Sublinear Time and Space Algorithms 2020B – Lecture 5
Hash functions, Heavy Hitters and Compressed Sensing*

Robert Krauthgamer

1 Hash Functions (cont'd)

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 $p, q \in \{0, 1, 2, \ldots, M-1\} = [M]$ 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,

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

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 $h \in H$.

One can reduce the size of the range $[M]$ (from large $M \geq n$ to $M = 2$ or say $4/\alpha$), with a small overhead/loss.

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.

*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.

Exer: Show that the correctness of algorithm CountSketch (for ℓ_2 point query) can be implemented with 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.

We will now see some applications of point queries.

2 Application 1: Heavy Hitters (Frequent Items)

Problem Definition: For parameter $\phi \in (0,1)$ and $p \in [1,\infty)$, define

$$HH^p_\phi(x) = \{i \in [n] : \ |x_i| \geq \phi \|x\|_p\}.$$

Observe that its cardinality is bounded by $\left|HH^p_\phi(x)\right| \leq 1/\phi^p$.

We will focus on $p = 1$ and ϕ is "not too small".

Approximate Heavy Hitters:

Parameters: $\phi, \varepsilon \in (0,1)$.

Goal: return a set $S \subseteq [n]$ such that

$$HH^p_\phi \subseteq S \subseteq HH^p_{\phi(1-\varepsilon)}.$$

Reduction from HH to point query (for $p = 1$):

Assume we have an algorithm for ℓ_1 point queries with parameter $\alpha = \varepsilon\phi/2$, and amplify its success probability to $1 - \frac{1}{3n}$ if needed.

1. compute an estimate $\tilde{x}_i$ for every $i \in [n]$ using this algorithm (this step takes time $O(n \log n)$ or even more)

2. report the set $S = \{i : \ \tilde{x}_i \geq (\phi - \varepsilon\phi/2)\|x\|_1\}$ (it is easy to know $\|x\|_1$ when $x \geq 0$, but more difficult in general)

Storage requirement: We can employ algorithm CountMin+ for ℓ_1 point queries, which requires $O(\alpha^{-1} \log n)$ words, and has error probability $1/n^2$, which is small enough. Then our approximate HH algorithm will take $O(\phi^{-1}\varepsilon^{-1} \log^2 n)$ bits.

Correctness: With probability $\geq 2/3$, all the n estimates are correct within additive $\varepsilon/2$. In this case, S contains all the ϕ-HH, and is contained in the $(\phi(1-\varepsilon))$-HH.

Exer: Extend the above approach to $p = 2$ (using CountSketch). How much storage it requires? Use the AMS sketch to estimate the ℓ_2-norm.

3 Application 2: Compressed Sensing (or Sparse Recovery)

Problem Definition: The input is a "signal" $x \in \mathbb{R}^n$, but instead of reading it directly we have only via linear measurements, i.e., we can observe/access $y_i = \langle A_i, x \rangle$ for $A_1, \ldots, A_m \in \mathbb{R}^n$ of our choice. Informally, the goal is to design few A_i's and then to use them recover x. We shall focus on non-adaptive A_i, i.e., the entire sequence has to be determined in advance.

Let $A_{m \times n}$ be a matrix whose rows are the A_i's, then we know that $Ax = y$. A trivial solution is to choose A that is invertible, which requires $m = n$. In general, this is optimal, because for smaller m there might be infinitely many solutions x to $Ax = y$.

Initial goal: Suppose that x is k-sparse (has at most k nonzeros, i.e., $\|x\|_0 = k$). What $m = m(n,k)$ is needed to recover x?

True goal: Suppose x is approximately k-sparse. For what m can we recover an approximation to x?

Remark: In most applications, it's preferable that A has bounded precision (i.e., the entries of A are integers of bounded magnitude), as otherwise y must be "acquired" with very high precision. Sometimes it's even important that A's entries are nonnegative.

CountMin Approach: Recall that CountMin is a (randomized) linear sketch of $x \in \mathbb{R}^n$, hence it can be viewed as multiplying x by some matrix A with $p = O(\alpha^{-1} \log n)$ rows.

Sparse 0-1 vector: Suppose first $x \in \{0,1\}^n$ and is k-sparse. Then $\|x\|_1 = k$, and a CountMin+ sketch of accuracy $\alpha = \frac{1}{3k}$ succeeds with probability at least $1 - 1/n$ in estimating all x_i's within additive $\pm\alpha\|x\|_1 \leq \pm\frac{1}{3}$, which can distinguish whether x_i is 0 or 1.

Sparse vector: If the nonzeros of x have different magnitudes, the above approach might require $\alpha \ll \frac{1}{k}$.

But a deeper inspection of CountMin shows that every coordinate has a good chance to "not collide" with any nonzero coordinate. This behavior is amplified by the repetitions + median trick's, and then WHP the estimator is exact, i.e., $\hat{x}_i = x_i$.

Exer: Show that a sketching matrix A with $m = O(k)$ rows (linear measurements) and whose entries are random Gaussians (or chosen uniformly from $[0,1]$) can recover with high probability every k-sparse input x. Show it also for an ε-coherent matrix for $\varepsilon = \frac{1}{10k}$.

Hint: It suffices that every $2k$ columns are linearly independent.