CS 454 Theory of Computation Fall 2018

Project # 1 Due: Sept 23, 2018

The project will be done in teams of size no larger than three. Each team will submit one solution – as a single program that when run asks if the user wants to solve Problem 1 or 2, then takes input for that problem (in case of problem 2, state clearly how the digits must be entered), displays the result and repeat the steps until the user choose option 3 (to quit). The project will be submitted through moodle. Please include all the files and create a zip file. Make sure that all the source files include the names of all the members of the team.

PROBLEM 1: Write a function count that computes the number of strings w of length n over $\{a, b, c\}$ with the following property: In any substring of length 4 of w, all three letters a, b and c occur at least once. For example, strings like *abbcaabca* satisfy the property, but a string like *bacaabbcabac* does not satisfy the property since the substring *aabb* does not have a c. The idea is to create a DFA M for the language $L = \{ w \mid$ in any substring of length 4 of w, all three letters a, b and c occur$\}$. Suppose M has m states. Assume that the states are labeled $0, 1, \ldots, m-1$ and that 0 is the start state. Create the transition matrix A associated with M of order m and use the expression $u\, A^n\, v$ where u is a row vector [1 0 0 … 0] (of length m) and v is the column vector $[v_0, v_1 \ldots v_{m-1}]^T$ where $v_j = 1$ if and only j is an accepting state, 0 else. A is defined as follows: A[i][j] = the number of transitions from state i to j in M. When your main function runs, it will ask for an integer input n, and output the number of strings of length n with the specified property. The range of n will be between 1 and 300. The answer should be exact, not a floating-point approximation so you should use a language that supports unlimited precision arithmetic like Java or Python or a library like GMP (in case of C++).

Some test cases:
Test case 1:
Input: $n = 137$ output: 6119266976149912241614898841866546736

Test case 2:
Input: $n = 100$ output: 9878022076381784001318849OO

PROBLEM 2: Write a function *MinString* that takes as input a DFA and outputs a string w of shortest length (lexicographically first in case of more than one string) accepted by the DFA. We already presented a Breadth-First Search (BFS) based algorithm to solve this problem. Use this function to write another function *smallestMultiple* that takes as input a positive integer k, and a subset S of $\{0, 1, 2, \ldots, 9\}$ and outputs the smallest positive integer y that is a multiple of k, and has only the digits (in decimal) from the set S.

Some test cases:
Test case 1: $k = 26147$, Digits permitted: {1 3}
Output: 1113313113

Input: $k = 198217$, Digits permitted: {1}
Output: integer containing 10962 ones (Your output will be a string of this many 1's.)