Übungen zum Sequenzanalyse-Praktikum

Universität Bielefeld, SoSe 2015
Dr. Roland Wittler · M.Sc. Linda Sundermann
http://wiki.techfak.uni-bielefeld.de/gi/Teaching/2015summer/SequaPrak
praktikum-seqan@CeBiTec.Uni-Bielefeld.DE

Übungsblatt 8 vom 08.06.2015
Abgabe bis Donnerstag, 24:00 Uhr.

Aufgabe 1 (Suffixbäume)

Lade die unter `http://wiki.techfak.uni-bielefeld.de/gi/Teaching/2015summer/SequaPrak` bereitgestellte Java-Klasse zur Erstellung und Verwendung von Suffixbäumen von der Veranstaltungsseite herunter. 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 speichern hier aber einfach immer nur eine Sequenz unter dem Index "0". In der ebenfalls bereitgestellten Main-Klasse sind Hilfsfunktionen vorgegeben.

1. Lade Chromosom 20 des Referenzgenoms "hg18" von der UCSC-Homepage herunter und extrahiere die Zeilen 1000–2000, z.B. mithilfe der Unix-Befehle `head` und `tail`.
2. Vervollständige die drei TODOs im Teil "Suffix tree experiment" der Main-Methode, um das Laufzeit- und Speicherverhalten der Suffixbaum-Konstruktion zu analysieren.
3. Stelle Laufzeit und Speicherbedarf jeweils in 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 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?