Projekt iz programske potpore

Bioinformatika - Poravnavnaje i mapiranje sekvenci genoma

Voditelji:

- Prof. dr. sc. Mile Šikić
- Mag. ing. Rober Vaser

Sadržaj

1 Uvod

U sklopu predmeta 'Projekt iz programske potpore' na Sveučilištu u Zagrebu, Fakultet elektrotehnike i računarstva, studenti u međusobnoj suradnji pod nadzorom profesora i asistenata rješavaju praktične probleme s ciljem upoznavanja alata i tehnika korištenih u struci. Grupa okupljena pod vodstvom prof. dr. sc. Mile Šikića bavi se poravnavanjem i mapiranja genoma koristeći moderni C++, sustav za upravljene izvornim kodom (eng. version control) git, automatizirani sustav izgradnje (eng. build system) CMake, sustav kontinuirane integracije TravisCI. Po završetku projekta, studenti bi trebali stječi vještine korištenja spomenutih tehnologija te ujedno biti u stanju implementirati osnovne algoritme iz područja bioinformatike.

2 Rad na projektu

Prije početka rada na projektu, studenti su dužni proći navedene lekcije ako nisu upoznati sa zadanim tehnologijama:

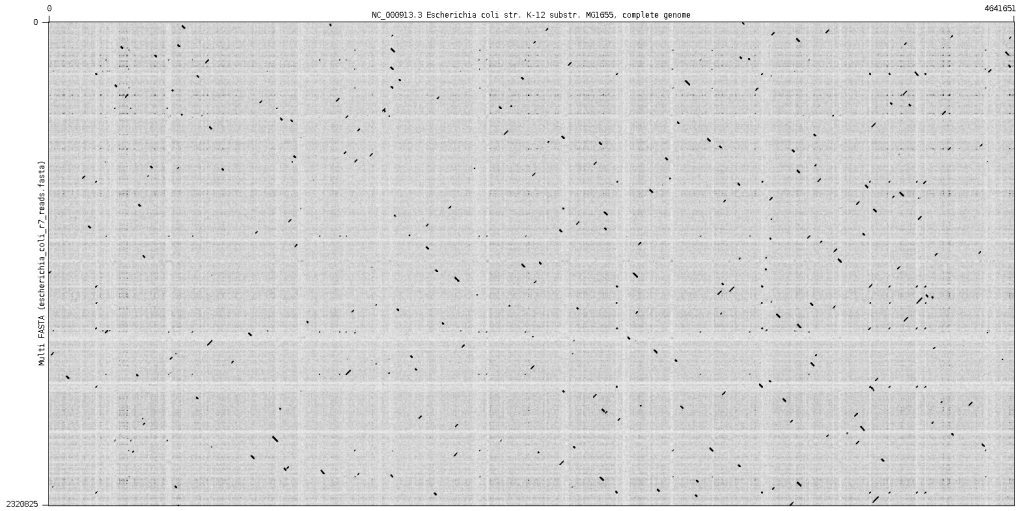
C++ , GitHub , CMake , GoogleTest , TravisCI

Također, poželjno je pridržavati se Googleovih smjernica za pisanje i oblikovanje C++ koda. Po početku rada studenti su podjelljeni u timove: blue, orange, pink i brown. [1] Svaki tim ima svoju git granu i dozvoljeno je kreiranje novih s imanovanjem formata: 'white_feature_one'.

2.0.1 Cilj

Studenti će implementirati biblioteku koja podržava nekoliko algoritama za mapiranje velikog broja relativno malih podnizova znakova na veliki niz znakova koji predstavlja referentni genom. Cilj je povezati implementirane biblioteke u jedan program, obično nazvan eng. mapper, s ciljem poravnanja skeniranih sekvenci.

[1] White?



Slika 1: Vizualni prikaz

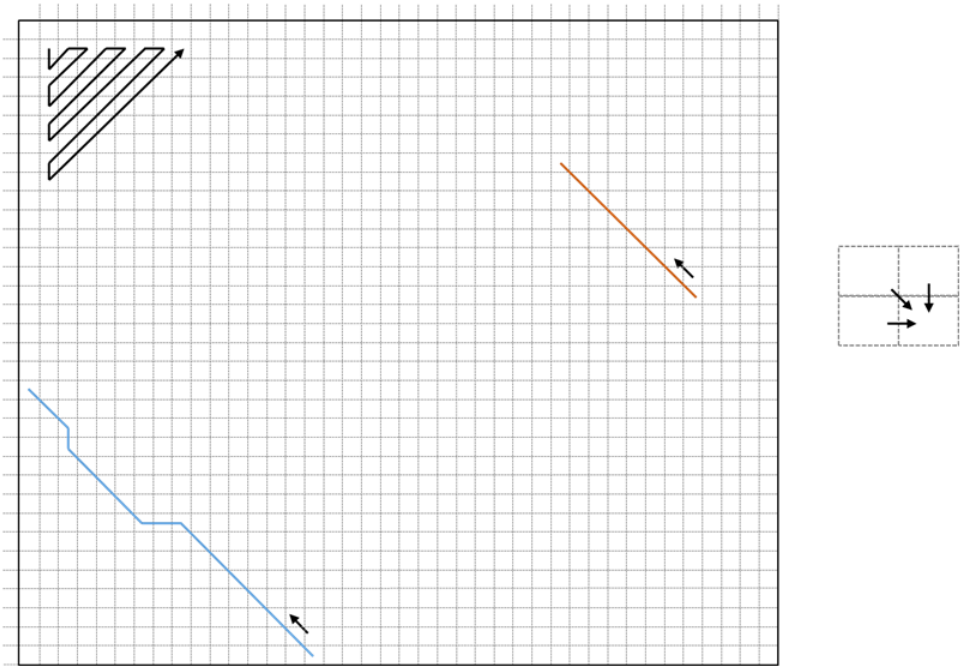
Zadaci

2.1.1 Alati, okruženje, učitavanje podataka

Svaki tim dužan je držati README.md svoje grane usklađenim s README.md glavne (eng. master) grane. Kao uvod u projekt, svaki tim treba postaviti strukturu projekta i inicijalizirati CMake postavke za izgradnju glavnog programa s imenom formata `<ime tima>_mapper` (npr. `white_mapper`) Program preko naredbene konzole mora prihvatiti dva argumenta: put do skeniranih sekvenci i put do referentnog genoma. Skenirane sekvence trebaju biti u FASTA ili FASTQ formatu, a referentni genom je isključivo podržan u FASTA formatu. Povodom učitavanja podataka i njihovog spremanja u memoriju, potrebno je ispisati određene statistike na `stderr` (duljina najkraće i najdulje sekvence, prosječna duljina sekvence...) Studenti nisu dužni samostalno implementirati parser za navedene tipove podataka već se mogu koristiti gotovim rješenjem bioparser dodajući ga u svoj projekt putem git submodula i CMake, ali moraju provjeriti jesu li ekstenzije predanih datoteka podržane (`.fasta` `.fa` `.fastq` `.fq` `.fasta.gz` `.fa.gz` `.fastq.gz` `.fq.gz`)

2.1.2 Poravnanje sekvenci

Cilj ovog podzadatka je implementirati razne algoritme bazirane na dinamičkom programiranju za poravnavanje nizova genoma kako bi se ustvrdila njihova sličnost. Ona se temelji na broju transformacija potrebnih za pretvaranje jednog niza u drugi. Generalni koncept udaljenosti uređivanja (eng. edit distance) detaljnije je objašnjen na Wikipediji i u knjizi FER - Bioinformatika



Zadani algoritmi:

- Needleman-Wunsch za globalno poravnanje
- Smith-Waterman za lokalno poravnanje
- Sufiks/prefiks derivati Needleman-Wunsch algoritma za polu-globalno poravnanje. Detaljnije u FER - Bioinformatika

Sve implementacije trebaju biti sadržane unutar namespacea jedne biblioteke, s nazivom formata `<ime tima>_alignment` (npr. `white_alignment`). Bibiloteka treba sadržavati dvije funkcije s navedenim prototipima:

```
int pairwise_alignment(
        const char* query, unsigned int query_length,
        const char* target, unsigned int target_length,
        AlignmentType type,
        int match,
        int mismatch,
        int gap);
```

```
int pairwise_alignment(
        const char* query, unsigned int query_length,
        const char* target, unsigned int target_length,
        AlignmentType type,
        int match,
        int mismatch,
        int gap);
```

Drugi prototip funkcije prima dodatna dva argumenta preko kojih vraća početak poravnanja unutar `target` sekvence i odgovarajući CIGAR string.

2.1.3 DNA Minimizeri

Slijedeći korak je bio implementirati "minimizere", mali substringovi koje često zovemo k-meri. K-meri se koriste za brz pronalazak regija koje treba poravnati. Pošto sekvenciranje samo po sebi zauzima velike količine priručne memorije, ovom metodom se to može znantno smanjiti. To radimo na način da slijed nukleinskim baza izrežemo na manje dijeove duljine k, i među svakih 2 (window) baza spremimo najmanji. To je pobliže opisano u ovom radu: Funkcija koja to treba obavljati je zadana:

```
std::vector<
    std::tuple<
        unsigned int, unsigned int, bool>> minimizers(
const char* sequence, unsigned int sequence_length,
unsigned int k,
unsigned int window_length);
```

Osim toga, moramo i ispisati broj različitih minimizera, postotak unikatnih minimizera i broj pojavljivanja najčešćeg minimizera kada top f nije uzeto

u obzir. Moramo dodati opcionalne argumente `k`, `w` i `f`. Predodređne vrijednosti za `k, w, f` bi trebale biti `15, 5, 0.001`.

2.1.4 Mapiranje sekvenci

Posljednji zadatak bio je poravnati ulazne fragmente s danim referentnim genomom. Nakon kreiranja indeksa minimizera iz referentnog genoma i uklanjanja f najčešćih minimizera, za svaki fragment zasebno pronalaze se poklapanja minimizera fragmenta i referentnog genoma. Iz liste poklapanja, najduži lanac najbolji je kandidat za dobro poravnanje. Tu regiju (lanac) najlakše je pronaći rješavanjem problema najdužeg rastućeg podniza (eng. longest increasing subsequence problem) nad listom minimizera koji se poklapaju. Potom se nad pronađenim regijama može pozvati procedura poravnavanja.
Maper ispisuje pronađene regije u PAF formatu. Ako je argument `c` pozvan u naredbenom retku, potrebno je u ispis dodati i CIGAR string. Dodatno, argumentom `t` unosi se broj dretvi koje maper koristi. Za ovaj dio zadatka studenti mogu koristiti OpenMP ili thread_pool.

Rezultati

Konačni cilj je bio poravnati 2 zadana fragmenta na referencu E. coli.

Rezultati

3 Literatura

– FER - Bioinformatika