SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 6141

Gornja granica u sastavljanju genoma

Luka Požega

Zagreb, lipanj 2019.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA
ODBOR ZA ZAVRŠNI RAD MODULA

Zagreb, 13. ožujka 2019.

ZAVRŠNI ZADATAK br. 6141

Pristupnik: **Luka Požega (0036499279)**
Studij: Računarstvo
Modul: Računarska znanost

Zadatak: **Gornja granica u sastavljanju genoma**

Opis zadatka:

Sastavljanje genoma jedan je od najsloženijih zadataka u području bioinformatike i računalne biologije. S obzirom da je problem NP težak, koriste se brojne heurističke metode. Isto tako unaprijed je nemoguće znati jesu li sekvencirani fragmenti sa svih djelova genoma i s kojom dubinom. Cilj ovoga završnog rada je, za poznati genom i sekvencirana očitanja, utvrditi je li iz tih očitanja moguće sastaviti potpun genom. Kao prvi korak potrebno je koristiti mapiranje očitanje na poznati genom, alatima kao što je Minimap2. Nakon toga od mapiranih očitanja izgraditi graf i pronaći najbolji put kroz taj graf. Koristeći alat Red pronaći sve ponavljajuće regije u genomu. Napraviti analizu uspješnosti i usporediti rezultate s Quast XL metodom.
Programski kod je potrebno komentirati i pri pisanju pratiti neki od standardnih stilova. Kompletnu aplikaciju postaviti na repozitorij Github.
U svezi dobivanja detaljnih informacija obratiti se Robertu Vaseru, mag. ing.

Zadatak uručen pristupniku: 15. ožujka 2019.
Rok za predaju rada: 14. lipnja 2019.

Mentor:
Mile Šikić
Prof. dr. sc. Mile Šikić

Djelovođa:
Izv. prof. dr. sc. Tomislav Hrkać

Predsjednik odbora za završni rad modula:
Marko Čupić
Doc. dr. sc. Marko Čupić

Zahvalio bih se Mili Šikiću i Robertu Vaseru na prenesenom znanju i ukazanom strpljenju.

SADRŽAJ

1. Uvod

Bioinformatika je interdisciplinarna grana znanosti koja se bavi razvojem metoda i alata za razumijevanje opsežnih bioloških podataka dobivenih biološkim istraživanjima. Jedan od najsloženijih problema bioinformatike i računalne biologije je sastavljanje genoma.

Danas je nemoguće očitati genom u potpunosti. Iz tog razloga se višestruko čitaju fragmenti od 20 do 30000 baza, ovisno o korištenoj tehnologiji. Najčešća metoda očitavanja genoma je tzv. *shotgun sequencing*. Kako bi se evaluiralo sekvenciranje, potrebno je odrediti je li iz očitanja moguće sastaviti cijeli genom. Problem spada u NP klasu teških problema, stoga se koriste razne heurističke metode u njegovom rješavanju. Postoje dva načina sastavljanja genoma: jedan je bez poznavanja referentnog genoma te drugi koji je vođen referencom. Ovaj rad se bavi sastavljanjem genoma vođeno referencom. Prilikom tog načina se koristi referenca s kojom se uspoređuju očitanja te se ovisno o informaciji lokacije očitanja na genomu nastoji sastaviti genom. Jedan od poznatijih alata danas koji se bavi ovom problematikom je QUAST [1].

U radu se nalazi kratka biološka pozadina problema, korišteni algoritmi i strukture podataka, rezultati rada te njihova usporedba s drugim alatima.

2. Teorijski uvod

2.1. Sekvenciranje genoma

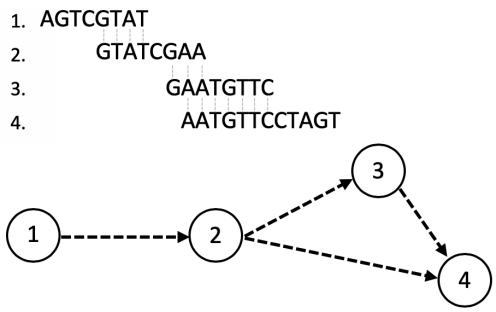
Deoksiribonukleinska kiselina (DNA) je nukleinska kiselina u obliku dvostruke uzvojnice. DNA sadrži genetske upute za specifični biološki razvoj staničnih oblika života i većine virusa [2]. Osnovna građevna jedinica joj je nukleotid, koji je pak građen od šećera zvanog deoksiriboza, fosfatne skupine i dušične baze. Kod DNA postoje četiri dušične baze: adenin (A), gvanin (G), citozin (C) i timin(T). Nukleotidi su poredani jedan do drugog tako da čine polimer. Odavde proizlazi zapis DNA kao niz znakova odnosno baza poput: 'ATGCTA'. Gen se nalazi na DNA, čini ga oko 1000 nukleotida. Genom je pak sav genetski materijal organizma. Konačno, kromosom je jedinica stanične jezgre koja se sastoji od DNA, RNA i histonskih i nehistonskih proteina.

Sekvenciranje genoma se svodi upravo na utvrđivanje redoslijeda nukleotida u genomu. Ima nekoliko metoda sekvenciranja genoma, a danas je najkorištenija *shotgun* metoda. Genom se prvo nekoliko puta umnoži, a zatim se svaki od duplikata razlama na male dijelove koji se potom očitavaju. Danas se ovo sekvenciranje radi uređajima 3. generacije u koje spadaju uređaji *PacBio* i *Oxford Nanopore*. Kod treće generacije se mogu čitati duži fragmenti, ali je zato greška povećana. *PacBio* uređaj omogućuje očitanja do 40 tisuća baza (40kbp) uz grešku do približno 15% [3].

2.2. Graf sastavljanja

Dva niza znakova (u našem slučaju baza) mogu se preklapati na različite načine. Ako niz n može biti mapiran na podniz niza m, tada kažemo da niz m sadrži niz n. Kada se kraj niza n mapira na početak niza m, tada kažemo da se nizovi m i n preklapaju. Ovaj slučaj ćemo dalje razmatrati te koristiti u kreiranju grafa. Čvorove grafa će predstavljati očitanja genoma, a usmjereni brid između dva čvora će postojati ako se ta dva očitanja preklapaju. Broj znakova u preklapanju će predstavljati duljinu brida. Jednostavan

primjer grafa preklapanja prikazuje **slika** 2.1.

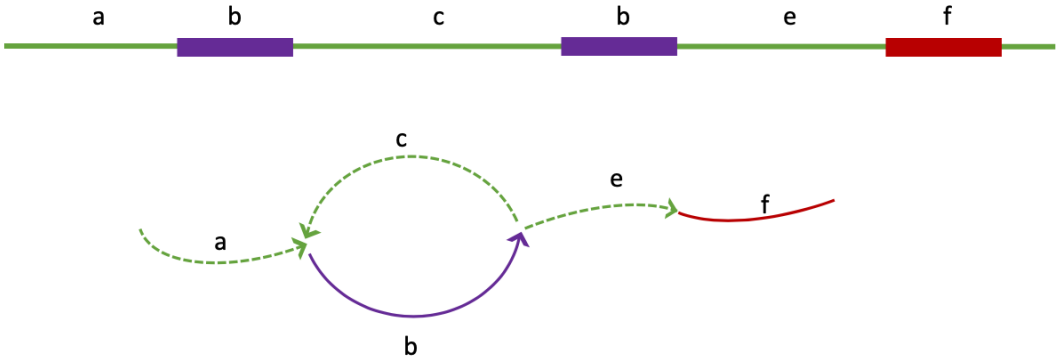


Slika 2.1: Jednostavan primjer grafa preklapanja

Možemo kreirati usmjereni graf $G = (V, E, l)$ koji ne sadrži višestruke bridove, gdje je V skup svih očitanja genoma, E su preklapanja očitanja, a l skup duljina preklapanja. Kako bi sastavljanje genoma pomoću ovog grafa bilo moguće mora biti zadovoljen jedan uvjet: niti jedno očitanje ne smije biti sadržano u drugome. Kraj grafa se definira čvor iz kojeg ne izlaze bridovi, u kojeg je usmjeren barem jedan brid. Početak je suprotno, čvor iz kojeg samo izlaze bridovi. Sastavljanje genoma nad ovako definiranim grafom je postupak u kojem je potrebno naći put od početka do kraja grafa, te će taj put predstavljati sastavljeni genom. Budući da ne mora postojati izlazni brid iz svakog čvora, tj. ne mora postojati očitanje kojim će se spojiti druga dva očitanja, moguće je da u grafu nastane više od jedne komponente. Genom tada neće biti moguće u potpunosti sastaviti, nego će biti sastavljena njegova dva ili više dijelova. Mjera koju uzimamo pri sastavljanju je pokrivenost, tj. koliki dio genoma je moguće sastaviti od jedne komponente. Za gornju granicu sastavljanja jednog kromosoma će se uzeti ona komponenta koja pokriva najveći dio kromosoma.

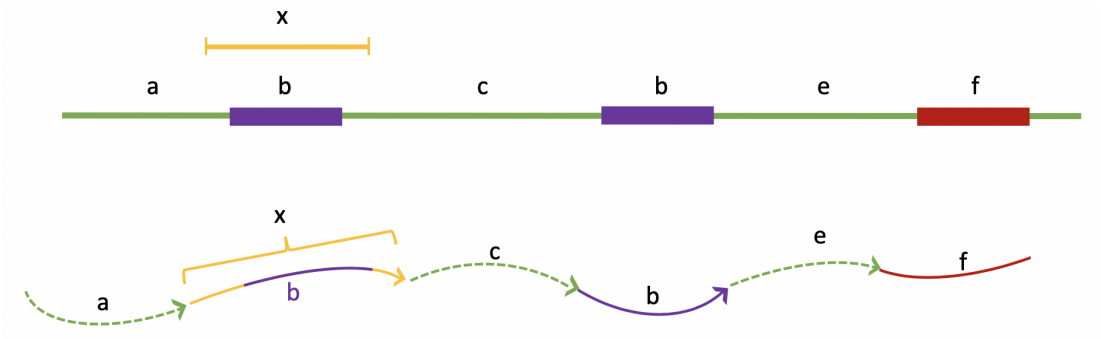
Problemi kod sastavljanja nastaju kada se u genomu pojavljuju regije jednakih nizova. U daljnjem tekst takve regije će se nazivati ponavljajuće regije. Postoje dvije vrste ponavljajućih regija: raspršene i tandem regije. Raspršene regije su ona ponavljanja koja se nalaze na različitim dijelovima genoma, tj. ne slijede jedna drugu. Tandem regije se nazivaju one regije koje unutar sebe imaju ponavljajući niz. Ponavljajuće regije u genomu će se na grafu odraziti kao ciklusi u kojima pravi put ne može biti određen. Stoga se početak ciklusa tretira kao prekid u sastavljanju genoma, a sastavljanje se nastavlja nakon ciklusa, baš kao da očitanja na tom dijelu nikad nije ni bilo. Iz teorije grafova, Eulerov put je put koji prolazi svim bridovima točno jednom. Kako bi sastavili genom mora postojati Eulerov put kroz graf sastavljanja. Ako postoje ponavljajuće regije, takav put ne postoji.

Rješenje ovog problema postiže se premošćivanjem. Potrebno je naći očitanje koje



Slika 2.2: Repetitivne regije u genomu i njihov prikaz u grafu

će se na genom mapirati prije ponavljajuće regije, a završiti poslije nje. Dakle, ako je ponavljajuća regija duljine l, kako bi ona bila premošćena mora postojati očitanje koje je duljine m, tako da je $m > l$ i da očitanje sadrži tu ponavljajuću regiju. Ako postoji takvo očitanje onda je moguće sa sigurnošću tvrditi koji put u grafu je potrebno odabrati u njegovom obilasku, kako je prikazano na **slici** 2.3. Nije potrebno premostiti sve ponavljajuće regije nego samo dok ne ostane najviše jedna neke vrste tj. ponavljajuća regija ne smije imati svog para drugdje na genomu koji nije premošten.



Slika 2.3: Premošćivanje ponavljajuće regije

2.3. Postojeća rješenja

2.3.1. QUAST-LG

QUAST v5 [1] ili QUATS-LG je alat razvijen 2018. godine u laboratoriju za algoritamsku biologiju u Sankt-Peterburgu. Prvenstveno je bio zamišljen kao alat koji će vrednovati rezultate koji drugi alati za sastavljanje daju. Od verzije 5, podržava i tako zvani *upper bound assembly*, tj. sastavljanje genoma na temelju predanih očitanja i

reference. Jednom kad su ti podaci obrađeni, rezultati se još jednom mapiraju na referencu te se određuje kvaliteta sastavljanja. Rezultati sastavljanja su kontizi (eng. *contig*) koji zapravo predstavljaju više očitanja sastavljenih u veću sekvencu na temelju njihovih pozicija u mapiranju. Nakon evaluiranja, alat ispisuje osnovnu statistiku, ali i nudi interaktivni prikaz kroz web preglednik.

QUAST-LG koristi ostale alate minimap2 [4] i RED [5] u svom analiziranju rezultata i sastavljanju genoma. Jedna od glavnih razlika između QUAST-LG alata i alata opisanog u ovom radu je već navedena dodatna značajka u evaluiranju dobivenih rezultata.

3. Podaci

Podaci koje koristi alat opisan u ovom radu zapisani su u FASTA [6] ili FASTQ [7] formatu. Ove datoteke mogu sadržavati više sekvenci, a svakoj je pridijeljen identifikator te sami niz baza koji predstavlja sekvencu. FASTQ datoteke dodatno imaju i niz ASCII znakova koji predstavljaju kvalitetu očitanja svake baze.

Tablica 3.1: FASTA i FASTQ format

Br. retka	FASTA	FASTQ
1	">" + identifikator	"@" + identifikator
2	Sekvenca	Sekvenca
3	-	"+" + neobavezni identifikator
4	-	Kvaliteta pojedine baze

Budući da svaki znak 4. reda predstavlja kvalitetu očitanja točno jedne baze, 4. red će imati jednako znakova kao i drugi. Dodatno, poslije identifikatora u oba formata moguće je dodati i kratki opis. Primjer FASTA formata dan je u nastavku:

>a019fb87-85b7-495a-b6dc-789a2f8c4572_Basecall_2D_000_2d
GCCAAAAGCACGCCGGGCGACCATAATGAGAAGATGCTCACCGCCAG

Budući da je alat vođen referencom prilikom sastavljanja genoma, potreban mu je i cijeli genom uz očitanja. Jedan od korištenih podataka za testiranje je genom bakterije *Escherichia coli*. Očitanja dobivena sekvenciranjem mogu se preuzeti pomoću poveznice `https://nanopore.s3.climb.ac.uk/MAP006-1_2D_pass.fasta`, a cijeli genom bakterije je moguće preuzeti ovdje: `ftp://ftp.ncbi.nlm.nih.gov/genomes/all/GCF/000/005/845/GCF_000005845.2_ASM584v2/GCF_000005845.2_ASM584v2_genomic.fna.gz`. Osim toga moguće je preuzeti i ostale genome raznih organizama, poput npr. kvasaca, na kojima je također testiran ovaj alat. Genom *Escherichie coli* se sastoji od 4.6 milijuna baza, dok genom kvasca *Saccharomyces cerevisiae* broji oko 12 milijuna baza.

4. Metode

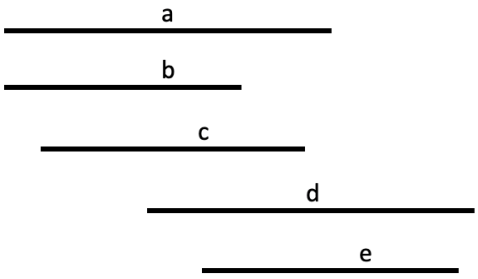
4.1. *Sweep line algoritam*

Da bi bilo moguće izgraditi graf sastavljanja, svako očitanje tj. vrh ne smije biti sadržan u drugom očitanju. Budući da se cijeli genom čita nekoliko puta takav slučaj se često pojavljuje u ulaznom skupu podataka. Stoga je prvi korak ovog alata da se ulazni skup pročisti od takvih očitanja. Prilikom mapiranja svakom očitanju je određen početak i kraj na referentnom genomu.

Naivni pristup ovom problemu bi bio da se svaka dva očitanja uspoređuju. Neka je l i r par očitanja za koje treba provjeriti je li jedno sadržano u drugom. Ako l počinje prije r te završava poslije r, očitanje r je sadržano u l i valjalo bi ga izbaciti iz skupa podataka. Ovaj pristup je ispravan, ali uz kvadratnu vremensku složenost, $O(n^2)$, koja na velikom skupu podataka može biti znatno osjetna na performansama alata.

Problem je moguće riješiti i drugim pristupom, tako zvanim algoritmom *Sweep line*. Algoritam se općenito koristi kada je potrebno naći elemente skupa koji su sadržani jedni u drugima. Njegova složenost je $O(nlogn)$. Prvo je potrebno sortirati sva očitanja po početnoj poziciji, a zatim po daljoj krajnjoj poziciji. Nakon toga se za svako očitanje obilaze sva sljedeća dok god je krajnja pozicija sljedećeg očitanja manja od krajnje pozicije trenutnog. Sva tako posjećena očitanja je potrebno izbaciti. Kada se završi postupak za jedno očitanje, samo se prelazi na ono sljedeće po početnoj poziciji.

Slika 4.1 prikazuje skup podataka nakon što je sortiran prvo po manjoj početnoj poziciji, a zatim po većoj daljnjoj poziciji. Prateći algoritam, prvo ćemo uzeti očitanje a i naći sva očitanje koja imaju kraj manji od njega. Takva očitanja će biti sadržana u a i treba ih izbaciti. Za ovaj slučaj to su očitanja b i c. Očitanje d ima kraj veći od a te tu treba stati i prijeći na iduće očitanje, koje je upravo d. Ono u sebi sadrži e pa ćemo i njega izbaciti. Algoritam 1 prikazuje pseudokod. Funkcija $inc(x)$ vraća sljedbenika elementa x u nizu kojem x pripada ako on postoji. U najgorem slučaju složenost je $O(nlogn)$ što je znatno bolje od kvadratne složenosti.



Slika 4.1: Prikaz sortiranih očitanja prije sweep line algoritma

Algorithm 1 Sweep line algoritam

Ulaz: A – Skup očitanja.

Izlaz: Skup bez očitanja sadržanih u drugim očitanjima A.

$sort(A)$ *// sortiraj po početnoj, a zatim po krajnjoj poziciji*

for (i in A) **do**

  $j := inc(i)$ *// j = element koji slijedi i*

  while $j.end <= i.end$ **do**

    $to\_remove = j$

    $j := inc(j)$

    $A.remove(to\_remove)$

  end while

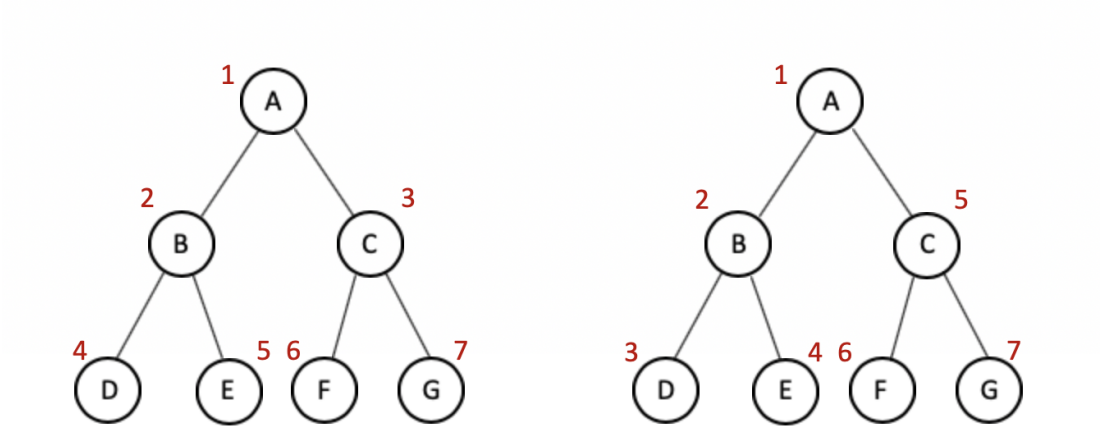
end for

4.2. Pretraživanje u dubinu

Graf sastavljanje se vrlo intuitivno može prevesti u stablo. Svako očitanje će predstavljati jedan čvor, a njegova djeca će biti sva očitanja u koja je taj čvor usmjeren, tj. sva očitanja koja se nastavljaju na odabrano očitanje. Sastavljanje genoma u ovako definiranom problemu se zapravo svodi na pretraživanje stabla od početnog čvora do onog konačnog.

Postoji nekoliko načina pretraživanja stabla, od kojih su vjerojatno najpoznatiji pretraživanje u širinu (eng. Breadth-first search, BFS) i pretraživanje u dubinu (Depth-first search, DFS). Ideja pretraživanja u širinu je da se čvorovi obilaze razinu po razinu, s lijeva na desno. Tek kada su posjećeni svi čvorovi razine prelazi se na sljedeću razinu. Pretraživanje u dubinu je također jednostavno, prvo se obilazi najljevije neposjećeno dijete svakog čvora, te tako zapravo pretražujemo kroz sve razine dok ne dođemo do

one zadnje.



Slika 4.2: Pretraživanje u širinu (lijevo) i pretraživanje u dubinu (desno)

Slika 4.2 prikazuje primjer pretraživanja za dva navedena algoritma. Redoslijed obilaženja čvorova kod pretraživanja u širinu na danom primjeru bi bio: ABCDEFG, a redoslijed kod pretraživanja u dubinu: ABDECFG. Faktor grananja kod stabla je broj djece koje čvorovi u stablu imaju i označava se sa b. Ako sa m označimo maksimalnu dubinu stabla, a sa d dubinu na kojoj prvi puta nalazimo rješenje, tada je vremenska složenost pretraživanja u dubinu $O(b^m)$. Vremenska složenost pretraživanja u širinu je $O(b^{d+1})$.

Za problem sastavljanja genoma pretraživanje u dubinu će nam biti sasvim dovoljno odnosno davat će i najbolje rezultate. Ova pretpostavka vrijedi jer ako je moguće naći put kroz graf onda ga je sigurno moguće naći uzimajući najdulja očitanja. Uvijek će sljedeći čvor za obilazak biti ono dijete kojemu se kraj mapira najdalje na genomu. Tako će nam u stablu najljevije dijete biti zapravo ono najdalje na genomu koje slijedi trenutni čvor. Na ovaj način ćemo obići samo ona nužna očitanja i njih će biti minimalno. Jednom kada je nađen završni čvor, moguće se vratiti na početak prateći roditelje svakog posjećenog čvora do početnog stanja.

Algoritam 2 predstavlja pseudokod pretraživanja u dubinu. Funkcija $expand$ vraća svu djecu predanog čvora. Struktura koju koristi pretraživanje u dubinu je stog. Jedina razlika kod pretraživanja u širinu je što se u toj liniji inicijalizira red, a ne stog.

4.3. Needleman–Wunsch algoritam

Needleman–Wunsch algoritam je algoritam koji globalno poravnava dva niza, od kraja do kraja. Točnije, nastoji jedan niz pretvoriti u drugi i odrediti cijenu koja je za to

Algorithm 2 Pretraživanje u dubinu

Ulaz: S – Početni čvor u stablu.
Ulaz: G – Ciljni čvor u stablu.
$L := Stack()$
$L.push(S)$
while $lenght(L) > 0$ **do**
  $n := L.pop()$
  if $(goal(n))$ **then**
    return n
  end if
  for $(i \in expand(n))$ **do**
    $L.push(i)$
  end for
end while
return $fail$

potrebna. Jedine operacije koje koristi su umetanje, brisanje i zamjena znakova. Algoritam služi za, na primjer, određivanje cijene pretvorbe niza "GCATGCU" u niz "GATTACA". Spada u algoritme dinamičkog programiranja. Za ideju ima podjelu većih problema na manje te prvo rješavanje manjih koji konačno daju rješenje inicijalnog problema.

Ako su q i t dva niza koja treba poravnati, tada je potrebno izgraditi matricu duljine $t \times q$. Potrebno je odrediti i cijenu umetanja, brisanja i zamjene znaka ako se naiđe na nepodudaranja. Algoritam kreće od gornjeg lijevog polja matrice i kreće se isključivo desno, dolje ili dijagonalno dolje-desno. Svako polje određuje par znakova, jedan znak na svakom nizu ovisno o poziciji širine odnosno visine. Znakovi se uspoređuju i na temelju njihovog odnosa i predefiniranih cijena se odabire sljedeće polje u matrici. Kretanje prema dolje ili desno označavaju brisanje odnosno umetanje, a dijagonalno zamjenu ili podudaranje znakova. Podudaranja se nagrađuju pozitivnim brojem ili nulom, a sve ostalo akcije negativnim. Uzimajući to u obzir, cilj je dobiti što veći broj tijekom obilaska matrice. U donjem desno kutu bit će određena cijena koja je potrebna da se jedan niz pretvori u drugi. Ako je to potrebno mogu se odrediti i akcije koje su potrebne za pretvorbu, što se postiže vraćanjem u nazad kroz matricu, krećući se po poljima koja imaju najveću cijenu.

Algoritam 3 prikazuje pseudokod. Funkcija $w(s[i], t[j])$ određuje cijenu zamjene ili podudaranja znakova, a d je cijena umetanja odnosno brisanja. Loša strana ovog

Algorithm 3 Needleman–Wunsch algoritam

Ulaz: t, q – dva niza znakova.
Ulaz: r – Cijena poravnavanja.
$V[0,0] = 0$
for $(i := 0; i < length(s), inc(i))$ **do**
  $V[i,0] = d * i$
end for
for $(j := 0; j < length(t); inc(j))$ **do**
  $V[0,j] = d * j$
end for
for $(i := 0; i < length(s); inc(i))$ **do**
  for $(j := 0; j < length(t); inc(j))$ **do**
    $MATCH = V[i-1, j-1] + w(s[i], t[j])$
    $INSERTION = V[i, j-1] + d$
    $DELETION = V[i-1, j] + d$
    $V[i,j] = max(MATCH, INSERTION, DELETION)$
  end for
end for

algoritma je što mu je vremenska složenost $O(mn)$ gdje su m i n duljine nizova za poravnavanje.

4.4. Integracija algoritama

Na samom početku programa se prvo učitavaju podaci iz FASTA/FASTQ i PAF datoteka pomoću biblioteke bioparser [8]. Ponavljajuće regije su zapisane u datoteci koju generira RED [5] alat, te se i ta datoteka analizira i pohranjuju se sve ponavljajuće regije u jedan vektor.

Prvi korak u programu nakon samog učitavanja podataka je upravo uklanjanje onih očitanja koja su sadržana u drugim očitanjima. Taj postupak je nužan jer je inače nemoguće izgraditi graf sastavljanja. Za to služi *sweep line* algoritam koji upravo miče takva očitanja iz predanog skupa podataka. Algoritam se ne koristi samo na ovom dijelu, već i kod premošćivanja ponavljajućih regija. Koristi se uz manju modifikaciju tako što se uspoređuju ponavljajuće regije i preostala očitanja. Ponavljajuće regije su baš kao i očitanja sortirane po početnoj poziciji. Za svako očitanje se promatra koje ponavljajuće regije počinju poslije očitanja, a završavaju prije njega na genomu.

Tako se oba vektora slijedno pretražuju, a regije sadržane u očitanjima se izbacuju iz vektora ponavljajućih regija. Osim toga regija može biti premošćena ako se dva očitanja preklapaju preko nje, a počinju prije i završavaju poslije ponavljajuće regije.

Nakon ovog postupka su nam ostale ponavljajuće regije koje ne možemo premostiti. Unatoč tome, ne mora značiti da će one nužno generirati točke prekida prilikom sastavljanja jer je moguće da su svi njihovi parovi na genomu uspješno premošćeni. Odnosno od preostalog vektora ponavljajućih regija potrebno je ostaviti samo one koje imaju unutar njega svog para, ostale neće stvarati nejednoznačnost u grafu sastavljanja. Stoga je potrebno usporediti sve parove ponavljajućih regija u svrhu otkrivanja onih koje su slične. To je moguće napraviti tako da se za svaki par promatra cijena pretvorbe jeden regije u drugu. Za to nam je potreban Needleman–Wunsch algoritam. Konkretno je uzeto da je umetanje, brisanje ili zamjena znakova cijene -1, a podudaranje se vrednuje s 0. Ako se za svaku nejednakost u nizovima oduzme 1, a za jednakosti se ne dodaje ništa, to znači da će na kraju vrijednost polja u donjem desnom kutu biti negativna vrijednost broja nepoklapanja znakova dvije ponavljajuće regije. Ako je apsolutna vrijednost tog polja manja od 15% duljine kraće ponavljajuće regije onda kažemo da su regije par. To znači da se razlikuju u manje od 15% znakova. Regije koje nemaju para mogu se izbaciti. Preostali vektor ponavljajućih regija će generirati prekidne točke prilikom sastavljanja genoma.

Budući da su sva očitanja sortirana po početnoj poziciji za svako od njih je moguće jednostavno odrediti s kojim očitanjima se preklapaju. Tako se dobiva graf sastavljanja s početnim vrhovima svake komponente. Za svaki početni vrh je napravljeno pretraživanje u dubinu. Ako jedan kromosom ima više komponenti tada se prilikom obilaska računa koja komponenta pokriva najveći dio kromosoma i ona se uzima kao gornja granica. Graf svih kromosoma je zapisan u jednu datoteku u GFA [9] formatu.

5. Implementacija

Alat se oslanja na nekoliko vanjskih programa: Minimap2 [4], RED [5] i bioparser [8]. Minimap2 i RED su korišteni kako bi se izgenerirali ulazni podaci u određenim formatima, a bioparser se konkretno koristi u implementaciji alata.

5.1. Vanjski korišteni alati

Minimap2 je alat razvijen u svrhu mapiranja očitanja na referencu. Jedan od mogućih rezultata je i PAF format [10] koji je ulazna datoteka u ovdje opisan alat. PAF datoteka sadrži informacije o svakom očitanju i kako se ono mapira na referencu, od koje do koje pozicije, na koji lanac itd.

RED je alat koji pronalazi ponavljajuće regije. Za sastavljanje genoma informacija o ponavljajućim regijama je neophodna stoga je i rezultat ovog alata važan. Osim ovog alata moguće je koristiti i alat koji je napisala Sara Bakić za pronalaženje ponavljajućih regija na genomu, a koji je dostupan na: `https://github.com/lbcb-edu/BSc-thesis-18-19/tree/sbakic`. Oba alata u jednu datoteku zapisuju ponavljajuće regije u zasebne retke s njihovim pozicijama na genomu. Ta datoteka se učitava u program i za svaku ponavljajuću regiju se promatra može li se premostiti ili određuje prekidnu točku pri sastavljanju.

Bioparser koji je napisao Robert Vaser se koristi na samom početku alata, kako bi se pročitale datoteke u PAF, FASTA ili FASTQ formatu. Bioparser omogućuje parsiranje ovih datoteka koje se kasnije mogu koristiti kroz zadane klase opisane u README datoteci na poveznici: `https://github.com/rvaser/bioparser/blob/master/README.md`

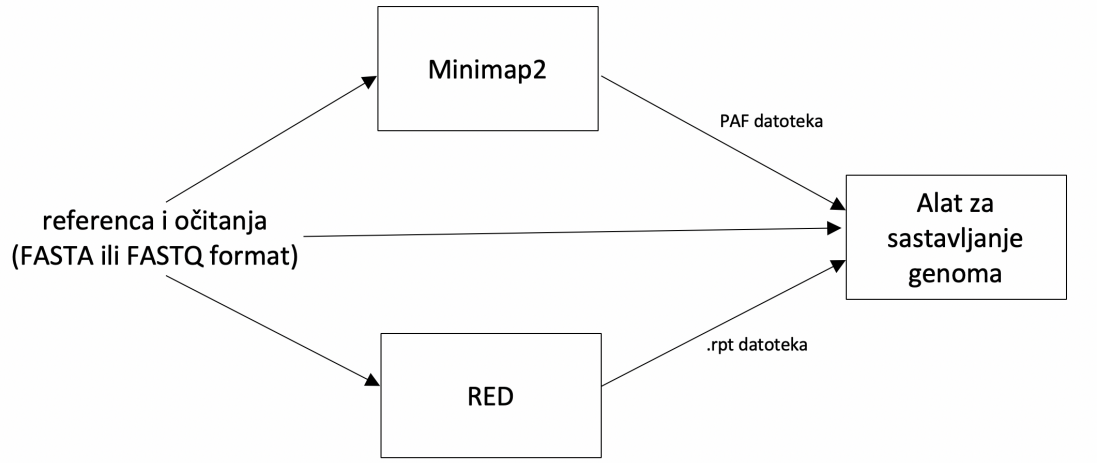
5.2. Struktura

Ovaj alat je dostupan na: `https://github.com/lbcb-edu/BSc-thesis-18-19/tree/lpozega`. Osmišljen je za brzo sastavljanje genoma iz predanih očitanja vo-

đeno referencom. Cijeli program je napisan u C++ programskom jeziku. Prevodi se pomoću programskog alata $cmake$. Organiziran je kao glavni program $main.cpp$ te kao zasebna biblioteka za učitavanje i obradu ponavljajućih regija s kojom se komunicira preko sučelja $repeats\_parser.h$. Biblioteka sadrži metoda za učitavanje regije, pronalaženje premoštenih te usporedbu parova regija. Korišteno je nekoliko klasa koje su navedene u nastavku:

- *PAFObject.cpp* - klasa napisana prema uputama alata bioparser, koristi se za čuvanje i manipulaciju podataka dobivenih iz PAF datoteke.
- *FASTAQObject.cpp* - klasa napisana prema uputama alata bioparser, koristi se za spremanja učitanih sekvenci i očitanja
- *Vertex* - klasa koja predstavlja vrh u grafu

Kako je potrebno predati četiri datoteke - očitanja, referencu, mapiranja i ponavljajuće regije, napravljena je manja skripta koja pokreće određene alata kako bi se izgenerirali potrebne datoteke koje se konačno prosljeđuju alatu za sastavljanje genoma. Organizaciju skripte prikazuje **slika** 5.1.



Slika 5.1: Rad skripte za pokretanja alata

6. Rezultati

U svrhu evaluiranja ovog alata, testiran je na dva skupa podataka, genomu bakterije *Escherichie coli* i genomu kvasca *Saccharomyces cerevisiae*. Prvo će biti prikazani rezultati, potom slijedi diskusija i usporedba sa sličnim rješenjima. Alat kao rezultate daje statistiku o pokrivenosti svih kromosoma zadanim očitanjima, datoteku za sastavljanje grafa u GFA formatu [9], popis svih korištenih očitanja i njihove identifikatore te datoteku u FASTA formatu koja sadrži genom kreiran od očitanja.

6.1. Testiranje

Svi testovi su napravljeni na računalu sa sljedećim specifikacijama:

- OS: MacOS Mojave 10.14.5
- Arhitektura: x86_64
- Procesor: Intel Core i7 2.2GHz
- RAM: 16GB

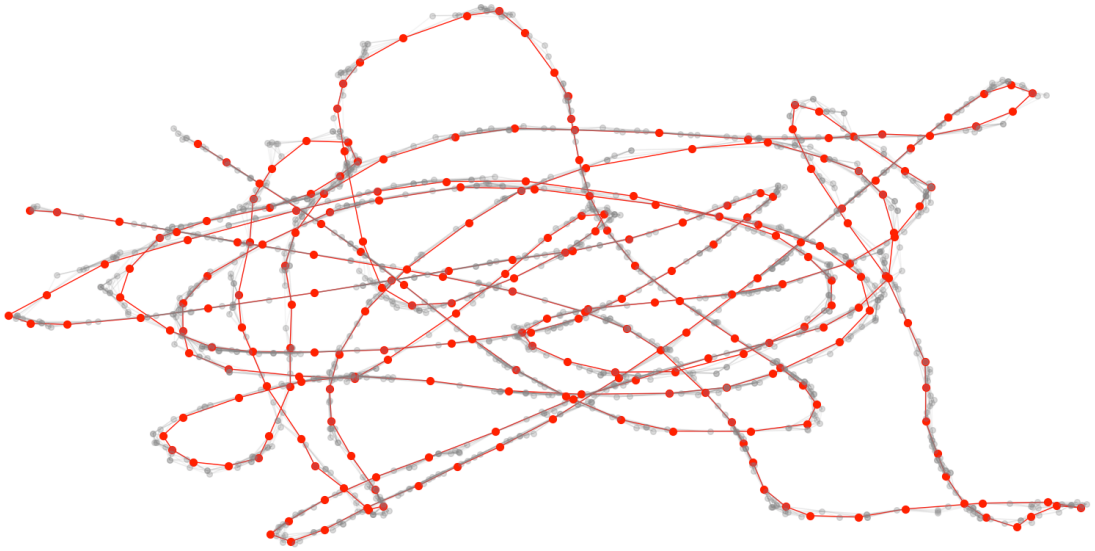
6.1.1. *Escherichia coli*

Genom *Escherichie* coli moguće je preuzeti ovdje:`ftp://ftp.ncbi.nlm.nih.gov/genomes/all/GCF/000/005/845/GCF_000005845.2_ASM584v2/GCF_000005845.2_ASM584v2_genomic.fna.gz`, a njezina očitanja dobivena sekvenciranjem ovdje: `https://nanopore.s3.climb.ac.uk/MAP006-1_2D_pass.fasta`. Genom ove bakterije se sastoji od 4.6 milijuna baza.

Tablica 6.1: Rezultati alata za Escherichiu coli

	Korišteno očitanja	Pokrivenost genoma
Escherichia coli (NC_000913.3)	264	99.99%

Tablica 6.1 prikazuje rezultate za bakteriju. Radi vizualnog prikaza kreirana je i datoteka koja prikazuje obilazak grafa sastavljanja za dane ulazne podatke. Datoteka je zatim korištena u Python programu s dodatkom NetworkX [11] koji iscrtava posjećene vrhove u grafu s obzirom na cijeli pročišćeni skup podataka očitanja. **Slika** 6.1 prikazuje taj graf - crveni vrhovi predstavljaju posjećena očitanja, dok sivi ona koja nisu posjećena.



Slika 6.1: Obilazak grafa prilikom sastavljanja E. coli

6.1.2. *Saccharomyces cerevisiae*

Saccharomyces cerevisiae poznatiji i kao pekarski kvasac je sekvenciran i uređajem PacBio i uređajem Oxford Nanopore, a u svrhu testiranja ovdje su korištena očitanja dobivena prvim uređajem. Genom ovog kvasca se sastoji od oko 12 milijuna baza. Budući da je genom nešto veći, sastavljanje i traje duže, a tablica 6.2 prikazuje dobivene rezultate. Ovaj kvasac se sastoji od 17 kromosoma koji su zasebno zapisani u FASTA formatu, tj. ne kao jedan dugački niz. Svaki od kromosoma ima i svoj identifikator.

Kao što se može vidjeti 11 kromosoma je moguće sastaviti gotovo u cijelosti, dok je ostalih 6 raznoliko pokriveno očitanjima. Rezultati grafa sastavljanja su ispisani u GFA formatu koji je potreban za prikaz genoma pomoću Bandage alata [12]. Ovaj alat također omogućava i interakciju s iscrtanim grafom. Svako očitanje korišteno u grafu je prikazano kao pravokutnik duljine ovisno o duljini očitanja. Vrhovi pravokutnika su povezani s onim pravokutnicima s kojim se očitanje preklapa. Prikaz je dan **slikom** 6.2.

Tablica 6.2: Rezultati alata za *Saccharomyces cerevisiae*

	Oznaka	Korišteno očitanja	Pokrivenost genoma
1.	$fragment_1$	6	99.99%
2.	$fragment_2$	36	99.99%
3.	$fragment_3$	61	99.99%
4.	$fragment_4$	27	99.99%
5.	$fragment_5$	17	99.99%
6.	$fragment_6$	16	99.99%
7.	$fragment_7$	44	84.50%
8.	$fragment_8$	44	60.36%
9.	$fragment_9$	11	46.93%
10.	$fragment_{10}$	70	99.99%
11.	$fragment_{11}$	62	99.21%
12.	$fragment_{12}$	52	99.99%
13.	$fragment_{13}$	37	86.75%
14.	$fragment_{14}$	35	99.99%
15.	$fragment_{15}$	25	52.36%
16.	$fragment_{16}$	46	66.44%
17.	$fragment_{17}$	101	99.99%

Drugi prikaz rezultata je napravljen uporabom Gepard alata [13] tako što se uspoređuje sastavljeni genom od očitanja naspram ulaznog genoma. Poklapanje u bazama između genoma je predočenom crnom točkom u koordinatnom sustavu. Tako pravokutne linije zapravo predstavljaju kontinuirano poklapanje baza, odnosno ispravno izgrađene sekvence kao što se može vidjeti na **slici** 6.3.

Alat ima jednostavan zadatak određivanja je li moguće iz očitanja izgraditi željeni genom i u kojem postotku. S obzirom na tu jednostavnost nema velike memorijske i vremenske zahtjeve kao što pokazuje tablica 6.3.

Tablica 6.3: Zahtjevnost alata

	Vrijeme(sec)	Memorija(GB)
Escherichia coli	5	0.3
Saccharomyces cerevisiae	66	1.5

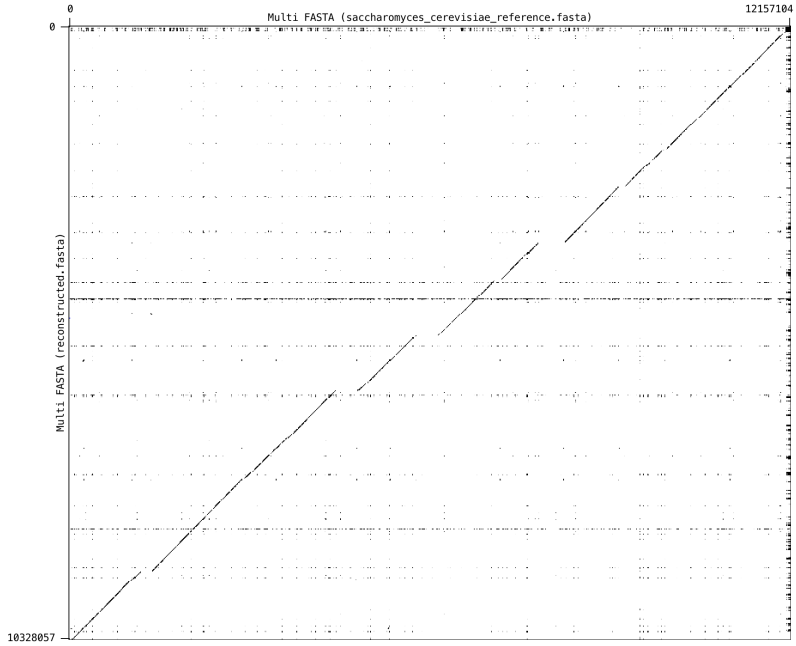
Slika 6.2: Prikaz rezultata sastavljanje kvasca *Saccharomyces cerevisiae* pomoću Bandage alata

6.2. Diskusija

Alat glavnu primjenu nalazi pri određivanju kvalitete sekvenciranja tj. koliko je zadovoljiv skup podataka koji je dobiven sekvenciranjem. Uz to, pomoću njega se može evaluirati rezultate alata koji sastavljaju genom bez reference.

Uspoređujući se s QUAST-LG alatom ovaj alat je samo dio njegovih funkcionalnosti. Uzimajući tu činjenicu u obzir, odrađuje traženje gornje granice u sastavljanju dosta brže što može biti korisno ako je to jedina potrebna informacija. Što se tiče kvalitete, QUAST-ov *upper-bound-assembly* daje jednake rezultate za skup podataka *Escherichie coli*.

Budući da alat ovisi o rezultatima Minimap2 i RED programa, to može utjecati na njegovu kvalitetu i točnost.



Slika 6.3: Prikaz rezultata sastavljanje kvasca *Saccharomyces cerevisiae* pomoću Gepard alata

7. Zaključak

Svrha ovog rada je bila izraditi alat koji će na temelju predanog skupa podataka očitanja i reference odrediti koja je gornja granica u kvaliteti prilikom sastavljanja genoma. Alat je u potpunosti napisan u programskom jeziku C++. U radu je korišteno nekoliko algoritama poput *sweep line* algoritma, pretraživanja u dubinu i Needleman–Wunsch algoritma. Graf se sastavlja pomoću grafa i njegovog obilaska uzimajući u obzir sve ponavljajuće regije i njihove pozicije na genomu. Rezultati pokazuju da u relativno malom vremenu je alat sposoban odrediti tražene informacije. Testiran je na dva skupa podataka: genomu bakterije *Escherichie coli* i kvascu *Saccharomyces cerevisiae*. Rezultati su potom evaluirani s obzirom na referencu i uspoređeni s alatom QUAST-LG.

Alat može biti korišten za evaluiranje skupa podataka dobivenog sekvenciranjem i za evaluiranje drugih alata koji sastavljaju genom.

LITERATURA

[1] Mikheenko A., Prjibelski A., Saveliev V., Antipov D., and Gurevich A. *Versatile genome assembly evaluation with QUAST-LG.*, 2018. URL https://www.ncbi.nlm.nih.gov/pubmed/29949969.

[2] Wikipedia. Deoksiribonukleinska kiselina, . URL https://hr.wikipedia.org/wiki/Deoksiribonukleinska_kiselina.

[3] Anhony Rhoads and Kin Fai Au. *PacBio Sequencing and Its Applications*, 2015. URL https://www.ncbi.nlm.nih.gov/pubmed/26542840.

[4] Li H. *Minimap2: pairwise alignment for nucleotide sequences.*, 2018. URL https://www.ncbi.nlm.nih.gov/pubmed/29750242.

[5] Girgis HZ. *Red: an intelligent, rapid, accurate tool for detecting repeats de-novo on the genomic scale.*, 2015. URL https://www.ncbi.nlm.nih.gov/pubmed/26206263.

[6] Wikipedia. FASTA format, . URL https://en.wikipedia.org/wiki/FASTA_format.

[7] Wikipedia. FASTQ format, . URL https://en.wikipedia.org/wiki/FASTQ_format.

[8] Vaser R. Bioparser. URL https://github.com/rvaser/bioparser.

[9] GFA format. URL https://github.com/GFA-spec/GFA-spec/blob/master/GFA2.md.

[10] PAF format. URL https://github.com/lh3/miniasm/blob/master/PAF.md.

[11] Aric Hagberg, Dan Schult, and Pieter Swart. Networkx. URL https://networkx.github.io/.

[12] Ryan R. Wick, Mark B. Schultz, Justin Zobel, and Kathryn E. Holt. *Bandage: interactive visualization of de novo genome assemblies*, 2015. URL https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4595904/.

[13] Krumsiek J., Arnold R., and Rattei T. *Gepard: a rapid and sensitive tool for creating dotplots on genome scale.*, 2007. URL https://www.ncbi.nlm.nih.gov/pubmed/17309896.

Gornja granica u sastavljanju genoma

Sažetak

U ovom radu je opisan alat za računanje gornje granice u sastavljanju genoma. U potpunosti je napisan u jeziku C++ i testiran na nekoliko skupova podataka. Alat ispisuje postotak pokrivenosti genoma ovisno o ulaznim podacima dobivenih sekvenciranjem. Također se kreira i datoteka u GFA format koja služi za sastavljanje grafa. U radu je korišteno nekoliko algoritama poput *sweep line* algoritma, pretraživanja u dubinu i Needleman–Wunsch algoritma. Testiran je na dva skupa podataka: genomu bakterije *Escherichie coli* i kvascu *Saccharomyces cerevisiae*. Glavnu primjenu će naći u evaluiranju kvalitete podatak dobivenih sekvenciranjem, kao i kod evaluiranja drugih alata za sastavljanje genoma.

Ključne riječi: gornja granica, sastavljanje genoma, graf

Upper bound in genome assembly

Abstract

In this thesis, a tool for genome upper bound assembly was created. It was fully written in C++ and tested on couple of datasets. Tool outputs percentage of the genome coverage that is possible to build in an ideal case considering sequenced data provided at the input. It also creats a graph file in GFA format. Algorithms like sweep line, depth-first search and Needleman–Wunsch were implemented. Tool was tested on datasets of *Escerichia coli* and yeast *Saccharomyces cerevisiae*. It might be mainly used in evaluating quality of the sequenced data and other de novo assembly methods.

Keywords: upper bound, genome assembly, graph