**Arbeitsgruppe für Genominformatik** **Universität Hamburg**
**Stefan Kurtz** **Zentrum für Bioinformatik**

# Genominformatik
# Sommersemester 2017
# Übungen zur Vorlesung: Ausgabe am 25.04.2017

Punkteverteilung: Aufgabe 2.1: 7 Punkte Aufgabe 2.2: 3 Punkte,

Abgabe bis zum 04.05.2017, 23:59 Uhr.

**Aufgabe 2.1** Implementieren Sie den in der Vorlesung vorgestellten optimierten output-sensitiven Algorithmus zur Berechnung der Edit-Distanz von zwei Sequenzen $u$ und $v$ für die Einheitskostenfunktion. Dieser Algorithmus trimmt die Front-Werte mit Hilfe von zwei Strategien. Eine Strategie basiert auf der regionalen Alignmentqualität, die andere auf dem Abstand zum Kopf der Front.

Gliedern Sie die einzelnen Funktionen wie im Skript beschrieben.

Falls Sie Ihre Lösung in C implementieren, gilt: Im Material zur Vorlesung finden Sie einen Testrahmen und eine Datei mit passenden Typ-Deklarationen und Funktionsköpfen. Die `main`-Funktion in `outsensenseedist-mn.c` implementiert einen Optionsparser. Hier ist die Ausgabe der Optionsliste:

```
Usage: outsenseedist.x options
    -t use trimming method
    -e compute edit operation list (does not work with trimming)
    -p compute and verify tracepoints after generating alignment without trimming
    -w <length of word to generate>
    -r <referencefile>
    -s <file with pair of sample positions>
    -g <sequence1> <sequence2>
    -d <maximal lag to front head>, default 30
    -h <length of match history <= 64>, default 60
    -m <minimal percentage of matches in history>, default 55%
    -v verify intermediate results after generating alignment without trimming
```

Beachten Sie, dass die regionale Alignmentqualität über die Option `-m` als Anteil (in %) der Treffer relativ zur history size bestimmt wird. Diese Umrechnung wird bereits im Testrahmen vorgenommen. Nachdem Sie in der Datei `outsenseedist-trim.c` die drei Funktionen `add_matches`, `evaluatefrontentry` und `fastedisttrim` implementiert haben, können Sie Ihr Programm mit `make` compilieren. Damit Sie sich auf die Implementierung der Trimming-Strategien kümmern können habe ich in der Datei `outsenseedist.c` eine Version ohne Trimming realisiert, an der Sie sich orientieren können.

Vergessen Sie beim Testen die Option `-t` nicht, damit die Trimming-basierte Funktion aufgerufen wird. Zum Debuggen werden Sie wahrscheinlich die Option `-g` verwenden, die als Argumente die zwei zu vergleichenden Sequenzen bekommt. Beispiel: Durch den Aufruf (siehe auch Ziel `trim-lecture` im Makefile)

```
outsenseedist.x -g FREIZEIT ZEITGEIST -m 67 -h 3 -d 2 -t
```

werden die beiden Sequenzen aus dem Skript miteinander verglichen, und zwar mit Trimming der Fronten für die Parameter $hy = 3$, $rq = \frac{67}{100} \cdot hy = 2$ und $\mathbf{L} = 2$. Damit sollten sich die Werte aus dem Skript reproduzieren lassen. Um die *front*-Werte anzuzeigen, müssen Sie im C-Programm `#undef SHOWFRONT` durch `#define SHOWFRONT` ersetzen.

Sie sollten versuchen so weit zu kommen, das Ihr Programm die Ziele `trim-match` und `trim-nomatch` besteht. Für das erreichen der maximalen Punktzahl ist das aber nicht unbedingt erforderlich.

**Aufgabe 2.2** Schreiben Sie ein Programm, das den einfachen Chaining-Algorithmus aus der Vorlesung implementiert. Ihr Programm soll eine Menge von Treffern (Matches) in dem folgenden Format einlesen:

```
start_in_seq_1 end_in_seq_1 start_in_seq_2 end_in_seq_2 weight
```

Beispiel mit nur drei Treffern:

```
10 100 500 590 88
90 110 1000 1020 21
110 200 600 690 90
```

Nach der Berechnung der optimalen Kette soll Ihr Programm diese als Sequenz von Treffern wie folgt ausgeben:

```
10 100 500 590 88
110 200 600 690 90
# optimal chain of length 2 with score 178.0
# 182 bp covered on sequence 1 (coverage 26.00%)
# 182 bp covered on sequence 2 (coverage 26.00%)
```

Für die Berechnung der Überdeckung müssen Sie Ihrem Programm die Länge der beiden Sequenzen (im obigen Fall jeweils 700) übergeben.

Die Datei `matches.txt` in STiNE enthält im obigen Format 619 Treffer mit Identität $\geq 90\%$ zwischen den Genomen von zwei *E. coli*-Stämmen. Ein Genom (Sequenz 1) enthält 5 132 068 bp und das andere Genom (Sequenz 2) 5 449 314 bp.

Wenn Ihr Programm richtig funktioniert, dann enthält die optimale Kette 159 Treffer und hat einen Score von 3 838 964. Die Gesamtlänge der Treffer auf Sequenz 1 und 2 ist 2 013 745 bzw. 2 013 793. Die prozentuale Überdeckung ist damit 39,24% (Sequenz 1) und 36,95%. Zu Testzwecken verifizieren Sie bitte, dass Ihr Programm, nennen wir es 'chainer' die Ausgabe in der Datei `chain.txt` liefert. Diese enthält eine optimale Kette im obigen Format. D.h. der folgende Aufruf sollte keine Fehler liefern:

```
./chainer 5132068 5449314 matches.txt | diff - chain.txt
```

**Die Lösungen zu diesen Aufgaben werden am 09.05.2017 besprochen.**