CS 454 Theory of Computation Project 2 Fall 2020

Due: Nov 15, 2020

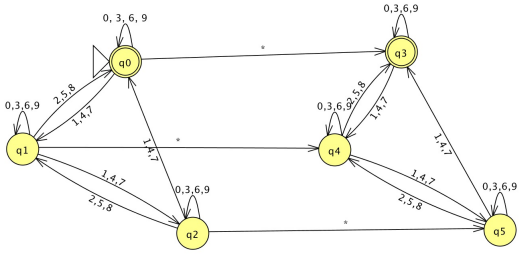
The projects will be done in teams of up to three. Each group will submit one program. Include the names of all members in the zip file. As with project 1, state the role of each member in the project.

In this project, you will solve a computational problem that is closely related to both Problems 1 and 2 of Project 1. The goal of this project is to show how the counting technique using DFA that we used in Project 1 can be extended to situations in which the construction of the DFA is a two step process: first we design an NFA, then convert it to a DFA. The problem is stated below.

We say that a number N is *weakly divisible* by an integer k if N is divisible by k, or removal of one of the digits of N gives an integer that it divisible by k. As an example, consider 653. Although it is not divisible by 7, removal of 5 gives 63 which is divisible by 7. Thus, 653 is weakly divisible by 7. Another larger example is 741842607938866199443579680083706254648829519399268 - a 50 digit number that is NOT weakly divisible by 7. This means none of the 51 numbers (the number itself and the 50 numbers you get by deleting one of the 50 digits are not divisible by 7).

The goal of the project is to determine, for two given integers N and k, the number of k digit integers that are weakly divisible by k. For example, when $N = 7$ and $k = 2$, the two digit numbers that are weakly divisible by 7 are 10, 14, 17, 20, 21, 27, 28, 30, 35, 37, 40, 42, 47, 49, 50, 56, 57, 60, 63, 67, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 84, 87, 90, 91, 97, 98 so the output is 37.

Algorithm to solve this problem: Given a positive integer N, we know how to build a DFA that accepts the decimal representation of all the integer multiples of N. Recall that this DFA M has n states - one state for each remainder $0, 1, \cdots, N-1$. We used such a DFA in Problem 2 of Project 1. It is not hard to modify the idea behind this DFA construction to build an NFA with $2N$ states that accepts the decimal representation of integers that are weakly divisible by N. The idea is as follows: Make two copies of the DFA M, and add transitions on all inputs $0, 1, \cdots, 9$ from state i of copy 1 to state i of the second copy, for all i. Call this NFA M_1. The figure below shows the NFA that accepts all the (positive) integers that are weakly divisible by 3. (This NFA accepts integers that begin with 0 such as 039, but in your construction, the NFA should reject such strings.) The next step is to convert M_1 to a DFA M_2 using subset construction. The final step is to count the number of strings of length k accepted by M_2 using the algorithm we used in Problem 1 of Project 1. Code for converting from NFA to DFA is widely available (some pointers to Python, Java and C++ implementations will be provided soon). But it is not hard to implement it yourself. The latter has the advantage of getting the DFA in the correct format to apply the final step of counting the number of strings accepted by a DFA.



What should be submitted? We know that converting the NFA to DFA can increase the number of states exponentially so we will fix N as 7. So the NFA M_1 has 14 states so you can construct this by hand. Conversion to DFA will result in a DFA M_2 with at most $2^{14} = 16384$ states (which is smaller than the size of DFA's in test cases of Project 1). The input to your program will be integer k, and output shall be the number of strings of length k accepted by M_2. Your program will be tested for the range of k from 1 to 200. As with project 1, the outputs can be much larger than 64-bit long long int, so you should use a package like gmp in case of C++. Some larger test cases will be provided soon.