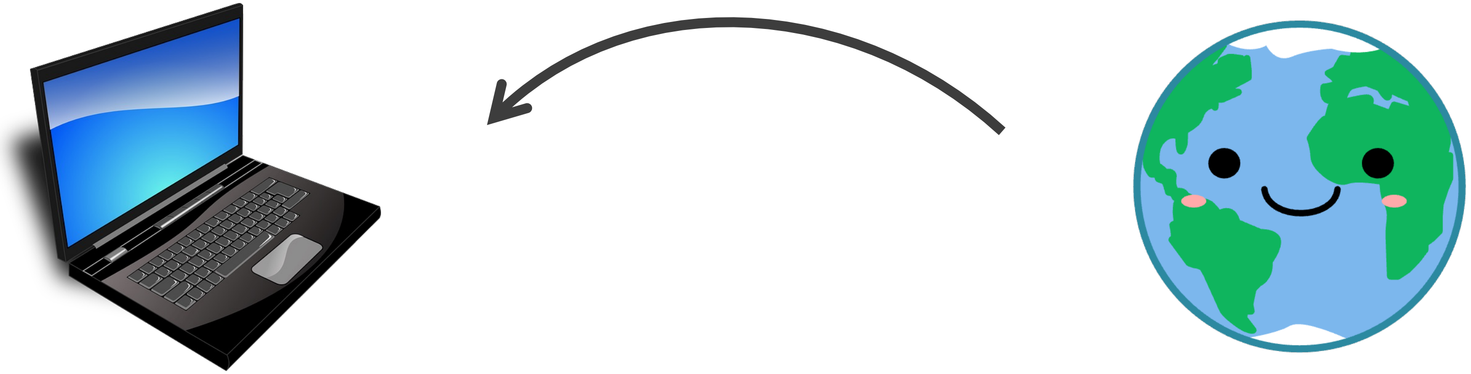
Adversarially Robust Streaming Algorithms

Omri Ben-Eliezer

Part 1: Model and motivation

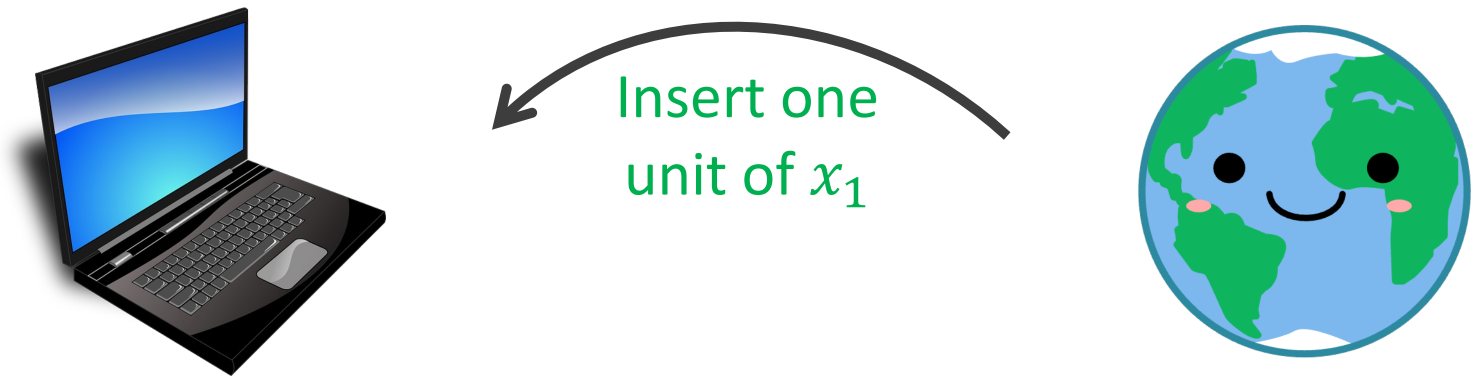
Streaming: The classical model

- Stream $(x_1, \Delta_1), \ldots, (x_m, \Delta_m)$ of updates, $x_i \in [n]$, $\Delta_i \in \mathbb{R}$
- Goal: compute $(1+\epsilon)$-approx of f over (frequency vector v of) stream
  (f = # distinct elements / ℓ_2-norm / entropy / heavy hitters / …)



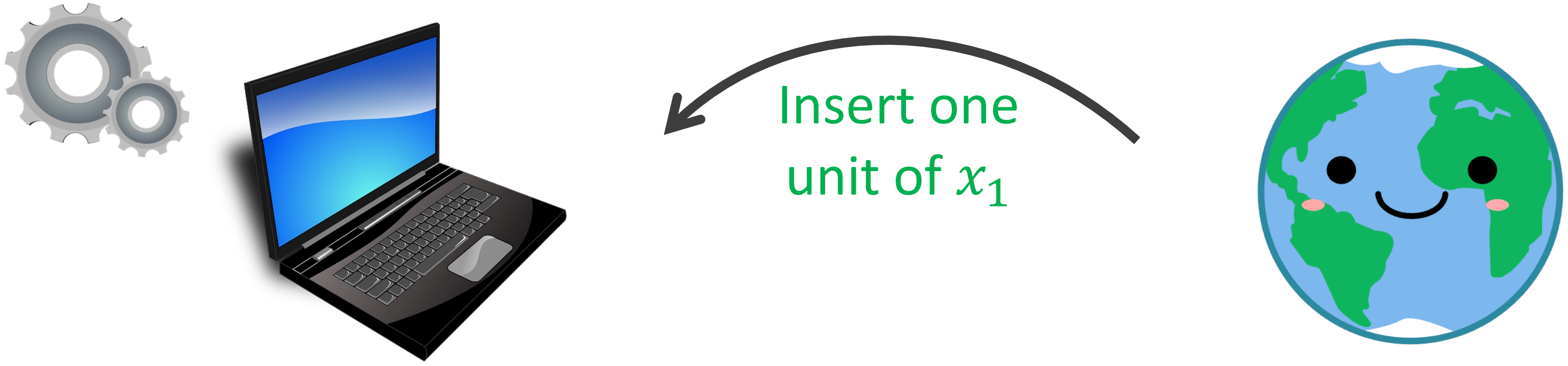
Streaming: The classical model

- Stream $(x_1, \Delta_1), \ldots, (x_m, \Delta_m)$ of updates, $x_i \in [n]$, $\Delta_i \in \mathbb{R}$
- Goal: compute $(1+\epsilon)$-approx of f over (frequency vector v of) stream
  (f = # distinct elements / ℓ_2-norm / entropy / heavy hitters / …)



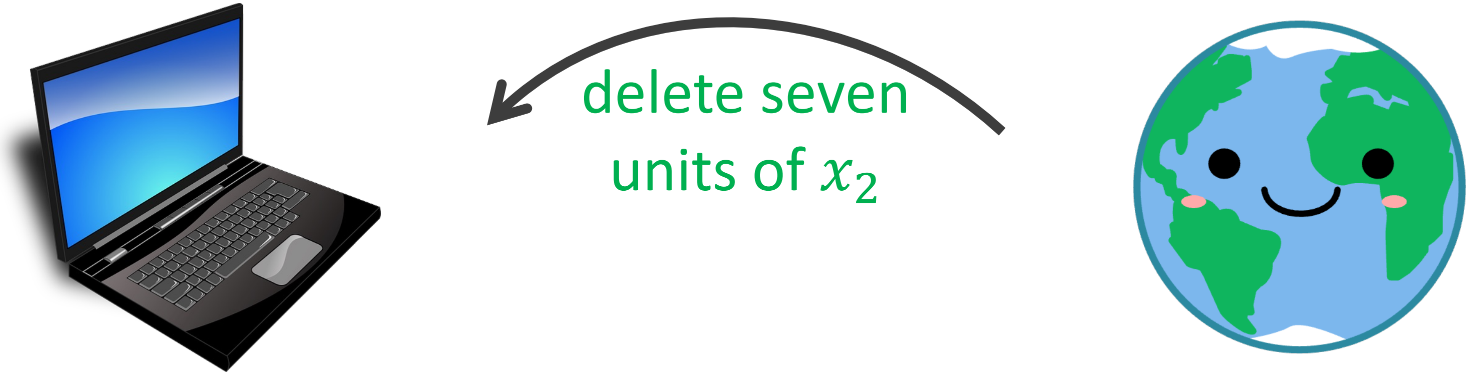
Streaming: The classical model

- Stream $(x_1, \Delta_1), \ldots, (x_m, \Delta_m)$ of updates, $x_i \in [n]$, $\Delta_i \in \mathbb{R}$
- Goal: compute $(1 + \epsilon)$-approx of f over (frequency vector v of) stream
  (f = # distinct elements / ℓ_2-norm / entropy / heavy hitters / …)



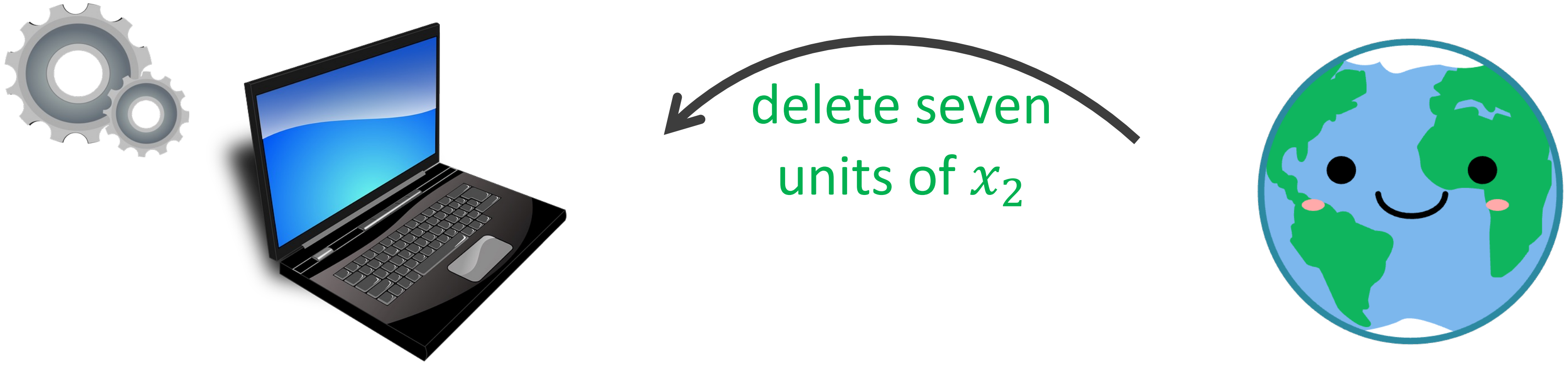
Streaming: The classical model

- Stream $(x_1, \Delta_1), \ldots, (x_m, \Delta_m)$ of updates, $x_i \in [n]$, $\Delta_i \in \mathbb{R}$
- Goal: compute $(1+\epsilon)$-approx of f over (frequency vector v of) stream
  (f = # distinct elements / ℓ_2-norm / entropy / heavy hitters / …)



Streaming: The classical model

- Stream $(x_1, \Delta_1), \ldots, (x_m, \Delta_m)$ of updates, $x_i \in [n]$, $\Delta_i \in \mathbb{R}$
- Goal: compute $(1+\epsilon)$-approx of f over (frequency vector v of) stream
  (f = # distinct elements / ℓ_2-norm / entropy / heavy hitters / …)



Streaming: The classical model

$m = n^{O(1)}$

- Stream $(x_1, \Delta_1), \ldots, (x_m, \Delta_m)$ of updates, $x_i \in [n]$, $\Delta_i \in \mathbb{R}$
- Goal: compute $(1+\epsilon)$-approx of f over (frequency vector v of) stream
  (f = # distinct elements / ℓ_2-norm / entropy / heavy hitters / …)

delete seven units of x_2

$$F_p(v) = \sum_{i=1}^{n} v_i^p$$

Randomized streaming algorithms

Problem	Space complexity
distinct elements (F_0)	$\tilde{O}(\epsilon^{-2} + \log n)$ [IW'03, KNW'10, B'18, W'23, …]
F_2-estimation	$\tilde{O}(\epsilon^{-2} \log n)$ [AMS'96,KNW'10,BDN'17,…]
F_2-heavy hitters	$\tilde{O}(\epsilon^{-2} \log^2 n)$ [BCINWW'17]
entropy estimation	$\tilde{O}(\epsilon^{-2} \log^3 n)$ [CC'13]

*Deterministic algorithms: exponentially less efficient

Randomized streaming algorithms

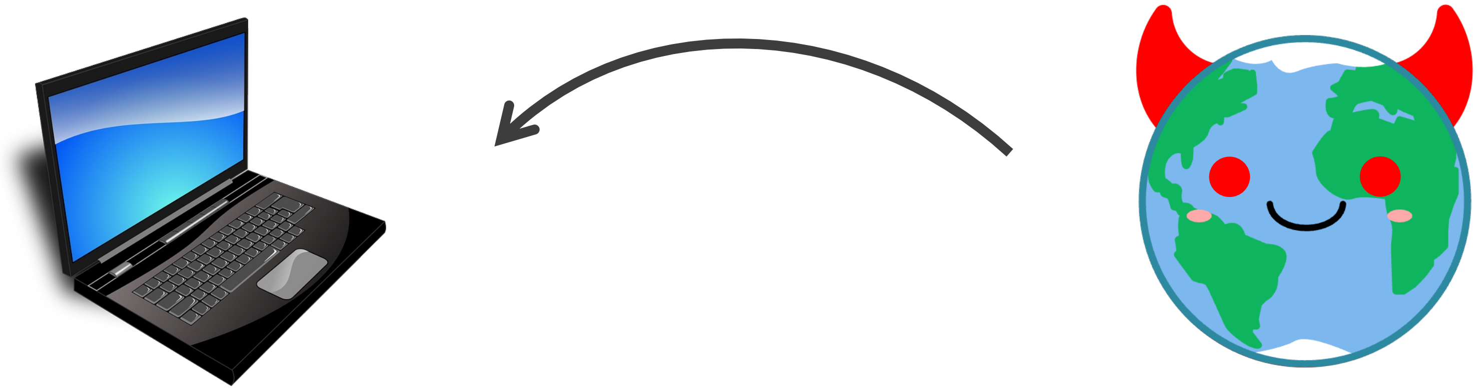
Problem	Space complexity
distinct elements (F_0)	$\tilde{O}(\epsilon^{-2} + \log n)$ [IW'03, KNW'10, B'18, W'23, …]
F_2-estimation	$\tilde{O}(\epsilon^{-2} \log n)$ [AMS'96,KNW'10,BDN'17,…]
F_2-heavy hitters	$\tilde{O}(\epsilon^{-2} \log^2 n)$ [BCINWW'17]
entropy estimation	$\tilde{O}(\epsilon^{-2} \log^3 n)$ [CC'13]

But… proofs assume stream is fixed in advance!

*Deterministic algorithms: exponentially less efficient

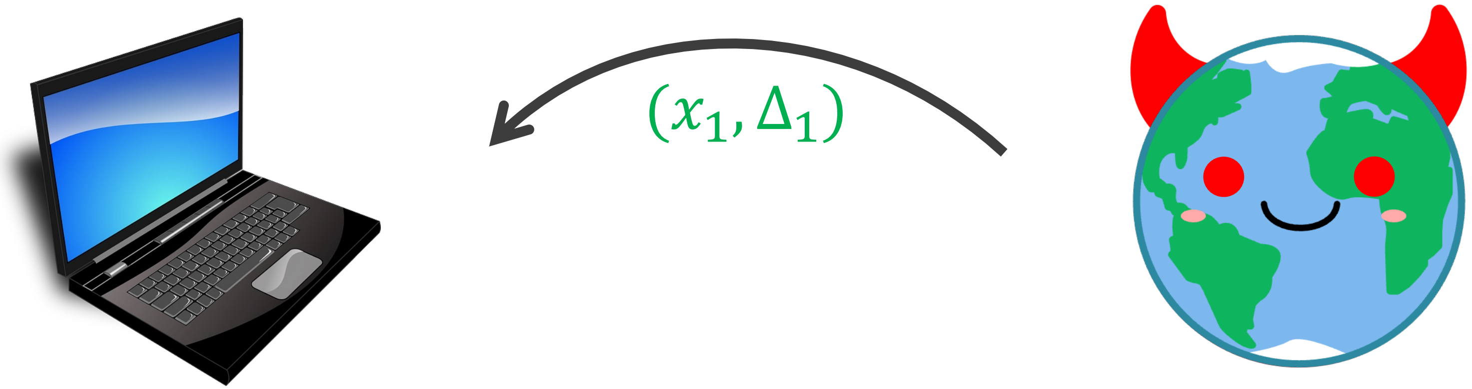
Streaming: The adaptive/adversarial model

- Feedback loop
- World's goal: break $(1+\epsilon)$-approximation



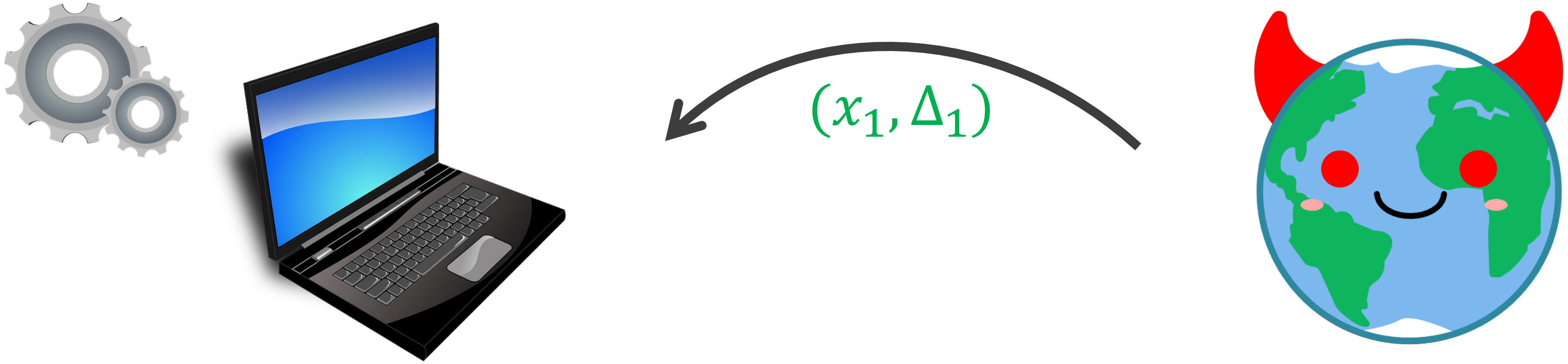
Streaming: The adaptive/adversarial model

- Feedback loop
- World's goal: break $(1 + \epsilon)$-approximation



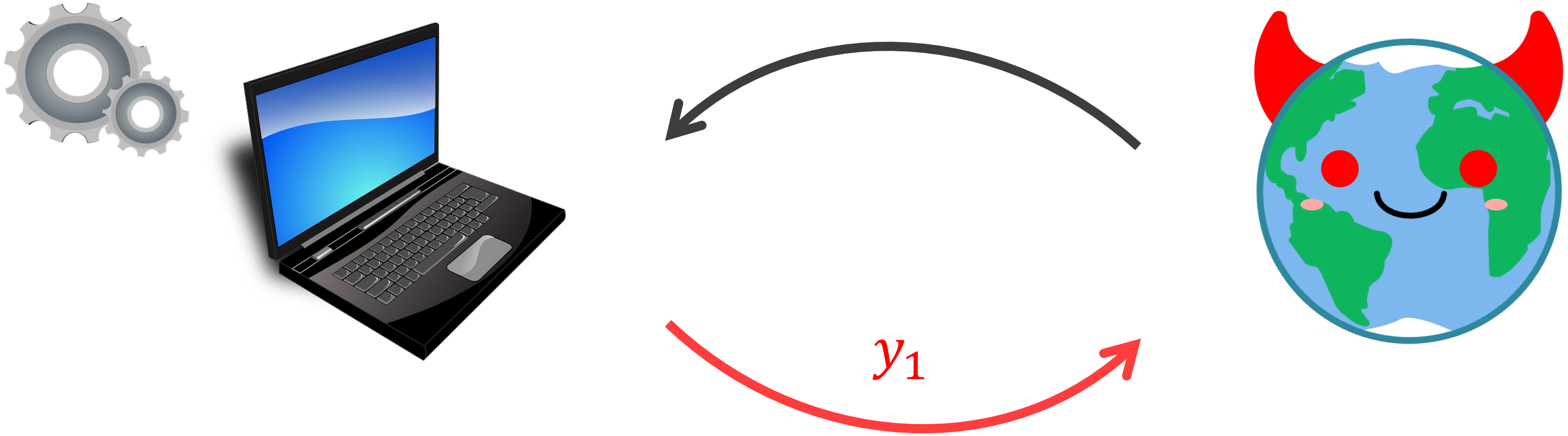
Streaming: The adaptive/adversarial model

- Feedback loop
- World's goal: break $(1 + \epsilon)$-approximation



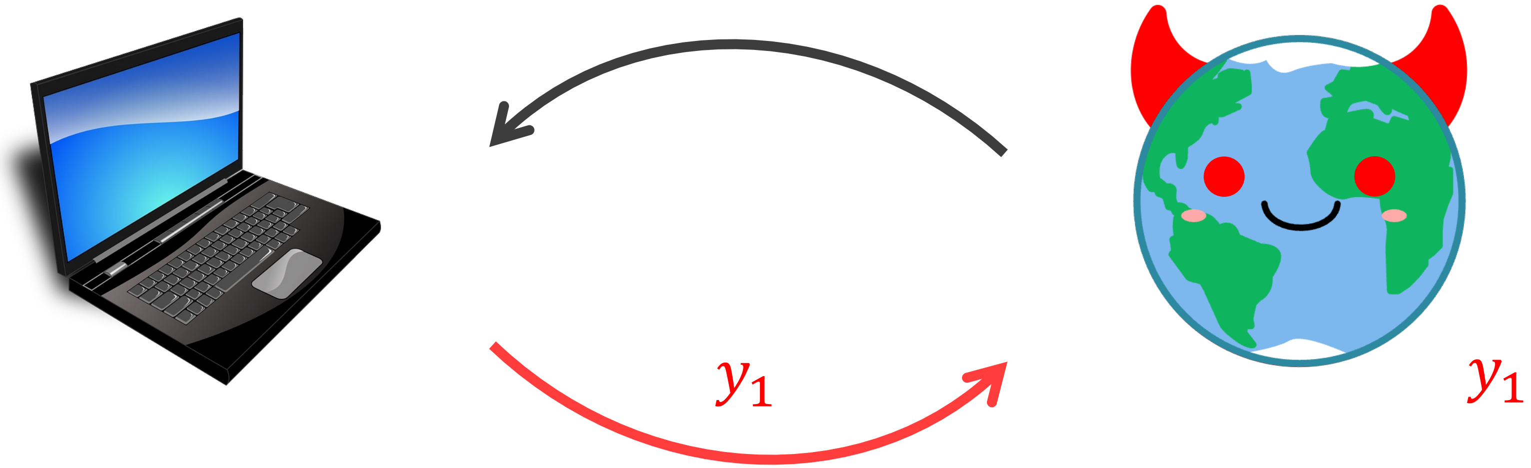
Streaming: The adaptive/adversarial model

- Feedback loop
- World's goal: break $(1+\epsilon)$-approximation



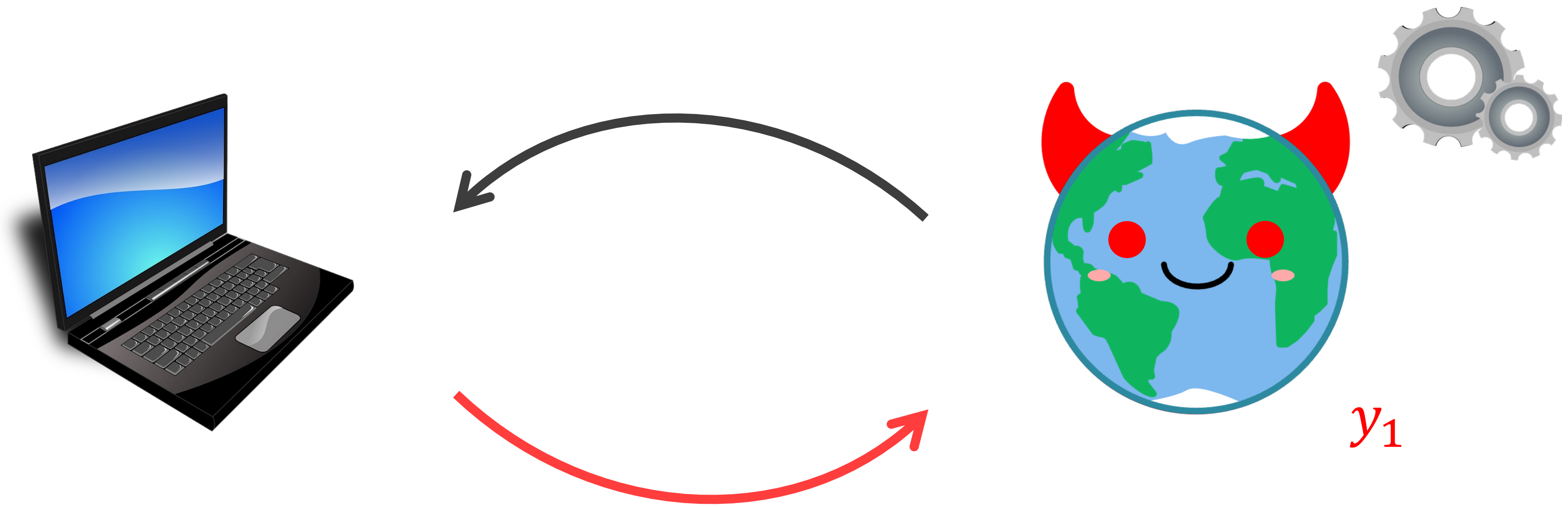
Streaming: The adaptive/adversarial model

- Feedback loop
- World's goal: break $(1+\epsilon)$-approximation



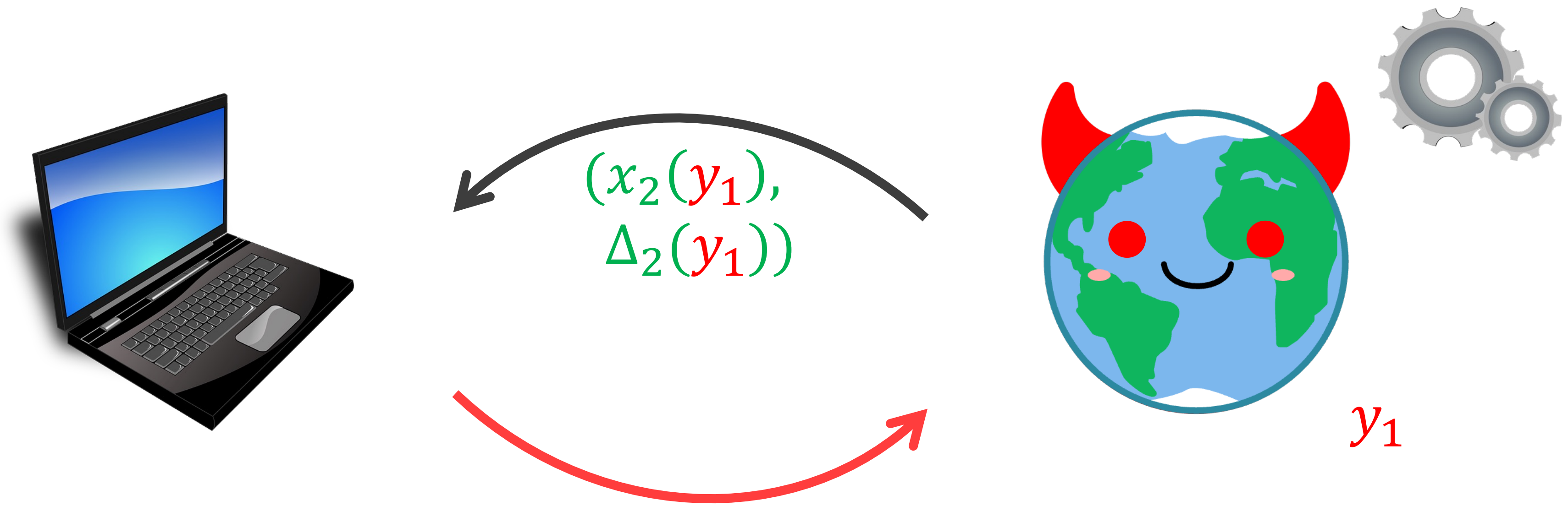
Streaming: The adaptive/adversarial model

- Feedback loop
- World's goal: break $(1+\epsilon)$-approximation



Streaming: The adaptive/adversarial model

- Feedback loop
- World's goal: break $(1+\epsilon)$-approximation



Key streaming technique: **Linear sketching**

F_2-estimation [Alon-Matias-Szegedy'96]:

Use $|Sv|_2$ to estimate $|v|_2$.

Given stream update (x_t, Δ),

update: $Sv \leftarrow Sv + \Delta \cdot S_{*,x_t}$

4-wise independent

$$\boldsymbol{S} = \begin{array}{c} \left[\begin{array}{ccccccc} \epsilon & -\epsilon & -\epsilon & \epsilon & -\epsilon & \epsilon & \epsilon \\ -\epsilon & \epsilon & -\epsilon & -\epsilon & \epsilon & -\epsilon & -\epsilon \\ \epsilon & -\epsilon & \epsilon & \epsilon & -\epsilon & -\epsilon & -\epsilon \\ \epsilon & \epsilon & -\epsilon & \epsilon & \epsilon & -\epsilon & \epsilon \end{array}\right] \\ n \end{array} \frac{1}{\epsilon^2}$$

$$\mathbb{E}_S[|Sv|_2^2] = \frac{1}{\epsilon^2}\mathbb{E}_S\left[\left(\sum_{j=1}^{n} S_{1j}v_j\right)^2\right] = \mathbb{E}_S\left[\sum_{j=1}^{n} v_j^2 + \sum_{i\neq j} S_{1i}S_{1j}v_iv_j\right] = |v|_2^2$$

Linear sketching: not robust [Hardt-Woodruff'13]

Also [AGM'12, BJWY'20, CLNSSS'22, CNSS'23,…]

Use $|Sv|_2$ to estimate $|v|_2$.

What if S_{1*} correlations learned by adversary?

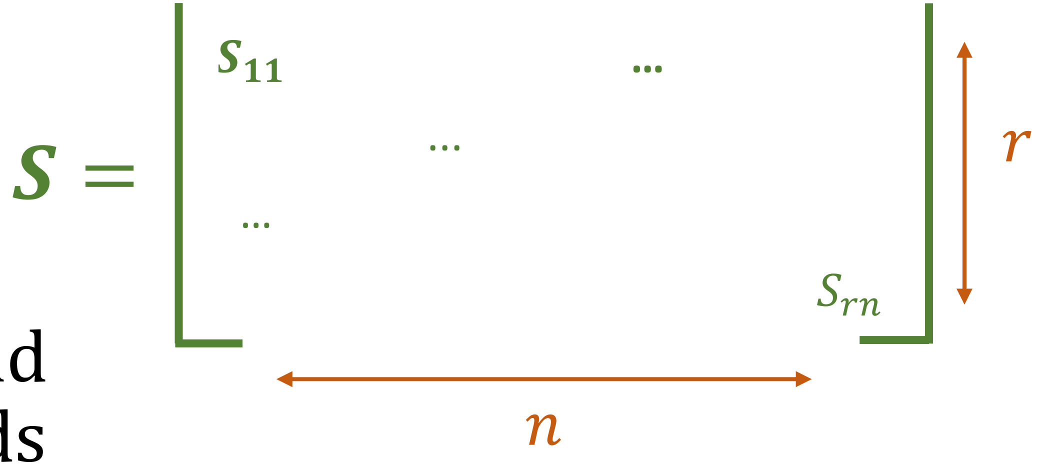
$$\boldsymbol{S} = \begin{array}{cc} \begin{bmatrix} \epsilon & -\epsilon & -\epsilon & \epsilon & -\epsilon & \epsilon & \epsilon \\ -\epsilon & \epsilon & -\epsilon & -\epsilon & \epsilon & -\epsilon & -\epsilon \\ \epsilon & -\epsilon & \epsilon & \epsilon & -\epsilon & -\epsilon & -\epsilon \\ \epsilon & \epsilon & -\epsilon & \epsilon & \epsilon & -\epsilon & \epsilon \end{bmatrix} & \updownarrow \frac{1}{\epsilon^2} \\ \xleftrightarrow{\quad n \quad} & \end{array}$$

$$\mathbb{E}_S[|Sv|_2^2] = \frac{1}{\epsilon^2}\mathbb{E}_S\left[\left(\sum_{j=1}^{n} S_{1j}v_j\right)^2\right] = \mathbb{E}_S\left[\sum_{j=1}^{n} v_j^2 + \sum_{i\neq j} S_{1i}S_{1j}v_iv_j\right] \neq |v|_2^2$$

Linear sketching: not robust [Hardt-Woodruff’13]

Also [AGM’12, BJWY’20, CLNSSS’22, CNSS’23,…]

Use $|Sv|_2$ to estimate $|v|_2$.

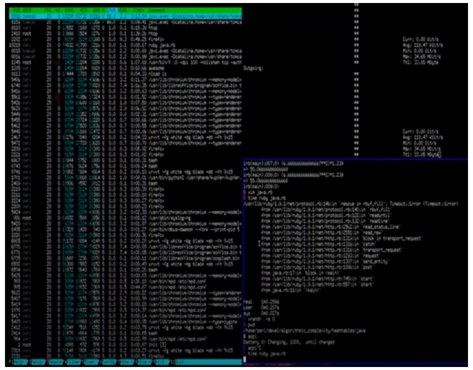


- Actual attack of [HW13] works for general $r \times n$ sketch matrices and breaks B-approx. in $poly(rB)$ rounds

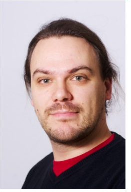
- Proof idea: find vectors correlated with row space, boost, “peel” dimension. Iterate until remainder of sketch is constant dimensional

The need for robustness - examples

Actual adversarial attacks



Alexander **“alech”** Klink
n.runs AG

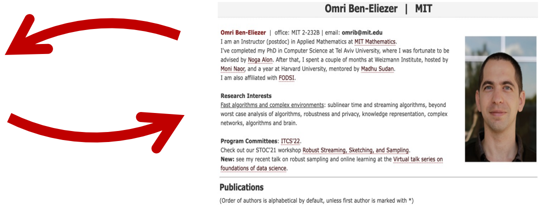


Julian **“zeri”** Wälde
TU Darmstadt

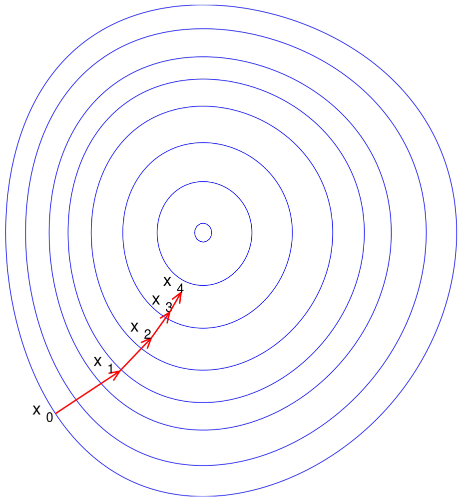
#hashDoS

Search Engine Optimization (SEO)

Google



Gradient descent



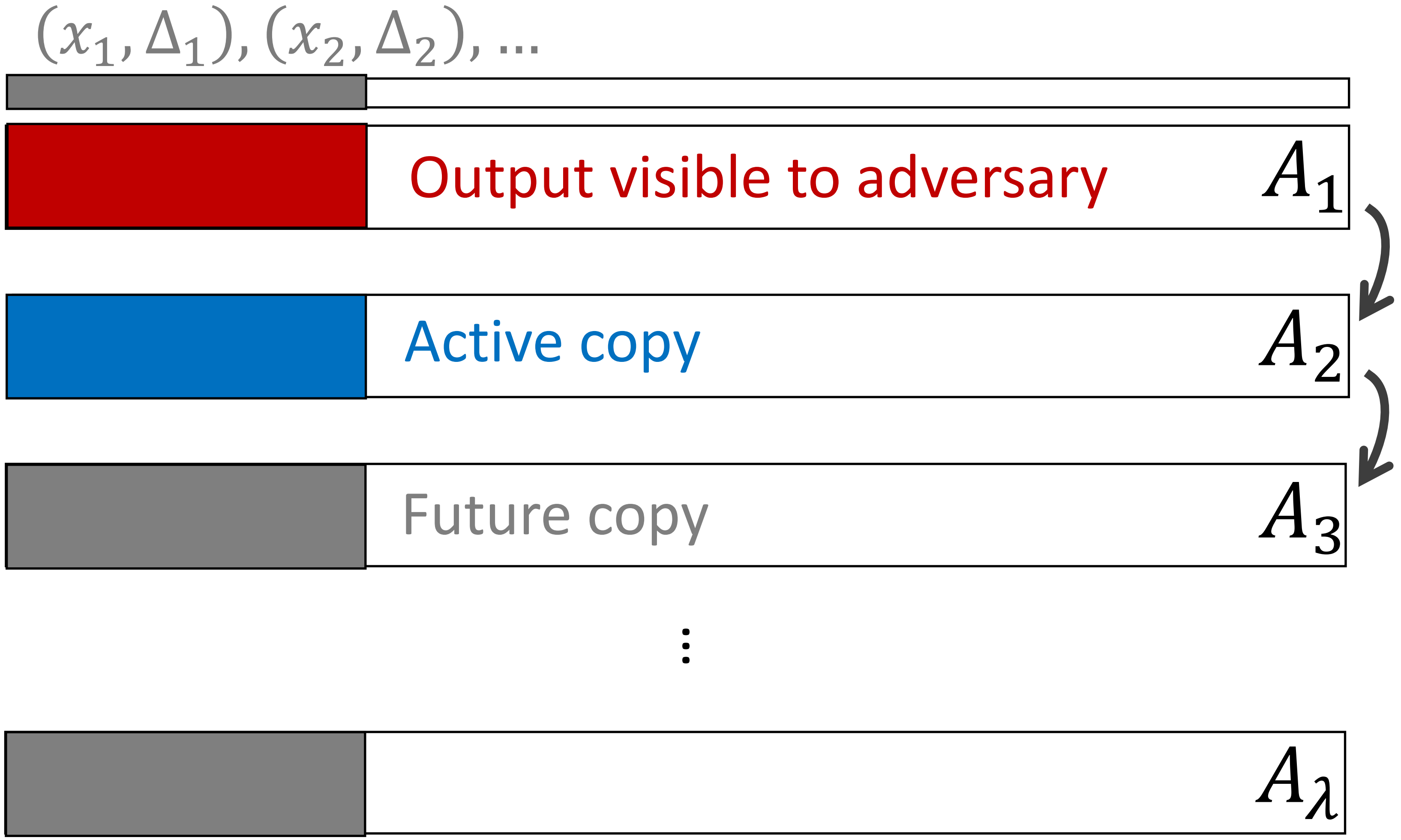
Part 2: Frameworks

How to make streaming algorithms robust?

- Sketch Switching [B., Jayaram, Woodruff, Yogev '20]
- Differential privacy [Hassidim, Kaplan, Mansour, Matias, Stemmer '20]
- Difference estimators [Woodruff, Zhou '21]
- "Best of both worlds" [Attias, Cohen, Shechner, Stemmer '23]

Sketch switching [B., Jayaram, Woodruff, Yogev '20]

- Run λ copies of algorithm
- Maintain frozen-output visible copy + active copy.
- If frozen output incorrect:
  visible copy ← active copy
  active copy ← next copy

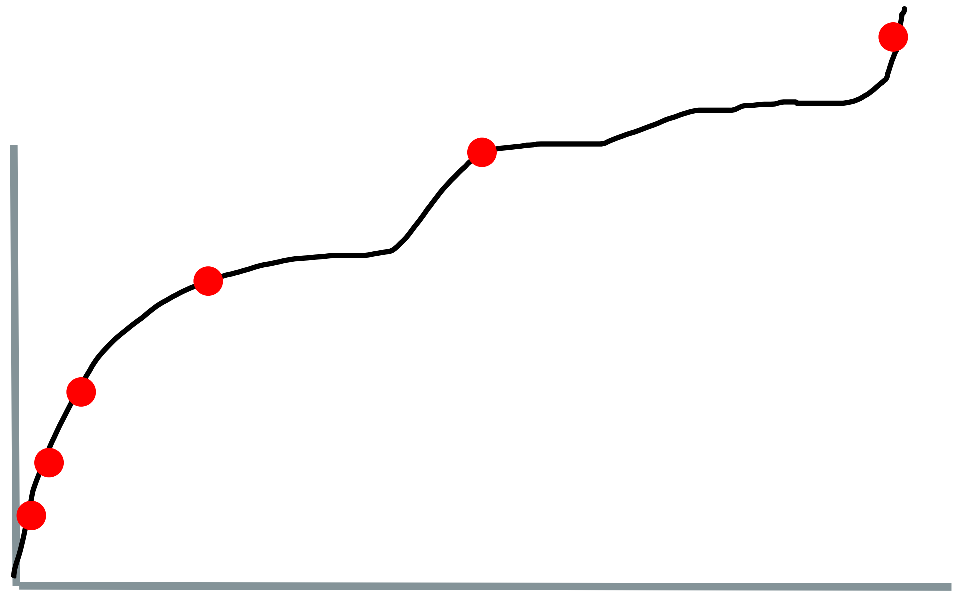


Key notion: Flip number

$$\lambda_{\epsilon,m}(f) := \max\left\{ k \;\middle|\; \begin{array}{c} \exists j_1,\ldots,j_k \in [m] \\ \forall l \in [k-1] \end{array} : f\left(v^{(j_{l+1})}\right) \notin (1 \pm \epsilon) \cdot f\left(v^{(j_l)}\right) \right\}$$

Theorem ([BJWY'20]):
Let $\mathcal{A}$ be a ("classical") ϵ-tracking streaming algorithm for f.

There exists an adversarially robust algorithm $\mathcal{A}'$ for ϵ-tracking f using space $\lambda(f) \cdot Space(\mathcal{A})$.

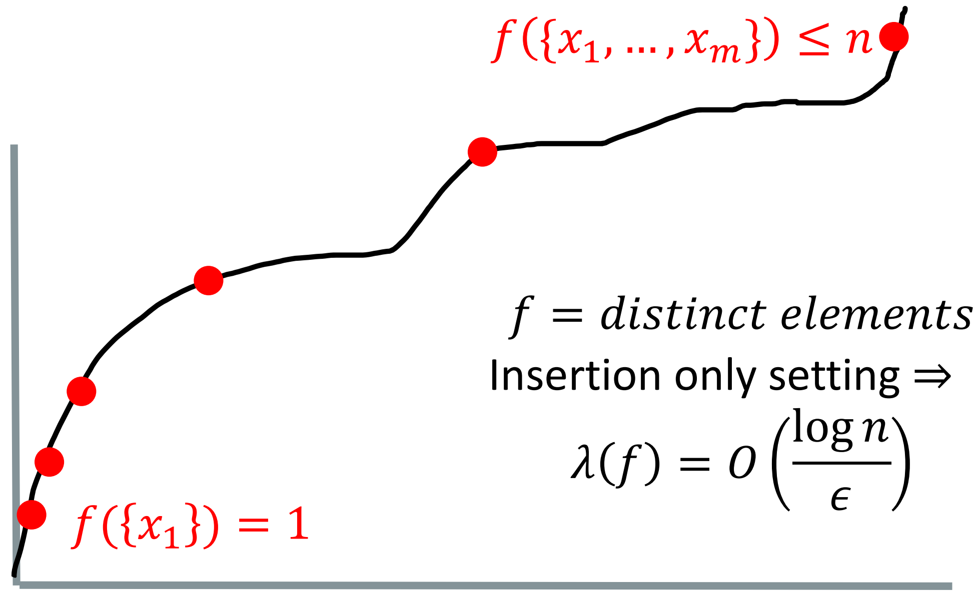


Key notion: Flip number

$$\lambda_{\epsilon,m}(f) := \max\left\{ k \;\middle|\; \begin{matrix} \exists j_1, \dots, j_k \in [m] \\ \forall l \in [k-1] \end{matrix} : f\left(v^{(j_{l+1})}\right) \notin (1 \pm \epsilon) \cdot f\left(v^{(j_l)}\right) \right\}$$

Theorem ([BJWY'20]):
Let $\mathcal{A}$ be a ("classical") ϵ-tracking streaming algorithm for f.

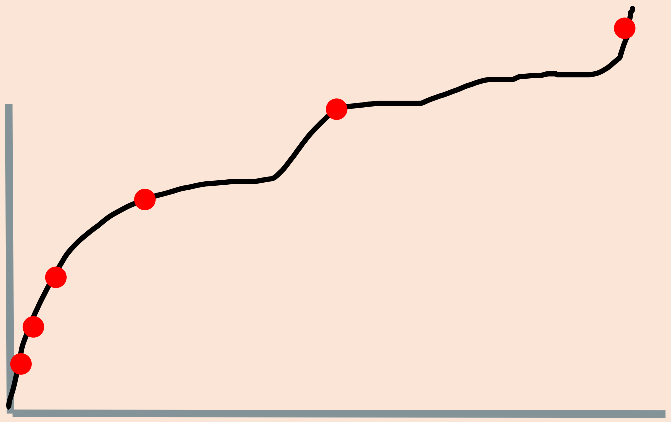
There exists an adversarially robust algorithm $\mathcal{A}'$ for ϵ-tracking f using space $\lambda(f) \cdot Space(\mathcal{A})$.



Key notion: Flip number

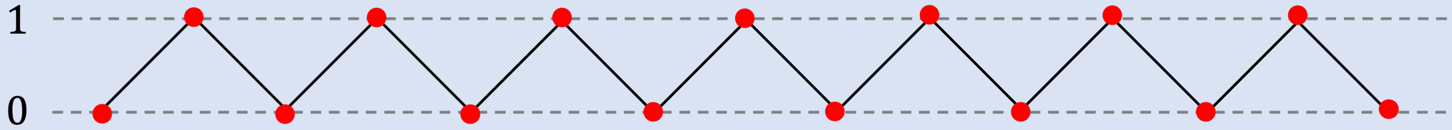
Insertion only ($\Delta > 0$):

$$\lambda = O\left(\frac{\log n}{\varepsilon}\right)$$



Turnstile model (general Δ, insertion/deletion)**:**

$\lambda = m$ in worst case!

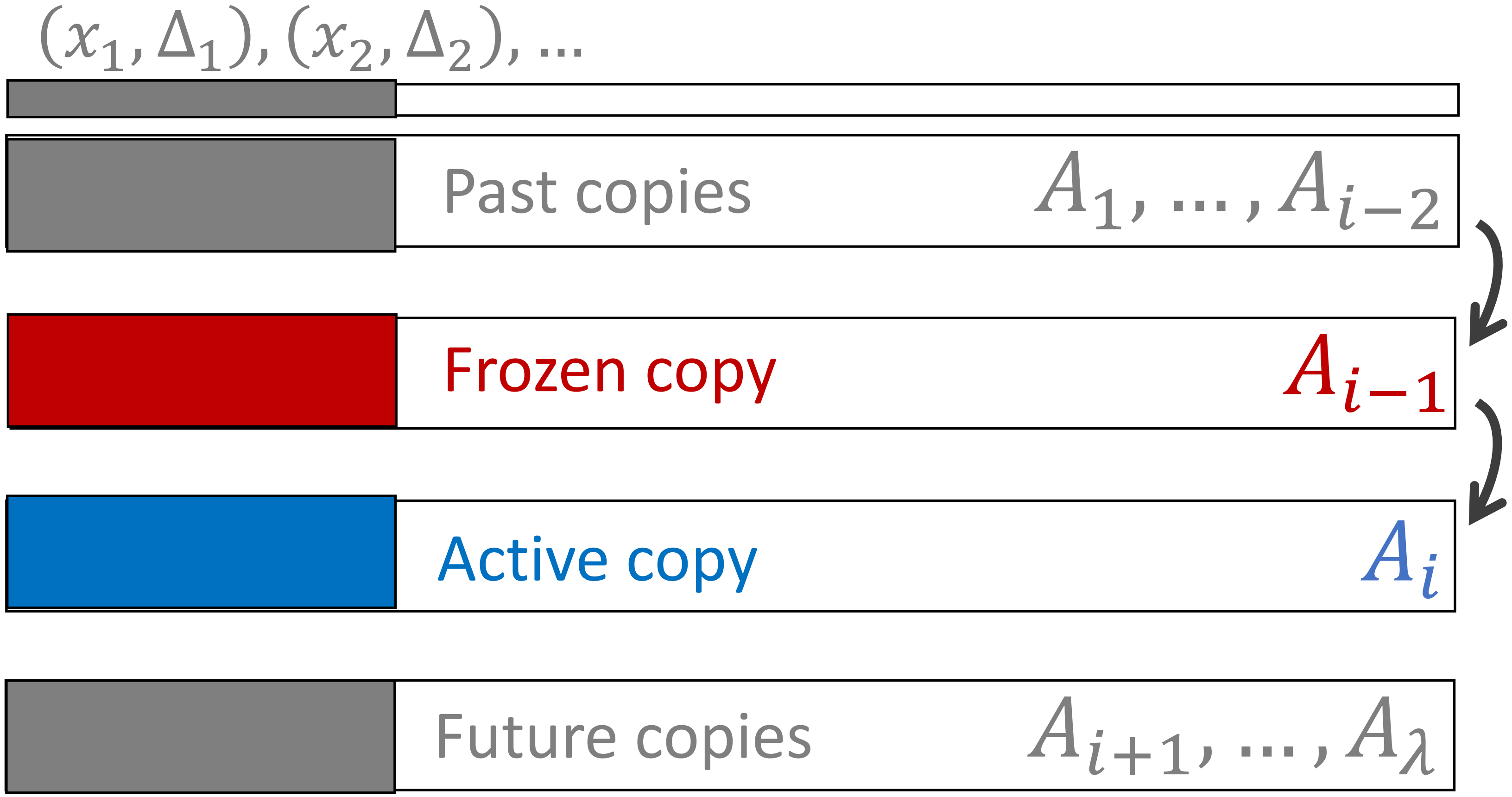


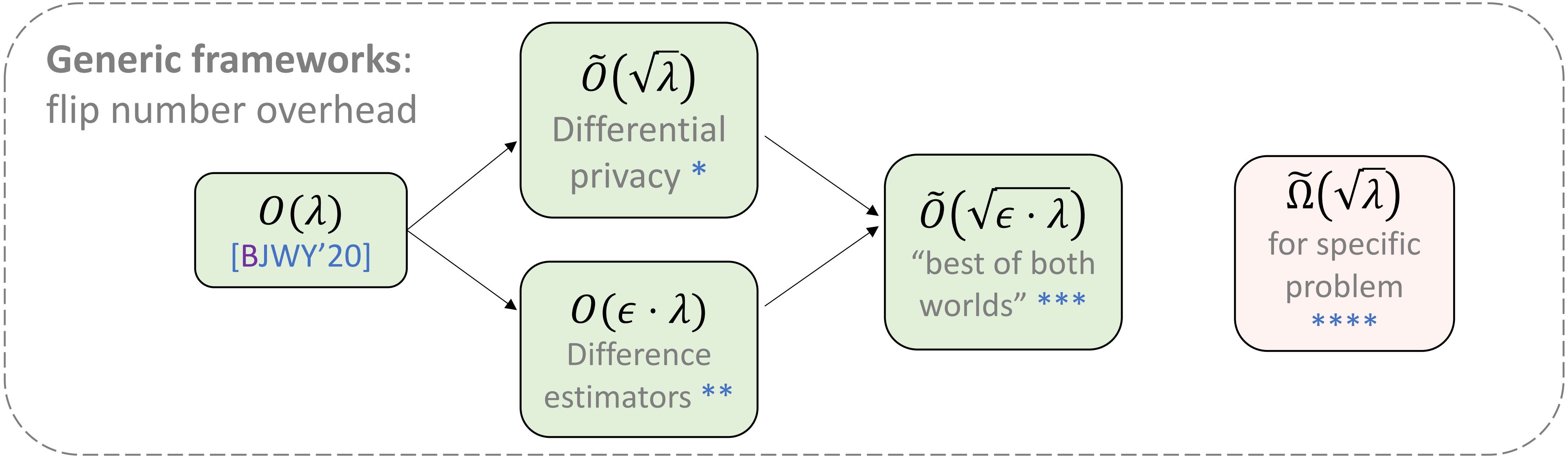
Sketch switching: Proof idea

1. Assume deterministic
(Yao's minimax [Yao'77])

2. Only need active copy A_i
to be correct over
specific (fixed) stream:

"A_i-oblivious part"
+
"frozen output part"





* Hassidim-Kaplan-Mansour-Matias-Stemmer '20

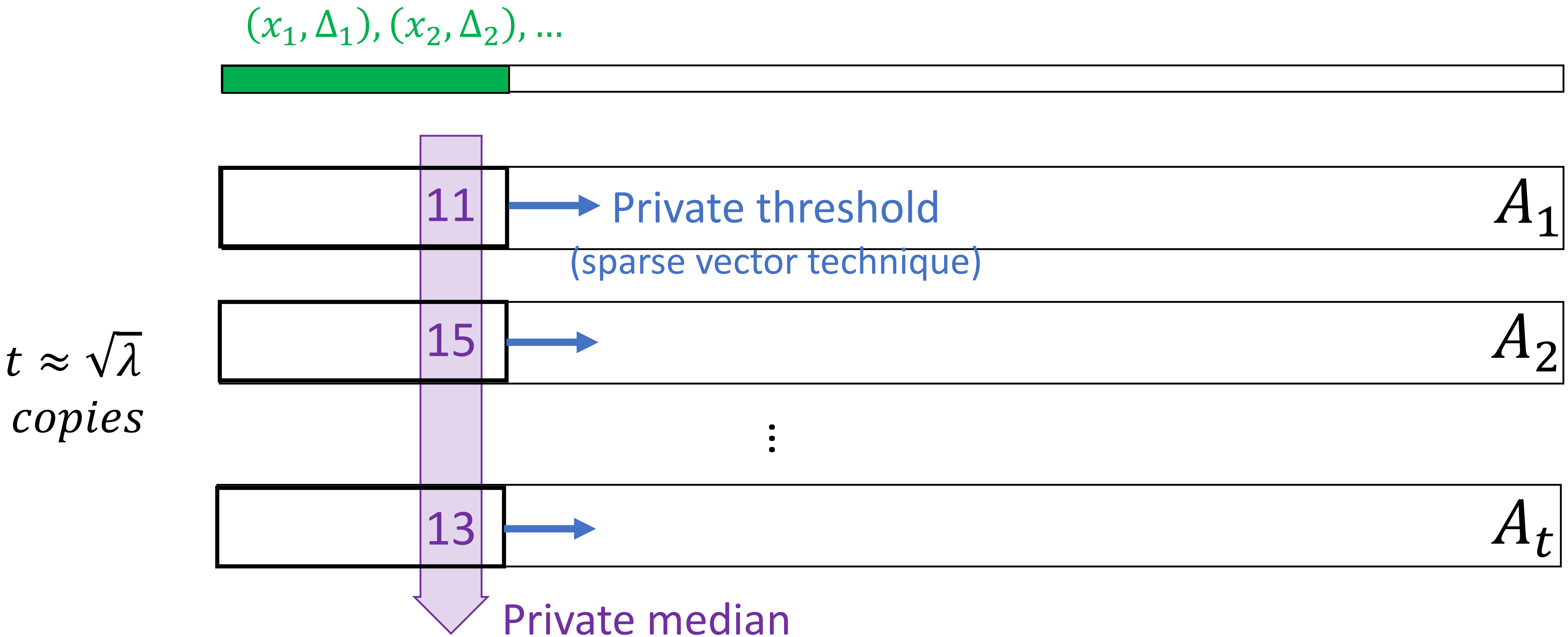
** Woodruff-Zhou '21

*** Attias-Cohen-Shechner-Stemmer '23

**** Kaplan-Mansour-Nissim-Stemmer '21

Differential privacy framework

[Hassidim-Kaplan-Mansour-Matias-Stemmer'20]



Differential privacy framework

[Hassidim-Kaplan-Mansour-Matias-Stemmer'20]

Why it works:

- Key idea -- **Advanced composition** [Dwork-Rothblum-Vadhan'10] in differential privacy supports $\approx \frac{1}{\varepsilon^2}$ adaptive interactions provided "privacy level" ε.
- Interaction with each flip $\Rightarrow \varepsilon \approx \frac{1}{\sqrt{\lambda}}$
- ε-DP median requires $\approx \frac{1}{\varepsilon} \approx \sqrt{\lambda}$ copies to be accurate

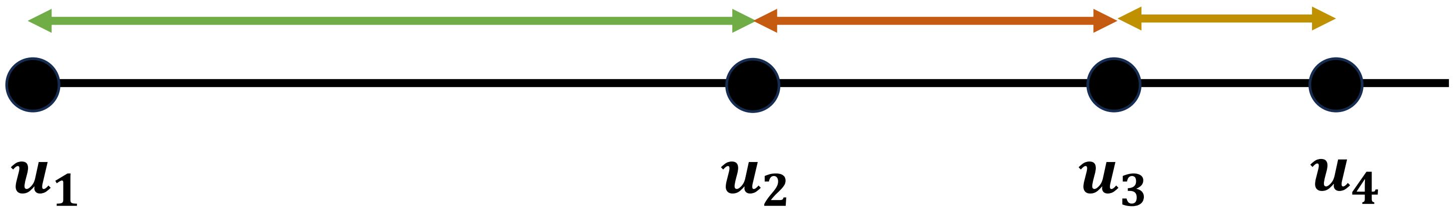
Extremely useful in applications!
(e.g., robust count sketch [Cohen, Lyu, Nelson, Sarlós, Shechner, Stemmer '22], robust dynamic graph algorithms [Beimel, Kaplan, Mansour, Nissim, Saranurak, Stemmer '22], …)

Difference estimators framework

[Woodruff, Zhou '21]

- Key notion: ε-difference estimator for vectors u and v approximates $f(u+v) - f(u)$ to $\pm\varepsilon \cdot f(u)$ error

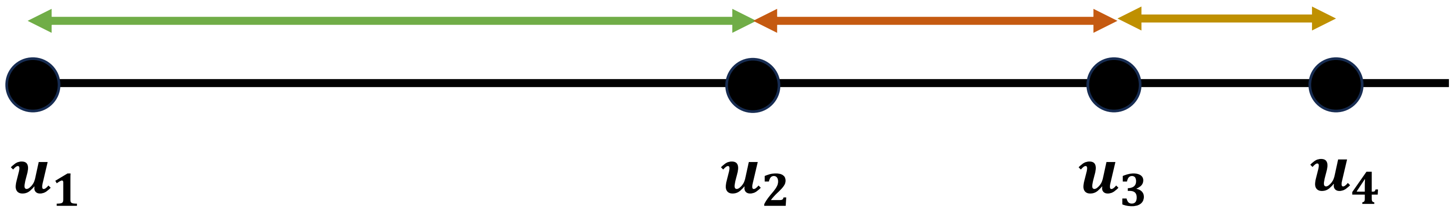
- Framework stitches difference estimators at different scales



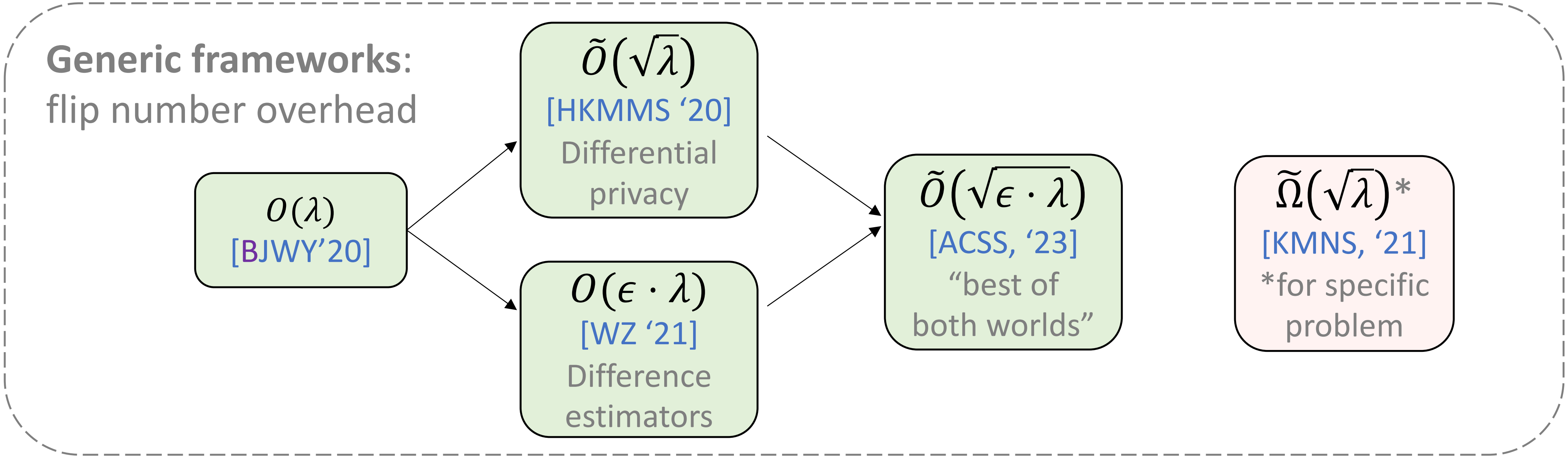
Difference estimators framework

[Woodruff, Zhou '21]

- Key notion: ε-difference estimator for vectors u and v approximates $f(u+v) - f(u)$ to $\pm\varepsilon \cdot f(u)$ error
- Framework stitches difference estimators at different scales
- Near-optimal space complexity for insertion only + sliding window model, resolving [Braverman-Ostrovsky'07]



Problem	[BJWY'20]	[HKMMS'20]	[WZ'21]
Distinct Elements (F_0) insert only	$\tilde{O}\left(\frac{1}{\epsilon^3} + \frac{\log n}{\epsilon}\right)$	$O\left(\frac{\log^{1.5} n}{\epsilon^{2.5}} + \frac{\log^{2.5} n}{\epsilon^{1.5}}\right)$	$\tilde{O}\left(\frac{1}{\epsilon^2} + \frac{\log n}{\epsilon}\right)$
F_p estimation, $p \in (0,2]$ insert only	$\tilde{O}\left(\frac{\log n}{\epsilon^3}\right)$	$O\left(\frac{\log^{1.5} n}{\epsilon^{2.5}}\right)$	$\tilde{O}\left(\frac{\log n}{\epsilon^2}\right)$
F_p with deletions, flip number λ	$\tilde{O}\left(\frac{\lambda \cdot \log^2 n}{\epsilon^2}\right)$	$O\left(\frac{\log^3 n \sqrt{\lambda}}{\epsilon^2}\right)$	$\tilde{O}\left(\frac{\lambda \cdot \log^2 n}{\epsilon}\right)$



Is flip number always the right parameter?

Generic frameworks: flip number overhead

$O(\lambda)$
[BJWY'20]

$\tilde{O}(\sqrt{\lambda})$
[HKMMS '20]
Differential privacy

$O(\epsilon \cdot \lambda)$
[WZ '21]
Difference estimators

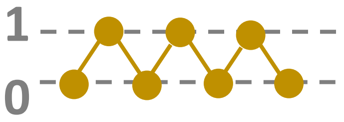
$\tilde{O}(\sqrt{\epsilon \cdot \lambda})$
[ACSS, '23]
"best of both worlds"

$\tilde{\Omega}(\sqrt{\lambda})^*$
[KMNS, '21]
*for specific problem

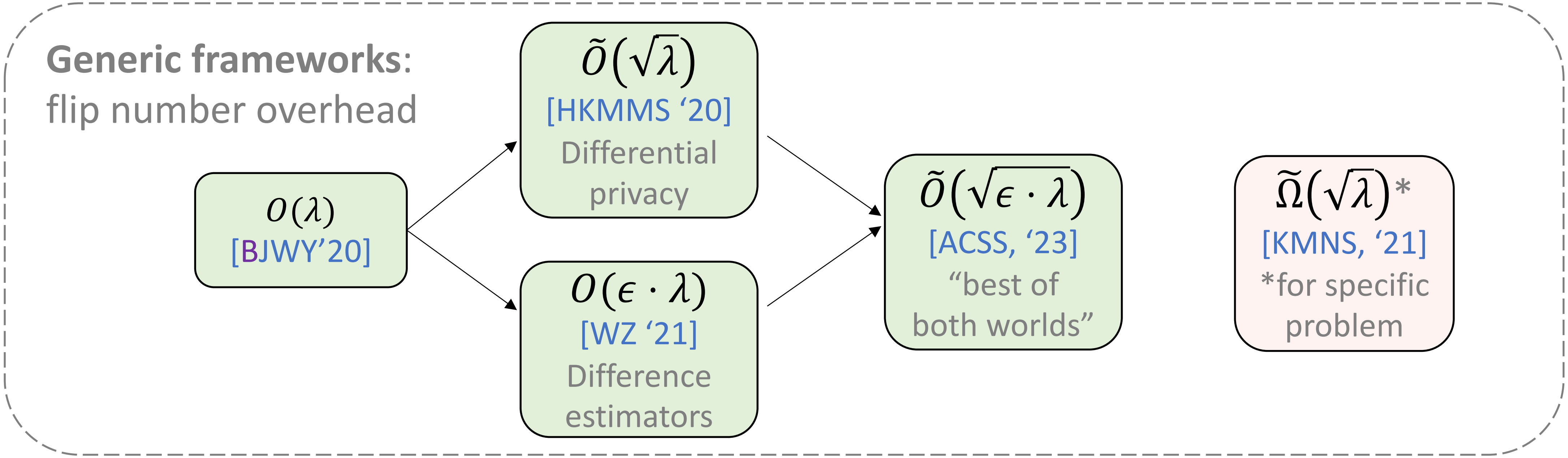
Is flip number always the right parameter?

[B., Eden, Onak '22]: **No**. for insertion-deletion streams ($\boldsymbol{\lambda = m}$) we get

$\tilde{O}(m^{1/3})$ for distinct elements, $\tilde{O}(m^{2/5})$ for ℓ_2-estimation



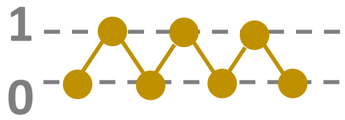
Technique: differential privacy + sparse recovery



Is flip number always the right parameter?

[B., Eden, Onak '22]: **No**. for insertion-deletion streams ($\lambda = m$) we get

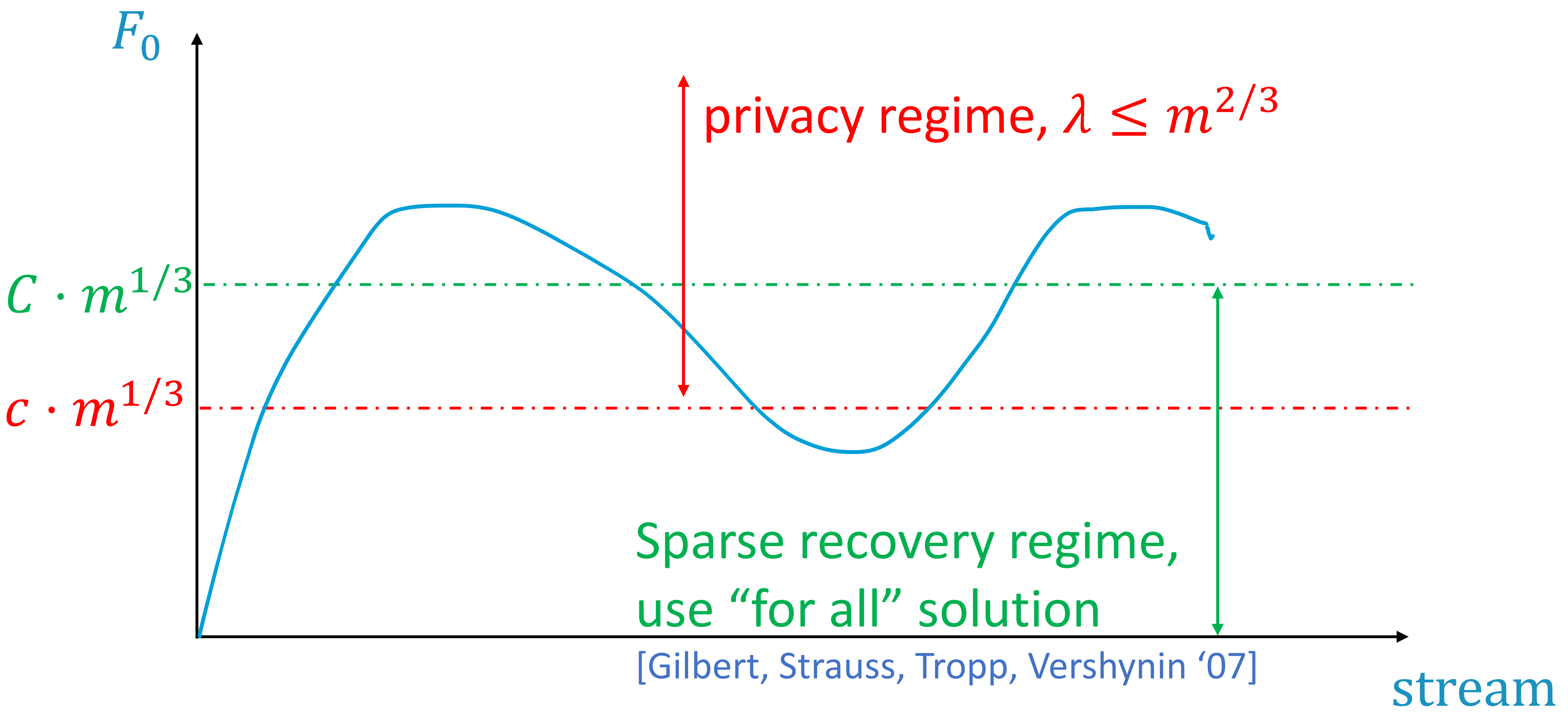
$\tilde{O}(m^{1/3})$ for distinct elements, $\tilde{O}(m^{2/5})$ for ℓ_2-estimation



Technique: differential privacy + sparse recovery

Best known, but exponentially worse than static.

Dense-sparse tradeoff [B., Eden, Onak '22]



Part 3: Lessons

Lesson 1: Sampling over sketching

- **Linear sketching**: extremely useful in static settings, but breaks in $r^{O(1)}$ rounds in adversarial settings [HW'13, BJWY'20, CLNSSS'22, CNSS'23,...]

$$\boldsymbol{S} = \begin{array}{c} \begin{bmatrix} s_{11} & \cdots & s_{1n} \\ \cdots & & \\ s_{r1} & \cdots & s_{rn} \end{bmatrix} \\ n \end{array} \begin{array}{c} r \\ \\ \end{array}$$

- **Random sampling**: another fundamental technique, more reliable in adversarial settings?

Lesson 1: Sampling over sketching

How many uniform samples needed so that ***sample*** will be **representative** of ***data***?

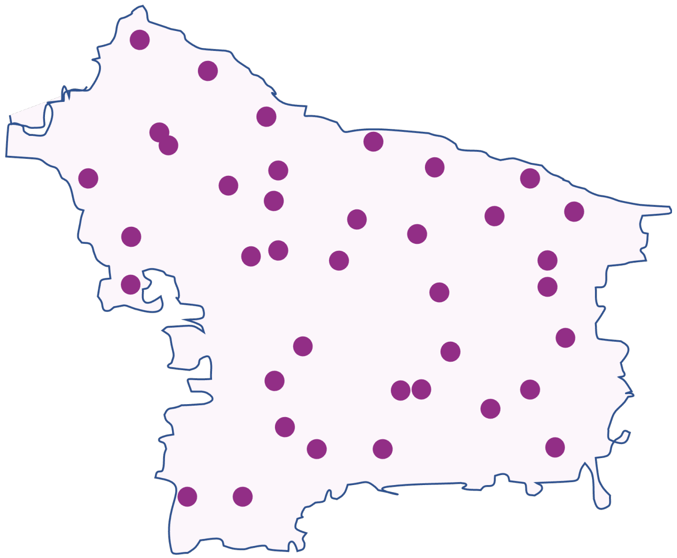
Oblivious case: $\Theta(\frac{\text{VC-DIM}}{\epsilon^2})$

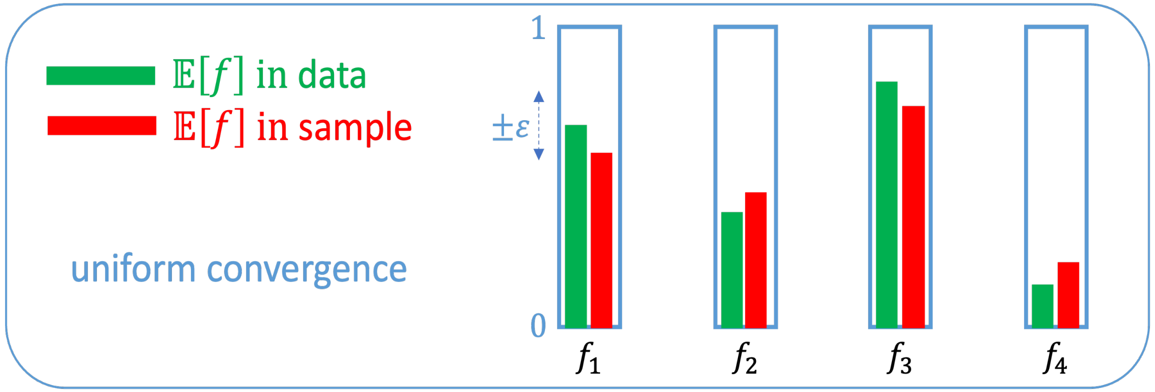
[B., Yogev '19] Not enough for robust case!

But $VC \cdot \frac{\log m}{\epsilon^2}$ suffices.

[Alon, B., Dagan, Moran, Naor, Yogev '21]:

$$\Theta\left(\frac{\text{Littlestone-DIM}}{\varepsilon^2}\right)$$





Lesson 1: Sampling over sketching

[Braverman, Hassidim, Matias, Schain, Silwal, Zhou ‘21]:

- Importance sampling & merge and reduce robust “for free”

⇒ Robustness of many existing algorithms!

Meta-approach	Applications
Merge and reduce (Theorem 1.1)	Coreset construction, support vector machine, Gaussian mixture models, k-means clustering, k-median clustering, projective clustering, principal component analysis, M-estimators, Bayesian logistic regression, generative adversarial networks (GANs), k-line center, j-subspace approximation, Bregman clustering
Row sampling (Theorem 1.2)	Linear regression, generalized regression, spectral approximation, low-rank approximation, projection-cost preservation, L_1-subspace embedding
Edge sampling (Theorem 1.3)	Graph sparsification

(Taken from [BHMSSZ’21])

Lesson 1: Sampling over sketching

Formal separation between ”sketching type” vs. “sampling type” algos?

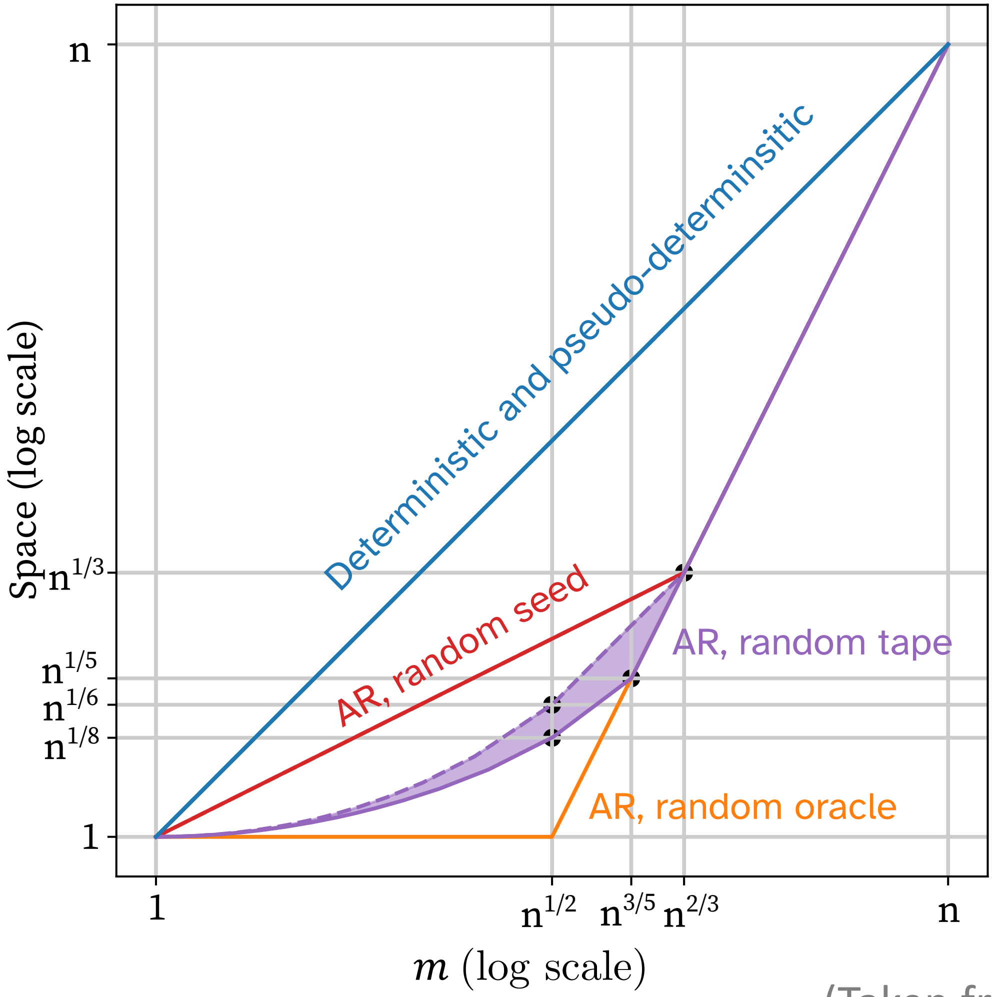
Yes! For ”missing item finding” [Stoeckl’23, Chakrabarti-Stoeckl’23, Magen’24] (see also Menuhin-Naor’22)

Random oracle = fixed random string, no storage cost

Random tape = randomness on the fly, costs to store it

Random seed = randomness at beginning

Pseudo-deterministic [GGMW’19] = “deterministic with prob. $1-\delta$”



(Taken from [CS’23])

Speaking about separations: robust graph coloring

Static streams: $(\Delta + 1)$-coloring in $\tilde{O}(n)$ space [Assadi, Chen, Khanna'19]

Adversarially robust [Chakrabarti, Ghosh, Stoeckl'22], [Assadi, Chakrabarti, Ghosh, Stoeckl'23]

"No free lunch" lower bound ☹:

$O(\Delta)$ colors requires $\Omega(n\Delta)$ space, $O(n)$ space implies $\Omega(\Delta^2)$ colors

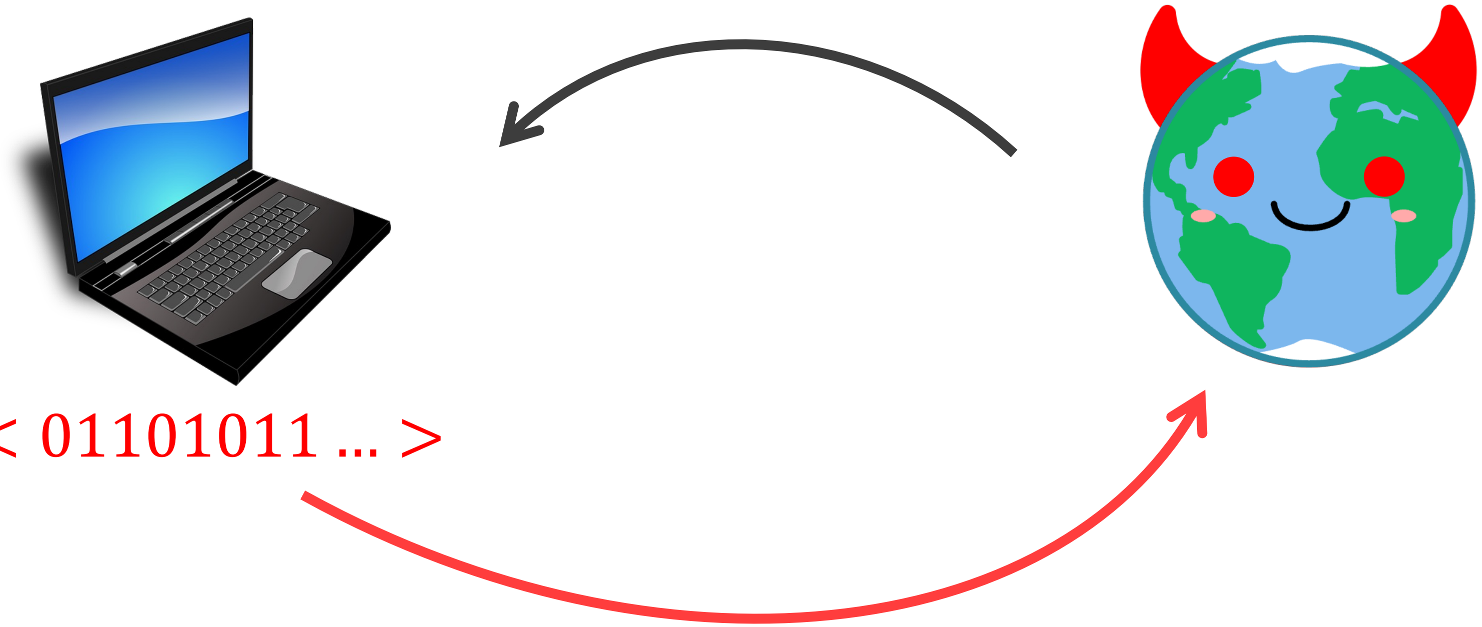
Upper bounds via sketch switching + additional techniques:

$O(\Delta^{2.5})$ colors in $\tilde{O}(n)$ space, or $O(\Delta^2)$ colors in $\tilde{O}(n\Delta^{1/3})$ space

Deterministic [Assadi, Chen, Sun '22]: $\tilde{O}(n)$ space implies $\exp(\Delta^{\Omega(1)})$ colors!

Lesson 2: White box vs black box

White box: adversary can see internal state of algorithm
[Ajtai, Braverman, Jayram, Silwal, Sun, Woodruff, Zhou ‘22], [Feng, Woodruff ‘23]



Lesson 2: White box vs black box

White box: adversary can see internal state of algorithm
[Ajtai, Braverman, Jayram, Silwal, Sun, Woodruff, Zhou ‘22], [Feng, Woodruff ‘23]

- Algorithms (e.g., n^{ε}-approx. F_0) from cryptographic primitives (e.g., SIS) and computationally-bounded adversary assumptions
- $\Omega(n)$ lower bound for approximate white-box F_p-estimation via reductions from deterministic GAP-EQUALITY problem in communication complexity

Lesson 3: Connections and applications [partial list]

- **Techniques/notions**:
  "new": differential privacy [HKMMS'20], cryptography [ABJSSW'21], statistics / learning theory [ABDMNY'21], adaptive data analysis [KMNS'21]
  "standard": sparse recovery [BEO'22], communication complexity [CGS'22]
  + novel techniques like difference estimators [WZ'21]

- **Applications/implications** in:
  dynamic data structures [BKMNSS'22], graph coloring [CGS'22, ACGS'23], sliding window streaming [WZ'21], machine learning [BHMSSZ'21, WZZ'23, CSWZZZ'23], analytics [RZCP'24], …

- **Many connections waiting to be explored!**
  A few examples: algorithmic game theory, cognitive science, reinforcement learning

Lesson 4 (personal): randomness of history

Wenn ich nur erst die Sätze habe! Die Beweise werde ich schon finden.

-- Bernhard Riemann (1826-1866)

A few (important/favorite) open questions

- $m^{o(1)}$ space for robust streaming F_0 or F_2?
- Alternatively, $m^{\Omega(1)}$ lower bound for same problems?
  (Not known even for pseudo-deterministic algorithms!)

- Interesting beyond worst case adversarial models [Cherapanamjeri, Silwal, Woodruff, Zhang, Zhang, Zhou'23, Sedigurchi, Stemmer, Shechner'23]
  - streaming vs. streaming?
  - Distribution-maintaining adversaries?
  - Models for robust dynamic graph algorithms?
- "Practical deployment"