CS485 Lecture 34
16 April 2007
Scribe: Alex Tsiatas

Data Streams

- **Four data stream problems:**
  - 1. Number of occurrences of a specific element in the stream
  - 2. Number of distinct elements in the stream
  - 3. Counts of frequently occurring elements
  - 4. Keep a sketch of the data that is sufficient to answer questions later, without knowing in advance what those questions will be.
  - The exact answer is not necessary – we are willing to get a good approximation
    - Computing the exact answer requires too much space
    - Warning: In the literature, there are many randomized approximation algorithms that appear to be very good: the expectation of the algorithm's output is the actual answer that is desired. However, in many cases, the variance is too high to make the use of such algorithms feasible.
- **Problem 1: Number of Occurrences**
  - Suppose the data stream is a string of 0's and 1's (0011010....) of length n.
  - The exact answer requires log(n) space: counting up to n 1's exactly requires storing the integer n, which uses log(n) bits.
  - We can approximate the answer in log(log(n)) space:
    - Let m be the number of ones.
    - We keep a value k such that 2^k is approximately m.
    - How do we do this?
      - In computing the exact answer, an algorithm would add one to the count whenever a 1 was encountered in the data stream.
      - For this approximation, for each 1, add one to k with probability $\frac{1}{2^k}$
        - Flip a coin with bias $\frac{1}{2^k}$ and add one to k if it comes out heads.
    - Claim: at the end of the data stream, m will be approximately $2^k - 1$.
    - Proof:
      - It is a fact that when flipping a coin with bias a, it will take $1/a$ flips, in expectation, before seeing heads.
      - This is because the expected number of times before seeing heads is:

$$\sum_{i=1}^{\infty} i \times Pr\left[\textit{Flip } i \textit{ is the first heads outcome}\right]$$

$$\sum_{i=1}^{\infty} \frac{a}{1-a} i (1-a)^i = \frac{1}{a}$$

      - Thus, it will take, in expectation, 2^k ones before k is incremented to k+1.
      - The total number of ones, then, to produce the ending value of k is
        $1 + 2 + 4 + 8 + 16 + ... = 2^k - 1$.

- **Problem 2: Number of Distinct Elements**
  - Suppose the data stream is a_1, a_2, a_3, ... with all $a_i \in \{A = 0, 1, 2, 3, \cdots, m\}$ (finite $|A|$)
  - This problem requires $O(m)$ space to compute exactly
    - With fewer than $O(m)$ space, there would be two different streams with the same memory configuration (by the pigeonhole pigeonhole), and this could lead to a wrong answer.
  - Consider the question: Are there greater than t distinct elements?
    - We will approximate this problem: if the number of distinct elements in between $0.5t$ and $2t$, then we won't care what answer an algorithm gives. If the number of distinct elements is less than $0.5t$, then with high probability, an algorithm should output NO, and if the number of distinct elements in greater than $2t$, then with high probability, an algorithm should output YES.
  - This algorithm needs a hash function h which maps the stream elements in A to integers $\{1, 2, 3, ..., t\}$.
    - More hash function details later in these notes
  - Approximation algorithm:
    - Calculate $h(a_i)$ for each element in the stream
    - Answer "YES" if for some i, $h(a_i) = 1$
  - This algorithm is very simple – almost *too* simple. Analysis:
    - If the $|A| > 2t$, we want the algorithm to answer YES.
    - Suppose we had exactly t elements.
      - Pr(Specific element mapped to 1) = $1/t$
      - Pr(Specific element not mapped to 1) = $1 - 1/t$
      - Pr(All elements not mapped to 1) = $(1 - 1/t)^{|A|}$
      - Pr(At least one element mapped to 1) = $1 - (1 - 1/t)^{|A|}$
    - Now, if $|A| > 2t$, the probability of all elements not being mapped to 1 is bounded from above by $(1- 1/t)^{2t}$ which is approximately $1/e^2$ (0.135) for large enough t.
    - Therefore, the probability of at least one element being mapped to 1 is bounded from below by 0.865.
    - Now, suppose we have $|A| < 0.5t$. Then, the probability that nothing is mapped to 1 is bounded from below by $(1 - 1/t)^{0.5t}$. For large enough t, this is approximately equal to $\sqrt{1/e} \neq 0.6$
    - These probabilities can be inflated higher with multiple iterations of the algorithm. The approximation bounds of $0.5t$ and $2t$ can also be adjusted.
- **Problem 3: Count the frequently occurring elements**
  - Idea: Every time the algorithm sees an element, start a count for that element with a small constant probability $1/c$. (for example, 1/1000).
    - This way, the less frequently elements get skipped, but we still count the most frequent elements.
  - Problem: it is hard to choose an appropriate c without knowing the length of the stream.
  - Fix this problem by changing the probabilities as the stream gets longer (for example, from 1/1000 to 1/1,000,000). Then adjust the old counts down to account for the new, smaller probabilities.
  - More details to come on Wed 4/18.

- **Hash Functions and their properties**
  - Problem 2 (number of distinct elements) required a hash function h.
    - It can't be just any old hash function – it has to have certain properties.
  - We will consider a family of hash functions $H = \{h_1, h_2, ..., h_m\}$ where each function h_i maps the set $\{0, 1, 2, ..., M\}$ to the set $\{1, 2, 3, ..., m\}$
  - Property 1: For fixed x, $h_i(x)$ should be equally likely to be any member of $\{1, 2, ..., m\}$ if h_i is randomly selected from H.
    - Simple hash function that satisfies Property 1: $h_i(x) = i$ for all i and x in their respective sets.
    - This is too simple – it turns out, it is not strong enough for Problem 2.
  - Property 2: the **2-universal hash function** property: For x and y distinct in members of $\{0, 1, 2, ..., M\}$, and u and v members of $\{1, 2, ..., m\}$, we would like $h(x) = u$ and $h(y) = v$ with probability $1/m^2$.
    - A hash function that satisfies this property: $h_{ab}(x) = ax + b \bmod m$. (This actually produces members of the set $\{0, 1, 2, ..., m\text{-}1\}$ but we can always add one without loss of generality, since the sets are the same size, and the representation as integers is arbitrary.) Then take $M=m^2$ to have a hash function for every pair (a, b).
    - This function satisfies the first property, since h_i is randomly selected.
    - This function also satisfies the second property, as the system of equations $ax + b = u \bmod m$, and $ay + b = v \bmod m$ has a unique solution (a, b), and choosing that pair randomly occurs with probability $1/m^2$, since there are m^2 pairs.