1 Approximate Counting (continued)

Suppose we want to count e.g. the number of suspicious packages that we encounter. We can do this in $O(n)$ 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

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

- the end estimator is $2^X - 1$

Claim 1. If $X_n$ = the value after $n$ ticks then $E[2^X - 1] = n$

Proof. (Given last time) \qed

Claim 2. Storing $X$ takes $O(\lg \lg n)$ bits with probability $\geq 90$

Proof. By the Markov Bound,

$$ \Pr[2^X - 1 > 2n] \leq \frac{1}{2} \quad (2) $$

suppose,

$$ 2^X - 1 \leq 2n \quad (3) $$

$$ X_n \leq \lg(2n + 1) \quad (4) $$

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:

$$ \lg(X_n) \leq \lg(\lg(2n + 1)) \quad (5) $$

$$ = O(\lg \lg n) \quad (6) $$
Claim 3. $\text{var}[2^{X_n} - 1] \leq \frac{3n(n+1)}{2} + 1$

Proof.

\[
\text{var}[2^{X_n} - 1] = \mathbb{E}[(2^{X_n} - 1)^2] - n^2
\]
\[
= \mathbb{E}[2^{2X_n}] + 1 - 2\mathbb{E}[2^{X_n}] - n^2
\]
\[
\leq \mathbb{E}[2^{2X_n}]
\]
\[
\mathbb{E}[2^{2X_n}] = \sum_i 2^{2i} \mathbb{P}[X_n = i]
\]
\[
= \sum_i 2^{2i} \left( 2^{-i} \mathbb{P}[X_{n-1} = i - 1] + (1 - 2^{-i}) \mathbb{P}[X_{n-1} = i] \right)
\]
\[
= \sum_i 2^{i+1} \mathbb{P}[X_{n-1} = i - 1] + \sum_i 2^{2i} \mathbb{P}[X_{n-1} = i] - \sum_i 2^i \mathbb{P}[X_{n-1} = i]
\]
\[
= 4\mathbb{E}[2^{X_{n-1}}] + \mathbb{E}[2^{2X_{n-1}}] - \mathbb{E}[2^{X_{n-1}}]
\]
\[
= 3\mathbb{E}[2^{X_{n-1}}] + \mathbb{E}[2^{2X_{n-1}}]
\]
\[
= 3n + \mathbb{E}[2^{2X_{n-1}}]
\]

We now apply induction

**Base case:**

\[
\mathbb{E}[2^{X_0}] = 1
\]

**Inductive case:**

\[
\mathbb{E}[2^{2X_n}] = \sum_{i=1}^{n} 3i + 1
\]
\[
= \frac{3}{2} n(n+1) + 1
\]

1.2 Bound for $2^{X_n}$

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

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

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

\[
\lambda^2 = 2 \text{var}[2^{X_n} - 1] = 3n(n + 1) + 2
\]

With probability $\geq \frac{1}{2}$, the estimator $= n \pm \lambda \approx n \pm \sqrt{3n}$.

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,

$$
\mathbb{E}[Y_n] = \mathbb{E}\left[\frac{1}{k} \sum_{j} X^j_n\right]
= \frac{1}{k} (n + n + ... + n) = n
\hspace{1cm} (12)
$$

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

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

Proof.

$$
\text{var}[Y_n] = \text{var}\left[\frac{1}{k} \sum_{j=1}^{k} (2^{X^j_n} - 1)\right]
= \frac{1}{k^2} \sum_{j=1}^{k} \text{var}\left(2^{X^j_n} - 1\right)
= \frac{1}{k^2} \sum_{j=1}^{k} \text{var}\left(2^{X^j_n} - 1\right)
\hspace{1cm} (13)
$$

$\square$
1.3.1 Bound on Morris+

We apply Chebyshev’s inequality to $Y_n$, and we want

$$\lambda^2 = 2 \text{var}[Y_n] = \frac{3n(n+2) + 2}{k}$$

$$\Rightarrow \lambda = \frac{1}{\sqrt{k}} \mathcal{O}(n)$$

Thus, estimator

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

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 $\mathcal{O}(\frac{1}{\epsilon^2} \log \log n)$.

**Observation 8.** Follows:

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

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

- It is possible to get

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

But it requires calculating higher moments.

2 Hashing

![Hashing Function and Collisions](image)

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

$$[N] = \{1, 2, ..., N\}$$  \hspace{1cm} (18)

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

$$U = 2^{128}$$  \hspace{1cm} (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 $O(m)$
2. Use a binary tree. Time is $O(\lg m)$

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

**Definition 10.** A hash function $h$, is

$$h : [U] \rightarrow [n]$$  \hspace{1cm} (20)

Typically,

$$n \approx m \ll U$$  \hspace{1cm} (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.

$$h(x) = h(y)$$  \hspace{1cm} (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**

$$h(x) = \lfloor \{ \frac{\sqrt{5} - 1}{2} x \} n \rfloor$$  \hspace{1cm} (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 $[n]$? $O(U \cdot \lg n)$ 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

\[
\forall x, y \in [U] \quad P[h(x) = h(y)] = \frac{1}{n}
\]

(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.

\[
\mathbb{E}[C] = \mathbb{E} \left[ \sum_{x,y \in S} \mathbb{1}_{h(x)=h(y)} \right]
\]

\[
= \left( \frac{m}{2} \right) \mathbb{E}_h \left[ \mathbb{1}_{h(x)=h(y)} \right]
\]

\[
= \left( \frac{m}{2} \right) \frac{1}{n}
\]

\[
= \frac{m(m-1)}{2n}
\]

(25)

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