SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

SEMINAR

Vlastita implementacija filtera: The Logarithmic Dynamic Cuckoo Filter

Fran Ostroški, Elena Wachtler
Voditelj: *Mirjana Domazet-Lošo*

Zagreb, svibanj 2023.

SADRŽAJ

1. Uvod

Spomenuti glavnu temu projekta, da se radi o projektu u sklopu kolegija Bioinformatika 1. Spomenuti sve izvore iz literature, točno što je koji od njih napravio i kako se jedan razlikuje od drugog (ovo se boduje, čak 3 boda, baš se moraju spomenuti izvori u tekstu, tako piše u uputama).

Franov komentar: ako misliš da treba nešt izmijenit ili nadodat, napiši.

Cuckoo filtar je probabilistička struktura podataka koja se koristi za određivanje pripadnosti zadanog elementa nekom skupu. Sam filtar nastao je kao proširenje već postojećih Bloom filtara, a naziv je dobio prema ptici kukavici koja izbacuje jaja iz tuđih gnijezda kako bi ubacila svoja.
Iako je riječ o općenitim podatcima, cuckoo filtar često se primijenjuje u bioinformatici jer je pogodna za određivanje pripadnosti podnizova nukleotidnih slijedova nekom većem nizu nukleotida ili usporedbu nalaze li se podnizovi jednog slijeda u nekom drugom slijedu.

Pripadnost elementa nekom skupu ovom strukturom ne možemo garantirati, jer zbog implementacije algoritma postoji malena šansa za "false positive" i "false negative" rezultate. Upravo zbog toga što postoji određena vjerojatnost pogreške cuckoo filtar je probabilistički. Međutim, nepripadnost nekom skupu može se pouzdano odrediti.

U ovom projektu u sklopu kolegija Bioinformatika 1 na Fakultetu elektrotehnike i računarstva predstavit ćemo svoju implementaciju cuckoo filtra.

NAPOMENA: u bibtex fileu nedostaje jedan izvor, nisam mogla naći njegov .bib: Fan et al. 2013. Cuckoo Filter: Better Than Bloom; https://www.cs.cmu.edu/ binfan/papers/login cuckoofilter.pdf - našla sam samo za Practically Better Than Bloom.

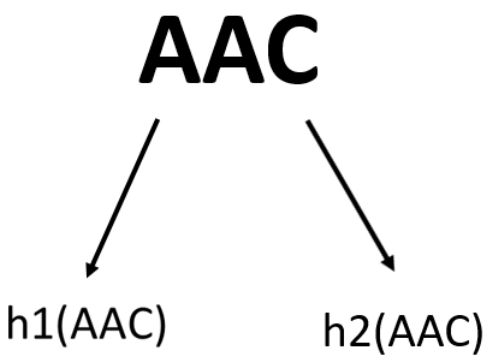
2. Opis algoritma

Mogu se i ovdje koristiti izvori, možda čak i bolje nego da se koriste u Uvodu, iako se mogu i u uvodu spomenuti.

Vizualizacija algoritma na jednostavnom primjeru.

U općenitom slučaju, cuckoo filtar sastoji se nizova memorijskih lokacija koje se nazivaju *bucketima*. Bucketi mogu imati mjesta za više unosa, ali u ovom jednostavnom primjeru svaki bucket moći će primiti po 1 uneseni podatak.

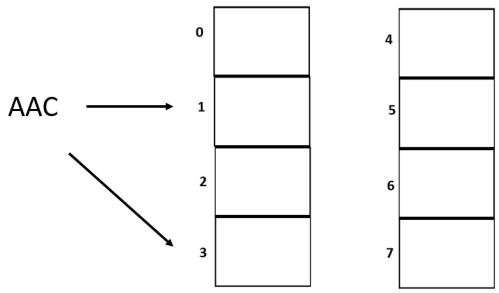
Recimo da imamo jedan kratak nukleotidni slijed, npr. AACTGAT, te kao ulaze u filtar želimo unijeti sve njegove k-mere, za k = 3. To su : AAC, ACT, CTG, TGA i GAT .
U prvom koraku algoritma za svaki unos računaju se dva sažetka (engl. *hash*) pomoću odabranih funkcija sažimanja (hash funkcija).



Slika 2.1: Generiranje hasha - Fran Ostroški

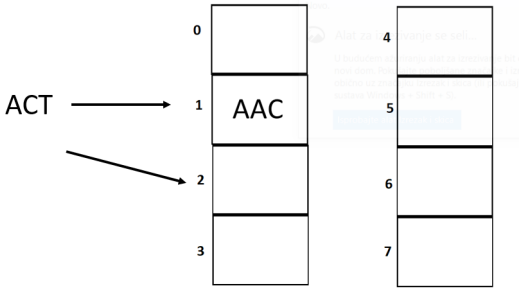
U drugom koraku se iz generiranih sažetaka određuje indeks bucketa u koji će biti smješten ulazni niz. To se najčešće određuje uporabom operacije dijeljenja *modulo* s ukupnim brojem bucketa. Kada se generiaju oba indeksa, element će se spremiti na

jedno od dvije lokacije s tim indeksima. Svrha drugog sažetka je da bi se umanjila vjerojatnost kolizija i potreba za premještanjem.



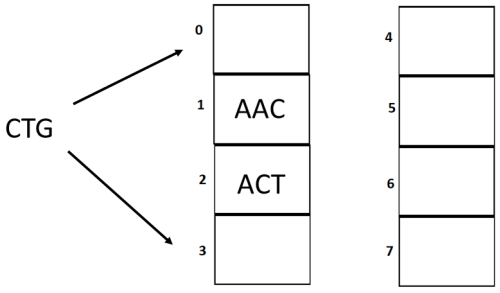
Slika 2.2: Odabir lokacije za niz AAC - prikaz po uzoru na [2]

Idući na redu je niz ACT . Nakon generiranja sažetaka smješta se u prazni bucket. Budući da je bucket na jednom od dobivenih indeksa već popunjen, spremamo ga na drugo, prazno mjesto.



Slika 2.3: Odabir lokacije za niz ACT - prikaz po uzoru na [2]

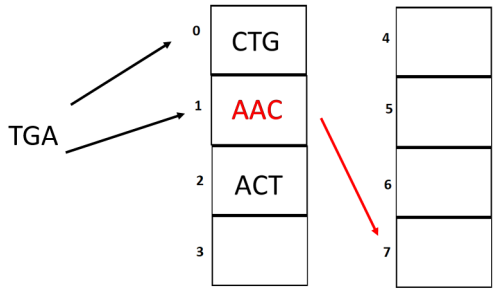
Isto korak se ponavlja za niz CTG.



Slika 2.4: Odabir lokacije za niz CTG - prikaz po uzoru na [2]

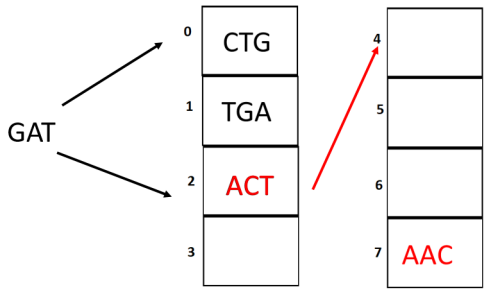
Za niz TGA dogodila se situacija da oba pokazivača pokazuju na indekse s već popunjenim bucketima. Sada se u algoritmu događa "izbacivanje". Obabrani element

izbacujemo, umećemo TGA, a za izbačeni element tražimo novu lokaciju. Odabir lokacije se razlikuje od algoritma do algoritma.

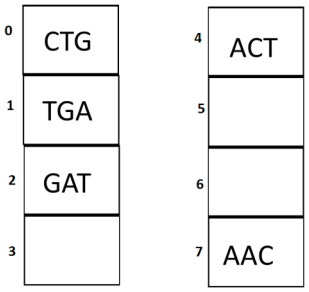


Slika 2.5: Izbacivanje AAC i ubacivanje TGA - prikaz po uzoru na [2]

Za niz GAC također se mora izvršiti izbacivanje starog elementa.



Slika 2.6: Izbacivanje ACT i ubacivanje GAC - prikaz po uzoru na [2]



Slika 2.7: Konačni razmještaj elemenata - prikaz po uzoru na [2]

Možemo primjetiti da što je popunjenost veća raste vjerojatnost da će doći do izbacivanja i relokacije starijih elemenata u filtru. Dogodit će se situacije da će element koji relociramo biti premješten na već popunjeni bucket. U tom slučaju prijašnji element će se relocirati, i tako dalje sve dok se ne pronađe slobodno mjesto ili se dosegne maksimalan broj operacija premještanja. U slučaju da se dosegne maksimalan broj, to

je obično indikator skore popunjenosti filtra. U tom slučaju mogu se poduzeti razne akcije, koje su ponovno različite za svaku implementaciju filtra.

NAPOMENA: Eventualno doradit opise, dodat usporedbe tih algoritama iz dokumentacije po čemu se razlikuju, ali neznam dal smijem njihove slike (npr. tablice s vremenskim složenostima i to) jer piše u uputama kao da treba dopuštenje autora.

3. Analiza

Analiza točnosti, vremena izvođenja i utroška memorije za različite testne slučajeve - ovo nosi 3 boda. Testira se na stvarnim podatcima, kod nas E. coli - rezultati prikazani u tablici ili grafu.

4. Zaključak

Zaključak.

5. Literatura

[1] Hanhua Chen, Liangyi Liao, Hai Jin, i Jie Wu. The dynamic cuckoo filter. U *2017 IEEE 25th International Conference on Network Protocols (ICNP)*, stranice 1–10, 2017. doi: 10.1109/ICNP.2017.8117563.

[2] Bin Fan, Dave G. Andersen, Michael Kaminsky, i Michael D. Mitzenmacher. Cuckoo filter: Practically better than bloom. U *Proceedings of the 10th ACM International on Conference on Emerging Networking Experiments and Technologies*, CoNEXT '14, stranica 75–88, New York, NY, USA, 2014. Association for Computing Machinery. ISBN 9781450332798. doi: 10.1145/2674005.2674994.

[3] Fan Zhang, Hanhua Chen, Hai Jin, i Pedro Reviriego. The logarithmic dynamic cuckoo filter. U *2021 IEEE 37th International Conference on Data Engineering (ICDE)*, stranice 948–959, 2021. doi: 10.1109/ICDE51399.2021.00087.

6. Sažetak

U ovome radu dan je pregled vlastite implementacije rješenja problema opisanog u znanstvenim radovima koja je napravljena u sklopu projekta iz kolegija Bioinformatika 1 na Fakultetu elektrotehnike i računarstva Sveučilišta u Zagrebu.