CSCI 4118 / 6106 Assignment 4 **posted: Oct. 18, 2024**
Instructor: Travis Gagie **due: 23:59 Oct. 25th, 2024**

You are allowed to work in groups of up to 3. One person in the group is to submit your answers and a list of all the group members, and each other person in the group is also to submit a list of all the group members. You can talk with other groups about the normal assignments but, for everything you write, someone in your group should be able to explain it.

This assignment is due **Friday** night at 23:59, Halifax time. There are no SDAs for this course. Late submissions will not be accepted unless you have an accommodation through the Accessibility Center that allows you to submit work up to 3 days late, in which case we will accept your assignment until 23:59 on **Monday**.

Travis has modified his Boyer-Moore-Li (BML) code to try to reduce the number of backward searches it performs, by using a Bloom filter of selected k-tuples from the text.

When jumping right from $P[x]$ to $P[x + L - 1]$, he first scans left from $P[x + L - k..x + L - 1]$ to see if he can find a k-tuple $P[i..i + k - 1]$ that should be in the Bloom filter but isn't (meaning the k-tuple's hash is 0 congruent to the insertion parameter but a search for it in the Bloom filter fails). If he finds such a $P[i..i + k - 1]$ with $i > x$, then he resets $x = i$ and continues; if not, he backward searches from $P[x + L - 1]$ as usual in BML.

This is silly! A much neater solution would be to scan the whole pattern and find all the maximal substrings $P[i_1..j_1], \ldots, P[i_t..j_t]$ such that each substring has length at least L and no substring contains a k-tuple that should be in the Bloom filter but isn't. (Notice that the substrings can overlap by $k - 2$.) Once you have those substrings, you can apply BML to each one separately (and in parallel, if you want).

(Ok, Travis's original solution could be faster when L is large and he avoids looking at most k-tuples, but ignore that for this assignment.)

Rewrite Travis's code (available in the assignment directory on Brightspace) to use this cleaner solution. Once you have the substrings, you can search them for MEMs of length at least L using either backward-forward or BML. Make sure your code reports the correct MEMs of length at least 10 for the `dataset.txt` and `patterns.txt` files from Assignment 3.

Set a counter that tracks the total number of backward steps. Experiment with k and the insertion parameter to see what tradeoffs you can get between the number of backward steps and the size of the Bloom filter (the number of bytes in `dataset.blm`). Write a paragraph summarizing those tradeoffs and submit it with your code on Brightspace. Watch out for overflows when k gets big! (If you want to change Travis's code from 32-bit to 64-bit to reduce them, that would be nice.)