SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

PROJEKT

IMPLEMENTACIJA HIRGC ALGORITMA

Duje Jurić

Voditelj: Mirjana Domazet-Lošo

Zagreb, Lipanj, 2025.

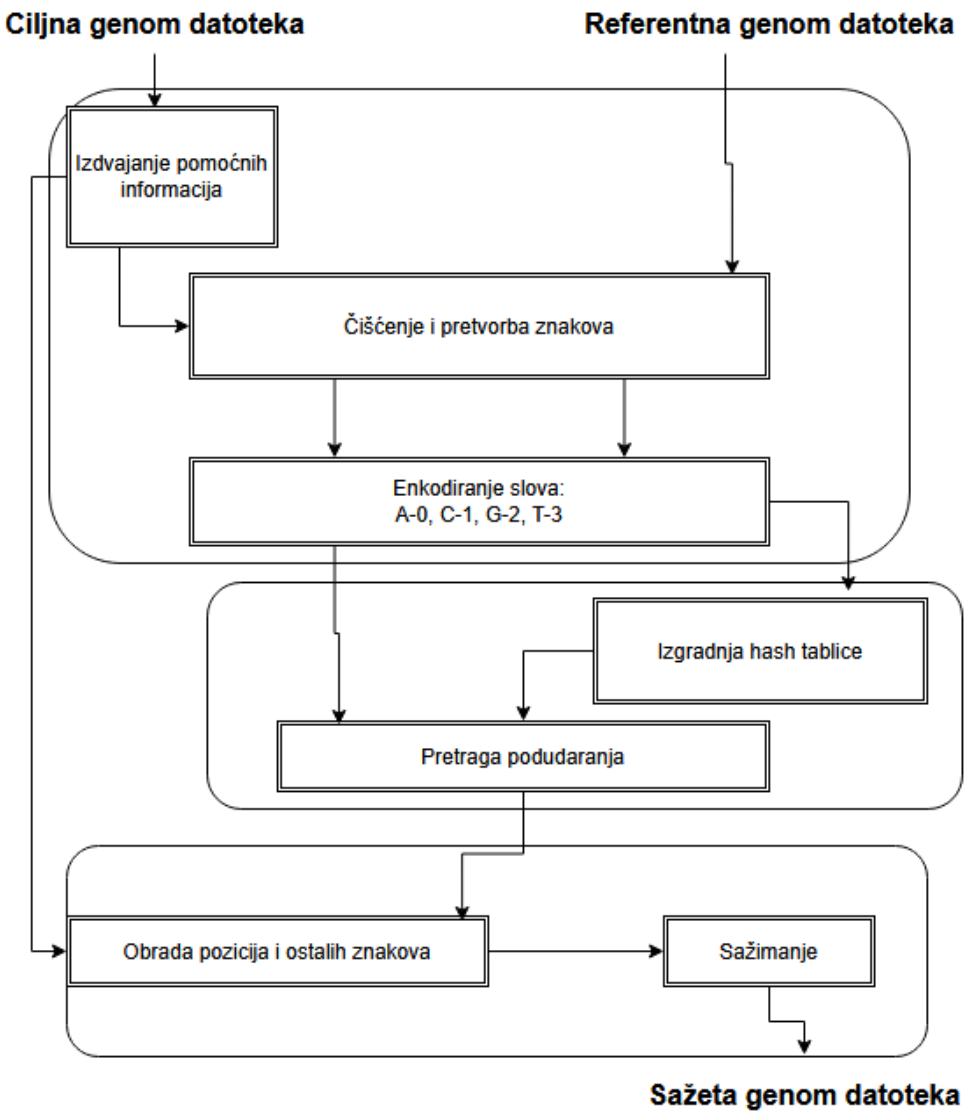
Sadržaj

1. Uvod

Ovaj projektni rad opisuje HiRGC (engl. *High-speed and high-ratio referential genome compression*), algoritam namijenjen za referencijalno sažimanje genomskih podataka. U radu uz objašnjenje principa rada algoritma, također je implementirana i vlastita verzija algoritma. Prikazana je usporedba s izvornim algoritmom kroz analizu performansi u brzini izvođenja, korištenju memorije i omjeru sažimanja[1].

2. Algoritam HiRGC

Algoritam HiRGC je visoko-performantni algoritam za referencijalno sažimanje genoma koji se temelji na 2-bitnom kodiranju i naprednom pretraživanju Greedy Matching algoritima korištenjem hash tablice[1].



Slika 2.1. Prikaz rada algoritma HiRGC[1]

Na slici 2.1. pirkazan je jednostavan rad algoritma HiRGC. Rad algoritma možemo podijeliti u tri glavne faze[1]:

- **Predobrada (engl. *Preprocessing*)**

  Cilj ove faze je pročistiti referentnu i ciljnu FASTA datoteku kako bi sadržavale isključivo znakove iz skupa (A, C, G, T). Algoritam izdvaja pomoćne informacije kao što su identifikatori, pozicije i duljine malih slova, pozicije znaka 'N' te svi ostali nestandardni znakovi. Zatim provodi čišćenje i pretvorbu znakova te sve preostale znakove iz skupa (A, C, G, T) enkodira.

- **Algoritam Greedy Matching**

  Faza u kojoj se provodi Greedy Matching algoritam kojim se pronalaze podudaranja između ciljne i referentne sekvence znakova. Algoritam prvo gradi hash tablicu. Za svaki podniz (k-torku) fiksne duljine k izračunava se njegova hash vrijednost i pozicija se pohranjuje u globalnu hash tablicu. Algoritam zatim iterira kroz ciljnu sekvencu. Za svaku poziciju traži najdulje moguće podudaranje s referentnom sekvencom koristeći izgrađenu hash tablicu. Zatim zapisuje rezultate svih podudaranja i nepodudaranja

- **Naknadna obrada (engl. *Post-processing*)**

  U ovoj fazi, informacije dobivene iz prethodnih faza dodatno se sažimaju kako bi se dobila konačna komprimirana datoteka. Delta encoding se koristi za sažimanje pozicija podudaranja i drugih pozicijskih informacija. Run-length encoding se koristi za sažimanje duljina sekvenci. Sve obrađene informacije (pomoćne informacije, podudaranja, nepodudaranja) spremaju se u tekstualnu datoteku.

U algoritam HiRGC također ubrajamo i dekompresiju koja predstavlja obrnuti proces kojim se originalna FASTA datoteka može precizno rekonstruirati iz sažete datoteke[1].

3. Testiranje i usporedba vlastite i izvorne implementacije algoritma HiRGC

Oba algoritma testirana su na 2 različita E. coli skupa podataka koja se oba sastoje od jedne referentne FASTA datoteke i jedne ciljne FASTA datoteke.

- **1. skup podataka**

  Referentna datoteka(NC_000913.3 Escherichia coli str. K-12 substring MG1655, complete genome) i ciljna datoteka(CP_000243.1 Escherichia coli UTI89, complete genome)

- **2. skup podataka**

  Referentna datoteka(NC_002695.2 Escherichia coli O157:H7 str. Sakai DNA, complete genome) i ciljna datoteka(CP017249.1 Escherichia coli strain NADC 5570/86-24/6565, complete sequence)

U provedbi usporedbe algoritma, korištene su tri kategorije: brzina izvođenja, korištenje memorije i omjer sažimanja. Na tablicama 3.1. i 3.2. prikazani su rezultati testiranja na oba skupa podataka.

Algoritam i proces	**Izvorna implementacija [kompresija]**	**Izvorna implementacija [dekompresija]**	**Vlastita implementacija [kompresija]**	**Vlastita implementacija [dekompresija]**
Brzina izvođenja [ms]	1420,126	332,972	2142,562	228,321
Korištena memorija [kb]	71,392	24,476	137,412	26,474
Omjer sažimanja [%]	40,7	-	47,9	-

Tablica 3.1. Usporedba algoritma na prvom skupu podataka

Algoritam i proces	**Izvorna implementacija [kompresija]**	**Izvorna implementacija [dekompresija]**	**Vlastita implementacija [kompresija]**	**Vlastita implementacija [dekompresija]**
Brzina izvođenja [ms]	3150,273	371,937	3797,322	203,231
Korištena memorija [kb]	81,328	26,504	148,092	25,464
Omjer sažimanja [%]	76,4	-	76,9	-

Tablica 3.2. Usporedba algoritma na drugom skupu podataka

4. Zaključak

U oba skupa podataka izvorna implementacija ostvarila je bolje rezultate u brzini izvođenja procesa kompresije, dok je vlastita implementacija ostvarila bolje rezultate u brzini izvođenja procesa dekompresije. Razlog tome je potencijalno jednostavnija implementacija dekompresije tekstualne datoteke i ne korištenje dodatnog PPMD algoritma sažimanja. Izvorna implementacija koristi manje memorije u oba procesa, osim u slučaju dekompresije drugog skupa podataka. Izvorna implementacija također konzistentno ostvaraje bolje omjere sažimanja.

Literatura

[1] L. W. J. L. Yuansheng Liu, Hui Peng, “High-speed and high-ratio referential genome compression”, https://academic.oup.com/bioinformatics/article/33/21/3364/3885699, 2017., [Online; Accessed: May 2025].