CS 454 Theory of Computation Spring 2023

Project 1, Due: February 17, 2023

The project will be done in teams of two or three members. (Three member teams should also solve Problem 2.) Only one submission is required per team. In addition to the code, please submit a short report that *states clearly the contribution of each member of the team.* Submission details will be specified in detail (along with test cases that shoud be included with the submission) in the canvas submission page. The project will be graded on (a) correctness on the test cases, (b) meeting the user interface requirements and (c) stating clearly the pre- and post-conditions, and the input/output relationship for each function. Around 5% of the total points will be assigned for (b) and (c) each.

PROBLEM 1:

The goal of the project is to compute the number of strings w of length n over $\{a, b, c, d\}$ with the following property: In any substring of length 6 of w, all three letters a, b, c and d occur at least once. For example, strings like *abbccdaabca* satisfy the property, but a string like *bad***aabcbc***dcabad* does not satisfy the property since the substring **aabcbc** does not have a d. The idea is to create a DFA M for the language:

$L = \{w \mid$ in any substring of length 6 of w, all the letters a, b, c and d occur$\}$.

By definition, all strings of length less than 6 are in L.

The idea behind an efficient solution to this problem is to design a DFA for L and to use an efficient algorithm to compute the number of strings of length n accepted by a DFA.

The algorithm for the counting the number of strings of a given length accepted by a DFA is briefly presented below:

Let $M = \langle Q, \Sigma, \delta, 0, F\rangle$ be a DFA. Assume that $Q = \{0, 1, \cdots, m-1\}$ and that 0 is the start state. The first step is to construct this DFA. This DFA has hundreds of states so you have to write a program to build it. The states will encode a buffer that holds the last 5 symbols scanned. When the next input symbol is read, the DFA checks if the last 6 symbols (the five from the buffer and the input read) meets the requirement that all the letters occur at least once. If this condition is not met, the DFA enters a fail state and thereafter it will remain there. If the condition is met, then the buffer is updated. Initially the buffer is empty so the first five transitions will just fill the buffer. Formally, if the current state q encodes a string $b_1b_2\cdots b_m$ where $m < 5$, then $\delta(q, a)$ encodes the state $b_1b_2\cdots b_ma$. If the state p encodes a buffer of length 5, say $b_1b_2b_3b_4b_5$, then $\delta(q, a)$ encodes the state $b_2b_3b_4b_5a$ if $b_1b_2b_3b_4b_5a$ contains at least once occurrence of each a, b, c and d, else $\delta(q, a)$ is the fail state. It is enconvenient to map each buffer string by a positive integer so that the states can be labeled 0, 1, 2, etc. One such encoding is to use base-4. For example, *babca* represents the integer $2 \times 4^4 + 4^3 + 2 \times 4^2 + 3 \times 4^1 + 4^0$.

The next step is to implement an algorithm that takes as input a DFA M and an integer n and computes the number of strings of length n accepted by M. This algorithm was presented and a number of examples were given, including some in quizzes 3 and 4. Specifically, this algorithm computes $N_j(n) =$ the number of strings of length n from any state j to an accepting state:

The number of strings of length n accepted by a DFA M is given by $N_0(n)$. The recurrence formula for $N_j(n)$ is given as follows: $N_j(n) = \Sigma_{x\in\{a,b,c,d\}} N_{\delta(j,x)}(n-1)$. Initial values $N_j(0)$ are given by: $N_j(0) = 1$ if $j \in F$, $N_j(0) = 0$ if $j \notin F$. Using this recurrence formula, you can compute $N_j(k)$ for all j for $k = 0, 1, \cdots, n$. As we noted in class, you only need to keep two

vectors *prev* and *next* of length m where m is the number of states. Using the values of $N_j(k)$ stored in *prev*, you can compute $N_j(k+1)$ for all $0 \leq j \leq m-1$ in *next*. Then copy *next* to *prev* and repeat.

When your main function is run, it will ask for an integer input n, and will 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 are included below:
$n = 6$ Answer: 1560
$n = 56$ Answer: 114451878183828768850216
Additional test cases will be provided later.

PROBLEM 2:

This problem involves counting the number of strings of length n in a slightly modified language L'. L' consists of strings over $\{a, b, c\}$ in which every substring of length 5 contains at least one occurrence of each symbol. (All strings of length ≤ 4 are included in L'.)