Prof. Dr. Knut Reinert
David Weese
Konrad Rudolph
Christoph Hartmann

Institut für Informatik
AG Algorithmische Bioinformatik

Freie Universität Berlin

Algorithmen und Datenstrukturen in der Bioinformatik

Elftes Übungsblatt WS 10/11

Abgabe Montag, 17.01.2011, 15:00 Uhr

Name: Übungsgruppe: A☐ B☐ C☐

Matrikelnummer:

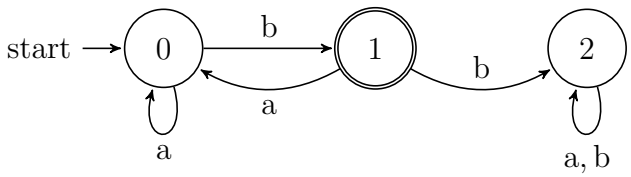
Niveau I

Aufgabe1: Additional Cost Matrices

Bestimmen Sie mit Hilfe der *Additional-Cost-Matrices* C-optimale Schnittstellen der 3 Sequenzen $S_1 = \texttt{ATA}$, $S_2 = \texttt{TAA}$ und $S_3 = \texttt{AA}$.

Aufgabe 2: DFA

Gegeben folgender Automat:



a) Finden Sie eine Grammatik, die die vom Automaten akzeptierte Sprache erzeugt.

b) Suchen Sie einen regulären Ausruck, der die Sprache repräsentiert

c) Beschreiben Sie die Sprache, die von diesem DFA akzeptiert wird, in eigenen Worten

Aufgabe 3: Schnittpositionen finden für affine Gapkosten

In der Vorlesung wurde gezeigt, wie man mit Hilfe der *Additional-Gap-Matrices* bei linearen Gapkosten die optimalen Schnittpositionen findet. Welche Änderungen muss man vornehmen, um diesen Algorithmus auf affine Gapkosten zu erweitern?

Aufgabe 4: Bonusaufgabe. LU.

Sie haben ein formales System über dem Alphabet $\sigma = \mathrm{L}, \mathrm{I}, \mathrm{U}$. Sie können aus Worten, die im System sind, neue Worte aus dem System erzeugen, indem Sie die folgenden Regeln anwenden:

Regel I Wenn Sie eine Kette besitzen, deren letzter Buchstabe I ist, können Sie am Schluss ein U zufügen (LUUI → LUUIU).

Regel II Angenommen Sie haben Lx Dann können es durch Lxx ersetzen. x steht hier für Zeichenfolgen beliebiger Länge (LUIUU → LUIUUUIUU).

Regel III Wenn in einer der Ketten Ihrer Sammlung III vorkommt, können Sie es durch U ersetzen (LIII → LU).

Regel IV Wenn UU in einer Ihrer Ketten vorkommt, kann man es streichen (UUU → U).

Sie haben zu Beginn nur ein Wort: LI. Wie können sie die folgenden Wörter erzeugen:

a) LUI

b) LL

c) LU

Programmieraufgabe (Abgabe Montag, 24.01.2011, 15:00)

P-Aufgabe 6: Implementieren Sie den Branch-and-Bound Algorithmus zum Finden eines k-Tupels von C-optimalen Schnittpositionen für Sequenzen $s_1, \ldots, s_k$. Die Sequenzen werden Ihrem Programm als k Kommandozeilenparameter übergeben. Ausgegeben werden sollen der Wert des minimalen *multiple additional cost* und ein k-Tupel $c_1, \ldots, c_k$ von Schnittpositionen, an dem dieser Wert angenommen wird.

Beispiel:

```
weese@peking:~$ ./aufgabe6 CT AGT G
6
1,2,1
```

Hinweise:

- Verwenden Sie das Levenshtein-Kostenschema.

- **Achtung:** In dieser Aufgabe werden Kosten minimiert (nicht Scores maximiert), der Needleman-Wunsch (aus P-A2) lässt sich aber mit geringen Änderungen anpassen.
- Eine Schnittposition ist die Länge der Sequenz links vom Schnitt.
- Setzen Sie alle paarweisen Gewichte $w_{i,j}$ auf 1.
- Für eine parallele Iteration über die Schnittposition c_1 gibt es einen **Zusatzpunkt**. Details gibt es im Tutorium.