CSE 347 Lecture 19

Randomized Algorithms II: Hashing

1. Pairwise independence: "universal" hash functions
2. Linear functions are universal

Hashing is great — it's hard to beat dictionaries that only cost $O(1)$ time for inserts, lookups, and deletes. But, so far it has been based on an unfulfilled promise that somehow we can find a good hash function that evenly spreads the elements that we've seen. And indeed, if our table has size m and our universe of elements has size $N \geq m^2$, then the pigeonhole principle tells us that for any fixed choice of hash function, there is a choice of m elements that will all hash to the same value. So, from this perspective, hashing seems impossible. Here, randomization will come to our rescue. Indeed, we already saw that if our elements are drawn uniformly at random from $0-(N-1)$, then the function $h(x) = x \bmod m$ is a fine choice, producing no collisions so long as we only inserted $O(\sqrt{m\delta})$ elements (w.p. $1-\delta$).

Unfortunately, the strategy we used to turn our naive trees into treaps - assigning a random priority - won't work here. Why? We need to remem-

ber what priority we assign to the element x, and when x arrives again in the future, we need to retrieve this value in $O(1)$ time. But then, even determining whether or not we already assigned a value to x is enough to support a dictionary, and so we would need something like a hash table to store the priority. This is circular.

Pairwise independence: "universal" hash functions

What we need is a way to choose a hash function h that we can compute in $O(1)$ time, that is "random enough" — we can't afford to assign every element x its own independent $h(x)$. The key observation is that what we really care about are collisions, which are events that only refer to two hash values at a time. It's enough that pairs $(h(x), h(y))$ are independently and uniformly distributed, as opposed to full, mutual independence among the n elements we've inserted so far.

In a little more detail, we'll have a family of hash functions $\mathcal{H}$, containing hash functions $h: \{0, 1, \ldots, N-1\} \to \{0, 1, \ldots, m-1\}$, i.e., mapping our universe of N elements to our table of size m. The actual function $h \in \mathcal{H}$ that we use will be chosen at random somehow. (Uniformly at random will turn out

to be fine for us, actually.) So, let's denote the random choice of hash function with the random variable H. That is, H is a random function, and for every element $x, y, z, \ldots \in \{0, 1, \ldots, N-1\}$, $H(x)$, $H(y)$, $H(z)$, ... are random values in $\{0, 1, \ldots, m-1\}$. The property that we really need is that it's unlikely for any two of our elements to collide:

Definition A random function H is a universal hash function if for every pair x, y in its domain, if its range has size m, then $\Pr[H(x) = H(y)] = \frac{1}{m}$.

A universal hash function is enough.

Theorem Let H be a universal hash function, and suppose S is a subset of the domain of H of size at most $\sqrt{2m\delta}$, where m is the size of the range of H. Then every $x, y \in S$ have $H(x) \neq H(y)$ with probability $1-\delta$.

Proof: For each pair $(x, y) \in S$, we define the "indicator" random variable $I[H(x) = H(y)] = \begin{cases} 1 \text{ if } H(x) = H(y) \\ 0 \text{ otherwise} \end{cases}$.

Now, the random variable $\sum_{x,y \in S,\, x \neq y} I[H(x) = H(y)]$ counts the number of pairs that collide, and so we wish to argue that it is 0 with probability $1-\delta$. We will do this in two steps: first we'll argue that it is small on average, then we'll argue that it is far enough from its average to be nonzero with probability at most δ.

The key property of its "expected" (average) value is

that this is a linear operation on random variables— so here $\mathbb{E}\left[\sum_{x,y\in S, x\neq y} I[H(x)=H(y)]\right] = \sum_{x,y\in S, x\neq y} \mathbb{E}[I[H(x)=H(y)]]$. Now, what is $\mathbb{E}[I[H(x)=H(y)]]$? By definition, $\mathbb{E}[I[H(x)=H(y)]] = 0\cdot \Pr[H(x)\neq H(y)] + 1\cdot \Pr[H(x)=H(y)] = \frac{1}{m}$ since H is universal. Now, since there are $\frac{|S|(|S|-1)}{2} < \frac{1}{2}|S|^2 \leq \frac{2m\delta}{2}$ pairs of distinct elements in S, $\mathbb{E}\left[\sum_{x,y\in S, x\neq y} I[H(x)=H(y)]\right] < \frac{m\delta}{m} = \delta$.

For our second step, we need Markov's Inequality:

Theorem (Markov's Inequality): For any nonnegative random variable X, $\Pr[X > k\cdot\mathbb{E}[X]] < \frac{1}{k}$ for all $k>0$.

(Proof? $\mathbb{E}[X] = \mathbb{E}[X \mid X > k\cdot\mathbb{E}[X]]\cdot \Pr[X > k\mathbb{E}[X]] + \mathbb{E}[X \mid X \leq k\cdot\mathbb{E}[X]]\cdot \Pr[X\leq k\cdot\mathbb{E}[X]]$
$> k\mathbb{E}[X]$ ≥ 0
If $\Pr[X > k\mathbb{E}[X]] \geq \frac{1}{k}$, this is $\geq k\mathbb{E}[X]\Pr[X > k\cdot\mathbb{E}[X]] \geq \mathbb{E}[X]$ ✳)

Now, here, $\Pr[\text{some } x\neq y \text{ have } H(x)=H(y)] \leq$
$\Pr\left[\sum_{x,y\in S, x\neq y} I[H(x)=H(y)] > \frac{1}{\delta}\cdot \underbrace{\mathbb{E}\left[\sum_{x,y\in S, x\neq y} I[H(x)=H(y)]\right]}_{(<1 \text{ by above...})}\right]$

This is less than δ by Markov's inequality. ▨

We can also bound the load on each bucket once we insert more than $\sqrt{2m\delta}$ elements—

Corollary: If H is universal and S contains $\leq \sqrt{Cm\delta}$ elements, then with probability $1-\delta$, no bucket contains more than C elements

Proof: Fix any $x\in S$. Now, the expected number of $y\neq x$ that hash to the same index is $\mathbb{E}\left[\sum_{y\neq x\in S} I[H(x)=H(y)]\right] < \frac{|S|}{m} \leq \frac{\sqrt{C\delta}}{\sqrt{m}}$

Markov's inequality now gives that the number is greater than $C \geq \sqrt{\frac{Cm}{\delta}}\cdot \mathbb{E}\left[\sum_{y\neq x\in S} I[H(x)=H(y)]\right]$ with probability at most

$\sqrt{\frac{\delta}{cm}}$. Therefore, by a union bound over the $|S| \leq \sqrt{cm\delta}$ events B_x = "More than c $y \in S$ have $H(y) = H(x)$", we find

$\Pr[$Some $x \in S$ has more than c $y \in S$ with $H(y) = H(x)]$

$< |S| \cdot \sqrt{\frac{\delta}{cm}} \leq \sqrt{cm\delta} \cdot \sqrt{\frac{\delta}{cm}} = \delta$ ▨

So as long as we don't insert more than $O(\sqrt{m})$ elements, we preserve $O(1)$ time accesses.

Linear hash functions are universal

Now it comes time to actually choose a family of hash functions. The simplest choice is as follows. we let m be a prime number. We then break the representation of elements x of our universe into l blocks of size at most $\log_2 m$ bits each — that is, $x = (x_1, x_2, \ldots, x_l)$.

We interpret each x_i as a number in the range $0, 1, \ldots, m-1$. Now, for each vector $\vec{a} \in \{0, 1, \ldots, m-1\}^l$ we define the hash function $h_{\vec{a}}(x) = \langle \vec{a}, x \rangle \bmod m = \sum_{i=1}^{l} a_i x_i \bmod m$.
H will simply be $h_{\vec{a}}$ for a uniform random choice of $\vec{a}$.

Theorem H is a universal hash function.

Proof: Fix $x = (x_1, \ldots, x_l)$ and $y = (y_1, \ldots, y_l)$, two distinct elements of the universe. Thus, there is some index i in which $x_i \neq y_i$. Let's consider any setting of the a_j s.t. $j \neq i$; a_i is still chosen independently and uniformly at random. Let's write $H(x) = a_i x_i + r \bmod m$ and $H(y) = a_i y_i + s \bmod m$. Now $H(x) = H(y)$ if and only if $a_i x_i + r = a_i y_i + s \bmod m$. We can rewrite this as

$a_i(x_i - y_i) = \underbrace{s-r}_{b} \bmod m$. How many choices for a_i satisfy this equation? One-

since $x_i - y_i = z \neq 0$, there is some $z^{-1} \in \{1, \ldots, m-1\}$ such that $a_i = a_i \cdot z \cdot z^{-1} = b \cdot z^{-1} \bmod m$. That is: a_i is uniquely determined to be $b \cdot z^{-1} \bmod m$. Therefore,

$$\Pr_{\vec{a}}[H(x) = H(y)] = \sum_{(a_j : j \neq i)} \Pr[A_i x_i + R = A_i y_i + S \bmod m \mid A_j = a_j \ \forall j \neq i] \cdot \Pr[A_j = a_j \ \forall j \neq i]$$

$$= \frac{1}{m} \cdot \underbrace{\sum_{(a_j : j \neq i)} \Pr[A_j = a_j \ \forall j \neq i]}_{=1} = \frac{1}{m} \ \square$$

We remark that if we add an extra "constant term" $b \in \{0, \ldots, m-1\}$, so that $h_{\vec{a},b}(x) = \sum_{i=1}^{l} a_i x_i + b \bmod m$, then this is still a universal choice of hash function that is now also individually uniformly distributed: indeed, whatever x and $\vec{a}$ are, $\Pr[H(x) = z] = \Pr[\sum_{i=1}^{l} a_i x_i + b = z \bmod m] = \frac{1}{m}$ since this equation is only satisfied by $b = z - \sum_{i=1}^{l} a_i x_i \bmod m$.

More generally, the marginal distribution on any two hash values $H(x)$ and $H(y)$ for $x \neq y$ is identical to two independent, uniformly random elements of $\{0, \ldots, m-1\}$:

$\Pr[H(x) = w \text{ and } H(y) = z] = \Pr[a_i x_i + r + b = w \text{ and } a_i y_i + s + b = z]$

Now, whatever $w - r$ and $z - s$ are, the system of equations $a_i x_i + b = w - r \bmod m$ and $a_i y_i + b = z - s \bmod m$ has a unique solution: $a_i(x_i - y_i) = w - r - (z - s) \bmod m$ where $x_i - y_i \neq 0$, so we must have $a_i = (x_i - y_i)^{-1}(w - r + s - z) \bmod m$, and likewise then we must have $b = w - r - a_i x_i \bmod m$. (Note that then $a_i y_i + b = a_i y_i + w - r - a_i x_i = a_i(y_i - x_i) + w - r$

$= (x_i - y_i)^{-1}(w - r + s - z)(y_i - x_i) + w - r$

$$= s - z - (w - r) + w - r \bmod m$$

so both constraints are satisfied by this choice of $a_i + b$. Thus, $\Pr[H(x) = w \text{ and } H(y) = z] = \frac{1}{m^2}$, exactly as if these were independent, uniform random variables. This is sometimes useful in designing algorithms.