Instruction of package for DNA sequence analysis

Deoxyribonucleic acid (**DNA**) is a molecule that carries the genetic instructions used in the growth, development, functioning and reproduction of all known living organisms and many viruses. DNA strand is polynucleotide, which is composed of simpler monomer units called nucleotides. Each nucleotide contains one of four nucleobases — adenine (A), thymine (T), guanine (G), or cytosine (C). Most DNA molecules consist of two biopolymer strands, which are bound together by base pairing (A with T, and C with G), and the two strands are reverse complement to each other. (https://en.wikipedia.org/wiki/DNA)

**<Class Sequence>**

This class represents the sequence of a DNA molecule. It reads a string composed of ‘A’, ‘T’, ‘G’, ‘C’, and ‘N’ (‘N’ represent any of the four nucleotides) and converts it into a list of DNA sequence, where the sequence can be represented as and convert between lower case and upper case. It support basic manipulations of DNA sequence, including insertion, deletion, mutation, reversion of the nucleotide order, and change its nucleotide to the complementary one (A and T, C and G). Moreover, DNA sequence can be printed to console in a given format and be drawn in a simple graph as a black line for visualization.

In bioinformatics, a **sequence alignment** is a way of arranging the sequences of DNA, RNA, or protein to identify regions of similarity that may be a consequence of functional, structural, or evolutionary relationships between the sequences.

**<Class Alignment>**

This class supports sequence alignment between two Sequence objects. It adopts a dynamic programming algorithm called Needleman-Wunsch algorithm. In the first step, it builds a 2-by-2 grid, and filled each cell with lowest penalty scores. In the second step, it traces back from the last cell to the first cell according to the scores. After alignment is done, the show() function allow user to visualize the result, where gaps in each sequence is filled by ‘~’, a match between two sequence is connected by ‘|’ and a mismatch is marked by ‘#’. In this class, the penalty scores for a match, mismatch, internal gap and external gap have been pre-assigned. However, a more sophisticated system may allow user to define these scores by themselves.

(https://en.wikipedia.org/wiki/Needleman\_Wunsch\_algorithm)

The **genetic code** is the set of rules by which information encoded within genetic material (DNA or mRNA sequences) is translated into proteins by living cells. Each codon consists of three DNA bases and encodes one out of 20 of protein creating amino acid. (https://en.wikipedia.org/wiki/Genetic\_code)

**<Class CodonTable>**

This class stores amino acids information and their genetic codes. Each amino acid has a name, a letter code and a molecular weight (kD). It supports functions to translate a DNA sequence to protein peptide sequence and show the result in the console.

A **restriction enzyme (cutter)** is an enzyme that cleaves DNA sequence at specific recognition sites within the molecule known as restriction sites. (https://en.wikipedia.org/wiki/Restriction\_enzyme)

**<Class Cutter>**

This class represents a cutter that recognizes a specific site. It enables search of a cutter against any DNA sequence using a string-matching algorithm termed Knuth–Morris–Pratt algorithm. The KMP algorithm searches for occurrences of a "word" (restriction sequence) within a main "text string" (DNA sequence) by employing the observation that when a mismatch occurs, the word itself embodies sufficient information to determine where the next match could begin, thus bypassing re-examination of previously matched characters. In the first step, it constructs a deterministic finite automaton (dfa) from restriction sequence (pattern). In the second step, it applies the dfa to search the pattern in a DNA sequence. (https://en.wikipedia.org/wiki/Knuth–Morris–Pratt\_algorithm)

**<Class CutterTable>**

This class takes a list of cutters and stores in a set. It supports search of all cutters in the table against a DNA sequence and shows the results in the console.

**<Class CuttingMap>**

This class takes a DNA sequence and a cutter table. It supports search of all cutters in the table against the DNA sequence and draw the results in a simple graph, which takes mouse inputs to select/show and unselect/hide a given cutter in the graph.

**<Class SeqAnalysis>**

This class takes the names of files as command-line arguments to create Sequence, CodonTable, CutterTable and CuttingMap objects for comprehensive sequence analysis.