Cuckoo filter

Cuckoo filter je probabilistička struktura podataka koja omogućuje brzo i efikasno testiranje pripadanje elemenata skupovima. Ovaj filter je zapravo poboljšana verzija 'bloom' filtera.*.*

Cuckoo filter vraća *'false positive'* vrijednosti pripadnosti skupu odnosno može se sa sigurnošću zaključiti da neki element ne pripada pojedinom skupu. Cuckoo filter je minimizirana varijanta *hash* tablica koja koristi *cuckoo hashing* za razrješavanje sukoba tokom popunjavanja tablice. Prostorna kompleksnost se minimizira tako što se zadržava samo *fingerprint* (eng. otisak prsta)od vrijednosti koju želimo dodati setu. *Fingerprint*  je *string* dobiven od podatka koristeći *hash* funkciju, te je *f-*bitne veličine, gdje se *f* odabere s obzirom koja se vjerojatnost *false positive* rješenja želi. Filter je gusto popunjen sa *fingerprint-*ovima cca 95% što omogućava dobru iskoristivost memorije. Kada se određeni podatak traži, pretražuje se *hash* tablica za njegov *fingerprint* i vraća vrijednost *true* ako je identični *fingerprint* pronađen. Kod konstruiranja filtera, veličina *fingerprint-*a određena je sa željenim *false positive* postotkom koji se označava sa . Što manji želimo, potrebniji je veći *fingerprint.*

Prednosti *cuckoo* filtera u odnosu na *bloom* filter :

* Podržava umetanje i brisanje podataka dinamički
* Ima bolje performanse pretraživanja elemenata, čak i kad je filter gotovo pun npr. Oko 95% popunjenosti
* Jednostavniji je za implementirati od drugih filtera sličnih performansi
* Koristi manje prostora od *bloom* filtera, ako je ciljani *false positive* manji od 3%

Postoje i limitacije *cuckoo* filtera: moguće je izbrisati jedino elemente za koje se zna da su umetnuti prije i umetanje nije uvijek moguće te je onda potrebno ponovno napraviti *hashing.*

Opis strukture podataka

A close up of a logo

Description automatically generated

Slika 1 Struktura podataka

Gdje je:

m - broj *bucketa*

b - broj ulaza odnosno *fingerprintsa* za svaki *bucket*

f - broj bitova korištenih za reprezentaciju *fingerprinta*

Cuckoo hashing

*Cuckoo hash* tablica se sastoji od polja *buckets-*a (eng. kante) gdje svaki podatak ima dvije

moguće kante u koje može ući, a one se određuju pomoću *hash* funkcija . Kada

se podatak pretražuje provjere se obje kante. Kada se podatak umeće može doći do dva scenarija :

1. Barem jedna kanta od dvije je slobodna i podatak se umeće u nju
2. Nijedna kanta nije slobodna, podatak se umeće u jednu od dva kandidata , izbacuje podatak koji se nalazi u njoj i taj se podatak onda naziva *victim* (eng. žrtva) i to se sprema na neku alternativnu lokaciju

-ako i sljedeća lokacija nije slobodna, *victim*  se ubacuje u tu lokaciju, a podatak koji je tamo bio postaje novi *victim* i izbacuje se te ide na novu lokaciju

-ovakva procedura se može ponavljati određeni broj puta koji je zadan i to se zove *number of kicks*(engl. broj izbačaja) i ako se i nakon određenog *number of kicks* nemože naći slobodna kanta filter se smatra punim i nije moguće umetanje

A close up of a device

Description automatically generated A close up of a device

Description automatically generated

Slika 2- a) scenarijj Slika 3 b) scenarijj

ALGORITMI

Algoritmi rade na principu *partial-key cuckoo hashing* što je varijanta standardnog *cuckoo hashinga* koja omogućava dinamički *insert.*

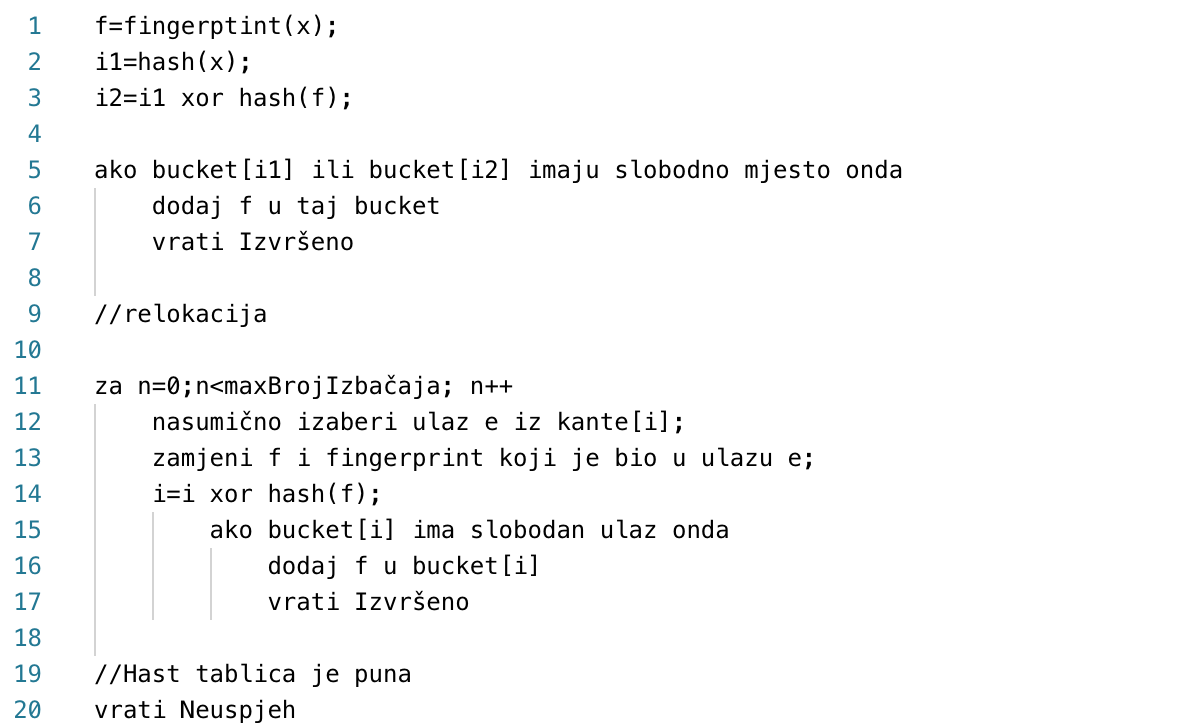
INSERT

Za svaki podatak *x* algoritam računa indekse od dva *bucket* kandidata :

h1(x) = hash(x)

h2(x) = h1(x) ⊕ hash(x’s fingerprint)

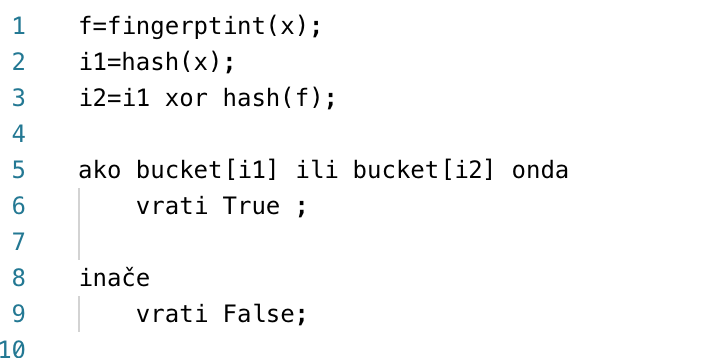
Xor operacija omogućava da se h1(x) može također izračunati od h2(x) i *fingerprint-*a koristeći istu formulu. Odnosno, da stavimo nešto u *bucket i* simulatano računamo i alternativni *bucket* *j*  koristeći *bucket* indeksa *i.* Koristi se *hash*-irani *fingerprint* jer bi inače podaci kada se izbace iz *bucket-*a bili smješteni svi blizu jedni drugih (ako je *fingerprint* mali s obzirom na veličinu tablice). Ovim postupkom se osigurava da se podatci relociraju na potpuno drugi dio tablice kako bi manje došlo do kolizija. Moguće je umetnuti dva ili više podatka s istim *fingerprintom*, ali do *2b* puta, gdje je *b* veličina *bucketa.*



Slika 4 - Pseudokod za insert

LOOKUP

Za podatak x, algoritam prvo računa *fingerprint* i sukladno tome dva *bucketa* koji su kanditati da sadrže x. Ako x postoji u ijednom vraća se pozitivna vrijednost. Zbog ovakvog načina pretraživanja nikad ne dolazi do *false negative* rezultata odnosno ako x stvarno postoji algoritam će ga uvijek pronaći (osim ako ne dođe do *bucket overflow-a*).



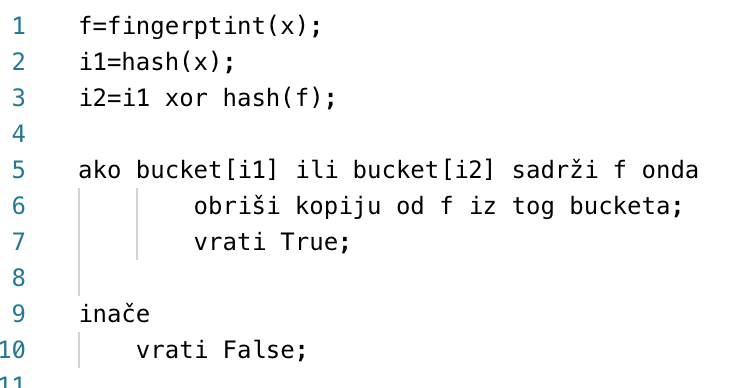
Slika 5 - Pseudokod za lookup

DELETE

Algoritam provjerava oba kandidata gdje bi se podatak mogao nalaziti bazirano na njegovom *fingerprintu* za podatak x, ako se nađe isti u jednoj od *buckets* kopija *fingerprinta* se briše .

U *cuckoo* filteru je izbjegnut problem lažnog brisanja kada dva podatka djele isti *candidate bucket* i imaju isti *fingerprint*. Ako su x i y u *bucket i1* i imaju *fingerprint* *f*  , algoritam *hashi-*inga osigurava da oboje također mogu biti u *bucket i2* zbog toga što je :

Kada se briše x nije bitno jel algoritam izbriše kopiju *f* od x ili y.



Slika 6 - Pseudokod za delete