5112

11-14

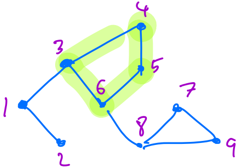
Hash Tables

What is a hash table?

A dictionary that uses a hash function to determine where to store items.

a data structure supporting
- Insert
- Query
- (Delete)
- (Scan) ← not supported by hash tables

Hash tables generalize direct mapping



In for example a DFS, can use a direct mapping to tell which nodes have been visited.

1	2	3	4	5	6	7	8	9
		x	x	x	x			

What is a hash function?

Have a universe $\mathcal{U}$ of keys.

e.g. integers between 0 and 2^{32}

strings of byte

points in $\mathbb{R}^3$.

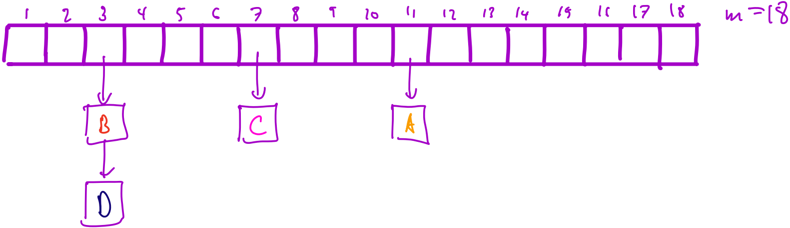
A hash function is a <u>randomized</u> function

$h: \mathcal{U} \longrightarrow \{0, \dots, m-1\}$

for some m.

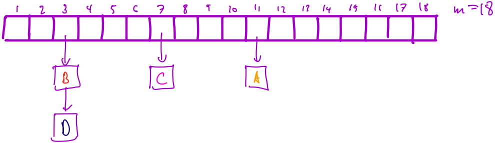
Chaining Hash Table

Hash each item to a bucket and store each bucket as a linked list.



$h(A) = 11 \quad h(B) = 3 \quad h(C) = 7 \quad h(D) = 3$

Hash each item to a bucket and store each bucket as a linked list.



What properties do we want from our hash function?

Could I just use random numbers?

Suppose I insert B, then query B.

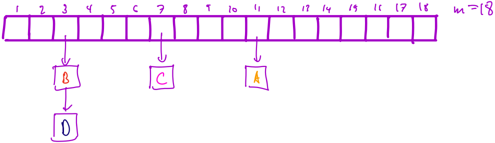
No! Need the same value each time.

Could I use a deterministic function?

Interesting question.

This is hard to analyze.

Hash each item to a bucket and store each bucket as a linked list.



What properties do we want from our hash function?

Expected cost of a query is proportional to the expected list length.

How can the hash function minimize this?

By distributing items to different buckets.

Totally Random Hash Functions

Choose h uniformly from the set of all functions from $\mathcal{U} \to \{0, \ldots, m-1\}$.

Equivalent to picking a random number from $\{0, \ldots, m-1\}$ for each $x \in \mathcal{U}$.

To store h, how many bits do I need?

$\log m$ bits for each $x \in \mathcal{U}$, so that $|\mathcal{U}| \log m$ bits.

way too big

Also need some way to efficiently compute.

Universal Hash Functions

Idea: pick h from a smaller set of potential hash functions.

A family of hash functions is called universal if for all $x \neq y \in \mathcal{U}$,

$$\Pr\{h(x) = h(y)\} \leq \frac{1}{m}$$

Universal Hash Functions

Example: $h_{a,b}(x) = \{(ax+b) \bmod p\} \bmod m.$

p is a fixed prime with $p > |\mathcal{U}|$,
$0 \le a, b < p$, with $a \neq 0$.

Number theory $\Rightarrow$ universal.
How to encode which hash function we're using:
Just need to provide p, a, b.

Universal Hash Functions

Example: Multiply-Shift.

Assume $m = 2^k$, a odd number $0 < a < 2^w$

$$h_a(x) = (ax \bmod 2^w) / 2^{w-k}$$

Multiplying x by a, then truncating to a word
Then shift right to get a k-bit result.
Not universal, but almost universal

k-wise independent hash functions

A family is k-wise independent if for all $x_1, \dots, x_k \in \mathcal{U}$, $t_1, \dots, t_k \in \{0, \dots, m-1\}$

$$\Pr_h\left\{h(x_1) = t_1 \wedge h(x_2) = t_2 \wedge \cdots \wedge h(x_k) = t_k\right\} = O\left(\frac{1}{m^k}\right)$$

Stronger condition than universal.

Lots of constructions

Hash Functions in Practice

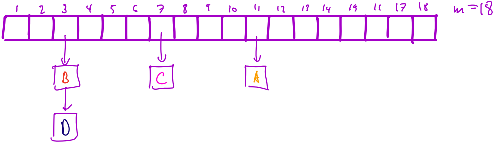
Basically none of the theory matters.

Murmur Hash: core loop that does a multiply and a rotation.

determined by a seed.

Note: These hash functions are not cryptographic.
Crypto hash functions are hard to invert.
" " " have extremely few collisions
(have > 256 bits of output)

Hash each item to a bucket and store each bucket as a linked list.



Expected cost of a query is proportional to the expected list length.

Suppose there are n items in the hash table.

Let C_t be the number of items hashing to t.

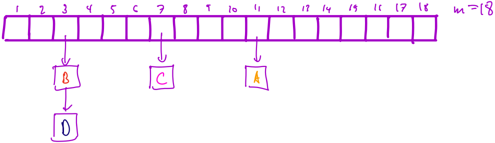
$$E\{\text{cost of a query}\} = E_h\{C_t\}$$

$$= \sum_i \Pr\{h(x_i) = t\}$$

$$= O\left(\frac{n}{m}\right) \quad (h \text{ is universal})$$

$$= O(1) \text{ if } m = \Omega(n)$$

Hash each item to a bucket and store each bucket as a linked list.



What about the worst case?

All items in the same bucket

$\Rightarrow \Theta(n)$.

Very unlikely!

If h is totally random, and $m = \Theta(n)$, then

$$C_t = O\left(\frac{\log n}{\log \log n}\right)$$

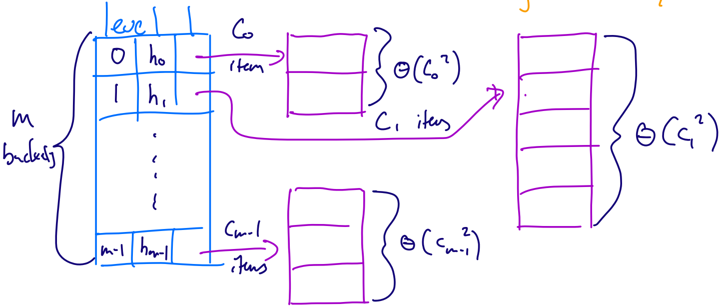
with high probability.

$1 - \frac{1}{n^c}$ for any c.

FKS Perfect Hashing

Static: Given all n items up front

In contrast to inserting them one by one.



How large does a hash table need to be to have no collisions?

$$E\{\#\text{ of collisions}\} = \sum_{i<j} \Pr\{h(x_i) = h(x_j)\}$$

$$= \frac{1}{m}\ \#\text{ of distinct pairs}$$

$$= \frac{1}{m}\frac{n(n-1)}{2} = \Theta\left(\frac{n^2}{m}\right)$$

If we choose m large enough that this expectation is $\approx \frac{1}{2}$, then can use Markov's inequality to say $\Pr\{\text{no collisions}\} \leq \frac{1}{2}$.

Markov's Inequality

If X is a non-negative R.V., and $a > 0$, then

$$\Pr\{X \geq a\} \leq \frac{E\{X\}}{a}.$$

$$\Pr\{\#\text{ of collisions} \geq 1\} \leq E\{\#\text{ of collisions}\} \leq \frac{1}{2}.$$