COMS 4995-8: Advanced Algorithms (Spring'21)	Jan 14, 2021

Lecture 2: Approximate Counting, Intro to Hashing

Instructor: *Alex Andoni* Scribes: *Akshat Agarwal, Julia Martin*

1 Recap

To exactly count up to n we cannot do better than $\Omega(\log n)$ space.
Hence, we were trying to device an approximate algorithm to count up to n which uses space $o(\log n)$.
Introduced Morris' Algorithm which is an Approximate and Randomized algorithm for doing just that.

2 Morris' Algorithm Continued

Approximate and Randomized counting algorithm with reduced space usage:
- Initialize $X = 0$
- onButtonPress: $X = \begin{cases} X, \text{ with probability } 1 - 2^{-X} \\ X + 1 \text{ with probability } 2^{-X} \end{cases}$
- Estimate: $\hat{n} = 2^X - 1$

Note:
- The bigger X becomes, the less likely it is to increment

Doubt 1: Is it a good estimator?

- $\hat{n}$ is a Random Variable
- Our Goal is to get $\hat{n}$ as close to n as possible
- A good start to achieve this goal could be to make sure that the expected value of $\hat{n}$ is n

Claim 1. *Yes, it is good.* $\mathbb{E}[\hat{n}] = n$

Proof. Define X_n as the value of counter X after n button presses, and with X_0 initialized to 0. We want the show that $\mathbb{E}[\hat{n}] = \mathbb{E}[2^{X_n} - 1] = n$. We will prove by induction.

Base case: for $n = 0$, $X_0 = 0$; $E[2^{X_0} - 1] = 0$
Inductive Step: Assume for inductive hypothesis that $E[2^{X_{n-1}} - 1] = n - 1$
We want to show $E[2^{X_n} - 1] = n$.

We have:

$$
\begin{aligned}
E[2^{X_n}-1] &= E_{X_n,X_{n-1},\ldots,X_1}[2^{X_n}-1] \\
&= E_{X_{n-1},\ldots,X_1}[E_{X_n}[2^{X_n}-1]] \\
&= E_{X_{n-1},\ldots,X_1}[2^{-X_{n-1}}(2^{X_{n-1}+1}-1)+(1-2^{-X_{n-1}})(2^{X_{n-1}}-1)] \\
&= E_{X_{n-1},\ldots,X_1}[2-2^{-X_{n-1}}+2^{X_{n-1}}-1-1+2^{-X_{n-1}}] \\
&= E_{X_{n-1},\ldots,X_1}[2^{X_{n-1}}] \\
&= E[(2^{X_{n-1}}-1)+1] \\
&= (n-1)+1 && \text{(by inductive hypothesis)} \\
&= n
\end{aligned}
$$

□

Doubt 2: Did we save space?

- Did the hassle pay off?
- Now we want to prove that the number of bits necessary to denote n has improved from $\log(n)$

Claim 2. *Yes we saved space. The number of bits taken by X, i.e. log(X), is of the order of log(log(n)) with probability $\geq$ 90%*

Proof. Apply the **Markov Bound** on $\hat{n}$, we know $\mathbb{E}[\hat{n}] = n$:

$$\Pr[\hat{n} > 10n] \leq \frac{n}{10n} = 0.1$$

$$\implies \Pr[2^{X_n}-1 \leq 10n] \geq 0.9$$

Now when $2^{X_n}-1 \leq 10n$ (this happens with 90% probability)

$$\implies 2^{X_n} \leq 10n+1$$

$$\implies \log\log 2^{X_n} \leq \log\log(10n+1)$$

$$\implies \log X_n \leq \log\log(10n+1)$$

Hence we get, $\Pr[\log X = O(\log\log n)] \geq 0.9$

□

Doubt 3: How far below the actual n can our estimator $\hat{n}$ be?

- Using Markov Bounds we saw that 90% of times $\hat{n} \leq 10n$
- With High Probability, our guess will be above n by no more than a constant factor multiple
- Now we want to check how low below the actual n can our guess be.
- We will find this using Chebyshev bounds. But for that we first need the Variance of $\hat{n}$

Claim 3. $\mathrm{Var}[\hat{n}] \leq \frac{3}{2}n(n+1) + 1 = O(n^2)$

Proof. We know,

$$\begin{aligned}
\mathrm{Var}[\hat{n}] &= \mathrm{Var}[2^{X_n} - 1] \\
&= \mathbb{E}[(2^{X_n} - 1)^2] - \mathbb{E}[(2^{X_n} - 1)]^2 \\
&= \mathbb{E}[(2^{X_n} - 1)^2] - n^2 \\
&= \mathbb{E}[2^{2X_n}] + \underbrace{1 - 2\underbrace{\mathbb{E}[2^{X_n}]}_{n+1} - n^2}_{\leq 0} && (-n^2 - 2n - 1 \leq 0) \\
&\leq \mathbb{E}[2^{2X_n}]
\end{aligned}$$

Inductive hypothesis: $\mathbb{E}[2^{2X_n}] \leq \frac{3}{2}n(n+1) + 1$
Base case: $n = 0 : \mathbb{E}[2^0] = 1 \leq 1$
Assume the inductive hypothesis for $\mathbb{E}[2^{2X_{n-1}}]$

Now we want to compute the expectation:

$$\begin{aligned}
E[2^{2X_n}] &= E_{X_n, X_{n-1}, \ldots, X_1}[2^{2X_n}] \\
&= E_{X_{n-1}, \ldots, X_1}[E_{X_n}[2^{2X_n}]] \\
&= E_{X_{n-1}, \ldots, X_1}[2^{-X_{n-1}}(2^{2(X_{n-1}+1)}) + (1 - 2^{-X_{n-1}})(2^{2X_{n-1}})] \\
&= E_{X_{n-1}, \ldots, X_1}[4 \cdot 2^{X_{n-1}} + 2^{2X_{n-1}} - 2^{X_{n-1}}] \\
&= E_{X_{n-1}, \ldots, X_1}[3 \cdot 2^{X_{n-1}} + 2^{2X_{n-1}}] \\
&= E[3 \cdot 2^{X_{n-1}} + 2^{2X_{n-1}}] \\
&= 3 \cdot \underbrace{\mathbb{E}[2^{X_{n-1}}]}_{n, \text{ by Claim 1}} + \underbrace{\mathbb{E}[2^{2X_{n-1}}]}_{\leq \frac{3}{2}(n-1)n+1} \\
&\leq \frac{3}{2}n(n+1) + 1 \\
&= O(n^2)
\end{aligned}$$

□

Claim 4. *Using the Chebyshev bound, we can find a lower bound and a tighter upper bound, such that:* $\hat{n} \in [n - 5n, n + 5n]$

Proof. So far, we have:

$$E[\hat{n}] = n$$

$$\mathrm{Var}[\hat{n}] \leq \frac{3}{2}n(n+1) + 1 \leq 2n^2$$

Apply the **Chebyshev Bound** on $\hat{n}$

$$\Pr[|\hat{n} - \mathbb{E}(\hat{n})| > \lambda] \leq \frac{\mathrm{Var}[\hat{n}]}{\lambda^2}$$

$$\Pr[|\hat{n} - n| > \lambda] \leq \frac{2n^2}{\lambda^2}$$

We approximately want this probability to be ≤ 0.1, therefore:

$$\frac{2n^2}{\lambda^2} = 0.1$$

$$2n^2 \leq 0.1\lambda^2$$

$$\lambda \geq \sqrt{20n}$$

Enough to have $\lambda = 5n$, We have:

$$\Pr[|\hat{n} - n| > 5n] \leq 0.1$$

With Probability 90%:

$$\hat{n} \in [n - 5n, n + 5n]$$

□

Doubt 4: Is this good enough?

• The upper bound is still fine. But the lower bound suggests that there is a high probability that we might end up with negative values of n.
• Can we do better?

GOAL: $\hat{n} \in [n - \epsilon n, n + \epsilon n]$; for small $\epsilon > 0$

• Basically, $\hat{n} \in (1 \pm \epsilon)n$
• ϵ will be a small quantity like 0.1, suggesting a 10% error margin
• For this we will see **Morris+ Algorithm**

3 Morris+ Algorithm

- Take k counters, compute i.i.d.
- Press button for each counter completely independently {x_1,x_2,...,x_k}
- Each of them is Morris Algorithm with counter x^{i}, {i = 1,...,k}

Estimator $\hat{n}^{\mathrm{k}}$ is equal to the average of estimators $\{\mathrm{x}^1,...,\mathrm{x}^{\mathrm{k}}\}$

Claim 5. $E[\hat{n}^k] = n$

Proof. Note: $2^{\mathrm{x^i}} - 1 = n$

$$E[\hat{n}^k] = E[\frac{1}{k}\sum_{i=1}^{k}(2^{\mathrm{x^i}} - 1)] = E[\frac{1}{k}\sum_{i=1}^{k} n] = n$$

□

Claim 6. *Space is $O(kloglogn)$ with probability $\geq$ 90%. No counter is larger and occupying more than logn space*

Claim 7. $var[\hat{n}^k] = \frac{1}{k}var[\hat{n}]$ *(variance of one counter)*

Proof.

$$var[\hat{n}^{\mathrm{k}}] = var[\frac{1}{k}\sum_{i=1}^{k}(2^{\mathrm{x^i}} - 1)] = \frac{1}{k^2}var[\sum_{i=1}^{k}(2^{\mathrm{x^i}} - 1)] = \frac{1}{k^2}\sum_{i=1}^{k}var[(2^{\mathrm{x^i}} - 1)] = \frac{1}{k}var[\hat{n}]$$

□

Goal: Want $\hat{n} = (1 \pm \varepsilon)n$
Chebyshev: $P[(\hat{n}^k - n) > \varepsilon n] \leq \frac{var[\hat{n}^{\mathrm{k}}]}{\varepsilon^2 n^2}$,$\varepsilon^2 n^2 \leq 0.1$

$$var[\hat{n}^{\mathrm{k}}] \leq 0.1\varepsilon^2 n^2$$

$$\frac{var[\hat{n}]}{k} \leq 0.1\varepsilon^2 n^2$$

$$k \geq \frac{var[\hat{n}]}{0.1\varepsilon^2 n^2}$$

It is enough to require $k = \frac{2n^2}{0.1\varepsilon^2 n^2} = \frac{20}{\varepsilon^2} = \Theta(\frac{1}{\varepsilon^2})$ So, it is enough to repeat Morris Algorithm $\frac{1}{\varepsilon^2}$ times *Theorem*: Morris + Algorithm can achieve $(1 \pm \varepsilon)$ approximation with 90% probability using space $O(\frac{loglogn}{\varepsilon^2})$

4 Hashing

Problem: Dictionary, data structure problem Given a large, fixed universe U of items Given set $S\epsilon U$, $|S| = n$, preprocess S into data structure such that it can answer: "Is $x\epsilon S$?"

Solution:
1. Given S, iterate over $S \rightarrow O(n)$ runtime
2. Binary search $\rightarrow O(logn) \rightarrow$ querytime *log n*, space: *O(n)*
3. Full-index: Store a table T of size |U| with {T_i} = 1 iff $i\epsilon S \rightarrow$ query time = *O(1)* and space = O(|U|)

Can we combine for *O(1)* query time and *O(n)* space? This is what hashing tries to do. If we use

IP addresses, each with 32 bits, as an example: $|\mathrm{U}| = 2^{32}$ = all possible IPs, all binary strings of length 32 Hashing says U is large but the set in our data structure is small, so we reduce U

U (h) $\rightarrow$ 1,...,m

Property $\star$: *h: U* $\rightarrow$ *[m] satisfies:* $\forall$ *i* ϵ *S and given x we have: h(i) = h(x) iff i = x.*

Solution to dictionary problem:

1. Compute *h(i), i* ϵ *S* to reduce universe from U to m
2. Store table {T_j} = 1 iff *j = h(i), i* ϵ *S*
3. At query x, check if T_h(x) = 1

Solution performance: space = *O(m)* and query time = *O(1)* to query table + time to compute *h(x)*

Solution 4: pick h randomly and hope that our property $\star$ holds with good probability
Define: Collision *h(i) = h(x)* $\star$ says that they should only collide if $i = x$, then solution 4 should be correct