Algoritmi u poravnavanju očitanja

Vukoša, Zorana

Undergraduate thesis / Završni rad

2017

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **University of Zagreb, Faculty of Science / Sveučilište u Zagrebu, Prirodoslovno-matematički fakultet**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:217:508446

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2024-03-28**



Repository / Repozitorij:

Repository of the Faculty of Science - University of Zagreb







SVEUČILIŠTE U ZAGREBU

PRIRODOSLOVNO-MATEMATIČKI FAKULTET

BIOLOŠKI ODSJEK

SEMINARSKI RAD

Algoritmi u poravnavanju očitanja

Algorithms in Read Alignment

Zorana Vukoša

Preddiplomski studij Molekularne biologije

Mentor: dr. sc. Goran Igaly

Zagreb, 2017

Sadržaj

1. Uvod 2
2. Sekvenciranje nove generacije (NGS) 3
3. Human Genome Project 4
4. Poravnavanje očitanja 5
5. Algoritmi za pretraživanje niza 5
5.1. Procjena algoritama 5
5.2. Online algoritmi 6
5.3. Offline algoritmi 10
6. Približno uparivanje nizova 13
6.1. Pigeonhole lema 13
6.2. Q-gram lema 13
7. Literatura 15
8. Sažetak 17
9. Summary 17

1. Uvod

Deoksiribonukleinska kiselina (DNA), sastavljena od četiriju baza (adenina, timina, gvanina i citozina), je molekula koja prenosi genetske informacije o rastu, razvoju, djelovanju i reproduciranju svih živih organizama i mnogih virusa. Uzimajući to u obzir, jasno je zašto znanstvenici ulažu toliko truda u razvijanje i usavršavanje raznih pristupa i tehnika za otkrivanje sekvence dvostruke spiralne zavojnice.

Prve dvije metode predstavljene 1977. su *metoda lančane terminacije*, koju je osmislio Frederick Sanger i *Maxam-Gilbertov pristup*, koji je odbačen zbog toksičnosti i tehničke zahtjevnosti. Sangerova metoda je točna i može očitati do 1000 baznih parova, ali je prespora i preskupa. Sekvenciranje nove generacije (*Next Generation Sequencing* ili NGS) je termin koji opisuje tehnologiju sekvenciranja DNA koja je doprinijela genomskom istraživanju zbog povećane brzine i učinkovitosti, koji su za posljedicu imali smanjivanje troška sekvenciranja. (Mardis, 2006, 2009; Schuster, 2008; Shendure and Ji, 2008; Ng et al., 2009; Tucker et al., 2009; Metzker, 2010). NGS, kao metoda koja pruža veliki broj kratkih očitanja koja trebaju biti obrađena, otvara mnoge izazove za bioinformatiku. Razvoj NGS-a znatno utječe na razvoj genomike, iz koje proističu mnoga velika dostignuća o povezanosti gena, broju kopija DNA i analizi genske ekspresije. Koristi od NGS sekvenciranja neće biti u potpunosti primjećene i cijenjene sve dok ne dođe do razvoja visoko-učinkovitih računala i do uznapredovanja bioinformatike (Zhang et al., 2011). Kako se algoritmi u bioiformatici budu razvijali i prilagođavali potrebama NGS-a, tako će se pronalaziti sve više i više odgovora na pitanja skrivena u DNA molekuli.

2. Sekvenciranje nove generacije (NGS)

Visoko učinkovite tehnologije nove generacije postale su dostupne početkom dvadeset i prvog stoljeća (Barba, 2014). Sekvenciranja kratkih očitanja NGS-a se svrstavaju u dvije općenite kategorije: *sekvenciranje ligacijom* (Sequencing by ligation ili SBL) i *sekvenciranje sintetiziranjem* (Sequencing by synthesis ili SBS). Oba pristupa koriste fragmentaciju molekule DNA i amplifikaciju tih dijelova na čvrstoj podlozi, zbog čega postoji milijun pojedinačnih SBL i/ili SBS centara sa svojim kloniranim DNA predloškom.

SBL pristupi koriste probnu sekvencu koja je je vezana za fluorescirajuću molekulu. Takva sekvenca hibridizira s fragmentom DNA te se ligira za susjedni oligonukleotid za skeniranje. Emisijski spektar fluora otkriva identitet baze koja je komplementarna specifičnim mjestima unutar probne sekvence. Platforme koje sekvenciraju SBL pristupom su *SOLiD* i *Complete Genomics*.

SBS pristupi koriste polimerazu i signal, poput fluorescirajuće molekule ili promjene u ionskoj gustoći koje prijavljuju stapanje nukleotida u elongiranu nit. Platforme koje spadaju pod SBS pristupe su *Illumina, 454* i *Ion Torrent*.

3. Human Genome Project

Imajući na umu da sekvenciranje kratkih fragmenata pruža podatke koji su u odnosu na veličinu ljudskoga genoma (83,234.83 Mb) jako kratki, možemo zaključiti kako je komplicirano provoditi analizu podataka sekvenciranja. Srećom, visokokvalitetni referentni genomi za mnogo modelnih organizama su već prisutni što čini poravnanje očitanja sekvenciranja praktičnijim pristupom naspram analize podataka sekvenciranja koje nalazimo u sekvenciranju nove generacije. (Mardis, 2013).

Human Genome Project, započet 1990., službeno je završen 2003. Sastojao se od dvije faze; u prvoj, takozvanoj *shotgun fazi*, ljudski genom je fragmentiran te su ti fragmenti podijeljeni na manje preklapajuće segmente. Tada su ti segmenti klonirani, njihovi klonovi sekvencirani te podatci obrađeni kako bi se stvorila *contig* sekvenca. Druga faza projekta, zvana *završna faza*, se sastojala u ispunjavanja praznina i razrješenju sekvenci DNA u dvoznačnim područjima koji nisu dobiveni u *shotgun fazi*.

Ovaj pristup se također naziva metoda hijerarhijskog *shotguna* jer su članovi tima sistematično stvorili preklapajuće klonove mapirane na svaki pojedinačni ljudski kromosom koji su individualno sekvencirani korištenjem *shotgun pristupa*. Fragmenti DNA su mapirani u kromosomska područja skeniranjem STS-ova. Svaki od klonova iz knjižnice sadrži otisak DNA fragmenta koji se može usporediti s otiscima svih ostalih klonova iz knjižnice s ciljem nalaženja preklapajućih klonova. *Fluorescencijska in situ hibridizacija* (FISH) je također korištena u mapiranju klonova iz knjižnice u određena kromoskomska područja (Chial, 2008.). Primijetimo da je ovaj pristup različit od pristupa korištenih u NGS sekvenciranju zbog toga što je u NGS-u, točna pozicija određena samo u koraku poravnanja očitanja korištenjem različitih algoritama.

4. Poravnavanje očitanja

Poravnavanje očitanja je korak u *sekvenciranju nove generacije* koji pomaže u mapiranju svih očitanja na specifična mjesta unutar genoma. Pošto redoslijed baza u DNA molekuli može biti predstavljen kao niz, referentna sekvenca se smatra kao tekst a očitanja kao uzorak koji treba biti poravnan naspram teksta. Algoritmi koji provode pretraživanje niza nastali su mnogo ranije, ali su sada pronašli novu primjenu unutar bioinformatike.

5. Algoritmi za pretraživanje niza

U računalnim znanostima, algoritmi za pretraživanje niza, ponekad zvani *algoritmi za uklapanje nizova*, su važna vrsta algoritama koji pokušavaju pronaći sva mjesta gdje se pojavljuje niz dužine *m* (zvane *uzorak* ili *šifra*) unutar teksta duljine *n* znakova.

Niz *S* je konačna sekvenca znakova.
Znakovi su preuzeti iz *DNA abecede* ($\Sigma = \{A,C,G,T\}$).
Prva pozicija se naziva *nultom pozicijom*.
Podniz od *S* je niz koji se pojavljuje unutar niza *S*.
Prefiks od *S* je podniz kojim započinje niz *S*.
Sufiks je podniz na kraju niza *S*.

Algoritmi se mogu klasificirati na temelju navedenog predobradnog kriterija. *Online algoritmi* su algoritmi koji ne koriste predobradu teksta. Algoritmi koji koriste predobradu teksta nazivamo *offline algoritmima*.

5.1. Procjena algoritama

Prilikom procjene algoritma promatraju se sljedeće stavke:

1. Najgora moguća algoritamska kompleksnost
2. Očekivana algoritamska kompleksnost
3. Prostorna kompleksnost

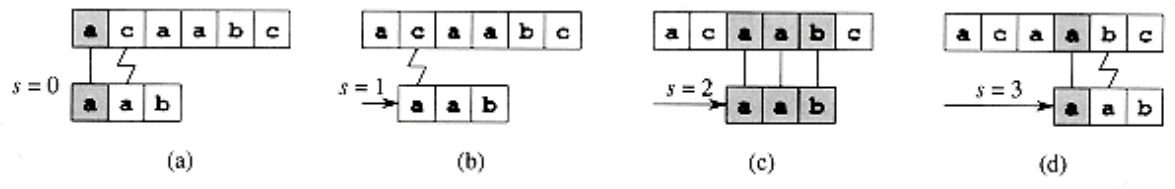
Najgora moguća kompleksnost je tradicionalna mjera algoritamske efikasnosti koja prikazuje kako bi algoritam djelovao uz dani uzorak i sekvencu koja zahtjeva provjeru najvećeg broja znakova. Očekivana kompleksnost prikazuje kako se određeni algoritam obično ponaša u radu. Prostorna kompleksnost je mjera za prostor koji algoritam mora zauzimati tokom izvedbe; preferiraju se algoritmi koji zauzimaju prostor koji linearno ovisi o veličini problema.

5.2. Online algoritmi

Online algoritmi čine kategoriju algoritama koji ne predobrađuju tekst (referentnu sekvencu), no ovisno o tipu algoritma, ponekad predobrađuju uzorak.

5.2.1. Naivno uklapanje nizova

Algoritam naivnog uklapanja je jednostavan algoritam koji djeluje prebacivanjem prozora duljine *m* na tekst (duljine *n*). Ovaj algoritam eksplicitno pokušava svaki mogući pomak *s* (Slika 1). Vremenska kompleksnost u najgorem slučaju je O((n – m)m), za koje je O(mn), pod pretpostavkom da je *m* "relativno malen" (m < n/2) u usporedbi sa *n*. Očekivana kompleksnost je O(n+m). *Algoritam naivnog uklapanja* je neefikasan je su informacije dobivene o tekstu za jednu vrijednost *s* sasvim ignorirane u usporedbi s drugim vrijednostima *s*.



Slika 1. Rad algoritma naivnog uklapanja za uzorak P=aab i tekst T=acaabc. U svakom prikazu okomite linije pokazuju mjesta gdje su se tekst i uzorak podudarili, a izlomljena linija povezuje prvi znak koji nije podudaran s tekstom. (c) Uzorak P se pojavljuje u tekstu na poziciji s = 2. http://staff.ustc.edu.cn/~csli/graduate/algorithms/book6/chap34.htm, The Chen Group, University of Science and Technology of China.

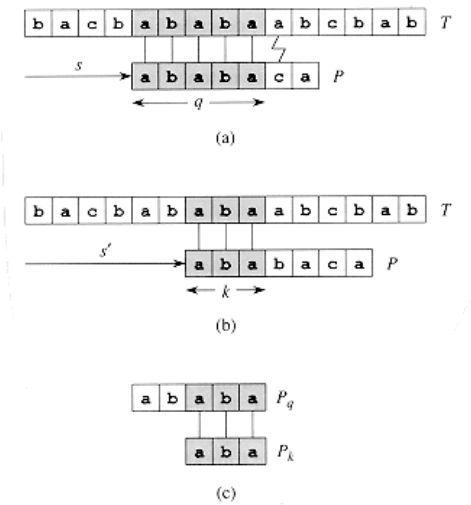
5.2.2. Knuth–Morris–Prattov algoritam

Knuth-Morris-Pratov algoritam (KMP) je algoritam koji se sastoji od dvije faze

1. Pretprocesna faza
2. Faza pretraživanja

U pretprocesnoj fazi, koja se također naziva **prefiksnom funkcijom**, konstruiran je cijeli niz koji pokazuje najdulje prave prefikse koji su istovremeno i sufiksi. U biti, P se miče u odnosu na sebe samoga i određuje se kada njegovi prefiksi postaju sufiksi u pomaku. Takav niz prikazuje broj znakova koji se treba preskočiti u fazi pretraživanja.

U fazi pretraživanja, KMP pokušava uštedjeti vrijeme preskakanjem unutarnji i vanjskih ponavljanja. U vanjskim ponavljanjima, cilj je preskočiti ponavljanja za kojih nema para. Algoritam neće pokušati sve moguće pozicije jer iz niza napravljenog u pretprocesu i trenutnog djelomičnog uparivanja, zna sljedeću moguću poziciju gdje će se uzorak i tekst podudariti. U unutarnjim odstupanjima, cilj je izbjeći testiranje znakova koji su već dokazani kao podudarni. Ako postoji netrivijalno preklapanje sa zadnjim parcijalnim parom, testiranje broja znakova jednakih duljini preklapanja se može izbjeći (Slika 2).



Slika 2. Prefiksna funkcija π. (a) Uzorak P = ababaca je poravnat s tekstom T tako da su prvih q = 5 znakova podudarni. Znakovi koji se podudaraju su prikazani osjenčano i povezani su okomitim linijama. (b) Koristeći znanje o 5 podudarenih znakova, možemo zaključiti da je pomak od s + 1 nevažeći, a pomak od s + 2 važeći jer je dosljedan svemu što znamo o tekstu. (c) Korisne informacije za takve zaključke se izračunaju ranije usporedbom uzorka sa samim sobom. Najdulji prefiks od P koji je također i sufiks od P5 je P3. Ta informacija je predizračunata i pospremljena u tablicu π[5] = 3. S obzirom da su q znakovi podudarni s tekstom sljedeći potencijalni pomak je s' = s + (q - π[q]). Kako se uzorak i tekst preklapaju u 3 znaka, nije potrebno testirati prva tri znaka.
(http://staff.ustc.edu.cn/~csli/graduate/algorithms/book6/chap34.htm).

Knuth-Morris-Prattov algoritam obrađuje u vremenu O(m + n). Vrijeme potrebno za pretprocesiranje i potragu su linearni naspram veličini uzorka i teksta. Prostorni zahtjev je O(m) (Sleit et al., 2007).

5.2.3. Boyer-Mooreov algoritam

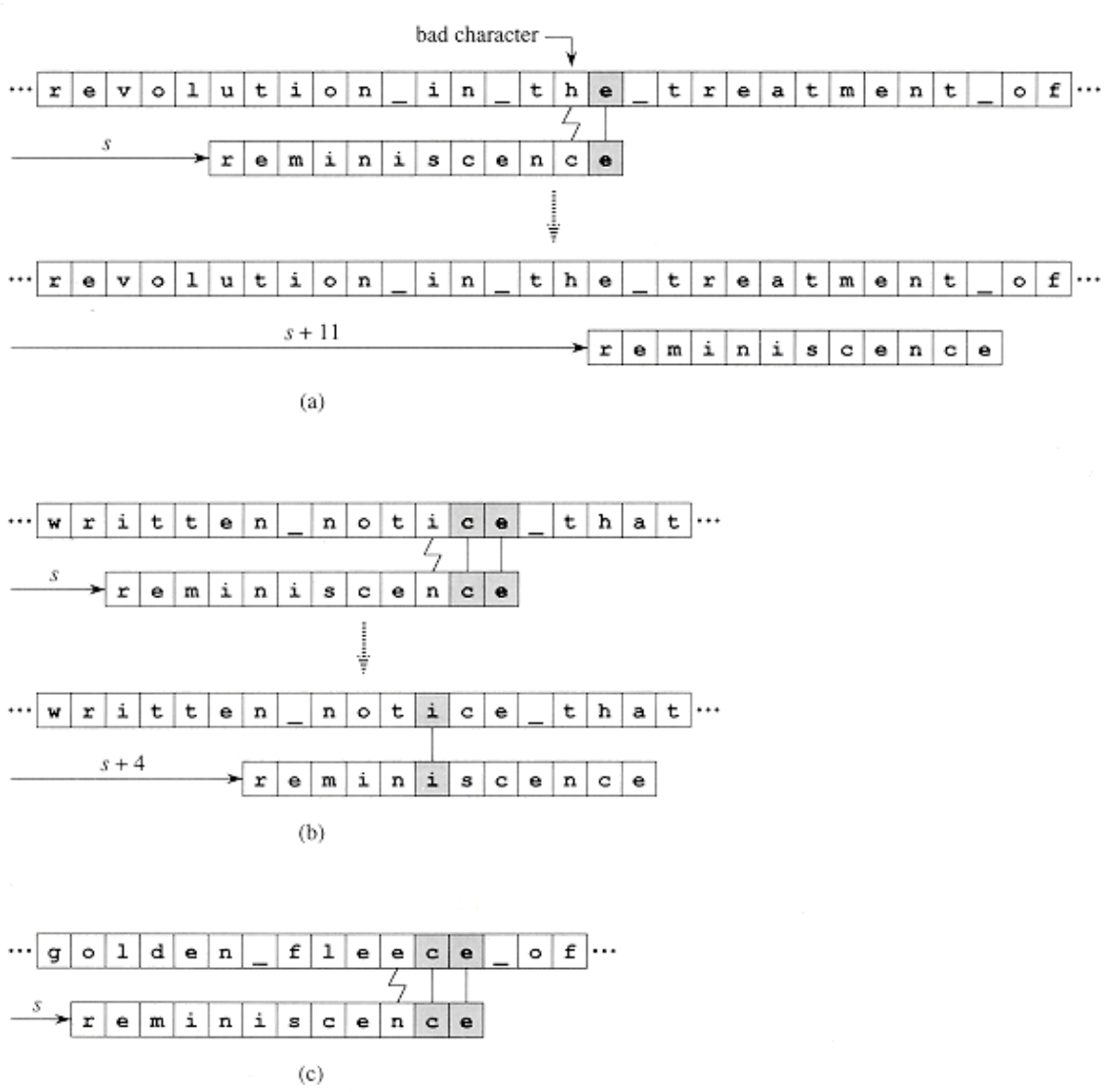
Boyer-Mooreov algoritam je dobro dizajniran za sljedeće vrste problema:

1) Veliki uzorak P
2) Velika abeceda Σ.

Iako prva situacija nije neuobičajena u bioinformatici, tradicionalna nukleotidna abeceda Σ ={A, T, C, G} je malena te Boyer-Mooreov algoritam možda neće pružiti najbolje prednosti. Abeceda aminokiselina se sastoji od 20 znakova, dakle za ovu abecedu, Boyer-Moore je dobar izbor. Algoritam se temelji na tri ključne ideje: skeniranje s desna na lijevo, zakon loših znakova te zakon dobrih sufiksa. (Boyer, Moore, 1977).

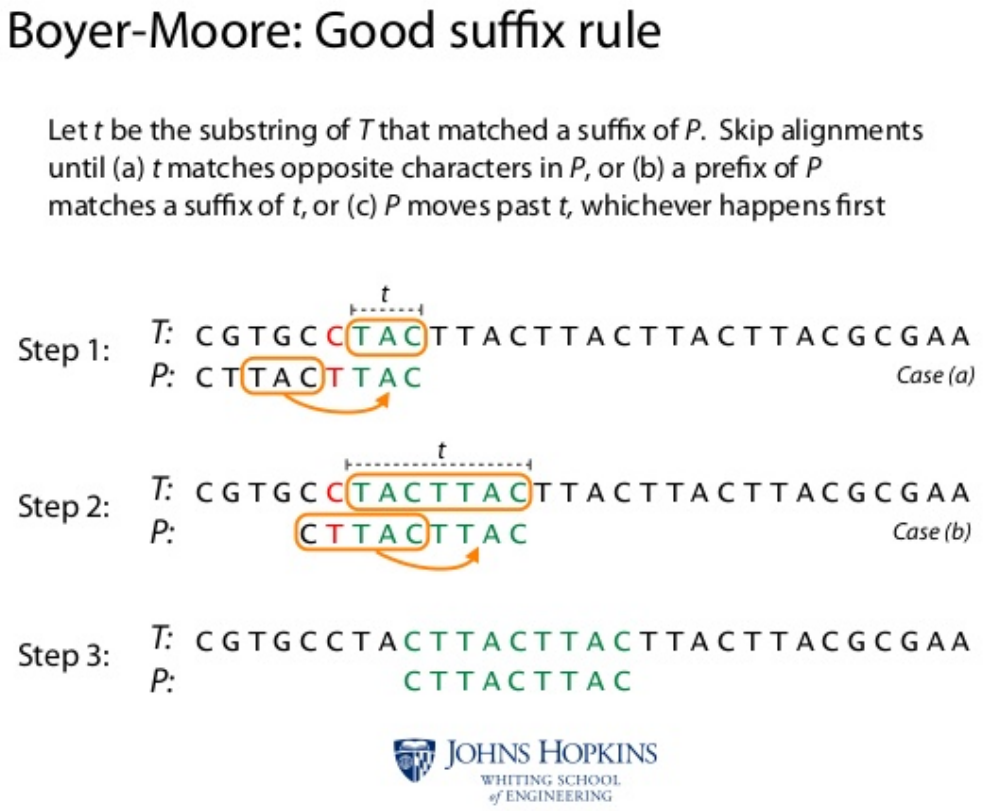
Skeniranjem s desna na lijevo je moguće ponekad dobiti više informacija od skeniranja s lijeva na desno. Par s desna na lijevo s dodatnim znanjem o P dozvoljava pomicanje P puno dalje, npr. ako je uzorak testiran sa znakovima u nizu koji se ne pojavljuju u nizu – to znači da se cijeli uzorak može prebaciti preko tog znaka.

Zakon loših znakova je heuristika koja predlaže količinu kojom se *s* može sigurno povećati bez izostavljanja opravdanih pomaka. Djelovanje je prikazano na Slici 3: ako dođe do nepodudarnosti u znakovima, traži se najdesnija pojava lošeg znaka u uzorku prema lijevo od pozicije te se uzorak pomiče dok krivi par ne postanje podudaran. Ako uzorak uopće nema takav znak, onda se uzorak pomiče za duljinu *m*. Ako se loš znak pojavi desno od krivog uparenja, onda heuristika lošeg znaka predlaže smanjenje *s*. Ovu preporuku algoritam Boyer-Moore ignorira pošto heuristika dobrog sufiksa predlaže pomak udesno u svim slučajevima. Ovaj pomak može biti predviđen za svako slovo i spremljen unutar tablice. Ova tablica se naziva **tablica prijenosa loših znakova** i napravljena je u pretprocesnoj fazi.



Slika 3. Zakon loših znakova. (a) Loš znak h se ne pojavljuje nigdje u uzorku, tako da se cijeli uzorak može pomaknuti za m znakova dok se prođe taj loš znak. (b) Pomak do najdesnije pojave lošeg znaka u uzorku lijevo od pozicije koja je dala nepodudarnost je sljedeća važeća pozicija. (c) Najdesnija pojava lošeg znaka se nalazi na poziciji desno od nepodudarnosti. Zakon lošeg znaka predlaže negativni pomak, koji se ignorira. (http://staff.ustc.edu.cn/~csli/graduate/algorithms/book6/chap34.htm)

Pravilo dobrog sufiksa je heuristika koja predlaže pomak kada imamo sufiks koji se podudara s tekstom, a iza kojeg slijedi nepodudarnost. Ako postoji još jedno pojavljivanje iste sekvence u uzorku poput one podudarne s tekstom, uzorak se pomiče na način da poravnanje dobrog sufiksa uz tekst bude sačuvano. Ako ne postoji takav sufiks u uzorku, ali prefiks uzorka je "sufiks sufiksa", ona se uzorak pomiče na to mjesto (Slika 4). Ovaj pomak se može predvidjeti za svako slovo i pohraniti u tablicu. Ova tablica se naziva **tablica pomaka dobrog sufiksa** i napravljena je u pretprocesnoj fazi.



Slika 4. Boyer-Mooreov algoritam: pravilo dobrog sufiksa. a) U ovom slučaju uzorak je pomaknut sve dok se "dobar sufiks" "TAC" ne podudari s tekstom. b) U ovom slučaju prefiks uzorka je "sufiks sufiksa", tako da je prefiks "CTTAC" pomaknut dok se ne poravna s tom pozicijom u tekstu. c) Pronađena je podudarnost (Langmead, Johns Hopkins official textbook).

Pravila djeluju neovisno, ali Boyer-Mooreov algoritam odabire ono pravilo koje predlaže veći pomak.

U pretprocesnoj fazi, Booyer-Mooreov algoritam zauzima O(m + |Σ|), (Σ = {A,C,G,T}), a u pretraživačkoj fazi O(mn) u najgorem slučaju. (Sleit et al., 2007). Ako se uzorak ne pojavljuje u tekstu, zauzima O(m+n) u najgorem slučaju. Prosječno trajanje se smanjuje kako se uzorak povećava, kako bi bio pogodniji za dulja očitanja.

5.3. Offline algoritmi

Offline algoritmi se odnose na algoritme koji pretprocesiraju tekst koristeći **strategije indeksiranja**. Ovakvim algoritmi zahtijevaju da je velika podatkovna struktura spremljena u memoriji. Mnogo algoritama brzih poravnavanja stvaraju pomoćne podatkovne strukture, zvane indeksi, za sekvence očitanja ili referentne sekvence, ili ponekad oboje. Uzimajući u

obzir stavke indeksa, algoritmi poravnavanja mogu biti svrstani u tri kategorije: algoritmi temeljeni na hash tablicama, algoritmi temeljeni na stablima sufiksa i algoritmi temeljeni na svrstavanju spajanja. Treća kategorija se sastoji samo od *Slidera* te njegovog potomka, *SlideraII*. Ovaj pregled će se fokusirati na prve dvije kategorije. (Li, Homer, 2010).

5.3.1. Algoritmi temeljeni na hash tablicama

Algoritmi temeljeni na indeksiranju po hash tablicama se povijesno mogu pratiti od početka BLAST-a (Altschul et al., 1990). Namjera hash tablice je da podjeli veliki niz, u ovom slučaju – referentni genom, u svaki mogući k-mer, npr. podniz dužine *k* i da ih stavi u poredanu strukturu koja ima mapiranu poziciju k-mera s obzirom na taj niz. U fazi pretraživanja, koristi se *seed-and-extend* strategija što znači da, nakon pronalaženja točnih parova u hash tablici zvanima *sjeme (seed)*, potvrda znakova koji prate sjeme se također provjerava. Hash tablica pretražuje binarnim pregledom koji djeluje na principu podjele tablice na dvije polovice i provjere je li pogodak prije ili poslije te polovice. U slučaju da ne dođe do pogotka, nastavlja se s binarnim pregledom koji dijeli tablicu dok ne pronađe pogodak.

5.3.2. Algoritmi temeljeni na pokušajima sufiksa/prefiksa

Algoritmi bazirani na hash tablici daju loše rezultate kada očitanja padaju u ponavljajuća područja pošto se treba pregledati mnogo sjemena. Pametne podatkovne strukture su odnedavno korištene sa svrhom rješavanja ovog problema: stablo sufiksa, niz sufiksa te FM indeksi. (Schbath et al, 2012). Ove strukture sažimaju područja ponavljanja kako bi ona bila pregledana samo jednom jer se te identične kopije preklapaju na pojedinom putu u stablu.

Stablo sufiksa je struktura koja prikazuje sve moguće sufikse koji završavaju znakom $ kako bi se osiguralo da nijedan sufiks nije prefiks drugog sufiksa. Svaki član ima barem dvije grane koji ne počinju istim slovom. Ako uzorak može biti poslovkan od korijena do člana – onda postoji u referentnom genomu. Pretraga strukture podataka se odvija u vremenu O(m), gdje je *m* duljina očitanja. Stabla sufiksa postižu linearni prostor dok dopuštaju vremenski linearno pretraživanje, no još uvijek zahtijevaju 12-17 bita po nukleotidu (Kurtz et al, 2004), čineći ga nepraktičnim za pohranu stabla sufiksa ljudskog genoma u memoriji. (Li, Homer, 2010).

Niz sufiksa je leksikografski raspoređen niz sufiksa niza *s*. Kako bismo ih pohranili, nije potrebno zadržati vektor niza; dovoljno je samo zadržati indeks svakog sufiksa u raspoređenome nizu (Vladu, Negruseri, 2005). Količina memorije iskorištene tijekom implementacije niza sufiksa je 3 do 5 puta manja od količine potrebne za stablo sufiksa, ali je još uvijek zahtijeva 16 puta više memorije od samoga genoma. Kompleksnost najgoreg slučaja iznosi O(n log n).

FM indeks je sabijeni indeks sastavljen od tri strukture podataka:

1. Burrows-Wheelerov tekst *s*
2. tablica C koja pohranjuje ukupan broj pojavljivanja znakova koji su leksikografski manji od slova c ($c \in \Sigma$, $\Sigma = \{A, T, C, G\}$)
3. struktura *occ* koja prikazuje broj pojavljivanja znakova c u prefiksu BW teksta ($c \in \Sigma$, $\Sigma = \{A, T, C, G\}$)

FM indeks može biti konstruiran u linearnom vremenu. Prostorni zahtjev je O(n).

6. Približno uparivanje nizova

Postoji mnogo razlika između referentnog genoma i genoma čiju sekvencu želimo razriješiti. Prvi razlog je u tome što je svaka DNA sekvenca jedinstvena. Drugi razlog nalazimo u sekvencijskoj pogrešci. Uzimajući to u obzir, jasno je zašto je važno koristiti približno uparivanje nizova u poravnavanju očitanja. Ranije pojašnjeni algoritmi također mogu biti prilagođeni kako bi obavljali približno uparivanje nizova. Poseban pristup korišten u približnom uparivanju je filtriranje.

Filtriranje je popularan način određivanja približnog para i uobičajeno koristi jednu od dvije leme: *pigeonhole* lemu ili *q-gram* lemu. Obje određuju minimalnu duljinu ili broj točnih q-grama koje očitanje s određenim brojem pogrešaka dijeli s referentnom sekvencom (Reinert et al, 2015).

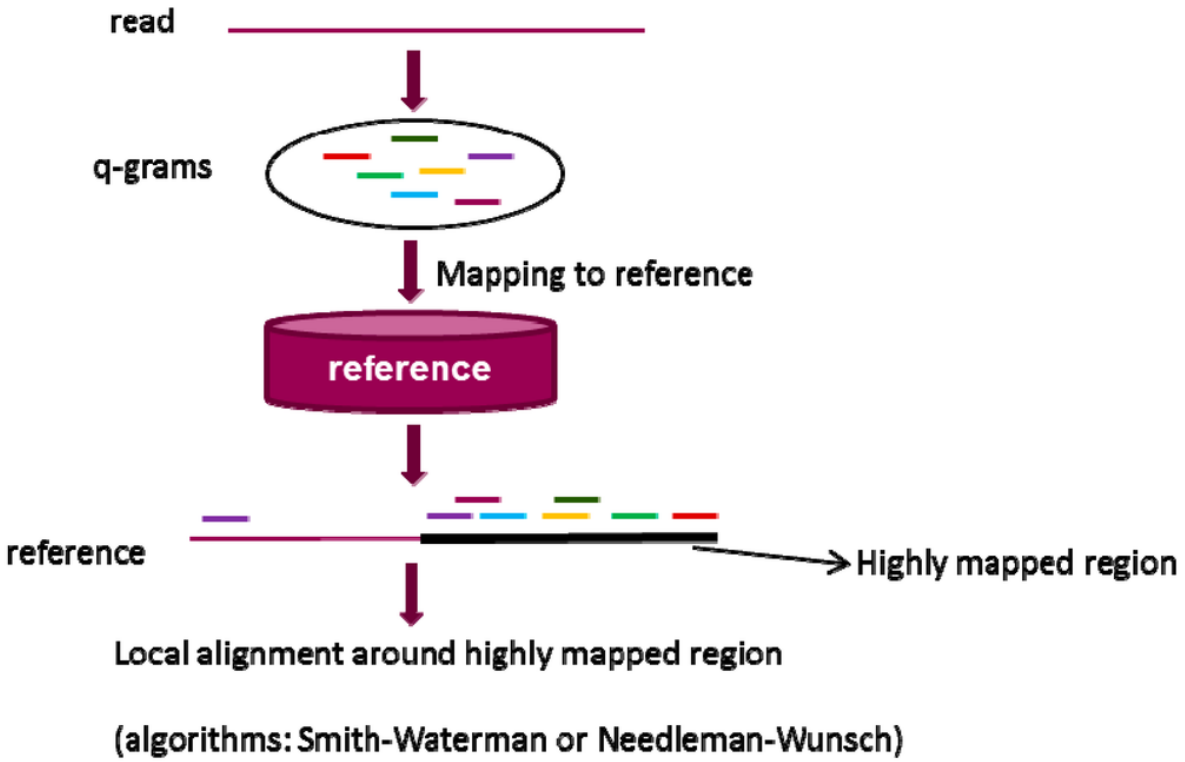
6.1. Pigeonhole lema

U matematici, **princip pigeonhole** navodi da ako *n* stvari stavimo u *m* posuda, sa n>m>0, tada barem jedna posuda mora sadržavati barem više od jedne stvari. Princip se koristi u mapiranju očitanja na način, da ako očitanje sadrži k pogrešaka i podijelimo očitanje u k+1 nepreklapajućih dijelova, tada se barem jedan od njih pojavljuje bez pogreške u približnom paru. (Baeza-Yates, Navarro, 1999). U svojem najjednostavnijem obliku, očitanje se dijeli na k+1 komada (zvanih sjemenjem) istih duljina. Neki algoritmi dijele očitanje na k+2 sjemena i traže barem dva ista sjemena, a neki dijele na manje dijelove, ali toleriraju jednu ili dvije pogreške unutar njih. (Langmead, Salzberg, 2012). U fazi pretraživanja, prijavljuje se pojavljivanje jednakog para sjemenke i referentnog genoma i okolna sjemena se razmatraju kao kandidati za približan par.

6.2. Q-gram lema

Q-gram lema dijeli očitanje duljine *n* na sve moguće preklapajuće sjemenke duljine *q* (n – q + 1). Postavlja donju granicu za broj q-grama koji očitanje s k pogrešaka dijeli sa referencom. Ovaj broj je određen najgorim slučajem, gdje su te pogreške distribuirane jednako cijelom duljinom očitanja *n* i mijenjaju najviše kq od n-q+1 preklapajućih q-grama.

Broj nedirnutih q-grama je n-(k+1)q+1 tako kad filtar pronađe takvu regiju on provjerava okolno područje i uzima ga kao potencijalnog kandidata za par. Cijela shema je prikazana na Slici 5.



Slika 5. Q-gram filtriranje. Visoko mapirana regija po donjoj granici je odabrana kao kandidat za približni par, nakon čega je provjereno i okruženje.

7. Literatura

Altschul, S. (1990). Basic Local Alignment Search Tool. *Journal of Molecular Biology*, 215(3), pp.403-410.

Barba, M., Czosnek, H. and Hadidi, A. (2014). Historical Perspective, Development and Applications of Next-Generation Sequencing in Plant Virology. *Viruses*, 6(1), pp.106-136.

Boyer, R. and Moore, J. (1977). A fast string searching algorithm. *Communications of the ACM*, 20(10), pp.762-772.

Chial, H. (2008) DNA sequencing technologies key to the Human Genome Project. *Nature Education* 1(1):219

Langmead, Johns Hopkins official textbook

Langmead, B. and Salzberg, S. (2012). Fast gapped-read alignment with Bowtie 2. *Nature Methods*, 9(4), pp.357-359.

Li, H. and Homer, N. (2010). A survey of sequence alignment algorithms for next-generation sequencing. *Briefings in Bioinformatics*, 11(5), pp.473-483.

Mardis, E. (2013). Next-Generation Sequencing Platforms. *Annual Review of Analytical Chemistry*, 6(1), pp.287-303.

Reinert, K., Langmead, B., Weese, D. and Evers, D. (2015). Alignment of Next-Generation Sequencing Reads. *Annual Review of Genomics and Human Genetics*, 16(1), pp.133-151.

Schbath, S., Martin, V., Zytnicki, M., Fayolle, J., Loux, V. and Gibrat, J. (2012). Mapping Reads on a Genomic Sequence: An Algorithmic Overview and a Practical Comparative Analysis. *Journal of Computational Biology*, 19(6), pp.796-813.

Sleit, A., AlMobaidee, W., Al-Areqi, S. and Yahya, A. (2007). A Dynamic Object Fragmentation and Replication Algorithm In Distributed Database Systems. *American Journal of Applied Sciences*, 4(8), pp.613-618.

http://staff.ustc.edu.cn/~csli/graduate/algorithms/book6/chap34.htm , The Chen Group, University of Science and Technology of China

www.cs.otago.ac.nz/cosc348/alignments/Lecture04_StringSearch.pdf

8. Sažetak

Poravnanje očitanja je korak u sekvenciranju nove generacije gdje se sva očitanja dobivena s platformi za sekvenciranje uspoređuju s referentnom sekvencom kako bi se dobila željena sekvenca. Razni algoritmi su korišteni kako bi olakšali uređivanje velikih brojeva očitavanja na njihovim točnim pozicijama u odnosu na reference. Algoritmi navedeni u ovom radu su temeljni alati za poravnanje očitanja. Razvoj ideje o ubrzanju i poboljšanju algoritma je prikazan od najosnovnijeg algoritma naivnog uparivanja nizova do *offline algoritama* koji pretprocesiraju tekst jednom te onda koriste indeks stvoren u toj fazi kako bi pronašli očitanja u fazi pretraživanja. Algoritmi će se unapređivati zajedno s platformama za sekvenciranje te u budućnosti možemo očekivati brže i točnije načine za sekvenciranje željenih genoma.

9. Summary

Read alignment is a step in Next generation sequencing where all the reads obtained from sequencing platforms are compared to the reference sequence in order to get the wanted sequence. Various algorithms are used to facilitate ordering of the large number of reads to their exact position in respect to the reference. Algorithms mentioned in this thesis are basis of all the read alignment tools. Evolution of the idea on how to speed up and improve algorithm is shown from the most basic Naive string matching algorithm to offline algorithms that preprocess text once and then use the index created in that phase to match the reads in the searching phase. Algorithms are going to evolve together with the sequencing platforms, so we expect that faster and more accurate ways for sequencing the desired genomes will be found.