SVEUČILIŠTE U ZAGREBU

FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

HRCM algoritam

Leon Hegedić, Ivan Terzić

Voditelj: Mirjana Domazet-Lošo

Zagreb, svibanj, 2024.

**Sadržaj**

# Uvod

U području bioinformatike čest je rad s dugačkim nizovima znakova. Neka taj dugi niz znakova predstavlja niz baza A, C, G i T u DNA molekuli. Ta sekvenca baza određuje genetsku informaciju koja je cijelom svojom duljinom bitna [1], tj. ne smije se izostaviti niti jedna baza u zapisu baze. Ako se uzme u obzir da se tim nizom može predstaviti i cijeli genom pojedine vrste koji može biti dug od 50 000 pa sve do preko 100 000 000 000 znakova, odnosno baza (za primjer, genom čovjeka sastoji se od ~3 000 000 000 baza). [2] Ako se uzme u obzir da se slova u današnjim računalima pohranjuju s jednim bajtom, odnosno 8 bitova, neki niz baza dug 1 000 000 znakova zauzeo bi 1MB, a 100 000 000 gotovo 100MB, a to nije ni približno dovoljno za pohranu genoma na primjer kralježnjaka. Jasno je da je potreban efikasan način za pohranu tih niza baza zbog samog prijenosa, preuzimanja, ali i pohrane na uređajima. Ovaj se rad bavi upravo takvim algoritmom: HRCM algoritmom. Algoritam radi kompresiju datoteka genoma pomoću referentnog niza baza te pohranjuje, s manjim memorijskim zauzećem, inicijalnu datoteku genoma u drugom obliku sa svim potpunim informacijama. Rad se bazira na proučavanju algoritma, opisu teoretskih pojmova nužnih za razumijevanje algoritma, opisu samog algoritma te dokumentiraju testiranja algoritma.

# Opis algoritma

HRCM algoritam sastoji se od dvije glavne faze: kompresija i dekompresija. U fazi kompresije se sekvence spremaju u datoteke sažetog oblika (manjeg memorijskog zauzeća) koje sadržavaju potpunu informaciju o sekvenci, dakle ne gube se informacije ni o malim slovima, ni specijalnim znakovima niti „X” znakovima. U fazi dekompresije se ti pohranjeni podatci vraćaju u svoj izvorni oblik, kako je primljen na ulazu u algoritam.

## Kompresija

Algoritam na svoj ulaz u fazi kompresije prima referentnu datoteku genoma (referentnu sekvencu) i jednu ili više sekvenci koje se trebaju komprimirati, sve datoteke u FASTA formatu. Sekvence za komprimiranje komprimiraju se pomoću referentnog genoma. Sama faza kompresije algoritma dijeli se u 3 glavna koraka: izvlačenje informacija sekvenci, pronalazak podudarnih informacija u nizovima te kodiranje informacija o sekvencama.

### Izvlačenje informacija o sekvencama

U prvom koraku kompresije, na ulazu samog algoritma, nalaze se datoteke sekvenci namijenjene za kompresiju te referentna sekvenca u FASTA formatu. To znači da je u prvom redu identifikator genoma te u svakom redu ispod niz znakova fiksne duljine za svaki red koji se sastoji od osnovnih baza, malih slova, koje označavaju nesigurnost kod očitanja, slova N i slova X ili procijepa -. Za svaku se sekvencu pohranjuju sljedeće informacije: identifikator, sekvenca osnovnih baza (A, C, G i T), informacije o pozicijama malih slova, informacije o slovima N, informacije o ostalim znakovima te broj koji označava duljinu retka. Za referentnu sekvencu pohranjuju se samo sekvenca osnovnih baza i informacije o malim slovima, ona ne mora pamtiti informacije o ostalim slovima, itd.

Sam proces obavlja se na sljedeći način: obzirom na FASTA format, prva linija u datoteci je tzv. identifikator, on se izvlači i sprema kao identifikator pojedine datoteke. Nakon toga se, za prvu liniju koja sadrži niz baza, gleda njena duljina te pohranjuje kao duljina retka. Nakon toga se u nizu baza redom bilježe pozicije malih slova te potom i N slova i posebnih slova. Obzirom da je očekivano da će algoritam raditi s izuzetno dugim nizovima znakova, pohrana informacija o malim i N slovima se radi pomoću pohrane relativnog indeksa (udaljenosti) od prošle pojave malog ili N slova (naravno, prošla pojava malog slova se gleda od prošle pojave malog slova, itd.) i pohrane duljine tog niza znakova. Tako se izbjegava pohrana velikih brojeva kao pozicija, što bi bilo očekivano za slučaj kad bi se koristili apsolutni indeksi u nizu znakova. Nakon pohrane tih informacija, znakovi se spremaju u jedan niz znakova, brišu se posebni znakovi i N znakovi te se dobije čist niz znakova A, C, G ili T, odnosno baza. Za referentu sekvencu postupak je analogan, osim što se spremaju samo podatci potrebni za taj niz, dakle niz baza i podaci o malim slovima. Iako se informacije ne spremaju, također se brišu N znakovi i specijalni znakovi. Ovako zapisani nizovi znakova ulaze u drugu fazu kompresije: pronalazak podudarnih informacija.

### Pronalazak podudarnih informacija

U drugoj fazi kompresije, pronalasku podudarnih informacija, kao ulaz primamo nizove znakova zapisane kako je opisano u prošlom odlomku, točno 1 referentan niz znakova i barem jednu sekvencu koja je namijenjena za kompresiju. U ovoj fazi cilj je pronaći podudarnosti između referentne sekvence i sekvenci koje se komprimiraju kako bi se što više smanjilo memorijsko zauzeće zapisa. Ova se faza sastoji od pronalaska podudaranja u nizu baza te u malim slovima. Ukoliko postoji više od jedne sekvence koje se trebaju komprimirati, algoritam koristi i tzv. podudaranje drugog reda (engl. *second-level matching*).

Prvo se u ovoj fazi radi pronalazak podudaranja u sekvencama pojedine sekvence za kompresiju i referentne sekvence. Za to, potrebno je napraviti hash tablicu pojedinih k-mera u referentnoj sekvenci. Ovaj postupak funkcionira kao posmični prozor koji ide nad nizom te onda izračunava sažetak, odnosno hash vrijednost za pojedini k-mer. A, C, G i T se kodiraju kao 0, 1, 2 i 3, time se svako slovo može kodirati s 2 bita jer su 00, 01, 10, 11 vrijednosti koje predstavljaju te brojeve u oblicima bitova. Izračunava se hash vrijednost za prvi k-mer u prozoru na način da se gleda vrijednost k-mera tako da su vrijednosti znaka koji je prije na značajnijem mjestu, to se ostvaruje posmakom za dva bita ulijevo kad se dodaju znakovi. Kreiranje hash tablica odvija se pomoću dva polja: H i L, koje se prije početka kreacije tablice moraju inicijalizirati. Nakon izračuna sažetka k-mera, recimo da je njegova vrijednost v, njegov indeks se pohranjuje u polje H pod indeksom v, a originalna vrijednost koja je bila u H se sprema u L na indeksu i. Time efektivno polje H drži zadnji indeks pojave k-mera s određenim sažetkom, a L „pokazivač“ na prethodni indeks čime to polje efektivno postaje lista s pokazivačima na prethodni indeks k-mera s istom vrijednosti sažetka. Time se na efektivan način kreira hash struktura koja će znatno ubrzati proces pronalaženja najduljih podnizova koji se podudaraju među dvije sekvence znakova. U procesu pronalaženja najduljih podnizova koji se podudaraju, za svaki se k-mer neke sekvence za kompresiju računa sažetak na isti način, s posmakom, kao i kod izrade hash tablice. Nakon izračuna sažetka se gleda u tablicu H, ako je zapis ekvivalentan onom kod inicijalizacije, u ovom slučaju -1, očito je po procesu izrade hash tablice da onda ne postoji u referentnoj sekvenci k-mer, odnosno podniz, koji se podudara s danim podnizom. Stoga se taj dio sekvence, taj znak, sprema u informacije o znakovima koji se ne podudaraju. U suprotnom, ako postoji indeks u tablici H, postoji bar jedan k-mer koji se podudara između dva niza i onda se ulazi u proces pronalaska najduljeg podudaranja. Tijekom ovog se procesa prolazi kroz sve relevantne k-mere pomoću polja L te se gleda za koju pojavu k-mera se može naći najdulji zajednički niz između referentne sekvence i one koja se komprimira. Informacije se spremaju kao trojka: pozicija (opet relativna u odnosu na prošlu poziciju), duljina tog najvećeg pronađenog podudaranja i niz znakova koji se ne podudaraju.

Kada algoritam za kompresiju primi više datoteka, automatski prelazi u način *batch* kompresije. Ovdje se rezultati prve razine podudaranja koriste za daljnje podudaranje u drugoj razini, kako bi se dodatno smanjili zahtjevi na memorijsko zauzeće korištenjem više sekvenci za pronalazak podudaranja. Podudaranje drugog reda, kao i ono prvog reda, također koristi hash podudaranje, ali umjesto načina koji je implementiran u prvom redu (izvlačenje sažetka iz kodiranih baza) koristi složenije entitete koji uključuju poziciju, duljinu i nepodudarnosti, odnosno izlaz iz podudaranja prve razine, kompresija se izvodi po nizu struktura podataka za podudarnosti iz podudaranja prve razine. Prije tog mora se provesti pretraga najduljih zajedničkih nizova u svim predanim sekvencama za kompresiju. Kako bi se limitirala potrošnja memorije, može se postaviti postotak druge razine podudaranja, p, što znači da se p% sekvenci koje treba komprimirati koristi kao referentne sekvence druge razine. Prvo se stvara hash tablica podataka o podudaranju, outputa iz *first-level matchinga*. U kreaciju sažetka uključuju se sva tri elementa pojedinog objekta toga izlaza, dakle pozicija, duljina i svaku nepodudarnu bazu. Veliki prosti broj se koristi u množenju tih podataka za sažetak. U slučaju da postoji konflikt, kao i u podudaranju prvog reda, koristi se odvojeno ulančavanje s H i L listama. Kod faze pretrage, nakon kreacije hash tablice, se na identičan način kao kod podudaranja prvog reda, naravno koristeći ispravnu hash funkciju, računaju sažeci i traži se identičan zapis u hash tablici. Međutim, identična hash vrijednost ovdje ne znači i identičan entitet jer preslikavanje između entiteta i hash vrijednosti u ovom slučaju nije jednoznačno za razliku od prve razine. Kada se pronađe identična hash vrijednost, potrebno je provjeriti jesu li entiteti identični na razini samog entiteta, ne samo hash vrijednosti. Ako jesu isti, pretraga se nastavlja kao kod prve razine uzastopno dok se ne pronađe nepodudarni entitet, a zatim se bilježe ID sekvence, pozicija i duljina podudarnog segmenta. Nakon pretrage svih referentnih sekvenci, najduži podudarni podniz uzima se za zamjenu i sprema se, ponovo, trojka ID sekvence, pozicija, duljina. Ako se identičan entitet ne pronađe nakon pretrage svih referentnih sekvenci, segment se sprema kao nepodudaran segment.

Neovisno o podudaranjima prvog i drugog reda odvija se treća glavna faza drugog dijela kompresije: podudaranje informacija o malim slovima. Na ovaj se način jednostavno smanjuje prostor za pohranu podataka na način da, ako za sekvencu za određen zapis o malim slovima taj zapis već postoji u zapisu o malim slovima referentne sekvence, pohranjuje se lokacija tog zapisa, inače ako ne postoji ostaje inicijalan zapis.

### Kodiranje

S prethodno opisanim podatcima kao izlazima iz faza podudaranja, bilo nizova, bilo malih slova, ulazi se u fazu kodiranja, same kompresije. Duljine linija i identifikatori se kodiraju s tzv. *run-length encodingom*, specijalni znakovi s entropijskim kodiranje, a pozicije se kodiraju inkrementalnim kodiranje. Sve se tr informacije kodiraju i komprimiraju u datoteku arhive.

## Dekompresija

Dekompresija je jednostavniji postupak od kompresije te je u ovom slučaju samo reverzan postupak. Arhiva se prvo dekomprimira, te se s algoritmima opisanim u kodiranju dekodiraju pojedine informacije. Informacije o sekvencama i malim slovima se na reverzan način također rekonstruiraju u originalne vrijednosti koristeći i podudarne i nepodudarne informacije te se onda na taj način vraćaju u originalne datoteke genoma. Ovdje se kod obrnute faze podudaranja ne moraju koristiti hash vrijednosti jer su one u fazi kompresije služile samo za pronalazak bržeg podudaranja, te tablice ovdje nisu bitne.

# Primjer izvođenja

# Testiranje

# Zaključak

# Literatura

[1] <https://www.fer.unizg.hr/_download/repository/Bioinformatika_1%20-%20Uvodno_2023_2024.pdf>

[2] https://www.fer.unizg.hr/\_download/repository/Bioinformatika\_%201\_2-3\_predavanje\_Dinami%C4%8Dko\_programiranje.pdf

[3] Haichang Yao, Yimu Ji, Kui Li, Shangdong Liu, Jing He, Ruchuan Wang, "HRCM: An Efficient Hybrid Referential Compression Method for Genomic Big Data", BioMed Research International, vol. 2019, Article ID 3108950, 13 pages, 2019. https://doi.org/10.1155/2019/3108950