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

Bioinformatik 1 - Übungsblatt 2

Abgabe bis Dienstag 11.11.2014 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 (40 P.):

Es soll eine Klasse `Graph` erstellt werden, die einen Overlap-Graphen mittels einer Adjazenzliste realisiert. Zu diesem Zweck wird eine Klasse `Node` benötigt, welche die in einen Knoten eingehenden und ausgehenden Kanten verwaltet. Zusätzlich speichert `Node` auch die vom Knoten repräsentierte Sequenz. Die Klasse `Edge` bietet ein Interface für den lesenden Zugriff auf die Kanten an. Die Klasse `Graph` soll mindestens folgende Methoden bereitstellen:

- `bool hasNode(const Sequence& seq) const` gibt `true` zurück, falls ein Knoten für die gegebene Sequenz existiert.
- `Node& getNode(const Sequence& seq)` gibt den Knoten für die übergebene Sequenz zurück. Existiert kein Knoten für diese Sequenz, wird ein neuer Knoten erstellt und zurückgegeben.
- `void removeNode(Node& node)` löscht den angegebenen Knoten. Achten Sie darauf, dass diese Methode je nach Ihrer Implementierung alle existierenden Referenzen auf andere Knoten ungültig machen kann. Passen Sie daher auf, alle existierenden Referenzen ggf. zu aktualisieren.
- `Edge getEdge(Node& src, Node& target)` gibt die Kante zwischen `src` und `target` zurück, falls diese existiert. Existiert keine Kante zwischen `src` und `target` wird diese erstellt.
- `Edge getEdge(const Sequence& src, const Sequence& target)` identifiziert die zu den übergebenen Sequenzen gehörenden Knoten und verbindet diese mittels `getEdge(Node&, Node&)`. Existieren einer oder beide der Knoten nicht, so werden sie erzeugt.

- `std::ostream& operator << (std::ostream& strm, const Graph& graph)` ein globaler Ausgabeoperator, der den Graphen im Graphviz Format ausgibt.
- Implementieren Sie einen globalen Eingabeoperator `std::istream& operator >> (std::istream& strm, Graph& graph)`, welcher die Beispieldatei fragments.fta einliest und für jede Sequenz einen Knoten anlegt. Des Weiteren soll eine gerichtete Kante zwischen zwei Knoten erzeugt werden, wenn die Sequenzen der Knoten eine maximale Überlappung > 0 haben. Speichern Sie die berechnete Überlappung im Graphen. Verwenden Sie dazu die Methode `overlap` Ihrer Sequenzklasse.

Die Klasse Node soll folgende Mindestanforderungen erfüllen:

- `explicit Node(const Sequence& sequence)` erstellt einen Knoten für die gegebene Sequenz.
- `std::vector<Edge> getOutEdges() const` gibt eine Liste aller ausgehenden Kanten zurück.
- `std::vector<Edge> getInEdges()` const gibt eine Liste aller eingehenden Kanten zurück.
- `Edge buildEdgeTo(Node& node)` erstellt eine Kante von diesem zum übergebenen Knoten mit korrektem Kantengewicht. Falls diese Kante bereits existiert, wird die bestehende Kante zurückgegeben.
- `void removeEdgeTo(Node& node)` falls eine ausgehende Kante zu `node` besteht, wird diese entfernt. Achten Sie darauf, dass diese Methode je nach Ihrer Implementierung alle existierenden Referenzen auf andere Knoten ungültig machen kann. Passen Sie daher auf, alle existierenden Referenzen ggf. zu aktualisieren.

Die Klasse Edge soll eine unveränderliche (immutable) Repräsentation einer Kante darstellen. Implementieren Sie dazu folgende Zugriffsmethoden:

- const Node& getSource() const liefert den Ausgangsknoten der Kante zurück.
- const Node& getTarget() const liefert den Zielknoten der Kante zurück.
- unsigned int getOverlap() const liefert die Überlappung der Knoten zurück.

Generieren Sie mittels Graphviz und Ihres Ausgabeoperators eine Visualisierung für die Datei fragments.fta. Der hierfür benötigte Code ist Teil der Abgabe.

Achten Sie darauf keine Daten doppelt zu speichern. Denken Sie daran Copykonstruktoren, Destruktoren und Zuweisungsoperatoren zu implementieren, wenn Sie mit dynamischem Speicher arbeiten. Verwenden Sie `forward declarations`, um zyklische Quellcodeabhängigkeiten aufzulösen. Erweitern Sie das Interface der Klassen ggf. um eigene Methoden, um die Aufgabe leichter bearbeiten zu können.

Aufgabe 2 (10 P.):
Sei $G = (V, E)$ ein ungerichteter Graph mit m Kanten ($|E| = m$) und bezeichne $d(v)$ den Grad eines Knotens v.

(a) Zeigen Sie:

$$\sum_{v \in V} d(v) = 2m$$

(b) Zeigen Sie außerdem: Die Anzahl der Knoten mit ungeradem Grad ist gerade.

Achten Sie auf formale Korrektheit!