FIT3155 S1/2025: Assignment 2
(Due night 11:55pm on Fri 09 May 2025)

[Weight: 20 marks.]

This assignment has just one question. Your assignment will be marked on the *performance/efficiency* of your program. You must write all the code yourself, and should not use any external library routines, except those that are considered standard. Standard data structures (e.g. list, dictionary, tuple, set etc.) that do not conflict with your assessment objectives are allowed, but ensure that their use is space/time efficient for the purpose you are using them. Also the usual input/output and other unavoidable routines are exempted.

Follow these procedures while submitting this assignment:

The assignment should be submitted online via moodle strictly as follows:

- All your scripts MUST contain your name and student ID.
- Upload a .zip archive via Moodle with the filename of the form `<student_ID>.zip`.
  - Your archive should extract to a directory which is your student ID.
  - Include your a2.py script and a2report.pdf in this directory. Nothing else should be included.
- Submit your archive electronically via Moodle.
- Strictly adhere to the specifications listed in each question.
- Do NOT hard-code input filenames in your solutions. These should be passed from the command line.

Academic integrity, plagiarism and collusion

Monash University is committed to upholding high standards of honesty and academic integrity. As a Monash student your responsibilities include developing the knowledge and skills to avoid plagiarism and collusion. Read carefully the material available at https://www.monash.edu/students/academic/policies/academic-integrity to understand your responsibilities. As per FIT policy, all submissions will be scanned via MOSS or JPLAG.

Generative AI not allowed!

This unit prohibits you from using generative AI to answer this assessed task.

DL-distance ≤ 1 multiple pattern matching within a collection of texts

The statement of the task for this assignment is straightforward. Your program will be given a collection of text files and multiple pattern files. You will have to report all occurrences of each pattern that matches, under the DL-distance ≤ 1 threshold, the regions of texts in the collection.[1]

Note that you are constrained in this assignment to use the suffix tree data structure as your primary/foundational search data structure on which you will be running the DL-distance ≤ 1 searches. Further, to optimize the performance of this task, together with using a suffix tree, you may also employ supplementary data structure(s) or algorithm(s) chosen from this unit or its prerequisite units.

Details about the input text and pattern files:

- Each text/pattern file contains a string of characters drawn from the alphabet composed of 7-bit ASCII printable characters (32–126) and ASCII whitespace control codes (8–13, 32). You are free to assume '$' symbol does not appear in text/pattern files.
- Each text/pattern file contains at least one character from the above alphabet. It is safe to assume that the longest pattern is at most as long as the shortest text.
- Multiple text files can contain identical contents. Further, a larger text file may contain the contents of a smaller text file. The same applies to the pattern files.

Strictly follow the following specification to address this task:

Program name: `a2.py`

Argument to your program: Your program/script will accept one argument: filename of a `run-configuration` file. This `run-configuration` file will contain all information needed for you to read and run your program: it specifies the list of filenames of the input text files followed by the list of filenames of the patterns. (The filenames themselves can include the full path.) The precise format of the `run-configuration` file is given below:

```
<N=number of text files> <M=number of pattern files>
1 <text_filename_1.txt>
2 <text_filename_2.txt>
. ...
. ...
N <text_filename_N.txt>
1 <pattern_filename_1.txt>
2 <pattern_filename_2.txt>
. ...
. ...
M <pattern_filename_M.txt>
```

Command line usage of your script:

```
a2.py  <run-configuration-fileaname>
```

[1] Refer to Question 1 of Assignment 1 for the definition of DL-distance.

Output file name: `output_a2.txt`

Output format:

```
<pattern number> <text number> <position of occurrence> <DL-distance>
...
...
```

Further notes about the output:

- The `pattern number` and `text number` in each line of the output file should follow the numbering given in the `run-configuration` file (first column).
- Pattern numbers and text numbers need **not** be in a sorted order – you can report them in any order your program identifies them.
- The `position of occurrence` is the 1-based position where the pattern (specified by pattern number) was observed in text (specified by the text number) under a `DL-distance` of either 0 or 1 (cf. Question 1 of Assignment 1).

Written PDF Report: `a2report.pdf`

Write a report addressing these questions:

1. How did you handle multiple texts in your approach?
2. How did you handle DL-distance = 0 search on multiple patterns in your approach?
3. How did you handle DL-distance = 1 search on multiple patterns in your approach?
   - Report one subsection for each of the 4 edit operations: (1) substitution, (2) transposition, (3) insertion (of a character in a pattern), and (4) deletion (of a character in a pattern).
4. Were changes made to the suffix tree data structure (compared to the version delivered in your lecture)? If, yes, state the changes you had to make and justify them.
5. Did you use any other supporting data structures or algorithms not already covered in your responses to the above?
6. What is the worst-case time AND space complexity of your suffix tree construction over multiple files?
7. What is the worst-case time AND space complexity of your search approach under the DL-distance ≤ 1 threshold?

```
-=o0o=-
  END
-=o0o=-
```