Lec30-hashing

Thursday, November 30, 2023 4:33 PM

Let's consider a dictionary. How do we look up words in a dictionary? Binary search works. What about an index in a book? That jumps us straight to the relevant page.

Dictionary ADT (Abstract Data Type)

- add(x): add key x to S
- query(q): is key $q \in S$
- delete(x): remove key x from S

} insertion only dictionary just needs two ops

← static dictionary only needs this
e.g. English language changes slowly.

Static dictionary implementations

- sorted list
- binary tree
- skiplists

} also work for dynamic case
$O(\log n)$ search
↳ can we do better? Want $O(1)$ search.

Hashing basics

- Keys from large universe U. e.g. all strings ≤100 char — Andrew IDs
  e.g. numbers from 1 to 10^{12}. — SSN
  e.g. all integers

- Set $S \subseteq U$ of keys we care about. e.g. 

Notice: we can encode most pertinent information as integers.

Ex. $U = \{1, 2, \ldots, 10^{12}\}$

$N = |S| = 1{,}000{,}000$ (set S much smaller than U)

binary search tree / skiplists take $O(\log N)$ query time

Alt: bitstring of size 10^{12}. Take $O(1)$ query time (assuming random access model)
↳ takes too much space

Idea: hash function $h: U \to \{0, \ldots, M-1\}$, & use array A of size M.

Store $x \in S$ in $A[h(x)]$.

If lucky or if $M \gg N$, no collisions, so $O(1)$ query time.

Collision resolution: Keep linked list in each entry of A. (separate chaining)

Lookup time for x = $O(\text{length}(A[h(x)]))$.

Intuition: If $M = \Theta(N)$, and h spreads S out evenly in A, then length $(A[h(x)])$ is small.

Impossibility result: $\forall$ hash function h, if $|U| \geq (N-1)M + 1$, then $\exists S$ of size $|S| = N$ that all hash to same location.

proof. Pigeon-hole principle. Suppose every location in A has only $N-1$ items from U that hash to it. Then $|U| \leq M(N-1)$, a contradiction. Thus, $\exists S$ with $|S| = N$ that all hash to one location ☐

Randomness to the Rescue

Def. A randomized algorithm H for constructing hash functions $h: U \to \{0, \ldots, M-1\}$ is universal if $\forall x \neq y \in U$,

$$\Pr_{h \in H}[h(x) = h(y)] \leq \frac{1}{M}.$$

Often, we will also say a set H of hash functions is a universal family of hash functions if the algorithm "choose $h \in H$ uniformly at random" is universal.

Important: By definition "any" 2 distinct items must collide with probability at most $\frac{1}{M}$.

Intuition: If $h: U \to [M]$ maps every $x \in U$ uniformly at random, then collision prob. is exactly $\frac{1}{M}$.

Alt interpretation of definition: Count all hash functions in family H. In some, $h(x) = h(y)$ for a specific $x \neq y$.

In some, $h(x) = h(y)$ for a specific $x \neq y$.

Then we need $\forall x \neq y \in U$, $\dfrac{|\{h \in H \mid h(x) = h(y)\}|}{|H|} \leq \frac{1}{M}$.

Example: $U = \{a, b\}$ $M = 2$

$h : \{a, b\} \to \{0, 1\}$

✓

	a	b
h_1	0	0
h_2	0	1

$a \neq b$
$h_1(a) = h_1(b)$
$h_2(a) \neq h_2(b)$
} collide half the time $\leq \frac{1}{M}$

✓

	a	b
h_1	0	1
h_2	1	0

$h_1(a) \neq h_1(b)$
$h_2(a) \neq h_2(b)$
} don't collide

✓

	a	b
h_1	0	0
h_2	1	0
h_3	0	1

$h_1(a) = h_1(b)$
$h_2(a) \neq h_2(b)$
$h_3(a) \neq h_3(b)$
} collide $\frac{1}{3}$ of time $\leq \frac{1}{M}$.

✗

	a	b
h_1	0	0
h_2	1	1

$h_1(a) = h_1(b)$
$h_2(a) = h_2(b)$
} collide $1 \geq \frac{1}{2}$ of time

Ex. $U = \{a, b, c\}$ $M = 2$

✗

	a	b	c
h_1	0	0	1
h_2	1	1	0
h_3	1	0	1

collisions: prob

$a \neq b$	$a \neq c$	$b \neq c$
collide	diff	diff
collide	diff	diff
diff	diff	collide
$\frac{2}{3}$	$\frac{0}{3}$	$\frac{1}{3}$

↳ not universal because of single pair (a, b).

Ex. $U = \{a, b, c\}$ $M = 3$

	a	b	c
h_0	0	0	0

$a \neq b$	$a \neq c$	$b \neq c$
collide	collide	collide

] Notice: some hash functions can

	a	b	c
h_0	0	0	0
h_1	0	1	2
h_2	1	2	0
h_3	2	0	1

collision $p \to b$

a + b	a + c	b + c
collide	collide	collide
diff	diff	diff
diff	diff	diff
diff	diff	diff
$\frac{1}{4}$	$\frac{1}{4}$	$\frac{1}{4}$

$\leq \frac{1}{3}$

Notice: some hash functions can be super bad while still being universal

$\Rightarrow$ universal

Thm If H is universal, then $\forall S \subseteq U$ of size N, $\forall x \in U$, if we construct h at random according to H ($h \in H$ random), the expected number of collisions between x & other elements in S is $\leq \frac{N}{M}$.

proof. By universality, $\Pr\left(h(x) = h(y) \,|\, y \neq x\right) \leq \frac{1}{M}$.

Let $C_{xy} = \begin{cases} 1 & \text{if } h(x) = h(y) \\ 0 & \text{otherwise} \end{cases}$

Let C_x be r.v. denoting total # collisions for x.

$$C_x = \sum_{\substack{y \in S \\ y \neq x}} C_{xy}$$

$$\mathbb{E}[C_x] = \sum_y \mathbb{E}[C_{xy}] \leq \frac{N}{M}.$$

↑ linearity of expectation

Thus, the expected lookup time is $O\left(\frac{N}{M}\right)$.

If, e.g. $M = 2N$, then lookup is $O(1)$. (assuming h takes $O(1)$ to compute)

Constructing universal hash function

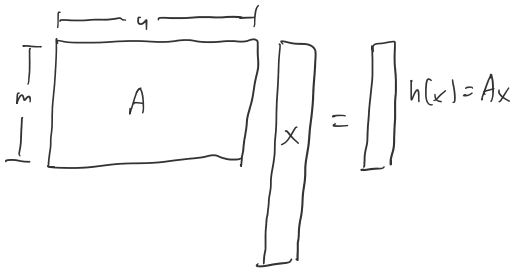
Matrix method:

Let keys be u-bits long. ($U = \{0,1\}^u \cong \{1, \ldots, 2^u\}$)

Let $M = 2^m$, so indices are m-bits long.

Let $A \in \{0,1\}^{m \times u}$ matrix where each entry is i.i.d. uniform.

Then let $h(x) = Ax$ where all additions are mod-2.



Claim: For $x \neq y$, $\Pr_h[h(x) = h(y)] = \frac{1}{M}$.

proof: Some bit $x_i \neq y_i$ since $x \neq y$. WLOG, say $x_i = 0$, $y_i = 1$.

Let's imagine setting all of A except col i.

Consider row j of A, and entries $[Ax]_j$, $[Ay]_j$.

Flipping A_{ji} will keep $[Ax]_j$ the same, while flipping $[Ay]_j$.

So some setting of A_{ji} causes $[Ax]_j = [Ay]_j$ and some setting $[Ax]_j \neq [Ay]_j$.

$\Rightarrow$ then every entry has $\frac{1}{2}$ prob to be equal.

$\Rightarrow$ $\Pr[Ax = Ay] = \left(\frac{1}{2}\right)^m$

Some other methods involve finite field arithmetic for other primes.

We now can get $O(1)$ expected runtime for hash table.

Can we do better if S is static?

Perfect Hashing

Method 1: (quadratic space)

Theorem: If H is universal and $M = N^2$, then $\Pr_{h \in H}(\text{no collisions in } S) \geq \frac{1}{2}$.

proof: $\binom{N}{2}$ pairs (x,y) to check.

proof. $\binom{N}{2}$ pairs (x,y) to check.

$\Pr(h(x)=h(y)) \le \frac{1}{M}$ by universality.

$\Rightarrow \Pr(\text{any collision}) \le \frac{\binom{N}{2}}{M} \le \frac{1}{2}$ (union bound) ☐

Recall birthday problem. If 22 people in room, $\frac{1}{2}$ chance of shared birthday, since only 365 days, and $\binom{22}{2}$ pairs. Here we just ensure lots of open slots.

$\Rightarrow$ Using quadratic space, just choose hash functions until we get a perfect one, as each hash function has at least $\frac{1}{2}$ chance of being perfect.

Method 2 (linear space) (2-level scheme)

Idea: First use universal hash function to $M = N$ size array.

Let L_i be the load in array bin i.

Then use 2nd-level universal hash function on each bin mapping to array of size L_i^2. (Method 1)

Thm. If $h \in H$ is universal, $\Pr\left[\sum_i (L_i)^2 > 4N\right] < \frac{1}{2}$.

proof. $\mathbb{E}\left[\sum_i (L_i)^2\right] = \mathbb{E}\left[\sum_x \sum_y C_{xy}\right]$ $\quad C_{xy} = \begin{cases} 1 & \text{if } h(x)=h(y) \\ 0 & \text{else} \end{cases}$ (even if $x=y$)

If $L_i = 3$ with elements (a,b,c), then we get collisions $\begin{pmatrix} aa & ba & ca \\ ab & bb & cb \\ ac & bc & cc \end{pmatrix} = 9$

$$= N + \sum_x \sum_{y \neq x} \mathbb{E}[C_{xy}]$$

$$\le N + \frac{N(N-1)}{M} < 2N \quad (\text{since } N=M).$$

By Markov's inequality, $\Pr\left(\sum_i (L_i)^2 > 4N\right) < \frac{1}{2}$. ☐

Thus, we just try hCH repeatedly to get one with few enough collisions that squaring the bin loads is still linear.