Hashing of strings (vectors)

- scalar product $h_t(x) = x \cdot t = \sum_{i=0}^{d-1} x_i \cdot t_i \pmod p$, $h_t : \mathbb{Z}_p^d \to \mathbb{Z}_p$
  $(x_0, \dots, x_{d-1}) \in \mathbb{Z}_p^d$, p prime, $d \geq 1$, $t \in \mathbb{Z}_p^d$
- $\mathcal{G} = \{ h_t \mid t \in \mathbb{Z}_p^d \}$ is 1-universal for any $d \geq 1$, any prime p. (lecture 7)

Note: Composing a universal family with mod m may lose universality. ← target # buckets

Q: How can $\mathcal{G}$ "gain" 2-independence? ← (stronger than universality)

- linear congruence $h_{a,b}(x) = ((ax+b) \bmod p) \bmod m$
  $h_{a,b} : \mathbb{Z}_p \to [m]$, $a,b \in \mathbb{Z}_p$, $p \geq m$, p prime
- $\mathcal{L} = \{ h_{a,b} \mid a,b \in \mathbb{Z}_p \}$ is $(2,4)$-independent for any $p \geq m$, and $(2,2)$-independent if $p \geq 4m$ where p is prime. (lecture 8)

Lem: Let $\mathcal{F}$ be c-universal family of $f : U \to [r]$, let $\mathcal{G}$ be $(2,d)$-independent family of $g : [r] \to [m]$. Then the family $\mathcal{H} = \mathcal{F} \circ \mathcal{G} = \{ f \circ g \mid f \in \mathcal{F}, g \in \mathcal{G} \}$ is $(2, c')$-independent where $c' = \left(\frac{cm}{r} + 1\right) d$.

general composition

$(f \circ g)(x) = g(f(x))$

Pf: $x_1, x_2 \in U$ distinct, $i_1, i_2 \in [m]$ (not a collision)

careful: $(2,d)$-ind. of $\mathcal{G}$ applies only if $f(x_1) \neq f(x_2)$!

"match event" M: $g(f(x_1)) = i_1 \wedge g(f(x_2)) = i_2$

"collision event" C: $f(x_1) = f(x_2)$ ← 0 if $i_1 \neq i_2$

$$\Pr[M] = \underbrace{\Pr[M \mid \neg C]}_{(2,d)\text{-ind.}} \cdot \underbrace{\Pr[\neg C]}_{\text{triv.} \leq 1} + \underbrace{\Pr[M \mid C]}_{(1,d)\text{-ind.}} \cdot \underbrace{\Pr[C]}_{c\text{-univ.}} \leq \frac{d}{m^2} \cdot 1 + \frac{d}{m} \cdot \frac{c}{r} = \frac{c'}{m^2}$$

Note: if $\Pr[C] = 0$, then $\Pr[M] \leq \frac{d}{m^2}$ ($\Pr[C] = 1$ doesn't happen). □

$\Rightarrow$ $\mathcal{G} \circ \mathcal{L}$ is $(2,8)$-ind. for any $p \geq m$, and $(2, 5/2)$-ind. if $p \geq 4m$.

Rolling hashing (another option for hashing of strings)

$h_a(x) = \sum_{i=0}^{d-1} x_i \cdot a^i \pmod p$, $h_a : \mathbb{Z}_p^d \to \mathbb{Z}_p$, $a \in \mathbb{Z}_p$, $d \geq 1$, p prime

($x = (x_0, \dots, x_{d-1})$)

evaluating a polynomial with coefficients x_i at a, or scalar product with $(a^0, a^1, \dots, a^{d-1})$

Thm: $R = \{h_a \mid a \in \mathbb{Z}_p\}$ is **d-universal** for any $d \geq 1$, p prime.

Pf: $x, y \in \mathbb{Z}_p^d$ distinct

$$\Pr_a[h_a(x) = h_a(y)] = \Pr_a\left[\sum_{i=0}^{d-1} (x_i - y_i) a^i = 0\right] \leq \frac{d}{p}$$

(a is a root of a (nonzero) polynomial of degree at most $d-1$; there are at most $d-1$ roots)

by Lem.

$\Rightarrow$ **$R \circ \mathcal{L}$ is $(2, \delta/2)$-independent if $p \geq 4d/\delta$.**

R versus $\mathcal{G}$: d-univ. / 1-univ., single parameter / d parameters, rolling

Rolling

motivation: Robin-Karp substring search – compare hashes in $O(1)$ time instead of entire substrings

$\dots, x_0, x_1, \dots, x_{d-1}, x_d, \dots$ input file (string)

search window: x, x' — x substring of length d, x' next

$$h_a(x) = x_0 + x_1 a + \dots + x_{d-1} a^{d-1}$$
$$h_a(x') = x_1 + \dots + x_{d-1} a^{d-2} + x_d a^{d-1} \pmod p$$

$$h_a(x') = \frac{h_a(x) - x_0}{a} + x_d a^{d-1}$$

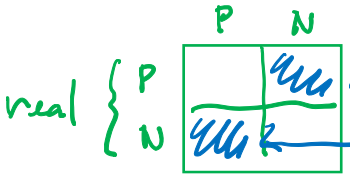
if a^{d-1} precomputed in $O(1)$ time

Remarks:

- alphabet size $k \leq p$ (prime), if $k \ll p$ (e.g. for $p \geq 4d/\delta$) we may encode multiple characters into one: $\boxed{x_0 \mid x_1 \mid x_2}$ $\quad x_0 + k x_1 + k^2 x_2 < p$
- if strings have variable lengths: use padding with a blank char to a fixed (maximal) length (encode by 0)

Bloom filters (application of hashing)

filter: (fast and efficient) test with a one-sided error (ideally)



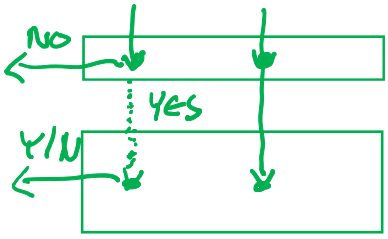
"sensitivity" = 100%

"specificity high"

⇒ such tests filters out negatives

Bloom filter (approximative representation of sets with 0 false negatives)

Find(x) Insert(x) (Delete problematic)

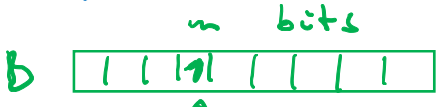


⇒ answers NO = definitely not in the set

YES = possibly in the set

aim: memory efficient and small probability of false positives (trade-off)

Simple B. filter



$h: U \to [m]$ uniformly in random from a c-universal family

INSERT(x): $B[h(x)] := 1$

Find(x) = YES $\Leftrightarrow$ $B[h(x)] = 1$

⇒ supports (only) Find, Insert, has no false negatives

- What is the probability of false positives?

$x_1, \dots, x_n$ already inserted, y distinct (true negative)

$$\Pr_h[y \text{ is FP}] = \Pr[\exists i\ h(y) = h(x_i)] \leq \sum_{i=1}^{n} \Pr[h(y) = h(x_i)] \leq \frac{cn}{m}$$

union bound; c-univ.

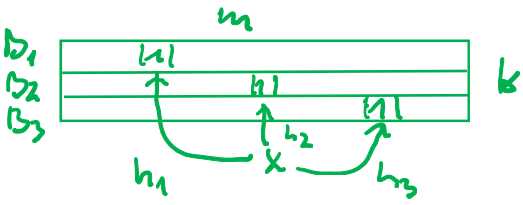
⇒ for $\Pr[\text{FP}] \leq \varepsilon$ (given target) we need $m = \lceil n/\varepsilon \rceil$ bits (say $c=1$)

ex: for $n = 10^6$, $\varepsilon = 0.01$ we have $m = 100$ Mb.

linearly dep. on $1/\varepsilon$

multi-band filter (conjunction of simple filters)

k "bands" per m bits. $h_i : U \to [m]$ chosen independently ($i = 1..k$) from a c-universal family (say $c=1$)



INSERT(x): $B_i[h_i(x)] := 1 \quad \forall i = 1..k$

FIND(x) = YES $\iff B_i[h_i(x)] = 1 \ \forall i$

$\Rightarrow$ $O(k)$ time, no false negatives

Probability of false positives: $x_1, \dots, x_n$ inserted, y distinct (true negative)

$$\Pr[FP] = \Pr[\forall i \exists j \ h_i(j) = h_i(x_j)] = \prod_{i=1}^{k} \Pr[\exists j \ h_i(j) = h_i(x_j)] \le \prod_{i=1}^{k} \frac{cn}{m} = \frac{1}{2^k}$$

$h_1, \dots, h_k$ independent

if $m = 2n$

$\Rightarrow$ for $\Pr[FP] \le \varepsilon$ we can take $m = 2n$, $k = \lceil \log(1/\varepsilon) \rceil$,

uses $M = 2n \lceil \log 1/\varepsilon \rceil$ bits (logarithmically in $1/\varepsilon$)

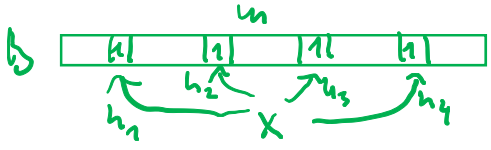
ex: for $n = 10^6$, $\varepsilon = 0.01$ we have $k = 7 \Rightarrow M = 14$ Mb

$\varepsilon = 0.001$ $k = 10$ 20 Mb

Note: It is known that any DS with no false positives and $\Pr[FP] \le \varepsilon$ needs at least $n \log(1/\varepsilon)$ bits. (optimal up to factor 2)

Furthermore, if h_i are totally random (and independent) then taking $m = n/\ln 2 \doteq 1.44\, n$, $k = \lceil \log(1/\varepsilon) \rceil$ also works.

single-band filter (compressed multiband)



$h_i : U \to [m]$ chosen independently ($i = 1..k$)

(assume they are totally random)

INSERT(x): $B[h_i(x)] := 1 \quad \forall i = 1..k$

FIND(x) = YES $\iff B[h_i(x)] = 1 \ \forall i$

$\Rightarrow$ $O(k)$ time, no false negatives

$x_1, \dots, x_n$ inserted $\Rightarrow$ at most kn bits set to 1

$p = \Pr[B[i]=0] = (1-1/m)^{nk} \approx e^{-nk/m} \Rightarrow k \doteq -m/n \cdot \ln p$

$B[i]$: fixed bit; $h_1, \dots, h_k$: totally random independent hashes; $1+x \approx e^x$ (Taylor's expansion)

y distinct from $x_1, \dots, x_n$ (true negative)

$\Pr[FP] = \Pr[\forall i\ B[h_i(y)]=1] \approx (1-p)^k = (1-p)^{-m/n \ln p} = e^{-m/n \ln p \ln(1-p)}$

("independent")

minimized for $p \approx 1/2$ (looks like random bits)

$\Rightarrow$ for $\Pr[FP] \le \varepsilon$ set $k = \lceil \log(1/\varepsilon) \rceil$

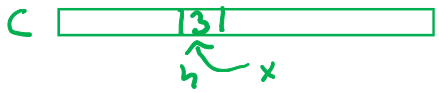
and $m = n/\ln 2 \cdot k \doteq 1.44\, n \lceil \log(1/\varepsilon) \rceil$

Counting filters (to implement Delete)

idea: since shared bits cannot be deleted, replace bits with counters

for simple filter: INSERT / DELETE (x): increment/decrement $C[h(x)]$

FIND(x) = YES $\Leftrightarrow$ $C[h(x)] > 0$



problem: counters are limited, b bits $\Rightarrow$ $2^b - 1 =: t$ max value

if $C[i] = t$, the counter is "stuck" (no inc/dec anymore)

$\Rightarrow$ new type of false positives!

However, the probability of having some counter stuck is small

(assuming a totally random hashing function for simple filter):

$$\Pr_h[C[i] \ge t] \le \binom{n}{t}\left(\frac{1}{m}\right)^t \le \left(\frac{ne}{mt}\right)^t = \left(\frac{e \ln 2}{t}\right)^t \quad \text{for } m = n/\ln 2 \approx 1.44\, n.$$

if we had unlimited; there is a t-tuple hashed to i, rest anywhere; using $\binom{n}{t} \le \left(\frac{ne}{t}\right)^t$

ex: $b=4 \Rightarrow t=15$, $\Pr(C[i] \ge t) \le 3.06 \cdot 10^{-14}$ "4-bits suffice"

$m = 10^9 \Rightarrow \Pr[\exists i\ \Pr(C[i] \ge t)] \le 3.06 \cdot 10^{-5}$

Note: if stuck counters accumulated $\Rightarrow$ rebuild