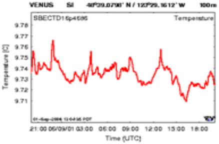
Data Stream Model

Data stream: In data stream model, input data is presented to the algorithm as a stream of items in no particularly order. The data stream is read only once and cannot be stored (entirely) due to the large volume.

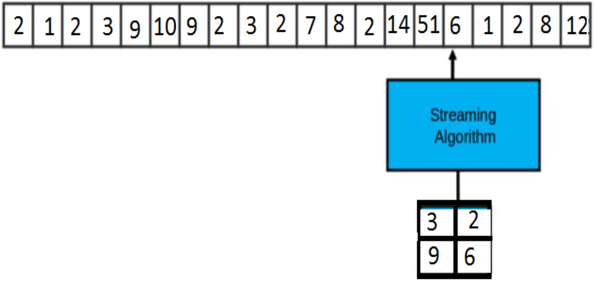
Examples of data streams:

- sensor data : temperature, pressure, ...
- website visits, click streams
- user queries (search)
- social network activities
- business transactions
- call center records



Data Stream Model

Streaming Algorithm is an algorithm that processes a data stream and has small memory compared with the amount of data it processes.



Sublinear space usage: Assuming n is the size of input typically a streaming algorithm has $o(n)$ (for example $\log^2(n)$, $\sqrt{n}$, etc) space usage.

Data Stream Model: two motivating puzzles

Missing elements in a permutation: Suppose the stream is a permutation of $\{1, \ldots, n\}$ with one element missing. How much space is needed to find the missing element?

$$7, 8, 3, 9, 1, 5, 2, 6, 10, 12, 11$$

What if 2 elements are missing?

Can we generalize to k missing elements?

Majority element: Suppose the stream is a sequence of numbers $a_1, \ldots, a_m$. Suppose one element is repeated at least $\frac{m}{2}$ times. How can we find the majority element?

$$2, 3, 2, 1, 2, 9, 8, 2, 5, 2, 2, 4, 6, 2, 2, 5, 2$$

Frequency Moments

Let $A = a_1, a_2, \ldots, a_m$ be the input stream where each $a_i \in \{1, \ldots, n\}$. Let f_i denote the number of repetitions of i in A. We define the k-th frequency moment of A

$$F_k = \sum_{i=1}^{n} f_i^k$$

Example: $A = 2, 3, 2, 1, 2, 9, 8, 2, 5, 2, 2, 4, 6, 2, 2, 5, 2$
$f_1 = 1, f_2 = 9, f_3 = 1, f_4 = 1, f_5 = 2, f_6 = 1, f_7 = 0, f_8 = 1, f_9 = 1$

$F_0 = \text{number of distinct elements } = 8$

$F_1 = \sum_{i=1}^{n} f_i = 17 = m$

$F_2 = \sum_{i=1}^{n} f_i^2 = 1^2 + 9^2 + 1^2 + 1^2 + 2^2 + 1^2 + 0^2 + 1^2 + 1^2 = 91$

$F_\infty = \max_{i=1}^{n} f_i$

Computing F_k in small space

Trivial facts:

- We can compute F_1 exactly in $O(1)$ words ($O(\log m)$ bits) of space.
- When k is a constant, we can compute F_k exactly in $O(n)$ words of space.

Nontrivial facts:

- Assuming $k \neq 1$, any randomized streaming algorithm that computes F_k exactly requires $\Omega(n)$ space.
- Assuming $k \neq 1$, any deterministic streaming algorithm that computes a constant factor approximation of F_k requires $\Omega(n)$ space.
- Both randomization and approximation is needed to compute F_k in sublinear space.

Approximating F_2

The space complexity of approximating the frequency moments *

Noga Alon † Yossi Matias ‡ Mario Szegedy §

February 22, 2002

Abstract

The frequency moments of a sequence containing m_i elements of type i, for $1 \leq i \leq n$, are the numbers $F_k = \sum_{i=1}^{n} m_i^k$. We consider the space complexity of randomized algorithms that approximate the numbers F_k, when the elements of the sequence are given one by one and cannot be stored. Surprisingly, it turns out that the numbers F_0, F_1 and F_2 can be approximated in logarithmic space, whereas the approximation of F_k for $k \geq 6$ requires $n^{\Omega(1)}$ space. Applications to data bases are mentioned as well.

Theorem [AMS99] There is a randomized streaming algorithm that approximates F_2 within $1+\epsilon$ factor using $O(\frac{1}{\epsilon^2}(\log n + \log m))$ bits of space. The algorithm succeeds with probability $3/4$.

AMS idea:

For each coordinate of $\boldsymbol{f}$, we pick a random number r_i independently from $\{-1, +1\}$. For now, suppose we have stored the vector $\boldsymbol{r} = (r_1, \ldots, r_n)$.

Let $Z = \sum_{i=1}^{n} r_i f_i$. Note that Z is computed as the input stream arrives. We analyze $X = Z^2$.

$E[X] = E[(\sum_{i=1}^{n} r_i f_i)^2] = \sum_{i=1}^{n} E[r_i^2] f_i^2 + \sum_{i \neq j} E[r_i r_j] f_i f_j$

Because r_i and r_j are independent, we get

$E[X] = \sum_{i=1}^{n} E[r_i^2] f_i^2 + \sum_{i \neq j} E[r_i] E[r_j] f_i f_j$

Because $E[r_i] = 0$ and $E[r_i^2] = 1$, we get

$E[X] = \sum_{i=1}^{n} f_i^2 = F_2$

Using Chebyshev Inequality, we can say

$$Pr(|X - E[X]| > \epsilon E[X]) = \frac{Var[X]}{\epsilon^2 E^2[X]}$$

Because r_i's are independent,

$$E[X^2] = E[Z^4] = E[(\sum_{i=1}^{n} r_i f_i)^4] =$$

$$\sum_{i=1}^{n} E[r_i^4] f_i^4 + 6 \sum_{i,j} E[r_i^2 r_j^2] f_i^2 f_j^2 + 4 \sum_{i,j} E[r_i r_j^3] f_i f_j^3$$

$$= \sum_{i=1}^{n} f_i^4 + 6 \sum_{i,j} f_i^2 f_j^2$$

$$Var[X] = E[X^2] - E^2[X] \leq 4 \sum_{i,j} f_i^2 f_j^2 \leq 2F_2^2$$

Consequently,

$$Pr(|X - E[X]| > \epsilon E[X]) = \frac{Var[X]}{\epsilon^2 E^2[X]} \leq \frac{2F_2^2}{\epsilon^2 F_2^2} = \frac{2}{\epsilon^2}$$

Repeat to Decrease the Variance

To decrease the variance of X, we compute $s = \frac{8}{\epsilon^2}$ independent copies $Y_1, \ldots, Y_s$ of X and output their average $Y = \frac{1}{s}(Y_1 + \ldots + Y_s)$

Note that $E[Y] = E[X] = F_2$ and $Var[Y] = \frac{1}{s}Var[X]$

$$Pr(|Y - E[Y]| \geq \epsilon E[Y]) \leq \frac{Var[Y]}{\epsilon^2 E^2[Y]} \leq \frac{1}{4}$$

$$Pr(|Y - F_2| \geq \epsilon F_2) \leq \frac{Var[Y]}{\epsilon^2 E^2[Y]} \leq \frac{1}{4}$$

Do we need to store the vector $\boldsymbol{r}$?

We do not need to store the random vector $\boldsymbol{r} = (r_1, \ldots, r_n)$.

It is enough the random coefficients r_i's to be 4-wise independent. We do not need them to be mutually independent! Check the analysis of $E[X]$ and $E[X^2]$.

Fact: We can generate a set of n k-wise independent random numbers using $O(k \log n)$ random bits.

How? See next slides.

Independence

Mutually independence: Let $X_1, \ldots, X_n$ be discrete random variables. We say $X_1, \ldots, X_n$ are (mutually) independent if for all values $\alpha_1, \ldots, \alpha_n$ we have

$$Pr(X_1 = \alpha_1, \ldots, X_n = \alpha_n) = \prod_{i=1}^{n} Pr(X_i = \alpha_i)$$

k-wise independence: Let $X_1, \ldots, X_n$ be discrete random variables. We say $X_1, \ldots, X_n$ are k-wise independent if for every subset $S = \{s_1, \ldots, s_\ell\} \subseteq \{1, \ldots, n\}$ of cardinality at most k and all values $\alpha_1, \ldots, \alpha_\ell$, we have

$$Pr(X_{s_1} = \alpha_1, \ldots, X_{s_\ell} = \alpha_\ell) = \prod_{i=1}^{\ell} Pr(X_{s_i} = \alpha_i)$$

k-wise Independence: Special case

Let $X_1, \ldots, X_n$ be $\{0,1\}$-valued random variables where for each i we have $Pr(X_i = 0) = Pr(X_i = 1)$. We say $X_1, \ldots, X_n$ are k-wise independent if for every subset $S = \{s_1, \ldots, s_\ell\} \subseteq \{1, \ldots, n\}$ of cardinality at most k and all values $\alpha_1, \ldots, \alpha_\ell$ we have

$$Pr(X_{s_1} = \alpha_1, \ldots, X_{s_\ell} = \alpha_\ell) = (\frac{1}{2})^\ell$$

Constructing k-wise independent random bits

First Idea: Assume n is even. We let $X_1, \ldots, X_n$ be random (cyclic) shift of $\underbrace{1, 0, 1, 0, 1, 0, \ldots, 1, 0}_{n}$. We have

$Pr(X_i = 0) = Pr(X_i = 1) = \frac{1}{2}$

Question: How much randomness is used in this construction?

Answer: $\log n$ bits

But X_i and X_j are not independent. ☹

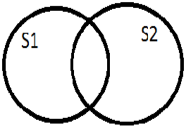
$$Pr(X_1 = 0, X_2 = 0) = 0 \neq \frac{1}{4}$$

Second Idea: [Pair-wise Independent Random Bits] Let $Y_1, \ldots, Y_m$ be mutually independent random bits. We construct $n = 2^m - 1$ random bits from $Y_1, \ldots, Y_m$. For each non-empty subset $S \subseteq [m] = \{1, \ldots, m\}$ we let

$$X_S = \sum_{r \in S} Y_r \mod 2$$

Claim: The random bits $\{X_S\}_{S \subseteq [m]}$ are pair-wise independent.

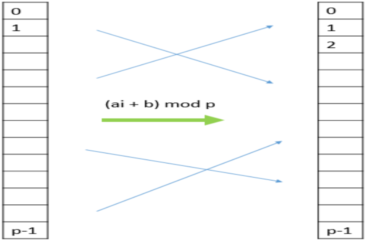
Proof: Exercise.



Conclusion: We can generate n pairwise random bits from $\log n + 1$ mutually independent random bits.

Third Idea: [Pair-wise Independent Random Numbers] Let p be a prime where $p \geq n$. We choose the random numbers a and b from $\mathbb{Z}_p$ independently. We let

$$X_i = (ai + b) \mod p$$



- $\forall i \in [n], \alpha \in \mathbb{Z}_p, \;\; Pr(X_i = \alpha) = \frac{1}{p}$
- $\forall i, j \in [n], \alpha_1, \alpha_2 \in \mathbb{Z}_p, \;\; Pr(X_i = \alpha_1, X_j = \alpha_2) = (\frac{1}{p})^2$

Fourth Idea: [Third Idea Generalized] Consider the finite field $\mathbb{F}_p$ where p is large enough. Let $Y_0, Y_1, \ldots, Y_{\ell-1}$ be independent samples from $\mathbb{F}_p$. For $a \in \mathbb{F}_p$, we define

$$X_a = \sum_{i=0}^{\ell-1} Y_i a^i = Y_0 + Y_1 a + Y_2 a^2 + \ldots + Y_\ell a^\ell$$

Note that all computations are done in the field $\mathbb{F}_p$

Lemma: The random variables $\{X_a\}_{a \in \mathbb{F}_p}$ are (ℓ)-wise independent.

Therefore we can generate k-wise independent $(0, 1)$-valued random variables $X_1, \ldots, X_n$ from $O(k)$ random numbers from $\mathbb{F}_p$ where $p \geq n$. To generate any X_i, we keep at most $O(k \log n)$ bits.

In particular, we can generate 4-wise independent $(-1, +1)$-valued random variables $r_1, \ldots, r_n$ by keeping $O(\log n)$ random bits.