Grundlagen der Bioinformatik
Assignment 1: Substring Search

SS 2017

Yvonne Lichtblau

Allgemeines

Ablauf der Übung

- Insgesamt 5 Übungszettel
- Abgabe in Gruppen von 3 Personen
- Pro Übung ~zwei Wochen Bearbeitungszeit

- 6 Pflichttermine (ca. alle zwei Wochen)
  - Ausgabe neuer Übungszettel
  - Vorstellung der Lösungen letzter Übungszettel

- Termine dazwischen
  - Klärung von Fragen zur aktuellen Übung
  - Übungen nach Wunsch (vorher Email)

- Webseite:
  https://www.informatik.hu-berlin.de/de/forschung/gebiete/wbi/teaching/archive/SS17/ue_bioinfo/

Termine im Einzelnen

- **27./28.04.2017, 1. Übung: Exact String Matching**
  (Abgabe: 09.05.2017)
- **11./12.05.2017, 2. Übung: Alignments**
  (Abgabe: 30.05.2017), Korrektur 1. Übung
- **01./02.06.2017, 3. Übung: Hierarchical Clustering**
  (Abgabe 13.06.2017), Korrektur 2. Übung
- **15./16.06.2017, 4. Übung: Introduction to R**
  (Abgabe: 27.06.2017), Korrektur 3. Übung
- **29./30.06.2017, 5. Übung: Microarray Analysis**
  (Abgabe: 11.07.2017), Korrektur zu 4. Übung
- **13./14.07.2017**, Korrektur 5. Übung

- Ansonsten jeden Donnerstag/Freitag Klärung von Fragen

Übungsschein

- Schein: Voraussetzung für die Prüfung

- Abgabe der Übungszettel in Gruppen von 3 Personen
  - Mindestens 51% der Punkte von jedem Zettel benötigt
  - Ein Zettel darf ausgelassen werden
  - Gruppen bestehen/scheitern nur als Ganzes

- Vorstellung der Lösungen der letzten Übung durch 2-3 Gruppen
  - Wir losen vorstellende Gruppen aus
  - Eine Gruppe muss Lösung vortragen
    → immer einen Vortrag parat haben
  - Ziel: Jede(r) Gruppe (Student) trägt einmal vor

- Klausurtermin: 14.08.2017, Raum 3.001, 11-14 Uhr

Aufgaben

- Implementationen in JAVA
- I.d.R. sind Eingabe, Ausgabe und Aufrufform vorgegeben
- Textaufgaben
  - Abgabe als PDF
  - Müssen Gruppennnamen beinhalten
- Code-Abgaben
  - Abgabe als Jar (class und source files!)
  - Abgaben ohne Quellcode werden ignoriert!
  - Dateiname: GdBioinf[ÜbungNr]_[Gruppe].jar
  - Kompilierung unter Java 1.7 (oder niedriger)
  - **Jar Datei auf gruenau2 testen!**
    (gruenau2 ist einer der Rechner hier am Institut, auf den ihr euch mit ssh einloggen können: ssh username@gruenau2.informatik.hu-berlin.de)
- Nichteinhaltung der Regeln: Punkteabzug

Assignment 1
Substring Search

Overview – Assignment 2 (20P)

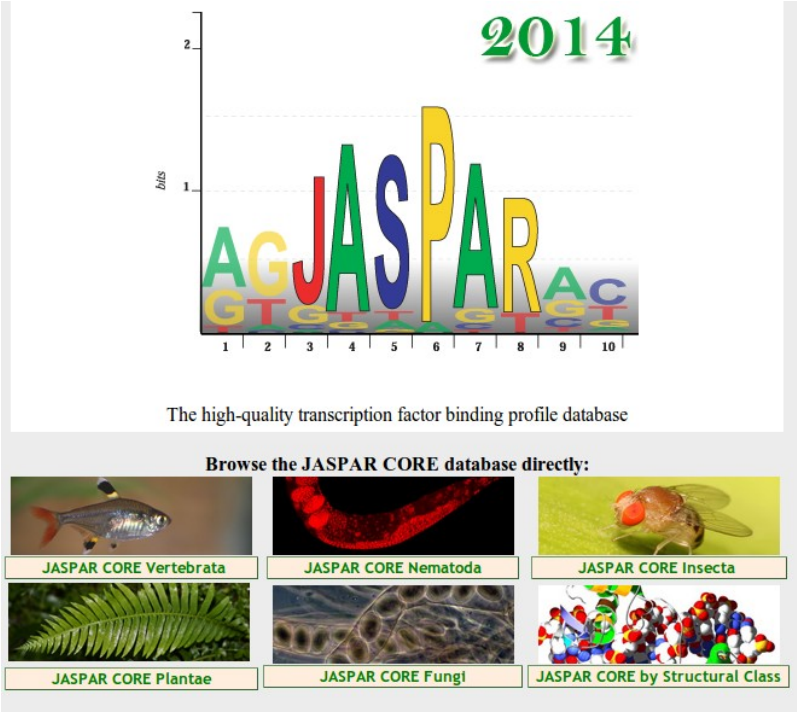
(1) Analyse transcription factor *GATA2* (4P)

(2) Substring search (10P)

(3) Properties of Boyer Moore Algorithm (6P)

(1) Transcription Factor *GATA2* (4P)

- *GATA2* is a transcription factor with established or assumed roles in a variety of different human cancers
- Seach *GATA2* in the JASPAR database



- JASPAR contains a set of transcription factor DNA-binding preferences, modeled as matrices
- Profiles derived from published collections of TF-binding sites
- Profile can be used to scan query sequences for presence of potential binding sites

(1) Transcription factor *GATA2* (4P)

- Search human *GATA2* in the JASPAR database
- Compute the information content of each position in the position specific weight matrix (PSWM, aka frequency matrix)
    - Find the exact formula on the web

- Submit
  - URL to the JASPAR information on *GATA2* (version .1, length 5)
  - Formula for information content used in sequence logos (1P)
  - Frequency matrix and IC of every position of the PSWM (2P)
    - Indicate process of calculation for at least one position
  - List of cancer types to which *GATA2* is associated and supporting papers from PubMed, 8x(Cancer, Title, PMID) (1P)
  - http://www.ncbi.nlm.nih.gov/pubmed/
    - Useful databases: Uniprot (http://www.uniprot.org), OMIM (http://www.ncbi.nlm.nih.gov/omim/), NCBI (http://www.ncbi.nlm.nih.gov)

(2) Substring Search (10P)

(a) Load a template string (~60MB) into main memory (3P)
- File: sequence.fasta
- Don't use the concatenation parameter +

(b) Load a set of patterns (0P)
- File: patterns.fasta

(c) Search all exact occurrences of all patterns in the template and print first ten positions to STDOUT (7P)

(2.a) Load a sequence (3P)

- You need to load sequences in FASTA Format
  - „A sequence in FASTA format begins with a single-line description, followed by lines of sequence data. The description line is distinguished from the sequence by a greater-than symbol („>“) in the first column. … The sequence ends if another line starting with a „>“ appears; this indicatesthe start of another sequence“
  - Example:

```
> gi|5524211|gb|AAD44166.1| cytochrome b
LCLYTHIGRNIYYGSYLYSETWNTGIMLLLITMATAFMGYVLPWGQMS
EWIWGGFSVDKATLNRFFAFHFILPFTMVALAGVHLTFLHETGSNNPL
LLLLLALLSPDMLGDPDNHMPADPLNTPLHIKPEWYFLFAYAILRSVP
GLMPFLHTSKHRSMMLRPLSQALFWTLTMDLLTLTWIGSQP
>gi|5454351|gb| cytochrome x
LLLITMATAFMGYVLPWGQMSLCLYTHIGRNIYYGSYLYSETWNTGIM
LLLITMATAFMGYVLPWGQMS
```

  - File: sequence.fasta

(2.b) Load a Set of Patterns (0P)

- You will get another file which contains a set of sequences in FASTA format. These should be used as patterns.
- File: pattern.fasta
- Format as on previous slide

(2.c) Substring Search (7P)

- Implement an algorithm of your choice to search all occurrences of all patterns in the template

- Methods indexOf(„AT“), equals(„AT“), etc. are not accepted
  - Use charAt() (to access a string like an array)

- Submit:
  - A Java Archive including class files and source code
  - Commandline:

```
java -jar GdBioinf1_[Gruppe].jar pattern.fasta sequence.fasta
```

  - Print pattern, number of occurrences and first ten positions to STDOUT:

```
tccgga: 2506
[29562, 30667, 134810, 244142, 276754, 315062, 318466, 330540,
344995, 347336]
gctacc: 6799
...
```

  - Runtime of the algorithm and a sentence on the implementation (naive, Boyer Moore, ...)

For Orientation

Number of occurrences:

- tccgga: 2506
- gctacc: 6799
- taataa: 28279
- cctcagc: 17520
- cctgcagg: 2425
- ggcgcgcc: 141
- ccccccccccc: 140
- aaaaaaaaaaa: 52695
- aaaaaaaaaaaa: 44140
- aaaaaaaaaaaaaaa: 25063
- aaaaaaaaaaaaaaaaaaaa: 8571

(3) Properties of the Boyer Moore Algorithm (6P)

- Give a template and a pattern such that the BM algorithm, as presented in the lecture, needs to calculate in the order of |T| * |P| caracter comparisons and explain why (3P)

- Many implementations of the BM algorithm actually drop the good suffix rule, especially for larger alphabets. Give an argument why and when this can be useful (3P)

Abgabe

- Abgabe bis Dienstag den 09.05.2017 um 23:59 Uhr

- Fragen zur Übung gerne per Email:
  yvonne.lichtblau@informatik.hu-berlin.de
- Abgabe als .zip hochladen:https://box.hu-berlin.de/u/d/9df9c4263b/
  - Dateiname: GdBioinf[ÜbungNr]_[Gruppe].zip
    (z.B. GdBioinf1_Gruppe2.zip)
  - Inhalt des .zip:
    - PDF mit Antworten zu 1, 2 und 3
    - Code als Jar Datei (GdBioinf[ÜbungNr]_[Gruppe].jar)
    - Sourcecode
- .jar auf gruenau2 testen!

- Wir testen mit neuen Pattern
  (Länge: 4-50bp, Alphabet: E = {acgtn})