Algorithmen in der Biologie

Dr. Hans-Joachim Böckenhauer
Dr. Dennis Komm

Zusammenfassung des 3. Abends

Zürich, 7. Mai 2014

1 Wahrscheinlichkeitstheorie

Als Wissenschaftler benutzen wir *Wahrscheinlichkeiten* in zweierlei Hinsicht. Zum einen gibt es den „echten“ Zufall, beispielsweise beim radioaktiven Zerfall. Zum anderen benutzen wir den Zufall als Modell für komplexe Situationen, die wir prinzipiell genau vorhersagen könnten, hierzu praktisch allerdings so viele Parameter kennen müssen, dass dies schlichtweg nicht praktikabel ist.

Nehmen wir beispielsweise den Münzwurf. Würden wir die genaue Beschaffenheit der Münze, die Erdanziehung, die Windrichtung und -intensität, die Wurfgeschwindigkeit etc. ganz genau kennen, könnten wir vorhersagen, ob sie mit „Kopf“ oder „Zahl“ oben liegen bleibt. Allerdings tun wir uns bereits schwer, wenn wir überlegen, welche Parameter wir alle beachten müssten, und so nehmen wir den Ausgang des Münzwurfs als zufällig an.

Ein Zufallsexperiment besteht aus zwei Parametern, einer Menge von so genannten *Ereignissen Erg*, die das Experiment haben kann, und einer *Wahrscheinlichkeitsverteilung Wahr* „auf“ diesen Ereignissen, die ihnen jeweils eine Wahrscheinlichkeit zuordnet. Beim Münzwurf ist beispielsweise

$$Erg = \{\text{Kopf}, \text{Zahl}\}$$

und

$$Wahr(\{\text{Kopf}\}) = \frac{1}{2} \quad \text{und} \quad Wahr(\{\text{Zahl}\}) = \frac{1}{2}.$$

Ein anderes Beispiel wäre das Werfen ein Würfels. Hier gibt es 6 Ausgänge des Experimentes und somit ist

$$Erg = \{\boxdot, \boxdot, \boxdot, \boxdot, \boxdot, \boxdot\}$$

und, wenn der Würfel fair ist, gilt ausserdem

$$Wahr(\{\boxdot\}) = \frac{1}{6}, \quad Wahr(\{\boxdot\}) = \frac{1}{6}, \quad Wahr(\{\boxdot\}) = \frac{1}{6},$$
$$Wahr(\{\boxdot\}) = \frac{1}{6}, \quad Wahr(\{\boxdot\}) = \frac{1}{6} \quad \text{und} \quad Wahr(\{\boxdot\}) = \frac{1}{6}.$$

Wenn wir nun *zusammengesetzte Ereignisse* betrachten, so addieren wir die entsprechenden Wahrscheinlichkeiten. Beispielsweise ist das zusammengesetzte Ereignis, dass eine gerade Zahl geworfen wird, durch die Menge

$$Erg_{\text{gerade}} = \{\boxdot, \boxdot, \boxdot\}$$

gegeben und es gilt

$$Wahr(Erg_{\text{gerade}}) = \frac{1}{6} + \frac{1}{6} + \frac{1}{6} = \frac{1}{2}.$$

Anders sieht es aus, wenn wir uns die Wahrscheinlichkeit ansehen, dass beispielsweise zwei Ereignisse unabhängig voneinander nacheinander eintreffen. Hier werden die Wahrscheinlichkeiten multipliziert. Die Wahrscheinlichkeit, dass beim zweifachen Würfeln erst die 2 und dann die 5 gewürfelt wird, ist

$$Wahr(\{\boxdot\}) \cdot Wahr(\{\boxtimes\}) = \frac{1}{6} \cdot \frac{1}{6} = \frac{1}{36}.$$

Wir interessieren uns im Folgenden für wahrscheinlichste Folgen von Ereignissen, die zu Beobachtungen geführt haben, die wir gemacht haben. Um diese zu berechnen, müssen wir geschickt Wahrscheinlichkeiten von einzelnen Ereignissen multiplizieren.

2 Das unfaire Kasino

Hierzu bleiben wir zunächst beim Würfeln und betrachten einführend das folgende Beispiel. Nehmen wir an, Sie sind in einem Kasino zu Gast, bei dem Sie ein Würfelspiel beobachten. Eine Person hat hierbei einen Würfel und würfelt wiederholt. Das Besondere ist, dass sie den Würfel von Zeit zu Zeit wechselt. Ein Würfel W_1 der beiden ist fair, es gilt also

$$\begin{aligned} Wahr(W_1 \text{ zeigt } \boxdot) &= Wahr(W_1 \text{ zeigt } \boxdot) = Wahr(W_1 \text{ zeigt } \boxdot) = Wahr(W_1 \text{ zeigt } \boxdot) \\ &= Wahr(W_1 \text{ zeigt } \boxtimes) = Wahr(W_1 \text{ zeigt } \boxdot) = \frac{1}{6}. \end{aligned}$$

Der zweite Würfel W_2 ist jedoch gezinkt und es gilt

$$Wahr(W_2 \text{ zeigt } \boxdot) = \frac{1}{2}$$

und

$$\begin{aligned} Wahr(W_2 \text{ zeigt } \boxdot) &= Wahr(W_2 \text{ zeigt } \boxdot) = Wahr(W_2 \text{ zeigt } \boxdot) \\ &= Wahr(W_2 \text{ zeigt } \boxdot) = Wahr(W_2 \text{ zeigt } \boxtimes) = \frac{1}{10}, \end{aligned}$$

eine 6 wird also in der Hälfte aller Fälle gewürfelt. Ohne dass Sie es merken, wird der Würfel ausgewechselt, und zwar jeweils mit einer Wahrscheinlichkeit von

$$Wahr(\text{Wechsel}) = \frac{1}{20}.$$

Zu Beginn wird entweder W_1 oder W_2 mit einer Wahrscheinlichkeit von jeweils $1/2$ genommen. Wir können die Situation nun also wie in Abbildung 1 darstellen. Wir sehen, dass wir eine Struktur erhalten haben, die sehr ähnlich zu den endlichen Automaten ist, die wir am ersten Abend kennengelernt haben. Die Zustände W_1 und W_2 entsprechen den Situationen, dass der faire Würfel W_1 oder der unfaire Würfel W_2 benutzt werden. Ausserdem haben wir noch einen Startzustand q_0, der der Situation entspricht, dass noch kein Würfel ausgewählt wurde. Die Kanten sind mit den Wahrscheinlichkeiten beschriftet, mit denen wir von einem Zustand in einen anderen „übergehen." Deswegen werden diese Wahrscheinlichkeiten auch *Übergangswahrscheinlichkeiten* genannt. Ferner heisst ein endlicher Automat, dessen Kantenbeschriftungen Wahrscheinlichkeiten entsprechen, *Markov-Modell*. In den Zuständen finden

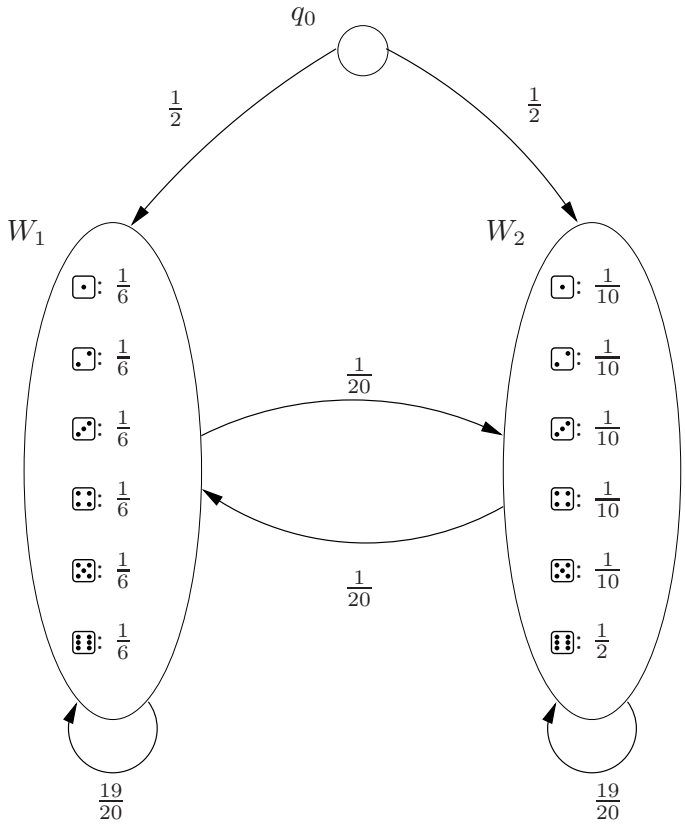


Abbildung 1. Ein Markov-Modell für das unfaire Kasino

sich zudem noch die *Emissionswahrscheinlichkeiten*, das heisst die Wahrscheinlichkeiten, im entsprechenden Zustand ein gegebenes Ereignis zu beobachten.

Wie bereits angedeutet, interessieren wir uns nun für Folgen von Ereignissen. Angenommen, wir beobachten

⚀, ⚅, ⚅, ⚄, ⚅, ⚁, ⚃

und nehmen an, dass zunächst der gezinkte Würfel die ersten fünf Mal benutzt wurde und dann der faire. Wie wahrscheinlich ist dies? Die Wahrscheinlichkeit, dass zu Beginn der unfaire Würfel W_2 genommen wurde, ist $1/2$. Anschliessend wurde die 1 mit einer Wahrscheinlichkeit von $1/10$ gewürfelt, da wir ja den unfairen Würfel W_1 benutzen. Dass wir danach wieder W_2 verwenden, passiert mit einer Wahrscheinlichkeit von $19/20$ (was sich sofort daraus ergibt, dass mit einer Wahrscheinlichkeit von $1/20$ gewechselt wird). Danach wird die 6 mit einer Wahrscheinlichkeit von $1/2$ gewürfelt. Somit kommen wir für die ersten zwei Würfe insgesamt auf eine Wahrscheinlichkeit von

$$\mathit{Wahr}(W_2 \text{ wird gewählt}) \cdot \mathit{Wahr}(W_2 \text{ zeigt } ⚀) \cdot \mathit{Wahr}(\text{kein Wechsel}) \cdot \mathit{Wahr}(W_2 \text{ zeigt } ⚅)$$

und somit

$$\frac{1}{2} \cdot \frac{1}{10} \cdot \frac{19}{20} \cdot \frac{1}{2},$$

indem wir alle diese Wahrscheinlichkeiten multiplizieren. Führen wir dieses Vorgehen fort, berechnet sich die Gesamtwahrscheinlichkeit zu

$$\frac{1}{2} \cdot \frac{1}{10} \cdot \frac{19}{20} \cdot \frac{1}{2} \cdot \frac{19}{20} \cdot \frac{1}{2} \cdot \frac{19}{20} \cdot \frac{1}{10} \cdot \frac{19}{20} \cdot \frac{1}{2} \cdot \frac{1}{20} \cdot \frac{1}{6} \cdot \frac{19}{20} \cdot \frac{1}{6},$$

was insgesamt zu einer Wahrscheinlichkeit von

$$\frac{2\,476\,099}{3\,686\,400\,000\,000}$$

führt.

3 Ein biologisches Experiment

Im Labor beobachten wir eine Bakterien-Kultur in einer Petrischale, die eine Nährlösung enthält. Die Bakterien sind genetisch so verändert, dass von einem *Marker-Protein* eine grün fluoreszierende Variante (GFP, *Green Fluorescent Protein*) produziert wird. Dieses Marker-Protein wird mit grosser Wahrscheinlichkeit hergestellt, wenn ein Bakterium gesund ist. Wir setzen die Nährlösung sich verändernden äusseren Einflüssen aus (zum Beispiel einem Temperaturanstieg) und notieren, was wir hinsichtlich der Lichtintensität der GFPs in aufeinander folgenden Zeitschritten beobachten können. Wir können für dieses Experiment nun folgendes Markov-Modell erstellen.

- Die Bakterien sind entweder *krank*, *OK* oder *gesund*. Hieraus ergeben sich drei entsprechende Zustände.
- Die Emissionen sind *hoch*, *mittel* oder *niedrig* und entsprechen der beobachteten Lichtintensität in einem Zustand. Im Zustand *Krank* ist die Wahrscheinlichkeit (also die Emissionswahrscheinlichkeit) beispielsweise grösser, eine hohe Lichtintensität zu beobachten als im Zustand *Gesund*.
- Für jeden der drei Zustände gibt es eine zuvor experimentell bestimmte Wahrscheinlichkeit, in einen anderen Zustand überzugehen. Wenn ein Bakterium beispielsweise zu einem Zeitschritt im Zustand *Krank* ist, dann ist es mit einer Wahrscheinlichkeit von 3/5 auch noch im nächsten Zeitschritt krank. Mit einer Wahrscheinlichkeit von 2/5 ist es danach hingegen im Zustand *OK*.

Während unseres Experimentes beoachten wir nun beispielsweise in aufeinanderfolgenden Zeitschritten die Lichtintensitäten

hoch, *mittel*, *mittel*, *hoch*, *hoch*, *niedrig*

und alles, was wir wissen, ist, dass das entsprechende Bakterium einen „Weg“ durch unser Markov-Modell genommen hat. Allerdings wissen wir nicht, welchen. Was wir ausrechnen wollen, ist ein wahrscheinlichster Weg. Für den Informatiker ergibt sich also das folgende Problem.

Eingabe: Ein Markov-Modell und eine Reihe von Beobachtungen. **Ausgabe:** Ein wahrscheinlichster Weg durch das gegebene Markov-Modell, der zu den gemachten Beobachtungen führte.

In einer solchen Situation sprechen wir auch von einem *Hidden-Markov-Modell* (HMM). Wir bewegen uns nämlich „durch“ ein Markov-Modell, wissen jedoch nicht, in welchem Zustand wir zu einem gegebenen Zeitpunkt sind. Das Problem, einen wahrscheinlichsten Weg durch ein HMM zu berechnen, lässt sich effizient mit der letzte Woche vorgestellten Methode der dynamischen Programmierung lösen. Der Algorithmus wird nach seinem Erfinder *Viterbi*-Algorithmus genannt. Eine genauere Beschreibung ist auf den Vortragsfolien dargestellt.

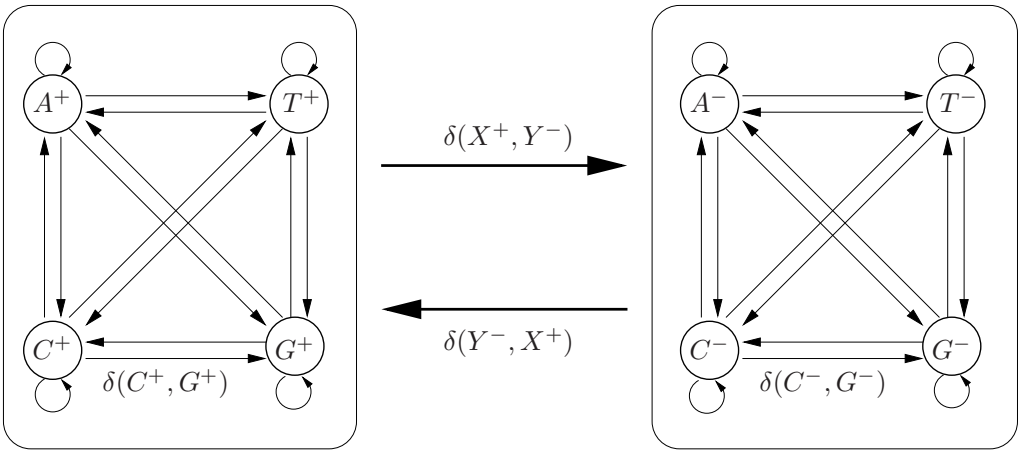


Abbildung 2. Ein HMM zum Auffinden der *CG*-Inseln in der DNA

4 Suche nach kodierenden Bereichen in der DNA

Eine weitere Anwendung der HMM ist die Suche nach kodierenden Bereichen (*Genen*) in der DNA. Dieser Ansatz basiert auf der folgenden Beobachtung: In der DNA-Sequenz kommt die Buchstabenabfolge *CG* im Allgemeinen viel seltener vor als dies in einem zufälligen String der Fall wäre. Es gibt jedoch Bereiche der DNA, in denen die Buchstabenfolge *CG* viel häufiger vorkommt, diese Bereiche nennt man *CG-Inseln* (oder auch *CpG-Inseln*). Solche *CG*-Inseln treten in der Regel in der Nähe von kodierenden Bereichen der DNA auf, deshalb ist es von Interesse, sie zu lokalisieren.

Wir können nun ein HMM entwerfen, mit dem es uns gelingt, die Lage der *CG*-Inseln vorherzusagen. Im Prinzip ist dies eine ähnliche Aufgabe wie die Vorhersage der Zeitpunkte, zu denen im unfaire Kasino der gezinkte Würfel benutzt wurde. Die Zahl 6 auf dem gezinkten Würfel entspricht hier dem Teilstring *CG*. Weil wir hier aber kein einzelnes Zeichen mit abweichender Häufigkeit suchen, sondern einen Teilstring der Länge 2, wählen wir eine etwas andere Modellierung.

Wir verwenden acht Zustände $A^+, C^+, G^+, T^+, A^-, C^-, G^-, T^-$, in denen aber jeweils mit Wahrscheinlichkeit 1 nur ein bestimmtes Zeichen ausgegeben wird, nämlich das Zeichen X in den Zuständen X^+ und X^-. Die mit „+“ markierten Zustände modellieren hierbei die *CG*-Inseln, die mit „−“ markierten Zustände die anderen Bereiche der DNA. Die abweichenden Häufigkeiten des Teilstrings *CG* modellieren wir hier durch verschiedene Übergangswahrscheinlichkeiten zwischen den Zuständen. Dadurch ergibt sich das in Abbildung 2 gezeigte HMM.

Wir müssen nun nur noch erklären, wie wir die Übergangswahrscheinlichkeiten in unserem Modell wählen. Wir bezeichnen im Folgenden die Übergangswahrscheinlichkeit von einem Zustand X in einen Zustand Y mit $\delta(X, Y)$. Dann muss in unserem Modell gelten, dass $\delta(C^+, G^+) \gg \delta(C^-, G^-)$. Die genauen Werte können empirisch bestimmt werden, entsprechend der Häufigkeit des Teilstrings *CG* innerhalb und ausserhalb der *CG*-Inseln. Wir setzen hier als Wert innerhalb einer *CG*-Insel $\delta(C^+, G^+) = 0.3$, damit ergibt sich $\delta(C^+, A^+) = \delta(C^+, C^+) = \delta(C^+, T^+) \approx 0.23$. Ausserhalb einer *CG*-Insel setzen wir $\delta(C^-, G^-) = 0.04$ und damit $\delta(C^-, A^-) = \delta(C^-, C^-) = \delta(C^-, T^-) \approx 0.32$. Diese Werte entsprechen ungefähr der realen Situation in einem Säugetiergenom. Für alle anderen Übergänge nehmen wir an, dass der Folgebuchstabe jeweils mit gleicher Wahrscheinlichkeit auftritt. Natürlich ist dies eine vereinfachte Modellierung, auch hier

könnte man die Übergänge zwischen den einzelnen Buchstaben noch verschieden gewichten, abhängig von empirisch ermittelten Werten.

Die Übergangswahrscheinlichkeit von einem Zustand X^+ der CG-Insel zu einem Zustand Y^- ausserhalb und umgekehrt hängt von der durchschnittlichen Länge einer CG-Insel und dem durchschnittlichen Abstand von zwei benachbarten CG-Inseln in der DNA-Sequenz ab. Die typische Länge einer CG-Insel liegt zwischen 500 und 2000, wir nehmen im Folgenden einen Wert von 1000 an. Damit ist dann ein Wert $\delta(X^+, Y^-) = 1/4000 = 0.00025$ sinnvoll: die CG-Insel wird mit einer Wahrscheinlichkeit von 1/1000 beendet, diese Abbruchwahrscheinlichkeit verteilt sich dann gleichmässig auf die vier möglichen Folgebuchstaben. Für die umgekehrte Richtung haben wir angenommen, dass $\delta(Y^-, X^+) = 1/100000$, dies entspricht einem durchschnittlichen Abstand von 25000 zwischen zwei benachbarten CG-Inseln.

Wie in den vorangegangenen Beispielen können wir nun den Viterbi-Algorithmus nutzen, um einen wahrscheinlichsten Pfad in dem HMM zu berechnen, der uns dann eine Vorhersage liefert, wo die CG-Inseln in der gegebenen DNA-Sequenz liegen, nämlich dort, wo der Pfad durch die mit „+“ markierten Zustände führt. Für den Start wählen wir entsprechend dem ersten Zeichen X der Sequenz mit gleicher Wahrscheinlichkeit den Zustand X^+ oder X^-.

5 Ein Rechner aus DNA

Bisher haben wir untersucht, wie man die Methoden der Informatik verwenden kann, um biologische Fragestellungen zu lösen. Wir wollen nun kurz die umgekehrte Richtung anschauen: Kann man die Methoden der Molekularbiologie auch nutzen, um damit Berechnungsprobleme zu lösen?

Wir wissen, 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 Rechnens mit DNA (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.

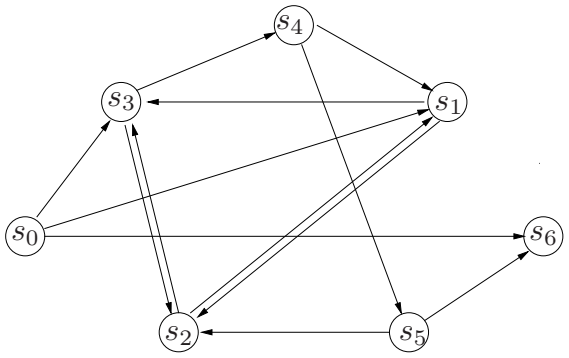
Empty?(T) Teste, ob T leer ist oder mindestens eine DNA-Sequenz beinhaltet.

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.

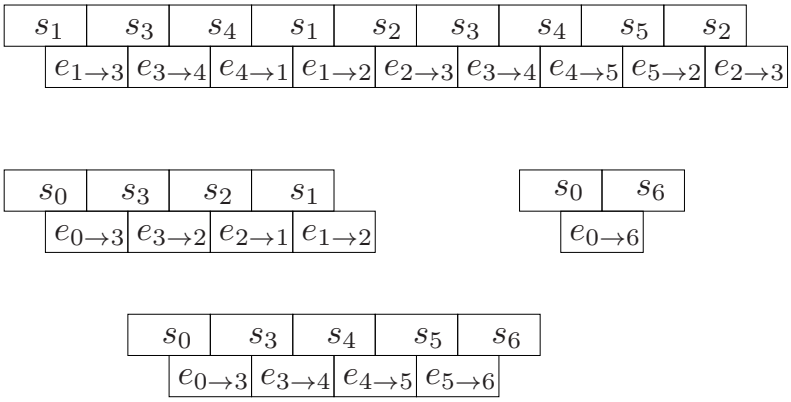
Man beachte, dass damit auch die Richtung der Strasse berücksichtigt wird. In unserem Beispiel bedeutet das

$e_{2\to 3}$ = `CATATAGGCT CGATAAGCTC`;

$e_{3\to 2}$ = `GAATTTCGAT ATAGCCTAGC`;

$e_{3\to 4}$ = `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.

6 Weiterführende Literatur

[1] Eine Einführung in die Hidden-Markov-Modelle und Anwendung bei CG-Inseln findet sich in Kapitel 9.4 des Buchs

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

[2] Eine sehr schöne und ausführliche Darstellung des DNA-Computing finden Sie in dem Buch

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

[3] Eine kurze Darstellung ist enthalten in

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