SVEUČILIŠTE U ZAGREBU

FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Bioinformatika

PROJEKT

Improving Bloom Filter Performance on Sequence Data Using k-mer Bloom Filters

Daria Matković

Voditelj: *Mirjana Domazet-Lošo*

Zagreb, sječanj, 2019

Sadržaj

1. Uvod

Bloom filter je učinkovita probabilistička podatkovna struktura koja se koristi za ispitivanje članstva elemenata u skupu. Bloomov filter može dati lažno pozitivan odgovor, tj. neki element može smatrati članom skupa, iako on to nije, ali Bloomov filter sa sigurnošću može provjeriti da neki element ne pripada skupu.

U ovom projektu Bloom filter je služio za spremanje velikog broja k-merova. K-mer je podniz duljine k. K-merovi se nalaze u fasta datoteci koja je preuzeta sa stranice http://bacteria.ensembl.org/index.html.

Za ovaj projekt korištena je već gotova implementacija Bloom filtera koja je dostupna na https://github.com/mavam/libbf/. Gotova implementacija Bloom filtera se automatski instalira.

2. Metode za poboljšanje performansi Bloom filtera

U ovom projektu su razmatrane dvije vrste poboljšanja performansi Bloom Filtra. Jedan način poboljšanja je tako da se smanjuje broj lažno pozitivnih rezultata, a drugi način poboljšanja performansi Bloom filtera je smanjenjem memorije koju zauzima Bloom filter. Također u oba slučaja želimo da vrijeme potrebno za inicijalizaciju i provjeru članova bude minimalno.

2.1. Metode za smanjenje broja lažno pozitivnih rezultata

Metode za smanjenje broja lažno pozitivnih rezultata su one-sided k-mer Bloom filter i two-sided k-mer Bloom filter.

One-sided k-mer Bloom filter osim k-mera koji mu je upit ispituje i njegove susjede, te upitni k-mer smatra članom skupa tek kada je jedan od susjeda (lijevi ili desnii) također član skupa.

Two-sided k-mer Bloom filter također provjerava susjede i upitni k-mer smatra članom skupa samo ako su njegovi i lijevi i desni susjedi članovi skupa.

Na primjer ako je upitni k-mer: ACCTGATT, onda će lijevi susjed biti XACCTGAT, a desni susjed će biti CCTGATTX, gdje je X = {A, C, T, G}. Dakle skup mogućih lijevih susjeda je: AACCTGAT, CACCTGAT, TACCTGAT, GACCTGAT, a skup mogućih desnih susjeda je: CCTGATTA, CCTGATTC, CCTGATTT, CCTGATTG.

Za one-sided k-mer Bloom filter potrebno je u skupu pronaći k-mer i jedan od osam susjeda, a za Two-sided Bloom filter potrebno je u skupu pronaći k-mer, jedan lijevi i jedan desni susjed.

2.2. Metode za smanjenje korištene memorije

Za smanjenje korištene memorije potrebno je odabrati samo neke k-merove koji se dodaju u Bloom filter. Postoje 3 metode:

a) Metoda odabirom najboljeg indeksa

Skup spremljenih k-merova se odabire tako da se iz dobivene sekvence sprema svaki s-ti k-mer, počevši od određenog indeksa. Indeks od kojeg se počinju odabirati

kmerovi odabran je tako da skup svakog s-tog k-mera koji počinje od tog indeksa treba imati najviše preklapanja sa do sada spremljnim skupom k-merova. Za ovu metodu koristilo se stroga metoda provjeravanja članstva testnog k-mera u skupu spremljenih k-merova.

b) Prorjeđivanje „hitting set" metodom

Koristi se pohlepna metoda kojom se u set dodaju k-merovi koji su se najviše puta pojavili u setu susjeda svih k-merova. Za ovu metodu koristila se opuštena metoda ispitivanja.

c) Metoda odabira svakog s-tog k-kmera iz sekvence

Iz sekvence se svaki s-ti k-mer sprema u Bloom filter. Za ovu metodu koristile su se i opuštena i stroga metoda ispitivanja.

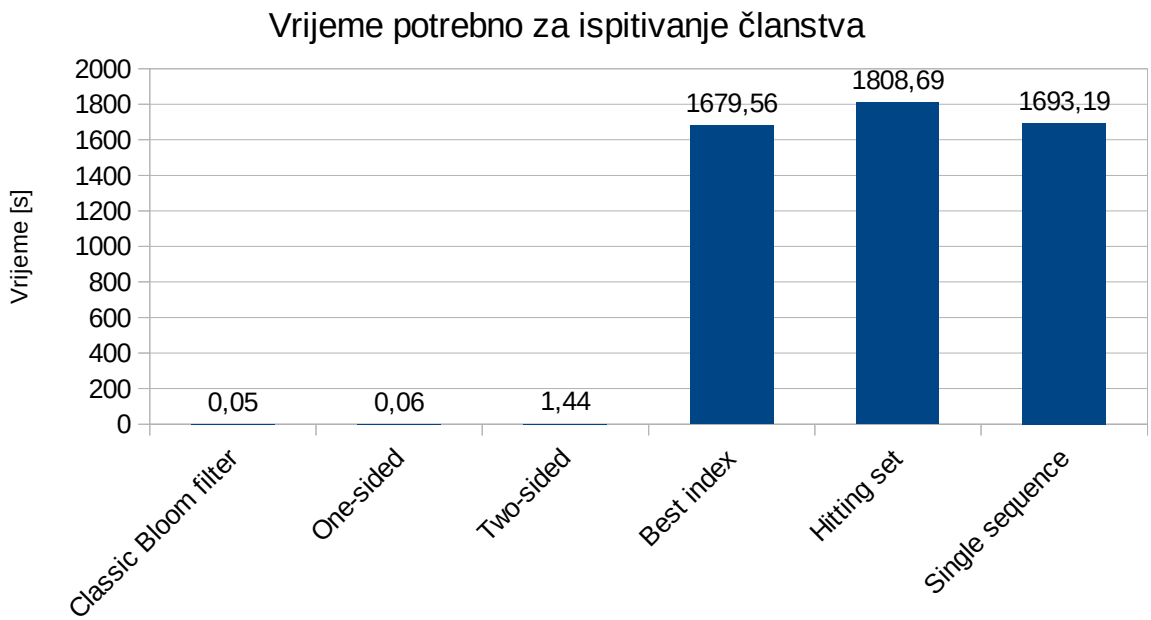
Nakon što se spreme samo određeni k-merovi u Bloom filter, mogu se koristiti dvije metode za provjeru nalazi li se pojedini k-kmer u spremljnom skupu, a to su opuštena metoda provjeravanja i stroga metoda provjeravanja. Na slici 1 prikazani su algoritmi koji opisuju svaku od metoda.

```
1: function DECIDE_PRESENT(query, Contains_left, Contains_right)
2:     if Contains_right == true and Contains_left == true then
3:         return true
4:     if Contains_right == true or Contains_left == true then
5:         if EDGE_k-mer_SET.contains(query) then
6:             return true
7:     return false

8: function STRICT-CONTAINS_NEIGHBOURS(query, left_dist, right_dist)
9:     Contains_left ← CONTAINS_SET( S_DISTANT_LEFT_NEIGHBOUR_SET(query, left_dist))
10:    Contains_right ← CONTAINS_SET(S_DISTANT_RIGHT_NEIGHBOUR_SET(query, right_dist))
11:    return DECIDE_PRESENT(query, Contains_left, Contains_right)

12: function RELAXED-CONTAINS_NEIGHBOURS(query, l_dist, r_dist)
13:    Contains_left ← CONTAINS_SET( ⋃_{i≤l_dist} S_DISTANT_LEFT_NEIGHBOUR_SET(query, i))
14:    Contains_right ← CONTAINS_SET( ⋃_{i≤r_dist} S_DISTANT_RIGHT_NEIGHBOUR_SET(query, i))
15:    return DECIDE_PRESENT(query, Contains_left, Contains_right)

16: function STRICT-CONTAINS(query, s)
17:    if BF.CONTAINS(query) then
18:        if STRICT-CONTAINS_NEIGHBOURS(query, s, s) then
19:            return true
20:    for i ← 0 to s − 1 do
21:        if STRICT-CONTAINS_NEIGHBOURS(query, i, s − (i + 1)) then
22:            return true
23:    return false

24: function RELAXED-CONTAINS(query, s)
25:    if BF.CONTAINS(query) then
26:        if RELAXED-CONTAINS_NEIGHBOURS(query, s, s) then
27:            return true
28:    else
29:        for i ← 0 to s − 1 do
30:            if RELAXED-CONTAINS_NEIGHBOURS(query, i, s − (i + 1)) then
31:                return true
32:    return false
```

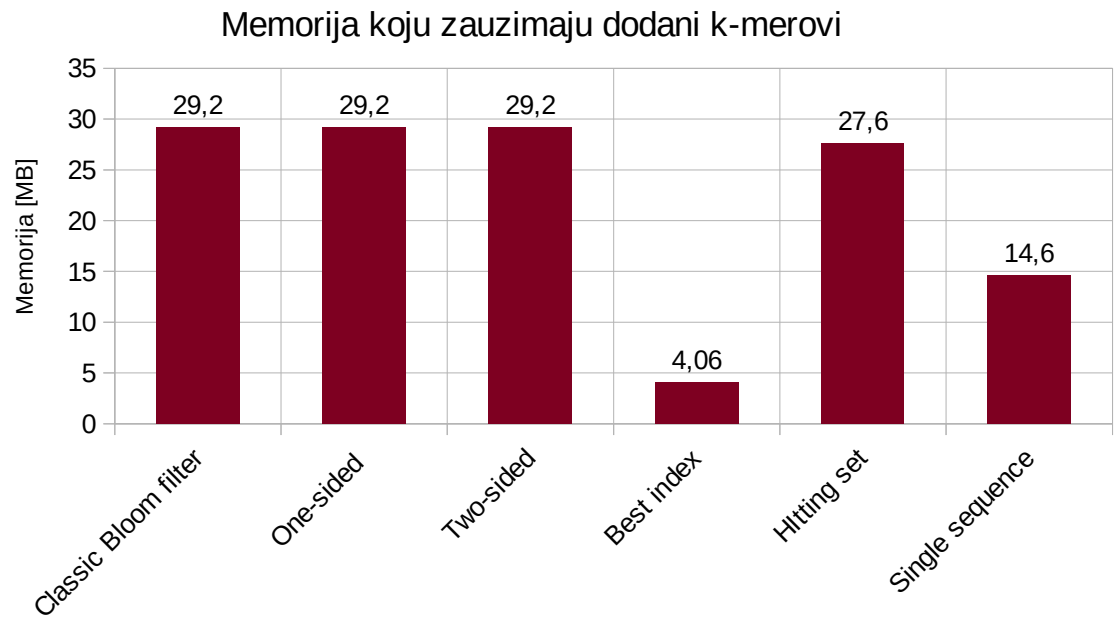
Algoritam 1 Algoritam provjere članstva k-mera kod metoda koje prorjeđuju broj spremljenih k-merova

3. Rezultati

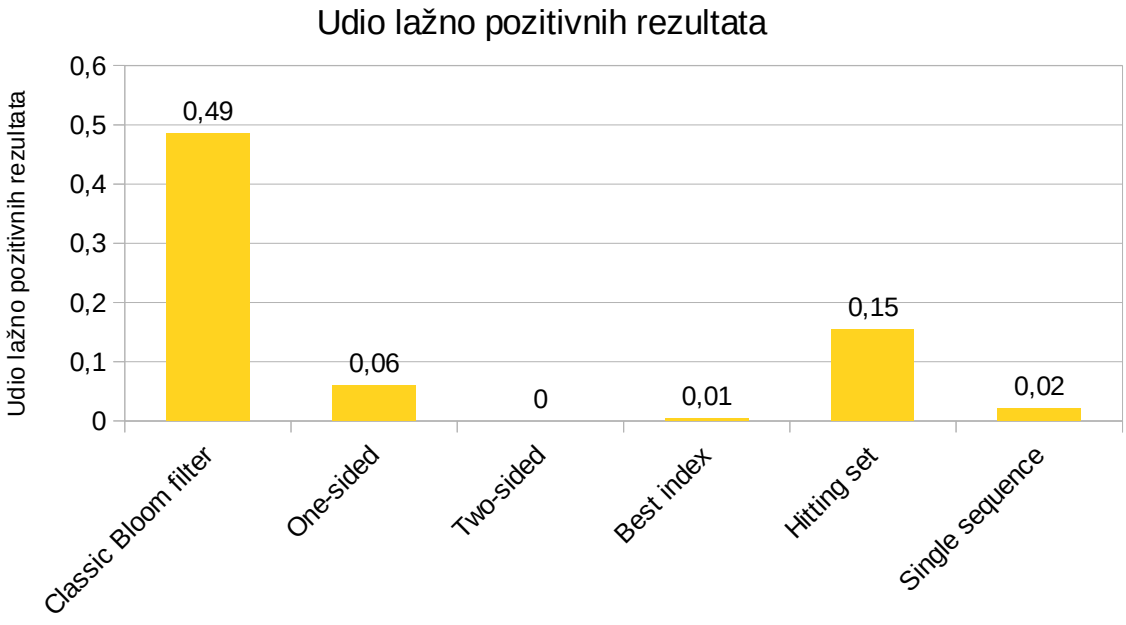
Za testiranje je korištena vibrio natriegens fasta koja ima cca 1e6 znakova. Na grafikonu 1 prikazano je vrijeme ispitivanja članstva testnih k-merova, za svaku od metoda. Na grafikonu 3 prikazan je udio lažno pozitivnih primjera. Ispitivanje je provedeno sa 10000 testnih slučajno generiranih k-merova. Može se primjetiti kako je ispitivanje kraće kada su u Bloom filter dodani samo neki k-merovi. Također klasični Bloom filter ima kraće vrijeme ispitivanja od one-sided i two-sided metoda koje uz ispitne k-merove provjeravaju i njihove susjede. Na grafikonu 2 se vidi da je utrošak memorije za spremanje prorjeđenog skupa prosječno manji nego kod spremanja k-merova kod klasičnog Bloom filtera, one-sided i two-sided Bloom filtera. Na grafikonu 3 vidi se da su metode za prorjeđivanje lošije jer u prosjeku imaju veći broj lažno negativnih rezultata nego druge metode. Najmanji broj lažno negativnih rezultata ima two-sided metoda jer se kod te metode u Bloom filter dodaju svi k-merovi iz fasta datoteke i uz upitni k-mer se provjeravaju i susjedni k-merovi.



Grafikon 1: Vrijeme potrebno za ispitvanje



Grafikon 2 Memorija koju zauzimaju dodani k-merovi



Grafikon 3: Udio lažno pozitivnih rezultata

4. Zaključak i usporedba sa originalnom implementacijom

Može se zaključiti kako su metode za smanjenje broja lažno pozitivnih rezultata (one-sided i two-sided) po tom kriteriju uspješnije odrađuju ispitivanje od drugih metoda, ali zato je ispitivanje duže. Metode za smanjenje utroška memorije su dobre po tom kriteriju ali broj lažno pozitivnih rezultata je veći kod prorjeđenih skupova, nego kod two-sided metode, pa po tom kriteriju nisu toliko dobre.

U originalnoj implementaciji koristile su se drugačije fasta datoteke, drukčija duljina k-merova.

5. Literatura

[1] D. Pellow, D. Filippova, C. Kingsford, Improving Bloom Filter Performance on Sequence Data Using k-mer Bloom Filters, Journal of computational biology, Volume 24, Number 6, 2017

[2] Predavanja iz kolegija Napredni modeli i baze podataka:

https://www.fer.unizg.hr/_download/repository/9._NoSQL_4_od_4.pdf

[3] Ulazni podaci (fasta datoteke): http://bacteria.ensembl.org/index.html

[4] Korištena osnovna implementacija Bloom filtera: https://github.com/mavam/libbf/