# Counting Inversions in Lists

Anupam Gupta  Francis X. Zane
Lucent Bell Labs
600 Mountain Avenue
Murray Hill NJ 07974
{anupamg,francis}@research.bell-labs.com

**Abstract**

In a recent paper, Ajtai et al. [1] give a streaming algorithm to count the number of inversions in a stream $L \in [m]^n$ using two passes and $O(\epsilon^{-1}\sqrt{n}\log n(\log m + \log n))$ space. Here, we present a simple randomized streaming algorithm for the same problem that uses one pass and $O(\epsilon^{-3}\log^2 n \log m)$ space. Our algorithm is based on estimating quantiles of the items already seen in the stream, and using that to estimate the number of inversions involving each element.

## 1 Preliminaries

Let $L$ be the list of elements appearing as a stream with the $i$'th element being denoted by $L[i]$. The quantity we want to approximate is $K(L)$, the number of inversions in $L$; this is the number of pairs $i < j$ such that $L[i] > L[j]$. In order to simplify our notation, we restate this in an equivalent form, that of counting *non-inversions* in the list when the total order, and thus the results of all strict comparisons, has been reversed. Thus, all comparisons between list elements in what follows use this reversed total order. Formally, let $a(j) = |\{i < j| \; L[i] < L[j]\}|$, the number of non-inversions involving element $L[j]$. Clearly $K(L)$, the number of inversions in the original ordering, is equal to $\sum_j a_j$.

The length of the list $L$ is $n$; it suffices for our purposes that $n$ be an upper bound for the list length. The elements in $L$ are drawn from an alphabet $\Sigma = \{1, 2, \ldots, m\}$ of $m$ elements. The subsequence $L[1..i]$ is denoted by $L_i$.

The $\phi$ quantile of a set of $s$ elements, for $\phi \in [0, 1]$, is the element at the $\lceil \phi s \rceil$ position in the sorted order of the set. Hence $\phi = 1$ denotes the maximum element, and $0 < \phi \leq 1/s$ denotes the minimum element; we abuse notation and define the 0-quantile to the minimum as well.

For $\phi \leq \frac{1}{2}$, an $\epsilon$-approximate $\phi$ quantile of $s$ elements is an element that lies in a position between $\lceil s\phi(1-\epsilon)\rceil$ and $\lceil s\phi(1+\epsilon)\rceil$. The reader familiar with the literature will notice that this is a stronger guarantee than usual. Most previous papers defined this to be an error margin of $\pm\epsilon s$ instead of $\pm\epsilon\phi s$ [3, 4, 2]; however, [4, Sec. 7] looks at some issues we face here.

In the rest of the paper, $\alpha = (1+\epsilon)$, where $\epsilon$ is the allowed error. We will make the reasonable assumption in this paper that $\epsilon \leq 1/2$. We have not made an attempt to optimize constants.

## 2 The Algorithm

Suppose that for all $t = 2^{-0}, 2^{-1}, 2^{-2}, \ldots$, we knew the $t$ quantiles $Q_j(t)$ of $L_{j-1}$ exactly. We claim that we can use this to approximate $a(j)$ to within a factor of 2.

First, observe that $Q_j(1) \geq Q_j(\frac{1}{2}) \geq Q_j(\frac{1}{4}) \ldots \geq Q_j(0)$. Given a list element $L[j]$, if $L[j] < Q_j(0)$, then it is smaller than all elements before it and $a(j) = 0$; if $L[j] \geq Q_j(1)$, we have $a(j) = (j-1)$. Otherwise, let $k$ be such that $Q_j(2^{-(k+1)}) < L[j] \leq Q_j(2^{-k})$. Now we know that at least $(j-1)\,2^{-(k+1)}$ elements in the list before $L[j]$ are smaller than it, and less than $(j-1)\,2^{-k}$ are; this allows us to pin $a(j)$ down to within a factor of 2. Now, if we replace $2^{-k}$ by $\alpha^{-k}$ in the above analysis, where $\alpha = (1+\epsilon)$, then we can approximate $a(j)$ to within $\alpha$.

Of course, we have not specified how to find the quantiles using small space in a single pass; and indeed, Munro and Paterson showed that is not possible to do so [5]. Their results show that computing the median exactly in one pass takes linear space. However, our saving grace is that if we could find $\epsilon$-approximate quantiles, we would be done.

To see this, let $\widehat{Q}_j(x)$ be an $\epsilon/4$-approximate $x$-quantile. If $L[j]$ lies between $\widehat{Q}_j(\alpha^{-(k+1)})$ and $\widehat{Q}_j(\alpha^{-k})$, we know that $a(j)$ is less than $(j-1)\alpha^{-k}(1+\epsilon/4)$, and at least $(j-1)\alpha^{-k}(1-\epsilon/4)/(1+\epsilon)$, giving us an approximation of $(1+O(\epsilon))$. (Hence we should have chosen $\epsilon = \epsilon/$(some constant).)

However, note that all these approximate quantiles were needed for calculating a single $a(j)$; to guess the value of $K(L)$, we need good approximations for $Q_j(x)$ for all $1 \leq j \leq n$. In the next section, we show the following theorem:

THEOREM 2.1. *There exists a one pass randomized streaming data structure which for each* $1 \leq j \leq n$ *and* $0 \leq$

*$k \le \log_\alpha n$ gives an $\epsilon$-approximation to $Q_j(\alpha^k)$ with probability $1 - 1/n^2$. The total storage required is space for $O(\log^2 n/\epsilon^3)$ numbers from $\Sigma$.*

Clearly, every time a new element $L[j]$ is read from the stream, this data structure may be used $O(\log n/\epsilon)$ times to estimate $a(j)$. Using a trivial union bound shows that with high probability, all the answers received will be correct, and hence an $(1+\epsilon)$-approximation for the number of non-inversions of a stream can be obtained. Since each of the elements of $\Sigma$ use $\log m$ space, and counting inversions and non-inversions are equivalent (by reversing the total order), this gives us the following theorem:

THEOREM 2.2. *There is a randomized streaming algorithm to calculate the number of inversions in a stream that uses $O(\log^2 n \log m/\epsilon^3)$ space.*

## 3 Finding Quantiles

In this section, we shall build a suite of $O(\epsilon^{-1}\log n)$ samplers, which we shall use to answer all queries $Q_j(x)$, where $x$ is some power of $(1+\epsilon)^{-1}$. Each of these samplers will store $O(\epsilon^{-2}\log n)$ elements of the alphabet $\Sigma$, bringing the total space requirements to $O(\epsilon^{-3}\log^2 n \log m)$. The main techniques used here are fairly routine, and are given here largely for both concreteness and completeness. For the rest of the section, let $\alpha = (1+\epsilon)$, and $\beta = (1+\epsilon/10)$. Let $T = \lceil 8\epsilon^{-2}\ln n \rceil$.

The $i$-th sampler, $A_i$, does the following: each element from the stream is picked with probability $p_i = T/\beta^i$. However, not all picked elements are retained by $A_i$; it just maintains the *smallest* $T$ elements it picked in a list $H_i$. When queried, it returns the *maximum* of the elements in $H_i$.

Note that the values $p_i$ make sense only for $\beta^i \ge T$, or for $i \ge \log_\beta T$; hence we keep samplers $A_i$ for $\log_\beta T \le i \le (\log_\beta n + 1)$. We also keep a "sampler" $A_0$ which picks all the elements and maintains the sorted list of all the $T$ smallest items seen in the stream till now. We should point out that on the arrival of the $j$-th element, all these updates are made after all the queries to estimate $a(j)$ have already been made.

Upon getting a query asking for $Q_j(\alpha^{-k})$, we define $q = (j-1)\alpha^{-k}$, and see if $q \le T$. If so, we can just read off the element at the $\lceil q \rceil$-th position in $A_0$'s list. If not, we find the value of $k'$ such that $\beta^{k'} \le q \le \beta^{k'+1}$. We then query the sampler $A_{k'}$, and return the result as the $\epsilon$-approximate quantile $\widehat{Q}_j(\alpha^{-k})$.

We now prove that, for any fixed query $Q_j(\alpha^{-k})$, the chance of error is $O(1/n^2)$, and hence taking a union bound over the entire process, we would not make an error with constant probability. Let us see how errors can be made: clearly, if we are reading a value from $A_0$, there is no error. If not, then we can err in two ways. Let $\mathcal{E}_1$ be the event that the top element returned by $A_{k'}$ is too low; i.e., it falls among the $N_1 = (j-1)\alpha^{-k}(1-\epsilon)$ smallest elements of $L_j$. Similarly, $\mathcal{E}_2$ is the event that we are too high; i.e., fewer than $T$ elements are chosen from among the smallest $N_2 = (j-1)\alpha^{-k}(1+\epsilon)$ elements of $L_j$.

Let $X_i$ be the indicator variable for the $i$-th smallest element of $L_j$ being picked, and let $S_1 = \sum_{i=1}^{N_1} X_i$. Then $\mu_1 = ES_1 = N_1(T/\beta^{k'}) \le \beta(1-\epsilon)T \le (1-4\epsilon/5)T$. Now setting $\lambda = 4\epsilon/5T$, we can bound

$$\mathbf{Pr}[\mathcal{E}_1] \le \mathbf{Pr}[\,S_1 - \mu_1 > \lambda\,] \le e^{-\lambda^2/3\mu_1} \le 1/n^2.$$

Similarly, to bound $\mathbf{Pr}[\mathcal{E}_2]$, we can set $S_2 = \sum_{i=1}^{N_2} X_i$. Then $\mu_2 = ES_2 = N_2(T/\beta^{k'}) \ge (1+\epsilon)T$, and setting $\lambda = \epsilon T$, we can bound

$$\mathbf{Pr}[\mathcal{E}_2] \le \mathbf{Pr}[\,\mu_2 - S_2 > \lambda\,] \le e^{-\lambda^2/2\mu_2} \le 1/n^2.$$

## References

[1] Miklós Ajtai, T.S. Jayram, S. Ravi Kumar, and D. Sivakumar. Counting inversions in a data stream. In *Proceedings of the 34th Annual ACM Symposium on Theory of Computing*, pages 370–379, 2002.

[2] Michael Greenwald and Sanjeev Khanna. Space-efficient online computation of quantile summaries. In *Proceedings ACM SIGMOD International Conference on Management of Data*, pages 58–66, 2001.

[3] Gurmeet Singh Manku, Sridhar Rajagopalan, and Bruce G. Lindsay. Approximate medians and other quantiles in one pass and with limited memory. In *Proceedings ACM SIGMOD International Conference on Management of Data*, pages 426–435, 1998.

[4] Gurmeet Singh Manku, Sridhar Rajagopalan, and Bruce G. Lindsay. Random sampling techniques for space efficient online computation of order statistics of large datasets. In *Proceedings ACM SIGMOD International Conference on Management of Data*, pages 251–262, 1999.

[5] J. I. Munro and M. S. Paterson. Selection and sorting with limited storage. *Theoret. Comput. Sci.*, 12(3):315–323, 1980.