Quantiles and Selection

Lecture 16

October 20, 2020

Part I

Introduction

Selection

Selection: Given a sequence of numbers $a_1, a_2, \ldots, a_n$ and integer $k \in [n]$ want to find the rank k element (the k'th element after sorting)

Median: rank $n/2$ element

Selection

Selection: Given a sequence of numbers $a_1, a_2, \ldots, a_n$ and integer $k \in [n]$ want to find the rank k element (the k'th element after sorting)

Median: rank $n/2$ element

Offline solutions:

- Sort and pick the k'th element. $O(n \log n)$ time. Can find all ranks in constant time after sorting.
- $O(n)$ time algorithm for Selection of given rank k. Randomized QuickSelect or deterministic Median-of-Medians algorithm (clever but slow).

Selection in Streaming

Question: Suppose $a_1, a_2, \ldots, a_n$ arrive in a stream. Can we do Selection in small space?

Selection in Streaming

Question: Suppose $a_1, a_2, \ldots, a_n$ arrive in a stream. Can we do Selection in small space?

Exact Selection in one pass requires $\Omega(n)$ space. Need to store all elements so trivial solution is optimal.

Selection in Streaming

Question: Suppose $a_1, a_2, \ldots, a_n$ arrive in a stream. Can we do Selection in small space?

Exact Selection in one pass requires $\Omega(n)$ space. Need to store all elements so trivial solution is optimal.

Relaxations:

- Approximate selection. Recall sampling to find ϵ-approximate median using $O(\frac{1}{\epsilon^2} \log(1/\delta))$ samples. Can do this in streaming with reservoir sampling.
- *Multiple* passes.
- Assume random order arrival of elements.

Selection in Multiple Passes

Multipass model: See same stream p times for some $p \geq 1$. With larger p one can do more with same memory bound.

Initially motivated by database applications where random access main memory is small and large external memory (such as tapes) that allow for reasonably fast sequential scans.

Selection in Multiple Passes

Multipass model: See same stream p times for some $p \geq 1$. With larger p one can do more with same memory bound.

Initially motivated by database applications where random access main memory is small and large external memory (such as tapes) that allow for reasonably fast sequential scans.

Selection in multiple passes:

- $\Theta(n)$ space allows 1 pass.
- $O(1)$ space. How many passes?

Selection in Multiple Passes

Multipass model: See same stream p times for some $p \geq 1$. With larger p one can do more with same memory bound.

Initially motivated by database applications where random access main memory is small and large external memory (such as tapes) that allow for reasonably fast sequential scans.

Selection in multiple passes:

- $\Theta(n)$ space allows 1 pass.
- $O(1)$ space. How many passes? $O(\log n)$ suffices. Implement Quick Select in $O(1)$ space.

Selection in Multiple Passes

Multipass model: See same stream p times for some $p \geq 1$. With larger p one can do more with same memory bound.

Initially motivated by database applications where random access main memory is small and large external memory (such as tapes) that allow for reasonably fast sequential scans.

Selection in multiple passes:

- $\Theta(n)$ space allows 1 pass.
- $O(1)$ space. How many passes? $O(\log n)$ suffices. Implement Quick Select in $O(1)$ space.
- p passes?

Selection in Multiple Passes

Multipass model: See same stream p times for some $p \geq 1$. With larger p one can do more with same memory bound.

Initially motivated by database applications where random access main memory is small and large external memory (such as tapes) that allow for reasonably fast sequential scans.

Selection in multiple passes:

- $\Theta(n)$ space allows 1 pass.
- $O(1)$ space. How many passes? $O(\log n)$ suffices. Implement Quick Select in $O(1)$ space.
- p passes? $O(n^{1/p}\text{polylog}(n))$ space suffices. Hence $O(\sqrt{n}\log n)$ for 2 passes. [Munro-Paterson 1980]

Quantiles

Large numerical/ordered data: say heights/weights/salaries of the population of the country.

Exact selection is not as interesting as high-level summary. Pick some granularity and bucket data into groups of roughly equal size.

Example: For $\alpha = 1, 2, \ldots, 100$ want α percentile salaries

More precision: For $\alpha = 0.1, 0.2, \ldots, 100$ want α percentile salaries

Quantiles

Large numerical/ordered data: say heights/weights/salaries of the population of the country.

Exact selection is not as interesting as high-level summary. Pick some granularity and bucket data into groups of roughly equal size.

Example: For $\alpha = 1, 2, \ldots, 100$ want α percentile salaries

More precision: For $\alpha = 0.1, 0.2, \ldots, 100$ want α percentile salaries

In terms of Selection:

- want rank k element for $k = \frac{\alpha}{100} n$ for each α
- allows for ϵ-*approximate* Selection (additive error ϵn where ϵ is granularity in quantile)

Quantile Summaries or Approximate Selection in Streaming

See stream of numbers $a_1, a_2, \ldots, a_n$.
Parameter $\epsilon \in (0, 1)$

Maintain a small space summary such that given any $k \in [n]$ can output number a from stream such that

$$k - \epsilon n \leq \text{rank}(a) \leq k + \epsilon n$$

Quantile Summaries or Approximate Selection in Streaming

See stream of numbers $a_1, a_2, \ldots, a_n$.
Parameter $\epsilon \in (0, 1)$

Maintain a small space summary such that given any $k \in [n]$ can output number a from stream such that

$$k - \epsilon n \leq \text{rank}(a) \leq k + \epsilon n$$

Offline: can do with $O(1/\epsilon)$ space. Store rank $\epsilon i/n$ elements for $i = 1, 2, \ldots, 1/\epsilon$

Q: Can we do it in streaming and how much space do we need?

Quantile Summaries or Approximate Selection in Streaming

See stream of numbers $a_1, a_2, \ldots, a_n$ Parameter $\epsilon \in (0, 1)$

Maintain a small space summary such that given any $k \in [n]$ can output number a from stream such that

$$k - \epsilon n \leq \text{rank}(a) \leq k + \epsilon n$$

Q: Can we do it in streaming and how much space do we need?

- $O(\frac{1}{\epsilon} \log^2 n)$ space using merge and reduce approach
- Involved $O(\frac{1}{\epsilon} \log(n/\epsilon))$ space algorithm that is near optimal

Both are deterministic algorithms. Can be used to derive Munro-Paterson multi-pass Selection algorithm

Part II

Approximate Quantiles in Streaming

Quantile Summary

See stream of numbers $a_1, a_2, \ldots, a_n$. Parameter $\epsilon \in (0, 1)$
Note: Items can be from any ordered set, use only comparison

What should we store?

Quantile Summary

See stream of numbers $a_1, a_2, \ldots, a_n$. Parameter $\epsilon \in (0, 1)$
Note: Items can be from any ordered set, use only comparison

What should we store? Take cue from offline solution. Equally spaced $1/\epsilon$ elements from sorted list.

Quantile Summary

See stream of numbers $a_1, a_2, \ldots, a_n$. Parameter $\epsilon \in (0, 1)$
Note: Items can be from any ordered set, use only comparison

What should we store? Take cue from offline solution. Equally spaced $1/\epsilon$ elements from sorted list.

Quantile Summary:

- $Q = \{q_1, q_2, \ldots, q_\ell\}$ where each q_i is an element of stream. Wlog $q_1 < q_2 < \ldots < q_\ell$ and q_1 is smallest and q_ℓ is largest in stream
- For each $q_i \in Q$ an interval $I(q_i) = [\mathbf{rmin}_Q(q_i), \mathbf{rmax}_Q(q_i)]$ where $\mathbf{rmin}_Q(q_i) \leq \mathbf{rank}(q_i) \leq \mathbf{rmax}_Q(q_i)$

Quantile Summary

Quantile Summary:

- $Q = \{q_1, q_2, \ldots, q_\ell\}$. Also $q_1 < q_2 < \ldots < q_\ell$ and q_1 is smallest and q_ℓ is largest
- For each $q_i \in Q$ an interval $I(q_i) = [\mathbf{rmin}_Q(q_i), \mathbf{rmax}_Q(q_i)]$ where $\mathbf{rmin}_Q(q_i) \leq \mathbf{rank}(q_i) \leq \mathbf{rmax}_Q(q_i)$

Given $k \in [n]$ want to use Q to answer ϵ-approximate rank k query. How?

Quantile Summary

Quantile Summary:

- $Q = \{q_1, q_2, \ldots, q_\ell\}$. Also $q_1 < q_2 < \ldots < q_\ell$ and q_1 is smallest and q_ℓ is largest
- For each $q_i \in Q$ an interval $I(q_i) = [\mathbf{rmin}_Q(q_i), \mathbf{rmax}_Q(q_i)]$ where $\mathbf{rmin}_Q(q_i) \le \mathbf{rank}(q_i) \le \mathbf{rmax}_Q(q_i)$

Given $k \in [n]$ want to use Q to answer ϵ-approximate rank k query. How?

Suppose $I(q_i) \subseteq [k - \epsilon n, k + \epsilon n]$ then it is clear that q_i is good to output since

$$k - \epsilon n \le \mathbf{rmin}(q_i) \le \mathbf{rank}(q_i) \le \mathbf{rmax}(q_i) \le k + \epsilon n.$$

ϵ-Approximate Quantile Summary

Quantile Summary:

- $Q = \{q_1, q_2, \ldots, q_\ell\}$. Also $q_1 < q_2 < \ldots < q_\ell$ and q_1 is smallest and q_ℓ is largest
- For each $q_i \in Q$ an interval $I(q_i) = [\mathbf{rmin}_Q(q_i), \mathbf{rmax}_Q(q_i)]$ where $\mathbf{rmin}_Q(q_i) \leq \mathbf{rank}(q_i) \leq \mathbf{rmax}_Q(q_i)$

Maintain key invariant: For each i,

$$\mathbf{rmax}(q_{i+1}) - \mathbf{rmin}(q_i) \leq 2\epsilon n$$

also implies $\mathbf{rank}(q_{i+1}) - \mathbf{rank}(q_i) \leq 2\epsilon n$

ϵ-Approximate Quantile Summary

Quantile Summary:

- $Q = \{q_1, q_2, \ldots, q_\ell\}$. Also $q_1 < q_2 < \ldots < q_\ell$ and q_1 is smallest and q_ℓ is largest
- For each $q_i \in Q$ an interval $I(q_i) = [\mathbf{rmin}_Q(q_i), \mathbf{rmax}_Q(q_i)]$ where $\mathbf{rmin}_Q(q_i) \leq \mathbf{rank}(q_i) \leq \mathbf{rmax}_Q(q_i)$

Maintain key invariant: For each i,

$$\mathbf{rmax}(q_{i+1}) - \mathbf{rmin}(q_i) \leq 2\epsilon n$$

also implies $\mathbf{rank}(q_{i+1}) - \mathbf{rank}(q_i) \leq 2\epsilon n$

Lemma

With invariant quantile summary can be used to answer ϵ-approximate rank queries.

Proof of Lemma

Maintain key invariant: For each i,

$$\mathbf{rmax}(q_{i+1}) - \mathbf{rmin}(q_i) \leq 2\epsilon n$$

Claim: There exists q_j such that $I(q_j) \subseteq [k - \epsilon n, k + \epsilon n]$

Proof of Lemma

Maintain key invariant: For each i,

$$\mathbf{rmax}(q_{i+1}) - \mathbf{rmin}(q_i) \leq 2\epsilon n$$

Claim: There exists q_j such that $I(q_j) \subseteq [k - \epsilon n, k + \epsilon n]$

- If $k \geq (1 - \epsilon)n$ then q_ℓ satisfies condition.

Proof of Lemma

Maintain key invariant: For each i,

$$\mathbf{rmax}(q_{i+1}) - \mathbf{rmin}(q_i) \leq 2\epsilon n$$

Claim: There exists q_j such that $I(q_j) \subseteq [k - \epsilon n, k + \epsilon n]$

- If $k \geq (1 - \epsilon)n$ then q_ℓ satisfies condition.
- Let j be smallest index such that $\mathbf{rmax}(q_j) \geq k + \epsilon n$ (exists since $\mathbf{rmax}(q_\ell) = n$ and $k < (1 - \epsilon)n$).

Proof of Lemma

Maintain key invariant: For each i,

$$\mathbf{rmax}(q_{i+1}) - \mathbf{rmin}(q_i) \leq 2\epsilon n$$

Claim: There exists q_j such that $I(q_j) \subseteq [k - \epsilon n, k + \epsilon n]$

- If $k \geq (1 - \epsilon)n$ then q_ℓ satisfies condition.
- Let j be smallest index such that $\mathbf{rmax}(q_j) \geq k + \epsilon n$ (exists since $\mathbf{rmax}(q_\ell) = n$ and $k < (1 - \epsilon)n$).
- q_{j-1} satisfies condition. Suppose not. By choice of j, $\mathbf{rmax}(q_{j-1}) < k + \epsilon n$. Since condition is not satisfied by q_{j-1}, $\mathbf{rmin}(q_{j-1}) < k - \epsilon n$ but then

  $$\mathbf{rmax}(q_j) - \mathbf{rmin}(q_{j-1}) > k + \epsilon n - (k - \epsilon n) > 2\epsilon n$$

  contradiction to invariant.

Maintaining ϵ-Approx Quantile Summary in Streaming

Question: How to maintain ϵ-approximate quantile summary in small space in streaming setting?

Merge and Reduce/Prune Framework
(also useful in other settings)

Merge: given ϵ_1-approx Q_1 for multiset S_1 and ϵ_2-approx Q_2 for multiset S_2 obtain approx Q for $S = S_1 \cup S_1$

Prune: Given ϵ-approx Q for S of size ℓ, prune to size h without increasing error by too much

Merging Summaries

$Q_1 = \{q_1, q_2, \ldots, q_\ell\}$ and intervals $I_1(q_1), \ldots, I_1(q_\ell)$ for multiset S_1 with $n_1 = |S_1|$

$Q_2 = \{s_1, s_2, \ldots, s_m\}$ and intervals $I_2(s_1), \ldots, I_2(s_m)$ for multiset S_2 with $n_2 = |S_1|$

$Q = \{z_1, z_2, \ldots, z_{\ell+m}\}$ which is sorted version of $\{q_1, q_2, \ldots, q_\ell, s_1, \ldots, s_m\}$ for multiset $S = S_1 \uplus S_2$ with $n = n_1 + n_2$

Merging Summaries

$Q_1 = \{q_1, q_2, \ldots, q_\ell\}$ and intervals $I_1(q_1), \ldots, I_1(q_\ell)$ for multiset S_1 with $n_1 = |S_1|$

$Q_2 = \{s_1, s_2, \ldots, s_m\}$ and intervals $I_2(s_1), \ldots, I_2(s_m)$ for multiset S_2 with $n_2 = |S_1|$

$Q = \{z_1, z_2, \ldots, z_{\ell+m}\}$ which is sorted version of $\{q_1, q_2, \ldots, q_\ell, s_1, \ldots, s_m\}$ for multiset $S = S_1 \uplus S_2$ with $n = n_1 + n_2$

How do we find intervals for Q while maintaining key invariant?

Consider z_i and assume wlog that $z_i = q_j$ for some $1 \leq j \leq \ell$

Merging

Consider z_i and assume wlog that $z_i = q_j$ for some $1 \leq j \leq \ell$

Find s_t, s_{t+1} such that $s_t \leq q_j \leq s_{t+1}$ (ignore corner cases)

We know that $\mathbf{rmin}_{Q_1}(q_j)$ elements in S_1 are smaller than q_j and also $\mathbf{rmin}_{Q_2}(s_t)$ elements in S_2 are smaller than q_j. Hence it safe to set

$$\mathbf{rmin}_Q(z_i) = \mathbf{rmin}_{Q_1}(q_j) + \mathbf{rmin}_{Q_2}(s_t)$$

Merging

Consider z_i and assume wlog that $z_i = q_j$ for some $1 \leq j \leq \ell$

Find s_t, s_{t+1} such that $s_t \leq q_j \leq s_{t+1}$ (ignore corner cases)

We know that $\mathbf{rmin}_{Q_1}(q_j)$ elements in S_1 are smaller than q_j and also $\mathbf{rmin}_{Q_2}(s_t)$ elements in S_2 are smaller than q_j. Hence it safe to set

$$\mathbf{rmin}_Q(z_i) = \mathbf{rmin}_{Q_1}(q_j) + \mathbf{rmin}_{Q_2}(s_t)$$

Similarly it is safe to set

$$\mathbf{rmax}_Q(z_i) = \mathbf{rmax}_{Q_1}(q_j) + \mathbf{rmax}_{Q_2}(s_{t+1}) - 1$$

Merging

Lemma

If Q_1 is an ϵ_1-approx quantile summary for S_1 and Q_2 is an ϵ_2-approx quantile summary for S_2 then Q is an $\epsilon = \max\{\epsilon_1, \epsilon_2\}$-approx quantile summary for $S = S_1 \uplus S_2$.

Hence error does not increase but $|Q| = |Q_1| + |Q_2|$.

For proof need to verify key invariant. $Q = \{z_1, z_2, \ldots, z_{\ell+m}\}$.
Need to show that

$$\mathbf{rmax}_Q(z_{i+1}) - \mathbf{rmin}_Q(z_i) \leq 2\epsilon(n_1 + n_2).$$

Merging Analysis

Need to show that

$$\mathbf{rmax}_Q(z_{i+1}) - \mathbf{rmin}_Q(z_i) \leq 2\epsilon(n_1 + n_2).$$

Case 1: z_i, z_{i+1} in same summary, say Q_1 wlog. Then $z_i = q_j$ and $z_{i+1} = q_{j+1}$ for some j.

Merging Analysis

Need to show that

$$\mathbf{rmax}_Q(z_{i+1}) - \mathbf{rmin}_Q(z_i) \leq 2\epsilon(n_1 + n_2).$$

Case 1: z_i, z_{i+1} in same summary, say Q_1 wlog. Then $z_i = q_j$ and $z_{i+1} = q_{j+1}$ for some j.
This implies that there are s_t, s_{t+1} in Q_2 such that $s_t \leq q_j < q_{j+1} \leq s_{t+1}$.

Hence $\mathbf{rmax}_Q(z_{i+1}) - \mathbf{rmin}_Q(z_i)$

$$\begin{aligned} &= \mathbf{rmax}_{Q_1}(q_{j+1}) + \mathbf{rmax}_{Q_2}(s_{t+1}) - 1 - (\mathbf{rmin}_{Q_1}(q_j) + \mathbf{rmin}_{Q_2}(s_t)) \\ &\leq (\mathbf{rmax}_{Q_1}(q_{j+1}) - \mathbf{rmin}_{Q_1}(q_j)) + (\mathbf{rmax}_{Q_2}(s_{t+1}) - \mathbf{rmin}_{Q_2}(s_t)) \\ &\leq 2\epsilon n_1 + 2\epsilon n_2 \leq 2\epsilon(n_1 + n_2) \end{aligned}$$

Merging Analysis

Case 2: z_i, z_{i+1} in different summaries, say Q_1, Q_2 wlog. Then $z_i = q_j$ and $z_{i+1} = s_{t+1}$ for some j, t.

Merging Analysis

Case 2: z_i, z_{i+1} in different summaries, say Q_1, Q_2 wlog. Then $z_i = q_j$ and $z_{i+1} = s_{t+1}$ for some j, t.
This implies that $s_t \leq q_j \leq s_{t+1} \leq q_{j+1}$ (ignoring corner cases)

Hence $\mathbf{rmax}_Q(z_{i+1}) - \mathbf{rmin}_Q(z_i)$

$$
\begin{aligned}
&= \mathbf{rmax}_{Q_1}(q_{j+1}) + \mathbf{rmax}_{Q_2}(s_{t+1}) - 1 - (\mathbf{rmin}_{Q_1}(q_j) + \mathbf{rmin}_{Q_2}(s_t)) \\
&\leq (\mathbf{rmax}_{Q_1}(q_{j+1}) - \mathbf{rmin}_{Q_1}(q_j)) + (\mathbf{rmax}_{Q_2}(s_{t+1}) - \mathbf{rmin}_{Q_2}(s_t)) \\
&\leq 2\epsilon n_1 + 2\epsilon n_2 \leq 2\epsilon(n_1 + n_2)
\end{aligned}
$$

Pruning/Reducing Summary

Merging keeps accuracy but increases summary size.

Reduce/Prune: reduce size at expense of accuracy.

Lemma

Given ϵ-approx quantile Q and integer $h \geq 3$ can find Q' such that $|Q'| \leq h + 1$ and Q' is ϵ'-approximate for $\epsilon' \leq \epsilon + \frac{1}{2h}$.

Pruning/Reducing Summary

Merging keeps accuracy but increases summary size.

Reduce/Prune: reduce size at expense of accuracy.

Lemma

Given ϵ-approx quantile Q and integer $h \geq 3$ can find Q' such that $|Q'| \leq h + 1$ and Q' is ϵ'-approximate for $\epsilon' \leq \epsilon + \frac{1}{2h}$.

$Q = \{q_1, q_2, \ldots, q_\ell\}$ and wlog assume $\ell > h + 1$.

Query Q for ranks $1, n/h, 2n/h, \ldots, n$.

Create Q' from output of queries. Use same intervals as those in Q.

Pruning/Reducing Analysis

$Q = \{q_1, q_2, \ldots, q_\ell\}$ and wlog assume $\ell > h + 1$.
Query Q for ranks $1, \frac{n/h}{,} \frac{2n}{h}, \ldots, n$.
$Q' = \{q'_1, q'_2, \ldots, q'_{h+1}\}$

Suppose $q'_i = q_a$ and $q'_{i+1} = q_b$ for some $a < b$.
$I(q_a) \subseteq [in/h - \epsilon n, in/h + \epsilon n]$ and
$I(q_b) \subseteq [(i+1)n/h - \epsilon n, (i+1)n/h + \epsilon n]$

Pruning/Reducing Analysis

$Q = \{q_1, q_2, \ldots, q_\ell\}$ and wlog assume $\ell > h + 1$.
Query Q for ranks $1, \frac{n/h}{,} \frac{2n}{h}, \ldots, n$.
$Q' = \{q'_1, q'_2, \ldots, q'_{h+1}\}$

Suppose $q'_i = q_a$ and $q'_{i+1} = q_b$ for some $a < b$.
$I(q_a) \subseteq [in/h - \epsilon n, in/h + \epsilon n]$ and
$I(q_b) \subseteq [(i+1)n/h - \epsilon n, (i+1)n/h + \epsilon n]$

Therefore,

$$\begin{aligned}
\mathbf{rmax}_{Q'}(q'_{i+1}) - \mathbf{rmin}_{Q'}(q'_i) &\leq (i+1)n/h + \epsilon n - (in/h - \epsilon n) \\
&\leq 2\epsilon n + n/h \\
&\leq 2(\epsilon + 1/(2h))n.
\end{aligned}$$

Merge and Reduce Streaming Quantiles

Stream: $a_1, a_2, \ldots, a_n$ and given $\epsilon \in (0,1)$

Want to maintain ϵ-approximate quantile summary.

$O(\frac{1}{\epsilon}\log^2 n)$ space algorithm based on reduce and merge.

- Come up with a solution as if the whole stream is available offline
- Show how it can implemented in small space in streaming setting.

Merge and Reduce for Streaming Quantiles

Stream: $a_1, a_2, \ldots, a_n$ and given $\epsilon \in (0, 1)$

- Imagine a rooted binary tree with $a_1, a_2, \ldots, a_n$ as leaves in that order (not sorted)
- At each internal node v let S_v be leaves under v.
- Compute a summary Q_v for S_v bottom up. Q_r is output where r is root. Summary at leaf is optimal simply stores element.
- To compute Q_v with children a, b Merge Q_a and Q_b and Prune to size $h + 1$
- Guarantees that Q_r has size $h + 1$

Merge and Reduce for Streaming Quantiles

Stream: $a_1, a_2, \ldots, a_n$ and given $\epsilon \in (0, 1)$

- Imagine a rooted binary tree with $a_1, a_2, \ldots, a_n$ as leaves in that order (not sorted)
- At each internal node v let S_v be leaves under v.
- Compute a summary Q_v for S_v bottom up. Q_r is output where r is root. Summary at leaf is optimal simply stores element.
- To compute Q_v with children a, b Merge Q_a and Q_b and Prune to size $h + 1$
- Guarantees that Q_r has size $h + 1$

How should we choose h to ensure ϵ-approx Q_r?

Merge and Reduce for Streaming Quantiles

If each leaf summary has error ϵ' then Merging does not increase error but Pruning adds $1/(2h)$ at each level. Hence ϵ_r at root with depth d satisfies

$$\epsilon_r \leq \epsilon' + d/(2h) \leq \epsilon' + \log n/(2h)$$

Merge and Reduce for Streaming Quantiles

If each leaf summary has error ϵ' then Merging does not increase error but Pruning adds $1/(2h)$ at each level. Hence ϵ_r at root with depth d satisfies

$$\epsilon_r \leq \epsilon' + d/(2h) \leq \epsilon' + \log n/(2h)$$

To ensure $\epsilon_r \leq \epsilon$ we set $h = \Omega(\frac{1}{\epsilon} \log n)$. Hence each summary size is $O(\frac{1}{\epsilon} \log n)$ numbers

Merge and Reduce for Streaming Quantiles

If each leaf summary has error ϵ' then Merging does not increase error but Pruning adds $1/(2h)$ at each level. Hence ϵ_r at root with depth d satisfies

$$\epsilon_r \leq \epsilon' + d/(2h) \leq \epsilon' + \log n/(2h)$$

To ensure $\epsilon_r \leq \epsilon$ we set $h = \Omega(\frac{1}{\epsilon} \log n)$. Hence each summary size is $O(\frac{1}{\epsilon} \log n)$ numbers

How can we implement offline algorithm in streaming setting and how much space does it require?

Merge and Reduce for Streaming Quantiles

To ensure $\epsilon_r \leq \epsilon$ we set $h = \Omega(\frac{1}{\epsilon} \log n)$. Hence each summary size is $O(\frac{1}{\epsilon} \log n)$ numbers

How can we implement offline algorithm in streaming setting and how much space does it require?

Only Q_r needed so sufficient to keep only those summaries in the "imaginary" binary tree that suffice to create Q_r. Suffices to keep $O(d)$ summaries where d is depth. Hence total space is $O(\frac{1}{\epsilon} \log^2 n)$.

Merge and Reduce for Streaming Quantiles

To ensure $\epsilon_r \leq \epsilon$ we set $h = \Omega(\frac{1}{\epsilon} \log n)$. Hence each summary size is $O(\frac{1}{\epsilon} \log n)$ numbers

How can we implement offline algorithm in streaming setting and how much space does it require?

Only Q_r needed so sufficient to keep only those summaries in the "imaginary" binary tree that suffice to create Q_r. Suffices to keep $O(d)$ summaries where d is depth. Hence total space is $O(\frac{1}{\epsilon} \log^2 n)$.

Need to know n in advance to set h. Otherwise use squaring.

Handling unknown n

Length of stream not known. Use some standard tricks/ideas.

- Start by assuming an estimate n_0 for n where n_0 is some constant. Create data structure assuming $\leq n_0$ items.
- when n exceeds current estimate *double* estimate and start a new data structure with new estimate
- or when n exceeds current estimate *square* estimate

Handling unknown n

Length of stream not known. Use some standard tricks/ideas.

- Start by assuming an estimate n_0 for n where n_0 is some constant. Create data structure assuming $\leq n_0$ items.
- when n exceeds current estimate *double* estimate and start a new data structure with new estimate
- or when n exceeds current estimate *square* estimate

Observation: When estimate changes we create new data structure and freeze past data structures. Error of each data structure is bounded by ϵ. To answer queries we can Merge the data structures without increasing error.

Handling unknown n

Length of stream not known. Use some standard tricks/ideas.

- Start by assuming an estimate n_0 for n where n_0 is some constant. Create data structure assuming $\leq n_0$ items.
- when n exceeds current estimate *double* estimate and start a new data structure with new estimate
- or when n exceeds current estimate *square* estimate

Observation: When estimate changes we create new data structure and freeze past data structures. Error of each data structure is bounded by ϵ. To answer queries we can Merge the data structures without increasing error.
Observation: Since space is poly-logarithmic in n when n is known, squaring strategy guarantees only constant factor loss even when n is not known.

Improvements

Instead of binary tree all the way use at first level $1/\epsilon$ nodes. Depth goes to $\log(\epsilon n)$ and hence space improves to $O(\frac{1}{\epsilon}\log^2(\epsilon n))$.

Improvements

Instead of binary tree all the way use at first level $\mathbf{1/\epsilon}$ nodes. Depth goes to $\mathbf{\log(\epsilon n)}$ and hence space improves to $\mathbf{O(\frac{1}{\epsilon}\log^2(\epsilon n))}$.

[Greenwald-Khanna] gave a more involved scheme that achieves $\mathbf{O(\frac{1}{\epsilon}\log(\epsilon n))}$ space. Near-optimal.

Part III

Multipass Selection

Multipass Selection

Selection in multiple passes:

- 1-pass requires and can be done in $O(n)$ space
- $O(1)$ space. $O(\log n)$ suffices. Implement Quick Select in $O(1)$ space.
- p passes? $O(n^{1/p}\text{polylog}(n))$ space suffices. Hence $O(\sqrt{n}\log n)$ for 2 passes. [Munro-Paterson 1980]

Multipass Selection

Selection in multiple passes:

- 1-pass requires and can be done in $O(n)$ space
- $O(1)$ space. $O(\log n)$ suffices. Implement Quick Select in $O(1)$ space.
- p passes? $O(n^{1/p}\text{polylog}(n))$ space suffices. Hence $O(\sqrt{n}\log n)$ for 2 passes. [Munro-Paterson 1980]

Goal: Derive p-pass algorithm via approximate quantile summary

$p = 2$ case

Goal: Selection of rank k element in 2-passes using $\tilde{O}(\sqrt{n})$ space

Pass 1:

- Store $\epsilon = 1/\sqrt{n}$-approximate summary. Space is $\tilde{O}(1/\epsilon) = \tilde{O}(\sqrt{n})$.
- Summary allows to find two numbers $a < b$ such that $\text{rank}(a) \geq k - O(\epsilon)n$ and $\text{rank}(b) \leq k + O(\epsilon)n$

$p = 2$ case

Goal: Selection of rank k element in 2-passes using $\tilde{O}(\sqrt{n})$ space

Pass 1:

- Store $\epsilon = 1/\sqrt{n}$-approximate summary. Space is $\tilde{O}(1/\epsilon) = \tilde{O}(\sqrt{n})$.
- Summary allows to find two numbers $a < b$ such that $\text{rank}(a) \geq k - O(\epsilon)n$ and $\text{rank}(b) \leq k + O(\epsilon)n$

Pass 2:

- Store all numbers between a and b; $O(\sqrt{n})$ numbers.
- Compute exact rank of a and b. How?
- Find rank k element from stored elements and knowing rank of a, b. How?

General p

Goal: Selection of rank k element in p-passes using $\tilde{O}(n^{1/p})$ space

Pass 1:

- Store $\epsilon = 1/n^{1/p}$-approximate summary. Space is $\tilde{O}(1/\epsilon) = \tilde{O}(n^{1/p})$.
- Summary allows to find two numbers $a < b$ such that $\text{rank}(a) \geq k - O(n^{1-1/p})$ and $\text{rank}(b) \leq k + O(n^{1-1/p})$
- In subsequent passes one can restrict attention to numbers between a and b. Only $n^{1-1/p}$ of them. Hence in one pass reduce to $n^{1-1/p}$ numbers.

General p

Goal: Selection of rank k element in p-passes using $\tilde{O}(n^{1/p})$ space

Pass 1:

- Store $\epsilon = 1/n^{1/p}$-approximate summary. Space is $\tilde{O}(1/\epsilon) = \tilde{O}(n^{1/p})$.
- Summary allows to find two numbers $a < b$ such that $\text{rank}(a) \geq k - O(n^{1-1/p})$ and $\text{rank}(b) \leq k + O(n^{1-1/p})$
- In subsequent passes one can restrict attention to numbers between a and b. Only $n^{1-1/p}$ of them. Hence in one pass reduce to $n^{1-1/p}$ numbers.

After $(p-1)$ passes we have $n^{1/p}$ numbers left and we can store all of them in p'th pass and solve exactly.

Random Order Streams

$\Omega(n)$ lower bound for Selection in adversarial setting. Can we do better if we assume non-worst case input?

Random Order Stream Model:

- Adversary picks some input.
- Algorithm sees a random permutation of the input. Adversary power is weakened.
- Several interesting results in this model.

For Exact Selection in random order streams.

- $O(\sqrt{n})$ space in 1-pass suffices with high probability. [Munro-Paterson]
- $O(\log \log n)$ passes suffice with $O(\text{poly}(\log n))$ space whp. [Guha-MacGregor]