CSCI3100: Software Engineering

Assignment 4

April 21, 2025

```
Due date: 2 May 2025 Total marks: 70
```

Revision	Date	Description
1.1.0	30 Apr	Fix bug in Q4: the command should be 'pytest …'. My apologies, and thanks.
1.0.0	21 Apr	Initial release

1 Questions

A DNA sequence can be viewed as a string of characters: T, G, A C. In DNA identification, a target DNA sequence is compared to a collection of confirmed DNA sequences to identify the one it most closely matches. One heuristic approach to evaluating similarity is to measure the length of the longest common substring shared between two DNA sequences — the longer the substring, the more similar the two DNA sequences are deemed to be. Since DNA sequences are long, efficient algorithms for string analysis must be available.

Please answer the following questions with respect to the source code in the package `asgn_4_package.tar.gz`, in which some string analysis methods including the calculation of the longest common string are implemented.

The configuration of the execution environment is as follows:

- Python 3.9.12
- Standard python interpreter
- CPU: 2.8 GHz
- Memory: 16 GB

Python dependencies:

- pytest
- pytest-benchmark
- pytest-cov

---

1. In `lib_raw.py`, the function `longest_common_substr()` contains a bug. Please address the following:

    1. Write test cases for the function in `tests/test_lib_raw.py` using `pytest`. [10 pt]
    2. Correct the bug in `lib_raw.py`. [10 pt]

2. Use `cProfile` to profile the function `StringAnalysis.analyze()`.

    1. Complete the code skeleton provided in `profiling/profiling_main.py`. [5 pt]
    2. Record the profiling results in `README.txt`. [5 pt]

3. Based on the profiling results from Question 2:

    1. Identify the function that takes the most time when considering both the time spent within the function itself and the time spent in all functions it calls. [5 pt]
    2. Identify the function that takes the most time when considering only the time spent within the function itself. [5 pt]

    Write your answers in `README.txt`.

4. Optimise the function `longest_common_substr()`. Only proceed with this question after preparing the regression test cases in Question 1.

    1. Implement the body of the benchmarking function `test_longest_common_substr_raw()` in `test_lib_benchmark.py`. [5 pt]

        Use the following command to run `pytest-benchmark`:

        ```
        pytest --benchmark-autosave --benchmark-compare=0001
        ```

        Here, `0001` refers to the ID of the saved benchmarking result, representing the initial result. For further details, refer to the pytest-benchmark documentation.

    2. Optimise the `longest_common_substr()` function in `lib_raw.py` to improve its runtime. A reduction of approximately 30% or more in runtime is achievable. Marks are given depending on how much run time can be reduced. [20 pt]

        When optimising:

    1. Make incremental changes: Verify the effectiveness of each modification before proceeding to the next.

    2. Avoid altering the overall approach of the algorithm, as it is deemed sufficiently effective.

    3. Use both the profiling results and intuition to guide the optimisation process.

3. Record the benchmarking results in `README.txt`. [5 pt]

2 Submission

1. What to submit:
    1. Your answers written in a Word or PDF document, if any
    2. The associated source code files, if any
2. Pack everything in a zip file named ".zip". E.g. if your student ID is `1234567890`, name the zip file as `1234567890.zip`
3. Submit the zip file to Blackboard