Übungen zur Vorlesung Sequenzanalyse

Universität Bielefeld, WS 2025/26
Prof. Dr. Jens Stoye
Leonard Bohnenkämper
Tutorien: Lennart Finke, Sofie Jans

`https://gi.cebitec.uni-bielefeld.de/teaching/2025winter/sa1`

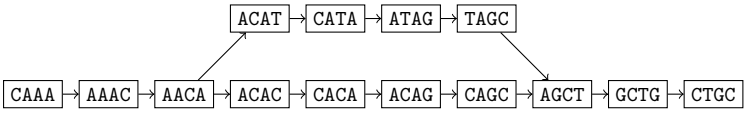
Übungsblatt 7 vom 11.12.2025
Abgabe bis 18.12.2025 bis 9:30 Uhr per Mail an den Tutor/die Tutorin

Aufgabe 1 (Genomassemblierung - Overlap Layout Consensus) (6 Punkte)
Berechne ein Assembly folgender "Reads" mithilfe der Overlap-Layout-Consensus Strategie. Du kannst davon ausgehen, dass sie vom gleichen Strang sequenziert wurden (d.h. du musst nur Vorwärts-Matches beachten). Verwende die Scores +1 für Match, −1 für Mismatch/Gap und einen Mindestwert für den Score von 6.

$$r_1 = \texttt{CACAGCTGC}, r_2 = \texttt{CAAACACAG}, r_3 = \texttt{ATCACAGCT}.$$

Aufgabe 2 (Genomassemblierung - De Bruijn Graph) (10 Punkte)
In dieser Aufgabe implementieren wir einen De Bruijn Graphen im Übungsrepository[1] und machen ein paar initiale Schritte in Richtung Genomassemblierung. Den De Bruijn Graphen definieren wir für diese Aufgabe wie folgt: Für ein Menge an Sequenzen, sind die Knoten des Graphen als die Substrings der Länge k in den Sequenzen. Eine Kante (u, v) verbindet zwei Knoten u und v, wenn die $k-1$ letzten Zeichen von u gleich der $k-1$ ersten Zeichen von v sind.

1. Implementiere die De-Bruijn-Graph Klasse `DBG` in `dbg.py`. Bearbeite dazu die folgenden Punkte:
   (a) `add_kmer`: Füge ein k-mer dem Graphen hinzu.
   (b) `add_sequence`: Füge alle k-mere einer Sequenz dem Graphen hinzu.
   (c) `build_dbg`: Aus einer Liste von Fasta-Sequenzen, konstruiere den entsprechenden De Bruijn Graphen.
   (d) `right_neighbors`/`left_neighbors`: Gib eine Liste der rechten/linken Nachbarn eines Knoten aus. Überlege dir, warum du dazu die Kanten nicht zusätzlich zu den k-meren speichern musst.
2. Binde die Funktionen `build_dbg`, `is_resolved` und `get_resolved_string` in `main_assemble.py` ein.
   (a) Welche der Beispielsequenzen im Unterverzeichnis `data/` kannst du damit schon erfolgreich assemblieren? Welchen Wert für k benötigst du dazu jeweils? Probiere dazu verschiedene Werte für k aus.
   (b) Warum sollte man für echte Readdaten immer k kleiner der Readlänge wählen?
   (c) Welchen Tradeoff beobachtest du für größeres oder kleineres k?
3. Der De Bruijn Graph für $k = 4$ der Beispieldatei `bubble.fasta` sieht wie folgt aus:



   Wie könnte diese Struktur entstanden sein? Beziehe die Readsequenzen in der Analyse mit ein.

Aufgabe 3 (N50) (4 Punkte)
Informiere dich darüber, was der N50-Wert eines Assemblies ist. Gib eine kurze Definition. Schreibe ein Skript (`main_n50.py`), das diesen für eine gegebene multiple Fasta-Datei berechnet. Du kannst dazu die fasta-Musterlösung (`w08/fasta.py`) im Übungsrepository nutzen.

[1] `https://gitlab.ub.uni-bielefeld.de/lbohnenkaemper/sqa-ex/-/tree/main/w08`