CS-448 Sublinear Algorithms for Big Data Analysis March 5, 2025

Lecture 3

Lecturer: Michael Kapralov

1 Distinct Elements Problem

In the *distinct elements problem*, we are given a stream of elements between 1 and n, and we wish to compute the number of distinct elements occurring in the stream. More formally, let x be a n-dimensional vector, where x_i denotes the frequency of i in the stream, we want to compute $||x||_0$ (the number of distinct elements). To obtain the exact solution, the best we can do is store all the distinct elements occurring in the stream. However, this can be the entire set i.e $\Omega(n)$, and our goal is to obtain an $o(n)$-space algorithm. To obtain a sublinear space algorithm we relax our requirements, and we ask for an algorithm that provides an approximation $\tilde{k}$ of $||x||_0$ such that $||x||_0 \leq \tilde{k} \leq (1+\epsilon)k$ with probability at least $1-\delta$. Before solving this problem, we reduce it to a simpler decision problem.

1.1 Distinct Elements Decision Problem

In this version of the problem, we are given a threshold t and we want to distinguish between the following two cases with "high" probability.

- **YES** : $||x||_0 \geq 2t$
- **NO** : $||x||_0 < t$

```
(1) Select a set $S \subseteq [n]$ by including every $i \in n$ independently with probability $\frac{1}{t}$
(2) Maintain $C = \sum_{i \in S} x_i$
(3) Output YES if $C > 0$, and NO otherwise.
```

Let $||x||_0 = k$, then the probability of outputting **YES** is as follows

$$\begin{aligned}\Pr[C > 0] &= \Pr[\text{S contains an element in the stream}] \\ &= 1 - \Pr[\text{S doesn't contain an element in the stream}] \\ &= 1 - \left(1 - \frac{1}{t}\right)^k\end{aligned}$$

which behaves differently under the different regimes.

- **NO** case: $\Pr[C > 0] = 1 - (1-\frac{1}{t})^k \leq 1 - (1-\frac{1}{t})^t \approx 1 - e^{-1} \approx 0.64$
- **YES** case: $\Pr[C > 0] = 1 - (1-\frac{1}{t})^k \geq 1 - (1-\frac{1}{t})^{2t} \approx 1 - e^{-2} \approx 0.86$

Note that these bounds hold for t large enough since we used the approximation $(1-\frac{1}{x})^x \approx e^{-1}$. From the above analysis, we know that we can differentiate between the **YES** and the **NO** case. However, there is a small gap between the two cases, and this would result in a large error probability.

To amplify the gap between them and obtain an arbitrarily small probability of error, we do the following. Let

$$Y_i = \begin{cases} 1 & \text{if the experiment output } \textbf{YES} \\ 0 & \text{o.w.} \end{cases}$$

(1) Repeat the experiment m times independently.

(2) Output **YES** if $\sum_{i=1}^{m} Y_i \geq 0.7m$, and **NO** otherwise.

One can easily show that we can distinguish between the two cases with probability at least $1-\delta$ by setting $m = C \log \frac{1}{\delta}$ for some large enough $C > 0$. Similar to the Morris++ analysis, we select m by showing that the error probability decay exponentially in m. Note that there is nothing magical in the choice of 0.7, any constant in the range $(0.64, 0.86)$ would work. This gives an algorithm for distinguishing between the two cases with space complexity $O(\log n \log(1/\delta))$ that succeeds with probability at least $1-\delta$, but our bound on the space complexity does not take into account the storage needed for S. If S is truly a random subset as defined above, however, it is not possible to store it compactly. We next design a version of the algorithm that uses pseudorandom S that is easy to store, but works almost as well for our purpose.

Note that the algorithm we designed is a 'linear sketch' the information that is stored about x is a linear function of x. This in particular means that our algorithm works even in *dynamic streams*, where elements can both arrive and depart.

2 Pairwise independent hash families

We now introduce hash families with the limited independence.

Definition 1 *A family of hash functions $H = \{h : [n] \to U\}$ is a pair-wise independent if for every pair of distinct elements $x, y \in [n]$ and every $a, b \in U$, we have:*

$$Pr_{h \in H}[h(x) = a \wedge h(y) = b] = \frac{1}{|U|^2}.$$

An example of pair-wise hash family is $\{ax + b \mod p\}$ where $a, b, x \in \mathbb{Z}_p$ and a, b are selected independently uniformly at random and p is a prime number.

Claim 2 *Let $a, b \in \mathbb{Z}_p$ be selected independently uniformly at random, and let $h_{a,b} : \mathbb{Z}_p \to \mathbb{Z}_p$ be defined by $h(x) = ax + b \mod p$. Then the family $\mathcal{H} := \{h_{a,b}\}_{a \in \mathbb{Z}_p, b \in \mathbb{Z}_p}$ is pairwise independent when p is prime.*

The proof is left as an exercise. Note that if p is not prime, the hash family above is not pairwise independent. Suppose that p is a power of two and $x - y = p/2$. Then we have that $h_{a,b}(x) - h_{a,b}(y) = a(x-y) = ap/2$ is 0 with probability $1/2$ and $p/2$ with probability $1/2$. Thus, the distribution of the pair $(h_{a,b}(x), h_{a,b}(y))$ is not uniform over $\mathbb{Z}_p^2$, and the hash family is not pairwise independent.

We now get the following small space method for generating the set S:

(1) Choose a parameter $B = \Theta(t)$.

(2) Select a hash function h from a pair-wise independent family H.

(3) Let $S = \{i \in [n], h(i) = 1\}$

3 Analysis with pairwise independent hash functions

- **NO** case: $k \leq t$

$$\begin{aligned}\Pr\left[\sum_{i\in S} x_i > 0\right] &\leq \Pr\left[\text{supp}(x)\cap S \neq \emptyset\right]\\ &\leq \sum_{i\in\text{supp}(x)} \Pr\left[i\in S\right]\\ &\leq \frac{k}{B}\\ &\leq \frac{t}{B}\end{aligned}$$

where the second inequality follows from *union bound*.

- **YES** case: $k \geq 2t$

$$\begin{aligned}\Pr\left[\sum_{i\in S} x_i > 0\right] &\geq \sum \Pr\left[i\in S\right] - \sum_{i,j\in\text{supp}(x), i\neq j} \Pr\left[i\in S \wedge j\in S\right]\\ &= \frac{k}{B} - \sum_{i,j\in\text{supp}(x), i\neq j} \frac{1}{B^2}\\ &= \frac{k}{B} - \frac{k(k-1)}{B^2},\end{aligned}$$

where the first inequality follows from *inclusion-exclusion principle* and the second inequality follows from the definition of *pair-wise independent hash family.* Now select an arbitrary subset Z of the support of X of size $2t$, and note that $\sum_{i\in S} x_i \geq \sum_{i\in S\cap Z} x_i$, and it thus suffices to lower bound the probability that $\sum_{i\in S\cap Z} x_i > 0$. We now apply the analysis above with $k = 2t$, getting

$$\begin{aligned}\Pr\left[\sum_{i\in S} x_i > 0\right] &\geq \Pr\left[\sum_{i\in S\cap Z} x_i > 0\right]\\ &\geq \frac{k}{B} - \frac{k(k-1)}{B^2}\\ &\geq \frac{2t}{B} - \frac{4t^2}{B^2}.\end{aligned}$$

By taking $B = 4t$, we can obtain the following inequality.

$$\begin{aligned}\Pr[\textbf{Yes}] - \Pr[\textbf{NO}] &\geq \frac{t}{B} - \frac{4t^2}{B^2}\\ &\geq \frac{t}{B}\left(1 - \frac{1}{4}\right)\\ &\geq \frac{3}{4}\frac{t}{B}\end{aligned}$$

Once we have ALG_t that distinguish $k < t$ and $k \geq 2t$, to solve the *distinct elements problem*, we can run ALG_t for $t = 1, 2, 4, \cdots, n$. Thus the overall space complexity of the algorithm is $O(\log n \log \frac{1}{\delta})$ with the failure probability at most $\delta \log n$.