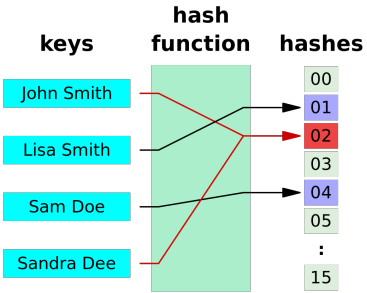
Napredni algoritmi i strukture podataka

Bloom filter, Count-min Sketch, HyperLogLog

Univerzitet u Novom Sadu
Fakultet Tehničkih Nauka

Hash funkcije

- Determinističke funkcije koje preslikavaju ulaze varijabilne dužine na izlaze fiksne dužine
- Različiti ulazi mogu imati iste izlaze - **kolizija**
- "Dobre" hash funkcije su otporne na kolizije, tačnije teško je naći dva ulaza koji imaju isti izlaz
- Kriptografske hash funkcije treba da budu one-way funkcije - na osnovu izlaza ne možemo zaključiti šta je bio ulaz



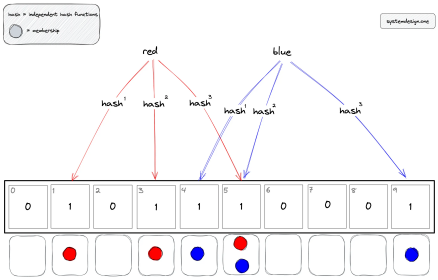
Bloom filter

Bloom filter je probabilistička struktura koja nam daje odgovor na pitanje: Da li je element prisutan u skupu?

- Element **sigurno** nije prisutan
- Element je **možda** prisutan

Bloom filter se sastoji iz:

- Niza bitova veličine **m**
- **k** hash funkcija



Bloom filter - dodavanje

Kada želimo da dodamo element u set:

- koristeći **k** hash funkcija $(h_1(x), h_2(x), \dots h_k(x))$ potrebno je da izračunamo indekse u setu koje ćemo prebaciti sa **0** na **1**.
- ako se desi kolizija, tj. da je bit već postavljen na vrednost **1**, sve ok nastavljamo dalje
- Za set veličine **m**, imamo **k** hash funkcija, onda je proces dobijanja indeksa sledeći:

$$h_1(\text{"key"}) \ \% \ m = i_1$$
$$h_2(\text{"key"}) \ \% \ m = i_2$$
$$\vdots$$
$$h_k(\text{"key"}) \ \% \ m = i_k$$

Gde $i_{ith} \in \{0, 1, \dots m-1\}$

Bloom filter - upit

Kada želimo da proverimo da li je element prisutan u setu:

- koristeći **k** hash funkcija $(h_1(x), h_2(x), \ldots h_k(x))$ potrebno je da izračunamo indekse u setu gde treba da proverimo da li je vrednost **0**
- Da bi smatrali da je element u setu, svih **k** indeksa treba da vrate vrednost **1**
- Ova operacija može da dovede do **false-positive** resultata

Bloom filter - parametri

- Parametre **m** i **k** nećemo nasumično birati
- Njih biramo shodno tome koju verovatnoću **false-positive** dopuštamo u sistemu
- Ako pretpostavimo da će set sadržati **n** elemenata, onda verovatnoću **p** možemo izračuanti sa: $p = (1 - [1 - \frac{1}{m}]^{kn})^k$
- Veličinu bit seta **m** možemo izračunati na sledeći način: $m = -\frac{n \ln p}{(\ln 2)^2}$
- Optimalan broj hash funkcija **k**, možemo izračunati na sledeći način: $k = \frac{m}{n} \ln 2$

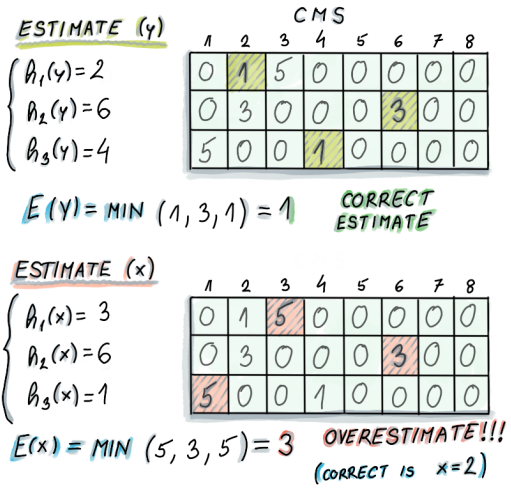
Count-min sketch

Count-min sketch je probabilistička struktura koja nam daje odgovor na pitanje: Koliko puta se neki element ponavlja u skupu?

- Element se ponavlja **najviše** N puta

Bloom filter se sastoji iz:

- 2D niza celobrojnih vrednosti veličine $k \times m$
- **k** hash funkcija



Count-min sketch - dodavanje

Za element sa ključem **K**, postupak dodavanja je sledeći:

- Propustimo **K** kroz **svaku hash funkciju**: $\forall\ h_i \in \{1, ..., k\}$
- Dobijemo vrednost kolone: $j = h_i(K)\ \%\ m$
- Na preseku reda i kolone povećamo vrednost za **1**: $CMS[i, j] += 1$

Count-min sketch - upit

Ako želimo da vidimo učestalost elementa sa ključem **K**, postupak je sledeći:

- Propustimo **K** kroz **svaku hash funkciju**: $\forall\ h_i \in \{1, ..., k\}$
- Dobijemo vrednost kolone: $j = h_i(K)\ \%\ m$
- Formiramo niz vrednosti sa odgovarajućih pozicija $R[i] = CMS[i, j], i \in \{0, ..., k\}$
- Uzmemo minimum iz niza i to je procena učestalosti elementa pod ključem **K** $E(K) = min(R[i]), i \in \{1, ..., k\}$

Count-min sketch - parametri

- Parametre **k** i **m** nećemo nasumično birati
- Kao i kod Bloom Filtera možemo da se oslonimo na malo matematike
- Ako hoćemo da definišemo tabelu veličine $k \times m$, treba da izaberemo očekivanu stopu greške (ε) sa nekom verovatnoćom $(1 - \delta)$
- Dobijamo $k = [\ln \frac{1}{\delta}]$ i $w = [\frac{e}{\varepsilon}]$, gde je e Ojlerov broj

ε	$1 - \delta$	w	d	wd
0.1	0.9	28	3	84
0.1	0.99	28	5	140
0.1	0.999	28	7	196
0.01	0.9	272	3	816
0.01	0.99	272	5	1360
0.01	0.999	272	7	1940
0.001	0.999	2719	7	19033

(Introduction to Probabilistic Data Structures, DZone)

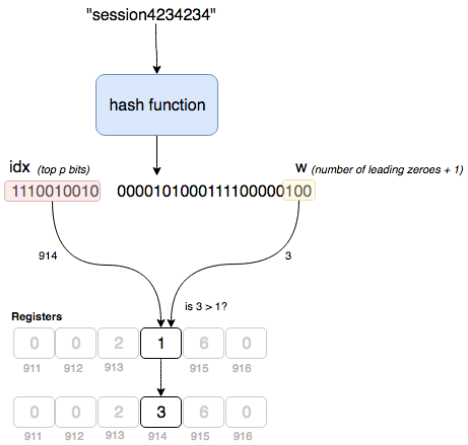
Napomena: $d = k, w = m$

HyperLogLog

HyperLogLog je probabilistička struktura koja nam daje odgovor na pitanje: Koliko jedinstvenih elemenata se nalazi u skupu?

HyperLogLog se sastoji iz:

- Niza celobrojnih vrednosti veličine **m**
- Jedne hash funkcije



HyperLogLog - parametri

- Oslanjamo se na nekoliko parametara:
  - **p** koliko vodećih bitova koristimo za baket
  - **m** veličina seta
- Prvo moramo da odredimo koliko vodećih bitova korisitmo za baket **p** (kolika je preciznost) obično u intervalu $[4, 16]$
- Veća vrednost **p** smanjuje grešku u brojanju, koristeći više memorije
- Nakon toga treba da izračunamo koliki nam set **m** treba koristeći formulu $\mathrm{m} = 2^{\mathrm{p}}$

HyperLogLog - dodavanje

- Pretpostavimo da nakon hash funkcije i pretvaranja u binarni oblik, naš ključ **K** ima vrednost *1011011101101100000*
- Pretpostavimo da za preciznost odaberemo vrednost **4** ($p = 4$)
- Kao rezultat toga, znamo da je veličina seta $m = 2^4$ tj. **16** (po formuli $m = 2^p$)
- Iz dobijene binarne vrednosti *1011011101101100000* zaključujemo da je vrednost bucket-a gde ćemo upisati vrednost *1011* tj. **11**
- Vrednost koju upisujemo u baket *11* je *6 (5+1)*, zato što je broj nula sa kraja **5**, od ostalog dela binarnog zapisa *011101101100000*

HyperLogLog - upit

- Durand-Flajolet je izveo konstantu da ispravi pristrasnost ka većim procenama (algoritam se zove LogLog).
- $\text{CARDINALITY}_{\text{HLL}} = \text{constant} * m * \frac{m}{\sum_{j=1}^{m} 2^{-R_j}}$
- R_j označavaju registar ili pojedinačnu promenljivu koja sadrži najduži niz uzastopnih nula
- Izraz $\sum_{j=1}^{m} 2^{-R_j}$ se naziva *harmonijska sredina* čime se postiže smanjenje greške bez povećanja potrebne memorije (za dokaz konsultovati originalan rad)
- Vrednost promenljive constant se obično računa i stvar je procene

Zadaci

Bloom filter zadaci

- Implementirati Bloon filter strukturu podataka
- Omogućiti da se struktura serijalizuje i deserijalizuje sa diska
- Za formule, koristiti funkcije date u helper fajlovima

Count-min sketch zadaci

- Implementirati Count-min sketch strukturu podataka
- Omogućiti da se struktura serijalizuje i deserijalizuje sa diska
- Za formule, koristiti funkcije date u helper fajlovima

HyperLogLog zadaci

- Implementirati HyperLogLog strukturu podataka
- Omogućiti da se struktura serijalizuje i deserijalizuje sa diska
- Za formule, koristiti funkcije date u helper fajlovima