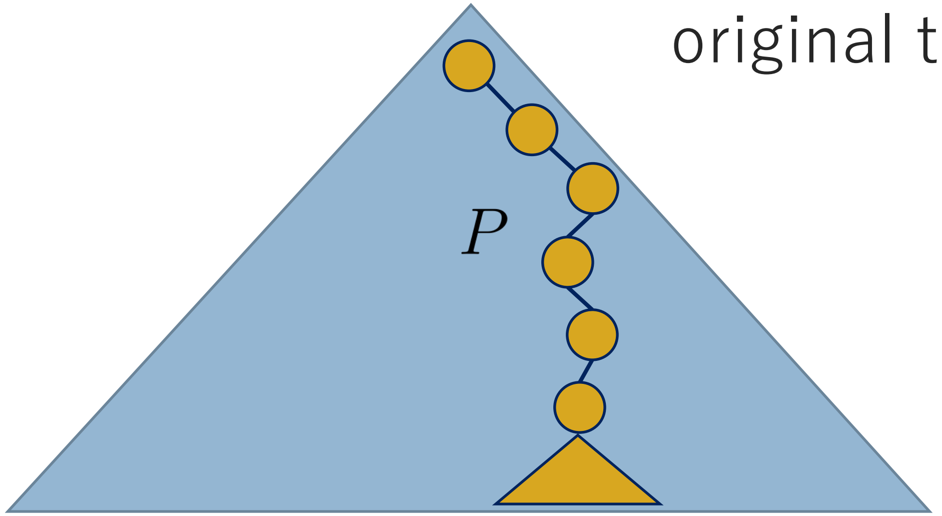


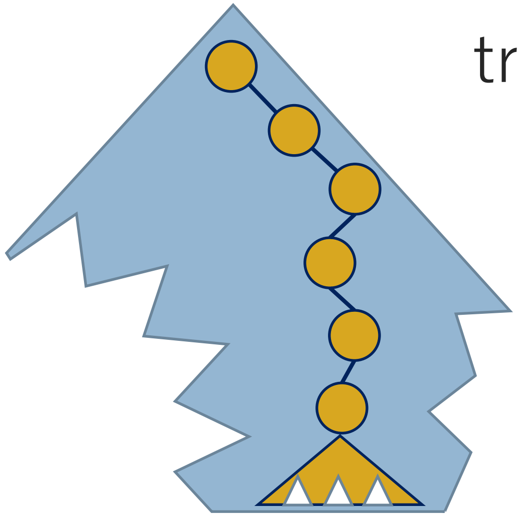
Theorem 2 $\exp\left(k^{1/3} + \log_k n\right)$ traces to reconstruct complete k-ary trees if $k \geq c \log^2(n)$

Proof: cover X by subtrees

Lemma 1

if trace contains P, we can find w.h.p. positions of all internal nodes in P

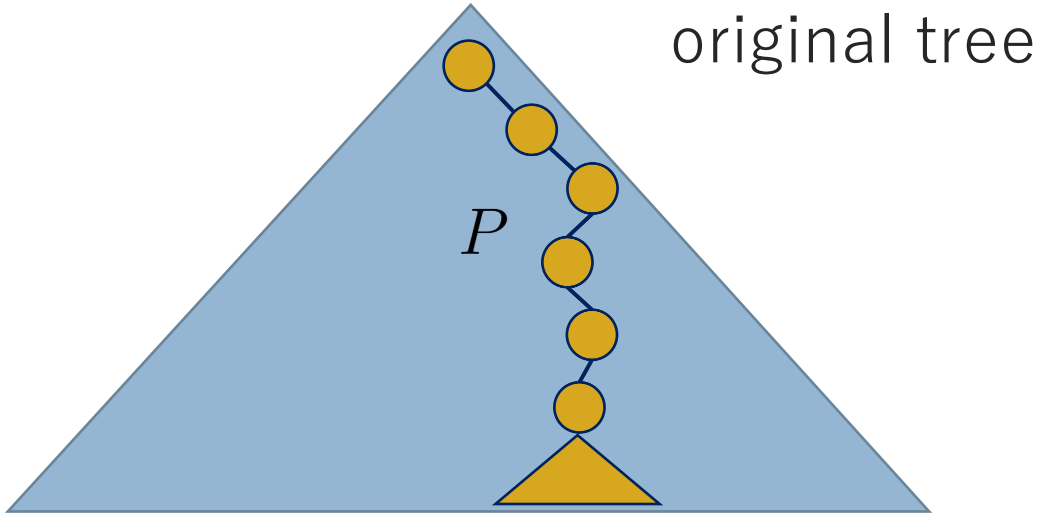


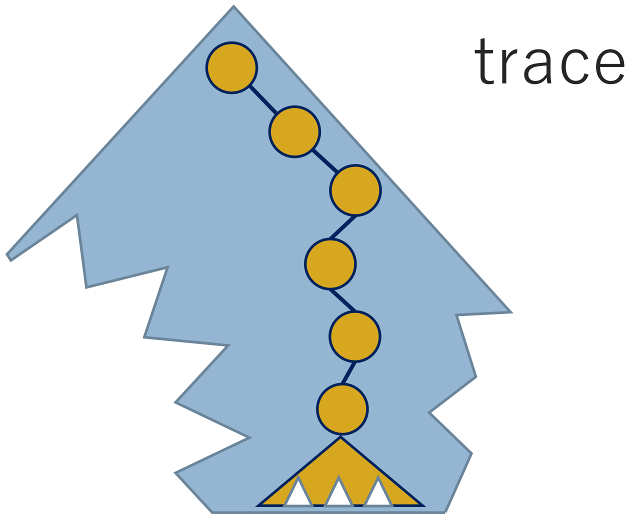


Theorem 2 $\exp\left(k^{1/3} + \log_k n\right)$ traces to reconstruct complete k-ary trees if $k \geq c \log^2(n)$

Proof: cover X by subtrees

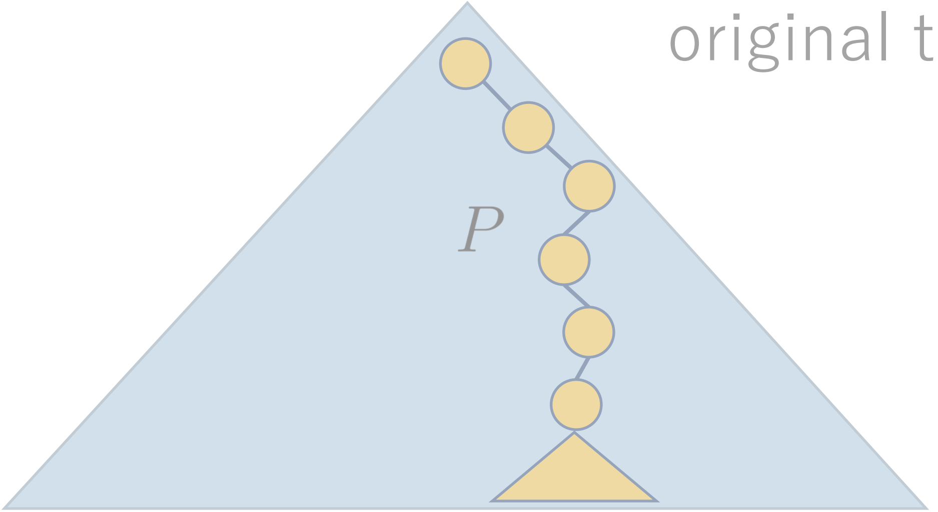
Lemma 1
if trace contains P, we can find w.h.p. positions of all internal nodes in P





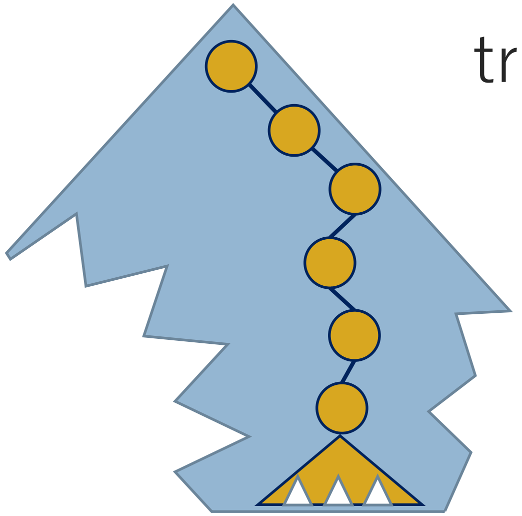
P survives with prob. $\exp(-d) = \exp(-\log_k n)$

Theorem 2 $\exp\left(k^{1/3} + \log_k n\right)$ traces to reconstruct complete k-ary trees if $k \geq c\log^2(n)$

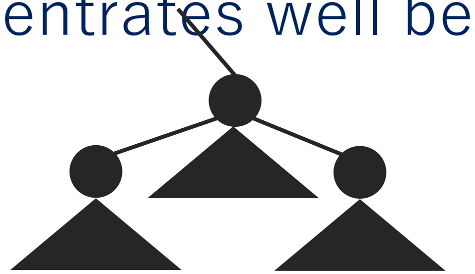


Proof: cover X by subtrees

Lemma 1
if trace contains P, we can find w.h.p. positions of all internal nodes in P



Idea: estimate # deleted nodes at every level, concentrates well because k is large!



P survives with prob. $\exp(-d) = \exp(-\log_k n)$

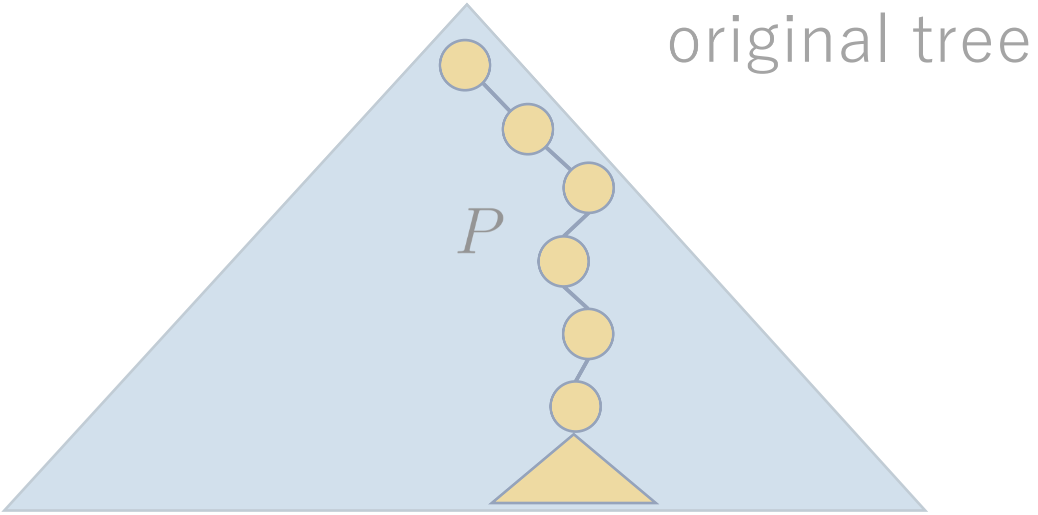
Theorem 2 $\exp\left(k^{1/3} + \log_k n\right)$ traces to reconstruct complete k-ary trees if $k \geq c\log^2(n)$

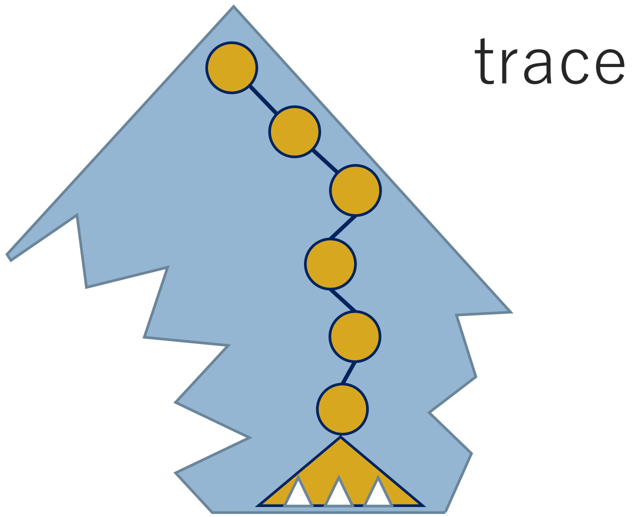
Proof: cover X by subtrees

Lemma 1
if trace contains P, we can find w.h.p. positions of all internal nodes in P

Lemma 2
Reconstruct k leaves w.h.p. using T_k traces

$$T_k \leq \exp\left(k^{1/3}\right)$$





P survives with prob. $\exp(-d) = \exp(-\log_k n)$

Trace Reconstruction Variants

- **coded TR:** encoded initial string X [Cheraghchi, Gabrys, Milenkovic, Ribeiro '19; Brakensiek, Li, Spang '19]
- **average-case:** X random → $\exp((\log n)^{1/3})$ [Peres-Zhai '17; Holden, Pemantle, Peres '18]
- **population recovery:** multiple unknown strings [Ban, Chen, Freilich, Servedio, Sinha '19]
- **matrix version:** delete random rows/cols [Krishnamurthy, Mazumdar, McGregor, Pal '19]
- **fixed # deletions:** e.g., 1, 2, 3, ... [Levenshtein '01; Gabrys, Yaakobi '18]
- **Tree TR:** reconstruct labelled trees **[Davies, Racz, Rashtchian '19]**

Deterministic Variants

- **k-deck:** reconstruct from all k-substrings $k \leq O(\sqrt{n})$ [Krasikov, Roditty '97]
- **Graph Reconstruction Conj:** all (n-1)-vertex subgraphs? [Kelly '57; Ulam '60]

Average case

Assume X is drawn uniformly at random from $\{0,1\}^n$

Know $\exp(\log^{1/3}(n))$ traces suffice

Can't improve avg. case upper bound w/out also improving worst case upper bound

Average case

Assume X is drawn uniformly at random from $\{0,1\}^n$

Know $\exp(\log^{1/3}(n))$ traces suffice

Can't improve avg. case upper bound w/out also improving worst case upper bound

Coded TR

Design codes which require less traces to reconstruct

Recently shown to be roughly equivalent to Average case TR

Average case

Assume X is drawn uniformly at random from $\{0,1\}^n$

Know $\exp(\log^{1/3}(n))$ traces suffice

Can't improve avg. case upper bound w/out also improving worst case upper bound

Coded TR

Design codes which require less traces to reconstruct

Recently shown to be roughly equivalent to Average case TR

Population recovery

Multiple unknown strings to learn

Algorithm observes traces, but doesn't know which original string they are from

When strings are random, this is just a clustering problem, then avg. case result applies

What's up with the *k*-deck?

The ***k*-deck** of a string X is the multi-set of all length k substrings
Strings with the same k-deck are ***k*-equivalent**
K-decks are unique when $k \geq Cn^{1/2}$. **Improvements here very interesting**

What's up with the *k*-deck?

The ***k*-deck** of a string X is the multi-set of all length k substrings
Strings with the same k-deck are ***k*-equivalent**
K-decks are unique when $k \geq Cn^{1/2}$. **Improvements here very interesting**

The *k*-deck & TR

Lower bounds for TR distinguish between strings with small hamming distance (4).

What's up with the *k*-deck?

The ***k*-deck** of a string X is the multi-set of all length k substrings
Strings with the same k-deck are ***k*-equivalent**
K-decks are unique when $k \geq Cn^{1/2}$. **Improvements here very interesting**

The *k*-deck & TR

Lower bounds for TR distinguish between strings with small hamming distance (4).
However...
If strings X and Y are k-equivalent then their hamming distance is ≥2k.
The k-deck of a binary string X can be determined exactly with exp(O(klog(n))) traces

What's up with the *k*-deck?

The ***k*-deck** of a string *X* is the multi-set of all length *k* substrings
Strings with the same *k*-deck are ***k*-equivalent**
K-decks are unique when $k \geq Cn^{1/2}$. **Improvements here very interesting**

The *k*-deck & TR

Lower bounds for TR distinguish between strings with small hamming distance (4).
However…
If strings X and Y are k-equivalent then their hamming distance is ≥2k.
The k-deck of a binary string X can be determined exactly with exp(O(klog(n))) traces

So, we already know that we can distinguish between strings with small Hamming distance using only traces from the deletion channel.
Are there better lower bounds using strings whose hamming distance is say, log(n).

Other Open Questions

- Families of graphs needing only $\text{polylog}(n)$ traces?
- **Approximate TR let εn bits of string be incorrect.**
- More practical deletion models for string TR (burst insertions/ deletions)
- Other useful structure for trace reconstruction?
- Applications for Tree TR to computational biology or sensors or … ?

Thanks!

Sami Davies
www.samidavies.com
daviess@uw.edu