COMS 4995-3: Advanced Algorithms — Jan 18, 2017

Lecture 2 – Approximate Counting and Hashing

Instructor: *Alex Andoni* — Scribes: *Arushi Gupta, Yogesh Garg*

1 Approximate Counting (continued)

Suppose we want to count e.g. the number of suspicious packages that we encounter. We can do this in $\mathcal{O}(n)$ space complexity, but if we also want to limit the space complexity, we have to resort to

- Approximation
- Randomization

We saw one solution using Morris Algorithm last time.

1.1 The Morris Algorithm

– Initialize X=0

– at each tick, set

$$X = \begin{cases} X+1 \text{ w.p. } 2^{-X} \\ \text{unchanged otherwise} \end{cases} \tag{1}$$

– the end estimator is $2^X - 1$

Claim 1. *If* X_n *= the value after* n *ticks then* $\mathbb{E}[2^{X_n} - 1] = n$

Proof. (Given last time) □

Claim 2. *Storing* X *takes* $\mathcal{O}(\lg \lg n)$ *bits with probability* $\geq 90\%$

Proof. By the Markov Bound,

$$\mathbb{P}[2^{X_n} - 1 > 2n] \leq \frac{1}{2} \tag{2}$$

suppose,

$$2^{X_n} - 1 \leq 2n \tag{3}$$

$$X_n \leq \lg(2n+1) \tag{4}$$

Recall that the number of bits required to store X_n is $\lg(X_n)$. This means that the space required to store X is:

$$\lg(X_n) \leq \lg\lg(2n+1) \tag{5}$$

$$= \mathcal{O}(\lg\lg n) \tag{6}$$

$\square$

Claim 3. $\mathrm{var}[2^{X_n} - 1] \leq \frac{3n(n+1)}{2} + 1$

Proof.

$$\begin{aligned}
\mathrm{var}[2^{X_n} - 1] &= \mathbb{E}[(2^{X_n} - 1)^2] - n^2 \\
&= \mathbb{E}[2^{2X_n}] + \underbrace{1 - 2\mathbb{E}[2^{X_n}] - n^2}_{\leq 0} \\
&\leq \mathbb{E}[2^{2X_n}] \\
\mathbb{E}[2^{2X_n}] &= \sum_i 2^{2i}\mathbb{P}[X_n = i] \\
&= \sum_i 2^{2i}\left(2^{-(i-1)}\mathbb{P}[X_{n-1} = i-1] + (1 - 2^{-i})\mathbb{P}[X_{n-1} = i]\right) \qquad (7) \\
&= \sum_i 2^{i+1}\mathbb{P}[X_{n-1} = i-1] + \sum_i 2^{2i}\mathbb{P}[X_{n-1} = i] - \sum_i 2^{i}\mathbb{P}[X_{n-1} = i] \\
&= 4\mathbb{E}[2^{X_{n-1}}] + \mathbb{E}[2^{2X_{n-1}}] - \mathbb{E}[2^{X_{n-1}}] \\
&= 3\underbrace{\mathbb{E}[2^{X_{n-1}}]}_{=n} + \mathbb{E}[2^{2X_{n-1}}] \\
&= 3n + \mathbb{E}[2^{2X_{n-1}}]
\end{aligned}$$

We now apply induction
Base case:

$$\mathbb{E}[2^{X_0}] = 1 \qquad (8)$$

Inductive case:

$$\begin{aligned}
\mathbb{E}[2^{2X_n}] &= \sum_{i=1}^{n} 3i + 1 \\
&= \frac{3}{2}n(n+1) + 1
\end{aligned} \qquad (9)$$

$\square$

1.2 Bound for 2^{X_n}

Applying the Chebyshev's inequality to 2^{X_n},

$$\mathbb{P}[|(2^{X_n} - 1) - n| > \lambda)] \leq \frac{\mathrm{var}[2^{X_n} - 1]}{\lambda^2} \qquad (10)$$

target probability is $= \frac{1}{2}$ which means that we want

$$\lambda^2 = 2\,\mathrm{var}[2^{X_n} - 1] = 3n(n+1) + 2 \qquad (11)$$

With probability $\geq \frac{1}{2}$, the estimator $= n \pm \lambda \approx n \pm \sqrt{3}n$.
This gives us an upper bound but we can observe that the estimator can output zeros, so this is not a

particularly good bound. If possible, we would like a tighter concentration bound.

We can use an often used trick to achieve this: We can compute a few such estimators and average them

1.3 The Morris + Algorithm

- run in parallel k copies of the Morris Algorithm
- counters are called $X^1, .., X^k$
- our new estimator is $Y = \frac{1}{k}\sum_{j=1}^{k}(2^{X^j} - 1)$

Claim 4. $\mathbb{E}[Y_n] = n$

Proof. Using linearity of expectation,

$$\begin{aligned}\mathbb{E}[Y_n] &= \mathbb{E}[\frac{1}{k}\sum_j X_n^j] \\ &= \frac{1}{k}\underbrace{(n + n + ... + n)}_{\text{k times}} = n\end{aligned} \tag{12}$$

$\square$

Claim 5. *Space is* $\mathcal{O}(k \lg\lg n)$

Claim 6. $\mathrm{var}[Y_n] = \frac{1}{k}\mathcal{O}(n^2)$

Proof.

$$\begin{aligned}\mathrm{var}[Y_n] &= \mathrm{var}\left[\frac{1}{k}\sum_{j=1}^{k}\left(2^{X_n^j} - 1\right)\right] \\ &= \sum_{j=1}^{k}\mathrm{var}\left[\frac{1}{k}\left(2^{X_n^j} - 1\right)\right] \\ &= \sum_{j=1}^{k}\frac{1}{k^2}\,\mathrm{var}\left[2^{X_n^j} - 1\right] \\ &= \frac{1}{k}\underbrace{\mathrm{var}\left[2^{X_n} - 1\right]}_{\mathcal{O}(n^2)} \\ &= \frac{1}{k}\mathcal{O}(n^2)\end{aligned} \tag{13}$$

$\square$

1.3.1 Bound on Morris+

We apply Chebyshev's inequality to Y_n, and we want

$$\begin{aligned} \lambda^2 = 2\operatorname{var}[Y_n] = \frac{3n(n+2)+2}{k} \\ \Rightarrow \lambda = \frac{1}{\sqrt{k}}\mathcal{O}(n) \end{aligned} \tag{14}$$

Thus, estimator

$$Y_n = n \pm \frac{1}{\sqrt{k}}\mathcal{O}(n) \tag{15}$$

with probability $\geq \frac{1}{2}$, where $k = \mathcal{O}(\frac{1}{\epsilon^2})$ and we have a $1+\epsilon$ approximation.

Theorem 7. *For $k = \mathcal{O}(\frac{1}{\epsilon^2})$, Morris+ algorithm outputs*
a value $\in [n - \epsilon n, n + \epsilon n]$
w.p. $\geq \frac{1}{2}$.
Space is $\mathcal{O}\left(\frac{1}{\epsilon^2}\lg\lg n\right)$.

Observation 8. *Follows:*

- If we want probability $\geq 1 - \delta$, where δ = small failure probability, the same argument gives

$$k = \mathcal{O}(\frac{1}{\epsilon^2}\frac{1}{\delta}) \tag{16}$$

- It is possible to get

$$k = \mathcal{O}(\frac{1}{\epsilon^2}\log\frac{1}{\delta}) \tag{17}$$

  But it requires calculating higher moments.

2 Hashing

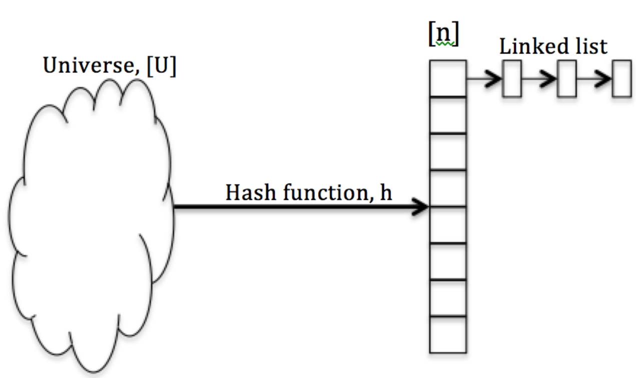

Figure 1: Hashing Function and Collisions

We can think about a universe (a big set of numbers) of size $U \in \mathbb{N}$. We use the notation

$$[N] = \{1, 2, ..., N\} \tag{18}$$

As an example, we can think about the universe of IP addresses, of which there are

$$U = 2^{128} \tag{19}$$

We would like to solve the following (static) dictionary problem:

Definition 9. *Given a set $S \subset [U]$, $|S| = m$, We need to resolve query Q_x : given x, decide quickly whether $x \in S$ or not.*

Possible solutions:

1. Just use a list. The time is $\mathcal{O}(m)$
2. Use a binary tree. Time is $\mathcal{O}(\lg m)$

This is not sufficient, we would like to get a better performance.

Definition 10. *A hash function h, is*

$$h : [U] \to [n] \tag{20}$$

Typically,

$$n \approx m \ll U \tag{21}$$

So a hash function maps members of a large universe into members of a smaller set. Since U is larger than n, there are likely to be collisions, i.e. it is likely that two elements of $[U]$ will map to the same element in $[n]$, i.e.

$$h(x) = h(y) \tag{22}$$

How do we deal with collisions? Well, we can store a linked list associated with each element of $[n]$, that contains all the elements of $[U]$ that map to that element of $[n]$. So how long does it take for us to solve our original problems Q_x of determining whether $x \in S$? It depends on the size of the bucket, the linked list, associated with x. We want to minimize collisions so that this bucket is as small as possible. So how do we choose h?

2.1 Knuth's solution

$$h(x) = \lfloor \{ \frac{\sqrt{5} - 1}{2} x \} n \rfloor \tag{23}$$

Where the braces represent the fractional part of the expression they surround. The problem with this, is that because it's a deterministic function, it's possible to construct a specific set, S, for which there are lots of collisions , and this can be a security risk.

2.2 Solution 1: A completely randomized approach

So we can choose h so that it is not deterministic by making it completely random. We pick each $h(x)$ as a random number. But how much space would it take to store $[n]$? $\mathcal{O}(U \cdot \lg n)$ which is way too big.

2.3 Solution 1.5

Store a bit array of size U

2.4 Solution 2: Limited Randomness

Definition 11. *$h : [U] \to [n]$ is called universal if*

$$\begin{aligned} &\forall x, y \in [U] \\ &\mathbb{P}[h(x) = h(y)] = \frac{1}{n} \end{aligned} \tag{24}$$

Notice that we are considering the fraction $\frac{1}{n}$ because this is what it would be in the completely randomized case.

Claim 12. *Suppose the number of collisions is C and h is a universal function. Then $\mathbb{E}[C] = \frac{m(m-1)}{2n}$*

Proof. Let $\mathbb{1}$ be the indicator function.

$$\begin{aligned} \mathbb{E}[C] &= \mathbb{E}\left[\sum_{x,y \in S} \mathbb{1}_{h(x)=h(y)}\right] \\ &= \binom{m}{2} \mathbb{E}_h\left[\mathbb{1}_{h(x)=h(y)}\right] \\ &= \binom{m}{2} \frac{1}{n} \\ &= \frac{m(m-1)}{2n} \end{aligned} \tag{25}$$

□

If we want $\mathbb{E}[C] < 1$, we need to set $n = \frac{m(m-1)}{2} = \Theta(m^2)$