Python project 2023

What is expected from you :

- A pyhton file `identify.py` with the requested functions. *Do not forget to fill in the documentation, use docstrings.*
- one double-sided page document (at maximum) containing your comments and observations on your functions. Some hints :
  - give toy examples to show on which cases your identifier finds the original set of proteins
  - how close is your solution to the *human_blood* protein set, knowing that this set should come from this tissue ?
  - may your solutions be used on real world cases ?
  - can your identifier deal with all the biological cases : incertainity, errors, duplication, repeated peptides : these are ideas ideas to improve your solution.
- ***your work is due for the 1st of December before midnight on Moodle***

Some notions of protein identification

Thanks to new Mass Spectometry (MS) technologies it is now possible to *read* the proteins present in a sample, quickly and cheaply. MS produces a large number of small segments of the proteins called *peptides*.

A peptide is therefore a string of Amino Acids (AA) (i.e. of characters among the 26 AA). Each peptide is identified by a unique identifier. The set of peptides produced by a MS machine is stored in a *FASTA* file (.fa or .fasta).

Here is an extract of such fasta file :

```
>nxp:NX_P01308-1
MALWMRLLPLLALLALWGPDPAAAFVNQHLCGSH
```

From the set of peptides produced by the MS machine when analyzing a sample, one could reconstruct the original set of proteins by doing what is called an **identification**. Identifying means selecting the proteins from a database in order to reconstruct the original set of proteins. The identification is done based on the overlaps between the peptides and the proteins in the database. For more details look at wikipedia:protein_identification.

The Data

Sample peptide fasta files are files containing small peptides from the MS machine: **small_peptides.fa** It is taken from peptide Atlas DB where many peptide fasta are available.

A *reference protein database file* is a fasta file that propose a complete list of possible proteins per tissue : : **human_blood_proteome.fa** The list is established from a list of genes expressed in that tissue, for example the *human peripheral blood* available in Protein Atlas. From this list of genes it is possible to create the protein database with the UniprotDB mapping tool.

Challenges for Protein Identification

There are several challenges to identify proteins from small peptides.

First, some peptides might not correspond to any protein. That is, they are not present in any of the proteins in the reference.

Second, a peptide can be present *in several protein sequences* of the database and it is not possible to associate that peptide with a unique protein.

Finally, it is possible that a protein sequence matches *randomly* with a peptide (that is, by chance), so *a single match* is not enough to identify a protein.

The basic approach.

In a first approach, *the peptides with several matches and the single-match peptides are discarded.*

The retained set of proteins are the proteins that have **at least two peptides** with a **unique** hit each.

You will write a script `identify.py`, which can be used to perform the identification of a *minimal set of proteins* from a set of *peptides* like those contained in the file `small_peptides.fa` and a protein sequences database like `human_blood_proteome.fa`.

Note that for simplicity sakes in this project we don't consider mismatch, and there aren't any identical peptides. Filter such identical peptides if they exist.

Your protein identification program in python

You will write the `identify.py` file that will contain (at least) the following functions:

- `read_fasta (fasta_file)` that loads a fasta file and returns the dictionary containing the sequences, peptides and proteins sequences in our case;
- `match_id(r,fasta_file)` that loads a protein fasta file and returns the ids of the proteins (possibly several) that match the peptide `r`, *or returns None otherwise*
- `unique_match_set(peptide_file, db_file)` that loads a peptide file (fasta) and a protein db file (fasta) and returns the protein sequences (possibly several) *that uniquely match at least two peptides,* or returns None otherwise*

One step further

To avoid discarding many peptides that have multiple matches, you will try to *assemble* peptides if they have sufficient *overlap* (above a threshold that you fix).

You will add at least the following functions :

- `overlap(r1, r2)` that, for 2 sequences `r1` and `r2`, will return the length of the maximal overlap between them, meaning the longest suffix of `r1` exactly matching with a prefix of `r2`. For example if :
  - `r1 = ATRYTY` and `r2 = YTYAAT`, the maximal overlap is 3 (`YTY`, though there is also an overlap of 1, `Y`)
  - `r1 = ATRYTP` and `r2 = YTPAAT`, the maximal overlap is 3 (`YTP`)
  - `r1 = ATRYTP` and `r2 = YTYAAT`, there is no overlap, so the maximal overlap is 0.
- `assembly_peptides(peptide_file, overlap_min)` that, for a set of peptides and a minimal overlap, produces a novel fasta file containing assemblies of peptides.

To compute all the possible assemblies of peptides, you will implement a **greedy solution** that, for each peptide, tries to **extend** it to the right and/or to the left with the peptide that overlaps the best. You will continue to extend till you will not find any other petide overlapping.

A peptide cannot be used more than once in an assembly, however a peptide may be used in several assemblies.