Übungen zur Vorlesung Sequenzanalyse I

Universität Bielefeld, WiSe 2009/2010
Prof. Dr. Jens Stoye · Dipl.-Inform. Nils Hoffmann

http://wiki.techfak.uni-bielefeld.de/gi/GILectures/2009winter/SequenzAnalyse

Blatt 5 vom 12.11.2009
Abgabe in einer Woche vor Beginn der Vorlesung.

Aufgabe 1 q-gram Distanz (2 Punkte)
Gegeben sind die Worte $x =$ `abcabdabbacadaba` und $y =$ `cadbcabbacadabada`.

1. Gib die entsprechenden q-gram Profile für x und y an.
2. Bestimme für $q = 4$ und $\Sigma =$ {`a,b,c,d`} die q-gram Distanz zwischen den Worten.

Aufgabe 2 Worte mit dem gleichen q-gram Profil (3 Punkte)
Gegeben sind $q = 3$, $\Sigma =$ {`a,b,c`} und $x =$ `bcabbabacabbac`.

1. Gib das q-gram Profil von x an.
2. Zeichne den De-Bruijn Graphen zu x und $q = 3$. Die Knoten des Graphen entsprechen den $q-1$-grams von x, die Kanten repräsentieren die q-grams.
3. Finde mit Hilfe des De-Bruijn Graphen ein Wort $y \neq x$ mit $d_q(x, y) = 0$.

Aufgabe 3 Rank und Unrank (2 Punkte)
Gegeben sind das Alphabet $\Sigma = \{0, 1, \ldots, 7\}$ und die Wortlänge $q = 4$.

1. Berechne den Rang des Wortes 3267 und danach (ohne vollständige Neuberechnung, sondern durch ein Update in konstanter Zeit) den Rang des Wortes 2671. Verwende die aufsteigende Variante der Codierung von 0 nach $q - 1$. Gib alle Zwischenschritte an.
2. Welches Wort hat den Rang 3267 unter Verwendung der aufsteigenden Codierung?

Aufgabe 4 q-gram Distanz, **Abgabe am 26.11.2009** (6 Punkte)

1. Implementiere eine JAVA-Klasse `QGramDistance`, die das Interface `StringDistance`[1] implementiert und mit der Methode `getDistance` die q-gram Distanz zwischen zwei Strings zurückgibt. Deine Klasse sollte zusätzlich zwei Methoden bieten, eine Methode `public void setQ(int q)`, um q zu setzen, und eine Methode `public void setSigma(int sigma)`, um die Alphabetgröße zu setzen.
2. Je nachdem, ob ein q-gram Profil dünn oder dicht besetzt ist, macht es Sinn, unterschiedliche Datenstrukturen zu verwenden, um dieses zu speichern. Verwende ein Array, um ein dichtes q-gram Profil zu speichern und verwende eine **ranking** Funktion (s.h. Skript S. 23 ff.), um den Index eines q-grams effizient zu berechnen (s.h. dazu auch Aufgabe 3). Implementiere eine JAVA-Klasse `DenseQGramProfile`, die ein q-gram Profil mit Hilfe eines Arrays erzeugt und speichert.
3. Bei dünn besetzten q-gram Profilen wird zuviel Speicherplatz verschwendet, wenn man ein dichtes Array benutzt. Implementiere daher eine JAVA-Klasse `SparseQGramProfile` und verwende intern eine `HashMap<Integer,Integer>`, die für den Rang jedes q-grams als Schlüssel die Häufigkeit des Auftretens speichert.
4. Beide Klassen sollten das Interface `QGramProfile` mit den Methoden `public void int getOccurrenceFor(int rank)` und `public int size()` implementieren. Die erste Methode liefert für den Rang eines q-grams dessen Häufigkeit zurück, während die zweite Methode die Anzahl aller möglichen q-grams bezogen auf die Alphabetgröße und q zurückgibt.
5. Benutze die Strings aus Aufgabe 1, um deine Methoden (dicht vs. dünn) zu evaluieren und gib die Ergebnisse an.

[1] s.h. dazu Übungsblatt 2 und 3

6. Auf der Homepage der Veranstaltung liegt das Genom des Organismus Xanthomonas campestris pv. campestris als FASTA-Datei `xanthomonas.fasta`. Verwende den `FASTAReader`, um die Datei einzulesen und erstelle das q-gram Profil. Bis zu welchem q kann man das dichte q-gram Profil noch komplett im Speicher halten? Wie weit kommt man mit dem dünn besetzten q-gram Profil?