Noise sensitivity from fractional query algorithms

Renan Gross, Weizmann Institute of Science

Boolean functions

A Boolean function is a function $f: \{-1,1\}^n \to \{-1,1\}$.

Boolean functions

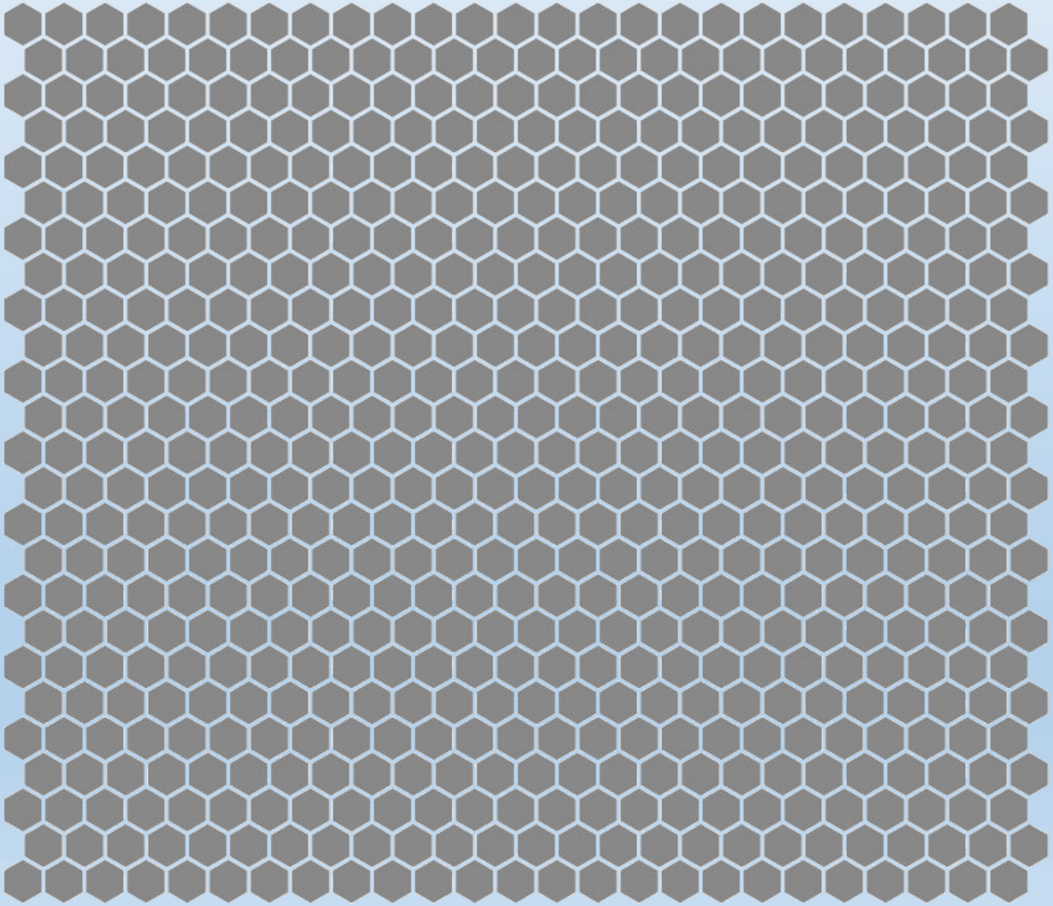
A Boolean function is a function $f: \{-1,1\}^n \to \{-1,1\}$.

Example: Majority

$$f(x) = \text{sign} \sum_{i=1}^{n} x_i$$

Percolation

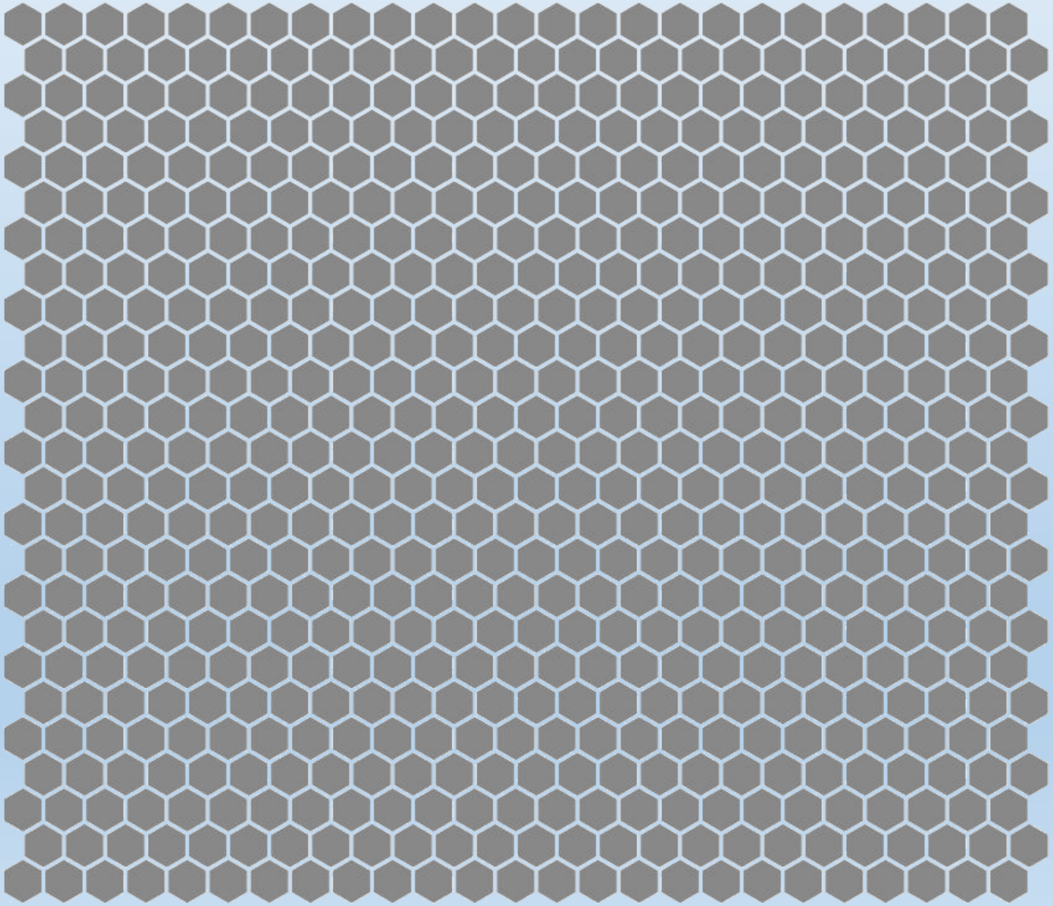
$f:\{-1,1\}^n \to \{-1,1\}$



Percolation

$f: \{-1,1\}^n \to \{-1,1\}$

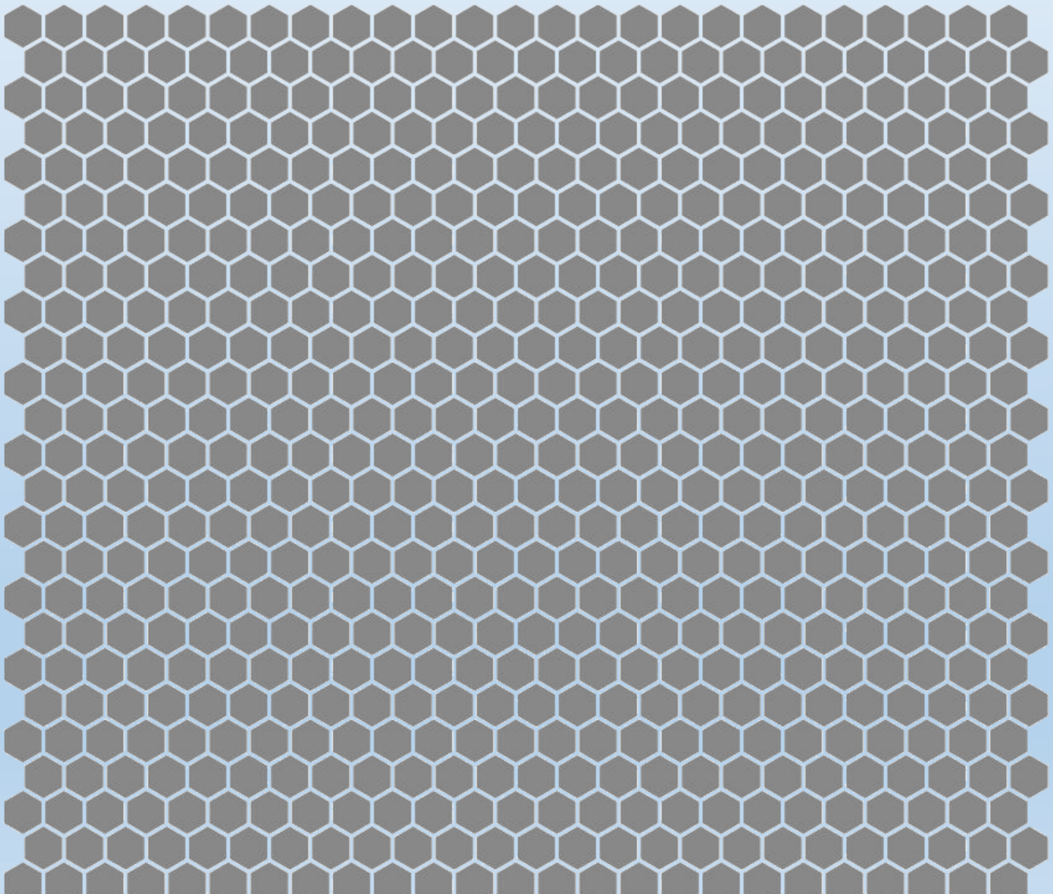
$$x = 1, 1, -1, 1, -1, -1, -1, -1, -1, 1, 1, -1, 1, -1, \ldots$$

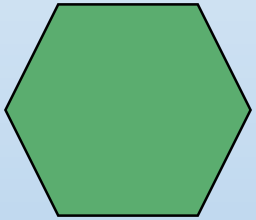


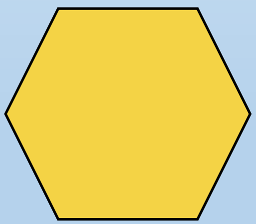
Percolation

$f\colon \{-1,1\}^n \rightarrow \{-1,1\}$

$\mathrm{x} = 1, 1, -1, 1, -1, -1, -1, -1, -1, 1, 1, -1, 1, -1, \ldots$



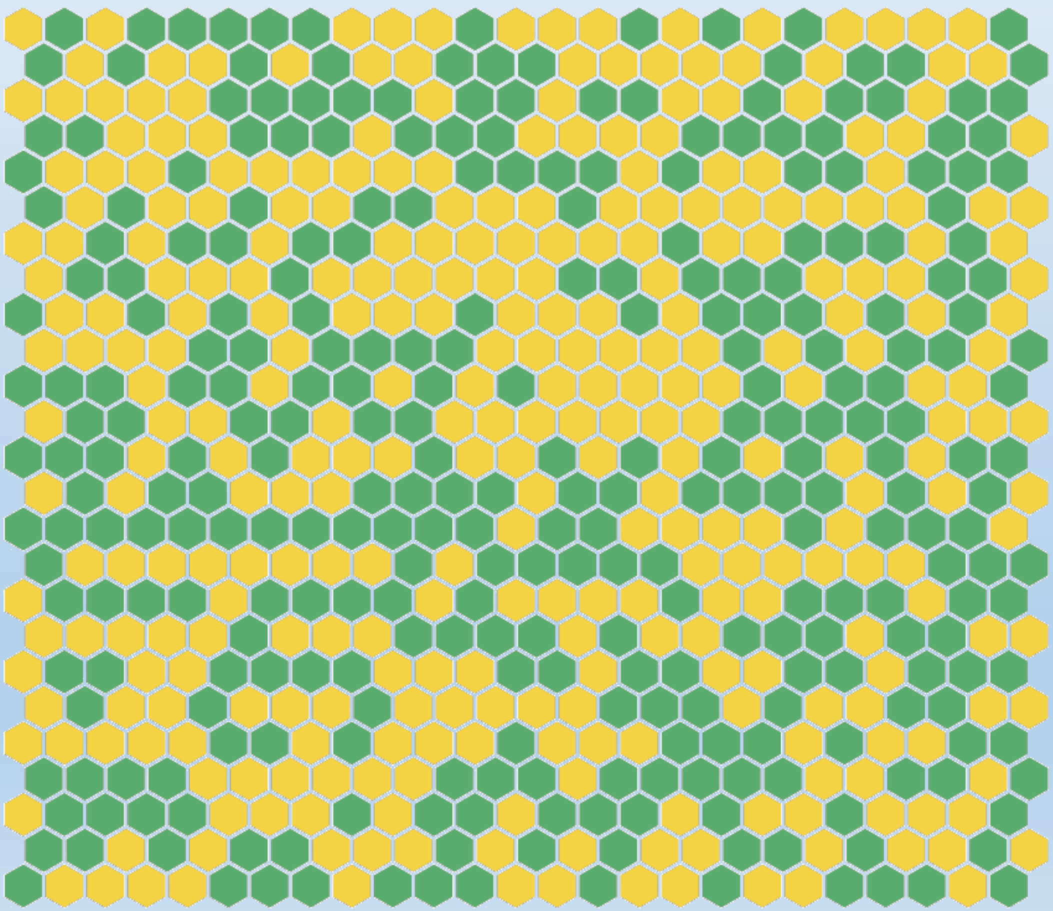
$1 =$ 

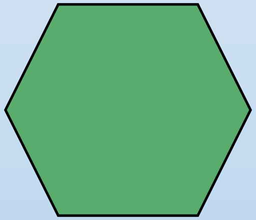
$-1 =$ 

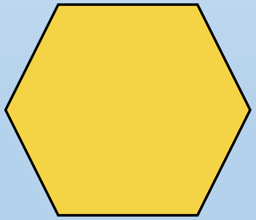
Percolation

$f\colon \{-1,1\}^n \to \{-1,1\}$

$\mathrm{x} = 1, 1, -1, 1, -1, -1, -1, -1, -1, 1, 1, -1, 1, -1, \ldots$



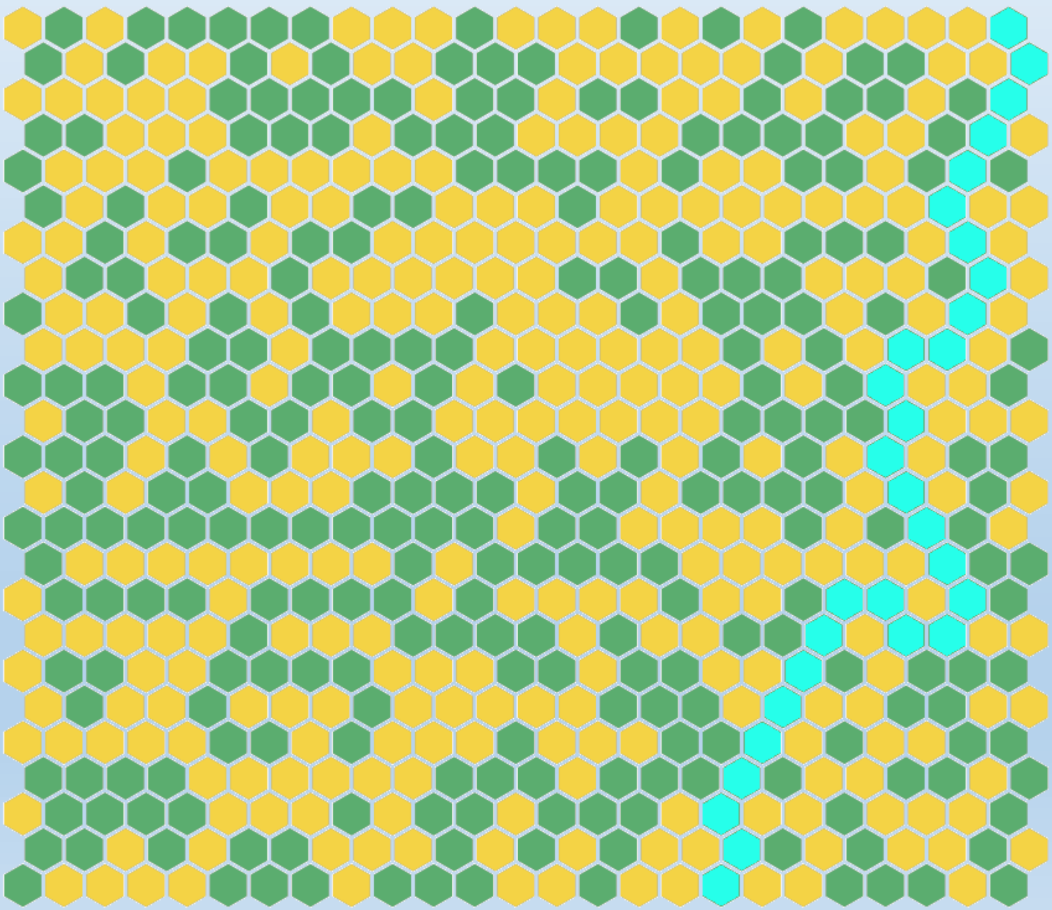
$1 =$ 

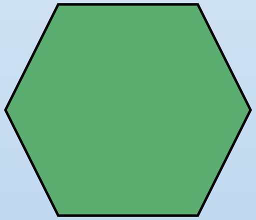
$-1 =$ 

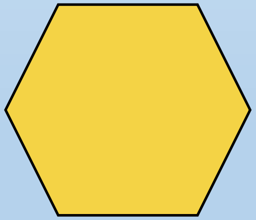
Percolation

$f\colon \{-1,1\}^n \to \{-1,1\}$

$$x = 1, 1, -1, 1, -1, -1, -1, -1, -1, 1, 1, -1, 1, -1, \ldots$$

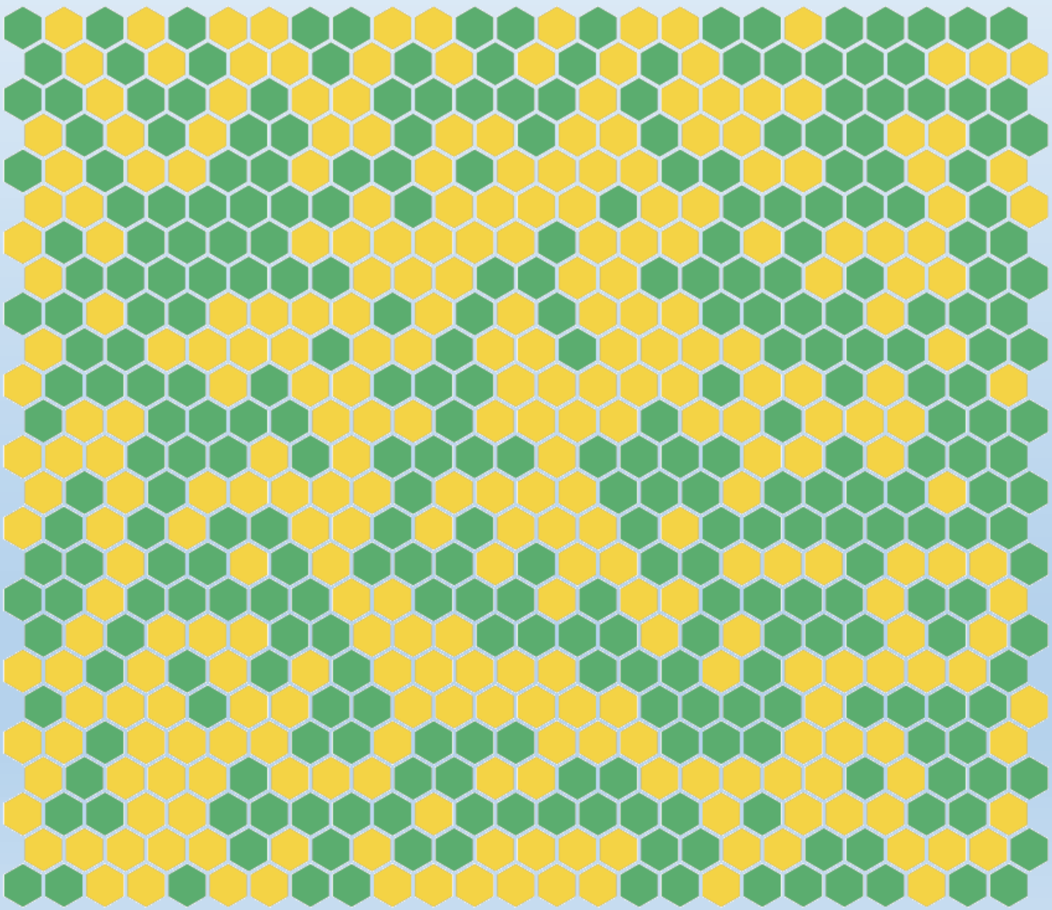


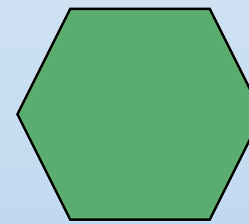
$1 =$ 

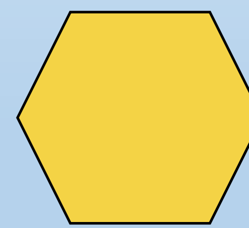
$-1 =$ 

Percolation

$f : \{-1,1\}^n \to \{-1,1\}$

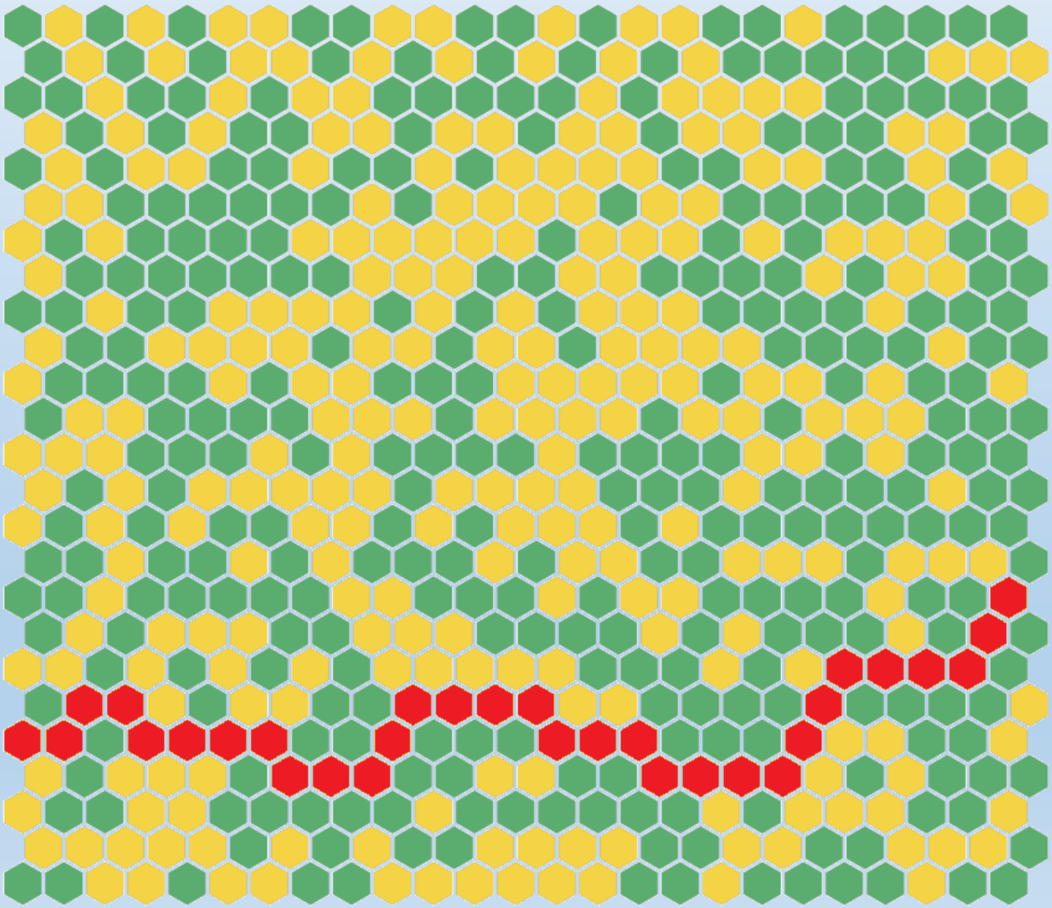


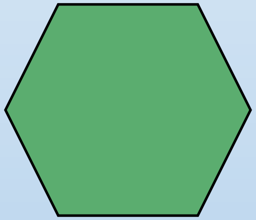
$1 =$ 

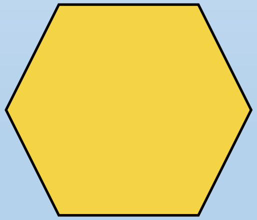
$-1 =$ 

Percolation

$f: \{-1,1\}^n \to \{-1,1\}$

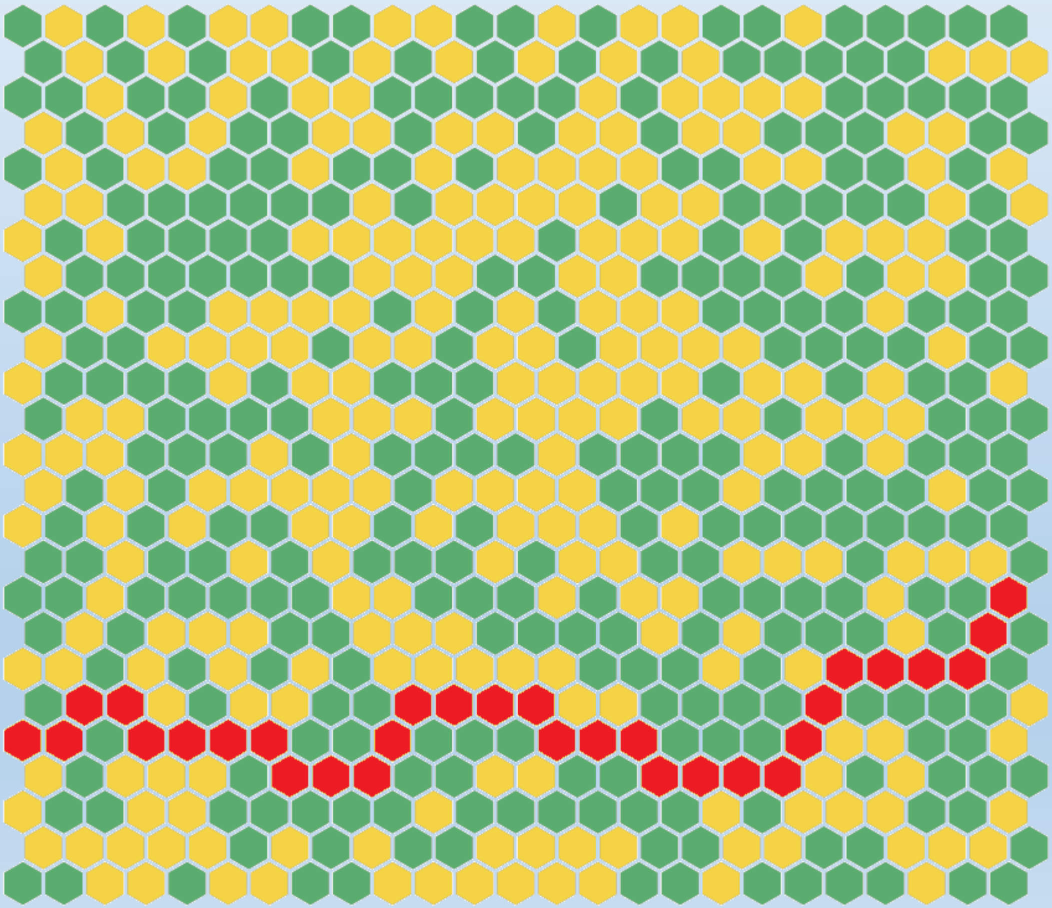


$1 =$ 

$-1 =$ 

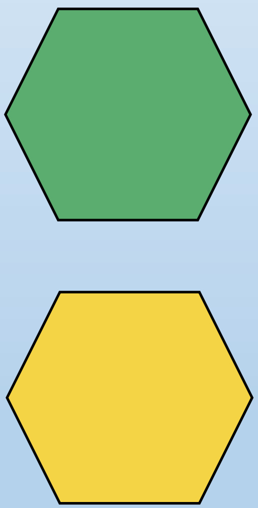
Percolation

$$f(x) = \begin{cases} 1 & \text{if green} \updownarrow \text{crossing} \\ -1 & \text{if yellow} \leftrightarrow \text{crossing} \end{cases}$$

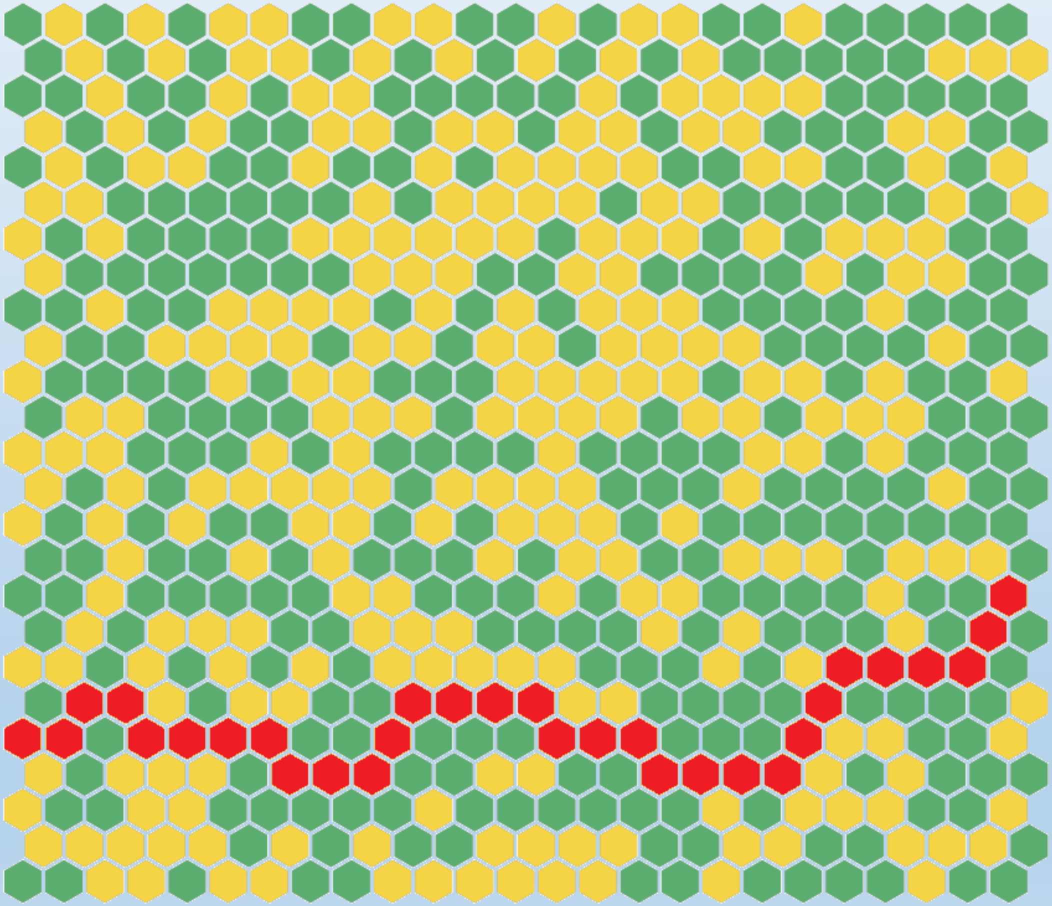


$1 =$

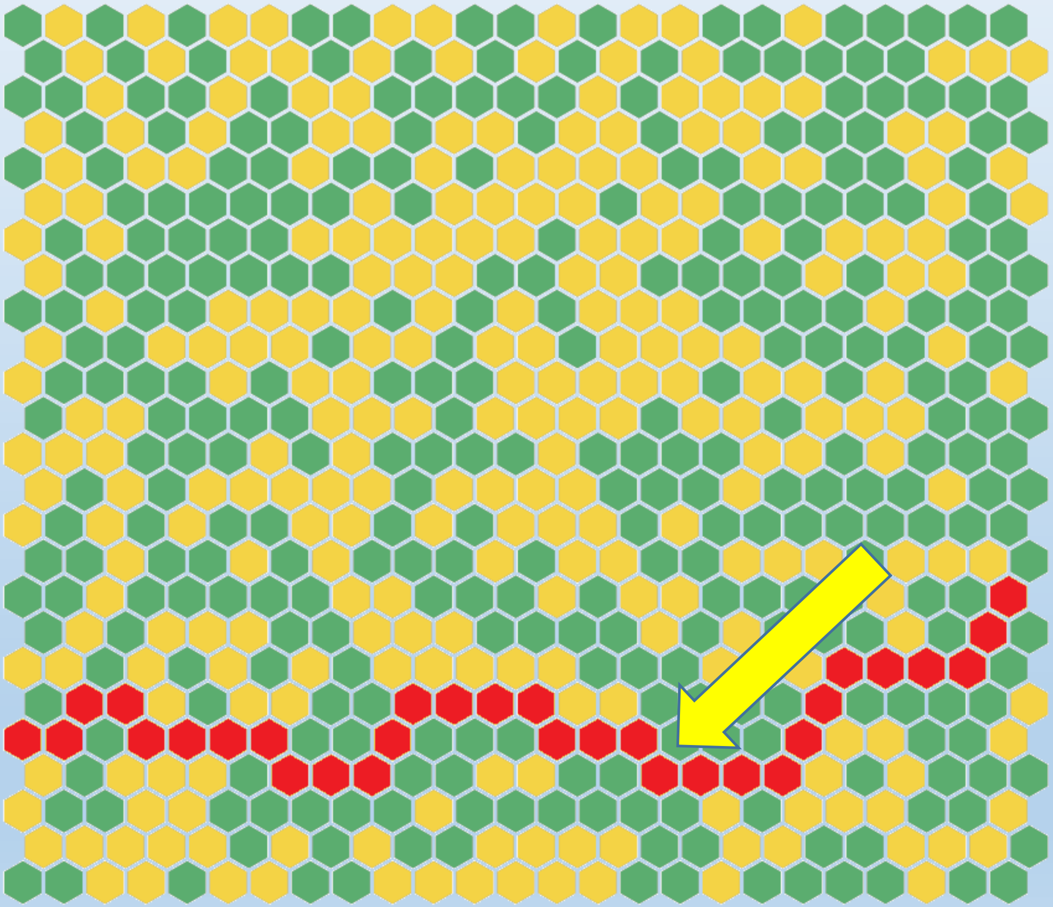
$-1 =$



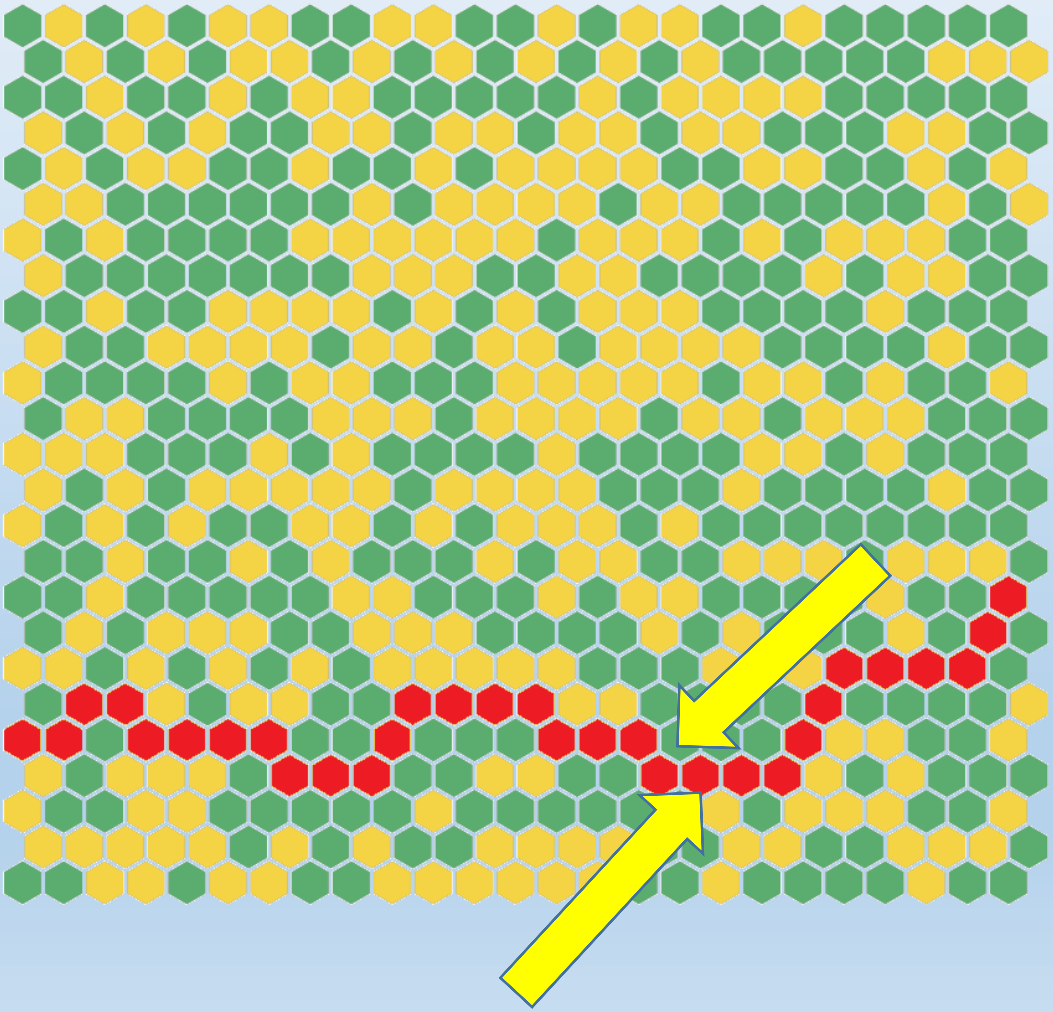
Noise sensitivity



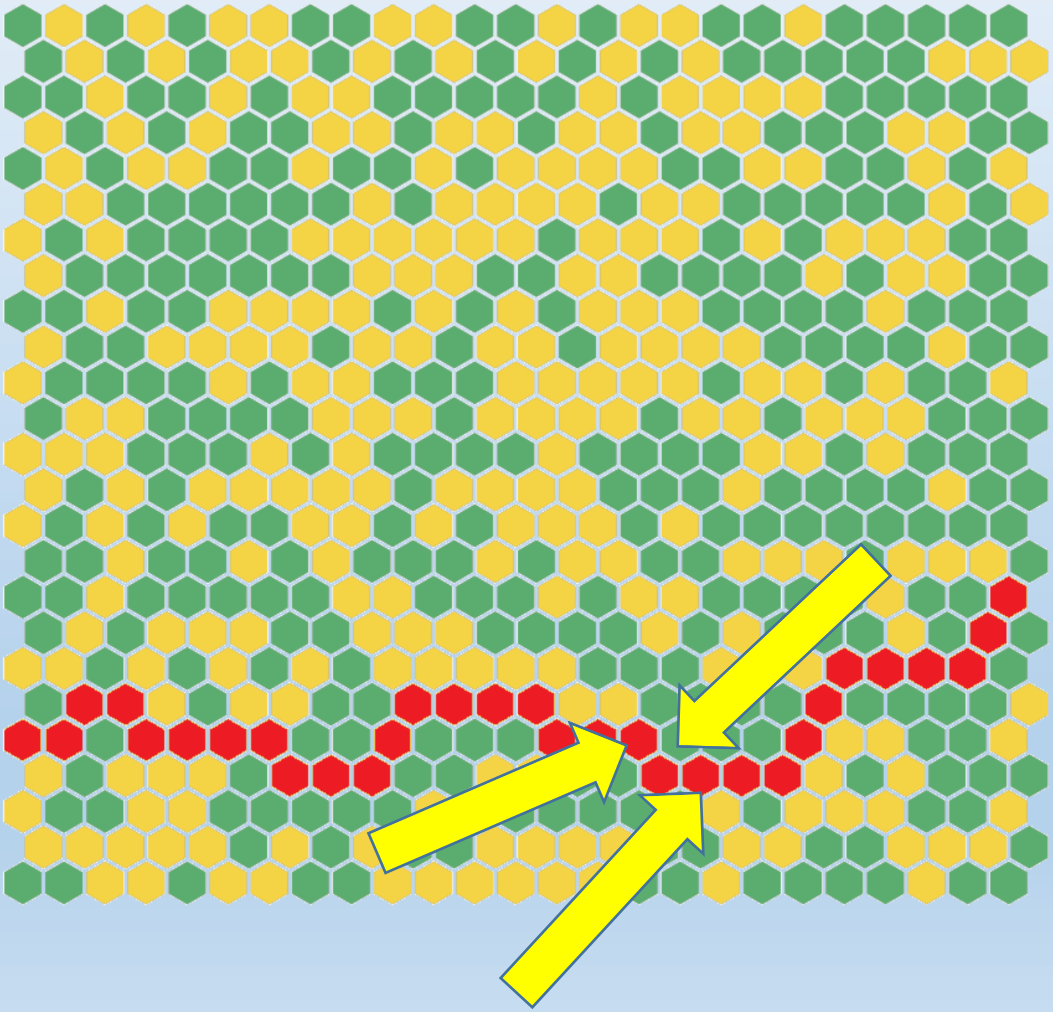
Noise sensitivity



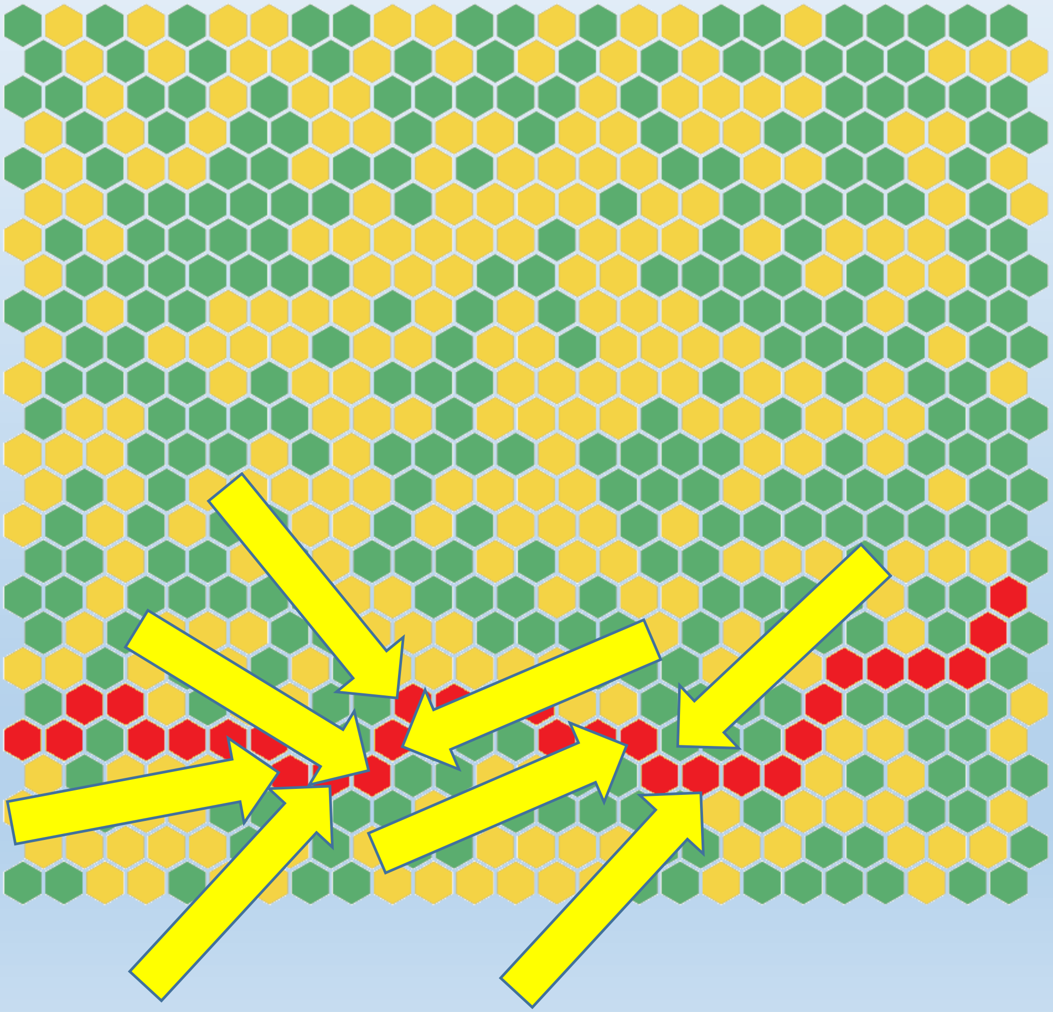
Noise sensitivity



Noise sensitivity



Noise sensitivity



Noise sensitivity

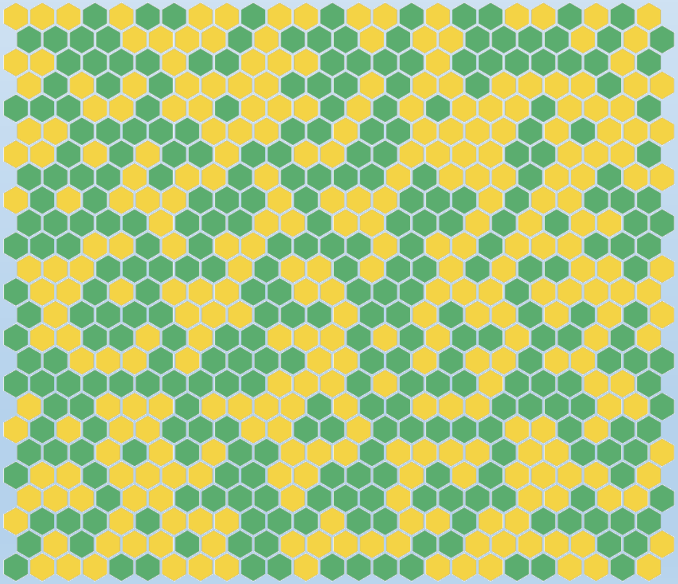
Pick $\varepsilon > 0$, and flip each bit with probability ε.

Did the function's value change?

Noise sensitivity

Pick $\varepsilon > 0$, and flip each bit with probability ε.

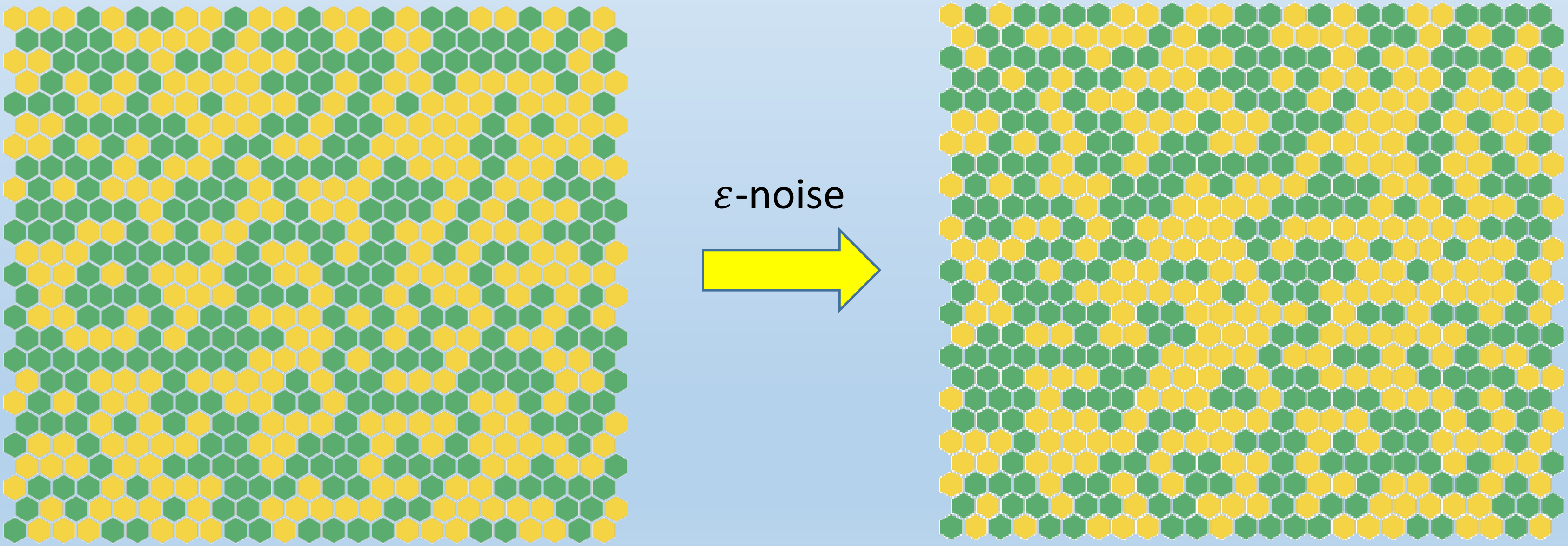
Did the function's value change?



Noise sensitivity

Pick $\varepsilon > 0$, and flip each bit with probability ε.

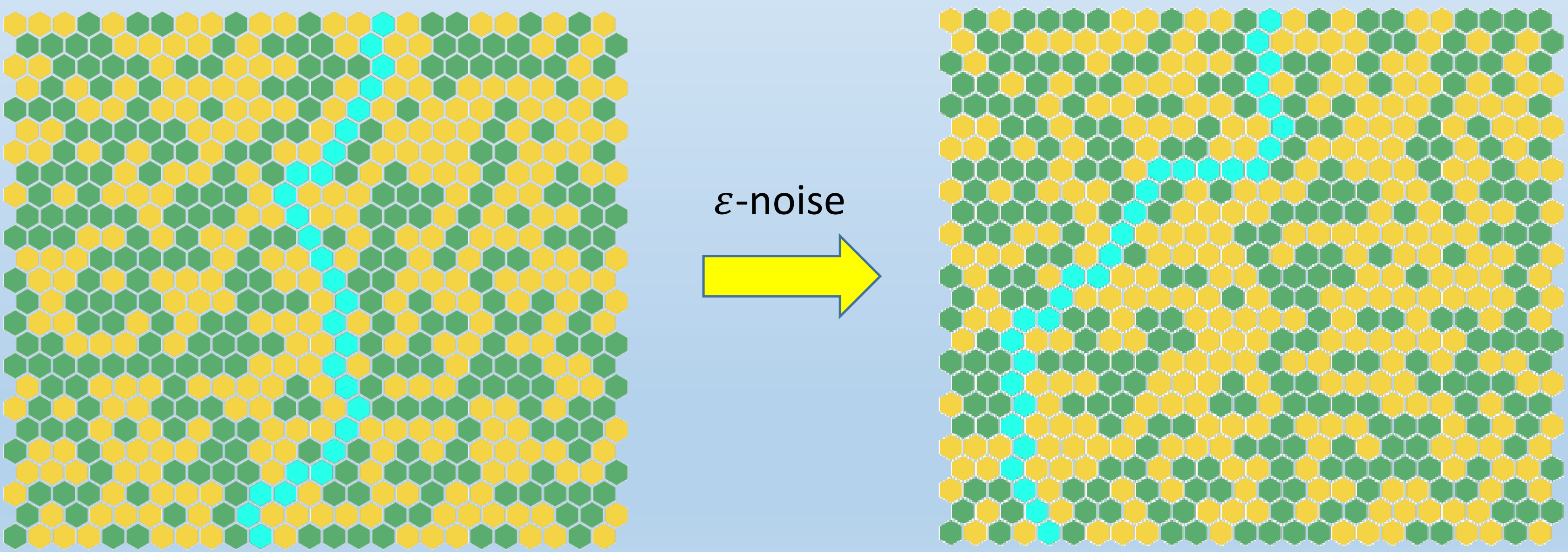
Did the function's value change?



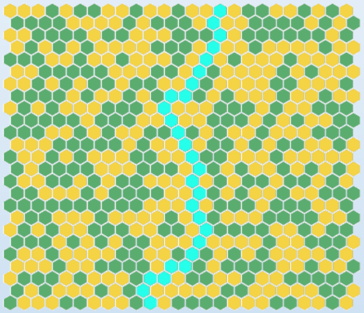
Noise sensitivity

Pick $\varepsilon > 0$, and flip each bit with probability ε.

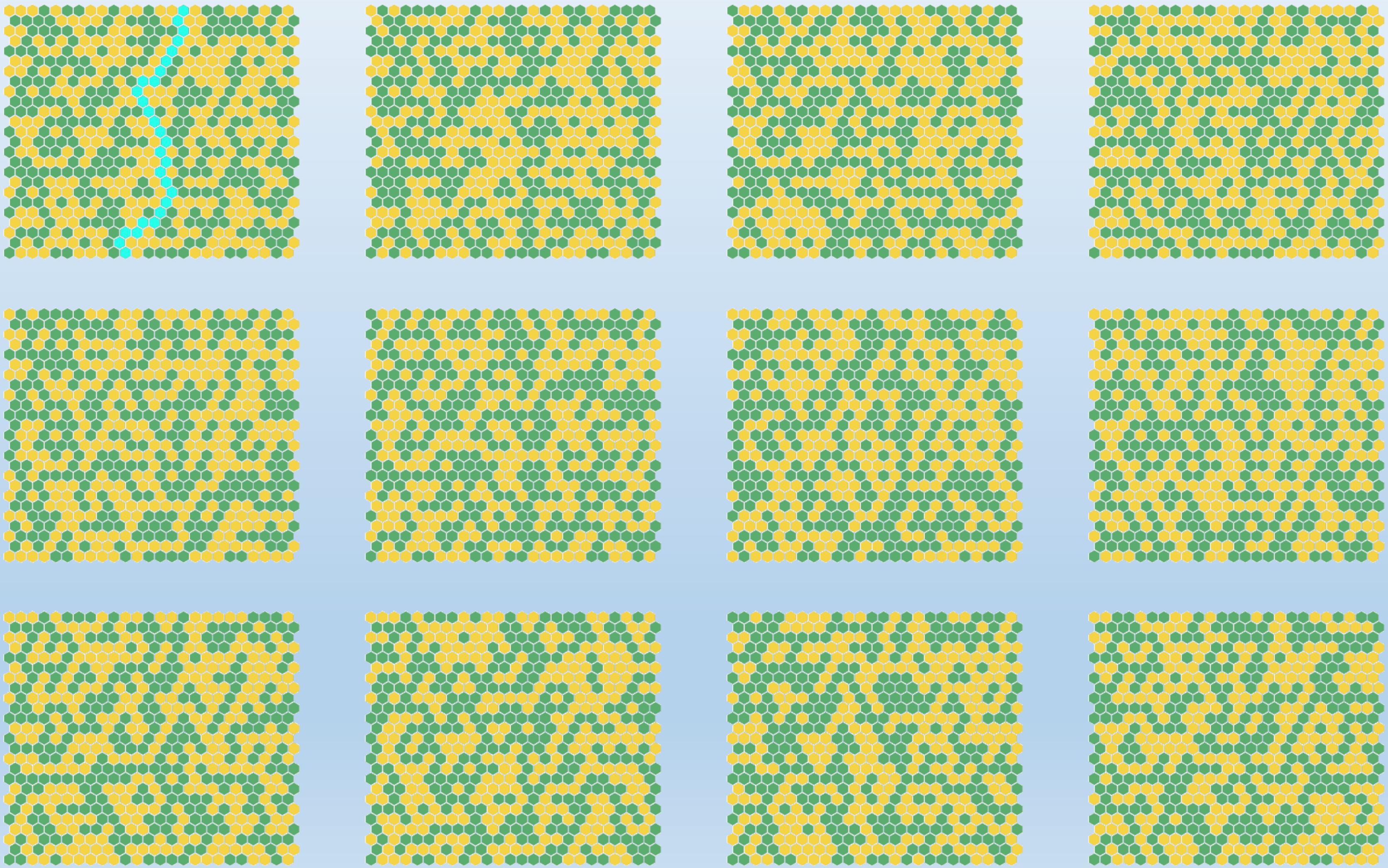
Did the function's value change?



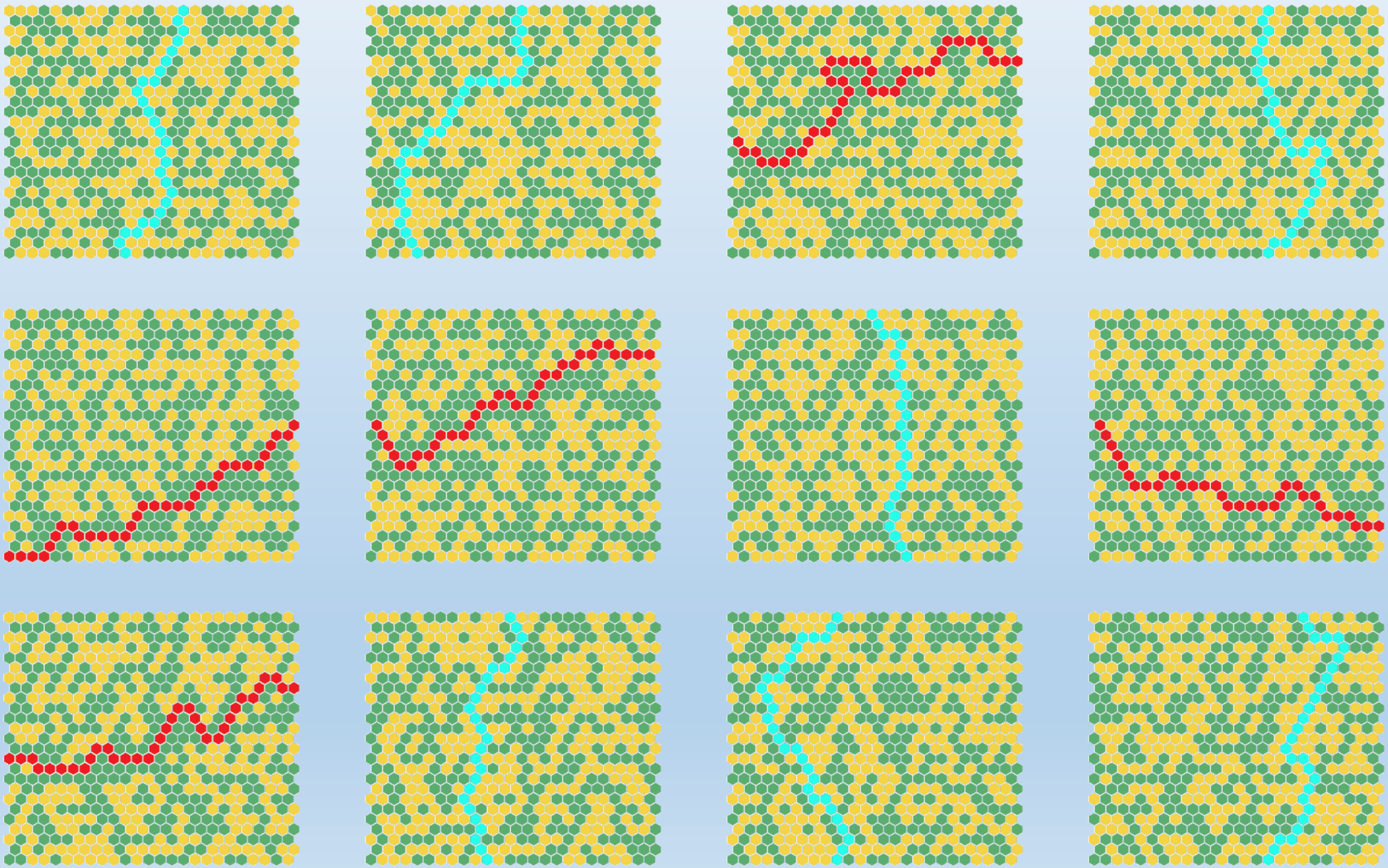
Noise sensitivity



Noise sensitivity



Noise sensitivity



Noise sensitivity

Definition: A sequence $f_n: \{-1,1\}^n \to \{-1,1\}$ of balanced Boolean functions is called "*noise sensitive*" if for all $\varepsilon > 0$,

$$\lim_{n\to\infty} \mathbb{E}[f_n(x) f_n(y)] = 0,$$

where x is random and y is an ε-noising of x.

Noise sensitivity

Definition: A sequence $f_n\colon \{-1,1\}^n \to \{-1,1\}$ of balanced Boolean functions is called "*noise sensitive*" if for all $\varepsilon > 0$,

$$\lim_{n\to\infty} \mathbb{E}[f_n(x) f_n(y)] = 0,$$

where x is random and y is an ε-noising of x.

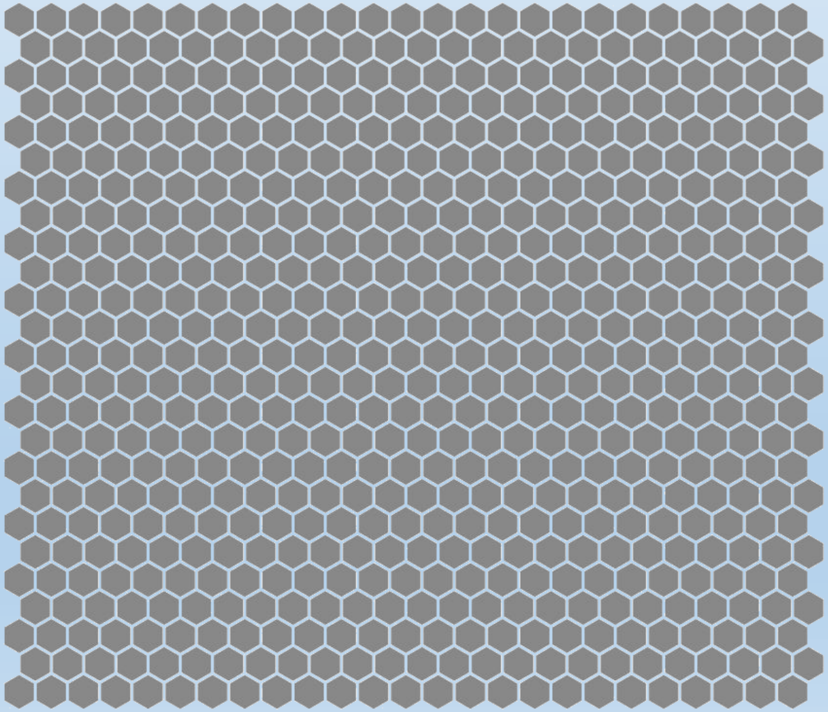
Is percolation crossing noise sensitive?

If so, how fast can ε go to 0 with n?

Decision trees

x is uniform random, but hidden from you.

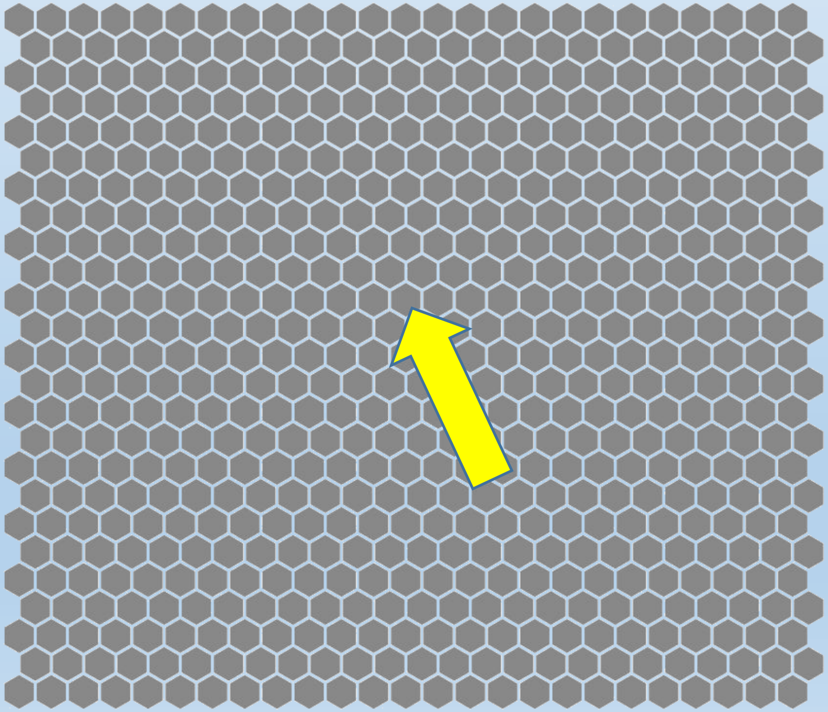
Reveal hexagons one by one, until $f(x)$ is found.



Decision trees

x is uniform random, but hidden from you.

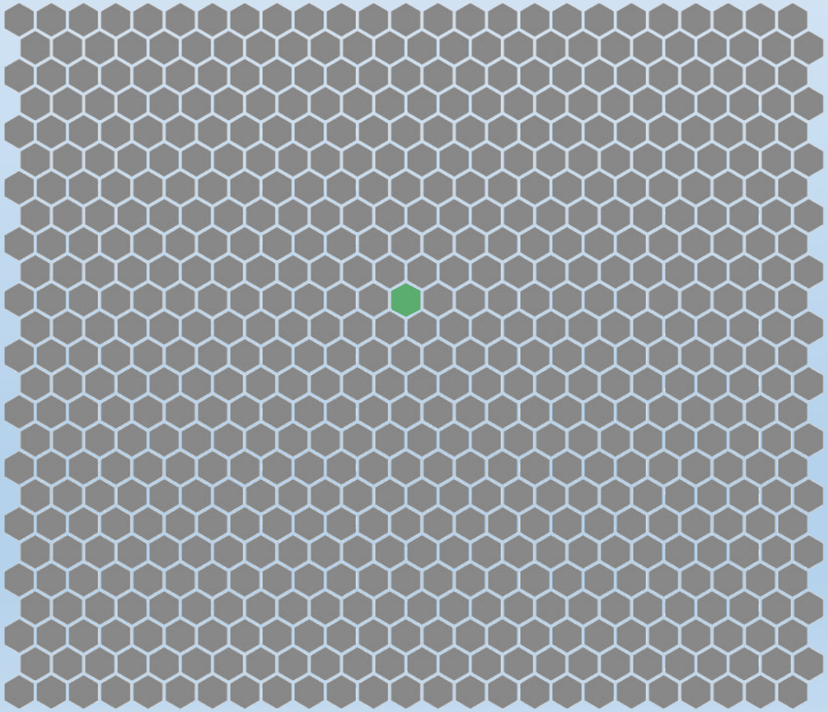
Reveal hexagons one by one, until $f(x)$ is found.



Decision trees

x is uniform random, but hidden from you.

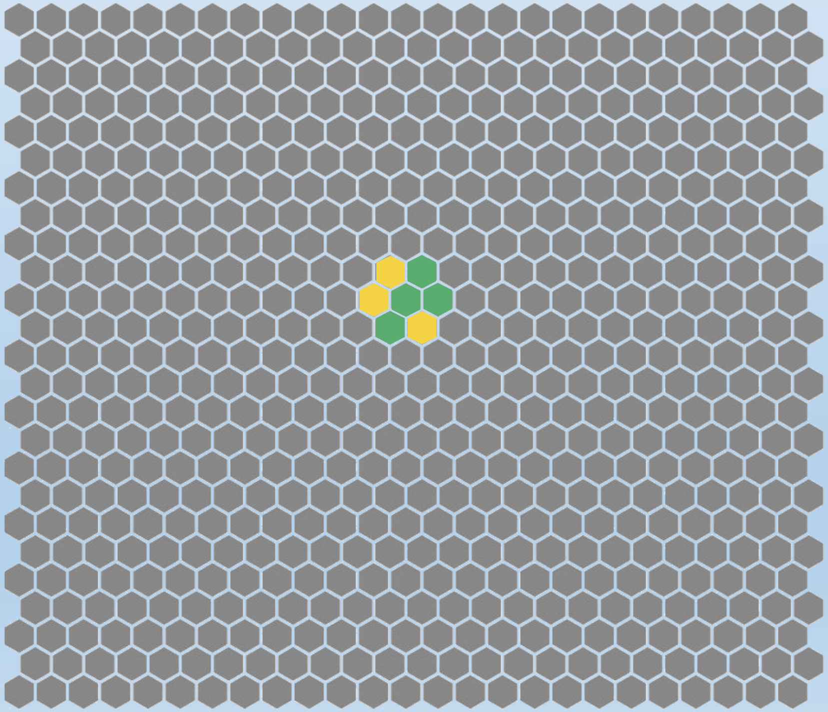
Reveal hexagons one by one, until $f(x)$ is found.



Decision trees

x is uniform random, but hidden from you.

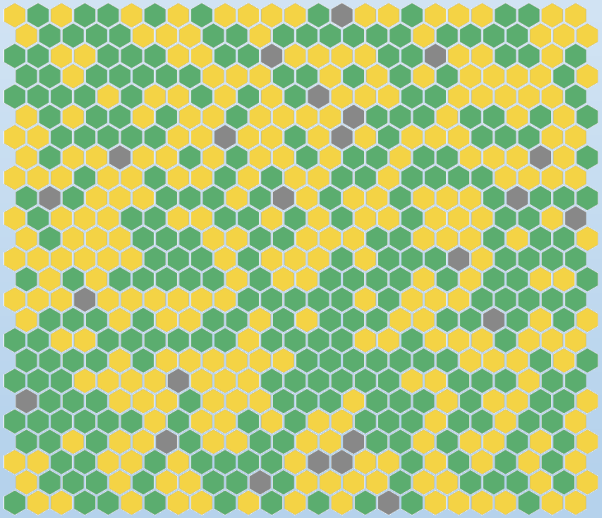
Reveal hexagons one by one, until $f(x)$ is found.



Decision trees

x is uniform random, but hidden from you.

Reveal hexagons one by one, until $f(x)$ is found.

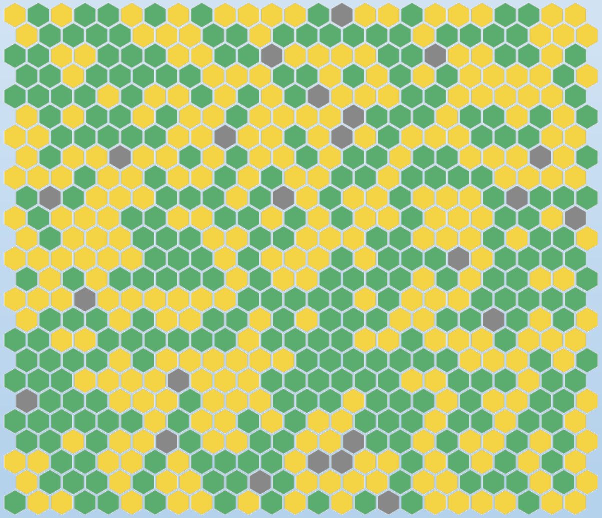


Reveal random bits

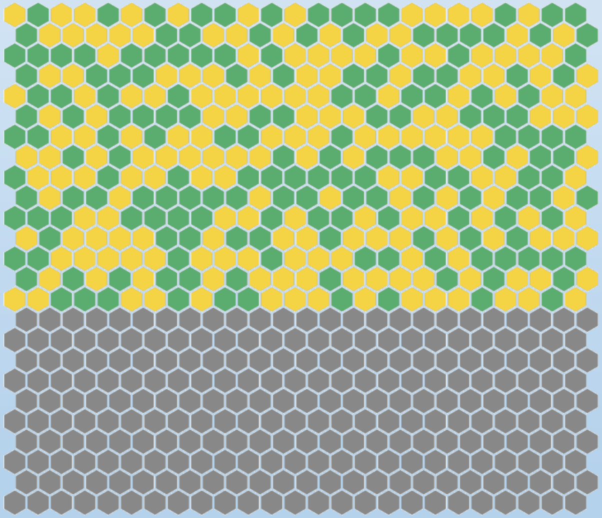
Decision trees

x is uniform random, but hidden from you.

Reveal hexagons one by one, until $f(x)$ is found.



Reveal random bits

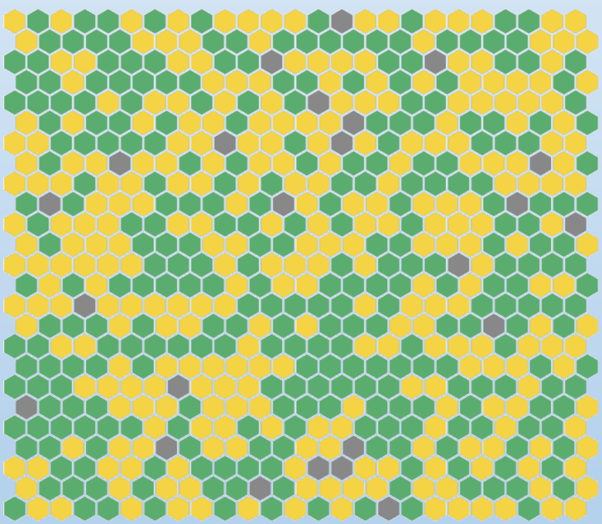


Reveal rows

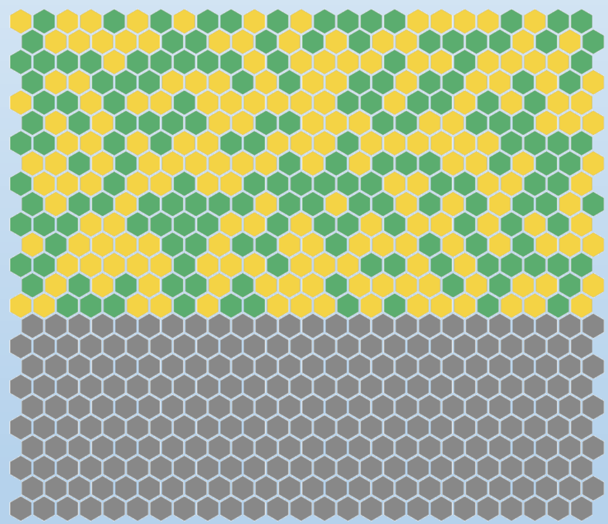
Decision trees

x is uniform random, but hidden from you.

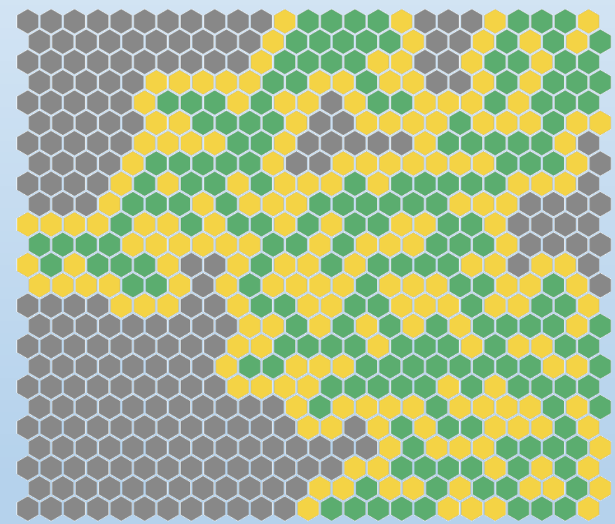
Reveal hexagons one by one, until $f(x)$ is found.



Reveal random bits



Reveal rows

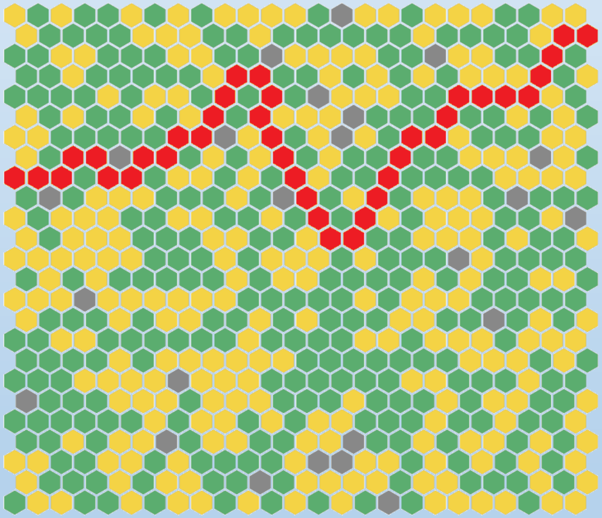


Random floodfill

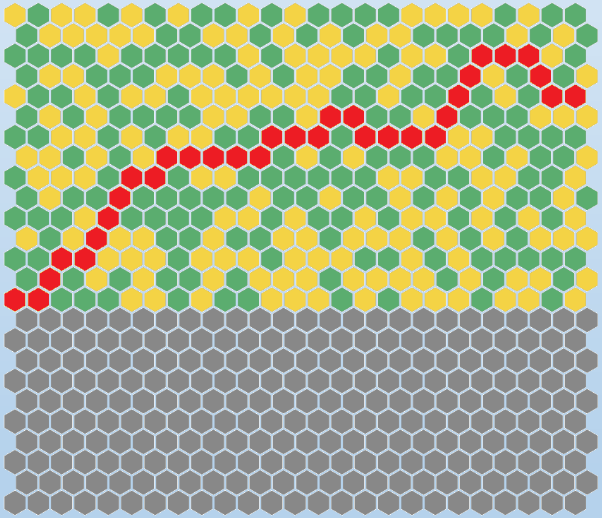
Decision trees

x is uniform random, but hidden from you.

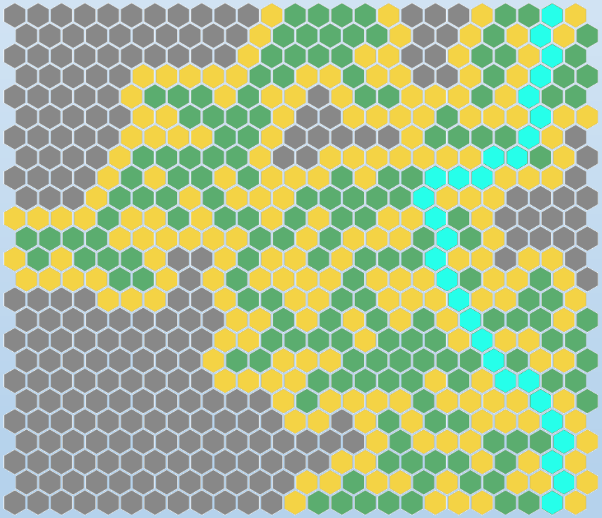
Reveal hexagons one by one, until $f(x)$ is found.



Reveal random bits



Reveal rows



Random floodfill

The Schramm-Steif Theorem

Let $f_n : \{-1,1\}^n \to \{-1,1\}$ be a sequence of Booelan functions.

Let T_n be a bit-reveal algorithm for f_n, and

$$\delta(n) := \max_i \delta_i = \max_i \mathbb{P}[T_n \text{ reveals bit } i]\,.$$

The Schramm-Steif Theorem

Let $f_n: \{-1,1\}^n \to \{-1,1\}$ be a sequence of Booelan functions.

Let T_n be a bit-reveal algorithm for f_n, and

$$\delta(n) := \max_i \delta_i = \max_i \mathbb{P}[T_n \text{ reveals bit } i] \,.$$

Theorem: If $\delta \to 0$, then f_n is noise sensitive.

The Schramm-Steif Theorem

Let $f_n: \{-1,1\}^n \to \{-1,1\}$ be a sequence of Booelan functions.

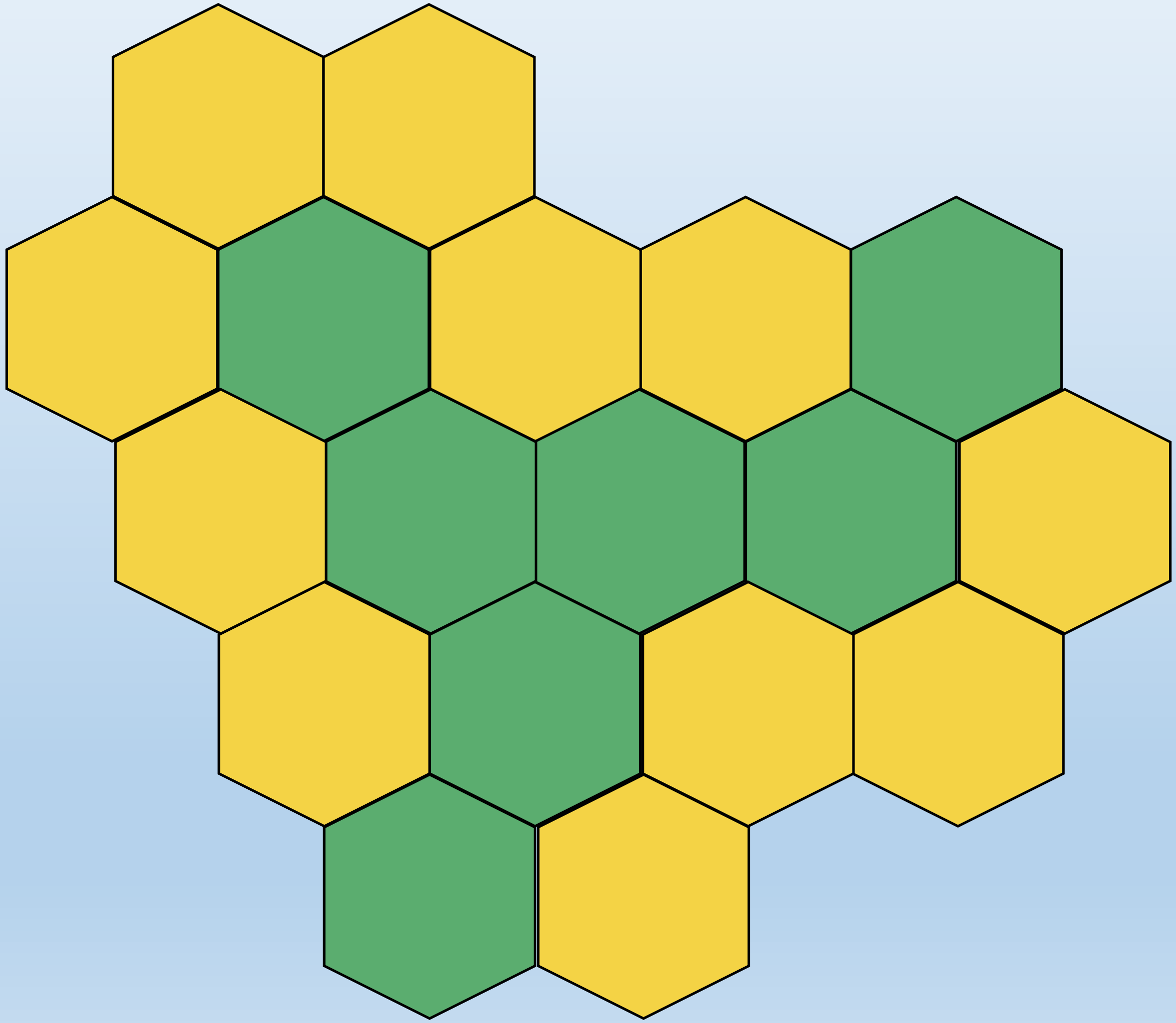
Let T_n be a bit-reveal algorithm for f_n, and

$$\delta(n) := \max_i \delta_i = \max_i \mathbb{P}[T_n \text{ reveals bit } i] .$$

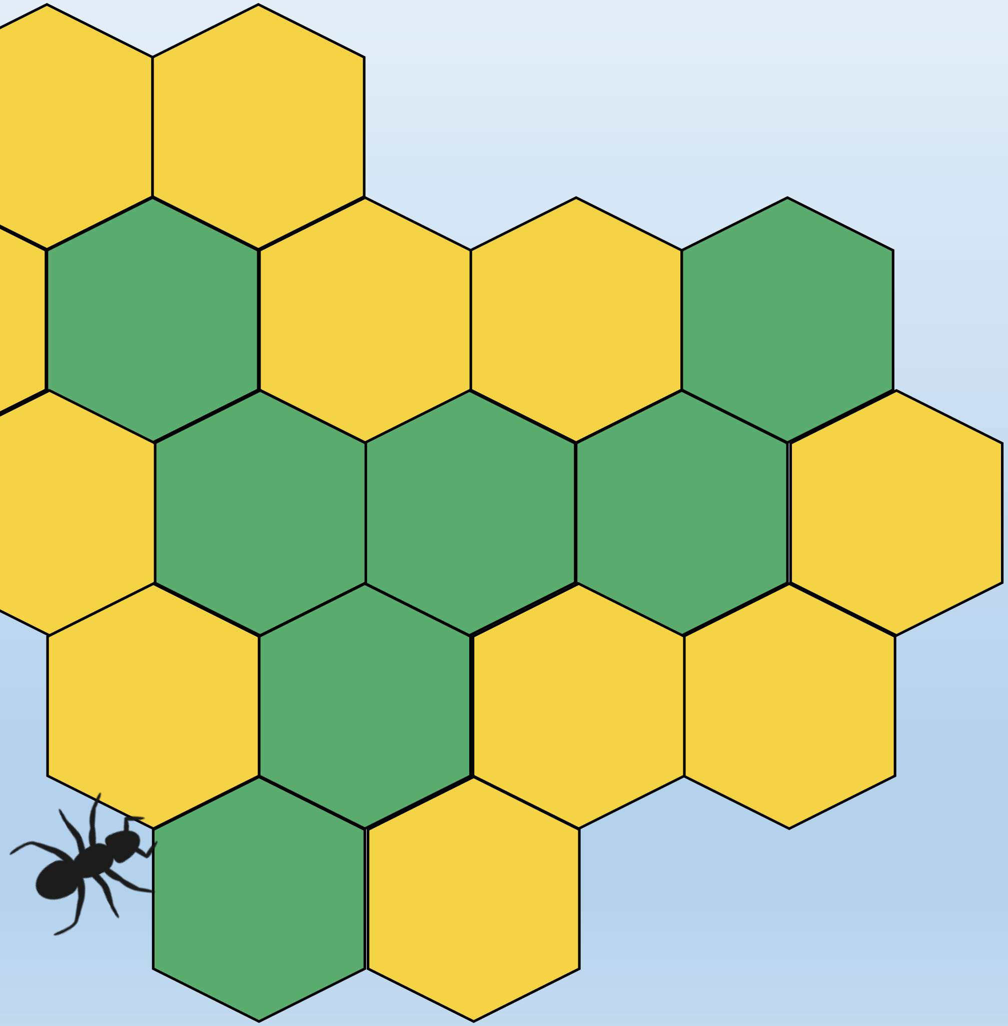
Theorem: If $\delta \to 0$, then f_n is noise sensitive.

The faster $\delta \to 0$, the more noise sensitive it is!

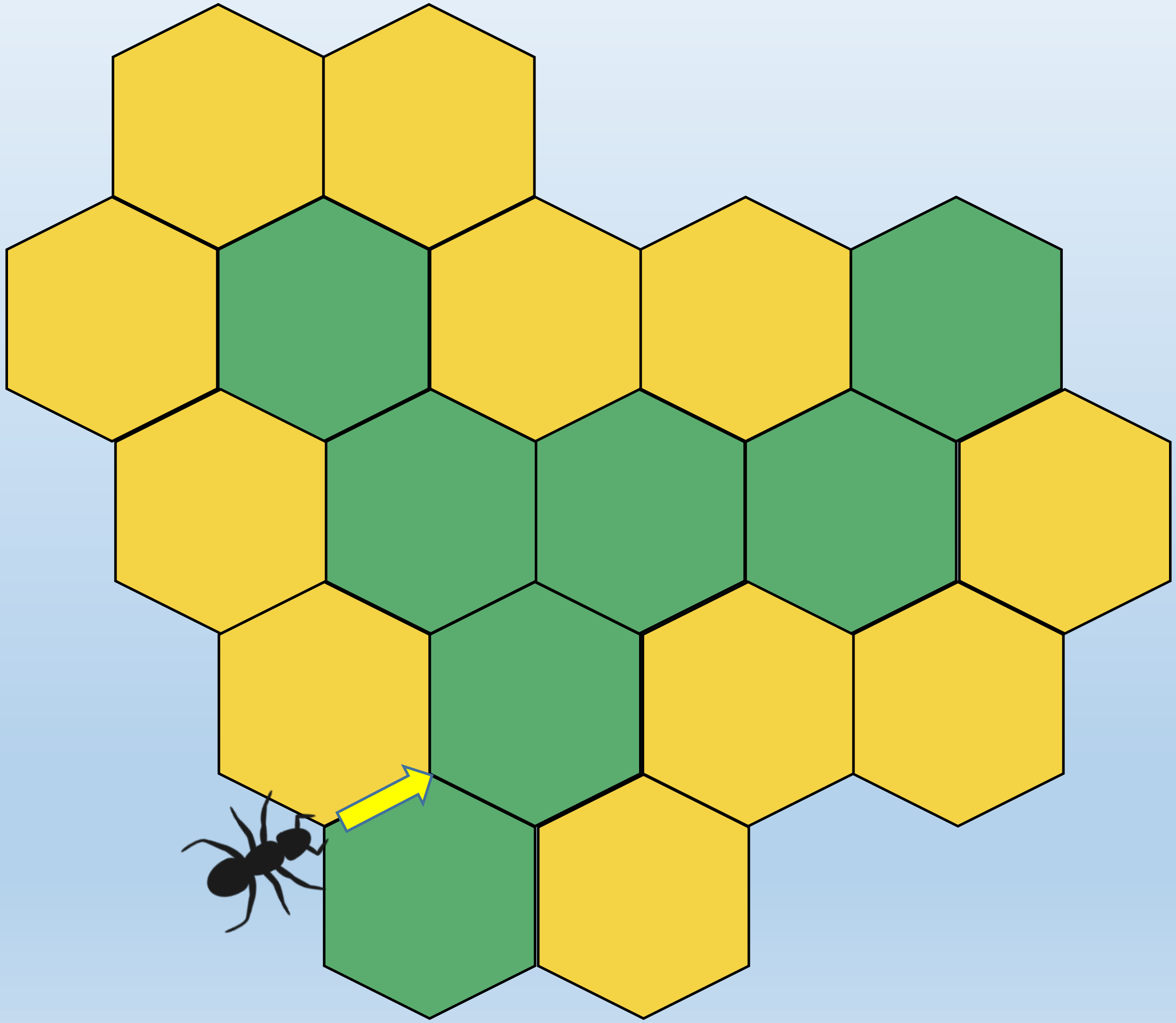
The interface algorithm



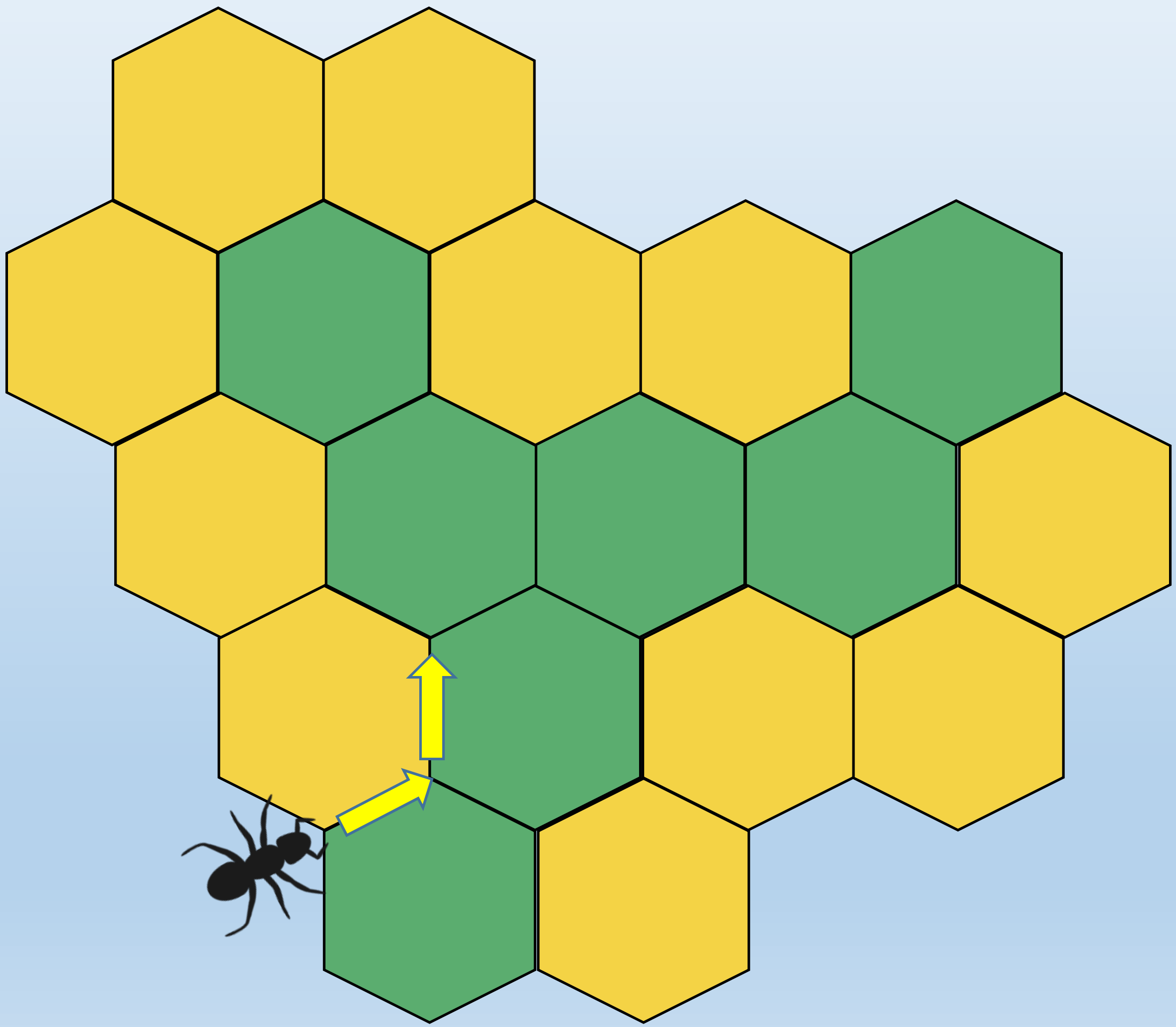
The interface algorithm



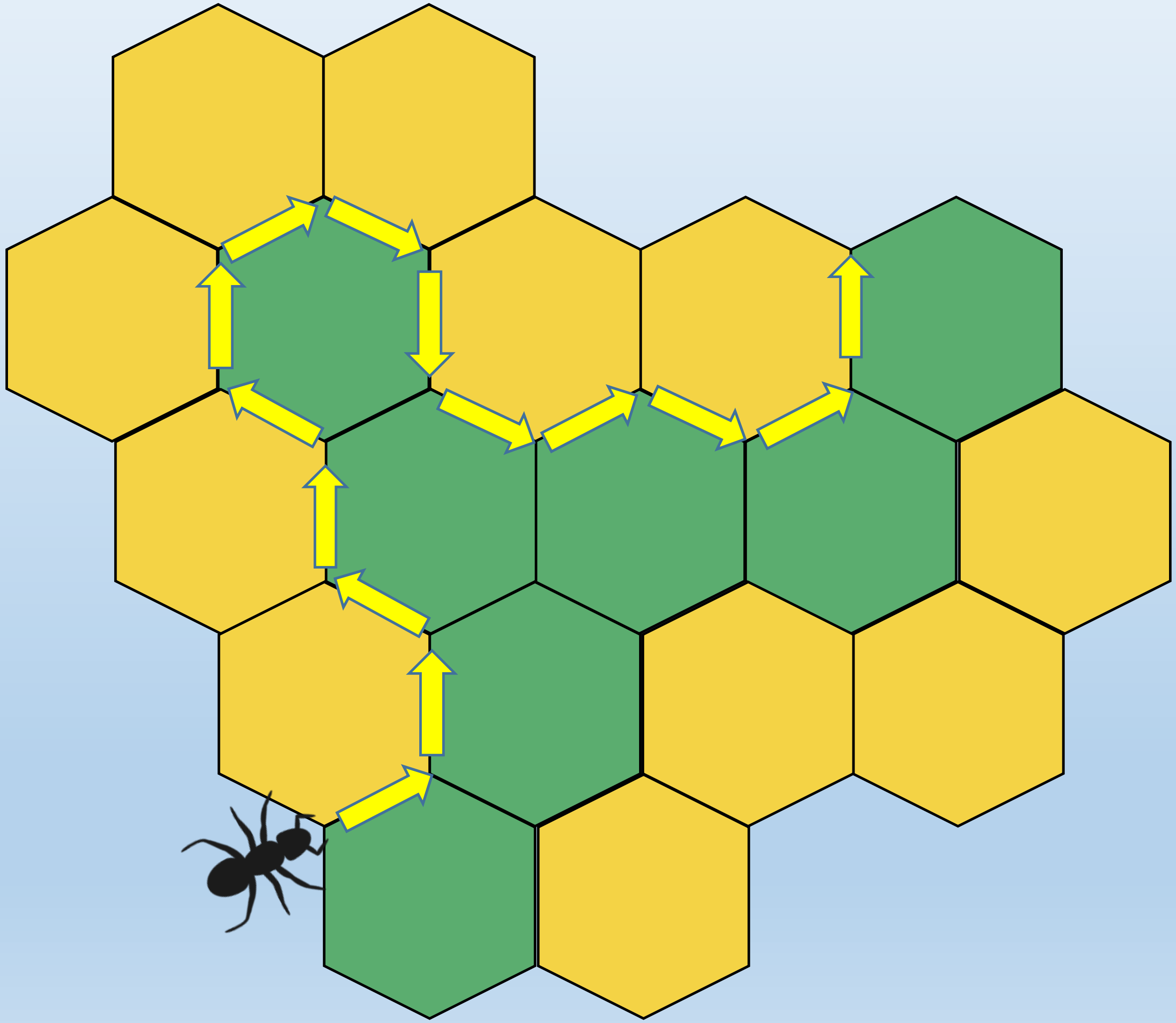
The interface algorithm



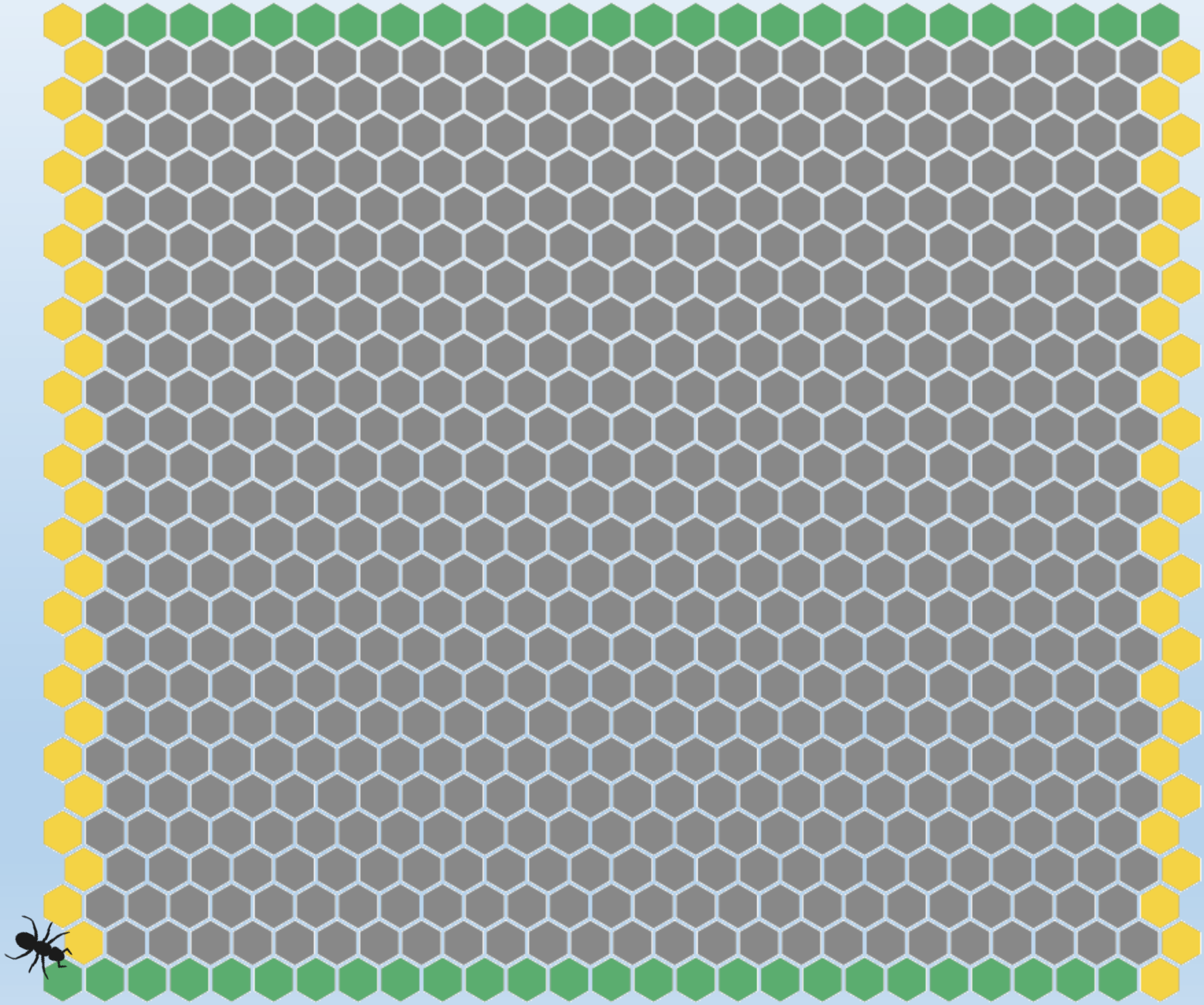
The interface algorithm



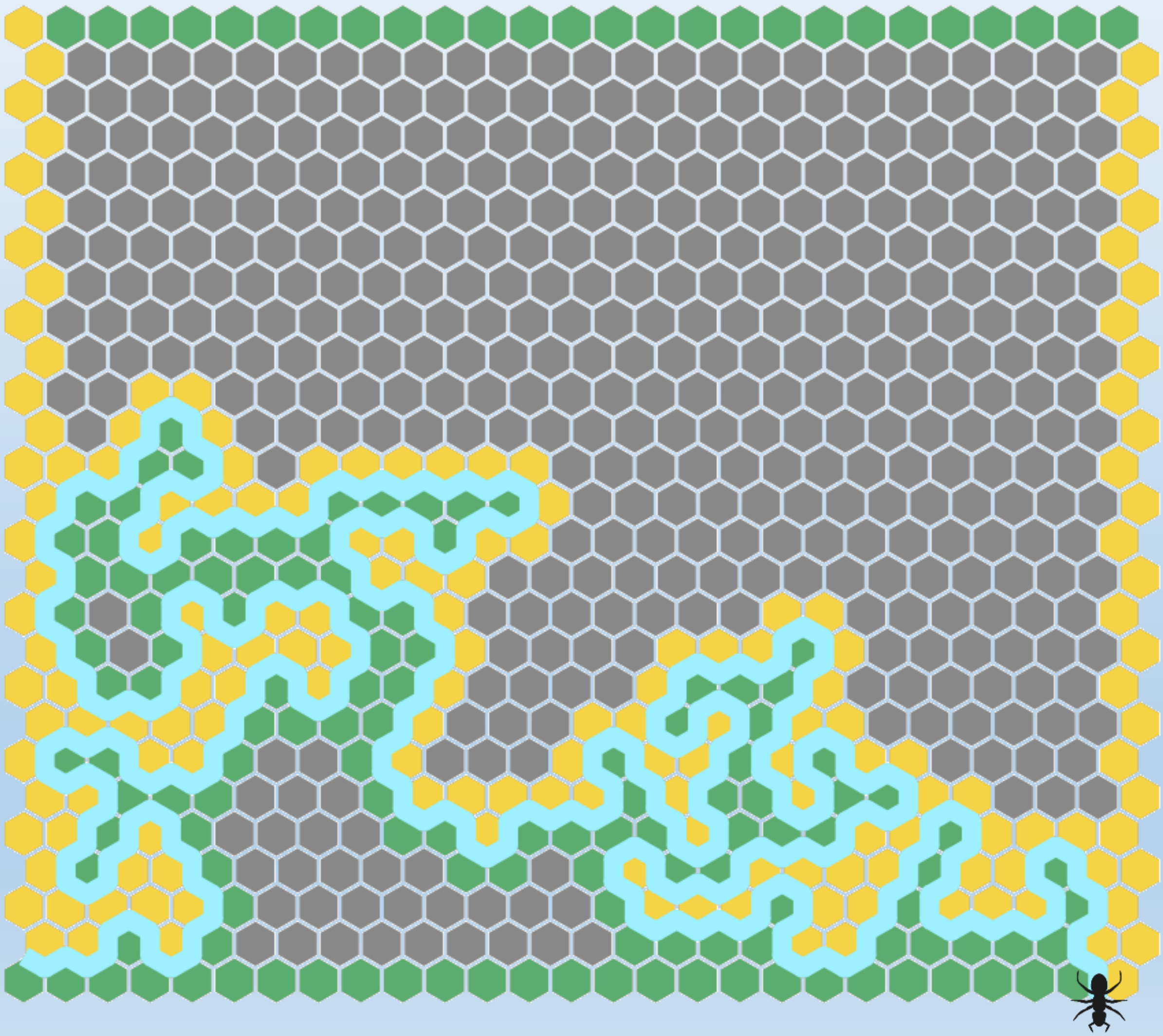
The interface algorithm



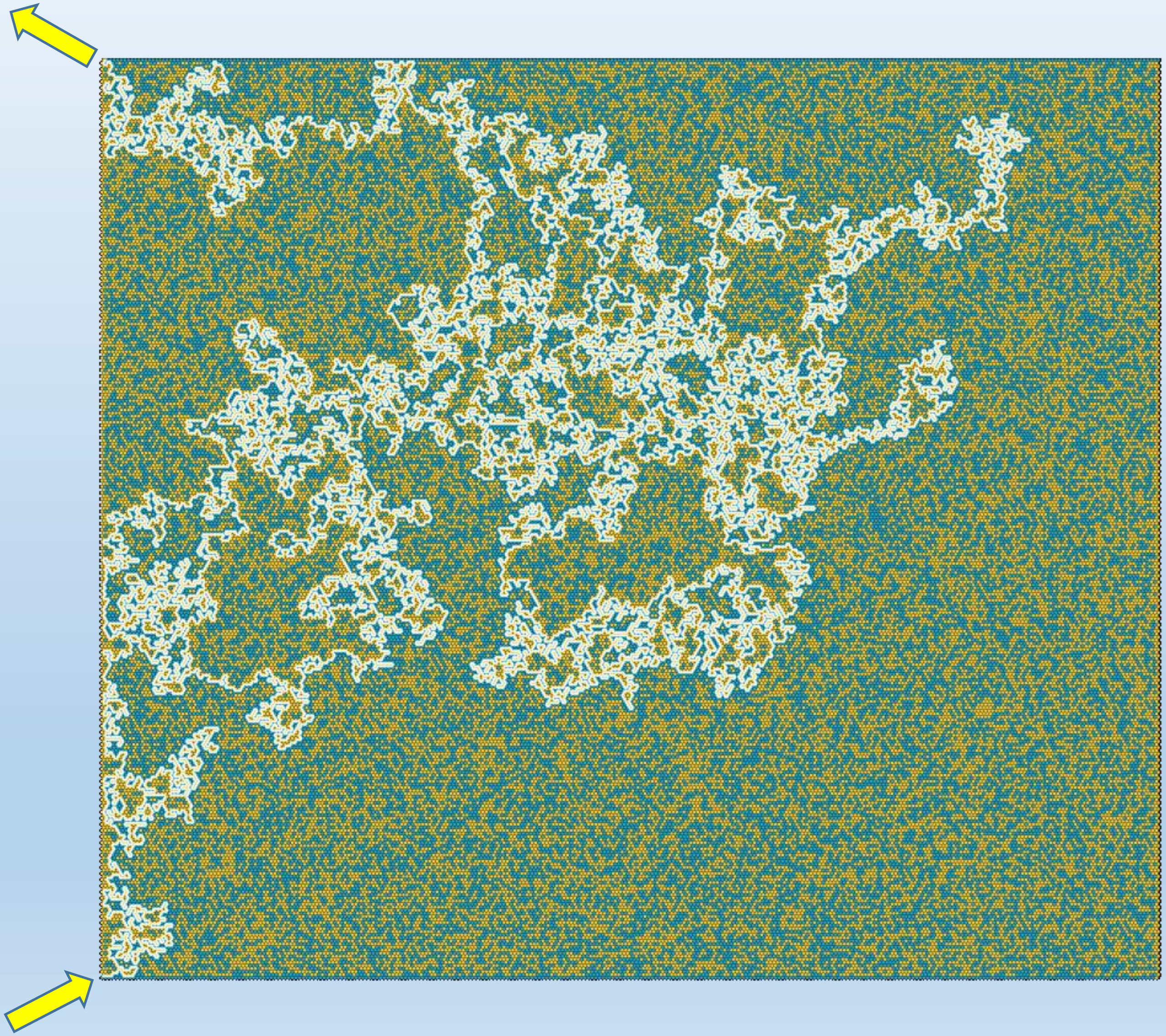
The interface algorithm



The interface algorithm



The interface algorithm



Fractional query algorithms

Fractional query algorithms

View 1:

True x	1	-1	1	-1	-1	1	1	-1

Fractional query algorithms

View 1:

True x	1	-1	1	-1	-1	1	1	-1
Known x	?	?	?	?	?	?	?	?

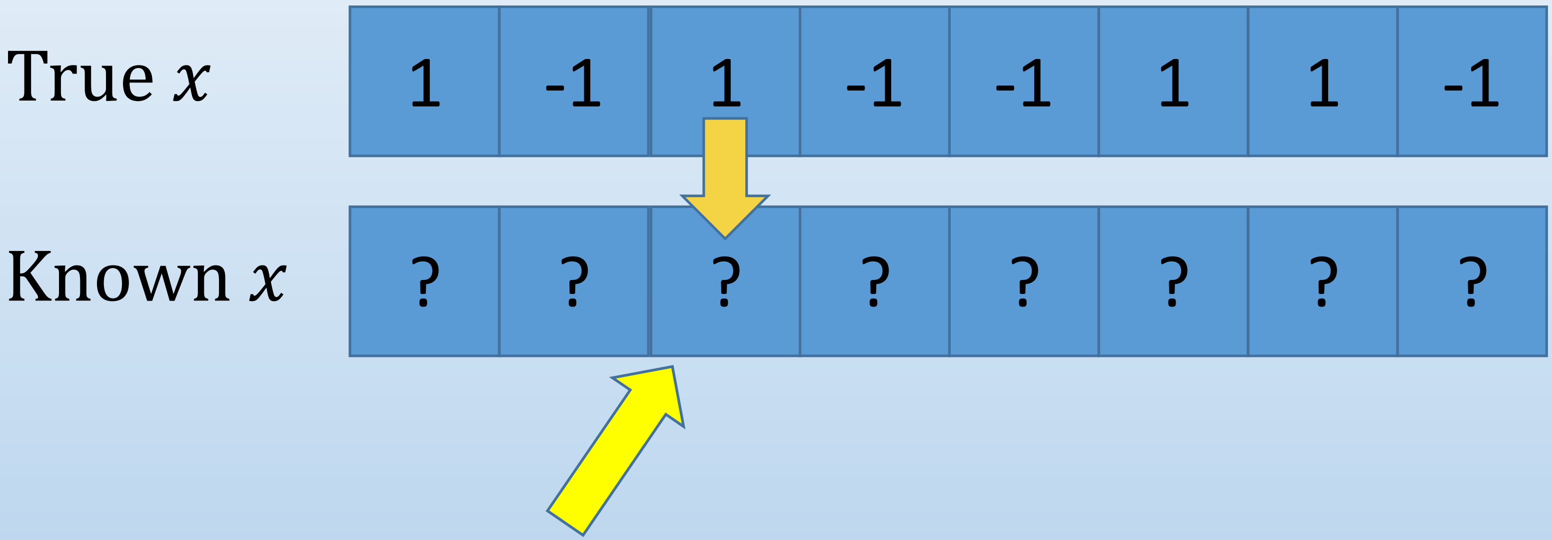
Fractional query algorithms

View 1:

True x	1	-1	1	-1	-1	1	1	-1
Known x	?	?	?	?	?	?	?	?

Fractional query algorithms

View 1:



Fractional query algorithms

View 1:

True x	1	-1	1	-1	-1	1	1	-1
Known x	?	?	1	?	?	?	?	?

Fractional query algorithms

View 1:

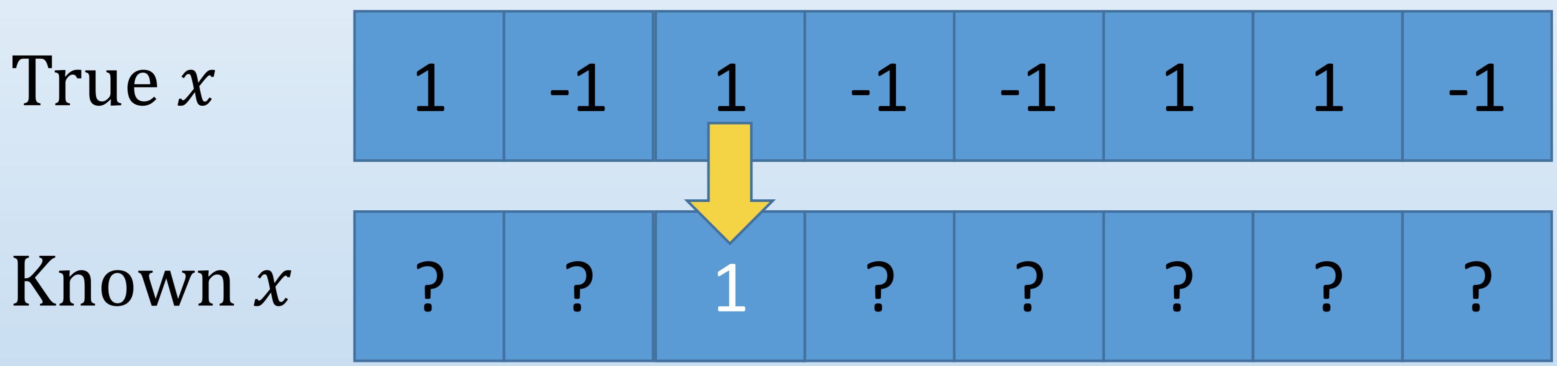
True x	1	-1	1	-1	-1	1	1	-1
Known x	?	?	1	?	?	?	?	?

View 2:

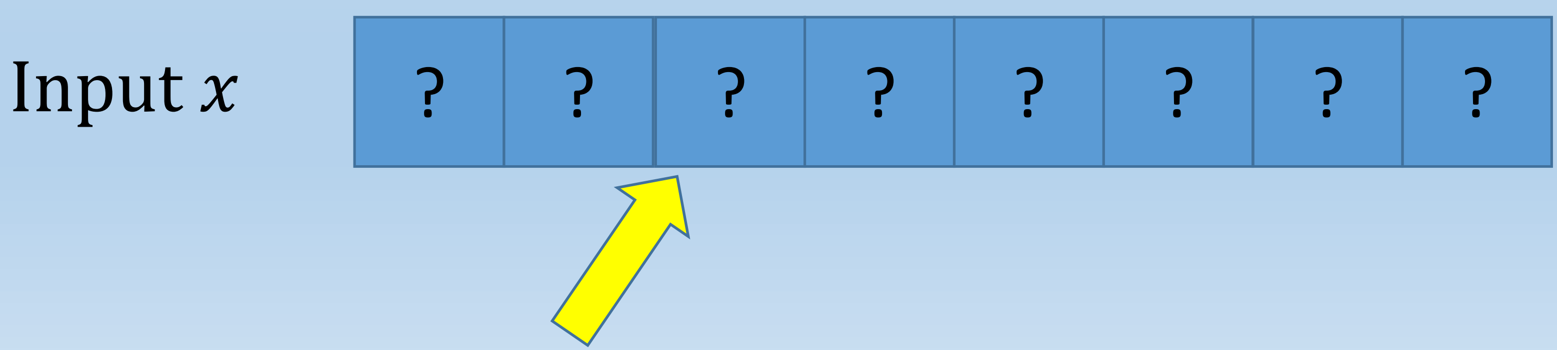
Input x	?	?	?	?	?	?	?	?

Fractional query algorithms

View 1:



View 2:



Fractional query algorithms

View 1:

True x	1	-1	1	-1	-1	1	1	-1
Known x	?	?	1	?	?	?	?	?

View 2:

Input x	?	?	?	?	?	?	?	?

$$x_i = \begin{cases} 1 & \text{w.p. } 1/2 \\ -1 & \text{w.p. } 1/2 \end{cases}$$

Fractional query algorithms

View 1:

True x	1	-1	1	-1	-1	1	1	-1
Known x	?	?	1	?	?	?	?	?

View 2:

Input x	?	?	1	?	?	?	?	?

$$x_i = \begin{cases} 1 & \text{w.p. } 1/2 \\ -1 & \text{w.p. } 1/2 \end{cases}$$

Fractional query algorithms

Input $x(t)$ is a stochastic process!

Fractional query algorithms

Input $x(t)$ is a stochastic process!

$x(0)$								
	0	0	0	0	0	0	0	0

Fractional query algorithms

Input $x(t)$ is a stochastic process!

$x(0)$	0	0	0	0	0	0	0	0	
$x(1)$	0	0	0	1	0	0	0	0	

Fractional query algorithms

Input $x(t)$ is a stochastic process!

$x(0)$	0	0	0	0	0	0	0	0
$x(1)$	0	0	0	1	0	0	0	0
$x(2)$	0	0	0	1	0	-1	0	0



Fractional query algorithms

Input $x(t)$ is a stochastic process!

$x(0)$	0	0	0	0	0	0	0	0	
$x(1)$	0	0	0	1	0	0	0	0	
$x(2)$	0	0	0	1	0	-1	0	0	
$\vdots$				$\vdots$					
$x(\tau)$	1	0	0	1	0	-1	-1	1	

Fractional query algorithms

Input $x(t)$ is a stochastic process!

$x(0)$	0	0	0	0	0	0	0	0	
$x(1)$	0	0	0	1	0	0	0	0	
$x(2)$	0	0	0	1	0	-1	0	0	
$\vdots$				$\vdots$					
$x(\tau)$	1	0	0	1	0	-1	-1	1	

$$\delta_i = \mathbb{P}[\text{bit } i \text{ is queried}] = \mathbb{E}[x_i(\tau)^2]$$

Fractional query algorithms

Let $\varepsilon > 0$.

Fractional query algorithms

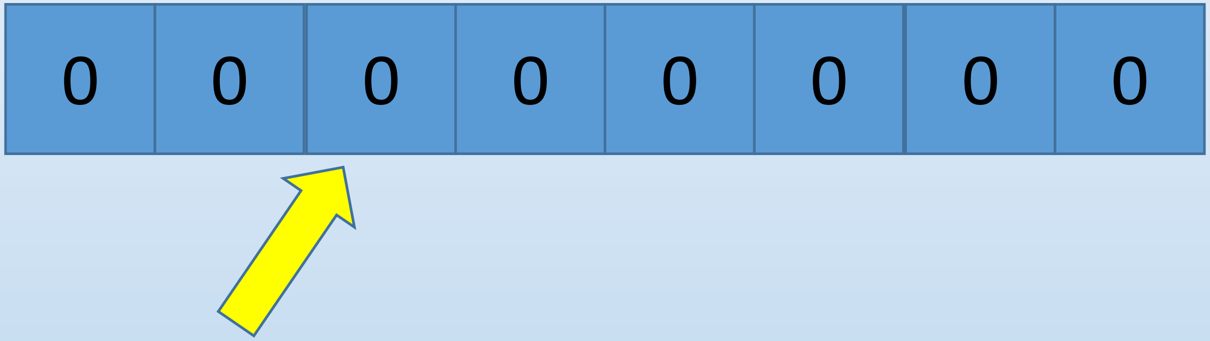
Let $\varepsilon > 0$.

$x(0)$	0	0	0	0	0	0	0	0

Fractional query algorithms

Let $\varepsilon > 0$.

$x(0)$



Fractional query algorithms

Let $\varepsilon > 0$.

$x(0)$								
	0	0	0	0	0	0	0	0

$$x_i = x_i + \begin{cases} \varepsilon & \text{w.p. } 1/2 \\ -\varepsilon & \text{w.p. } 1/2 \end{cases}$$

Fractional query algorithms

Let $\varepsilon > 0$.

$x(0)$								
	0	0	0	0	0	0	0	0

$$x_i = x_i + \begin{cases} \varepsilon & \text{w.p. } 1/2 \\ -\varepsilon & \text{w.p. } 1/2 \end{cases}$$

$x(1)$								
	0	0	ε	0	0	0	0	0

Fractional query algorithms

Let $\varepsilon > 0$.

$x(0)$								
	0	0	0	0	0	0	0	0

$$x_i = x_i + \begin{cases} \varepsilon & \text{w.p. } 1/2 \\ -\varepsilon & \text{w.p. } 1/2 \end{cases}$$

$x(1)$								
	0	0	ε	0	0	0	0	0

Fractional query algorithms

Let $\varepsilon > 0$.

$x(0)$

0	0	0	0	0	0	0	0

$$x_i = x_i + \begin{cases} \varepsilon & \text{w.p. } 1/2 \\ -\varepsilon & \text{w.p. } 1/2 \end{cases}$$

$x(2)$

0	0	ε	$-\varepsilon$	0	0	0	0

Fractional query algorithms

Let $\varepsilon > 0$.

$x(0)$

0	0	0	0	0	0	0	0

$$x_i = x_i + \begin{cases} \varepsilon & \text{w.p. } 1/2 \\ -\varepsilon & \text{w.p. } 1/2 \end{cases}$$

$x(2)$

0	0	ε	$-\varepsilon$	0	0	0	0

Fractional query algorithms

Let $\varepsilon > 0$.

$x(0)$

0	0	0	0	0	0	0	0

$$x_i = x_i + \begin{cases} \varepsilon & \text{w.p. } 1/2 \\ -\varepsilon & \text{w.p. } 1/2 \end{cases}$$

$x(3)$

0	0	0	$-\varepsilon$	0	0	0	0

Fractional query algorithms

Let $\varepsilon > 0$.

$x(0)$

0	0	0	0	0	0	0	0

$$x_i = x_i + \begin{cases} \varepsilon & \text{w.p. } 1/2 \\ -\varepsilon & \text{w.p. } 1/2 \end{cases}$$

$x(3)$

0	0	0	$-\varepsilon$	0	0	0	0

Can only do this if $x_i(t) \in (-1,1)$.

Computing with fractional inputs

Computing with fractional inputs

Every $f: \{-1,1\}^n \to \{-1,1\}$ can be written as

$$f(x) = \sum_{S \subseteq [n]} \hat{f}(S) \prod_{i \in S} x_i$$

This is a real-valued polynomial.

Computing with fractional inputs

Every $f: \{-1,1\}^n \to \{-1,1\}$ can be written as

$$f(x) = \sum_{S \subseteq [n]} \hat{f}(S) \prod_{i \in S} x_i$$

This is a real-valued polynomial.

$$f(X(t)) = \sum_{S \subseteq [n]} \hat{f}(S) \prod_{i \in S} X_i(t)$$

Computing with fractional inputs

Every $f: \{-1,1\}^n \to \{-1,1\}$ can be written as

$$f(x) = \sum_{S \subseteq [n]} \hat{f}(S) \prod_{i \in S} x_i$$

This is a real-valued polynomial.

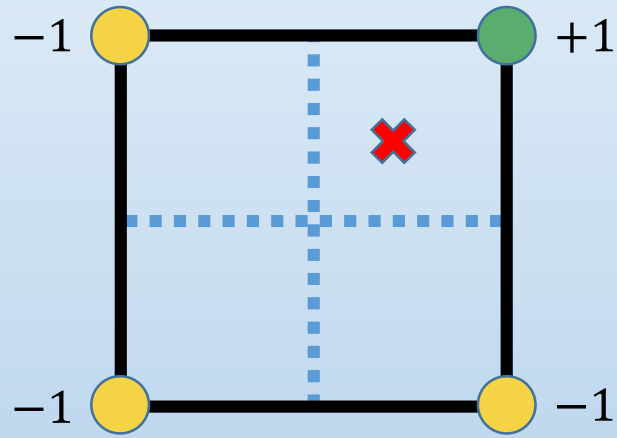
$$f(X(t)) = \sum_{S \subseteq [n]} \hat{f}(S) \prod_{i \in S} X_i(t)$$

In fact, it is an interpolation!

Computing with fractional inputs

Every $f: \{-1,1\}^n \to \{-1,1\}$ can be written as

$$f(x) = \sum_{S \subseteq [n]} \hat{f}(S) \prod_{i \in S} x_i$$



This is a real-valued polynomial.

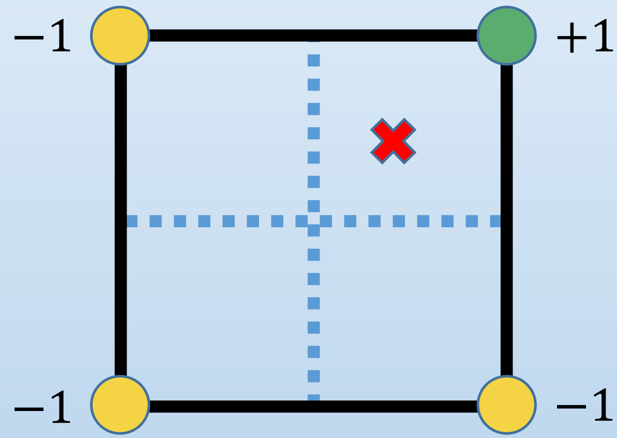
$$f(X(t)) = \sum_{S \subseteq [n]} \hat{f}(S) \prod_{i \in S} X_i(t)$$

In fact, it is an interpolation!

Computing with fractional inputs

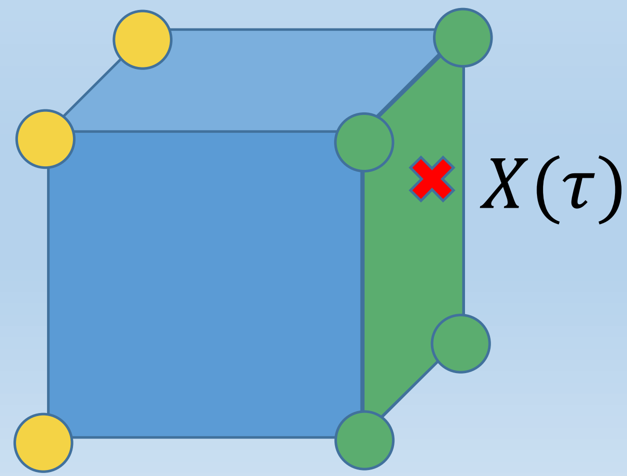
Every $f: \{-1,1\}^n \to \{-1,1\}$ can be written as

$$f(x) = \sum_{S \subseteq [n]} \hat{f}(S) \prod_{i \in S} x_i$$



This is a real-valued polynomial.

$$f(X(t)) = \sum_{S \subseteq [n]} \hat{f}(S) \prod_{i \in S} X_i(t)$$

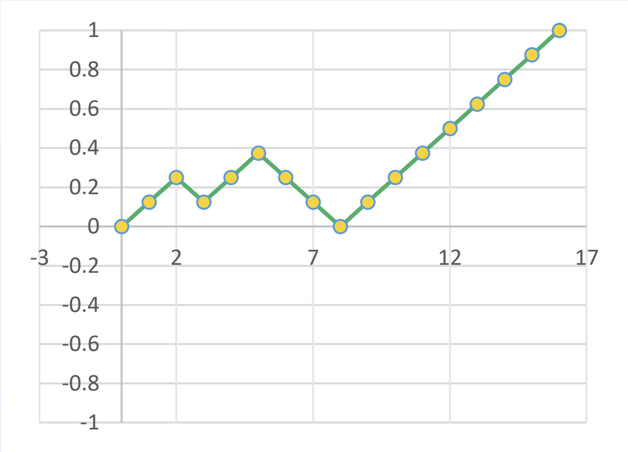


In fact, it is an interpolation!

Where is the unknown input?

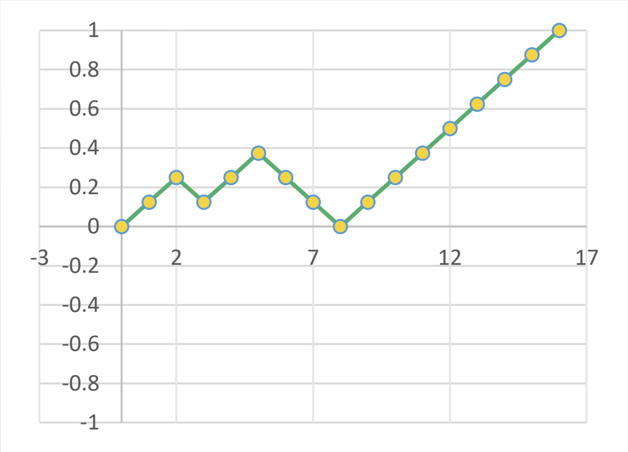
Where is the unknown input?

Individual bits perform random walks



Where is the unknown input?

Individual bits perform random walks



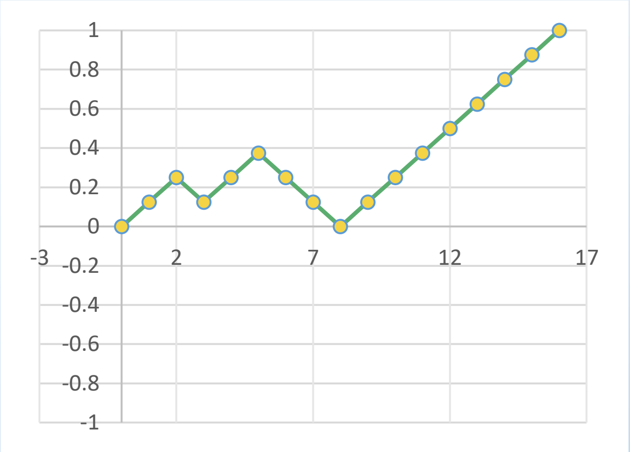
In the end,

$$X(\infty) \in \{-1,1\}^n$$

This is the "final input".

Where is the unknown input?

Individual bits perform random walks



In the end,

$$X(\infty) \in \{-1,1\}^n$$

This is the “final input”.

The current value gives a hint to the future:

$$\mathbb{P}[\, X_i(\infty) = 1 \mid X_i(t) \,] = \frac{1 + X_i(t)}{2}$$

Comparing with classical algorithms

Comparing with classical algorithms

For classical decision trees,

$$\delta_i = \mathbb{P}[\text{bit } i \text{ is queried}] = \mathbb{E}[X_i(\tau)^2].$$

We define δ_i similarly for fractional algorithms.

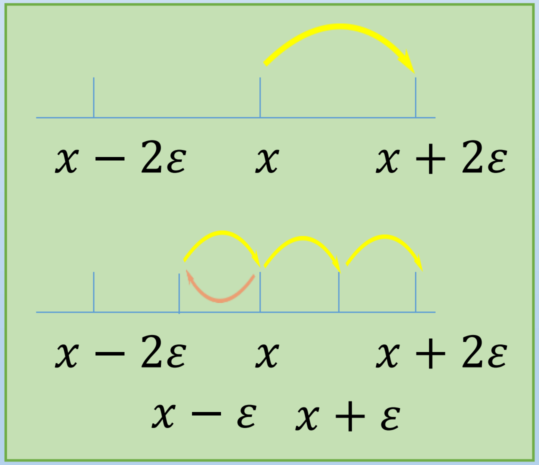
Comparing with classical algorithms

For classical decision trees,

$$\delta_i = \mathbb{P}[\text{bit } i \text{ is queried}] = \mathbb{E}[X_i(\tau)^2].$$

We define δ_i similarly for fractional algorithms.

Fact: $\min\limits_{\varepsilon\text{–algs}} \delta_i \leq \min\limits_{2\varepsilon\text{–algs}} \delta_i$

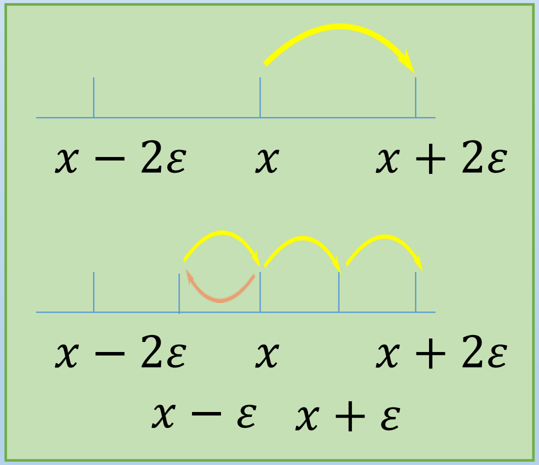


Comparing with classical algorithms

For classical decision trees,

$$\delta_i = \mathbb{P}[\text{bit } i \text{ is queried}] = \mathbb{E}[X_i(\tau)^2].$$

We define δ_i similarly for fractional algorithms.



Fact: $\min\limits_{\varepsilon\text{–algs}} \delta_i \leq \min\limits_{2\varepsilon\text{–algs}} \delta_i$

Why this cost?

Fact: $\mathbb{E}[X_i(\tau)^2] = \mathbb{E}[X_i]_\tau = \varepsilon^2 \mathbb{E}[\#\text{times } i \text{ was chosen}]$

The Schramm-Steif Theorem

Let $f_n: \{-1,1\}^n \to \{-1,1\}$ be a sequence of Booelan functions.

Let T_n be a bit-reveal algorithm for f_n, and

$$\delta(n) := \max_i \delta_i = \max_i \mathbb{P}[T_n \text{ reveals bit } i]\,.$$

Theorem: If $\delta \to 0$, then f_n is noise sensitive.

The faster $\delta \to 0$, the more noise sensitive it is!

The Schramm-Steif Theorem

Let $f_n\colon \{-1,1\}^n \to \{-1,1\}$ be a sequence of Booelan functions.

fractional

Let T_n be a bit-reveal algorithm for f_n, and

$$\delta(n) := \max_i \delta_i = \max_i \mathbb{P}[T_n \text{ reveals bit } i] \,.$$

Theorem: If $\delta \to 0$, then f_n is noise sensitive.

The faster $\delta \to 0$, the more noise sensitive it is!

The Schramm-Steif Theorem

Let $f_n\colon \{-1,1\}^n \to \{-1,1\}$ be a sequence of Booelan functions.

Let T_n be a ~~bit-reveal~~ fractional algorithm for f_n, and

$$\delta(n) := \max_i \delta_i = \max_i \mathbb{E}[X_i(\tau)^2].$$

Theorem: If $\delta \to 0$, then f_n is noise sensitive.

The faster $\delta \to 0$, the more noise sensitive it is!

The $\frac{\text{Schramm}}{\text{Steif}}$ Theorem

Let $f_n: \{-1,1\}^n \to \{-1,1\}$ be a sequence of Booelan functions.

Let T_n be a ~~bit-reveal~~ fractional algorithm for f_n, and

$$\delta(n) := \max_i \delta_i = \max_i \mathbb{E}[X_i(\tau)^2].$$

Theorem: If $\delta \to 0$, then f_n is noise sensitive.

The faster $\delta \to 0$, the more noise sensitive it is!

Sending $\varepsilon \to 0$

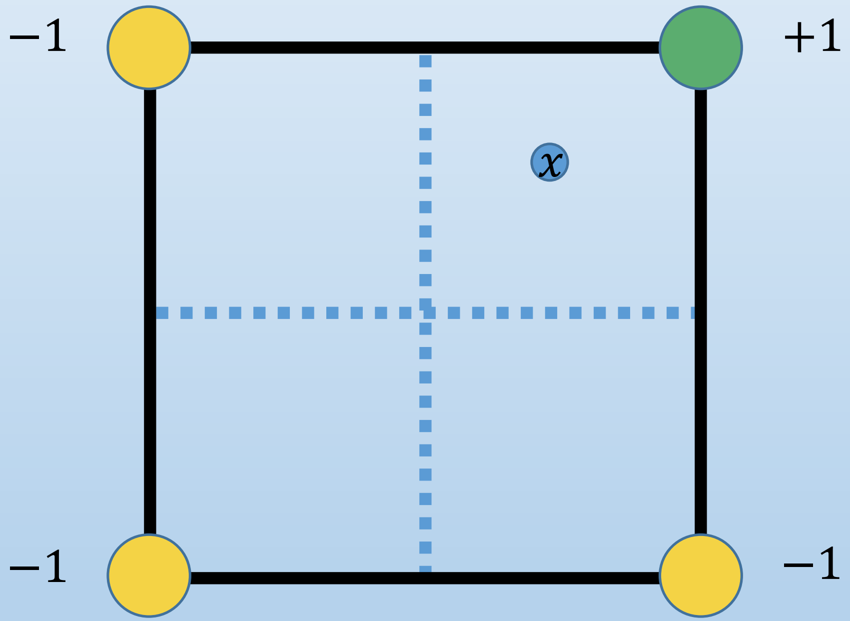
Sending $\varepsilon \to 0$

Let $u_\varepsilon(x)\colon [-1,1]^n \to \mathbb{R} = \text{best alg when } x(0) = x$.

Sending $\varepsilon \to 0$

Let $u_\varepsilon(x)$: $[-1,1]^n \to \mathbb{R}$ = best alg when $x(0) = x$.

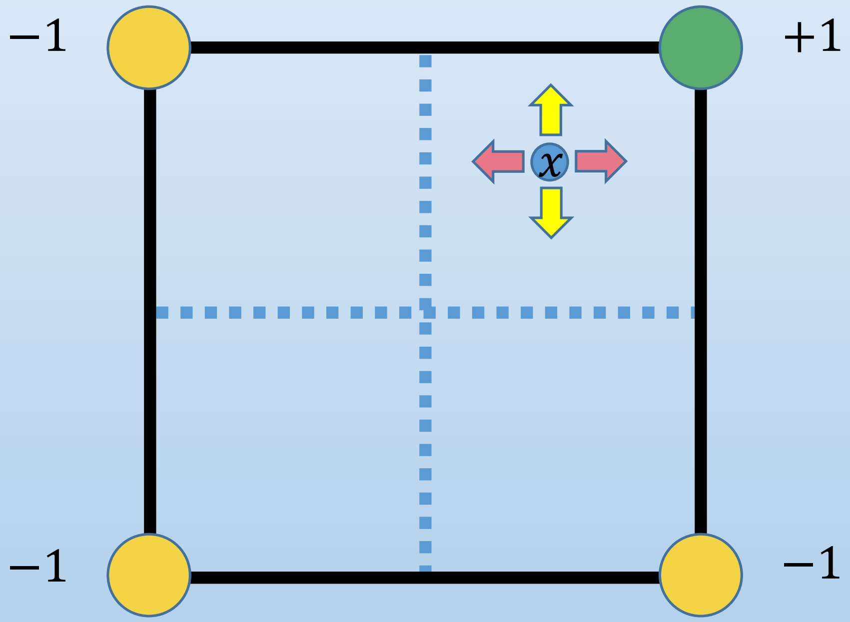
In which direction to go?



Sending $\varepsilon \to 0$

Let $u_\varepsilon(x)$: $[-1,1]^n \to \mathbb{R} = \text{best alg when } x(0) = x$.

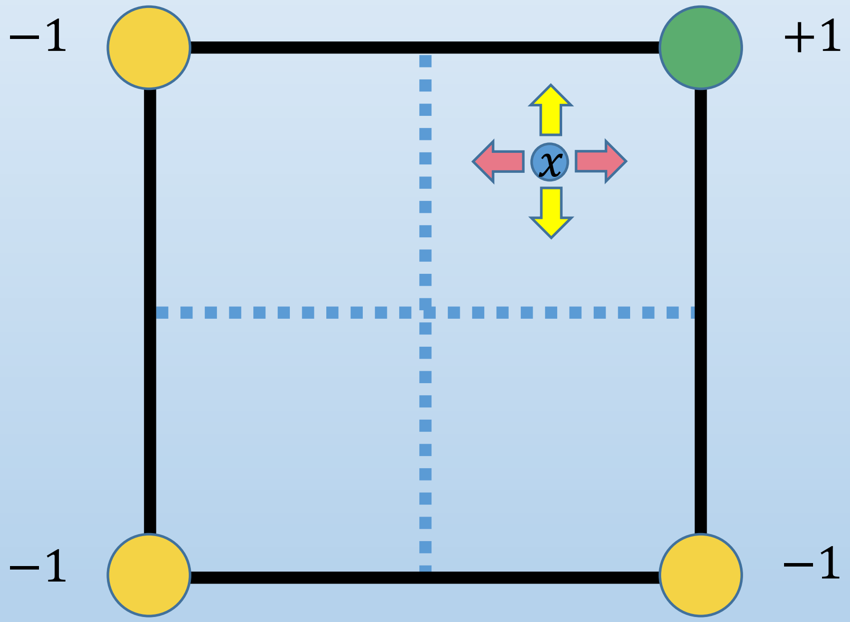
In which direction to go?



Sending $\varepsilon \to 0$

Let $u_\varepsilon(x)$: $[-1,1]^n \to \mathbb{R} = \text{best alg when } x(0) = x$.

In which direction to go?

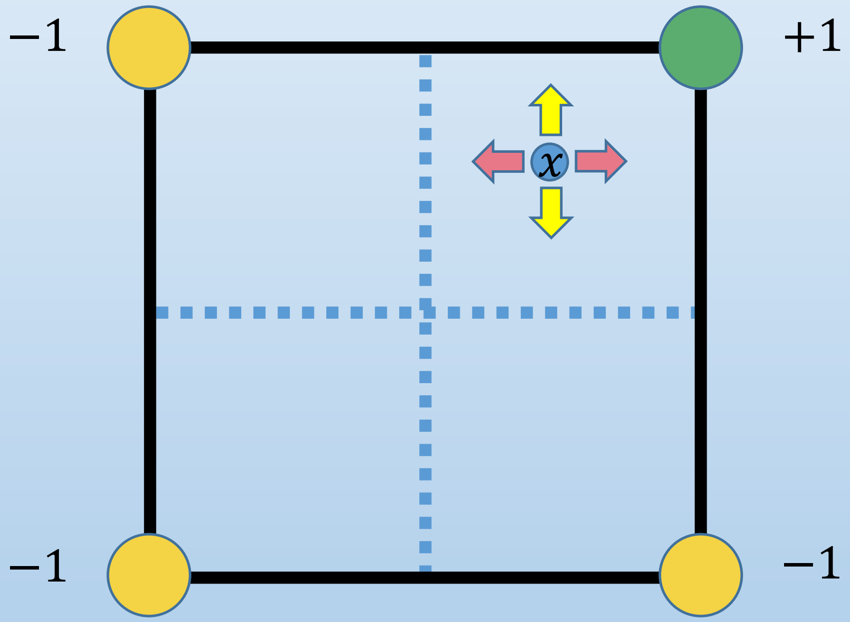


$$u_\varepsilon(x) = \min_i \frac{u_\varepsilon(x + \varepsilon e_i) + u_\varepsilon(x - \varepsilon e_i)}{2}$$

Sending $\varepsilon \to 0$

Let $u_\varepsilon(x)\colon [-1,1]^n \to \mathbb{R} = \text{best alg when } x(0) = x$.

In which direction to go?



$$u_\varepsilon(x) = \min_i \frac{u_\varepsilon(x + \varepsilon e_i) + u_\varepsilon(x - \varepsilon e_i)}{2} + \varepsilon^2$$

Sending $\varepsilon \to 0$

$$\left(u_\varepsilon(x) = \min_i \frac{u_\varepsilon(x + \varepsilon e_i) + u_\varepsilon(x - \varepsilon e_i)}{2} + \varepsilon^2 \right)$$

Sending $\varepsilon \to 0$

$$\left(u_\varepsilon(x) = \min_i \frac{u_\varepsilon(x + \varepsilon e_i) + u_\varepsilon(x - \varepsilon e_i)}{2} + \varepsilon^2 \right)$$

Theorem: Define $u = \lim_{\varepsilon \to 0} u_\varepsilon$. Then

$$\min_i \frac{\partial^2 u}{\partial x_i^2} + 2 = 0\,.$$

- "Axis-aligned Laplacian" equation.

Sending $\varepsilon \to 0$

$$\left(u_\varepsilon(x) = \min_i \frac{u_\varepsilon(x+\varepsilon e_i) + u_\varepsilon(x-\varepsilon e_i)}{2} + \varepsilon^2\right)$$

Theorem: Define $u = \lim_{\varepsilon \to 0} u_\varepsilon$. Then

$$\min_i \frac{\partial^2 u}{\partial x_i^2} + 2 = 0\,.$$

- "Axis-aligned Laplacian" equation.
- $u(0)$ might tell us something about δ!
  - Solving a PDE can give us noise-sensitivity.

The OR function

OR: $f(x) = 1$ if any $x_i = 1$.

The OR function

OR: $f(x) = 1$ if any $x_i = 1$.

- Classical alg: just query bits. $\mathbb{E}[\text{runtime}] = 2.$

The OR function

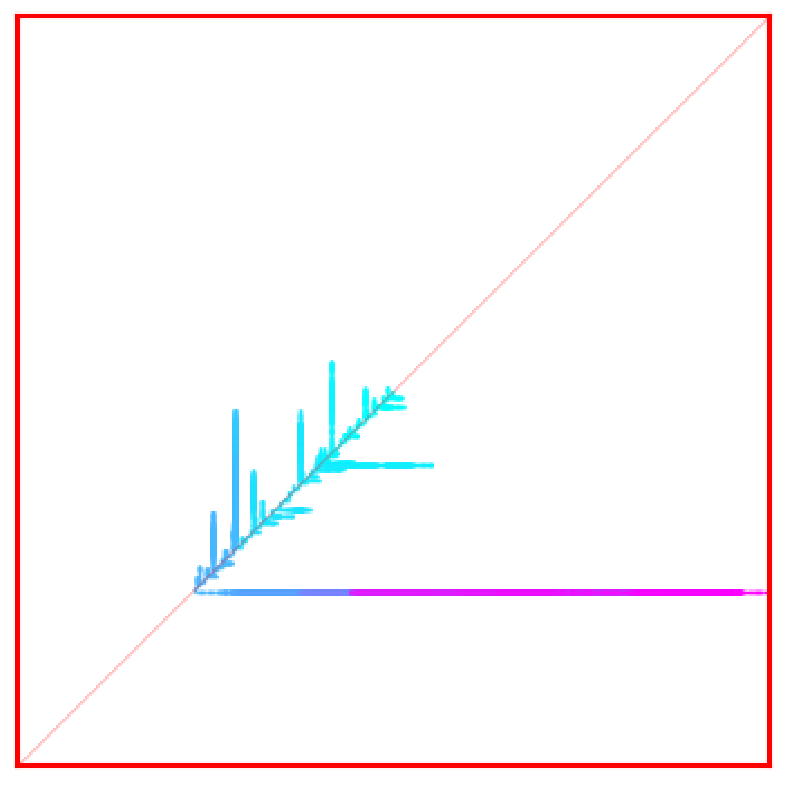
OR: $f(x) = 1$ if any $x_i = 1$.

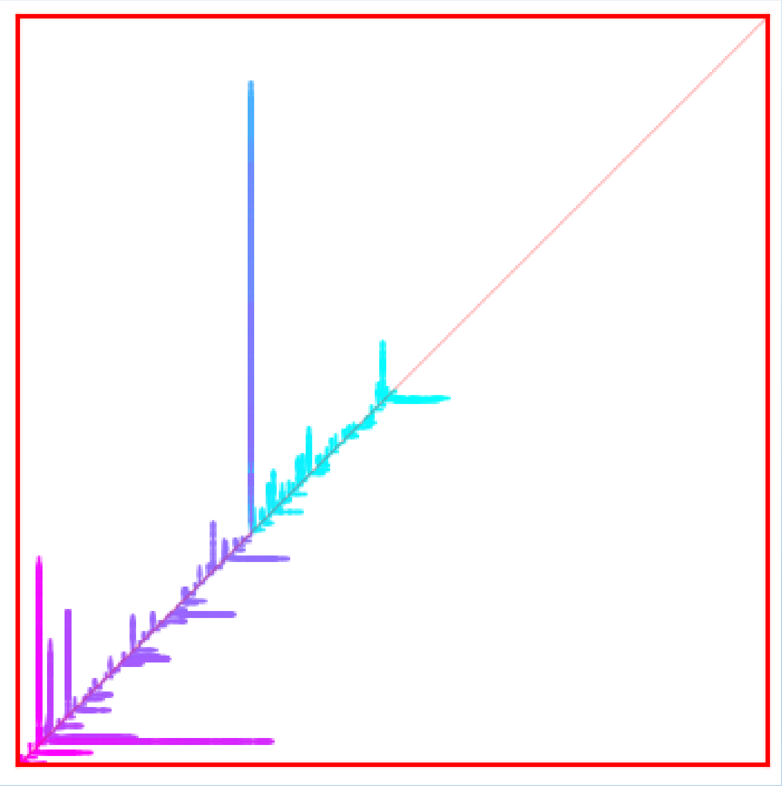
- Classical alg: just query bits. $\mathbb{E}[\text{runtime}] = 2$.

- Fractional alg: ???

The OR function

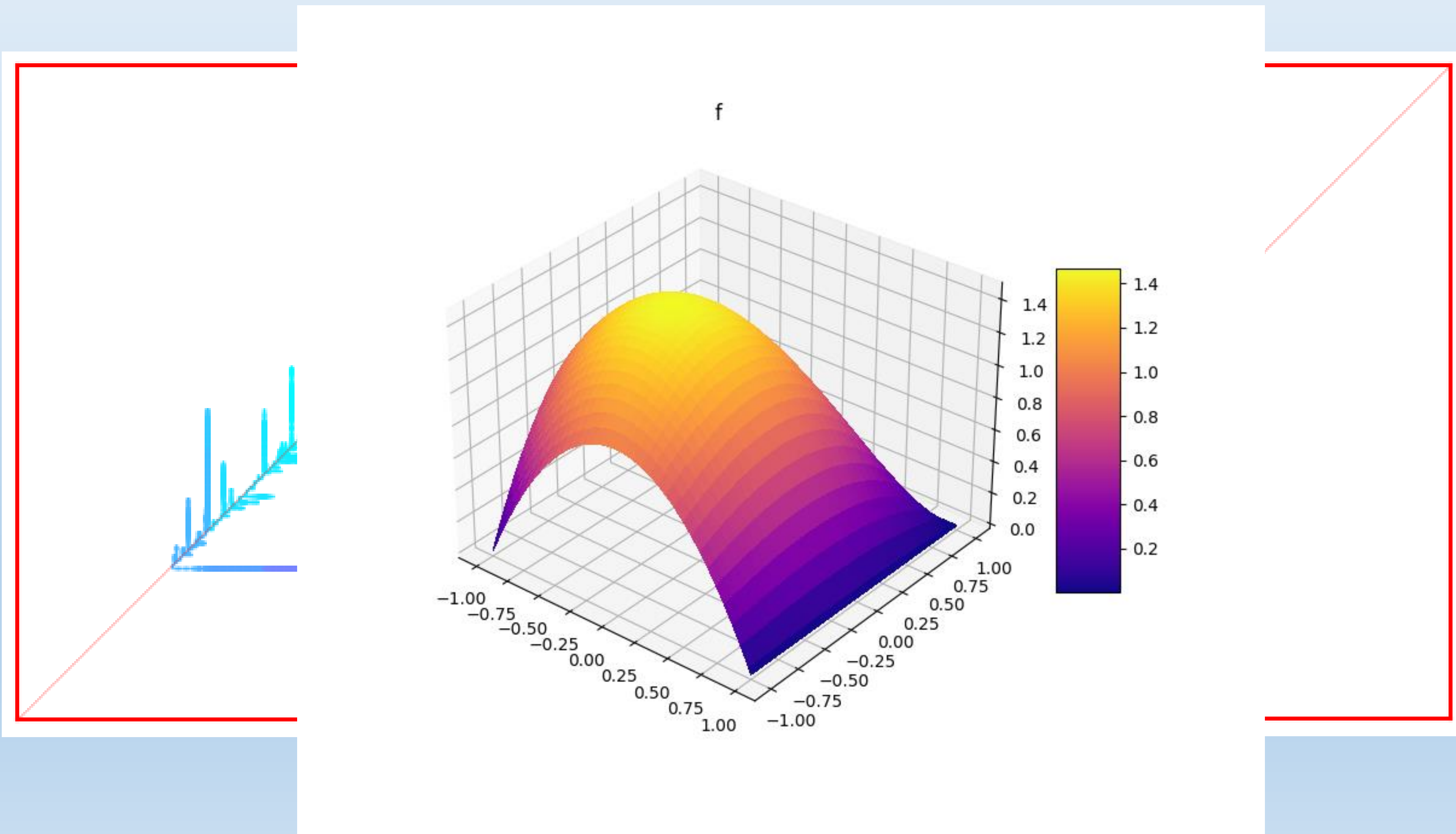
OR: $f(x) = 1$ if any $x_i = 1$.





The OR function

OR: $f(x) = 1$ if any $x_i = 1$.



The big question

The big question

- Is $P = NP$?

The big question

- ~~Is $P = NP$?~~

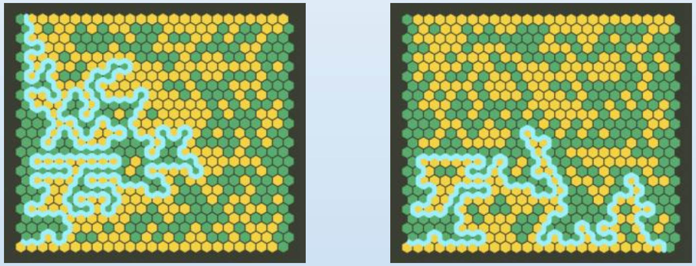
- Is there a class of functions f such that

$$\lim_{n\to\infty} \liminf_{\varepsilon\to 0} \frac{\delta(f,\varepsilon)}{\delta(f,1)} = 0?$$

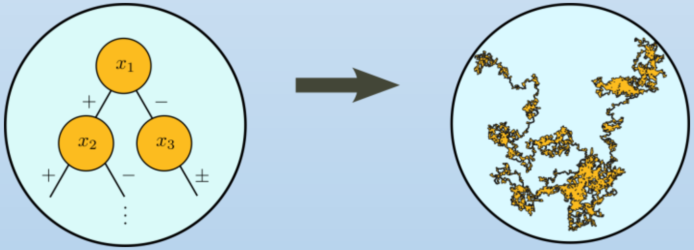
(specifically, what about percolation?)

Overview

- Boolean functions, noise-sensitivity, revealment algorithms

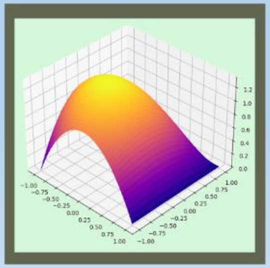


- Fractional algorithms can do better



- A limiting partial differential equation

$$\min_i \frac{\partial^2 u}{\partial x_i^2} = -2.$$

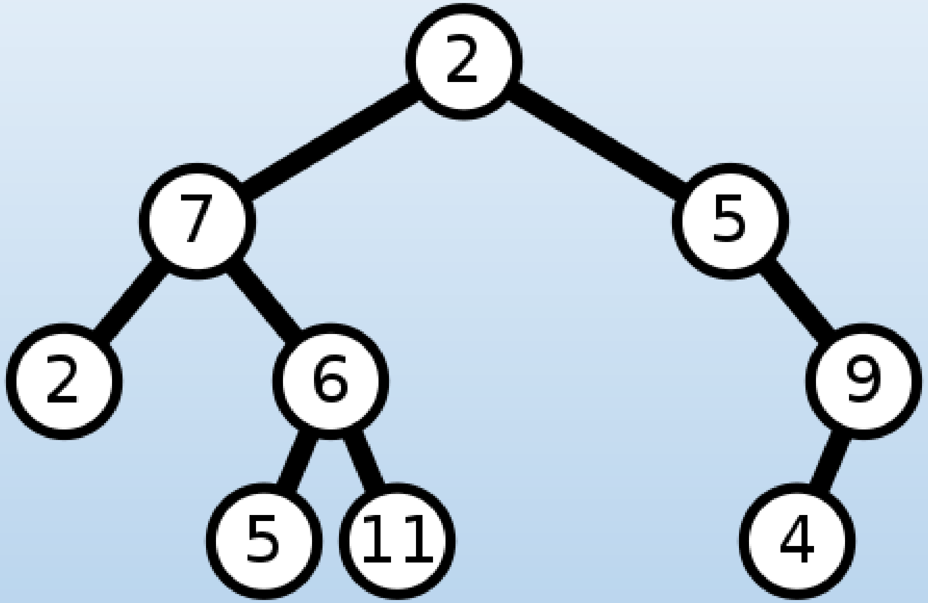


House

Person

Tree

https://practicalpie.com/house-person-tree-test/



Also a tree