§2.2. Streaming Model

Qin Zhang
Data Streams

- Continuous streams of data elements (massive possibly unbounded, rapid, time-varying)

- Some examples:
  1. network monitoring and traffic engineering
  2. financial data: stock data,
  3. web logs and click streams
  4. sensor networks, RFID tags
  5. ad auction
  6. flight logs on tape
  etc.

(next 4 slides, in courtesy of Jeff Phillips)
Network Routers

Internet Router

- data per day: at least 1 Terabyte
- packet takes 8 nanoseconds to pass through router
- few million packets per second

What statistics can we keep on data? For example, want to detect anomalies for security.
Cell phones connect through switches

- each message 1000 Bytes
- 500 million calls / day
- 1 Terabyte per month

Search for characteristics for dropped calls?
Serving Ads on web Google, Yahoo!, Microsoft

- Yahoo.com viewed 77 trillion times
- 2 million / hour
- Each page serves ads; which ones?

How to update ad delivery model?
All airplane logs over Washington, DC

- About 500 - 1000 flights per day.
- 50 years, total about 9 million flights
- Each flight has trajectory, passenger count, control dialog.

Stored on Tape. Can only make 1 (or $O(1)$) pass!
What statistics can be gathered?
Data Stream Management Systems (DSMS)

- A comparison between traditional DBMS and DSMS

<table>
<thead>
<tr>
<th>DBMS</th>
<th>DSMS</th>
</tr>
</thead>
<tbody>
<tr>
<td>● Persistent relations</td>
<td>● Transient streams (and persistent relations)</td>
</tr>
<tr>
<td>● One-time queries</td>
<td>● Continuous queries</td>
</tr>
<tr>
<td>● Random access</td>
<td>● Sequential access</td>
</tr>
<tr>
<td>● Access plan determined by query processor and physical DB design</td>
<td>● Unpredictable data characteristics and arrival patterns</td>
</tr>
</tbody>
</table>
The Big Picture of DSMS

Ad-hoc Query

Continuous monitoring

Standing query
Many challenges: System, Programming Language, etc.

This course focus on: Algorithmic Challenges

(More on the algorithms not proofs)

Read Complete Book Section 23.4 for how to write SQL-like queries in DSMS
The abstract model and challenges

- **The data stream model** (Alon, Matias and Szegedy 1996)

  \[ a_n \quad \cdots \quad a_2 \quad a_1 \]

- **Why hard?**
  1. Lack of space ... Cannot store everything.
  2. Need fast processing.
  3. Can only scan the data once.
Why hard? You do see everything but then “forget”!

- **Game 1**: A sequence of numbers
Why hard? You do see everything but then “forget”!

- **Game 1**: A sequence of numbers

52
Why hard? You do see everything but then “forget”!

- **Game 1**: A sequence of numbers

  ![45]
Why hard? You do see everything but then “forget”!

- **Game 1**: A sequence of numbers

18
Why hard? You do see everything but then “forget”!

- **Game 1:** A sequence of numbers

  23
Why hard? You do see everything but then “forget”!

- **Game 1**: A sequence of numbers

17
Why hard? You do see everything but then “forget”!

- **Game 1:** A sequence of numbers

   41
Why hard? You do see everything but then “forget”!

- **Game 1**: A sequence of numbers

  33
Why hard? You do see everything but then “forget”!

- **Game 1**: A sequence of numbers

  29
Why hard? You do see everything but then “forget”!

- **Game 1**: A sequence of numbers

  ![49]
Why hard? You do see everything but then “forget”!

- **Game 1**: A sequence of numbers

  12
Why hard? You do see everything but then “forget”!

- **Game 1**: A sequence of numbers

35
Why hard? You do see everything but then “forget”!

- **Game 1**: A sequence of numbers

  **Q**: What’s the median?
Why hard? You do see everything but then “forget”!

- **Game 1:** A sequence of numbers

  Q: What’s the median?

  A: 33

- **Game 1**: A sequence of numbers
  
  Q: What’s the median?
  
  A: 33

- **Game 2**: Relationships between Alice, Bob, Carol, Dave, Eva and Paul

- **Game 1**: A sequence of numbers

  Q: What’s the median?
  
  A: 33

- **Game 2**: Relationships between Alice, Bob, Carol, Dave, Eva and Paul

  Alice and Bob become friends

- **Game 1**: A sequence of numbers

  Q: What’s the median?

  A: 33

- **Game 2**: Relationships between Alice, Bob, Carol, Dave, Eva and Paul

  Carol and Eva become friends

- **Game 1**: A sequence of numbers

  Q: What’s the median?

  A: 33

- **Game 2**: Relationships between Alice, Bob, Carol, Dave, Eva and Paul

  Eva and Bob become friends

- **Game 1:** A sequence of numbers

  Q: What’s the median?

  A: 33

- **Game 2:** Relationships between Alice, Bob, Carol, Dave, Eva and Paul

  Dave and Paul become friends

- **Game 1**: A sequence of numbers
  
  Q: What’s the median?
  
  A: $33$

- **Game 2**: Relationships between Alice, Bob, Carol, Dave, Eva and Paul
  
  Alice and Paul become friends

- **Game 1**: A sequence of numbers
  
  Q: What’s the median?
  
  A: 33

- **Game 2**: Relationships between Alice, Bob, Carol, Dave, Eva and Paul
  
  Eva and Bob unfriends

- **Game 1**: A sequence of numbers

  Q: What’s the median?

  A: 33

- **Game 2**: Relationships between
  Alice, Bob, Carol, Dave, Eva and Paul

  Alice and Dave become friends

- **Game 1**: A sequence of numbers

  Q: What’s the median?
  
  A: 33

- **Game 2**: Relationships between Alice, Bob, Carol, Dave, Eva and Paul

  Bob and Paul become friends

- **Game 1**: A sequence of numbers

  Q: What’s the **median**?

  A: 33

- **Game 2**: Relationships between Alice, Bob, Carol, Dave, Eva and Paul

  Dave and Paul unfriends

- **Game 1**: A sequence of numbers

  Q: What’s the **median**?

  A: 33

- **Game 2**: Relationships between Alice, Bob, Carol, Dave, Eva and Paul

  Dave and Carol become friends

- **Game 1**: A sequence of numbers

  Q: What’s the median?

  A: 33

- **Game 2**: Relationships between Alice, Bob, Carol, Dave, Eva and Paul

  Q: Are Eva and Bob connected by friends?

- **Game 1:** A sequence of numbers
  
  Q: What’s the median?
  
  A: 33

- **Game 2:** Relationships between
  Alice, Bob, Carol, Dave, Eva and Paul

  Q: Are Eva and Bob connected by friends?
  
  A: YES. Eva ⇔ Carol ⇔ Dave ⇔ Alice ⇔ Bob

- **Game 1:** A sequence of numbers

  Q: What’s the median?

  A: 33

- **Game 2:** Relationships between
  Alice, Bob, Carol, Dave, Eva and Paul

  Q: Are Eva and Bob connected by friends?

  A: YES. Eva ↔ Carol ↔ Dave ↔ Alice ↔ Bob

- Have to allow approx/randomization given a small memory.
§1.1 Sampling

Uniformly sample an item
Tasks: Find a uniform sample from a stream of unknown length, can we do it in $O(1)$ space?
Tasks: Find a uniform sample from a stream of unknown length, can we do it in $O(1)$ space?

Algorithm: Store 1-st item. When the $i$-th ($i > 1$) item arrives

With probability $1/i$, replace the current sample;
With probability $1 - 1/i$, throw it away.
A toy example: Reservoir Sampling

Tasks: Find a uniform sample from a stream of unknown length, can we do it in $O(1)$ space?

Algorithm: Store 1-st item. When the $i$-th ($i > 1$) item arrives

  With probability $1/i$, replace the current sample;
  With probability $1 - 1/i$, throw it away.

Correctness: each item is included in the final sample w.p.

$$\frac{1}{i} \times \left(1 - \frac{1}{i+1}\right) \times \ldots \times \left(1 - \frac{1}{n}\right) = \frac{1}{n} \quad (n: \text{total } \# \text{ items})$$

Space: $O(1)$
§1.2 Distinct Elements

How many distinct elements?
Approximation needed.
The FM sketch

Denote the stream by $A = a_1, a_2, \ldots, a_m$, where $m$ is the length of the stream, which is unknown at the beginning. Let $n$ be the item universe.
The FM sketch

Denote the stream by $A = a_1, a_2, \ldots, a_m$, where $m$ is the length of the stream, which is unknown at the beginning. Let $n$ be the item universe.

**The algorithm** (Flajolet and Martin '83)

1. Choose a random hash function $h : [n] \rightarrow [n]$ from a 2-universal family. Set $z = 0$. Let $\text{zeros}(h(e))$ be the \# tailing zeros of the binary representation of $h(e)$.

2. For each new coming item $e$, if $\text{zeros}(h(e)) > z$, then set $z = \text{zeros}(h(e))$;

3. Output $2^z$. 
Denote the stream by \( A = a_1, a_2, \ldots, a_m \), where \( m \) is the length of the stream, which is unknown at the beginning. Let \( n \) be the item universe.

**The algorithm** (Flajoet and Martin ’83)

1. Choose a random hash function \( h : [n] \rightarrow [n] \) from a 2-universal family. Set \( z = 0 \). Let \( \text{zeros}(h(e)) \) be the number of trailing zeros of the binary representation of \( h(e) \).
2. For each new coming item \( e \), if \( \text{zeros}(h(e)) > z \), then set \( z = \text{zeros}(h(e)) \);
3. Output \( 2^z \).

**Example** (on board)
The FM sketch

Denote the stream by \( A = a_1, a_2, \ldots, a_m \), where \( m \) is the length of the stream, which is unknown at the beginning. Let \( n \) be the item universe.

**The algorithm** (Flajolet and Martin ’83)

1. Choose a random hash function \( h : [n] \rightarrow [n] \) from a 2-universal family. Set \( z = 0 \). Let \( \text{zeros}(h(e)) \) be the \# tailing zeros of the binary representation of \( h(e) \).
2. For each new coming item \( e \), if \( \text{zeros}(h(e)) > z \), then set \( z = \text{zeros}(h(e)) \);
3. Output \( 2^z \).

**Example** (on board)

**Theorem**

The number of distinct elements can be \( O(1) \)-approximated with probability \( 2/3 \) using \( O(\log n) \) bits (\( O(1) \) words).
Can we boost the success probability to $1 - \delta$?

The idea is to run $k = \Theta(\log(1/\delta))$ copies of this algorithm in parallel, using mutually independent random hash functions, and output the median of the $k$ answers.
§1.3 Heavy Hitters

What are most frequent items?
Heavy Hitters:

\[ HH_\phi = \{ i : f(i) \geq \phi m \} \]
Heavy hitters

- **Heavy Hitters:**

\[ HH_\phi = \{ i : f(i) \geq \phi m \} \]

- **(\(\epsilon, \phi\))-Heavy-Hitter Problem:**

Given \(\phi, \phi', (\text{often } \phi' = \phi - \epsilon)\), return a set \(S\) such that

\[ HH_\phi \subseteq S \subseteq HH_{\phi'} \]

In other words, we should
- return all \(i\) such that \(f(i) \geq \phi m\),
- and none \(i\) such that \(f(i) < (\phi - \epsilon)m\),
- and for those \(i\) whose \(f(i)\) is in between, may or may not return (arbitrary decision)
Algorithm Misra-Gries [Misra-Gries ’82]
The Misra-Gries Algorithm

Algorithm Misra-Gries [Misra-Gries ’82]

Maintain an array of size $1/\epsilon$, each entry is a tuple in the form of $(e, f(e))$. When a new item $e$ comes, we have two cases.

1. If $e$ is already in the array, increment $f(e)$ in $(e, f(e))$ by 1.

2. If $e$ is not in the array, create a new tuple $(e, 1)$ and try to insert it into the array.

In the case when there are already $1/\epsilon$ tuples in the array (i.e., the array is full), we decrement the $f(e)$ of all items $e$ in the array, and repeat doing it, until the $f(e)$ of some item $e$ becomes 0, and then we delete those tuples to find space for the new tuple.

Example (on board)
The Misra-Gries Algorithm (Analysis)

**Answering Query:** given \( e \in U \), return its frequency \( f(e) \)?

If \( e \) is already in the array, simply return its associated value, denoted by \( \tilde{f}(e) \).

Otherwise, return \( \tilde{f}(e) = 0 \)

**Analysis:** Can show that \( f(e) - \epsilon m \leq \tilde{f}(e) \leq f(e) \), where \( f(e) \) is the true frequency of \( e \) and \( \tilde{f}(e) \) is our estimation.

**Note:** this is enough to compute approximate heavy hitters.

The right inequality is straightforward from the algorithm since we never overestimate, but sometimes underestimate (when performing “decrement”)

For the left inequality, just notice that we can do at most \( \epsilon m \) decrements, since each decrement “consumes” \( 1/\epsilon \) items, and we have in total \( m \) items in the stream.
Space-saving: a variant of Misra-Gries

**Algorithm Space-saving** [Metwally et al. ’05]

When a new item $e$ comes, we have two cases.

1. If $e$ is already in the array. We just increment $f(e)$ by 1 and reinsert the $(e, f(e))$ into the array.

2. If $e$ is not in the array, we create a new tuple $(e, \text{MIN} + 1)$ where \( \text{MIN} = \min\{f(e_i) : e_i \text{ is in the array}\} \).

We always keep the array sorted according to $f(e_i)$, and then MIN is just the estimated frequency of the last item. If the length array is larger than $1/\varepsilon$, we delete the last tuple.

Cost: $O(1/\varepsilon)$ words.
Algorithm **Space-saving** [Metwally et al. ’05]

When a new item $e$ comes, we have two cases.

1. If $e$ is already in the array. We just increment $f(e)$ by 1 and reinsert the $(e, f(e))$ into the array.

2. If $e$ is not in the array, we create a new tuple $(e, \text{MIN} + 1)$ where $\text{MIN} = \min\{f(e_i) : e_i \text{ is in the array}\}$.

We always keep the array sorted according to $f(e_i)$, and then $\text{MIN}$ is just the estimated frequency of the last item. If the length of the array is larger than $1/\epsilon$, we delete the last tuple.

Cost: $O(1/\epsilon)$ words.

**Example** (on board)
**Answering Query**: given $e \in U$, return its frequency $f(e)$?

If $e$ is already in the array, simply return its associated value, denoted by $\tilde{f}(e)$

Otherwise, return $\tilde{f}(e) = \text{MIN}$

**Analysis**: Can show that $f(e) \leq \tilde{f}(e) \leq f(e) + \epsilon m$, where $f(e)$ is the true frequency of $e$ and $\tilde{f}(e)$ is our estimation.

**Note**: this is enough to compute approximate heavy hitters.
Which one is better?
• Data Streams: Algorithms and Applications
  by S. Muthukrishnan

  \(^a\)http://www.cs.rutgers.edu/~muthu/stream-1-1.ps
Maintain a sample for Sliding Windows

Tasks: Find a uniform sample from the last $w$ items.
Maintain a sample for Sliding Windows

**Tasks:** Find a uniform sample from the last $w$ items.

**Algorithm:**
- For each $x_i$, we pick a random value $v_i \in (0, 1)$.
- In a window $< x_{j-w+1}, \ldots, x_j >$, return value $x_i$ with smallest $v_i$.
- To do this, maintain the set of all $x_i$ in sliding window whose $v_i$ value is minimal among subsequent values.

Analogy: think about find the $\max$
Maintain a sample for Sliding Windows

**Tasks:** Find a uniform sample from the last \( w \) items.

**Algorithm:**

- For each \( x_i \), we pick a random value \( v_i \in (0, 1) \).
- In a window \( < x_{j-w+1}, \ldots, x_j > \), return value \( x_i \) with smallest \( v_i \).
- To do this, maintain the set of all \( x_i \) in sliding window whose \( v_i \) value is minimal among subsequent values.

Analogy: think about find the max

**Correctness:** Obvious.

**Space (expected):** \( 1/w + 1/(w - 1) + \ldots + 1/1 = \log w \).