

Preklapanje visoko pouzdanih očitavanja diploidnih ili poliploidnih organizama

Vučenić, Filip

Undergraduate thesis / Završni rad

2024

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **University of Zagreb, Faculty of Electrical Engineering and Computing / Sveučilište u Zagrebu, Fakultet elektrotehnike i računarstva**

Permanent link / Trajna poveznica: <https://urn.nsk.hr/urn:nbn:hr:168:132156>

Rights / Prava: [In copyright](#)/[Zaštićeno autorskim pravom.](#)

Download date / Datum preuzimanja: **2025-03-21**



Repository / Repozitorij:

[FER Repository - University of Zagreb Faculty of Electrical Engineering and Computing repository](#)



SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 1400

**PREKLAPANJE VISOKO POUZDANIH OČITANJA
DIPLOIDNIH ILI POLIPLOIDNIH ORGANIZAMA**

Filip Vučenik

Zagreb, lipanj 2024.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 1400

**PREKLAPANJE VISOKO POUZDANIH OČITANJA
DIPLOIDNIH ILI POLIPLOIDNIH ORGANIZAMA**

Filip Vučenik

Zagreb, lipanj 2024.

ZAVRŠNI ZADATAK br. 1400

Pristupnik: **Filip Vučenik (0036542094)**
Studij: Elektrotehnika i informacijska tehnologija i Računarstvo
Modul: Računarstvo
Mentor: prof. dr. sc. Mile Šikić

Zadatak: **Preklapanje visoko pouzdanih očitavanja diploidnih ili poliploidnih organizama**

Opis zadatka:

Treća generacija tehnologija DNA sekvenciranja koju predvode Oxford Nanopore Technologies i Pacific Biosciences omogućuje očitavanje uzoraka duljine preko 20 kbp s točnošću iznad 99%. Ove karakteristike igraju ključnu ulogu u kvaliteti sastavljanja genoma iz uzoraka, posebice kada je genom diploidan ili poliploidan, odnosno kada je genetski materijal organiziran u dva ili više skupova homolognih kromosoma. Važno je napomenuti da su razlike između homolognih kromosoma su male i iznose manje od 1% za čovjeka. Pronalaženje preklapanja između očitavanja prvi je korak u procesu rekonstrukcije genoma iz uzoraka, nakon kojeg se konstruira graf preklapanja. Kako prilikom konstruiranja grafa ne bi izgubili informacije o jednom od haplotipova, potrebno je prilikom traženja preklapanja voditi računa o informativnim pozicijama, bazama koje su unikatne za svaki od haplotipova, ili segmenata u ponavljajućim regijama. Glavni cilj ove teze je unapređivanje već postojećih algoritama za brzo traženje preklapanja ili osmišljavanje i implementacija novog algoritma pogodnom za ove vrste očitavanja. Rješenje mora biti pogodno za paralelnu arhitekturu i implementirano u jeziku C++. Izvorni kod treba biti iscrpno dokumentiran koristeći komentare i slijediti Google C++ Style Guide kada je to moguće. Cijeli programski proizvod potrebno je postaviti na GitHub pod jednu od OSI odabranih licenci.

Rok za predaju rada: 14. lipnja 2024.

Zahvaljujem mentoru prof.dr.sc Mili Škiću na savjetima. Zahvaljujem asistentu univ. mag. ing. el. techn. inf. Filipu Tomasu na diskusijama i pomoći tijekom izrade rada. Također, zahvaljujem svojoj djevojci, obitelji i prijateljima na potpori tijekom mog dosadašnjeg obrazovanja.

Sadržaj

1. Uvod	3
2. Sekvenciranje	4
2.1. Osnovne tehnologije	4
2.1.1. Sangerovo sekvenciranje	5
2.1.2. Sekvenceri nove generacije	5
2.1.3. Sekvenceri treće generacije	5
2.2. Simulacija sekvenciranja alatom Seqrequester	6
2.3. Simulacija sekvenciranja alatom NanoSim	7
3. Računanja preklapanja	8
3.1. Minimizers	8
3.2. Poravnanje slijedova	10
3.2.1. Globalno poravnanje	12
3.2.2. Lokalno poravnanje	13
3.3. Hifiasm	14
4. Informativne pozicije	15
5. Rezultati i rasprava	16
5.1. Odvajanje haplotipova seqrequester očitavanja	17
5.2. Odvajanje haplotipova nanosim očitavanja	18
6. Programsko ostvarenje	20
7. Zaključak	21

Literatura	22
Popis slika	24
Popis tablica	25
Sažetak	26
Abstract	27

1. Uvod

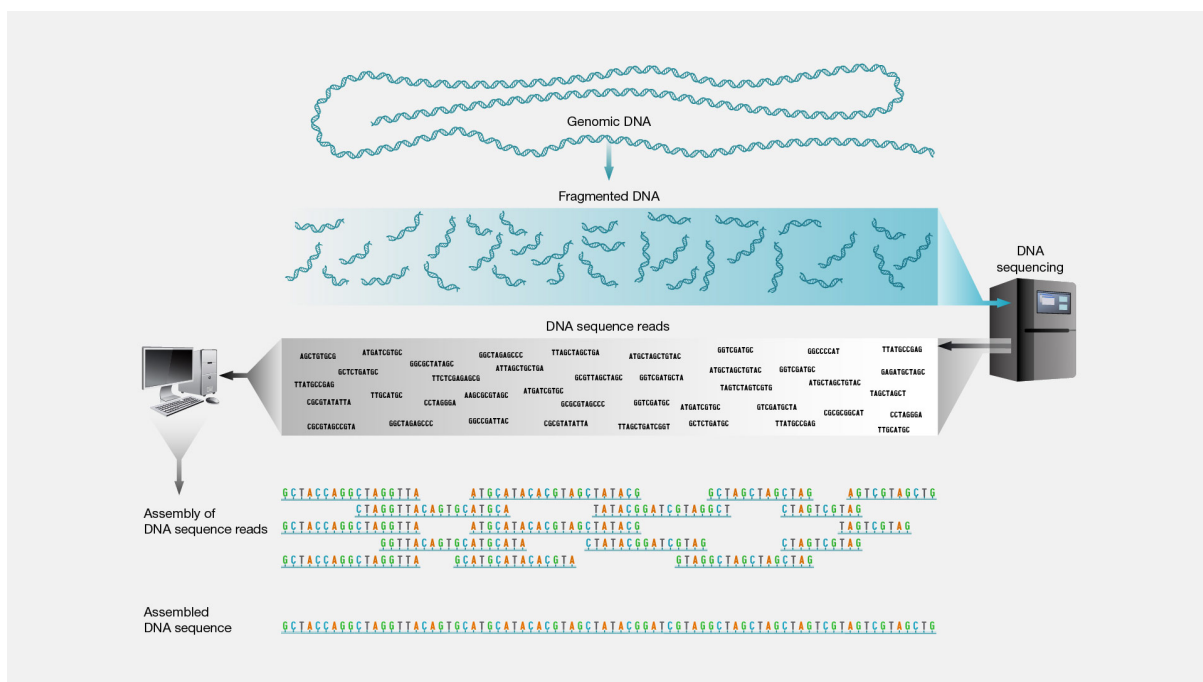
Razvojem tehnologije sekvenciranja te dolaskom i usavršavanjem treće generacije DNA sekvenciranja, danas su dostupna očitavanja duljina preko 20k baza sa točnošću iznad 99%. Zbog dobrih svojstava očitavanja (velika duljina i visoka točnost) pojavile su se De novo metode sastavljanja genoma (eng. *De novo assembly*). De novo strategije sastavljanja genoma oslanjaju se na dovoljnu duljinu očitavanja tako da iz samih očitavanja možemo rekonstruirati čitav genom (bitno je da su očitavanja dovoljno duga kako bi se mogle razlikovati ponavljajuće regije unutar genoma). Za razliku od De novo strategija za sastavljanje genoma kod strategija mapiranja očitavanja (eng. *Reference alignment*) potrebna nam je referenca jednog primjerka organizma pomoću koje onda zbog velike sličnosti dva genoma možemo sastaviti genom drugog organizma pomoću reference prvog. Kod diploidnih i poliploidnih organizama (organizmi koji imaju više od jednog seta kromosoma) potrebno je dodatno modificirati sastavljanje genoma dodavanjem mehanizma koji odvaja haplotipove (geni s jednog seta kromosoma) te ih zasebno sastavlja jer u protivnome dolazi do urušavanja haplotipova te gubimo informacije o ostalim kromosomima. Takav modificirani način sastavljanja naziva se De novo sastavljanje genoma sa razrješavanjem haplotipova (eng. *Haplotype-resolved de novo assembly*).

U okviru ovog završnog rada komentirat će se neke od tehnika sekvenciranja uz primjer simulacije sekvenciranja korištenjem alata otvorenog koda, dat će se primjeri algoritama za računanje preklapanja (bitan korak u De novo sastavljanju genoma) te primjeri algoritama za brzo računanje informativnih pozicija kao osnove za odvajanje očitavanja na pripadajuće haplotipove.

2. Sekvenciranje

2.1. Osnovne tehnologije

Sekvenciranje je tehnika čitanja nukleinskih kiselina iz DNA ili RNA te njihova tekstualna reprezentacija. Sve moderne tehnologije sekvenciranja bazirane su na strategiji takozvanog sekvenciranja sačmaricom (eng. *shotgun sequencing*). Osnovna karakteristika te metode je cijepanje osnovne DNA sekvence na male dijelove koji se zatim unose u sekvencer koji kao izlaz daje slijedove nukleinskih baza u tekstnom obliku. Na kraju se koriste računalni programi kako bi se izlomljeni slijedovi ponovno povezali u cjeloviti genom.



Slika 2.1. sekvenciranje sačmaricom

2.1.1. Sangerovo sekvenciranje

Prvu metoda sekvenciranja koja je postavila temelje modernog sekvenciranja izumio je Frederick Sanger 1977. godine. Glavna karakteristika Sangerovog sekvenciranja je vrlo visoka točnost očitavanja (veća od 99%) dok je glavni nedostatak vrlo visoka cijena sekvenciranja. Nakon uspješnog primjenjivanja njegove metode Ministarstvo energetike (eng. *Department of Energy*) i Nacionalni institut za zdravlje (eng. *National Institute of Health*) vlade SAD-a uspješno su realizirali projekt određivanja ljudskog genoma (eng. *Human genome project*). Projekt je proglašen završenim u travnju 2003. godine s ukupnim troškom od oko 3 milijarde američkih dolara. Zbog visoke cijene Sangerovog sekvenciranja počele su se razvijati nove tehnologije sekvenciranja koje bi riješile problem cijene te su se tako pojavili Sekvenceri nove generacije. Danas se Sangerovo sekvenciranje zbog svoje velike preciznosti još uvijek koristi kao mjerilo kvalitete novih tehnologija sekvenciranja.

2.1.2. Sekvenceri nove generacije

Sekvenceri nove generacije uspješno su riješili problem cijene sekvenciranja uz mali pad u kvaliteti očitavanja (taj je problem riješen u novijim verzijama sekvencera nove generacije), ali uz dodatnu brzinu samog sekvenciranja. Nedostatak sekvencera nove generacije jest mala duljina očitavanja i relativno skupa oprema.

2.1.3. Sekvenceri treće generacije

Sekvenceri treće generacije omogućili su sekvenciranje dugih očitavanja, a kod prvih verzija glavni je problem bio znatno povećanje pogreške naspram Sekvencera nove generacije i Sangerovog sekvenciranja. Kasnijim napretkom ta je greška u potpunosti smanjena te je to uzrokovalo pojavu assemblera koji se oslanjaju na De novo strategije sastavljanja genoma.

Tehnologija	Duljina (bp)	Točnost	Procjena cijene
Sangerovo sekvenciranje	500-1000	>99.99%	\$500 do \$1000 po megabazi (Mb)
Illumina (nova generacija)	50-300	99.9%	\$600 do \$1,500 po genomu
PacBio (treća generacija)	10k-15k	99.9%	\$5,000 do \$10,000 po genomu
Oxford Nanopore (treća generacija)	10k-100k	92-97%	\$1,000 do \$3,000 po genomu

Tablica 2.1. Usporedba tehnologija sekvenciranja

2.2. Simulacija sekvenciranja alatom Seqrequester

Seqrequester [1] je alat otvorenog koda koji služi za generiranje, modificiranje, ekstrakciju i računanje sažetaka DNA sekvenci. Ovdje će biti objašnjeno kako simulirati očitavanja diploidnog organizma korištenjem alata seqrequester. Organizam čiji će genom biti sekvenciran jest bakterija *Escherichia coli*. Kako je *Escherichia coli* haploidan organizam, prvi će korak biti mutiranje reference da bismo umjetno napravili dodatnu referencu koja će nam poslužiti za oponašanje kromosoma diploidnih organizama. U našim očitanjima imat ćemo neke promijenjenje pozicije (SNP eng. *Single Nucleotide Polymorphism*) koje će nam predstavljati varijacije u svojstvima diploidnih organizama. Mutiranje ćemo oponašati jednostavnom zamjenom nekih dušičnih baza tako da nam početna i mutirana referenca na kraju imaju jednak broj baza. Mutiranje bismo mogli još dodatno simulirati brisanjem, dodavanjem ili zamjenom nekih dijelova početne reference, ali radi lakšeg analiziranja zamijenit ćemo samo dušične baze. Za simuliranje očitavanja pokrećemo naredbu:

```
seqrequester simulate -genome <merged reference files> -coverage <coverage> -distribution <distribution>
```

U danoj naredbi, alatu seqrequester predajemo referentni genom iz kojeg će se sekvencirati očitavanja bez pogrešaka. Argument *coverage* govori nam koliko će se prosječno baza s neke pozicije nalaziti u očitanjima, a argument *distribution* prema kojoj će se distribuciji birati duljine naših očitavanja. Kao rezultat dobit ćemo očitavanja u fasta formatu. U FASTA formatu svako očitavanje sastoji se od opisnika i tekstualne reprezentacije nukleinskih baza.

2.3. Simulacija sekvenciranja alatom NanoSim

NanoSim [2, 3] je alat otvorenog koda koji služi za simuliranje realističnih Nanopore očitavanja (treća generacija sekvencera, ultra duga očitavanja). Simulacija se odvija u dva koraka. Prvi je korak treniranje modela na pravim očitanjim. Dok je drugi korak simulacija očitavanja korištenjem napravljenog modela (ili korištenjem gotovog modela). Za simuliranje očitavanja potrebno je pokrenuti simulaciju u *genome* modu danom naredbom:

simulator.py genome -c <model> -o <output file> -n <number of reads>

Za razliku od očitavanja dobivenih seqrequesterom NanoSim očitavanja su realistična te je zbog toga prije traženja preklapanja potrebno još provesti korak ispravljanja pogrešaka. Za ispravljanje pogrešaka korišten je Herro [4, 5], alat otvorenog koda koji služi za ispravljanje pogrešaka Nanopore očitavanja.

3. Računanja preklapanja

Računanje preklapanja među očitajima ključan je dio u rekonstrukciji originalnog genoma. Prvi korak je podijeliti sekvence na k -mere (k -mer je podniz znakova duljine k originalnog niza znakova). Kako postoji veliki broj k -mera (u znakovnom nizu duljine N ima $N - k + 1$ k -mera), a nisu svi potrebni u određivanju preklapanja dio njih se može izbaci na način da se promatraju samo minimalni k -meri unutar nekog prozora od w k -mera (*minimizeri*).

3.1. Minimizers

Minimizeri [6] su skup k -mera koji zadovoljavaju sljedeće svojstvo: **Ako se dva znakovna niza značajno preklapaju onda će postojati barem jedan zajednički minimizer u oba znakovna niza.** Ova definicija je poprilično apstraktna i općenita te ćemo pokazati način na koji je to svojstvo primjenjeno u postupku određivanja *minimizera*. Počnimo s određivanjem poretka kod k -mera, taj postupak općenito može biti dosta složen i imati veliki utjecaj na kvalitetu samih minimizera zbog toga što želimo izbjeći preveliku diskriminaciju određenih k -mera. Predloženo rješenje je definiranje poretka pomoću hash funkcije, opširnije [7]. Ovdje ćemo kao primjer radi lakšeg objašnjavanja uzeti da su k -meri poredani leksikografski (npr. najmanji 3 -mer je AAA i ima vrijednost 000). Kada smo odredili poredak k -mera sljedeći je postupak određivanje prozora, prozor se dobiva tako da se zajedno grupira w slijednih k -mera u blok na kojem se onda traži minimalni k -mer koji je onda proširen (w,k) minimizermom (skraćeno *minimizermom*), veličina bloka je uvijek $w + k - 1$. U slučaju da postoji više minimalnih k -mera svi se uzimaju kao *minimizeri*. Sada možemo preformulirati početno svojstvo: **Ako dva znakovna niza imaju zajednički znakovni podniz duljine $w + k - 1$, tada oni imaju zajednički (w,k) minimizer.** Parametri w i k imaju vrlo velik utjecaj na pokrivenost bloka. Ako pogledamo primjer na Tablici 3.1. mo-

žemo vidjeti kako pozicije 1, 2, 3, 7 i 12 nisu pokrivena s *minimizera* što je problem kada je bitno da su sve pozicije pokrivena *minimizera*.

Pozicija	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Vrijednost znaka	2	3	1	0	3	2	1	0	1	2	3	3	1	0	1
prozori sa podebljanim minimizera	2	3	1	0	3	2									
		3	1	0	3	2	1								
			1	0	3	2	1	0							
				0	3	2	1	0	1						
					3	2	1	0	1	2					
						2	1	0	1	2	3				
							1	0	1	2	3	3			
								0	1	2	3	3	1		
									1	2	3	3	1	0	
											2	3	3	1	0

Tablica 3.1. primjer *minimizera* sa $w=4$ $k=3$ [6]

Ukoliko zanemarimo početak i kraj nizova, možemo primjetiti da problem pokrivenosti nastaje kada su *minimizera* dva susjedna prozora udaljena za više od k . Kako je veličina prozora jednaka $k + w - 1$ ukoliko uzmemo $w \leq k$ tada osiguravamo da će *minimizera* dva susjedna prozora biti u najgorem slučaju biti udaljeni za k . U primjeru sa Tablice 3.2 možemo vidjeti da ukoliko uzmemo jednaku veličinu prozora i *k-mera* da osiguravamo da se nepokrivena pozicija mogu nalaziti samo na početku ili na kraju nizu, što se može riješiti tako da se uvijek uključe prvi i posljednji *k-meri* nizova.

Pozicija	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Vrijednost znaka	2	3	1	0	3	2	1	0	1	2	3	3	1	0	1
prozori sa podebljanim minimizera	2	3	1	0	3										
		3	1	0	3	2									
			1	0	3	2	1								
				0	3	2	1	0							
					3	2	1	0	1						
						2	1	0	1	2	3				
							1	0	1	2	3	3			
								0	1	2	3	3	1		
									1	2	3	3	1	0	
											2	3	3	1	0

Tablica 3.2. primjer *minimizera* sa $w=3$ $k=3$ [6]

Nakon gradnje *minimizera* sljedeći je korak pronaći sva preklapanja *minimizera* između

svih kombinacija očitavanja i proširiti ta preklapanja koliko je moguće da bismo dobili što veće konačno preklapanje.

3.2. Poravnanje slijedova

Poravnanje slijedova osnovna je tehnika za određivanje sličnosti između dva znakovna niza. Tehnika je primjenjiva na poravnanje bilo koja dva znakovna niza i temelji se na Levenshteinovoj udaljenost odnosno udaljenosti uređivanja (engl. *edit distance*). Udaljenost uređivanja je minimalni broj operacija koje su potrebne da bi se proizvoljan znakovni niz s pretvorio u proizvoljni znakovni niz t . Uz dozvoljene operacije: zamjene znakove (promjena jednog znaka iz niza s u znak koji se nalazi na odgovarajućoj poziciji u nizu t), umetanje (dodavanje jednog znaka u niz s u cilju da niz odgovara nizu t) i brisanje (micanje jednog znaka iz niza s da odgovara nizu t).

Algoritamsko rješenje problema udaljenosti uređivanja temelji se na tehnici dinamičkog programiranja. Ako su duljine nizova s i t , $|s| = n$ i $|t| = m$ stvaramo matricu V veličine $(m + 1) \times (n + 1)$, za svaku moguću operaciju (umetanje, brisanje i neslaganje) zadajemo cijenu (npr. možemo uzeti $d=1$ za operacije brisanja i umetanja, 1 za neslaganje i 0 ako su znakovi na pozicijama jednaki).

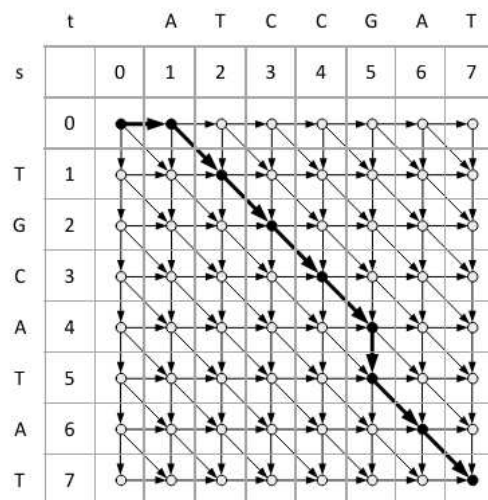
$$V(i, j) = \begin{cases} 0 & i = 0 \wedge j = 0 \\ d * i & j = 0 \\ d * j & i = 0 \\ \min \begin{cases} V(i - 1, j - 1) + w(s_i, t_j) \\ V(i - 1, j) + d & \text{inace} \\ V(i, j - 1) + d \end{cases} & \end{cases} \quad (3.1)$$

Zatim inicijaliziramo nulti redak i nulti stupac na cijenu operacije pomnoženu sa pozicijom ($V(0, j) = d*j$, odnosno $V(i, 0) = d*i$). Za sve preostale pozicije udaljenost izračunavamo na temelju prethodnih na način da uzimamo minimum poravnanja lijevog, gornjeg i gornje-lijevog susjeda zbrojenih sa cijenama prijelaza (prijelaz iz gornjeg i lijevog susjeda imat će cijenu d i označavat će brisanje i umetanje, prijelaz iz gornje-lijevog susjeda ovisit će o tome jesu li znakovi na trenutnim pozicijama jednaki te ako jesu cijena prijelaza je 0, a u slučaju da su različiti cijena prijelaza zadana za zamjenu znakova). Vrijednost udaljenosti uređivanja

zatim možemo pročitati sa pozicije $V(m,n)$. Ukoliko želimo saznati i koje tačno operacije moramo poduzeti i na taj način rekonstruirati niz t' iz t takav da je $t'=s$ moramo uz svaki element matrice V doadno pamtiti i njegovog prethodnika. Tablica 3.3 i slika 3.1 prikazuju jedan primjer udaljenost uređivanja između niza $s=TGCATAT$ i $t=ATCCGAT$. Tablica 3.3 prikazuje elemente matrice $V(i,j)$ dok slika 3.1 prikazuje rekonstrukciju niza t u niz s uz 4 operacije (iz niza s se briše prvo slovo A , treće slovo C se mijenja sa G , peto slovo G se mijenja sa A i na poziciju 6 se ubacuje slovo T).

	t	A	T	C	C	G	A	T
s	0	1	2	3	4	5	6	7
T	1	1	1	2	3	4	5	6
G	2	2	2	2	3	3	4	5
C	3	3	3	2	2	3	4	5
A	4	3	4	3	3	3	3	4
T	5	4	3	4	4	4	4	3
A	6	5	4	4	5	5	4	4
T	7	6	5	5	5	6	5	4

Tablica 3.3. Matrica udaljenosti poravnanja za nizove $s=TGCATAT$ i $t=ATCCGAT$



Slika 3.1. Mreža poravnanja za nizove $s=TGCATAT$ i $t=ATCCGAT$

3.2.1. Globalno poravnanje

Algoritmi za globalno poravnanje na efikasan način pokušavaju riješiti problem sličnosti dva niza. Rješenje problema prvi su formulirali Needleman i Wunsch 1970. [8] godine te je prvi algoritam imao kubnu složenost. Prvi algoritam sa kvadratnom složenošću predložen je 1972. godine, a rad Petra H. Sellersa iz 1974 [9] . godine pokazao je da su problemi maksimiziranja sličnosti i minimiziranja uređivanja ekvivalentni. Po radu Needlemana i Wunscha danas se algoritam za računanje sličnosti dva niza naziva Needleman-Wunschovim algoritmom za računanje sličnosti, a za razliku od algoritma za računanje udaljenosti uređivanja baziran je na maksimizaciji sličnosti (3.2)

$$V(i, j) = \begin{cases} 0 & i = 0 \wedge j = 0 \\ d * i & j = 0 \\ d * j & i = 0 \\ \max \begin{cases} V(i - 1, j - 1) + w(s_i, t_j) \\ V(i - 1, j) + d \quad \text{inace} \\ V(i, j - 1) + d \end{cases} & \end{cases} \quad (3.2)$$

Ovdje je bitno za primjetiti da se da bi algoritam pravilno funkcionirao cijene dodavanja, brisanja i zamjene moraju biti manje od nagrade za pravilno poravnanje. Najčešće je praksa da su cijene akcija negativne, a nagrada za pravilno poravnaje pozitivna, akcije brisanja i umetanja su najčešće jednake jer su to ekvivalenten akcije, a akcija zamjene može biti različita ovisno o tome želimo li dati nekoj akciji prioritet. Kao i kod algoritma za računanje udaljenosti uređivanja rješenje nam se nalazi u donjem desnom kutu matrice, $V(n, m)$.

3.2.2. Lokalno poravnanje

Algoritmi lokalnog poravnanja za razliku od algoritama globalnog poravnanja na efikasan način pokušavaju maksimizirati poravnanje na način da će poravnanje nekih regija dati maksimalan rezultat, dok se kod globalnog poravnanja uspoređuju cijeli znakovni nizovi. Algoritam lokalnog poravnanja moguće je konstruirati iz prethodno opisanog algoritma globalnog poravnanja uvođenjem malih promjena u početnom popunjavanju tablice i uvođenjem pravila da vrijednost elemenata matrice V ne može pasti ispod 0.

$$V(i, j) = \begin{cases} 0 & i = 0 \wedge j = 0 \\ \max \begin{cases} 0 \\ V(i-1, j-1) + w(s_i, t_j) \\ V(i-1, j) + d \\ V(i, j-1) + d \end{cases} & \text{inace} \end{cases} \quad (3.3)$$

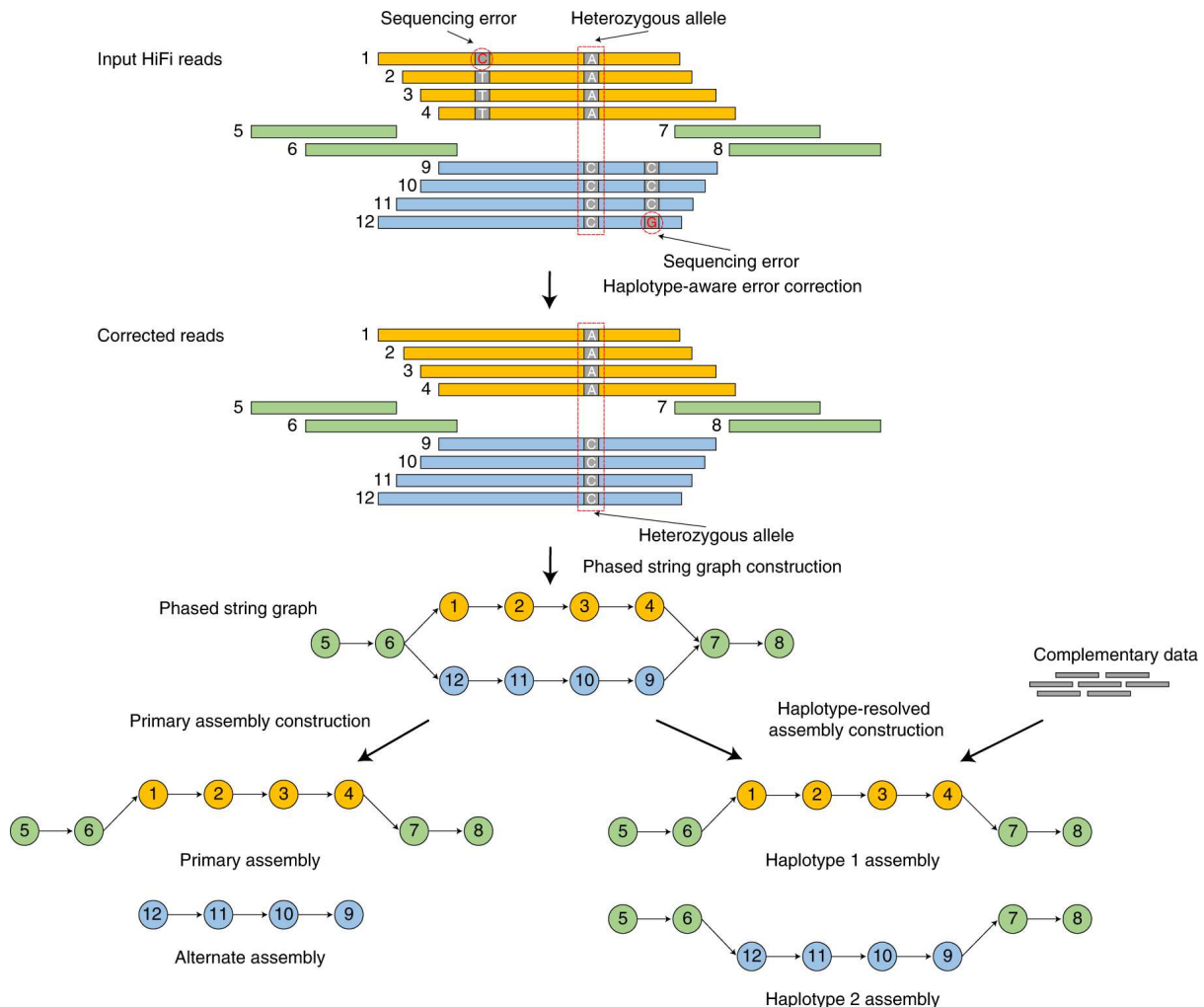
Ovom malom izmjenom pravila osigurali smo da zapravo tražimo dva podniza koja se najbolje preklapaju, to je osigurano time da je minimalna vrijednost koja se može postići 0 te nam tako svaka pozicija može biti početna pozicija podniza. Za razliku od globalnog poravnanja kao rješenje lokalnog poravnanja uzimamo maksimalni element matrice V i od njega možemo rekonstruirati podnizove na isti način kao i kod globalnog poravnanja. Algoritam lokalnog poravnanja naziva se još i Smith-Watermanov algoritam, a na Tablici 3.4. možemo i vidjeti lokalno poravnanje nizova s="ACCTAAGG" i t="GGCTCAATCA".

		G	G	C	T	C	A	A	T	C	A
	0	0	0	0	0	0	0	0	0	0	0
A	0	0	0	0	0	0	2	2	0	0	2
C	0	0	0	2	0	2	0	1	1	2	0
C	0	0	0	2	1	2	1	0	0	3	1
T	0	0	0	0	4	2	1	0	2	1	2
A	0	0	0	0	2	3	4	3	1	1	3
A	0	0	0	0	0	1	5	6	4	2	3
G	0	2	2	0	0	0	3	4	5	3	1
G	0	2	4	2	0	0	1	2	3	4	2

Tablica 3.4. Primjer lokalnog poravnanja dva niza s="ACCTAAGG" i t="GGCTCAATCA"

3.3. Hifiasm

Hifiasm [10, 11] je *de novo assembler* sa razrješavanjem haplotipova. Početno je dizajniran za duga HiFi očitavanja PacBio sekvencera treće generacije, a kasnije je prilagođen i za *nanopore* očitavanja. Princip rada *assemblera* sažeto je prikazan na slici 3.2.



Slika 3.2. De novo sastavljanje genoma sa razrješavanjem haplotipova korištenjem hifiasma

U okviru projektnog zadatka hifiasm je jedan od korištenih alata za računanje preklapanja između očitavanja. Iako to nije osnovna funkcija alata kada je alat pokrenut sa *-dbg-ovec* zastavicom kao rezultat dobivaju se preklapanja očitavanja razdvojena po haplotipovima u PAF (*Pairwise Mapping Format*) formatu. Razlog korištenja hifiasma za računanje preklapanja je mogućnost transparentne usporedbe rada alata za odvajanje haplotipova.

4. Informativne pozicije

Poravnanje više slijedova je tehnika u bioinformatici gdje se istovremeno uspoređuje tri ili više slijedova. Kod preklapanja više slijedova informativne pozicije su pozicije na kojima dolazi do razlika u poravnanju. Na primjeru poravnanja više slijedova (Tablica 4.1) možemo vidjeti da na pozicijama 1 i 3 sva 3 slijeda imaju jednako poravnanje i te pozicije nisu informativne. Na poziciji 2 postoje razlike u poravnanju, slijed 1 i 3 na poziciji 2 imaju znak A, dok slijed 4 ima znak G, slijed 2 na poziciji 2 ima umetanje te se on zanemaruje i zaljučujemo da je pozicija 2 informativna. Slično se može zaključiti i za poziciju 4 kojoj slijedovi 1 i 2 na poziciji 4 imaju znak C, a slijedovi 3 i 4 imaju znak T i pozicija 4 je također informativna. Na poziciji 5 slijedovi 1, 2 i 3 imaju znak T, a slijed 4 ima umetanje, kako nema alternative znaku T pozicija 5 nije informativna.

	Pozicija 1	Pozicija 2	Pozicija 3	Pozicija 4	Pozicija 5
Slijed 1	A	A	G	C	T
Slijed 2	A		G	C	T
Slijed 3	A	A	G	T	T
Slijed 4	A	G	G	T	

Tablica 4.1. Poravnanje više slijedova

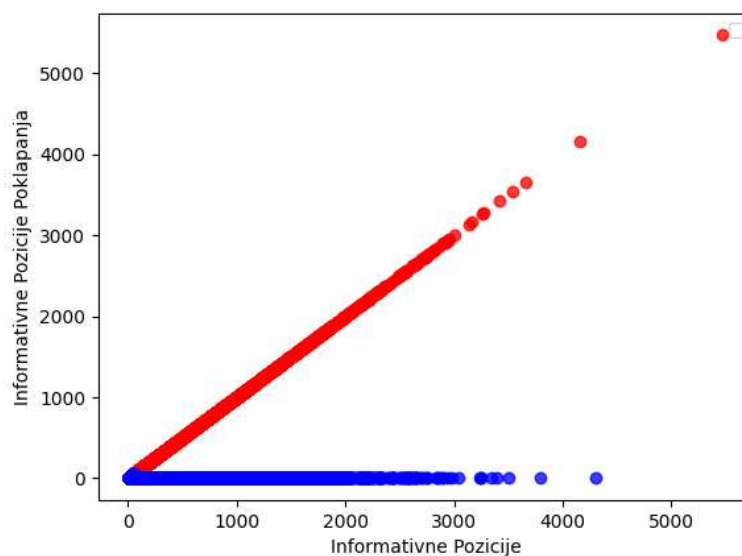
Bitno je primjetiti da se u slijedovima znaju nalaziti pogreške nastale prilikom sekvenciranja i da pogreške ne bi trebale biti ubrajane u informativne pozicije. Rješenje tog problema biti će detaljnije razmatrano u sljedećim poglavljima. Informativne pozicije su ključni dio u programskom dijelu projekta za određivanje haplotipova pojedinih očitavanja.

5. Rezultati i rasprava

U ovom poglavlju detaljnije ćemo komentirati način određivanja pripadanja očitavanja određenim haplotipovima. Kao što je već spomenuto za to su ključne informativne pozicije. Informativne pozicije računaju se tako što se provede poravnanje više slijedova nad preklapajućim dijelovima očitavanja, na način da se prebroje sve baze koje se nalaze na jednakim pozicijama preklapajućih dijelova. Uzmimo ponovno primjer uz tablicu 4.1. pozicija 1 ima 3A, pozicija 2 ima 2A,1G,... Nakon što smo to izračunali za sva preklapanja jednog očitavanja prelazimo na korak određivanja informativnih pozicija. Informativne pozicije određujemo na način da ponovo prolazimo po svim preklapanjima odabranog očitavanja i za svaku poziciju svakog preklapanja analiziramo podudaraju li se ili razlikuju baze očitavanja u preklapanju te također analiziramo i postoji li konsenzus oko toga koja se baza nalazi na toj poziciji. Kažemo da postoji konsenzus ako se jedna baza pojavljuje u više od 80% preklapanja te tada kažemo da pozicija tada nije informativna. Parametar koji određuje koliko se jedna baza mora pojavljivati u presjeku svih preklapanja određen je odokativno te bi se kao nastavak ovog završnog rada mogla raditi bolja procjena tog parametra. Svrha parametra je razlikovanje informativnih pozicija od pogrešaka u očitanjima. U slučaju da ne postoji konsenzus odnosno imamo informativnu poziciju, brojimo tu informativnu poziciju brojačem informativnih pozicija, ako uz postojanje informativne pozicije također imamo i podudaranje između baza očitavanja u preklapanju tada uvećavamo i brojač koji broji informativne pozicije sa prisustvom podudaranja između očitavanja. U kasnije koraku koristimo podatke o ukupnom broju informativnih pozicija i broju informativnih pozicija uz prisustvo podudaranja između očitavanja da bismo odredili pripadaju li očitavanja istom ili različitom haplotipu. To određujemo na način da promatramo postoji li dovoljan broj informativnih pozicija sa preklapanjem u odnosu na ukupan broj informativnih pozicija da možemo tvrditi da očitavanja dolaze sa istog haplotipa. U sljedeća dva podpoglavlja razmatrat ćemo pokušaje odvajanja haplotipova različitih očitavanja pomoću informativnih pozicija.

5.1. Odvajanje haplotipova seqrequester očitavanja

Podsjetimo se simulacije seqrequester očitavanja. Seqrequester generira haploidna očitavanja bez pogrešaka, kako bismo dobili diploidna očitavanja mutirali smo referencu (mutirali smo 1% baza) te smo tada sekvencirali obje reference uz prosječnih 15 pojavljivanja svake pozicije ($coverage = 15$). Izračunali smo preklapanja dobivenih očitavanja korištenjem hifiasm *assemblera*. Iz preklapanja smo izvukli informativne pozicije te je dobiven sljedeći graf (Slika 5.1).

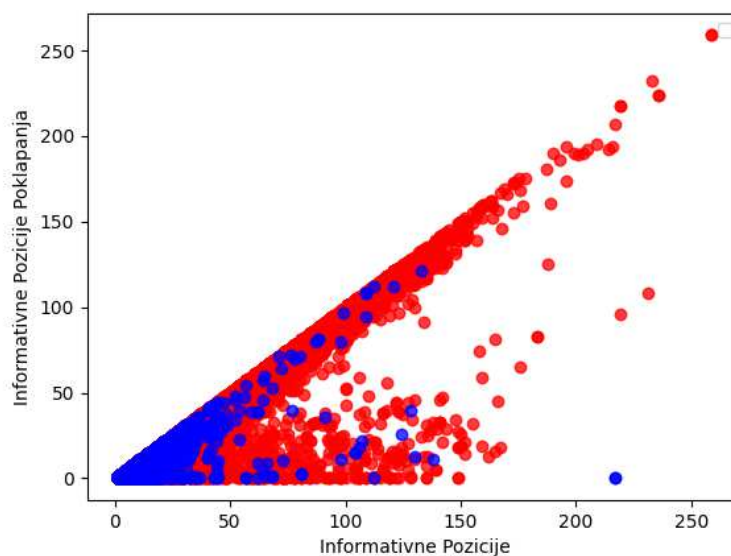


Slika 5.1. Informativne pozicije preklapanja seqrequester očitavanja

Na grafu su crvenom bojom označena preklapanja očitavanja koja dolaze sa istih haplotipova, dok su plavom bojom označena preklapanja očitavanja koja dolaze sa različitih haplotipova. Kod savršenih očitavanja ta su preklapanja savršeno odvojena zbog toga što kada nema pogrešaka nema miješanja informativnih pozicija sa pogreškama i uvijek se kod preklapanja očitavanja sa istih haplotipova broj informativnih pozicija preklapanja podudara s brojem informativnih pozicija. To se događa zbog toga što ako nema pogrešaka i dva očitavanja na informativnoj poziciji imaju jednak znak onda oni dolaze sa istog haplotipa. Suprotno vrijedi za očitavanja koja dolaze s različitih haplotipova jer se na njihovim informativnim pozicijama znakovi uvijek razlikuju. U usporedbi sa hifiasmom dobiveni su slični rezultati te su za savršena očitavanja haplotipovi odvojeni s točnošću većom od 99%. Pogreške koje su se dogodilo vjerojatno su nastale zbog nepostojanja informativnih pozicija kod nekih preklapanja na što se može utjecati povećanjem prosječnog broja pojavljivanja svih pozicija u očitanjima.

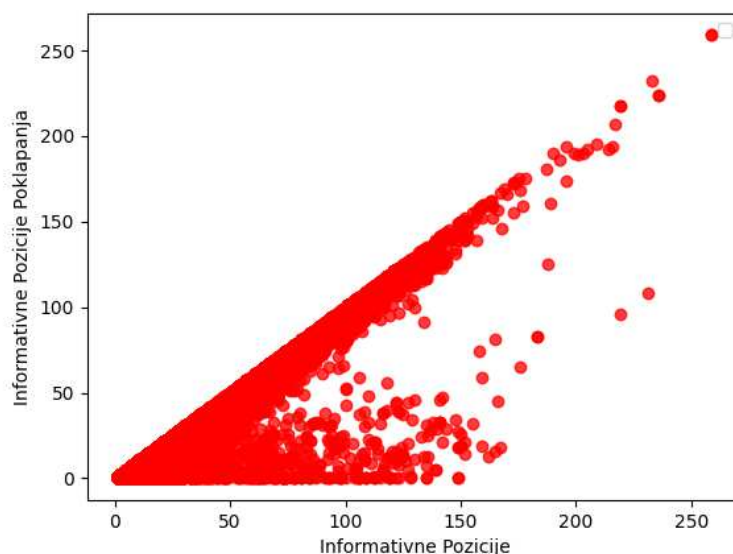
5.2. Odvajanje haplotipova nanosim očitavanja

Nanosim očitavanja dobili smo korištenjem alata nanosim koji simulira realistična Nanopore očitavanja (očitanja Oxford Nanopore Technologies sekvencera, sekvencer spada u sekvencere treće generacije). Ispravili smo pogreške realističnih nanopore očitavanja korištenjem alata Herro. Zatim smo jednako kao i za očitavanja dobivena sekruesterom izračunali preklapanja korištenjem hifiasm *assemblera* i izračunali informativne pozicije. Dobiveni graf (Slika 5.2) prikazan je u nastavku.



Slika 5.2. Informativne pozicije preklapanja nanosim očitavanja

Na grafu su ponovno crvenom bojom kao i prije označena preklapanja očitavanja koja dolaze sa istih haplotipova, dok su plavom bojom označena preklapanja očitavanja koja dolaze sa različitih haplotipova. Vizualno iz ovog grafa izgleda kao da bi bilo moguće odjeliti skupove pripadnosti haplotipovima s obzirom na broj informativnih pozicija i informativnih pozicija sa preklapanjem. Te su bile pokušane metode linearne regresije i SVM (*Support Vector Machine*) za klasifikaciju pripadnosti haplotipovima. Oba pokušaja klasifikacije tim metodama nisu bila uspješna zbog krive predodbe podataka na prvom grafu jer kada je graf bio ponovo nacrtan samo za podatke koji dolaze s istih haplotipova (Slika 5.3) vidimo da se skupovi preklapaju i da na ovaj način ne možemo odvojiti skupove da bismo ih mogli uspješno klasificirati.



Slika 5.3. Informativne pozicije preklapanja nanosim očitavanja

Pretpostavljeni razlog ovakvog rezultata su pogreške u računanju informativnih pozicija. Korištena metoda konsenzusa od 80% nije dovoljno dobro raspoznala informativne pozicije te se ovakav rezultat ostavlja kao motivacija za razvijanje metoda koje bi kvalitetnije raspoznavale informativne pozicije od pogrešaka. Drugi je razlog da su podaci sadržavali značajno više očitavanja koja dolaze s istog haplotipa (više od 93%), odokativnom procjenom parametara dobili smo točnost od 89%. To se može činiti kao dobar rezultat, ali ako se pogleda konfuzijska matrica možemo primjetiti da visok postotak točnosti dolazi zbog velikog predviđanja jednakosti haplotipova što je automatski predviđanje kada se ne pronađu informativne pozicije.

	Predviđeni jednaki haplotipovi	Predviđeni različiti haplotipovi
Stvarni jednaki haplotipovi	632028	31626
Stvarni različiti haplotipovi	45938	742

Tablica 5.1. Konfuzijska matrica

6. Programsko ostvarenje

Programski dio projektnog zadatka [12] podijeljen je na dva dijela. Prvi dio nalazi se unutar SNP_tool direktorija i tamo se nalazi implementacija alata za razdvajanje haplotipova implementirana u C++-u. Drugi dio nalazi se unutar Scripts direktorija i tamo se nalaze pomoćni programi za analizu i vizualizaciju implementirani u Pythonu.

Za povezivanje i prevođenje alata za razdvajanje haplotipova korišten je alat CMake. Kako bi se program povezo i preveo potrebno je koristiti sljedeću naredbu:

```
cmake -S . -B build && cd build && make && cd ..
```

Nakon prevođenja izvršni program nalazi se u bin poddirektoriju direktorija build pod nazivom **snp**. Programski kod organiziran je na način da se mogu koristiti različiti izvori preklapanja. Izvori preklapanja modelirani su sučeljem koje sadrži dvije metode: jednu za dohvat očitavanja i jednu za dohvat preklapanja. Ideja je da se jednom nakon prevođenja alata kasnije mogu dodavati novi izvori preklapanja bez potrebe za ponovnim prevođenjem. Zbog toga se izvori preklapanja prevode kao dinamičke biblioteke i pomoću metode tvornice se stvaraju dinamički. Podržani su izvori preklapanja u PAF formatu (argument paf), PAF formatu sa izračunatim porvananjem (argument pafPA), PAF formatu dobivenom alatom minimap2 [13] (argument pafM2) te preklapanja izračunata alatom ram [14] (argument ram). Dodatne dinamičke biblioteke mogu se dodati u lib poddirektorij. Izvršni kod alata pokreće se predajom argumenta za željeni izvor preklapanja i predajom argumenta potrebnih za stvaranje generatora željenog izvora. Primjer pokretanja alata sa seqrequester očitanjima Escherichie Coli i njenih preklapanja dobivenim hifiasmom:

```
./snp paf E-coli_reads.fasta E-coli_hifiasm_overlaps.paf
```

Kao izlaz program generira novu datoteku u nadograđenom PAF formatu, gdje se osim uobičajenih informacija nalaze i informacije o tome dolaze li očitavanja s istih ili različitih haplotipova (0 istih haplotipovi, 1 različiti haplotipovi) te omjer broja podudarajućih informativnih pozicija i ukupnog broja informativnih pozicija.

7. Zaključak

U ovom završnom radu pokazali smo da pomoću informativnih pozicija možemo dobro odvojiti haplotipove očitavanja simulirana alatom seqrequester. Karakteristika tih očitavanja je da nemaju simulirane greške te nam je to olakšalo postupak odvajanja haplotipova. Isto smo napravili i sa očitanjima simuliranim alatom nanosim. Karakteristika nanosim očitavanja je da su ona realistična i da sadrže greške. Pokušali smo s obzirom na informativne pozicije donjeti zaključak o podrijetlu haplotipova i napraviti binarnu klasifikaciju korištenjem metoda strojnog učenja (linearna regresija i SVM) i odokativnom procjenom parametra. Nismo uspjeli dobro odvojiti haplotipove te su metode strojnog učenja uvijek pokazivale jednu vrijednost, dok su se odokativnom procjenom parametra dobili malo bolji rezultat, ukupna točnost je bila vrlo niska. Pretpostavljeni razlog loših rezultata sa nanosim očitanjima je prejednostavan način računanja informativnih pozicija te se kao nastavak rada predlaže bolja procjena korištenog parametra ili razvijanje naprednijih metoda.

Literatura

- [1] Brian Walenz, <https://github.com/marbl/seqrequester>, [mrežno; stranica posjećena: svibanj 2024.].
- [2] C. Yang, J. Chu, R. L. Warren, i I. Birol, “NanoSim: nanopore sequence read simulator based on statistical characterization”, *GigaScience*, sv. 6, br. 4, str. gix010, 02 2017. <https://doi.org/10.1093/gigascience/gix010>
- [3] Yang, Chen and Chu, Justin and Warren, René L and Birol, Inanç, <https://github.com/bcgsc/NanoSim>, [mrežno; stranica posjećena: svibanj 2024.].
- [4] Dominik Stanojevic, <https://github.com/lbcb-sci/herro>, [mrežno; stranica posjećena: svibanj 2024.].
- [5] D. Stanojevic, D. Lin, P. Florez De Sessions, i M. Sikic, “Telomere-to-telomere phased genome assembly using error-corrected simplex nanopore reads”, 2024.
- [6] Michael Roberts, Wayne Hayes, Brian R. Hunt, Stephen M. Mount and James A. Yorke, https://academic.oup.com/bioinformatics/article-pdf/20/18/3363/48906547/bioinformatics_20_18_3363.pdf, [mrežno; stranica posjećena: lipanj 2024.].
- [7] G. Irving, “Inverse of a hash function”, <https://naml.us/post/inverse-of-a-hash-function/>, ožujak 2012., [mrežno; stranica posjećena: lipanj 2024.].
- [8] S. B. Needleman i C. D. Wunsch, “A general method applicable to the search for similarities in the amino acid sequence of two proteins”, *J. Mol. Biol.*, sv. 48, br. 3, str. 443–453, 1970.

- [9] P. H. Sellers, “On the theory of computation and evolutionary distances”, <https://digitalcommons.rockefeller.edu/cgi/viewcontent.cgi?article=1003&context=peter-sellers-memorial>, accessed: 2024-6-13.
- [10] H. Cheng, G. T. Concepcion, X. Feng, H. Zhang, i H. Li, “Haplotype-resolved de novo assembly using phased assembly graphs with hifiasm”, *Nat. Methods*, sv. 18, br. 2, str. 170–175, 2021. <https://doi.org/https://doi.org/10.1038/s41592-020-01056-5>
- [11] H. Cheng, E. D. Jarvis, O. Fedrigo, K.-P. Koepfli, L. Urban, N. J. Gemmell, i H. Li, “Haplotype-resolved assembly of diploid genomes without parental data”, *Nat. Biotechnol.*, sv. 40, br. 9, str. 1332–1335, 2022. <https://doi.org/https://doi.org/10.1038/s41587-022-01261-x>
- [12] F. Vučenik, “Programski kod završnog rada”, <https://github.com/filipvucenic/FinalProject/>, lipanj 2024., [mrežno; stranica posjećena: lipanj 2024.].
- [13] “minimap2”, <https://github.com/lh3/minimap2>, [mrežno; stranica posjećena: lipanj 2024.].
- [14] R. Vaser, “Ram”, <https://github.com/lbcb-sci/ram>, [mrežno; stranica posjećena: lipanj 2024.].

Popis slika

2.1. sekvenciranje sačmaricom	4
3.1. Mreža poravnanja za nizove $s=TG\text{CATAT}$ i $t=AT\text{CCGAT}$	11
3.2. De novo sastavljanje genoma sa razrješavanjem haplotipova korištenjem hi- fiasma	14
5.1. Informativne pozicije preklapanja seqrequester očitavanja	17
5.2. Informativne pozicije preklapanja nanosim očitavanja	18
5.3. Informativne pozicije preklapanja nanosim očitavanja	19

Popis tablica

2.1. Usporedba tehnologija sekvenciranja	6
3.1. primjer <i>minimizera</i> sa $w=4$ $k=3$ [6]	9
3.2. primjer <i>minimizera</i> sa $w=3$ $k=3$ [6]	9
3.3. Matrica udaljenosti poravnanja za nizove $s=TGCATAT$ i $t=ATCCGAT$. . .	11
3.4. Primjer lokalnog poravnanja dva niza $s="ACCTAAGG"$ i $t="GGCTCAATCA"$	13
4.1. Poravnanje više slijedova	15
5.1. Konfuzijska matrica	19

Sažetak

Ovaj završni radi bavi se pronalaženjem informativnih pozicija i njihovim korištenjem za predviđanje haplotipova u preklapanjima očitavanja genoma diploidnih organizama.

U završnom radu je dan pregled osnovnih tehnologija sekvenciranja te primjeri alata za simulaciju sekvenciranja. Objasnjene su metode računanja preklapanja sekvenciranih očitavanja te je dan uvid u računanje informativnih pozicija.

Ključne riječi: informativne pozicije; preklapanja; algoritmi

Abstract

This final thesis deals with finding informative positions and using them to predict haplotypes in the overlaps of genome reads of diploid organisms.

This thesis provides an overview of basic sequencing technologies and examples of tools for sequencing simulation. Here are explained methods for calculating overlaps of sequenced reads, and insights into the calculation of informative positions are provided.

Keywords: informative positions; overlaps; algorithms