Ludwig-Maximilians-Universität München
Institut für Informatik
Prof. Dr. Volker Heun
Markus Joppich
Jens Quedenfeld

Sommersemester 2016
Programmieraufgabe
18. Mai 2016

Algorithmische Bioinformatik I

Abgabetermin: Donnerstag den 2. Juni, vor der Vorlesung

Aufgabe 1 (Programmieraufgabe 12P)

Modifiziere den Algorithmus von Knuth, Morris und Pratt und den Algorithmus von Boyer und Moore, so, dass er alle Vorkommen eines Suchwarts $s \in \Sigma^m$ in einem Text $t \in \Sigma^n$ findet.

Implementiere diese Algorithmen für die oben genannte Modifikationen sowie die naive Suche in Java.

Beachten Sie folgende Vorgehensweise bei der **Implementierung**:

- Die Ausgabe für alle drei Varianten ist gleich.
- Start und Ende einer Lösung bilden ein Intervall.
- Der Aufruf der bereits in Java implementierten Suche wird mit 0 Punkten bewertet. Dies gilt auch für RegEx-Aufrufe, etc.

Da die Ausgabe von allen drei Varianten gleich ist, empfiehlt es sich **wiederverwendbare** Funktionen zu überlegen (insbesondere die auch die Verwendung einer abstrakten Klasse). Beachten Sie dies beim Entwurf Ihrer Implementierung. Die Abgabe erfolgt üben den Abgabeserver `https://services.bio.ifi.lmu.de/abgabeserver`.

Aufruf: `java -jar` *gruppennname*`_blatt02.jar [kmp|bm|naive] "`$s_0 \dots s_m$`" "`$t_0 \dots t_n$`"`

Das **erste** Argument $\in$ { `kmp`, `bm`, `naive` } gibt an, ob die Knuth-Morris-Pratt-, Boyer-Moore- oder naive Version gewählt werden soll. Sollten Sie eine Variante noch nicht implementiert haben, lassen Sie die Ausgabe einfach leer (oder rufen Sie eine implementierte Variante auf). Kombinieren Sie bei Boyer-Moore sowohl die **bad-character** als auch die **strong good-suffix** Regel.

Das **zweite** Argument gibt das zu suchende Wort s an, das **dritte** Argument einen Referenz-/Datenbank-/zu durchsuchenden String t.

Die Ausgabe soll **für jedes gefundene Vorkommen** zeilenweise und durch Tabulator/tab-getrennt jeweils das gefundene **Wort**, **Start**-, **End**-Position in t sowie die Anzahl benötigter **Vergleiche** angeben. Das erste Zeichen in t hat den **Index 0**.

Bitte vergessen Sie nicht, dass Ihre *jar*-Datei auch **Ihren Source-Code enthalten** muss!

Beispiel für naive:

```
s:               C  C  T  T  T  T  G  C
t:    GCTTCTGCTA C  C  T  T  T  T  G  C
idx   0123456789 10 11 12 13 14 15 16 17
```

Ausgabe:

```
CCTTTTGC  \t  10 \t 17 \t 21
```