Übung 4
Algorithmische Bioinformatik

WS 15/16

Yvonne Mayer

Lösungen/ Wettbewerb Übung 3

Lösungen Übung 3 vorstellen

- Zirkuläre Strings und Rotationen (8P)

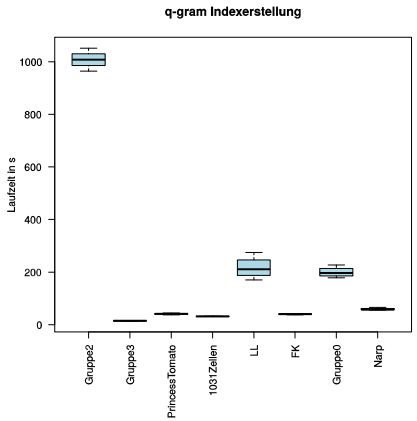
GruppeAB

- q-gram Index (12P)
  - Indexierung (4P)
  - Suche (5P)
  - Beschreibung Vorgehen (3P)

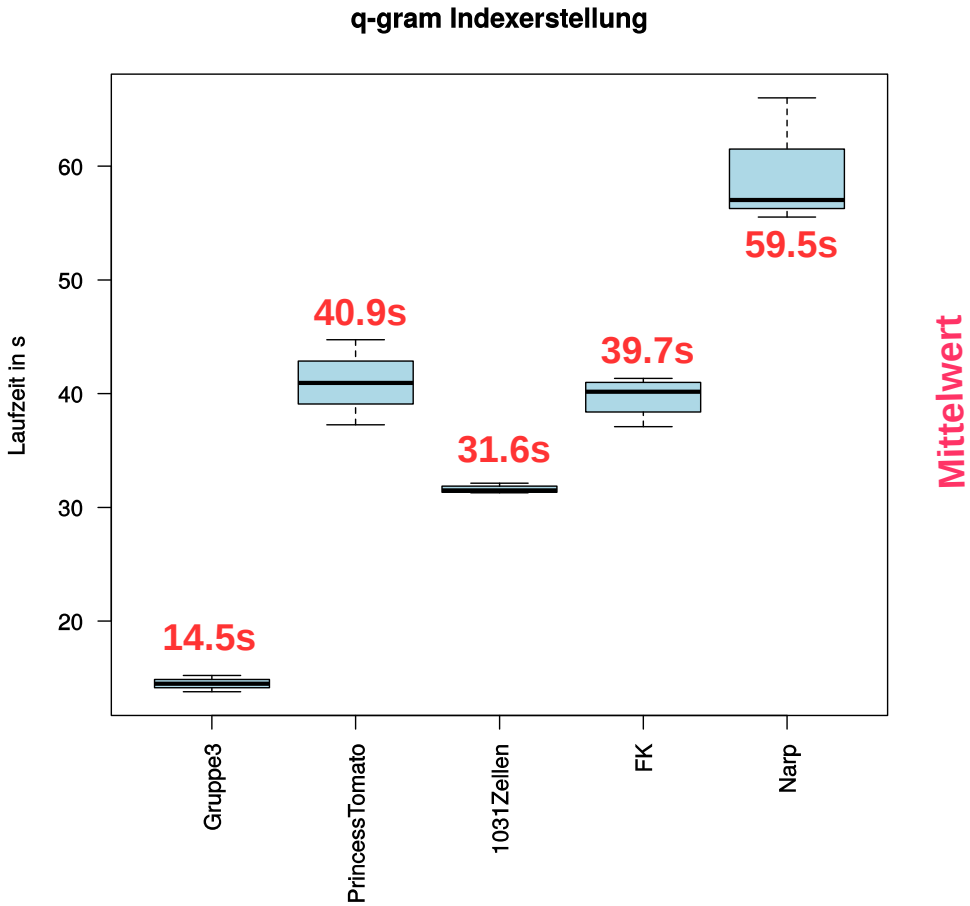
GruppeXY
GruppeYZ

Wettbewerb Part 1: Indexierung (q-grams)

Ergebnisse aller Gruppen:

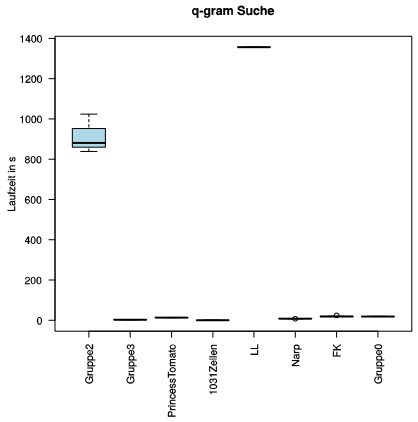


Ergebnisse aller Gruppen mit Laufzeit < 1min:

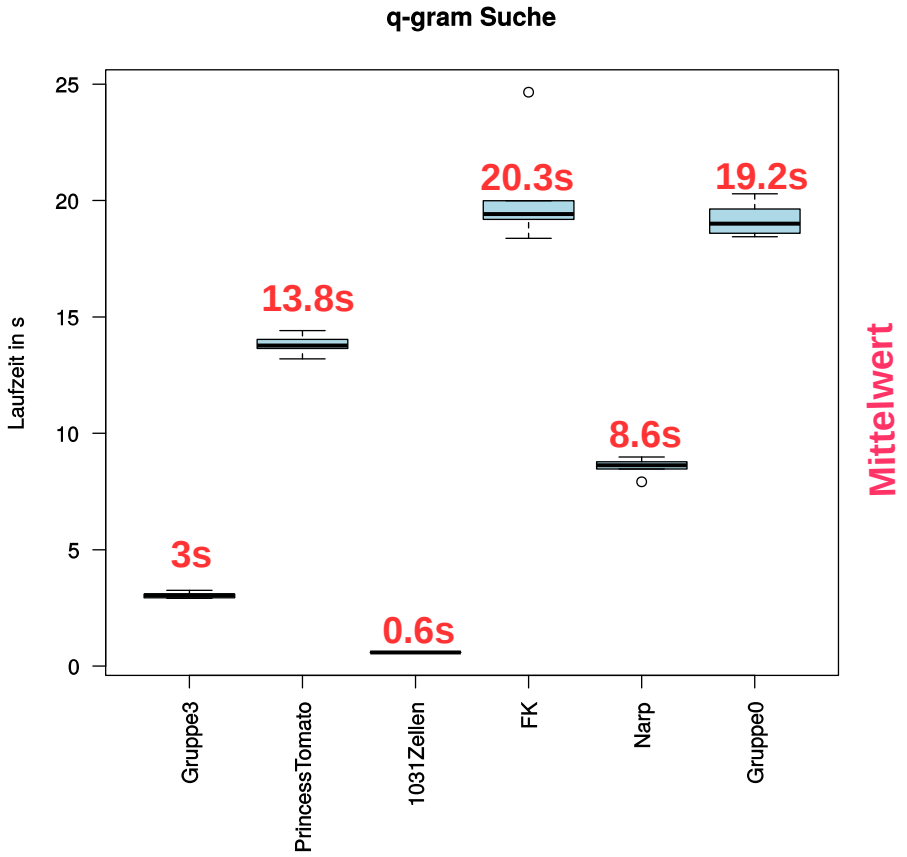


Wettbewerb Part 2: Suche (q-grams)

Ergebnisse aller Gruppen:



Ergebnisse aller Gruppen mit Laufzeit < 20s:



Wettbewerb

	Gruppe0	Gruppe1	Gruppe2	Gruppe3	PrincessTomato	1031Zellen	LL	Narp	FK
Challenge1 (beliebiges Stringmatching)	0	0	0	0	0	3	0	1	2
Challenge2 (Boyer Moore)	0	0	1	1	0	3	0	0	2
Challenge3a (q-grams: Index)	0	0	0	3	1	2	0	0	1
Challenge3b (q-grams: Suche)	0	0	0	2	0	3	0	1	0
Summe	0	0	1	6	1	11	0	2	5

(Ergebnisse unter Vorbehalt, bis Korrektur vollständig abgeschlossen)

Übung 4

t-Repeats

- In DNA kommen sehr häufig Repeats vor
  - Z.B. Short/Long Interspersed nuclear Sequences (SINEs/LINEs)
  - Beispiel SINE: Alu-Familie. Alu Repeats sind etwa 300 Nukleotide lang, kommen etwa 500.000 mal im menschlichen Genom vor und machen 10% des Erbguts aus
    [Alu repeats and human genomic diversity. Batzer MA, Deininger PL; Nature Review Genetics 2002]
- Definition:
  - Ein t-repeat ist ein Substring, der im Template mindestens t mal vorkommt (mit t>1)
  - Überlappungen sind erlaubt
- Beispiel: In ABABA sind
  - 2-Repeats: A, B, AB, ABA, BA
  - 3-Repeats: A

Übung 4.1

(1) t-Repeats (7 Punkte)

- Entwerfen Sie einen Algorithmus, der in linearer Zeit alle t-Repeats in einer Sequenz mit Hilfe von Suffixtrees findet.
- Nur Repeats sollen gefunden werden;
  Startpositionen oder Multiplizität sind nicht gefragt.

(Lösung als Text, nicht implementieren!)

Übung 4.2

(2) Suffixarrays (13 Punkte)

- Entwickeln Sie ein Programm, dass das Suffixarray zum Template im Hauptspeicher baut (nach Manber/Meyer) und in diesem Array alle Vorkommen der Patterns findet (10P).
- Verwenden Sie die Pattern aus Aufgabe 1 und ein neues, kürzeres Template (Download auf Übungsseite).
- Messen Sie die Gesamtlaufzeit über alle Pattern (1P)
  - nur Suche, nicht Arrayaufbau
- Kommentieren Sie Ihr Programm (2P).

Programmausgabe

- Ausgabe auf STDOUT
- Pro Paar (Template/Pattern) muss das Programm dann ausgeben (in jeweils neuer Zeile):
  - Pro Pattern:
    - Pattern
    - Patternlänge
    - Anzahl Fundstellen
    - Startpositionen der ersten zehn Fundstellen
  - Gesamtlaufzeit über alle Pattern

```
tccgga
Length: 6
Occurrences: 255
Positions: 29561, 30666, 134809, 244141, 276753, 315061, 318465, 330539, 344994, 347335
(…)
Runtime: 200ms
```

Programmaufruf

- Programm muss wie folgt aufrufbar sein:
  ```
  java -jar Assignment4_GrXY.jar f1 f2
  f1: Dateiname Template
  f2: Dateiname Patterns
  ```
  Ausgabe auf STDOUT
- Alle Dateien sind unkomprimiert.

Wettbewerb

- Wir vergeben Punkte für die schnellste Implementierung der Suffixarrays (Aufbau und Suche gemeinsam).

- Wir messen die Gesamtlaufzeit Ihres Verfahrens über alle Pattern (mittels Linux „`time`“), Parallelisierung lohnt nicht.

- Wir verwenden 20 neue Patterns, deren Länge zwischen 4 und 50 Zeichen liegt (Alphabet E = {ACGTN}).

- Wir verwenden ein anderes, ähnlich langes Template zum Testen.

- Wettbewerbspunkte:
  - Platz 1: 3 Punkte
  - Platz 2: 2 Punkte
  - Platz 3: 1 Punkt

Abgabe

- Abgabe bis Sonntag den 13.12.2015 um 23:59 Uhr
- Abgabe per Email an: mayeryvo@informatik.hu-berlin.de
  (gerne auch Fragen zur Übung per Email)
  - PDF mit Antworten zu allen Fragen (Aufgabe 4.1)
  - Jar Datei mit Quellcode
    (Dateiname: Assignment4_GrXY.jar, jar ggf. als rar verpacken)
  - Kompilierung unter Java 1.7, Jar Datei auf gruenau2 testen,
    Abgaben ohne Quellcode werden ignoriert!
- Bitte die geschätzte Bearbeitungszeit mitteilen
  (z.B. 10h für zwei Teilnehmer)

Zur Orientierung

Anzahl Vorkommen der Pattern im Template

- tccgga: 255
- gctacc: 696
- taataa: 2527
- cctcagc: 2114
- cctgcagg: 208
- ggcgcgcc: 21
- ccccccccccc: 8
- aaaaaaaaaaa: 7021
- aaaaaaaaaaaa: 5921
- aaaaaaaaaaaaaaa: 3436
- aaaaaaaaaaaaaaaaaaaa: 1228