Chapter 14

Streaming Data Model

14.1 Finding frequent elements in stream

A very useful statistics for many applications is to keep track of elements that occur *more frequently*. It can come in many flavours

- **Mode** : The element (or elements) with the highest frequency.
- **Majority**: An element with more than 50% occurrence - note that there may not be any.
- **Threshold**: Find out all elements that occur more than f fraction of the stream for any $0 < f \leq 1$. Finding majority is a special case with $f > 1/2$.

The above problems are hardly interesting from an algorithmic design perspective and can be reduced to sorting. Designing more efficient algorithms requires more thought (for example, finding mode). Accomplishing the same task in a streaming environment with limited memory presents interesting design challenges. Let us first review a well known algorithm for *Majority* finding among n elements known as *Boyer-Moore Voting algorithm*. Note that the above procedure scans the array sequentially and uses one counter variable. It is not obvious why it returns the majority element if it exists. If there is no majority element, this can be verified by making one additional pass to if the count of the element $a[maj - ind]$ exceeds $n/2$.

The algorithm tries to prune elements without affecting the majority, if one exists. At any point in the algorithm, the elements under consideration is the set of elements yet to be scanned plus the multiset of the elements accounted under the variable *count*. A key observation is, if there is a majority, it will remains a majority if some other element is deleted along with an instance of the majority element. Indeed if $n_1 > n/2$ then $n_1 - 1 > (n-2)/2$. So, among a pair of elements, we can delete both

Procedure Finding Majority of n elements in Array a

```
1 maj_ind ← 0 ;
  count ← 0 ;
2 for i = 1 to n do
3 |   if count = 0 then
  |   |   maj_ind ← i  (* initalize maj_ind *)
4 |   if a[maj_ind] = a[i] then
5 |   |   count ← count + 1
  |   else
6 |   |   count ← count − 1 ;
7 Return a[maj_ind] ;
```

Figure 14.1: Boyer-Moore Majority Voting Algorithm

as long as both are not majority. If both values are equal (and could be a candidate for majority), we simply increment the count so that we can pair them with distinct elements in future.

Alternately, we can argue that the count of any element can decrease by at most $n/2$ if there is a majority element. In the end, the element returned is only a candidate for majority and this needs to be verified using a second pass.

This idea can be generalized to finding out elements whose frequency is at least $\frac{n}{k}$ for any integer k. Instead of one counter, we shall use $k-1$ counters. When we scan the next element, we can either increment the count, if there exists a counter for the element or start a new counter if number of counters used is less than $k-1$. Otherwise, we decrease the counts of all the existing counters. If any counter becomes zero, we discard that element and instead assign a counter for the new element. In the end the counters return the elements that have non-zero counts. These are potentially the elements that have frequencies at least $\frac{n}{k}$ and need a second pass to verify them.

The proof of correctness is along the same lines as the majority. Note that there can be at most $k-1$ elements that have frequencies exceeding $\frac{n}{k}$, i.e., a fraction $\frac{1}{k}$. So, if we remove such an element along with $k-1$ distinct elements, it still continues to be at least $\frac{1}{k}$ fraction of the remaining elements - $n_1 > \frac{n}{k} \Rightarrow n_1 - 1 > \frac{n-k}{k}$. We can actually perform approximate counting using the Mishra-Gries algorithm (Figure Algorithm for threshold).

Exercise 14.1 *Let f_i be the frequency of element i in the stream. Then show that for a stram of length n, one can compute $\hat{f}_i$ such that*

$$f_i - \frac{n}{k} \leq \hat{f}_i \leq f_i$$

Procedure Algorithm for threshold

```
1 cur : current element of stream ;
  S: current set of elements with non-zero counts, |S| ≤ k ;
2 if cur ∈ S then
3 |   increment counter for cur
  else
  |   if |S| < k then
4 |   |   Start a new counter for cur, update S
  |   else
5 |   |   decrement all counters ;
6 |   |   If a counter becomes 0 delete it from S
  |
7 Return a[maj_ind] ;
```

Figure 14.2: Mishra-Gries streaming algorithm for frequent elements

The previous algorithms have the property that the data is scanned in the order it is presented and the amount of space is proportional to the number of counters where each counter has $\log n$ bits.

14.2 Distinct elements in a stream

The challenging aspect of this problem is to count the number of distinct elements d with limited memory m where $m \ll d$. Otherwise, we could simply hash the elements and the count the number of non-zero buckets. Uniformly sampling a subset could be misleading as multiple occurrence of any element would be picked up by the the uniform sample and it doesn't provide any significant information about the number of distinct elements.

Instead we will hash the incoming elements uniformly over a range, $[1, n]$ such that if there are k distinct elements then they will be roughly n/k apart. If g is the gap between two consecutive hashed elements, then we can estimate $k = n/g$. Alternately, we can use the position of the first hashed position as as estimate of g. This is the underlying idea behind the algorithm given in Figure 14.3.

This procedure will be analyzed rigorously using the property of universal hash family family discussed earlier. The parameter of interest will be the *expected* gap between consecutive hashed elements.

If k elements are uniformly mapped into the range $[1, n]$. we can compute the expected position of the first hashed location as a function of k and then estimate d. The algorithm keeps track of the smallest hashed location. At the end of the

Procedure Finding the number of distinct elements in a stream $S(m, n)$

```
1 Input A stream S = {x_1, x_2 ... x_m} where x_i ∈ [1, n] ;
2 Suppose p is a prime in the range [n, 2n]. Choose 1 ≤ a ≤ p − 1 and
    0 ≤ b ≤ p − 1 uniformly at random ;
3 Z ← ∞ ;
4 for i = 1 to m do
5 |   Y = (a · x_i + b) mod p ;
6 |   if Y < Z then
  |     Z ← Y
7 Return ⌈p/Z⌉ ;
```

Figure 14.3: Counting number of distinct elements

stream this is the value g. Suppose we can show that $g \in [g_1(n,k), g_2(n,k)]$ with high probability. Then we can invert this to claim that $k \in [k_1, k_2]$.

Let $Z_i = (a \cdot x_i + b) \mod p$ be the sequence of hashed values from the stream. Then we can claim the following.

Claim 14.1 *The numbers* Z_i *,*$0 \leq Z_i \leq p-1$ *are uniformly random in the range* $[0, p-1]$ *and are also pairwaise independent , viz., for* $r \neq s$

$$\Pr[Z_i = r, Z_k = s] = \Pr[Z_i = r] \cdot \Pr[Z_k = s] = \frac{1}{p(p-1)}$$

Proof: For some fixed $i_0 \in [0, p-1]$ and $x \in [1, n]$, we want to find the probability that x is mapped to i_0. So

$$\begin{aligned} i_0 &\equiv (ax + b) \mod p \\ i_0 - b &\equiv ax \mod p \\ x^{-1}(i_0 - b) &\equiv a \mod p \end{aligned}$$

where x^{-1} is the multiplicative inverse of x in the multiplicative prime field modulo p and it is unique since p is prime[1]. For any fixed b, there is a unique solution for a. As a is chosen uniformly at random, the probability of this happening is $\frac{1}{p}$ for any *fixed* choice of b. Therefore this is also the unconditional probability that x is mapped to i_0.

For the second part consider $i_0 \neq i_1$. We can consider $x \neq y$ such that x, y are mapped respectively to i_0 and i_1. We can write the simultaneous equations similar

[1]By our choice of p, $x \not\equiv 0 \mod p$

to the previous one.

$$\begin{bmatrix} x & 1 \\ y & 1 \end{bmatrix} \cdot \begin{bmatrix} a \\ b \end{bmatrix} \equiv_p \begin{bmatrix} i_0 \\ i_1 \end{bmatrix}$$

The 2×2 matrix is invertible for $x \neq y$ and therefore there is a unique solution corresponding to a fixed choice of (i_0, i_1). The probability that a, b matches the solution is $\frac{1}{p(p-1)}$ as they are chosen uniformly at random over all the $\binom{p}{2}$ pairs. $\square$

Let N be the number of distinct elements in the stream. Then we can claim the following

Claim 14.2 *For any constant $c \geq 2$,*

$$Z \in \left[\frac{p}{cN}, \frac{cp}{N}\right] \text{ with probability } \geq 1 - \frac{2}{c}$$

Proof: Note that if $Z = p/N$, then the algorithm returns N which is the number of distinct elements in the stream. Since Z is a random variable, we will only be able to bound the probability that it is within the interval $\left[\frac{p}{cN}, \frac{cp}{N}\right]$ with significant probability implying that the algorithm with return an answer in the range $[p/c, pc]$ with significant probability. Of course, there is a risk that it falls outside this window and that is the inherent nature of a *Monte Carlo* randomized algorithm.

First we will find the probability that $Z \leq s - 1$ for some arbitrary s. Let us define a family of indicator random variables in the following manner

$$X_i = \begin{cases} 1 & \text{if } (ax_i + b) \mod p \leq s - 1 \\ 0 & \text{otherwise} \end{cases}$$

So the total number of x_i that map to numbers in the range $[1, s-1]$ equals $\sum_{i=1}^{N} X_i$ since there are only N distinct x_i's and wlog, we are assuming that the first N are distinct (alternately we can choose N indices corresponding to N distinct x_i's). Let $X = \sum_i X_i$ and we therefore have

$$\mathbb{E}[X] = \mathbb{E}[\sum_i X_i] = \sum_i \mathbb{E}[X_i] = \sum_i \Pr[X_i = 1] = N \cdot \Pr[X_i = 1] = \frac{sN}{p}$$

The last equality follows from the previous result as there are $s = \{0, 1 \ldots s-1\}$ possibilities for x_i to be mapped and each has probability $\frac{1}{p}$.

If we choose $s = \frac{p}{cN}$ for some constant c, then $\mathbb{E}[X] = 1/c$. From Markov's inequality, $\Pr[X \geq 1] \leq \frac{1}{c}$, implying that with probability greater than $1 - 1/c$ no x_i will be mapped to numbers in the range $[0, \lceil \frac{p}{cN} \rceil]$.

For the other direction, we will will Chebychev inequality, which requires computing the variance of X, which we shall denote by $\sigma^2(X)$. We know that

$$\sigma^2[X] = \mathbb{E}[(X - \mathbb{E}[X])^2] = \mathbb{E}[X^2] - \mathbb{E}^2[X]$$

Since $X = \sum_i X_i$, we can calculate

$$\begin{aligned}
\mathbb{E}[X^2] &= \mathbb{E}[(\sum_i X_i)^2] \\
&= \mathbb{E}[\sum X_i^2 + \sum_{i \neq j} X_i \cdot X_j] \\
&= \mathbb{E}[\sum X_i^2] + \mathbb{E}[\sum_{i \neq j} X_i \cdot X_j] \\
&= \sum \mathbb{E}[X_i^2] + \sum_{i \neq j} \mathbb{E}[X_i] \cdot \mathbb{E}[X_j]
\end{aligned}$$

which follows from linearity of expectation and pairwise independence of X_i and X_j[2]. So the expression simplifies to $N \cdot \frac{s}{p} + N(N-1) \cdot \frac{s^2}{p^2}$. This yields the expression for

$$\sigma^2(X) = \frac{sN}{p} + \frac{N(N-1)s^2}{p^2} - \frac{s^2N^2}{p^2} = \frac{sN}{p} \cdot (1 - \frac{s}{p} \leq \frac{sN}{p}$$

For $s = \frac{cp}{N}$, the variance is bounded by c. From Chebychev's inequality, we know that for any random variable X,

$$\Pr[|X - \mathbb{E}[X]| \geq t] \leq \frac{\sigma^2(X)}{t^2}$$

Using $t = \mathbb{E}[X] = \frac{sN}{p} = c$, we obtain $\Pr[|X - \mathbb{E}[X]| \geq \mathbb{E}[X]] \leq \frac{c}{c^2} = \frac{1}{c}$. The event $|X - \mathbb{E}[X]| \geq \mathbb{E}[X]$ is the union of two disjoint events, namely

(i) $X \geq 2\mathbb{E}[X]$ and
(ii) $\mathbb{E}[X] - X \geq \mathbb{E}[X]$, or $X \leq 0$

Clearly, both events must have probability bounded by $\frac{1}{c}$ and specifically, the second event implies that the probability that none of the N elements is mapped to the interval $[0, \frac{cp}{N}]$ is less than $\frac{1}{c}$. Using the union bound yields the required result. □

So the algorithm outputs a number that is within the range $[\frac{N}{c}, cN]$ with probability $\geq 1 - \frac{2}{c}$.

[2]This needs to be rigorously proved from the previous result on pairwise independence of (x_i, x_j) being mapped to (i_0, i_1). We have to technically consider all pairs in the range $(1, s-1)$.

14.3 Frequency moment problem and applications

Suppose the set of elements in a stream $S = \{x_1, \ldots, x_m\}$ belong to a universe $U = \{e_1, \ldots, e_n\}$. Define the frequency f_i of element e_i as the number of occurrences of e_i in the stream S. The k^{th} frequency moment of the stream is defined as

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

Note that F_0 is exactly the number of distinct elements in the stream. F_1 counts the number of elements in the stream, and can be easily estimated by keeping a counter of size $O(\log m)$. The second frequency moment F_2 captures the non-uniformity in the data – if all n elements occur with equal frequency, i.e., m/n (assume that m is a multiple of n for the sake of this example), then F_2 is equal to m^2/n; whereas if the stream contains just one element (with frequency m), then F_2 is m^2. Thus, larger values of F_2 indicate non-uniformity in the stream. Higher frequency moments give similar statistics about the stream – as we increase k, we are putting more emphasis on higher frequency elements.

The idea behind estimating F_k is quite simple : suppose we sample an element uniformly at random from the stream, call it X. Suppose X happens to be the element e_i. Conditioned on this fact, X is equally likely to be any of the f_i occurrences of e_i. Now, we observe how many times e_i occurs in the stream for now onwards. Say it occurs r times. What can we say about the expected value of r^k ? Since e_i occurs f_i times in the stream, the random variable r is equally likely to be one of $\{1, \ldots, f_i\}$. Therefore,

$$\mathbb{E}[r^k] = \frac{1}{f_i} \sum_{j=1}^{i} j^k.$$

Looking at the above expression, we see that $\mathbb{E}[r^k - (r-1)^k] = \frac{1}{f_i} \cdot f_i^k$. Now, we remove the conditioning on X, we see that

$$\mathbb{E}[r^k - (r-1)^k] = \mathbb{E}[r^k - (r-1)^k | X = e_i] \Pr[X = e_i] = \frac{1}{f_i} \cdot f_i^k \cdot \frac{f_i}{m} = \frac{1}{m} \cdot F_k.$$

Therefore, the random variable $m(r^k - (r-1)^k)$ has expected value as F_k.

The only catch is that we do not know how to sample a uniformly random element of the stream. Since X is a random element of the stream, we want

$$\Pr[X = x_j] = \frac{1}{m},$$

for all values of $j = 1, \ldots, m$. However, we do not know m in advance, and so cannot use this expression directly. Fortunately, there is a more clever sampling procedure,

called *reservoir sampling*, described in Figure Combining reservoir sampling with the estimator for F_k. Note that at iteration i, the algorithm just tosses a coin with probability of Heads equal to $1/i$.

Exercise 14.2 *Prove by induction on i that after i steps, the random variable X is a uniformly chosen element from the stream* $\{x_1, \ldots, x_i\}$.

We now need to show that this algorithm gives a good approximation to F_k with high probability. So far, we have only shown that there is a random variable, namely $Y := m(r^k - (r-1)^k)$, which is equal to F_k in expectation. But now, we want to compute the probability that Y lies within $(1 \pm \varepsilon)F_k$. In order to do this, we need to estimate the variance of Y. If the variance is not too high, we can hope to use Chebychev's bound. We know that the variance of Y is at most $\mathbb{E}[Y^2]$. Therefore, it is enough to estimate the latter quantity. Since we are going to use Chebychev's inequality, we would like to bound $\mathbb{E}[Y^2]$ in terms of $E[Y^2]$, which is same as F_k^2. The first few steps for estimating $\mathbb{E}[Y]2$ are identical to those for estimating $\mathbb{E}[Y]$:

$$\begin{aligned}\mathbb{E}[Y^2] &= \sum_{i=1}^{n} \mathbb{E}[Y^2|X = e_i] \cdot \Pr[X = e_i] = \sum_{i=1}^{n} m^2 \cdot \mathbb{E}[(r^k - (r-1)^k)^2|X = e_i] \cdot \frac{f_i}{m} \\ &= \sum_{i=1}^{n} m f_i \cdot \frac{1}{f_i} \sum_{j=1}^{f_i} (j^k - (j-1)^k)^2 = m \cdot \sum_{i=1}^{n} \sum_{j=1}^{f_i} (j^k - (j-1)^k)^2. \qquad (14.3.1)\end{aligned}$$

We now show how to handle the expression $\sum_{j=1}^{f_i}(j^k - (j-1)^k)^2$. We first claim that

$$j^k - (j-1)^k \leq k \cdot j^{k-1}.$$

This follows from applying the mean value theorem to the function $f(x) = x^k$. Given two points $x_1 < x_2$, the mean value theorem states that there exists a number $\theta \in [x_1, x_2]$ such that $f'(\theta) = \frac{f(x_2)-f(x_1)}{x_2-x_1}$. We now substitute $j-1$ and j for x_1 and x_2 respectively, and observe that $f'(\theta) = k\theta^{k-1} \leq kx_2^{k-1}$ to get

$$j^k - (j-1)^k \leq k \cdot j^{k-1}.$$

Therefore,

$$\sum_{j=1}^{f_i}(j^k - (j-1)^k)^2 \leq \sum_{j=1}^{f_i} k \cdot j^{k-1} \cdot (j^k - (j-1)^k) \leq k \cdot f_i^{k-1} \sum_{j=1}^{f_i} (j^k - (j-1)^k) = k \cdot f_\star^{k-1} \cdot f_i^k,$$

where $f_\star$ denotes $\max_{i=1}^{n} f_i$. Substituting this in (14.3.1), we get

$$\mathbb{E}[Y^2] \leq k \cdot m \cdot f_\star^{k-1} F_k.$$

Recall that we wanted to bound $\mathbb{E}[Y^2]$ in terms of F_k^2. So we need to bound $m \cdot f_\star^{k-1}$ in terms of F_k. Clearly,

$$f_\star^{k-1} = (f_\star^k)^{\frac{k-1}{k}} \leq F_k^{\frac{k-1}{k}}.$$

In order to bound m, we apply Jensen's inequality to the convex function x^k to get

$$\left(\frac{\sum_{i=1}^n f_i}{n}\right)^k \leq \frac{\sum_{i=1}^n f_i^k}{n},$$

which implies that

$$m = \sum_{i=1}^n f_i \leq n^{1-1/k} \cdot F_k^{1/k}.$$

Combining all of the above inequalities, we see that

$$\mathbb{E}[Y^2] \leq k \cdot n^{1-1/k} \cdot F_k^2.$$

If we now use Chebychev's bound, we get

$$\Pr[|Y - F_k| \geq \varepsilon F_k] \leq \frac{\mathbb{E}[Y^2]}{\varepsilon^2 F_k^2} \leq k/\varepsilon^2 \cdot n^{1-1/k}.$$

The expression on the right hand side is (likely to be) larger than 1, and so this does not give us much information. The next idea is to further reduce the variance of Y by keeping several independent copies of it, and computing the average of all these copies. More formally, we maintain t i.i.d. random varaibles $Y_1, \ldots, Y_t$, each of which has the same distribution as that of Y. If we now define Z as the average of these random variables, linearity of expectation implies that $\mathbb{E}[Z]$ remains F_k. However, the variance of Z now decreases by a factor t.

Exercise 14.3 *Show that the variance of Z, denoted by $\sigma^2(Z)$, is equal to $\frac{1}{t} \cdot \sigma^2(Y) \leq \mathbb{E}[Y^2]/t$.*

Therefore, if we now use Z to estimate F_k, we get

$$\Pr[|Z - F_k| \geq \varepsilon F_k] \leq \frac{k}{t \cdot \varepsilon^2} \cdot n^{1-1/k}.$$

If we want to output an estimate within $(1 \pm \varepsilon)F_k$ with probability at least $1 - \delta$, we should pick t to be $\frac{1}{\delta\varepsilon^2} \cdot n^{1-1/k}$. It is easy to check that the space needed to update one copy of Y is $O(\log m + \log n)$. Thus, the total space requirement of our algorithm is $O\left(\frac{1}{\delta\varepsilon^2} \cdot n^{1-1/k} \cdot (\log m + \log n)\right)$.

Procedure Reservoir Sampling

```
1 X ← x_1 ;
2 for i = 2 to m do
3     Sample a binary random variable t_i, which is 1 with probability 1/i ;
4     if t_i = 1 then
5         X ← x_i
6 Return X
```

Procedure Combining reservoir sampling with the estimator for F_k

```
1 X ← x_1, r ← 1 ;
2 for i = 2 to m do
3     Sample a binary random variable t_i, which is 1 with probability 1/i ;
4     if t_i = 1 then
5         X ← x_i, r ← 1
6     else
7     if X = x_i then
          r ← r + 1 ;
8 Return m (r^k − (r − 1)^k);
```

Figure 14.4: Estimating F_k

14.3.1 The median of means trick

We now show that it is possible to obtain the same guarantees about Z, but we need to keep only $O\left(\frac{1}{\varepsilon^2} \cdot \log\left(\frac{1}{\delta}\right) \cdot n^{1-1/k}\right)$ copies of the estimator for F_k. Note that we have replaced the factor $1/\delta$ by $\log(1/\delta)$. The idea is that if we use only $t = \frac{4}{\varepsilon^2} \cdot n^{1-1/k}$ copies of the variable Y in the analysis above, then we will get

$$\Pr[|Z - F_k| \geq \varepsilon F_k] \leq 1/4.$$

Although this is not good enough for us, what if we keep several copies of Z (where each of these is average of several copies of Y) ? In fact, if we keep $\log(1/\delta)$ copies of Z, then at least one of these will give the desired accuracy with probability at least δ – indeed, the probability that all of them are at least εF_k far from F_k will be at most $(1/2)^{\log(1/\delta)} \leq \delta$. But we will not know *which* one of these copies is correct! Therefore, the plan is to keep slightly more copies of Z, say about $4\log(1/\delta)$. Using Chernoff bounds, we can show that with probability at least $1-\delta$, roughly a majority of these copies will give an estimate in the range $(1 \pm \varepsilon)F_k$. Therefore, the *median* of all these copies will give the desired answer. This is called the "median of means" trick.

We now give details of the above idea. We keep an array of variables Y_{ij}, where i varies from 1 to $\ell := 4\log(1/\delta)$ and j varies from 0 to $t := \frac{2}{\varepsilon^2} \cdot n^{1-1/k}$. Each row of this array (i.e., elements Y_{ij}, where we fix i and vary j) will correspond to one copy of the estimate described above. So, we define $Z_i = \sum_{j=1}^{t} Y_{ij}/t$. Finally, we define Z as the median of Z_i, for $i = 1, \ldots, \ell$. We now show that Z lies in the range $(1 \pm \varepsilon)F_k$ with probability at least $1-\delta$. Let E_i denote the event: $|Z_i - F_k| \geq \varepsilon F_k$. We already know that $\Pr[E_i] \leq 1/4$. Now, we want to show that the number of such events will be close to $\ell/4$. We can use Chernoff bound to prove this because these events are independent.

Exercise 14.4 *Prove that with probability at least $1-\delta$, the size of the set $\{i : E_i$ occurs$\}$ is at most $\ell/2$.*

Now assume the above happens. If we look at the sequence $Z_i, i = 1, \ldots, \ell$, at least half of them will lie in the range $(1 \pm \varepsilon)F_k$. The median of this sequence will also lie in the range $(1 \pm \varepsilon)F_k$ for the following reason: if the median is (say) above $(1+\varepsilon)F_k$, then at least half of the events E_i will occur, which is a contradiction. Thus, we have shown the following result:

Theorem 14.1 *We can estimate the frequency moment F_k of a stream with $(1 \pm \varepsilon)$ multiplicative error with probability at least $1-\delta$ using $O\left(\frac{1}{\varepsilon^2} \cdot \log\left(\frac{1}{\delta}\right) \cdot n^{1-1/k}\right) \cdot (\log m + \log n)$ space.*

14.3.2 The special case of second frequency moment

It turns out that we can estimate the second frequency moment F_2 using logarithmic space only (the above result shows that space requirement will be proportional to $\sqrt{n}$). The idea is again to have a random variable whose expected value is F_2, but now we will be able to control the variance in a much better way. We will use the idea of universal hash functions. We will require binary hash functions, i.e., they will map the set $U = \{e_1, \ldots, e_n\}$ to $\{-1, +1\}$. Recall that such a set of functions H is said to be k-universal if for any set S of indices of size at most k, and values $a_1, \ldots, a_k \in \{-1, +1\}$,

$$\Pr_{h \in H}[\wedge_{i \in S} x_i = a_i] = \frac{1}{2^{|S|}},$$

where h is a uniformly chosen hash function from H. Recall that we can construct such a set H which has $O(n^k)$ functions, and a hash function $h \in H$ can be stored using $O(k \log n)$ space only. We will need a set of 4-universal hash functions. Thus, we can store the hash function using $O(\log n)$ space only.

The algorithm for estimating F_2 is shown in Figure Second Frequency Moment. It maintains a running sum X – when the element x_t arrives, it first computes the hash value $h(x_t)$, and then adds $h(x_t)$ to X (so, we add either $+1$ or -1 to X). Finally, it outputs X^2. It is easy to check that expected value of X^2 is indeed F_2. First observe that if f_i denotes the frequency of element e_i. then $X = \sum_{i=1}^{n} f_i \cdot h(e_i)$. Therefore, using linearity of expectation,

$$\mathbb{E}[X^2] = \sum_{i=1}^{n} \sum_{j=1}^{n} f_i f_j \mathbb{E}[h(e_i)h(e_j)].$$

The sum above splits into two parts: if $i = j$, then $h(e_i)h(e_j) = h(e_i)^2 = 1$; and if $i \neq j$, then the fact that H is 4-universal implies that $h(e_i)$ and $h(e_j)$ are pair-wise independent random variables. Therefore, $\mathbb{E}[h(e_i)h(e_j)] = \mathbb{E}[h(e_i)] \cdot \mathbb{E}[h(e_j)] = 0$, because $h(e_i)$ is ± 1 with equal probability. So

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

As before, we want to show that X^2 comes close to F_2 with high probability. We need to bound the variance of X^2, which is at most $\mathbb{E}[X^4]$. As above, we expand take the fourth power of the expression of X:

$$\mathbb{E}[X^2] = \sum_{i,j,k,l=1}^{n} f_i f_j f_k f_l \mathbb{E}[h(e_i)h(e_j)h(e_k)h(e_l)].$$

Procedure Second Frequency Moment

$X \leftarrow 0$, $h \leftarrow$ uniformly chosen ± 1 hash function from a 4-universal family. ;
for $i = 1$ *to* m **do**
  $X \leftarrow X + h(x_i)$
Return X^2

Figure 14.5: Estimating F_2

Each of the summands is a product of 4 terms – $h(e_i), h(e_j), h(e_k), h(e_l)$. Consider such a term. If an index is distinct from the remaining three indices, then we see that its expected value is 0. For example, if i is different from j, k, l, then $\mathbb{E}[h(e_i)h(e_j)h(e_k)h(e_l)] = \mathbb{E}[h(e_i)]\mathbb{E}[h(e_j)h(e_k)h(e_l)]$ (we are using 4-universal property here – any set of 4 distinct hash values are mutually independent). But $\mathbb{E}[h(e_i)] = 0$, and so the expected value of the whole term is 0. Thus, there are only two cases when the summand need not be 0: (i) all the four indices i, j, k, l are same – in this case $\mathbb{E}[h(e_i)h(e_j)h(e_k)h(e_l)] = \mathbb{E}[h(e_i)^4] = 1$, because $h(e_i)^2 = 1$, or (ii) exactly two of i, j, k, l take one value and the other two indices take another value – for example, $i = j$, $k = l$, but $i \neq k$. In this case, we again get $\mathbb{E}[h(e_i)h(e_j)h(e_k)h(e_l)] = \mathbb{E}[h(e_i)^2 h(e_k)^2 = 1$. Thus, we can simplify

$$\mathbb{E}[X^4] = \sum_{i=1}^{n} f_i^4 + \sum_{i=1}^{n} \sum_{j \in \{1,\ldots,n\} \setminus \{i\}} f_i^2 f_j^2 \leq 2F_2^2.$$

Thus we see that the variance of the estimator X^2 is at most $2\mathbb{E}[X^2]^2$. Rest of the idea is the same as in the previous section.