SVEUČILIŠTE U ZAGREBU

FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

PROJEKT

The Logarithmic Dynamic Cuckoo Filter

David Iveković, Ana Ujević

Voditelj: Mirjana Domazet-Lošo

Zagreb, svibanj, 2024.

**Sadržaj**

[1. Uvod 2](#_Toc1048096407)

[2. Opis problema 3](#_Toc1252724988)

[2.1 Struktura LDCF 4](#_Toc419639630)

[2.2 Opis algoritma 4](#_Toc1495163768)

[3. Objašnjenje algoritma kroz primjer + vizualizacija 5](#_Toc1752204220)

[4. Rezultati mjerenja 6](#_Toc1404025159)

[5. Zaključak 7](#_Toc958132817)

[6. Satežak 8](#_Toc1045307715)

[7. Literatura 9](#_Toc1250958906)

# Uvod

Logaritamski dinamički Cuckoo filter (LDCF) je posebna vrsta strukture podataka koja se koristi za brzo provjeravanje prisustva ili odsustva elemenata u velikim skupovima podataka. On koristi kombinaciju logaritamske dinamike i Cuckoo filtra kako bi efikasno upravljao memorijom i brzo obavljao provjere.

Prije su se za filtriranje i provjeru prisutnosti elemenata u skupu podataka često koristile druge strukture podataka poput hash tablica ili Bloom filtara. Hash tablice su bile popularne zbog brzog pristupa elementima, ali su mogle zahtijevati puno memorije pogotovo kod velikih skupova podataka. Bloom filteri su smanjili potrebu za memorijom koristeći probabilistički pristup, ali su imali ograničenja u tome što su mogli dati lažno pozitivne rezultate (tj. reći da je element prisutan kada zapravo nije).

Taj problem je ublažen s dolaskom LDCF filtra jer je riješio problem prekomjerne potrošnje memorije uz istovremeno održavanje brze i pouzdane provjere prisutnosti elemenata. Danas se koristi u mnogim aplikacijama koje zahtijevaju efikasno upravljanje velikim skupovima podataka, poput baza podataka, sustava za pretraživanje i distribuiranih sustava pohrane podataka

Cilj našeg projekta je napraviti vlastitu LDCF implementaciju te napraviti analizu točnosti, analizu vremena izvođenja i analizu utroška memorije te ga usporediti sa originalnom implementacijom.

# Opis problema

Kao što je spomenuto u uvodu, s porastom velikih skupova podataka, javila se i potreba za boljim predstavljanjem i organiziranjem skupova podataka. Različite strukture podataka i tehnike, kao što su hash kodiranje, Bloom filter (BF), Cuckoo filter (CF) i Dynamic Cuckoo filter (DCF), korištene su u svrhu provjere članstva elemenata u skupovima podataka. Međutim, svaka od ovih tehnika ima svoje nedostatke, a potreba za još učinkovitijim strukturama podataka i dalje postoji.U tom kontekstu, Logarithmic Dynamic Cuckoo Filter (LDCF) predstavlja napredak koji kombinira prednosti prethodnih tehnika uz dodatna poboljšanja.

LDCF koristi kombinaciju hash funkcija, cuckoo hashiranja i dinamičkog prilagođavanja veličine kako bi osigurao visoku učinkovitost u vremenu izvođenja i memoriji. Ovi koncepti omogućuju efikasno upravljanje memorijskim resursima i održava performanse čak i s promjenjivim brojem elemenata.

## Struktura LDCF

Osnovni dizajn Logaritamskog Dinamičkog Cuckoo Filtera (LDCF) koristi **višerazinsku strukturu stabla** za organizaciju blokova Cuckoo filtera (CF), za razliku od klasičnih lančanih struktura. Ova struktura omogućuje LDCF-u da bude prilagodljiv dinamičkim promjenama u skupovima podataka, smanjujući istovremeno računske troškove testiranja i memorijske zahtjeve na logaritamsku razinu.

Struktura LDCF-a sastoji se od **više razina**, gdje svaka razina predstavlja određenu dubinu u hijerarhijskoj organizaciji. Na vrhu strukture nalazi se korijenska razina, koja sadrži blokove Cuckoo filtera koji se koriste kao temelj za ostale razine. Svaka razina sadrži blokove CF-a, pri čemu svaki blok sadrži određeni broj otisaka ili hash vrijednosti.

Ključna značajka ovog dizajna je sposobnost skaliranja i prilagođavanja strukture prema potrebama skupa podataka. Kada se skup podataka poveća ili smanji, LDCF može dinamički dodavati ili uklanjati blokove na odgovarajućim razinama, čime se održava učinkovitost operacija umetanja, provjere članstva i brisanja elemenata.

Dodatna prednost ove strukture je učinkovito upravljanje memorijom jer se memorijski resursi koriste samo onoliko koliko je potrebno za trenutni skup podataka.

## Opis algoritma

Počinje s jednim CF-om na najnižoj razini stabla. Kada se taj CF napuni, dodaju se dodatne razine stabla, a umetanje se nastavlja u nove CF-ove. Na svakoj razini, umetanje se odvija prema uzorku otisaka elemenata. Na primjer, na prvoj razini, elementi s otiscima koji počinju s '0' umetnut će se u jedan od CF-ova, dok će elementi s otiscima koji počinju s '1' biti umetnuti u drugi CF.

Ova struktura omogućuje učinkovito pretraživanje elemenata provjerom otisaka na svakoj razini stabla. Vrijeme potrebno za pretragu u LDCF-u s l razina je O(l), što je znatno manje od linearnog vremena potrebnog za pretragu u drugim strukturama podataka poput DCF filtra. Iako umetanje u LDCF može zahtijevati nešto više računalne obrade nego u DCF-u, to se kompenzira bržim pretraživanjem.

Algoritam LCDF koristi sljedeće funkcije:

* **Funkcija “insert”**

Operacija umetanja prolazi kroz stablo CF-a kako bi pronašla odgovarajući listni CF za umetanje novog elementa. Nakon pronalaska odgovarajućeg CF-a, element se umetne u praznu kantu (eng. bucket). Ako je kanta puna, vrši se nasumično premještanje elemenata kako bi se oslobodio prostor za novi element. Ako je cijeli CF pun, dodaju se novi CF-ovi na istoj razini stabla kako bi se proširila struktura. Ponekad se struktura proširi kako bi se osiguralo dovoljno mjesta za buduće umetanje.

* **Funkcija “append”**

Operacija nadopunjavanja se koristi kada je potrebno dodati elemente bez stvaranja novih CF-ova ili proširivanja strukture. Ako trenutni CF nije pun, element se jednostavno dodaje. Ako je pun, postupak je sličan operaciji umetanja.

* **Operacija “membership test”**

Membership test provjerava prisutnost određenog elementa unutar LDCF-a. Algoritam koristi hash funkciju za generiranje otiska (eng. fingerprint) elementa, te pretražuje kroz odgovarajuće CF-ove u stablu. Ako je element pronađen, vraća se pozitivni rezultat, inače negativni.

* **Funkcija “delete”**

Operacija brisanja uklanja određeni element iz LDCF-a. Algoritam pretražuje stablo CF-ova kako bi pronašao i uklonio element. Ako se element pronađe, uklanja se i struktura se ažurira kako bi odrazila promjene.

# Objašnjenje algoritma kroz primjer + vizualizacija

# Rezultati mjerenja

# Zaključak

# Satežak

# Literatura