CS 7880: Algorithms for Big Data (Fall'22) **Northeastern University**

Lecture 10 & 11

October 18, 2022

Instructor: Soheil Behnezhad *Scribe: Haoyu He*

Disclaimer: *These notes have not been edited by the instructor.*

1 Introduction to Streaming Algorithms

In this lecture, we first exposed to streaming algorithms which is used to process data streams where input data is coming one by one. Primarily, we focus on space complexity [1] rather than time complexity which we talked a lot in previous lessons. We can take an example of this.

Problem 1. There are n elements from a universe $[m] = \{1, 2, 3, ..., m\}$ arriving one by one. Our goal is to return one element uniformly at random. How much space do we need?

We have two easy and straightforward solutions to this. The first one is to create a list of n, store the n elements and then pick one from it uniformly. Space we need for this solution is $O(nlgm)$ since we have n entries and each we need $O(lgm)$ space

The second solution is to keep an array of length m, for each $i \in [m]$ count the number of time we see i and store it to the corresponding position. We then pick one from the array with probability respect to the number of times we see it and return it in m. Space we need for this solution is $O(mlgn)$ since we have m entries and each we need $O(lgn)$ space

The above two solutions require a lot of space and can we do better? Of course! Pick $i \in [n]$ uniformly at random. When we see the i-th element we return it. The space we need is $O(lgn + lgm)$ since we need to store i and the element we return.

Consider the same problem but suppose n is not known, can we still achieve with same space complexity?

Algorithm 1:
set s as an empty set
when the i-th element arrives, it has a probability of $\frac{1}{i}$ to replace the item in s
After all the elements arrived, return s

Claim 1. *Suppose $e_i \neq e_j$ for any $i \neq j$. For any i, probability of returning e_i is $\frac{1}{n}$.*

Proof.

$$\begin{aligned} Pr[\text{returning } e_i] &= \frac{1}{q} \times \Pi_{j=i+1}^{n}(1 - \frac{1}{j}) \\ &= \frac{1}{i} \times \frac{i}{i+1} \times \frac{i+1}{i+2} \times ... \times \frac{n-1}{n} \\ &= \frac{1}{n} \end{aligned}$$

Thus the space we need is $O(lgn + lgm)$ (one for counter and one for stored number) even if we don't know n beforehand.

2 Streaming Algorithms for Distinct Elements Counting

One interesting application of streaming algorithms is to count distinct elements arriving one after another and there has some reliable solutions to it[2]. Let's start with a simple example.

Problem 2. There are n elements from universe $[m] = \{1, 2, 3, ..., m\}$ arriving one by one. The goal is to return the number of distinct elements, DE, we have seen.

For example, let $m = 5, n = 7$ and the sequence we seen is $1, 2, 5, 4, 2, 1, 4$. The answer to this one should be four since we saw four distinct elements $1, 2, 4, 5$. There are also two trivial solutions, $O(m)$ and $O(nlgm)$. The first one is we create a boolean array of size m and set each entry as false. For each element we see, go to the corresponding entry and change it to true. Finally we count the number of entries being true. The space we need is this array which is $O(m)$. The second one is we create an empty array and for each element we see we append it to this array. After this, we count the number of distinct element in this array. Since the size of each entry is $O(lgm)$ and we have n of these, the space we need is $O(nlgm)$

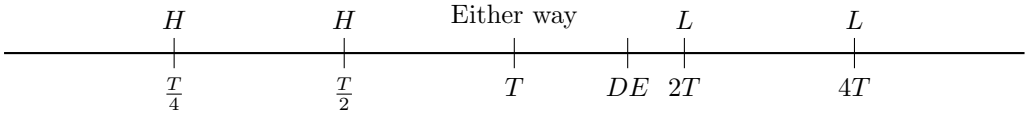
Claim 2. *There are two known lowerbounds regarding deterministic and randomized algorithms. The first one is every deterministic algorithm requires $\Omega(m)$ bits, even if it returns a 1.1-approximation. The second one is every randomized algorithm returning an exact solution requires $\Omega(n)$ bits of space.*

Theorem 3. *For any $\epsilon > 0, \delta > 0$, there is a randomized algorithm return $(1+\epsilon)$-approximation solution to distinct elements using the polynomial of $lgn, lgm\frac{1}{\epsilon}, lg(\frac{1}{\delta})$ space and succeed up to $1-\delta$.*

$$Pr[\widetilde{DE} < (1-\epsilon)DE] \geq 1 - \delta$$

2.1 Threshold testing approach to find DE

We propose a new algorithm to find DE. Given a threshold $0 \leq T < n$, the goal is to return L if this distinct element $DE \leq T$ and H if $DE > 2T$



We can see from the above figure that to the left of $\frac{T}{2}$ the algorithm will return H and to the right of the $2T$ will return L and can return anything on T

Algorithm $\mathscr{A}$:
For any $i = 0, 1, ..., lgn$
let $T_i = 2^i$, $\mathscr{A}$ be threshold tester.
For any T_i, run $\mathscr{A}$ with parameter $\delta' = \frac{\delta}{2lgn}$
let i be the smallest index such that $\mathscr{A}$ returns L for T_i

Claim 4. *let $DE \in [T_i, T_{i+1}]$ then with probability greater or equal to $1-\delta$, algorithm $\mathscr{A}$ returns T_i or T_{i+1}*

Proof. let ϵ_1 be the event that algorithm $\mathscr{A}$ return $T \in \{T_0, ..., T_{i-1}\}$ and ϵ_2 be the event that algorithm $\mathscr{A}$

return $T \in \{T_{i+2}, ...\}$. Thus we have

$$\begin{aligned} Pr[\epsilon_1] &\leq Pr[\exists j < i \text{ s.t. } \alpha \text{ returns } L \text{ for } T_j] \\ &\leq \Sigma_{j=0}^{i-1} Pr[\alpha \text{ returns } L \text{ for } T_j] \\ &\leq lgn \times \delta' \\ &= lgn \times \frac{\delta}{2lgn} \\ &= \frac{\delta}{2} \end{aligned}$$

$$\begin{aligned} Pr[\epsilon_2] &\leq Pr[\alpha \text{ returns } H \text{ for all } T_j, j \leq i+1] \\ &\leq Pr[\alpha \text{ returns } H \text{ for } T_{i+1}] \\ &\leq \delta' \\ &= \frac{\delta}{2lgn} \end{aligned}$$

Thus

$$\begin{aligned} Pr[\text{correct output}] &= 1 - Pr[\epsilon_1 \vee \epsilon_2] \\ &= 1 - Pr[\epsilon_1] - Pr[\epsilon_2] \\ &\leq 1 - \frac{\delta}{2} - \frac{\delta}{2lgn} \\ &\leq 1 - \delta \end{aligned}$$

The space complexity of this algorithm is $O(lgm)$ times the space complexity of $\mathscr{A}$ for each run.

Algorithm: a prelinminary version of algorithm $\mathscr{A}$
Pick a hash function: $h : [m] \to [n]$
For any $i \in [m]$, $h(i)$ is uniformly distributed on $[T]$ noted that the space complexity for this is $\Omega(mlgT)$.
For each element e_i arrives, check if e_i is 1
if so return H
otherwise return L

This is an visualization of this algorithm. In the figure above, each element in $[m]$ maps to the first element or any element in T is $\frac{1}{T}$

Lemma 5. *The followings are true for the algorithm α.*
If $DE \leq T$, then α returns L with a probability higher than or equal to $\frac{1}{e^{1.01}} \approx 0.36$
If $DE \geq 2T$, then α returns L with a probability less than or equal to $\frac{1}{e^2} \approx 0.13$

Proof. Let $i_1, ..., i_{DE}$ be distinct elements. Thus

$$Pr[\text{return } = Pr[h_{ij} \neq 1 \text{ for all } j \in [DE]]$$

So if $DE \geq 2T$, then $Pr[\text{return } L] = (1 - \frac{1}{T})^{DE} \leq (1 - \frac{1}{T})^{2DE} \leq \frac{1}{e^2}$. if $DE \leq T$, then $Pr[\text{return } L] = (1 - \frac{1}{T})^{DE} \geq (1 - \frac{1}{T})^T = \frac{1}{e^{1.01}}$

Exercise. Proof that $(1-\frac{1}{T})^T$ is increasing for $T \geq 2$.

If the above one is correct,

$$Pr[\text{return } L] = (1-\frac{1}{T})^{DE} \geq (1-\frac{1}{T})^T \geq (1-\frac{1}{100})^{100} = 0.366$$

Lemma 6. *There is an algorithm that give $T \geq 100$*

- *If $DE \leq T$, then the algorithm returns L with probability greater than $\frac{1}{e^{1.01}} \approx 0.36$*
- *If $DE \geq 2T$, then the algorithm returns L with probability greater than $\frac{1}{e^2} \approx 0.13$*

The gap between 0.13 and 0.36 with be equally divided and assign correspond probability to closer label to ensure we get some reasonable results on this.

Algorithm $\mathscr{A}$(refined version based on previous one)
Let $\lambda := \frac{1}{2}(\frac{1}{\epsilon^2} + \frac{1}{\epsilon^{1.01}})$
Run the algorithm of lemma 6 for $k \times \frac{lg(\frac{2}{\delta})^3}{(0.36-\delta)^2} = O(lg\frac{1}{\delta})$ times all in parallel
Let $N_i := 1$ if and only if the i-th run returns L
Let $N := \Sigma_{i=1}^k N_i$
If $N \geq k$ return L otherwise return H

Lemma 7. *Algorithm $\mathscr{A}$ returns the right answer with probability at least $1-\delta$.*

Proof. Suppose that $DE \leq T$. In this case we are supposed to return L.

$$E[N_1] \geq 0.36 \Rightarrow E[N] = k \times E[N_1] \geq 0.36k$$

$$\begin{aligned}
Pr[\text{returning } L] &= Pr[N > k] \\
&= 1 - Pr[N \leq \lambda k] \\
&= 1 - Pr[|N - E[N]| \geq (0.36 - k)] \\
&\geq 1 - 2e^{-\frac{(0.36-\lambda)^2 k^2}{3E[N]}} \\
&\geq 1 - 2e^{-\frac{(0.36-\lambda)^2 k^2}{3k}} \\
&= 1 - 2e^{-\frac{(0.36-\lambda)^2 k}{3}} \\
&= 1 - 2e^{-lg\frac{2}{\delta}} \\
&\geq 1 - \delta
\end{aligned}$$

The other case where $DE \geq 2T$ can be proved symmetrically.

Definition 8. A family $\mathscr{H} = \{h : [a] \to [b]\}$ is called a k-wise independent family of hash functions if for all distinct $x_1, x_2, ..., x_k \in [a]$ and any not necessary distinct $y_1, y_2, ..., y_k \in [b]$,

$$Pr_{h \sim \mathscr{H}}[h(x_1) = y_1, h(x_2) = y_2, ..., h(x_k) = y_k] = \frac{1}{b^k}$$

Lemma 9. *let $\mathscr{H} = \{h : [a] \to [b]\}$ be a family of k-wise independent hash function. Let $x_1, x_2, ..., x_k$ be distinct elements in $[a]$. Then*

- *for every $i \in [k]$, $h(\mathscr{H}_i)$ is uniform over $[b]$*

- $h(x_1), h(x_2), ..., h(x_k)$ *are mutally independent*

Proof. Take some arbitrary $y_1 \in [b]$.

$$\begin{aligned} Pr[h(x_1) = y_1] &= \Sigma_{y_1, y_2, ..., y_k \in [b]} Pr[h(x_1) = y_1, h(x_2) = y_2, ..., h(x_k) = y_k] \\ &= \Sigma_{y_2, y_3, ..., y_k \in [b]} \frac{1}{b^k} \\ &= \frac{b^{k-1}}{b^k} \\ &= \frac{1}{b} \end{aligned}$$

$$Pr[h(x_1) = y_1, h(x_2) = y_2, ..., h(x_k) = y_k] = \frac{1}{b^k} = \Pi_{i=1}^{k} Pr[h(x_i) = y_i]$$

Let $k \geq 2$ and let $p > k$ be a prime number. Let $\mathscr{H}$ be the set of degree $k-1$ polynomials over $\mathbb{F}_p$ (integers mod p).

$$\mathscr{H} = \{h : [p] \to [p], h(x) = c_{k-1}x^{k-1} + c_{k-2}x^{k-2} + .. + c_0 \bmod p\}$$

where $c_i \in \{0, 1, ..., p-1\}$

To sample h from $\mathscr{H}$ uniformly, it suffices to sample k independent coefficients $c_0, c_1, ..., c_{k-1}$ from $\{0, 1, ..., p-1\}$ uniformly.

Proposition 10. *For any a, b, there exists a k-wise independent family if hash functions mapping $[a] \to [b]$, that requires $O(k \times lg(a+b))$ bits of space to store.*

The threshold tester we discussed also works with 2-wise independence, so it only requires $O(lgm \times lg\frac{1}{\delta})$ space for each threshold tester. Thus the overall space is $O(lgm \times lgm \times lg\frac{lgm}{\delta}) = O(lg^2 m(lglgm + lg\frac{1}{\delta}))$

Take a 2-wise independent hash function $h : [m] \to [m]$ where m takes $O(lgm)$ bits. Let $t = \frac{100}{\epsilon^2}$, X be the t-th smallest $y \in [m]$ such that $h(x) = y$ for some x in the input. Finally, return $DE = \frac{mt}{X}$

Lemma 11. $Pr[|DE - \widetilde{DE}| \geq \epsilon DE] \leq \frac{1}{50}$

Proof. The proof can be divided into two parts and we focus on the lower tail first.

$$\begin{aligned} Pr[\widetilde{DE} < (1-\epsilon)DE] &= Pr[\frac{mt}{x} < (1-\epsilon)DE] \\ &= Pr[X > \frac{mt}{(1-\epsilon)DE}] \\ &= Pr[\text{less than } t \text{ distinct element needed to a value} \leq \frac{mt}{(1-\epsilon)DE} (\text{We refer this to } \tau)] \end{aligned}$$

For each $i \in [DE]$ define Y_i be the indicator of the event that the i-th distinct element is mapped to a value less than τ

Definition 12. $Y = \Sigma_{i=1}^{DE} Y_i$

$$E[Y_i] = E[Y_1] = Pr[h(x_1) < \tau] = \frac{\tau}{m} = \frac{t}{(1-\epsilon)DE}$$

By theorem of expectation of linearity, we have

$$E[Y] = DE \times E[Y_1] = \frac{t}{1-\epsilon}$$

$$Pr[x > \frac{mt}{1-\epsilon}DE] = Pr[Y < t]$$

So it suffices to prove Y is not going to be made smaller than its expected value

References

[1] Noga Alon, Yossi Matias, and Mario Szegedy. The space complexity of approximating the frequency moments. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pages 20–29, 1996. 1

[2] Ziv Bar-Yossef, TS Jayram, Ravi Kumar, D Sivakumar, and Luca Trevisan. Counting distinct elements in a data stream. In *International Workshop on Randomization and Approximation Techniques in Computer Science*, pages 1–10. Springer, 2002. 2