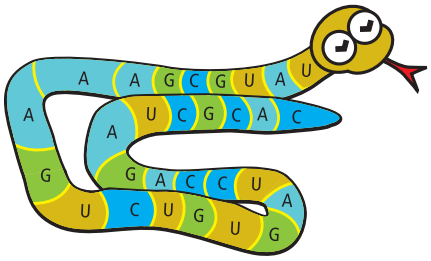
Beispielaufgabe Ribo-Natter

Die Ribo-Natter (Kosename: RNA) ist eine lange Schlange mit vielen farbigen Ringeln, die in folgenden Farben vorkommen: Aquamarin (kurz: A), Umbra (U), Grasgrün (G) und Cyan (C).
Diese Schlangenart ist ganz besonders eitel und möchte dem Betrachter ein möglichst gefälliges Aussehen präsentieren. Daher legt sie sich ganz flach so auf den Boden, dass – ihrer Meinung nach – eine farblich besonders harmonische Schlängelung entsteht. Dabei gibt es vier Punkte zu beachten:

- Sie möchte dem Betrachter gerne viele Harmoniestellen zeigen. Das sind Stellen, an denen A mit U bzw. C mit G in Berührung kommt.
- Ein Ringel darf zu höchstens einer Harmoniestelle gehören.
- Damit die Schlängelung nicht zu geknickt aussieht, müssen zwischen zwei Ringeln, die miteinander eine Harmoniestelle bilden, mindestens drei Ringel sein, die zu keiner Harmoniestelle gehören.
- Besonders schön sind Harmonieabschnitte, d.h. mehrere direkt aufeinander folgende Harmoniestellen.



Für diese Aufgabe nehmen wir an, dass die Ringel einer Schlange fortlaufend nummeriert sind, und zwar beginnend mit 1 vom Kopf her. Eine Harmoniestelle entspricht dann einem Nummernpaar, bei der oben abgebildeten Schlange z.B. (3,28); ein Harmonieabschnitt kann durch ein Paar von Nummernbereichen beschrieben werden: (3-7,28-24). Eine komplette Schlängelung wird durch die entstandenen Harmonieabschnitte beschrieben; die Schlängelung aus der Abbildung lautet also komplett: (3-7,28-24);(12-14,22-20)

Aufgabe

1. Schreibe ein Programm, das eine Schlange als Folge von Farbringeln einliest und ihren längsten Harmonieabschnitt findet. Demonstriere das Programm an drei Beispielen. Eines dieser Beispiele soll die folgende Schlange sein:
GGGAGCGUAGCUCAGUGCGGAGAGCGCCUGCUUUGCAC
GCAGGAGGUCUGCGGUUCGAUCCCGCGCGCUCCCACCA
2. Eine Schlängelung mit möglichst vielen Harmoniestellen wäre der Ribo-Natter am liebsten. Beschreibe, wie eine Schlange vorgehen könnte, um die maximale (Farb-)Harmonie zu finden.

Lösungsidee

In dieser Aufgabe wird das molekularbiologische Phänomen der RNA-Faltung in vereinfachter Form behandelt. Die RNA wird zur Schlange, die Basenbindungen werden zu Harmoniestellen. Als Harmonieabschnitte werden dann Blöcke von Harmoniestellen bezeichnet.

Die Schlängelungsregeln der Aufgabenstellung lassen zwei Optionen offen, die bei einer RNA-Faltung aber nicht vorkommen: spiralförmige Schlängelungen und Schlängelungen, bei denen mehr als zwei Stränge parallel liegen. Beide Varianten werden nicht berücksichtigt. Im Aufgabentext ist auch noch vorgegeben, dass sich die Schlange ganz flach auf den Boden legt. Das Problem ist also zweidimensional, Schlangenstücke liegen nicht übereinander.

Teilaufgabe 1: Finde den längsten Harmonieabschnitt

Bei der Bestimmung des längsten Harmonieabschnitts genügt es, von einem Knick der Schlange auszugehen. Durch mehr Knicke entstehen sicher keine längeren Harmonieabschnitte.

Naheliegend ist nun z. B., für jede mögliche Abschnittlänge alle Schlangenstücke dieser Länge mit allen möglichen (umgedrehten!) Partnerstücken auf Harmonie zu testen.

Wenn man mit der größten Länge anfängt (halbe Schlangenlänge, ggf. abgerundet, minus 3), kann man aufhören, sobald ein Segment komplett mit einem anderen harmoniert.

Anstatt bei jedem Test das Partnerstück umzudrehen, sollte man gleich eine komplette Kopie der Schlange umdrehen und auch noch die Ringelfarben gemäß der Harmoniezuordnung invertieren. Dann kann man Original und Kopie jeweils einfach von vorne durchlaufen, der Harmonietest vereinfacht sich zum Test auf Gleichheit. Dieses Vorgehen wird im unten beschriebenen Programm implementiert.

Teilaufgabe 2: Schlängelungen mit maximal vielen Harmoniestellen

Die Ermittlung der maximalen Harmonie, also der größtmöglichen Anzahl an Harmoniestellen, die eine Schlange bilden kann, ist schon komplizierter. Es handelt sich um ein Optimierungsproblem, und da gibt es wie immer die Möglichkeit des „brute force"-Ansatzes. D.h., die Schlange müsste alle Möglichkeiten der Schlängelung ausprobieren und sich die beste merken. Zwei Punkte sind im Unterschied zu Teilaufgabe 1 dabei wichtig: Mehrere Knicke können zu besseren Ergebnissen führen als einer, und die Länge der Harmonieabschnitte spielt keine Rolle; u.a. müssen auch einseitig einmal Ringel ausgelassen werden.

Um alle Schlängelungen auszuprobieren, kann man rekursiv vorgehen. Die ganze Schlange wird an allen möglichen Stellen in zwei Teile geteilt, die Berechnung der maximalen Harmonie dann für die Teile auf die gleiche Weise durchgeführt. Das beste Ergebnis unter den verschiedenen Teilungen wird genommen. Um vernünftige Ergebnisse zu bekommen, muss die Berechnung für kleine Stücke ohne weitere Teilung erfolgen. Nachteil dieser Methode ist, dass der gesuchte Wert für ein und dasselbe Teilstück ziemlich häufig neu berechnet werden muss. Es ist deshalb sinnvoll, einmal erfolgreich berechnete Werte in eine Tabelle einzutragen und bei einem rekursiven Aufruf erst einmal die Tabelle zu prüfen. Oder man geht andersherum vor, fängt mit den kurzen Stücken an und setzt die Ergebnisse für längere Stücke aus denen für die kürzeren zusammen. So kommt man ohne Rekursion aus.

Programm-Dokumentation

Eine Ribo-Natter kann sehr gut als Zeichenkette implementiert werden. Zur Bearbeitung von Zeichenketten eignet sich Perl sehr gut, so dass das Lösungsprogramm in Perl geschrieben wurde.

Das Perl-Programm ist kurz und besteht aus fünf Abschnitten: (1) Einlesen der Schlange aus einer Datei, (2) Erzeugen der gedrehten und invertierten Kopie, (3) Initialisierung einiger Hilfsvariablen, (4) Bestimmung des längsten Harmonieabschnitts und (5) Ausgabe des Ergebnisses. Teil 4 bearbeitet das eigentliche Problem und wird näher erläutert.

In der äußersten Schleife wird die Originalschlange von vorne durchlaufen. Die Zählvariable $i bestimmt die Anfangsposition eines zu einem möglichen Harmonieabschnitt gehörenden Teilstücks. Diese Anfangsposition kann maximal 4 Positionen vor dem Ende der Schlange liegen ($laenge-4), damit Regel 3 (Abstand zwischen zwei Ringeln einer Harmoniestelle) nicht verletzt wird. Um ein passendes Teilstück zu finden, wird die gedrehte und invertierte Kopie der Schlange genau wie das Original von vorne durchsucht. Die entsprechende Zählvariable der inneren Schleife $j wird zusätzlich durch die äußere Zählvariable begrenzt (je weiter hinten ein Harmonieabschnitt beginnt, desto weniger Ringel bleiben als mögliche Partner übrig).

Für jedes Paar von Anfangspositionen $i und $j in Original und Kopie wird geprüft, ob sich damit ein längerer Harmonieabschnitt als bisher bekannt bilden lässt. Dabei kann ein Harmonieabschnitt höchstens halb so lang werden wie die durch $i von vorne und $j von hinten begrenzte Schlange (2*$halaenge < $laenge-$i-$j-2;). Die beiden Teilstücke werden mittels der substr-Funktion gebildet; wenn sie übereinstimmen, gibt es eine neue (Zwischen-)Lösung und die entsprechenden Variablen werden aktualisiert.

Programm-Ablaufprotokolle

Hier drei Aufrufe des Programms. Die in der Eingabedatei enthaltene Schlange und der längste Harmonieabschnitt werden ausgegeben. bsp1.dat enthält das Pflichtbeispiel.

```
> perl ribo-natter.pl bsp1.dat
Schlange: GGGAGCGUAGCUCAGUGCGGAGAGCGC
CUGCUUUGCACGCAGGAGGUCUGCGGUUCGAUCCC
GCGCGCUCCCACCA
Laengster Harmonieabschnitt: (1-7,72-66)
>
> perl ribo-natter.pl bsp2.dat
Schlange: UUUUUGGGGGAACCAACAGUUGGUUCC
CCCAAAAA
Laengster Harmonieabschnitt: (1-16,35-20)
>
> perl ribo-natter.pl bsp3.dat
Schlange: AAAUAAACCCGCCC
Laengster Harmonieabschnitt: Keine
Harmoniestellen!
>
```

Programm-Text

```
#!/usr/bin/perl

# BWINF 20.1: Ribo-Natter

# Schlange aus übergebener Datei einlesen
open DATEI, „<$ARGV[0]“;
$schlange = <DATEI>;
close DATEI;

# Kopie in umgekehrter Reihenfolge erzeugen
$invschlange = reverse $schlange;
# in der Kopie A mit U und G mit C vertauschen
$invschlange =~ tr/ACGU/UGCA/;

# Länge der Schlange
$laenge = length $schlange;
# Länge des größten gefundenen Harmonieabschnitts
$maxha = 0;
# Loesungsspeicher, passend initialisiert
$loesung = „Keine Harmoniestellen!“;

# Originalschlange bis Ringel $laenge-4 durchlaufen
for ($i=0; $i<$laenge-4; $i++) {
  # Schlangenkopie bis Ringel $laenge-$i-4 durchlaufen
  for ($j=0; $j<$laenge-4-$i; $j++) {
    # nach größerer Übereinstimmung suchen
    for ($halaenge=$maxha+1;
         2*$halaenge < $laenge-$i-$j-2;
         $halaenge++) {
      # Vergleichsstück der Ausgangsschlange
      $m1 = substr($schlange, $i, $halaenge);
      # Vergleichsstück der Schlangenkopie
      $m2 = substr($invschlange, $j, $halaenge);
      # wenn beide eine Harmoniestelle bilden...
      if ($m1 eq $m2) {
        # ...neue Maximallänge festsetzen
        $maxha = $halaenge;
        # ...und Harmonieabschnittsbeschreibung bilden
        $loesung = ‘(‘.($i+1).’-’.($i+$halaenge).’,’;
        $loesung .= ($laenge-$j).’-’.($laenge-$j-
$halaenge+1).“)“;
      }
      # ansonsten...
      else {
            # ...Vergleich der Stellen $i und $j beenden
        last;
      }
    }
  } # Ende for-Schleife Schlangenkopie
} # Ende for-Schleife Originalschlange

# Schlange ausgeben
print „Schlange: $schlange\n“;
# Längsten Harmonieabschnitt ausgeben
print „Laengster Harmonieabschnitt“: $loesung\n";
```

Lösung nach Alexander Hullmann und Markus Völker