LECTURE 10

Last time

• Multipurpose sketches
• Count-min and count-sketch
• Range queries, heavy hitters, quantiles

Today

• Limitations of streaming algorithms
• Communication complexity

HW3, project proposal resubmission due Thursday
Recall: Frequency Moments Estimation

Input: a stream $\langle a_1, a_2, ..., a_m \rangle \in [n]^m$

• The frequency vector of the stream is $f = (f_1, ..., f_n)$, where $f_i$ is the number of times $i$ appears in the stream

• The $p$-th frequency moment is $F_p = \|f\|^p = \sum_{i=1}^{n} f_i^p$

  $F_0$ is the number of nonzero entries of $f$ (# of distinct elements)

  $F_1 = m$ (# of elements in the stream)

  $F_2 = \|f\|^2$ is a measure of non-uniformity

  used e.g. for anomaly detection in network analysis

  $F_\infty = \max_i f_i$ is the most frequent element

  We obtained streaming algorithms for $F_0, F_1, F_2$.

  What about $F_3$ to $F_\infty$?
Communication Complexity

A Method for Proving Lower Bounds
(Randomized) Communication Complexity

Goal: minimize the number of bits exchanged.

- **Communication complexity of a protocol** is the maximum number of bits exchanged by the protocol.
- **Communication complexity of a function** $C$, denoted $R(C)$, is the communication complexity of the best protocol for computing $C$. 

Partially based on slides by Eric Blais
Example: Set Disjointness $DISJ_k$

**Theorem** [Kalyanasundaram Schmitger 92, Razborov 92]

$$R(DISJ_k) \geq \Omega(k) \text{ for all } k \leq \frac{n}{2}.$$ 

**Input:** $S \subseteq [n], |S| = k.$

**Input:** $T \subseteq [n], |T| = k.$

Compute $DISJ_k(S, T)$

$$= \begin{cases} 
\text{accept} & \text{if } S \cap T = \emptyset \\
\text{reject} & \text{otherwise}
\end{cases}$$

**Example: 1101000101110101010101100...**
One-Way Communication Complexity

**Input:** $x$

**Shared random string:**

1101000101110101110101010110...

**Goal:** minimize the number of bits Alice sends to Bob.

One-way communication complexity of a function $C$, denoted $R\leftarrow(C)$, is the communication complexity of the best one-way protocol for computing $C$. 

**Input:** $y$

**Compute $C(x, y)$**
3-Player One-Way Communication Complexity

**Goal:** minimize $|m_1| + |m_2|$.
- Require correct output w.p. at least 2/3 over the random string.
Converting Streaming Algorithm to CC Protocol

Let \( \mathcal{P} \) be a streaming problem.

- Suppose there is a transformation \( x \rightarrow s_1, y \rightarrow s_2, z \rightarrow s_3 \) such that \( \mathcal{P}(s_1 \circ s_2 \circ s_3) \) suffices to compute \( C(x, y, z) \)

An \( s \)-bit algorithm \( A \) for \( \mathcal{P} \) gives a \( 2s \)-bit protocol for \( C \)

- Alice runs \( A \) on \( s_1 \) and sends memory state, \( m_1 \), to Bob
- Bob instantiates \( A \) with \( m_1 \), runs \( A \) on \( s_2 \), sends memory state, \( m_2 \), to Carol
- Carol instantiates \( A \) with \( m_2 \), runs \( A \) on \( s_3 \) to get \( \mathcal{P}(s_1 \circ s_2 \circ s_3) \) and computes \( C(x, y, z) \)

Based on Andrew McGregor's slides: https://people.cs.umass.edu/~mcgregor/711S18/lowerbounds-1.pdf
Converting Streaming Algorithm to CC Protocol

Let $\mathcal{P}$ be a streaming problem.

- Suppose there is a transformation $x \to s_1, y \to s_2, z \to s_3$ such that $\mathcal{P}(s_1 \circ s_2 \circ s_3)$ suffices to compute $C(x, y, z)$

<table>
<thead>
<tr>
<th>Input: $x$</th>
<th>$s_1$</th>
<th>$s_2$</th>
<th>$s_3$</th>
<th>Input: $y$</th>
<th>Input: $z$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$m_1$</td>
<td></td>
<td></td>
<td></td>
<td>$m_2$</td>
<td></td>
</tr>
</tbody>
</table>

An $s$-bit algorithm $A$ for $\mathcal{P}$ gives a $2s$-bit protocol for $C$

- If there are $p$ players than the protocol uses $(p - 1)s$ bits

- A lower bound $L$ for computing $C$ implies $b = \Omega \left( \frac{L}{p} \right)$

Based on Andrew McGregor’s slides: https://people.cs.umass.edu/~mcgregor/711S18/lowerbounds-1.pdf
A lower bound using CC method

Approximating $F_\infty$
**Application: Approximating $F_{\infty}$**

<table>
<thead>
<tr>
<th>Theorem</th>
</tr>
</thead>
<tbody>
<tr>
<td>Every algorithm that computes $4/3$-approximation of $F_{\infty}$ (w.p. $\geq 2/3$) needs $\Omega(n)$ space.</td>
</tr>
</tbody>
</table>

**Proof:** Reduction from Set Disjointness

On input $x, y \in \{0,1\}^n$, players generate $s_1 = \{j: x_j = 1\}$ and $s_2 = \{j: y_j = 1\}$

**Example:**

\[
\begin{pmatrix}
0 & 0 & 1 & 1 & 0 & 0 \\
1 & 0 & 1 & 0 & 1 & 0
\end{pmatrix}
\rightarrow
\langle 3,4; 1,3,5 \rangle
\]

- Then $F_{\infty} = 1$ if $x, y$ represent disjoint sets, and $F_{\infty} = 2$, otherwise.

**Output $\leq 4/3$**

**Output $\geq 3/2$**

- An $s$-space algorithm implies an $s$-bit protocol:

\[
s = \Omega(n)
\]

by communication complexity of Set Disjointness

Based on Andrew McGregor’s slides: https://people.cs.umass.edu/~mcgregor/711S18/lowerbounds-1.pdf
A lower bound using $CC$ method

Computing the median of a stream
Index

• Alice gets an $n$-bit string $x$, and Bob gets an index $j \in [n]$.

• Define $\text{Index}(x, j) = x_j$.

• One-way communication complexity of $\text{Index}(x, j)$ is $\Omega(n)$.
**Application: Finding the Median of a Stream**

### Theorem

Every algorithm that computes the median of an \((2n - 1)\)-element stream exactly (w.p. \(\geq 2/3\)) needs \(\Omega(n)\) space.

**Proof:** Reduction from Index.

- On input \(x \in \{0,1\}^n\), Alice generates \(s_1 = \{2i + x_i: i \in [n]\}\)
  
  **Example:** \[0011011\] \(\rightarrow\) \(\langle 2,4,7,9,10,13,15\rangle\)

- On input \(j \in [n]\), Bob generates
  
  \[s_2 = \{n - j\ \text{copies of} \ 0\ \text{and} \ j - 1\ \text{copies of} \ 2n + 2\}\]

  **Example:** \(j = 2\) \(\rightarrow\) \(\langle 0,0,0,0,0,16\rangle\)

- Then \(median(s_1 \circ s_2) = 2j + x_j\) and \(\text{Index}(x, j) = 2j + x_j \ mod \ 2\)

- An \(s\)-space algorithm implies an \(s\)-bit protocol:

\[
s = \Omega(n)
\]

by 1-way communication complexity of \(\text{Index}\)

---

Based on Andrew McGregor’s slides: https://people.cs.umass.edu/~mcgregor/711S18/lowerbounds-1.pdf
A lower bound using CC method

Approximating Frequency Moments

[Bar-Yossef, Jayram, Kumar, Sivakumar 04]
Multi-party Set Disjointness

- Consider a $p \times n$ binary matrix $M$ where each column has weight 0, 1 or $p$

Example:

$$\begin{pmatrix}
0 & 0 & 1 & 1 & 0 & 0 \\
1 & 0 & 1 & 0 & 1 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 1
\end{pmatrix}$$

- The input of player $i$ is row $i$ of $M$

$$DISJ^{(p)}(M) = \begin{cases} 
0 & \text{if there is a column of 1s} \\
1 & \text{otherwise}
\end{cases}$$

- Communication complexity of $DISJ^{(p)}(M)$ is $\Omega\left(\frac{n}{p}\right)$

Based on Andrew McGregor’s slides: https://people.cs.umass.edu/~mcgregor/711S18/lowerbounds-1.pdf
Application: Frequency Moments for $k > 2$

**Thm.** Every algorithm that 2-approximates $F_k$ (w.p. $\geq 2/3$) needs $\Omega\left(n^{1-\frac{2}{k}}\right)$ space

**Proof:** Reduction from multi-party Set Disjointness

- On input $M \in \{0,1\}^{p \times n}$, player $i$ generates $s_i = \{j: M_{ij} = 1\}$

  **Example:**
  $$
  \begin{pmatrix}
  0 & 0 & 1 & 1 & 0 & 0 \\
  1 & 0 & 1 & 0 & 1 & 0 \\
  0 & 0 & 1 & 0 & 0 & 0 \\
  0 & 0 & 1 & 0 & 0 & 1 \\
  \end{pmatrix}
  \rightarrow \langle 3,4; 1,3,5; 3; 3,6 \rangle
  $$

- If all columns have weight 0 or 1 then $F_k = \sum_{i=1}^{n} f_i^k \leq n$
- If there is a column of weight $p$ then $F_k \geq p^k$
- A 2-approximation of $F_k$ distinguishes the cases if $p^k > 4n \iff p > (4n)^{\frac{1}{k}}$
- An $s$-space algorithm implies $s(p - 1)$-bit protocol:
  $$
  s = \Omega\left(\frac{n}{p^2}\right) = \Omega\left(\frac{n}{(4n)^{\frac{2}{k}}}\right) = \Omega\left(n^{1-\frac{2}{k}}\right)
  $$

  by communication complexity of $DISJ^{(p)}$ for constant $k$

Based on Andrew McGregor’s slides: https://people.cs.umass.edu/~mcgregor/711S18/lowerbounds-1.pdf
A lower bound using CC method

Distinct Elements
Alice and Bob get $n$-bit strings $x$ and $y$, respectively.

Hamming distance $Ham(x, y)$ is the number of positions on which $x$ and $y$ differ.

Output: $Ham(x, y)$ with additive error $\sqrt{n}$ w.p. $\geq 2/3$

Communication complexity of $Ham(x, y)$ is $\Omega(n)$ even when $|x|$ and $|y|$ are known to both players.
**Application: Distinct Elements**

**Thm.** Every algorithm \((1 + \varepsilon)\)-approximating \(F_0\) (w.p. \(\geq 2/3\)) needs \(\Omega\left(\frac{1}{\varepsilon^2}\right)\) space

**Proof:** Reduction from Gap Hamming

On input \(x, y \in \{0,1\}^n\), players generate \(s_1 = \{j: x_j = 1\}\) and \(s_2 = \{j: y_j = 1\}\)

**Example:**

\[
\begin{pmatrix}
0 & 0 & 1 & 1 & 0 & 0 \\
1 & 0 & 1 & 0 & 1 & 0
\end{pmatrix}
\rightarrow (3,4; 1,3,5)
\]

- Then \(2F_0 = |x| + |y| + \text{Ham}(x, y)\)
- When \(|x|\) is known to Bob, \((1 + \varepsilon)\)-approximation of \(F_0\) gives an additive approximation to \(\text{Ham}(x, y)\)

\[
\varepsilon \cdot \frac{|x| + |y| + \text{Ham}(x, y)}{2} \leq \varepsilon n \leq \sqrt{n}
\]

- An \(s\)-space algorithm implies an \(s\)-bit protocol:

\[
s = \Omega(n) = \Omega\left(\frac{1}{\varepsilon^2}\right)
\]

by communication complexity of *Gap Hamming*

---

Based on Andrew McGregor’s slides: https://people.cs.umass.edu/~mcgregor/711S18/lowerbounds-1.pdf