Assignment 02

Programming with Python (for Bioinformatics)

Johanna Schmitz

08.05.2024; Submit before **21.05.2024, 23:59**

Assignment 02

Template

Update your git fork to get the template file (`assignments_02.py`) and tests (`test_assignment_02.py`).
For some tasks, you will have to read input files. For the tests, we provide the input files in the git repository, but we will test your code after the submission also on different input.
You have until **21.05.2024, 23:59** to submit your assignment in the CMS.

Assignment 01 - Task 1 (2 points)

Anagrams

A word w is an anagram of a word w‘, if w’ can be formed by rearranging the letters of w. For example, `'study'` is an anagram of `'dusty'`.

Task 1 : Group words by their anagram

Write a function `anagrams(words)` that gets a list of words as input and returns the largest subset of words that have the same anagrams (equal words should be counted only once). You may assume that the maximum is unique.
For example, `anagrams(['no', 'peels', 'sleep', 'on', 'leeps', 'fruit'])` should return `{'peels', 'sleep', 'leeps'}`, because the groups `{no, on}` and `{fruit}` contain less elements.

Hint

Store all words in a `dict` using the lexicographic smallest anagram of each word as the key.

Assignment 02 - Task 2 (3 points)

Hamming Distance

The hamming distance is defined as the number of positions where two strings of the same length differ (or equivalently, the minimum number of substitutions to convert one string into the other).

Task 2 : Hamming distance between strings

Write a function `hamming_distance` that gets a filename as an argument and returns the minimum and maximum hamming distance between all pairs of strings. The input file contains in each line a DNA or RNA sequence (upper or lower case letters, no spaces allowed), e.g.,

```
ACTGACT
actggtt
ACUGCUU
```

Assignment 02 - Task 2 (continued)

Task 2 : Hamming distance between strings (continued)

Before computing the hamming distance, convert RNA sequences to DNA sequences (U to T conversion). Then compute the case insensitive hamming distance between all pairs of strings.
Return the minimum and maximum hamming distance (here (1, 2)).

Invalid input

If the input is not valid, return (-1,-1).
For example, if not all lines have the same number of characters or if the strings are not DNA or RNA, the input is not valid.

Assignment 02 - Task 3 (2 points)

FASTA files

- **FASTA** file format is an ubiquitous format for storing biological sequences.
- FASTA files are plain text files. The contents are as follows:

```
> description_1
biological_sequence_1
biological_sequence_1_continued
> description_2
biological_sequence_2
...
```

- There are **header lines** (starting with >) and **sequence lines**.
- Each header starts a new sequence.
- Sequences may continue over several lines. These lines have to be concatenated (without interruption) to form the complete sequence.

Assignment 02 - Task 3 (continued)

Task 3 : Reading FASTA files

Write a function `common_kmers(fasta, k)` that first extracts all sequences from the fasta file, computes for each sequence its k-mers and then returns the set of all common k-mers. A k-mer is **common** if it occurs in **all** of the sequences (hint: solve using set intersection).

Hint

Store k-mers in a **set** data structure.
The subtask of getting all k-mers is very similar to Task 3 of assignment 01.

Assignment 02 - Task 4 (3 points)

Vampire numbers

Vampire numbers are composite natural numbers with an even number of digits n that can be divided into two numbers with $n/2$ digits each, called fangs, such that the product of both fangs is equal to the original number.
The concatenation of both fangs must contain the same digits as the original number, but an arbitrary order of the digits is allowed, with the exception that not both fangs can have trailing zeros.

Task 4 - Find vampire numbers

a. Write a function, `is_vampire(n)` that returns whether the number n is a vampire number or not.

b. Write a generator function `vampire_generator()`, that generates the infinite sequence of vampire numbers.

Assignment 02 - Task 4 (continued)

Examples

- 1260 is a vampire number, since $21 \times 60 = 1260$
- 125460 is a vampire number with two different pairs of fangs 204×615 and 246×510
- the first few vampire numbers are: $1260, 1395, 1435, 1530, 1827, 2187, \ldots$

Hint

- You can find more information on vampire numbers at en.wikipedia.org/wiki/Vampire_number
- Have a look at the **itertools** python package.