SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

SEMINAR

Algoritam za traženje maksimalnih egzaktnih preklapanja u dugačkim sekvencama koristeći rijetka sufiksna polja

Fran Jurišić, Denis Osvald, Tomislav Ljubej

Bioinformatika 2014/2015

Zagreb, siječanj 2015.

SADRŽAJ

1. Uvod

Maksimalna egzaktna preklapanja koriste se u algoritmima sekvenciranja da bi se identificirala potencijalna poravnanja, oslanjajući se na pretpostavku da će broj grešaka prilikom čitanja biti dovoljno mali da relativno velike regije ne budu promijenjene. Algoritmi koji traže maksimalna egzaktna preklapanja će identificirati takve regije koje se dalje mogu koristiti kao informacija za pozicije pokušaja poravnanja sekvenci.

U ovom projektu se ostvarila implementacija jednog takvog algoritma[4] u programskom jeziku C++11, koji koristi rijetka sufiksna polja kako bi smanjio memorijska opterećenja na cijenu računskih. Za konstrukciju sufiksnog polja se koristio algoritam SA-IS[5].

2. Opis algoritama

2.1. Sufiksno polje

Sufiksno polje razmatra sufikse definirane kao niz znakova od neke pozicije u referentnom nizu do njegovog završetka. U sufiksnom polju su te početne pozicije zapisane u redoslijedu koji odgovara leksikografskom poretku odgovarajućih sufiksa. Važno je primijetiti da se za potrebe sufiksnog polja referentni niz završava s posebnim znakom koji predstavlja kraj niza i smatra se leksikografski manjim od svih drugih znakova.

Algoritam SA-IS se temelji na identificiranju uzlaznih sekvenci u nizu znakova koje se zatim dovode u djelomični poredak na temelju usporedbe prvog znaka sekvence i njihovih međusobnih odnosa. Takvi djelomični poretci se promatraju kao problem ekvivalentan početnom te se rekurzivno rješavaju ako iz njih nije moguće izravno konstruirati sufiksno polje. Vremenska složenost ovog algoritma je $O(n)$.

Detaljan opis i primjer primjene algoritma postoji u materijalima popratnog kolegija [2], te se ovdje ne će ulaziti u njega s obzirom da algoritam za maksimalna egzaktna preklapanja ne ovisi o načinu konstrukcije sufiksnog polja.

Sufiksno polje je uglavnom dodatno popraćeno inverznim sufiksnim poljem (ISA) i poljem duljine zajedničkih prefiksa (LCP). Inverzno sufiksno polje na poziciji koja odgovara početnoj poziciji nekog sufiksa sadrži indeks tog sufiksa u sufiksnom polju, a polje duljine zajedničkih prefiksa sadrži duljine jednakih prefiksa koje imaju uzastopni sufiksi u sufiksnom polju.

2.2. Rijetko sufiksno polje

Rijetko sufiksno polje s parametrom k, je sufiksno polje koje sadrži samo indekse sufiksa čiji su indeksi višekratnici k. Iz punog sufiksnog polja je relativno lagano probrati potrebne indekse za rijetko sufiksno polje zato što je potrebno samo da njihov relativni poredak ostane očuvan. Važno je primijetiti da algoritmi često očekuju da

```
    0  1  2  3  4  5  6  7  8  9 10 11 12 13
    A  N  A  N  A  S  I  B  A  N  A  N  A  $

ISA 4 11  5 12  6 13  8  7  3 10  2  9  1  0

    SA                    LCP
    0 13 $                  -1
    1 12 A$                  0
    2 10 ANA$                1
    3  8 ANANA$              3
    4  0 ANANASIBANANA$      5
    5  2 ANASIBANANA$        3
    6  4 ASIBANANA$          1
    7  7 BANANA$             0
    8  6 IBANANA$            0
    9 11 NA$                 0
   10  9 NANA$               2
   11  1 NANASIBANANA$       4
   12  3 NASIBANANA$         2
   13  5 SIBANANA$           0
```

Slika 2.1: *Primjer SA, ISA i LCP.*

će prvi sufiks u sufiksnom polju biti završni znak, pa ako početni referenti niz ne ma završni znak na indeksu koji je višekratnik od k potrebno ga je proširiti s završnim znakovima dok to ne bude zadovoljeno.

Popratno rijetko inverzno sufiksno polje je također jednostavno dobiti, samo je bitna interpretacija da u takvom polju indeks i zapravo predstavlja sufiks indeksa $i * k$. Za konstrukciju polja duljine zajedničkih prefiksa smo koristili algoritam[3] namijenjen punim sufiksnim poljima, koji smo modificirali u algoritmu 1 za rad s rijetkim sufiksnim poljima, pri čemu su sve pretpostavke održane tako da je linearne složenosti.

2.3. Maksimalna egzaktna preklapanja

Algoritam za maksimalna egzaktna preklapanja[4] koristi rijetko sufiksno polje da bi binarnim pretraživanjem brzo pronašao intervale sufiksnog polja čiji sufiksi imaju minimalnu i maksimalnu duljinu preklapanja s ispitnim nizom.

Takve intervale podudaranja se traži tako da se krene od intervala koji sadrži cijeli SA (0 podudaranja), te se svakim znakom interval sužava tako da su u njemu samo sufiksi koji se podudaraju barem jedan znak više. Svaki sljedeći interval je moguće

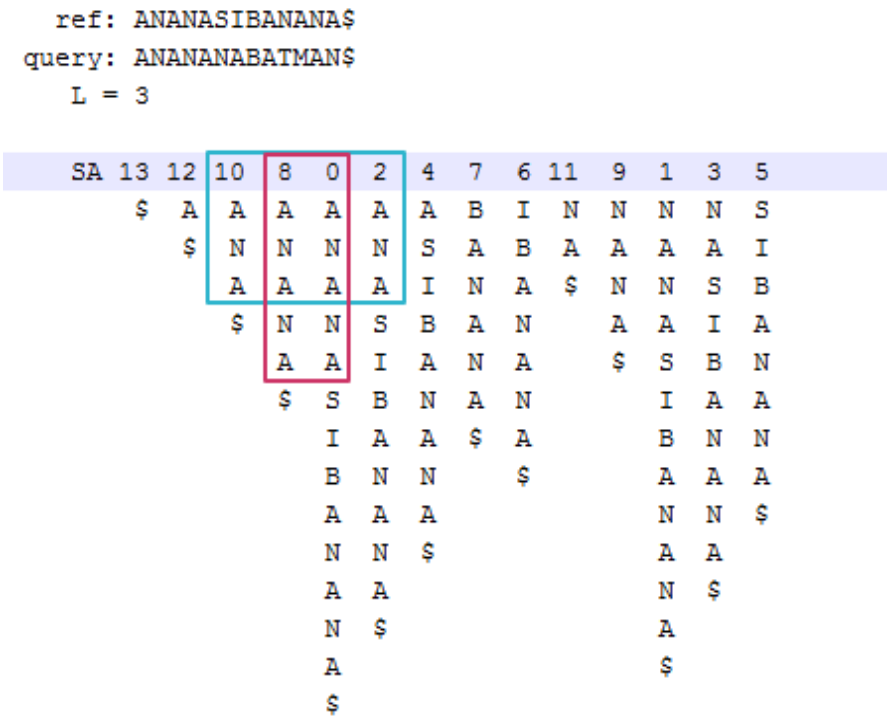
Algorithm 1 Algoritam za konstrukciju LCP

Require: S referenti niz
Require: SSA rijetko sufiksno polje
Require: ISA inverzno rijetko sufiksno polje
Require: n duljina rijetkog sufiksnog polja
Require: k faktor rijetkog sufiksnog polja

```
l ← 0
LCP[0] ← −1
for i = 0 → n − 1 do
    t ← ISA[i]
    j ← SA[t − 1]
    while S[i + l] == S[j + l] do
        l ← l + 1
    end while
    LCP[t] ← l
    if l > k − 1 then
        l ← l − k
    else
        l ← 0
    end if
end for
return LCP
```

pronaći binarnom pretragom gornje i donje granice, a u cijelom postupku su nam posebno važni interval minimalnog preklapanja, gdje se preklapa barem L znakova, i interval maksimalnog preklapanja, koji nije moguće dalje suziti i nužno je sadržan u intervalu minimalnog preklapanja. Svaki sufiks u minimalnom intervalu preklapanja je potencijalno maksimalno egzaktno preklapanje, a interval maksimalnog preklapanja služi da bi se brže napravilo proširenje preklapanja unaprijed korištenjem LCP-a.

Algoritam traži ovakve intervale za svaki sufiks ispitnog niza, pa je važno primijetiti da nije potrebno pokušavati proširiti preklapanje unazad, štoviše ako je moguće neki sufiks u intervalu preklapanja proširiti unazad on se zanemaruje zato što to znači da je podniz nekog drugog maksimalnog egzaktnog preklapanja.



Slika 2.2: *Primjer pretrage minimalnog i maksimalnog intervala preklapanja sufiksa.*

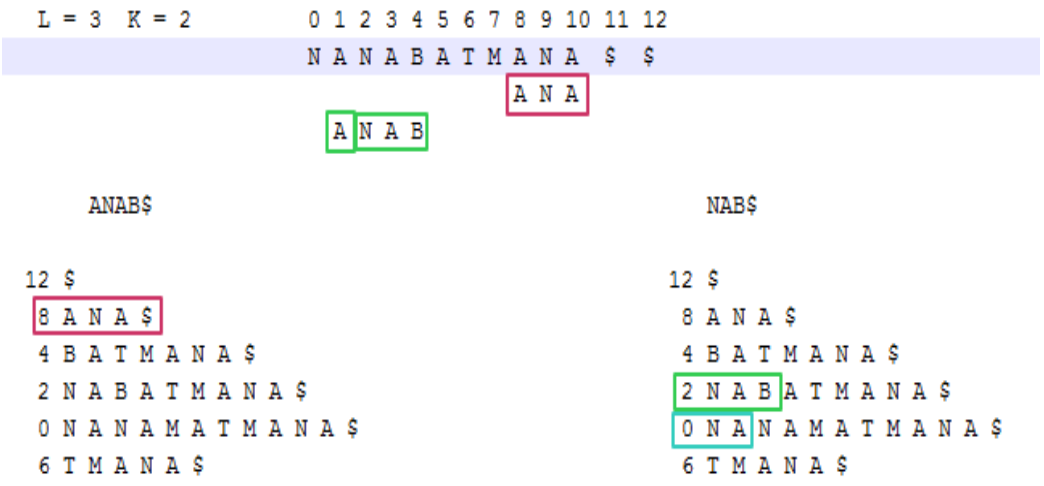
Ovaj grubi opis algoritma razmatra puno sufiksno polje, te je potrebno istaknuti promjene da bi mogao raditi s rijetkim sufiksnim poljem.

Sufiksi u intervalu preklapanja na rijetkom polju se mogu proširivati unazad, zato što je prvi sljedeći sufiks koji se može naći ovakvom pretragom udaljen k znakova. Ukoliko se preklapanje može proširiti barem k znakova, ono se odbacuje iz istog razloga kao i prije, zato što je podniz nekog drugog maksimalnog egzaktnog preklapanja, ali ako se može proširiti za manje od k znakova onda je jedinstveno maksimalno eg-

zaktno preklapanje.

Zbog mogućnosti proširenja unazad se mijenja kriterij minimalnog intervala preklapanja, u kojem sada ulaze sufiksi koji se moraju preklapati u barem $L-(k-1)$ znakova kako se proširenjem unazad mogu povećati za do $(k-1)$ znakova. Sada je važno i provjeriti za svako maksimalno prošireno preklapanje, da li je dovoljne duljine L.

Na slici 2.3 se može vidjeti primjer rada algoritma na referentom niz *NANABATMANA* i upitnom nizu *ANAB*, uz $L=3, K=2$. Pronađena su dva maksimalna egzaktna preklapanja *ANA* i *ANAB*, ti time da je drugo prilikom traženja intervala preklapanja u SA bilo pronađeno kao *NAB* i zatim prošireno još jedan znak unatrag. Pretragom intervala preklapanja SA pronađeno je još potencijalno preklapanje *NA*, ali se ono nije uspjelo proširiti unazad i doseći traženu duljinu L.



Slika 2.3: *Primjer traženja maksimalnih egzaktnih preklapanja - MEM.*

2.4. Sufiksne veze

Veliki dio računanja u algoritmu se svodi na binarne pretrage intervala preklapanja sufiksa, no prilikom svakog pomicanja u upitnom nizu, dio preklapanja koji slijedi nakon znakova preko kojih se pomaknulo je ostao očuvan. Ta informacija se nastoji iskoristiti, te se simulacijom sufiksnih veza postojeći interval preklapanja pomakne i djelomično očuva za sljedeći sufiks upitnog niza.

Nove granice intervala preklapanja se mogu dobiti s:

$$left = ISA[SA[left]/K + 1]$$

$$right = ISA[SA[right]/K + 1]$$

Kako bi se primijenilo ovu tehniku ubrzanja potrebno je, zbog rijetkog sufiksnog i inverznog sufiksnog polja, pomicati poziciju u upitnom nizu za K znakova.

Nadalje ovaj novi interval neće biti nužno jednak intervalu koji bi se dobio počevši pretragu od početka, ali će nužno biti sadržan u tom intervalu. Stoga je potrebno pokušati proširiti obje granice intervala promatrajući vrijednosti LCP-a, s ograničenim brojem koraka, tako da složenost u najgorem slučaju ostane jednaka složenosti bez simulacije sufiksnik veza $O(m^2 \log n)$ gdje je m duljina upitnog niza, a n duljina referentnog niza.

2.5. Paralelizacija

Paralelizacija na razini jednog upitnog niza se temelji na radu simulacije sufiksnih veza. Proces traženja intervala preklapanja u SA je potrebno pokrenuti za svaki sufiks upitnog niza, ali ubrzanje koje nastaje korištenjem simulacija sufiksnih veza pomiče poziciju u upitnom nizu K mjesta, te je potrebno da se sufiksi upitnog niza odrađuju u skokovima duljine K. Stoga potrebno je pokrenuti postupak uz početne indekse $0..K-1$, što se može odraditi paralelno.

3. Performanse

Testiranje performansi implementiranih SA i MEM algoritama rađeno je uzimajući kao referentni niz genom Caenorhabditis Elegans [1]. Taj je niz duljine približno 100 milijuna znakova. Iz njega je konstruiran velik broj upitnih podnizova s kojim su tražena preklapanja referentnog niza.

Testirane su performanse konstrukcije rijetkog sufiksnog niza iz referentnog, te zatim traženje preklapanja podnizova. Trajanje i zauzeće memorije su prikazani u tablici, a u svim prikazanim slučajevima traženi su MEM-ovi duljine 20 ili više (L=20).

numMEM	qlen	K	time (**real**,user) [s]	peakmem [KiB]
2,000,000	1000	1	**10m34.73s** 10m29.32s	1,387,048
20,000,000	200	1	**20m68.58s** 20m58.99s	1,633,144
2,000,000	1000	2	**18m35.75s** 22m55.45s	1,080,316
20,000,000	200	2	**36m53.92s** 46m08.97s	1,080,524
2,000,000	1000	3	**26m43.37s** 54m28.92s	884,392
20,000,000	200	3	**53m38.71s** 1h55m23.85s	884,032

Slika 3.1: *Rezultati testiranja performansi. Imajte na umu da se učitavalo puno sufiksno polje veličine oko 400,000 KiB prije nego što je pretvoreno u rijetko, što je u utjecalo na peakmem.*

4. Zaključak

Algoritam za pronalazak maksimalnih egzaktnih preklapanja omogućava vrlo brzi pronalazak područja preklapanja, a varijabili faktor rijetkosti k omogućava upravljanje omjerom vremenske i memorijske zahtjevnosti problema. U originalnom radu [4], zamijećeno je da algoritam najbolje radi za manje vrijednosti k, zato što je za ubrzanje koje se ostvaruje uporabom simuliranih sufiksnih veza potrebno da minimalni interval preklapanja ima barem k poklopljenih znakova. U praksi bi razuman pristup bio izravno konstruirati rijetko sufiksno polje i zaista ostvariti tu uštedu na memoriji.

Kada su pronađena maksimalna egzaktna preklapanja, bi slijedilo potražiti preklapanja koja daju sličan podatak o potencijalnoj poziciji upitnog niza u referentnom i izračunati izglednost raznih poravnanja.

5. Literatura

[1] Caenorhabditis elegans genome. http://www.ensembl.org/Caenorhabditis_elegans/Info/Index.

[2] Domazet-Lošo, M. Predavanje sufiksna polja, bioinformatika. 'http://www.fer.unizg.hr/_download/repository/Bioinformatika_-_6._predavanje\%5B1\%5D.pdf, Studeni 2014. Slide 21.

[3] Kasai, T., Lee, G., Arimura, H., Arikawa, S., i Park, K. Linear-time longest-common-prefix computation in suffix arrays and its applications. In Combinatorial Pattern Matching, 12th Annual Symposium, CPM 2001 Jerusalem, Israel, July 1–4, 2001 Proceedings, Springer Berlin Heidelberg, July 2001.

[4] Khan, Z., Bloom, S.B., Kruglyak, L., i Singh, M. A practical algorithm for finding maximal exact matches in large sequence datasets using sparse suffix arrays. http://bioinformatics.oxfordjournals.org/content/25/13/1609.long, Travanj 2009.

[5] Nong, G., Zhang, S., i Chan, W.H. Two efficient algorithms for linear time suffix array construction. *IEEE Transactions on Computers*, Vol. 60, No. 10, Listopad 2011. str. 91-110.

6. Sažetak

Ostvarili smo implementaciju algoritma koji pronalazi maksimalna egzaktna preklapanja u programskom jeziku C++ koristeći rijetka sufiksna polja, te implementaciju algoritma za konstrukciju sufiksnog polja SA-IS. Performanse algoritma su testirane na genomu crva Caenorhabditis Elegans koji se sastoji od oko 100 milijuna znakova. Pokazalo se da je moguće ekonomično smanjiti zauzeće memorije korištenjem rijetkih sufiksnih polja uz prihvatljivo povećanje vremena izvođenja.