ECS 189A Sublinear Algorithms for Big Data **Fall 2024**

Lecture 16: k-wise Independent Hashing

Lecturer: Jasper Lee *Scribe: Nivita Reddy*

1 Review / Motivation

Recall the count distinct algorithm from last class. We showed that the space complexity of the algorithm was $O(\frac{1}{\epsilon^2}\log n + \text{space for random function } g)$ bits. To store a random function $g : [n] \to [n^3]$ we need $O(n \log n)$ bits. However, this is not necessary - it turns out that we can make do with less randomness, and less space. Consider the proof used in the count distinct algorithm. We needed:

- Linearity of expectation
- No covariance between pairs of variables, i.e.,

$$\mathbb{E}[g(i) \cdot g(j)] = \mathbb{E}[g(i)] \cdot \mathbb{E}[g(j)]$$

Since this is the only non-trivial property we need, we can make do with something more efficient and simpler than a uniformly random function g.

2 Pairwise Independent Hash Families

Definition 16.1. 2-wise Independent Hash Family

Consider a hash family, which is a distribution $h \leftarrow \mathcal{H}$ over a set of functions $[N] \to [M]$. We say $\mathcal{H}$ is a pairwise hash family if for all $i \neq j \in [N]$, $a, b \in [M]$,

$$\mathbb{P}_{h \leftarrow \mathcal{H}}(h(i) = a \wedge h(j) = b) = \frac{1}{M^2}$$

Another interpretation: $\mathcal{H}$ is a joint distribution over $h(1), \ldots, h(N)$. Our definition requires every (distinct) pair $h(i)$, $h(j)$ to have marginals equal to the uniform product distribution and $h(i)$, $h(j) \leftarrow \text{Unif}[M]$.

Note: k-wise independence is a property of a hash family, not an individual hash function. The randomness is over the hash families; if we are just talking about a specific hash function, fixing a particular h, then everything is a fixed quantity and there is no randomness, which means

$$\mathbb{P}(h(i) = a \wedge h(j) = b) \in \{0, 1\}$$

Why is k-wise independence useful and what does it buy us? It allows us efficient implementation while still maintaining independence required for probability calculations.

Proposition 16.2. *Consider any function $f : [M] \to \mathbb{R}$. Let $F = \sum f(h(i))$ where $h \leftarrow \mathcal{H}$ for a 2-wise independent hash family $\mathcal{H}$. Then*

$$\operatorname{Var} F = \sum_i \operatorname{Var}(f(h(i))$$

which implies

$$\mathbb{P}(|F - \mathbb{E}[F]| \geq a) \leq \frac{\sum_i \operatorname{Var}(f(h(i))}{a^2}$$

Proof.

$$\begin{aligned}\operatorname{Var} F &= \sum_i \operatorname{Var}(f(h(i)) + \sum_{i \neq j} \operatorname{Cov}(f(h(i)), f(h(j))) \\ &= \sum_i \operatorname{Var}(f(h(i))\end{aligned}$$

Since $h(i), h(j)$ independent implies $f(h(i))$, $f(h(j))$ independent. □

Theorem 16.3. *(Few Collisions) Consider 2-wise independent hash family $\mathcal{H} : [N] \to [M]$. Then for all $i \neq j \in [N]$*

$$\mathbb{P}_{h \leftarrow \mathcal{H}}(h(i) = h(j)) \leq \frac{1}{M}$$

Proof. There are M possibilities for what $h(i)$ and $h(j)$ could be, each occurring with probability $\frac{1}{M^2}$, so we get $\frac{M}{M^2} = \frac{1}{M}$. □

Note: Any hash family satisfying the above condition is sometimes referred to as a universal hash family, which is a slightly weaker notion than 2-wise independence (although the two names are sometimes used interchangeably).

Corollary 16.4. *Consider 2-wise independent $\mathcal{H} : [N] \to [M]$. Fix set $\mathcal{S} \subseteq [N]$ and element $i \in [N]$. Let $X = |\{j \in S : h(j) = h(i)\}|$ = number of indices in S colliding with i. Then,*

$$\mathbb{E}[X] \leq \frac{|S|}{|M|}$$

Proof.

$$\mathbb{E}[X] = \sum_{j \in S} \mathbb{P}(h(j) = h(i)) \leq \frac{|S|}{|M|}$$

□

3 Construction for 2-wise independent hash family

Some Additional Background on Finite Fields:
A finite field or Galois Field is a set $\mathbb{F}$, with operations

$$+ : \mathbb{F} \times \mathbb{F} \to \mathbb{F}$$

$$\cdot : \mathbb{F} \times \mathbb{F} \to \mathbb{F}$$

such that

1. $\langle \mathbb{F}, + \rangle$ forms an abelian group
2. $\langle \mathbb{F} \setminus \{0\}, \cdot \rangle$ forms an abelian group (for a 0 element where $0 + a = a + 0 = a$)
3. For all $a, b, c \in \mathbb{F}$, $a(b + c) = ab + bc$

The *order* of a finite field is the number of elements in $\mathbb{F}$. For any integer $m > 0$, and prime p, there exists a finite field with order p^m elements. There are no other fields and each field is unique up to isomorphism.

Fact 16.5.

For any integer $m > 0$, there exists a finite field, with 2^m elements denoted $\mathbb{F}_{2^m}$ (or GF(2^m)) over $[0, \ldots, 2^m - 1]$.

To perform addition of 2 elements (which is equivalent to subtraction here), we take their XOR.

To perform multiplication of 2 elements, we take the bit representation of elements and represent them as polynomials. We then specify an irreducible polynomial $g(x)$ of degree m in GF(2) and do polynomial multiplication modulo $g(x)$.

Division is defined as follows: We are in a finite field, so for every $i \in \mathbb{F}_{2^m}$, there exists a unique $i^{-1} \in \mathbb{F}_{2^m}$ such that $i \cdot i^{-1} = i^{-1} \cdot i = 1$. To calculate $\frac{i}{j}$ we use the extended Euclidean algorithm to find j^{-1} and multiply i and j^{-1}.

Additional Notes:
For finite fields with order p for a prime p, elements may be represented by integers in the range $[0, \cdots, p - 1]$. Addition, subtraction, and multiplication are defined as usual, but done modulo p, and division is defined with inverses using a similar line of reasoning above: to calculate $\frac{i}{j}$ find $j^{-1} \pmod{p}$ and multiply i and $j^{-1} \pmod{p}$. For finite fields with order p^m for $p > 2$, operations described above can be generalized.

Now, moving on to constructing such a hash family.

Definition 16.6. Define hash family $\mathcal{H} : \{0,1\}^m \to \{0,1\}^m$ as the uniform distribution h_{X_1,X_2} where $h_{X_1,X_2}(u \in \{0,1\}^m) = X_1 + u \cdot X_2$ (defined for all $X_1, X_2 \in \{0,1\}^m$).

The space used is $O(m)$ bits and we need $O(1)$ finite field operations.

Theorem 16.7. *$\mathcal{H}_m$ is a pairwise independent hash family.*

Proof. Consider arbitrary $i \neq j \in \mathbb{F}_2^m$ and $a, b \in \mathbb{F}_2^m$. We want $\mathbb{P}(h(i) = a \wedge h(j) = b) = \frac{1}{M^2}$

$$h(i) = a \wedge h(j) = b \Leftrightarrow \begin{cases} X_1 + i \cdot X_2 = a \\ X_1 + j \cdot X_2 = b \end{cases}$$

Solving the system of equations we get

$$X_1 = \frac{i \cdot b - a \cdot j}{i - j}, X_2 = \frac{a - b}{i - j}$$

i, j, a, b are all points in the finite field, and addition, subtraction, multiplication, and division of points in the finite field gives us another point in the finite field. When we fix i, j, a, b, we fix the values in $\{0,1\}^m$ that we need X_1 and X_2 to be. Since X_1 and X_2 are each drawn independently and uniformly at random from $\{0,1\}^m$, X_1 and X_2 satisfy the values necessary with probability $\frac{1}{(2^m)^2}$. $\square$

Observations:

1. Taking subset of the domain preserves 2-wise independence
2. Deleting bits and coordinates from the range preserves 2-wise independence.

Corollary 16.8. *For every $m, l \geq 1$, there exists a hash family $\mathcal{H}_{m,l} : \{0,1\}^m \to \{0,1\}^l$ that is 2-wise independent and requires $O(\max(m,l))$ bits to store.*

4 Digression: Application to Max-CUT

Problem 16.9. (Max-CUT) Consider a simple weighted graph $G = (V, E)$ where $V = [n]$. The goal is to find a subset $U \subseteq V$ to maximize the cut across U, denoted $\delta(U)$, where

$$\delta(U) = \sum_{i \in U, j \notin U} w_{ij}$$

Algorithm 16.10

1: Take a random $g : V \to \{0,1\}$, where g is drawn from a 2-wise independent hash family
2: Return $U = \{i : g(i) = 1\}$

Proposition 16.11. *Algorithm 16.10 returns a set U so that*

$$\mathbb{E}[\delta(U)] = \frac{w(G)}{2}$$

where

$$w(G) = \sum_{i,j \in E} w_{ij}$$

Note that $\frac{w(G)}{2}$ is a 2-approximation of the maximum cut, since the maximum cut is upper bounded by the sum of weights in the graph.

Proof.

$$\begin{aligned}
\mathbb{E}[\delta(U)] &= \sum_{i,j \in E} w_{ij} \cdot \mathbb{P}[(i \in U \wedge j \notin U) \wedge (i \notin U \wedge j \in U)] \\
&= \sum_{i,j \in E} w_{ij} \cdot [\mathbb{P}(i \in U \wedge j \notin U) + \mathbb{P}(i \notin U \wedge j \in U)] && \text{Sum of disjoint events} \\
&= \sum_{i,j \in E} w_{ij} \cdot [\mathbb{P}(g(i) = 1 \wedge g(j) = 0) + \mathbb{P}(g(i) = 0 \wedge g(j) = 1)] \\
&= \sum_{i,j \in E} w_{ij} \cdot \left[\frac{1}{4} + \frac{1}{4}\right] && \text{By Pairwise Independence} \\
&= \frac{w(G)}{2}
\end{aligned}$$

□

Observations:

- Since in expectation, $\delta(U)$ is $\frac{w(G)}{2}$, over a distribution of cuts, the best possible cut generated by all the g in supp($\mathcal{H}$) will have weight at least $\frac{w(G)}{2}$
- $g \leftarrow \mathcal{H} : \{0,1\}^{\lceil \log |V| \rceil} \to \{0,1\}$
- By Corollary 16.8, g needs $O(\log |V|)$ bits to specify.
- So, in polynomial time, we can enumerate all possible g and take the best cut $\tilde{U}$.

Theorem 16.12. *There is a deterministic polytime algorithm for Max-CUT such that for an input graph G, the algorithm outputs U with $\delta(U) \geq \frac{w(G)}{2}$.*

Proof.

Denote U_g as the cut induced by a function g, where $g : V \to \{0,1\}$

Since $\mathbb{E}_{g \leftarrow \mathcal{H}}[\delta(U_g)] = \frac{w(G)}{2}$, there exists g_0 such that $\delta(U_{g_0}) \geq \frac{w(G)}{2}$

We have a polytime algorithm where we can enumerate all possible $g_0 \leftarrow \mathcal{H}$, where there are $2^{O(\log |V|)}$ of them. □

5 k-wise Independent Hash Families

For pairwise independent hash families, if we look at 2 particular inputs, the marginals look independent from each other. For k-wise independent hash families, if we look at a k-tuple of inputs, and what the function maps to, that should look independent.

Definition 16.13. k-wise independent hash family

Consider a hash family $\mathcal{H} : [N] \to [M]$. We say $\mathcal{H}$ is k-wise independent if for all distinct $i_1, \cdots, i_k \in [N]$ and all (not necessarily distinct) $a_1, \cdots, a_k \in [M]$,

$$\mathbb{P}(\bigwedge_{j \in [k]} h(i_j) = a_j) = \frac{1}{M^k}$$

Theorem 16.14. *(k-wise independence implies k-wise universality) Suppose $\mathcal{H} : [N] \to [M]$ is a k-wise independent hash family. Then for all distinct $i_i, \cdots, i_k$,*

$$\mathbb{P}_{h \leftarrow \mathcal{H}}(h(i_1) = \cdots = h(i_k)) \leq \frac{1}{M^{k-1}}$$

Proof. This follows from the same reasoning as Theorem 16.3. There are M possibilities of what these values can equal, each occurring with probability $\frac{1}{M^k}$, so we get $\frac{M}{M^k} = \frac{1}{M^{k-1}}$. □

Definition 16.15. Define hash family $\mathcal{H}_m^{(k)} : \{0,1\}^m \to \{0,1\}^m$ as the uniform distribution over $h_{X_1 \cdots X_k}$ where $h_{X_1 \cdots X_k}(U) = \sum_{i=1}^k u^{i-1} \cdot X_i$

The space used is $O(k \cdot m)$ bits and we need $O(k)$ finite field operations to compute the hash.

Theorem 16.16. *$\mathcal{H}_m^{(k)}$ is k-wise independent*

Proof. Consider arbitrary (distinct) $i_1, \ldots, i_k$ and arbitrary (not necessarily distinct) outputs $a_1, \ldots, a_k$. Then,

$$\bigwedge_{j \in [k]} h_{X_1,\ldots,X_k}(i_j) = a_j \Leftrightarrow \begin{pmatrix} 1 & i_1 & i_1^2 & \cdots & i_1^{k-1} \\ 1 & i_2 & i_2^2 & \cdots & i_2^{k-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & i_k & i_k^2 & \cdots & i_k^{k-1} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_k \end{pmatrix} = \begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_k \end{pmatrix}$$

The above matrix is a Vandermonde Matrix. According to Wikipedia, the determinant is $\prod_{j<l}(i_j - i_l)$ which is non-zero if and only if $i_1, \ldots i_k$ are distinct. Since each of our $i_1, \ldots i_k$ are distinct, we have an invertible matrix and therefore a unique solution to $(X_1, \ldots, X_k)$. This occurs with probability $\frac{1}{(2^m)^k}$. □

Corollary 16.17. *For any $m, l \geq 1$, there exists a hash family $\mathcal{H}_{m,l}^{(k)}$ that is k-wise independent and uses $O(k \cdot \max(m, l))$ bits.*

This follows from the same reasoning as 2-wise independence in Corollary 16.8.