Praktikum Data-Mining und Sequenzanalyse
Aufgaben zu Alignments

Markus Fleischauer

Allgemeine Projektanforderungen

- Versionskontrolle mit Git
- Build-Management mit Gradle
  - Verwendung eurer ”exactSearch” library als dependency
- Implementierung der Klassen mit den entsprechenden Algorithmen
- API Dokumentation mit javadoc
- Benutzerinterface als Kommandozeilenprogramm (CLI) inklusive Hilfe (–help)

1 Implementierung - Klassenstruktur

Die folgenden Punkte stellen die Mindestanforderungen an die Implementierung dar. Sollten Sie den Einsatz weiterer (abstrakter)Klassen, Interfaces oder Pakete für nötig oder sinnvoll halten, steht es ihnen frei diese umzusetzen.

- Package: *alignment.model*
  Ein Alignment soll zwei Sequenzen gleichen Typs und gleicher Länge Speichern und den Zugriff auf diese ermöglichen.
  - Definieren Sie eine geeignete API für ein Alignment durch ein Interface mit dem Namen **Alignment**.
  - Implementieren Sie die oben definierte API in geeigneter Weise aus. Die Aligment Klasse soll als Rückgabewert ihrer Alignmentalgorithmen dienen.
- Package: *alignment.algo*
  Dieses Paket soll sowohl API als auch die Implementierung verschiedener Alignment-Algorithmen enthalten. Eine Aligner bekommt 2 Sequenzen des gleichen Typs als Eingabe und gibt ein optimales Alignment (siehe alignment.model package) oder nur dessen Score zurück.
  - Geeignete API durch Interface **Aligner** definieren. Ein Aligner bekommt zwei Sequenzen als Eingabe und hat ein Alignment bzw. einen Score als Ausgabe.

  - Aligner mit den folgenden Algorithmen aus implementieren:
    * **Naiv Lokal**
    * **Naiv Global**
    * **Needleman-Wunsch**
      · Score-Berechnung mit linearem Speicher
      · Alignment-Berechnung mit quadratischem Speicher
    * **Smith-Waterman**
      · Score-Berechnung mit linearem Speicher
      · Alignment-Berechnung mit quadratischem Speicher
  - Die Klasse **Scorer/Evaluator** verwaltet eine Scoringmatrix und kann zwei Character bewerten (`score(char1,char2)`).
  - Ein FileReader soll das einlesen einer Scoringmatrix aus einer Datei ermöglichen.

- Package: *alignment.cli*
  Paket für den Kommandozeilenparser und die main() Methode:
  - Klasse mit `main()` Methode
  - Einlesen von Datei(en) mit Sequenzen
  - Parameter zum einlesen einer Datei mit Scoringmatrix
  - Parameter für den Alignment-Algorithmus

2 Validierung der Algorithmen (Unit Tests)

Gewährleisten Sie die Korrektheit ihrer Algorithmen durch die Implementierung geeigneter (Unit)Tests. *Hinweis: Die frühzeitige Implementierung von Tests erleichtert den Debuggingprozess erheblich.*

1. Überprüfen Sie mit Hilfe einige geeigneter (Unit)Tests anhand überschaubaren Beispiele die Korrektheit ihrer implementierten Algorithmen. *Hinweis: "Toy examples" mit bekannter Lösung, Sonder-, und Extremfälle, "correct failure".*

2. Suchen Sie in einer zufällig erzeugten Protein Sequenz der Länge 100 ein zufällig erzeugtes Pattern der Länge 10. Validieren Sie, dass beide Varianten (linearer/quadratischer Speicher) ihrer Smith-Waterman Implementierung den gleichen Score und schließlich das gleiche Ergebnis liefern.

3. Implementieren Sie einen weiteren Test nach obigen Vorbild und mit geeigneten Sequenzlängen, sodass Sie testen können ob ihr Smith-Waterman-Algorithmus und ihre naive lokale Alignment-Implementierung die gleichen Lösungen liefern. Der Test soll nicht länger als 1min laufen.

4. Aligniert 2 zufällig erzeugte Sequenzen in mit geeigneten zufälligen Längen. Validieren Sie, dass beide Varianten (linearer/quadratischer Speicher) ihrer Needleman-Wunsch Implementierung den gleichen Score und schließlich das gleiche Ergebnis liefern.

5. Implementieren Sie einen weiteren Test nach obigen Vorbild und mit geeigneten Sequenzlängen, sodass Sie testen können ob ihr Needleman-Wunsch-Algorithmus und ihre naive globale Alignment-Implementierung die gleichen Lösungen liefern. Der Test soll nicht länger als 1min laufen.

3 Anwendung der Implemetierung (Protokollaufgaben)

Allgemeine Hinweise:

- Die folgenden Aufgaben, sollen mit Hilfe ihres Kommandozeilenprogramms gelöst werden.
- Geben sie stets die Ausgeführten Befehle an, sodass die Ergebnisse reproduzierbar und nachvollziehbar sind.
- Geben sie das verwendete Testsystem an (CPU, RAM, HDD/SSD, OS(-Version), Java-Version, JVM Parameter).
- Erläutert Sie ihre Implementierung. Gab es Schwierigkeiten? Gibt es noch Fehler?

Daten:
Die Testdaten liegen im `FASTA`-Format vor und sind unter `https://bio.informatik.uni-jena.de/lehre/winter-1920/data-mining-und-sequenzanalyse/` zu finden.

Aufgaben:

1. Vergleicht die naiven Algorithmen anhand selbst gewählter, kleiner Beispiele mit den effizienten Algorithmen. Wie große Instanzen könnt Ihr in vertretbarer Zeit (z.B. einige Minuten) mit den verschiedenen Algorithmen behandeln?
2. Vergleicht die Proteine im Unterverzeichnis `RiboProtS1` per globalem und lokalem Alignment. Benutzt zwei unterschiedliche Matrizen. Welche Proteine sind am ähnlichsten? Welche Schlussfolgerungen ziehen Sie aus dem Vergleich?
3. Vergleicht die Laufzeit und den Speicherplatzverbrauch der Implementierungen mit linearem Speicherverbrauch mit der Implementierung mit quadratischem Speicherverbrauch beim globalen Alignment dieser Proteine (*Hinweis: Kein Traceback*). Überlegen Sie sich, wie Sie sinnvoll Angaben zum Speicherplatzbedarf machen können, d.h. diesen messen oder berechnen? Sollten die Proteine zu kurz für einen aussagekräftigen Vergleich sein, können Sie diese (mehrfach) duplizieren. Ab welcher Sequenzlänge lässt sich das Alignment aufgrund von Speicherproblemen nicht mehr berechnen?
4. Inwieweit eignen sich die Algorithmen für die inexakte Suche in großen Genomen? Gibt es Möglichkeiten ein exaktes Alignment (inklusive Traceback) mit linearem Speicherverbrauch zu berechnen? Gibt es noch andere Alternativen?