CS 598CSC: Algorithms for Big Data	Lecture date: Sept 25, 2014
Instructor: Chandra Chekuri	Scribe: Chandra Chekuri

Suppose we have a stream $a_1, a_2, \ldots, a_n$ of objects from an ordered universe. For simplicity we will assume that they are real numbers and more over that they are distinct (for simplicity). We would like to find the k'th ranked element for some $1 \leq k \leq n$. In particular we may be interested in the median element. We will discuss exact and approximate versions of these problems. Another terminology for finding rank k elements is quantiles. Given a number ϕ where $0 < \phi \leq 1$ we would like to return an element of rank ϕn. This normalization allows us to talk about ϵ-approximate quantiles for $\epsilon \in [0, 1)$. An ϵ-approximate quantile is an element whose rank is at least $(\phi - \epsilon)n$ and at most $(\phi + \epsilon)n$. In other words we are allowing an additive error of ϵn. There is a large amount of literature on quantiles and quantile queries. In fact one of the earliest "streaming" papers is the one by Munro and Paterson [5] who described a p-pass algorithm for selection using $\tilde{O}(n^{1/p})$ space for any $p \geq 2$. They also considered the "random-order" streaming setting which has become quite popular in recent years.

The material for these lectures is taken mainly from the excellent chapter/survey by Greenwald and Khanna [1]. We mainly refer to that chapter and describe here the outline of what we covered in lectures. We will omit proofs or give sketchy arguments and refer the reader to [1].

1 Approximate Quantiles and Summaries

Suppose we want to be able to answer ϵ-approximate quantile queries over an ordered set S of n elements. It is easy to see that we can simply pick elements of rank $i|S|/k$ for $1 \leq i \leq k \simeq 1/\epsilon$ and store them as a summary and use the summary to answer any quantile query with an ϵn additive error. However, to obtain these elements we first need to do selection which we can do in the offline setting if all we want is a concise summary. The question we will address is to find a summary in the streaming setting. In the sequel we will count space usage in terms of "words". Thus, the space usage for the preceding offline summary is $\Theta(1/\epsilon)$.

We will see two algorithms. The first will create an ϵ-approximate summary [4] with space $O(\frac{1}{\epsilon}\log^2(\epsilon n))$ and is inspired by the ideas from the work of Munro and Paterson. Greenwald and Khanna [2] gave a different summary that uses $O(\frac{1}{\epsilon}\log(\epsilon n))$ space.

Following [1] we will think of a quantile summary Q as storing a set of elements $\{q_1, q_2, \ldots, q_\ell\}$ from S along with an interval $[\mathbf{rmin}_Q(q_i), \mathbf{rmax}_Q(q_i)]$ for each q_i; $\mathbf{rmin}_Q(q_i)$ is a lower bound on the rank of q_i in S and $\mathbf{rmax}_Q(q_i)$ is an upper bound on the rank of q_i in S. It is convenient to assume that $q_1 < q_2 < \ldots < q_\ell$ and moreover that q_1 is the minimum element in S and that q_ℓ is the maximum element in S. For ease of notation we will simply use Q to refer to the quantile summary and also the (ordered) set $\{q_1, q_2, \ldots, q_\ell\}$.

Our first question is to ask whether a quantile summary Q can be used to give ϵ-approximate quantile queries. The following is intuitive and is worthwhile proving for oneself.

Lemma 1 *Suppose Q is a quantile summary for S such that for $1 \leq i < \ell$, $\mathbf{rmax}_Q(q_{i+1}) - \mathbf{rmin}_Q(q_i) \leq 2\epsilon|S|$. Then Q can be used for ϵ-approximate quantile queries over S.*

In the following, when we say that Q is an ϵ-approximate quantile summary we will implicitly be using the condition in the preceding lemma.

Given a quantile summary Q' for a multiset S' and a quantile summary Q'' for a multiset S'' we would like to combine Q' and Q'' into a quantile summary Q for $S = S' \cup S''$; by union here we view S as a multiset. Of course we would like to keep the approximation of the resulting summary similar to those of Q' and Q''. Here a lemma which shows that indeed we can easily combine.

Lemma 2 *Let Q' be an ϵ'-approximate quantile summary for multiset S' and Q'' be an ϵ''-approximate quantile summary for multiset S''. Then $Q = Q' \cup Q''$ yields an ϵ-approximate quantile summary for $S = S' \cup S''$ where $\epsilon \leq \frac{\epsilon' n' + \epsilon'' n''}{n' + n''} \leq \max\{\epsilon', \epsilon''\}$ where $n' = |S'|$ and $n'' = |S''|$.*

We will not prove the correctness but describe how the intervals are constructed for Q from those for Q' and Q''. Suppose $Q' = \{x_1, x_2, \ldots, x_a\}$ and $Q'' = \{y_1, y_2, \ldots, y_b\}$. Let $Q = Q' \cup Q'' = \{z_1, z_2, \ldots, z_{a+b}\}$. Consider some $z_i \in Q$ and suppose $z_i = x_r$ for some $1 \leq r \leq a$. Let y_s be the largest element of Q'' smaller than x_r and y_t be the smallest element of Q'' larger than x_r. We will ignore the cases where one or both of y_s, y_t are not defined. We set

$$\mathbf{rmin}_Q(z_i) = \mathbf{rmin}_{Q'}(x_r) + \mathbf{rmin}_{Q''}(y_s)$$

and

$$\mathbf{rmax}_Q(z_i) = \mathbf{rmax}_{Q'}(x_r) + \mathbf{rmax}_{Q''}(y_t) - 1.$$

It is easy to justify the above as valid intervals. One can then prove that with these settings, for $1 \leq i < a + b$ the following holds:

$$\mathbf{rmax}_Q(z_{i+1}) - \mathbf{rmin}_Q(z_i) \leq 2\epsilon(n' + n'').$$

We will refer to the above operation as COMBINE(Q', Q''). The following is easy to see.

Lemma 3 *Let $Q_1, \ldots, Q_h$ be ϵ-approximate quantile summaries for $S_1, S_2, \ldots, S_h$ respectively. Then $Q_1, Q_2, \ldots, Q_h$ can be combined in any arbitrary order to obtain an ϵ-approximate quantile summary $Q = Q_1 \cup \ldots \cup Q_h$ for $S_1 \cup \ldots \cup S_h$.*

Next we discuss the PRUNE(Q, k) operation on a quantile summary Q that reduces the size of Q to $k + 1$ while losing a bit in the approximation quality.

Lemma 4 *Let Q' be an ϵ-approximate quantile summary for S. Given an integer parameter k there is a quantile summary $Q \subseteq Q'$ for S such that $|Q| \leq k + 1$ and it is $(\epsilon + \frac{1}{2k})$-approximate.*

We sketch the proof. We simply query Q' for ranks $1, |S|/k, 2|S|/k, \ldots, |S|$ and choose these elements to be in Q. We retain their **rmin** and **rmax** values from Q'.

$$\mathbf{rmax}_Q(q_{i+1}) - \mathbf{rmin}_Q(q_i) \leq i|S|/k + \epsilon|S| - ((i-1)|S|/k - \epsilon|S|) \leq |S|/k + 2\epsilon|S|.$$

1.1 An $O(\frac{1}{\epsilon}\log^2(\epsilon n))$ space algorithm

The idea is inspired by the Munro-Paterson algorithm and was abstracted in the paper by Manku et al. We will describe the idea in an offline fashion though it can be implemented in the streaming setting. We will use several quantile summaries with k elements each for some parameter k, say ℓ of them. Each summary of size k will be called a buffer. We will need to reuse these buffers as more elements in the stream arrive; buffers will combined and pruned, in other words "collapsed" into a single buffer of the same size k. Pruning introduces error.

Assume n/k is a power of 2 for simplicity. Consider a complete binary tree with n/k leaves where each leaf corresponds to k consecutive elements of the stream. Think of assigning a buffer of size k to obtain a 0-error quantile summary for those k elements; technically we need $k+1$ elements but we will ignore this minor additive issue for sake of clarity of exposition. Now each internal node of the tree corresponds to a subset of the elements of the stream. Imagine assigning a buffer of size k to each internal node to maintain an approximate quantile summary for the elements of the stream in the sub-tree. To obtain a summary at node v we combine the summaries of its two children v_1 and v_2 and prune it back to size k introducing and additional $1/(2k)$ error in the approximation. The quantile summary at the root of size k will be our final summary that we output for the stream.

Our first observation is that in fact we can implement the tree-based scheme with $\ell = O(h)$ buffers where h is the height of the tree. Note that $h \simeq \log(n/k)$. The reason we only need $O(h)$ buffers is that if need a new buffer for the next k elements in the stream we can collapse two buffers corresponding to the children of an internal node — hence, at any time we need to maintain only one buffer per level of the tree (plus a temporary buffer to do the collapse operation).

Consider the quantile summary at the leaves. They have error 0 since we store all the elements in the buffer. However at each level the error increases by $1/(2k)$. Hence the error of the summary at the root is $h/(2k)$. Thus, to obtain an ϵ-approximate quantile summary we need $h/(2k) \leq \epsilon$. And $h = \log(n/k)$. One can see that for this to work out it suffices to choose $k > \frac{1}{2\epsilon}\log(2\epsilon n)$.

The total space usage is $\Theta(hk)$ and $h = \log(n/k)$ and thus the space usage is $O(\frac{1}{\epsilon}\log^2(\epsilon n))$.

One can choose d-ary trees instead of binary trees and some optimization can be done to improve the constants but the asymptotic dependence on ϵ does not improve with this high-level scheme.

1.2 An $O(\frac{1}{\epsilon}\log(\epsilon n)$ space algorithm

We now briefly describe the Greenwald-Khanna algorithm that obtains an improved space bound. The GK algorithm maintains a quantile summary as a collection of s tuples $t_0, t_2, \ldots, t_{s-1}$ where each tuple t_i is a triple (v_i, g_i, Δ_i): (i) a value v_i that is an element of the ordered set S (ii) the value g_i which is equal to $\mathbf{rmin}_{\mathrm{GK}}(v_i) - \mathbf{rmin}_{\mathrm{GK}}(v_{i-1})$ (for $i = 0$, $g_i = 0$) and (iii) the value Δ_i which equals $\mathbf{rmax}_{\mathrm{GK}}(v_i) - \mathbf{rmin}_{\mathrm{GK}}(v_i)$. The elements $v_0, v_1, \ldots, v_{s-1}$ are in ascending order and moreover v_0 will the minimum element in S and v_{s-1} is the maximum element in S. Note that $n = \sum_{j=1}^{s-1} g_j$. The summary also stores n the number of elements seen so far. With this set up we note that $\mathbf{rmin}_{\mathrm{GK}}(v_i) = \sum_{j \leq i} g_j$ and $\mathbf{rmax}_{\mathrm{GK}}(v_i) = \Delta_i + \sum_{j \leq i} g_j$.

Lemma 5 *Suppose Q is a GK quantile summary for a set $|S|$ such that $\max_i(g_i + \Delta_i) \leq 2\epsilon|S|$ then it can be used to answer quantile queries with ϵn additive error.*

The query can be answered as follows. Given rank r, find i such that $r - \mathbf{rmin}_{\mathrm{GK}}(v_i) \leq \epsilon n$ and $\mathbf{rmax}_{\mathrm{GK}}(v_i) - r \leq \epsilon n$ and output v_i; here n is the current size of S.

The quantile summary is updated via two operations. When a new element v arrives it is inserted into the summary. The quantile summary is compressed by merging consecutive elements to keep the summary within the desired space bounds.

We now describe the INSERT operation that takes a quantile summary Q and inserts a new element v. First we search over the elements in Q to find an i such that $v_i < v < v_{i+1}$; the case when v is the new smallest element or the new largest element are handled easily. A new tuple $t = (v, 1, \Delta)$ with $\Delta = \lfloor 2\epsilon n \rfloor - 1$ is added to the summary where t becomes the new $(i+1)$st tuple. Note that here n is the current size of the stream. It is not hard to see if the summary Q before

arrival of v satisfied the condition in Lemma 5 then Q satisfies the condition after inserting the tuple (note that n increased by 1). We note that that the first $1/(2\epsilon)$ elements are inserted into the summary with $\Delta_i = 0$.

Compression is the main ingredient. To understand the operation it is helpful to define the notion of a *capacity* of a tuple. Note that when v arrived $t = (v, 1, \Delta)$ is inserted where $\Delta \simeq 2\epsilon n'$ where n' is the time when v arrived. At time $n > n'$ the capacity of the tuple t_i is defined as $2\epsilon n - \Delta_i$. As n grows, the capacity of the tuple increases since we are allowed to have more error. We can merge tuples $t_{i'}, t_{i'+1}, \ldots, t_i$ into t_{i+1} at time n (which means we eliminate $t_{i'}, \ldots, t_i$) while ensuring the desired precision if $\sum_{j=i'}^{i+1} g_j + \Delta_{i+1} \leq 2\epsilon n$; g_{i+1} is updated to $\sum_{j=i'}^{i+1} g_j$ and Δ_{i+1} does not change. Note that this means that Δ of a tuple does not change once it is inserted.

Note that the insertion and merging operations preserve correctness of the summary. In order to obtain the desired space bound the merging/compression has to be done rather carefully. We will not go into details but mention that one of the key ideas is to keep track of the capacity of the tuples in geometrically increasing intervals and to ensure that the summary retains only a small number of tuples per interval.

2 Exact Selection

We will now describe a p-pass deterministic algorithm to select the rank k element in a stream using $\tilde{O}(n^{1/p})$-space; here $p \geq 2$. It is not hard to show that for $p = 1$ any deterministic algorithm needs $\Omega(n)$ space; one has to be a bit careful in arguing about bits vs words and the precise model but a near-linear lower bound is easy. Munro and Paterson described the $\tilde{O}(n^{1/p})$-space algorithm using p passes. We will not describe their precise algorithm but instead use the approximate quantile based analysis.

We will show that given space s and a stream of n items the problem can be effectively reduced in one pass to selecting from $O(n \log^2 n/s)$ items.

Suppose we can do the above. Choose $s = n^{1/p}(\log n)^{2-2/p}$. After i passes the problem is reduced to $n^{\frac{p-i}{p}}(\log n)^{\frac{2i}{p}}$ elements. Setting $i = p - 1$ we see that the number of elements left for the p'th pass is $O(s)$. Thus all of them can be stored and selection can be done offline.

We now describe how to use one pass to reduce the effective size of the elements under consideration to $O(n \log^2 n/s)$. The idea is that we will be able to select two elements a_1, b_1 from the stream such that $a_1 < b_1$ and the k'th ranked element is guaranteed to be in the interval $[a_1, b_1]$. Moreover, we are also guaranteed that the number of elements between a and b in the stream is $O(n \log^2 n/s)$. a_1 and b_1 are the left and right filter after pass 1. Initially $a_0 = -\infty$ and $b_0 = \infty$. After i passes we will have filters a_i, b_i. Note that during the $(i + 1)$st pass we can compute the exact rank of a_i and b_i.

How do we find a_1, b_1? We saw how to obtain an ϵ-approximate summary using $O(\frac{1}{\epsilon} \log^2 n)$ space. Thus, if we have space s, we can set $\epsilon' = \log^2 n/s$. Let $Q = \{q_1, q_2, \ldots, q_\ell\}$ be ϵ'-approximate quantile summary for the stream. We query Q for $r_1 = k - \epsilon' n - 1$ and $r_2 = k + \epsilon' n + 1$ and obtain a_1 and b_1 as the answers to the query (here we are ignoring the corner cases where $r_1 < 0$ or $r_2 > n$). Then, by the ϵ'-approximate guarantee of Q we have that the rank k element lies in the interval $[a_1, b_1]$ and moreover there are at most $O(\epsilon' n)$ elements in this range.

It is useful to work out the algebra for $p = 2$ which shows that the median can be computed in $O(\sqrt{n} \log^2 n)$ space.

2.1 Random Order Streams

Munro and Paterson also consider the random order stream model in their paper. Here we assume that the stream is a random permutation of an ordered set. It is also convenient to use a different model where the the i'th element is a real number drawn independently from the interval $[0, 1]$. We can ask whether the randomness can be taken advantage of. Indeed one can. They showed that with $O(\sqrt{n})$ space one can find the median with high probability. More generally they showed that in p passes one can find the median with space $O(n^{1/(2p)})$. Even though this space bound is better than for adversarial streams it still requires $\Omega(\log n)$ passes is we have only poly-logarithmic space, same as the adversarial setting. Guha and McGregor [3] showed that in fact $O(\log \log n)$ passes suffice (with high probability).

Here we describe the Munro-Paterson algorithm; see also `http://polylogblog.wordpress.com/2009/08/30/bite-sized-streams-exact-median-of-a-random-order-stream/`.

The algorithm maintains a set S of s consecutively ranked elements in the stream seen so far. It maintains two counters ℓ for the number of elements less then $\min S$ (the min element in S) which have been seen so far and h for the number of elements larger than $\max S$ which have been so far. It tries to maintain $h - \ell$ as close to 0 as possible to "capture" the median.

MunroPaterson(s):

$n \leftarrow 0$
$S \leftarrow \emptyset$
$\ell, h \leftarrow 0$
While (stream is not empty) do
  $n \leftarrow n + 1$
  a is the new element
  if ($a < \min S$) then $\ell \leftarrow \ell + 1$
  else if ($a > \max S$) then $h \leftarrow h + 1$
  else add a to S
  if ($|S| = s + 1$)
    if ($h < \ell$) discard $\max S$ from S and $h \leftarrow h + 1$
    else discard $\min S$ from S and $\ell \leftarrow \ell + 1$
endWhile
if $1 \leq (n + 1)/2 - \ell \leq s$ then
  return $(n + 1)/2 - \ell$-th smallest element in S as median
else return FAIL.

To analyze the algorithm we consider the random variable $d = h - \ell$ which starts at 0. In the first s iterations we simply fill up S to capacity and $h - \ell$ remains 0. After that, in each step d is either incremented or decremented by 1. Consider the start of iteration i when $i > s$. The total number of elements seen prior to i is $i - 1 = \ell + h + s$. In iteration i, since the permutation is random, the probability that a_i will be larger than $\max S$ is precisely $(h + 1)/(h + s + 1 + \ell)$. The probability that a_i will be smaller than $\min S$ is precisely $(\ell + 1)/(h + s + 1 + \ell)$ and thus with probability $(s - 1)/(h + s + 1 + \ell)$, a_i will be added to S.

Note that the algorithm fails only if $|d|$ at the end of the stream is greater than s. A sufficient condition for success is that $|d| \leq s$ *throughout* the algorithm. Let $p_{d,i}$ be the probability that $|d|$ increases by 1 conditioned on the fact that $0 < |d| < s$. Then we see that $p_{d,i} \leq 1/2$. Thus the process can be seen to be similar to a random walk on the line and some analysis shows that if we choose $s = \Omega(\sqrt{n})$ then with high probability $|d| < s$ throughout. Thus, $\Omega(\sqrt{n})$ space suffices to find the median with high probability when the stream is in random order.

Connection to CountMin sketch and deletions: Note that when we were discussion frequency moments we assume that the elements were drawn from a $[n]$ where n was known in advance while here we did not assume anything about the elements other than the fact that they came from an ordered universe (apologies for the confusion in notation since we used m previously for length of stream). If we know the range of the elements in advance and it is small compared to the length of the stream then CountMin and related techniques are better suited and provide the ability to handle deletions. The GK summary can also handle some deletions. We refer the reader to [1] for more details.

Lower Bounds: For median selection Munro and Paterson showed a lower bound of $\Omega(n^{1/p})$ on the space for p passes in a restricted model of computation. Guha and McGregor showed a lower bound of $\Omega(n^{1/p}/p^6)$ bits without any restriction. For random order streams $O(\log \log n)$ passes suffice with polylog(n) space for exact selection with high probability [3]. Moreover $\Omega(\log \log n)$ passes are indeed necessary; see [3] for references and discussion.

Bibliographic Notes: See the references for more information.

References

[1] Michael Greenwald and Sanjeev Khanna. Quantiles and equidepth histograms over streams. Available at `http://www.cis.upenn.edu/~mbgreen/papers/chapter.pdf`. To appear as a chapter in a forthcoming book on Data Stream Management.

[2] Michael Greenwald and Sanjeev Khanna. Space-efficient online computation of quantile summaries. In *ACM SIGMOD Record*, volume 30, pages 58–66. ACM, 2001.

[3] Sudipto Guha and Andrew McGregor. Stream order and order statistics: Quantile estimation in random-order streams. *SIAM Journal on Computing*, 38(5):2044–2059, 2009.

[4] Gurmeet Singh Manku, Sridhar Rajagopalan, and Bruce G Lindsay. Approximate medians and other quantiles in one pass and with limited memory. In *ACM SIGMOD Record*, volume 27, pages 426–435. ACM, 1998.

[5] J Ian Munro and Mike S Paterson. Selection and sorting with limited storage. *Theoretical computer science*, 12(3):315–323, 1980.