FIT3155 S1/2022: Assignment 1
(Due midnight 11:55pm on Sun 27 March 2022)

[Weight: $10 = 5 + 5$ marks.]

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. 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.
- Use `gzip` or `Winzip` to bundle your work into an archive which uses your student ID as the filename. (STRICTLY AVOID UPLOADING `.rar` ARCHIVES!)
  - Your archive should extract to a directory which is your student ID.
  - This directory should contain a subdirectory for each of the two questions, named as: `q1/ and q2/`.
  - Your corresponding scripts and work should be tucked within those subdirectories.
- Submit your zipped file electronically via Moodle.

Academic integrity, plagiarism and collusion

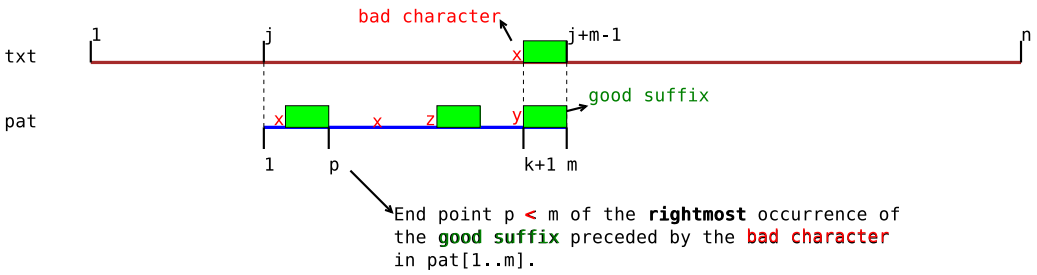
Monash University is committed to upholding high standards of honesty and academic integrity. As a Monash student your responsibilities include developing the knowledge and skills to avoid plagiarism and collusion. Read carefully the material available at https://www.monash.edu/students/academic/policies/academic-integrity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

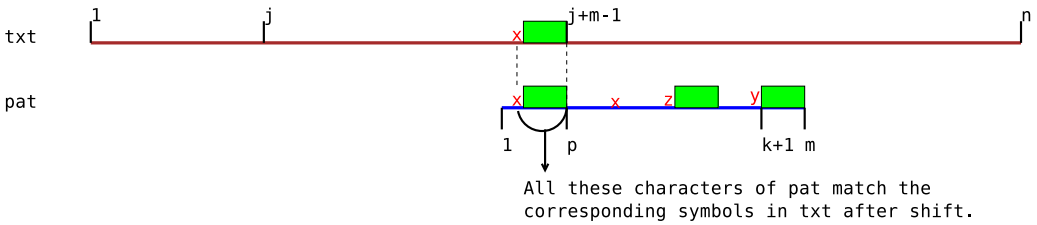
Assignment Questions

For the questions below, assume the alphabet $\aleph$ is composed of printable ASCII characters.

1. Given any text `txt`$[1 \ldots n]$ and any pattern `pat`$[1 \ldots m]$, your first task is to implement a modified version of the Boyer-Moore's algorithm, and report the positions of all occurrences of `pat` in `txt`.

Refer the following illustration of the modified shift rule:





Specifically, in any iteration, after a right-to-left scan, if a mismatch is observed at some position k in pat, shift by $m-p$ positions when there is a position $p < m$ in `pat` that is the **endpoint** of the **rightmost** instance of the good suffix, **such that the bad character precedes the good suffix**:

- $\texttt{pat}[p-m+k+1 \ldots p] \equiv \texttt{pat}[k+1 \ldots .m]$, and
- the preceding character, $\texttt{pat}[p-m+k]$, **is identical** to the bad character in the text identified during the right-to-left scan.

Otherwise, all other considerations of the Boyer-Moore algorithm discussed in Week 2 remain the same.

Strictly follow the following specification to address this question:

Program/Function name: `modified_BoyerMoore.py`

Arguments to your program/function: Two filenames, containing:

(a) the string corresponding to $\texttt{txt}[1...n]$ (without any line breaks).

(b) the string corresponding to $\texttt{pat}[1..m]$ (without any line breaks).

Do not hard-code the filenames/input in your function. Ensure that we will be able to run your function from a terminal (command line) supplying any pair of text and pattern filenames as arguments.

Output file name: `output_modified_BoyerMoore.txt`

- Each position where `pat` matches the `txt` should appear in a separate line. For example, when text = `abcdabcdabcd`, and pattern = `abc`, the output should be:

  1
  5
  9

2. Hint: use Z-algorithm to address this question.
Given some text `txt`$[1..n]$ and a pattern `pat`$[1..m]$, write a program to identify all positions within `txt`$[1..n]$ that matches the `pat`$[1..m]$ within a **Hamming distance** ≤ 1.

The Hamming distance between two strings of the same length is the number of corresponding positions where the characters between the two strings disagree.

Strictly follow the following specification to address this question:

Program/Function name: `hd1_patmatch.py`

Arguments to your program/function: Two filenames, containing:

(a) the string corresponding to `txt`$[1...n]$ (without any line breaks).
(b) the string corresponding to `pat`$[1..m]$ (without any line breaks).

Do not hard-code the filenames/input in your function. Ensure that we will be able to run your function from a terminal (command line) supplying any pair of text and pattern filenames as arguments.

Output file name: `output_hd1_patmatch.txt`

- Output format of each line of the output:

```
<position_in_txt>  <hamming_distance>
```

- Example output for text = `abcdyaacaacdaacz`, and pattern = `aacd`

```
1     1
6     1
9     0
13    1
```

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