SVEUČILIŠTE U ZAGREBU

**FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA**

BIOINFORMATIKA – PROJEKTNA DOKUMENTACIJA

**PRONALAŽENJE VARIJANTI GENA IZ PODATAKA DOBIVENIH SEKVENCIRANJEM**

Alen Štruklec

Zvonimir Kučiš

Zlatko Verk

Zagreb, siječanj 2020.

Sadržaj

[1. Opis problema 1](#_Toc29748508)

[2. Opis korištenog algoritma 2](#_Toc29748509)

[3. Testiranje i analiza rješenja 3](#_Toc29748510)

[Zaključak 4](#_Toc29748511)

[Literatura 5](#_Toc29748512)

# Opis problema

Kao ulazni podatak dan je skup očitanja dobiven sekvenciranjem koji sadrži nekoliko varijanti istog gena. Koristeći algoritme globalnog, poluglobalnog i lokalnog poravnanja potrebno je implementirati vremenski i prostorno optimalno rješenje koje na temelju udaljenosti izračunate jednim od gore navedenih algoritama vrši grupiranje očitanja kako bi se otkrile varijante gena prisutne u uzorku.

# Opis korištenog algoritma

Mjera sličnosti dvije sekvence može se odrediti na dva načina, Hammingovom udaljenosti ili udaljenošću uređivanja. Hammingova udaljenost predstavlja broj zamjena potrebnih za transformaciju jednog niza u drugi, a udaljenost uređivanja definirana je kao minimalan broj zamjena, umetanja i brisanja potrebnih za transformaciju jednog niza u drugi.

Na primjer:

* Zamjena: Trava -> Krava
* Umetanje: sir -> sVir
* Brisanje: Brod -> rod

Udaljenost uređivanja može se računati na različite načine i u svakom pokušaju moguće je dobiti različito rješenje, zato su razvijeni razni algoritmi za računanje udaljenosti uređivanja od kojih ćemo neke iskoristiti u ovom projektu.

Algoritmi globalnog, poluglobalnog i lokalnog poravnanja temelje se na konceptu dinamičkog programiranja.

Dinamičko programiranje može se koristiti za rješavanje problema koji imaju optimalnu podstrukturu i preklopljenost potproblema. Koristi se tzv. *bottom-up* pristup kojim se konačno rješenje gradi od rješenja istovrsnih manje složenih problema, a kako se već izračunati podatci ne bi ponovno izračunavali, rješenja se spremaju u matricu.

A close up of a keyboard

Description automatically generated

Primjer matrice prikazan je na slici iznad, a ona se gradi na sljedeći način:

Micanje u desno i dolje predstavlja pomicanje u jednom nizu, a ostanak na istom elementu u drugom nizu. Micanje po dijagonali predstavlja slaganje ili neslaganje između dva niza.

Parametri algoritma težine su kojima se kažnjava zamjena/neslaganje, preskakanje i slaganje znakova. U ovom primjeru one su jednake 1.

Algoritam počinje definiranjem vrijednosti na rubovima (retci i stupci s indeksom 0). Element na poziciji (0, 0) poprimi vrijednost 0, elementi nultog retka poprime vrijednost d\*j, a elementi nultog stupca d\*i gdje d predstavlja parametar brisanja/umetanja (pomicanje desno/dolje).

Ostali elementi računaju se kao minimum pomaka desno iz lijeve ćelije (V(i, j-1) + d), pomaka dolje iz gornje ćelije (V(i-1, j) + d) i dijagonalnog pomaka koji nagrađuje ako su elementi u oba niza ćelije gore-lijevo jednaki, a kažnjava ako su različiti.

Udaljenost poravnanja je rezultat izračunat za element u zadnjem retku i stupcu (m, n).

**Algoritam globalnog poravnanja (Needleman-Wunsch)**

*Needleman-Wunsch*-ov algoritam koristi se za traženje globalnog poravnanja. Globalno poravnanje traži put od početka do kraja oba slijeda.

Algoritam izgleda gotovo identično algoritmu koji je objašnjen u prethodnom odjeljku, samo što umjesto minimuma pomaka traži maksimum.

Implementacija algoritma u programskom jeziku python prikazana je u nastavku.

A screenshot of a social media post

Description automatically generated

**Polu-globalno poravnanje**

U slučaju da želimo pronaći preklapanje niza s podnizom nekog drugog niza, koristimo polu-globalno poravnanje. Ono ne penalizira praznine na početku ili kraju pojedinog niza. Praznine na početku niza omogućavamo tako da ih ne penaliziramo prilikom inicijalizacije (nulti redak/stupac inicijaliziran s 0), a praznine na kraju niza omogućavamo tako da na kraju poravnanja uzimamo maksimalnu vrijednost u zadnjem retku ili stupcu, ovisno o tome nad kojim nizom želimo omogućiti praznine na kraju.

**Lokalno poravnanje**

Lokalno poravnanje traži regije u čije će poravnanje imati najveći rezultat. Praznine na početku slijedova se ne penaliziraju (nulti redak i stupac inicijalizirani na 0) i u slučaju da vrijednost pojedinog elementa padne ispod 0, ta vrijednost se zamjenjuje s 0.

Rezultat poravnanja je najveći element u matrici i predstavlja kraj regije s maksimalnim rezultatom poravnanja sljedova.

U nastavku je prikazana implementacija algoritma lokalnog poravnanja u programskom jeziku python.

A screenshot of a social media post

Description automatically generated

# Testiranje i analiza rješenja

# Zaključak

Na kraju rada svakako morate napisati kratak zaključak duljine do najviše jedne stranice. Na ovoj stranici, nakon završetka teksta zaključka autor rada se mora vlastoručno potpisati.

# Literatura

1. Stallings, W. *Data and Computer Networks*. London: John Wiley, 2006.
2. Stallings, W. *Local Computer Networks*. London: John Wiley, 2006a.