Quantile sketch algorithms

— MATH PROJECT —

Author:

Levente Birszki

Applied Mathematics MSc

Supervisors:

Dr. Gábor Rétvári

Senior Research Fellow

Dr. Balázs Vass

Assistant Lecturer

Eötvös Loránd University
Faculty of Science

Budapest, 2023.

1 Introduction

Within the scope of the project, I dealt with the quantile sketch algorithms in data streams. It is a well-researched area of mathematics that has many practical applications such as big data [1], distributed systems [2] and the area that led me here, network traffic monitoring. Within the latter, quantile sketches are used, for example, catching heavy flows [3], attack detection [4] and even in traffic control [5]. The goal of my report is to present the problem and give an overview of the results so far, asymptotic limits, and practical implementations.

2 The quantile problem

A *sketch* $S(X)$ of some data set X with respect to some function f is a compression of X that allows us to compute, or approximately compute $f(X)$ given access only to $S(X)$. A *streaming* algorithm is processing data streams in which the input is presented as a sequence of items and can be examined only in one pass. Streaming algorithms often produce approximate answers based on a sketch of the data stream.

Given a stream of items $y_1, y_2, \ldots, y_n$ in some arbitrary order, and let $x_1 \leq x_2, \leq \ldots \leq x_n$ the sorted sequence. If its necessary, we can assume, that all the elements are distinct, since instead of y_i we can take (y_i, i) with lexicographical ordering.

Definition 2.1. Given an x element from the input stream. $r(x)$, the rank of x is the number of elements smaller or equal than x in the sorted input.

Definition 2.2. The q-quantile for $q \in [0,1]$ is defined to be the element in position $\lceil qn \rceil$ in the sorted sequence of the input. In other words, the element whose rank is $\lceil qn \rceil$. Denote this element with x_q.

There are different versions of this problem. Sometimes an element x is given, and we need to compute $r(x)$, and sometimes the opposite; given a rank r (or a quantile q) and the task is to return the item from the stream, with rank r (or $\lceil qn \rceil$). But usually if we can answer one question, we can also answer the other.

2.1 Theoretical results

Munro and Paterson [6] showed that $\Omega(n^{1/p})$ space is required to determine the quantile q with p passes. Furthermore, Blum, Floyd, Pratt, Rivest and Tarjan showed, that we need at least $1.5n$ comparisons to compute an exact median of a data set of size n [7]. This paper also shows, that $5.43n$ comparisons is always sufficient for any quantile.

Later Dor and Zwick showed, that the lower bound for the median is $(2+2^{-40})n$, [8], and the upper bound for an arbitrary quantile $2.9423n$ [9].

Typically we only have opportunity to a one-pass algorithm, and with limited space, therefore our main goal is to *approximate* the quantiles.

Definition 2.3. An element $\tilde{x}_q$ is an ε-approximate q quantile if $\lceil (q-\varepsilon)n \rceil \leq r(\tilde{x}_q) \leq \lceil (q+\varepsilon)n \rceil$. In other words $|r(x_q) - r(\tilde{x}_q)| \leq \varepsilon n$. This also known as rank error.

Remark. *There are other possible ways to define the error of an approximation, e.g. relative error, which is defined in the paper in which DDSketch was introduced [10]: $\tilde{x}_q$ is an α-accurate q-quantile if $|\tilde{x}_q - x_q| \leq \alpha x_q$, for a given x_q. Since most algorithms use the rank-error, I also use that in the following.*

In 1974 Yao showed, that computing an approximate median requires $\Omega(n)$ comparisons for any deterministic algorithm. In 2016 Hung and Ting [11] proved, that any comparison-based algorithm for finding ε-approximate quantiles needs $\Omega(\frac{1}{\varepsilon}\log\frac{1}{\varepsilon})$ space.

3 Major milestones

Definition 3.1. In the *single quantile approximation problem*, given an $x_1, \ldots, x_n$ input stream in arbitrary order, q, ε and δ. Construct a streaming algorithm, which computes an ε-approximate q-quantile with probability at least $1 - \delta$.

Definition 3.2. In the *all quantiles approximation problem*, given an $x_1, \ldots, x_n$ input stream in arbitrary order, ε and δ. Construct a streaming algorithm, which computes an ε-approximate q-quantile with probability at least $1 - \delta$ for all q simultaneously.

Definition 3.3. A sketching algorithm is (fully) mergeable, if given two sketches S_1 and S_2 created from inputs X_1 and X_2, a sketch S of $X := X_1 \sqcup X_2$ can be created with no degradation in quality of error or failure probability, and satisfying the same efficiency constraints as S_1, S_2.

Publication	Algorithm	Space Complexity	Notes
1988	MRL [12]	$O(\frac{1}{\varepsilon}\log^2 \varepsilon n)$	non-mergeable, all quantiles deterministic, comparison-based
1988	MRL [12]	$O(\frac{1}{\varepsilon}\log^2 \frac{1}{\varepsilon} + \frac{1}{\varepsilon}\log^2 \log \frac{1}{\delta})$	non-mergeable, all quantiles randomized, comparison-based
2001	GK [13]	$O\left(\frac{1}{\varepsilon}\log(\varepsilon n)\right)$	non-mergeable, all quantiles deterministic, comparison-based
2004	q-digest [14]	$O(\frac{1}{\varepsilon}\log u)$	mergeable, all quantiles deterministic, fixed universe (of size u)
2016	KLL [15]	$O(\frac{1}{\varepsilon}\log^2 \log \frac{1}{\delta})$	mergeable, singe quantile randomized, comparison-based
2016	KLL [15]	$O(\frac{1}{\varepsilon}\log^2 \log \frac{1}{\delta\varepsilon})$	mergeable, all quantiles randomized, comparison-based
2016	KLL [15]	$O(\frac{1}{\varepsilon}\log \log \frac{1}{\delta})$	non-mergeable, singe quantile randomized, comparison-based
2016	KLL [15]	$O(\frac{1}{\varepsilon}\log \log \frac{1}{\delta\varepsilon}))$	non-mergeable, all quantiles randomized, comparison-based
2017	FO [16]	$O(\frac{1}{\varepsilon}\log \frac{1}{\varepsilon})$	non-mergeable, all quantiles randomized, comparison-based
2019	SweepKLL [17]	$O(\frac{1}{\varepsilon}\log \log \frac{1}{\delta\varepsilon}))$	non-mergeable, all quantiles randomized, comparison-based runtime is $O(\log \frac{1}{\varepsilon})$ instead of $O(\frac{1}{\varepsilon})$

Its worth to mention two other sketches; QPipe [18] which is an accelerated version of SweepKLL, and can be fully implemented in the data plane of a programmable switch, and Moment Sketch [19], which has no rank error guarantees, but its widely used in practice.

4 MP-sketch

In 1998 Manku, Rajagopalan and Lindsay gave a solution to all quantile approximation problem [12], based on the work of Munro and Peterson. They gave a uniform framework for three sketching algorithms, including the original MP-sketch. In the followings I'll sum up the framework, based on their paper.

Remark. *Originally MRL was used for databases. If we want to use it for data streams, we need some clever sampling, e.g. reservoir sampling [20].*

In this framework, we have b buffers, each can store k elements. for each buffer X, we associate a positive integer $w(X)$, whits denotes its weight. Intuitively, the weight of a buffer is the number of elements represented by each element in the buffer. There are three operations on a buffer, *New*, *Collapse* and *Quantile*.

4.1 $New(X)$ operation

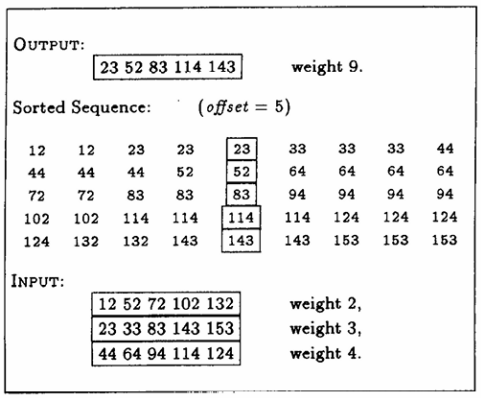
It takes an empty buffer X as an input. The operation simply populates the input buffer with the next k elements from the input stream, and set $w(X) = 1$. If the buffer cannot be filled completely, because there are less than k remaining elements in the input stream, an equal number of $-\infty$ and ∞ elements are added to make up the deficit.

4.2 $Collapse(X_1, X_2, \ldots, X_c)$ operation

It takes $c \geq 2$ full input buffers, $X_1, X_2, \ldots, X_c$ and outputs a buffer Y, which is physically use the same space as X_1. The weight of the output buffer is the sum of the weights of the input buffers, so $w(Y) = \sum_{i=1}^{c} w(X_i)$.

Consider making $w(X_i)$ copies of each element in X_i and sorting all the input buffers together, taking into account the multiple copies. The elements of Y are k equally spaced elements in this (sorted) sequence.

Figure 1: Collapse illustrated.

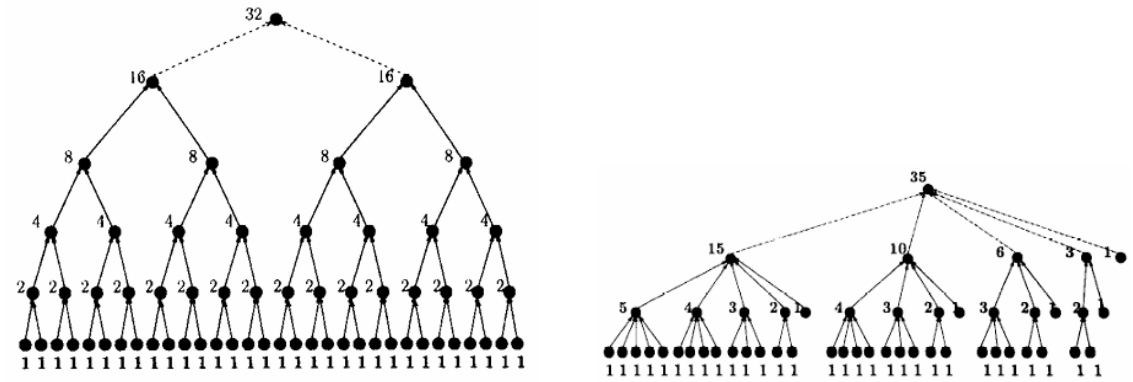


4.3 $Quantile(q)$ operation

This operation is invoked only after the end of the input stream, when all the elements are processed by the data structure, and there is only one full buffer X, as the result of a *Collapse* operation. It returns the $q \cdot k$ element of buffer X.

4.4 Algorithms

An algorithm for computing approximate quantiles consists of a series of invocations of *New* and *Collapse*, and then we can use *Quantile* as many times, as needed. The key difference between algorithms from this family is the collapse policy. *New* populates empty buffers, and *Collapse* reclaims some of them by collapsing a chosen set of full buffers. In figure 2 we can see two different collapsing policies.



(a) MP-sketch for $b = 6$ buffers.

(b) MRL-sketch for $b = 5$ buffers.

Figure 2: Different collapsing policies.

5 Our contribution

Our main goal is to create a sketching algorithm that improves its performance using its own predictions. If we have a sketching algorithm for quantile sketches, we can use it to get an approximation of the CDF of the input stream. We assume that if we knew something about the distribution of the input, we would be able to determine its quantiles more efficiently.

I made an implementation of MP-sketch. In this, if we call *Collapse* on a buffer X with $w(X) = 1$, we need to sort the buffer. After this, all the buffers collapse were called on are sorted. Finally we need to do a slightly modified merge sort.

The idea was, that a sketch could built from scratch, then build a second one, using the first. When the algorithm inserts a new element, ask the first sketch, what is the rank of that element. Then try to insert it into the buffers corresponding position. Thus, we get nearly sorted buffers, and we can use a sorting algorithm that performing well on this kind of input. I this idea works for this sketch, it can be extended to some more complicated algorithms, like KLL.

5.1 Measures

I tested on two different sized inputs. For both input cases sketch parameters were chosen such that it guarantees an error rate of $\varepsilon = 0.001$. In the first case, $n = 10^5, b = 6$ and $k = 3125$, while in the second case, $n = 10^9, b = 17$ and $k = 15259$. The proportion of runtime attributed to each operation is depicted in figure 3. The results depicted in the figure suggests that expediting the sorting process may be worthwhile, even at the expense of slowing down the insertion process.

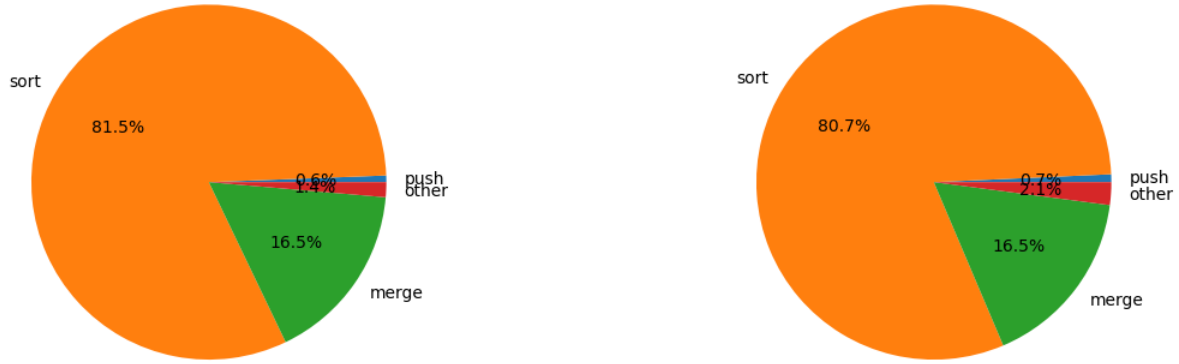


Figure 3: Operation proportions to the runtime in the original sketch. $n = 10^5$ left, $n = 10^9$ right.

In figure 4 we can see the proportions to the runtime of each operation, in case of biased sketches. This is already resembles, what we wanted to achive. In figure 5 we observe how the total runtime evolves in the sketches constructed for the first and second scenarios. Unfortunately, we didn't achieve any improvement with either parameterization; in fact, significant degradation is observed with excessively large buffers.

The cause of this is that there are too many outlier values in the "nearly sorted" buffer, so the used insertion sort cannot be faster than quicksort.

6 Future plans

Instead of sorting the buffers after the insertions, we could make a sorted list initially. For this, we need a data structure, on which we can index elements, but at the same time, sorting parts is not too expensive. This data structure could be the skiplist.

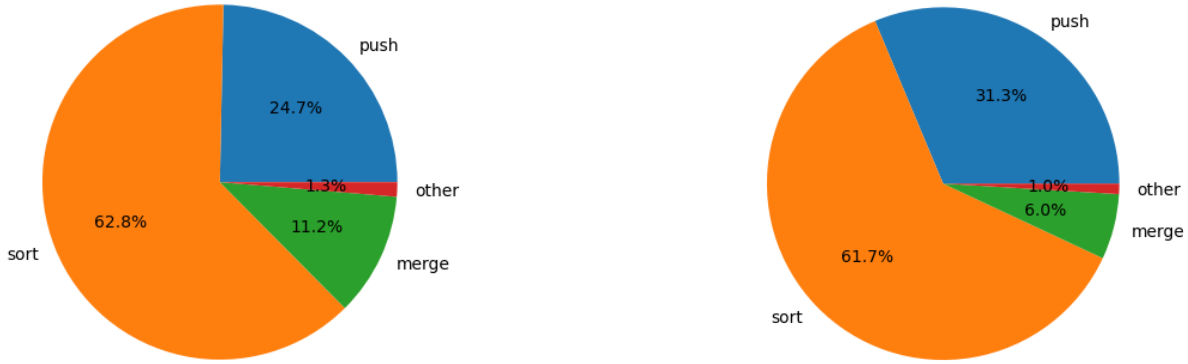


Figure 4: Operation proportions to the runtime in the biased sketch. $n = 10^5$ left, $n = 10^9$ right.

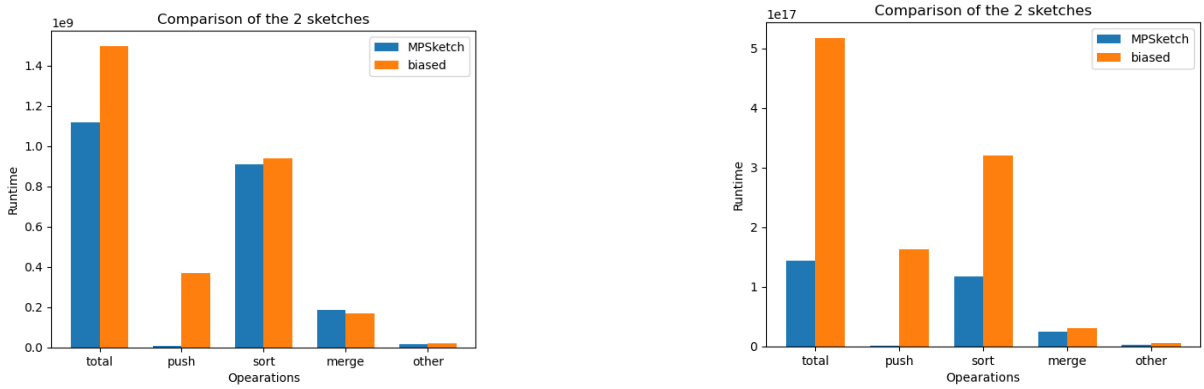


Figure 5: Total runtimes of operations. $n = 10^5$ left, $n = 10^9$ right.

Furthermore, in practice often we don't need to support querying all quantiles, or one single quantile, but only a few specific ones. This is the targeted quantile problem [21], and it would be worthwhile to explore the literature further on this area.

References

1. Chen, T. & Guestrin, C. *XGBoost: A Scalable Tree Boosting System* in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (ACM, Aug. 2016). http://dx.doi.org/10.1145/2939672.2939785.
2. DeWitt, D. J., Naughton, J. F. & Schneider, D. A. *Parallel Sorting on a Shared-Nothing Architecture Using Probabilistic Splitting* in (IEEE Computer Society Press, Miami, Florida, USA, 1991), 280–291. ISBN: 0818622954.
3. Cormode, G. & Hadjieleftheriou, M. Methods for finding frequent items in data streams. *The VLDB Journal* **19,** 3–20. ISSN: 0949-877X. https://doi.org/10.1007/s00778-009-0172-z (Feb. 2010).
4. Kompella, R. R., Singh, S. & Varghese, G. *On Scalable Attack Detection in the Network* in *Proceedings of the 4th ACM SIGCOMM Conference on Internet Measurement* (Association for Computing Machinery, Taormina, Sicily, Italy, 2004), 187–200. ISBN: 1581138210. https://doi.org/10.1145/1028788.1028812.
5. Vass, B., Sarkadi, C. & Rétvári, G. *Programmable Packet Scheduling With SP-PIFO: Theory, Algorithms and Evaluation* in *IEEE INFOCOM 2022 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)* (2022), 1–6.
6. Chan, T. M., Munro, J. I. & Raman, V. Selection and Sorting in the "Restore" Model. *ACM Trans. Algorithms* **14.** ISSN: 1549-6325. https://doi.org/10.1145/3168005 (Apr. 2018).
7. Blum, M., Floyd, R. W., Pratt, V. R., Rivest, R. L. & Tarjan, R. E. Time Bounds for Selection. *J. Comput. Syst. Sci.* **7,** 448–461. https://api.semanticscholar.org/CorpusID:3162077 (1973).
8. Dor, D. & Zwick, U. Finding the αn-th largest element. *Combinatorica* **16,** 41–58. ISSN: 1439-6912. https://doi.org/10.1007/BF01300126 (Mar. 1996).
9. Dor, D. & Zwick, U. *Median Selection Requires Comparisons* in *Proceedings of the 37th Annual Symposium on Foundations of Computer Science* (IEEE Computer Society, USA, 1996), 125.
10. Masson, C., Rim, J. E. & Lee, H. K. DDSketch: A fast and fully-mergeable quantile sketch with relative-error guarantees. *CoRR* **abs/1908.10693.** arXiv: 1908.10693. http://arxiv.org/abs/1908.10693 (2019).
11. Hung, R. Y. S. & Ting, H. F. *An Omega(1/e Log 1/e) Space Lower Bound for Finding e-Approximate Quantiles in a Data Stream* in *Proceedings of the 4th International Conference on Frontiers in Algorithmics* (Springer-Verlag, Wuhan, China, 2010), 89–100. ISBN: 3642145523.
12. Manku, G. S., Rajagopalan, S. & Lindsay, B. G. Approximate Medians and Other Quantiles in One Pass and with Limited Memory. *SIGMOD Rec.* **27,** 426–435. ISSN: 0163-5808. https://doi.org/10.1145/276305.276342 (June 1998).
13. Greenwald, M. & Khanna, S. Space-Efficient Online Computation of Quantile Summaries. **30,** 58–66. ISSN: 0163-5808. https://doi.org/10.1145/376284.375670 (May 2001).
14. Shrivastava, N., Buragohain, C., Agrawal, D. & Suri, S. Medians and Beyond: New Aggregation Techniques for Sensor Networks. *CoRR* **cs.DC/0408039.** http://arxiv.org/abs/cs.DC/0408039 (2004).
15. Karnin, Z., Lang, K. & Liberty, E. *Optimal Quantile Approximation in Streams* 2016. arXiv: 1603.05346 [cs.DS].
16. Felber, D. & Ostrovsky, R. *A randomized online quantile summary in $O(\frac{1}{\varepsilon}\log\frac{1}{\varepsilon})$ words* 2015. arXiv: 1503.01156 [cs.DS].
17. Ivkin, N., Liberty, E., Lang, K., Karnin, Z. & Braverman, V. *Streaming Quantiles Algorithms with Small Space and Update Time* 2019. arXiv: 1907.00236 [cs.DS].

18. Ivkin, N., Yu, Z., Braverman, V. & Jin, X. *QPipe: Quantiles Sketch Fully in the Data Plane* in *Proceedings of the 15th International Conference on Emerging Networking Experiments And Technologies* (Association for Computing Machinery, Orlando, Florida, 2019), 285–291. ISBN: 9781450369985. https://doi.org/10.1145/3359989.3365433.

19. Gan, E., Ding, J., Tai, K. S., Sharan, V. & Bailis, P. Moment-Based Quantile Sketches for Efficient High Cardinality Aggregation Queries. *Proc. VLDB Endow.* **11,** 1647–1660. ISSN: 2150-8097. https://doi.org/10.14778/3236187.3236212 (July 2018).

20. Li, K.-H. Reservoir-Sampling Algorithms of Time Complexity O(n(1 + Log(N/n))). *ACM Trans. Math. Softw.* **20,** 481–493. ISSN: 0098-3500. https://doi.org/10.1145/198429.198435 (Dec. 1994).

21. Cormode, G., Korn, F., Muthukrishnan, S. & Srivastava, D. *Effective computation of biased quantiles over data streams* in *21st International Conference on Data Engineering (ICDE'05)* (2005), 20–31.