ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich
Departement Informatik

Open Class – Sieben Wunder der Informatik

Prof. Dr. Juraj Hromkovič

Übungsaufgaben – Blatt 7

Zürich, 14. Dezember 2005

Zusammenfassung

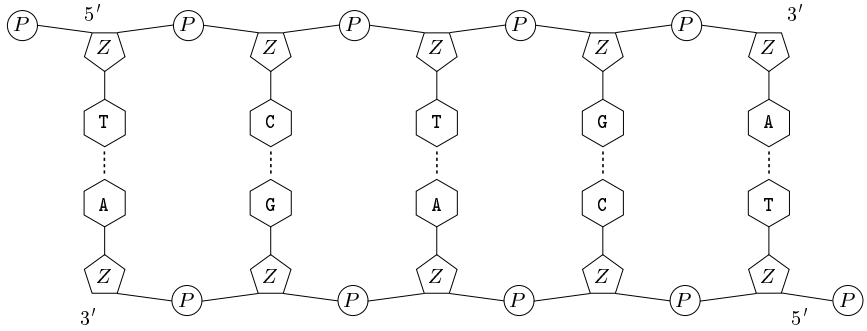
Eine sehr schöne und ausführliche Darstellung finden Sie in dem Buch

> G. Păun, G. Rozenberg, A. Salomaa: *DNA Computing.*
> Springer-Verlag 2005, ISBN 3-540-64196-3, Seiten 1–74.

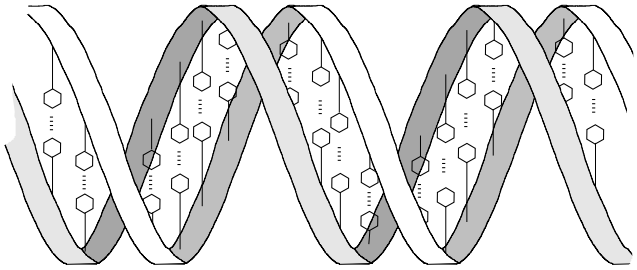
Eine kurze Darstellung ist enthalten in

> J. Hromkovič: *Algorithmics for Hard Problems.*
> Springer-Verlag 2004, ISBN 3-540-44134-4, Seiten 479–485.

DNA-Sequenzen können als Texte dargestellt werden, die sich nur aus den vier Buchstaben `A,C,G` und `T` zusammensetzen. Dabei stehen die Buchstaben für die vier Basen Adenin (`A`), Cytosin (`C`), Guanin (`G`) und Thymin (`T`), aus denen die DNA zusammengesetzt ist. DNA kommt häufig als Doppelstrang-Molekül vor, wobei sich die Basen `A` und `T` sowie `G` und `C` chemisch aneinander binden können.



(a)



(b)

Bildnachweis: H.-J. Böckenhauer, D. Bongartz: *Algorithmische Grundlagen der Bioinformatik*, Teubner 2003, Seite 20

In den vorhergehenden Veranstaltungen haben wir gelernt, dass man Daten (Informationen), wie zum Beispiel Programme, als Texte darstellen kann. In gewissem Sinne macht ein Rechner nichts anderes, als gegebene Eingabetexte in Ausgabetexte umzuwandeln. Man kann die Daten nicht nur durch 0-1-Folgen als Texte darstellen, sondern genauso gut auch als Texte bestehend aus den vier Buchstaben `A`,`C`,`G` und `T`. Die Idee des DNA-Computing ist nun die folgende:

- Daten werden durch DNA-Sequenzen dargestellt.
- Auf diesen DNA-Sequenzen werden chemische Operationen (als Rechneroperationen) ausgeführt.

Wir können beweisen, dass ein solcher DNA-Rechner genau dasselbe tun kann wie ein herkömmlicher digitaler (elektronischer) Rechner. Anders gesagt kann man jede Berechnung eines Rechners durch chemische Operationen auf DNA-Molekülen simulieren. Welches sind aber die chemischen „Operationen auf Reagenzgläsern“, die wir hierfür verwenden können? Wir listen im folgenden einige dieser Operationen auf:

Amplify(T) Die Anzahl der DNA-Sequenzen im Reagenzglas T wird verdoppelt.

Separate(T, w) Verteile den Inhalt von Reagenzglas T so auf zwei Reagenzgläser U und V, dass U alle DNA-Sequenzen enthält, die den Text w als Teiltext enthalten, und dass V die übrigen DNA-Sequenzen enthält.

Length-Separate(T, l) Entferne alle DNA-Sequenzen aus T, die nicht genau l Zeichen lang sind (mittels Gel-Elektrophorese).

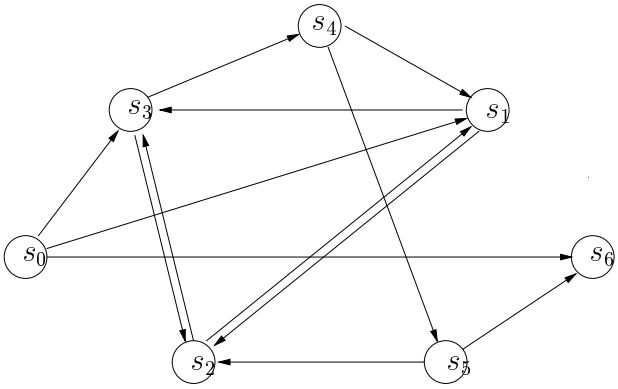
Concatenate(T) Wenn Reagenzglas T eine grosse Menge von DNA-Sequenzen enthält, dann werden zufällig viele davon miteinander verbunden (aneinandergehängt) und so längere DNA-Sequenzen gebildet.

Empty?(T) Teste, ob T leer ist oder mindestens eine DNA-Sequenz beinhaltet.

Separate-Prefix(T, u) Entferne alle Sequenzen aus T, die nicht mit dem Teiltext u beginnen.

Separate-Suffix(T, x) Entferne alle Sequenzen aus T, die nicht mit dem Teiltext x enden.

Der erste DNA-Algorithmus wurde 1994 von L. Adleman für das Problem des *Hamiltonischen Weges* entworfen und auch praktisch durchgeführt. Dieses Problem lässt sich wie folgt beschreiben: Gegeben ist ein Strassennetz zwischen n Städten (bestehend aus lauter Einbahnstrassen; es dürfen zwischen zwei Städten aber auch eine Strasse hin und eine zurück vorhanden sein). Unter diesen n Städten gibt es zwei besondere: START und ZIEL. Die Frage ist nun, ob es einen Weg vom START zum ZIEL in dem Strassennetz gibt, der alle n Städte jeweils genau einmal besucht. Die Antwort (Ausgabe) eines Programms für dieses Problem soll einfach JA oder NEIN sein.
Wir betrachten das folgende Beispiel:



Hierbei sei START $= s_0$ und ZIEL $= s_6$. Der einzige Weg durch dieses Strassennetz von START nach ZIEL, der auch alle anderen Städte genau einmal besucht, ist $s_0 \to s_1 \to s_2 \to s_3 \to s_4 \to s_5 \to s_6$.
Um diesen Weg mit Hilfe eines DNA-Algorithmus zu finden, kodieren wir zunächst die Städtenamen als gleichlange Texte der Länge 20 unter Verwendung der Symbole `A,C,G,T`, beispielsweise

$s_2 =$ `TATCGGATCG GTATATCCGA`;

$s_3 =$ `GCTATTCGAG CTTAAAGCTA`;

$s_4 =$ `GGCTAGGTAC CAGCATGCTT`.

Um nun auch die Strassen durch DNA-Sequenzen zu kodieren, wenden wir die Eigenschaft der DNA an, dass sich die Basen `A` und `T` sowie `C` und `G` ausschliesslich in diesen Paarungen miteinander binden können. Für eine Strasse $e_{i \to j}$ von s_i nach s_j

- spalten wir deren Texte jeweils in der Mitte auf;

- bilden vom zweiten Teil von s_i und vom ersten Teil von s_j das sogenannte Komplement: Wir ersetzen `A` durch `T`, `C` durch `G` und jeweils umgekehrt;
- erzeugen den Text für diese Strasse $e_{i\to j}$ durch Hintereinanderhängen dieser beiden Komplemente.

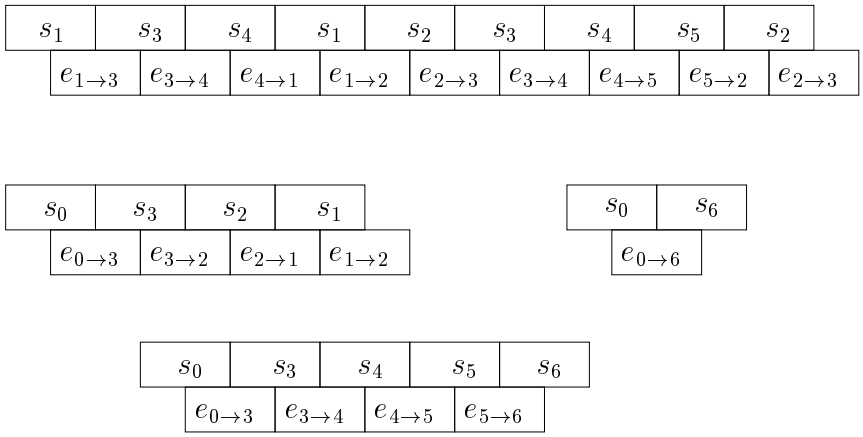
Man beachte, dass damit auch die Richtung der Strasse berücksichtigt wird. In unserem Beispiel bedeutet das

$$e_{2\to 3} = \texttt{CATATAGGCT CGATAAGCTC}$$

$$e_{3\to 2} = \texttt{GAATTTCGAT ATAGCCTAGC}$$

$$e_{3\to 4} = \texttt{GAATTTCGAT CCGATCCATG}$$

Wenn man jetzt die zu diesen Kodierungen passenden DNA-Sequenzen als Einzelstränge unter geeigneten Bedingungen in einem Reagenzglas zusammenbringt, dann können sie sich wie folgt zu Doppelsträngen zusammenlagern:



Jeder derartige Doppelstrang beschreibt damit irgendeinen Weg durch das Strassennetz. Damit das Ganze einwandfrei funktioniert, muss man sicherstellen, dass sich die Strassenkodierungen nur so anlagern können wie in dem Beispiel oben gezeigt. Insbesondere müssen alle Hälften von Städtekodierungen paarweise verschieden sein.
Nachdem wir die Städte und Strassen so kodiert haben, können wir den folgenden Algorithmus anwenden, um den gesuchten Weg zu finden:

1. Gebe DNA-Kodierungen von allen Städten und Strassen (als Einzelstränge) in ein Reagenzglas T.
2. Wiederhole $(2n \cdot \log_2 n)$-mal die Operation Amplify(T), um mindestens n^{2n} Kopien von jedem dieser DNA-Stränge zu erhalten.
3. Erzeuge mit Concatenate(T) eine grosse Menge von doppelsträngigen DNA-Sequenzen, die unterschiedlich lange Wege in dem Strassennetz repräsentieren.

4. Wende die Operation Length-Separate(T, l) an, wobei l die n-fache Länge der Kodierung einer einzelnen Stadt sei. Dann bleiben in T nur die Kodierungen solcher Wege erhalten, die genau n Städte lang sind.

5. Wende Separate-Prefix(T, s_0) an, um nur diejenigen DNA-Sequenzen in T zu behalten, die Kodierungen von Wegen entsprechen, die in $s_0 =$ START beginnen.

6. Wende Separate-Suffix(T, s_n) an, um nur diejenigen DNA-Sequenzen in T zu behalten, die Kodierungen von Wegen entsprechen, die in $s_n =$ ZIEL enden.

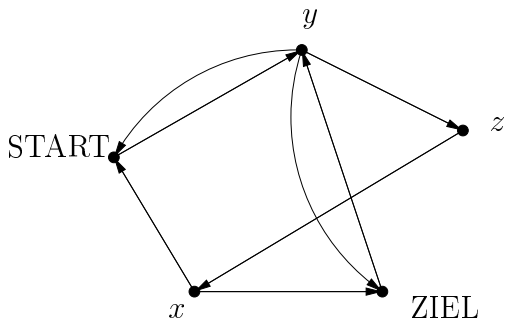
7. Wende $(n-2)$-mal Separate(T, x) an, für alle $n-2$ Kodierungen der restlichen Städte. Damit bleiben nur diejenigen Wege in T übrig, die jede Stadt mindestens einmal enthalten.

8. Untersuche den Inhalt von T mit Empty?(T) und gib die Antwort JA aus, falls noch eine DNA-Sequenz in T enthalten ist, sonst gib die Antwort NEIN aus.

Aufgaben

Aufgabe 19

Betrachten Sie das folgende Strassennetz:



(a) Geben Sie (möglichst kurze) Kodierungen (DNA-Darstellungen) der Städte an, so dass die folgenden Bedingungen erfüllt sind:

1. Jede Kodierung einer Stadt unterscheidet sich an mindestens vier Positionen von jeder Kodierung einer anderen Stadt.
2. Die erste und zweite Hälfte jeder Städte-Kodierung unterscheiden sich von jeder anderen ersten oder zweiten Hälfte einer Städte-Kodierung.

(b) Geben Sie DNA-Kodierungen für die Strassen an, so dass diese zu Ihren Städte-Kodierungen passen.

10 Punkte

Aufgabe 20

Beschreiben Sie detailliert die Vorgehensweise des Adleman-Algorithmus für die Eingabe aus Aufgabe 19, indem Sie eine Folge von Operatioenn angeben, bei denen Sie die Parameter (DNA-Sequenzen und Längenangaben) konkret angeben. **10 Punkte**

Bonus-Aufgabe 7

Reichen die in Aufgabe 19 (a) beschriebenen Anforderungen an die Städte-Kodierungen aus, um zu garantieren, dass jeder gebildete DNA-Doppelstrang einem Weg durch das Strassennetz entspricht? Begründen Sie Ihre Antwort. **10 Bonus-Punkte**

Ihre Lösungen zu den Aufgaben können Sie entweder persönlich bei der Open-Class-Veranstaltung am 11. Januar 2006 abgeben oder bis zum 11. Januar per E-Mail (möglichst als PDF-Datei) an `hjb@inf.ethz.ch` oder per Post an folgende Adresse schicken:

Dr. Hans-Joachim Böckenhauer
Informationstechnologie und Ausbildung
ETH Zentrum CAB F 11.1
Universitätsstrasse 6
8092 Zürich

Bitte vergessen Sie nicht, Ihre Lösung mit Ihrem Namen und Ihrer E-Mail-Adresse zu versehen.
Falls Ihre Lösung uns bis zum 9. Januar 2006 erreicht, können Sie die korrigierte Lösung bereits in der Veranstaltung am 11. Januar abholen (sonst bei der nächsten Veranstaltung).