Bloom filters

Recall: ($|X| = n$, $X \subseteq \mathcal{U}$)

- represents set X in low memory
- Insert(x)
- Search(x) { $x \in X \Rightarrow$ always True; $x \notin X \Rightarrow$ sometimes also True (false positive) }

- implementation:
  - bit array $B[0 \dots m-1]$
  - hash function $h: \mathcal{U} \to [m]$
  - Insert(x) → $B[h(x)] \leftarrow 1$
  - Search(x) → return $B[h(x)]$
- $\Pr[\text{false positive}] \leq n/m$

single band filter

- multiband filter = k independent filters
  - (B_i, h_i) = band
  - → $\Pr[FP] \leq (n/m)^k = \frac{1}{2^k}$ for $m = 2n$

also possible to use single table of $k \cdot m$ bits and k functions $h_i: \mathcal{U} \to [k \cdot m]$

Optimizing parameters

- assume n given

- $\varepsilon > 0$ given error probability (binary log!)

Thm: Set $m = 2n$, $k = \lceil \log(1/\varepsilon) \rceil$.
Then we get $\Pr[FP] \leq \varepsilon$.

↳ need $2n \cdot \lceil \log(1/\varepsilon) \rceil$ bits

↳ for $\varepsilon = 0.01 \approx 14$ Mb
vs ≈ 100 Mb for single band

- can we do better memory-wise?
  ↳ can we choose better m and k?

- given M bits
  - → find m, k s.t. $m \cdot k \leq M$ with ε smallest

- assume perfectly random hash functions

- single band in the filter:
  - → $\forall t: Pr[B_t[i] = 0] = (1 - 1/m)^n$ — m is parameter!
    - $Pr[h_t(x_j) \neq i]$
  - → $(1-1/m)^n \approx e^{-n/m} = p$
    - ↳ $\leq$ and "$=$" for $m \to \infty$

better bound for $Pr[FP]$ than n/m from union bound (but needs perfectly random functions)

value of p determines other parameters
- → $m \approx -n / \ln p$
  - ↳ ! base e log !
- → $k = \lfloor M/m \rfloor \approx -M/n \cdot \ln p$
  - ↳ bands must fit into M bits
- → $\varepsilon = Pr[FP] = Pr[\forall t\ B_t[i] = 1]$
  $= (1-p)^k = e^{k \cdot \ln(1-p)} \approx \exp\left(-M/n \cdot \ln(1-p) \cdot \ln(p) \right)$
  - $-M/n$: fixed
  - $\ln(1-p) \cdot \ln(p)$: maximize w.r.t. p to get min ε
    - ↳ maximize $\ln(1-p) \cdot \ln(p)$ s.t. $p \in (0,1)$ using basic calculus

min ε is for $p = 1/2$
k

$\Rightarrow \varepsilon = 2^{-k} = 2^{-M/n \cdot \ln 2}$

For fixed ε and minimizing M we get:
- → fix $m = n/\ln 2$
- → $k = \lceil \log(1/\varepsilon) \rceil$ (to get $Pr[FP] \leq \varepsilon$)
- → $M = k \cdot m \approx n \cdot \log(1/\varepsilon) \cdot (1/\ln 2) \doteq n \cdot \log(1/\varepsilon) \cdot 1.44$
  - ↳ vs $n \cdot \log(1/\varepsilon) \cdot 2$ for $m = 2n$

Note: It can be shown that no data structure can do better than $n \cdot \log(1/\varepsilon)$ for this problem

Counting filters

? How to implement Delete(x)?

→ setting $B[h(x)] \leftarrow 0$ could cause false negatives if $\exists y \in X$ s.t. $h(y) = h(x)$

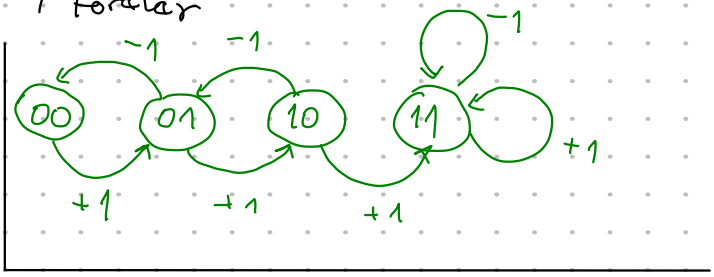
Solution: use counters

- $C[0 \dots m-1]$ array of b-bit counter
- Insert(x) → $C[h(x)]$++
- Delete(x) → $C[h(x)]$--
- Search(x) → return $C[h(x)] \overset{?}{>} 0$ ← still allows false positives
  - ↳ $\exists y \in X$ s.t. $h(y) = t \Rightarrow C[t] > 0$
  - ⇒ no false negatives

Problem: what if counter overflows?

Solution: counter gets "stuck" at $2^b - 1$ forever

"simple" filter is counting filter with b=1



? What is the probality that C[i] gets stuck?

- assume single hand and fully random hash function, after n inserts.
- $C'[i]$ = #increments of $C[i]$ (e.g. what would be in C[i] if we had unlimited precision)

$$Pr[C[i] = t] = \binom{n}{t} \cdot \left(\frac{1}{m}\right)^t \cdot \left(1 - 1/m\right)^{n-t}$$

↝ for each of x_j we toss a coin
- → with probability $1/m$ x_j falls into bucket i
- ↝ binomial distribution with $p = 1/m$

$$\Rightarrow Pr[C'[i] \geq t] \leq \binom{n}{t} \cdot (1/m)^t$$

↳ we don't care whether rest of the elements end up in i or not

→ $t = 2^b - 1$

$$Pr[C[i] \text{ is stuck}] \leq Pr[C'[i] \geq t] \leq \left(\frac{n \cdot a}{t}\right)^t \cdot \left(\frac{1}{m}\right)^t$$

↳ for any t: $\binom{n}{t} \leq \left(\frac{na}{t}\right)^t$

→ for $m = n/\ln 2$:

$$Pr[C[i] \text{ is stuck}] \leq \left(\frac{a \cdot \ln 2}{t}\right)^t \quad \rightarrow \text{for } b=4 \rightsquigarrow \leq 10^{-13}$$

$$Pr[\exists i \text{ s.t. } C[i] \text{ is stuck}] \leq m \cdot \left(\frac{a \ln 2}{t}\right)^t \rightarrow \text{for } m = 10^9 \rightsquigarrow \leq 10^{-4}$$

Note: We can periodically rebuild filter to get rid of stuck filters

↳ but we need to know "true" content of X