Sublinear Time and Space Algorithms 2018B – Lecture 4
Amplifying Success and Hash Functions[*]

Robert Krauthgamer

1 Amplifying Success Probability

To amplify the success probability of Algorithm CountMin (in general case), we use median of independent repetitions (instead of minimum), and analyze it using the following (standard) concentration bounds.

Chernoff-Hoeffding concentration bounds: Let $X = \sum_{i\in[n]} X_i$ where $X_i \in [0,1]$ for $i \in [n]$ are independently distributed random variables. Then

$$\begin{aligned} \forall t > 0, &\qquad \Pr[|X - \mathbb{E}[X]| \geq t] \leq 2e^{-2t^2/n}. \\ \forall 0 < \varepsilon \leq 1, &\qquad \Pr[X \leq (1-\varepsilon)\,\mathbb{E}[X]] \leq e^{-\varepsilon^2\,\mathbb{E}[X]/2}. \\ \forall 0 < \varepsilon \leq 1, &\qquad \Pr[X \geq (1+\varepsilon)\,\mathbb{E}[X]] \leq e^{-\varepsilon^2\,\mathbb{E}[X]/3}. \\ \forall t \geq 2e\,\mathbb{E}[X], &\qquad \Pr[X \geq t] \leq 2^{-t}. \end{aligned}$$

Algorithm CountMin++:

1. Run $k = O(\log n)$ independent copies of algorithm CountMin, keeping in memory the vectors $S^1, \ldots, S^k$ (and functions $h^1, \ldots, h^k$)

2. Output: To estimate x_i report the median of all basic estimates $\hat{x}_i = \text{median}\{S^l_{h^l(i)} : l \in [k]\}$

Exer: Prove that

$$\Pr[\hat{x}_i \in x_i \pm \alpha\|x\|_1] \leq 1/n^2.$$

Hint: Define an indicator Y_j for the event that copy $j \in [k]$ succeeds, then use one of the concentration bounds.

Exer: Use these concentration bounds to amplify the success probability of the algorithms we saw for Distinct Elements and for Probabilistic Counting (say from constant to $1 - 1/n^2$).

[*]These notes summarize the material covered in class, usually skipping proofs, details, examples and so forth, and possibly adding some remarks, or pointers. The exercises are for self-practice and need not be handed in. In the interest of brevity, most references and credits were omitted.

Hint: use independent repetitions + median.

Exer: Let $x \in \mathbb{R}^n$ be the frequency vector of a stream of m items (insertions only). Show how to use the CountMin++ algorithm (for ℓ_1 point queries) to estimate the median of x, which means to report an index $j \in [n]$ that with high probability satisfies $\sum_{i=1}^{j} x_j \in (\frac{1}{2} \pm \varepsilon)m$.

Hint: Use a dyadic decomposition to express any interval $[1..j]$ as the sum of at most $\log_2 n$ intervals from $O(n \log n)$ canonical intervals, then estimate each of these intervals.

2 Hash Functions

Independent random variables: Recall that two (discrete) random variables X, Y are independent if

$$\forall x, y \qquad \Pr[X = x, Y = y] = \Pr[X = x] \cdot \Pr[Y = y].$$

This is equivalent to saying that the conditioned random variable $X|Y$ has exactly the same distribution as X. In particular, it implies $\mathbb{E}[XY] = \mathbb{E}[X] \cdot \mathbb{E}[Y]$.

The above naturally extends to more than two variables, and then we say the random variables are mutually (or fully) independent.

Pairwise independence: A collection of random variables $X_1, \ldots, X_n$ is called *pairwise independent* if for all $i \neq j \in [n]$, the variables X_i and X_j are independent.

Example: Let $X, Y \in \{0, 1\}$ be random and independent bits, and let $Z = X \oplus Y$. Then X, Y, Z are clearly not mutually (fully) independent, but they are pairwise independent.

Observation: When $X_1, \ldots, X_n$ are pairwise independent, the variance $\mathrm{Var}(\sum_i X_i)$ is exactly the same as if they were fully independent, because

$$\mathrm{Var}(\sum_i X_i) = \mathbb{E}[(\sum_i X_i)^2] - (\mathbb{E}[\sum_i X_i])^2 = \sum_{i,j} \mathbb{E}[X_i X_j] - (\sum_i \mathbb{E}[X_i])^2.$$

A different way to see it, is via the following well-known (and easy) fact: If $X_1, \ldots, X_n$ are pairwise independent (and have finite variance), then $\mathrm{Var}(\sum_i X_i) = \sum_i \mathrm{Var}(X_i)$.

The above definition extends to k-wise independence, where every subset of k random variables should be independent.

Pairwise independent hash family: A family H of hash functions $h : [n] \to [M]$ is called *pairwise independent* if for all $i \neq j \in [n]$,

$$\forall x, y \qquad \Pr_{h \in H}[h(i) = x, h(j) = y] = \Pr[h(i) = x] \Pr[h(j) = y].$$

A common scenario is that each $h(i)$ is uniformly distributed over $[M]$.

Universal hashing: A family H of hash functions $h : [n] \to [M]$ is called *2-universal* if for all $i \neq j \in [n]$,

$$\forall x, y \qquad \Pr_{h \in H}[h(i) = x, h(j) = y] \leq 1/M.$$

Observe that 2-universality is a weaker requirement than (follows from) pairwise independence when each $h(i)$ is distributed uniformly over $[M]$, but it suffices for many algorithms.

Construction of pairwise independent hashing:

Assume $M \geq n$ and that M is a prime number (if not, we can pick a larger M that is a prime). Pick random $p, q \in \{0, 1, 2, \dots, M-1\} = [M]$ and set accordingly $h_{p,q}(i) = pi + q \pmod{M}$.

The family $H = \{h_{p,q} : p, q\}$ is pairwise independent because for all $i \neq j$ and all x, y,

$$\Pr_{h \in H}[h(i) \equiv x, h(j) \equiv y] = \Pr_{p,q}\left[\left(\begin{smallmatrix} i & 1 \\ j & 1 \end{smallmatrix}\right)\left(\begin{smallmatrix} p \\ q \end{smallmatrix}\right) \equiv \left(\begin{smallmatrix} x \\ y \end{smallmatrix}\right)\right] = \Pr_{p,q}\left[\left(\begin{smallmatrix} p \\ q \end{smallmatrix}\right) \equiv \left(\begin{smallmatrix} i & 1 \\ j & 1 \end{smallmatrix}\right)^{-1}\left(\begin{smallmatrix} x \\ y \end{smallmatrix}\right)\right] = \tfrac{1}{M^2},$$

where we relied on the above matrix being invertible.

Storing a function $h_{p,q}$ from this family can be done by storing p, q, which requires $\log|H| = O(\log M)$ bits. In general, $\log|H|$ bits suffice to store an index of $h \in H$.

Another construction for $M = 2$:

Let A be a 0-1 matrix of size $(2^t - 1) \times t$ with all possible (distinct) nonzero rows $A_i \in \{0,1\}^t$. For a random $p \in \{0,1\}^t$, define $h_p : [2^t] \to \{0,1\}$ by $h_p(i) := (Ap)_i = \langle A_i, p\rangle$, where all operations are performed in $GF[2]$ (i.e., modulo 2).

Storing the hash function requires $\log|H| = O(t)$ bits.

Exer: Prove that the family $H = \{h_p : p\}$ is pairwise independent.

Exer: Show that this construction generates k-wise independent bits whenever the matrix A satisfies that every k rows are linearly independent.

Exer: Show that the correctness of algorithm CountMin (for ℓ_1 point query) extends to using a universal hash function, and analyze how much additional storage the hash function requires.

Exer: Show that the correctness of algorithm CountSketch (for ℓ_2 point query) can be implemented with a limited (pairwise) independence and analyze how much additional storage the hash function requires.

Hint: use separate randomness for the hash functions and for the signs.

Exer: Show that algorithm AMS (for estimating ℓ_2 norm) works even if the random signs $\{r_i\}$ are only 4-wise independent.

Exer: Show that the correctness of algorithm Bottom k (for Distinct Elements) can be extended to using a pairwise independent hash function $h : [n] \to [n^3]$ (instead of continuous range $[0,1]$), and analyze how much additional storage the hash function requires.

Hint: Our analysis used events of the form $\{h(i) < \textit{threshold}\}$, and relied on independence for every pair $h(i), h(j)$.