Übungen zum Sequenzanalyse-Praktikum

Universität Bielefeld, WS 2023/24
M. Sc. Leonard Bohnenkämper · Dr. Roland Wittler
https://gi.cebitec.uni-bielefeld.de/teaching/2023winter/sequaprak
praktikum-seqan@CeBiTec.Uni-Bielefeld.DE

Übungsblatt 3 vom 31.10.2023
Abgabe bis Sonntag, 24:00 Uhr.

Aufgabe 1 (Suffixbäume)

Kopiere dir die im Ornder `/prj/seqan/Praktikum/Indexstrukturen/` bereitgestellte Java-Klasse zur Erstellung und Verwendung von Suffixbäumen. Diese Klasse stammt von Alessandro Bahgat Shehata und soll in den folgenden Aufgabenteilen verwendet und untersucht werden.
Sie implementiert eigentlich einen *generalized suffix tree* zur Indizierung mehrerer Sequenzen – wir verwenden hier aber nur den `SimpleSuffixTree` in der Klasse `SuffixTree`. In der ebenfalls bereitgestellten Main-Klasse sind Hilfsfunktionen vorgegeben.

1. Lade Chromosom 22 des Referenzgenoms "hg38" von der UCSC-Homepage herunter und extrahiere die ersten 1000 Zeilen, die keine Ns enthalten, z.B. mithilfe der Unix-Befehle `grep` und `head`.
2. Vervollständige die zwei TODOs im Teil "Suffix tree experiment" der Main-Methode, um das Laufzeit- und Speicherverhalten der Suffixbaum-Konstruktion zu analysieren.
3. Stelle Laufzeit und Speicherbedarf in jeweils einem Plot dar.
4. Auf welchem Algorithmus basiert die Implementation des Suffixbaums und welches Laufzeit- und Speicherverhalten würde man dementsprechend erwarten? Werden die Erwartungen erfüllt?

Aufgabe 2 (Bloom-Filter)

Die in Aufgabe 1 verwendete Main-Klasse enthält eine einfache Implementierung eines Bloom-Filters mit zwei Hashfunktionen. Diese soll nun vervollständigt und im Teil "Bloom filter experiment" der Main-Methode verwendet werden.

1. Implementiere die vorgegebene Funktion `add`, die für einen gegebenen String die Hashfunktionen auswertet und die entsprechenden Positionen im Bitarray setzt.
2. Implementiere die vorgegebene Funktion `mayContain`, die für einen gegebenen String die Hashfunktionen auswertet und die entsprechenden Positionen im Bitarray prüft.
3. In der Main-Methode wird ein Bloom-Filter mit zwei Hashfunktionen und einem Array der Länge 4096 verwendet um alle 12-mere aus den ersten zehn Zeilen der Sequenzdatei zu speichern. Wie groß ist theoretisch die Falsch-Positiv-Rate, also die Wahrscheinlichkeit, dass die Funktion `mayContain` für ein k-mer die Antwort `true` zurückgibt, obwohl dieses nie dem Bloom-Filter hinzugefügt wurde?
4. Prüfe für die sechs in der Main-Methode deklarierten k-mere, ob es sich jeweils um ein *True-Positive, False-Positive* oder *True-Negative* handelt.
5. Warum kann es keine *False-Negatives* geben?