CMPSCI 311: Introduction to Algorithms **Spring 2018**

Extra Credit Assignment

Released 4/18/2018 Due 11:59pm 5/1/2018 by email to me

Notes:

1. This assignment is extra credit. It is entirely optional.
2. If you are participating in the user study that Mark McCartin-Lim is organizing, you cannot get credit for this assignment (You will get credit there instead).
3. This assignment is worth up to 5 midterm-type points. The same is true for the user study.

Problem. Design and implement an efficient algorithm for the following task:

Given a string S, find the longest substring of S that appears at least twice in S.

Specifically, you will be given a string x of n characters. Your task is to find indices $i \neq j$ and a number k such that $x[i, \ldots, i+k] = x[j, \ldots, j+k]$ where k is as large as possible. Then you should output $x[i, \ldots, i+k]$ which is the longest repeated substring.

Other issues:

1. You may resolves ties arbitrarily. If there is a tie (two repeated substrings with the same length, which is the maximum), you may output whichever you like.
2. Observe that while i is not allowed to equal j, it could be the case that $j = i + 1$ so the substrings overlap. For example if $x =$ "aaaa", then the longest repeated substring is "aaa."

Collaboration. You must work independently. You may not work in groups or discuss your solution with anyone. You may use the internet and external resources for programming help *only*, you may not use other resources for algorithmic ideas.

Submission instructions. This extra credit assignment is due by 11:59pm on 5/1/2018. You should submit a single file to me via email (see below). I will not accept any late submissions.

You may use whatever programming language you like, but your program should consume the input string from `stdin` and write the output to `stdout` followed by a single newline. Make sure that the only output of your program is your answer, followed by a newline. Any sequence of bytes on `stdin` should be interpreted as a string and only the end-of-file (`EOF`) character terminates the string.

You must also prepare a `README` file with the following information.

1. The intuition for your algorithm.
2. High-level pseudocode.
3. Run-time analysis.

This file should be in plain ASCII text.

You will submit a gzip-ed tarball (extension `.tar.gz`) to me via email. The tarball should have at least two files `README` and `Makefile`, and potentially any source code files you want. If your gzip-ed tarball is called `submission.tar.gz` we will test your program using the following protocol:

```
$ gunzip submission.tar.gz
$ mkdir submission
$ mv submission.tar submission/
$ cd submission
$ tar -xf submission.tar
$ make -s compile
$ cat input1 | make -s run 1>output1 2>/dev/null
$ diff -q output1 answer1
```

If your program produces the correct answer, there should be no output. If your program is incorrect there will be some output. We will also time the execution of your algorithm and terminate it if it is too slow

Included with the assignment is a submission in python that meets the protocol but is an incorrect algorithm.

Do not submit pre-compiled files. Specify in the `compile` directive of your Makefile how to compile your source code.

Grading.

1. I will allocate up to 5 midterm-type points how I see fit.

Advice.

1. Spend some time on algorithm design before diving into implementation.
2. Make sure your submission follows the above protocol.
3. I strongly urge you to use python, and simply use the files that I have provided.