Hashing

- A very efficient implementation for dictionaries, has applications in cryptography, data streaming, complexity theory, etc.
- U: a very large universe of keys $|U|=m$
- We like to maintain a set $S \subset U$ where $|S| \ll |U|$
- Support the following operations:
  - MakeSet(S) : create a hash table of S
  - Insert(i,S) : add item i to S
  - Delete(i,S) : remove item i from S
  - Find(i,S) : find item i in S
- Using balanced BST oper's can be done in $O(\log n)$
- let's assume $U = \{1, \ldots, M\}$ and we can do arithmetic opr's in $O(1)$ time.
- We have a table $T[1..n]$ and a hash function $h: U \longrightarrow [n]$ where for each $x \in U$ we store x at index $h(x)$ of the table.

Collisions: we have a collision if for two different keys $x, y \in S$: $h(x) = h(y)$

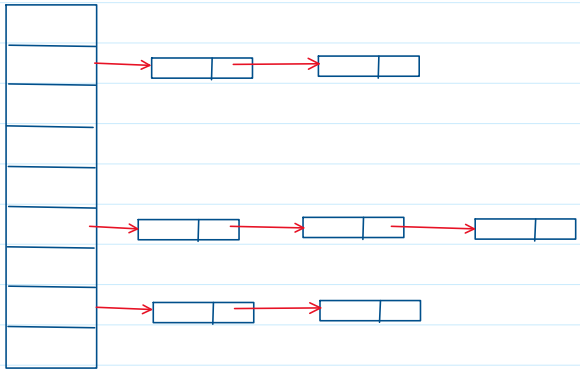
Perfect hash function: A hash function $h: U \longrightarrow [n]$ is perfect for set $S \subset U$ if it is collision-free.

$(\forall x, y \in S,\ x \neq y \longrightarrow h(x) \neq h(y))$

- For any set S there is a perfect hash h

- For any set S there is a perfect hash h but there is no h that is perfect for each S if $n < m$. (Exercise!)

- Completely Random hash: if $|S| = n$ and hash function is uniformly random (Balls & Bins):
  $\Pr[\text{collision}] = \frac{1}{n}$ and the maximum number of collisions per bucket $\Theta\left(\frac{\log n}{\log \log n}\right)$
  To manage collisions, one idea is Chaining



universal hash functions: A family H of hash functions $h: U \longrightarrow T$ is universal if for all $x, y \in U$, with $x \neq y$ and for h chosen u.r. from H:

$$\Pr[h(x) = h(y)] \leq \frac{1}{n}$$

- This is the same probability a truly random hash function will give.

- In fact most constructions of universal hash func's imply a stronger property called 2-universal hashing.

Definition: let H be a family of hash functions

$h: U \to T$. We say H is 2-universal (or 2-wise) if for all $x, y \in U$, $x \neq y$ and all $n_1, n_2 \in \{1, \dots, n\}$

$$\Pr[h(x) = n_1 \wedge h(y) = n_2] = \frac{1}{n^2}$$

- Question: Can we get better family of hash func's? i.e. with smaller probability of collision?

Not by much.

Lemma: For any family of hash functions $h: U \to T$

$$\exists x, y \in U \text{ s.t. } \Pr_{h \in H}[h(x) = h(y)] \geq \frac{1}{n} - \frac{1}{M}$$

- So when $M \gg n$ we cannot do much better.
- Why 2-universal hash func's are good?
- 2-universal is also called pair-wise independent. This can be generalized to k-wise independent.

Definition: A family of hash functions H mapping U to $[n]$ is called k-wise independent if for any k distinct keys $x_1 \dots x_k \in U$ and any k values $a_1 \dots a_k \in [n]$ we have

$$\Pr_{h \in H}[h(x_1) = a_1 \wedge \dots \wedge h(x_k) = a_k] \leq \frac{1}{n^k}$$

Theorem: Consider any sequence of opr's with at most s inserts using a hash function h drawn u.r. from a universal family H. Then the expected cost of each operation is $\leq 1 + \frac{s}{n}$

Proof: Consider inserting $x \in U$. If z is the number of elements in $h(x)$ then the cost of this opr

is $1+Z$. We bound $\mathbb{E}[Z]$

For each yes let $Z_y = 1$ iff $y \in h(x)$.

$$\mathbb{E}[Z] = \sum_{\text{yes}} \mathbb{E}[Z_y] \leq s \cdot \Pr[h(x) = h(y)] \leq \frac{s}{n}.$$

Construction of universal family

- For simplicity, suppose $|U| = 2^u$ and $n = 2^m$
- Take a $u \times m$ matrix A whose entries are 0/1 chosen u.r.
- For each $x \in U$ consider x as a u-bit vector and let $h(x) = A \cdot x \bmod 2$
- So there are $2^{m \cdot u}$ hash functions in this family

Theorem: This is a universal hash family.

Proof: $\forall x, y \in U$, with $x \neq y$, we need to show $\Pr[h(x) = h(y)] \leq \frac{1}{n}$

- Note $h(x) = h(y) \iff h(x-y) = \vec{0}$.
- So need to show $\forall z \in \{0,1\}^u$

$$\Pr_{h \in H}[h(z) = \vec{0}] = \Pr[Az = \vec{0}] \leq \frac{1}{2^m} = \frac{1}{n}$$

- Since $z \neq \vec{0}$, there is some entry of z that is non-zero, say $z_{i^*} = 1$
- if $A_1 \dots A_u$ are columns of A then $A \cdot z = \sum_{1 \leq i \leq u} z_i \cdot A_i$. For Az be zero we need to have $A_{i^*} = \sum_{i \neq i^*} z_i \cdot A_i$.
- But entries of A_{i^*} are all random and each bit matches the entry on the right with prob. $\frac{1}{2}$.

– [illegible] bit matches the entry on the right with prob. $\frac{1}{2}$

– So the probability of $A \cdot z = \vec{0}$ is $\frac{1}{2^m} = \frac{1}{n}$.

A second construction of universal hash

– let p be a prime $m \le p \le 2m$ → universe and $\mathbb{Z}_p = \{0, 1, \ldots, p-1\}$

– $\forall a, b \in \mathbb{Z}_p$ with $a \neq 0$ define $f_{ab} : U \longrightarrow \mathbb{Z}_p$ as:

$f_{ab}(x) = ax + b \mod p$

let $h_{ab}(x) = (f_{ab}(x)) \mod n$

– The hash family is : $H = \{ h_{ab} \mid a, b \in \mathbb{Z}_p, a \neq 0 \}$

– Note that family H has only $p(p-1)$ functions and each function needs $O(\log M)$ bits to represent

Theorem: H is universal.

Proof: For any $x, y \in U$ we need to show:

$$\Pr[h(x) = h(y)] \le \frac{1}{n}$$

– Note $f_{ab}(x) = (ax + b) \mod p$

and $h_{ab}(x) = f_{ab}(x) \mod n$

– if $h_{ab}(x) = h_{ab}(y) \longrightarrow f_{ab}(x) = f_{ab}(y) \mod n$

Claim: $\forall r, s \in \mathbb{Z}_p$:

$$\Pr[f_{ab}(x) = r \wedge f_{ab}(y) = s] = \begin{cases} 0 & \text{if } r = s \\ \frac{1}{p(p-1)} & \text{o.w} \end{cases}$$

Proof: if $f_{ab}(x) = r$ and $f_{ab}(y) = s$ then

$$\begin{cases} ax + b \mod p = r \\ ay + b \mod p = s \end{cases}$$

- this system (with a,b unknows) has a unique solution in $\mathbb{Z}_p$ for a,b : $a = \frac{r-s}{x-y}$
- Since $x \neq y$, a is non-zero iff $r \neq s$
- Thus there is exactly 1 out of $p(p-1)$ functions that gives $f_{ab}(x) = r$ and $f_{ab}(y) = s$.
- We have $h_{ab}(x) = h_{ab}(y)$ iff $r \equiv s \mod n$

$\longrightarrow \Pr[h_{ab}(x) = h_{ab}(y)] = \frac{1}{p(p-1)} \times |\{(r,s) \mid r \neq s \text{ and } r \equiv s \mod n\}|$

For each choice of r there are $\lceil \frac{p}{n} \rceil - 1$ other values for s s.t. $r \equiv s \mod n$

$\longrightarrow$ size of this set $\leq p\left(\lceil \frac{p}{n} \rceil - 1\right) \leq \frac{p(p-1)}{n}$

$\longrightarrow \Pr[h_{ab}(x) = h_{ab}(y)] \leq \frac{1}{p(p-1)} \cdot \frac{p(p-1)}{n} = \frac{1}{n}$.

Perfect hashing

Let us consider static dictionaries: S is fixed and we are only concerned about find operations.

Definition : A family H of hash functions is called perfect if for every subset $S \subseteq U$ there is a hash function $h \in H$ s.t. h is perfect for S.

- suppose $|S| = N$. We show a two level hashing that gives perfect hashing with $O(N)$ space and $O(1)$ look-up time and uses universal hashing.
- Suppose we use universal hashing for a set S of size N into a table of size $n = O(N)$

Claim: with probability $\geq \frac{1}{2}$ chain sizes are $O(\sqrt{n})$

proof: $\forall\ x,y \in S$ let $C_{xy} = \begin{cases} 1 & h(x)=h(y) \\ 0 & \text{o.w.} \end{cases}$

let $C = \sum_{x \neq y \in S} C_{xy}$

$$\mathbb{E}[C] = \mathbb{E}\left[\sum_{x \neq y} C_{xy}\right] = \sum_{x \neq y} \mathbb{E}[C_{xy}] \leq \binom{N}{2}\frac{1}{n} \quad \circledast$$

for $n=N$ this is $\sim \frac{N}{2}$. So by Markov's $\Pr[C \geq N] \leq \frac{1}{2}$.

if a chain is $\geq \sqrt{2N}$ then there are $\binom{\sqrt{2N}}{2} \geq N$ collisions. So with probability $\geq \frac{1}{2}$ max chain size is $O(\sqrt{N})$.

- In fact if L_i = length of chain at i then total # of collisions: $\sum_i \binom{L_i}{2}$

  So with probability $\geq \frac{1}{2}$: $\sum_i L_i^2 \leq 3N$.

- Call this first level hash function $h^*: U \longrightarrow [N]$

- For all L_i keys mapped into location i we build a table of size $M_i = L_i^2$ and use a second level universal hash function h_i^* to map the L_i keys into $[M_i]$ $\quad h_i^*: [L_i] \longrightarrow [M_i]$

- Using $\circledast$ with $n = M_i = L_i^2$, all keys are mapped to separate locations.

- To look up q: check location $i = h^*(q)$ then check location $h_i^*(q)$; two hash evaluations

Total space: N for h^*

$\sum_i O(L_i^2) = O(N)$ for second level

Summary: perfect hashing for a static set $S \subseteq U$ with

Summary: perfect hashing for a static set $S \subseteq U$ with $|S| = N$

1) pick a random h from a universal hash family $H: U \to N$ while the total # of collisions by h is $> 3N$, resample h
2) if $L_i = \#\{x \in S \mid h(x) = i\}$, we know $\sum \binom{L_i}{2} \leq 3N$
3) For each $i \in [N]$, choose a table of size $4L_i^2$ and pick a second-level hash h_i from a universal family $H: U \to [4L_i^2]$ map L_i keys in location i using h_i; prob of a collision is $\leq \frac{1}{2}$; if collision then resample h_i.
4) For a query q, first look at $h(q)$ of the first hash then location $h_{h(q)}(q)$ of the second hash.

Bloom Filters

- Hash tables are efficient data structures to find an item given a key
- what if most searches are for items that don't belong to the table?
- you may be ok with a data structure that is more efficient but has higher chance of false positive: if it says "no" then definitely the item is not present; but may say (falsely) yes for some that are not present. (this is used as a filter) we may use a secondary data structure
- if we have items from a universe $|U| = m$ and we

have N items we need $O(N \log m)$ bits to store.

- What if we hash each item to a bit?

Problem: if we have collision we have false positive

Here is how the hashing would work:

- to map items from $S \subseteq U$, we have an n-bit vector (hash table), all initially zero.

we take k (random) hash functions $h_1 \cdots h_k : U \to [n]$

given any element $x \in S$ set all $h_i(x)$ locations to 1

- To look-up: answer yes if all k bits of $h_i(x)$ are 1.

- Clearly, if we say "No" then x is not in the table. But could have false positive (i.e. all one 1 but item was never added)

- note that if we have added N items they set kN random bits to 1.

so Prob. that a bit is zero: $\left(1 - \frac{1}{n}\right)^{kN} \approx e^{-kN/n}$ → Say p

$\longrightarrow$ expected # of zero bits $\approx p \cdot n$

Probability of false positive $(1-p)^k = f$

- How to optimize f? what k?

$f = e^g \longrightarrow g = k \ln\left(1 - e^{-kN/n}\right)$

$dg = 0 \ldots \left(1 - e^{-kN/n}\right) + \frac{kN}{\quad} e^{-kN/n}$

$$\frac{dg}{dk} = \ln\left(1 - e^{-kN/n}\right) + \frac{kN}{n} \cdot \frac{e^{-kN/n}}{1 - e^{-kN/n}}$$

zero at $k = \ln 2 \left(\frac{n}{N}\right)$

this gives false positive probability of $0.6185^{n/N}$

if $n = 2N$ then false positive rate $\approx 38\%$ and with $n = 8N$ this drops to 2%

if the false positive error is set to ε then the # of bits we use is $n \approx 1.44 N \log\left(\frac{1}{\varepsilon}\right)$ which is about $1.44 \log \frac{1}{\varepsilon}$ per entry

(remember that previously it was $\log m$)