Algorithmen und Datenstrukturen in der Bioinformatik

Vorlesung von Prof. Dr. Knut Reinert
Wintersemester 2007/2008, FU Berlin

reinertATmi.fu-berlin.de

1.1 Ziel der Vorlesung

Aus dem Vorlesungsverzeichnis:

> Die Vorlesung gibt eine Einführung in grundlegende algorithmische Techniken und Datenstrukturen für Strings und Graphen. Dabei stehen bioinformatische Fragestellungen und die dafür erforderlichen Grundlagen im Vordergrund.

Es geht im Wesentlichen darum, *Paradigmen* (z. B. Dynamisches Programmieren, Divide–and–Conquer) und *Konzepte* (z. B. Formale Sprachen und erkennende Automaten) kennenzulernen, die in der algorithmischen Bioinformatik eine zentrale Rolle spielen.

Diese Paradigmen und Konzepte werden zunächst allgemein vorgestellt und dann anhand von wichtigen bioinformatischen Themen vertieft (z. B. Dynamisches Programmieren → Sequenzalignment).

1.2 Sprache · Folien · "Skript" · Tafel · Literatur

- Die Vorlesung wird auf Deutsch gehalten, es sei denn, Englisch ist von allen erwünscht. Die Folien sind auf Englisch (bis auf diesen Teil). Die Übungsblätter sind ebenfalls auf Englisch, können natürlich aber auf Deutsch bearbeitet werden. In den Tutorien wird ebenfalls Deutsch gesprochen. Die Prüfungen und das Testen der aktiven Teilnahme sind auf Deutsch.
- Die Folien
  - werden zeitnah online verfügbar gemacht. Eine Mitschrift des *Folien*inhaltes ist nicht nötig.
  - sind **KEIN** Skript. Hinzu kommen Erklärungen, Tafelanschrieb, Beispiele, Hinweise, Anekdoten, Overheadfolien, . . .
    → **Gehen Sie in die Vorlesung** und machen Sie sich Notizen.
    Lesen Sie ergänzende Literatur → (Liste jeweils am Anfang der Kapitel, vollständige Liste auf der Webseite)
- Folien"autoren": Clemens Gröpl (FU Berlin), Prof. Daniel Huson (Uni Tübingen), Prof. Knut Reinert (FUB), Gunnar Klau (FUB). . . ("Lecture Pool")

1.3 Übungen

- Für die Übungen sind Michael Bartel und Paul Pyl verantwortlich.
- Aktuelle Nachrichten werden bevorzugt über die Mailingliste verbreitet, Downloads gibt es auf der *Veranstaltungswebseite*
  `http://www.inf.fu-berlin.de/inst/ag-bio/file.php?p=ROOT/Teaching/Lectures/index.list.htm`
- Anmeldung und Einteilung in Übungsgruppen nehmen die Tutoren vor.

1.4 Übungen · Ablauf und Regeln

- Es wird zwölf Übungsblätter geben und zwar
  - neun Blätter mit schriftlichen Aufgaben und
  - drei Programmieraufgaben
- Die regulären Übungsblätter werden in den Übungen ausgeteilt und können von der Webseite heruntergeladen werden. Sie haben dann 6 Tage Zeit zur Bearbeitung und geben sie einen Tag vor der Übung ab.
- Die Blätter werden nicht korrigiert, auf allen Blättern darf höchstens eine Aufgabe angekreuzt werden die *nicht* bearbeitet wurde. Eine Aufgabe nicht ankreuzen heißt, dass Sie sie gelöst haben und eine Musterlösung an der Tafel präsentieren können. Falls Sie dies nicht können, bekommen Sie 10 Punkte (von 100 möglichen) abgezogen (pro Aufgabe die sie NICHT rechnen können), maximal bekommen sie 20 abgezogen.
- Die Übung wird also von Ihnen gestaltet, die Tutoren leiten die Übungen und ergänzen gegebenenfalls die Lösungen.

1.5 Prüfungsleistung

- Klausur am Semesterende: *19. Februar 2009*
- 2.Klausur in der ersten Woche des Sommersemesters.
- Beide Klausuren sind unabhängige Prüfungsversuche im Sinne der Studienordnung

1.6 Aktive Teilnahme

- Sie bearbeiten die wöchentlichen Übungsaufgaben *schriftlich* und präsentieren Ihre Lösungen in den Übungsstunden.
- Es werden zwei *Reviews* geschrieben. In diesen werden die Inhalte der Übungen abgeprüft. In jdem Review könne sie 40 Punkte erreichen.
- Es werden drei Programmieraufgaben gestellt[1]. Die Programmieraufgaben werden in *Code Reviews* bewertet. Sie dürfen Gruppen mit max. drei Teilnehmern bilden.

Um die aktive Teilnahme bescheinigt zu bekommen, müssen Sie 50 von 100 Punkten erreichen. Davon sind

[1] Verwenden Sie eine Programmiersprache Ihrer Wahl. Der Code muss aber bei der Abgabe auf einem Rechner laufen, und Sie müssen erklären können, was er tut.

1. 20 durch Vorrechnen bzw. code review zu erreichen.
2. jeweils 40 in jeder review zu erreichen.

1.7 Content

- Exact string matching (++)
- Pairwise sequence alignment by dynamic programming (+++)
- Fasta and Chaining problem (+)
- Database searching with Blast (+)
- Finite automata, languages, grammars, and regular expressions (+++)
- Multiple sequence alignment (+++)
- Clustering (+)
- Markov chains, hidden Markov models (+++)