

Napredni algoritmi i strukture podataka

Bloom filter, Count-min Sketch, SimHash



Univerzitet u Novom Sadu
Fakultet Tehničkih Nauka

Bloom filter - parametri

Bloom filter zahteva nekoliko parametara:

- ▶ Niz bitova veličine **m**, gde su svi bitovi inicijalno postavljeni na vrednost **0**
- ▶ **k** hash funkcija za izračunavanje heševa za dati ulaz
- ▶ Koristeći prethodne parametre, možemo odrediti poziciju bit-a koji treba da prebacimo sa **0** na **1**

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:

$$h_1(\text{"key"}) \% m = i_1$$

$$h_2(\text{"key"}) \% m = i_2$$

$$\vdots$$

$$h_k(\text{"key"}) \% m = i_k$$

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

Bloom filter - pretraga

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

- ▶ koristeći k hash funkcija ($h_1(x), h_2(x), \dots, h_k(x)$) 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** rezultata

Bloom filter - formule

- ▶ 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 izračunati 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 - 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 preciznost (ϵ) koju želimo da postignemo, kao i sigurnost sa kojom dolazimo do tačnosti (δ)
- ▶ Dobijamo $k = \lceil \ln \frac{1}{\delta} \rceil$ i $w = \lceil \frac{\epsilon}{\epsilon} \rceil$, gde je ϵ 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: $d = k, w = m$

Count-min sketch - Dodavanje

Ako dobijemo element iz stream-a sa ključem **K**, postupak dodavanja je sledeći:

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

Count-min sketch - Dobijanje vrednosti

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

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

SimHash - algoritam

- ▶ Set podataka (npr. tekst) podelimo na delove i uklonite zaustavne reči (ako ih ima)
- ▶ Dodelimo težine dobijenim vrednostima (npr. broj ponavljanja reči)
- ▶ Izračunamo **b**-bitni *hash* za svaki element iz dobijenog skupa, propuštajući element kroz *hash* funkciju
- ▶ Za svaku dobijenu vrednost uradimo konverziju **0** → **-1**
- ▶ Formiramo tabelu, tako što vrednosti stavimo jedne ispod drugih
- ▶ Sumiramo kolone, množeći težine sa vrednošću
- ▶ Ponovo izvršimo konverziju, ali sada za svaku vrednost u dobijenom rezultatu:
 - ▶ if $el > 0$, $el \leftarrow 1$
 - ▶ if $el < 0$, $el \leftarrow 0$
- ▶ Dobijamo **b**-bit fingerprint za ceo ulazni set
- ▶ Uradimo XOR operaciju sa drugim setom podataka i dobijamo Hemingovu udaljenost

Bloom filter zadaci

- ▶ Implementirati Bloom filter struktur podataka
- ▶ Omogućiti da se struktura serijalizuje, i deserijalizuje sa diska
- ▶ Za formule, koristiti funkcije date u helper.go fajlu

Count-min sketch zadaci

- ▶ Implementirati Count-min sketch struktur podataka
- ▶ Za formule, koristiti funkcije date u helper.go fajlu

SimHash sketch zadaci

- ▶ Implementirati SimHash strukture podataka
- ▶ Za formule, koristiti funkcije date u helper.go fajlu
- ▶ Koristiti date tekstualne fajlove za proveru Hemingovog rastojanja