Hashing Functions

He Sun

University of Edinburgh

Dictionary data type

Dictionary. Given a universe U of possible elements, maintain a subset $S \subseteq U$ so that inserting, deleting, and searching in S are efficient.

Dictionary data type

Dictionary. Given a universe U of possible elements, maintain a subset $S \subseteq U$ so that inserting, deleting, and searching in S are efficient.

Dictionary interface.

- `create`(): initialise a dictionary with $S = \emptyset$.

Dictionary data type

Dictionary. Given a universe U of possible elements, maintain a subset $S \subseteq U$ so that inserting, deleting, and searching in S are efficient.

Dictionary interface.

- $\texttt{create}()$: initialise a dictionary with $S = \emptyset$.
- $\mathsf{insert}(u)$: add element $u \in U$ to S.

Dictionary data type

Dictionary. Given a universe U of possible elements, maintain a subset $S \subseteq U$ so that inserting, deleting, and searching in S are efficient.

Dictionary interface.

- create(): initialise a dictionary with $S = \emptyset$.
- insert(u): add element $u \in U$ to S.
- delete(u): delete u from S (if u is currently in S).

Dictionary data type

Dictionary. Given a universe U of possible elements, maintain a subset $S \subseteq U$ so that inserting, deleting, and searching in S are efficient.

Dictionary interface.

- create(): initialise a dictionary with $S = \emptyset$.
- insert(u): add element $u \in U$ to S.
- delete(u): delete u from S (if u is currently in S).
- lookup(u): is u in S ?

Dictionary data type

Dictionary. Given a universe U of possible elements, maintain a subset $S \subseteq U$ so that inserting, deleting, and searching in S are efficient.

Dictionary interface.

- create(): initialise a dictionary with $S = \emptyset$.
- insert(u): add element $u \in U$ to S.
- delete(u): delete u from S (if u is currently in S).
- lookup(u): is u in S ?

Easy solution. Build an array b of length $|U|$, where $b[u]$ indicates if u appears in S.

Dictionary data type

Dictionary. Given a universe U of possible elements, maintain a subset $S \subseteq U$ so that inserting, deleting, and searching in S are efficient.

Dictionary interface.

- create(): initialise a dictionary with $S = \emptyset$.
- insert(u): add element $u \in U$ to S.
- delete(u): delete u from S (if u is currently in S).
- lookup(u): is u in S ?

Easy solution. Build an array b of length $|U|$, where $b[u]$ indicates if u appears in S.

Challenge. Universe U can be extremely large so defining an array b is infeasible.

Applications. File systems, databases, networks, cryptography, web caching.

Hashing

Hash function. $h : U \to [n]$, where $[n] := \{0, 1, \ldots, n-1\}$.

Hashing. Create an array a of length n. When processing element u, access array element $a[h(u)]$.

Hashing

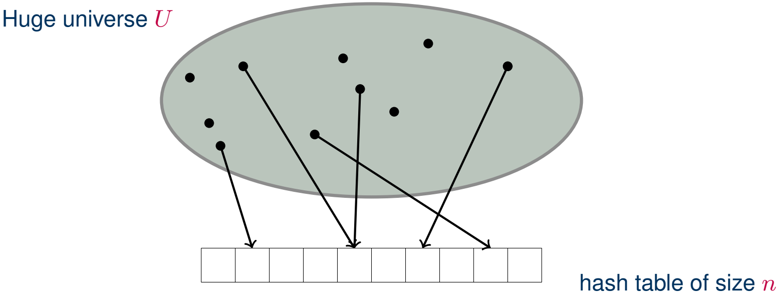
Hash function. $h : U \to [n]$, where $[n] := \{0, 1, \ldots, n-1\}$.

Hashing. Create an array a of length n. When processing element u, access array element $a[h(u)]$.

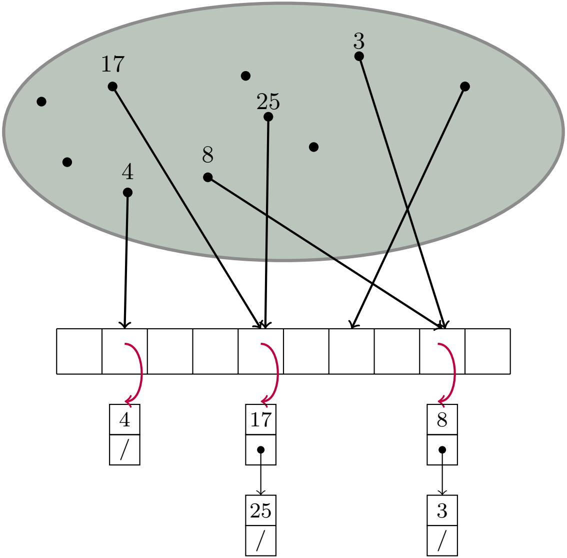
birthday paradox

Collision. When $h(u) = h(v)$ but $n \neq v$.

- collision is expected after $\Theta(\sqrt{n})$ random insertions.
- Separate chaining: $a[i]$ stores linked list of elements u with $h(u) = i$.



Hashing with chaining



Hashing performance

Ideal hash function. Map m elements uniformly at random to n hash slots.

- Running time depends on length of chains.

Hashing performance

Ideal hash function. Map m elements uniformly at random to n hash slots.

- Running time depends on length of chains.
- Average length of chain = m/n.

Hashing performance

Ideal hash function. Map m elements uniformly at random to n hash slots.

- Running time depends on length of chains.
- Average length of chain = m/n.
- Choose $n \approx m \Rightarrow$ expect $O(1)$ per insert, lookup, or delete.

Hashing performance

Ideal hash function. Map m elements uniformly at random to n hash slots.

- Running time depends on length of chains.
- Average length of chain = m/n.
- Choose $n \approx m \Rightarrow$ expect $O(1)$ per insert, lookup, or delete.

Challenge. Explicit hash function h that achieves $O(1)$ per operation.

Hashing performance

Ideal hash function. Map m elements uniformly at random to n hash slots.

- Running time depends on length of chains.
- Average length of chain = m/n.
- Choose $n \approx m \Rightarrow$ expect $O(1)$ per insert, lookup, or delete.

Challenge. Explicit hash function h that achieves $O(1)$ per operation.

Approach. Use randomisation for the choice of h.

adversary knows the randomised algorithm you are using, but doesn't know random choices that the algorithm makes.

Universal hashing (Carter-Wegman 1980s)

A universal family of hash functions is a set of hash functions H mapping a universe U to the set $\{0, 1, \ldots, n-1\}$ such that, for any pair of elements $u \neq v$,

$$\mathbb{P}_{h \in H}\left[h(u) = h(v)\right] \leq 1/n.$$

Universal hashing (Carter-Wegman 1980s)

A universal family of hash functions is a set of hash functions H mapping a universe U to the set $\{0, 1, \ldots, n-1\}$ such that, for any pair of elements $u \neq v$,

$$\mathbb{P}_{h \in H}\left[h(u) = h(v)\right] \leq 1/n.$$

Example. $U = \{a, b, c, d, e, f\}, n = 2$

	a	b	c	d	e	f
$h_1(x)$	0	1	0	1	0	1
$h_2(x)$	0	0	0	1	1	1

$$\mathbb{P}_{h \in H}[h(a) = h(b)] = 1/2$$

Universal hashing (Carter-Wegman 1980s)

A universal family of hash functions is a set of hash functions H mapping a universe U to the set $\{0, 1, \ldots, n-1\}$ such that, for any pair of elements $u \neq v$,

$$\mathbb{P}_{h \in H}\left[h(u) = h(v)\right] \leq 1/n.$$

Example. $U = \{a, b, c, d, e, f\}, n = 2$

	a	b	c	d	e	f
$h_1(x)$	0	1	0	1	0	1
$h_2(x)$	0	0	0	1	1	1

$$\mathbb{P}_{h \in H}[h(a) = h(b)] = 1/2$$
$$\mathbb{P}_{h \in H}[h(a) = h(c)] = 1$$

Universal hashing (Carter-Wegman 1980s)

A universal family of hash functions is a set of hash functions H mapping a universe U to the set $\{0, 1, \ldots, n-1\}$ such that, for any pair of elements $u \neq v$,

$$\mathbb{P}_{h \in H}\left[h(u) = h(v)\right] \leq 1/n.$$

Example. $U = \{a, b, c, d, e, f\}, n = 2$

	a	b	c	d	e	f
$h_1(x)$	0	1	0	1	0	1
$h_2(x)$	0	0	0	1	1	1

$$\mathbb{P}_{h \in H}[h(a) = h(b)] = 1/2$$
$$\mathbb{P}_{h \in H}[h(a) = h(c)] = 1$$
$$\mathbb{P}_{h \in H}[h(a) = h(d)] = 0$$

Universal hashing (Carter-Wegman 1980s)

A universal family of hash functions is a set of hash functions H mapping a universe U to the set $\{0, 1, \ldots, n-1\}$ such that, for any pair of elements $u \neq v$,

$$\mathbb{P}_{h \in H}\left[h(u) = h(v)\right] \leq 1/n.$$

Example. $U = \{a, b, c, d, e, f\}, n = 2$

	a	b	c	d	e	f
$h_1(x)$	0	1	0	1	0	1
$h_2(x)$	0	0	0	1	1	1

$\mathbb{P}_{h \in H}[h(a) = h(b)] = 1/2$
$\mathbb{P}_{h \in H}[h(a) = h(c)] = 1$
$\mathbb{P}_{h \in H}[h(a) = h(d)] = 0$
not universal

Universal hashing (Carter-Wegman 1980s)

A universal family of hash functions is a set of hash functions H mapping a universe U to the set $\{0, 1, \ldots, n-1\}$ such that, for any pair of elements $u \neq v$,

$$\mathbb{P}_{h \in H}\left[h(u) = h(v)\right] \leq 1/n.$$

Example. $U = \{a, b, c, d, e, f\}, n = 2$

	a	b	c	d	e	f
$h_1(x)$	0	1	0	1	0	1
$h_2(x)$	0	0	0	1	1	1

$\mathbb{P}_{h \in H}[h(a) = h(b)] = 1/2$
$\mathbb{P}_{h \in H}[h(a) = h(c)] = 1$
$\mathbb{P}_{h \in H}[h(a) = h(d)] = 0$
not universal

	a	b	c	d	e	f
$h_1(x)$	0	1	0	1	0	1
$h_2(x)$	0	0	0	1	1	1
$h_3(x)$	0	0	1	0	1	1
$h_4(x)$	1	0	0	1	1	0

$\mathbb{P}_{h \in H}[h(a) = h(b)] = 1/2$

Universal hashing (Carter-Wegman 1980s)

A universal family of hash functions is a set of hash functions H mapping a universe U to the set $\{0, 1, \ldots, n-1\}$ such that, for any pair of elements $u \neq v$,

$$\mathbb{P}_{h \in H}\left[h(u) = h(v)\right] \leq 1/n.$$

Example. $U = \{a, b, c, d, e, f\}, n = 2$

	a	b	c	d	e	f
$h_1(x)$	0	1	0	1	0	1
$h_2(x)$	0	0	0	1	1	1

$\mathbb{P}_{h \in H}[h(a) = h(b)] = 1/2$
$\mathbb{P}_{h \in H}[h(a) = h(c)] = 1$
$\mathbb{P}_{h \in H}[h(a) = h(d)] = 0$
not universal

	a	b	c	d	e	f
$h_1(x)$	0	1	0	1	0	1
$h_2(x)$	0	0	0	1	1	1
$h_3(x)$	0	0	1	0	1	1
$h_4(x)$	1	0	0	1	1	0

$\mathbb{P}_{h \in H}[h(a) = h(b)] = 1/2$
$\mathbb{P}_{h \in H}[h(a) = h(c)] = 1/2, \cdots$

Universal hashing (Carter-Wegman 1980s)

A universal family of hash functions is a set of hash functions H mapping a universe U to the set $\{0, 1, \ldots, n-1\}$ such that, for any pair of elements $u \neq v$,

$$\mathbb{P}_{h \in H}\left[h(u) = h(v)\right] \leq 1/n.$$

Example. $U = \{a, b, c, d, e, f\}, n = 2$

	a	b	c	d	e	f
$h_1(x)$	0	1	0	1	0	1
$h_2(x)$	0	0	0	1	1	1

$\mathbb{P}_{h \in H}[h(a) = h(b)] = 1/2$
$\mathbb{P}_{h \in H}[h(a) = h(c)] = 1$
$\mathbb{P}_{h \in H}[h(a) = h(d)] = 0$
not universal

	a	b	c	d	e	f
$h_1(x)$	0	1	0	1	0	1
$h_2(x)$	0	0	0	1	1	1
$h_3(x)$	0	0	1	0	1	1
$h_4(x)$	1	0	0	1	1	0

$\mathbb{P}_{h \in H}[h(a) = h(b)] = 1/2$
$\mathbb{P}_{h \in H}[h(a) = h(c)] = 1/2, \cdots$
$\mathbb{P}_{h \in H}[h(a) = h(f)] = 0, \cdots$

Universal hashing (Carter-Wegman 1980s)

A universal family of hash functions is a set of hash functions H mapping a universe U to the set $\{0, 1, \ldots, n-1\}$ such that, for any pair of elements $u \neq v$,

$$\mathbb{P}_{h \in H}\left[h(u) = h(v)\right] \leq 1/n.$$

Example. $U = \{a, b, c, d, e, f\}, n = 2$

	a	b	c	d	e	f
$h_1(x)$	0	1	0	1	0	1
$h_2(x)$	0	0	0	1	1	1

$\mathbb{P}_{h \in H}[h(a) = h(b)] = 1/2$
$\mathbb{P}_{h \in H}[h(a) = h(c)] = 1$
$\mathbb{P}_{h \in H}[h(a) = h(d)] = 0$
not universal

	a	b	c	d	e	f
$h_1(x)$	0	1	0	1	0	1
$h_2(x)$	0	0	0	1	1	1
$h_3(x)$	0	0	1	0	1	1
$h_4(x)$	1	0	0	1	1	0

$\mathbb{P}_{h \in H}[h(a) = h(b)] = 1/2$
$\mathbb{P}_{h \in H}[h(a) = h(c)] = 1/2, \cdots$
$\mathbb{P}_{h \in H}[h(a) = h(f)] = 0, \cdots$
universal

Other expected features of hash functions:

- We can select a random h efficiently;

Universal hashing (Carter-Wegman 1980s)

A universal family of hash functions is a set of hash functions H mapping a universe U to the set $\{0, 1, \ldots, n-1\}$ such that, for any pair of elements $u \neq v$,

$$\mathbb{P}_{h \in H}\left[h(u) = h(v)\right] \leq 1/n.$$

Example. $U = \{a, b, c, d, e, f\}, n = 2$

	a	b	c	d	e	f
$h_1(x)$	0	1	0	1	0	1
$h_2(x)$	0	0	0	1	1	1

$\mathbb{P}_{h \in H}[h(a) = h(b)] = 1/2$
$\mathbb{P}_{h \in H}[h(a) = h(c)] = 1$
$\mathbb{P}_{h \in H}[h(a) = h(d)] = 0$
not universal

	a	b	c	d	e	f
$h_1(x)$	0	1	0	1	0	1
$h_2(x)$	0	0	0	1	1	1
$h_3(x)$	0	0	1	0	1	1
$h_4(x)$	1	0	0	1	1	0

$\mathbb{P}_{h \in H}[h(a) = h(b)] = 1/2$
$\mathbb{P}_{h \in H}[h(a) = h(c)] = 1/2, \cdots$
$\mathbb{P}_{h \in H}[h(a) = h(f)] = 0, \cdots$
universal

Other expected features of hash functions:

- We can select a random h efficiently;
- We can also compute $h(u)$ efficiently.

Universal hashing (Carter-Wegman 1980s)

A universal family of hash functions is a set of hash functions H mapping a universe U to the set $\{0, 1, \ldots, n-1\}$ such that, for any pair of elements $u \neq v$,

$$\mathbb{P}_{h \in H}\left[h(u) = h(v)\right] \leq 1/n.$$

Example. $U = \{a, b, c, d, e, f\}, n = 2$

	a	b	c	d	e	f
$h_1(x)$	0	1	0	1	0	1
$h_2(x)$	0	0	0	1	1	1

$\mathbb{P}_{h \in H}[h(a) = h(b)] = 1/2$
$\mathbb{P}_{h \in H}[h(a) = h(c)] = 1$
$\mathbb{P}_{h \in H}[h(a) = h(d)] = 0$
not universal

	a	b	c	d	e	f
$h_1(x)$	0	1	0	1	0	1
$h_2(x)$	0	0	0	1	1	1
$h_3(x)$	0	0	1	0	1	1
$h_4(x)$	1	0	0	1	1	0

$\mathbb{P}_{h \in H}[h(a) = h(b)] = 1/2$
$\mathbb{P}_{h \in H}[h(a) = h(c)] = 1/2, \cdots$
$\mathbb{P}_{h \in H}[h(a) = h(f)] = 0, \cdots$
universal

Other expected features of hash functions:

- We can select a random h efficiently;
- We can also compute $h(u)$ efficiently.

Universal hashing: analysis

LEMMA

Let H be a universal family of hash functions mapping a universe U to the set $\{0, 1, \ldots, n-1\}$. Let $h \in H$ be chosen uniformly at random from H; let $S \subseteq U$ be a subset of size at most n, and $u \notin S$. Then, the expected number of items in S that collide with u is at most 1.

Universal hashing: analysis

LEMMA

Let H be a universal family of hash functions mapping a universe U to the set $\{0, 1, \ldots, n-1\}$. Let $h \in H$ be chosen uniformly at random from H; let $S \subseteq U$ be a subset of size at most n, and $u \notin S$. Then, the expected number of items in S that collide with u is at most 1.

Proof. For any $s \in S$, define random variable $X_s = 1$ if $h(s) = h(u)$, and 0 otherwise. Let X be a random variable counting the total number of collisions with u, so $X = \sum_{s \in S} X_s$.

Universal hashing: analysis

Lemma

Let H be a universal family of hash functions mapping a universe U to the set $\{0, 1, \ldots, n-1\}$. Let $h \in H$ be chosen uniformly at random from H; let $S \subseteq U$ be a subset of size at most n, and $u \notin S$. Then, the expected number of items in S that collide with u is at most 1.

Proof. For any $s \in S$, define random variable $X_s = 1$ if $h(s) = h(u)$, and 0 otherwise. Let X be a random variable counting the total number of collisions with u, so $X = \sum_{s \in S} X_s$.

$$\mathbb{E}[X] = \mathbb{E}\left[\sum_{s \in S} X_s\right] = \sum_{s \in S} \mathbb{E}[X_s] = \sum_{s \in S} \mathbb{P}[X_s = 1] \leq \sum_{s \in S} \frac{1}{n}$$
$$= |S|/n \leq 1$$

linearity of expectation

def. of universal hasing

Universal hashing: analysis

Lemma

Let H be a universal family of hash functions mapping a universe U to the set $\{0, 1, \ldots, n-1\}$. Let $h \in H$ be chosen uniformly at random from H; let $S \subseteq U$ be a subset of size at most n, and $u \notin S$. Then, the expected number of items in S that collide with u is at most 1.

Proof. For any $s \in S$, define random variable $X_s = 1$ if $h(s) = h(u)$, and 0 otherwise. Let X be a random variable counting the total number of collisions with u, so $X = \sum_{s \in S} X_s$.

$$\mathbb{E}[X] = \mathbb{E}\left[\sum_{s \in S} X_s\right] = \sum_{s \in S} \mathbb{E}[X_s] = \sum_{s \in S} \mathbb{P}[X_s = 1] \leq \sum_{s \in S} \frac{1}{n}$$
$$= |S|/n \leq 1$$

linearity of expectation

def. of universal hasing

Q: How can we design a universal class of hash functions?

Designing a universal family of hash functions

Modulus. We will use a prime number p for the size of the hash table.

Designing a universal family of hash functions

Modulus. We will use a prime number p for the size of the hash table.

Integer encoding. Identify each element $u \in U$ with a base-p integer of r digits: $x = (x_1, x_2, \ldots, x_r)$.

Designing a universal family of hash functions

Modulus. We will use a prime number p for the size of the hash table.

Integer encoding. Identify each element $u \in U$ with a base-p integer of r digits: $x = (x_1, x_2, \ldots, x_r)$.

Hash functions. Let A = set of all r-digits $(a_1, a_2, \ldots, a_r)$, where $0 \leq a_i < p$. For each $a = (a_1, a_2, \ldots, a_r)$ with $0 \leq a_i < p$, define

$$h_a(x) = \left(\sum_{i=1}^{r} a_i x_i \right) \mod p.$$

Designing a universal family of hash functions

Modulus. We will use a prime number p for the size of the hash table.

Integer encoding. Identify each element $u \in U$ with a base-p integer of r digits: $x = (x_1, x_2, \ldots, x_r)$.

Hash functions. Let $A =$ set of all r-digits $(a_1, a_2, \ldots, a_r)$, where $0 \leq a_i < p$. For each $a = (a_1, a_2, \ldots, a_r)$ with $0 \leq a_i < p$, define

$$h_a(x) = \left(\sum_{i=1}^{r} a_i x_i \right) \mod p.$$

Hash function family. $H = \{h_a : a \in A\}$

Designing a universal family of hash functions

Theorem

$H = \{h_a : a \in A\}$ is a universal family of hash functions.

Designing a universal family of hash functions

Theorem

$H = \{h_a : a \in A\}$ is a universal family of hash functions.

Proof: Let $x = (x_1, x_2, \ldots, x_r)$ and $y = (y_1, \ldots, y_r)$ be two distinct elements of U. We need to show that $\mathbb{P}[h_a(x) = h_a(y)] \leq 1/p$.

Designing a universal family of hash functions

THEOREM

$H = \{h_a : a \in A\}$ is a universal family of hash functions.

Proof: Let $x = (x_1, x_2, \ldots, x_r)$ and $y = (y_1, \ldots, y_r)$ be two distinct elements of U. We need to show that $\mathbb{P}[h_a(x) = h_a(y)] \leq 1/p$.

- Since $x \neq y$, there exists an integer j such that $x_j \neq y_j$.

Designing a universal family of hash functions

Theorem

$H = \{h_a : a \in A\}$ is a universal family of hash functions.

Proof: Let $x = (x_1, x_2, \ldots, x_r)$ and $y = (y_1, \ldots, y_r)$ be two distinct elements of U. We need to show that $\mathbb{P}[h_a(x) = h_a(y)] \leq 1/p$.

- Since $x \neq y$, there exists an integer j such that $x_j \neq y_j$.
- We have $h_a(x) = h_a(y)$ iff $\sum_{i=1}^r a_i x_i \equiv \sum_{i=1}^r a_i y_i \mod p$, i.e.,

$$a_j \underbrace{(y_j - x_j)}_{z} \equiv \underbrace{\sum_{i \neq j} a_i (x_i - y_i)}_{m} \mod p$$

Designing a universal family of hash functions

Theorem

$H = \{h_a : a \in A\}$ is a universal family of hash functions.

Proof: Let $x = (x_1, x_2, \ldots, x_r)$ and $y = (y_1, \ldots, y_r)$ be two distinct elements of U. We need to show that $\mathbb{P}[h_a(x) = h_a(y)] \leq 1/p$.

- Since $x \neq y$, there exists an integer j such that $x_j \neq y_j$.
- We have $h_a(x) = h_a(y)$ iff $\sum_{i=1}^{r} a_i x_i \equiv \sum_{i=1}^{r} a_i y_i \mod p$, i.e.,

$$a_j \underbrace{(y_j - x_j)}_{z} \equiv \underbrace{\sum_{i \neq j} a_i (x_i - y_i)}_{m} \mod p$$

- Can assume a was chosen uniformly at random by first selecting all coordinates a_i where $i \neq j$, then selecting a_j at random. Thus, we can assume a_i is fixed for all coordinates $i \neq j$.

Designing a universal family of hash functions

Theorem

$H = \{h_a : a \in A\}$ is a universal family of hash functions.

Proof: Let $x = (x_1, x_2, \ldots, x_r)$ and $y = (y_1, \ldots, y_r)$ be two distinct elements of U. We need to show that $\mathbb{P}[h_a(x) = h_a(y)] \leq 1/p$.

- Since $x \neq y$, there exists an integer j such that $x_j \neq y_j$.
- We have $h_a(x) = h_a(y)$ iff $\sum_{i=1}^{r} a_i x_i \equiv \sum_{i=1}^{r} a_i y_i \mod p$, i.e.,

$$a_j \underbrace{(y_j - x_j)}_{z} \equiv \underbrace{\sum_{i \neq j} a_i (x_i - y_i)}_{m} \mod p$$

- Can assume a was chosen uniformly at random by first selecting all coordinates a_i where $i \neq j$, then selecting a_j at random. Thus, we can assume a_i is fixed for all coordinates $i \neq j$.
- Since p is prime, $a_j z \equiv m \mod p$ has at most one solution among p possibilities. $\Leftarrow$ See lemma on the next slide.

Designing a universal family of hash functions

Theorem

$H = \{h_a : a \in A\}$ is a universal family of hash functions.

Proof: Let $x = (x_1, x_2, \ldots, x_r)$ and $y = (y_1, \ldots, y_r)$ be two distinct elements of U. We need to show that $\mathbb{P}[h_a(x) = h_a(y)] \leq 1/p$.

- Since $x \neq y$, there exists an integer j such that $x_j \neq y_j$.
- We have $h_a(x) = h_a(y)$ iff $\sum_{i=1}^{r} a_i x_i \equiv \sum_{i=1}^{r} a_i y_i \mod p$, i.e.,

$$a_j \underbrace{(y_j - x_j)}_{z} \equiv \underbrace{\sum_{i \neq j} a_i (x_i - y_i)}_{m} \mod p$$

- Can assume a was chosen uniformly at random by first selecting all coordinates a_i where $i \neq j$, then selecting a_j at random. Thus, we can assume a_i is fixed for all coordinates $i \neq j$.
- Since p is prime, $a_j z \equiv m \mod p$ has at most one solution among p possibilities. $\Leftarrow$ See lemma on the next slide.
- Thus $\mathbb{P}[h_a(x) = h_a(y)] \leq 1/p$.

Number theory fact

Fact

Let p be prime, and let $z \not\equiv 0 \mod p$. Then $az \equiv m \mod p$ has at most one solution $0 \leq a < p$.

Number theory fact

Fact

Let p be prime, and let $z \not\equiv 0 \mod p$. Then $az \equiv m \mod p$ has at most one solution $0 \leq a < p$.

Proof. The proof is by contradiction.

- Suppose $0 \leq a_1 < p$ and $0 \leq a_2 < p$ are two different solutions.

Number theory fact

Fact

Let p be prime, and let $z \not\equiv 0 \mod p$. Then $az \equiv m \mod p$ has at most one solution $0 \leq a < p$.

Proof. The proof is by contradiction.

- Suppose $0 \leq a_1 < p$ and $0 \leq a_2 < p$ are two different solutions.
- Then $(a_1 - a_2)z \equiv 0 \mod p$; hence $(a_1 - a_2)z$ is divisible by p.

Number theory fact

Fact

Let p be prime, and let $z \not\equiv 0 \mod p$. Then $az \equiv m \mod p$ has at most one solution $0 \leq a < p$.

Proof. The proof is by contradiction.

- Suppose $0 \leq a_1 < p$ and $0 \leq a_2 < p$ are two different solutions.
- Then $(a_1 - a_2)z \equiv 0 \mod p$; hence $(a_1 - a_2)z$ is divisible by p.
- Since $z \not\equiv 0 \mod p$, we know that z is not divisible by p.

Number theory fact

FACT

Let p be prime, and let $z \not\equiv 0 \mod p$. Then $az \equiv m \mod p$ has at most one solution $0 \leq a < p$.

Proof. The proof is by contradiction.

- Suppose $0 \leq a_1 < p$ and $0 \leq a_2 < p$ are two different solutions.
- Then $(a_1 - a_2)z \equiv 0 \mod p$; hence $(a_1 - a_2)z$ is divisible by p.
- Since $z \not\equiv 0 \mod p$, we know that z is not divisible by p.
- It follows that $(a_1 - a_2)$ is divisible by p.
- This implies $a_1 = a_2$.

use the fact that p is prime

Number theory fact

Fact

Let p be prime, and let $z \not\equiv 0 \mod p$. Then $az \equiv m \mod p$ has at most one solution $0 \leq a < p$.

Proof. The proof is by contradiction.

- Suppose $0 \leq a_1 < p$ and $0 \leq a_2 < p$ are two different solutions.
- Then $(a_1 - a_2)z \equiv 0 \mod p$; hence $(a_1 - a_2)z$ is divisible by p.
- Since $z \not\equiv 0 \mod p$, we know that z is not divisible by p.
- It follows that $(a_1 - a_2)$ is divisible by p.
- This implies $a_1 = a_2$.

use the fact that p is prime

Bonus fact. Can replace "at most one" with "exactly one" in above fact.

Universal hashing: summary

Goal. Given a universe U, maintain a subset $S \subseteq U$ so that insert, delete, and lookup are efficient.

Universal hashing: summary

Goal. Given a universe U, maintain a subset $S \subseteq U$ so that insert, delete, and lookup are efficient.

Universal hash function family. $H = \{h_a : a \in A\}$,

$$h_a(x) = \left(\sum_{i=1}^{r} a_i x_i\right) \mod p$$

- Choose p so that $n \leq p \leq 2n$, where $n = |S|$.
- Fact: There exits a prime number between n and $2n$.

Universal hashing: summary

Goal. Given a universe U, maintain a subset $S \subseteq U$ so that insert, delete, and lookup are efficient.

Universal hash function family. $H = \{h_a : a \in A\}$,

$$h_a(x) = \left(\sum_{i=1}^{r} a_i x_i \right) \mod p$$

- Choose p so that $n \leq p \leq 2n$, where $n = |S|$.
- Fact: There exits a prime number between n and $2n$.

Consequence.

- Space used $= \Theta(n)$.
- Expected number of collisions per operation is ≤ 1.

$\Rightarrow O(1)$ time per insert, delete, or lookup

Applications of hashing: finger printing

Problem. Suppose there are two documents X and Y located at two different places, and we want to know if these two documents are the same.

Applications of hashing: finger printing

Problem. Suppose there are two documents X and Y located at two different places, and we want to know if these two documents are the same.

A naive solution. Send two documents to the same place, and make a deterministic comparison.

This method has zero-error, but produces high communication cost.

Applications of hashing: finger printing (cont.)

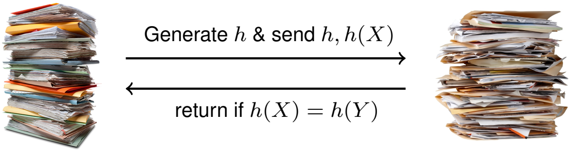
An alternative solution. Use a universal hash function h to map each document to a k-bit string. We only need to send h, and $h(X)$(or $h(Y)$) instead.

Generate h & send $h, h(X)$ $\longrightarrow$

$\longleftarrow$ return if $h(X) = h(Y)$

Applications of hashing: finger printing (cont.)

An alternative solution. Use a universal hash function h to map each document to a k-bit string. We only need to send h, and $h(X)$(or $h(Y)$) instead.



Analysis of the error probability.

$$
\begin{aligned}
\mathbb{P}_{h\in H}[\text{err}] &= \mathbb{P}_{h\in H}[h(X) \neq h(Y) | X = Y] + \mathbb{P}_{h\in H}[h(X) = h(Y) | X \neq Y] \\
&= 0 + \mathbb{P}_{h\in H}[h(X) = h(Y) | X \neq Y] \\
&\leq 1/2^k.
\end{aligned}
$$

Pairwise independent hash functions

PAIRWISE INDEPENDENCE

A family of functions $H = \{h \mid h : U \mapsto [n]\}$ is pairwise independent if, for any h chosen uniformly at random from H, the following holds:

1. $h(x)$ is uniformly distributed in $[n]$ for any $x \in U$;
2. For any $x_1 \neq x_2 \in U$, $h(x_1)$ and $h(x_2)$ are independent.

Pairwise independent hash functions

PAIRWISE INDEPENDENCE

A family of functions $H = \{h \mid h : U \mapsto [n]\}$ is pairwise independent if, for any h chosen uniformly at random from H, the following holds:

1. $h(x)$ is uniformly distributed in $[n]$ for any $x \in U$;
2. For any $x_1 \neq x_2 \in U$, $h(x_1)$ and $h(x_2)$ are independent.

These two conditions state that for any different $x_1 \neq x_2 \in U$, and any $y_1, y_2 \in [n]$, it holds that

$$\mathbb{P}_{h \in \mathcal{H}}\left[h(x_1) = y_1 \wedge h(x_2) = y_2\right] = \frac{1}{n^2},$$

where the probability above is over all random choices of a function from H.

Construction of pairwise independent hash functions

THEOREM

Let p be a prime number, and let $h_{a,b} = (ax+b) \bmod p$. Define

$$H = \{h_{a,b} \mid 0 \leq a, b \leq p-1\}.$$

Then H is a family of pairwise independent hash functions.

Construction of pairwise independent hash functions

Theorem

Let p be a prime number, and let $h_{a,b} = (ax + b) \bmod p$. Define

$$H = \{h_{a,b} \mid 0 \leq a, b \leq p - 1\}.$$

Then H is a family of pairwise independent hash functions.

Recall $\mathbb{Z}_p = \{0, 1, \ldots, p - 1\}$

Proof. We need to show that, for any two $x_1 \neq x_2 \in \mathbb{Z}_p$ and any $y_1, y_2 \in \mathbb{Z}_p$, it holds

$$\mathbb{P}_{h \in H}\left[h(x_1) = y_1 \wedge h(x_2) = y_2\right] = 1/p^2.$$

Construction of pairwise independent hash functions

THEOREM

Let p be a prime number, and let $h_{a,b} = (ax+b) \bmod p$. Define

$$H = \{h_{a,b} \mid 0 \leq a, b \leq p-1\}.$$

Then H is a family of pairwise independent hash functions.

Recall $\mathbb{Z}_p = \{0, 1, \ldots, p-1\}$

Proof. We need to show that, for any two $x_1 \neq x_2 \in \mathbb{Z}_p$ and any $y_1, y_2 \in \mathbb{Z}_p$, it holds

$$\mathbb{P}_{h \in H}\left[h(x_1) = y_1 \wedge h(x_2) = y_2\right] = 1/p^2.$$

For any a, b, the conditions that $h_{a,b}(x_1) = y_1$ and $h_{a,b}(x_2) = y_2$ yield two equations

$$ax_1 + b = y_1 \mod p,$$

$$ax_2 + b = y_2 \mod p.$$

Construction of pairwise independent hash functions

Theorem

Let p be a prime number, and let $h_{a,b} = (ax+b) \bmod p$. Define

$$H = \{h_{a,b} \mid 0 \le a, b \le p-1\}.$$

Then H is a family of pairwise independent hash functions.

Recall $\mathbb{Z}_p = \{0, 1, \ldots, p-1\}$

Proof. We need to show that, for any two $x_1 \neq x_2 \in \mathbb{Z}_p$ and any $y_1, y_2 \in \mathbb{Z}_p$, it holds

$$\mathbb{P}_{h \in H}\left[h(x_1) = y_1 \wedge h(x_2) = y_2\right] = 1/p^2.$$

For any a, b, the conditions that $h_{a,b}(x_1) = y_1$ and $h_{a,b}(x_2) = y_2$ yield two equations

$$ax_1 + b = y_1 \mod p,$$

$$ax_2 + b = y_2 \mod p.$$

Such system has a unique solution of a and b, out of p^2 possible pairs of (a, b). Hence, the equation above holds.

Generalisation: k-wise independence

The set $H = \{h : U \to [n]\}$ is call a set of k-wise independent family of hash functions if for any distinct $x_1, \ldots, x_k \in U$, and any $y_1, \ldots, y_k \in [n]$,

$$\mathbb{P}_{h\in H}\left[h(x_1) = y_1 \wedge h(x_2) = y_2 \wedge \cdots \wedge h(x_k) = y_k\right] = \frac{1}{n^k}$$

Generalisation: k-wise independence

The set $H = \{h : U \to [n]\}$ is call a set of k-wise independent family of hash functions if for any distinct $x_1, \ldots, x_k \in U$, and any $y_1, \ldots, y_k \in [n]$,

$$\mathbb{P}_{h \in H}\left[h(x_1) = y_1 \wedge h(x_2) = y_2 \wedge \cdots \wedge h(x_k) = y_k\right] = \frac{1}{n^k}$$

Construction of k-wise hash functions

Let p be a prime, and $k \geq 2$ be an integer. Assume that a seed $s = (a_0, \ldots, a_{k-1})$ is chosen uniformly at random from $\mathbb{Z}_p^k$. Then, the set of functions $H = \left\{h_s | s \in \mathbb{Z}_p^k\right\}$, where

$$h_s(x) = \sum_{i=0}^{k-1} a_i x^i \mod p$$

is k-wise independent.