Faux-determinism

Shalev Ben-David
University of Waterloo
Institute for Quantum computing

M. H. Ebtehaj
University of Waterloo
Institute for Quantum computing

CCC 2026
Lisbon, Portugal

Overview

- Define faux deterministic algorithms in decision tree model which
- roughly characterizes “computationally indistinguishable” from deterministic

Introduction Some preliminary definitions

Motivation Search problems and why randomness could be different for them

Definition Defining faux deterministic model

Construction Building the construction for FIND1 problem

Introduction

Some preliminary definitions

Decision tree model

Decision tree model

Goal: compute $f : \{0,1\}^* \to \{0,1,\perp\}$

Decision tree model

Goal: compute $f : \{0,1\}^* \to \{0,1,\perp\}$

Decision tree T:

Decision tree model

Goal: compute $f : \{0,1\}^* \to \{0,1,\perp\}$

Decision tree T:

- Given an input string $x \in \{0,1\}^n$ to f

$x =$

1	0	1	...	x_i	...	0	0	1

Decision tree model

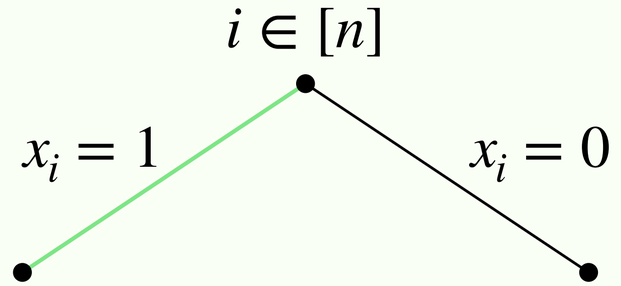
Goal: compute $f : \{0,1\}^* \to \{0,1,\perp\}$

Decision tree T:

- Given an input string $x \in \{0,1\}^n$ to f
- Query indices $i \in [n]$ adaptively

$x =$

1	0	1	...	x_i	...	0	0	1



Decision tree model

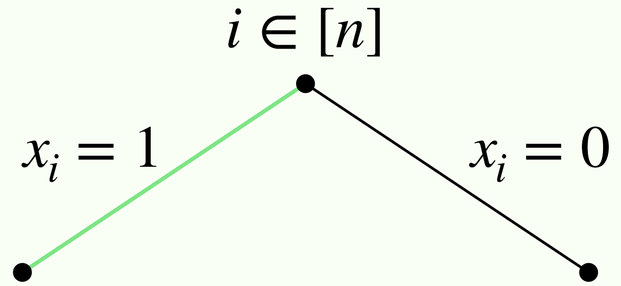
Goal: compute $f : \{0,1\}^* \to \{0,1,\perp\}$

Decision tree T:

- Given an input string $x \in \{0,1\}^n$ to f
- Query indices $i \in [n]$ adaptively
- After for $d = \text{polylog}(n)$ many queries

$x =$

1	0	1	...	x_i	...	0	0	1

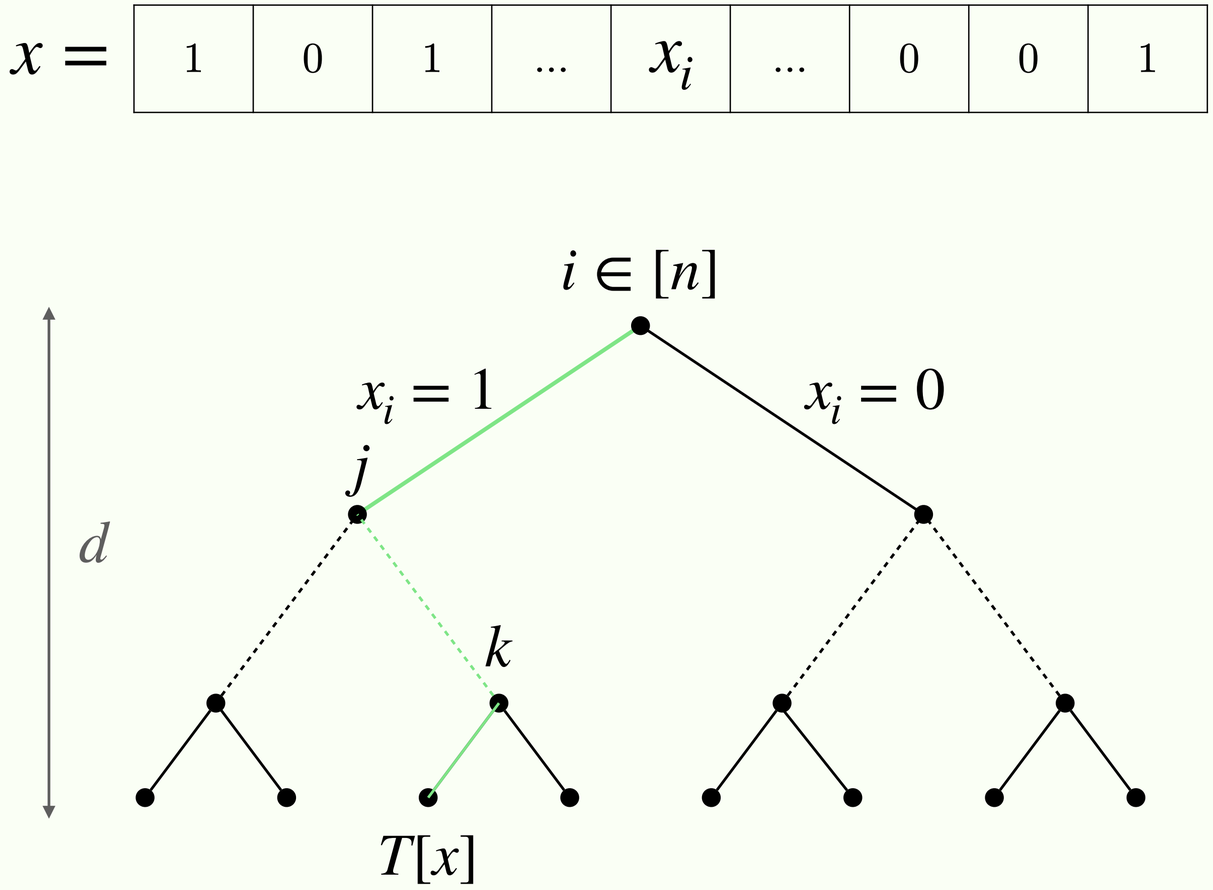


Decision tree model

Goal: compute $f : \{0,1\}^* \to \{0,1,\perp\}$

Decision tree T:

- Given an input string $x \in \{0,1\}^n$ to f
- Query indices $i \in [n]$ adaptively
- After for $d = \text{polylog}(n)$ many queries
- output $T[x] = \text{f(x)}$



Randomized decision tree

Randomized decision tree

Goal: compute $f : \{0,1\}^* \to \{0,1,\perp\}$

Randomized decision tree

Goal: compute $f : \{0,1\}^* \to \{0,1,\perp\}$

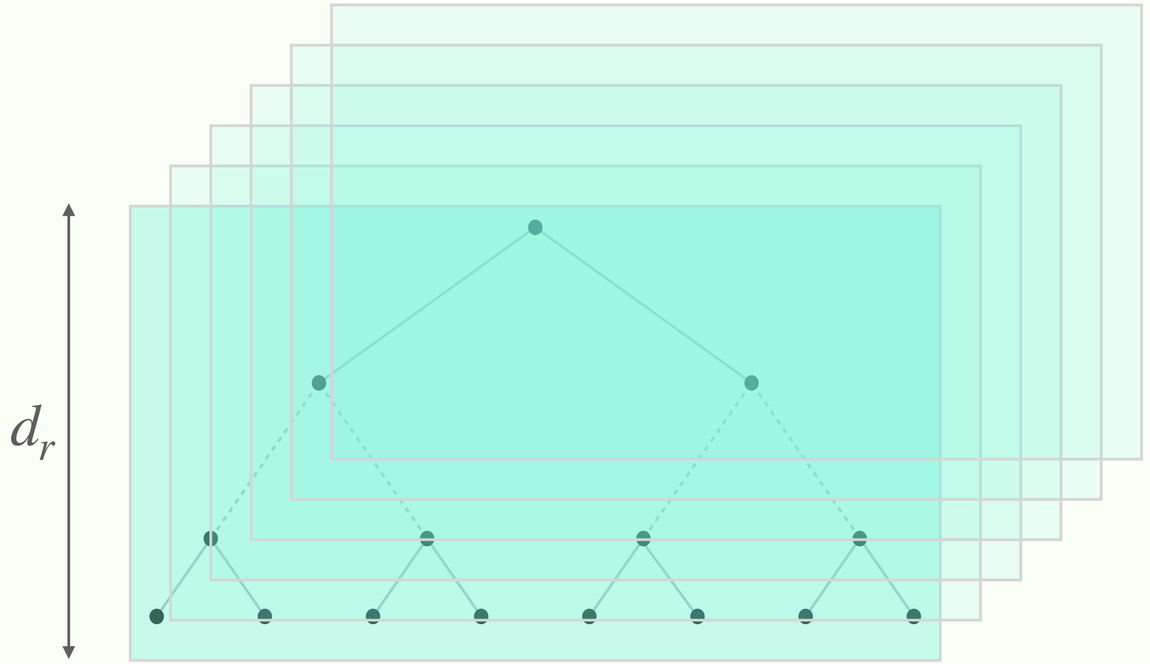
Randomized decision tree

Randomized decision tree

Goal: compute $f : \{0,1\}^* \to \{0,1,\perp\}$

Randomized decision tree

- Distribution $\mathcal{T} = \{T_r : r \sim R\}$ where $\max_r d_r = \text{polylog}(n)$



Randomized decision tree

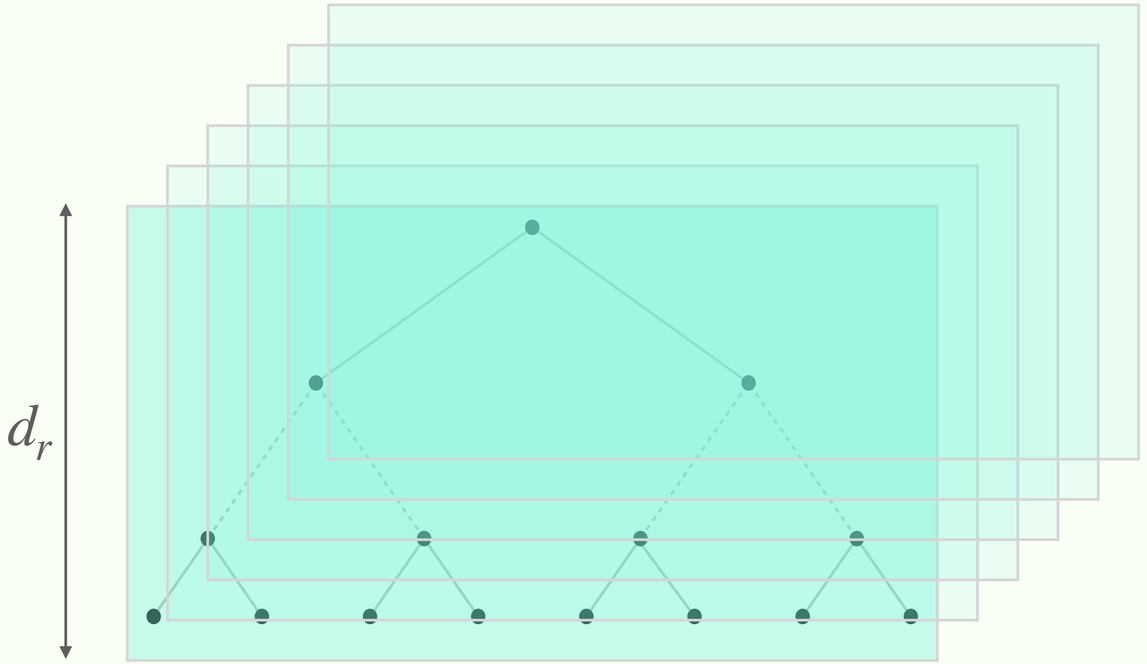
Goal: compute $f : \{0,1\}^* \to \{0,1,\perp\}$

Randomized decision tree

- Distribution $\mathcal{T} = \{T_r : r \sim R\}$ where $\max_r d_r = \text{polylog}(n)$
- Given an input string $x \in \{0,1\}^n$

$x =$

1	0	1	...	x_i	...	0	0	1



Randomized decision tree

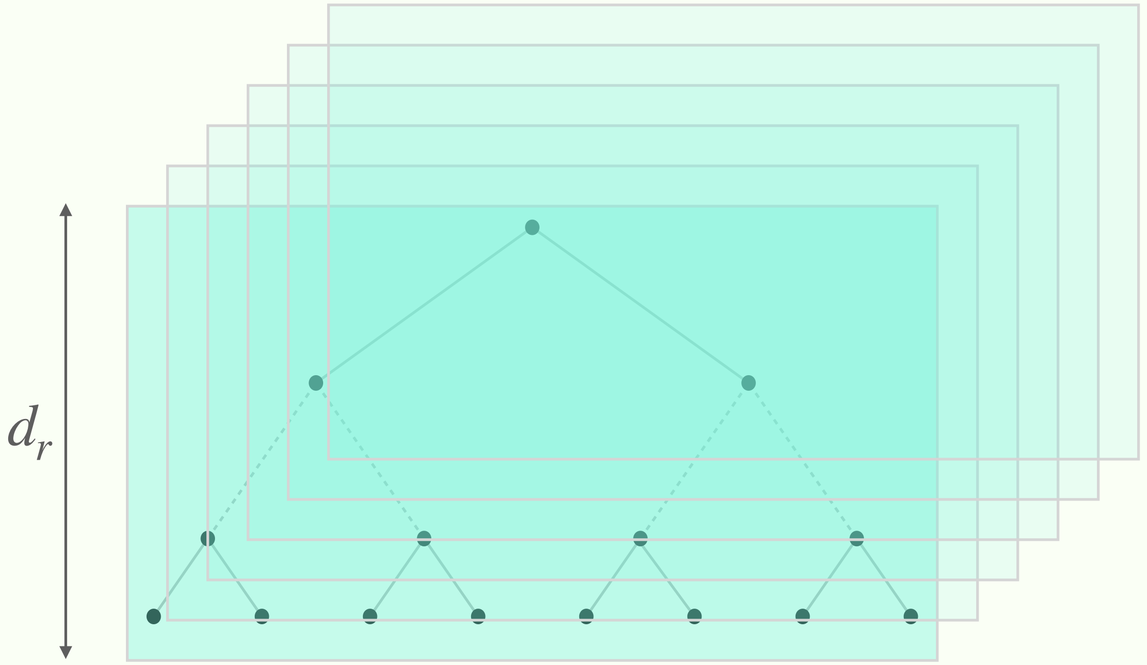
Goal: compute $f : \{0,1\}^* \to \{0,1,\perp\}$

Randomized decision tree

- Distribution $\mathcal{T} = \{T_r : r \sim R\}$ where $\max_r d_r = \text{polylog}(n)$
- Given an input string $x \in \{0,1\}^n$
- Sample a random seed $r \sim \text{Unif}$

$x =$

1	0	1	...	x_i	...	0	0	1



Randomized decision tree

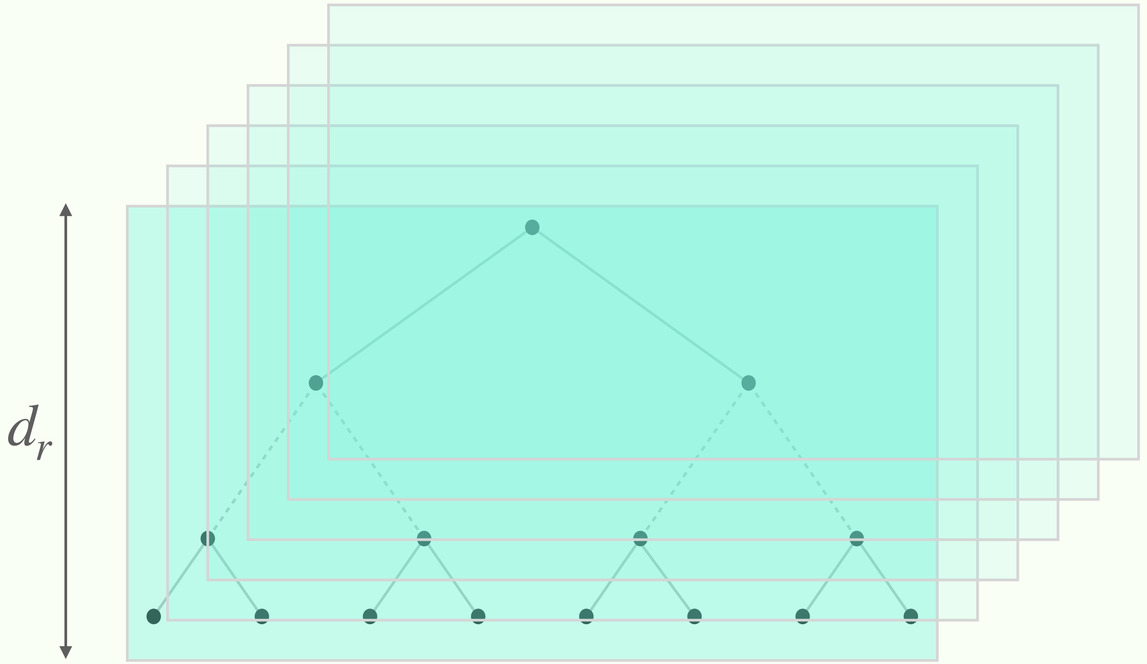
Goal: compute $f : \{0,1\}^* \to \{0,1,\perp\}$

Randomized decision tree

- Distribution $\mathcal{T} = \{T_r : r \sim R\}$ where $\max_r d_r = \text{polylog}(n)$
- Given an input string $x \in \{0,1\}^n$
- Sample a random seed $r \sim \text{Unif}$
- Output $T_r[x]$

$x =$

1	0	1	...	x_i	...	0	0	1



Randomized decision tree

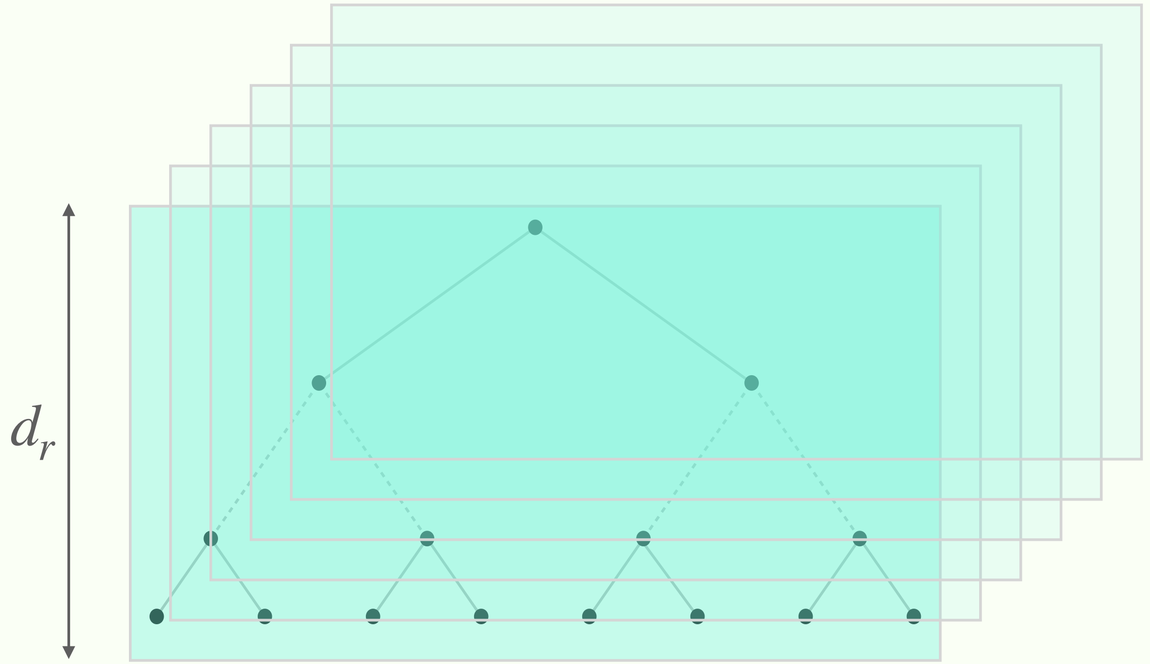
Goal: compute $f : \{0,1\}^* \to \{0,1,\perp\}$

Randomized decision tree

- Distribution $\mathcal{T} = \{T_r : r \sim R\}$ where $\max_r d_r = \text{polylog}(n)$
- Given an input string $x \in \{0,1\}^n$
- Sample a random seed $r \sim \text{Unif}$
- Output $T_r[x]$
- Correctness: $\Pr_r[T_r[x] = f(x)] \geq 2/3$

$x =$

1	0	1	...	x_i	...	0	0	1



Randomized decision tree

Goal: compute $f : \{0,1\}^* \to \{0,1,\perp\}$

Randomized decision tree

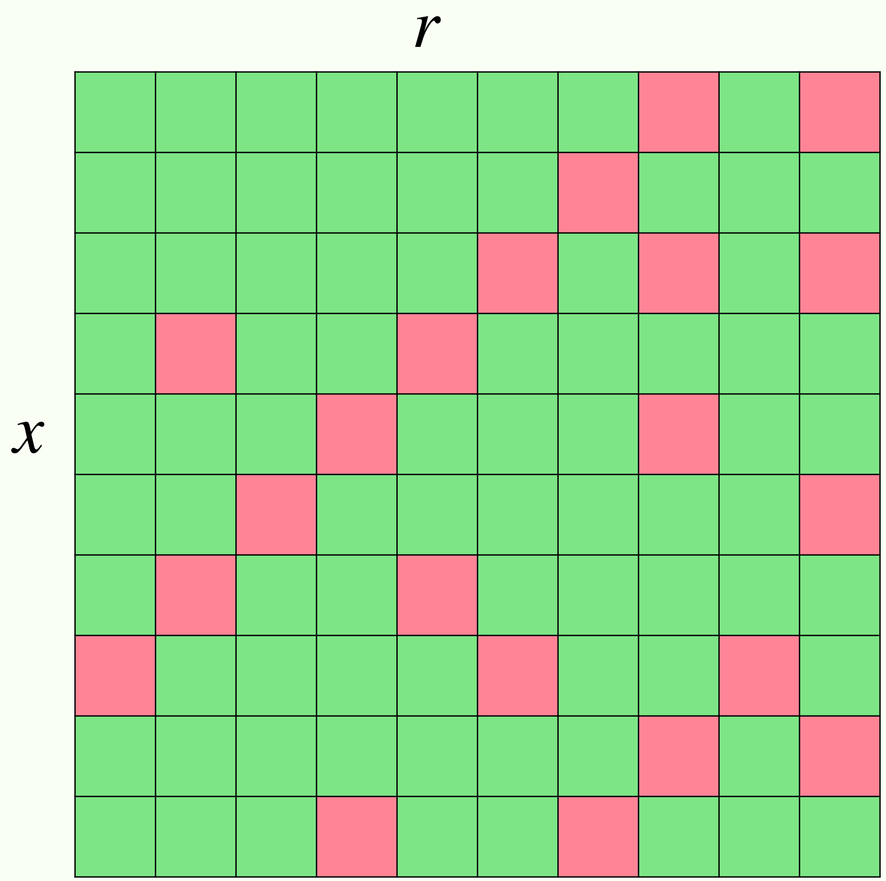
- Distribution $\mathcal{T} = \{T_r : r \sim R\}$ where $\max_r d_r = \text{polylog}(n)$
- Given an input string $x \in \{0,1\}^n$
- Sample a random seed $r \sim \text{Unif}$
- Output $T_r[x]$
- Correctness: $\Pr_r[T_r[x] = f(x)] \geq 2/3$

Randomized decision tree

Goal: compute $f : \{0,1\}^* \to \{0,1,\perp\}$

Randomized decision tree

- Distribution $\mathcal{T} = \{T_r : r \sim R\}$ where $\max_r d_r = \text{polylog}(n)$
- Given an input string $x \in \{0,1\}^n$
- Sample a random seed $r \sim \text{Unif}$
- Output $T_r[x]$
- Correctness: $\Pr_r[T_r[x] = f(x)] \geq 2/3$

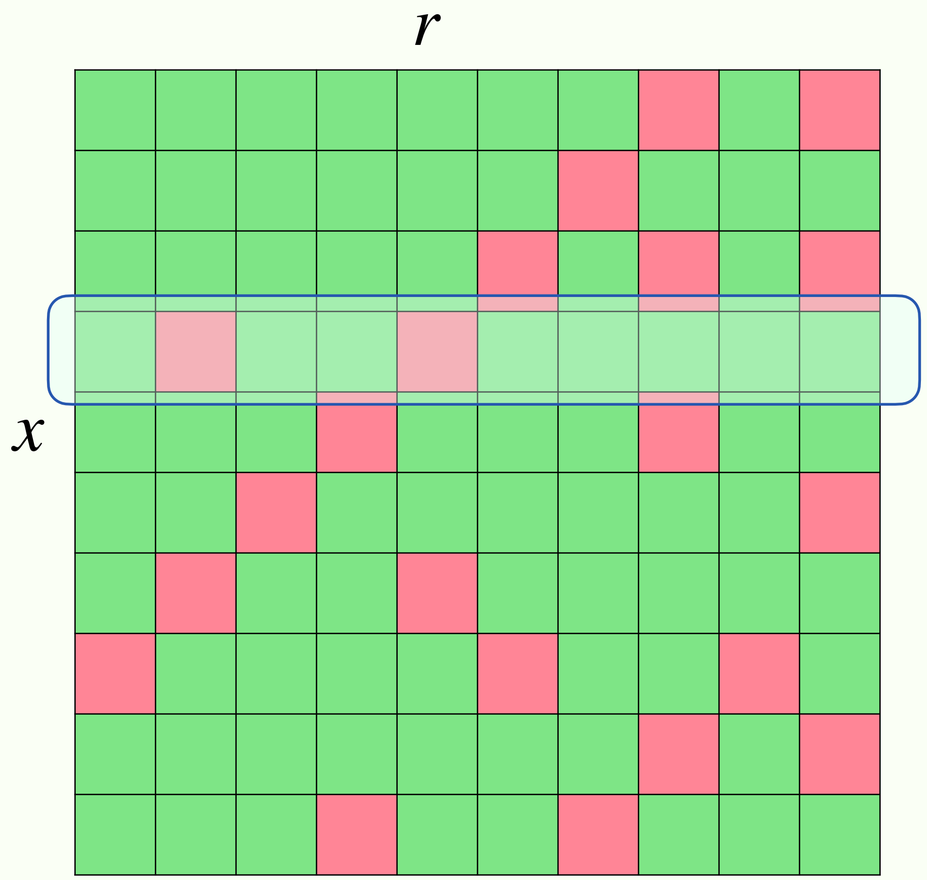


Randomized decision tree

Goal: compute $f : \{0,1\}^* \to \{0,1,\perp\}$

Randomized decision tree

- Distribution $\mathcal{T} = \{T_r : r \sim R\}$ where $\max_r d_r = \text{polylog}(n)$
- Given an input string $x \in \{0,1\}^n$
- Sample a random seed $r \sim \text{Unif}$
- Output $T_r[x]$
- Correctness: $\Pr_r[T_r[x] = f(x)] \geq 2/3$



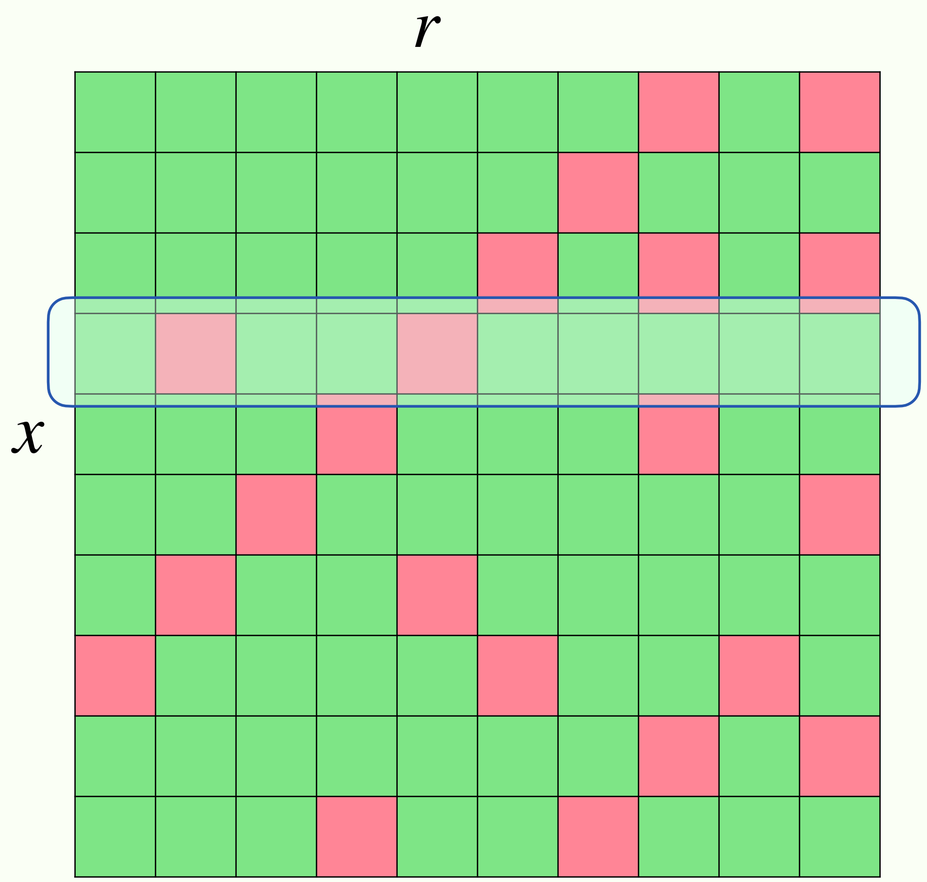
Randomized decision tree

Goal: compute $f : \{0,1\}^* \to \{0,1,\perp\}$

Randomized decision tree

- Distribution $\mathcal{T} = \{T_r : r \sim R\}$ where $\max_r d_r = \text{polylog}(n)$
- Given an input string $x \in \{0,1\}^n$
- Sample a random seed $r \sim \text{Unif}$
- Output $T_r[x]$
- Correctness: $\Pr_r[T_r[x] = f(x)] \geq 2/3$

- Error amplification to $\epsilon = 1/\text{quasipoly}(n)$

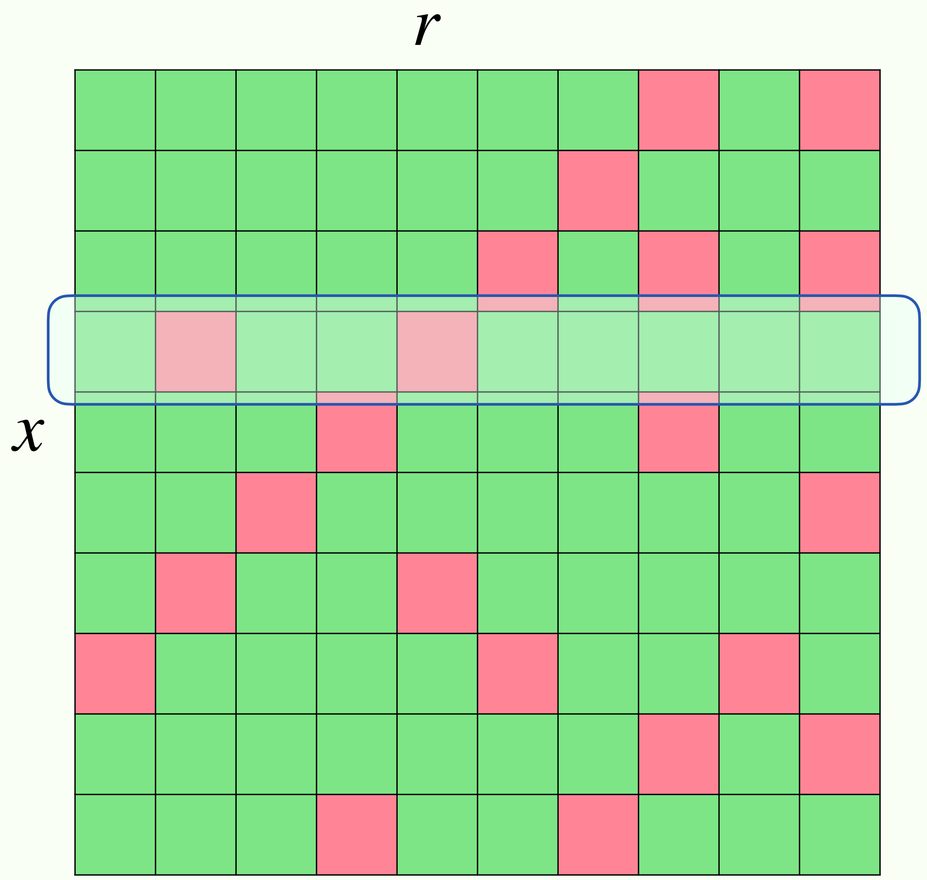


Randomized decision tree

Goal: compute $f : \{0,1\}^* \to \{0,1,\perp\}$

Randomized decision tree

- Distribution $\mathcal{T} = \{T_r : r \sim R\}$ where $\max_r d_r = \text{polylog}(n)$
- Given an input string $x \in \{0,1\}^n$
- Sample a random seed $r \sim \text{Unif}$
- Output $T_r[x]$
- Correctness: $\Pr_r[T_r[x] = f(x)] \geq 2/3$



- Error amplification to $\epsilon = 1/\text{quasipoly}(n)$
- indistinguishable from deterministic algorithm

Computational distinguishability from deterministic algorithms

Computational distinguishability from deterministic algorithms

Algorithm
$\mathscr{T} = \{T_r\}$

r

x

1	1	1	1	1	0	1	0	1	
0	0	1	0	0	0	0	0	0	1

Computational distinguishability from deterministic algorithms

Adversary

Algorithm
$\mathscr{T} = \{T_r\}$

r

x

1	1	1	1	1	0	1	0	1	
0	0	1	0	0	0	0	0	0	1

Computational distinguishability from deterministic algorithms

Adversary

Algorithm
$\mathscr{T} = \{T_r\}$

Goal: distinguish $\mathscr{T}$ from a deterministic algorithm

r

x

1	1	1	1	1	0	1	0	1	
0	0	1	0	0	0	0	0	0	1

Computational distinguishability from deterministic algorithms

Adversary

Algorithm
$\mathscr{T} = \{T_r\}$

Goal: distinguish $\mathscr{T}$ from a deterministic algorithm
Input: Black-box access to query inputs $x \in \{0,1\}^n \equiv [2^n]$

r

x

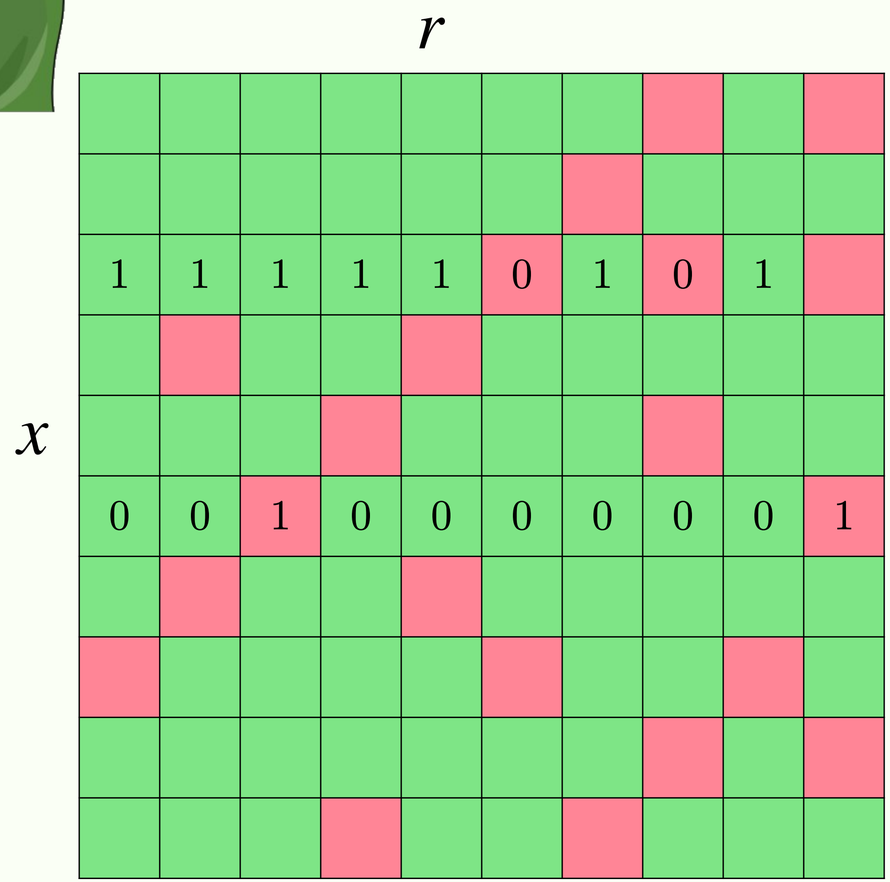
1	1	1	1	1	0	1	0	1	
0	0	1	0	0	0	0	0	0	1

Computational distinguishability from deterministic algorithms

Adversary

Algorithm
$\mathscr{T} = \{T_r\}$

Goal: distinguish $\mathscr{T}$ from a deterministic algorithm
Input: Black-box access to query inputs $x \in \{0,1\}^n \equiv [2^n]$
Budget: polylog(2^n) = poly(n) many queries



Computational distinguishability from deterministic algorithms

Adversary

Algorithm
$\mathcal{T} = \{T_r\}$

Goal: distinguish $\mathcal{T}$ from a deterministic algorithm
Input: Black-box access to query inputs $x \in \{0,1\}^n \equiv [2^n]$
Budget: polylog(2^n) = poly(n) many queries

What is $f(x_1)$?

r

x

1	1	1	1	1	0	1	0	1	
0	0	1	0	0	0	0	0	0	1

Computational distinguishability from deterministic algorithms

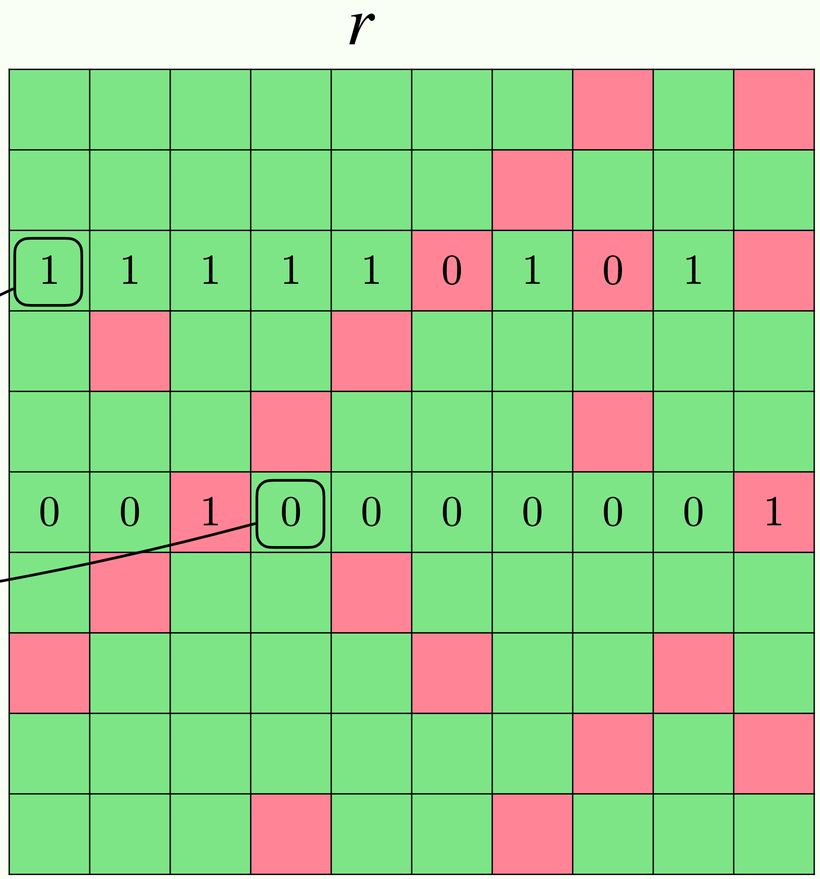
Adversary

Algorithm
$\mathcal{T} = \{T_r\}$

Goal: distinguish $\mathcal{T}$ from a deterministic algorithm
Input: Black-box access to query inputs $x \in \{0,1\}^n \equiv [2^n]$
Budget: polylog(2^n) = poly(n) many queries

What is $f(x_1)$?

What is $f(x_2)$?



Computational distinguishability from deterministic algorithms

Adversary

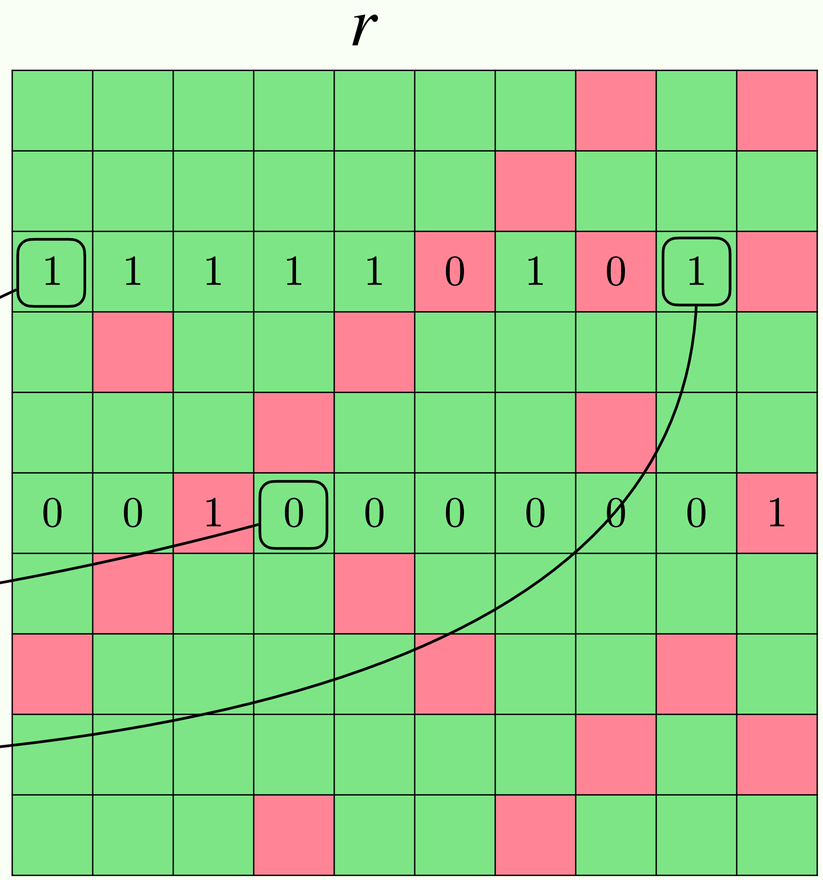
Algorithm
$\mathcal{T} = \{T_r\}$

Goal: distinguish $\mathcal{T}$ from a deterministic algorithm
Input: Black-box access to query inputs $x \in \{0,1\}^n \equiv [2^n]$
Budget: polylog(2^n) = poly(n) many queries

What is $f(x_1)$?

What is $f(x_2)$?

What is $f(x_1)$?

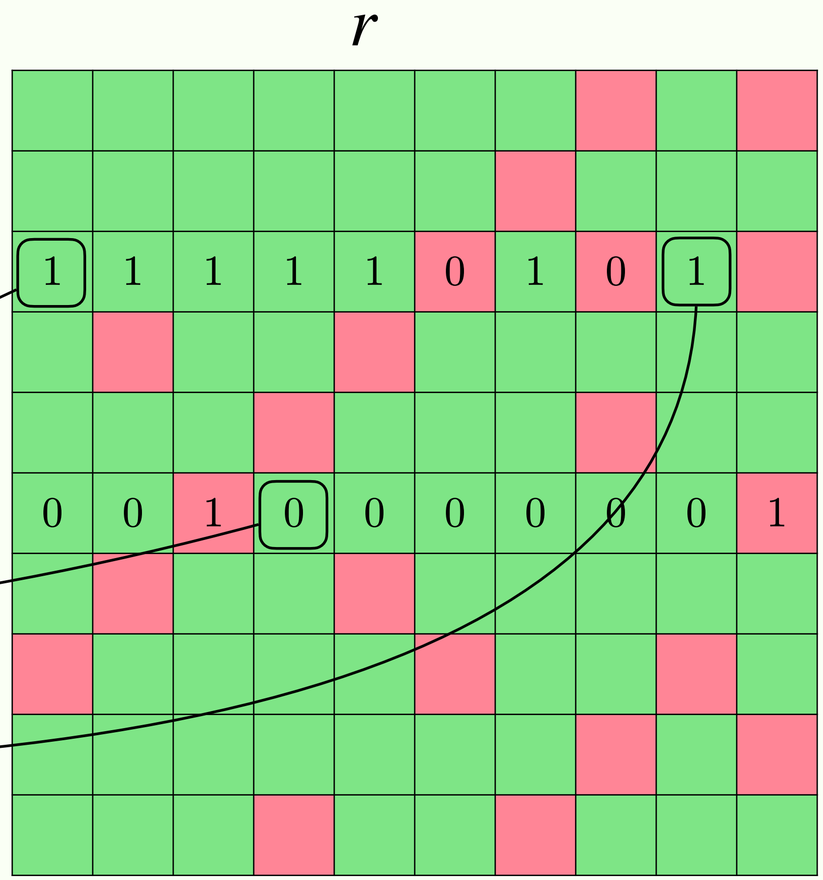


Computational distinguishability from deterministic algorithms

Adversary

Algorithm
$\mathcal{T} = \{T_r\}$

Goal: distinguish $\mathcal{T}$ from a deterministic algorithm
Input: Black-box access to query inputs $x \in \{0,1\}^n \equiv [2^n]$
Budget: polylog(2^n) = poly(n) many queries



What is $f(x_1)$?

What is $f(x_2)$?

What is $f(x_1)$?

Success: $\text{poly}(n) \times \dfrac{1}{\text{quasipoly}(n)} \ll 1$

Motivation

Search problems and why randomness could be different for them

Algorithms for search problems

Algorithms for search problems

Goal: compute **search** problem $S \subset \{0,1\}^* \times \Gamma$ $\qquad (y \in S(x) \iff (x,y) \in S)$

Algorithms for search problems

Goal: compute **search** problem $S \subset \{0,1\}^* \times \Gamma$ $\quad (y \in S(x) \iff (x,y) \in S)$

Randomized tree $\mathcal{T} = \{T_r : r \sim R\}$:

Algorithms for search problems

Goal: compute **search** problem $S \subset \{0,1\}^* \times \Gamma$ $\qquad (y \in S(x) \iff (x,y) \in S)$

Randomized tree $\mathcal{T} = \{T_r : r \sim R\}$:

- input $x \in \{0,1\}^n$ and seed r

Algorithms for search problems

Goal: compute **search** problem $S \subset \{0,1\}^* \times \Gamma$

$(y \in S(x) \iff (x, y) \in S)$

Randomized tree $\mathcal{T} = \{T_r : r \sim R\}$:

- input $x \in \{0,1\}^n$ and seed r
- Correctness: $\Pr_r[T_r[x] \in S(x)] \geq 2/3$

Algorithms for search problems

Goal: compute **search** problem $S \subset \{0,1\}^* \times \Gamma$ $\qquad (y \in S(x) \iff (x,y) \in S)$

Randomized tree $\mathcal{T} = \{T_r : r \sim R\}$:

- input $x \in \{0,1\}^n$ and seed r
- Correctness: $\Pr_r[T_r[x] \in S(x)] \geq 2/3$
- $\mathsf{FBPP}^{\mathsf{dt}} = \{S : \exists$ efficient $\mathcal{T}$ computing $S\}$

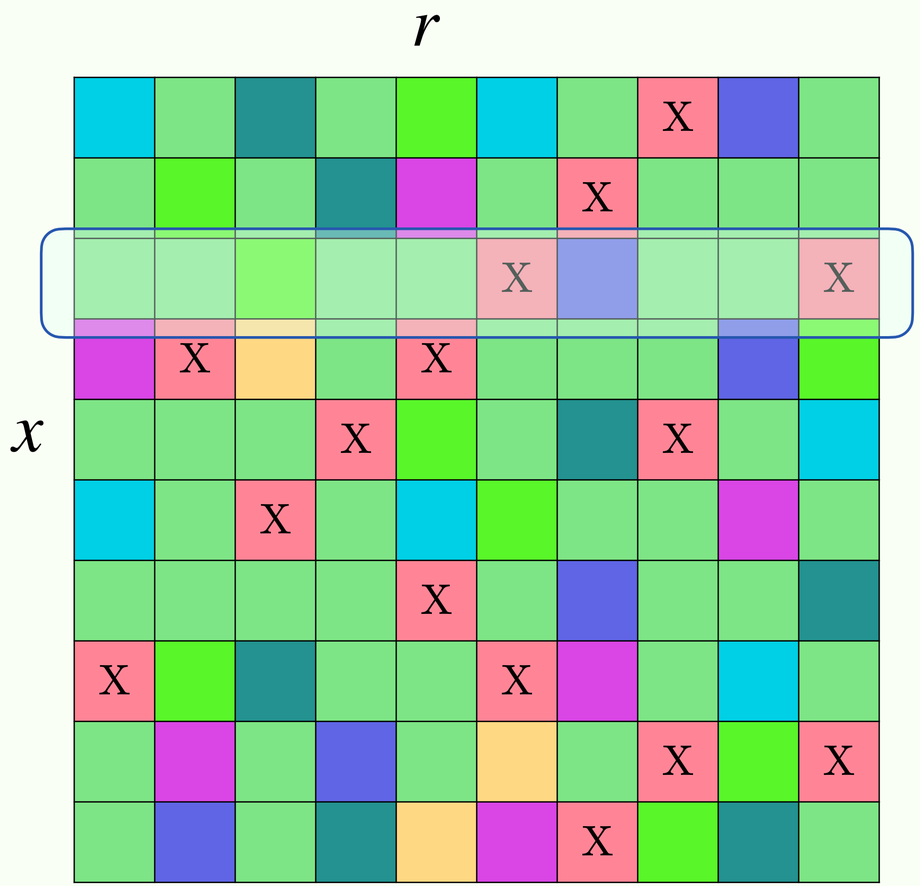
Algorithms for search problems

Goal: compute **search** problem $S \subset \{0,1\}^* \times \Gamma$

Randomized tree $\mathcal{T} = \{T_r : r \sim R\}$:

- input $x \in \{0,1\}^n$ and seed r
- Correctness: $\Pr_r[T_r[x] \in S(x)] \geq 2/3$
- $\mathsf{FBPP}^{\mathsf{dt}} = \{S : \exists \text{ efficient } \mathcal{T} \text{ computing } S\}$

$(y \in S(x) \iff (x, y) \in S)$



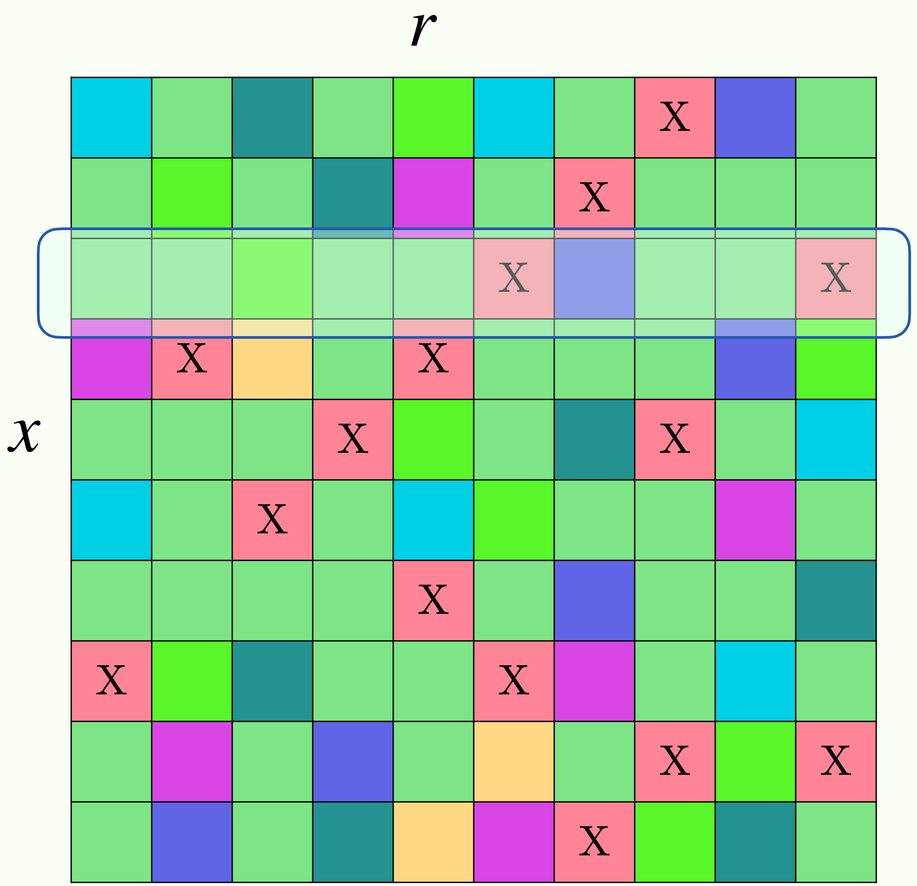
Algorithms for search problems

Goal: compute **search** problem $S \subset \{0,1\}^* \times \Gamma$

Randomized tree $\mathcal{T} = \{T_r : r \sim R\}$:

- input $x \in \{0,1\}^n$ and seed r
- Correctness: $\Pr_r[T_r[x] \in S(x)] \geq 2/3$
- $\mathsf{FBPP}^{\mathsf{dt}} = \{S : \exists \text{ efficient } \mathcal{T} \text{ computing } S\}$
- ~~Error amplification~~

$(y \in S(x) \iff (x, y) \in S)$



Algorithms for search problems

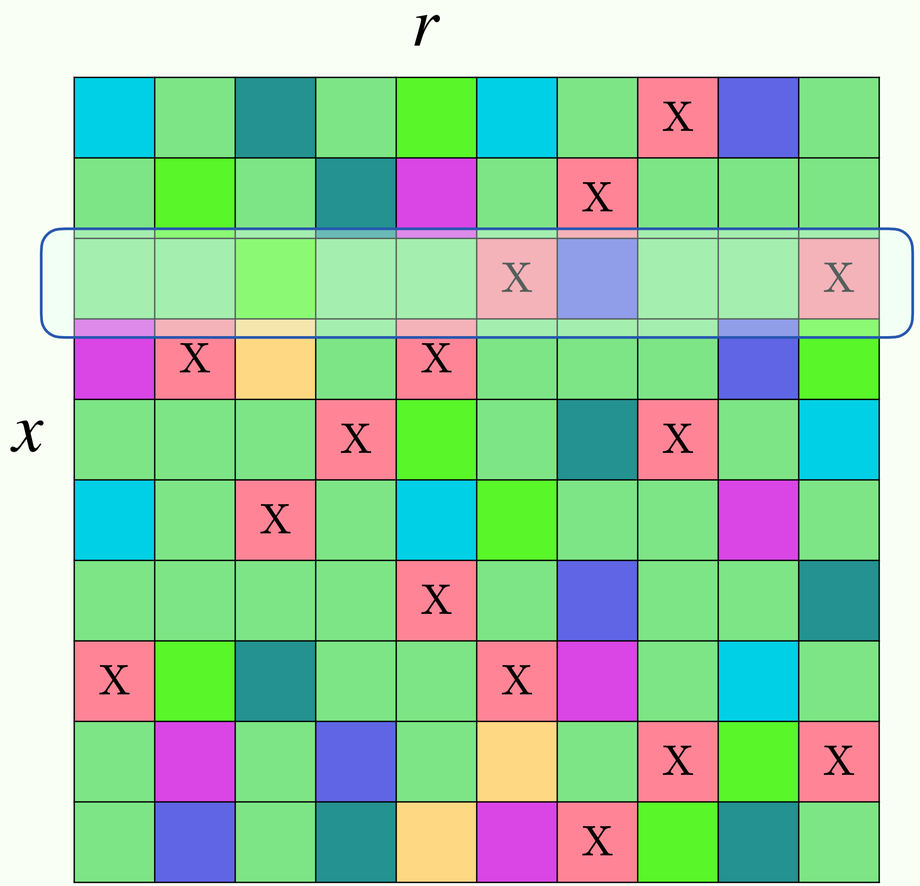
Goal: compute **search** problem $S \subset \{0,1\}^* \times \Gamma$

Randomized tree $\mathcal{T} = \{T_r : r \sim R\}$:

- input $x \in \{0,1\}^n$ and seed r
- Correctness: $\Pr_r[T_r[x] \in S(x)] \geq 2/3$
- $\mathsf{FBPP}^{\mathsf{dt}} = \{S : \exists$ efficient $\mathcal{T}$ computing $S\}$

- ~~Error amplification~~ ←fixed by verifiability

$(y \in S(x) \iff (x, y) \in S)$



Algorithms for search problems

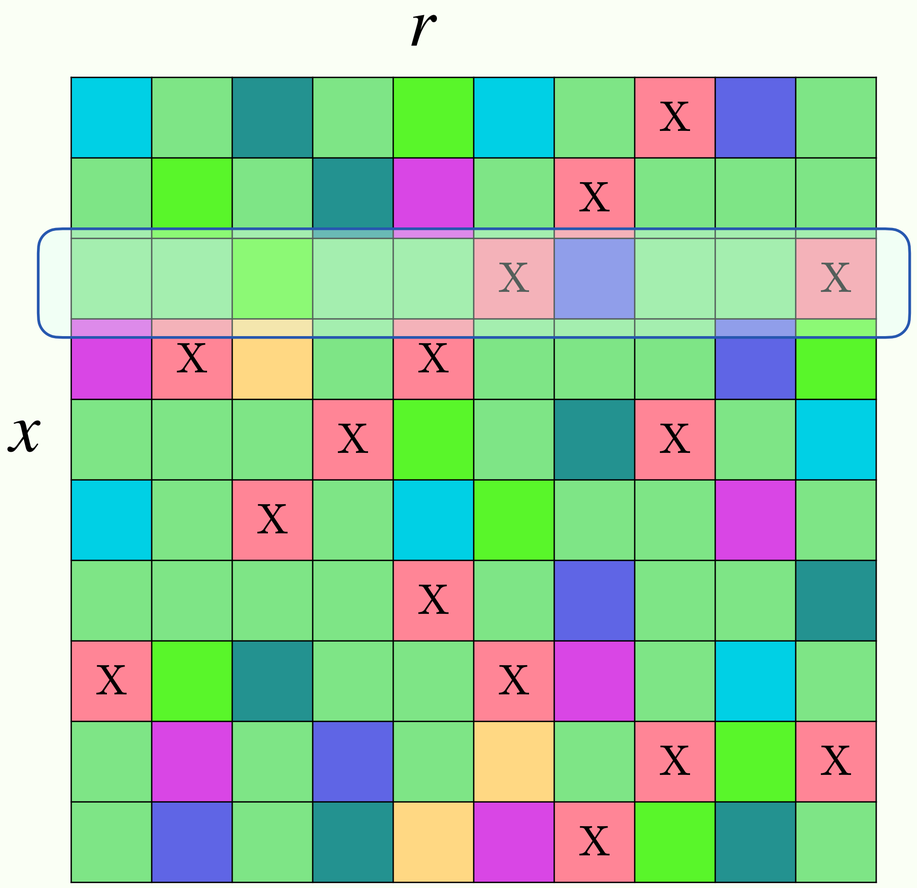
Goal: compute **search** problem $S \subset \{0,1\}^* \times \Gamma$

Randomized tree $\mathcal{T} = \{T_r : r \sim R\}$:

- input $x \in \{0,1\}^n$ and seed r
- Correctness: $\Pr_r[T_r[x] \in S(x)] \geq 2/3$
- $\mathsf{FBPP}^{\mathsf{dt}} = \{S : \exists$ efficient $\mathcal{T}$ computing $S\}$

- ~~Error amplification~~ ←fixed by verifiability
- $S \in \mathsf{FBPP}^{\mathsf{dt}} \cap \mathsf{FNP}^{\mathsf{dt}} \Rightarrow \epsilon = 1/\text{quasipoly}(n)$

$(y \in S(x) \iff (x, y) \in S)$



Algorithms for search problems

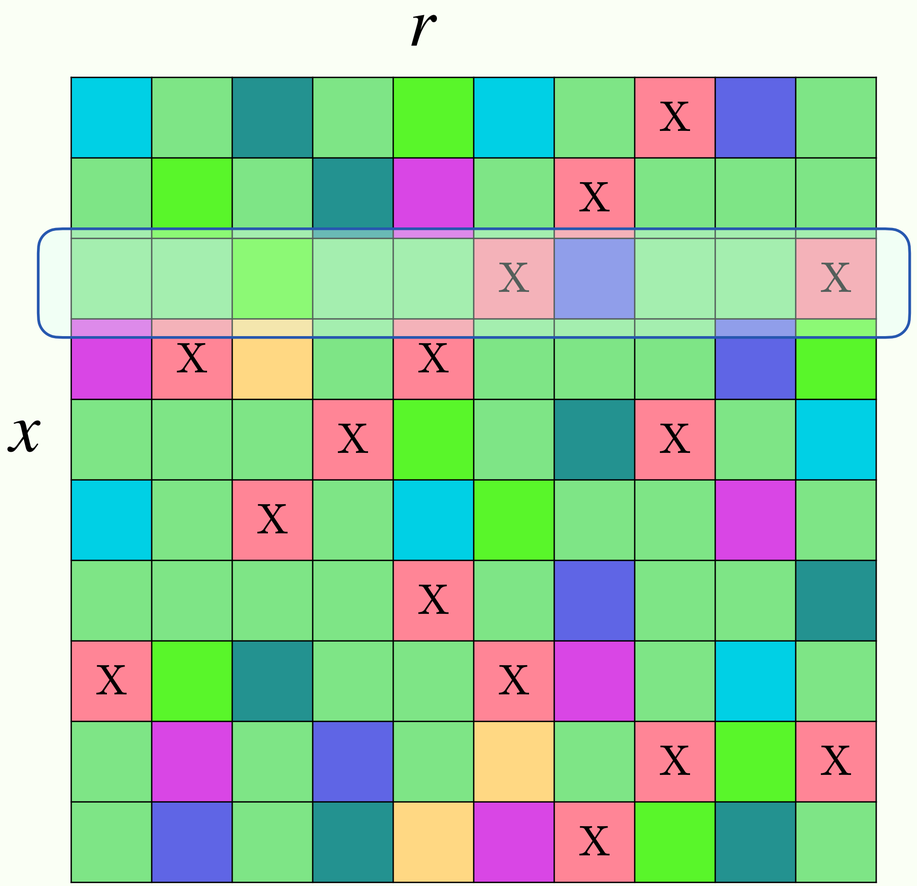
Goal: compute **search** problem $S \subset \{0,1\}^* \times \Gamma$

Randomized tree $\mathcal{T} = \{T_r : r \sim R\}$:

- input $x \in \{0,1\}^n$ and seed r
- Correctness: $\Pr_r[T_r[x] \in S(x)] \geq 2/3$
- $\mathsf{FBPP}^{\mathsf{dt}} = \{S : \exists$ efficient $\mathcal{T}$ computing $S\}$

- ~~Error amplification~~ ←fixed by verifiability
- $S \in \mathsf{FBPP}^{\mathsf{dt}} \cap \mathsf{FNP}^{\mathsf{dt}} \Rightarrow \epsilon = 1/\text{quasipoly}(n)$
- ~~indistinguishable from deterministic algorithm?~~

$(y \in S(x) \iff (x, y) \in S)$



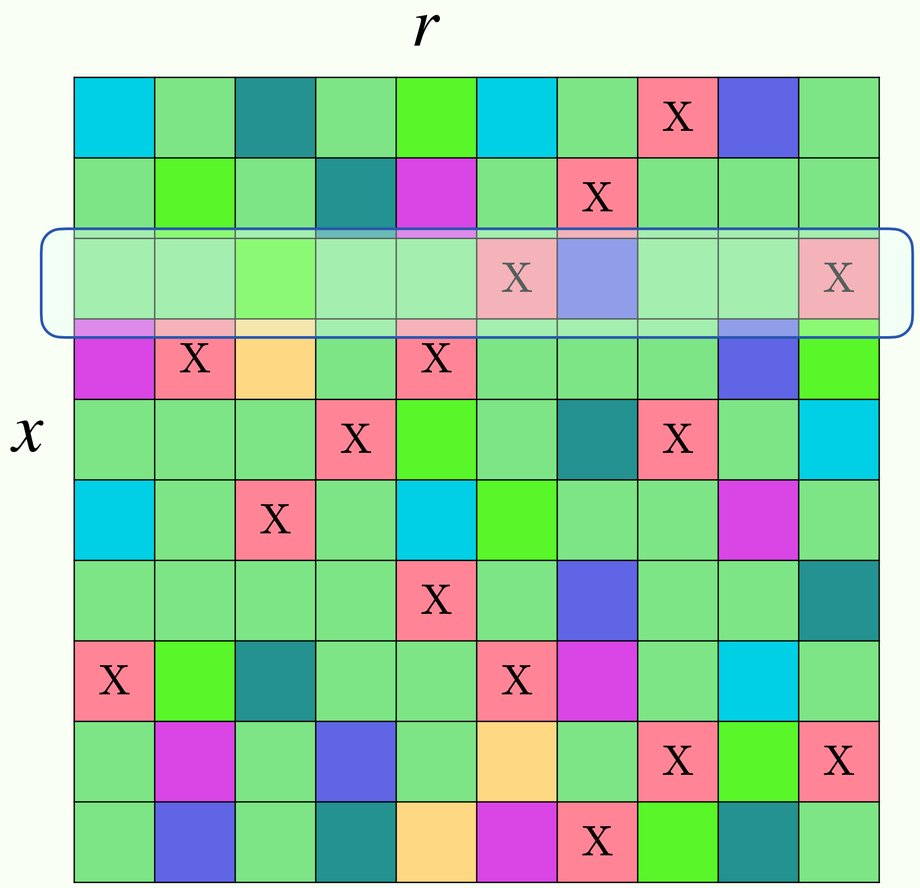
Algorithms for search problems

Goal: compute **search** problem $S \subset \{0,1\}^* \times \Gamma$

$(y \in S(x) \iff (x,y) \in S)$

Randomized tree $\mathcal{T} = \{T_r : r \sim R\}$:

- input $x \in \{0,1\}^n$ and seed r
- Correctness: $\Pr_r[T_r[x] \in S(x)] \geq 2/3$
- $\mathsf{FBPP}^{\mathsf{dt}} = \{S : \exists$ efficient $\mathcal{T}$ computing $S\}$



- ~~Error amplification~~ ←fixed by verifiability
- $S \in \mathsf{FBPP}^{\mathsf{dt}} \cap \mathsf{FNP}^{\mathsf{dt}} \Rightarrow \epsilon = 1/\text{quasipoly}(n)$
- ~~indistinguishable from deterministic algorithm?~~

Problem: not necessarily consistent answers

(different outputs for the same input)

Algorithms for search problems

Goal: compute **search** problem $S \subset \{0,1\}^* \times \Gamma$

$(y \in S(x) \iff (x,y) \in S)$

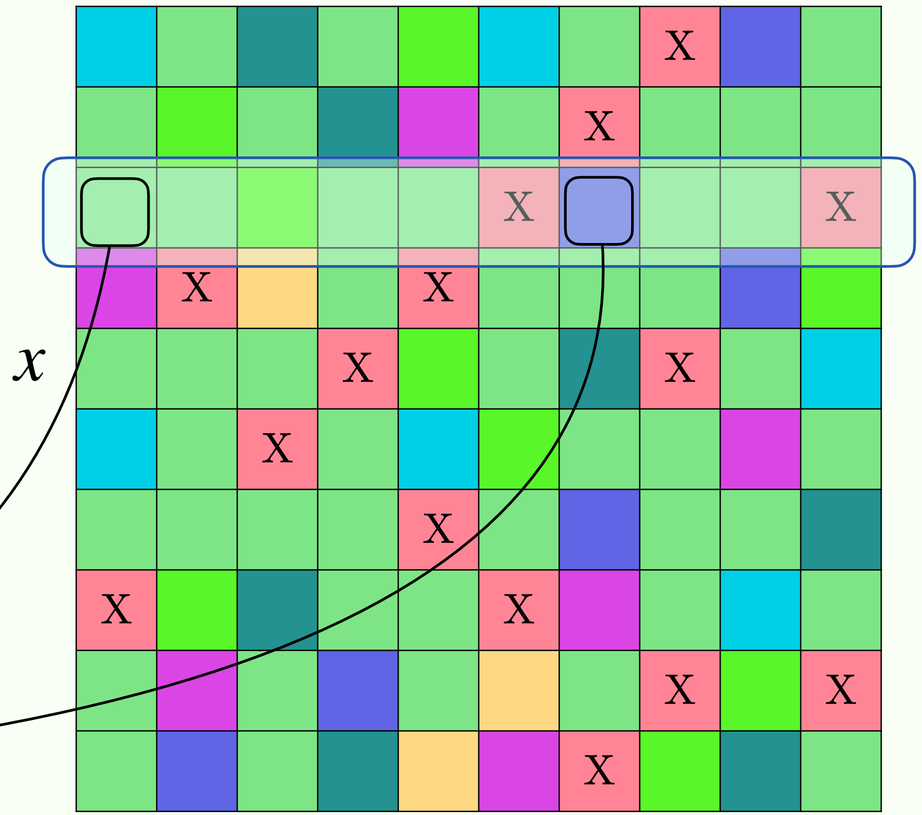
Randomized tree $\mathcal{T} = \{T_r : r \sim R\}$:

- input $x \in \{0,1\}^n$ and seed r
- Correctness: $\Pr_r[T_r[x] \in S(x)] \geq 2/3$
- $\mathsf{FBPP}^{\mathsf{dt}} = \{S : \exists$ efficient $\mathcal{T}$ computing $S\}$

- ~~Error amplification~~ ←fixed by verifiability
- $S \in \mathsf{FBPP}^{\mathsf{dt}} \cap \mathsf{FNP}^{\mathsf{dt}} \Rightarrow \epsilon = 1/\text{quasipoly}(n)$
- ~~indistinguishable from deterministic algorithm?~~

What is $f(x_1)$?

What is $f(x_1)$?



Problem: not necessarily consistent answers

(different outputs for the same input)

Algorithms for search problems

Goal: compute **search** problem $S \subset \{0,1\}^* \times \Gamma$

$(y \in S(x) \iff (x,y) \in S)$

Randomized tree $\mathcal{T} = \{T_r : r \sim R\}$:

- input $x \in \{0,1\}^n$ and seed r
- Correctness: $\Pr_r[T_r[x] \in S(x)] \geq 2/3$
- $\mathsf{FBPP}^{\mathsf{dt}} = \{S : \exists$ efficient $\mathcal{T}$ computing $S\}$

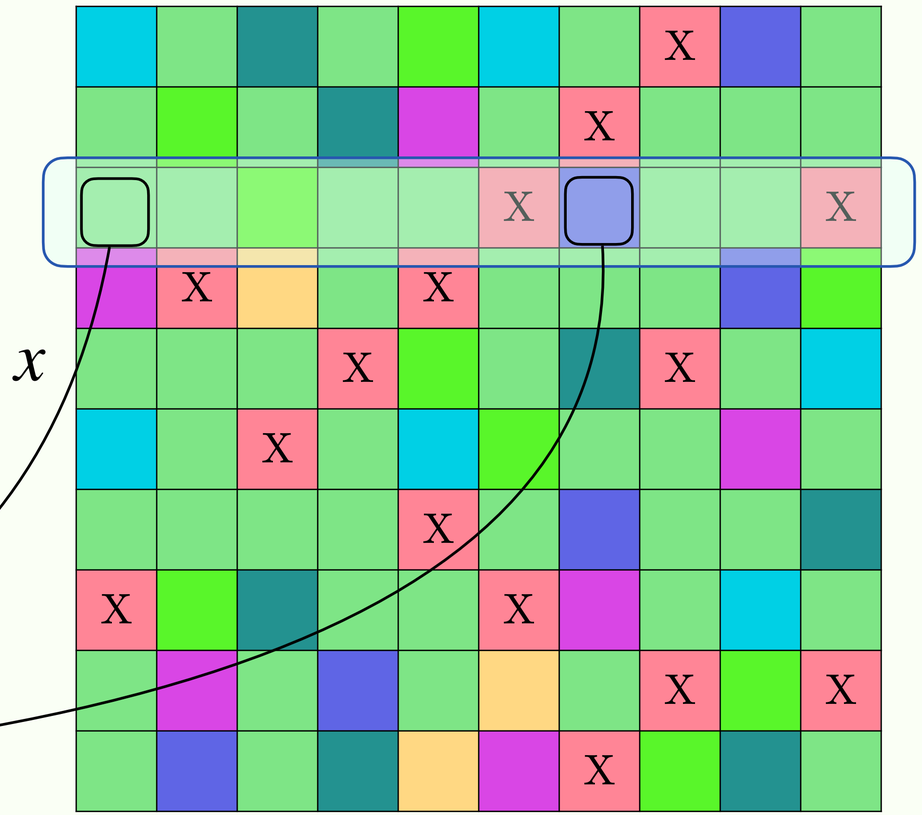
- ~~Error amplification~~ ←fixed by verifiability
- $S \in \mathsf{FBPP}^{\mathsf{dt}} \cap \mathsf{FNP}^{\mathsf{dt}} \Rightarrow \epsilon = 1/\text{quasipoly}(n)$
- ~~indistinguishable from deterministic algorithm?~~

What is $f(x_1)$?

What is $f(x_1)$?

Problem: not necessarily consistent answers

(different outputs for the same input)



Pseudo-deterministic algorithm?

Pseudo-deterministic algorithms for search problems [GG11]

- **Goal:** compute **search** problem $S \subset \{0,1\}^* \times \Gamma$

Pseudo-deterministic algorithms for search problems [GG11]

- **Goal:** compute **search** problem $S \subset \{0,1\}^* \times \Gamma$

Pseudo-deterministic algorithm:

Pseudo-deterministic algorithms for search problems [GG11]

- **Goal:** compute **search** problem $S \subset \{0,1\}^* \times \Gamma$

Pseudo-deterministic algorithm:

- $\mathcal{T} = \{T_r : r \sim R\}$

Pseudo-deterministic algorithms for search problems [GG11]

- **Goal:** compute **search** problem $S \subset \{0,1\}^* \times \Gamma$

Pseudo-deterministic algorithm:

- $\mathcal{T} = \{T_r : r \sim R\}$
- input $x \in \{0,1\}^n$, seed r, output $T_r[x]$

Pseudo-deterministic algorithms for search problems [GG11]

- **Goal:** compute **search** problem $S \subset \{0,1\}^* \times \Gamma$

Pseudo-deterministic algorithm:

- $\mathcal{T} = \{T_r : r \sim R\}$
- input $x \in \{0,1\}^n$, seed r, output $T_r[x]$
- Correctness: there exists a canonical $y_x \in S(x)$ such that $\Pr_r[T_r[x] = y_x] \geq 2/3$

Pseudo-deterministic algorithms for search problems [GG11]

- **Goal:** compute **search** problem $S \subset \{0,1\}^* \times \Gamma$

Pseudo-deterministic algorithm:

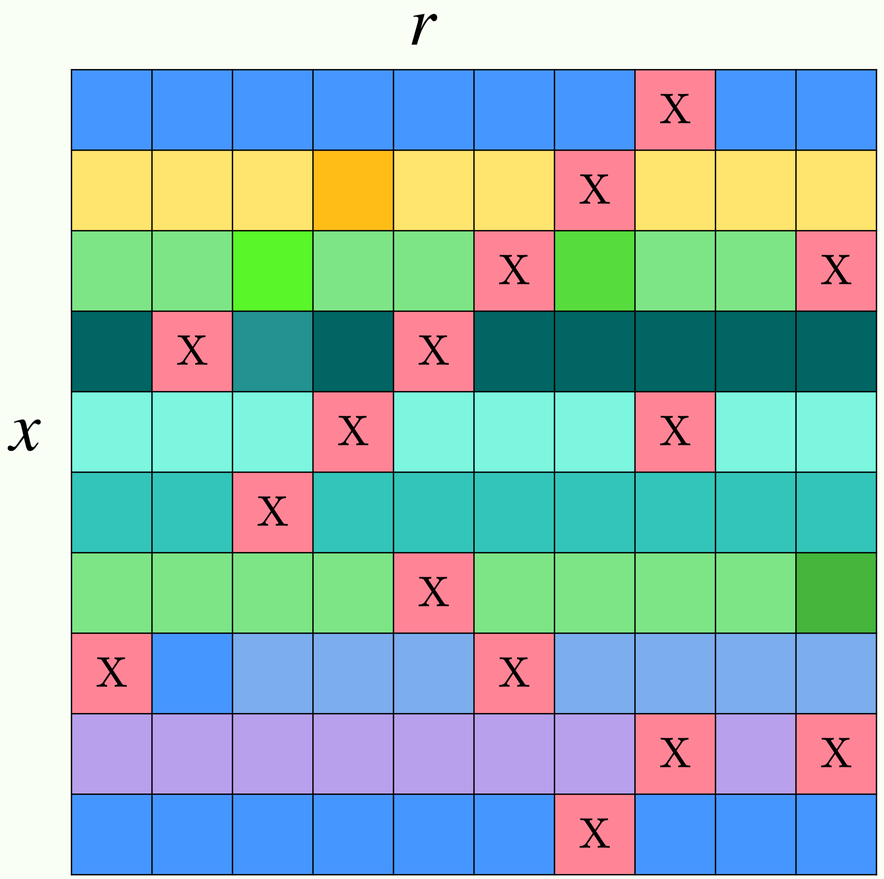
- $\mathcal{T} = \{T_r : r \sim R\}$
- input $x \in \{0,1\}^n$, seed r, output $T_r[x]$
- Correctness: there exists a canonical $y_x \in S(x)$ such that $\Pr_r[T_r[x] = y_x] \geq 2/3$
- Class: $\mathsf{PseudoP}^{\mathrm{dt}}$

Pseudo-deterministic algorithms for search problems [GG11]

- **Goal:** compute **search** problem $S \subset \{0,1\}^* \times \Gamma$

Pseudo-deterministic algorithm:

- $\mathcal{T} = \{T_r : r \sim R\}$
- input $x \in \{0,1\}^n$, seed r, output $T_r[x]$
- Correctness: there exists a canonical $y_x \in S(x)$ such that $\Pr_r[T_r[x] = y_x] \geq 2/3$
- Class: $\mathsf{PseudoP}^{\mathrm{dt}}$

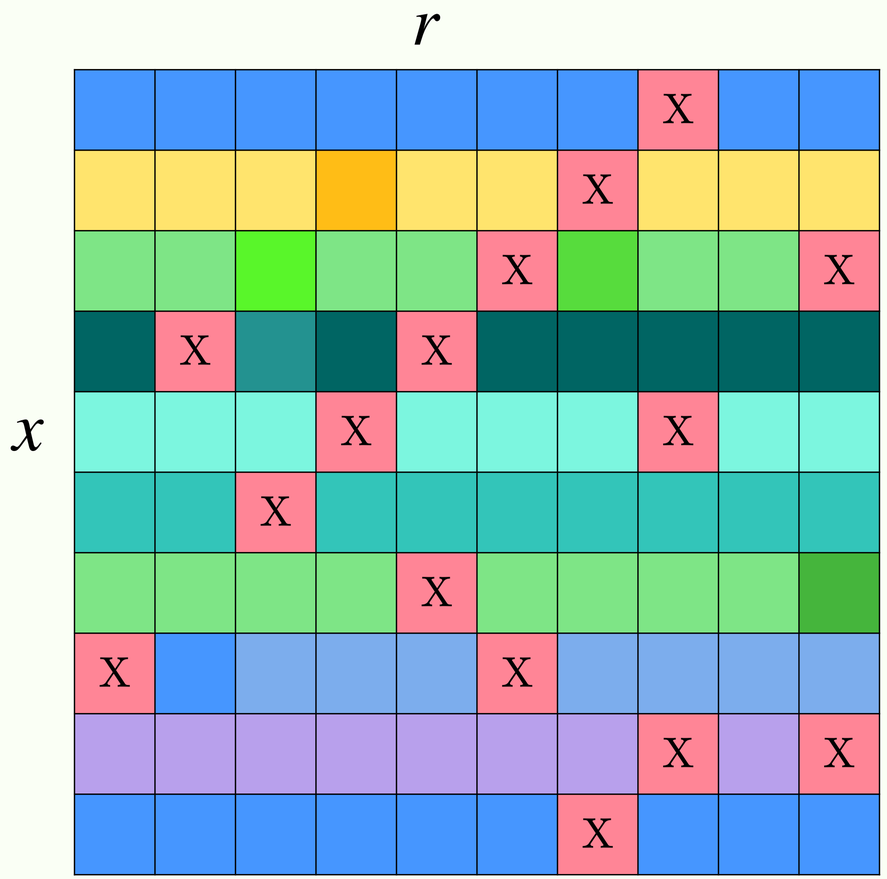


Pseudo-deterministic algorithms for search problems [GG11]

- **Goal:** compute **search** problem $S \subset \{0,1\}^* \times \Gamma$

Pseudo-deterministic algorithm:

- $\mathcal{T} = \{T_r : r \sim R\}$
- input $x \in \{0,1\}^n$, seed r, output $T_r[x]$
- Correctness: there exists a canonical $y_x \in S(x)$ such that $\Pr_r[T_r[x] = y_x] \geq 2/3$
- Class: $\mathsf{PseudoP}^{\mathrm{dt}}$



- indistinguishable from deterministic algorithm

Are we done?!

Are we done?!

- [GIPS'21] proved an exponential lower-bound for pseudo-deterministic complexity for the FIND1 problem: a complete problem for $\mathsf{FBPP}^{\mathsf{dt}} \cap \mathsf{FNP}^{\mathsf{dt}}$:

Are we done?!

- [GIPS'21] proved an exponential lower-bound for pseudo-deterministic complexity for the FIND1 problem: a complete problem for $\mathsf{FBPP}^{\mathsf{dt}} \cap \mathsf{FNP}^{\mathsf{dt}}$:

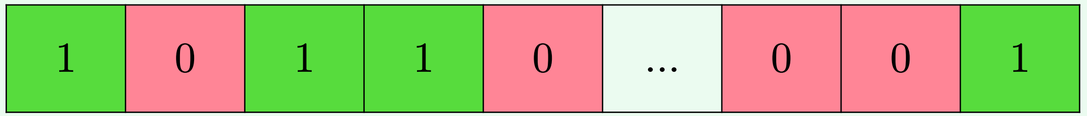
FIND1 Problem

Are we done?!

- [GIPS'21] proved an exponential lower-bound for pseudo-deterministic complexity for the FIND1 problem: a complete problem for $\mathsf{FBPP}^{\mathsf{dt}} \cap \mathsf{FNP}^{\mathsf{dt}}$:

FIND1 Problem

- Input: $x \in \{0,1\}^n$ with Hamming weight $|x| \geq n/2$

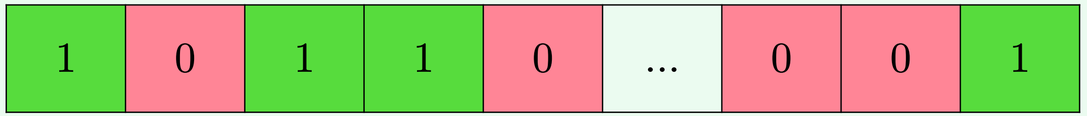


Are we done?!

- [GIPS'21] proved an exponential lower-bound for pseudo-deterministic complexity for the FIND1 problem: a complete problem for $\mathsf{FBPP}^{\mathsf{dt}} \cap \mathsf{FNP}^{\mathsf{dt}}$:

FIND1 Problem

- Input: $x \in \{0,1\}^n$ with Hamming weight $|x| \geq n/2$
- Output: Find i such that $x_i = 1$

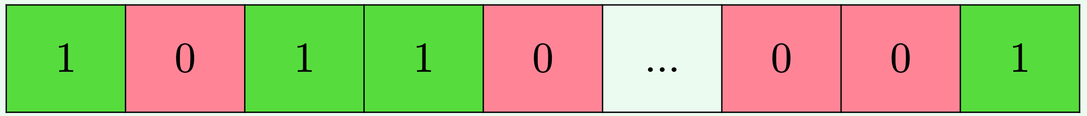


Are we done?!

- [GIPS'21] proved an exponential lower-bound for pseudo-deterministic complexity for the FIND1 problem: a complete problem for $\mathsf{FBPP}^{\mathsf{dt}} \cap \mathsf{FNP}^{\mathsf{dt}}$:

FIND1 Problem

- Input: $x \in \{0,1\}^n$ with Hamming weight $|x| \geq n/2$
- Output: Find i such that $x_i = 1$

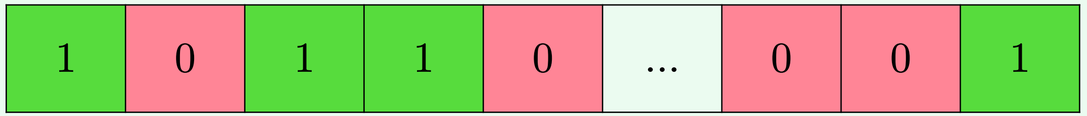


Are we done?!

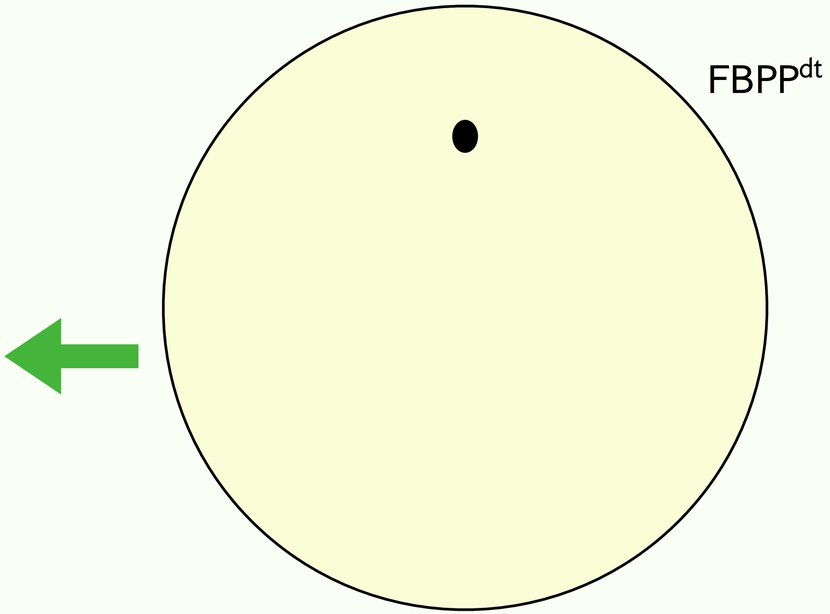
- [GIPS'21] proved an exponential lower-bound for pseudo-deterministic complexity for the FIND1 problem: a complete problem for $\mathsf{FBPP}^{dt} \cap \mathsf{FNP}^{dt}$:

FIND1 Problem

- Input: $x \in \{0,1\}^n$ with Hamming weight $|x| \geq n/2$
- Output: Find i such that $x_i = 1$



- Randomized and verification $\Theta(1)$ queries

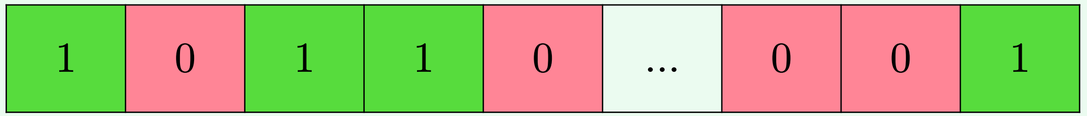


Are we done?!

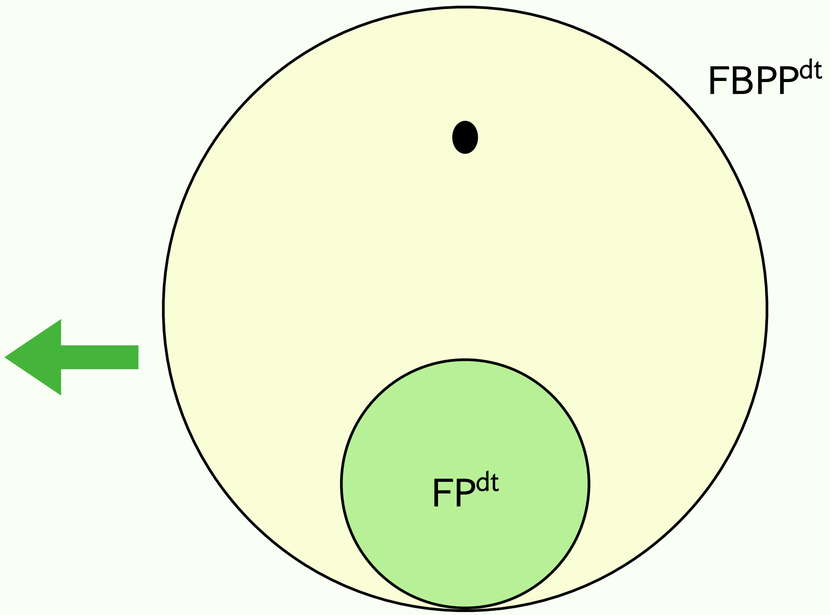
- [GIPS'21] proved an exponential lower-bound for pseudo-deterministic complexity for the FIND1 problem: a complete problem for $\mathsf{FBPP}^{\mathsf{dt}} \cap \mathsf{FNP}^{\mathsf{dt}}$:

FIND1 Problem

- Input: $x \in \{0,1\}^n$ with Hamming weight $|x| \geq n/2$
- Output: Find i such that $x_i = 1$



- Randomized and verification — $\Theta(1)$ queries
- Deterministic complexity — $\Theta(n)$ queries

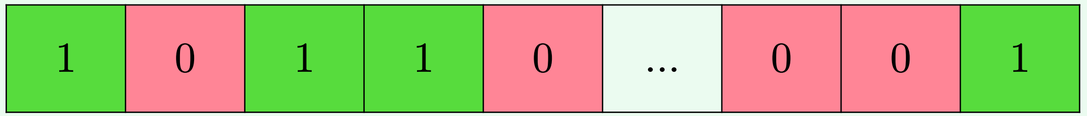


Are we done?!

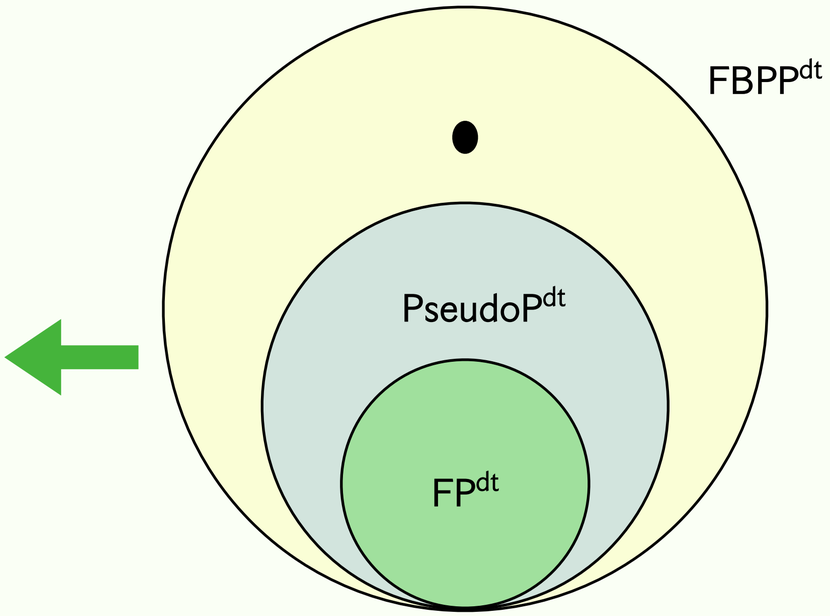
- [GIPS'21] proved an exponential lower-bound for pseudo-deterministic complexity for the FIND1 problem: a complete problem for $\mathsf{FBPP}^{\mathsf{dt}} \cap \mathsf{FNP}^{\mathsf{dt}}$:

FIND1 Problem

- Input: $x \in \{0,1\}^n$ with Hamming weight $|x| \geq n/2$
- Output: Find i such that $x_i = 1$



- Randomized and verification $\Theta(1)$ queries
- Deterministic complexity $\Theta(n)$ queries
- Pseudo-deterministic complexity $\Omega(\sqrt{n})$ queries

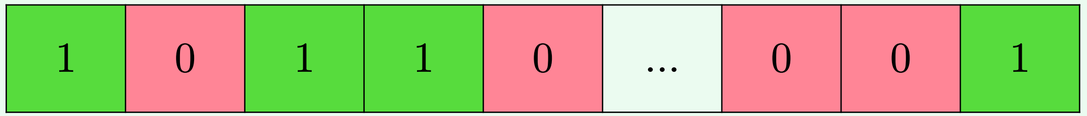


Are we done?!

- [GIPS'21] proved an exponential lower-bound for pseudo-deterministic complexity for the FIND1 problem: a complete problem for $\mathsf{FBPP}^{\mathsf{dt}} \cap \mathsf{FNP}^{\mathsf{dt}}$:

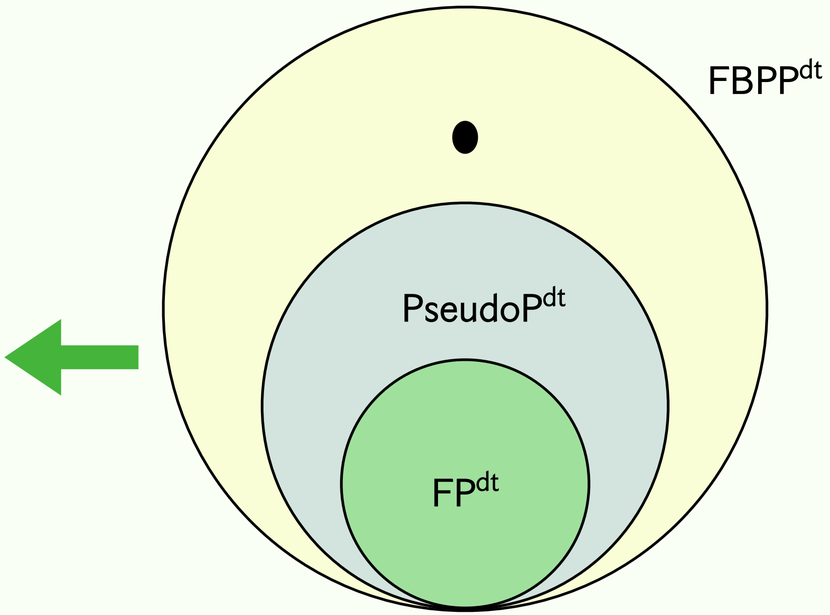
FIND1 Problem

- Input: $x \in \{0,1\}^n$ with Hamming weight $|x| \geq n/2$
- Output: Find i such that $x_i = 1$



- Randomized and verification $\Theta(1)$ queries
- Deterministic complexity $\Theta(n)$ queries
- Pseudo-deterministic complexity $\Omega(\sqrt{n})$ queries

Can we get a cheaper "pseudo"-deterministic algorithm?

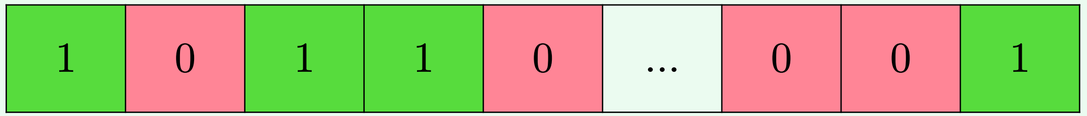


Are we done?!

- [GIPS'21] proved an exponential lower-bound for pseudo-deterministic complexity for the FIND1 problem: a complete problem for $\mathsf{FBPP}^{\mathsf{dt}} \cap \mathsf{FNP}^{\mathsf{dt}}$:

FIND1 Problem

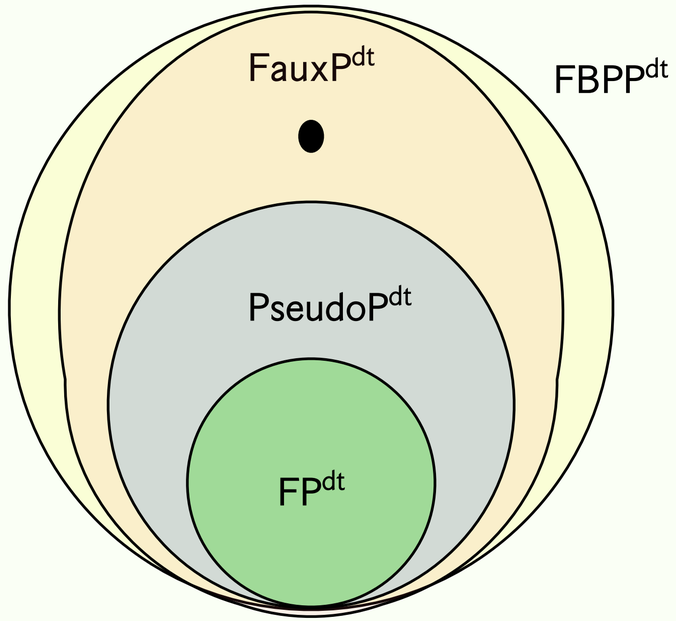
- Input: $x \in \{0,1\}^n$ with Hamming weight $|x| \geq n/2$
- Output: Find i such that $x_i = 1$



- Randomized and verification $\Theta(1)$ queries
- Deterministic complexity $\Theta(n)$ queries
- Pseudo-deterministic complexity $\Omega(\sqrt{n})$ queries

Can we get a cheaper "pseudo"-deterministic algorithm?

Yes! Faux-determinism~
an incorrect input to the algorithm can't be found efficiently

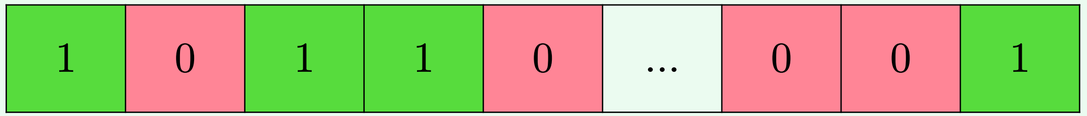


Are we done?!

- [GIPS'21] proved an exponential lower-bound for pseudo-deterministic complexity for the FIND1 problem: a complete problem for $\mathsf{FBPP}^{\mathsf{dt}} \cap \mathsf{FNP}^{\mathsf{dt}}$:

FIND1 Problem

- Input: $x \in \{0,1\}^n$ with Hamming weight $|x| \geq n/2$
- Output: Find i such that $x_i = 1$

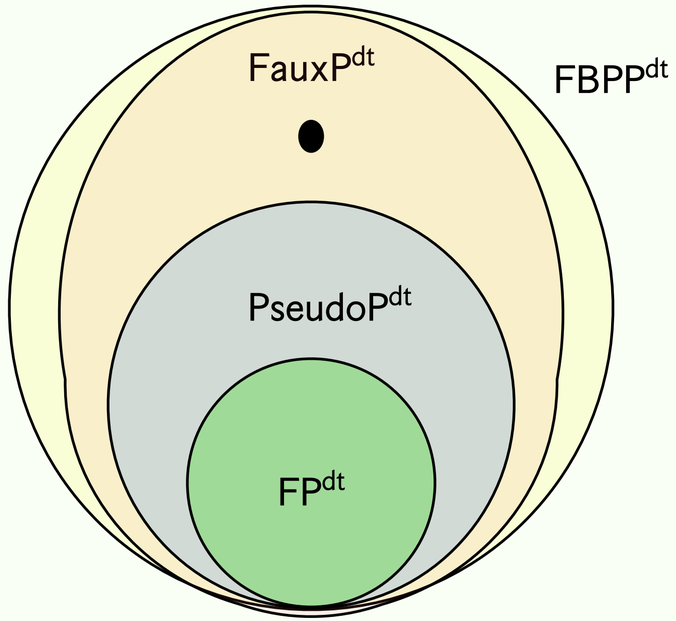


- Randomized and verification $\Theta(1)$ queries
- Deterministic complexity $\Theta(n)$ queries
- Pseudo-deterministic complexity $\Omega(\sqrt{n})$ queries

Can we get a cheaper "pseudo"-deterministic algorithm?

Yes! Faux-determinism~
an incorrect input to the algorithm can't be found efficiently

What is stopping us from proving a $\Theta(n)$ lower bound?



Are we done?!

- [GIPS'21] proved an exponential lower-bound for pseudo-deterministic complexity for the FIND1 problem: a complete problem for $\mathsf{FBPP}^{\mathsf{dt}} \cap \mathsf{FNP}^{\mathsf{dt}}$:

FIND1 Problem

- Input: $x \in \{0,1\}^n$ with Hamming weight $|x| \geq n/2$
- Output: Find i such that $x_i = 1$

1	0	1	1	0	...	0	0	1

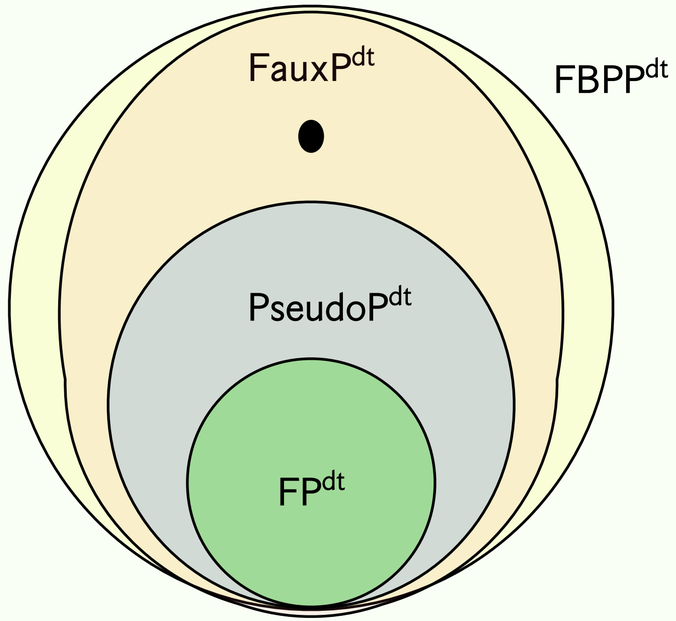
- Randomized and verification $\Theta(1)$ queries
- Deterministic complexity $\Theta(n)$ queries
- Pseudo-deterministic complexity $\Omega(\sqrt{n})$ queries

Can we get a cheaper "pseudo"-deterministic algorithm?

Yes! Faux-determinism~
an incorrect input to the algorithm can't be found efficiently

What is stopping us from proving a $\Theta(n)$ lower bound?

Requires a non-constructive proof?



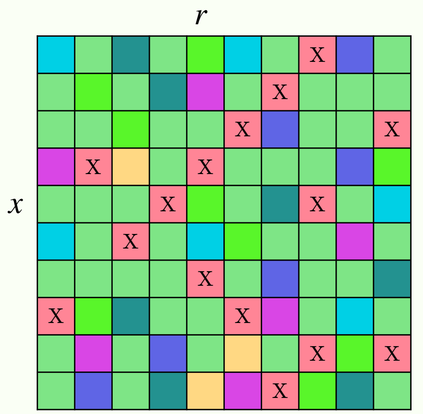
Faux-determinism

Defining faux deterministic algorithms

Faux-deterministic algorithms for search problems

Faux-deterministic algorithms for search problems

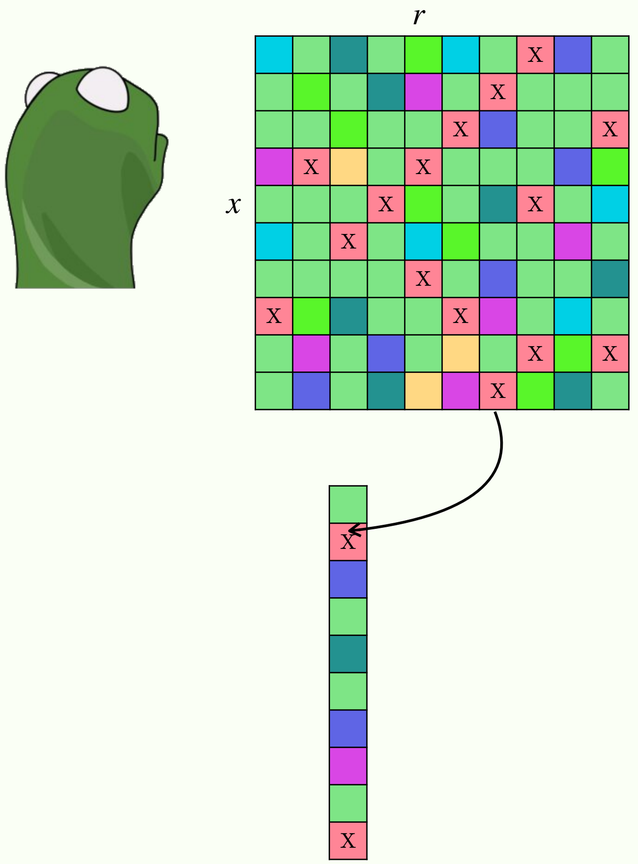
Faux-deterministic tree $\mathscr{T} = \{T_r : r \sim R\}$:



Faux-deterministic algorithms for search problems

Faux-deterministic tree $\mathcal{T} = \{T_r : r \sim R\}$:

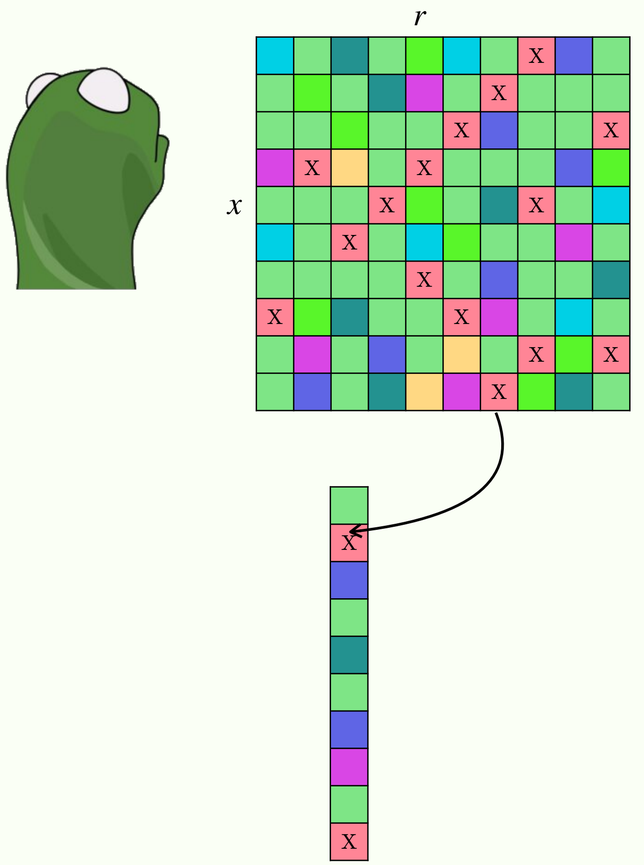
- Sample a random seed $r \sim \text{Unif}$ and **fix it**



Faux-deterministic algorithms for search problems

Faux-deterministic tree $\mathcal{T} = \{T_r : r \sim R\}$:

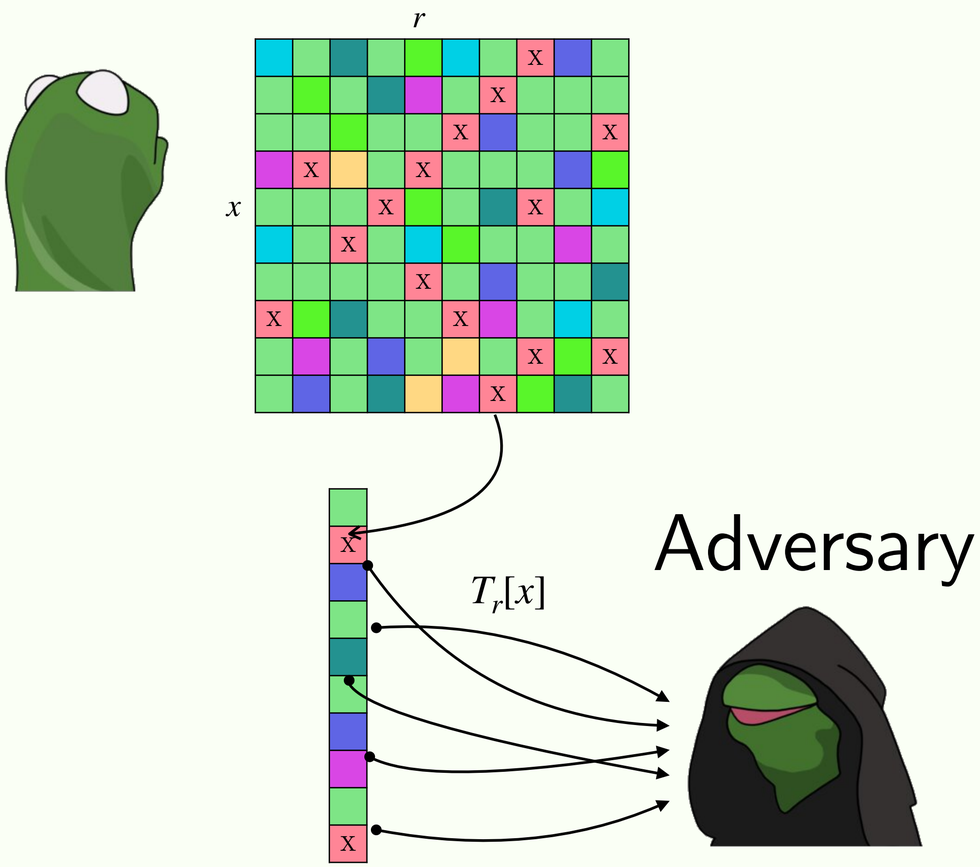
- Sample a random seed $r \sim \text{Unif}$ and **fix it**
- Give truth table of $T_r \in \Gamma^{\{0,1\}^n}$ to any adversary A



Faux-deterministic algorithms for search problems

Faux-deterministic tree $\mathcal{T} = \{T_r : r \sim R\}$:

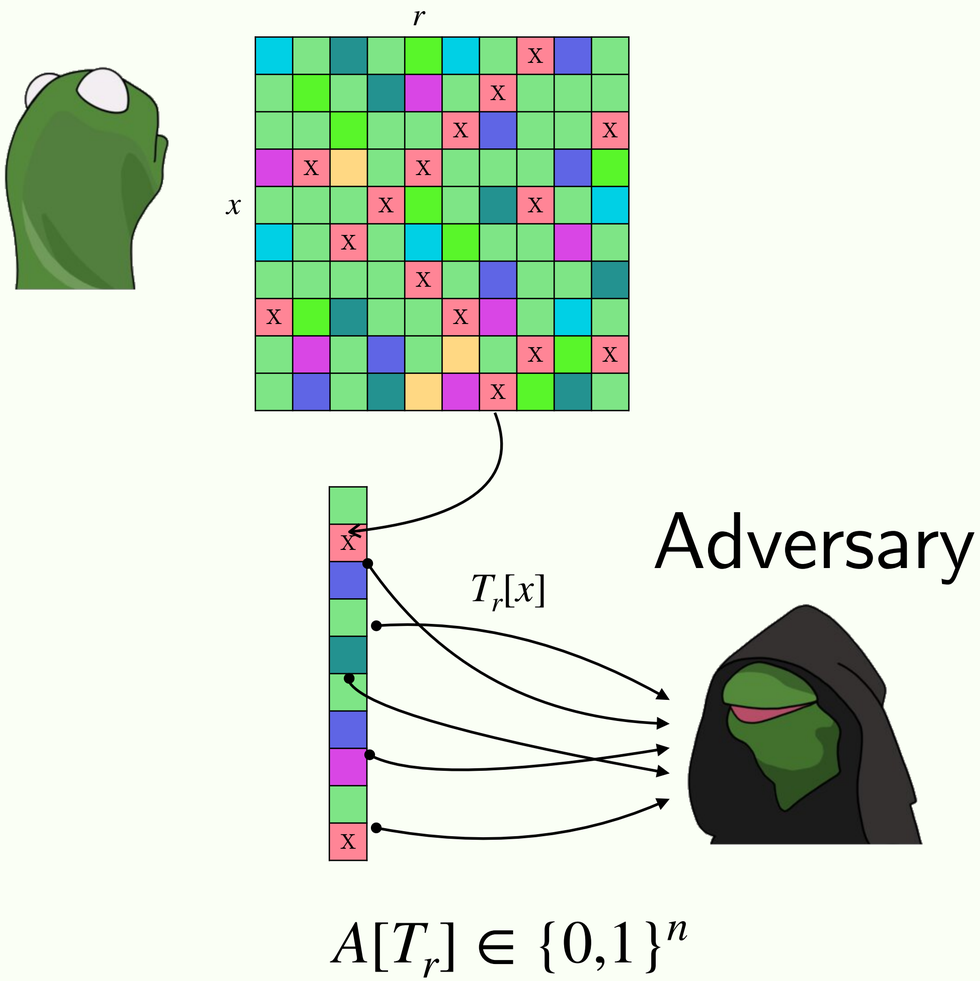
- Sample a random seed $r \sim$ Unif and **fix it**
- Give truth table of $T_r \in \Gamma^{\{0,1\}^n}$ to any adversary A
- A makes poly(n) queries to the truth table of $T_r \in \Gamma^{\{0,1\}^n}$



Faux-deterministic algorithms for search problems

Faux-deterministic tree $\mathcal{T} = \{T_r : r \sim R\}$:

- Sample a random seed $r \sim \text{Unif}$ and **fix it**
- Give truth table of $T_r \in \Gamma^{\{0,1\}^n}$ to any adversary A
- A makes poly(n) queries to the truth table of $T_r \in \Gamma^{\{0,1\}^n}$

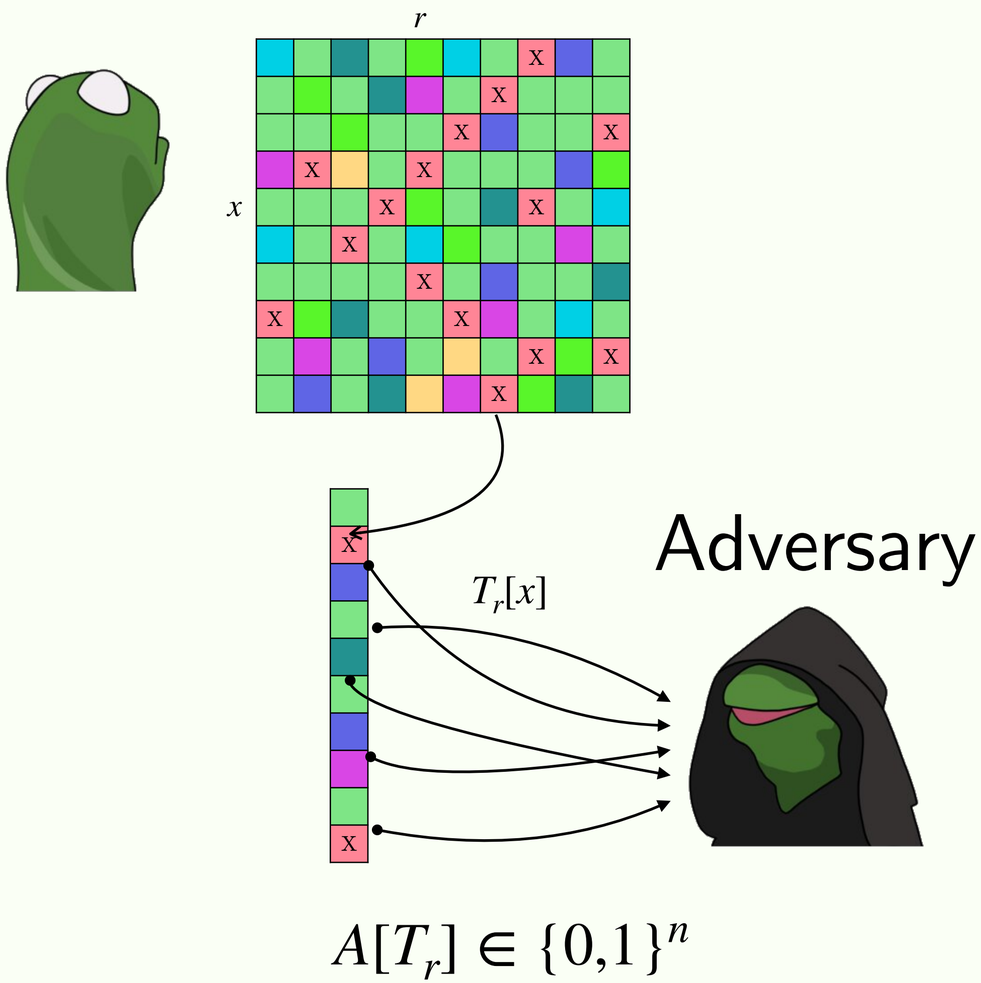


$A[T_r] \in \{0,1\}^n$

Faux-deterministic algorithms for search problems

Faux-deterministic tree $\mathcal{T} = \{T_r : r \sim R\}$:

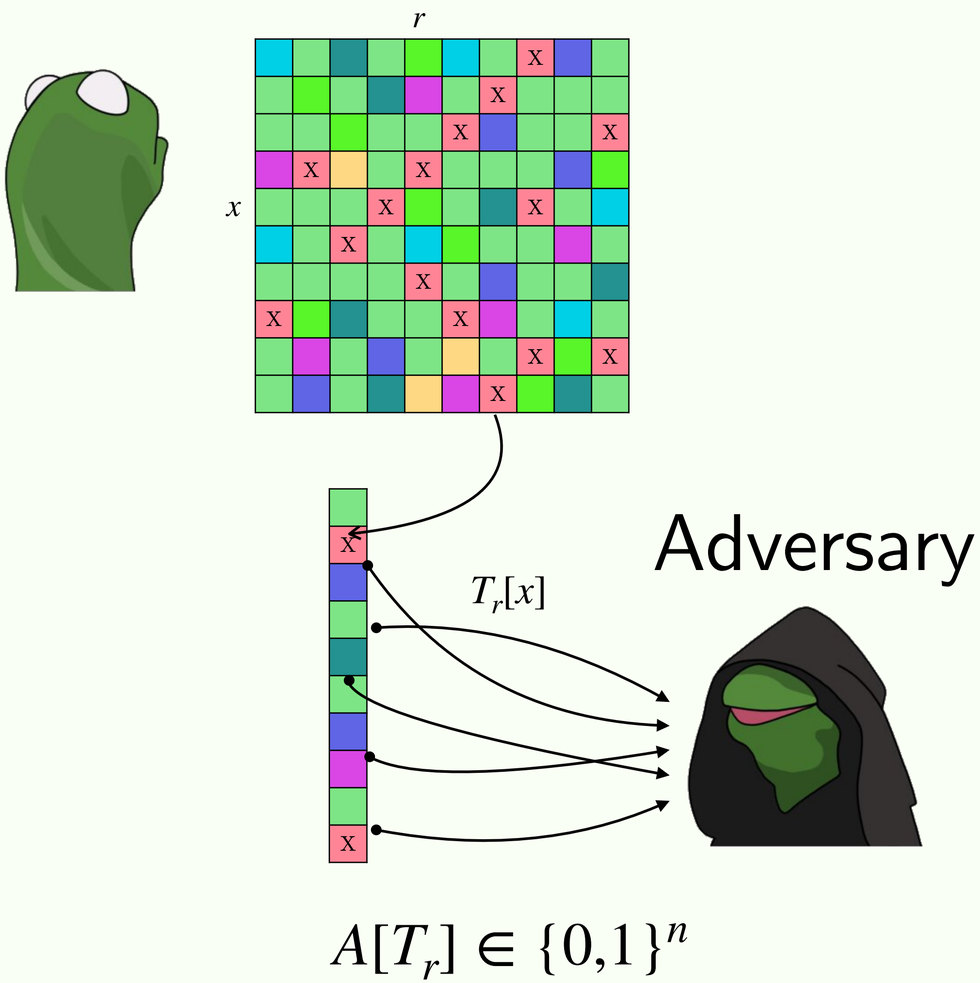
- Sample a random seed $r \sim \text{Unif}$ and **fix it**
- Give truth table of $T_r \in \Gamma^{\{0,1\}^n}$ to any adversary A
- A makes poly(n) queries to the truth table of $T_r \in \Gamma^{\{0,1\}^n}$
- Succeeds to find a "bad" input $T_r[x] \notin S(x)$ with negligible probability



Faux-deterministic algorithms for search problems

Faux-deterministic tree $\mathcal{T} = \{T_r : r \sim R\}$:

- Sample a random seed $r \sim \text{Unif}$ and **fix it**
- Give truth table of $T_r \in \Gamma^{\{0,1\}^n}$ to any adversary A
- A makes poly(n) queries to the truth table of $T_r \in \Gamma^{\{0,1\}^n}$
- Succeeds to find a "bad" input $T_r[x] \notin S(x)$ with negligible probability
- $\mathsf{FauxP}^{\mathsf{dt}} = \{S : \exists \text{polylog depth faux-deterministc tree for } S\}$

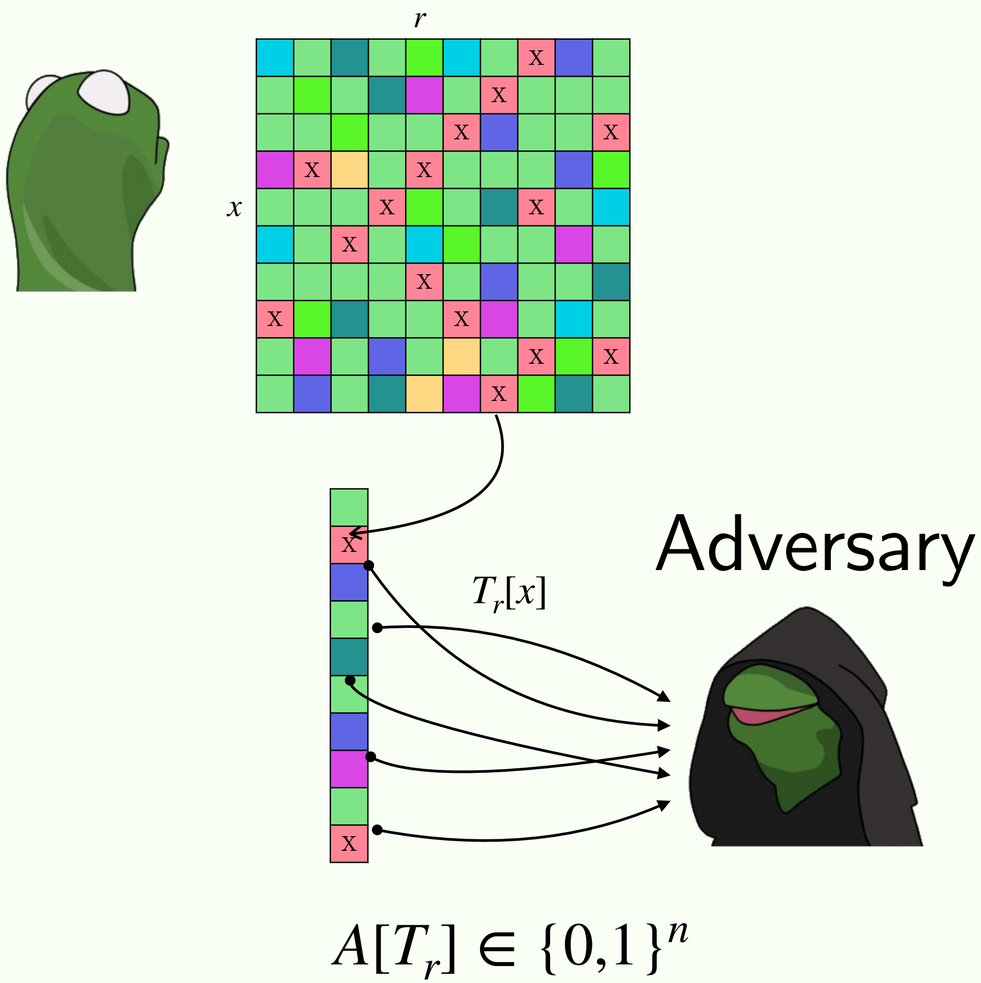


Faux-deterministic algorithms for search problems

Faux-deterministic tree $\mathcal{T} = \{T_r : r \sim R\}$:

- Sample a random seed $r \sim \text{Unif}$ and **fix it**
- Give truth table of $T_r \in \Gamma^{\{0,1\}^n}$ to any adversary A
- A makes poly(n) queries to the truth table of $T_r \in \Gamma^{\{0,1\}^n}$
- Succeeds to find a "bad" input $T_r[x] \notin S(x)$ with negligible probability
- $\mathsf{FauxP}^{\mathsf{dt}} = \{S : \exists \text{polylog depth faux-deterministc tree for } S\}$

Remark: the tree is secret, but the distribution is known to the adversary



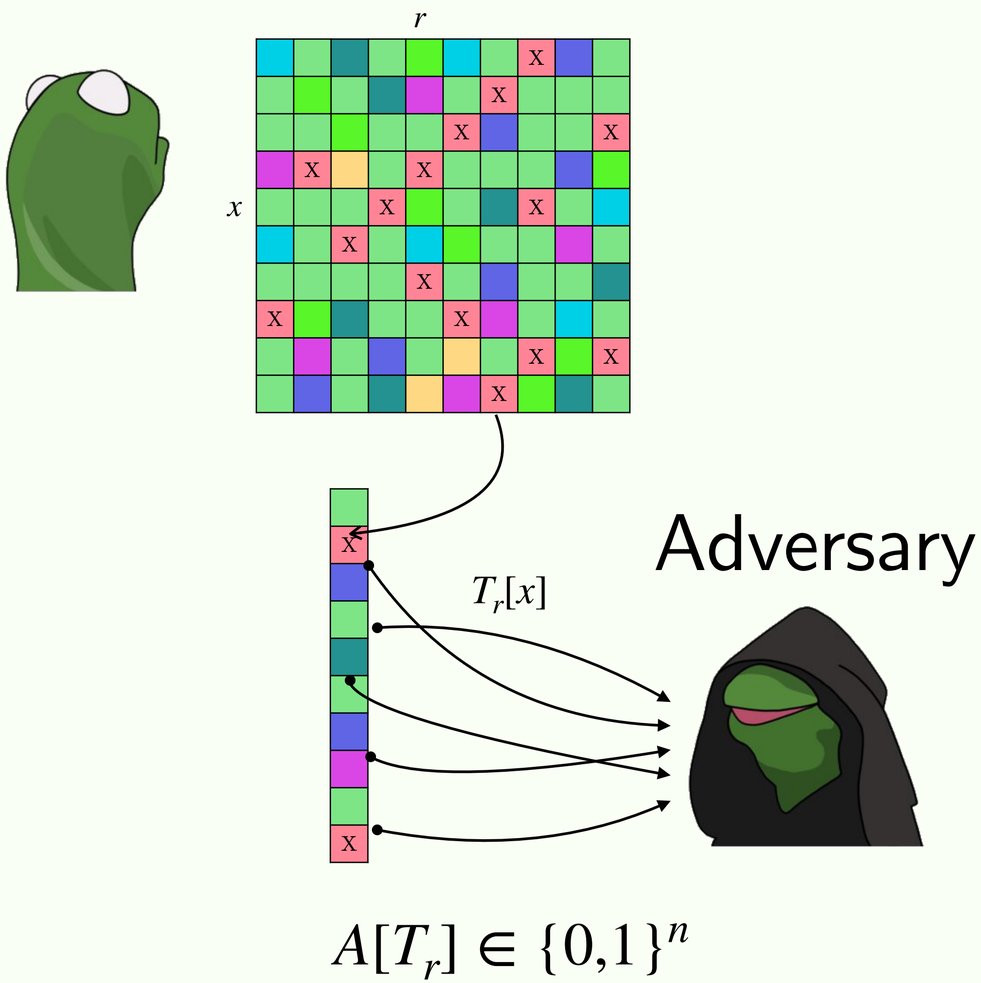
Faux-deterministic algorithms for search problems

Faux-deterministic tree $\mathcal{T} = \{T_r : r \sim R\}$:

- Sample a random seed $r \sim \text{Unif}$ and **fix it**
- Give truth table of $T_r \in \Gamma^{\{0,1\}^n}$ to any adversary A
- A makes poly(n) queries to the truth table of $T_r \in \Gamma^{\{0,1\}^n}$
- Succeeds to find a "bad" input $T_r[x] \notin S(x)$ with negligible probability
- $\mathsf{FauxP}^{\mathsf{dt}} = \{S : \exists \text{polylog depth faux-deterministc tree for } S\}$

Remark: the tree is secret, but the distribution is known to the adversary

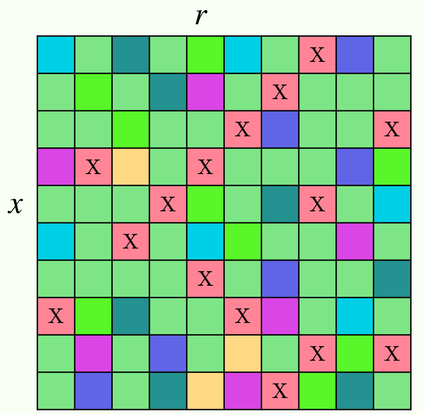
Remark: "set it and forget it" paradigm: the given tree is a fixed deterministic tree

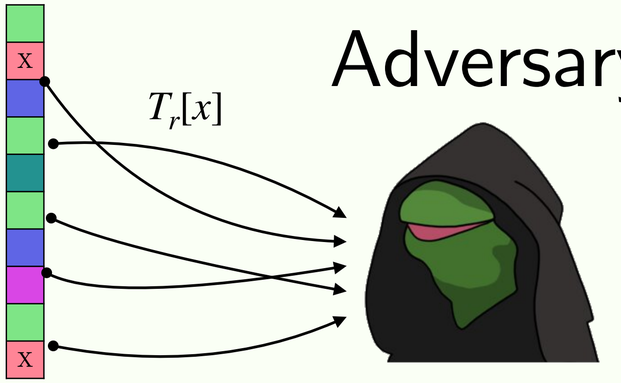


Faux-deterministic algorithms for search problems

Faux-deterministic tree $\mathcal{T} = \{T_r : r \sim R\}$:

- Sample a random seed $r \sim \text{Unif}$ and **fix it**
- Give truth table of $T_r \in \Gamma^{\{0,1\}^n}$ to any adversary A
- A makes poly(n) queries to the truth table of $T_r \in \Gamma^{\{0,1\}^n}$
- Succeeds to find a "bad" input $T_r[x] \notin S(x)$ with negligible probability
- $\mathsf{FauxP}^{\mathsf{dt}} = \{S : \exists \text{polylog depth faux-deterministc tree for } S\}$





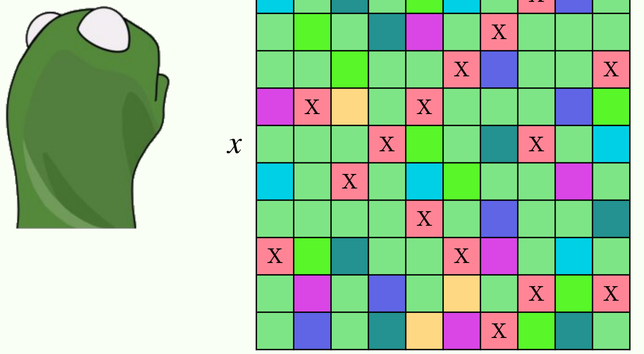
$A[T_r] \in \{0,1\}^n$

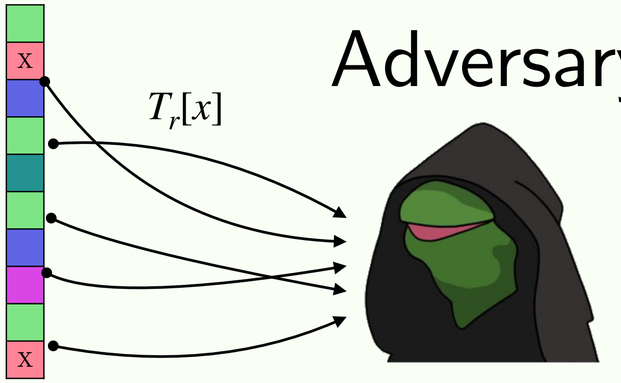
Faux-deterministic algorithms for search problems

Faux-deterministic tree $\mathcal{T} = \{T_r : r \sim R\}$:

- Sample a random seed $r \sim \text{Unif}$ and **fix it**
- Give truth table of $T_r \in \Gamma^{\{0,1\}^n}$ to any adversary A
- A makes poly(n) queries to the truth table of $T_r \in \Gamma^{\{0,1\}^n}$
- Succeeds to find a "bad" input $T_r[x] \notin S(x)$ with negligible probability
- $\mathsf{FauxP}^{\mathrm{dt}} = \{S : \exists \text{polylog depth faux-deterministc tree for } S\}$

Lemma. $\mathsf{FP}^{\mathrm{dt}} \subsetneq \mathsf{PseudoP}^{\mathrm{dt}} \subsetneq \mathsf{FauxP}^{\mathrm{dt}} \subsetneq \mathsf{FBPP}^{\mathrm{dt}}$





$A[T_r] \in \{0,1\}^n$

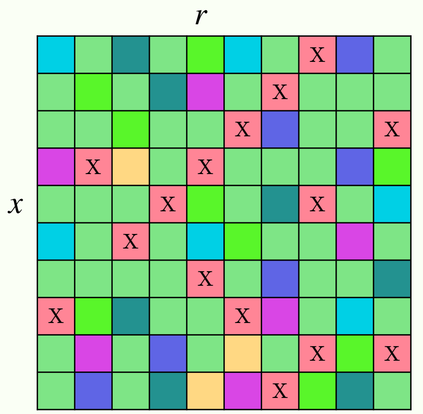
Faux-deterministic algorithms for search problems

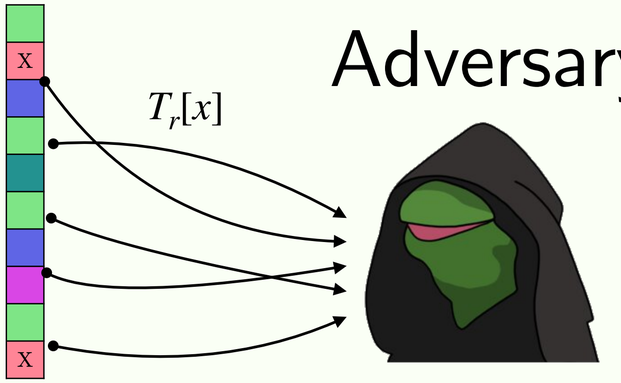
Faux-deterministic tree $\mathcal{T} = \{T_r : r \sim R\}$:

- Sample a random seed $r \sim \text{Unif}$ and **fix it**
- Give truth table of $T_r \in \Gamma^{\{0,1\}^n}$ to any adversary A
- A makes poly(n) queries to the truth table of $T_r \in \Gamma^{\{0,1\}^n}$
- Succeeds to find a "bad" input $T_r[x] \notin S(x)$ with negligible probability
- $\mathsf{FauxP}^{\mathrm{dt}} = \{S : \exists \text{polylog depth faux-deterministc tree for } S\}$

Lemma. $\mathsf{FP}^{\mathrm{dt}} \subsetneq \mathsf{PseudoP}^{\mathrm{dt}} \subsetneq \mathsf{FauxP}^{\mathrm{dt}} \subsetneq \mathsf{FBPP}^{\mathrm{dt}}$

Theorem (faux derandomization).
FIND1 $\in \mathsf{FauxP}^{\mathrm{dt}}$, and by completeness of FIND1,
$\mathsf{FBPP}^{\mathrm{dt}} \cap \mathrm{FNP}^{\mathrm{dt}} \subset \mathsf{FauxP}^{\mathrm{dt}}$





$A[T_r] \in \{0,1\}^n$

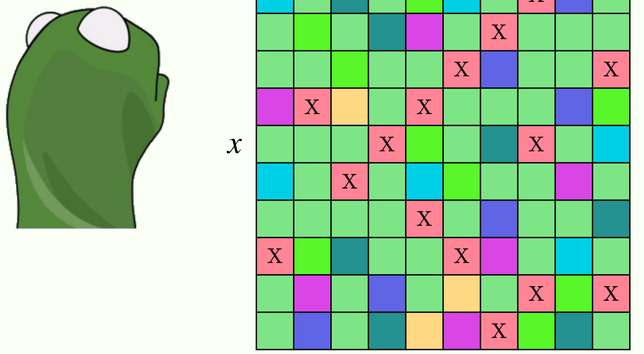
Faux-deterministic algorithms for search problems

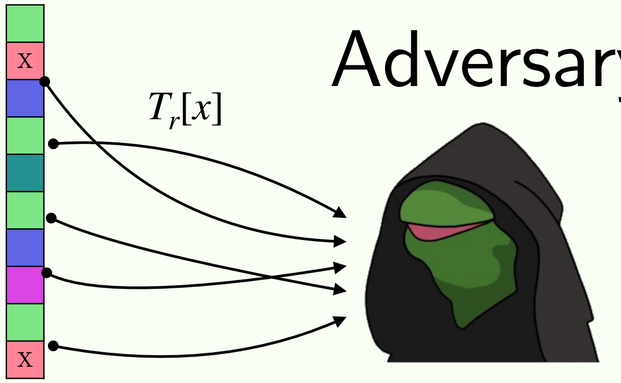
Faux-deterministic tree $\mathcal{T} = \{T_r : r \sim R\}$:

- Sample a random seed $r \sim \text{Unif}$ and **fix it**
- Give truth table of $T_r \in \Gamma^{\{0,1\}^n}$ to any adversary A
- A makes poly(n) queries to the truth table of $T_r \in \Gamma^{\{0,1\}^n}$
- Succeeds to find a "bad" input $T_r[x] \notin S(x)$ with negligible probability
- $\mathsf{FauxP}^{\mathrm{dt}} = \{S : \exists \text{polylog depth faux-deterministc tree for } S\}$

Lemma. $\mathsf{FP}^{\mathrm{dt}} \subsetneq \mathsf{PseudoP}^{\mathrm{dt}} \subsetneq \mathsf{FauxP}^{\mathrm{dt}} \subsetneq \mathsf{FBPP}^{\mathrm{dt}}$

Theorem (faux derandomization).
FIND1 $\in \mathsf{FauxP}^{\mathrm{dt}}$, and by completeness of FIND1,
$\mathsf{FBPP}^{\mathrm{dt}} \cap \mathrm{FNP}^{\mathrm{dt}} \subset \mathsf{FauxP}^{\mathrm{dt}}$





$A[T_r] \in \{0,1\}^n$

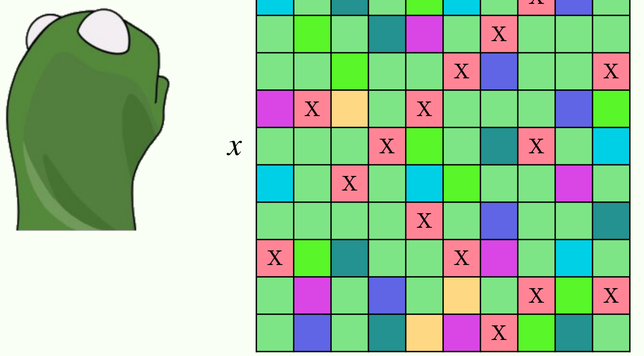
Faux-deterministic algorithms for search problems

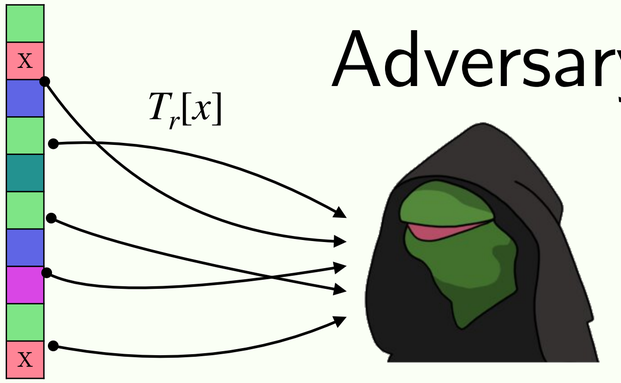
Faux-deterministic tree $\mathcal{T} = \{T_r : r \sim R\}$:

- Sample a random seed $r \sim \text{Unif}$ and **fix it**
- Give truth table of $T_r \in \Gamma^{\{0,1\}^n}$ to any adversary A
- A makes poly(n) queries to the truth table of $T_r \in \Gamma^{\{0,1\}^n}$
- Succeeds to find a "bad" input $T_r[x] \notin S(x)$ with negligible probability
- $\mathsf{FauxP}^{\mathrm{dt}} = \{S : \exists \text{polylog depth faux-deterministc tree for } S\}$

Lemma. $\mathsf{FP}^{\mathrm{dt}} \subsetneq \mathsf{PseudoP}^{\mathrm{dt}} \subsetneq \mathsf{FauxP}^{\mathrm{dt}} \subsetneq \mathsf{FBPP}^{\mathrm{dt}}$

Theorem (faux derandomization).
FIND1 $\in \mathsf{FauxP}^{\mathrm{dt}}$, and by completeness of FIND1,
$\mathsf{FBPP}^{\mathrm{dt}} \cap \mathrm{FNP}^{\mathrm{dt}} \subset \mathsf{FauxP}^{\mathrm{dt}}$





$A[T_r] \in \{0,1\}^n$

See the construction

Faux-deterministic Construction for FIND1

Failure + Failure = Success ?

Faux-deterministic construction for FIND1

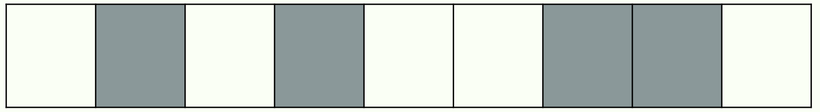
Attempt 1: non-adaptive approach

Faux-deterministic construction for FIND1

Attempt 1: non-adaptive approach

- Choose a random block $B \subset [n]$ of width $|B| = w = \text{polylog}(n)$

B



Faux-deterministic construction for FIND1

Attempt 1: non-adaptive approach

- Choose a random block $B \subset [n]$ of width $|B| = w = \text{polylog}(n)$
- Given x, query $\{x_i : i \in B\}$

B

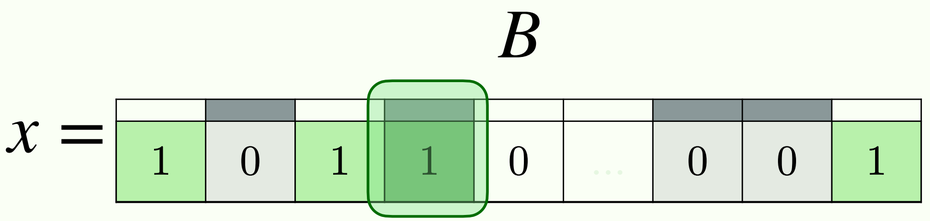
$x =$

1	0	1	1	0	...	0	0	1

Faux-deterministic construction for FIND1

Attempt 1: non-adaptive approach

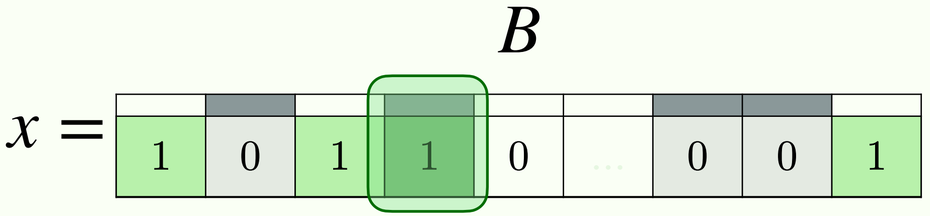
- Choose a random block $B \subset [n]$ of width $|B| = w = \text{polylog}(n)$
- Given x, query $\{x_i : i \in B\}$
- Output $B(x) = \{i \in B : x_i = 1\}$ or $\perp$



Faux-deterministic construction for FIND1

Attempt 1: non-adaptive approach

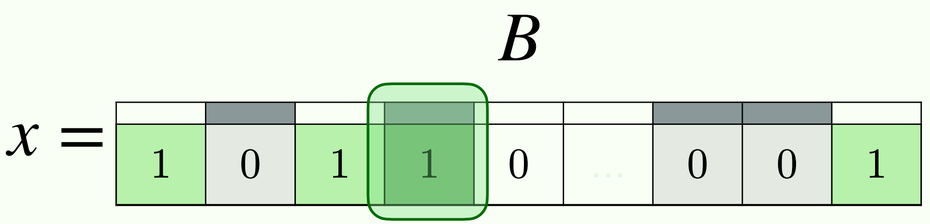
- Choose a random block $B \subset [n]$ of width $|B| = w = \text{polylog}(n)$
- Given x, query $\{x_i : i \in B\}$
- Output $B(x) = \{i \in B : x_i = 1\}$ or $\perp$
- Error $\epsilon = 2^{-W} = 1/\text{quasipoly}(n)$



Faux-deterministic construction for FIND1

Attempt 1: non-adaptive approach

- Choose a random block $B \subset [n]$ of width $|B| = w = \text{polylog}(n)$
- Given x, query $\{x_i : i \in B\}$
- Output $B(x) = \{i \in B : x_i = 1\}$ or $\perp$
- Error $\epsilon = 2^{-W} = 1/\text{quasipoly}(n)$

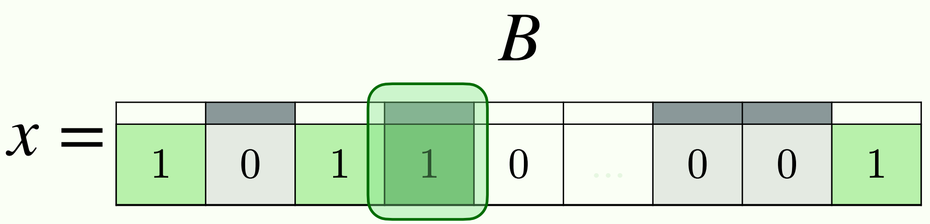


- Start with $x_0 = 1^n$

Faux-deterministic construction for FIND1

Attempt 1: non-adaptive approach

- Choose a random block $B \subset [n]$ of width $|B| = w = \text{polylog}(n)$
- Given x, query $\{x_i : i \in B\}$
- Output $B(x) = \{i \in B : x_i = 1\}$ or $\perp$
- Error $\epsilon = 2^{-W} = 1/\text{quasipoly}(n)$



- Start with $x_0 = 1^n$
- Flip the index $x \leftarrow x^{B(x)}$

Faux-deterministic construction for FIND1

Attempt 1: non-adaptive approach

- Choose a random block $B \subset [n]$ of width $|B| = w = \text{polylog}(n)$
- Given x, query $\{x_i : i \in B\}$
- Output $B(x) = \{i \in B : x_i = 1\}$ or $\perp$
- Error $\epsilon = 2^{-W} = 1/\text{quasipoly}(n)$

B

$x =$

1	0	1	1	0	...	0	0	1

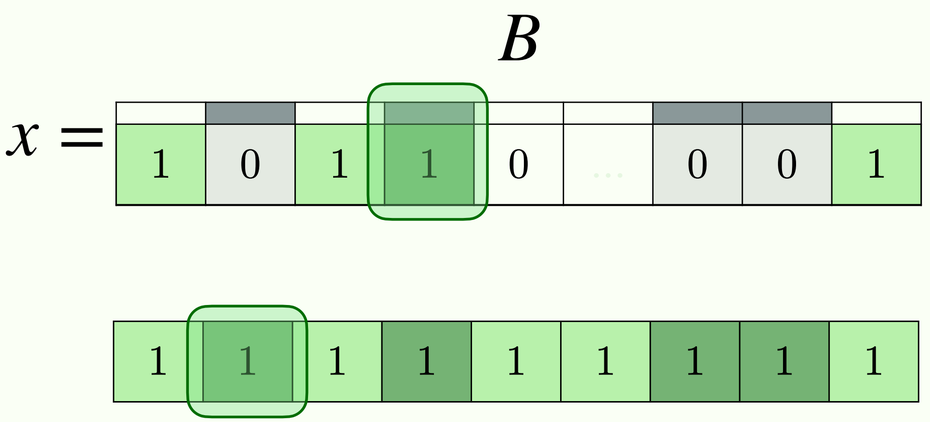
1	1	1	1	1	1	1	1	1

- Start with $x_0 = 1^n$
- Flip the index $x \leftarrow x^{B(x)}$

Faux-deterministic construction for FIND1

Attempt 1: non-adaptive approach

- Choose a random block $B \subset [n]$ of width $|B| = w = \text{polylog}(n)$
- Given x, query $\{x_i : i \in B\}$
- Output $B(x) = \{i \in B : x_i = 1\}$ or $\perp$
- Error $\epsilon = 2^{-W} = 1/\text{quasipoly}(n)$



- Start with $x_0 = 1^n$
- Flip the index $x \leftarrow x^{B(x)}$

Faux-deterministic construction for FIND1

Attempt 1: non-adaptive approach

- Choose a random block $B \subset [n]$ of width $|B| = w = \text{polylog}(n)$
- Given x, query $\{x_i : i \in B\}$
- Output $B(x) = \{i \in B : x_i = 1\}$ or $\perp$
- Error $\epsilon = 2^{-W} = 1/\text{quasipoly}(n)$

B

$x =$

1	0	1	1	0	...	0	0	1

1	1	1	1	1	1	1	1	1

- Start with $x_0 = 1^n$
- Flip the index $x \leftarrow x^{B(x)}$
- Until we find $x_i = \begin{cases} 1 & i \notin B \\ 0 & i \in B \end{cases}$

Faux-deterministic construction for FIND1

Attempt 1: non-adaptive approach

- Choose a random block $B \subset [n]$ of width $|B| = w = \text{polylog}(n)$
- Given x, query $\{x_i : i \in B\}$
- Output $B(x) = \{i \in B : x_i = 1\}$ or $\perp$
- Error $\epsilon = 2^{-W} = 1/\text{quasipoly}(n)$

B

$x =$

1	0	1	1	0	...	0	0	1

1	1	1	1	1	1	1	1	1

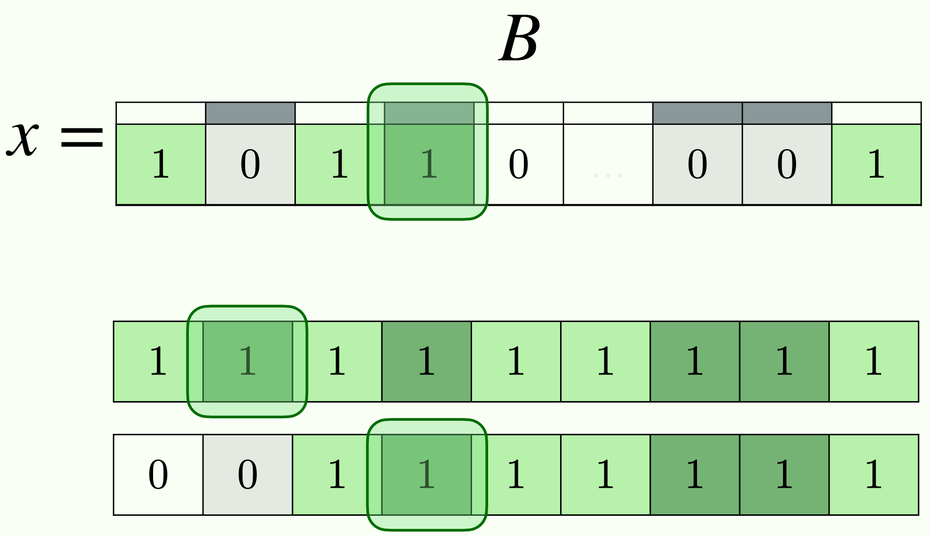
0	0	1	1	1	1	1	1	1

- Start with $x_0 = 1^n$
- Flip the index $x \leftarrow x^{B(x)}$
- Until we find $x_i = \begin{cases} 1 & i \notin B \\ 0 & i \in B \end{cases}$

Faux-deterministic construction for FIND1

Attempt 1: non-adaptive approach

- Choose a random block $B \subset [n]$ of width $|B| = w = \text{polylog}(n)$
- Given x, query $\{x_i : i \in B\}$
- Output $B(x) = \{i \in B : x_i = 1\}$ or $\perp$
- Error $\epsilon = 2^{-W} = 1/\text{quasipoly}(n)$

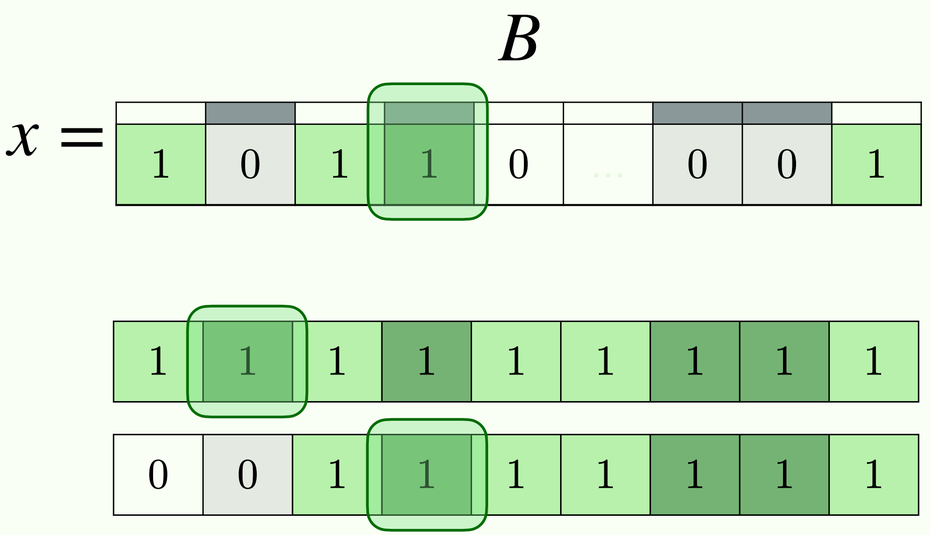


- Start with $x_0 = 1^n$
- Flip the index $x \leftarrow x^{B(x)}$
- Until we find $x_i = \begin{cases} 1 & i \notin B \\ 0 & i \in B \end{cases}$

Faux-deterministic construction for FIND1

Attempt 1: non-adaptive approach

- Choose a random block $B \subset [n]$ of width $|B| = w = \text{polylog}(n)$
- Given x, query $\{x_i : i \in B\}$
- Output $B(x) = \{i \in B : x_i = 1\}$ or $\perp$
- Error $\epsilon = 2^{-W} = 1/\text{quasipoly}(n)$

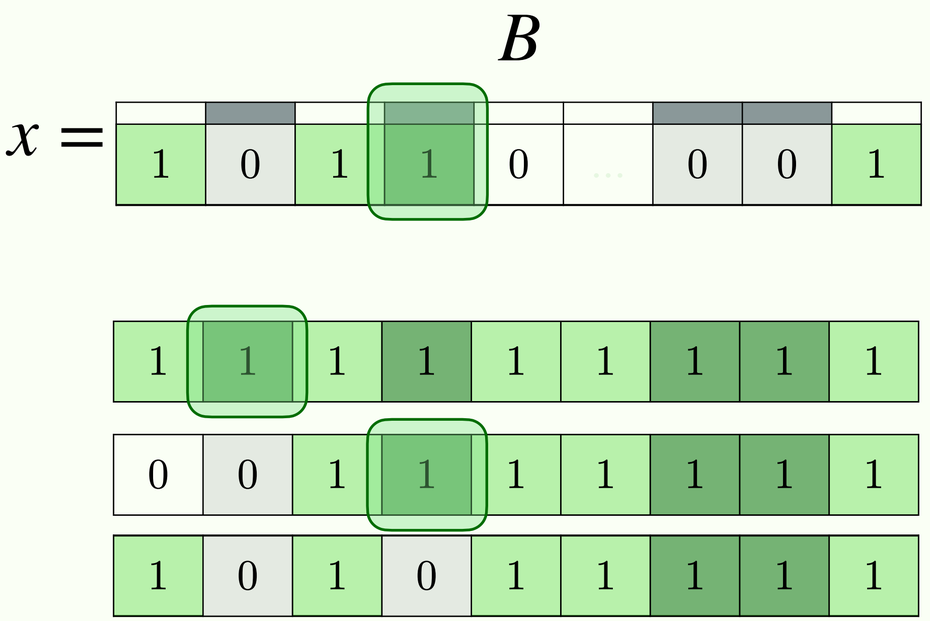


- Start with $x_0 = 1^n$
- Flip the index $x \leftarrow x^{B(x)}$
- Until we find $x_i = \begin{cases} 1 & i \notin B \\ 0 & i \in B \end{cases}$
- w queries at most

Faux-deterministic construction for FIND1

Attempt 1: non-adaptive approach

- Choose a random block $B \subset [n]$ of width $|B| = w = \text{polylog}(n)$
- Given x, query $\{x_i : i \in B\}$
- Output $B(x) = \{i \in B : x_i = 1\}$ or $\perp$
- Error $\epsilon = 2^{-W} = 1/\text{quasipoly}(n)$

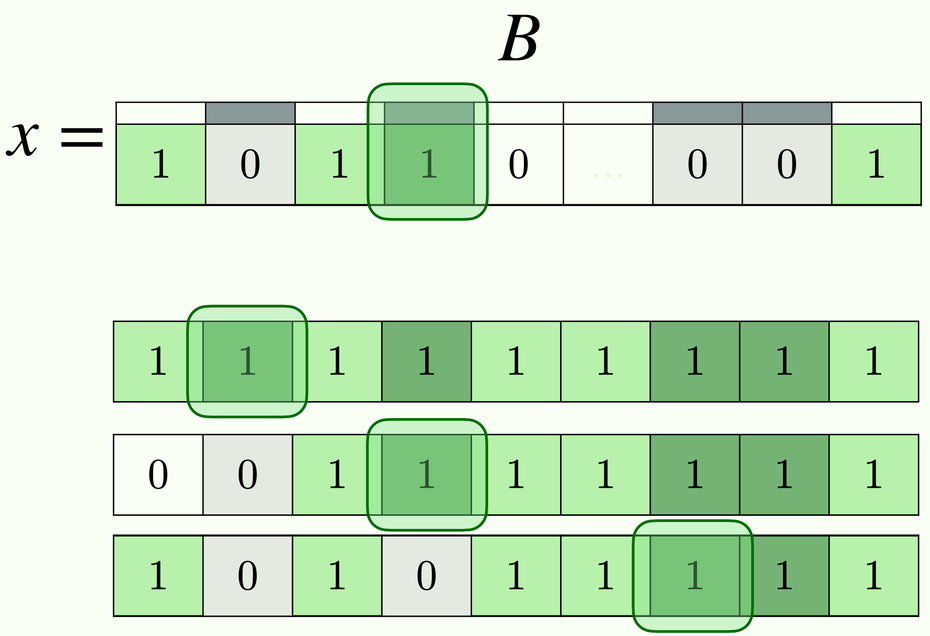


- Start with $x_0 = 1^n$
- Flip the index $x \leftarrow x^{B(x)}$
- Until we find $x_i = \begin{cases} 1 & i \notin B \\ 0 & i \in B \end{cases}$
- w queries at most

Faux-deterministic construction for FIND1

Attempt 1: non-adaptive approach

- Choose a random block $B \subset [n]$ of width $|B| = w = \text{polylog}(n)$
- Given x, query $\{x_i : i \in B\}$
- Output $B(x) = \{i \in B : x_i = 1\}$ or $\perp$
- Error $\epsilon = 2^{-W} = 1/\text{quasipoly}(n)$



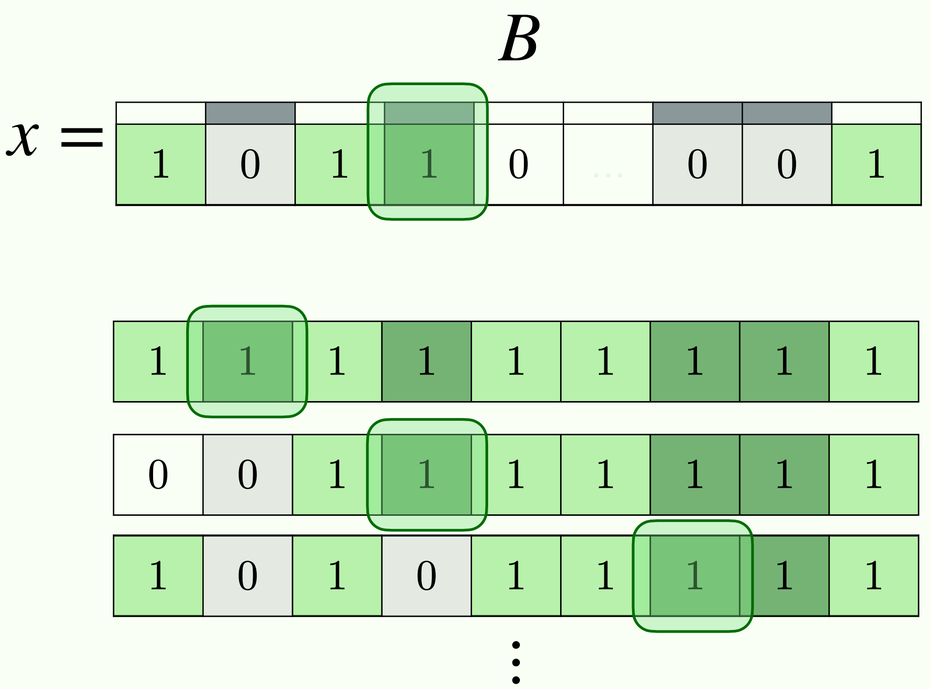
- Start with $x_0 = 1^n$
- Flip the index $x \leftarrow x^{B(x)}$
- Until we find $x_i = \begin{cases} 1 & i \notin B \\ 0 & i \in B \end{cases}$
- w queries at most

Faux-deterministic construction for FIND1

Attempt 1: non-adaptive approach

- Choose a random block $B \subset [n]$ of width $|B| = w = \text{polylog}(n)$
- Given x, query $\{x_i : i \in B\}$
- Output $B(x) = \{i \in B : x_i = 1\}$ or $\perp$
- Error $\epsilon = 2^{-W} = 1/\text{quasipoly}(n)$

- Start with $x_0 = 1^n$
- Flip the index $x \leftarrow x^{B(x)}$
- Until we find $x_i = \begin{cases} 1 & i \notin B \\ 0 & i \in B \end{cases}$
- w queries at most

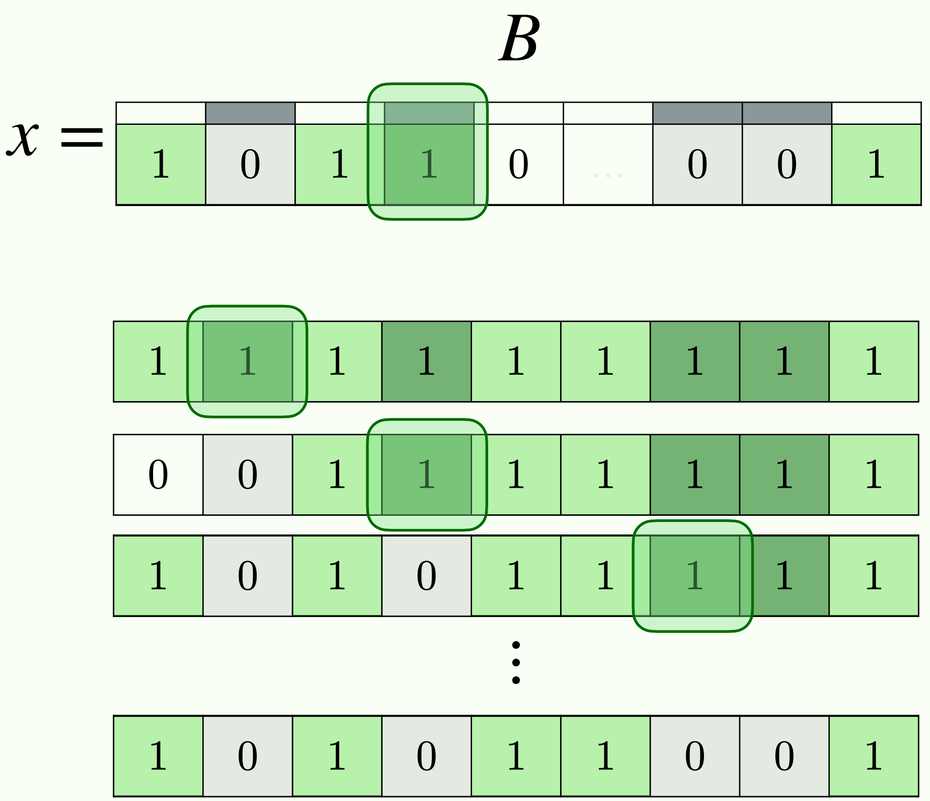


Faux-deterministic construction for FIND1

Attempt 1: non-adaptive approach

- Choose a random block $B \subset [n]$ of width $|B| = w = \text{polylog}(n)$
- Given x, query $\{x_i : i \in B\}$
- Output $B(x) = \{i \in B : x_i = 1\}$ or $\perp$
- Error $\epsilon = 2^{-W} = 1/\text{quasipoly}(n)$

- Start with $x_0 = 1^n$
- Flip the index $x \leftarrow x^{B(x)}$
- Until we find $x_i = \begin{cases} 1 & i \notin B \\ 0 & i \in B \end{cases}$
- w queries at most

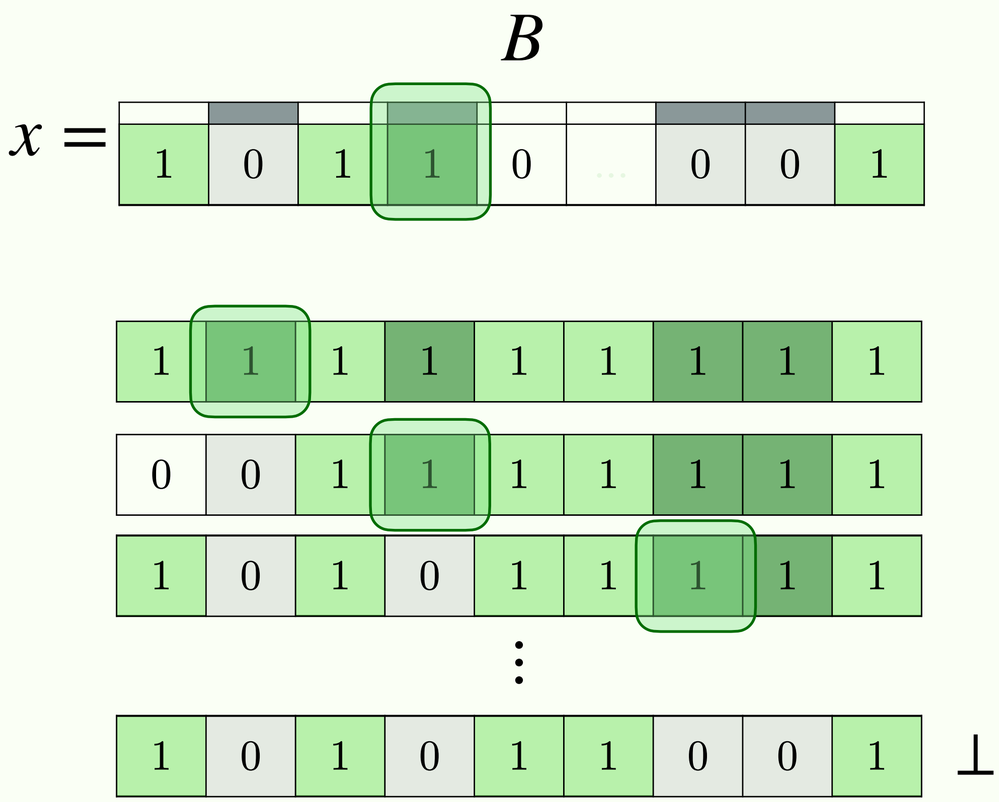


Faux-deterministic construction for FIND1

Attempt 1: non-adaptive approach

- Choose a random block $B \subset [n]$ of width $|B| = w = \text{polylog}(n)$
- Given x, query $\{x_i : i \in B\}$
- Output $B(x) = \{i \in B : x_i = 1\}$ or $\perp$
- Error $\epsilon = 2^{-W} = 1/\text{quasipoly}(n)$

- Start with $x_0 = 1^n$
- Flip the index $x \leftarrow x^{B(x)}$
- Until we find $x_i = \begin{cases} 1 & i \notin B \\ 0 & i \in B \end{cases}$
- w queries at most



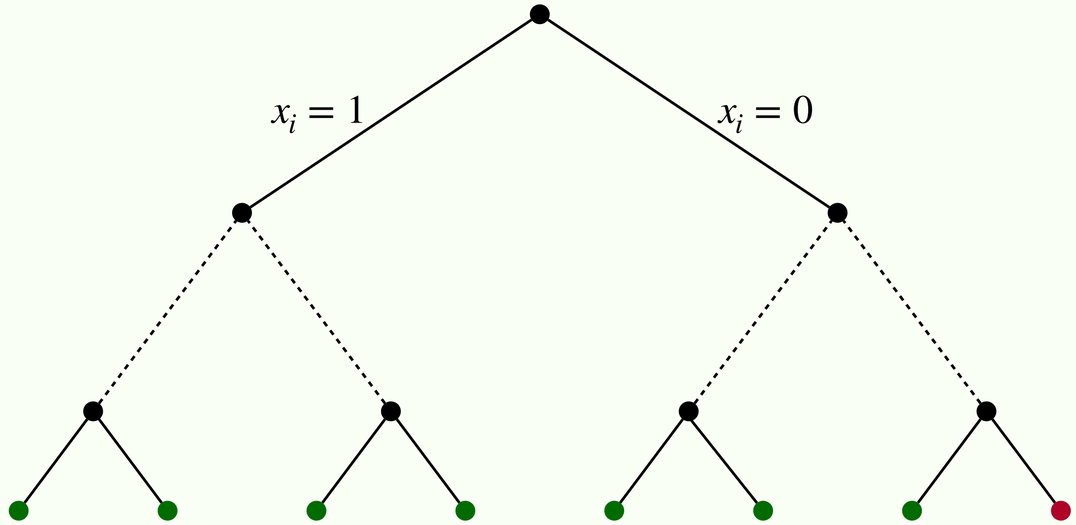
Faux-deterministic construction for FIND1

Attempt 2: adaptive approach

Faux-deterministic construction for FIND1

Attempt 2: adaptive approach

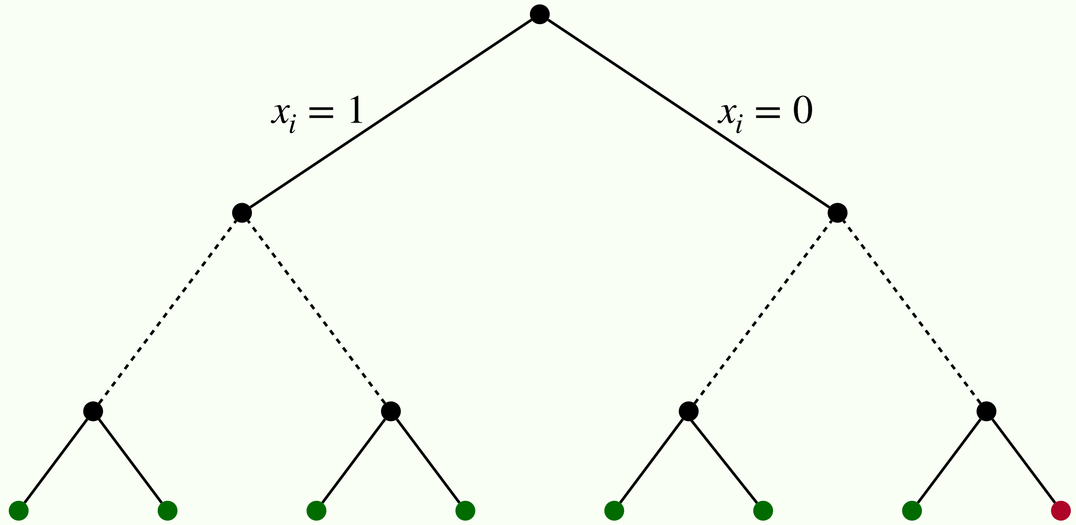
- choose a random decision tree T of depth $d = \text{polylog(n)}$



Faux-deterministic construction for FIND1

Attempt 2: adaptive approach

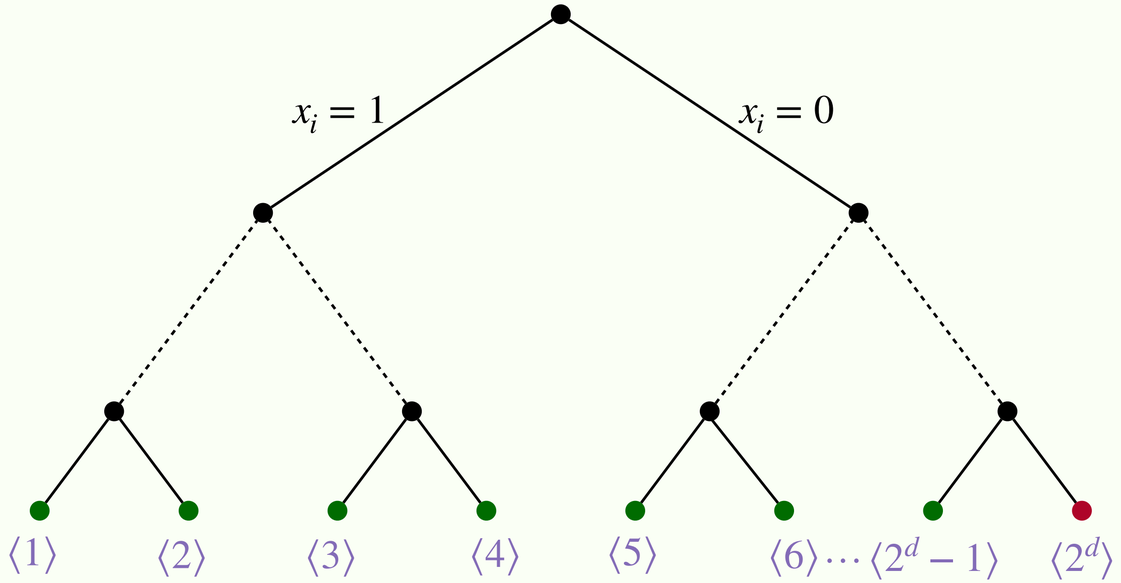
- choose a random decision tree T of depth $d = \text{polylog(n)}$
- run $T[x]$ to reach leaf $\langle \ell \rangle$



Faux-deterministic construction for FIND1

Attempt 2: adaptive approach

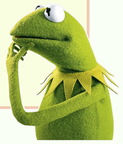
- choose a random decision tree T of depth $d = \text{polylog(n)}$
- run $T[x]$ to reach leaf $\langle \ell \rangle$

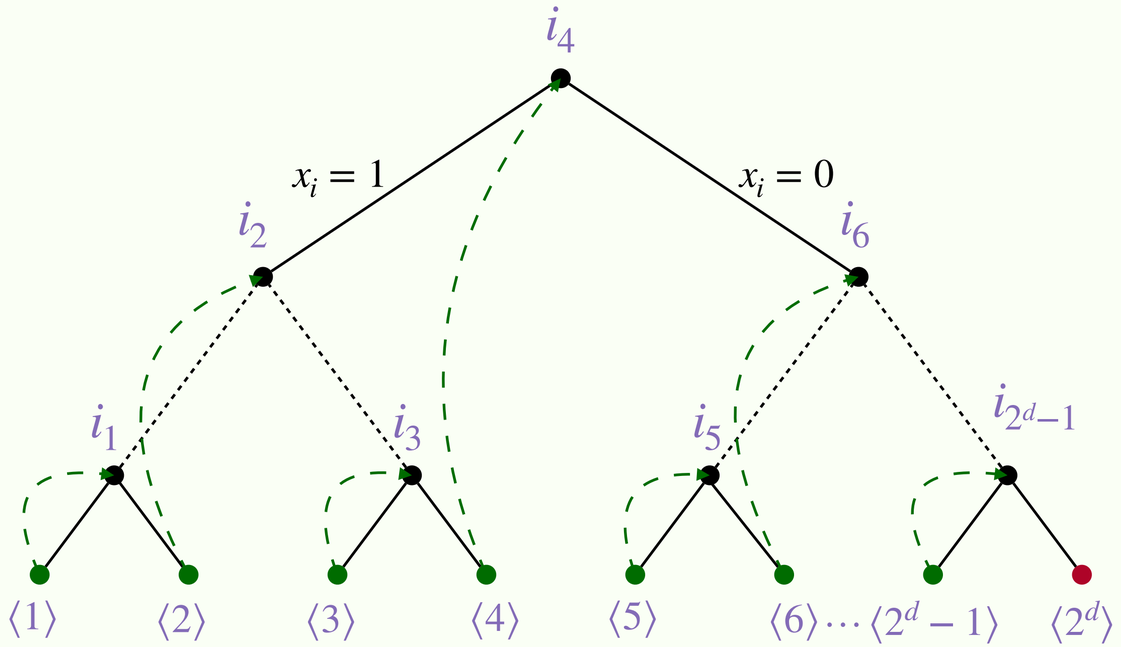


Faux-deterministic construction for FIND1

Attempt 2: adaptive approach

- choose a random decision tree T of depth $d = \text{polylog(n)}$
- run $T[x]$ to reach leaf $\langle \ell \rangle$

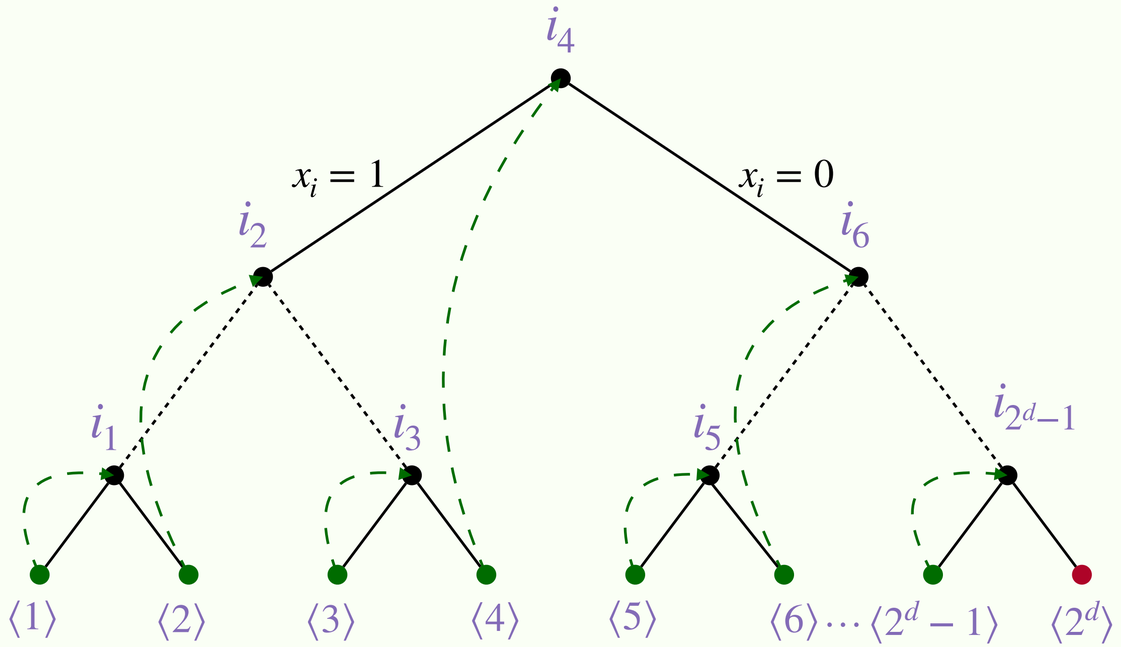




Faux-deterministic construction for FIND1

Attempt 2: adaptive approach

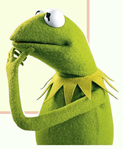
- choose a random decision tree T of depth $d = \text{polylog(n)}$
- run $T[x]$ to reach leaf $\langle \ell \rangle$
- output the last found 1-index: i_ℓ

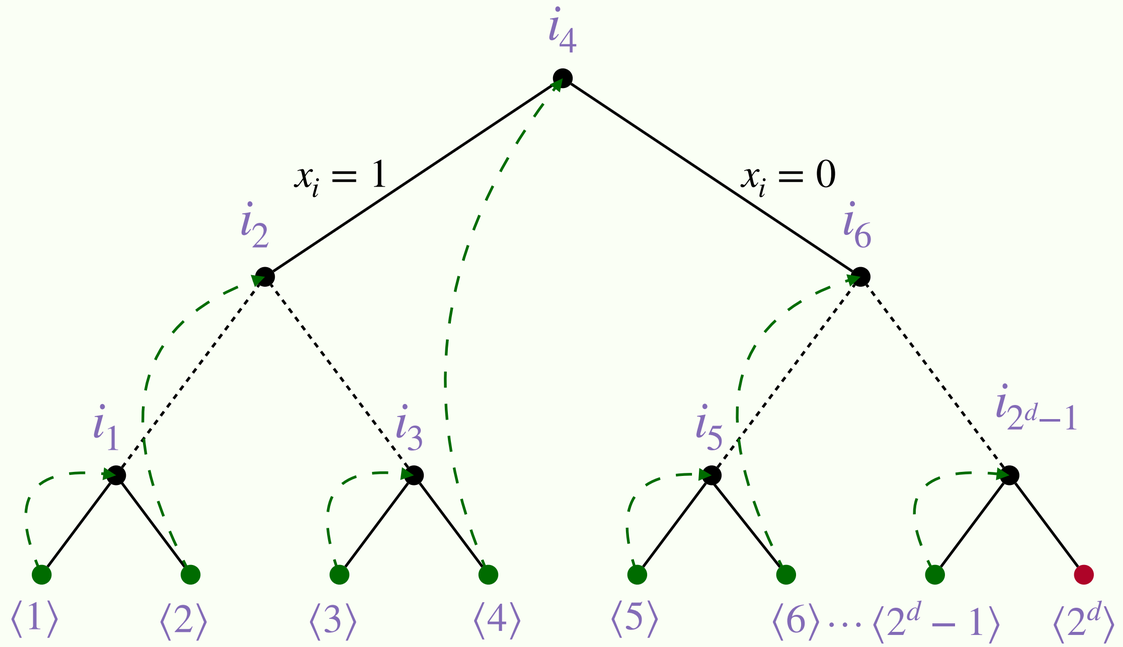


Faux-deterministic construction for FIND1

Attempt 2: adaptive approach

- choose a random decision tree T of depth $d = \text{polylog(n)}$
- run $T[x]$ to reach leaf $\langle \ell \rangle$
- output the last found 1-index: i_ℓ
- $\epsilon = 2^{-d} = 1/\text{quasipoly}(n)$

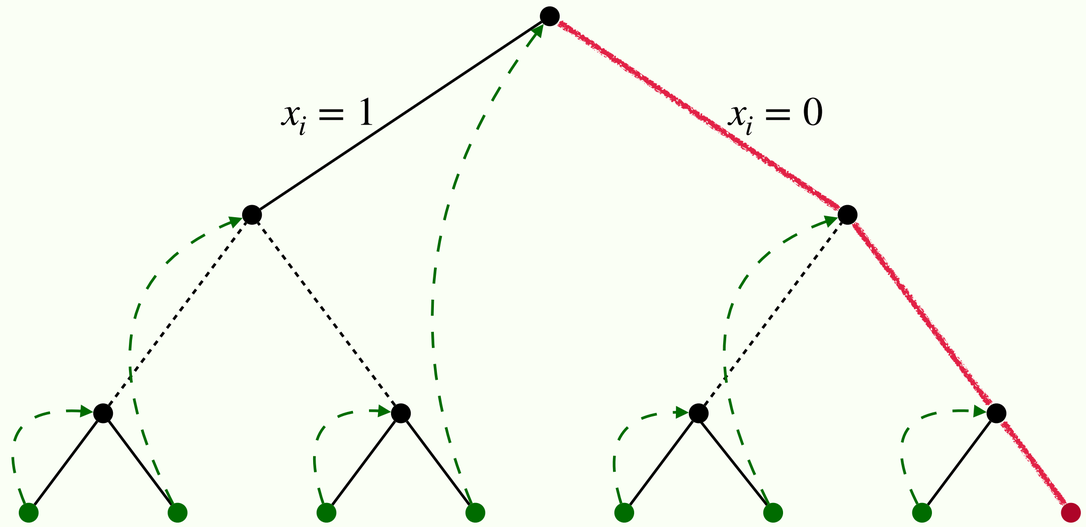




Faux-deterministic construction for FIND1

Attempt 2: adaptive approach

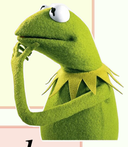
- choose a random decision tree T of depth $d = \text{polylog(n)}$
- run $T[x]$ to reach leaf $\langle \ell \rangle$
- output the last found 1-index: i_ℓ
- $\epsilon = 2^{-d} = 1/\text{quasipoly}(n)$



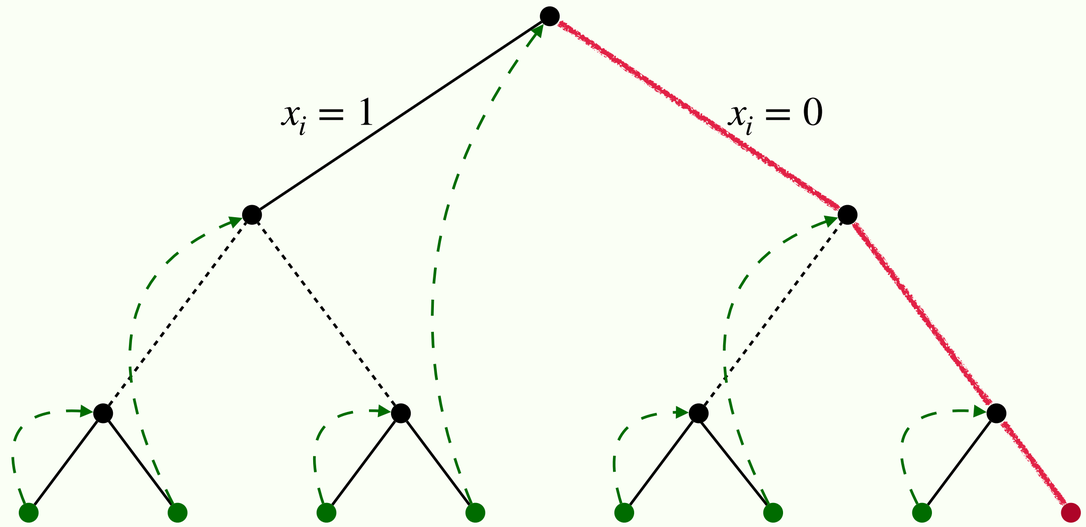
Faux-deterministic construction for FIND1

Attempt 2: adaptive approach

- choose a random decision tree T of depth $d = \text{polylog(n)}$
- run $T[x]$ to reach leaf $\langle \ell \rangle$
- output the last found 1-index: i_ℓ
- $\epsilon = 2^{-d} = 1/\text{quasipoly}(n)$



- Goal: find the all 0-path: input with leaf $\langle 2^d \rangle$
- Previous attack fails... requires 2^d queries...

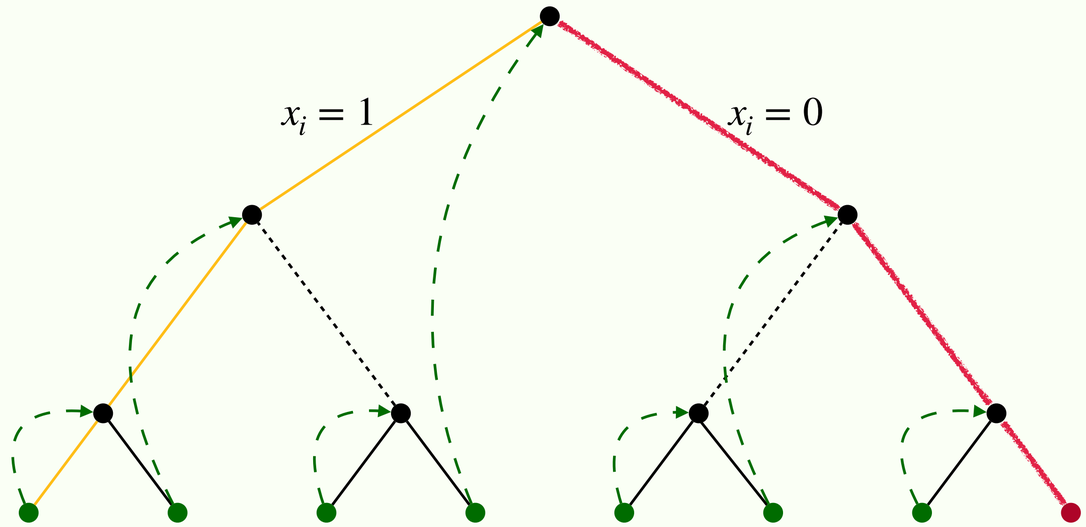


Faux-deterministic construction for FIND1

Attempt 2: adaptive approach

- choose a random decision tree T of depth $d = \text{polylog(n)}$
- run $T[x]$ to reach leaf $\langle \ell \rangle$
- output the last found 1-index: i_ℓ
- $\epsilon = 2^{-d} = 1/\text{quasipoly}(n)$

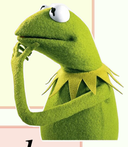
- Goal: find the all 0-path: input with leaf $\langle 2^d \rangle$
- Previous attack fails... requires 2^d queries...



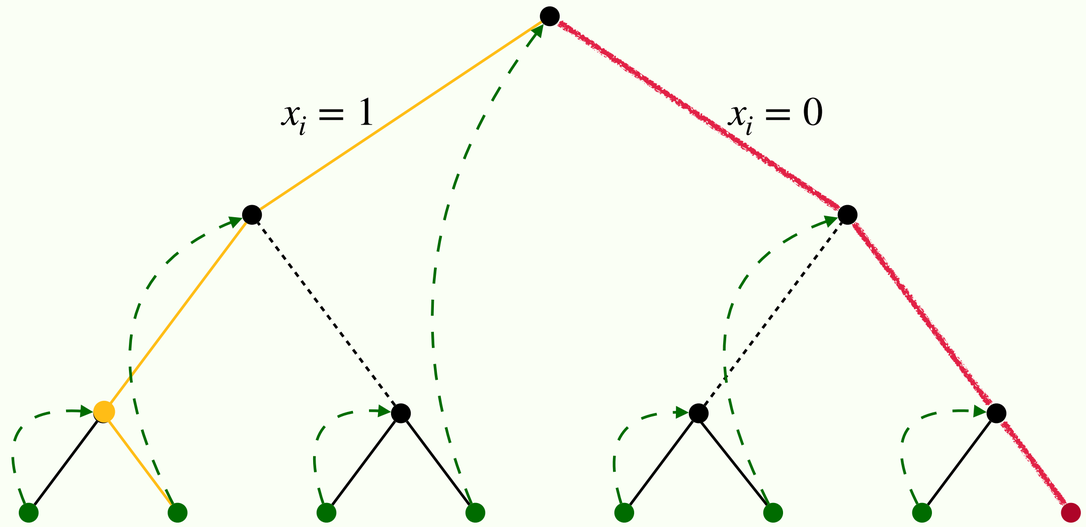
Faux-deterministic construction for FIND1

Attempt 2: adaptive approach

- choose a random decision tree T of depth $d = \text{polylog(n)}$
- run $T[x]$ to reach leaf $\langle \ell \rangle$
- output the last found 1-index: i_ℓ
- $\epsilon = 2^{-d} = 1/\text{quasipoly}(n)$



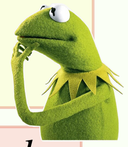
- Goal: find the all 0-path: input with leaf $\langle 2^d \rangle$
- Previous attack fails... requires 2^d queries...



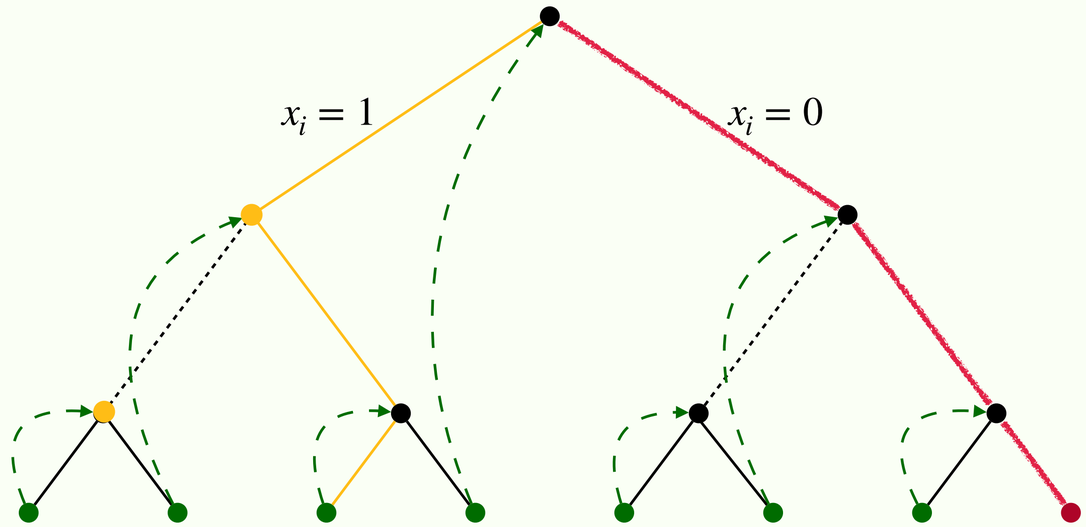
Faux-deterministic construction for FIND1

Attempt 2: adaptive approach

- choose a random decision tree T of depth $d = \text{polylog(n)}$
- run $T[x]$ to reach leaf $\langle \ell \rangle$
- output the last found 1-index: i_ℓ
- $\epsilon = 2^{-d} = 1/\text{quasipoly}(n)$



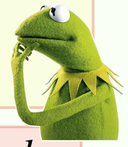
- Goal: find the all 0-path: input with leaf $\langle 2^d \rangle$
- Previous attack fails... requires 2^d queries...



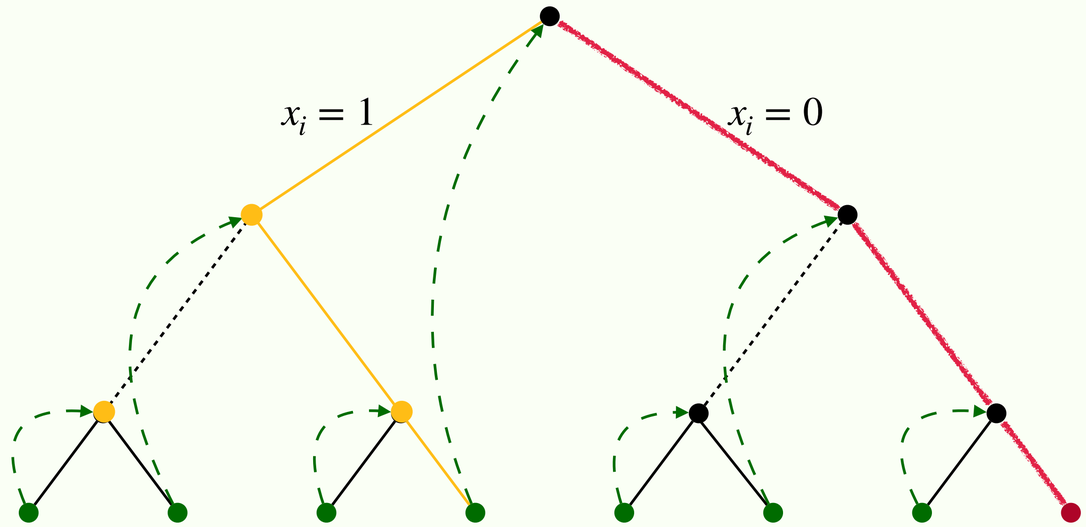
Faux-deterministic construction for FIND1

Attempt 2: adaptive approach

- choose a random decision tree T of depth $d = \text{polylog(n)}$
- run $T[x]$ to reach leaf $\langle \ell \rangle$
- output the last found 1-index: i_ℓ
- $\epsilon = 2^{-d} = 1/\text{quasipoly}(n)$



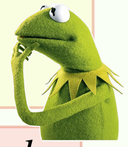
- Goal: find the all 0-path: input with leaf $\langle 2^d \rangle$
- Previous attack fails... requires 2^d queries...



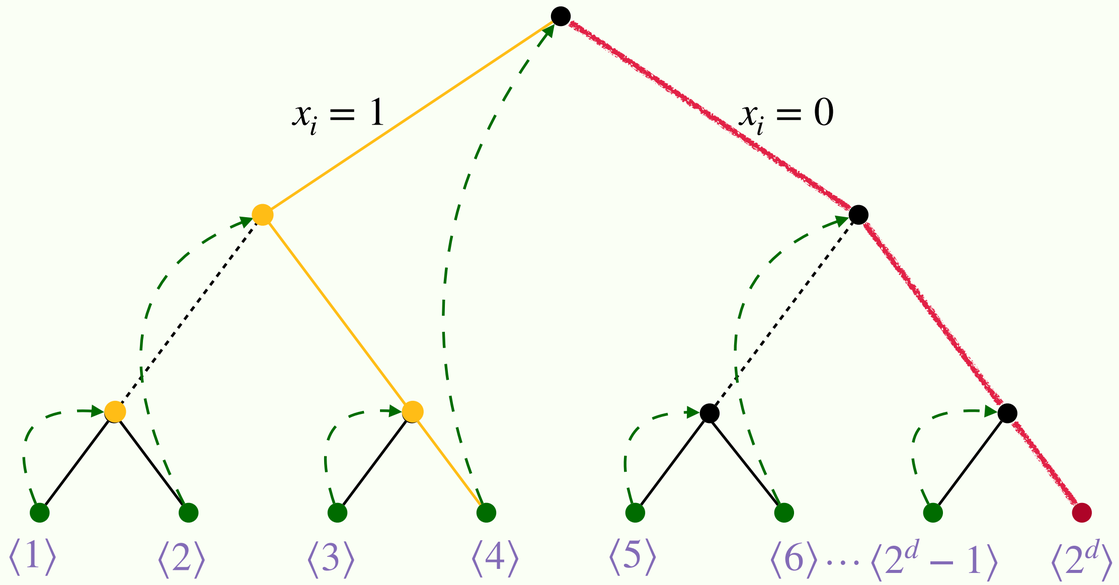
Faux-deterministic construction for FIND1

Attempt 2: adaptive approach

- choose a random decision tree T of depth $d = \text{polylog(n)}$
- run $T[x]$ to reach leaf $\langle \ell \rangle$
- output the last found 1-index: i_ℓ
- $\epsilon = 2^{-d} = 1/\text{quasipoly}(n)$



- Goal: find the all 0-path: input with leaf $\langle 2^d \rangle$
- Previous attack fails... requires 2^d queries...

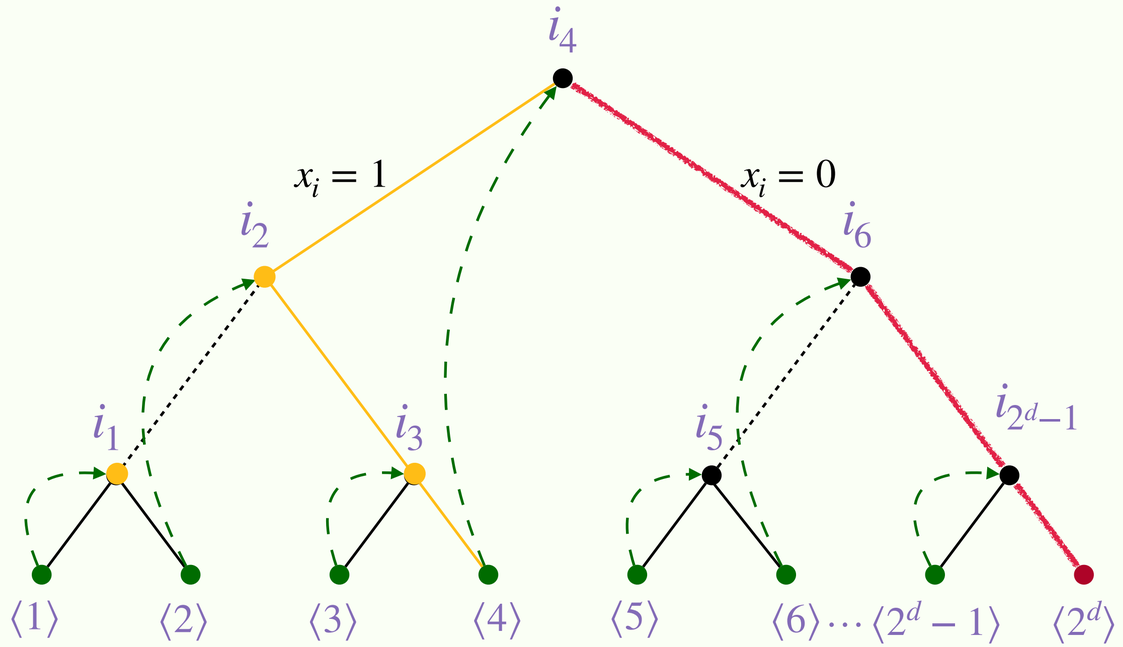


Faux-deterministic construction for FIND1

Attempt 2: adaptive approach

- choose a random decision tree T of depth $d = \text{polylog(n)}$
- run $T[x]$ to reach leaf $\langle \ell \rangle$
- output the last found 1-index: i_ℓ
- $\epsilon = 2^{-d} = 1/\text{quasipoly}(n)$

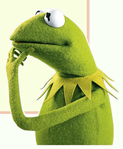
- Goal: find the all 0-path: input with leaf $\langle 2^d \rangle$
- Previous attack fails... requires 2^d queries...

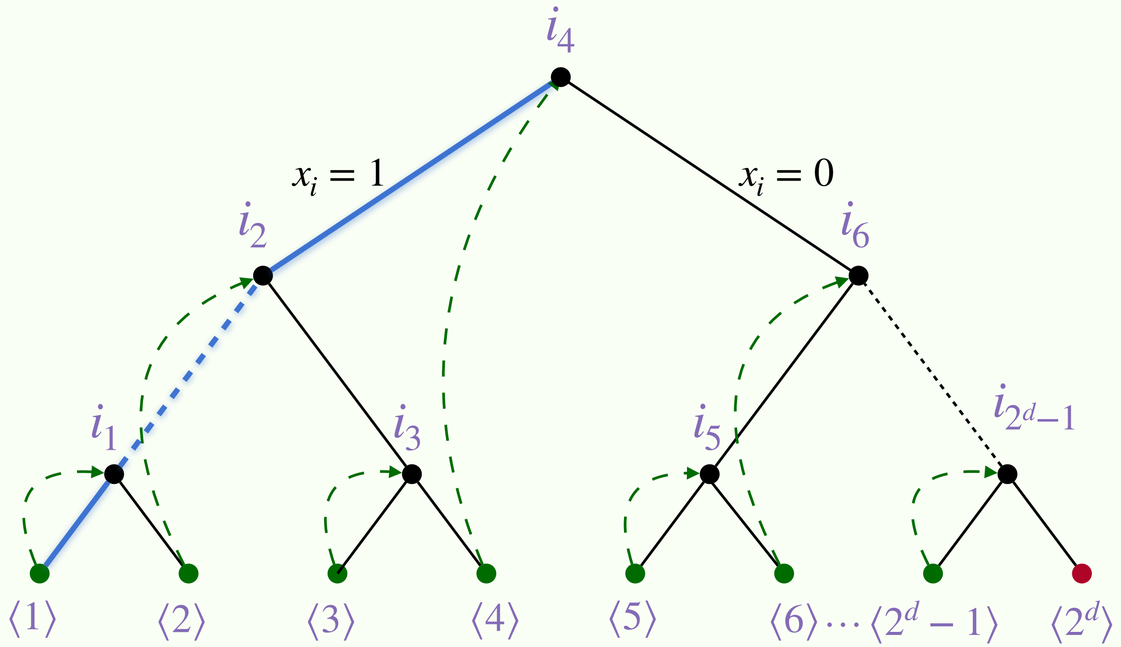


Faux-deterministic construction for FIND1

Attempt 2: adaptive approach

- choose a random decision tree T of depth $d = \text{polylog(n)}$
- run $T[x]$ to reach leaf $\langle \ell \rangle$
- output the last found 1-index: i_ℓ
- $\epsilon = 2^{-d} = 1/\text{quasipoly}(n)$



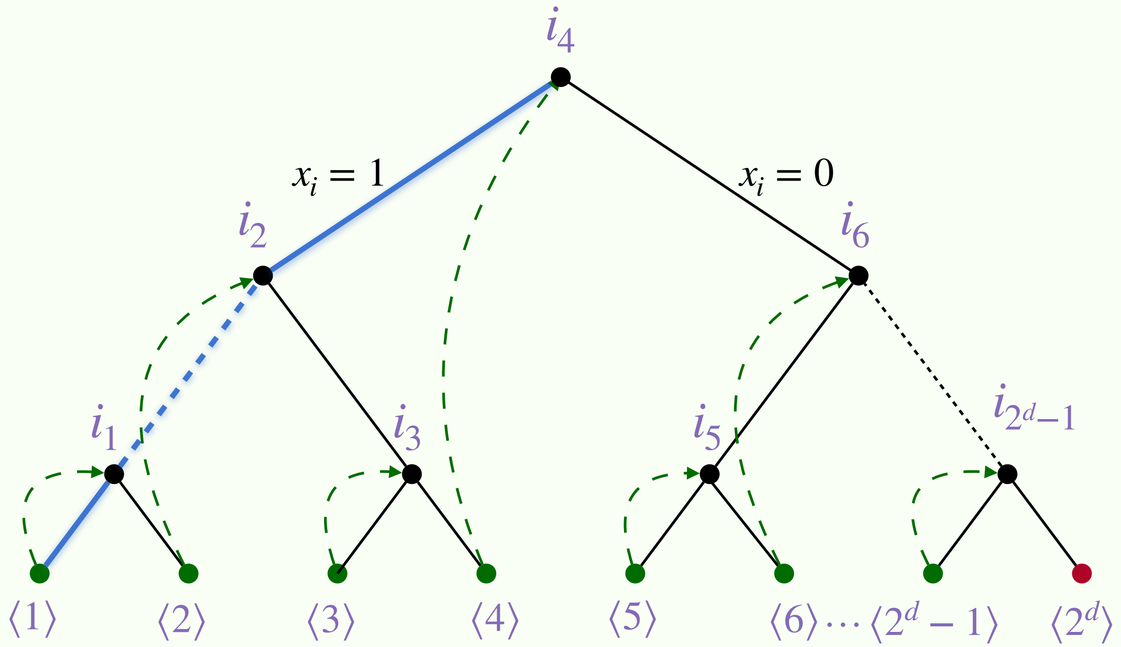


Faux-deterministic construction for FIND1

Attempt 2: adaptive approach

- choose a random decision tree T of depth $d = \text{polylog(n)}$
- run $T[x]$ to reach leaf $\langle \ell \rangle$
- output the last found 1-index: i_ℓ
- $\epsilon = 2^{-d} = 1/\text{quasipoly}(n)$

- Goal: find the all 0-path: input with leaf $\langle 2^d \rangle$
- Previous attack fails as it requires 2^d queries
- Comes with a huge disadvantage

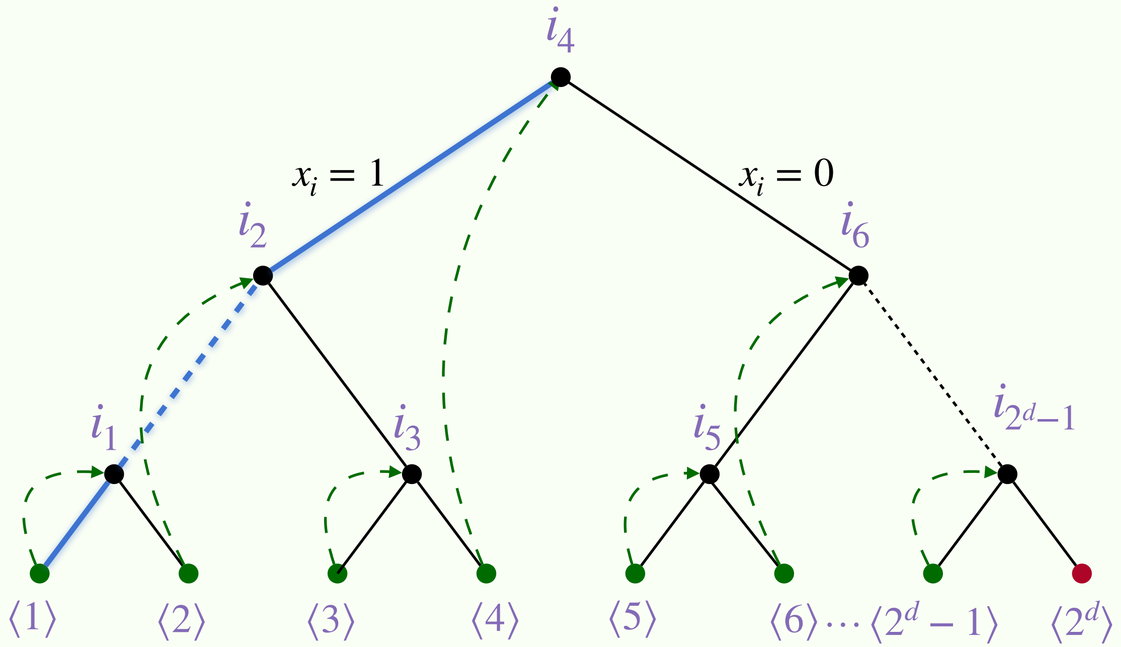


Faux-deterministic construction for FIND1

Attempt 2: adaptive approach

- choose a random decision tree T of depth $d = \text{polylog(n)}$
- run $T[x]$ to reach leaf $\langle \ell \rangle$
- output the last found 1-index: i_ℓ
- $\epsilon = 2^{-d} = 1/\text{quasipoly}(n)$

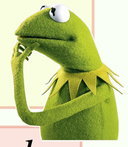
- Goal: find the all 0-path: input with leaf $\langle 2^d \rangle$
- Previous attack fails as it requires 2^d queries
- Comes with a huge disadvantage
- Queried indices are sensitive($f(x) \neq f(x^j)$) $\Rightarrow$ identifiable



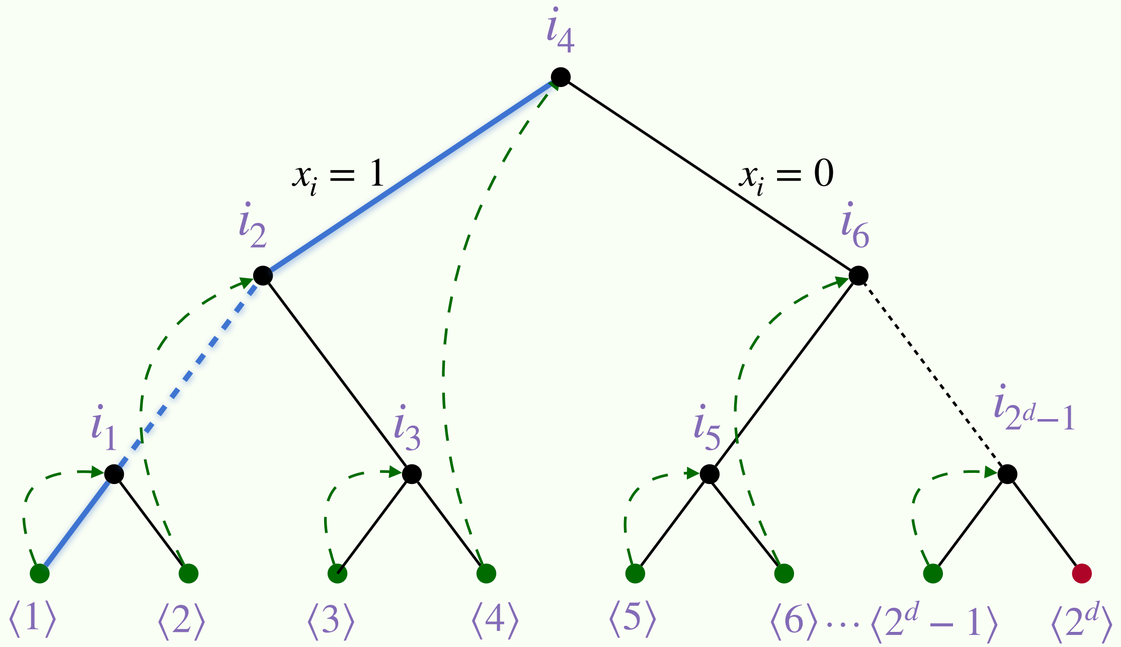
Faux-deterministic construction for FIND1

Attempt 2: adaptive approach

- choose a random decision tree T of depth $d = \text{polylog(n)}$
- run $T[x]$ to reach leaf $\langle \ell \rangle$
- output the last found 1-index: i_ℓ
- $\epsilon = 2^{-d} = 1/\text{quasipoly}(n)$



- Goal: find the all 0-path: input with leaf $\langle 2^d \rangle$
- Previous attack fails as it requires 2^d queries
- Comes with a huge disadvantage
- Queried indices are sensitive$(f(x) \neq f(x^j)) \Rightarrow$ identifiable
- Can obtain poly(n) query attack.

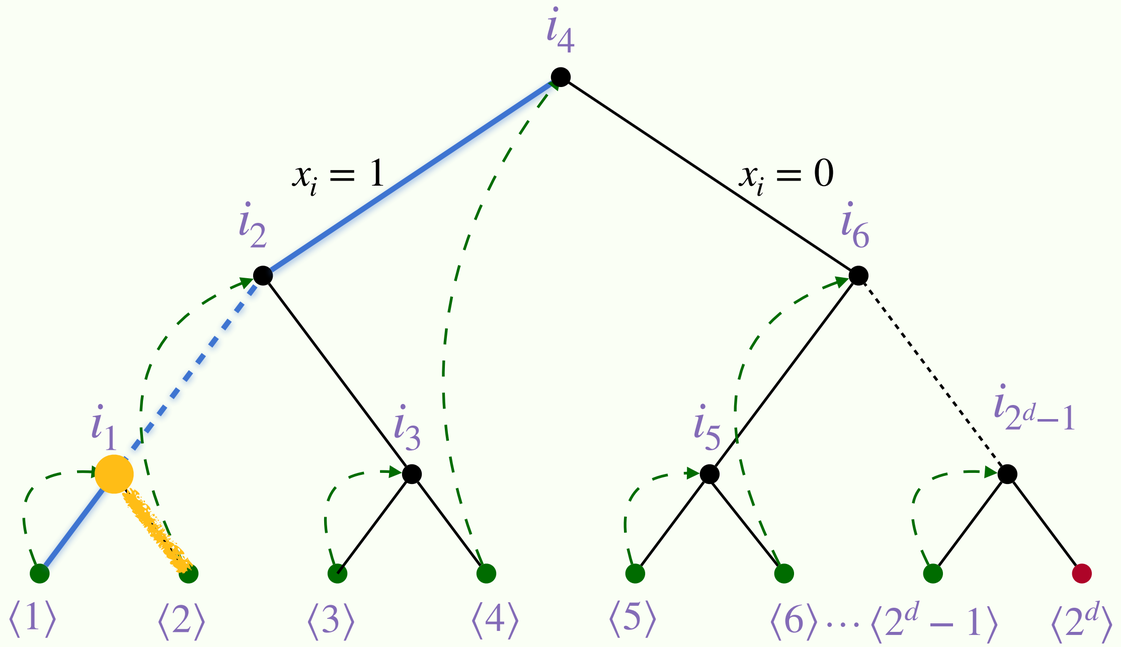


Faux-deterministic construction for FIND1

Attempt 2: adaptive approach

- choose a random decision tree T of depth $d = \text{polylog(n)}$
- run $T[x]$ to reach leaf $\langle \ell \rangle$
- output the last found 1-index: i_ℓ
- $\epsilon = 2^{-d} = 1/\text{quasipoly}(n)$

- Goal: find the all 0-path: input with leaf $\langle 2^d \rangle$
- Previous attack fails as it requires 2^d queries
- Comes with a huge disadvantage
- Queried indices are sensitive$(f(x) \neq f(x^j)) \Rightarrow$ identifiable
- Can obtain poly(n) query attack.

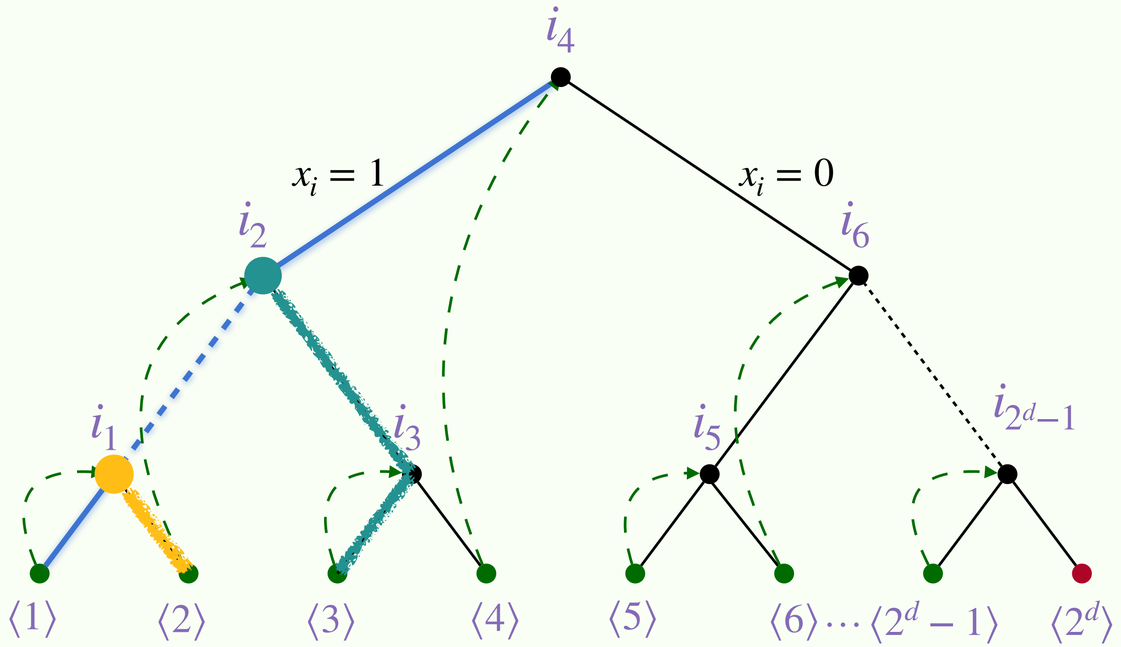


Faux-deterministic construction for FIND1

Attempt 2: adaptive approach

- choose a random decision tree T of depth $d = \text{polylog(n)}$
- run $T[x]$ to reach leaf $\langle \ell \rangle$
- output the last found 1-index: i_ℓ
- $\epsilon = 2^{-d} = 1/\text{quasipoly}(n)$

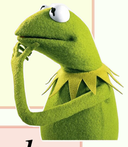
- Goal: find the all 0-path: input with leaf $\langle 2^d \rangle$
- Previous attack fails as it requires 2^d queries
- Comes with a huge disadvantage
- Queried indices are sensitive($f(x) \neq f(x^j)$) $\Rightarrow$ identifiable
- Can obtain poly(n) query attack.



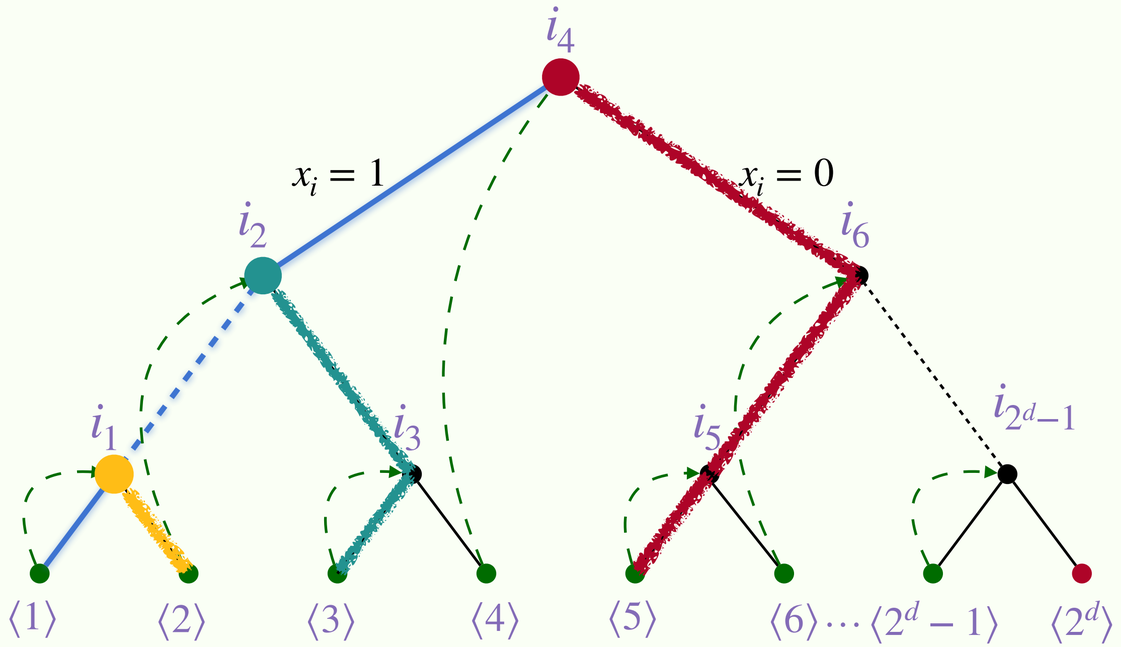
Faux-deterministic construction for FIND1

Attempt 2: adaptive approach

- choose a random decision tree T of depth $d = \text{polylog(n)}$
- run $T[x]$ to reach leaf $\langle \ell \rangle$
- output the last found 1-index: i_ℓ
- $\epsilon = 2^{-d} = 1/\text{quasipoly}(n)$



- Goal: find the all 0-path: input with leaf $\langle 2^d \rangle$
- Previous attack fails as it requires 2^d queries
- Comes with a huge disadvantage
- Queried indices are sensitive($f(x) \neq f(x^j)$) $\Rightarrow$ identifiable
- Can obtain poly(n) query attack.



Faux-deterministic construction for FIND1

The successful design: a mixed approach

Faux-deterministic construction for FIND1

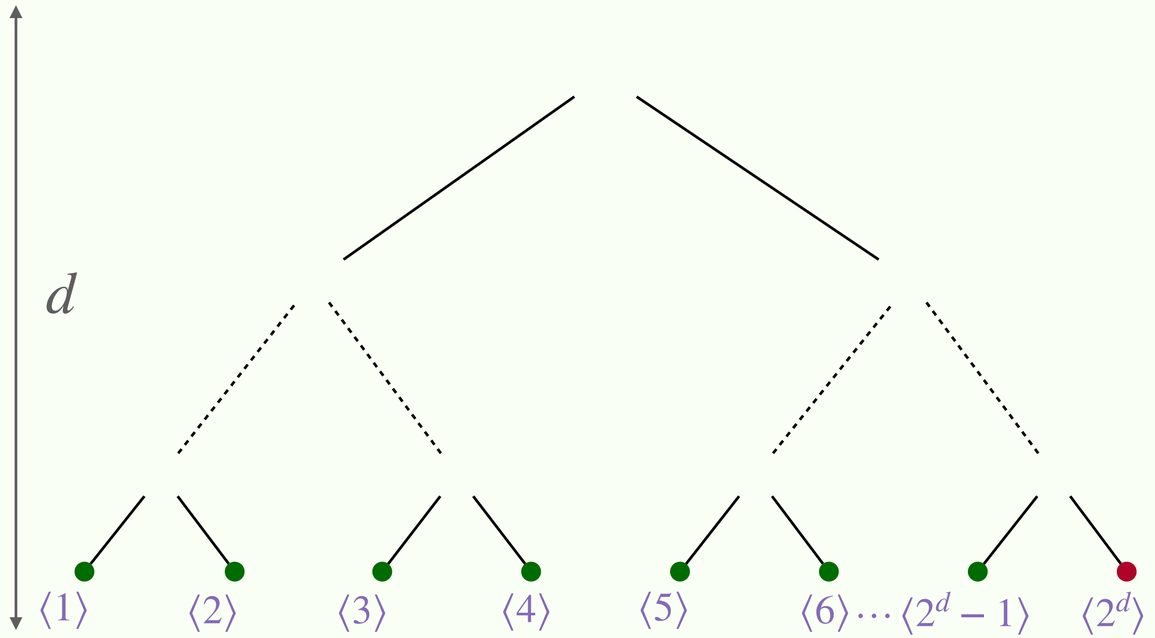
The successful design: a mixed approach

Let $q = \text{polylog}(n)$, and $w, d := \Theta(\sqrt{q})$

Faux-deterministic construction for FIND1

The successful design: a mixed approach

Let $q = \text{polylog}(n)$, and $w, d := \Theta(\sqrt{q})$

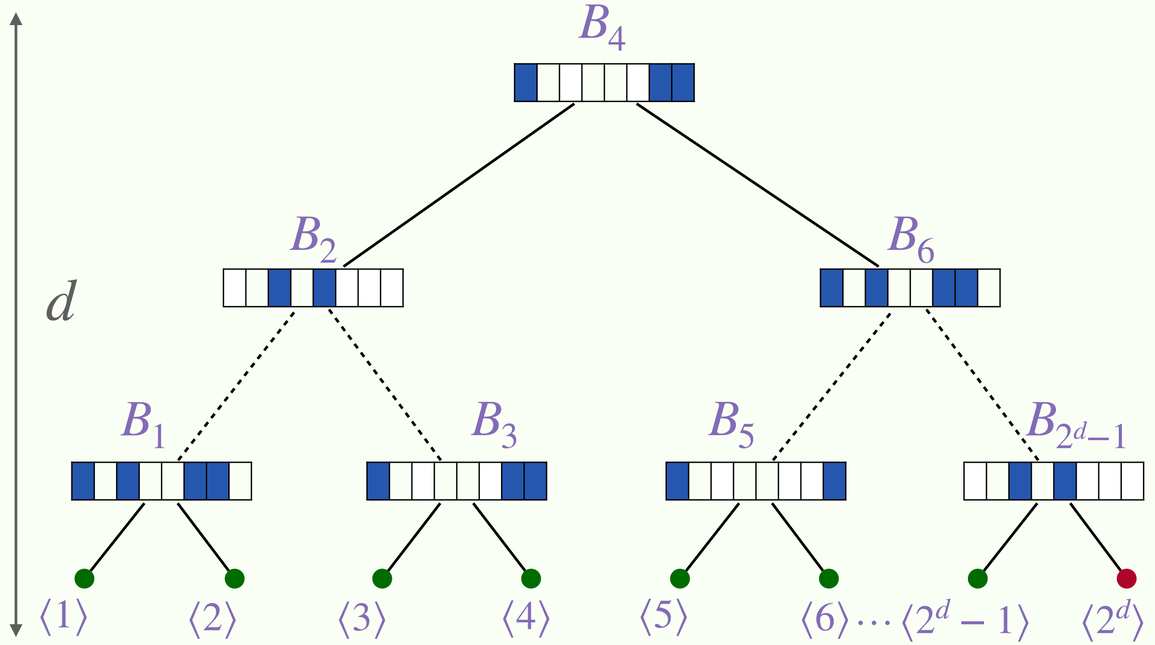


Faux-deterministic construction for FIND1

The successful design: a mixed approach

Let $q = \text{polylog}(n)$, and $w, d := \Theta(\sqrt{q})$

- Sample $2^d - 1$ random blocks $B_v \subset [n]$, $|B_v| = w$

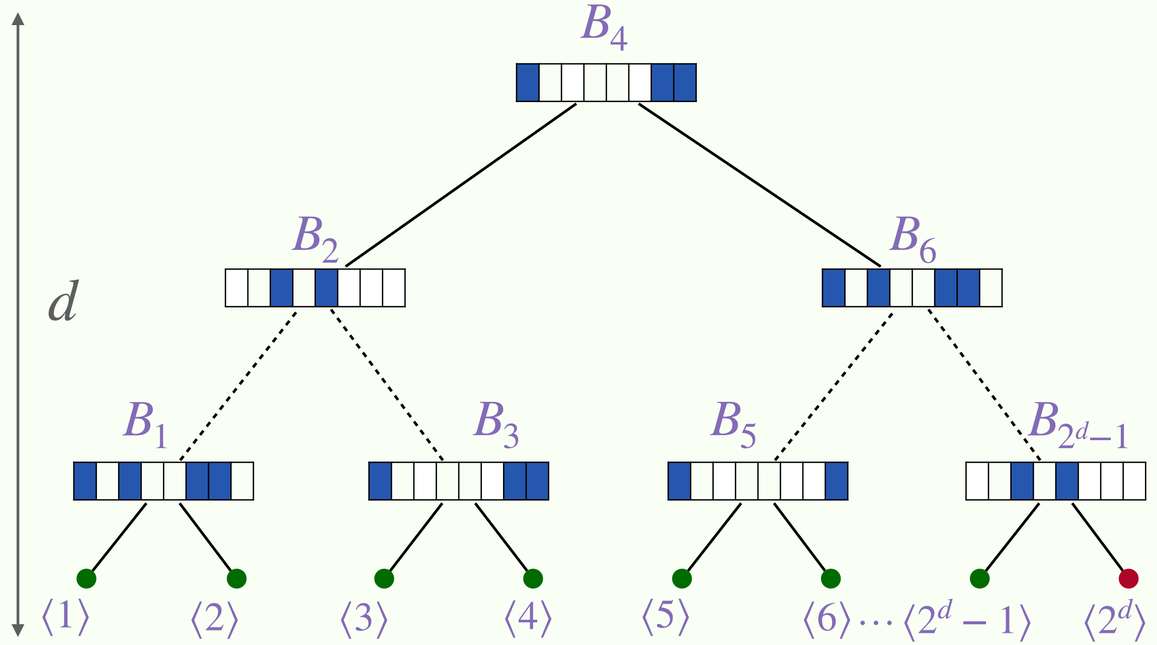


Faux-deterministic construction for FIND1

The successful design: a mixed approach

Let $q = \text{polylog}(n)$, and $w, d := \Theta(\sqrt{q})$

- Sample $2^d - 1$ random blocks $B_v \subset [n]$, $|B_v| = w$
- $B_v(x) = \{i \in B_v : x_i = 1\}$ or $\perp$

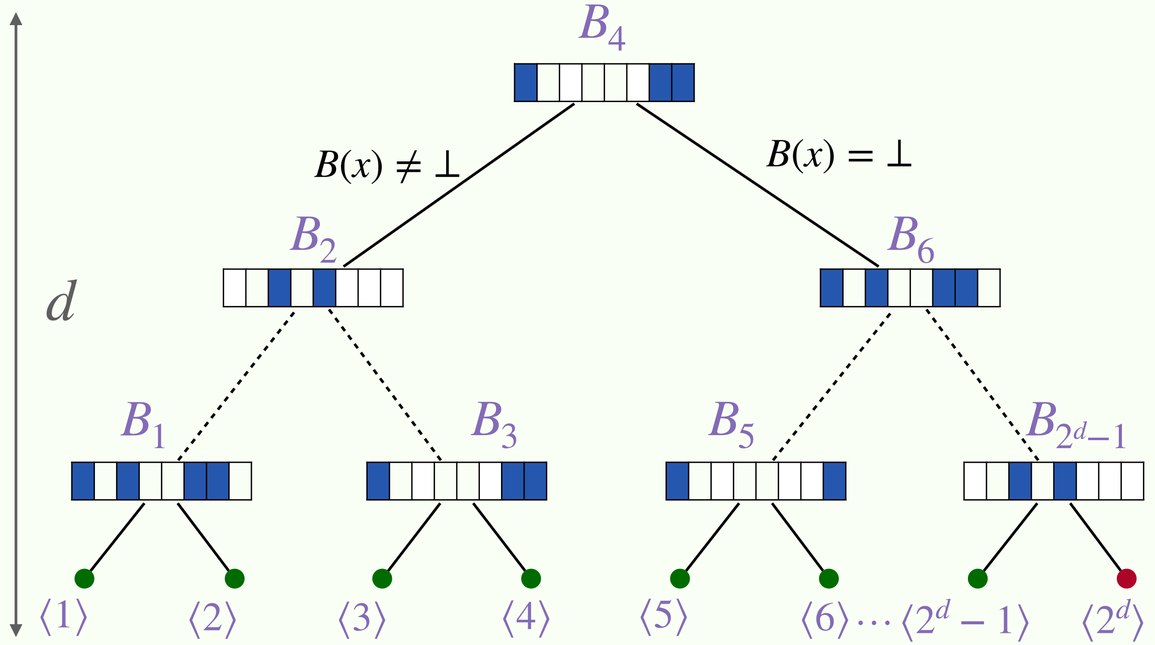


Faux-deterministic construction for FIND1

The successful design: a mixed approach

Let $q = \text{polylog}(n)$, and $w, d := \Theta(\sqrt{q})$

- Sample $2^d - 1$ random blocks $B_v \subset [n]$, $|B_v| = w$
- $B_v(x) = \{i \in B_v : x_i = 1\}$ or $\perp$
- Go left if $B_v(x) \neq \perp$ and right otherwise

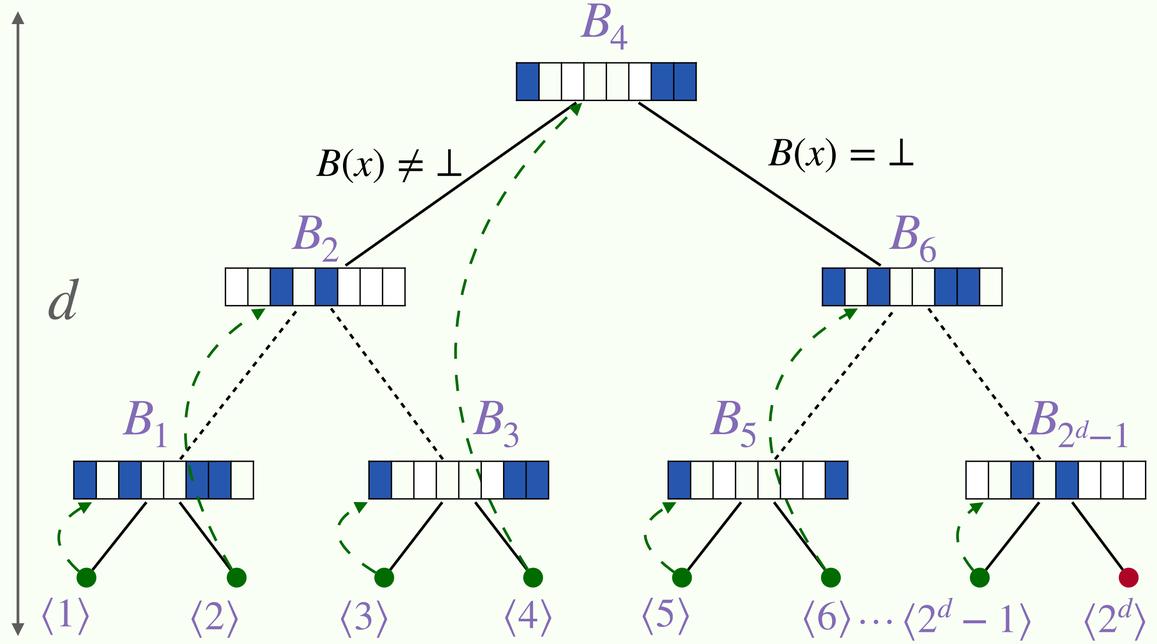


Faux-deterministic construction for FIND1

The successful design: a mixed approach

Let $q = \text{polylog}(n)$, and $w, d := \Theta(\sqrt{q})$

- Sample $2^d - 1$ random blocks $B_v \subset [n]$, $|B_v| = w$
- $B_v(x) = \{i \in B_v : x_i = 1\}$ or $\perp$
- Go left if $B_v(x) \neq \perp$ and right otherwise
- When reached leaf $\langle \ell \rangle$, output $B_\ell(x)$

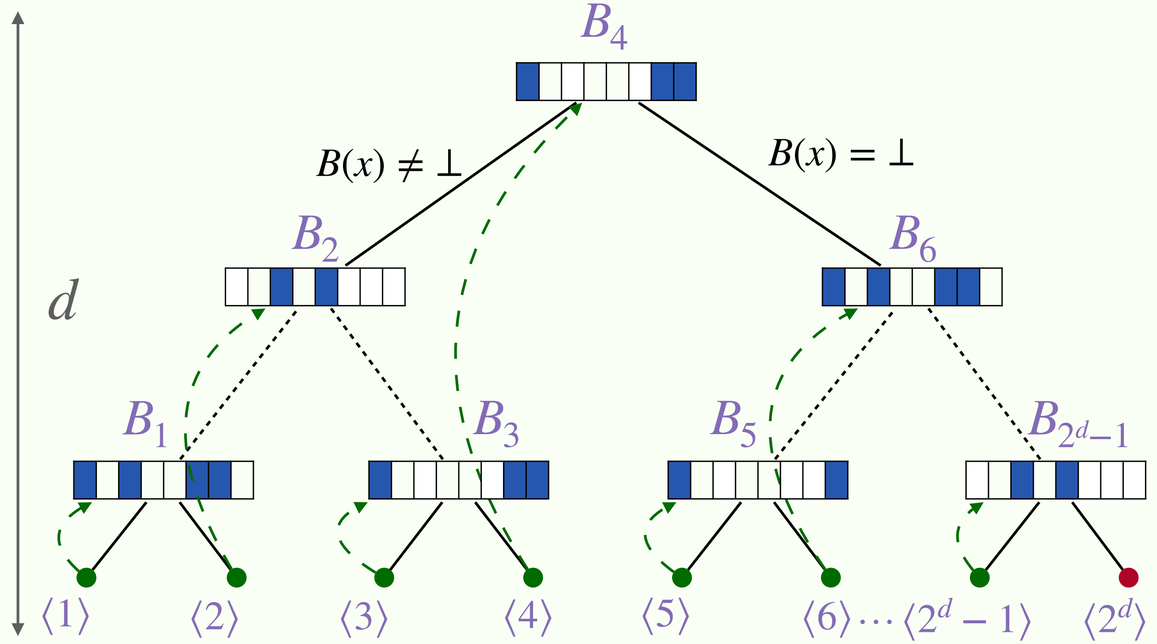


Faux-deterministic construction for FIND1

The successful design: a mixed approach

Let $q = \text{polylog}(n)$, and $w, d := \Theta(\sqrt{q})$

- Sample $2^d - 1$ random blocks $B_v \subset [n]$, $|B_v| = w$
- $B_v(x) = \{i \in B_v : x_i = 1\}$ or $\perp$
- Go left if $B_v(x) \neq \perp$ and right otherwise
- When reached leaf $\langle \ell \rangle$, output $B_\ell(x)$



Claim. Let $q = \text{polylog}(n)$, and $w, d := \Theta(\sqrt{q})$. Then, any successful adversary requires at least $2^{\Omega(\sqrt{q})} = \text{quasipoly(n)}$ many queries.

Proof idea. Cannot skip any of the blocks, as earlier queries are hidden in the $\vee$ blocks

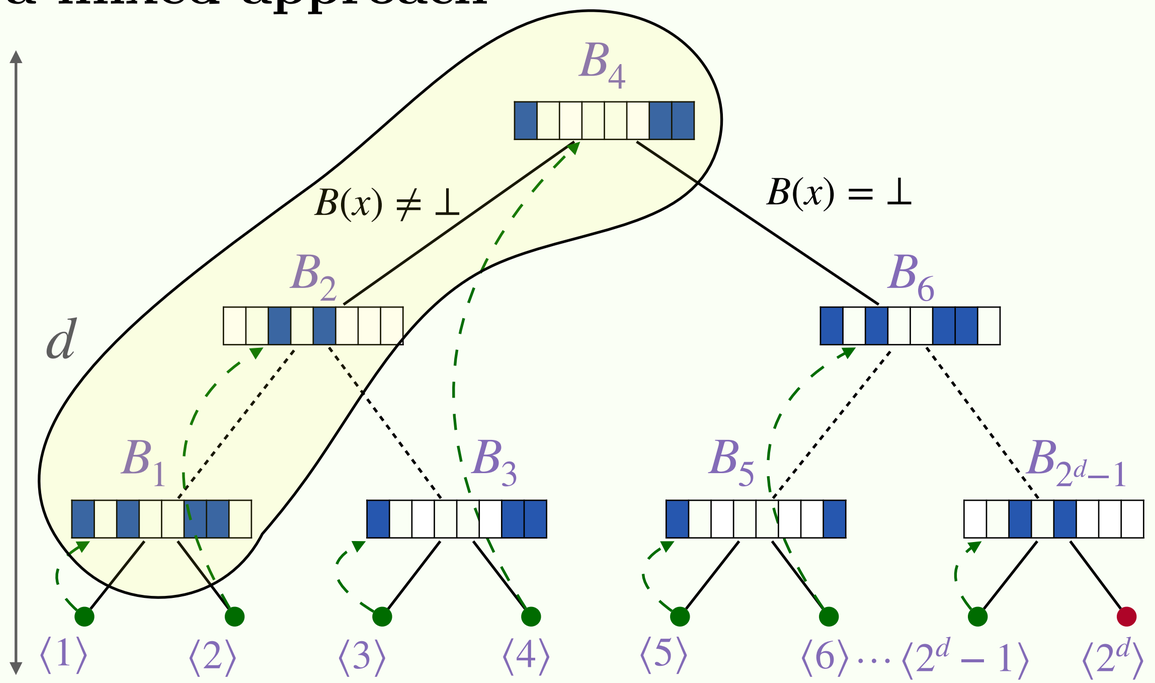
$\forall x, \Pr[x \text{ reaches } \langle \ell \rangle \mid B_1, \ldots, B_{\ell-1}] \geq (1 - 2^{-w})^d$

Faux-deterministic construction for FIND1

The successful design: a mixed approach

Let $q = \text{polylog}(n)$, and $w, d := \Theta(\sqrt{q})$

- Sample $2^d - 1$ random blocks $B_v \subset [n]$, $|B_v| = w$
- $B_v(x) = \{i \in B_v : x_i = 1\}$ or $\perp$
- Go left if $B_v(x) \neq \perp$ and right otherwise
- When reached leaf $\langle \ell \rangle$, output $B_\ell(x)$



Claim. Let $q = \text{polylog}(n)$, and $w, d := \Theta(\sqrt{q})$. Then, any successful adversary requires at least $2^{\Omega(\sqrt{q})} = \text{quasipoly(n)}$ many queries.

Proof idea. Cannot skip any of the blocks, as earlier queries are hidden in the $\vee$ blocks

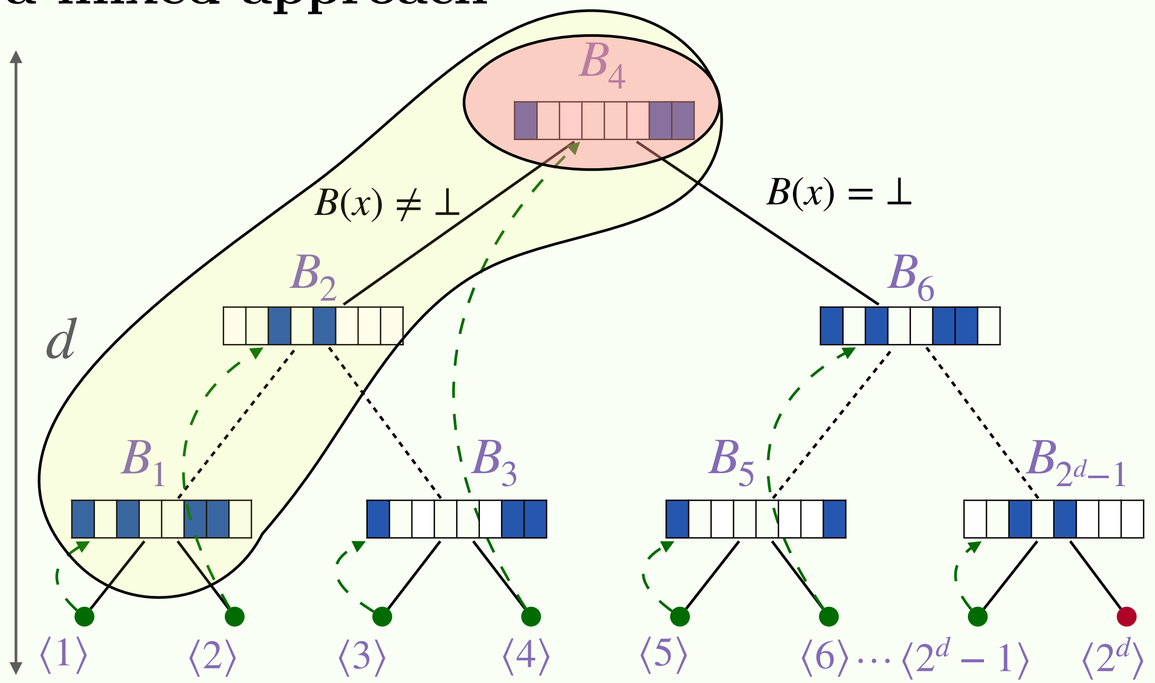
$\forall x, \Pr[x \text{ reaches } \langle \ell \rangle \mid B_1, \ldots, B_{\ell-1}] \geq (1 - 2^{-w})^d$

Faux-deterministic construction for FIND1

The successful design: a mixed approach

Let $q = \text{polylog}(n)$, and $w, d := \Theta(\sqrt{q})$

- Sample $2^d - 1$ random blocks $B_v \subset [n]$, $|B_v| = w$
- $B_v(x) = \{i \in B_v : x_i = 1\}$ or $\perp$
- Go left if $B_v(x) \neq \perp$ and right otherwise
- When reached leaf $\langle \ell \rangle$, output $B_\ell(x)$



Claim. Let $q = \text{polylog}(n)$, and $w, d := \Theta(\sqrt{q})$. Then, any successful adversary requires at least $2^{\Omega(\sqrt{q})} = \text{quasipoly(n)}$ many queries.

Proof idea. Cannot skip any of the blocks, as earlier queries are hidden in the $\vee$ blocks

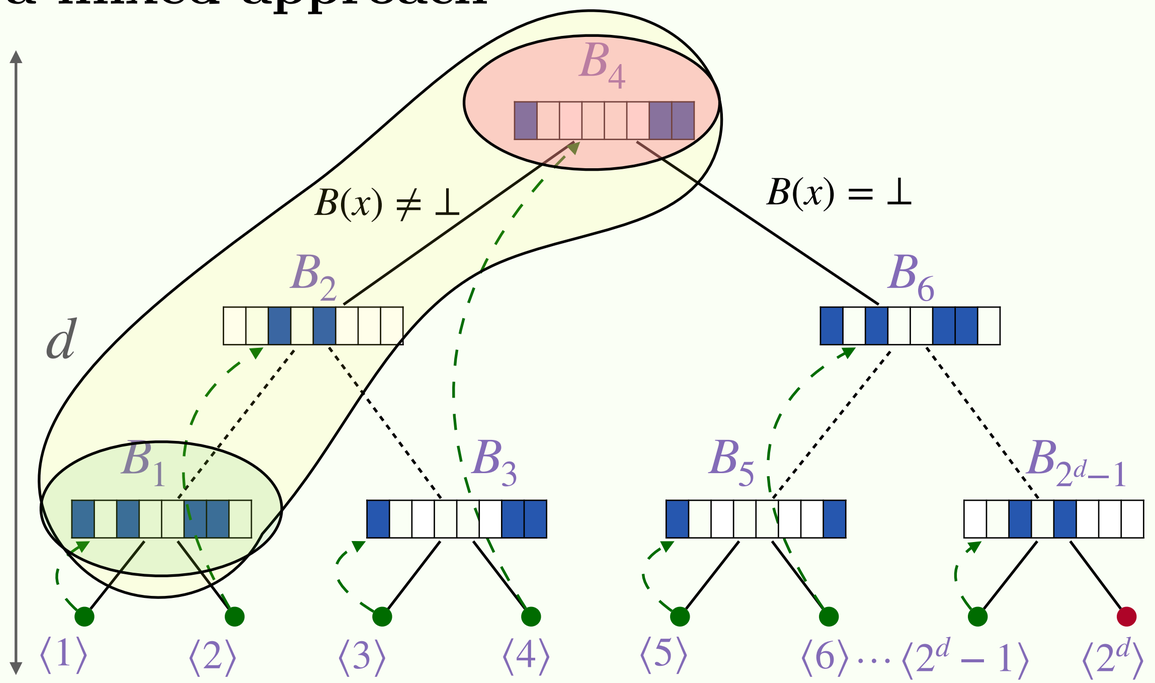
$\forall x, \Pr[x \text{ reaches } \langle \ell \rangle \mid B_1, \ldots, B_{\ell-1}] \geq (1 - 2^{-w})^d$

Faux-deterministic construction for FIND1

The successful design: a mixed approach

Let $q = \text{polylog}(n)$, and $w, d := \Theta(\sqrt{q})$

- Sample $2^d - 1$ random blocks $B_v \subset [n]$, $|B_v| = w$
- $B_v(x) = \{i \in B_v : x_i = 1\}$ or $\perp$
- Go left if $B_v(x) \neq \perp$ and right otherwise
- When reached leaf $\langle \ell \rangle$, output $B_\ell(x)$



Claim. Let $q = \text{polylog}(n)$, and $w, d := \Theta(\sqrt{q})$. Then, any successful adversary requires at least $2^{\Omega(\sqrt{q})} = \text{quasipoly(n)}$ many queries.

Proof idea. Cannot skip any of the blocks, as earlier queries are hidden in the $\vee$ blocks

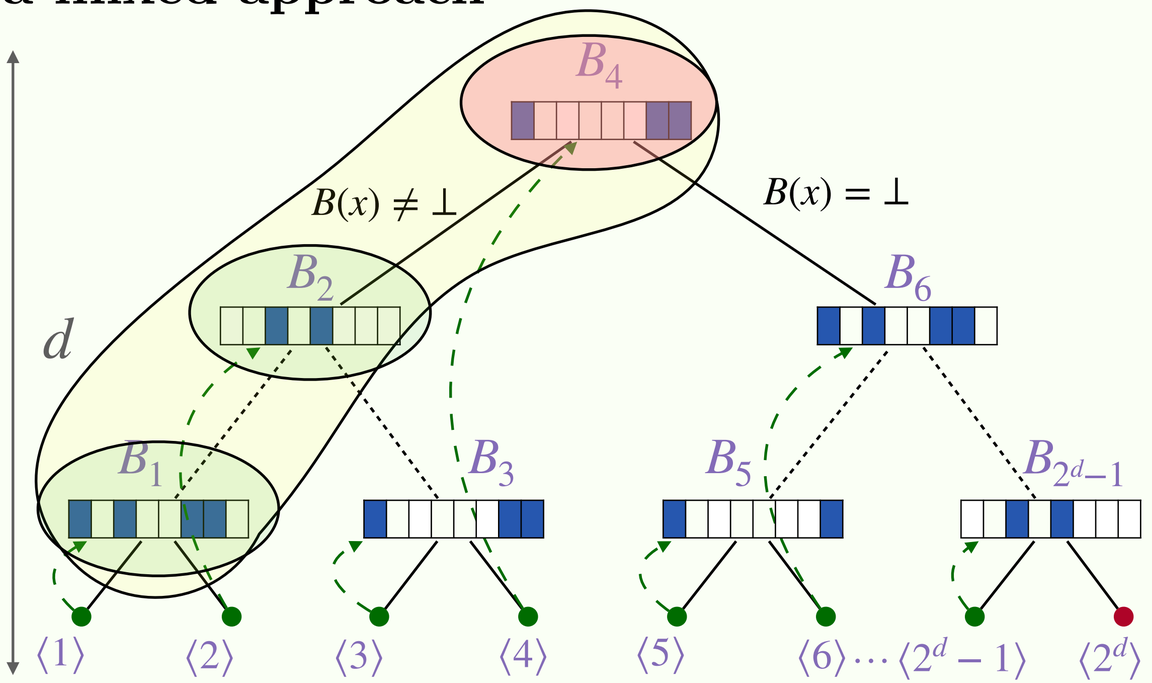
$\forall x, \Pr[x \text{ reaches } \langle \ell \rangle \mid B_1, \ldots, B_{\ell-1}] \geq (1 - 2^{-w})^d$

Faux-deterministic construction for FIND1

The successful design: a mixed approach

Let $q = \text{polylog}(n)$, and $w, d := \Theta(\sqrt{q})$

- Sample $2^d - 1$ random blocks $B_v \subset [n]$, $|B_v| = w$
- $B_v(x) = \{i \in B_v : x_i = 1\}$ or $\perp$
- Go left if $B_v(x) \neq \perp$ and right otherwise
- When reached leaf $\langle \ell \rangle$, output $B_\ell(x)$



Claim. Let $q = \text{polylog}(n)$, and $w, d := \Theta(\sqrt{q})$. Then, any successful adversary requires at least $2^{\Omega(\sqrt{q})} = \text{quasipoly(n)}$ many queries.

Proof idea. Cannot skip any of the blocks, as earlier queries are hidden in the $\vee$ blocks

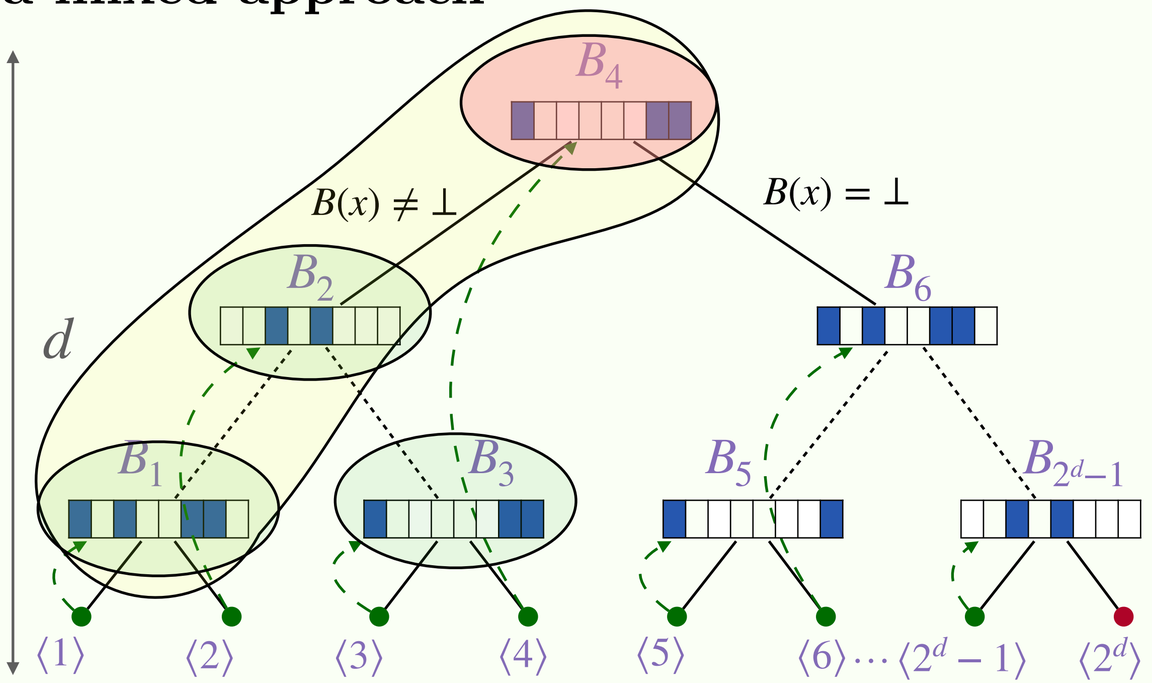
$\forall x, \Pr[x \text{ reaches } \langle \ell \rangle \mid B_1, \ldots, B_{\ell-1}] \geq (1 - 2^{-w})^d$

Faux-deterministic construction for FIND1

The successful design: a mixed approach

Let $q = \text{polylog}(n)$, and $w, d := \Theta(\sqrt{q})$

- Sample $2^d - 1$ random blocks $B_v \subset [n]$, $|B_v| = w$
- $B_v(x) = \{i \in B_v : x_i = 1\}$ or $\perp$
- Go left if $B_v(x) \neq \perp$ and right otherwise
- When reached leaf $\langle \ell \rangle$, output $B_\ell(x)$



Claim. Let $q = \text{polylog}(n)$, and $w, d := \Theta(\sqrt{q})$. Then, any successful adversary requires at least $2^{\Omega(\sqrt{q})} = \text{quasipoly(n)}$ many queries.

Proof idea. Cannot skip any of the blocks, as earlier queries are hidden in the $\vee$ blocks

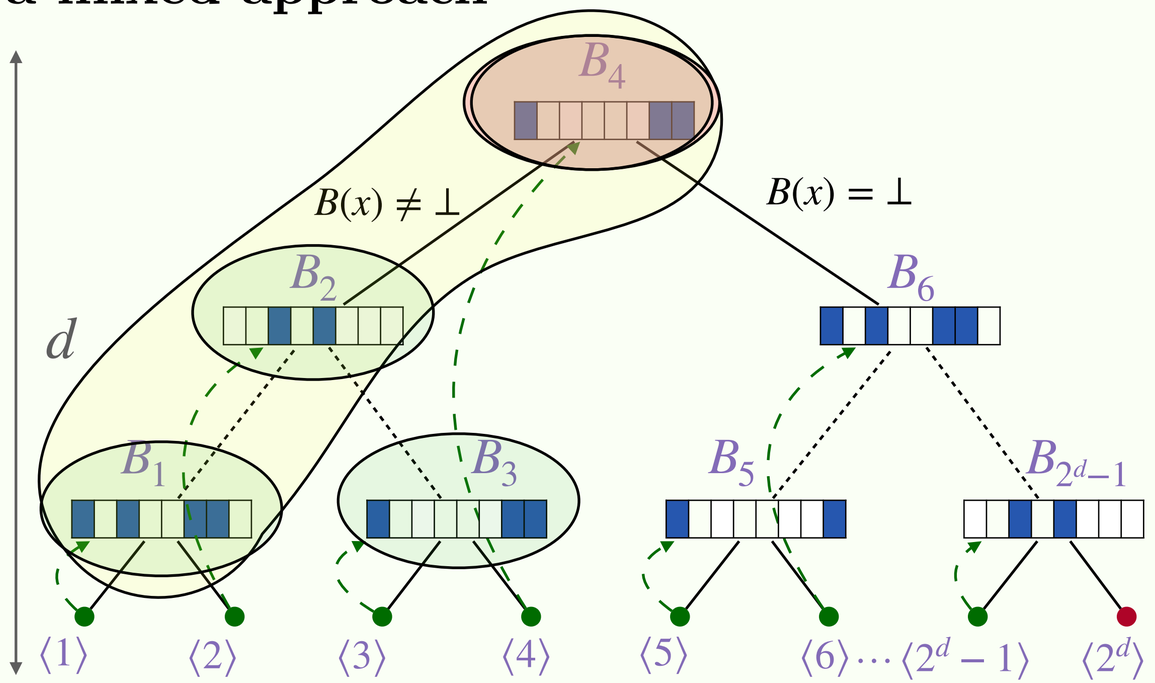
$\forall x, \Pr[x \text{ reaches } \langle \ell \rangle \mid B_1, \ldots, B_{\ell-1}] \geq (1 - 2^{-w})^d$

Faux-deterministic construction for FIND1

The successful design: a mixed approach

Let $q = \text{polylog}(n)$, and $w, d := \Theta(\sqrt{q})$

- Sample $2^d - 1$ random blocks $B_v \subset [n]$, $|B_v| = w$
- $B_v(x) = \{i \in B_v : x_i = 1\}$ or $\perp$
- Go left if $B_v(x) \neq \perp$ and right otherwise
- When reached leaf $\langle \ell \rangle$, output $B_\ell(x)$



Claim. Let $q = \text{polylog}(n)$, and $w, d := \Theta(\sqrt{q})$. Then, any successful adversary requires at least $2^{\Omega(\sqrt{q})} = \text{quasipoly(n)}$ many queries.

Proof idea. Cannot skip any of the blocks, as earlier queries are hidden in the $\vee$ blocks

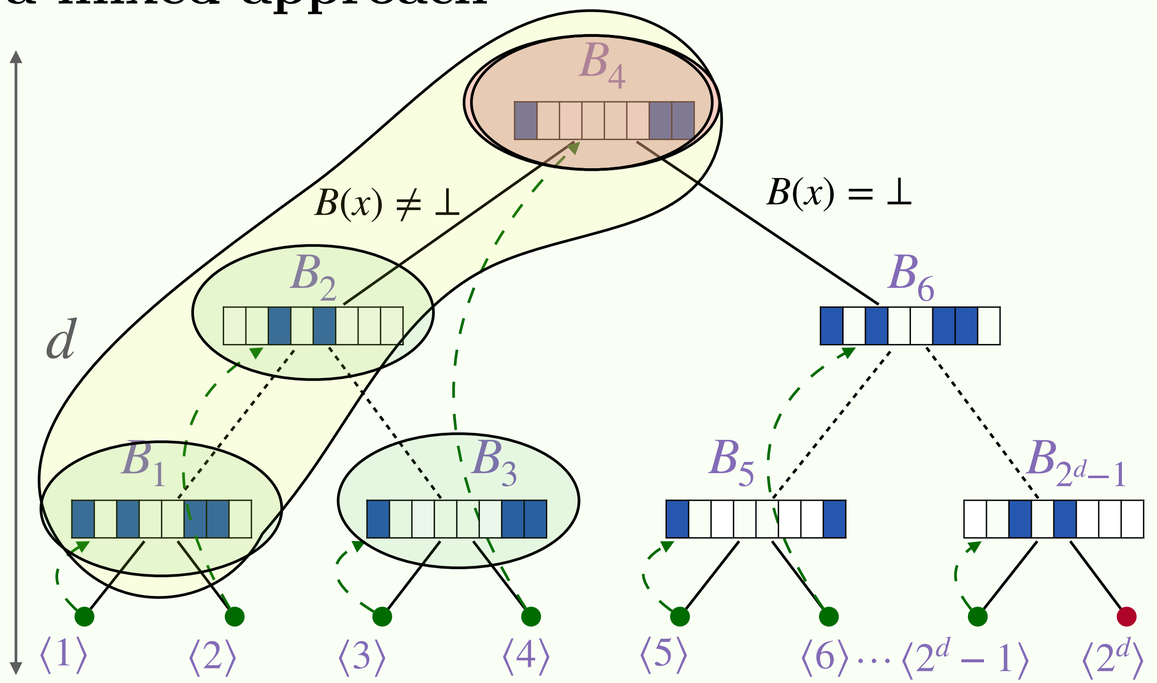
$\forall x, \Pr[x \text{ reaches } \langle \ell \rangle \,|\, B_1, \ldots, B_{\ell-1}] \geq (1 - 2^{-w})^d$

Faux-deterministic construction for FIND1

The successful design: a mixed approach

Let $q = \text{polylog}(n)$, and $w, d := \Theta(\sqrt{q})$

- Sample $2^d - 1$ random blocks $B_v \subset [n]$, $|B_v| = w$
- $B_v(x) = \{i \in B_v : x_i = 1\}$ or $\perp$
- Go left if $B_v(x) \neq \perp$ and right otherwise
- When reached leaf $\langle \ell \rangle$, output $B_\ell(x)$



Claim. Let $q = \text{polylog}(n)$, and $w, d := \Theta(\sqrt{q})$. Then, any successful adversary requires at least $2^{\Omega(\sqrt{q})} = \text{quasipoly(n)}$ many queries.

Proof idea. Cannot skip any of the blocks, as earlier queries are hidden in the $\vee$ blocks

$\forall x, \Pr[x \text{ reaches } \langle \ell \rangle \mid B_1, \ldots, B_{\ell-1}] \geq (1 - 2^{-w})^d$

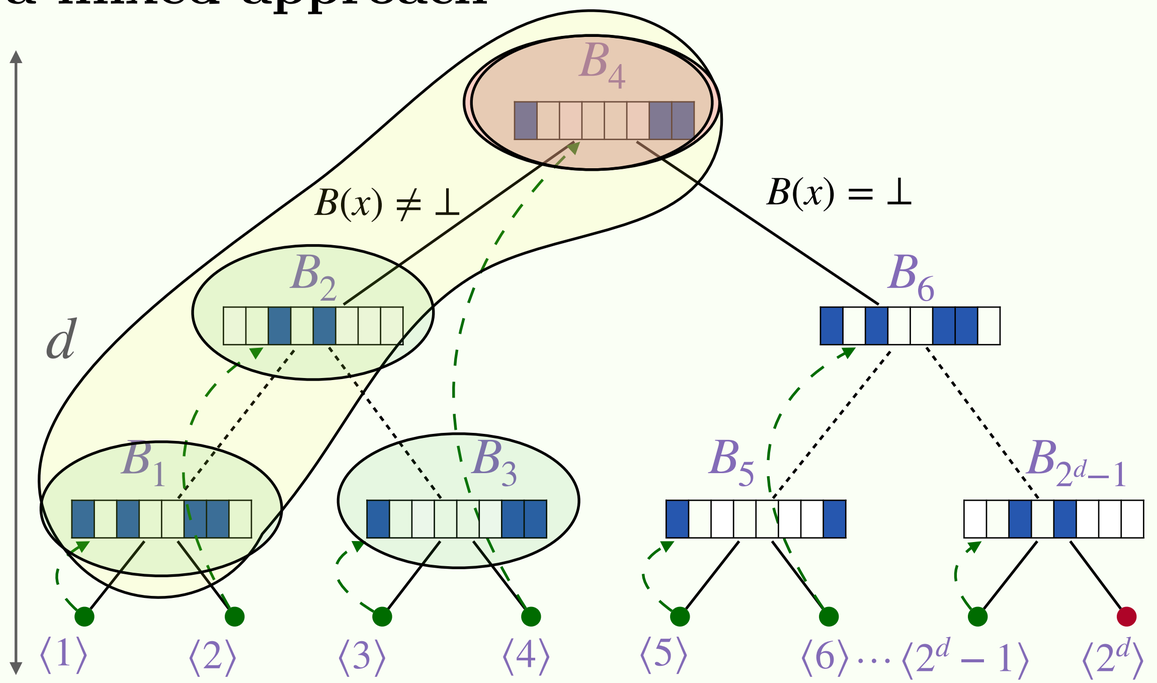
Lemma. this is almost tight: for any algorithm with q queries, there exists a adversary with $2^{O(q)}$ many queries

Faux-deterministic construction for FIND1

The successful design: a mixed approach

Let $q = \text{polylog}(n)$, and $w, d := \Theta(\sqrt{q})$

- Sample $2^d - 1$ random blocks $B_v \subset [n]$, $|B_v| = w$
- $B_v(x) = \{i \in B_v : x_i = 1\}$ or $\perp$
- Go left if $B_v(x) \neq \perp$ and right otherwise
- When reached leaf $\langle \ell \rangle$, output $B_\ell(x)$



Claim. Let $q = \text{polylog}(n)$, and $w, d := \Theta(\sqrt{q})$. Then, any successful adversary requires at least $2^{\Omega(\sqrt{q})} = \text{quasipoly(n)}$ many queries.

Proof idea. Cannot skip any of the blocks, as earlier queries are hidden in the $\vee$ blocks

$\forall x, \Pr[x \text{ reaches } \langle \ell \rangle \mid B_1, \ldots, B_{\ell-1}] \geq (1 - 2^{-w})^d$

Lemma. this is almost tight: for any algorithm with q queries, there exists a adversary with $2^{O(q)}$ many queries

Lemma. A similar bound of $2^{\Omega(\sqrt{q})}$ many queries can be proven against quantum adversaries.

Conclusion

In summary:

- Faux determinism: fix the randomness once; failures may exist, but bounded-query adversaries cannot find them.
- Faux derandomization: every efficiently verifiable randomized search problem has an efficient faux-deterministic algorithm
- Pseudo-deterministic lower bounds need to be non-constructive in a sense

Some future directions?

- Closing the gap $\sqrt{n} \lesssim D^{\text{pseudo}}(\text{FIND1}) \lesssim n$
- Constructing a faux-deterministic algorithm resisting $2^{\Omega(q)}$ many queries?
- In which models does faux determinists algorithms make sense
- Relation to other complexity measures?

Thank you!

References.

[**GG11**] Eran Gat and Shafi Goldwasser.
Probabilistic Search Algorithms with Unique Answers and Their Cryptographic Applications.
Electronic Colloquium on Computational Complexity, Report TR11-136, 2011.

[**GIPS21**] Shafi Goldwasser, Russell Impagliazzo, Toniann Pitassi, and Rahul Santhanam.
On the Pseudo-Deterministic Query Complexity of NP Search Problems.
Proceedings of the 36th Computational Complexity Conference (CCC 2021), LIPIcs 200, Article 36, 2021.

?