Nearly Optimal Distinct Elements and Heavy Hitters on Sliding Windows

Vladimir Braverman

Elena Grigorescu

Harry Lang

David P. Woodruff

Samson Zhou

Johns Hopkins University

Carnegie Mellon University

Purdue University

Streaming / Sliding Window Model

- Input: Elements of an underlying data set S, which arrive sequentially
- Output: Evaluation (or approximation) of a given function
- Goal: Use space *sublinear* in the size of the input S

- Sliding Window: "Only the m most recent updates form the underlying data set S"
  - Recent interactions, time sensitive

Heavy-Hitters

- Given a set S of m elements from $[n]$, let f_i be the frequency of element i. (How often it appears)
- Let L_2 be the norm of the frequency vector:

$$L_2 = \sqrt{f_1^2 + f_2^2 + \cdots + f_n^2}$$

- Goal: Given a set S of m elements from $[n]$ and a parameter ϵ, output the elements i such that $f_i > \epsilon\, L_2$...and no elements j such that $f_j < \frac{\epsilon}{16}\, L_2$.

Heavy-Hitters in the Insertion-Only Model

- Insertion-Only: "Elements of the data stream are *permanent*"
- CountSketch [Charikar, Chen, Farach-Colton 04]:
  - Algorithm for finding L_2 heavy hitters using $O\left(\frac{1}{\epsilon^2}\log^2 n\right)$ space.
  - $O\left(\frac{1}{\epsilon^2}\right)$ by $O(\log n)$ table of counters
  - Each element hashes to a bucket in each row and adds $\{-1,+1\}$.
- BPTree [BravermanChestnutIvkinNelsonWangWoodruff17]:
  - Algorithm for finding L_2 heavy hitters using $O\left(\frac{1}{\epsilon^2}\log n\log\frac{1}{\epsilon}\right)$ space.

Heavy-Hitters in the Sliding Window Model

Upper Bound	Lower Bound
$O\left(\frac{1}{\epsilon^4}\log^3 n\right)$ [BGO14]	$\Omega\left(\frac{1}{\epsilon^2}\log n\right)$ [JST11]
$O\left(\frac{1}{\epsilon^2}\log^2 n\left(\log^2\log n+\log\frac{1}{\epsilon}\right)\right)$ [Here]	$\Omega\left(\frac{1}{\epsilon^2}\log^2 n\right)$ [Here]

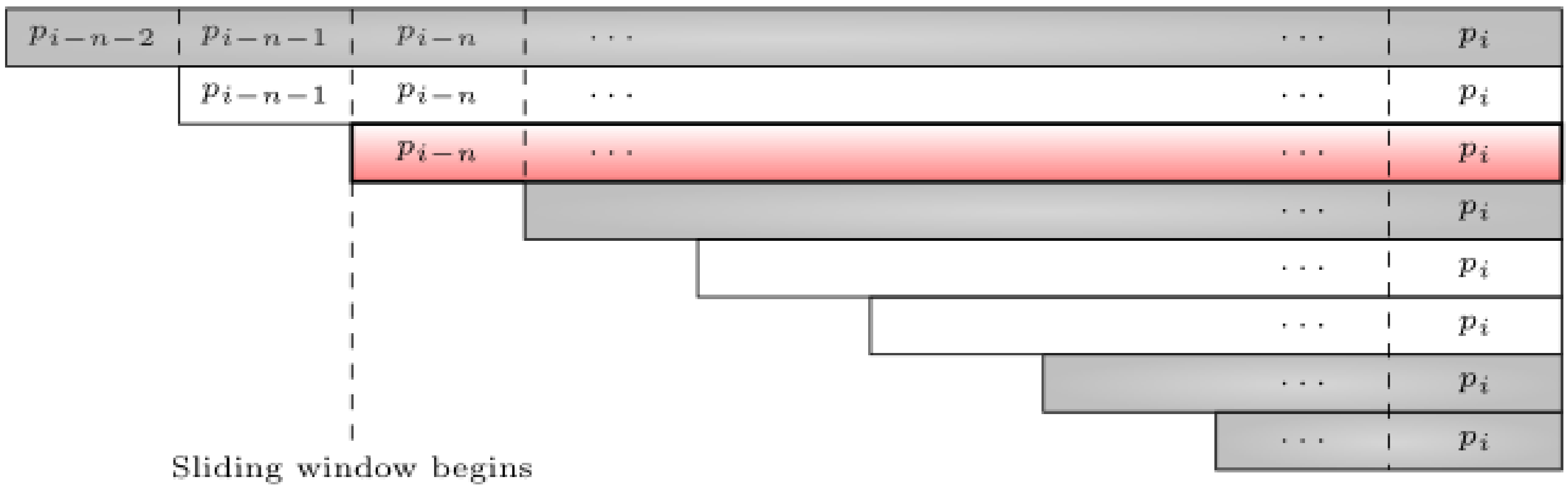
Optimal up to $\log\log n, \log\frac{1}{\epsilon}$ factors

The ϵ Matters!

- CountSketch [Charikar, Chen, Farach-Colton 04]:
  - Algorithm for finding L_2 heavy hitters using $O\left(\frac{1}{\epsilon^2}\log^2 n\right)$ space.
- Count-Min Sketch [Cormode, Muthukrishnan 05]:
  - Algorithm for finding L_1 heavy hitters using $O\left(\frac{1}{\epsilon}\log^2 n\right)$ space.
  - Outputs are always *biased.*

Technical Ingredient: Smooth Histogram

- Data structure for converting insertion-only streaming algorithms to sliding window algorithms for *smooth functions* [Braverman, Ostrovsky 07]



Smooth Histogram

- Maintain several instances of the algorithm, starting at times $m_1, m_2, \ldots, m_t$: $A_1, A_2, \ldots, A_t$
- Each instance is "roughly $(1+\epsilon)$ apart":

$$f(A_t) \leq (1+\epsilon) f(A_{t-1}) \leq (1+\epsilon)^2 f(A_{t-2}) \leq \cdots$$

- Smooth functions: If A_1 and A_2 are "close", they will remain close.
- If f is bounded by a polynomial of n, then $O\left(\frac{1}{\epsilon^2}\log n\right)$ instances of the algorithm are needed

Review

- Need $O\left(\frac{1}{\epsilon^2}\log n\right)$ instances of the algorithm
- Each instance uses $O\left(\frac{1}{\epsilon^2}\log n \log\frac{1}{\epsilon}\right)$ space $\rightarrow$ $O\left(\frac{1}{\epsilon^4}\log^2 n \log\frac{1}{\epsilon}\right)$ space needed
  - First attempt already improves upon $O\left(\frac{1}{\epsilon^4}\log^3 n\right)$ [BGO14]!
- Need a few ingredients to shave off $O\left(\frac{1}{\epsilon^2}\right)$ factor.
- Idea: use $O(\log n)$ instances of the algorithm?

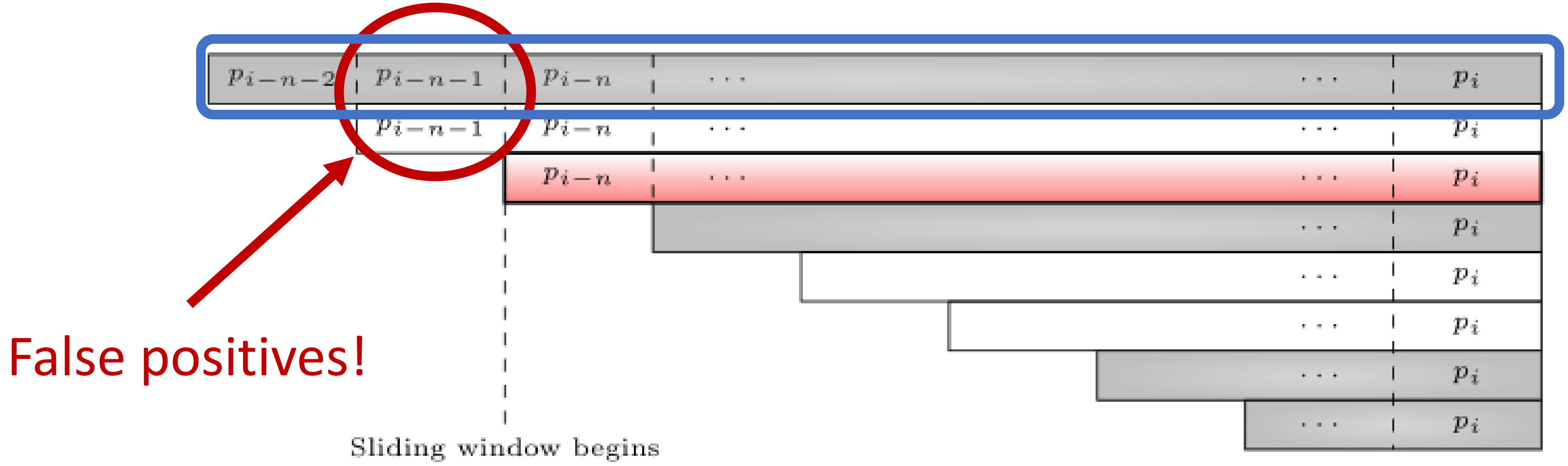
Heavy-Hitters in the Sliding Window Model

- StrongL2Estimator [Blasiok, Ding, Nelson 17]:
  - Algorithm for approximating L_2 within 2 *at all times* using $O(\log n \log\log n)$ space.
- Algorithm (second attempt):
  1. Maintain a 2-approximation of L_2 (several instances of StrongL2Estimator)
  2. Return the heavy-hitters for each instance (BPTree)

Heavy-Hitters in the Sliding Window Model

❖ Algorithm (second attempt):

1. Maintain a 2-approximation of L_2 (several instances of StrongL2Estimator)
2. Return the heavy-hitters for each instance (BPTree)



Heavy-Hitters in the Sliding Window Model

- SmoothCounter:
  - Algorithm for approximating f_i for a given item i within 2 using $O(\log n)$ space.
- Algorithm:
  1. Maintain a 2-approximation of L_2 (several instances of StrongL2Estimator)
  2. Report the heavy-hitters early for first instance (BPTree)
  3. Maintain a counter for each reported item i (SmoothCounter)
  4. Confirm whether $f_i > \frac{\epsilon}{16} L_2$.
- Space Complexity: $O(\log n)$ instances of StrongL2Estimator, BPTree, $O\left(\frac{1}{\epsilon^2} \log n\right)$ instances of SmoothCounter

Heavy-Hitters in the Sliding Window Model

- SmoothCounter:
  - Algorithm for approximating f_i for a given item i within 2 using $O(\log n)$ space.
- Algorithm:

Our result: $O\left(\frac{1}{\epsilon^2}\log^2 n\left(\log^2\log n+\log\frac{1}{\epsilon}\right)\right)$ bits of space

- Space Complexity: $O(\log n)$ instances of StrongL2Estimator, BPTree, $O\left(\frac{1}{\epsilon^2}\log n\right)$ instances of SmoothCounter

Technical Caveats

- Maintenance of smooth histogram uses correctness of all instances
  - Need union bound over $\Omega(n)$ instances → additional $\log n$ factor space
  - We show that it suffices to union bound over $O(\text{polylog}\, n)$ instances
    - First use Khintchine's inequality to show that *all* instances maintain a $\log n$ -approximation of L_2.
    - Then use properties of L_2 to show that any instance whose output is "too low" or "too high" cannot impact the output of the smooth histogram.
- Lower Bound
  - Augmented Index: Identities of each heavy hitter in the suffix gives information about the value of $S[i]$.

Questions?

Distinct Elements (L_0 Norm)

- Given a set S of m elements from $[n]$, let F be the number of distinct elements in S. (How many elements of $[n]$ appear *at least once* in S)
- Goal: Give $(1+\epsilon)$-approximation of F.
- Best-known algorithm: $O\left(\frac{1}{\epsilon^3}\log^2 n + \frac{1}{\epsilon}\log^3 n\right)$ bits of space [Kane, Nelson, Woodruff 10, Braverman, Ostrovsky 07]

Our result: $O\left(\frac{1}{\epsilon^2}\log n\left(\log\log n \log\frac{1}{\epsilon}\right) + \frac{1}{\epsilon}\log^2 n\right)$ bits of space

Distinct Elements

Upper Bound	Lower Bound
$O\left(\frac{1}{\epsilon^3}\log^2 n + \frac{1}{\epsilon}\log^3 n\right)$ [KNW10, BO07]	$\Omega\left(\frac{1}{\epsilon^2} + \log n\right)$ [AMS99, IW03]
$O\left(\frac{1}{\epsilon^2}\log n\left(\log\log n \log\frac{1}{\epsilon}\right) + \frac{1}{\epsilon}\log^2 n\right)$ [Here]	$\Omega\left(\frac{1}{\epsilon^2}\log n + \frac{1}{\epsilon}\log^2 n\right)$ [Here]

Optimal up to $\log\log n, \log\frac{1}{\epsilon}$ factors

Distinct Elements

n

Distinct Elements

n

S: 3

Distinct Elements

		1												

n

S: 3

Distinct Elements

		1												

n

S: 3, 5

Distinct Elements

		1		1										

n

S: 3, 5

Distinct Elements

		1		1										

n

S: 3, 5, 9

Distinct Elements

		1		1				1						

n

S: 3, 5, 9

Distinct Elements

		1		1				1						

n

S: 3, 5, 9, 3

Distinct Elements

		1	1	1				1						

n

S: 3, 5, 9, 3, 4

Distinct Elements

	1	1	1	1				1			1			

$\rightarrow 6$

n

S: 3, 5, 9, 3, 4, 2, 12

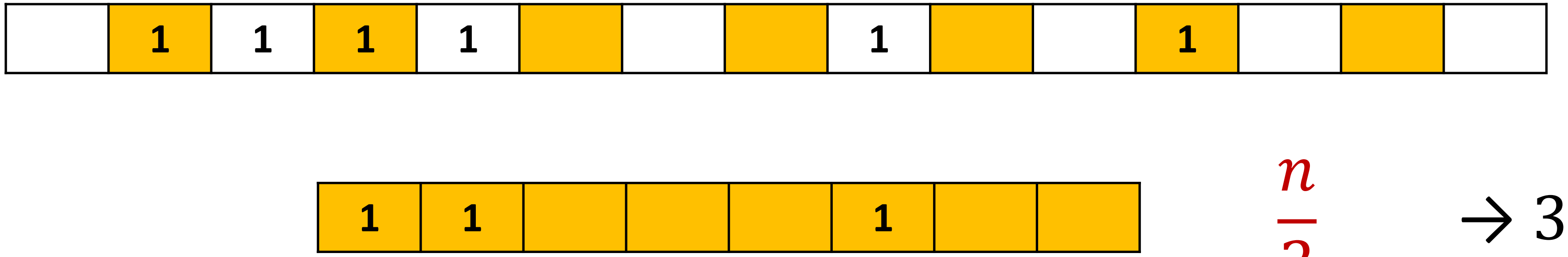
Distinct Elements

	1	1	1	1				1			1			

n

S: 3, 5, 9, 3, 4, 2, 12

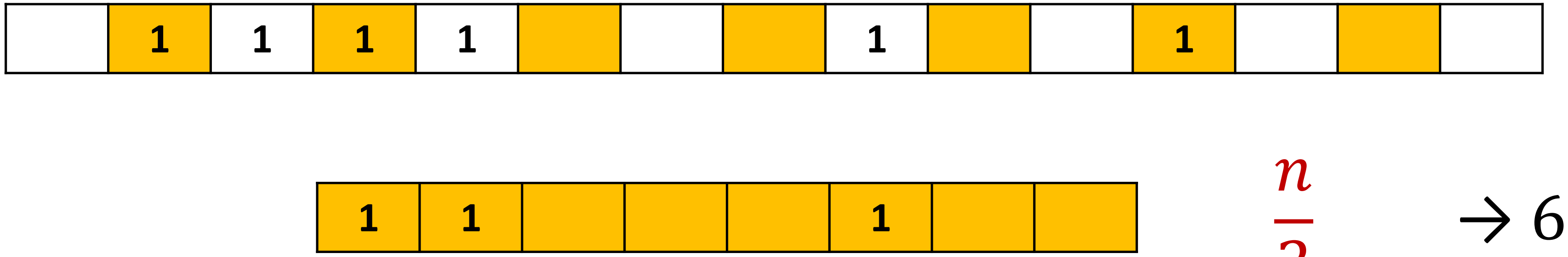
Distinct Elements



$\frac{n}{2} \to 3$

S: 3, 5, 9, 3, 4, 2, 12

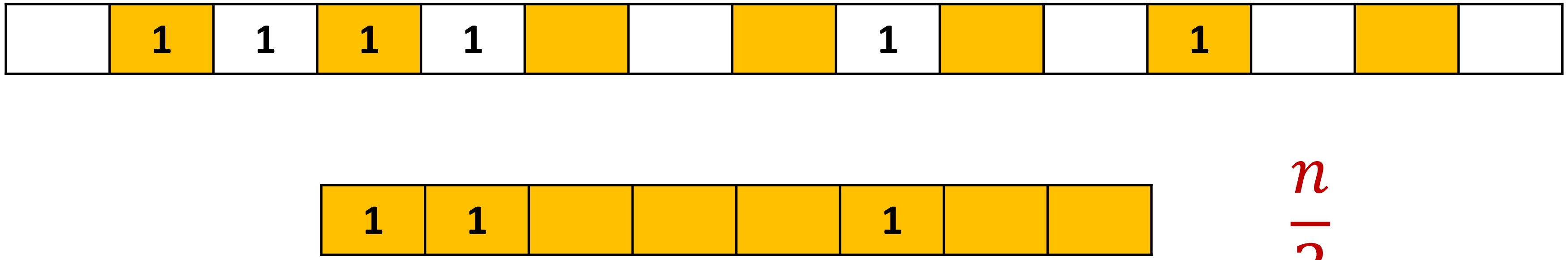
Distinct Elements



$$\frac{n}{2} \to 6$$

Sample with probability $\frac{1}{2}$, rescale by 2

Distinct Elements



$$\frac{n}{2}$$

$$\mathrm{E}[X] = 2 \sum_{i \text{ in } S} E[X_i]$$

Distinct Elements

	1	1	1	1				1			1			

1	1				1		

$$\frac{n}{2}$$

$$\mathrm{E}[X] = 2 \sum_{i \text{ in } S} E[X_i]$$

$$X_i = 0 \text{ w.p.} \frac{1}{2}$$

$$X_i = 1 \text{ w.p.} \frac{1}{2}$$

Distinct Elements

	1	1	1	1				1			1			

1	1				1		

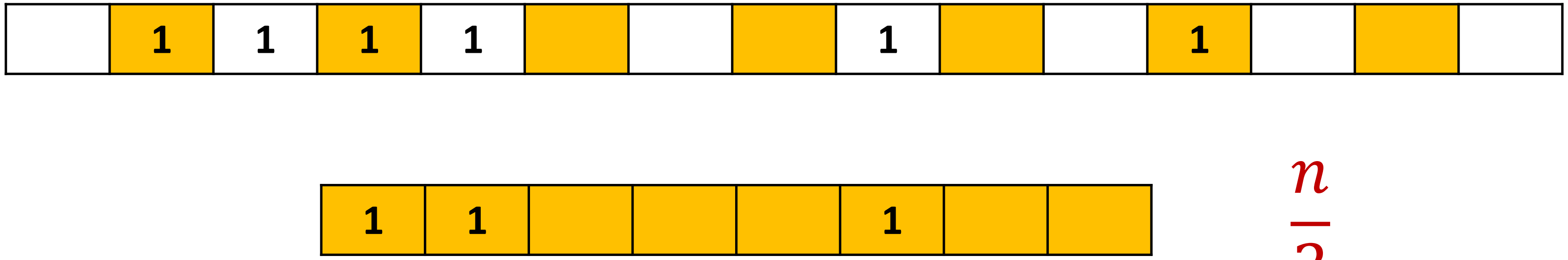
$$\frac{n}{2}$$

$$\mathrm{E}[X] = 2\sum_{i \text{ in } S} \frac{1}{2}$$

$$X_i = 0 \text{ w.p.} \frac{1}{2}$$

$$X_i = 1 \text{ w.p.} \frac{1}{2}$$

Distinct Elements



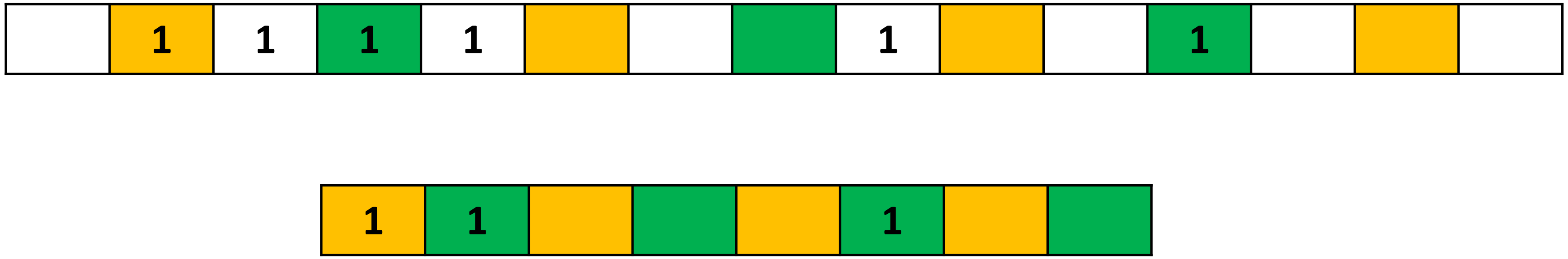
$\frac{n}{2}$

$$\mathrm{E}[X] = 2 \sum_{i \text{ in } S} \frac{1}{2} = F$$

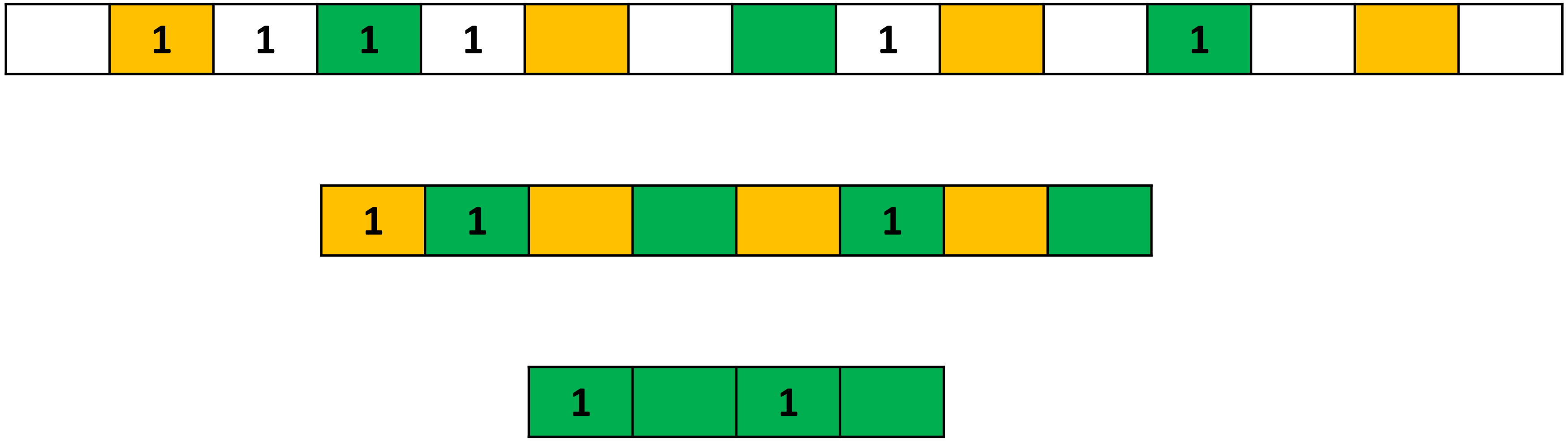
$$X_i = 0 \text{ w.p.} \frac{1}{2}$$

$$X_i = 1 \text{ w.p.} \frac{1}{2}$$

Distinct Elements

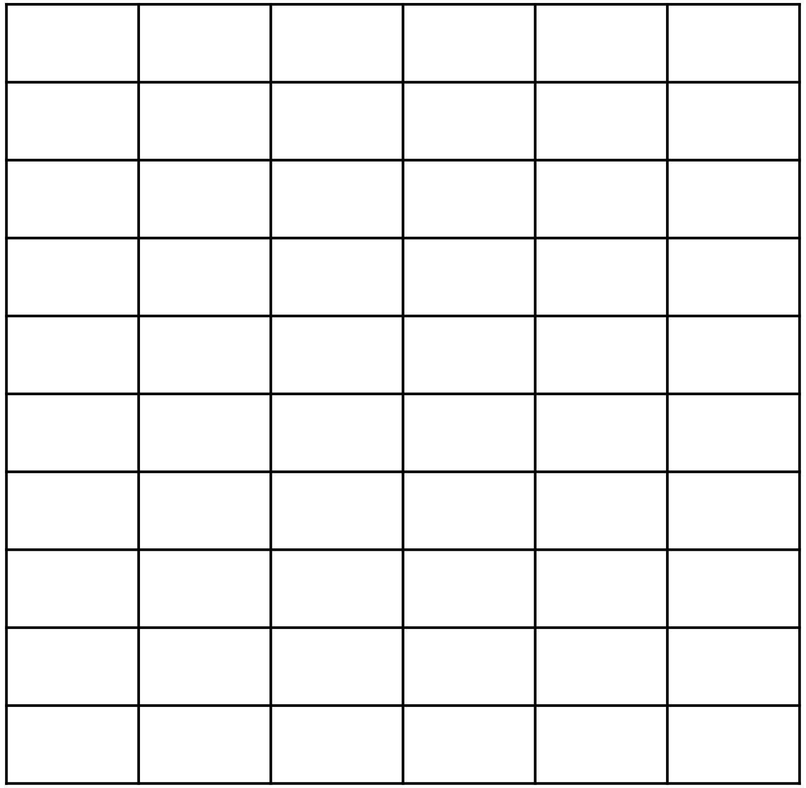


Distinct Elements



Distinct Elements

$\log n$



$$\frac{1}{\epsilon^2}$$

Distinct Elements

$\log n$

	1				
	1				

$$\frac{1}{\epsilon^2}$$

Distinct Elements

$\log n$

	1	1			
	1				

$\frac{1}{\epsilon^2}$

Distinct Elements

	1	1	1		
	1		1		
			1		
			1		
			1		
			1		

$\log n$

$$\frac{1}{\epsilon^2}$$

Distinct Elements

1	1	1	1		
	1		1		
			1		
			1		
			1		
			1		

$\log n$

$$\frac{1}{\epsilon^2}$$

Distinct Elements

1	1	1	1	1	
	1		1		
			1		
			1		
			1		
			1		

$\log n$

$\frac{1}{\epsilon^2}$

Distinct Elements

1	1	1	1	1	
	1		1		
	1		1		
			1		
			1		
			1		

$\log n$

$$\frac{1}{\epsilon^2}$$

Distinct Elements

1	1	1	1	1	1
	1		1		
	1		1		
			1		
			1		
			1		

$\log n$

$\frac{1}{\epsilon^2}$

Distinct Elements

1	1	1	1	1	1
	1		1	1	
	1		1		
			1		
			1		
			1		

$\log n$

$$\frac{1}{\epsilon^2}$$

Distinct Elements on Sliding Windows

- ❖ Need $O\left(\frac{1}{\epsilon}\log n\right)$ instances of the algorithm
- ❖ Each instance uses $O\left(\frac{1}{\epsilon^2}\log n\right)$ bits of space

- ❖ Key observation: Instances are monotonic

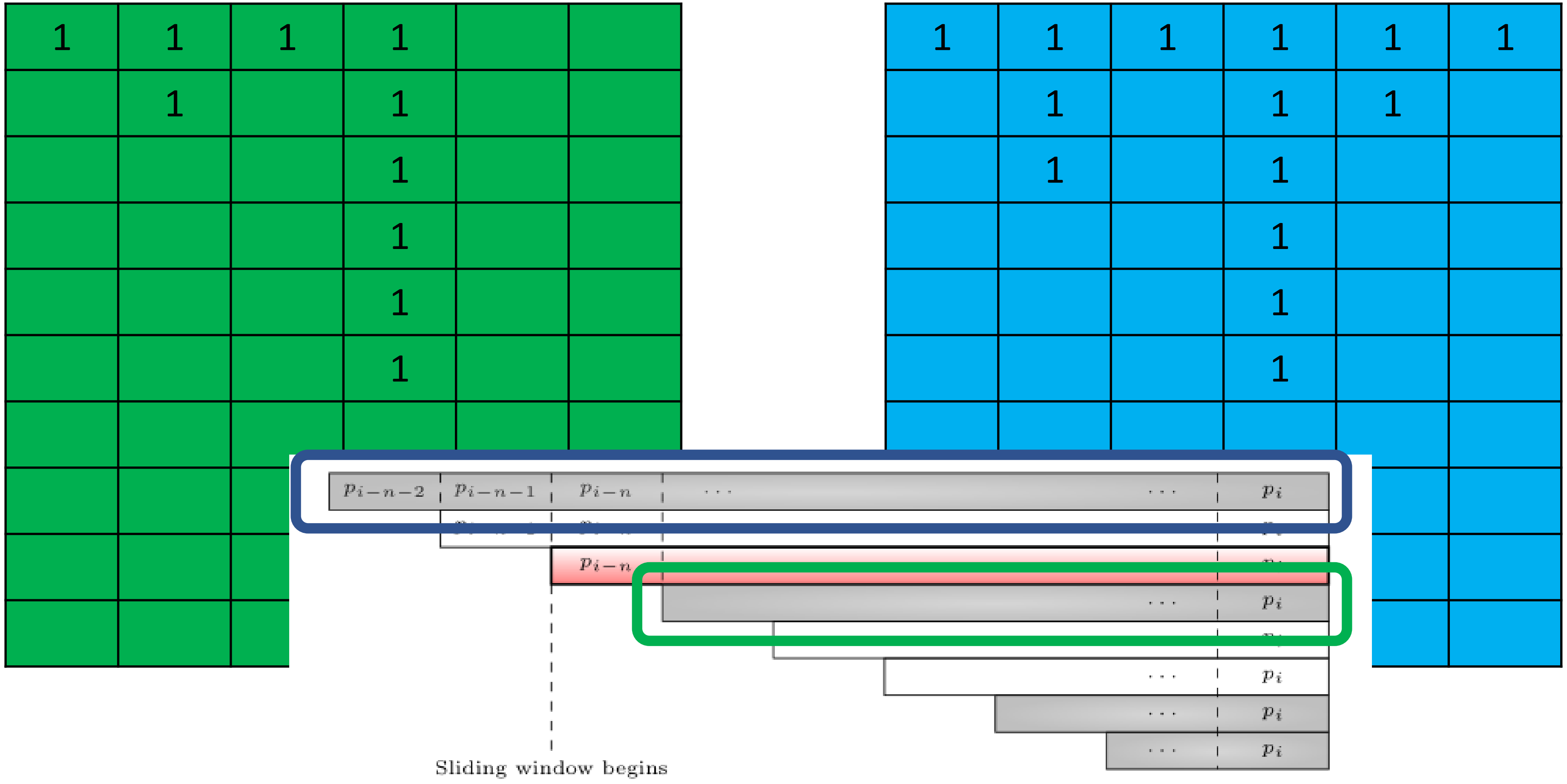
No need to repeat entire table for each instance

$\log n$

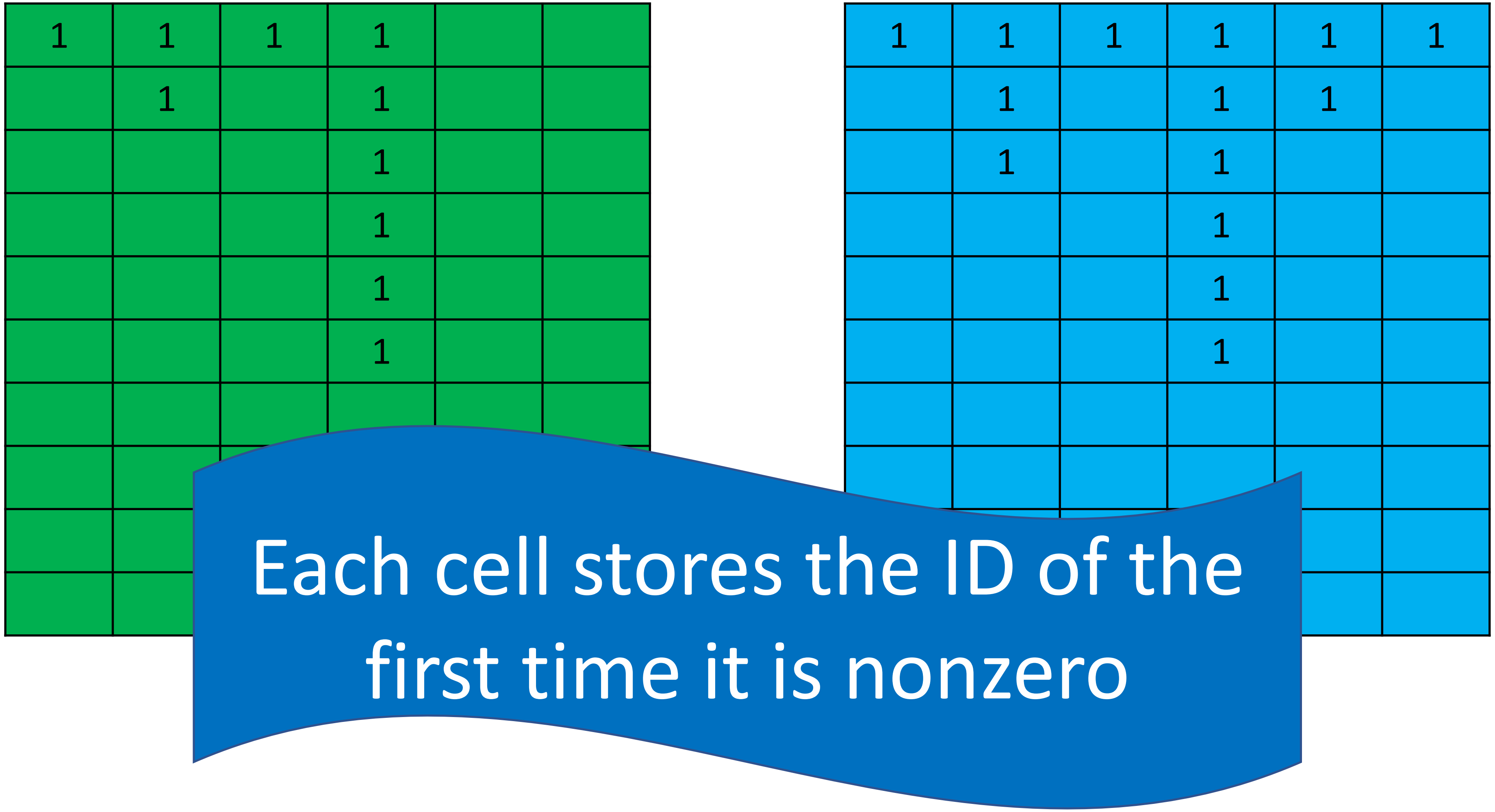
1	1	1	1	1	1
	1		1	1	
	1		1		
			1		
			1		
			1		

$\frac{1}{\epsilon^2}$

Distinct Elements on Sliding Windows



Distinct Elements on Sliding Windows



Distinct Elements on Sliding Windows

1	1	1	1		
	1		1		
			1		
			1		
			1		
			1		

1	1	1	1	1	1
	1		1	1	
	1		1		
			1		
			1		
			1		

Distinct Elements on Sliding Windows

1	1	1	1		
	1		1		
			1		
			1		
			1		
			1		

1	1	1	1	1	1
	1		1	1	
	1		1		
			1		
			1		
			1		

1	1	1	1	2	2
	1		1	2	
	2		1		
			1		
			1		
			1		

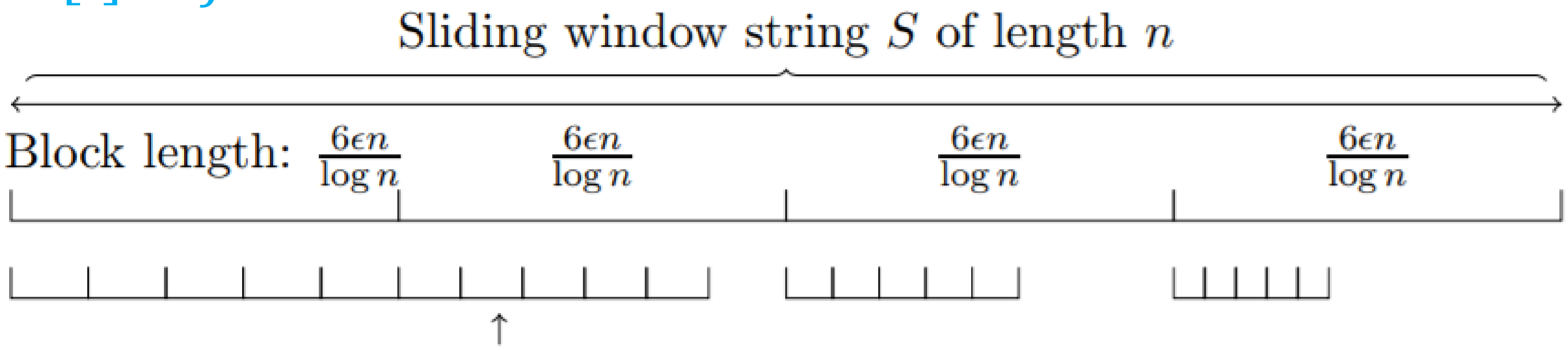
Distinct Elements on Sliding Windows

- $O\left(\frac{1}{\epsilon^2}\log n\right)$ cells, each using $O\left(\log\log n + \log\frac{1}{\epsilon}\right)$ space
- Can get down to $O\left(\frac{1}{\epsilon^2}\log n \log\frac{1}{\epsilon}\right)$ with one more observation
- Monotonic columns
- Encode each column using $O\left(\log n \log\frac{1}{\epsilon}\right)$ bits.

1	1	1	1	2	2
3	1	3	1	2	3
4	2	3	1	3	4
4	3	3	1		
4		4	1		
			1		
			4		

Technical Caveats

- Maintenance of smooth histogram uses correctness of all instances
- $\Omega\left(\frac{1}{\epsilon^2}\log n\right)$ Lower Bound
  - IndexGreater problem: Alice is given S, Bob is given i, j and must decide whether $S[i] \leq j$.



Elements $\{0, 1, \ldots, (1+2\epsilon)^i - 1\}$ inserted into piece x_i of block i.

Alice: $\underbrace{x_1}\ldots x_m$, where $m = \frac{1}{6\epsilon}\log n$.

Each x_k is $\frac{1}{2}\log n$ bits.

Technical Caveats

- GapHamming: Alice receives x and Bob receives y, each of length n, and must decide if $\mathrm{HAM}(x,y)=\frac{n}{2}+\sqrt{n}$ or $\mathrm{HAM}(x,y)=\frac{n}{2}-\sqrt{n}$.
- $\Omega\left(\frac{1}{\epsilon}\log^2 n\right)$ Lower Bound
  - Idea: Embed $O(\log n)$ instances of GapHamming into the stream.
  - $(1+\epsilon)$-approximation for $\epsilon\leq\frac{2}{\sqrt{n}}$ already decides.

Questions?


thank you
danke
謝謝
ngiyabonga
teşekkür ederim
gracias
merci
obrigado
bedankt
dziękuję
sukriya
kop khun krap
go raibh maith agat
mochchakkeram
tapadh leat
terima kasih
grazie
arigatô
dakujem
takk
taiku
sagolun
hvala
maururu
рахмат
Баярлалаа
спасибо
faafetai lava
mersi
kia ora
barka
welalin
tack
spas
dank je
misaotra
matondo
paldies
grazzi
mahalo
vinaka
blagodaram
спасибі
kiitos
dankie
nanni
nandri
dhanyavad
köszönöm
bayarlalaa
gràçie
enkosi
sobodi
děkuji
akun
dankon
ačiū
хвала
asante
manana
obrigada
murakoze
tenki
djiere dieuf
tau
дякую
mamnun
chokrane
chnorakaloutioun
gratias ago
gràcies
sulpáy
trugarez
mèsi
didi madloba
najis tuke
kam sah hamnida
rahmat
ありがとう
tanemirt
rahmet
diolch
dhanyavadagalu
shukriya
mercé
мерси
তোমাকে ধন্যবাদ
감사합니다
xièxie
ευχαριστώ