Optimal Quantile Approximation in Streams

Zohar Karnin, Kevin Lang, Edo Liberty

Jay Dharmadhikari

The Quantiles Problem

- We receive numbers (or comparable objects) arriving in a stream
- Answer queries about rank of x - how many items in the stream are less than x?
- Equivalent problem: What is the median of the stream? P-th percentile?
- Applications:
  - Is this credit card transaction within this user's normal spending habits?
  - What's the 99th percentile of latency on my network?
  - Is this blood pressure reading an outlier for this patient?

The Quantiles Problem

- Stream consists of n items $x_1 \ldots x_n$
- Data structure to answer $R(x) := |\{x_i : x_i \leq x\}|$
  - Not enough space to compute exactly, instead give an approximation $\tilde{R}(x)$
  - Additive ε-approximation: $\Pr[|\tilde{R}(x) - R(x)| \leq \varepsilon n] \geq 1 - \delta$
  - **Goal: Low space complexity**

Related Work

- Desirable quality: **mergeable** (parallel sketches can be merged)
- "Single quantiles" problem: approximation holds for one rank query
- "All quantiles" problem: approximation holds for ALL rank queries

	Single quantile	All quantiles	Randomized	Mergeable
MRL [3]	$(1/\varepsilon)\log^2(\varepsilon n)$	$(1/\varepsilon)\log^2(\varepsilon n)$	No	Yes
GK [5]	$(1/\varepsilon)\log(\varepsilon n)$	$(1/\varepsilon)\log(\varepsilon n)$	No	No
KLL [This paper]	$(1/\varepsilon)\log^2\log(1/\delta)$	$(1/\varepsilon)\log^2\log(1/\delta\varepsilon)$	Yes	Yes
KLL [This paper]	$(1/\varepsilon)\log\log(1/\delta)$	$(1/\varepsilon)\log\log(1/\delta\varepsilon)$	Yes	No

Compactors

stream of k items → Compactor → stream of $k/2$ items

Compaction Algorithm

Wait for compactor to fill up

1, 7, 5, -23, 4, 9

Sort the elements

-23, 1, 4, 5, 7, 9

Color the elements (even and odd)

-23, 1, 4, 5, 7, 9

Output red or blue elements

-23, 4, 7

Compaction Algorithm

Claim: For any x, after one compact operation, $R_{\text{stream}}(x) \approx 2 \cdot R_{\text{compact}}(x)$

Formally: $|R_{\text{stream}}(x) - 2 \cdot R_{\text{compact}}(x)| \leq 1$

$x = 6, R(x) = 4$

-23, 1, 4, 5, 7, 9

$x = 6, 2 \cdot R(x) = 4$

-23, 4, 7

$x = 4, R(x) = 3$

-23, 1, 4, 5, 7, 9

$x = 4, 2 \cdot R(x) = 4$

-23, 4, 7

$x = 4, R(x) = 3$

-23, 1, 4, 5, 7, 9

$x = 4, 2 \cdot R(x) = 2$

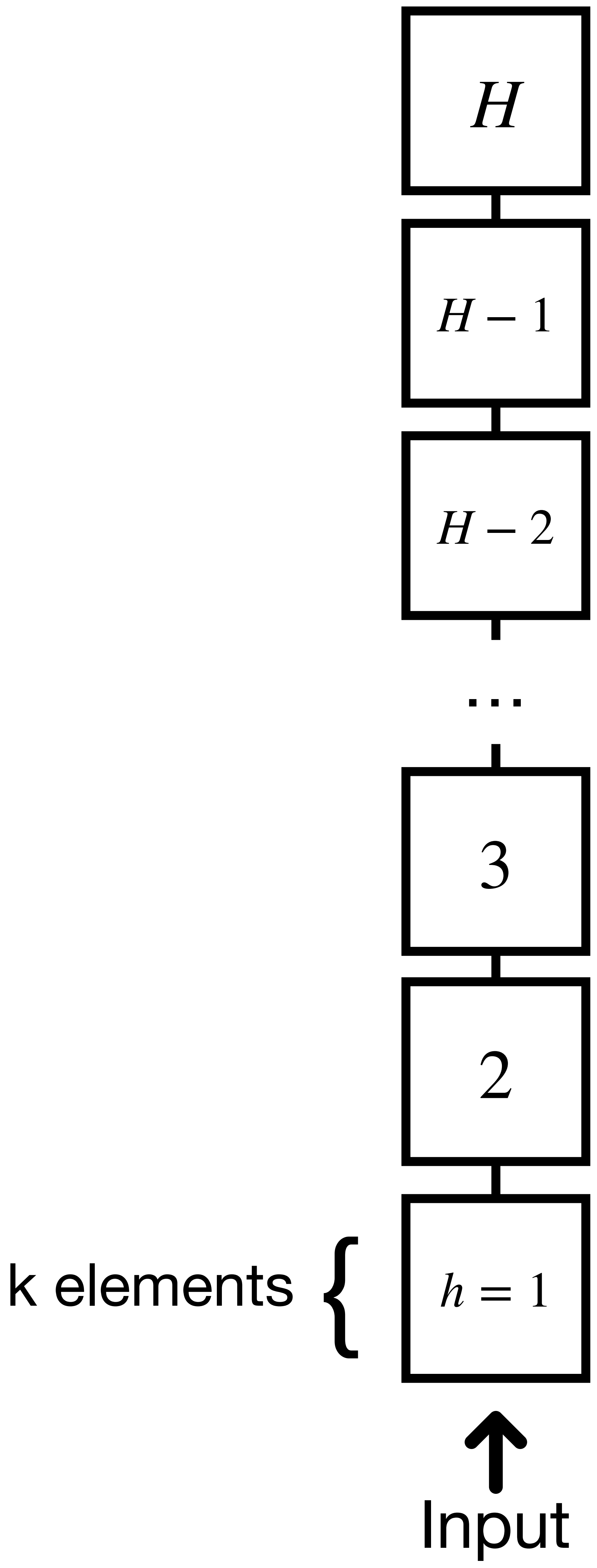
1, 5, 9

Compacting Compactors?

- If we chain two compactors together, we get an output of size $n/4$
- But the error blows up. **Why**?
  - Each element in Compactor 1's output stream "represents" two elements of original stream
  - After Compactor 2, each element in final stream "represents" four elements of original stream
  - If CompactX is the stream after x compactors, $|R_{\text{stream}}(x) - 2 \cdot R_{\text{CompactX}}(x)| \leq 2^x$
- **Fix**: when analyzing multiple compactors chained together, assign each element a weight w
  - Every time an element goes through a compactor, double its weight
  - For every compact operation: $|R_{\text{before}}(x) - 2 \cdot R_{\text{after}}(x)| \leq w$

Naive Compactor Algorithm

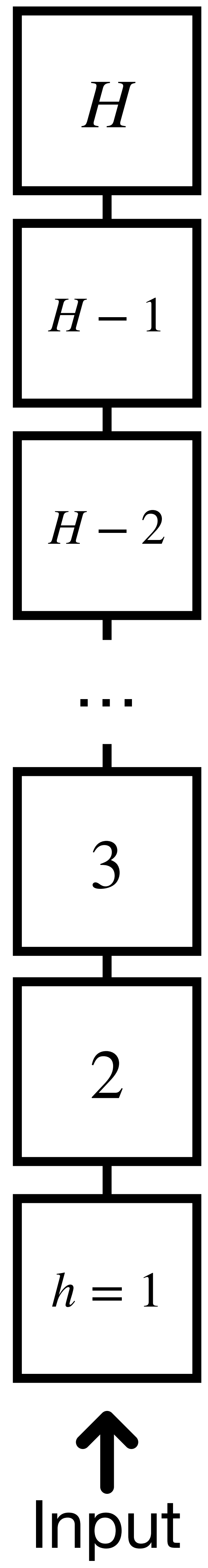
- Each compactor can hold at most k elements (we will pick k later)
- Maintain a chain of $H = \lceil \log(n/k) \rceil$ compactors
- When compactor h is full, it sends $k/2$ elements with doubled weight to compactor $h+1$ (Observe that the weight of elements in compactor h is 2^{h-1})
- How do we answer queries?



Naive Compactor Algorithm

To compute the rank of x:

- For each compactor, calculate how many elements in it are $\leq x$
- Add each element's weight (2^{h-1}) to a running sum
- After all compactors are done, output the sum

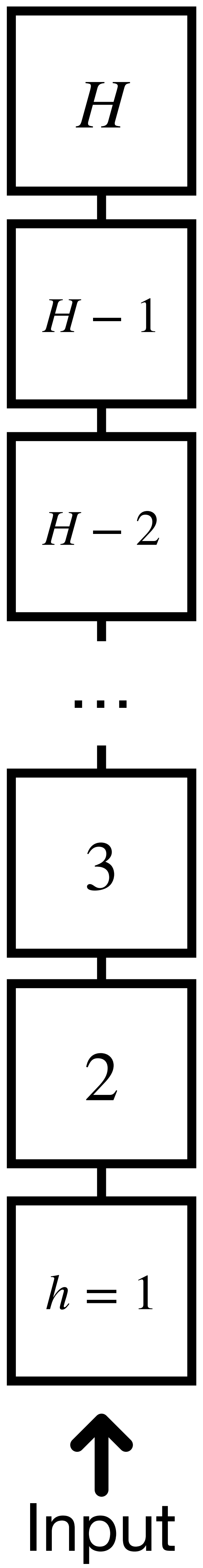


Naive Compactor Algorithm

1. Let $w_h = 2^{h-1}$ be the weight of items that compactor h receives. Then, the number of compactions it performs is at most $m_h = n/kw_h$
2. Remember that the compaction operation of a compactor h can introduce error at most w_h to any rank query. So the total error that compactor h will introduce is $m_h w_h$.
3. The total error on a rank query across all compactors will be

$$\sum_{h=1}^{H} m_h w_h \quad = \sum_{h=1}^{H} n/k \quad = Hn/k \quad \leq n \log(n/k)/k$$

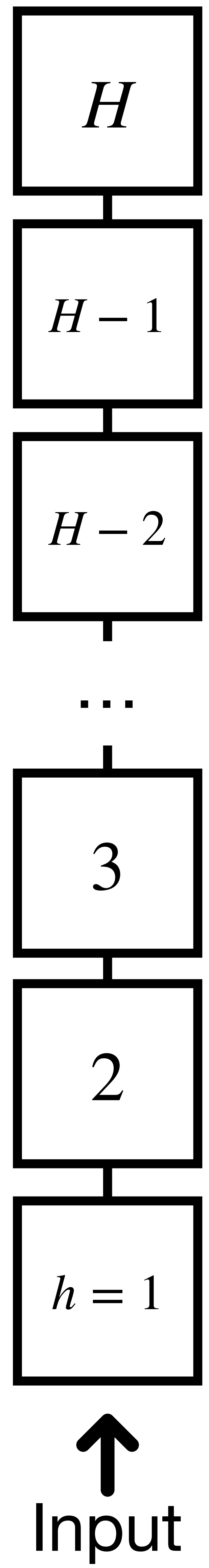
4. Since we have H compactors, the space used will be $kH \leq k \log(n/k)$.
5. Pick $k = O(1/\varepsilon \log(\varepsilon n))$ to get error εn and space $O(1/\varepsilon \log^2(\epsilon n))$



Contribution #1

Idea. The lower levels are too accurate - we can make them smaller to save space without sacrificing error

- k_h := Size of the buffer at height h
- We will set $k_h \approx k_H (2/3)^{H-h}$ (rounding up to 2)



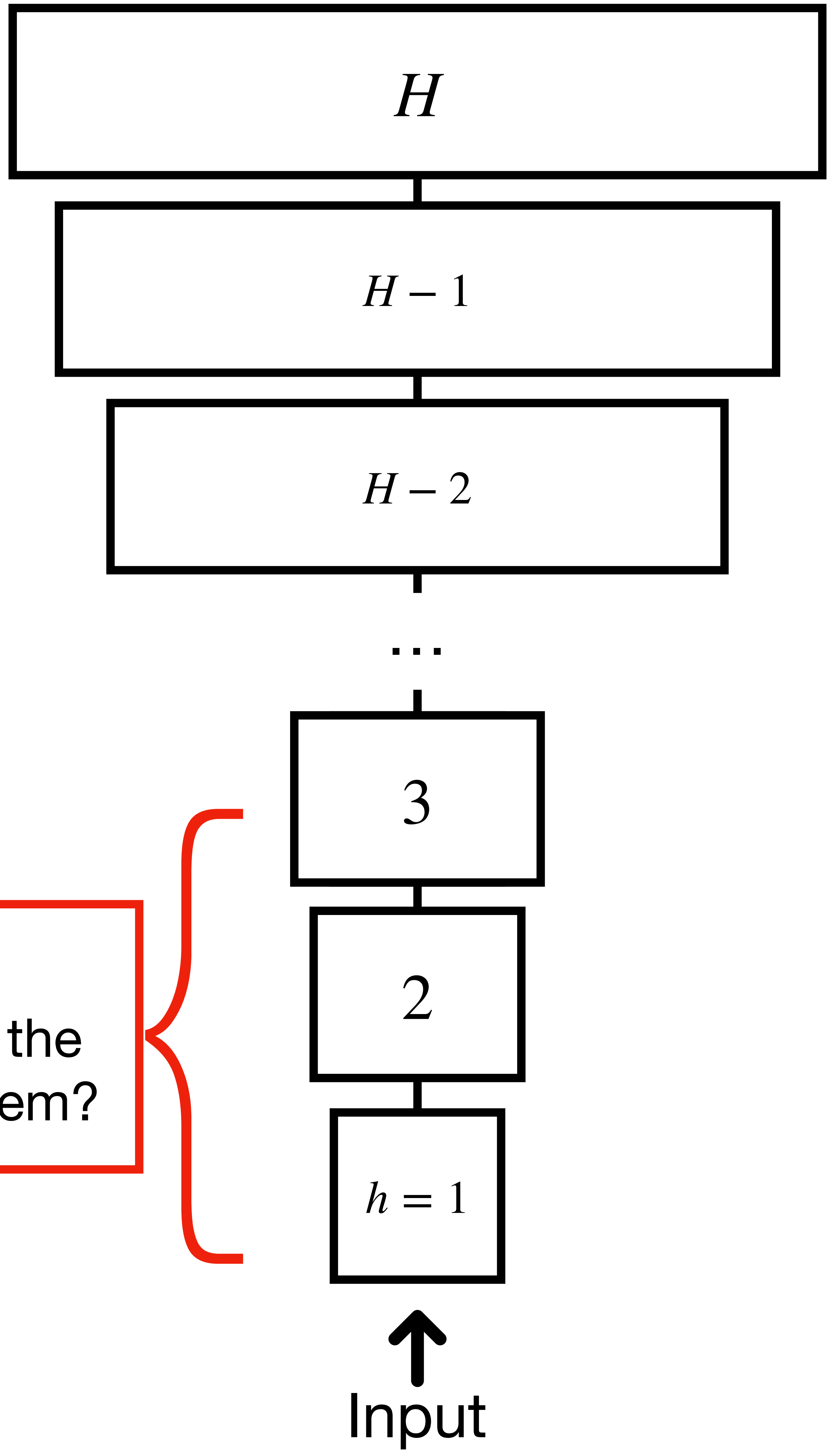
Contribution #1

Idea. The lower levels are <u>too accurate</u> - we can make them smaller to save space without sacrificing error

- k_h := Size of the buffer at height h
- We will set $k_h \approx k_H(2/3)^{H-h}$ (rounding up to 2)

Theorem. For $k = \sqrt{\log(1/\delta)}/\varepsilon$, the new algorithm has error $\leq \varepsilon n$ and space $O(\sqrt{\log(1/\delta)}/\varepsilon + \boxed{\log(\epsilon n)})$

We will have a lot of compactors of size 2 at the bottom… do we need them?



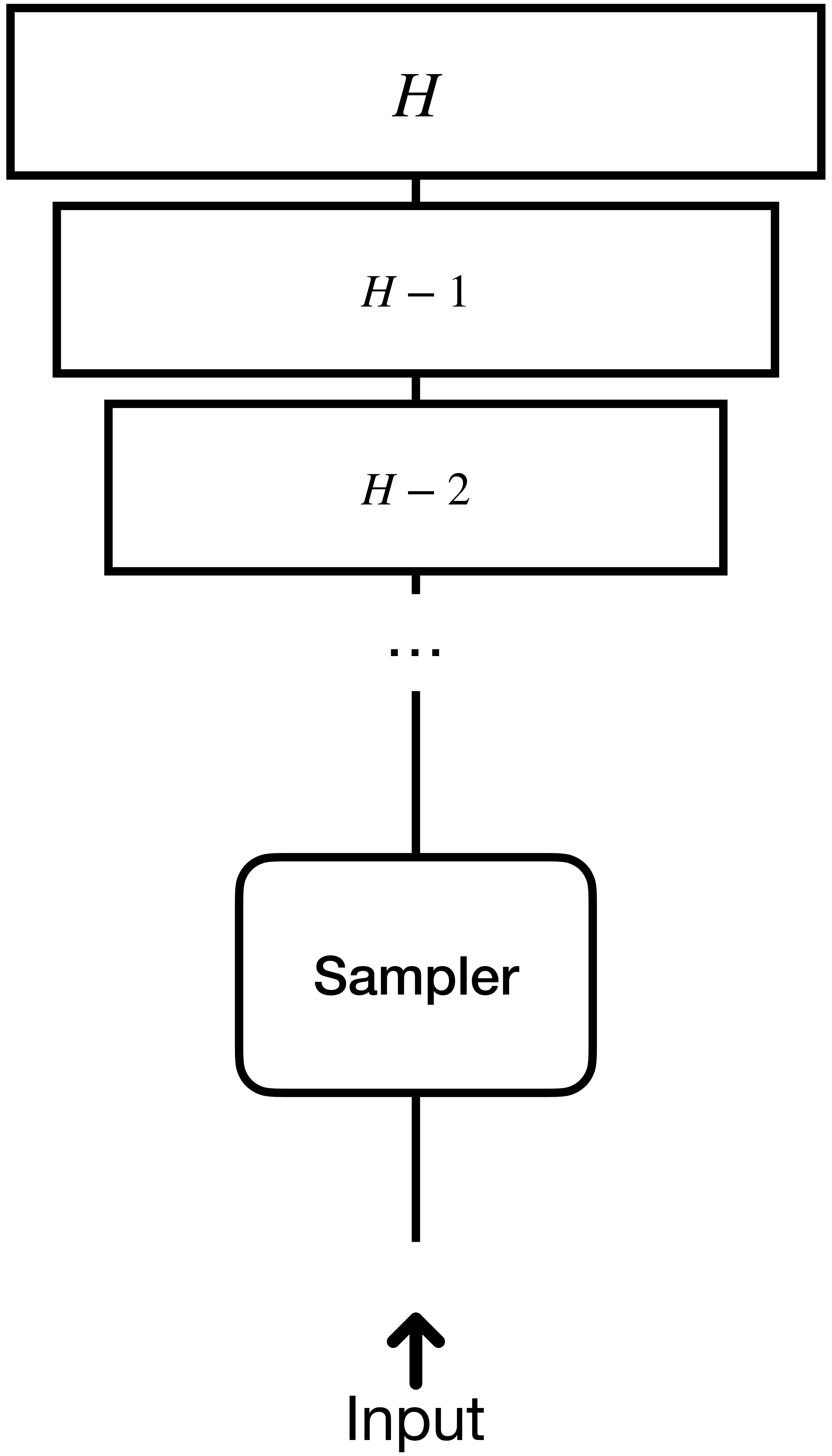
Contribution #1

The bottom H' compactors of size 2 are simulating sampling an item from $2^{H'}$ of the stream. We don't need compactors for this - we can do it with a sampler in $O(1)$ space.

After replacing H' compactors with a sampler, the space taken by the remaining compactors is

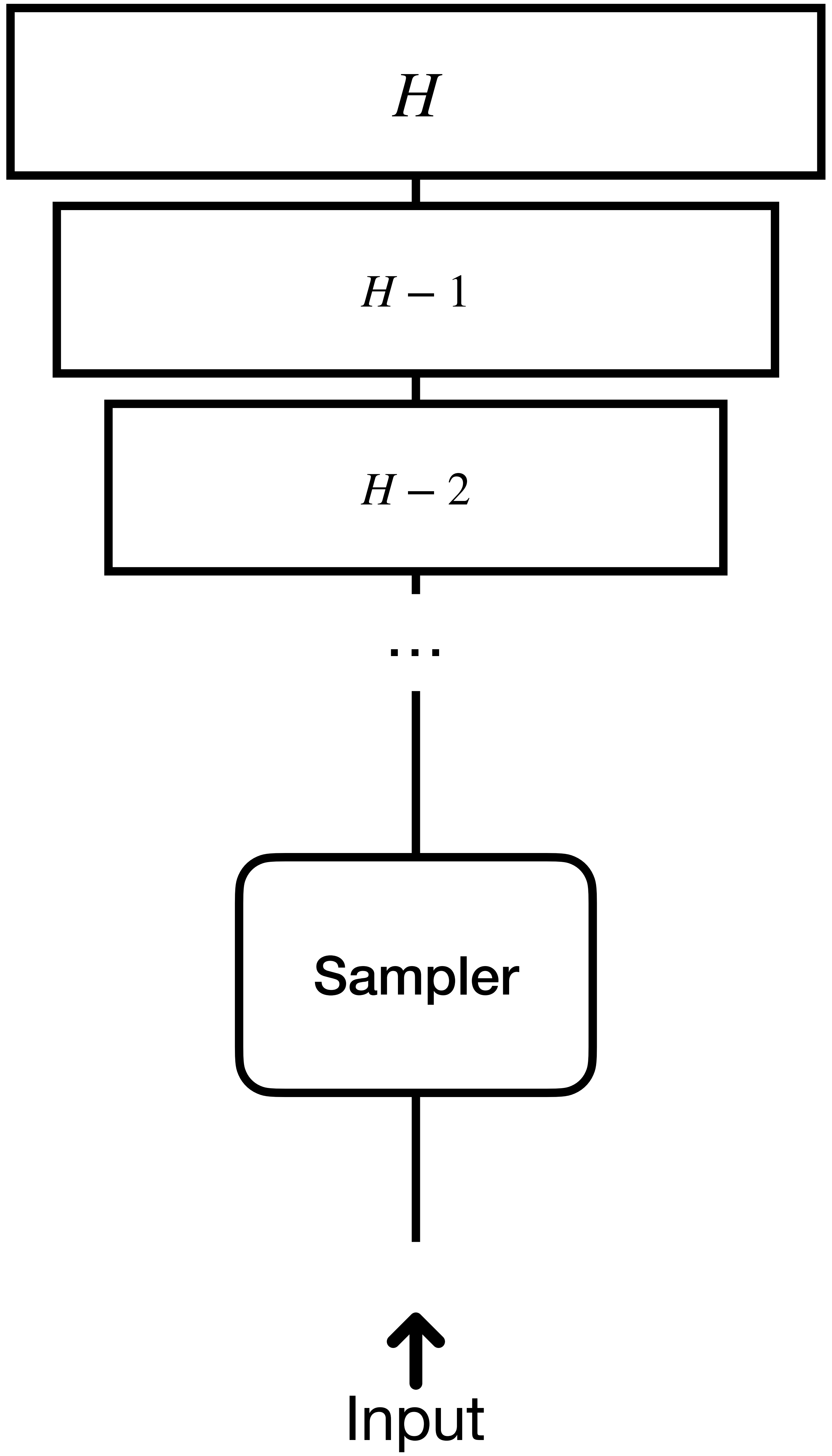
$$\sum_{h=H'+1}^{H} k(2/3)^{H-h} \leq 3k = O(k)$$

With a sampler, the algorithm achieves space $O(\sqrt{\log(1/\delta)}/\varepsilon)$



Contribution #2

Observation. In the top $\log \log 1/\delta$ compactors, the number of compaction operations are so few, worst-case error occurs often.

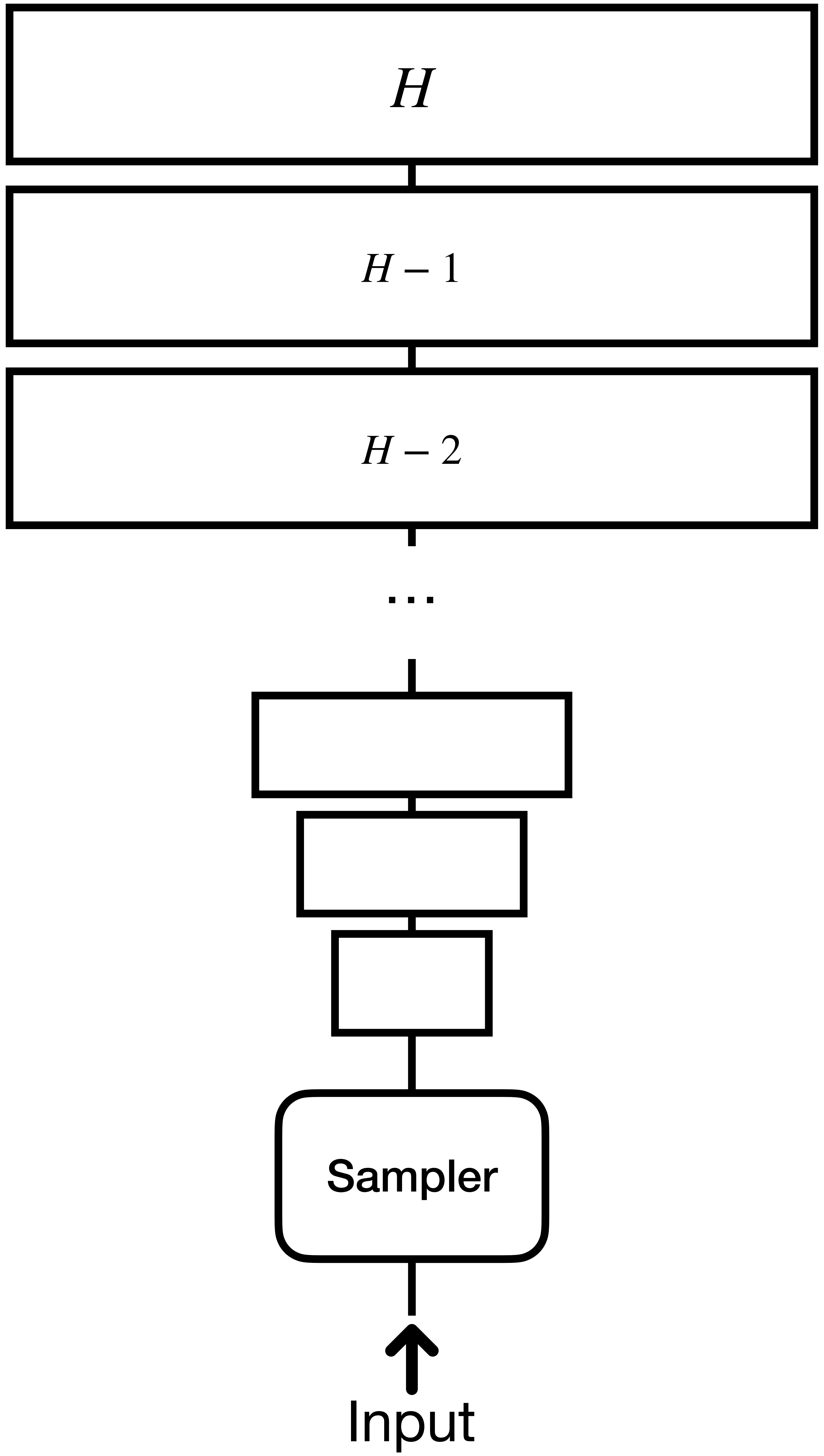


Contribution #2

Observation. In the top $\log\log 1/\delta$ compactors, the number of compaction operations are so few, worst-case error occurs often.

Idea. Fix the size of the top $\log\log 1/\delta$ compactors to k.

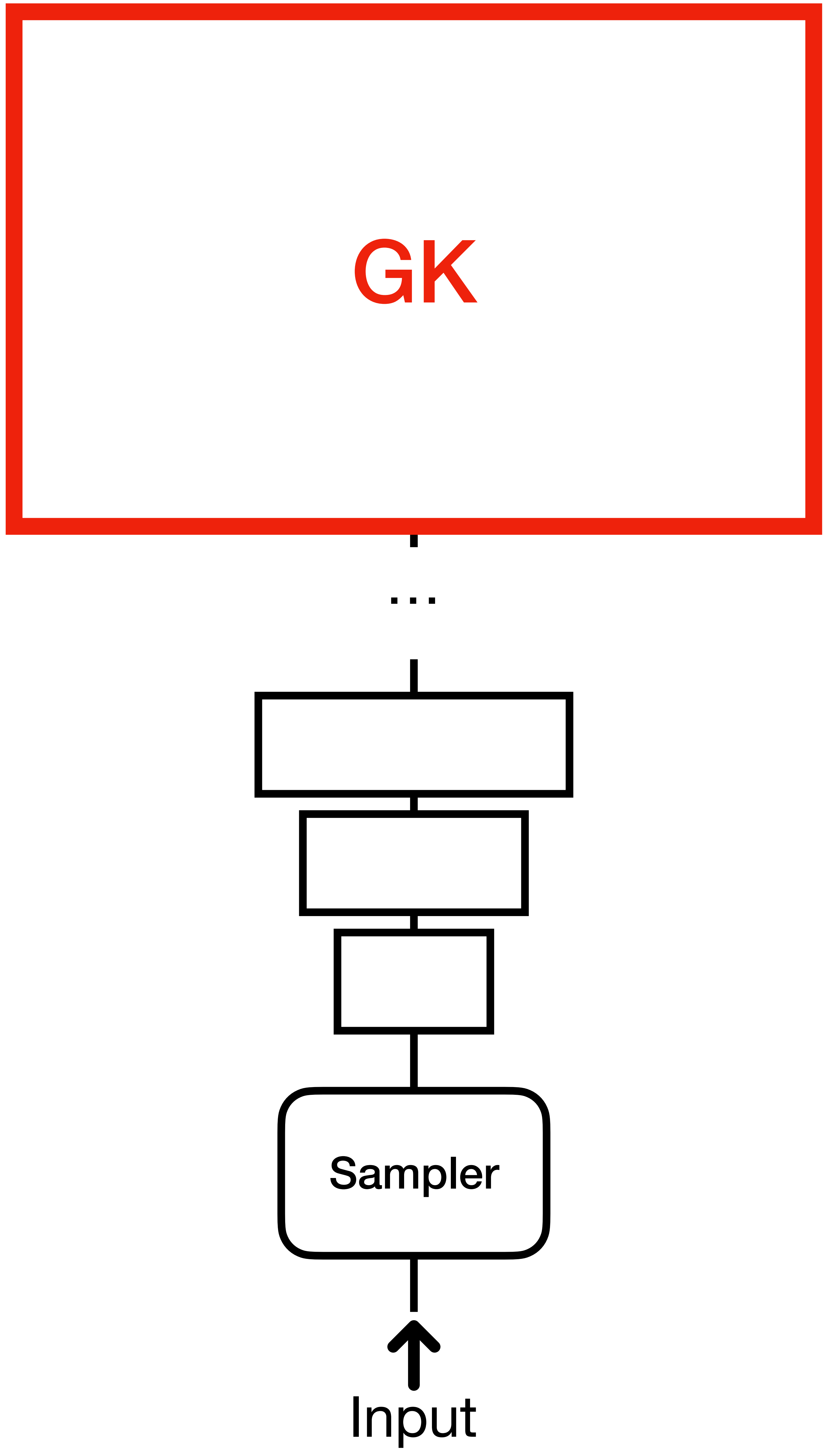
1. We can analyze the error of the top compactors as a mini Naive Compactor Algorithm!
2. The top compactors dominate the space complexity by using $O(k \log\log 1/\delta)$, while the rest contribute $O(k)$.
3. Set $k = \frac{1}{\varepsilon}\log\log 1/\delta$ to get space usage $O((1/\varepsilon)\log^2\log(1/\delta))$ 🙂



Contribution #3

We can improve the space complexity further by replacing the top $\log\log 1/\delta$ compactors with a GK sketch, but we sacrifice merge-ability.

Theorem. There is an algorithm that computes an additive ε-approximation and uses space $O((1/\varepsilon)\log\log(1/\delta))$.



KLL Is Optimal

Theorem. Any algorithm that computes rank within additive error ε with probability $1 - \delta$ must use space $\Omega((1/\varepsilon)\log\log(1/\delta))$.

[Hung, Ting 2010]

References

[1] Lu Wang, Ge Luo, Ke Yi, and Graham Cormode. Quantiles over data streams: An experimental study. In *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, SIGMOD ’13, pages 737–748, New York, NY, USA, 2013. ACM.

[2] Michael B. Greenwald and Sanjeev Khanna. Quantiles and equidepth histograms over streams. In J. Gehrke, M. Garofalakis, and R. Rastogi, editors, *In Data Stream Management: Processing High-Speed Data Streams*. Springer, 2016.

[3] Gurmeet Singh Manku, Sridhar Rajagopalan, and Bruce G. Lindsay. Random sampling techniques for space-efficient online computation of order statistics of large datasets. *SIGMOD Rec.*, 28(2):251–262, June 1999.

[4] J.I. Munro and M.S. Paterson. Selection and sorting with limited storage. *Theoretical Computer Science*, 12(3):315–323, 1980.

[5] Michael Greenwald and Sanjeev Khanna. Space-efficient online computation of quantile summaries. In *Proceedings of the 2001 ACM SIGMOD International Conference on Management of Data*, SIGMOD ’01, pages 58–66, New York, NY, USA, 2001. ACM.

[6] David Felber and Rafail Ostrovsky. A randomized online quantile summary in \(O((1/ε)\log(1/ε))\) words. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2015, August 24–26, 2015, Princeton, NJ, USA*, pages 775–785, 2015.

[7] Pankaj K. Agarwal, Graham Cormode, Zengfeng Huang, Jeff Phillips, Zhewei Wei, and Ke Yi. Mergeable summaries. In *Proceedings of the 31st Symposium on Principles of Database Systems*, PODS ’12, pages 23–34, New York, NY, USA, 2012. ACM.

[8] Nisheeth Shrivastava, Chiranjeeb Buragohain, Divyakant Agrawal, and Subhash Suri. Medians and beyond: New aggregation techniques for sensor networks. In *Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems*, SenSys ’04, pages 239–249, New York, NY, USA, 2004. ACM.

[9] Andrej Brodnik, Alejandro Lopez-Ortiz, Venkatesh Raman, and Alfredo Viola. *Space-Efficient Data Structures, Streams, and Algorithms: Papers in Honor of J. Ian Munro, on the Occasion of His 66th Birthday*, volume 8066. Springer, 2013.

[10] Regant YS Hung and Hingfung F Ting. A \((1/ε)\log(1/ε) \) space lower bound for finding ε-approximate quantiles in a data stream. In *Frontiers in Algorithmics*, pages 89–100. Springer, 2010.