CS358: Applied Algorithms

Assignment 5a: Suffix Array Construction (due 11/06/25 at 10PM)

Instructor: Sam McCauley

Instructions

Code can be submitted by committing and pushing them to gitlab. I strongly suggest that you access `evolene.cs.williams.edu` through a web browser to make sure everything was uploaded as you expected. You may collaborate on code with your classmates, as well as the instructor and TA, but you may not use any LLM assistance. Please contact me at srm2@williams.edu if you have any questions or find any problems with the assignment materials.

Problem Description

In this problem, we will be finding the suffix array of a given text. In particular, you will receive as input a text T, and your goal is to return an array of integers A, such that if $A[i] = j$, then the jth suffix of T is the ith suffix of T in sorted order. As in class, we assume that the final character of T is a special character that is before any other in alphabetical order—we usually wrote \$ for this character in class, but this will be the null character `\0` in the code.

Input: The suffix array function is given a text string `str` of length `strLen`; note that `strLen` *does* include the terminal character.

This string is read from an input file, and stored in an array of 8-bit `char`s (this means you can assume that there are at most 256 different characters in any string).

Output: You should modify the function `suffixArray` in the file `suffixarray.c` so that it returns the suffix array of the given string, as an array of 32-bit signed integers. (Signed integers are not important for the final suffix array, but the implementation we discussed in class includes values that are temporarily below 0.)

You do not need to modify anything outside of this function.

Your code will be automatically tested in `test.c` against a working suffix array implementation (this working implementation is in `sais.c`; this is an $O(n)$ time implementation which is faster but more complicated)

Test Files: There are five shorter strings to help you test your code: `test.txt test2.txt test3.txt test4.txt test5.txt`. Finally, `timeData.txt` is a much larger file for testing. Your code on `timeData.txt` will likely require around 40 seconds to run.

A simple run of the program can proceed as follows:

```
./test.out test.txt
```

You can also use the `-v` flag, in which case the algorithm will output the original string, the suffix array of your algorithm, as well as the correct suffix array.

```
./test.out -v test2.txt
```

Questions

> **Implementation 1.** Implement the prefix doubling algorithm for suffix array construction in `suffixarray.c`.