1 Lecture 6

Frequency Moments and Counting Distinct Elements

Definition 1.1. Suppose that we receive m objects (not necessarily distinct) that are members of the set $[n]$. This set of m objects will, as they appear in a stream, be labeled σ; we have $B << m$ bits to process σ. Let g be a real valued non-negative function that operates on streams.

Definition 1.2. We say that a streaming algorithm $\mathcal{A}$ provides (ϵ, δ) additive approximation if

$$\mathbb{P}\left[|\mathcal{A} - g| > \epsilon\right] < \delta$$

J

Definition 1.3. We say that a streaming algorithm $\mathcal{A}$ provides (ϵ, δ) relative approximation if

$$\mathbb{P}\left[\frac{\mathcal{A}(\delta)}{g(\delta)} - 1 > \epsilon\right] < \delta$$

Remark 1.4. Many streaming problems are trivial to solve deterministically, if we have full access to the data that enters a stream. To solve them deterministically, however, there are lower bounds on the space and running time of working algorithms. These bounds are often unacceptable. To achieve useful space and running times, thus, streaming algorithms often end up using randomized algorithms.

Remark 1.5. The task of counting how many distinct elements appear in a stream has obvious, if costly, deterministic solutions: Put all objects in a binary search tree; when a new stream object appears, search the tree for the object and, if not found, add it to the tree. We call this task the *distinct elements problem.*

This requires $O(k)$ space and $O(m \log k)$ running time where k is the true number of distinct objects. Even if you were told in advance that there would be k objects, a hashing solution would require $O(k)$ space and $O(m)$ running time.

Remark 1.6. A cute idea is to assume the existence of a hash function $h : [n] \to [n^3]$ that closely "resembles" the theoretical, non-existent hash function that maps all objects with perfect randomness to some real in $[0, 1]$. Then keep track of the random varible $X = \min_{e_i \in \text{ all objects}} h(e_i)$. The minimum statistic (for k uniform random variables over $[0, 1]$) has the property that $\mathbb{E}\left[[]\, X\right] = \frac{1}{k+1}$. Thus, at the end of streaming, take X, invert it and subtract 1 from it to obtain a statistical estimator of the number of distinct objects.

Convince yourself that the following integral represents the expectation.

$$\mathbb{E}\left[X\right] = \int_0^1 (\frac{1}{1-0}) \binom{k}{1} (1-x)^{k-1}$$

Theorem 1.7. Using the definitions set above, we can achieve a $1 + \epsilon$ accurate estimate of X with probability $1 - \delta$; if we kept track of the minimum hash value seen using only $O(\log n)$ bits, then

1.1 BJKST

In what follows, I give a high level overview of how to understand the analysis of BJKST

- What does the algorithm do?
  - The algorithm repeatedly hashes elements as they appear and stores the t smallest hash values.We use some hash function taken from a 2-universal hash family; moreover, this hash function has the formula $[n] \mapsto [n^3]$. It returns a statistic that depends on the average hash value of the tth smallest hash value.
- How do we analyze the algorithm?
  - First, we need to understand the statistics at place. Assume that v is the hash value of the tth smallest hash. Then:
    - The minimum of the hash function, assuming d distinct elements would fall around $\frac{1}{d+1}$ if we were working with an ideal hash function.
    - Since we work over $[N]$ instead of $[0,1]^3$, however, it seems reaonable that this will instead be $\frac{N}{d+1}$. Moreover, the tth hash – let's call it v – will hash to $\frac{tN}{d+1}$ on average (we're waving our hands, here). So

$$v = \frac{tN}{d+1} \implies (d+1) = \frac{tN}{v} \implies \approx d = \frac{tN}{v}$$

- We will have the algorithm described above return $\frac{tN}{v}$ as our estimate for d. We would like to have a bound of the form:

$$\mathbb{P}\left[|D-d| > \epsilon d\right] \leq 1/3$$

  For then we can apply the median trick.

  - Show that there is a low chance that more than half of this experiment's trials are all bad. On average, $l/3$ of the trials will be bad; we want to bound the probability that more than $l/2$ of the trials will be bad, for that will imply the median is good. Use the bound $\mathbb{P}\left[X \geq (1+\delta)\mu\right] \leq \exp \frac{-\delta^2\mu}{3}$.
- This is decomposable into exactly two, disjoint events:

$$\mathbb{P}\left[D > d(\epsilon+1)\right] + \mathbb{P}\left[D < d(1-\epsilon)\right]$$

- Focus on

$$\mathbb{P}\left[D < d(1-\epsilon)\right]$$

  It is more tractable to explore this inequality when we work with the concrete objet v, the object whose hash was the tth smallest of all hashes.

$$D < d(1-\epsilon) \iff tN/v < d(1-\epsilon) \iff v > \frac{tN}{d(1-\epsilon)}$$

- The foregoing occurs iff lesss than t objects fell into $\frac{tN}{d(1-\epsilon)}$.
    - * The probability that a single distinct element falls in this range is

$$\frac{t}{d(1-\epsilon)}$$

Let $X = \sum_{i=1}^{d} X_i$. We have that

$$\begin{aligned} \mathbb{E}[X_i] &= \frac{t}{d(1-\epsilon)} > \frac{t(1+\epsilon)}{d} \\ \mathbb{E}[X] &= \frac{t}{1-\epsilon} > \frac{t(1+\epsilon)}{1} \\ \mathrm{Var}[X_i] < \mathbb{E}[X_i] &< \frac{t(1+3\epsilon/2)}{d} \\ \mathrm{Var}[X] < \mathbb{E}[X] &< \frac{t(1+3\epsilon/2)}{1} \end{aligned}$$

Notice that we were originally interested in the value $\mathbb{P}[X < t]$

$$\mathbb{P}[X < t] \leq \mathbb{P}[|X - \mathbb{E}[X]| > \epsilon t] \leq \mathrm{Var}[X]/(\epsilon t)^2 \leq \frac{t(1+3\epsilon/2)}{c}$$

If we make c sufficiently large, then we get a desireable inequality

---

- Now focus on

$$\mathbb{P}[D > d(\epsilon+1)]$$

- As before, we find that the event $D > d(\epsilon+1)$ is synonymous with the event that $v < \frac{tN}{(1+\epsilon)d}$.
    - This is the event that at least t values hash inside $[1 \dots \frac{tN}{(1+\epsilon)d}]$
- The probability that any one value hashes inside this range is

$$\frac{t}{(1+\epsilon)d} \leq \frac{t(1-\epsilon/2)}{d}$$

The latter inequality follows from the fact that $\frac{1}{1+\epsilon}$ is an alternating series[1] and that $\epsilon < 1/2$. We want to bound

$$\mathbb{P}[X \geq t]$$

[1]You can prove that odd partial sums are decreasing and even partial sums are increasing.

$$\mathbb{P}[X \geq t] \leq \mathbb{P}[|X - \mathbb{E}[X]| > \epsilon t/2 - q] \leq \frac{4\mathrm{Var}[X]}{(\epsilon t - 2q)^2} \approx \frac{4\mathrm{Var}[X]}{c}$$

Now choose c to make RHS less than $\frac{1}{6}$

- The preceding holds, because as before, it can be shown that

$$\mathrm{Var}[X] < \frac{t(1-\epsilon/2)}{}$$

- Recall that we need c/ϵ^2 elements. c here is fixed, once we find a sufficiently large value for it. Thus we need $\log(1/\epsilon^2)\log n$ bits; and we perform the median trick $\log(1/\delta)$ times.
  - In total we need $O(\frac{1}{\epsilon^2}\log(1/\delta)\log n)$ bits.

2 Lecture 7

Definition 2.1. A stream is a sequence $O = \{e_1 \dots e_m\}$ where $e_i \in [n]$. For each $j \in [n]$, $f_j = |O \cap j|$, which is the number of times that j appears in O.

Definition 2.2. A function $g : \mathbb{R} \to \mathbb{R}$ of a stream σ satisfies:

$$g(0) = 0$$
$$g(\sigma) = \sum_{i=1}^{n} g_i(f_i)$$

Example 2.3. Let $g_i = h(x) = x^k$ for each i. Then $g(\sigma)$ is the kth frequency moment.

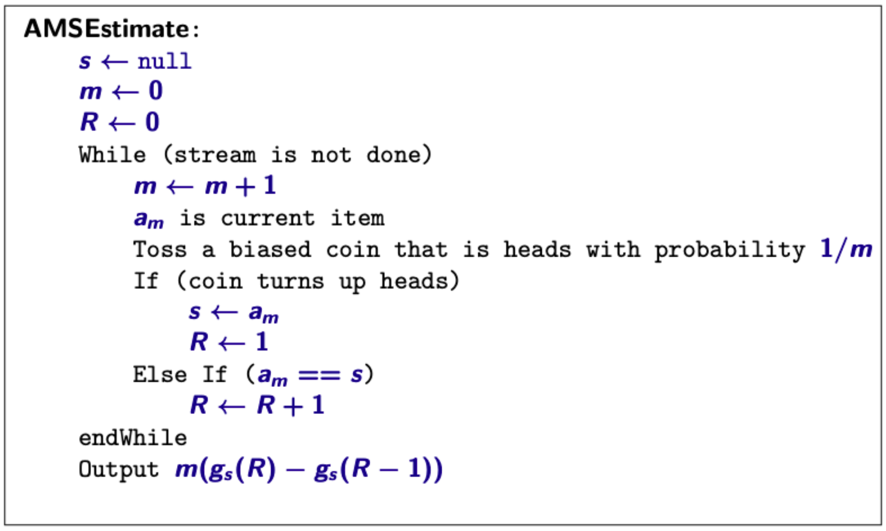
Remark 2.4. AMS sampling does the following:

- Draw an element e_J uniformly at random from $\{e_1 \dots e_m\}$.
  - $e_J = i$ for some $i \in [n]$.
- Count how many times i appears out of $\{e_j | j \geq J\}$.
- Let

$$R = \left|\{e_j | j \geq J\}\right|$$

- Output $g_i(R) - g_i(R-1)$

Lemma 2.5. Let Y be the output of AMS. Then $\mathbb{E}[Y] = g(\sigma) = \sum_{i=1}^{n} g_i(f_i)$.



```
AMSEstimate:
    $s \leftarrow$ null
    $m \leftarrow 0$
    $R \leftarrow 0$
    While (stream is not done)
        $m \leftarrow m + 1$
        $a_m$ is current item
        Toss a biased coin that is heads with probability $1/m$
        If (coin turns up heads)
            $s \leftarrow a_m$
            $R \leftarrow 1$
        Else If ($a_m == s$)
            $R \leftarrow R + 1$
    endWhile
    Output $m(g_s(R) - g_s(R - 1))$
```

Proof.

Let e_J be the random variable whose values is the element in $[n]$ that we choose.

$$\begin{aligned}
\mathbb{E}[Y] &= \mathbb{E}[\mathbb{E}[Y|e_J]] \\
&= \sum_{i=1}^{n} \mathbb{P}[e_J = i]\,\mathbb{E}[Y|e_J = i] \\
&= \sum_{i=1}^{n} \mathbb{P}[e_J = i]\,\mathbb{E}[Y|e_J = i] \\
&= \sum_{i=1}^{n} \frac{f_i}{m}\mathbb{E}[Y|e_J = i] \\
&= \sum_{i=1}^{n} \frac{f_i}{m} \sum_{l=1}^{f_i} \frac{1}{f_i} m\,(g(l) = g(l-1)) \\
&= \sum_{i=1}^{n} g_i(f_i)
\end{aligned}$$

□

Remark 2.6. At this point, calculations are given that bound $\mathrm{Var}[Y]$. The calculations are not very illuminating, so I omit showing them. They are at the very end of the February 5th slides:
`https://courses.engr.illinois.edu/cs498abd/sp2019/slides/07.pdf`

3 Lecture 8

Estimation of F_2

Remark 3.1. F_2 is special for "fundamental" reasons.

Unresolved 3.2. The previous lecture's AMS algorithm purportedly establishes that we can estimate F_2 in $\sqrt{n}\log n$ space. The analysis revealed that we need $O(\sqrt{n})$ versions of the algorithm to run in parallel. It is not clear, however, that each version requires $\log n$ space – indeed, the algorithm needs counters that are at most m, which would suggest $\log m$ space. In the event that $m \in O(n^k)$ for some k, then this would be okay.

Update: It is likely that we need $\log(n)$ space to store the hash function.

Lemma 3.3. Consider the following algorithm:

```
AMS-$F_2$-Estimate:
  Suppose that $h$ is chosen from a 4-wise independent hash family with formula $[n] \to \{-1, 1\}$.
  $z \leftarrow 0$
  For each $e_j$:
      $z \leftarrow h(e_j)$
  Output $z^2$.
```

This algorithm estimates F_2 with (ϵ, δ) accuracy if we use $O(\log(1/\delta)\frac{1}{\epsilon^2}\log n)$ bits.

Proof. See lecture notes 8. `https://courses.engr.illinois.edu/cs498abd/sp2019/slides/08.pdf` □

Remark 3.4. This algorithm can be extended to estimate not just the 2 norm of the frequency vector but the square of the euclidean norm of any vector (and hence the euclidean norm itself). This is again covered in his lecture notes. In fact, wherever in the proof of the AMS-F_2 algoritm one uses f_i, just replace it by x_i.

Remark 3.5. k-wise independent families satisfy the property that for any $S = \{a_1 ... a_k\} \subseteq [n]$, we have that all elements in S are independent. In particular, a k-wise independent hash family guarantees that the hash of any input is uniformly distributed and that any 4 hashes are independent.

Definition 3.6. A sketch c of a stream σ is a function such that $c(\sigma_1 \cdot \sigma_2) = c(\sigma_1) * c(\sigma_2)$. Here, $*$ is some operation and σ_i are subsets of the stream and $\cdot$ is concatenation.

Remark 3.7. The previous algorithm A is a linear sketch[2]. For if we use the same hash function over two streams σ_1 and σ_2 then $(A(\sigma_1) + A(\sigma_2))^2 = A(\sigma_1 \cdot \sigma_2)^2$.

Remark 3.8. If we wish to run this algorithm in order to attain (ϵ, δ) accuracy, then could formulate it as follows:

Create the matrix

$$M = \begin{bmatrix} h_1(1) ... h_1(n) \\ \vdots \\ h_{O(\frac{1}{\epsilon^2}\log(1/\delta))}(1) ... h_{O(\frac{1}{\epsilon^2}\log(1/\delta))}(n) \end{bmatrix}$$

[2]Note that the algorithm returns z and one must additionally post-process this return value by returning z^2, so that we treat the algorithm as solely returning z

Then when the token (i_t, δ_{i_t}) where $i_t \in [n]$ for $t \in [m]$ appears, update a vector Z of initially $O(\frac{1}{\epsilon^2}\log(1/\delta))$ zeros so that $Z = Z + \delta_i * M(e_{i_t})$. At the end, we will have a vector of algorithm estimates, which we should take the median q of and then output q^2. In this formulation, we see that if we havea single M across two runs of this experiment and let

$$x_q = \sum_{t=1}^{m} \delta_{i_t} e_{i_t}$$

for $q \in \{1, 2\}$ (for both runs) then,

$$M(x_1 + x_2) = Mx_1 + Mx_2$$

Here $x_1 + x_2$ represents the action of taking the concatenation of both streams. This gives some insight as to why the algorithm is called a linear sketch.

4 Lecture 9

Heavy Hitters

Remark 4.1. Note that the algorithm to the determine the majority element can be shown to output the correct member, if there is in fact a majority element. If there is no majority element, there may be garbage.

Remark 4.2. The deterministic algorithm that returns those elements whose frequency is greater than m/k for some k will at least output those elements $i \in [n]$ such that $f_i \geq m/k$; however, this algorithm may also erroneously return those i such that $f_i < m/k$.

Unresolved 4.3. A second pass will

<++>

<++>