SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 2322

Program za simulaciju pročitanih nizova nukleotida

Vanessa Županović

Zagreb, lipanj 2012.

Sadržaj

Uvod....1

1. Sekvencioniranje genoma....2

2. Programi za simulaciju pročitanih nizova nukleotida....6

2.1. Wgsim....6

2.2. ART....7

2.3. DWGSIM....8

3. Simulator....9

3.1. Pojednostavljeni algoritam simulatora....11

3.2. Formati ulaza i izlaza....14

3.2.1. FASTA format....14

3.2.2. FASTQ format....15

3.2.2.1 Ilumina verzija....16

3.3. Instalacija i pokretanje simulatora pod operacijskim sustavom Linux....17

3.4. Simulacija....21

3.5. Rezultati i diskusija....23

3.6. Primjena kontinuiranih i diskretnih distribucija u procesu simulacije....28

Zaključak....31

Literatura....32

Sažetak....33

Summary....34

Skraćenice..35

Privitak..36

Uvod

Današnje vrijeme karakterizira iznimna popularnost sekvencioniranja genoma. *GenBank* – baza svih trenutno poznatih, javno dostupnih DNA sekvenci, koja je početkom 1987. sadržavala 15 milijuna (15 000 000) nukleotida do samo 1992. uosmerostručava svoju veličinu na 120 milijuna (120 000 000) nukleotida . Ta ista baza danas broji otprilike 126 551 501 141 nukleotida u 135 440 924 zapisa u tradicionalnim *GenBank* odjeljenjima te 191 401 393 188 nukleotida u 62 715 288 zapisa u tzv. WGS (engl. *"Whole-genome shotgun")* odjeljenju. Nedavno razvijene platforme za sekvencioniranje druge generacije pokreću revoluciju na polju medicinskih istraživanja i biologije. U usporedbi s tradicionalnom Sangerovom tehnologijom sekvencioniranja, nove platforme generiraju podatke mnogo brže te je i sam proces sekvencioniranja postao značajno jeftiniji. Nakon što je završen proces sekvencioniranja jednog od sljedećih, vrlo izazovnih, koraka obrade predstavlja upravo pokušaj da se niz kratkih očitanja nastalih u procesu sekvencioniranja mapira na referentni genom što je u osnovi i zadaća kompjuterskih programa znanih kao *"asembleri"*. Kroz proteklih nekoliko godina razvijen je čitav niz takvih programa čija je glavna namjena upravo "sklapanje" kratkih očitanja generiranih platformama za sekvencioniranje druge generacije. Samo neki od tih alata jesu: SSAKE, VCAKE, Euler-sr, Edna, Velvet, AbySS i SOAPdenovo [1]. Performanse navedenih alata u ovisnosti o ogromnim varijacijama potencijalnih različitih uvjeta nisu u potpunosti istražene, stoga ih je potrebno testirati na poznatom genomu s poznatim očitanjima te mjestima mapiranja na referentni genom. U tu svrhu razvijen je čitav niz alata koji bi isto trebao omogućiti. Kroz naredna poglavlja dan je kratak osvrt na već postojeća rješenja te je opisan rad jednog takvog simulatora izrađenog u demonstracijske svrhe (poglavlje 3. *Simulator*). Također, u već spomenutom poglavlju, detaljno su opisani i formati ulaza i izlaza "demonstracijskog simulatora" te je izvršeno testiranje na velikom broju testnih sekvenci od čega su neki rezultati testiranja odabrani te analizirani i prikazani grafički. Dan je i kratak uvod u samo sekvencioniranje genoma s naglaskom na važnost istog (poglavlje 1. *Sekvencioniranje genoma*) te kratak osvrt na već postojeće programe za simulaciju pročitanih nizova nukleotida (poglavlje 2. *Programi za simulaciju pročitanih nizova nukleotida*).

1. Sekvencioniranje genoma

Sekvencioniranje DNA predstavlja jednu od najvažnijih tehnika za proučavanje nekog gena, kojom se određuje točan slijed nukleotida unutar promatrane sekvence. S pomoću računalnih programa za analizu sekvencija DNA, podaci o redoslijedu mogu se rabiti za:

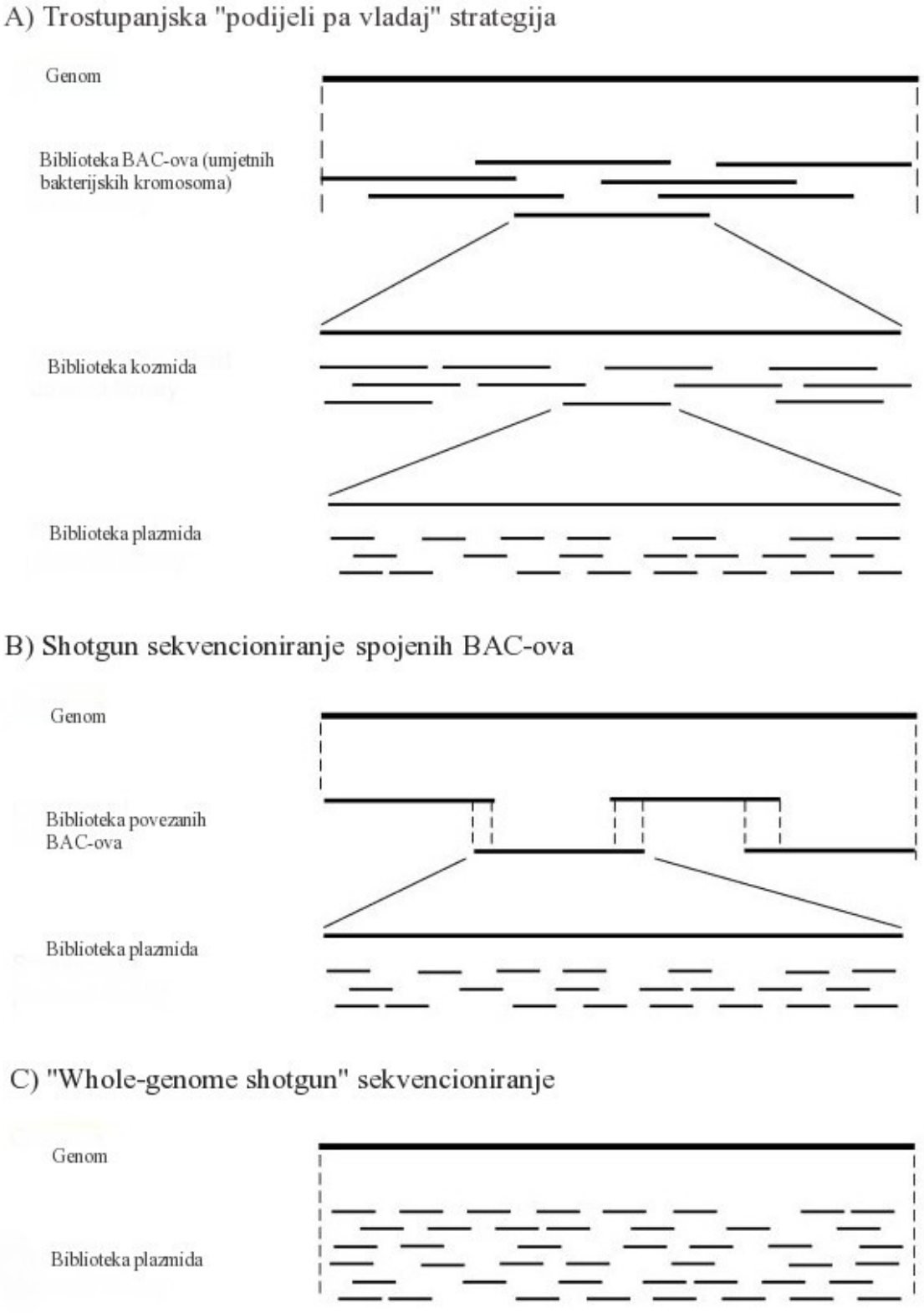
i) određivanje svih poznatih restrikcijskih mjesta unutar gena

ii) predviđanje proteina kodiranog tim genom

iii) dobivanje podataka o mogućim intronskim mjestima unutar genomske DNA

iv) te za dobivanje uvida u mehanizme kojima se provodi nadzor nad genskim izražavanjem i to analizom S' nekodirajućeg područja koje može sadržavati sljedove DNA za koje se zna da na sebe vežu specifične jezgrene proteine koji služe regulaciji transkripcije (transkripcijski regulacijski proteini ili transkripcijski faktori – dovode gen pod vremenski ili tkivno – specifičan nadzor).

Kod sekvencioniranja genoma razlikujemo tri strategije: tzv. engl. *"clone-by-clone"* [2] shotgun pristup (trostupanjska "*podijeli pa vladaj strategija*" [2]), shotgun sekvencioniranje BAC-ova [3] (umjetnih bakterijskih kromosoma) te engl. *"Whole-genome shotgun"* (WGS) [2]. Sve tri metode ilustrirane su sljedećom slikom:



Slika 1.1 Strategije za sekvencioniranje genoma. A) Klon po klon trostupanjska "podijeli pa vladaj" strategija. B) sekvencioniranje BAC-ova povezanih konektorima (STC). C) "*whole-genome shotgun*".

Važno je napomenuti da sve one uključuju kloniranje te tzv. engl. *"Shotgun sequencing"* [2] u nekoj od svojih faza, no ono što ih razlikuje je upravo način mapiranja klonova te trenutak pokretanja nasumičnog "Shotgun sequencing-a". WGS [2] je prvotno bio korišten za sekvencioniranje virusnog genoma, te se , kao što je bilo spomenuto ranije, temelji na principu nasumičnog odabira vrlo velikog broja malih klonova te pokušaju da se iz istih poravnavanjem rekonstruira polazna sekvenca. Ova se metoda pokazala vrlo uspješnom pri analizi relativno malih nepotpunih genoma kao što su virus koji napada cvjetaču (8031 pb) te bakteriofag lambda (48 502 pb), no pri sekvenciranju ljudskog genoma je gotovo beskorisna zbog veličine istog te vrlo velikog broja ponavljajućih regija.

Nakon utvrđivanja slijeda baza pojedinog fragmenta potrebno je rekonstruirati polaznu sekvencu što je posao računalnih programa znanih kao "asembleri". Ovi programi nastoje pronaći i analizirati preklapanja ili identične sljedove na krajevima pojedinih čitanja. Same algoritme za poravnanja moguće je podijeliti u tri velike kategorije: algoritmi bazirani na tablicama raspršenog adresiranja, algoritmi bazirani na sufiksnom stablu te naposljetku, algoritmi bazirani na merge sortiranju.

Ideja indeksiranja hash tablicom javlja se još kod BLAST-a [1], te svi algoritmi temeljeni na istoj slijede jednaku tzv. "posijaj i proširi" paradigmu. Primjerice BLAST čuva poziciju svake k-torke (podsekvence ulaznog niza) u hash tablici gdje ključ predstavlja upravo sama k-torka, te tijekom rada pretražuje postojeću bazu sekvenci s namjerom da pronađe podudaranja (uz pomoć prethodno navedene hash tablice) koja se nazivaju sjeme (engl. *"seed"*) nakon čega naposljetku proširuje i međusobno spaja sjemena. Inicijalno bez procjepa (engl. *"gaps"*) koje naknadno redefinira pomoću Smith-Waterman algoritma [1].

Algoritmi bazirani na prefiksnom i sufiksnom stablu implicitno uključuju dva koraka: identificiranje preciznog podudaranja te izgradnja grubih poravnanja zasnovanih na preciznom podudaranju. Kako bi pronašli precizna poravnanja ovi se algoritmi oslanjaju na sigurnu reprezentaciju sufiksnog odnosno prefiksnog stabla kao što su samo sufiksno stablo, poboljšano sufiksno polje te FM indeks [1]. FM indeks predstavlja podatkovnu strukturu koja omogućava pronalaženje određenog podniza unutar niza ili kolekcije nizova pri čemu je vremenska složenost istoga sublinearna, tj. ukoliko je S tekstualni dokument duljine n te D garnitura tekstualnih dokumenata ukupne duljine n, svako je pojavljivanje uzorka P moguće odrediti u vremenu manjem od O(n). Glavna namjena FM indeks-a je efikasna pretraga vrlo velikih tekstualnih datoteka kod kojih ne samo da je potrebna kompresija samog teksta već kompresiji moraju biti podvrgnuti i indeksi. FM indeks se

temelji na Burrows-Wheelerovoj transformaciji (BWT) [5], tj, algoritmu korištenom u tehnikama kompresije podataka. Ukoliko se niz znakova podvrgne ovoj transformaciji, niti jedan od znakova ne mijenja vrijednost već dolazi do izmjene redoslijeda pojavljivanja znakova. Ukoliko originalni string karakterizira učestalo pojavljivanje određenog podniza, onda će transformirani niz sadržavati više mjesta gdje se jedan znak ponavlja više puta unutar istog retka. Transformacija se obavlja sortiranjem svih rotacija tekstualnog zapisa leksikografskim poretkom te uzimanjem zadnje kolone na koju se postavlja EOF pokazivač. Po svojoj strukturi FM indeks je vrlo sličan sufiksnom polju. Glavna prednost kod uporabe sufiksnog stabla jest da je poravnanje na više identičnih kopija podniza potrebno učiniti samo jednom budući da se identične kopije svedu na samo jedan put unutar stabla dok je kod klasične tablice raspršenog adresiranja poravnanje potrebno činiti za svaku kopiju zasebno [1].

Tijekom protekle dvije godine objavljeno je više od 20 alata za poravnanje kratkih čitanja, od kojih se tek nekolicina svakodnevno rabi.

Tablica 1.1 daje pregled alata koji se prema informacijama sakupljenim sa "*SEQ answers*" foruma danas najviše rabe

Program	Algoritam indeksiranja
Bfast	Hashing ref.
Bowtie	FM index
BWA	FM index
MAQ	Hashing reads
Mosaik	Hashing ref.
Novoalign	Hashing ref.

2. Programi za simulaciju pročitanih nizova nukleotida

Sekvencioniranje genoma je područje kojeg kroz proteklih nekoliko godina karakterizira iznimno brz razvoj. Kao rezultat sekvencioniranja nastaje vrlo velik broj kratkih pročitanih nizova nukleotida. Važno je napomenuti da metoda ne garantira 100%-tnu točnost. Ukoliko referentni genom postoji, pročitani se nizovi pokušavaju mapirati na njega, te se isti u odnosu na sekvencionirani razlikuje u pojedinim nukelotidima, obrisanim i dodanim dijelovima. U nastavku je dan kratak pregled alata za simulaciju rezultata sekvencioniranja.

2.1. Wgsim

Wgsim [3] predstavlja mali alat za simulaciju pročitanih nizova nukleotida iz referentnog genoma. Može simulirati diploidne genome sa SNP-ovima [8] i "briši/ubaci" polimorfizmom. Izlaz Wgsim-a čine dvije FASTQ datoteke unutar kojih se nalazi niz čitanja u čijim se zaglavljima bilježe stvarne koordinate početka čitanja, kao i broj polimorfizama te pogrešaka pri sekvencioniranju. Ulaz Wgsim-a predstavlja tekstualna datoteka unutar koje se nalazi sekvenca u FASTA formatu.

Primjer pokretanja simulacije:

```
wgsim -e 0.02 -d 400 -1 70 -2 100 -r 0.02 Danio_rerio.Zv9.60.dna.chromosome.9.fa
izlaz1.txt izlaz2.txt > preusmjeravanje.txt
[wgsim] seed = 1336867565
[wgsim_core] calculating the total length of the reference sequence...
[wgsim_core] 1 sequences, total length: 58232459
```

Primjer zapisa unutar jedne od izlaznih datoteka:

```
@9_8035764_8036233_1:0:0_0:0:0_0/2
ACAGCAAGAAGGTCGCTGGTTCGAGCCTCGGCTCAGTTGCCATTTCTGTGTGGAGTTTGCATGTTCTCCT
+
2222222222222222222222222222222222222222222222222222222222222222222222
@9_40163028_40163471_3:0:0_2:0:0_1/2
CTAAAAATGATAAATTCAACATTATGTTTTATTAGAATAAAAAAAAAAAATATTTATTAAGAATTAAATGC
+
2222222222222222222222222222222222222222222222222222222222222222222222
@9_6022137_6022585_0:0:0_2:0:0_2/2
ATCTAAGTTACGGAGAAATGCAGAGGTCTTTTTTTCTTTCTTTCGCCGTGCGGTATCAAACATTGCATTC
+
2222222222222222222222222222222222222222222222222222222222222222222222
@9_32695336_32695738_2:0:0_1:0:0_3/2
AAGTCATACTTAACTGACCGCTAGCAGTTTGTGAATCTTAATGGACAGCCTTCACAAATCTGCCCAGAAA
```

2.2. ART

ART [7] predstavlja set alata za simulaciju pročitanih nizova nukleotida. Ta je funkcionalnost esencijalna za testiranje i vrednovanje izlaznih podataka alata za sekvencioniranje druge generacije. ART simulira čitanja tako da oponaša stvaran proces sekvencioniranja s empiričkim modelima pogrešaka ili kvalitativnim profilima sažetima iz velike rekalibrirane baze podataka o sekvencama. ART također može simulirati tako da rabi svoj vlastiti model pogrešaka ili kvalitativni profil. Podržava simulaciju čitanja s jednog kraja te čitanja s oba kraja na tri veće komercijalne platforme za sekvencioniranje druge generacije (Iluminina Solexa, Rocheov 454 i Applied Biosystemov Solid). ART je implementiran u programskom jeziku C++ s optimiziranim algoritmima što ga čini vrlo efikasnim u simulaciji čitanja koja zapisuje u izlaznu tekstualnu datoteku u FASTQ formatu, nadalje ART je u potpunosti besplatan te javno dostupan u verzijama za Linux, Macintosh i Windows.

Primjer rada s alatom ART (za svaku pojedinu platformu) :

```
454 read simulation

Single-end reads
art_454 [ -s ] [ -p read_profile ] <INPUT_SEQ_FILE> <OUTPUT_FILE_PREFIX> <FOLD_COVERAGE>
Primjer:
art_454 seq_reference.fa ./outdir/dat_single_end 20
Paired-end reads
art_454 [ -s ] [ -p read_profile ] <INPUT_SEQ_FILE> <OUTPUT_FILE_PREFIX> <FOLD_COVERAGE>
<MEAN_FRAG_LEN> <STD_DE>
Primjer:
art_454 seq_reference.fa ./outdir/dat_paired_end 20 500 20

Illumina read simulation

Single-end reads
art_illumina [options] -i <INPUT_SEQ_FILE> -l <READ_LEN> -f <FOLD_COVERAGE> -o
<OUTPUT_FILE_PREFIX>
Primjer:
art_illumina -sam -i seq_reference.fa -l 50 -f 10 -o ./outdir/dat_single_end

Paired-end reads
art_illumina [options] -i <INPUT_SEQ_FILE> -l <READ_LEN> -f <FOLD_COVERAGE> -o
<OUTPUT_FILE_PREFIX> -m <MEAN_FRAG_LEN> -s <STD_DE>
Primjer:
art_illumina -p -sam -i seq_reference.fa -l 50 -f 20 -m 200 -s 10 -o
d./outdir/dat_paired_end
```

```
Mate-pair reads
art_illumina [options] -i <INPUT_SEQ_FILE> -l <READ_LEN> -f <FOLD_COVERAGE> -o
<OUTPUT_FILE_PREFIX> -m <MEAN_FRAG_LEN> -s <STD_DE>
Primjer:
art_illumina -mp -sam -i seq_reference.fa -l 50 -f 20 -m 2050 -s 50 -o
d./outdir/dat_paired_end

SOLID read simulation

Single-end reads
art_SOLiD [ -s ] [ -p read_profile ] <INPUT_SEQ_FILE> <OUTPUT_FILE_PREFIX> <READ_LEN>
<FOLD_COVERAGE>
Primjer:
art_SOLiD -s seq_reference.fa ./outdir/dat_single_end 32 10

Paired-end reads
art_SOLiD [ -s ] [ -p read_profile ] <INPUT_SEQ_FILE> <OUTPUT_FILE_PREFIX> <READ_LEN>
<FOLD_COVERAGE> <MEAN_FRAG_LEN> <STD_DE>
Primjer:
art_SOLiD seq_reference.fa ./outdir/dat_paired_end 25 10 500 20
```

2.3. DWGSIM

Dwgsim [16] se bazira na već ranije navedenom alatu Wgsim, no za razliku od istog može rukovati s ABI SOLID podacima kao i s različitim pretpostavkama o poravnanjima i poziciji obrisanih, odnosno umetnutih nukleotida .

Način uporabe je ostao isti kao i kod Wgsim-a tj. program se pokreće naredbom:

```
dwgsim[options]  <in.ref.fa> <out.prefix>
```

Nadalje, zaglavlje svakog čitanja sadrži više informacija, nego zaglavlje čitanja kod Wgsim-a. Dodatne informacije jesu: nasumičan kraj prvog čitanja, nasumičan kraj drugog čitanja, kraj prvog lanca, kraj drugog lanca te broj SNP-ova na prvom kraju. (SNP [5], engl. *"Single-nucleotide polymorphism"* označava varijaciju unutar DNA sekvence pri kojoj se jedan nukleotid u genomu ili nekoj drugoj dijeljenoj sekvenci razlikuje kod istih bioloških vrsta ili para kromosoma unutar iste jedinke. SNP-ovi su najčešće zastupljeni u nekodirajućim regijama).

Izlaz iz programa predstavljaju tri datoteke u FASTQ formatu.

3. Simulator

Simulator predstavlja command – line program napisan u programskom jeziku C standardom c99, čiji je osnovni zadatak simulacija pogrešaka pri čitanju, raznih mutacija, brisanja te ubacivanja nukleotida koja se mogu dogoditi na sekvenci DNA. Program se vrlo jednostavno prevodi i pokreće na bilo kojem unixolikom operativnom sustavu pri čemu se na standardni izlaz ispisuju opće informacije o sekvenci kao što su duljina, naziv, frekvencija pojavljivanja pojedinih nukleotida te vrijeme trajanja same simulacije. Ispis ovih podataka moguće je prema potrebi i preusmjeriti u datoteku ili na standardni ulaz nekog drugog programa. Način prevođenja i pokretanja pod operacijskim sustavom Linux, bit će detaljno razrađeno u poglavlju *3.3 Instalacija i pokretanje pod operacijskim sustavom Linux*. Ulaz programa predstavlja tekstualna datoteka koja sadrži sekvencu u FASTA formatu dok izlaz (osim osnovnih podataka o sekvenci) uključuje i dvije tekstualne datoteke koje sadrže zapise o svakom pojedinom očitanju proizvoljno određene duljine (prema osnovnim postavkama simulatora duljina jednog očitanja iznosi 70 pb).

Ulazni parametri programa jesu:

i) broj čitanja: budući da je lokacija početka svakog čitanja odabrana nasumično, potrebo je osigurati dovoljan broj čitanja kako bi se adekvatno pokrio cijeli genom. Inicijalno je broj čitanja postavljen na 1000000, no korisnik ga prema potrebi može i redefinirati pomoću zastavice -N. Veći broj čitanja rezultira većom pokrivenošću (tzv. engl. *"Depth of coverage*" – dubina prekrivanja). Ukupan broj čitanja i dubina prekrivanja povezani su sljedećom formulom:

$$N_{čitanja} = \frac{DuljinaReferentneSekvencexDubinaPrikrivanja}{IndividualnaDuljinaČitanja}$$

Veća dubina prekrivanja nema direktnog utjecaja na kasnije algoritme za poravnanja sekvenci budući da se svaki fragment poravnava nezavisno.

ii) duljina očitanja: prosječan broj parova baza po svakom očitanju, ovisno o alatu varira od 30b do 100b, u ovom alatu inicijalno iznosi 70b. Budući da isti podržava tzv. engl. "*pair end read*" duljinu pojedinog očitanja moguće je zasebno definirati

za svako očitanje (*"read"*) koristeći pri tom -1 n za prvi, odnosno -2 n zastavicu za drugi član para.

iii) Stopa brisanja/ubacivanja nukleotida: uz obične pogreške koje nastaju pri čitanju potrebno je stimulirati i kratka ubacivanja te brisanja nizova nukleotida koje, između ostalog, mogu nastati i kao posljedica samog čitanja, no mogu biti uzrokovane i varijacijama unutar populacije. Vrijednost ovog parametra inicijalno je postavljena na 0.15.

iv) BCER(engl. *"Base Calling Error Rate"*) [1] : procjena vjerojatnosti pogrešnog očitanja određene baze. Vrijednost BCER-a obično varira od 0.0% do 2.0%. Unutar ovog alata inicijalno je postavljena na 2.0% te ju korisnik po potrebi može redefinirati koristeći zastavicu -e . Pri promjeni nasumično odabrane baze (odabir baza vrši se prema uniformnoj razdiobi) vrijednosti vjerojatnosti prijelaza u bilo koju od preostalih baza su jednake i iznose 0.33.

v) Distanca između dva čitanja (udaljenost krajeva početka čitanja): srednja vrijednost je inicijalno postavljena na 500b. Indeks početka čitanja odabire se prema uniformnoj razdiobi dok se sama duljina ravna prema normalnoj razdiobi s već unaprijed definiranom standardnom devijacijom koja prema osnovnim postavkama simulatora iznosi 50b. (Sama standardna devijacija jest, također, ulazni parametar , stoga je moguće zadati i proizvoljnu vrijednost iste.)

vi) Frekvencija mutacija: implicitno određuje broj baza koje je potrebno mutirati. Inicijalno je postavljenja na 1.0%, te ju je moguće redefinirati pomoću zastavice -r [p/100] gdje p označava novu frekvenciju mutacija, ovaj parametar značajno utječe na brzinu same simulacije. Točan broj mutacija unutar pojedinog čitanja bilježi se u zaglavlju pripadajućeg FASTQ zapisa unutar jedne od izlaznih datoteka.

vii) Sjeme za generator slučajnih brojeva. Ukoliko se ne definira uzima se broj sekundi proteklih od siječnja 1970 (sistemsko vrijeme). Redefinirati ga je moguće zastavicom -S [novo_sjeme]. Iznos sjemena ispisuje se prilikom pokretanja programa. Odabirom fiksne vrijednosti sjemena moguće je ponoviti rezultate neke od prethodnih simulacija.

viii) Imena datoteka u koje će biti pohranjen izlaz u FASTQ formatu. Podaci o prvom čitanju (njih N) bivaju zapisani u prvu datoteku, odnosno drugu ukoliko se radi o čitanju s druge strane fragmenta ulazne sekvence, imena datoteka kao

argumenti pri pozivu programa su obavezna te se simulacija bez definiranih izlaznih tekstualnih datoteka neće moći izvršiti.

ix) Sama ulazna sekvenca u FASTA formatu.

x) Prosječna duljina procjepa: između ostalog, simulator pogreške prilikom čitanja generira i na način da izbacuje pojedine fragmente sekvence prosječne duljine -g pri čemu novonastale krajeve sljepljuje. Duljina procjepa može poprimiti proizvoljnu vrijednost pri čemu valja paziti da se ista ne postavi na apsurdno velik broj. Ovaj je parametar inicijalno postavljen na 1.

xi) Vjerojatnost da je u pojedinom čitanju došlo do ubacivanja nukleotida. Ova je vrijednost inicijalno postavljena 0.3, dakle 30% od ukupnog ubacivanja-brisanja nukleotida otpada upravo na ubacivanje. Iznos ovog parametra poprima vrijednost iz intervala [0-100]% te se podešava zastavicom -X.

xii) Iznos standardne devijacije prilikom generiranja čitanja. Vrijednost je inicijalno postavljena na 50b. Redefinira se zastavicom -D.

Svaki je ulazni parametar, sukladno potrebama, moguće proizvoljno odrediti, pri čemu se mora voditi računa da se unose razumne vrijednosti. Vrijeme izvođenja programa ovisi upravo o ulaznim parametrima, tj. veća frekvencija mutacija i procjepa te veći BCER direktno produljuju vrijeme izvođenja. Vrijeme izvođenja također linearno zavisi i o veličini ulazne sekvence.

Izlaz programa predstavlja niz sekvenci (kao posljedica N čitanja) u FASTQ formatu. Unutar zaglavlja svake sekvence zapisuju se prave koordinate početka čitanja (indeksi) kao i jedinstveni naziv sekvence, redni broj sekvence (heksadekadski), ukupan broj supstitucija, pogrešnih očitanja te brisanja/umetanja nukleotida.

3.1. Pojednostavljeni algoritam simulatora

Kroz sljedeći algoritam ukratko su opisani osnovni koraci rada simulatora:

1) Sa standardnog ulaza učitaj naziv datoteke koja sadrži sekvencu u FASTA formatu, proizvoljne vrijednosti parametara te nazive dvaju izlaznih datoteka u FASTQ ili txt formatu.

2) Inicijaliziraj generator pseudoslučajnih brojeva na vrijednost koju je putem komandne linije definirao korisnik ili u slučaju da ista nije definirana kao sjeme generatora postavi trenutno sistemsko vrijeme.

3) Iz ulazne datoteke učitaj u memoriju zadanu sekvencu kako bi ju bilo moguće nesmetano obrađivati.

4) Ponavljaj:

   - Nasumično odaberi dužinu čitanja iz normalne distribucije

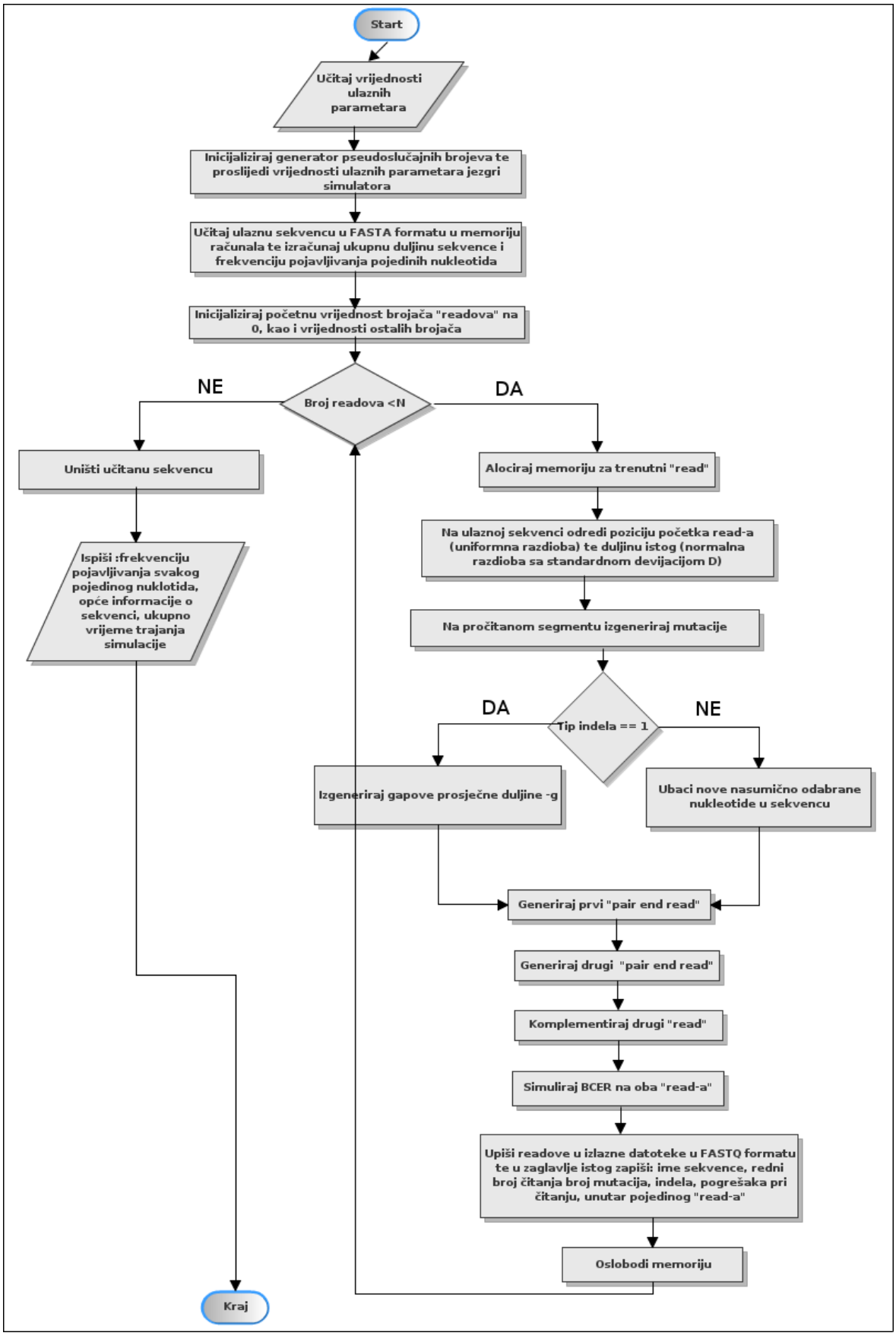
   - Nasumično odaberi poziciju od koje ćeš započeti čitanje (uniformna razdioba)

   - Koordinate početka i kraja čitanja pohrani u memoriju

   - U trenutni fragment ubacuj ili briši nukleotide iz istog te supstituiraj baze u skladu s parametrima koje je definirao korisnik ili u skladu s osnovnim postavkama simulatora.

   - U izlazne datoteke ispiši očitanje (engl. "read") nastalo čitanjem fragmenta s lijevog, odnosno komplement desnog kraja u FASTQ formatu. U zaglavlju zabilježi stvarne koordinate fragmenta, ukupan broj supstitucija, pogrešaka tijekom čitanja te broj nasumično ubačenih/obrisanih nukleotida.

5) Na standardni izlaz ispiši ukupno trajanje simulacije te opće informacije i statistiku za ulaznu sekvencu.



Slika 1.2 dijagram toka simulatora.

3.2. Formati ulaza i izlaza

3.2.1. FASTA format

Tijekom kratke povijesti bioinformatike izumljeni su različiti formati za skladištenje podataka o ispitnim sekvencama unutar tekstualnih datoteka. Zbog svoje jednostavnosti i lakoće parsiranja jezicima kao što su python, ruby i perl, najveću popularnost uživa upravo FASTA [11] format.

Najjednostavnije rečeno FASTA format čine linije teksta terminirane znakom novog retka kako bi se isti mogao lakše prikazati na zaslonu računala ili isprintati na list papira. Duljina jedne linije nije specificirana, no u praksi se pokazalo da ju je najbolje ograničiti na 80 znakova (prema osnovnim postavkama simulatora duljina jedne linije iznosi 70 znakova te ona ne smije biti duža od 120 znakova). Zaglavlje formata čini linija (ili linije) započete znakom “>” (katkad se rabi i znak “;” iako se njegova upotreba ne preporuća, budući da izvorno služi za označavanje komentara). Ono može sadržavati gotovo proizvoljne informacije, no tipično se u njemu nalazi jedinstveni identifikator sekvence te ime DNA ili gena od kojeg sekvenca potječe tipično odvojeno znakom “|” od ostatka dodatnih informacija o sekvenci koje slijede kao što je primjerice eksperiment pri kojem je sekvenca dobivena. Važno je napomenuti da zaglavlje može biti i prazno te da sekvenca može sadržavati i više zaglavlja koja se u tom slučaju odvajaju znakom ^A (control-A). Mnogi programi za obradu netom opisanog formata ne dozvoljavaju da zaglavlje čini više od jedne linije. Nakon zaglavlja slijedi sama sekvenca koja se ovisno o tome radi li se o proteinu ili nukleinskoj kiselini kodira odgovarajućim znakovima.

Kodovi za nukleinske kiseline određuju se prema sljedećoj tablici (kodovi za proteinske sekvence u ovom radu neće biti razmatrani):

Tablica 1.2 kodovi za nukleinske kisline [13]

Kod za nukleinsku kiselinu	Značenje	Mnemonički
Y	C, T ili U	pirimidinske(pYrimidine)
X	maskirano	-
W	A, T ili U	Slaba interakcija(Weak)
V	niti T niti U (i.e. A, C or G)	V nakon U
U	U	Uracil
T	T	Timin
S	C ili G	Jaka interakcija(Strong)
R	A ili G	purinske

Kod za nukleinsku kiselinu	Značenje	Mnemonički
N	A C G T U	A**N**y-bilo koja
M	A ili C	Baze s a**M**ino skupinom
K	G, T ili U	Baze koje su **K**etoni
H	ne G (npr., A, C, T or U)	**H** dolazi nakon G
G	G	Gvanin
D	ne C (npr. A, G, T or U)	**D** dolazi nakon C
C	C	Citozin
B	ne A (npr. C, G, T or U)	**B** dolazi nakon A
A	A	Adenin
-	Procjep neodređene duljine	-

Primjer sekvence u FASTA formatu:

```
>9 dna:chromosome chromosome:Zv9:9:1:58232459:1
TATTATTATTATTTAATTTTGTTATGTTGAAAAATGTGTATGCATACAACCAAGGCTTGT
TTTGACAATTTATTTAAAACTTGTGGTTAAAAATAATTAATTATAACATGAAAAAGTCCA
GTATAAGTCTGAAATTTCATTTAAAATGGTCTTTAAAAGCTCTTAAAAGTCCTAAATTTG
ACTTTTGACACCAGTAGGAAATCTGTTAATGTTTTATGCCTAATGTTCATGTAATAAGCA
GGATAAAGTACATGCAAGACGGTTAAAAATATTTAGCGTTTAGAACTTTTATAAGTCTTT
```

Unutar simulatora sekvence u FASTA formatu se parsiraju pomoću zaglavlja kseq.h čija je osnovna namjena upravo parsiranje FASTA/FASTQ formata. Zaglavlje, također, omogućava parsiranje oba formata sadržana u istoj tekstualnoj datoteci te je u kombinaciji s zlib-om (biblioteka za rad s podacima pod kompresijom) prilagođeno radu s kompresiranim datotekama, budući da ulazne sekvence mogu prelaziti i red veličine od 1 Gb.

3.2.2. FASTQ format

Nakon završetka simulacije rezultat obrade ulazne sekvence biva pohranjen u FASTQ [10] formatu po tzv. Sanger standardu (uz ovu verziju FASTQ formata u upotrebi je i Ilumina verzija o kojoj će nešto više biti rečeno u nastavku rada). FASTQ format je nastao iz običnog FASTA formata, no za razliku od istog prva linija ne započinje identifikatorom “>” već ga zamjenjuje “@”. FASTQ format, također, sadrži dvije nove linije kroz koje se bilježi ocjena kvalitete točnosti očitanja određene baze, kodirana jednim ASCII znakom, te identifikator sekvence (ova linija započinje znakom “+” nakon kojeg slijedi identifikator sekvence te ga je prema želji moguće i izostaviti).

Standardni FASTQ format tzv. Sanger FASTQ format rabi ASCII znakove korespodentne numeričkim vrijednostima od 33 (prvi znak kojeg je moguće ispisati na standardni izlaz :

"!") do 73 (znak "I").Ovakvo bilježenje ocjene kvalitete jest inicijalno bilo implementirano u programu Phread čiji je glavni zadatak određivanje ocjene kvalitete točnosti očitanja određenih baza unutar DNA.

Kvaliteta točnosti označava se s Q te računa prema sljedećoj formuli:

$$Q = -10\log(p)$$

gdje p predstavlja vjerojatnost krivog određivanja promatrane baze tzv. BCER – (engl. *"Base calling error rate"*). Primjerice ukoliko je p = 0.05 (postoji 5%-tna šansa da je baza određena krivo, pripadajuća vrijednost Q je 13, a odgovarajući ASCII znak "."). U programima za simulaciju pročitanih nizova nukleotida ovaj se parametar najčešće postavlja na vrijednost 0.02 ma da ga je moguće podesiti na bilo koju drugu proizvoljnu vrijednost (veće vrijednosti pridonose sporosti simulacije). Budući da on ostaje fiksan tijekom čitave simulacije, sve se baze unutar sekvence kodiraju jednakim ASCII znakom.

Primjer izlaza simulatora za BCER = 0.02 :

```
@9_49928595_49929101_4:25:7_2:22:10_0/1
AATCATGGTCGAAGGCCATTAAACGGGGCCCATGGAAGCGTCCGTTATTAAGTGCCCGTCCCGACTTGGA
+
2222222222222222222222222222222222222222222222222222222222222222222222
```

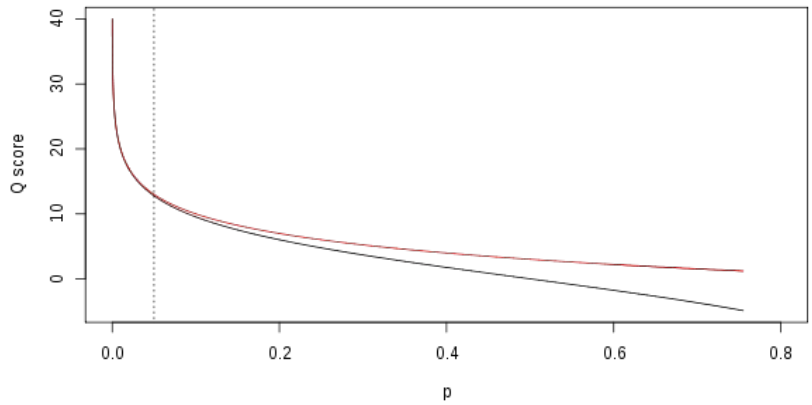
3.2.2.1 Ilumina verzija

Sekvenca procesirana Ilumina cjevovodima rezultira formatom koji je generalno sličan prethodno opisanom, međutim sadrži dvije bitne razlike. Umjesto kodiranja znakovima od "!" do "I" (ASCII vrijednosti od 33 do 73) kodiranje započinje ASCII znakom "B" (vrijednost 66, procjena kvalitete: 2) te završava sa "h" (vrijednost 73, procjena kvalitete: 40). Kako bi se dobila navedena korespodentna vrijednost potrebno je od vrijednost Q oduzeti 64 (u prethodno opisanoj verziji, koja ujedno predstavlja i izlaz simulatora, oduzima se 33). Minimalna moguća vrijednost je 2 koja odgovara vjerojatnosti pogreške većoj od 0.75. Dodatna posljedica ovakvog označavanja kvalitete jest da "@" više nije dio palete mogućih znakova, što je rezultiralo smanjenjem mogućnosti dvoznačnog parsiranja. Nadalje Id-jevi Ilumina sekvenci su sistematična imena, npr. u imenu: @HWI-EAS385_0064:3:24:1024:5245#0/1, @HWI-EAS385_0064 određuje stroj, 3 označava treću liniju u toku ćelija, 24 je broj pločice u toku ćelija, 1024 i 5245 su x i y koordinate pločice respektivno, dok "#0" označava da uzorak nije multipleksiran. Ukoliko bi uzorak ipak bio multipleksiran to bi bilo označeno s "#ATCACG". Konačno /1 označava prvi član para , drugi odgovarajući član biva označen s /2 .

Kvaliteta točnosti označava se s Q te računa prema sljedećoj formuli:

$$Q = -10\log\left(\frac{p}{1-p}\right)$$

Iako su mapiranja asimptotski jednaka kod nižih vrijednosti BCER-a za obije vrste formata, do odstupanja dolazi kod viših vrijednosti što je i jasno vidljivo na sljedećem priloženom grafu.



Slika 1.3 Odnos između Q i p za Sanger (crvena krivulja) i Solexa format (crna krivulja). Vertikalna točkana linija postavljena je na p=0.05, odnosno Q=13

3.3. Instalacija i pokretanje simulatora pod operacijskim sustavom Linux

Prije početka instalacije samog programa potrebno je instalirati zlib zaglavlje na sljedeći način:

Sa stranice http://www.zlib.net skinuti aktualnu verziju zlib-a (kompresirana datoteka oblika tar.gz) te u terminal unjeti sljedeće naredbe:

```
tar -xvzf zlib-1.2.5.tar.gz

cd zlib-1.2.5

./configure –prefix=/usr/local/zlib

make

sudo make install

aptitude install zlib1g-dev #instalacija samog zaglavlja
```

Za pohranu programa preporuća se stvaranje novog kazala proizvoljnog naziva primjerice:

```
mkdir rezultati_simulacije
```

Program i pripadna zaglavlja (“kseq.h”, “genran.h”) potrebno je pohraniti unutar istog, prethodno kreiranog kazala.

Nakon pozicioniranja u prethodno kreirano kazalo (naredba cd) program se jednostavno prevodi sljedećom naredbom:

```
gcc -o simulator -lz -lm -std=gnu99 simulator_v1.c
```

Važna napomena: prilikom prevođenja potrebno je naglasiti prevoditelju da se radi o c99 standardu (-std=gnu99). C99 standard za razliku od osnovnog standarda dozvoljava upotrebu jednorazinskih funkcija, deklaraciju varijabli unutar bloka (prema osnovnom standardu varijable se deklariraju na početku bloka), uvodi nove tipove podataka kao što su: long long int, boolean, complex (za baratanje kompleksnim brojevima) te nova zaglavlja kao što su stdbool.h, complex.h, tgmath.h, inttypes.h. Važno je napomenuti da ga karakterizira mnogo agresivnija optimizacija koda.

Prikaz funkcije koja na stderr ispisuje izbornik programa u slučaju da korisnik unese ulazne parametre u krivom formatu (prikaz izbornika moguće je dobiti i ukucavanjem naziva programa bez argumenata: ./simulator):

```
static int simu_usage(){
        fprintf(stderr,"***********************************************************\n");
        fprintf(stderr,"Program: simulator (short read simulator)\n");
        fprintf(stderr,"Version %s\n",PACKAGE_VERSION);
        fprintf(stderr,"\nUsage: ./simulator [options] <in_seq.fa> <out_read1.fq> <out_read2.fq>\n\n");
        fprintf(stderr,"Options: -r FLOAT rate of mutations [default 0.001]\n");
        fprintf(stderr,"         -e FLOAT base error rate [default 0.02]\n");
        fprintf(stderr,"         -1 INT length of first read [default 70bp]\n");
        fprintf(stderr,"         -2 INT length of second read [default 70bp]\n");
        fprintf(stderr,"         -N INT number of read pairs [default 1000000]\n");
        fprintf(stderr,"         -R FLOAT fraction of indels [default 0.15]\n");
        fprintf(stderr,"         -S INT seed for random generator [default -1]\n");
        fprintf(stderr,"         -d INT outer distance between the two ends [default 500pb]\n");
        fprintf(stderr,"         -g INT average gap size [default 1]\n");
        fprintf(stderr,"         -D INT standard deviation [default 50pb]\n");
        fprintf(stderr,"         -X FLOAT probability that indel is extended [default 0.3]\n");
        fprintf(stderr,"\n***********************************************************\n");
        return 1;
}
```

Program se pokreće sljedećom naredbom:

```
./simulator -N [xxx] -r [yyy] -e [zzz] -S [aaa] -R [www] -d [bbb] -1 [l_1] -2
[l_2] -D [iii] -X [fff] -g [qqq] <ime_sekvence.fa> <izlaz1.fq> <izlaz2.fq>
```

gdje:

-N [xxx] označava ukupan broj čitanja. Parametar može poprimiti bilo koju cjelobrojnu vrijednost, primjerice -N 500000. Prema osnovnim postavkama programa iznosi 1000000.

-r [yyy] označava frekveciju mutacija, može poprimiti vrijednosti od 0 do 1 tj. 0 do 100%, primjerice -r 0.1. *(*važna napomena: visoka frekvencija mutacija značajno usporava rad programa!)*

-e [zzz] označava BCER, tipično se vrijednost BCER-a u simulatorima kreće od 0 do 0.06 .

-S [aaa] označava sjeme za generator slučajnih brojeva, [aaa] jest proizvoljna cjelobrojna konstanta. Ukoliko se za sjeme postavi vrijednost manja od 1, ista će se automatski zamijeniti s vrijednošću sistemskog vremena. Za istu vrijednost sjemena generira se isti izlaz, što omogućava ponavljanje pokusa.

-R [www] označava stopu brisanja/ubacivanja nasumično generiranih baza u pojedini fragment, može poprimiti bilo koji realan broj od 0 do 1. Prema osnovnim postavkama programa poprima vrijednost 0.15, točan broj brisanja/ubacivanja unutar pojedinog čitanja (s bilo kojeg kraja) zabilježen je u zaglavlju odgovarajućeg FASTQ zapisa unutar izlazne datoteke.

-d [bbb] označava distancu između dva paralelna čitanja, inicijalno je postavljena na 500b, može poprimiti bilo koju cjelobrojnu vrijednost pri čemu valja paziti da unesena distanca ne premaši duljinu sekvence. U tom slučaju simulacija se naglo prekida i izlazi se iz programa.

-1 označava duljinu čitanja s prvog kraja, dok -2 označava duljinu čitanja s drugog kraja

-X [fff] označava vjerojatnost da prilikom generiranja čitanja došlo do ubacivanja novih nukleotida, ova je vrijednost inicijalno postavljena na 0.3, tj. 30% stope brisanja/ubacivanja opada upravo na ubacivanje, valja napomenuti da je ubacivanje skuplja operacija od brisanja, stoga vrijednost ovog parametra značajno utječe na sporost simulacije.

-g [ggg] označava srednju duljinu procjepa. Inicijalno je ova vrijednost postavljena na 1, iznos pojedine duljine procjepa ravna se prema Poissonovoj razdiobi upravo oko definirane srednje vrijednosti.

-D [iii] označava iznos standardne devijacije prilikom generiranja pojedinog čitanja (duljine čitanja ravnaju se prema normalnoj razdiobi). Prema osnovnim postavkama programa iznosi 50b.

<ime_sekvence.fa>, **<izlaz1.fq>**, **<izlaz2.fq>** su proizvoljne datoteke. **<ime_sekvence.fa>** sadrži ulaznu sekvencu u FASTA formatu, dok su preostale dvije izlazi u FASTQ formatu.

Prilikom pokretanja programa bilo koji od parametara (osim naziva datoteka) je moguće izostaviti pri čemu će isti poprimiti već unaprijed definiranu vrijednost.

Za ispis svih mogućih parametara i objašenjenja dovoljno je ukucati samo ime programa: ./simulator pri čemu se poziva prethodno navedena funkcija simu_usage() koja će parametre zajedno s unaprijed definiranim vrijednostima ispisati na stderr.

Primjer pokretanja programa s proizvoljno definiranim ulaznim parametrima:

```
./simulator -e 0.02 -r 0.01 -d 400 -N 1000000 -1 100 -2 70 -g 3 -X 0.1 -S 1991
Mus_musculus.NCBIM37.61.dna_rm.chromosome.1.fa iz1.txt iz2.txt
[main] simulator seed = 1991
[core] calculating the total length of the sequnce...
[core] name: 1
[core] comment: dna_rm:chromosome chromosome:NCBIM37:1:1:197195432:1
[core] 1 sequences, total length: 197195432
[core] frequency per sequence [1/1] A - 15.671207 | C - 10.681949 | G - 10.651283
| T - 15.627029 | *N - 47.368532 - unknown nucleotides (percentage)
[core] transferring sequence into memory and generating errors...
[main] return value: 0 OK
[main] Total time taken: 46.030000 sec
```

Frekvencija mutacija u gornjem primjeru postavljena je na vrijednost 0.01 (1%). Broj čitanja ograničen je na 1000000. Ulazna sekvenca je preuzeta s http://www.ensembl.org/index.html te prethodno pohranjena u isto kazalo kao i program. Stupanj pogrešnog očitavanja baza postavljen je na vrijednost 0.02 (2%) dok se izlaz pohranjuje u dvije tekstualne datoteke iz1.txt i iz2.txt. Memorija potrebna za pohranu sekvence alocira se dinamički te je jednaka ukupnoj duljini sekvence pomnoženoj s veličinom podatkovnog tipa char. Memorija za pojedino očitanje također se alocira dinamički te poprima vrijednost duljine očitanja također pomnožene s veličinom znakovnog podatkovnog tipa (char).

Sve važnije funkcije programa kao što su učitavanje sekvence u memoriju, generiranje mutacija i procjepa te računanje postotka pojedinih nukleotida obavljaju se unutar jezgre istog (jezgru simulatora predstavlja funkcija core()). Jezgra prilikom obrade sekvence poziva niz pomoćnih funkcija kao što su funkcije za generiranje slučajnih brojeva odgovarajućom razdiobom ,funkcije za promjenu vrijednosti pojedine baze te funkcija za generiranje BCER-a.

Vrijednosti pojedinih parametara prosljeđuju se programu putem argumenata komandne linije te parsiraju funkcijom optarg() pri čemu varijabla optind predstavlja indeks elementa unutar argv koji će sljedeći biti obrađen.

3.4. Simulacija

Za potrebe testiranja programa odabrano je nekoliko testnih sekvenci preuzetih sa http://www.ensembl.org/info/about/species.html. Bitno je da su sekvence različitih veličina te da sadrže različit postotak neutvrđenih nukleotida. Vrijednosti ulaznih parametara tijekom prve serije testiranja će biti fiksirane kako bi se mogla utvrditi ovisnost vremena izvođenja i duljine sekvence te ukupnog zauzeća memorije tijekom simulacije. U drugoj seriji testiranja pratit će se utjecaj različite vrijednosti ulaznog parametara BCER na ukupno vrijeme izvođenja programa.

```
lexy@ubuntu:~/test_projekt$ ./simulator
********************************************************
Program: simulator (short read simulator)
Version 0.0.1

Usage: ./simulator [options] <in_seq.fa> <out_read1.fq> <out_read2.fq>

Options: -r FLOAT rate of mutations [default 0.001]
         -e FLOAT base error rate [default 0.02]
         -1 INT length of the first read [default 70bp]
         -2 INT length of the second read [default 70bp]
         -N INT number of read pairs [default 1000000]
         -R FLOAT fraction of indels [default 0.15]
         -S INT seed for random generator [default -1]
         -d INT outer distance between the two ends [default 500pb]
         -g INT average gap size [default 1]
         -D INT standard deviation [default 50pb]
         -X FLOAT probability that indel is extended [default 0.3]

********************************************************
lexy@ubuntu:~/test_projekt$ ./simulator -e 0.02 -r 0.01 -d 400 -N 1000000 -1 100 -2 70 -X 0.5 -S 1991 -R 0.05 Danio_rerio.Zv9.60.dna.chromosome.25.fa iz1.txt
 iz2.txt
[main] simulator seed = 1991
[core] calculating the total length of the sequnce...
[core] name: 25
[core] comment: dna:chromosome chromosome:Zv9:25:1:38499472:1
[core] 1 sequences, total length: 38499472
[core] frequency per sequence [1/1] A - 31.518708 | C - 18.217008 | G - 18.282643 | T - 31.755781 | *N - 0.225860 - unknown nucleotides (percentage)
[core] transferring sequence into memory and generating errors...
[main] return value: 0 OK
[main] Total time taken: 21.240000 sec
```

Prvi read (70pb):

```
@1_113060550_113061001_1:1:0_2:4:0_1/1
ACCTAGTAAAGGGGAGTACCAAGTAATGGGGAGTACAAACCTGAAAGAGATAGGAACAGCTATCTTGAGGG
+
2222222222222222222222222222222222222222222222222222222222222222222222
@1_38652045_38652411_0:3:2_2:2:0_2/1
GTAGACACTTTTTGAAGGAAGATTTTTTTGTTATTATTACTAACTGTAAATCCACTGCTTTGCATAGCCCG
+
2222222222222222222222222222222222222222222222222222222222222222222222
...
```

Drugi read (100pb):

```
@1_113060550_113061001_2:1:0_2:4:0_1/2
TTACAATGTTGTGTCAATTTCTGCTGTACAGTCATACATATACAGACACATTCTTTCTTTCCTACTATCTTCCTTCATGATCTATCCCAAGAGATTGAAC
+
22222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222
@1_38652045_38652411_3:3:2_2:2:0_2/2
TGCAGTGTAATCTTAATGACCAGGAAAACGGAGGAGAAAAACCCATAACATCTAGTATTCGTTGAACCACCACATGTTTTGAGGTGGTGAAATGTTGGAA
+
22222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222
...
```

Slika 1.4 primjer simulacije i izgled korisničkog sučelja simulatora.

U sljedećoj tablici dan je prikaz testnih sekvenci:

Tablica 1.3 Prikaz testnih sekvenci i veličine istih u Mb.

Naziv Sekvence	**Veličina(Mb)**
Drosophila_melanogaster.BDGP5.65.dna_rm.chromosome.XHet.fa	0.20
Drosophila_melanogaster.BDGP5.65.dna.chromosome.3LHet.fa	2.5
Drosophila_melanogaster.BDGP5.65.dna_rm.chromosome.U.fa	9.7
Drosophila_melanogaster.BDGP5.65.dna_rm.chromosome.Uextra.fa	28.1
Danio_rerio.Zv9.60.dna_rm.chromosome.25.fa	37.3
Canis_familiaris.BROADD2.48.dna.chromosome.29.fa	43.5
Gorilla_gorilla.gorGor3.1.64.dna.chromosome.18.fa	76.4
Homo_sapiens.GRCh37.66.dna.chromosome.HSCHR12_1_CTG2.fa	129.8
Mus_musculus.NCBIM37.61.dna_rm.chromosome.1.fa	191.2
Sus_scrofa.Sscrofa10.2.67.dna.chromosome.1.fa	305.7
Monodelphis_domestica.BROADO5.50.dna.chromosome.3.fa	511.9

Vrijednosti ulaznih parametara u prvom krugu testiranja, jednake su za sve testne sekvence te iznose:

- -e 0.02 (BCER)
- -r 0.01 (frekvencija supstitucija)
- -d 400 srednja udaljenost dva kraja čitanja
- -N 1000 000 ukupan broj čitanja
- -1 100 duljina prvog očitanja
- -2 70 duljina drugog očitanja
- -g 3 srednja vrijednost procjepa
- -X 0.1 vjerojatnost da je došlo do ubacivanja novih baza u sekvencu
- -S 1991 sjeme za generator pseudoslučajnih brojeva
- iz1.txt, iz2.txt – izlazne tekstualne datoteke.

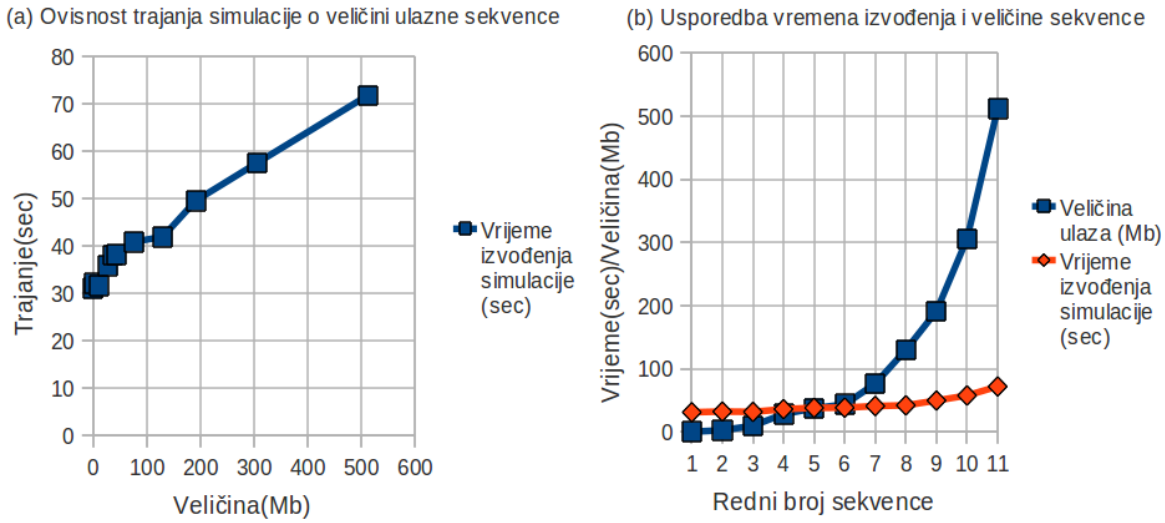
U drugom krugu testiranja ulazna sekvenca se ne mijenja, dok vrijednost parametra BCER raste linearno od 0 do 0.4 [0-40]%. Vrijednosti svih ostalih parametara su fiksirane.

3.5. Rezultati i diskusija

Tablica 1.4 Rezultati izvođenja simulacije. U prvom stupcu tablice nalazi se naziv same sekvence. U drugom i trećem stupcu bilježi se ukupno vrijeme trajanja simulacije bez, odnosno s optimiranjem prilikom prevođenja. Posljednji stupac sadrži podatak o maksimalnom zauzeću memorije tijekom simulacije.

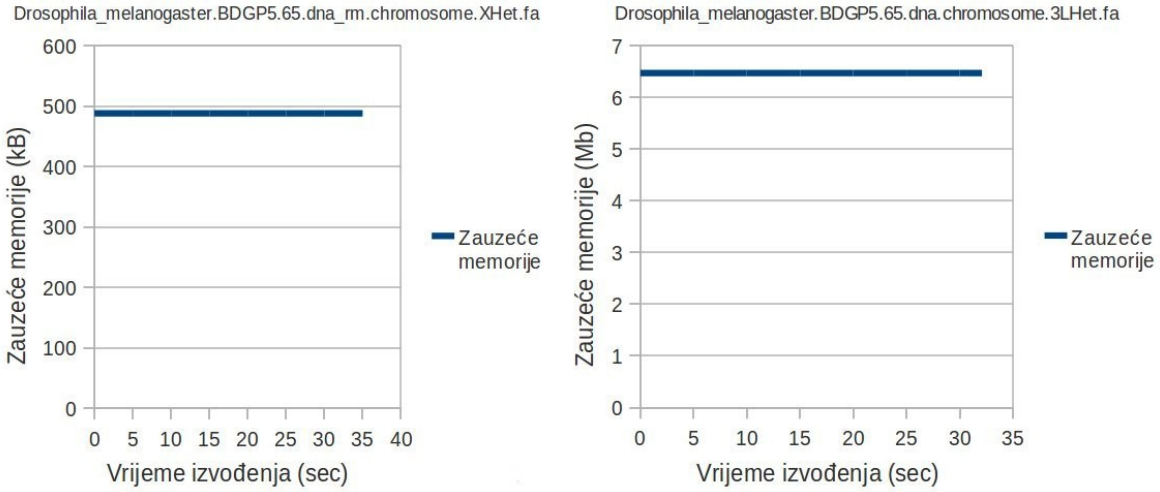
Naziv sekvence	**Vrijeme izvođenja(sec)**	**Vrijeme izvođenja (optimizacija prilikom prevođenja)**	**Zauzeće memorije(max)**
Drosophila_melanogaster.BDGP5.65.dna_rm.chromosome.XHet.fa	31.06	29.13	488.1kB
Drosophila_melanogaster.BDGP5.65.dna.chromosome.3LHet.fa	32.09	29.59	6.469Mb
Drosophila_melanogaster.BDGP5.65.dna_rm.chromosome.U.fa	31.62	28.54	25.62Mb
Drosophila_melanogaster.BDGP5.65.dna_rm.chromosome.Uextra.fa	35.91	32.14	59.69Mb
Danio_rerio.Zv9.60.dna_rm.chromosome.25.fa	38.07	33	100.7Mb
Canis_familiaris.BROADD2.48.dna.chromosome.29.fa	38.29	33.28	106.8Mb
Gorilla_gorilla.gorGor3.1.64.dna.chromosome.18.fa	40.91	34.59	203.2Mb
Homo_sapiens.GRCh37.66.dna.chromosome.HSCHR12_1_CTG2.fa	41.97	33.85	255.7Mb
Mus_musculus.NCBIM37.61.dna_rm.chromosome.1.fa	49.59	37.41	444.1Mb
Sus_scrofa.Sscrofa10.2.67.dna.chromosome.1.fa	57.56	42.13	812.74Mb
Monodelphis_domestica.BROADO5.50.dna.chromosome.3.fa	71.8	49.23	1015.52 Mb

U nastavku slijedi grafički prikaz ovisnosti trajanja simulacije o veličini ulazne sekvence. Ta je ovisnost linearna s manjim odstupanjima, tj. trajanje simulacije raste proporcionalno s veličinom ulazne sekvence. Također, data je i međusobna usporedba vremena izvođenja i veličine sekvence. Svaka je od tih veličina prikazana zasebnom linijom na istom grafu za svaku pojedinu sekvencu. Unatoč gotovo eksponencijalnom rastu veličine ulaza, porast vremena izvođenja je mali, gotovo konstantan.

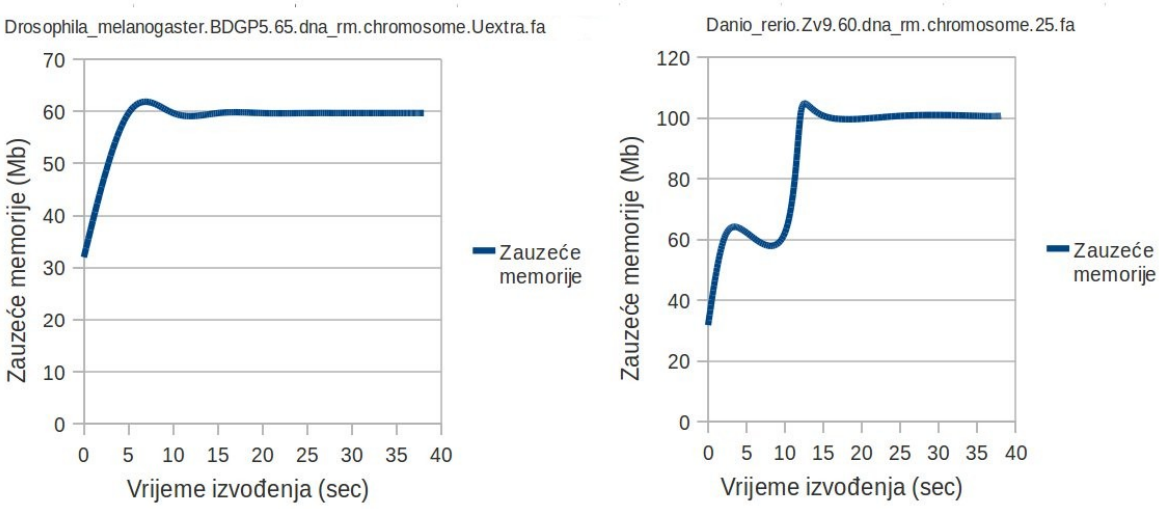


Slika 1.5 Grafički prikaz ovisnosti trajanja simulacije o veličini ulazne sekvence (linearan porast vremena izvođenja, lijevo) te usporedba vremena izvođenja i veličine sekvence (desno) za svaku pojedinu testnu sekvencu.

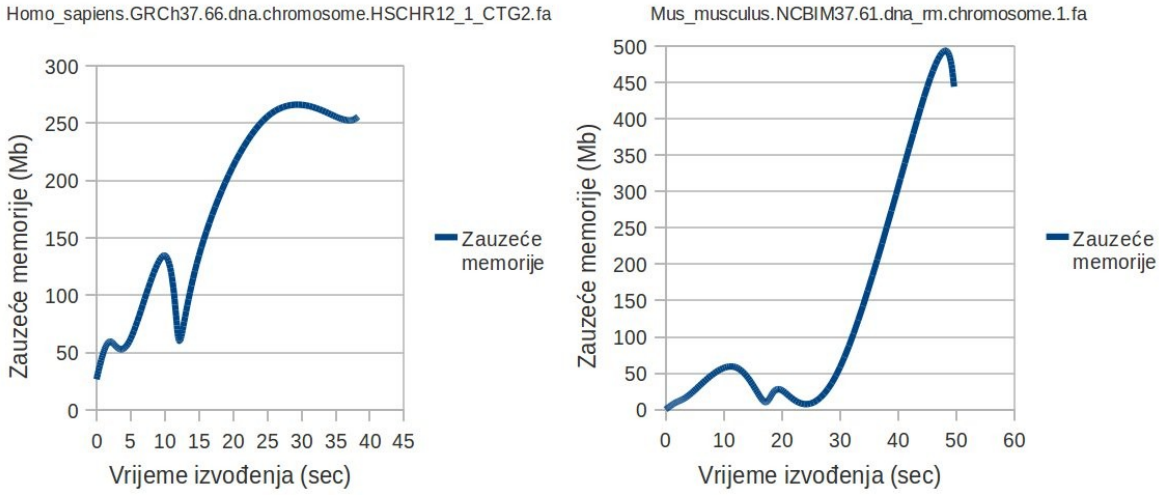
Kroz sljedećih nekoliko grafova prikazana je količina memorije koju simulator rabi tijekom čitavog procesa obrade pojedinih sekvenci. Za male sekvence zauzeće memorije u vremenu je konstantno dok se kod većih sekvenci nakon početnih varijacija, količina memorije u uporabi stabilizira oko vrijednosti približno jednake trenutnoj maksimalnoj iskorištenosti memorije (iznos maksimalne iskorištenosti memorije, za svaku pojedinu testnu sekvencu, naveden je u Tablici 1.4)



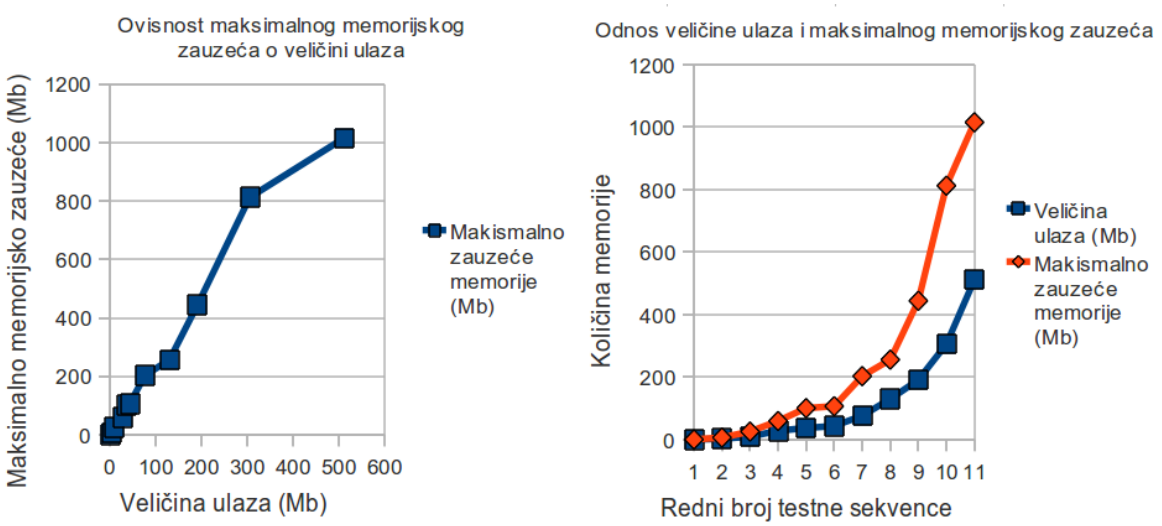
Slika 1.6 Grafički prikaz ukupnog zauzeća radne memorije od strane simulatora tijekom čitave simulacije. Zauzeće memorije za manje sekvence je gotovo konstantno (veličina testne sekvence na lijevom grafu je 0.2 Mb, a na desnom 2.5 Mb).



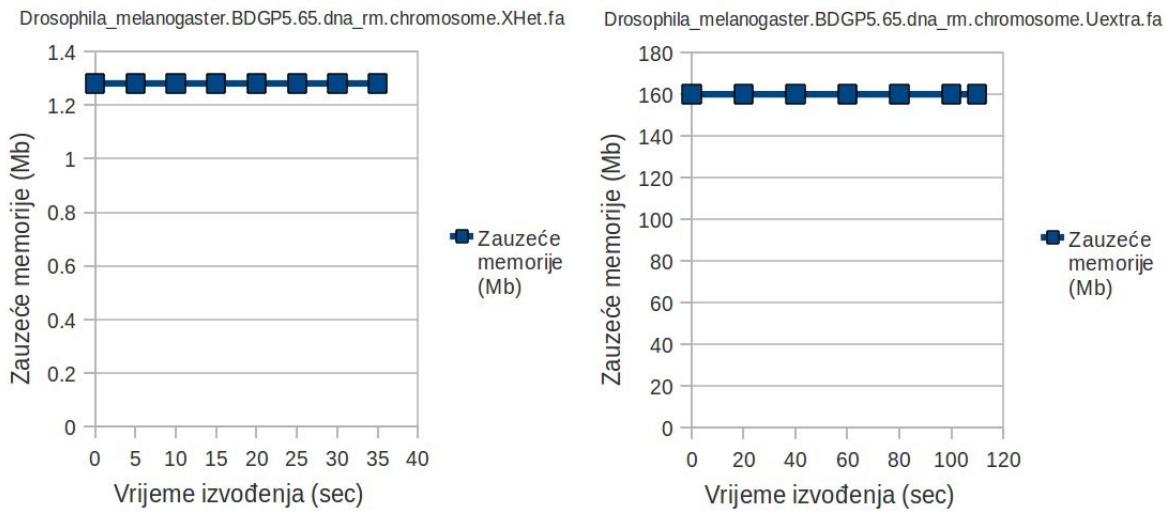
Slika 1.7 Grafički prikaz zauzeća memorije od strane simulatora tijekom čitave simulacije. Red veličine testnih sekvenci u ovom primjeru se kreće oko nekoliko desetaka Mb. Na slici su jasno vidljive početne varijacije u količini zauzete memorije, no ista se nakon određenog vremena stabilizira oko vrijednosti bliske maksimalnom iskorištenju memorije (veličina testne sekvence na lijevom grafu je 28.1 Mb, a na desnom 37.3 Mb).



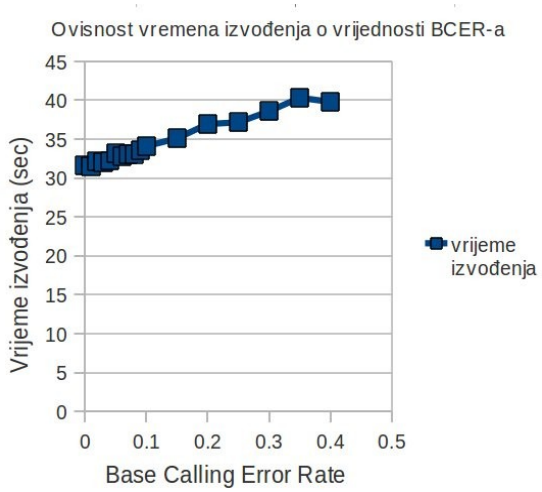
Slika 1.8 Grafički prikaz zauzeća memorije tijekom čitave simulacije od strane simulatora za sekvence reda veličine nekoliko stotina Mb. Na slici su jasno vidljive izrazite varijacije u količini memorije koju program rabi. Maksimum iskorištenosti memorije se postiže relativno brzo.



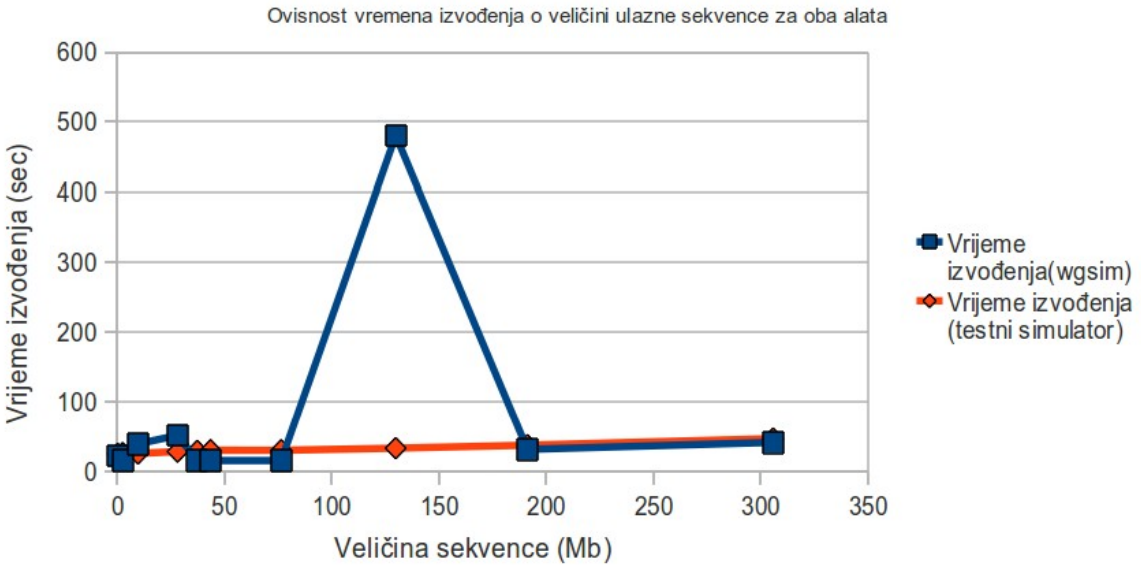
Slika 1.9 Grafički prikaz ovisnosti maksimalnog zauzeća radne memorije tijekom rada simulatora o veličini ulazne sekvence (lijevo) te prikaz odnosa veličine pojedinog ulaza i pripadnog maksimalnog zauzeća memorije (desno). Maksimalno memorijsko zauzeće otprilike odgovara dvostrukoj veličini ulazne sekvence na što se još pridodaje i maksimalna moguća veličina samog fragmenta za zadane ulazne parametre te sve ostale pomoćne alokacije memorije.



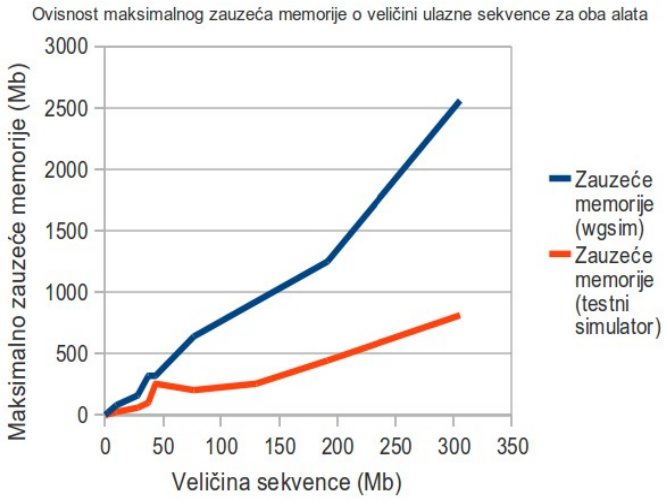
Slika 1.10 Grafički prikaz količine zauzete memorije tijekom simulacije kod alata WGSIM. Grafovi su dati isključivo za usporedbu performansi. Količina zauzete memorije je konstanta te iznosi 1.281 Mb za lijevu odnosno 160Mb za desnu sekvencu (te vrijednosti kod testnog simulatora iznose 500kB za lijevu, odnosno maksimalno 60Mb za desnu testnu sekvencu). Napomena: testiranje je vršeno s prethodno navedenim parametrima.



Slika 1.11 Prikaz ovisnosti vremena trajanja simulacije o vrijednosti BCER-a. Na grafu je jasno vidljiv polagani porast vremena izvođenja u skladu s porastom vrijednosti BCER-a. Veća vrijednost BCER-a uvjetuje većem broju redefinicija vrijednosti pojedinih baza. Složenost tog postupka iznosi O(n) gdje n predstavlja ukupan broj pogrešaka po čitanju dobiven Poissonovom razdiobom. Računanje pseudoslučajnog broja netom spomenutom razdiobom jest vremenski vrlo skupa operacija što pridonosi sporosti cjelokupnog postupka. Kako bi vrijeme izvođenja programa bilo u prihvatljivim granicama potrebno je ovaj parametar postaviti na što manju vrijednost. U praksi se vrijednost BCER-a najčešće kreće između 0.01 i 0.06 ([1-6]%).



Slika 1.12 Prikaz ovisnosti vremena trajanja simulacije o veličini ulazne testne sekvence za oba alata (Wgsim, testni simulator). Važno je napomenti da je usporedba poprilično labava budući da se simulatori u određenim parametrima ipak razlikuju. Kako bi se razlike donekle izjednačile maknuta je zastavica -g 3 s ciljem da se smanji broj obrisanih nukleotida po jednoj iteraciji. Wgism u osnovi obrađuje sekvence malo brže od testnog simulatora, no nailazi na probleme pri obradi sekvence rednog broja 9 *(Mus_musculus.NCBIM37.61.dna_rm.chromosome.1.fa)*za koju obrada traje duže od 5 min. Također, nešto sporije obrađuje testne sekvence pod rednim brojem 3 *(Drosophila_melanogaster.BDGP5.65.dna_rm.chromosome.U.fa)* te pod rednim brojem 4 *(Drosophila_melanogaster.BDGP5.65.dna_rm.chromosome.Uextra.fa)*.



Slika 1.12 Ovisnost maksimalnog zauzeća memorije o veličini ulazne sekvence za alate WGSIM i testni simulator. Za male testne sekvence maksimalna potrošnja memorije kod WGSIM-a je dvostruko veća od maksimalne potrošenje memorije kod testnog simulatora. S rastom veličine ulaznih testnih sekvenci razlika u iskorištenju memorije se još više povećava. Za ulaznu sekvencu veličine 305.7 Mb maksimalno memorijsko zauzeće kod alata WGSIM iznosi 2560 Mb dok je isto kod testnog simulatora značajno manje (812.74 Mb).

Važne napomene: sva testiranja izvršena su na računalu čiji je procesor modela Pentium(R) Dual-Core (2.00GHz), te radna memorija (RAM) 4.00GB. Svi grafički prikazi (izuzev zadnja 2) dobiveni su prevođenjem bez zastavica -On za optimizaciju. Uključivanje zastavica u proces prevođenja značajno smanjuje ukupno vrijeme trajanja simulacije (Tablica 1.4).

3.6. Primjena kontinuiranih i diskretnih distribucija u procesu simulacije

Kod generiranja fragmenata program se služi sljedećim distribucijama:

-uniformnom: Ukoliko je područje vrijednosti varijable $S=\{x1,x2,..,xn\}$ diskretan skup tada slučajna varijabla koja poprima vrijednost iz skupa S s jednakim vjerojatnostima opisuje pokus biranja na sreću elementa skupa S te je vjerojatnost njezine realizacije jednaka:

$$P\left(X=x_k\right)=\frac{1}{n}\,k=1,2,\ldots,n$$

Prilikom odabira početka generiranja svakog pojedinog čitanja, prema ovoj razdiobi, vrijednosti skupa S predstavljaju upravo indeksi pojedinih baza unutar sekvence. Uniformna se razdioba koristi također prilikom generiranja mutacija, tj. svakoj je bazi

pridružen jedan prirodan broj, vjerojatnost pojavljivanja svakog od pojedinih brojeva je jednaka te iznosi 1/3, što zapravo odgovara vjerojatnosti slučajne promjene vrijednosti baze. Pretpostavimo sad da je skup S interval [a,b]. Točaka unutar tog intervala ima beskonačno mnogo. Zamislimo postupak odabira na sreću nekog broja unutar tog intervala. Kažemo da biramo na sreću broj iz intervala [a,b] ako je vjerojatnost da će on biti odabran unutar nekog podintervala proporcionalna duljini tog istog podintervala, dakle, za slučajnu varijablu X kažemo da je uniformno (jednoliko) distribuirana na intervalu [a,b] ako je zadana funkcijom gustoće:

$$f(x)=\frac{1}{b-a}, a \le x \le b$$

-normalnom (Gaussovom): potrebno je naglasiti da je normalna razdioba najvažnija neprekinuta razdioba ukoliko uzmemo u obzir učestalost i važnost modela u kojima se ona pojavljuje. Prvenstveni razlog tomu je što se ta razdioba javlja kao granična u svim situacijama kada je slučajna varijabla dobivena kao zbroj velikog broja međusobno nezavisnih pribrojnika. Slučajna varijabla X ima normalnu razdiobu s parametrima:

$$f(x)=\frac{1}{\sigma\sqrt{2\pi}}\exp\left(\frac{-(x-a)^2}{2\sigma^2}\right)$$

Unutar programa vrijednosti parametara a i sigma moguće je proizvoljno odabrati prema potrebi.

Srednja vrijednost čitanja iznosi d sa standardnom devijacijom D, i jedan i drugi parametar su varijabilni te ih je moguće proizvoljno definirati u skladu s potrebama.

Sljedeći isječak koda predstavlja funkciju koja računa pseudoslučajan broj prema normalnoj razdiobi:

```
50. double ran_normal()
51. {
52. static int iset = 0;
53. static double gset;
54. double fac, rsq, v1, v2;
55. if (iset == 0) {
56.     do {
57.         v1 = 2.0 * ran_uniform() - 1.0;
58.         v2 = 2.0 * ran_uniform() - 1.0;
59.         rsq = v1 * v1 + v2 * v2;
60.     } while (rsq >= 1.0 || rsq == 0.0);
61.     fac = sqrt(-2.0 * log(rsq) / rsq);
62.     gset = v1 * fac;
63.     iset = 1;
64.     return v2 * fac;
65. } else {
66.     iset = 0;
```

```
67.     return gset;
68.   }
69.}
```

-Poissonovom: prema ovoj razdiobi ravna se duljina svakog pojedinog procjepa. Poissonovu razdiobu možemo dobiti kao ograničeni slučaj binomne, kad broj pokusa neograničeno raste. Ulogu vjerojatnosti p pojavljivanja događaja zamjenjuje intenzitet lambda pojavljivanja događaja. Kažemo da slučajna varijabla X ima Poissonovu razdiobu s parametrom $\lambda > 0$ i pišemo $X \sim P(\lambda)$ ako ona poprima vrijednost unutar skupa $\{0,1,2...,n\}$ s vjerojatnostima:

$$p_k = P(X=k) = \frac{\lambda^k e^{-\lambda}}{k!}$$

Za očekivanje i disperziju ove razdiobe vrijedi:

$$m_x = \lambda \quad \sigma_x^2 = \lambda$$

Sljedeći isječak koda predstavlja funkciju za računanje pseudoslučajnog broja prema Poissonovoj razdiobi:

```
83.const int PoissonRandomNumber(const double lambda)
84.{
85.  int k=0;
86.  const int max_k = 1000;
87.  double p = ran_uniform();
88.  double P = exp(-lambda);
89.  double sum=P;
90.  if (sum>=p) return 0;
91.  for (k=1; k<max_k; ++k) {
92.    P*=lambda/(double)k;
93.    sum+=P;
94.    if (sum>=p) break;
95.  }
96.  return k;
97.}
```

Zaključak

Tijekom nekoliko proteklih godina obilježen je značajan porast popularnosti sekvencioniranja genoma ponajviše zbog razvoja tehnika za sekvencioniranje druge generacije koje omogućavaju vrlo brzo, precizno te cjenovno prihvatljivo sekvencioniranje. Nove tehnologije za sekvencioniranje predstavljaju ogroman izazov alatima za poravnanje sekvenci tzv. asemblerima čija je osnovna namjena poravnanje malih fragmenata dobivenih sekvencioniranjem u odnosu na referentni genom. U svrhu testiranja asemblera razvijen je čitav niz simulatora čija je zadaća simulirati rezultate sekvencioniranja pri čemu se u svrhe testiranja bilježe točan početak čitanja, broj mutacija, pogrešaka te indel-a pri svakom pojedinom čitanju. Pri simulaciji netom navedenih karakteristika čitanja simulator se služi sljedećim razdiobama: uniformnom (odabir pozicije početka čitanja), normalnom (definiranje duljine fragmenta) te Poissonovom (simulacija pogrešaka pri iščitavanju baza). Sam proces simulacije je vremenski vrlo zahtjevan te troši mnogo memorije. Obrada sekvence reda veličine nekoliko stotina Mb traje oko minute te troši memorije u rasponu od minimalno 200 Mb do maksimalno 1 Gb. Postoji, također, ovisnost vremena trajanja simulacije o vrijednosti ulaznih parametara, stoga će primjerice velika vrijednost BCER-a i frekvencije mutacija značajno usporiti samu simulaciju.

Literatura

[1] Heng Li, Nils Homer, *A survey of sequence alghoritms for next generation sequencing.Oxford University Press* 2010 .

[2] Richard C. Deonier, Simon Avare, Michael S. Waterman, *Computational Genome Analysis: An introduction*. 2005.

[3] http://en.wikipedia.org/wiki/Bacterial_artificial_chromosome

[4] Heng Li, Nils Homer, *A survey of sequence alghoritms for next generation sequencing*.Oxford University Press 2010 .

[5] http://en.wikipedia.org/wiki/Burrows%E2%80%93Wheeler_transform

[6] https://github.com/lh3/wgsim

[7] http://www.niehs.nih.gov/research/resources/software/biostatistics/art/

[8] http://en.wikipedia.org/wiki/Single_nucleotide_polymorphism

[9] Metthew Ruffalo, Thomas LaFramboise, Mehmet Koyuturk, *Comparative analysis of alghoritms for next-generation sequencing read alignment*. Oxford University Press 2011.

[10] Young Lin, Jian Li, Hui Shen, Lei Zhang, Christoper J Papasian, Hong-Wen Deng, *Comparative Studies of de novo Assembly Tools for Next-generation Sequencing and Technologies*. Oxford University Press 2011.

[11] Guenter Kahl, T. Matthias Iarbers, *Tag Based Next Generation Sequencing*. 2012.

[12] http://en.wikipedia.org/wiki/FASTQ_format

[13] http://en.wikipedia.org/wiki/FASTA_format

[14] http://www.ornl.gov/sci/techresources/Human_Genome/faq/seqfacts.shtml

[15] http://www.genomenewsnetwork.org/resources/whats_a_genome/Chp2_1.shtml

[16] http://sourceforge.net/apps/mediawiki/dnaa/index.php?title=Whole_Genome_Simulation

[17] Neven Elezović, *Diskretna Vjerojatnost*. Zagreb, 2007.

Sažetak

Sekvencioniranje DNA jedna je od najvažnijih tehnika za proučavanje nekog gena kojom se određuje točan redoslijed nukleotida tog gena. S pomoću računalnih programa za analizu sekvencija DNA, podaci o redoslijedu mogu se rabiti za: utvrđivanje poznatih restrikcijskih mjesta unutar gena, predviđanje proteina kodiranog tim genom, dobivanje podataka o mogućim intronskim mjestima te uvide u mehanizme kojima se provodi nadzor nad genskim izražavanjem. Kao rezultat sekvencioniranja nastaje vrlo velik broj pročitanih nizova nukleotida. Ukoliko referentni genom postoji ti se nizovi pokušavaju mapirati na njega. Nizovi se od referentnog genoma razlikuju u pojedinim nukleotidima, obrisanim i dodanim dijelovima, stoga je razvijen čitav niz alata kojima se pokušavaju simulirati netom navedene razlike. Neki od postojećih alata su: ART, WGSIM, DWGSIM. Najveći problem postojećih alata je rukovanje samo s relativno kratkim čitanjima.

Ključne riječi: sekvencioniranje, simulacija, genom, BCER, mutacije, ubacivanje, brisanje, alati, sekvence, DNA, nukleotidi

Summary

Genome sequencing is figuring out the order of DNA nucleotides (or bases) in genome that makes an organism's DNA. With the help of various tools for DNA sequence analysis, the obtained data could be used for determining well known restriction points inside genes, prediction of proteins and introns identification. Sequencing reads are then mapped to a reference genome. Reads differ from reference genome in deleted/extended parts and some nucleotides, so the large number of tools for short read simulation such as ART, WGSIM and DWGSIM are developed. Most of these tools can only handle the short reads.

Keywords: sequencing, simulation, genome, BCER, mutation rate, insertion, deletion, tools, sequences, DNA, nucleotide

Skraćenice

BAC	Bacterial Artificial Cromosome	umjetni bakterijski kromosom
BCER	Base calling error rate	greška prilikom određivanja baza
BWT	Burrow-Wheeler transformation	Burrows-Wheelerova transformacija
DNA	Deoxyribonucleic Acid	deoksiribonukleinska kiselina
SNP	Single-nucleotide polymorphism	jednonuklotidni polimorfizam
WGS	Whole-genome shotgun	sekvencioniranje kompletnog genoma

Privitak

Rezultati mjerenja

Mjerenje ovisnosti vremena izvršavanja te ukupnog zauzeća memorije o vrijednostima ulaznih parametara izvedeno je pomoću alata "Valgrind". Simulacija ja za svaku testnu sekvencu pokretana dva puta od čega jednom pod netom navedenim alatom kako bi se utvrdilo prosječno te maksimalno zauzeće radne memorije od strane simulatora tijekom čitave simulacije. Tijekom prvog pokretanja simulacije utvrđivala se ispravnost izlaza te ukupno vremensko trajanje simulacije.

Naredba za pokretanje simulacije pod alatom "Valgrind" s namjerom analize memorijskog zauzeća kroz vrijeme:

valgrind - -tool massif /.ime_programa

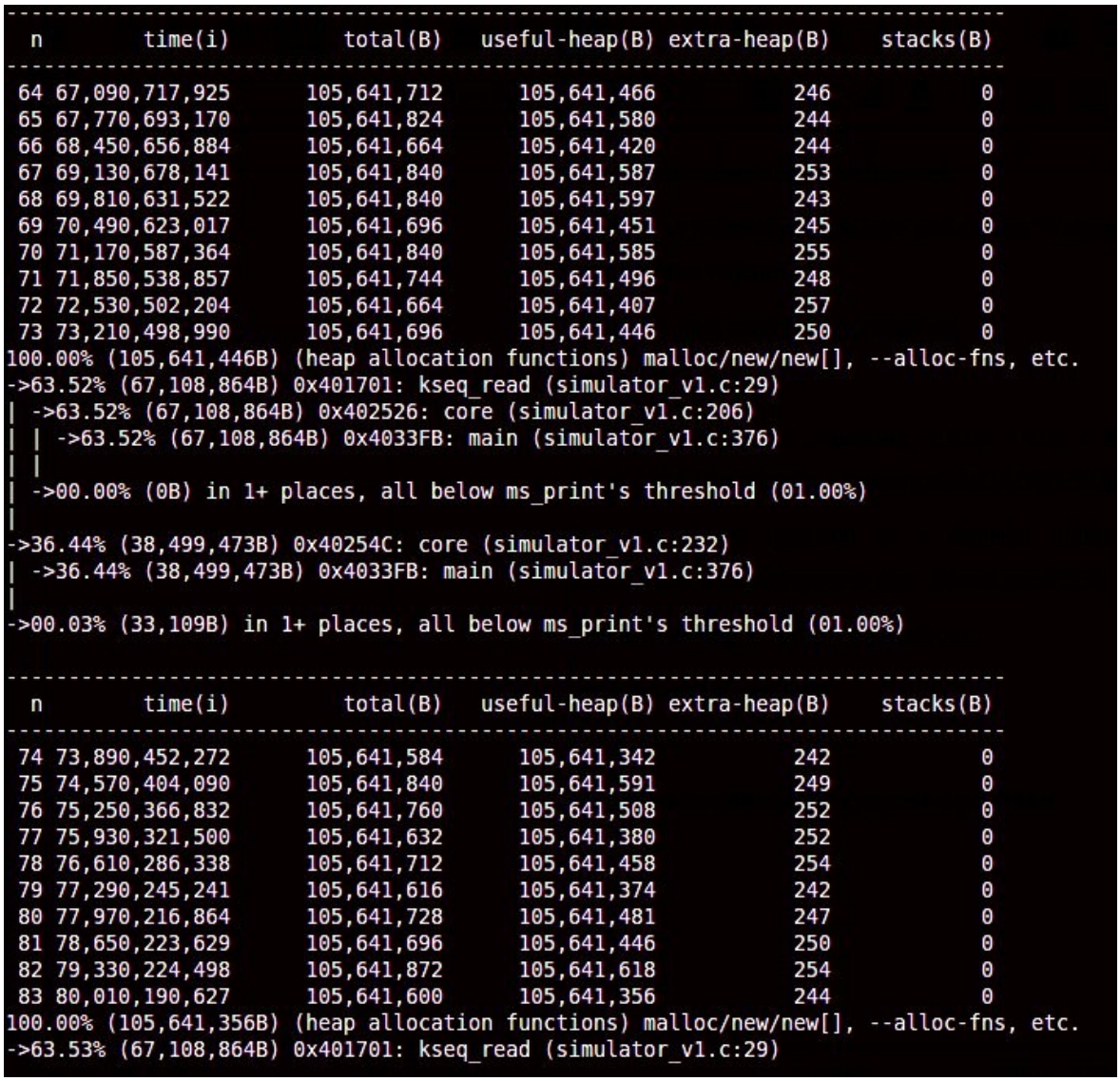
npr.:

valgrind --tool=massif ./simulator -e 0.02 -r 0.01 -d 400 -N 1000000 -1 100 -2 70 -g 3 -X 0.1 -S 1991 Monodelphis_domestica.BROADO5.50.dna.chromosome.3.fa iz1.txt iz2.txt

Rezultati obrade spremaju se u datoteku oblika: massif.out.<pid> gdje je <pid> identifikator dotičnog procesa (npr. Massif.out.4323).

Datoteku je moguće pročitati naredbom:

ms_print massif_out.<pid>



Slika 1.13 screenshoot ispisa alata Valgrind za jednu od testnih sekvenci

Upute za korištenje programske podrške

Detaljne upute za korištenje i instalaciju programske podrške date su u poglavlju **(3.Simulator)**. Program je moguće prevesti i pokretanjem priloženog makefile-a što se čini naredbom: *"make"* nakon pozicioniranja u kazalo u koje su prethodno spremljeni izvorni kod simulatora i zaglavlja: "kseq.h" i "genran.h", ukoliko se koristi makefile vrijeme izvođenja simulacije bit će značajno kraće budući da se u tom slučaju kod prevođenja provodi optimizacija (Tablica 1.4).