Abschlussprojekt: Bestimmung der Sekundärstuktur einer optimal gefalteten RNA Sequenz

Die Rolle der RNA wird oft als Botenstoff betrachtet (sogenannte mRNA). Hierbei werden Teile der DNA zu den Ribosomen transportiert, um dort die Produktion von Proteinen anzuleiten. Weiterhin spielt aber RNA auch eine wichtige Rolle als Katalysator in einer Zelle, ähnlich einem Enzym. Man unterscheidet dabei nach ihrer Funktion folgende RNA-Typen:

transfer RNA (tRNA) Diese RNA's haben typischerweise eine Länge von 85 Basen. Sie spielen eine wichtige Rolle bei dem Prozess der Translation. Die Gestalt der tRNA erlaubt, dass sich eine bestimmte Aminosäure an die Seite der tRNA bindet und transporiert wird. Es gibt also für jede der 20 Aminosäuren eine zugehörige Form der tRNA, meistens sogar mehrere, nämlich mindestens eine für jedes der 61 Codons (ausser dem Stopcodon). Auf der anderen Seite bindet die tRNA während der Translation im Ribosom mit dem entsprechenden Codon mit Hilfe eines 3-Basen langen Anticodons. Somit wird die richtige Aminosäure an das enstehende Protein angehängt.

ribosomale RNA (rRNA) Das Ribosom selbst besteht aus mehreren RNA-Molekühlen (rRNAs) und Proteinen. Bei den Säugetieren sind das nach dem Gewicht gleich viele. Die darin vorkommenden RNAs heißen 18S, 28S, 5.8S und 5S. Die rRNA ist am Aufbau und der enzymatischen Aktivität des Ribosoms beteiligt. Während der Translation bindet sich das Ribosom gleichzeitig an eine mRNA und zwei tRNAs.

small nuclear RNA (snRNA) Unter diesem Begriff subsummiert man üblicherweise die sechs RNAs die bei dem Splicing-Prozess involviert sind. Man spricht dabei auch vom *spliceosome*, als der Gesamtheit der beteiligten RNAs und Proteine. Die sechs menschlichen snRNAs heißen U1-U6.

signal recognition particle RNA (SRP RNA) Das *signal recognition particle* ist ein Komplex aus sechs Proteinen und einem RNA Molekül aus ca. 600 Basen. Dieser Komplex erkennt und transportiert spezifische Proteine zu dem oder durch das endoplasmatischen Rediculum.

Bei seiner Funktion als Katalysator spielt bei der RNA deren Struktur, also deren Faltung eine wichtige Rolle. Dabei ist zu beachten, dass die RNA nur aus einem Strang besteht, und sich Wasserstoffbrücken zu Nukleotiden auf dem selben Strang herausbilden können. Diese Wasserstoffbrücken legen zum großen Teil die 3-dimensionale Struktur der RNA fest und deren chemische Eigenschaften.

Bei der Beschreibung der Struktur unterscheidet man zwischen:

Primärstruktur Das ist die lineare Abfolge der in der RNA vorkommenden Nukleotide, üblicherweise von $5'$ nach $3'$ gelesen.

Sekundärstruktur Diese beschreibt die Menge der Nukeotide die nicht direkt benachbart sind und Wasserstoffbrücken in der RNA besitzen. Neben den "Watson-Crick Paaren"

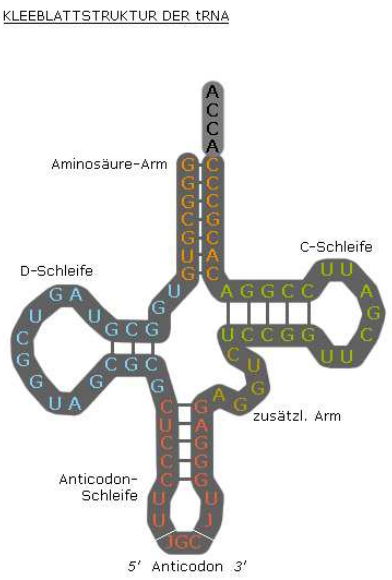
AU und GC können auch Guanin und Uracil eine einfache Wasserstoffbrücke bilden (sogenannte *wobble pairs*).

Tertiärstruktur Diese beschreibt die genaue Position eines jeden Atoms der RNA im dreidimensionalen Raum.

Quartärstruktur Diese beschreibt die relative Position der RNA im 3-dimensionalen Raum, innerhalb eines Komplexes der aus mehren Proteinen und RNAs besteht.

Die Tertiärstruktur sagt am meisten über die chemischen Eigenschaften der RNA aus. Bei kleinen RNAs kann sie mit Hilfe der Röntgenstrukturanalyse bestimmt werden. Bei großen RNAs funktioniert das nicht mehr.

Wenn man die Tertiärstruktur nicht bestimmen kann, so gibt einem die Sekundärstruktur wertvolle Hinweise. Darum ist es für die Biologie interessant die Sekundärstruktur von RNA Molekülen zu bestimmen. Dabei ist zu beachten, dass auch die RNA möglichst energetisch niedrige, also stabile Zustände annimmt. Die Ausbildung von Wasserstoffbrücken fördert diese Stabilität. Also ist es gerechtfertigt nach der Sekundärstruktur mit der maximalen Anzahl von Wasserstoffbrücken zu suchen.



Ziel dieses Projektes ist es, ein Programm zu erstellen, welches für eine gegebene DNA Sequenz die daraus entstehende mRNA ermittelt, dessen optimale Sekudärstruktur berechnet und diese graphisch abbildet. Für den graphischen Teil ist der Programmcode in der Datei `GUI.py` bereits vorhanden (mit der zusätzlichen Datei `drawing.py`). Die Funktionen für die Berechnungen der Sekundärstruktur müssen von Ihnen in der Datei `Biologic.py` vervollständigt werden. Die GUI kann über die Konsole mit

```
python GUI.py
```

aus dem Ordner, in dem sich diese Datei befindet, gestartet werden. Die Datei `Biologic.py` und die Datei `drawing.py` sollten sich hierfür im gleichen Ordner befinden.

Aufgabe 1 In der Datei `Biologic.py` findet sich die Klasse `biologic`. Ergänzen Sie diese um die Methoden:

- `set_dna(self, dna)`: Es soll eine DNA-Sequenz, gegeben als String, intern als Instanzvariable gespeichert werden.
- `validate_dna(self)`: Es soll die intern gespeicherte DNA-Sequenz überprüft werden, ob es eine gültige DNA-Sequenz ist. Das Ergebnis ist ein Wahrheitswert. Diese Methode wird bereits bei der DNA-Eingabe verwendet.
- `get_rna(self)`: Es soll die RNA-Sequenz als String zurück gegeben werden, die intern in einem Objekt der Klasse gespeichert ist.

Aufgabe 2 Es sollte nun möglich sein eine DNA-Sequenz (Menüeintrag 'DNA' -> 'Enter sequence') manuell einzugeben und diese sollte angezeigt werden. Implementieren Sie nun zwei weitere Funktionen:

- `read_fasta_file(self, name)`: Diese Methode soll eine DNA-Sequenz aus einer FASTA-Datei name einlesen. Die GUI verwendet dazu ein Dialogfenster, Sie bekommen also einen vollqualifizierten Namen und müssen sich nicht um den Pfad weiter kümmern. Diese Methode hat kein Ergebnis, weist aber der internen DNA-Sequenz einen Wert zu.
- `read_from_ncbi(self, id, mail)`: Diese Methode liest eine DNA-Sequenz von der NCBI Nukleotid-Datenbank. Als Parameter werden die Accession-Number (id) und die zu verwendende email-Adresse (mail) verwendet. Auch hier wird die interne DNA-Sequenz gesetzt und mit dem Resultat von der NCBI-Anfrage belegt.

Sie können die Funktionen über die beiden anderen Untermenüs im Menü 'DNA' testen.

Aufgabe 3 Im nächsten Schritt wollen wir die Transkription der DNA in RNA durchführen.

Bei den Doppelsträngen der DNA unterscheidet man den codogenen Strang (auch Watson-Strang) und den codierenden Strang (auch Crick-Strang). Dabei ist der eine Strang jeweils das reverse Komplement zu dem anderen Strang.

Der codogene Strang wird bei der Transkription für den Aufbau der RNA benutzt. Die Basensequenz der RNA gleicht also der Sequenz des codierenden Strangs, nur dass jedes Vorkommen von Thymin (T) durch Uracil (U) ersetzt wird.

Wir gehen nun davon aus, dass die DNA-Sequenz als codogener Strang vorliegt. Schreiben Sie eine Funktion `transcript(self)` die für eine DNA-Sequenz, die als codogener Strang angenommen wird, die RNA Sequenz berechnet, die sich aus der Transkription ergibt. Diese Methode soll als Ergebnis die translatierte RNA zurückgeben, aber sich diese RNA auch intern speichern

Für die Implementierung sollen Sie keine zusätzlichen Bibliotheken (wie z.B. Biopython) verwenden. Sie können die Methode testen, wenn Sie im Menü 'RNA' den Unterpunkt 'Transcribe' auswählen

Aufgabe 4 Als nächstes soll die RNA-Sequenz in eine Aminosäurensequenz translatiert werden.

Für die Translation sollen Sie den genetischen Code aus der Datei `genCode.txt` einlesen. Dort ist der Code zeilenweise abgelegt. In jeder Zeile steht am Anfang die Aminosäure. Dahinter folgen die codierende Codons, durch Leerzeichen getrennt. Die Translation des Stop-Codons wird mit '_' dargestellt.

Lesen Sie die Datei zeilenweise ein und erzeugen sich ein entsprechendes Dictionary. Mit diesem eingelesenen Dictionary sollen Sie dann die Translation implementieren.

Die Methode `translate(self,initPos)` soll die intern gespeicherte RNA ab der gegebenen Startposition `initPos` in eine Aminosäurensequenz gemäß dem genetischen Code übersetzen. Als Ergebnis soll diese Aminosäurensequenz zurückgegeben werden.

Auch hier sollen Sie die Funktionalität selbst implementieren, ohne Verwendung von BioPython. Sie können Ihr Programm testen, indem Sie im Menü 'Protein' den Unterpunkt 'Translate' verwenden.

Aufgabe 5 Im nächsten Schritt sollen die codierende Sequenzen in der translatierten Form, also Proteine, gefunden werden.

Für die Berechnung möglicher Proteine sollen alle 6 möglichen Leseraster (reading frames) anwendbar sein: Leseraster 0 beginnt in der RNA bei Position 0, Leseraster 1 und 2 bei Position 1 und 2, Leseraster 3 beginnt bei der revertierten RNA bei Position 0, 4 und 5 bei der revertierten RNA bei 1 und 2.

Implementieren Sie eine Methode `get_proteins(self, rf_number)`, die für die gespeicherte RNA alle möglichen Proteine zu dem durch `rf_number` gegebenem Leseraster berechnet. Die Translation eines Proteins beginnt mit dem Startcodon, also der Aminosäure 'M' und endet bei einem Stopcodon, hier mit '_' übersetzt. Die Methode soll die gefundenen Proteine als Liste von Strings zurückgeben.

Sie können die Methode testen mit dem Untermenüpunkt 'Extract Protein' im Menü 'Protein' (Voraussetzung: transkribierte RNA wurde erzeugt).

Aufgabe 6 Implementieren Sie eine Funktion zur Auswertung einer gegebenen Sekundärstruktur für eine RNA.

Die zweidimensionale Struktur einer RNA läßt sich als eine Folge von ausgeglichenen Klammern und Punkten darstellen. Hierbei wird eine Basenpaarung durch ein Klammerpaar dargestellt. Zu beachten ist, dass sich Basenpaarungen nicht überlappen können. Also z.B. die Folge

((...)..(((....)).))..

beschreibt eine Sekundärstruktur für die RNA

AGAAGCCCCUUCCCCCGGUAUUU

In dieser Aufgabe sollen Sie die Methode `eval_rna(self, struct)` entwickeln, welche für die intern gespeicherte RNA-Sequenz und einer dazu gegebenen Sekundärstrukur `struct` (in oben dargestellter Schreibweise) die Anzahl der Wasserstoffbrücken zählt.

Implementieren Sie dazu die Funktion `get_bonds()`. Diese Funktion soll ein Dictionary liefern, welches die Anzahl der Wasserstoffbrücken für eine Paar von Nukleotiden liefert. Dabei bilden die Watson-Crick Paare mit 'C' und 'G' jeweils 3 Wasserstoffbrücken, mit 'A' und 'U' jeweils zwei und die *wobble pairs* eine Wasserstoffbrücke.

Für die Implementierung der Methode `eval_rna(self, struct)` gilt dann: die Basen die an der Position einer öffnenden Klammer und an der Position der dazugehörigen schließenden Klammer zu finden sind, bilden ein Basenpaar. Für dieses Basenpaar kann die Anzahl der gebildeten Wasserstoffbrücken aus dem `Dictionary` entnommen werden. Die Methode soll die Anzahl der gesamten Wasserstoffbrücken als Wert zurückgeben.

Sie können diese Methode testen im Untermenüpunkt 'Evaluate Structure' im Menü 'RNA' (Voraussetzung: transkribierte RNA wurde erzeugt).

Aufgabe 7 Jetzt geht es darum die Sekundärstruktur zu finden, welche die Anzahl der Wasserstoffbrücken maximiert, unter Beachtung folgender Einschränkungen:

- **Primary Proximity Constraint**: Eine RNA kann sich nicht so stark falten, dass eine Base eine Bindung mit einer ihrer sechs Nachbarn eingeht.

  Wenn (i, j) *eine Verbindung eingehen, dann gilt:* $|i - j| > 3$

- **Nesting Constraint**: Sogenannte *Pseudoknoten* sind verboten.

  Wenn (i, j) *und* (p, q) *eine Verbindung eingehen mit* $i < p < j$ *und* $p < q$*, dann ist* $q < j$*.*

Nehmen wir dazu als Beispiel die RNA-Sequenz UUUCAGUAGCA. Bezüglich der Faltung gibt es zwei Möglichkeiten: Entweder bilden das U an der ersten Stelle und das A an der letzten Stelle eine Wasserstoffbrücke, oder nicht. Im ersten Fall finden wir die dafür beste Sekundärstruktur, in dem wir zuerst die beste Sekundärstruktur für die Sequenz UUCAGUAGC berechnen und dann die Paarung (U,A) addieren. Also:

$(U, A) +$ (beste Faltung für $UUCAGUAGC$)

Im zweiten Fall können wir die Sequenz an einer beliebigen Stelle teilen. Wir finden dann die beste Sekundärstruktur in dem wir die beiden optimalen Teilstrukturen miteinander kombinieren. Dabei gibt es 10 Möglichkeiten

(beste Faltung für U) + (beste Faltung für $UUCAGUUAGCA$)
(beste Faltung für UU) + (beste Faltung für $UCAGUUAGCA$)
(beste Faltung für UUU) + (beste Faltung für $CAGUUAGCA$)
(beste Faltung für $UUUC$) + (beste Faltung für $AGUUAGCA$)
(beste Faltung für $UUUCA$) + (beste Faltung für $GUUAGCA$)
(beste Faltung für $UUUCAG$) + (beste Faltung für $UUAGCA$)
(beste Faltung für $UUUCAGU$) + (beste Faltung für $UAGCA$)
(beste Faltung für $UUUCAGUU$) + (beste Faltung für $AGCA$)
(beste Faltung für $UUUCAGUUA$) + (beste Faltung für GCA)
(beste Faltung für $UUUCAGUUAG$) + (beste Faltung für CA)
(beste Faltung für $UUUCAGUUAGC$) + (beste Faltung für A)

Daraus ergibt sich als Strategie, dass man zuerst die beste Faltung für kürzere Sequenzen berechnet, diese sich zwischenspeichert und dann mit deren Hilfe die optimale Faltung für längere Sequenzen herleitet. Diese Art von Vorgehen nennt man dynamisches Programmieren.

Für unser spezielles Problem verwenden wir eine quadratische Matrix (der Länge entsprechend der Länge der RNA), die wir mit den Ergebnissen von Teilproblemen füllen, also mit der jeweils maximal möglichen Anzahl von Wasserstoffbrücken. Ein Eintrag in der Matrix in Zeile i und Spalte j speichert die optimale Anzahl für die Teilsequenz von Base i bis Base j. Da Wasserstoffbrücken erst ab Sequenzen der Länge 5 möglich sind (siehe obige Einschränkungen), füllen wir die Matrix nach und nach für alle Teilsequenzen aufsteigend der Länge nach, ab Länge 5. Am Ende können wir das optimale Ergebnis für die gesamte Sequenz aus der Matrix ablesen.

Implementieren Sie die Methode foldRna(self) wie folgt:

- Initialisieren Sie eine entsprechende quadratische Matrix mit den Werten 0.
- Schreiben Sie eine verschachtelte Schleife. In der äußeren Schleife iterieren Sie über die Länge der Teilsequenz (ausgehend von 5). In der inneren Schleife iterieren Sie über die möglichen Startpositionen der aktuellen Teilsequenzlänge (beginnend bei 0). Dieser Schleifenaufbau ermöglicht uns, die Matrix diagonal zu durchlaufen.
- Berechnen Sie die Endposition der Teilsequenz mit Hilfe der Startposition und der Länge.
- Berechnen Sie die Anzahl der Wasserstoffbrücken, die bei einer Verbindung der Basen an der End- und der Startposition enstehen.
- Nun verwenden wir die Fallunterscheidung von oben um die maximale Anzahl der Wasserstoffbrücken für die aktuelle Teilsequenz zu bestimmen. Berechnen Sie hierzu den ersten Fall und speichern sie das Ergebnis in der Matrix im entsprechenden Eintrag ab. (Hinweis: die Anzahl der Waserstoffbrücken der inneren Teilsequenz ist bereits in der Matrix gespeichert.)
- Nun muss getestet werden, ob der zweite Fall (Aufteilung in zwei beliebige Teilsequenzen) zu einer höheren Anzahl an Wasserstoffbrücken als der bereits berechnete erste Fall führt. Vergleichen Sie die Summe der möglichen Teilsequenzpaare unter Beachtung des Primary Proximity Constraint mit der bereits berechneten Anzahl der Wasserstoffbrücken für den ersten Fall. Speichern Sie das Maximum in der Matrix im entsprechenden Eintrag ab.(Hinweis: das erfordert eine weitere Schleife.)

Diese Methode soll für die aktuell gespeicherte RNA die maximal mögliche Anzahl von Wasserstoffbrücken berechnen und als Ergebnis zurückgeben. Die Methode kann mit dem Untermenüpunkt 'Optimal Structure' im Menü 'RNA' ausgeführt werden.

Aufgabe 8 Nun soll aus der Faltung von der letzten Aufgabe die optimale Struktur abgeleitet werden,

Bisher haben wir nur die maximale Anzahl der Wasserstoffbrückenbindungen berechnet. Jetzt wollen wir die (bzw. eine) optimale Struktur daraus ableiten. Dazu können wir rekursiv die Einträge in unserer Matrix zurückverfolgen. Dies soll in der Methode trace(self) umgesetzt werden

- Verwenden Sie eine rekursive Hilfsfunktion.
- Beginnen Sie die Rekursion mit dem Eintrag in der Matrix von der letzten Aufgabe (folding) an der Stelle $(0, LaengeRNA - 1)$. Hierbei ist wiederum die aktuell gespeicherte RNA gemeint.
- Unterscheiden Sie folgende Fälle für einen Eintrag an der Stelle (i, j):
  - Ist der Eintrag $= 0$, so gibt es zwischen i und j keine Wasserstoffbrücken. Wir geben also entsprechend viele Punkte '.' aus, für jede Base zwischen i und j jeweils einen.
  - Ist der Eintrag gleich dem Eintrag an der Stelle $(i + 1, j)$, so ist die Base an der Stelle i keine Bindung eingegangen. Wir erzeugen einen '.' und hängen das Ergebnis der Zurückverfolgung für die Sequenz $(i + 1, j)$ an.
  - Ist der Eintrag gleich dem Eintrag an der Stelle $(i + 1, j - 1)$ plus der Anzahl der Wasserstoffbrücken, die bei der Paarung von i-ter und j-ter Base entstehen, so geben wir eine öffnende Klammer '(' aus, gefolgt von dem Ergebnis der Zurückverfolgung für die Sequenz $(i+1, j-1)$, gefolgt von einer schliessenden Klammer ')'.
  - Im letzten Fall suchen wir, ob es ein k zwischen i und j gibt, so dass sich der Eintrag an der Stelle (i, j) als Summe der Einträge an den Positionen (i, k) und $(k + 1, j)$ ergibt. D.h. die i-te Base bindet mit der k-ten Base. In diesem Fall konkatenieren wir das Ergebnis der Rückverfolgung von der Sequenz (i, k) mit dem Ergebnis der Rückverfolgung der Sequenz $(k + 1, j)$.

Die Methode `trace(self)` soll die Sekundärstruktur in Klammerschreibweise als `String` zurückgeben. Sie kann über den gleichen Menüeintrag wie die letzte Aufgabe getestet werden (eventuell müssen Sie in der Anzeige zur richtigen Stelle scrollen).