1 Introduction

Usually, a streaming algorithm is used in scenarios in which there are a lot of data (items) arriving and there is a space or time limitation for storage of data and processing later. More precisely, streaming algorithms are on-line algorithms which process the data streams. Each data stream is a long sequence of items arriving rapidly, denoted by $I_1, I_2, ..., I_t, ..., I_m$ where $I_t$ is the $t^{th}$ item and $m$ is the length of data stream. There are various data stream models depending on how to represent $I_t$:

- **Time Series Model.** In this model, $I_t$ is represented as $a_{i_t}$ where $a_{i_t} \in \{a_1, a_2, ..., a_n\}$. So the data stream is the sequence of items such that each item belongs to $\{a_1, a_2, ..., a_n\}$.

- **Cash Register Model.** In this model, $< a_1(t), a_2(t), ..., a_n(t) >$ is the state at time $t$. Upon arriving item $I_t$ which is modeled as pair of $(j, c_t)$, $a_i(t)$ will be calculated as follows:

  $$a_i(t) = \begin{cases} a_i(t-1) + c_t & \text{if } i = j \\ a_i(t-1) & \text{if } i \neq j \end{cases}$$

  Note that $c_t \geq 1$ and can not have a negative value.

- **Turnstile Model.** This model is similar to the Cash Register model with the difference that $|c_t| \geq 1$ which implies $c_t$ can have negative value.

2 Frequency Moments Algorithms

This section focuses on time series model. Let $m_i = |\{t | I_t = a_i\}|$ denote the the number of occurrences of $a_i$ in the sequence. For $k \geq 0$, the frequency moments $F_k$ is defined

$$F_k = \sum_{i=1}^{n} m_i^k \quad (1)$$

The numbers $F_k$ provide useful statistics on the sequence. For example, $F_0$ represents the number of distinct items appearing in the sequence, $F_1$ is the length of sequence, and $F_2$ is Gini’s index of homogeneity which can be used to show the diversity of items.

Flajolet and Martin [4] studied the algorithm for $F_0$. Later on, Alon et al. [2] showed that $F_2$ can be approximated randomly using only $\Theta(\log n + \log m)$ bits of memory. Moreover, they present a randomized approximation algorithm for $F_k$ with $\tilde{\Theta}(n^{1-\frac{k}{2}}) = \Theta(n^{1-\frac{k}{2}}(\log n + \log m))$. Following subsections will review these algorithms.

2.1 Estimating $F_k$

The basic idea behind this randomized approximation algorithm is to define a random variable whose expected value is close to $F_k$ and can be calculated under the space constraint. There are two parameters associated with the randomized approximation algorithm: (1) the error probability $\delta$ which demonstrates the probability that the algorithm fails, and (2) approximation ratio $\epsilon$. The output of algorithm, denoted by $Y$, should be calculated based on space constraints and satisfy the following inequality: 
\[ \mathbb{P}(|Y - F_k| > \epsilon F_k) \leq \delta \]  

Let constants \( s_1 \) and \( s_2 \) be defined as follows:

\[
s_1 = \frac{8k}{\epsilon^2 n^{1 - \frac{1}{k}}} \quad \quad s_2 = 2 \log \frac{1}{\delta} \quad \quad (3)
\]

The output of the algorithm \( Y \) is the median of \( s_2 \) random variables \( Y_1, Y_2, \ldots, Y_{s_2} \) where \( Y_i \) is the average of \( s_1 \) random variables \( X_{ij}, 1 \leq j \leq s_1 \). Note that all \( X_{ij} \) are independent identically distributed random variables. Each \( X = X_{ij} \) is calculated in the same way using only \( O(\log n + \log m) \) bits as follows: Choose randomly \( p \in [1, m] \), then see the value of \( a_p \). Maintain \( r = |\{q|q \geq p \text{ and } a_q = a_p\}|. \) Define \( X = m(r^k - (r - 1)^k) \). Note that in order to calculate \( X \), we only require to store \( a_p \) (log \( n \) bits) and \( r \) (at most log \( m \) bits). Now, we will show that \( E(X) = F_k \).

By definition of \( E(X) \), we have

\[
E(X) = \frac{m}{m} [((1^k + (2^k - 1)^k) + \ldots + (m_i^k - (m_1 - 1)^k))
\]
\[
\quad + (1^k + (2^k - 1)^k) + \ldots + (m_k^k - (m_2 - 1)^k)) + \ldots
\]
\[
\quad + (1^k + (2^k - 1)^k) + \ldots + (m_n^k - (m_n - 1)^k))] = \sum_{i=1}^{n} m_i^k = F_k
\]

Alon et al. [2] showed that

\[
E(X^2) \leq kF_1F_{2k-1} \leq kn^{1 - \frac{1}{k}} \left( \sum_{i=1}^{n} m_i^k \right)^2
\]

As \( \text{Var}(X) = E(X^2) - (E(X))^2 \), they can conclude that

\[
\text{Var}(X) \leq kn^{1 - \frac{1}{k}} F_k^2
\]

Thus, we have:

\[
\text{Var}(Y_i) = \frac{\text{Var}(X)}{s_1} \leq \frac{kn^{1 - \frac{1}{k}} F_k^2}{s_1}
\]

Note that \( E(Y_i) = E(X) = F_k \). Therefore, by Chebyshev’s inequality, we have:

\[
\mathbb{P}(|Y_i - F_k| > \epsilon F_k) \leq \frac{\text{Var}(Y_i)}{\epsilon^2 (F_k)^2} \leq \frac{kn^{1 - \frac{1}{k}} F_k^2}{s_1 \epsilon^2 (F_k)^2}
\]

### 2.2 Estimating \( F_2 \)

Using the algorithm presented in Section 2.1, \( F_2 \) can be computed employing \( O(\sqrt{n}(\log n + \log m)) \) memory bits. This section will present an improvement algorithm for \( F_2 \) which uses only \( O(\log n + \log m) \) bits of memory. Let constants \( s_1 \) and \( s_2 \) be defined as follows:

\[
s_1 = \frac{16}{\epsilon^2} \quad s_2 = 2 \log \frac{1}{\delta} \quad \quad (4)
\]
Fix a set $V = \{v_1, v_2, ..., v_h\}$ such that $|V| = h = O(n^2)$ and each $v_i \in \{1, -1\}^n$ is four-wise independent\(^1\). In other words, $V$ is the set of $O(n^2)$ vectors of length $n$ with 1 and $-1$ entities which are four-wise independent.

As with the previous algorithm, the output of the algorithm $Y$ is the median of $s_2$ random variables $Y_1, Y_2, ..., Y_{s_2}$ where $Y_i$ is the average of $s_1$ random variables $X_{ij}$, $1 \geq j \geq s_1$. Note that all $X_{ij}$ are independent identically distributed random variables. Each $X = X_{ij}$ is calculated in the same way using only $O(\log n + \log m)$ bits as follows: Choose uniformly random $p \in [1, h]$, and then look up $v_p = (b_1, b_2, ..., b_n)$. Then, define $Z = \sum_{i=1}^n b_i m_i$ (note that $Z$ can be computed by $O(\log n + \log m)$ memory bits). Afterward, define $X = Z^2$. We will show that $E(X) = F_2$ and $\text{Var}(X) \leq F_2$.

$$E(X) = E\left(\sum_{i=1}^n b_i m_i^2\right) = \sum_{i=1}^n m_i^2 E(b_i^2) + \sum_{i \neq j} m_i m_j E(b_i) E(b_j)$$

As random variables $b_i$ are pair-wise independent, $E[b_i] = 0$ for all $i$ (this is because $\text{Prob}(b_i = 1) = \text{Prob}(b_i = -1) = \frac{1}{2}$). Moreover, $E[b_i^2] = 1$ for all $i$. So we have:

$$E(X) = \sum_{i=1}^n m_i^2 = F_2$$

Similarly, we can conclude that

$$E(X^2) = \sum_{i=1}^n m_i^4 + 6 \sum_{1 \leq i < j \leq n} m_i^2 m_j^2$$

So we have:

$$\text{Var}(X) = E(X^2) - E(X)^2$$

$$= \sum_{i=1}^n m_i^4 + 6 \sum_{1 \leq i < j \leq n} m_i^2 m_j^2 - \left(\sum_{i=1}^n m_i^2\right)^2$$

$$= 4 \sum_{1 \leq i < j \leq n} m_i^2 m_j^2$$

$$\leq 2F_2^2$$

Thus, we have:

$$\text{Var}(Y_i) = \frac{\text{Var}(X)}{s_1} \leq \frac{2F_2^2}{s_1}$$

Note that $E(Y_i) = E(X) = F_2$. Therefore, by Chebyshev’s inequality, we have:

$$\text{Prob}[|Y_i - F_2| > \epsilon F_2] \leq \frac{\text{Var}(Y_i)}{\epsilon^2 F_2^2} \leq \frac{2F_2^2}{s_1 \epsilon^2 F_2^2}$$

\(^1\)A probability distribution over $\{-1, 1\}^n$ is 4-wise independent if for every four distinct coordinates $i_1 < i_2 < i_3 < i_4$ and every choice of $b_1, b_2, b_3, b_4 \in \{-1, 1\}$ exactly a $\frac{1}{16}$ fraction of vectors have $b_j$ in their coordinate number $i_j$ for $j = 1, ..., 4$ [2].
3 Count-Min Sketch

The turnstile model introduced in Section 1 uses the vector \( \vec{a}(t) = < a_1(t), ..., a_i(t), ..., a_n(t) > \).

Note that \( a_i(t) \) represents the value of variable \( a_i \) at time \( t \) and \( a_i(0) = 0 \) for all \( i \). If we have limitation in storing \( a_i(t) \) then we need to approximate the value of \( a_i(t) \). Suppose \( Q(i) \) is a function which return \( Z_i \), an estimate of \( a_i \). The goal is to produce \( Z_i \geq a_i(t) \) while satisfying the following property with the probability of \( 1 - \delta \):

\[
Z_i \leq a_i(t) + \epsilon ||a||_1
\]

where \( ||a||_1 = \sum_{i=1}^{n} a_i(t) \).

Data Structure. A Count-Min(CM) sketch [3] with the parameters \( (\epsilon, \delta) \) is presented by a two-dimensional matrix \( Count_{d \times w} \) with \( d \) rows and \( w \) columns where \( w = \left\lceil \frac{e}{\epsilon} \right\rceil \) and \( d = \left\lceil \ln \frac{1}{\delta} \right\rceil \).

Moreover, we need \( d \) hash functions such that

\[
h_1...h_d : \{1, 2, ..., n\} \rightarrow \{1...w\}
\]

and they are a family of pair-wise independent hash functions.

Update Procedure. Upon receiving an item \( I_t = (i_t, c_t) \), we update the matrix \( Count \) as follows:

\[
Count[j, h_j(i_t)] = Count[j, h_j(i_t)] + c_t \quad \forall 1 \leq j \leq d
\]

Approximation. \( Q(i) \) is calculated as follows:

\[
Q(i) = \min_j Count[j, h_j(i_t)]
\]

4 Next Lecture Preview

Markov Chain. Given a set of states denoted by \( S = \{s_1, s_2, ..., s_n\} \), the process starts from one state and moves to other states consequently. Each move is called a step. If the chain is in state \( s_i \) at time \( t \), then it moves to state \( s_j \) at time \( t + 1 \) with \( p_{ij} \) probability. This probability is independent from which states the chain was before reaching to current state (i.e., memoryless property of Markov chain).

Random Walk on a Graph. Given a graph and a starting node, we select randomly a neighbor and move to it. Then, we select randomly a neighbor of current node and move to it and so on. The random sequence of nodes selected this way is a random walk on the graph. Every Markov chain can be viewed as random walk on a directed graph.

Suppose a uniform random walk on a directed graph. Let \( h_{ij} \) be the expected time to go from \( v_i \) to \( v_j \). Let define \( c_{ij} \) the commute time from \( v_i \) to \( v_j \) and vice versa. We have \( c_{ij} = h_{ij} + h_{ji} \) where \( h_{ij} \neq h_{ji} \). Let \( c_u(G) \) be the required time to visit every nodes in \( G \) starting at \( u \). We define \( c(G) \) as follows:

\[
c(G) = \max_u c_u(G)
\]

It has been shown that \( c(G) \leq 2m(n - 1) \) where \( m = |E| \) and \( n = |V| \). This result is used to solve USTCON problem (undirected s-t connectivity problem) which is the problem of deciding if there is a path between two nodes in an undirected graph [1].
References


