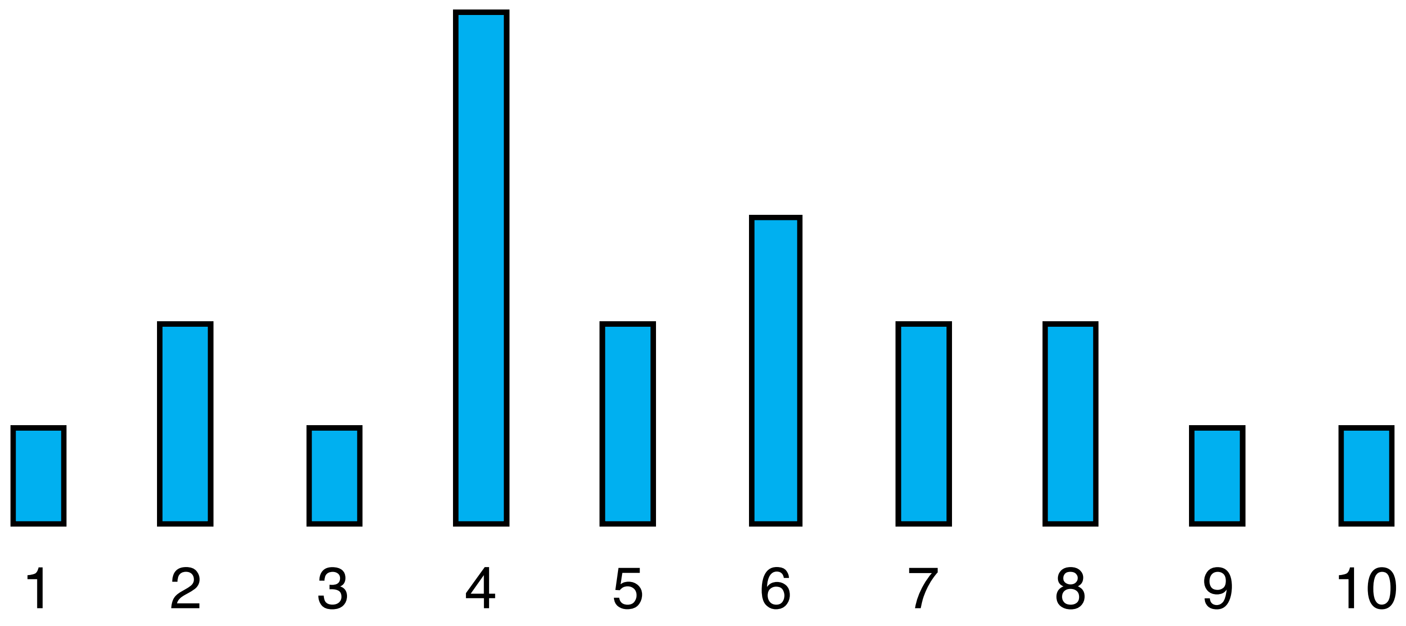


Improved Frequency Estimation Algorithms with and without Predictions

Anders Aamand* Justin Y. Chen* Huy Nguyen$ Sandeep Silwal* Ali Vakilian^

*MIT, $Northeastern, ^TTIC

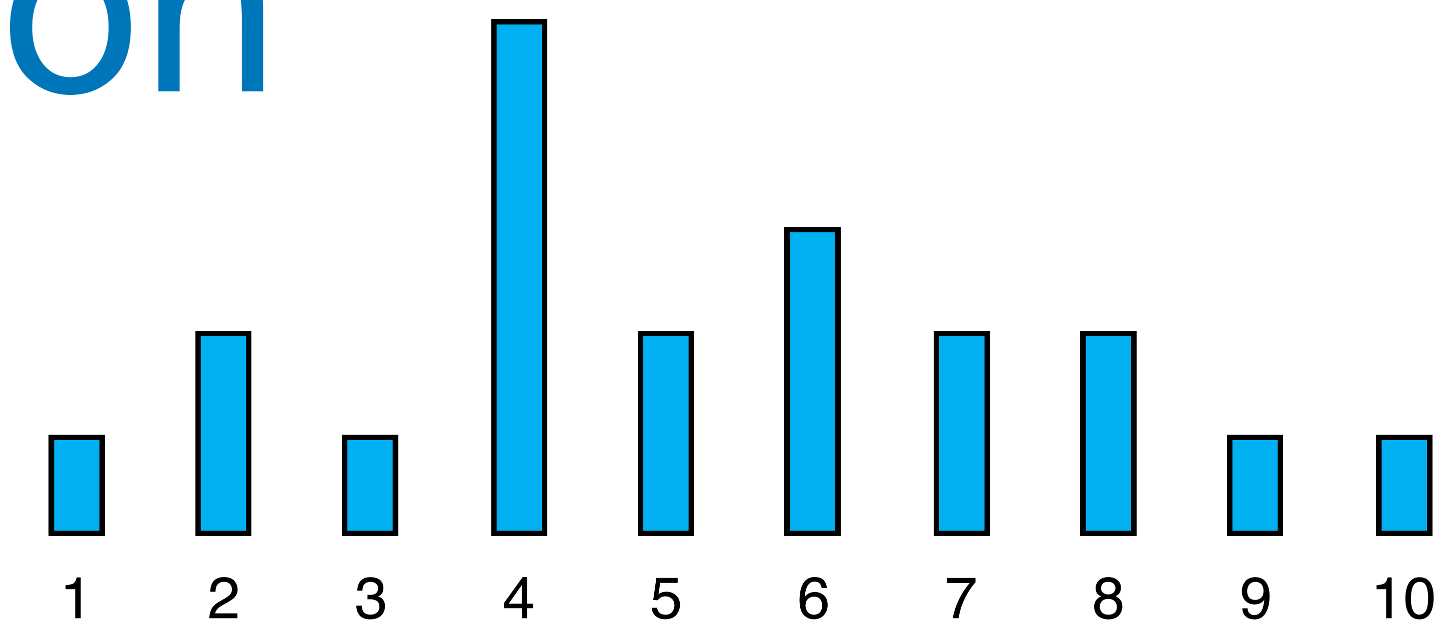
Thanks to Piotr Indyk for some slides

Will appear at NeurIPS '23

TLDR: Better frequency estimation algorithms under natural assumptions

Some background first …

Frequency Estimation



Data stream S: a sequence of items from [n]
- E.g.: 8, 1, 7, 4, 6, 4, 10, 4, 4, 6, 8, 7, 5, 4, 2, 5, 6, 3, 9, 2

- Goal: at the end of the stream, given item $i \in [n]$ output an estimation $\tilde{f}_i$ of the frequency f_i in S

- Sub-linear space ?

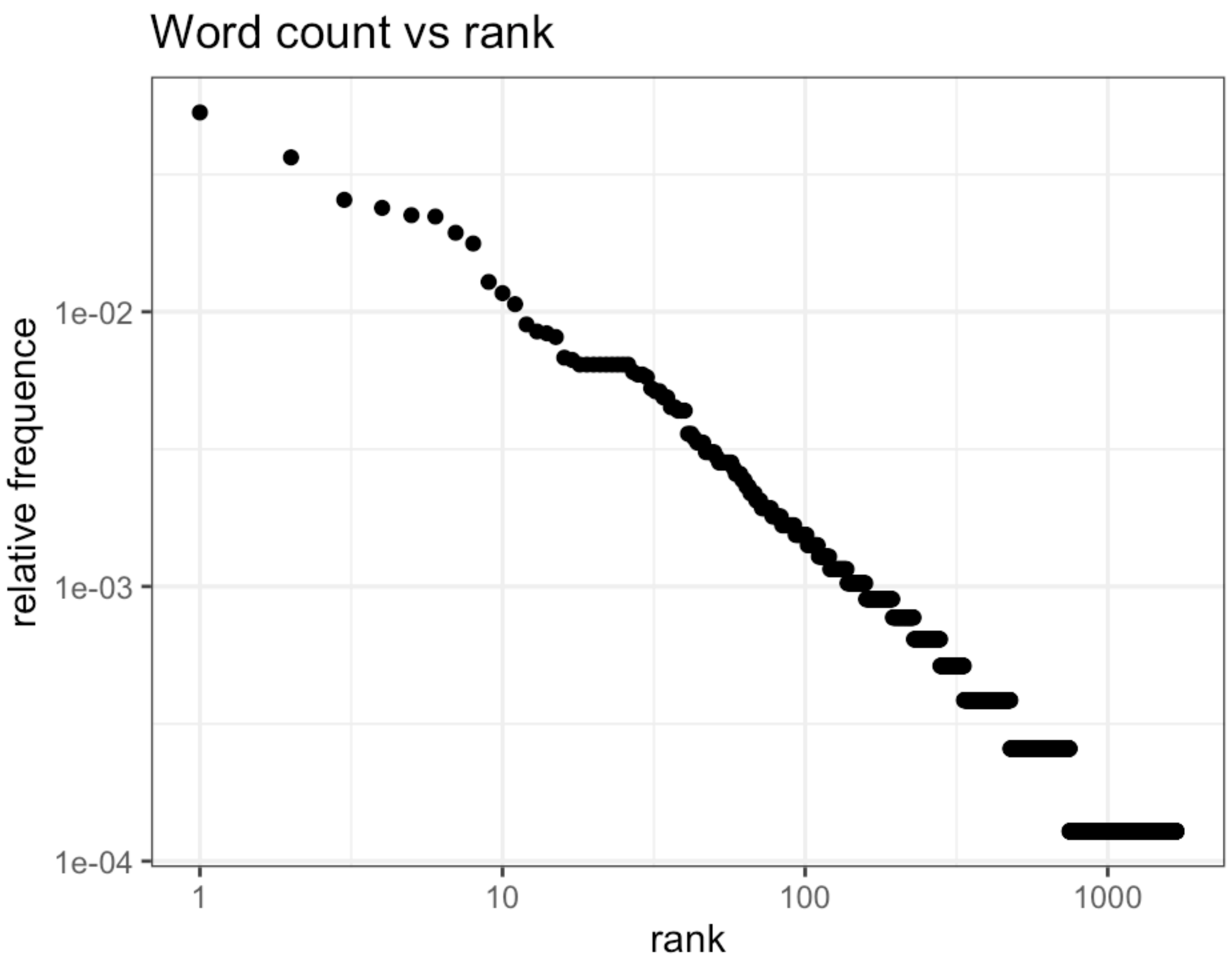
Two heroes (assumptions)

- Learning-augmented: Access to heavy hitter predictions

- Zipfian: True frequencies are heavy-tailed

  (for theory analysis)

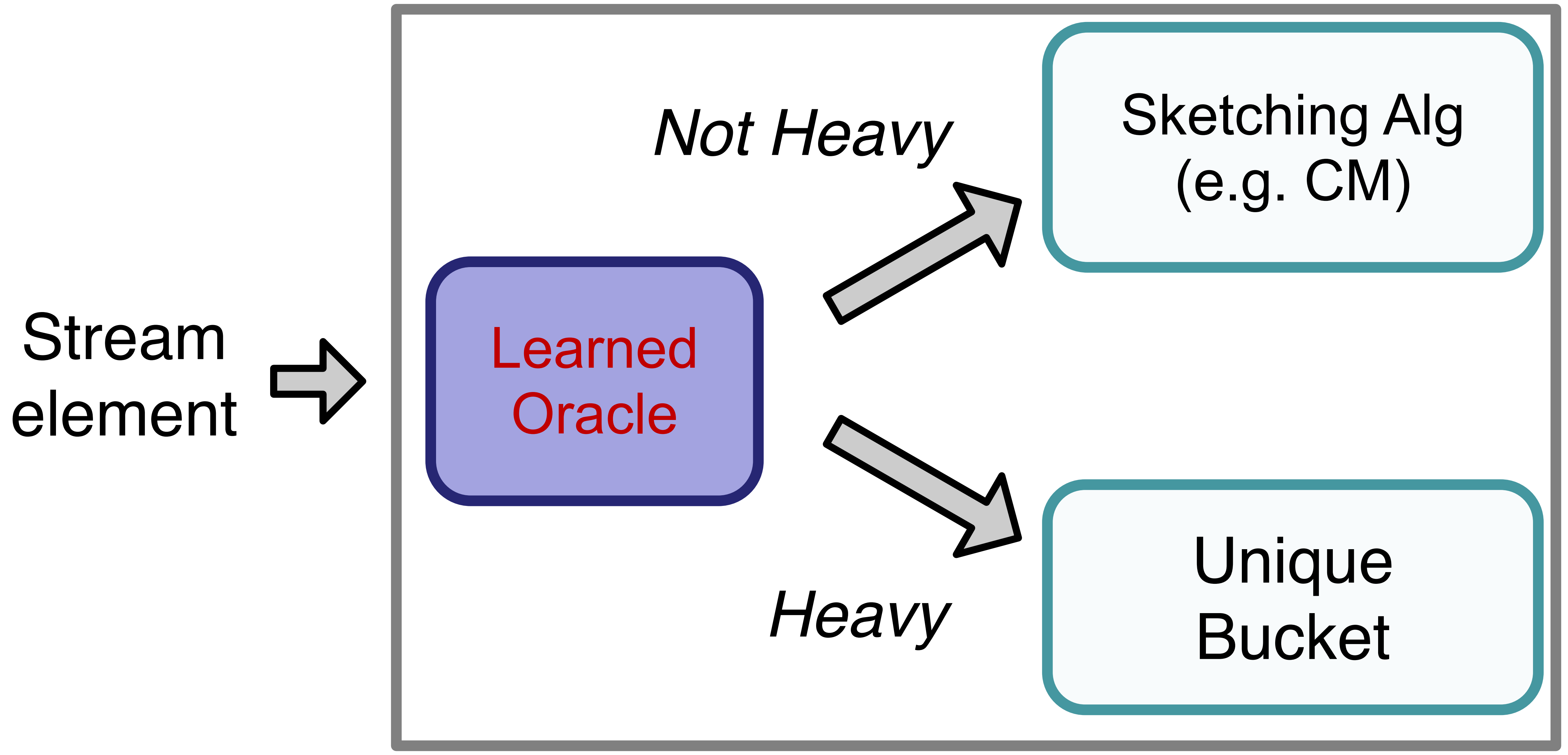
Word frequencies in “Moby Dick” (Wikipedia)



The story so far [Hsu-Indyk-Katabi-Vakilian, ICLR'19]

- Augment sketching algorithms like CountMin (CM) or CountSketch (CS) with heavy-hitter predictions.

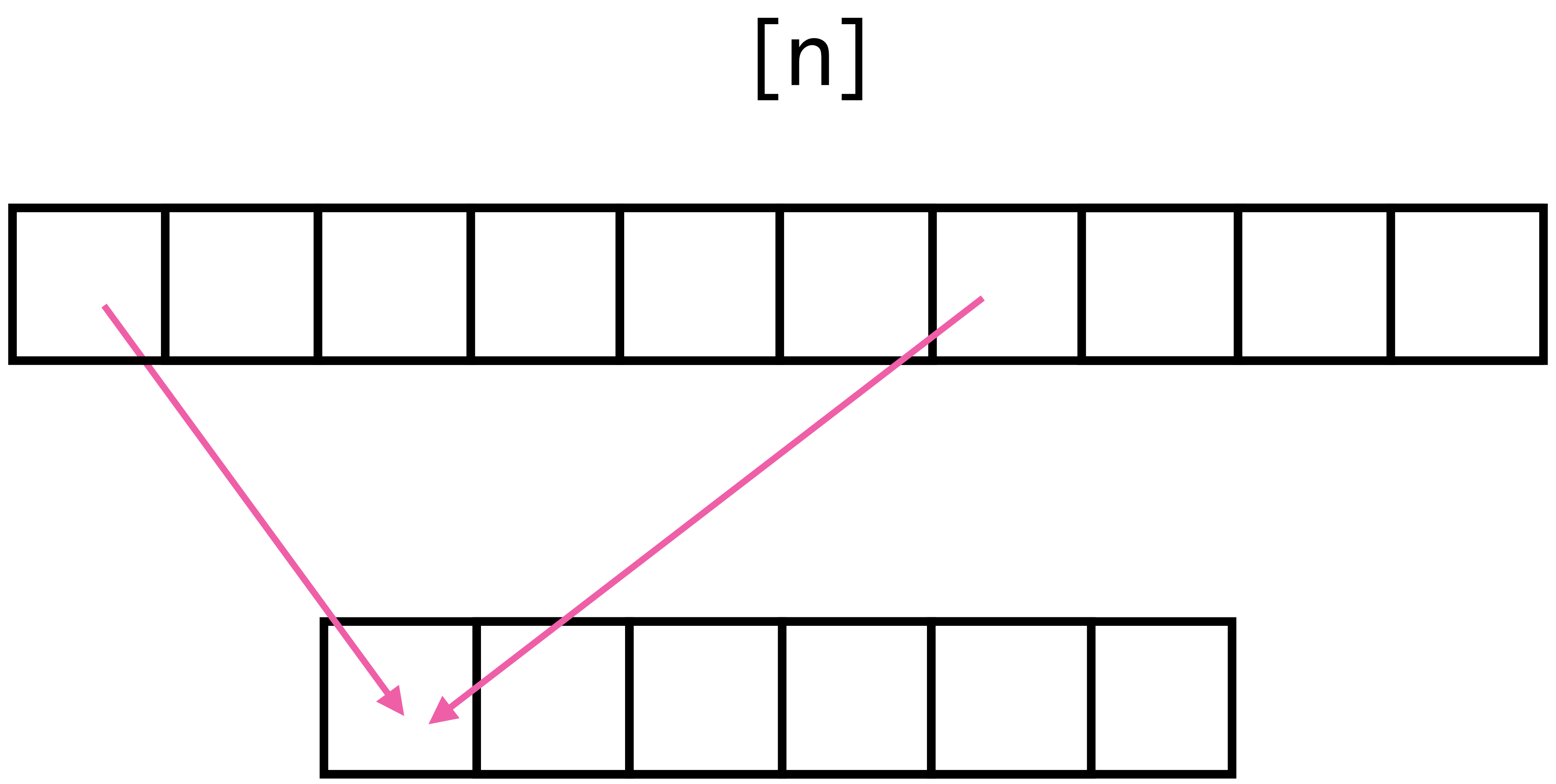
"Learned CM/CS"



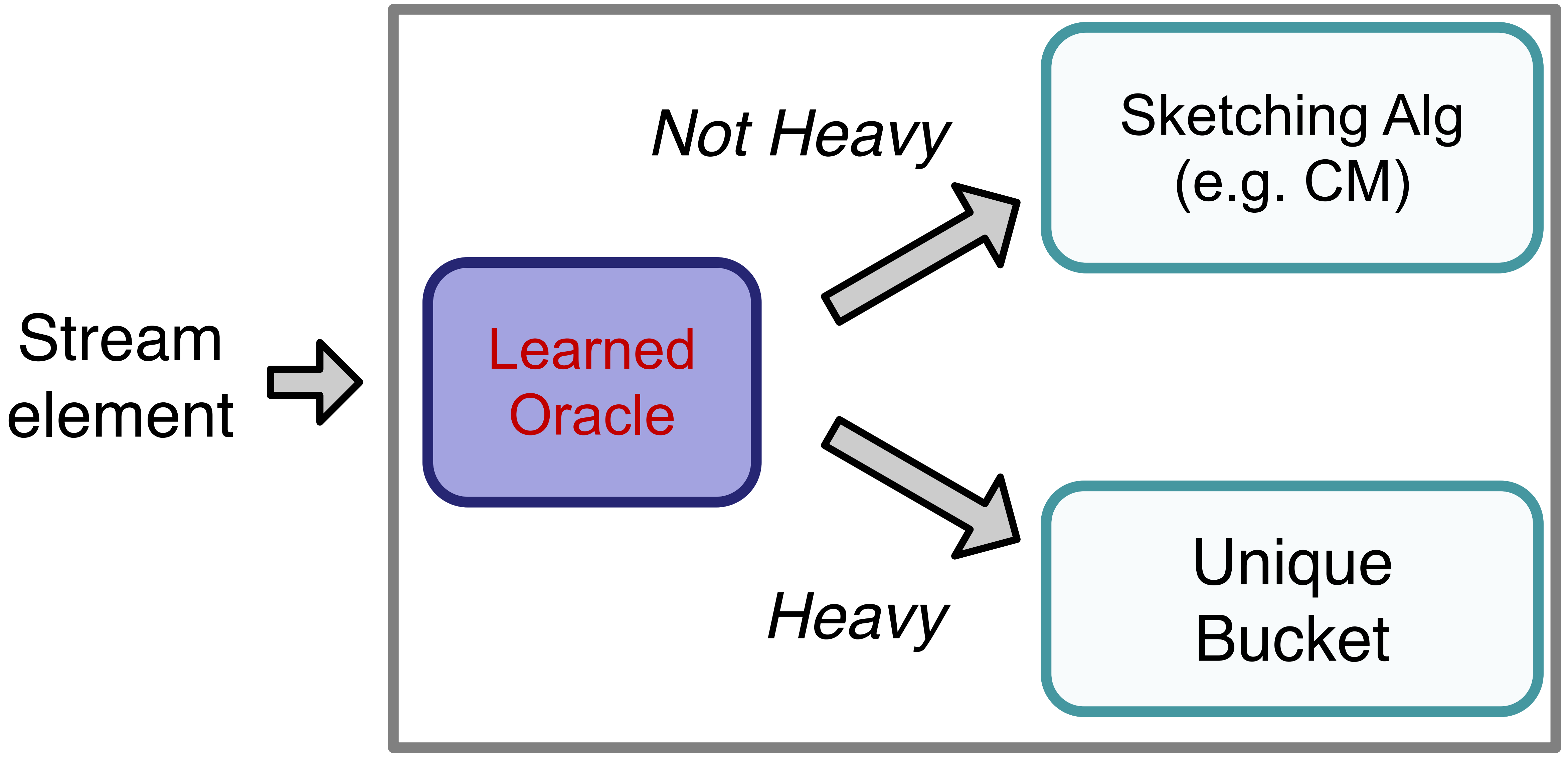
The story so far [Hsu et al.]

- Augment sketching algorithms like CountMin (CM) or CountSketch (CS) with heavy-hitter predictions.

- Short and sweet intuition: heavy elements are responsible for error!



“Learned CM/CS”



Theoretical Analysis of [Hsu et al.]

$$f_i \propto 1/i$$

- Analyzed it under Zipfian distribution
- Error metric: $\mathbb{E}\left[\sum_i f_i \cdot |\tilde{f}_i - f_i|\right]$

  > "Average" error (Want better predictions for larger frequencies)

  > Generalizes to other weights such as uniform

  > Used in practical works ([Roy et al'16])

Theoretical Analysis of [Hsu et al.]

$f_i \propto 1/i$

- Analyzed it under Zipfian distribution
- Error metric: $\sum_i f_i \cdot |\tilde{f}_i - f_i|$
- B words of space
- Predictor correctly predicts top B heavy elements

CountMin (CM)	1
Learned-CM	$O\left(\frac{\log(n/B)}{\log n}\right)^2$

Is the story over?

CountMin (CM)	1
Learned-CM	$O\left(\frac{\log(n/B)}{\log n}\right)^2$

Next Steps Questions:

- Advantage only when space (B) is large?
- What about CountSketch?

Is the story over?

CountMin (CM)	1
Learned-CM	$O\left(\frac{\log(n/B)}{\log n}\right)^2$

Next Steps Questions:

- Advantage only when space (B) is large?
- What about CountSketch?

Meta:

- Can we just get better algorithms under Zipfian assumptions?
- Do we even need predictions? “New” algorithms?

Our Results

Space = B words, n = # of elements, Error metric: $\sum_i f_i \cdot |\tilde{f}_i - f_i|$

CountMin (CM)	1
Learned-CM	$\Theta\left(\frac{\log(n/B)}{\log n}\right)^2$
CountSketch (CS)	$\Theta\left(\frac{1}{\log n}\right)$
Learned-CS	$\Theta\left(\frac{\log(n/B)}{(\log n)^2}\right)$

Characterize exact performance of classical algos and learned variants (for our error metric and Zipfian data)

Our Results

Space = B words, n = # of elements, Error metric: $\sum_i f_i \cdot |\tilde{f}_i - f_i|$

CountMin (CM)	1
Learned-CM	$\Theta\left(\frac{\log(n/B)}{\log n}\right)^2$
CountSketch (CS)	$\Theta\left(\frac{1}{\log n}\right)$
Learned-CS	$\Theta\left(\frac{\log(n/B)}{(\log n)^2}\right)$

Learned variants are only useful over classic counterparts for large space

Space = B words, n = # of elements
Error metric: $\sum_i f_i \cdot |\tilde{f}_i - f_i|$

Learned-CS	$\Theta\left(\frac{\log(n/B)}{\log n}\right)^2$
New Alg	$O\left(\frac{\log B}{(\log n)^2}\right)$
Learned-New Alg	$O\left(\frac{1}{(\log n)^2}\right)$

- For small space (B ~ polylog n): New alg outperforms even prior learned variants *without predictions*
- Even better *with predictions*

Space = B words, n = # of elements

Error metric: $\sum_i f_i \cdot |\tilde{f}_i - f_i|$

CountMin (CM)	1
Learned-CM	$\Theta\left(\frac{\log(n/B)}{\log n}\right)^2$
CountSketch (CS)	$\Theta\left(\frac{1}{\log n}\right)$
Learned-CS	$\Theta\left(\frac{\log(n/B)}{(\log n)^2}\right)$
New Alg	$O\left(\frac{\log B}{(\log n)^2}\right)$
Learned-New Alg	$O\left(\frac{1}{(\log n)^2}\right)$

- For small space (B ~ polylog n): New alg outperforms even prior learned variants *without predictions*
- Even better *with predictions*

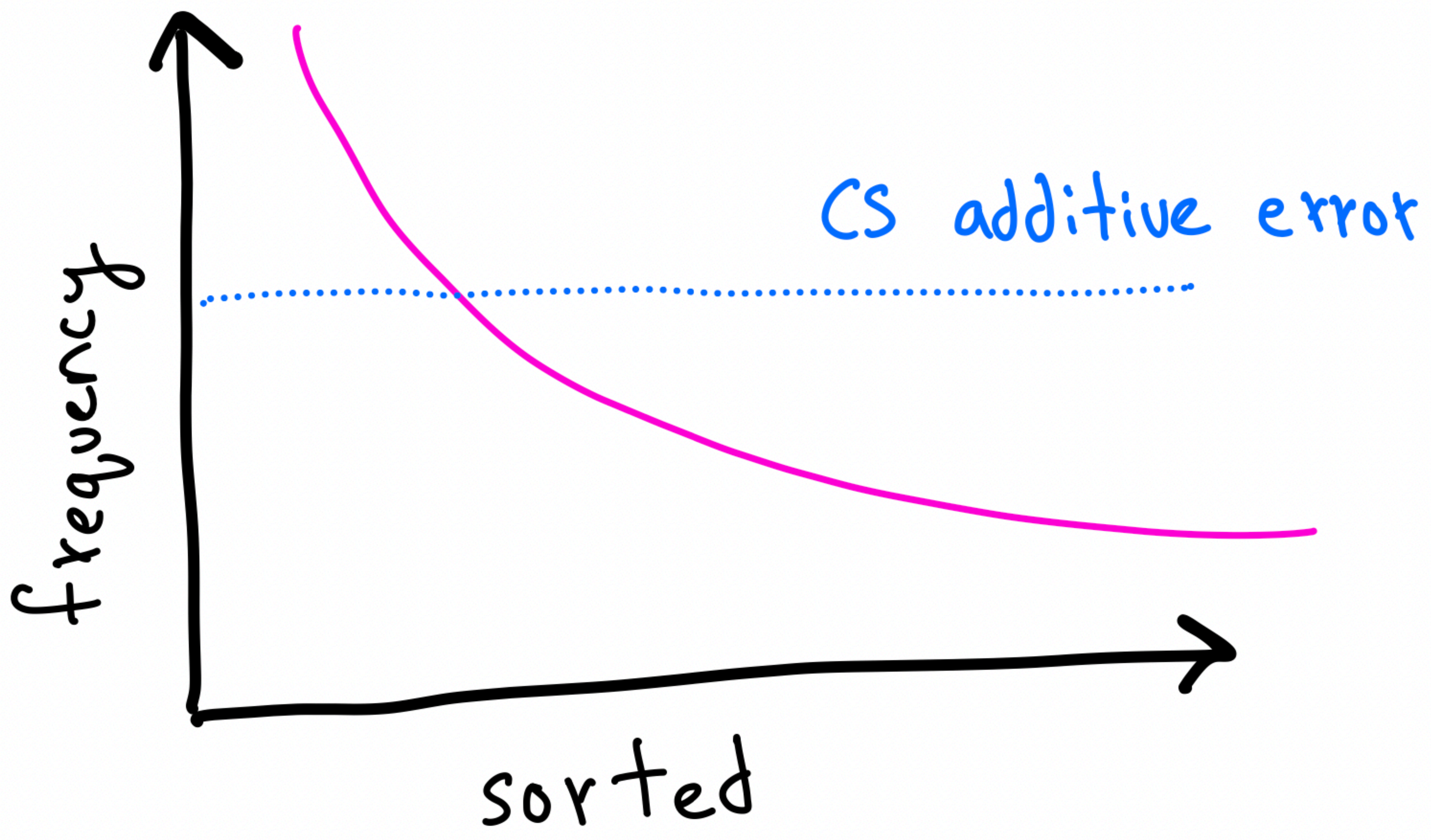
Space = B words, n = # of elements
Error metric: $\sum_i f_i \cdot |\tilde{f}_i - f_i|$

CountMin (CM)	1
Learned-CM	$\Theta\left(\frac{\log(n/B)}{\log n}\right)^2$
CountSketch (CS)	$\Theta\left(\frac{1}{\log n}\right)$
Learned-CS	$\Theta\left(\frac{\log(n/B)}{(\log n)^2}\right)$
New Alg	$O\left(\frac{\log B}{(\log n)^2}\right)$
Learned-New Alg	$O\left(\frac{1}{(\log n)^2}\right)$

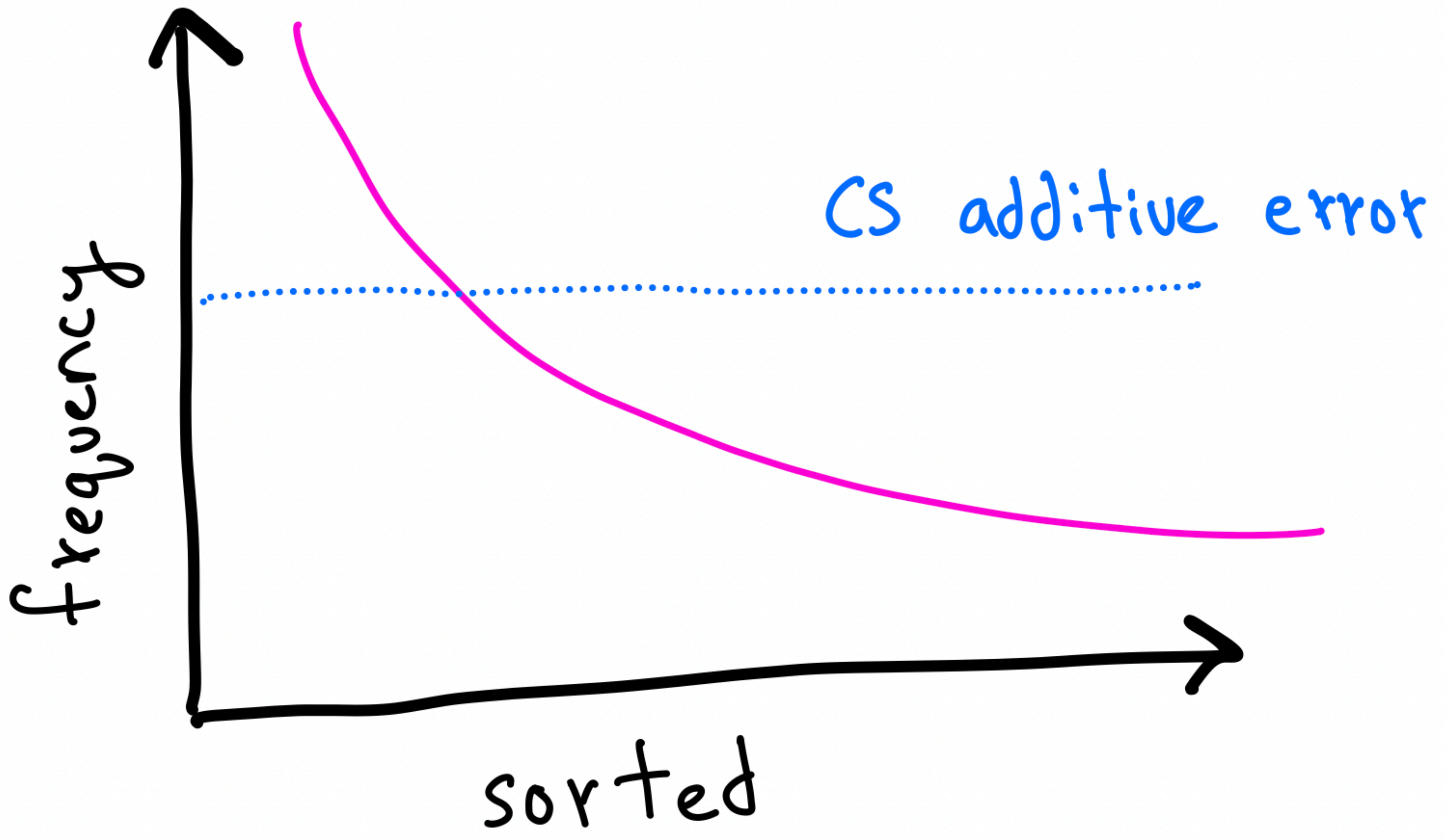
Zipfian assumption is also common in theory (e.g. [Milton-Price ’14])

Usually studied for ‘point wise’ error

New Algorithm Intuition (No predictions)



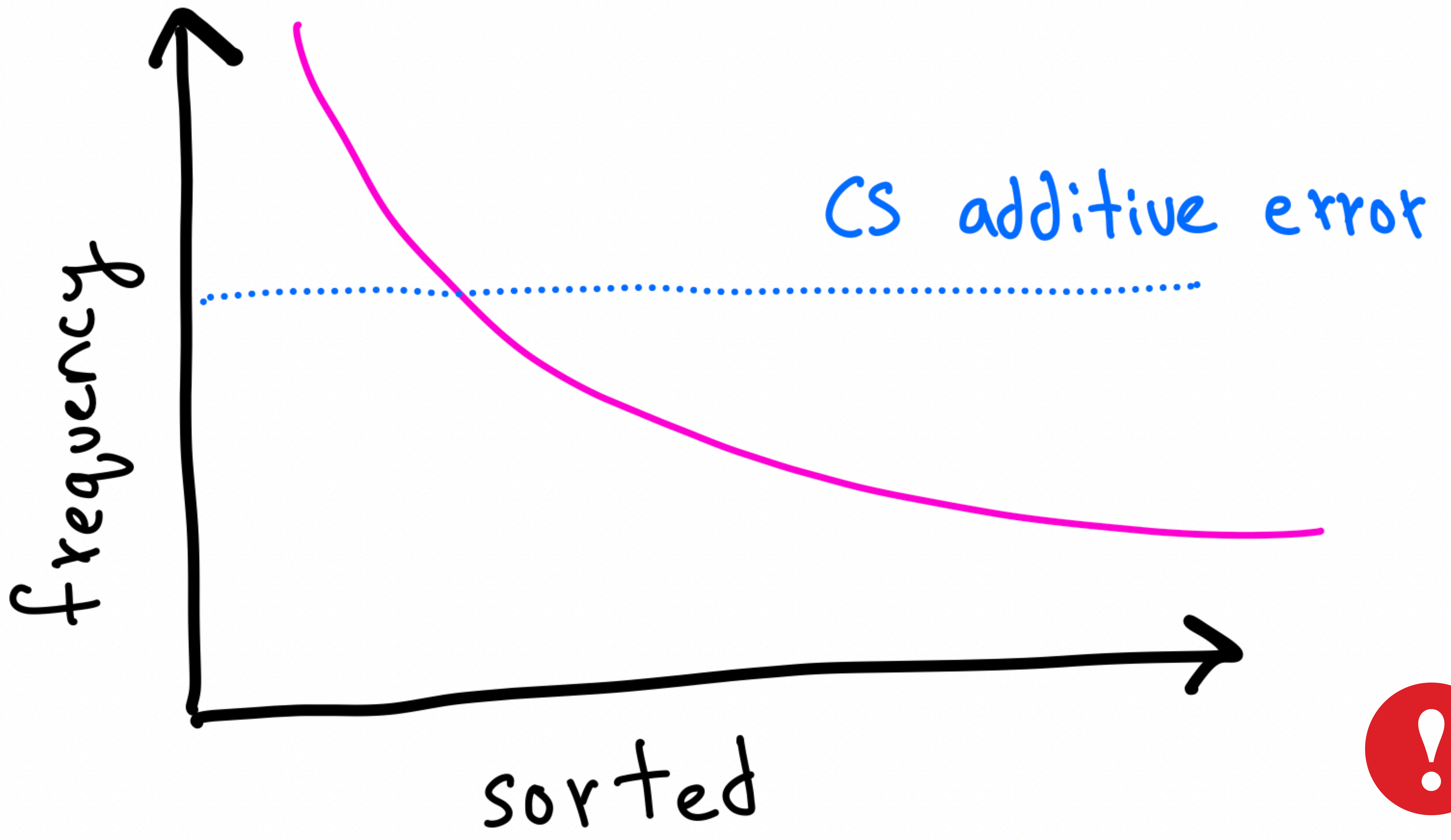
New Algorithm Intuition



Why would you incur large error for small elements?

Better to predict them as 0!

New Algorithm Intuition

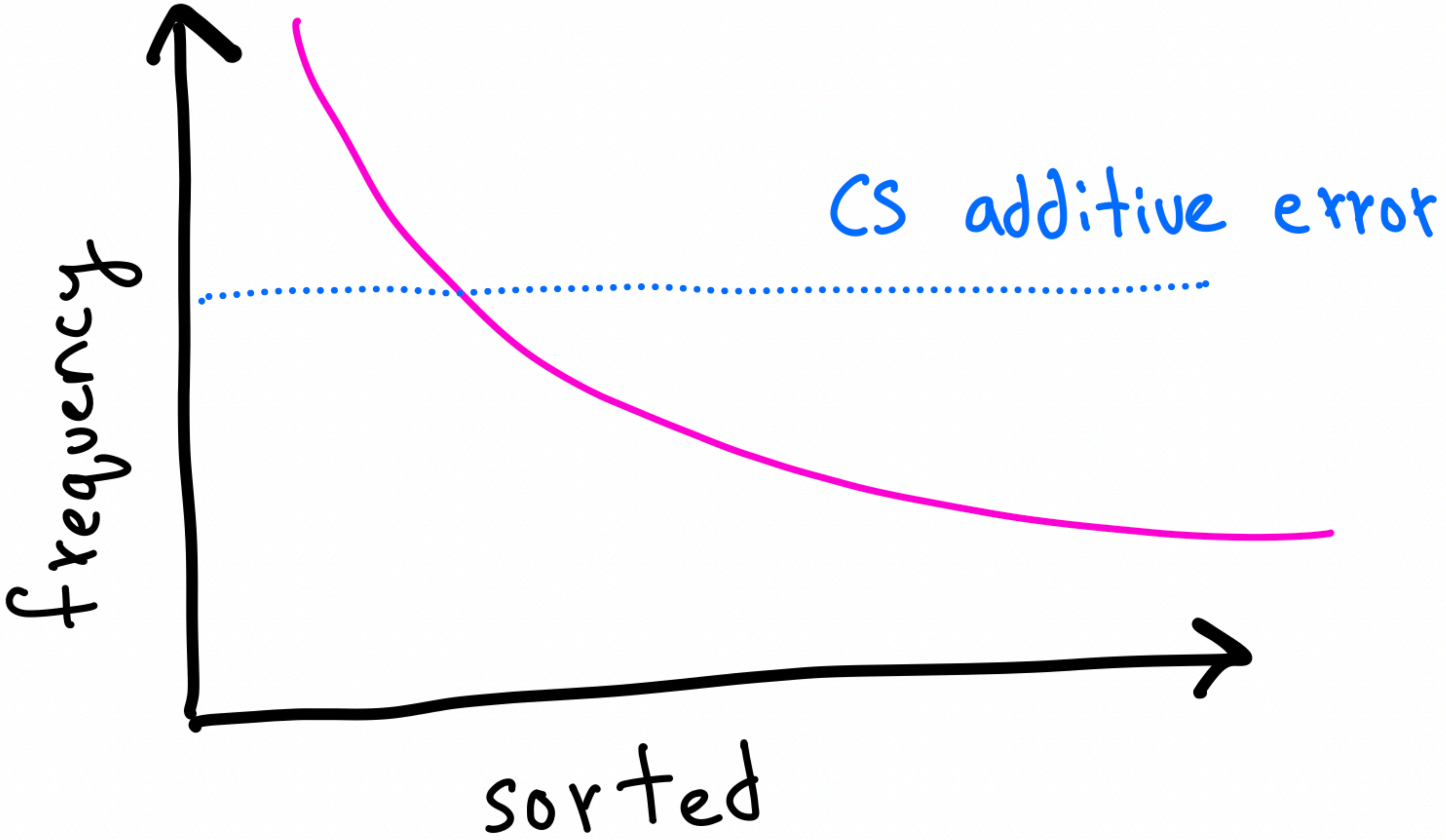


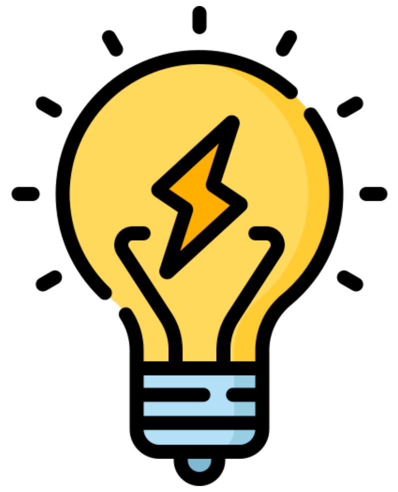
Why would you incur large error for small elements?

Better to predict them as 0!

Don't know before hand which elements are small

New Algorithm Intuition



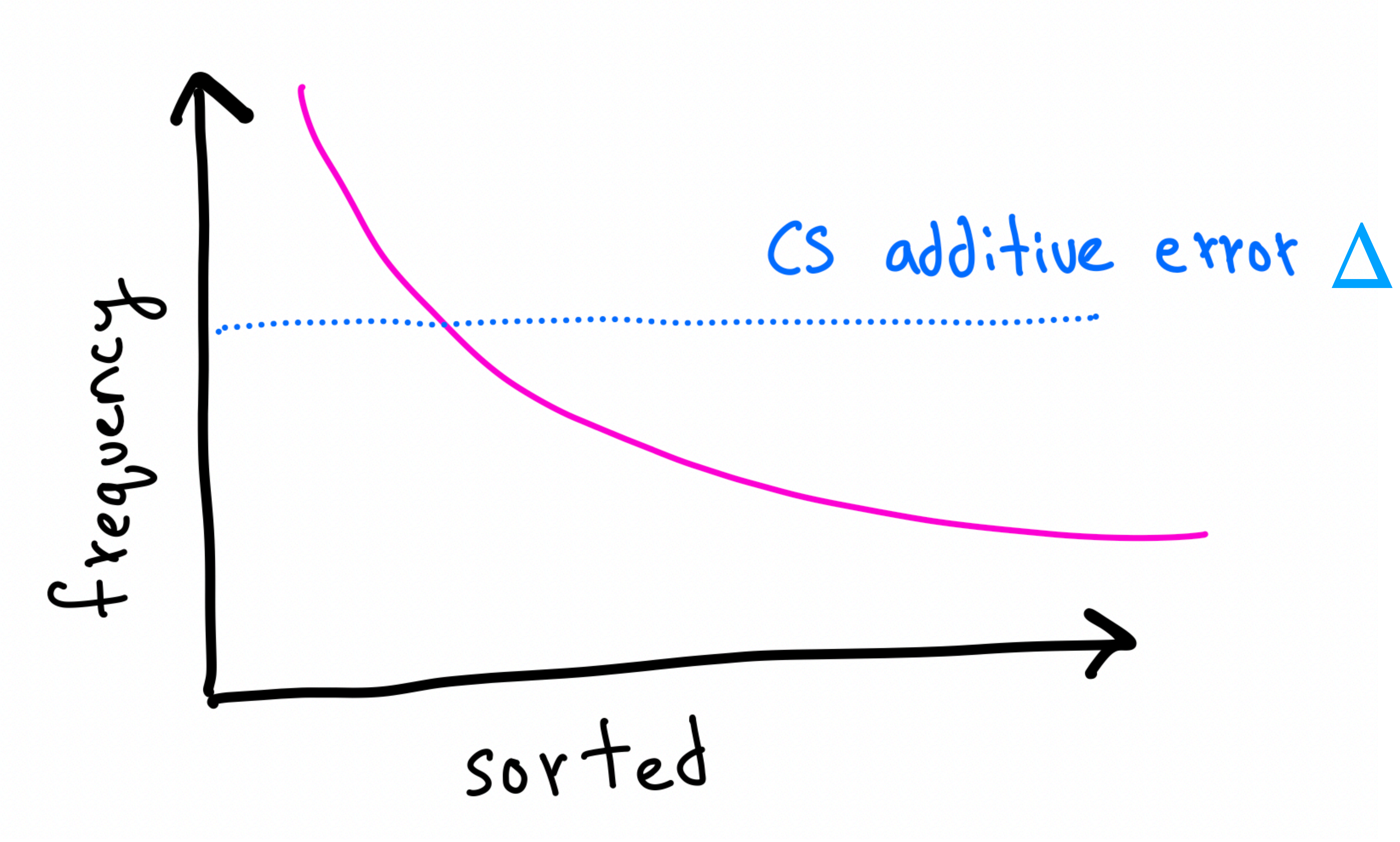


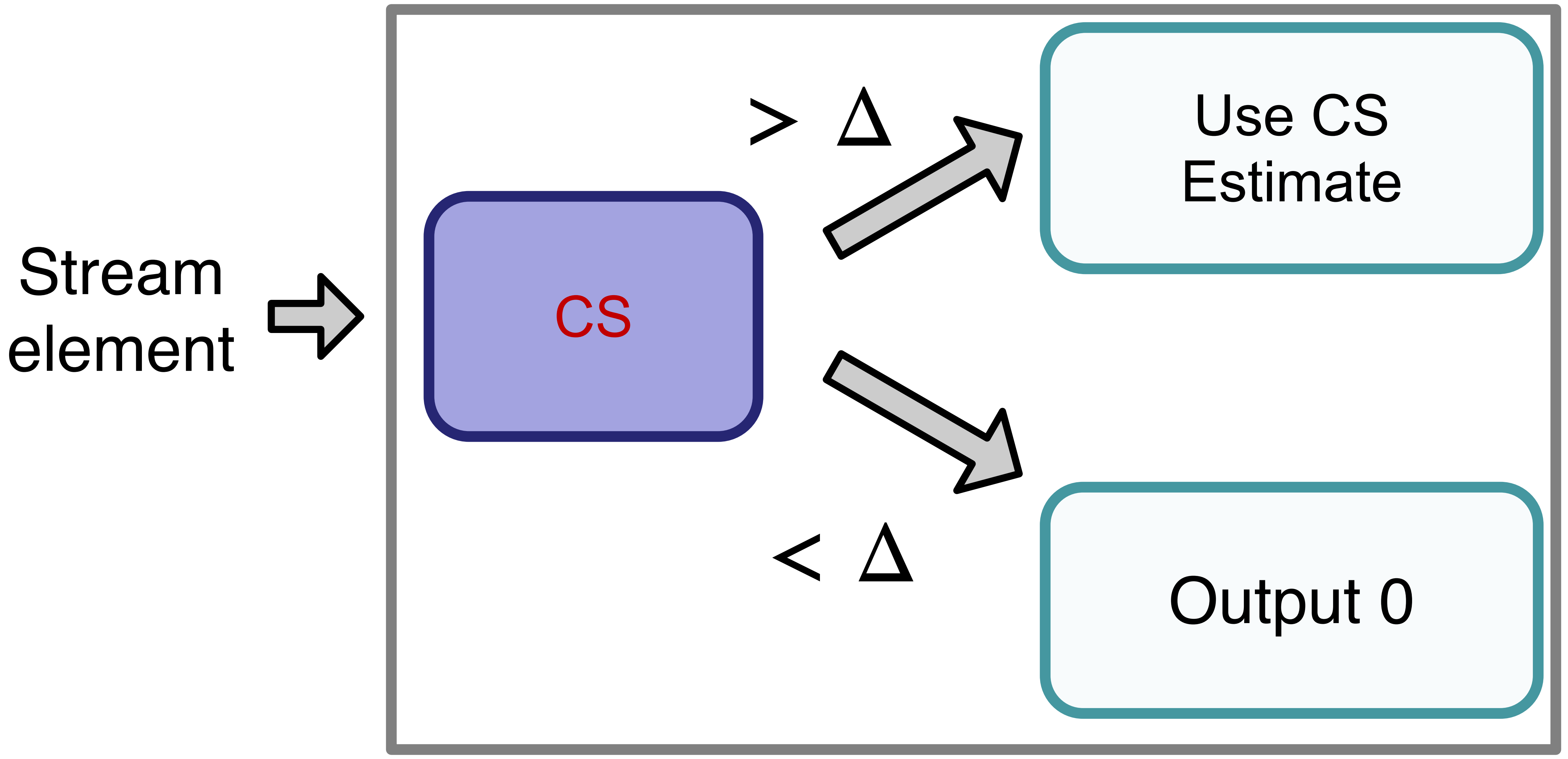
Why would you incur large error for small elements?

Better to predict them as 0!

Use the estimate of CS itself!

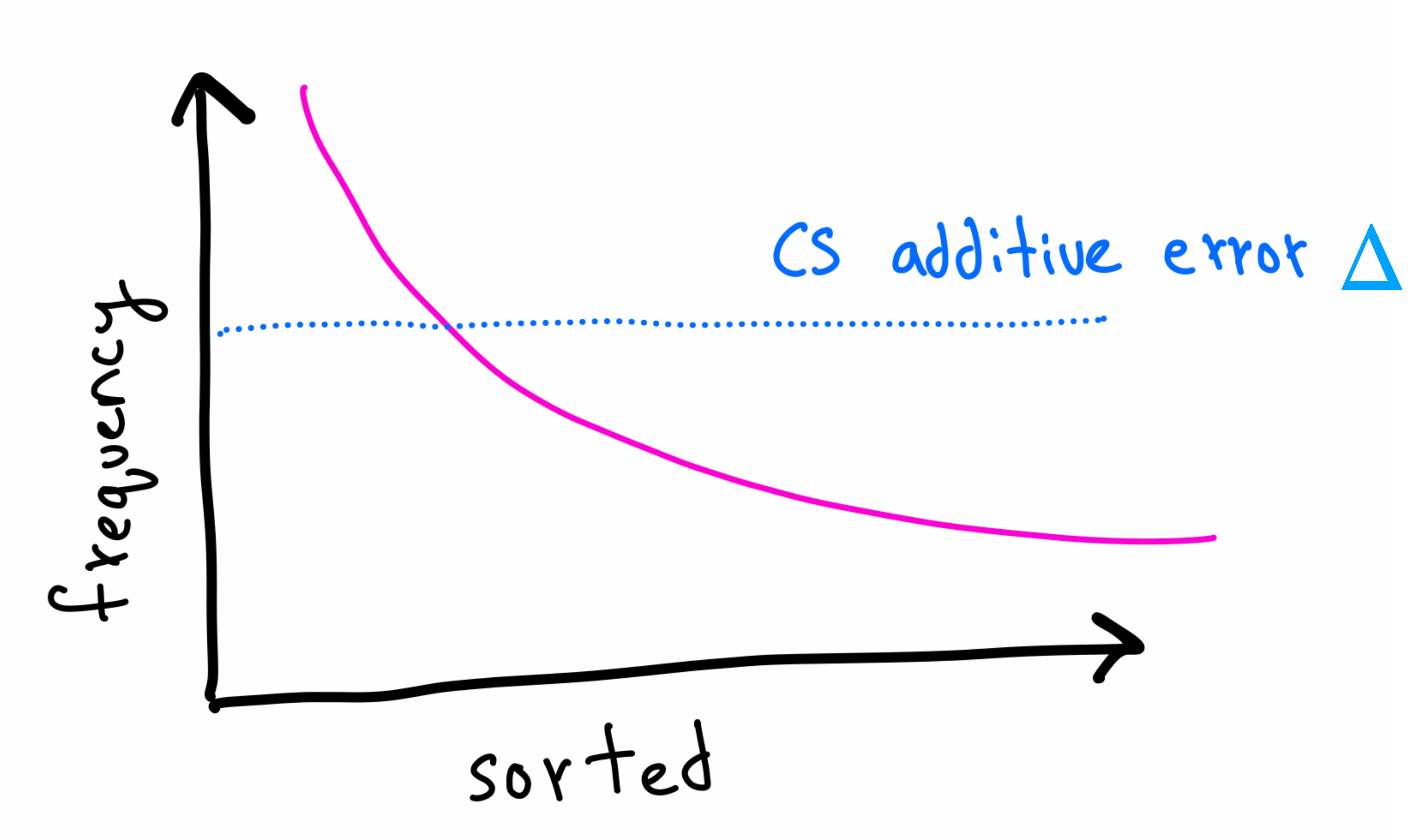
New Algorithm Intuition

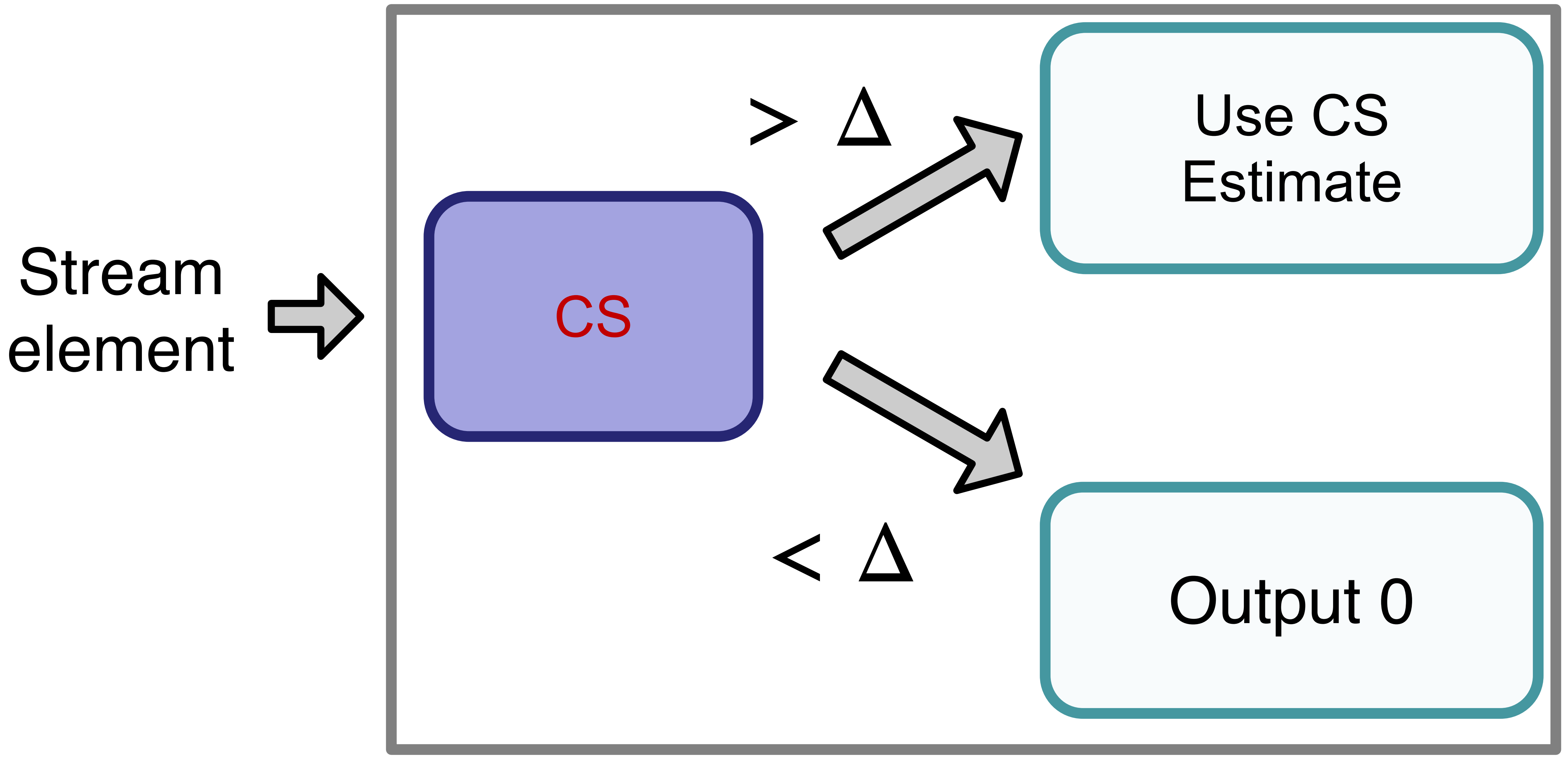




Note: The additive error of CS (Δ) is a known quantity.

New Algorithm Intuition





Note: The additive error of CS (Δ) is a known quantity.

Problem: Can potentially output 0 for large frequencies!

Space = B words, n = # of elements

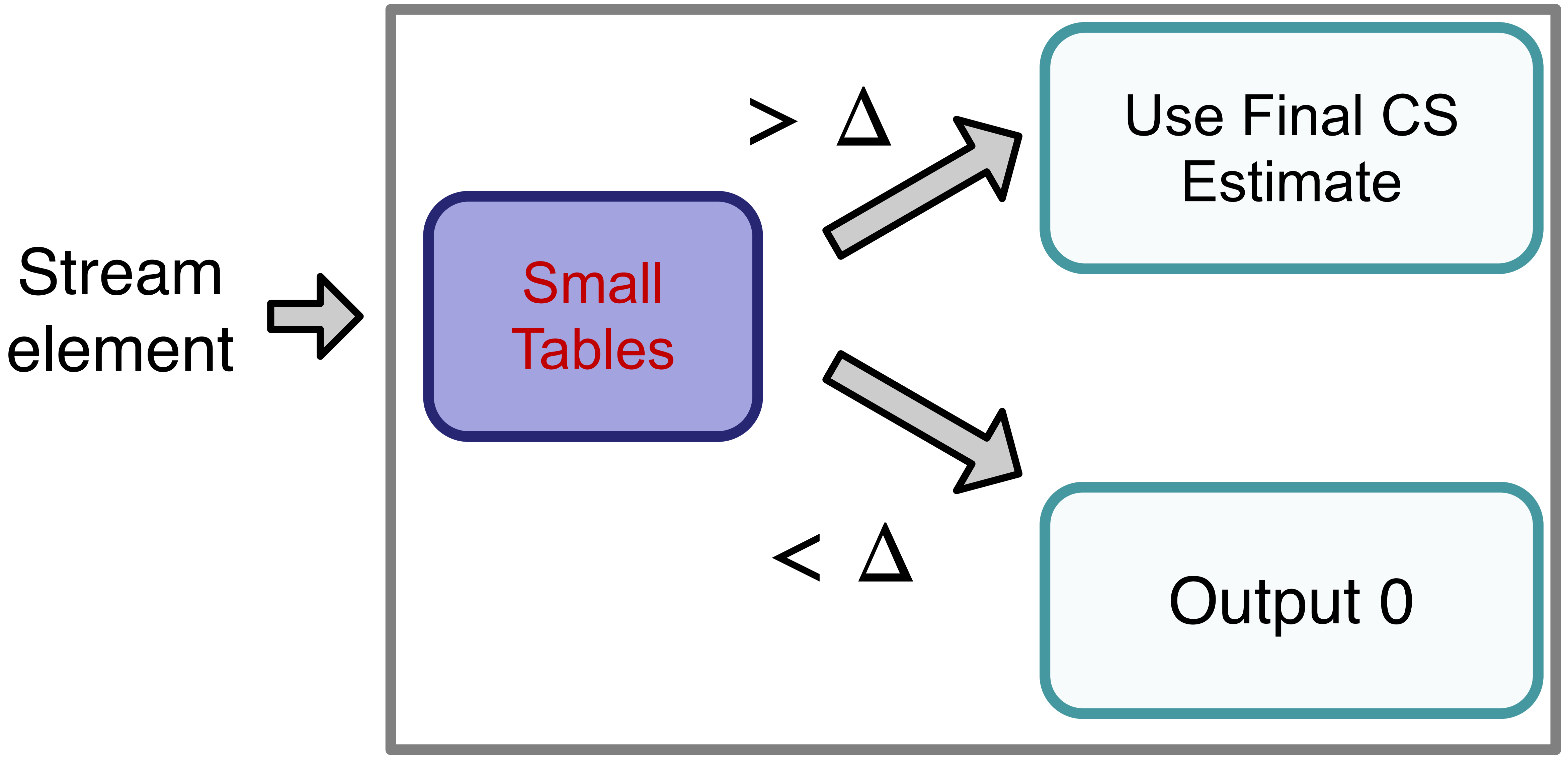
Algorithm 7 Frequency update algorithm

procedure UPDATE
    $T \leftarrow \Theta(\log \log n)$
    for $j = 1$ to $T - 1$ **do**
        $S_j \leftarrow$ CountSketch table with 3 rows and $\frac{B}{6T}$ columns
    end for
    $S_T \leftarrow$ CountSketch table with 3 rows and $\frac{B}{6}$ columns
    Input stream update in all of the T CountSketch tables

end procedure

Many small noisy CS tables

Space = B words, n = # of elements

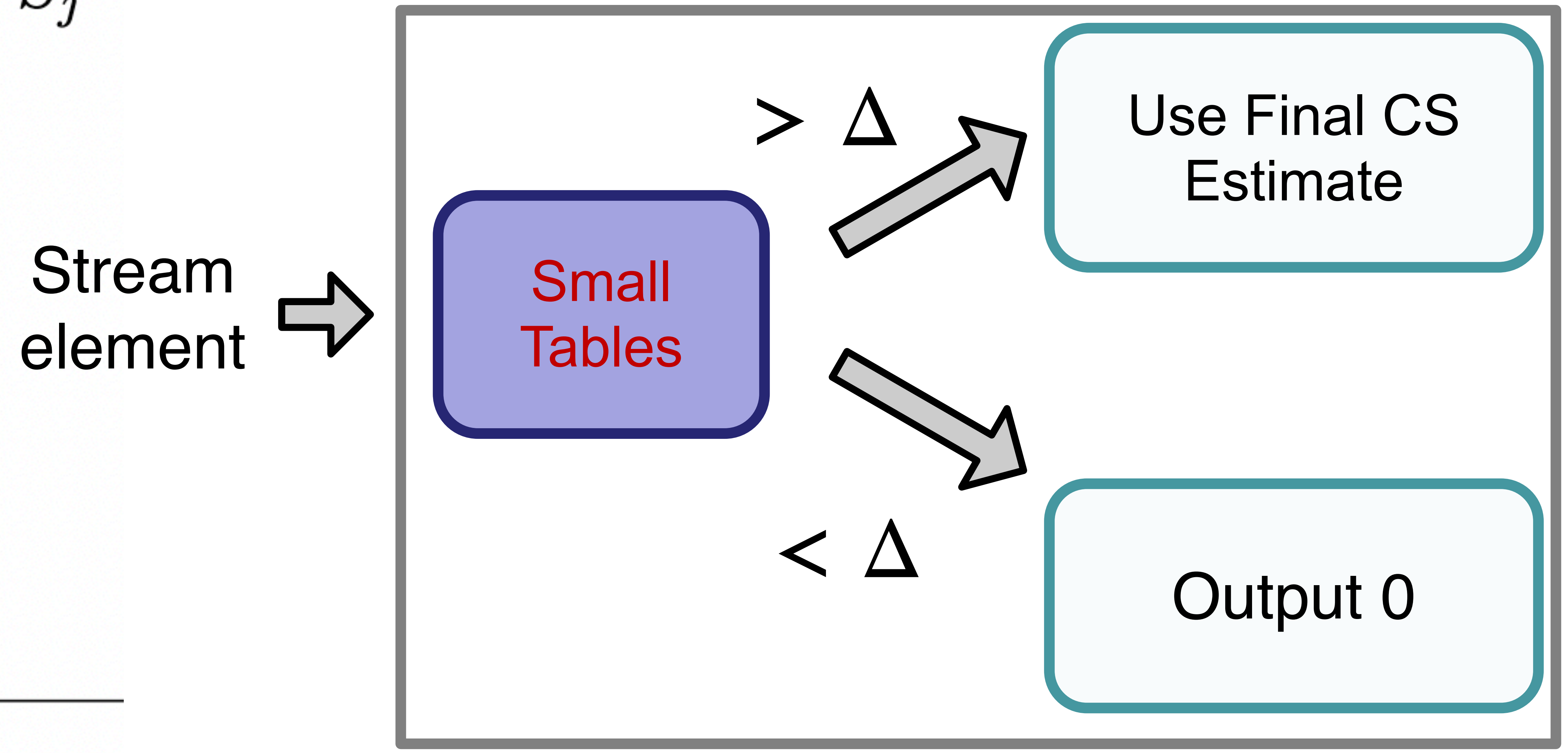


Idea:

Use estimate of smaller noisy tables to determine if we should output 0 or listen to the large CS table

Algorithm 7 Query

procedure QUERY
    for $j = 1$ to $T - 1$ **do**
        $\hat{f}_i^j \leftarrow$ estimate of the ith frequency given by table S_j
    end for
    $\tilde{f}_i \leftarrow \text{Median}(\hat{f}_i^1, \ldots, \hat{f}_i^{T-1})$
    if $\tilde{f}_i <$ 'Appropriate Threshold' **then**
        Return 0
    else
        Return $\hat{f}_i^T$, the estimate given by table S_T
    end if
end procedure



Idea:

Use estimate of smaller noisy tables to determine if we should output 0 or listen to the large CS table

Space = B words, n = # of elements

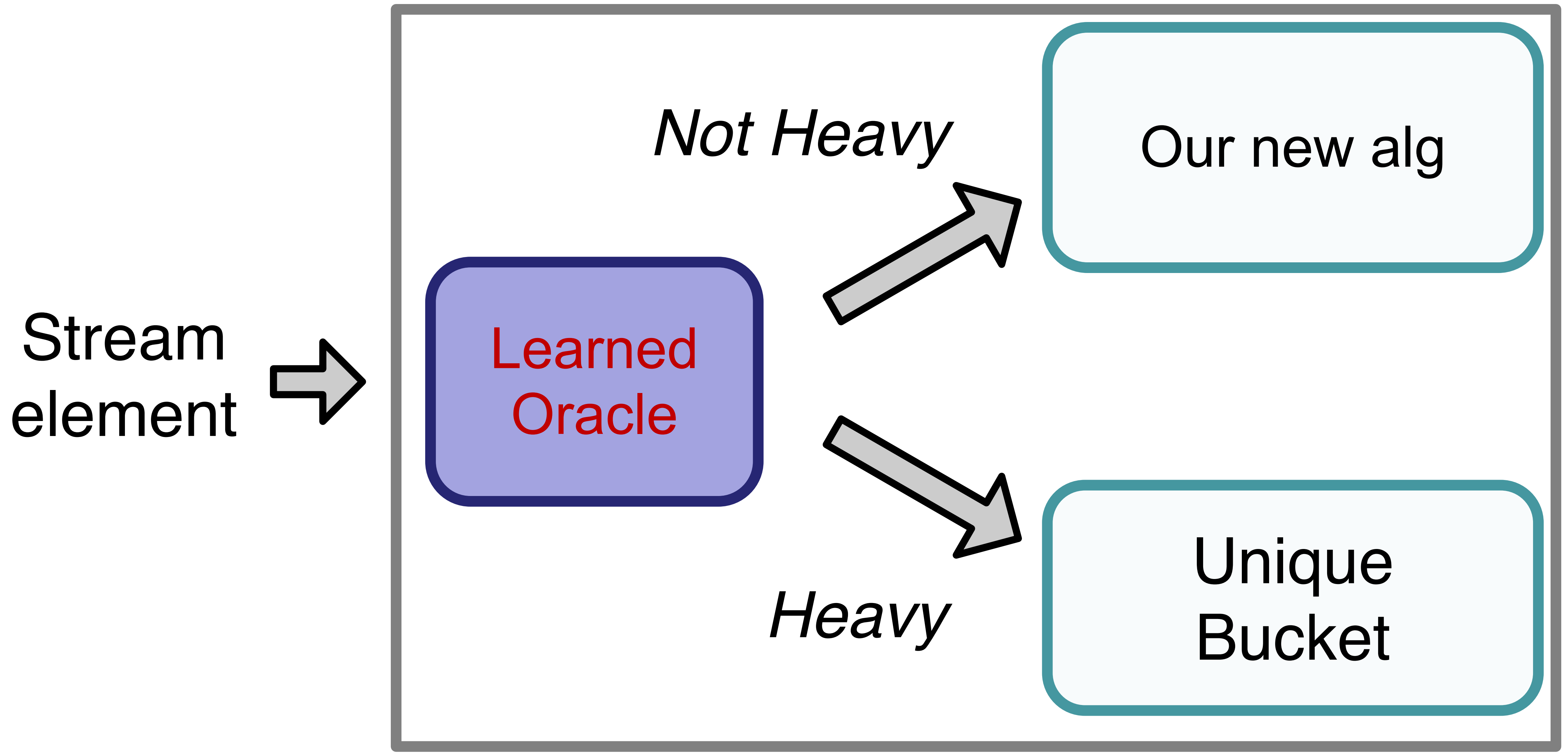
Algorithm 7 Frequency update algorithm

procedure UPDATE
    $T \leftarrow \Theta(\log \log n)$
    for $j = 1$ to $T - 1$ **do**
        $S_j \leftarrow$ CountSketch table with 3 rows and $\frac{B}{6T}$ columns
    end for
    $S_T \leftarrow$ CountSketch table with 3 rows and $\frac{B}{6}$ columns
    Input stream update in all of the T CountSketch tables
end procedure

Algorithm 7 Query

procedure QUERY
    for $j = 1$ to $T - 1$ **do**
        $\hat{f}_i^j \leftarrow$ estimate of the ith frequency given by table S_j
    end for
    $\tilde{f}_i \leftarrow \text{Median}(\hat{f}_i^1, \ldots, \hat{f}_i^{T-1})$
    if $\tilde{f}_i <$ 'Appropriate Threshold' **then**
        Return 0
    else
        Return $\hat{f}_i^T$, the estimate given by table S_T
    end if
end procedure

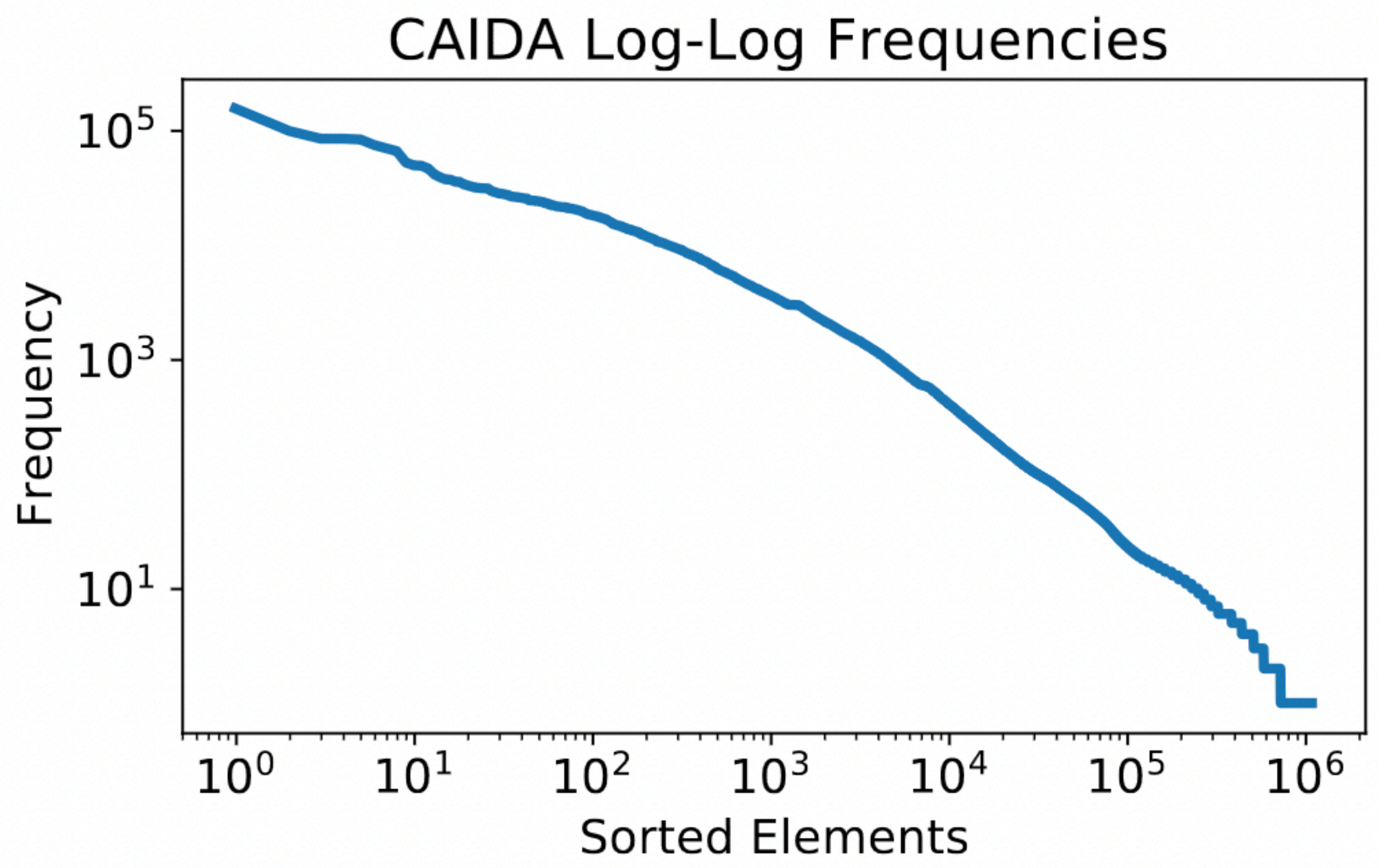
Learning-augmented Version

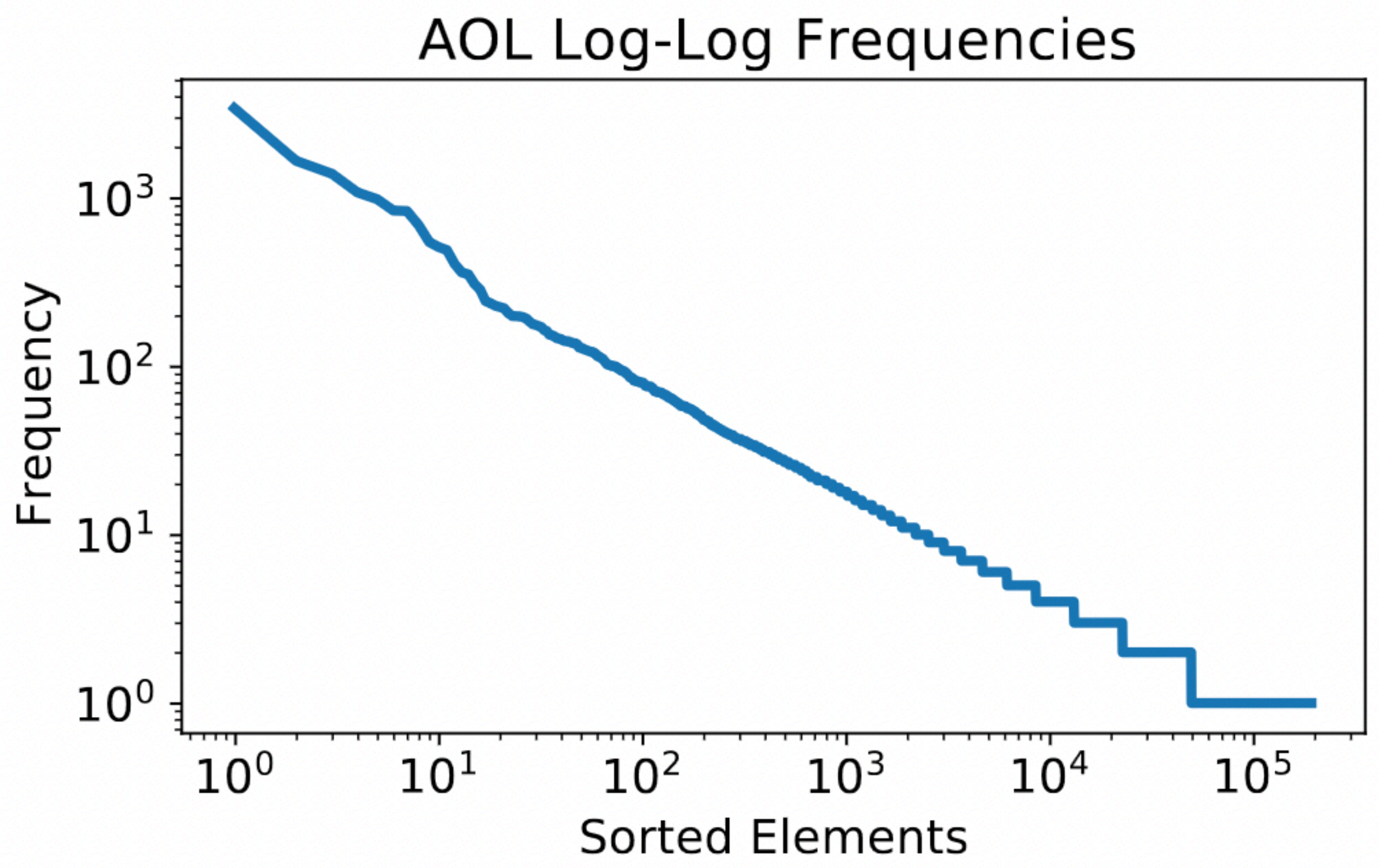


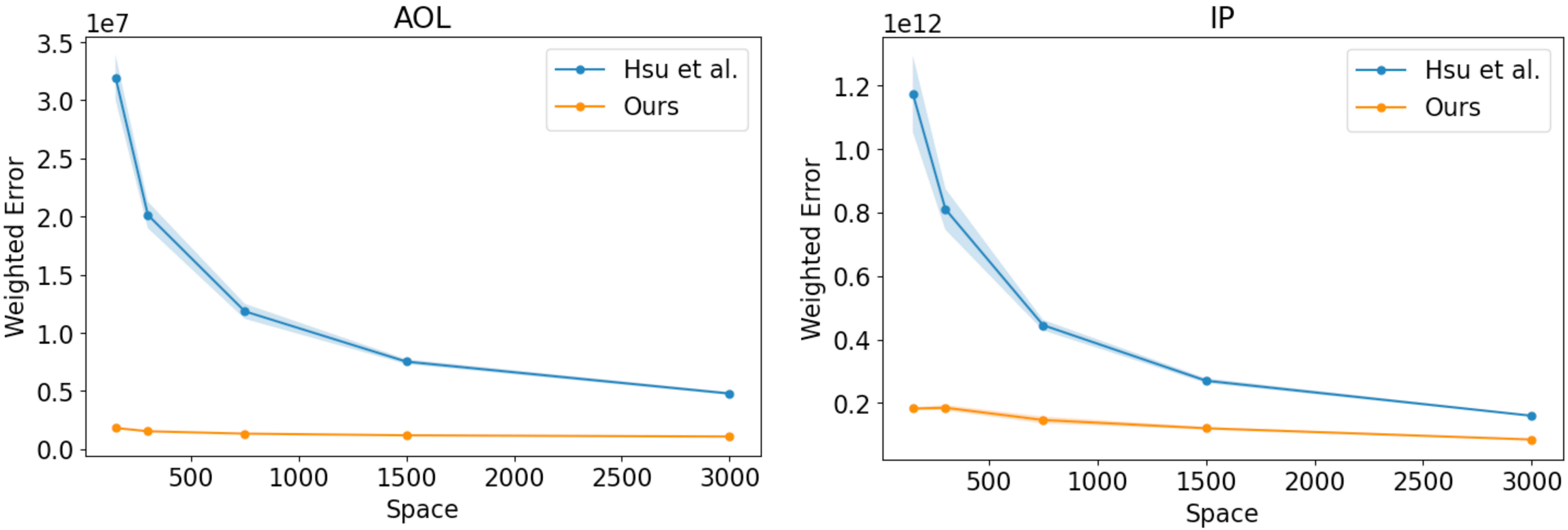
Some Empirical Results

Test on two real-world datasets

- Internet search queries (AOL Dataset)
- Internet Traffic/IP (CAIDA Dataset)
- [Hsu et al.] obtained predictions (predictor trained on past versions of the data)






AOL
1e7
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
Weighted Error
500
1000
1500
2000
2500
3000
Space
Hsu et al.
Ours
IP
1e12
1.2
1.0
0.8
0.6
0.4
0.2
Weighted Error
500
1000
1500
2000
2500
3000
Space
Hsu et al.
Ours

Why it works

Quick stretch time!

Algorithm 7 Frequency update algorithm

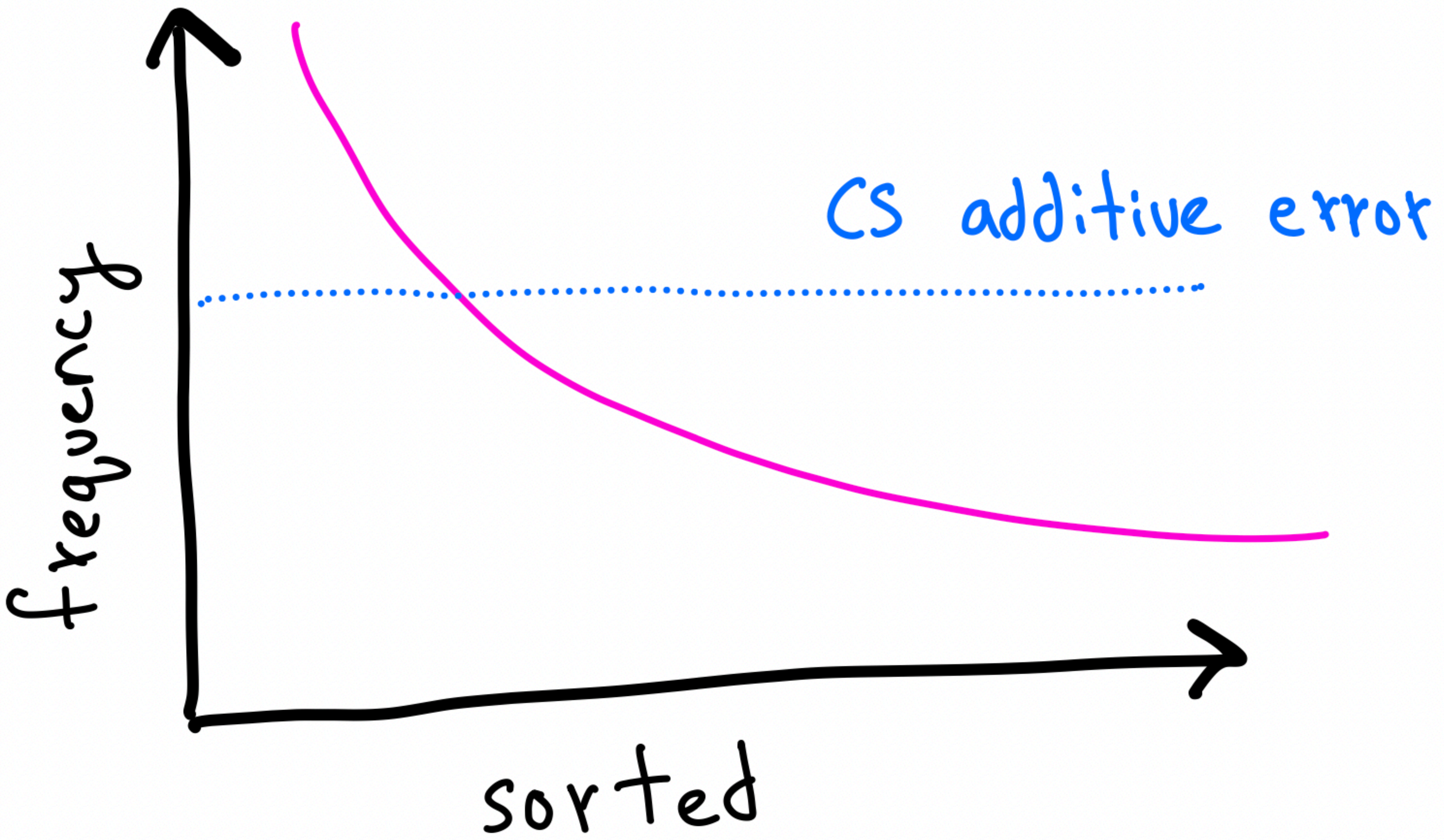
```
procedure UPDATE
    T ← Θ(log log n)
    for j = 1 to T − 1 do
        S_j ← CountSketch table with 3 rows and B/(6T) columns
    end for
    S_T ← CountSketch table with 3 rows and B/6 columns
    Input stream update in all of the T CountSketch tables

end procedure
```

Algorithm 7 Query

```
procedure QUERY
    for j = 1 to T − 1 do
        f̂_i^j ← estimate of the ith frequency given by table S_j
    end for
    f̃_i ← Median(f̂_i^1, ..., f̂_i^{T−1})
    if f̃_i < 'Appropriate Threshold' then
        Return 0
    else
        Return f̂_i^T, the estimate given by table S_T
    end if
end procedure
```

Why it works



Ideal Analysis: Pay CS error for 'large' frequencies, pay f_i error for 'small' frequencies

Why it works

Ideal: Pay CS error for ‘large’ frequencies, pay f_i error for ‘small’ frequencies

But it’s a bit asymmetric.

Why it works

Ideal: Pay CS error for ‘large’ frequencies, pay f_i error for ‘small’ frequencies

But it’s a bit asymmetric.

- Small elements: Even if something goes ‘wrong’, we only pay CS (= bounded) error

- Probability of going wrong ~ 1/log n [O(log log n) small tables for boosting]

Ideal: Pay CS error for 'large' frequencies, pay f_i error for 'small' frequencies

But it's a bit asymmetric.

- Small elements: Even if something goes 'wrong', we only pay CS (= bounded) error
- Probability of going wrong ~ 1/log n [O(log log n) small tables for boosting]

CountSketch (CS)	$\Theta\left(\frac{1}{\log n}\right)$
New Alg	$O\left(\frac{\log B}{(\log n)^2}\right)$

Why it works

Ideal: Pay CS error for ‘large’ frequencies, pay f_i error for ‘small’ frequencies

Large elements: If something goes ‘wrong’, we pay huge error: f_i !

Handling large elements

To output 0 on a large element, all estimates of small tables are *wayyyy off*

Algorithm 7 Query

---

procedure QUERY
    for $j = 1$ to $T - 1$ **do**
        $\hat{f}_i^j \leftarrow$ estimate of the ith frequency given by table S_j
    end for
    $\tilde{f}_i \leftarrow \text{Median}(\hat{f}_i^1, \ldots, \hat{f}_i^{T-1})$
    if $\tilde{f}_i <$ 'Appropriate Threshold' **then**
        Return 0
    else
        Return $\hat{f}_i^T$, the estimate given by table S_T
    end if
end procedure

Handling large elements

To output 0 on a element i, all estimates of small tables are *wayyyy off*

A majority of small estimates need to be off by $O(f_i)$.

Bound “probability deviation is very large” for single small table:

Handling large elements

To output 0 on a element i, all estimates of small tables are *wayyyy off*

A majority of small estimates need to be off by $O(f_i)$.

Bound "probability deviation is very large" for single small table:

- Event 1: An element with frequency > s collides with our element i [few elements due to Zipfian assumption!]

Handling large elements

To output 0 on a element i, all estimates of small tables are *wayyyy off*

A majority of small estimates need to be off by $O(f_i)$.

Probability deviation is very large for single small table:

- Event 1: An element with frequency > s collides with our element i [few elements due to Zipfian assumption!]
- Event 2: "*Many*" elements with frequency < s collide with element i

Handling large elements

To output 0 on a element i, all estimates of small tables are *wayyyy off*

A majority of small estimates need to be off by $O(f_i)$.

- Event 1: An element with frequency > s collides with our element i [few elements due to Zipfian assumption!]

- Event 2: "*Many*" elements with frequency < s collide with element i [Small probability due to many]

Has to be "many" because we know the error is huge

Small probability of colliding on "many" elements!

Why it works

Ideal: Pay CS error for ‘large’ frequencies, pay f_i error for ‘small’ frequencies

But it’s a bit asymmetric.

- Small elements: Even if something goes ‘wrong’, we only pay CS (= bounded) error

- Probability of going wrong ~ 1/log n

- Large elements: If something goes ‘wrong’, we pay huge error: f_i !

- Probability estimate is “O(f_i) off” exactly cancels f_i factor

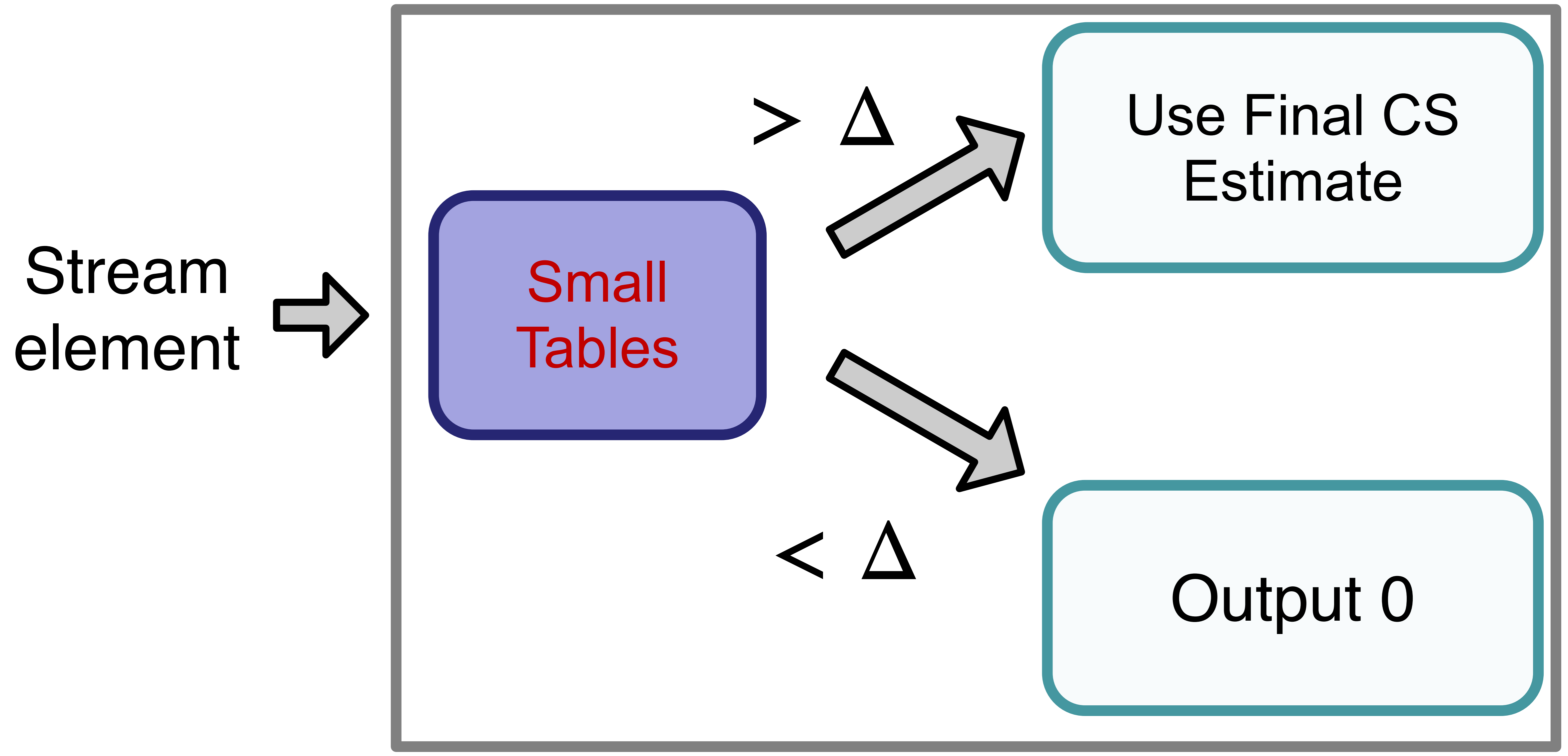
Thank you!

1) Better understanding of frequency estimation for Zipfian data?

2) ‘Better’ ways to use predictions?

3) Other problems? Note: lots of work on learning-augmented algo design:
https://algorithms-with-predictions.github.io/

Beyond Zipfian



Note:

Can obtain good errors without Zipfian assumption by estimating the ‘additive error’ of CountSketch on the fly

Our Results

Space = B words, n = # of elements, Error metric: $\sum_i f_i \cdot |\tilde{f}_i - f_i|$

CountMin (CM)	1
Learned-CM	$\Theta\left(\frac{\log(n/B)}{\log n}\right)^2$
CountSketch (CS)	$\Theta\left(\frac{1}{\log n}\right)$
Learned-CS	$\Theta\left(\frac{\log(n/B)}{(\log n)^2}\right)$

Note: Our paper is a merger with an older manuscript

"(Learned) Frequency Estimation Algorithms under Zipfian Distribution"

Which analyzed the tight behavior of CS/CM and their learned variants (under Zipfian)