CMSC5706 Topics in Theoretical Computer Science

Week 3: Streaming and Sketching

Instructor: Shengyu Zhang
Map

- Motivations and model
- Problem 1: Missing numbers
- Problem 2: Count-Min sketch
- Lower bounds
  - Communication complexity
Motivations

- Big mass of data.
- Data comes as a stream.
  - Cannot see future data.
- Relatively small space. “sketch”
  - Cannot store past data
- Need to process each item fast.
  - Quick update time.
- Examples: Phone calls, Internet packets, satellite pictures, …
Problem 1: Missing numbers

- A set of numbers $S = \{1,2, \ldots, n\}$
- $n - 1$ of them come in a stream $x_1, x_2, \ldots, x_{n-1}$; one number is missing.
  
  $3, 25, 6, 19, 1, 10, \ldots$

- Task: identify which one is missing.
  - Using small space.
A simple algorithm

- **Maintain the sum** of the input numbers.

- $\textit{sum} = 0$

- **for** $i = 1$ to $n - 1$

  \[ \textit{sum} = \textit{sum} + x_i \]

- **return** $\frac{n(n+1)}{2} - \textit{sum}$
Space complexity

- \textit{sum} is at most $\frac{n(n+1)}{2}$ during the algorithm.
- Thus it takes at most $\log_2 \frac{n(n+1)}{2} = O(\log_2 n)$ bits to write it down.
- Space complexity: $O(\log_2 n)$.
- Much smaller than storing the whole stream, which takes at least $O(n \log n)$. 
More complicated

- Now the task gets harder.
- \( n - 2 \) of them come in a stream
  \( x_1, x_2, \ldots, x_{n-2} \), two numbers are missing.

  3, 25, 6, 19, 1, 10, ...

- Task: identify which two are missing.
  - Using small space.
First try

- Maintain the sum and product of the input numbers.

- \( \text{sum} = 0; \text{product} = 1 \)
- \( \textbf{for} \ i = 1 \ \textbf{to} \ n - 2 \)
  
  \[
  \text{sum} = \text{sum} + x_i \\
  \text{product} = \text{product} \cdot x_i
  \]
- \( a = \frac{n(n+1)}{2} - \text{sum}, \ b = n!/\text{product} \)
- solve equations \( x + y = a, x \cdot y = b \)
- \textbf{return} \ (x, y)
Problem and solution

- Issue: \textit{product} is at least \((n - 2)!\).
- Thus even writing down the number needs 
  \(\log_2(n - 2)! = \Theta(n \log n)\) bits.
  - Too much compared to \(O(\log n)\) before.

- How to do?
Improvement

- Note that we don’t need to maintain product.
- We can maintain anything, as long as finally we can reconstruct the solution from the stored results.
- One summary that is much smaller than product: sum of squares.
- Recall: \(1^2 + 2^2 + \cdots + n^2 = \frac{n(n+1)(2n+1)}{6}\)
Improvement

- Maintain the **sum** and **sum of squares** of the input numbers.

- \( sum = 0 \); \( sos = 0 \)

- **for** \( i = 1 \) to \( n - 2 \)
  
  \[ sum = sum + x_i \]
  \[ sos = sos + x_i^2 \]

- \( a = \frac{n(n+1)}{2} - sum \), \( b = \frac{n(n+1)(2n+1)}{6} - sos \)

- solve equations \( x + y = a \), \( x^2 + y^2 = b \).

- **return** \((x, y)\)
Space complexity

- $sos$ is at most $\frac{n(n+1)(2n+1)}{6}$ during the algorithm.

- Thus it takes at most $\log_2 \frac{n(n+1)(2n+1)}{6} = O(\log_2 n)$ bits to write it down.

- Space complexity: $O(\log_2 n)$. 
Further question

- Now assume that the numbers are from an arbitrary set \( S = \{s_1, s_2, \ldots, s_n\} \).
- \( n - k \) of them come in a stream \( x_1, x_2, \ldots, x_{n-k} \); \( k \) numbers are missing.

Task: identify which \( k \) are missing.
  - Using small space.
First try

- Maintain $\sum_i x_i, \sum_i x_i^2, \ldots, \sum_i x_i^k$ of the input numbers.

- $sum_1 = 0; sum_2 = 0; \ldots; sum_k = 0$

- for $i = 1$ to $n - k$
  for $d = 1$ to $k$
    $sum_d = sum_d + x_i^d$

- solve system of equations

- return $(y_1, \ldots, y_k)$
Space complexity

- \( sum_d \) is at most \( O(n^d) \) during the algorithm.
- Thus it takes at most \( O(k \log_2 n^k) = O(k^2 \log_2 n) \) bits to write it down.

- Space complexity: \( O(k^2 \log_2 n) \).
Problem 2: high frequency estimation

- Consider an array $F[1..n]$ of size $n$.
- Items like $(i_1, +), (i_2, -), ..., (i_T, +)$ come in a stream.
  
  $(3, +), (3, +), (2, +), (3, -), ...$

- $F[i] ++$ when $(i, +)$ comes, and $F[i] --$ when $(i, -)$ comes
  
  Assumption: $F[i] \geq 0$ all the time.

- Task: Answer queries like “what is $F[18]$”? 
Approximation and error

- Unlike the previous algorithm, here deterministic algorithm needs a lot of space.
- But if we allow
  - approximation: only estimate $F[i]$ up to certain precision
  - error: algorithm fails with some small probability
- then we’ll have an efficient randomized algorithm.
- Pick $\log(1/\delta)$ hash functions $h_j: [n] \rightarrow [e/\epsilon]$ 
  - uniformly at random from a family of pairwise independent hash functions.
- $e/\epsilon \ll n$, so it’s space efficient.
- For each $i \in [n]$, different $h_j$’s map it to different “buckets”.
- Idea: only maintain counters for buckets.
Algorithm

- for $j = 1$ to $\log(1/\delta)$
  
  for $d = 1$ to $e/\epsilon$
  
  count$(j, k) = 0$

- for $t = 1$ to $T$
  
  if item $t$ is $(i, +/-)$
  
    for $j = 1$ to $\log(1/\delta)$
    
    count$\left(j, h_j(i)\right) = \begin{cases} + & \text{if item } t \text{ is } (i, +) \\ - & \text{if item } t \text{ is } (i, -) \end{cases}$

- On query $F[i]$: return $F'[i] = \min_j \text{count} \left(j, h_j(i)\right)$
Guarantee

- At any time of query:
- Define $\|F\| = \sum_i F[i]$

- Theorem.
  - $F'[i] \geq F[i]$
  - $F'[i] \leq F[i] + \epsilon \|F\|$ with probability $\geq 1 - \delta$. 
Analysis

- $F'[i] \geq F[i]$ is easy:
- Any time when $F[i]$ increases by 1, we increase $\text{count}(j, h_j(i))$ for each $j$.
- Thus $\min_j \text{count}(j, h_j(i))$ also increases by 1.
- Thus we never miss any increment.
Analysis

- Next: $F'[i] \leq F[i] + \epsilon \|F\|$ with prob. $\geq 1 - \delta$.
- $X_{ji}$: the contribution of items other than $i$ to count $(j, h_j(i))$.

Claim. $\mathbb{E}[X_{ji}] = \frac{\epsilon}{e} (\|F\| - F[i]) \leq \frac{\epsilon}{e} \|F\|.$

Proof. For each fixed item $i' \neq i$, the probability of $h_j(i') = h_j(i)$ is $\epsilon/e$.

There are $\|F\| - F[i]$ many items $i' \neq i$ (counting multiplicity), thus $\mathbb{E}[X_{ji}] = \frac{\epsilon}{e} (\|F\| - F[i])$. 
\begin{itemize}
  \item \[ \Pr[F'[i] > F[i] + \epsilon \|F\|] = \]
  \[ \Pr[F[i] + X_{ji} > F[i] + \epsilon \|F\|, \forall j] \]
  \item \( F'[i] = F[i] + X_{ji} \) by definition
  \item \( \min_j \text{ count } (j, h_j(i)) > F[i] + \epsilon \|F\| \)
    \[ \Leftrightarrow F[i] + X_{ji} > F[i] + \epsilon \|F\|, \forall j \]
  \item \[ \Pr[F[i] + X_{ji} > F[i] + \epsilon \|F\|, \forall j] \]
    \[ = \Pr[F[i] + X_{ji} > F[i] + \epsilon \|F\|]^{\log 1/\delta} \]
\end{itemize}

because different \( h_j \)'s are independently chosen.
\[ \Pr[F[i] + X_{ji} > F[i] + \epsilon \|F\|] = \Pr[X_{ji} > \epsilon \|F\|] \]

Recall: \[ \mathbb{E}[X_{ji}] = \frac{\epsilon}{e} (\|F\| - F[i]) \leq \frac{\epsilon}{e} \|F\| \]

By Markov’s inequality,
\[ \Pr[X_{ji} > \epsilon \|F\|] \leq \Pr[X_{ji} > e\mathbb{E}[X_{ji}]] < 1/e \]

Putting everything together,
\[ \Pr[F'[i] > F[i] + \epsilon \|F\|] \leq \left( \frac{1}{e} \right)^{\log \frac{1}{\delta}} = \delta \]
Lower bounds

- Theorem. In order to estimate $F[i]$ within an error of $\epsilon \|F\|$ with probability $2/3$, one needs to use $\Omega \left( \frac{1}{\epsilon} \right)$ space.
- Proof. We will use one-way communication complexity.
Communication complexity

- Two parties, Alice and Bob, jointly compute a function $f$ on input $(x, y)$.
  - $x$ known only to Alice and $y$ only to Bob.
- **Communication complexity**: how many bits are needed to be exchanged?
One-way communication complexity

- Theorem. Index function needs $\Omega(n)$ communication bits.
  - even for randomized protocols.
Lower bound

- Theorem. In order to estimate $F[i]$ within an error of $\epsilon \|F\|$ with probability 2/3, one needs to use $\Omega\left(\frac{1}{\epsilon}\right)$ space.

- **Proof.** Given an Index problem input $(x, i)$, with $n = 1/2\epsilon$.

- Let $F$ be: $F[i] = 2x_i$ for $i = 1, \ldots, n$, and $F[0] = 2 \cdot |\{i \in [n]: x_i = 0\}|$.

- $\|F\| = 2n = 1/\epsilon$. Thus $\epsilon \|F\| = 1$. 
If one can estimate $F[i]$ within error $\epsilon \|F\| = 1$ using space $s$, then

Alice can use this way to transmit the space to Bob.

- communication: $s$ bits.

Bob then gets $F'[i]$ which differ from $F[i]$ by 1.

Bob can then determine whether $x_i = 0$ or $x_i = 1$.

Thus the communication lower bound implies $s = \Omega(n) = \Omega \left( \frac{1}{\epsilon} \right)$, as desired.
One thing left

- Pairwise independent hash family
- A family of functions $H = \{h|h : N \rightarrow M\}$ is pairwise independent if the following two conditions hold when we pick $h \in H$ uniformly at random:
  - $\forall x \in N$, the random variable $h(x)$ is uniformly distributed in $M$
  - $\forall x_1 \neq x_2 \in N$, the random variables $h(x_1)$ and $h(x_2)$ are independent.
Note that the condition is equivalent to the following.

For any two different $x_1 \neq x_2 \in N$, and any $y_1, y_2 \in M$, it holds that

$$\Pr_{h \in H}[h(x_1) = y_1 \text{ and } h(x_2) = y_2] = \frac{1}{|M|^2}$$
There is an easy construction of the pairwise independent hash function family.

Let $p$ be a prime, and define

$$h_{a,b}(x) = (ax + b) \mod p$$

Define family

$$H = \{h_{a,b} : 0 \leq a, b \leq p - 1\}$$

Theorem. $H$ is a family of pairwise independent hash functions.
It is enough to show that
\[ \Pr_{h \in H}[h(x_1) = y_1 \text{ and } h(x_2) = y_2] = 1/p^2 \]

For any \( x_1 \neq x_2, y_1 \) and \( y_2 \), there is a unique pair \((a, b)\) s.t. \( h_{a,b}(x_1) = y_1 \) and \( h_{a,b}(x_2) = y_2 \).

Indeed, this is just
\[
ax_1 + b = y_1 \mod p \\
ax_2 + b = y_2 \mod p
\]

which has a unique solution for \((a, b)\) because \[
\begin{vmatrix} x_1 & 1 \\ x_2 & 1 \end{vmatrix} \neq 0 \text{ due to } x_1 \neq x_2.\]