Abteilung für Genominformatik **Universität Hamburg**
Stefan Kurtz **Zentrum für Bioinformatik**

Genominformatik
Sommersemester 2014
Übungen zur Vorlesung: Ausgabe am 05.05.2014

Punkteverteilung: Aufgabe 3.1: 5 Punkte, Aufgabe 3.2: 2 Punkte, Aufgabe 3.3: 5 Punkte.

Abgabe bis zum 15.5.2014, 23:59 Uhr.

Aufgabe 3.1 In der Vorlesung wurde ein Algorithmus beschrieben, der für eine gegebene RNA-Sequenz die minimale freie Energie liefert, unter der Annahme, dass Basen-Paarungen unabhängig voneinander sind. Dieser Algorithmus wird auch „Nussinov Algorithmus“ genannt. Schreiben Sie in der Programmiersprache Ihrer Wahl ein Programm `nussinov`, das eine RNA-Sequenz von der Kommandozeile liest und darauf den Nussinov-Algorithmus anwendet. Es soll eine RNA-Sekundärstruktur mit minimaler freier Energie (mfe) bestimmt und ausgegeben werden. Als Nebenbedingung soll gelten, dass Hairpin loops mindestens drei ungepaarte Basen enthalten müssen, d.h. $l_{\min} = 3$. Die Basenpaare sollen unterschiedlich bewertet werden: $\alpha(\mathtt{G},\mathtt{C}) = \alpha(\mathtt{C},\mathtt{G}) = -3$, $\alpha(\mathtt{A},\mathtt{U}) = \alpha(\mathtt{U},\mathtt{A}) = -2$, $\alpha(\mathtt{G},\mathtt{U}) = \alpha(\mathtt{U},\mathtt{G}) = -1$ und $\alpha(x,y) = \infty$ sonst. Berechnen Sie dazu die Matrix E und wenden den traceback-Algorithmus, wie in der Vorlesung beschrieben, an. In den Materialien zur Übung finden Sie eine Datei `phenyalanines.txt`, die drei ähnliche Sequenzen enthält, und zwar in den Zeilen, die nicht mit # oder > beginnen. Ihre Programm soll exakt das Ergebnis in der xml-Datei `fold-phenyalanines.xml` reproduzieren.

Aufgabe 3.2 Wir betrachten ein Beispiel für die Anwendung der *Write Only Top Down*-Konstruktion von Suffixbäumen. Geben Sie die einzelnen R-Mengen an, die die Unterbäume des Suffixbaums für den String $S = babaabaaaab\$$ bestimmen. Beachten Sie dabei die Alphabetordnung $a < b < \$$.

Aufgabe 3.3 Implementieren Sie ein Programm `wotd`, das die *Write Only Top Down*-Konstruktion von Suffixbäumen implementiert. Die $-Kante soll immer die letzte Kante sein, die von einem Knoten ausgeht. Alle anderen Kanten sollen alphabetisch entsprechend des ersten Zeichens der Kantenmarkierung sortiert werden.

Beachten Sie, dass es zur Lösung dieser Aufgabe nicht notwendig ist, den konstruierten Baum im Hauptspeicher zu repräsentieren. Es reicht, wenn die R-Mengen und die Gruppen berechnet werden, wobei die konstruierten Knoten und Kanten “on the fly” ausgegeben werden.

Als Argument des Programms soll die Sequenz ohne Sentinel übergeben werden. Den Sentinel müssen Sie selbst an die Sequenz anhängen. Verwenden Sie in Ihrem Programm intern das Zeichen mit dem ASCII-Code 255 als Sentinel und verwenden Sie statt dieses Zeichens das Symbol $ in der Ausgabe. Ihr Programm soll eine der drei folgenden Ausgabeformate haben:

Text Bei diesem Format werden nur die Kantenmarkierungen und Blattmarkierungen zeilenweise ausgeben, und zwar mit führenden Leerzeichen wie folgt: Wenn der Pfad zu einem verzweigenden Knoten i Kanten hat, dann sollen die Markierungen der Kanten mit $2i$ führenden Leerzeichen ausgegeben werden. Bei Blattkanten wird nach der Kantenmarkierung in der gleichen Zeile die Startposition des Suffixes, der zu dem Blatt korrespondiert, ausgegeben. Kantenmarkierung und Blattmarkierung werden durch einen Tabulator getrennt. Die Ausgabe der Kanten erfolgt in alphabetischer Reihenfolge von oben nach unten, siehe Abbildung 1(a).

dot Das dot-Format besteht aus der Liste aller Kanten des Baumes. Für jede Kante wird jeweils der Quellknoten, der Zielknoten und in eckigen Klammern die Kantenmarkierung angegeben. Für einen internen Knoten $\overline{w}$ kann man die Bezeichner w verwenden. Für ein Blatt, das zum Suffix $S[i \ldots n-1]\$$ korrespondiert, kann man i verwenden, Eine Ausgabe in diesem Format kann durch das Programm `dot` aus dem *Graphviz*-Paket (`http://www.graphviz.org`) visualisiert werden, siehe Abbildungen 1(b) und 1(c).

tikz Das *tikz*-Format wird durch das *tikz*-Paket, das Teil von LaTeX ist, unterstützt. Hiermit wurden die Bäume aus dem Vorlesungsskript gezeichnet. Ein Baum besteht aus einem Wurzelknoten, der durch die `\node`-Befehl spezifiziert wird. Dieser wird durch ein Semikolon am Ende abgeschlossen. Weiterhin besteht der Baum aus Kindern, für die jeweils das Schlüsselwort `child` verwendet wird. Alle Teile, die zu einem Kind gehören, werden durch `{` und `}` begrenzt. Ein Kind wird durch das Schlüsselwort `node` beschrieben, optional mit einer Annotation in `{}`-Klammern. Eine Kante wird durch Schlüsselwort `edge from parent` beschrieben, optional mit einer Markierung in `{}`-Klammern nach dem Schlüsselwort `node`. Dieses kann mit Optionen versehen werden, um die Lage der Markierung relativ zur Kante selbst anzugeben, bzw. den Stil des Knotens, entsprechend der Stil-Definitionen in der Präambel des *tikz*-Formates. Ein Beispiel findet man in Abbildungen 1(d) und 1(e).

Die Textdateien für alle drei Formate finden Sie in den Materialien zu dieser Übung. Ausserdem finden sie darin noch entsprechende Dateien für den Suffixbaum der Sequenz $caccaccac\$$.

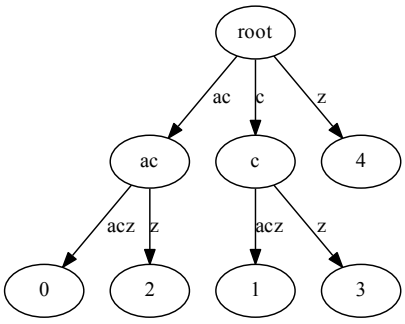
Die Lösungen zu diesen Aufgaben werden am 19.05.2014 besprochen.

```
ac
  ac$ 0
  $ 2
c
  ac$ 1
  $ 3
$ 4
```

(a) Text-Format für $ST(acac\$)$

```
digraph suffixtree_acacz {
  root -> ac [label=ac];
  ac -> 0 [label=acz];
  ac -> 2 [label=z];
  root -> c [label=c];
  c -> 1 [label=acz];
  c -> 3 [label=z];
  root -> 4 [label=z];
}
```

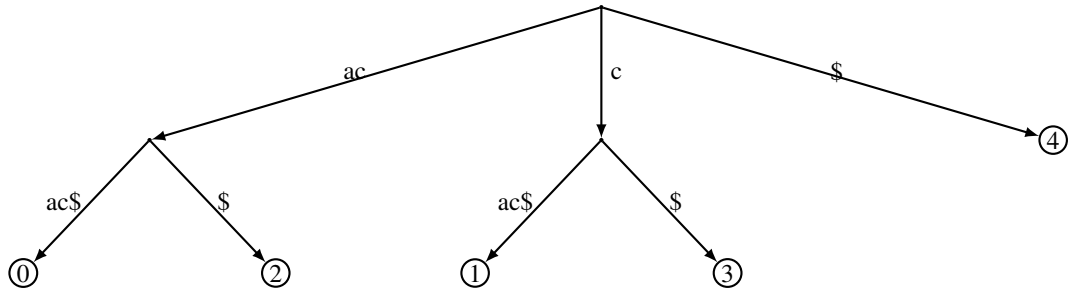
(b) dot-Format für $ST(acac\$)$. Beachten Sie, dass als Sentinel das Zeichen `z` verwendet wird, da $ im dot-Format nicht in einem `label` verwendet werden darf.



(c) Visualisierung des Dotfiles

```
\begin{tikzpicture}[
  >=stealth', thick, level distance=50pt,
  level 1/.style={sibling distance=170pt},
  level 2/.style={sibling distance=95pt},
  level 3/.style={sibling distance=55pt},
  level 4/.style={sibling distance=15pt},
  every node/.style={scale=.8},
  edge from parent/.style={draw,-latex},
  root node/.style={circle,draw,inner sep=0pt},
  leaf node/.style={circle,draw,inner sep=1pt},
  inner node/.style={circle,draw,inner sep=0pt}
]
\node[root node] {}
child {
  node[inner node] {}
  child {
    node[leaf node] {0}
    edge from parent node[left] {ac\$}
  }
  child {
    node[leaf node] {2}
    edge from parent node[right] {\$}
  }
  edge from parent node[left] {ac}
}
child {
  node[inner node] {}
  child {
    node[leaf node] {1}
    edge from parent node[left] {ac\$}
  }
  child {
    node[leaf node] {3}
    edge from parent node[right] {\$}
  }
  edge from parent node[right] {c}
}
child {
  node[leaf node] {4}
  edge from parent node[right] {\$}
}
;
\end{tikzpicture}
```

(d) tikz-Format für $ST(acac\$)$



(e) tikz-Ausgabe für $ST(acac\$)$

Abbildung 1: Visualisierung von Suffixbäumen