Chair for Clinical Bioinformatics
Professor Dr. Andreas Keller
Dr. Christina Backes
Dipl.-Ing. Thomas Großmann
B. Sc. Yvonne Saara Gladbach

Bioinformatik 1 - Übungsblatt 4

Abgabe bis Dienstag 25.11.2014 um 11:59

Vorname, Nachname:
Matrikelnummer:

Anmerkungen :
Die Übungen dürfen in Gruppen bestehend aus max. 2 Personen abgegeben werden. Der Quellcode ist als Quellcodepaket abzugeben, muss auf den Cip-Pool Rechnern kompilieren, gut dokumentiert und getestet sein (Testfälle sind mit abzugeben!). Schicken Sie Ihre Abgabe als tar.gz Paket per Mail an Ihre Tutorin Yvonne Gladbach (yvonne.gladbach@ccb.uni-saarland.de) mit Betreff: "Übung X, Name Y, Name Z". Das Paket darf keine temporären oder binären Dateien enthalten. Beim Entpacken des Paketes soll ein übergeordnetes Verzeichnis erstellt werden, welches den Quellcode enthält. Der Quellcode muss durch ein mitgeliefertes Makefile kompilierbar sein.

Aufgabe 1 (25 P.):
Implementieren Sie eine Klasse `Alignment`. Verwenden Sie dazu volgendes Interface:

```
class EditDistance
{
        public:
                float operator() (char a , char b) const ;
};

template <typename Distance>
class Alignment
{
        public:
                Alignment();
                Alignment (const Sequence& seq1 , const
                   Sequence& seq2);
                Alignment (const Alignment& a);
                virtual ~Alignment();

                Alignment& operator= (const Alignment& a);
                bool operator== (Alignment a);
                void print (ostream& os = cout);
                void setSequence1 (const Sequence& seq);
                void setSequence2 (const Sequence& seq);
                const Sequence& getSequence1() const;
                const Sequence& getSequence2() const;
```

```
        void computeGlobalAlignment ();
        const std::pair<std::string, std::string>&
            getAlignment () const;
    private:
        Distance distance_;
};
```

Die Klasse `Alignment` soll also ein vollständiges Klasseninterface haben. Ein Ausgabeoperator soll das berechnete Alignment ausgeben. Verwenden Sie für die Sequenzen die auf Übungsblatt 1 schon von Ihnen implementierte Sequenzklasse, die Sie ggf. auch um Methoden erweitern können.
Implementieren Sie eine Methode `computeGlobalAlignment()`, die für die vorher gesetzten Sequenzen ein optimales Alignment unter Verwendung der Bewertungsfunktion berechnet.
Die Klasse `EditDistance` soll eine Methode `operator()` haben, die fur zwei Zeichen die jeweilige Bewertung zurückliefert.
Testen Sie Ihr Programm mit dem Alignmentproblem aus Aufgabe 2.

Aufgabe 2 (10 P.):
Gegeben sei folgendes Sequenz-Alignment-Problem:

$$\Sigma = \{A, C, G, T\}$$
$$s = ACGCA$$
$$p = ACCG$$

und die Bewertungsfunktion D:

	A	C	T	G	-
A	3	-1	-1	-1	-2
C	-1	3	-1	-1	-2
T	-1	-1	3	-1	-2
G	-1	-1	-1	3	-2
-	-2	-2	-2	-2	0

(a) Um welche Art von Bewertungsfunktion handelt es sich bei der Matrix D? Begründen Sie Ihre Antwort.

(b) Berechnen Sie unter Verwendung der gegebenen Bewertungsfunktion D die folgende Tabelle mit Hilfe des aus der Vorlesung bekannten "Dynamische Programmierung" - Algorithmus:

		A	C	G	C	A
A						
C						
C						
G						

(c) Führen Sie das Backtracking durch die Matrix durch und geben Sie ein mögliches optimales globales sowie lokales Alignment an.

Aufgabe 3 (8 P.):
Laden Sie die Gensequenz vom Gen "TTN" von der Vorlsungsseite (vom NCBI). Die Sequenzbibliothek besteht aus verschiedem $k = 6,\ k = 8,\ k = 10, \ldots, k = 40$.

(a) Erstellen Sie Histogramme für die verschiedenen k-mere wie oft jedes k-mer vorkommt.

(b) Abhängig von k, wie viele Sequenzen kommen weniger als 10mal vor?

Aufgabe 4 (7 P.):

(a) Erstellen Sie das 4-Spektrum zu den Fragmenten:
ACGAACG, *TGCTGAC*, *ACTGCT*, *AACGG*, *CTGACGA*

(b) Zeichnen Sie den De-Bruijn-Graphen zu diesem 4-Spektrum.

(c) Geben Sie einen Euler-Pfad durch den De-Bruijn-Graphen und den daraus resultierenden Text T an.