

# CS 454 Theory of Computer Science

## Final Project Report

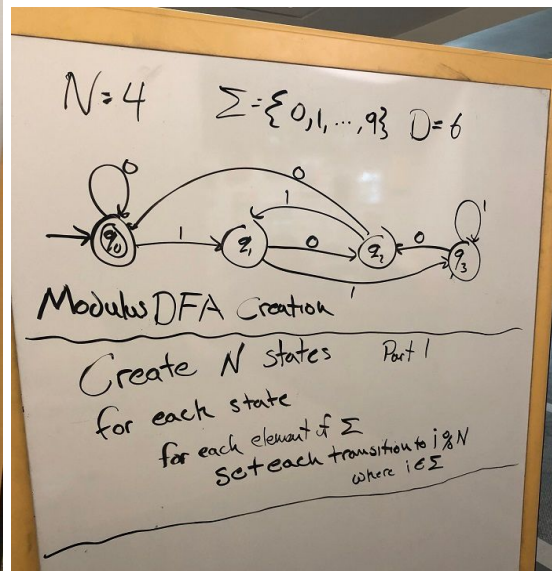
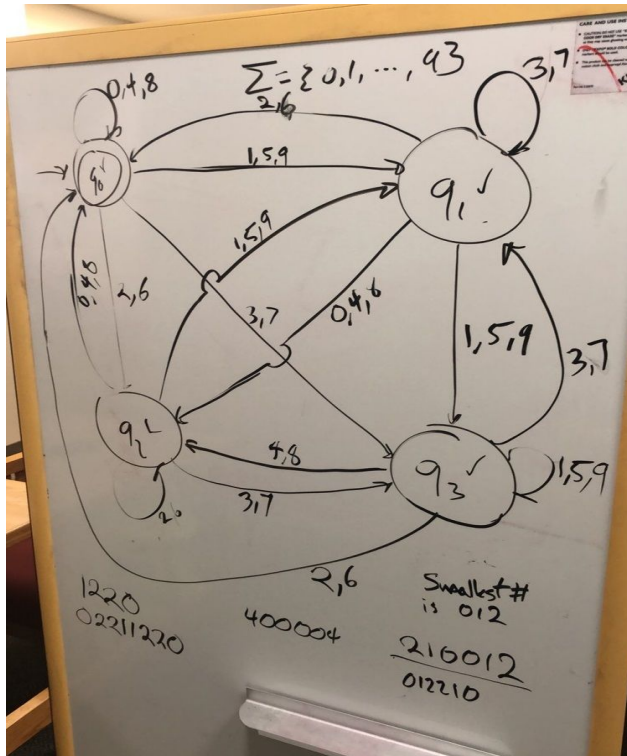
### 1. Problem Statement

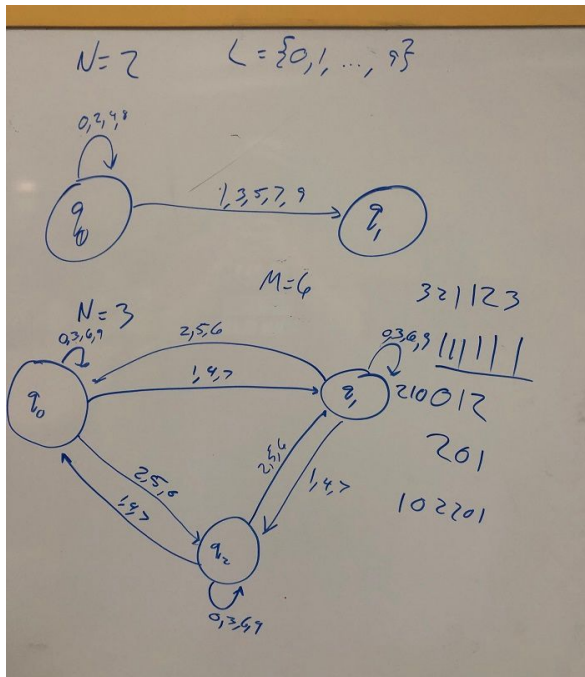
The goal of this problem is to count the number of palindromes of a given number of digits that is divisible by an integer  $N$ . This problem is closely related to Problem 2 of Project 1. The input are two integers  $N$  and  $M$ . The output is the smallest palindrome of length  $M$  that is a multiple of  $N$ . A related problem is: compute the number of palindromes of length  $M$  that are divisible by  $N$ . You can work on either one.

### 2. Problem Background

Our code eventually boiled down to some basic features as many of our creative ideas on how the problem might be solved failed to produce accurate results. We create a basic finite automata transition table that you can choose to print to screen or not. This  $N \times 10$  table is used to generate numbers divisible by  $N$ . At each position in the table the value is the next state you would reach by transitioning with the column digit.

**Handwritten Problem solving:**





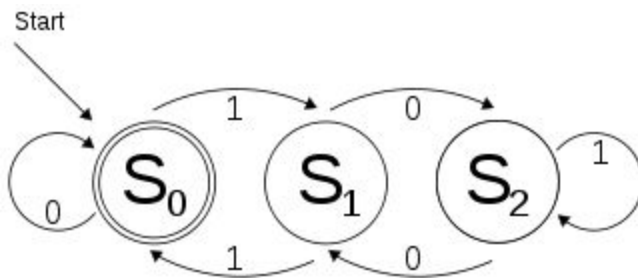
$N=5$

State	0	1	2	3	4	5	6	7	8	9
$q_0$	$q_0$	$q_1$	$q_2$	$q_3$	$q_4$	$q_0$	$q_1$	$q_2$	$q_3$	$q_4$
$q_1$	$q_0$	$q_1$	$q_2$	$q_3$	$q_4$	$q_0$	$q_1$	$q_2$	$q_3$	$q_4$
$q_2$	$q_0$	$q_1$	$q_2$	$q_3$	$q_4$	$q_0$	$q_1$	$q_2$	$q_3$	$q_4$
$q_3$	$q_0$	$q_1$	$q_2$	$q_3$	$q_4$	$q_0$	$q_1$	$q_2$	$q_3$	$q_4$
$q_4$	$q_0$	$q_1$	$q_2$	$q_3$	$q_4$	$q_0$	$q_1$	$q_2$	$q_3$	$q_4$

$125$   
 $521125$   $+23$   
 $52125$   $324423$

### 3. Algorithms Used

Very basic Idea show below but a foundation we built up from.



can be represented by a dictionary like this:

```
dfa = {0: {'0': 0, '1': 1},
       1: {'0': 2, '1': 0},
       2: {'0': 1, '1': 2}}
```

```
def generate_DFA_Table(M,N):
    states = [[0 for k in range(10)]for i in range(N)]
    for i in range(N):
        for k in range(10):
            states[i][k] = ((10*i) + k) %N

    return states
```

This is the algorithm we used to come up with the state transitions.

#### 4, Data Structures Used

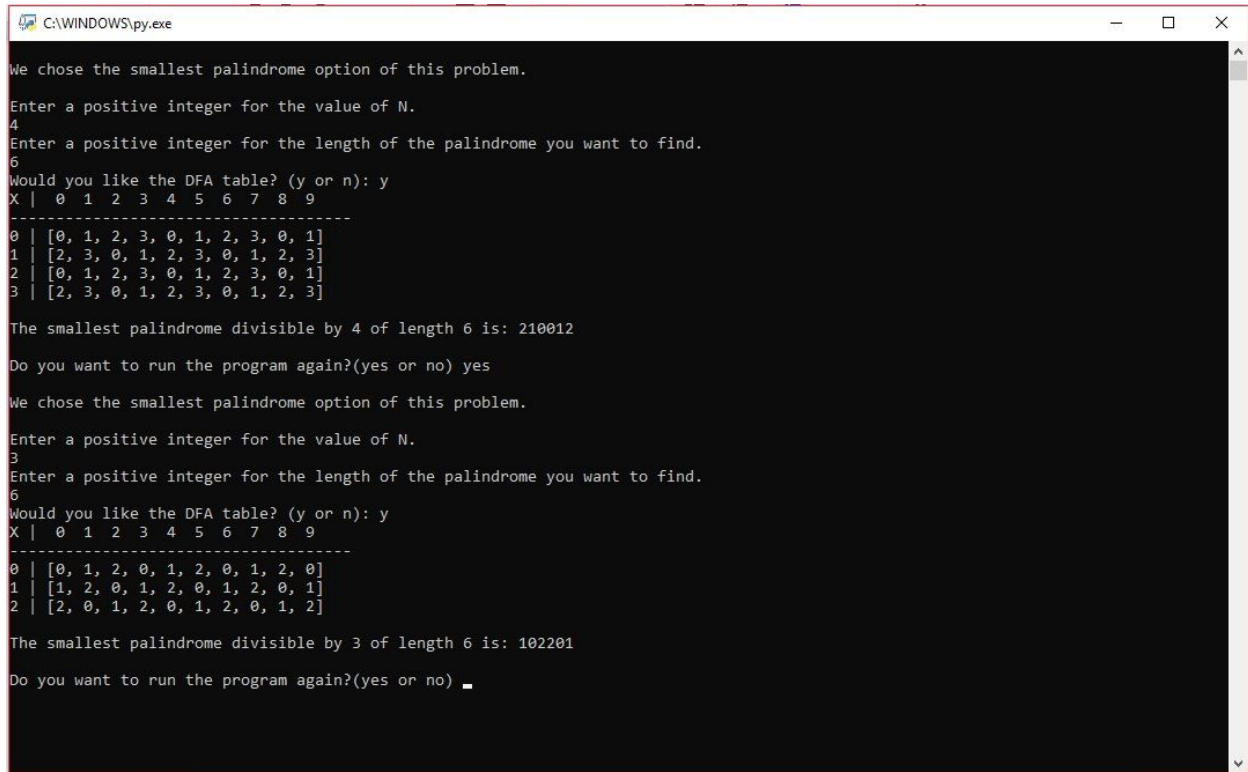
- DFA transition table (N x 10 matrix)
  - One of the data structures we decided upon was a DFA transition table of ( N X 10 matrix ) in size. The reason being is this data structure provided optimal use in finding the path that granted us with the smallest palindrome. The way that this data structure works is it takes the modules of a given number. With that data the transition table then knows what state to step into next. This process is what runs our entire program and is essentially the hub of data to state transitions.
- Visited Queue
  - The visited Queue data structure is another very vital data structure used in our program. What it does is essentially exactly how it sounds. It checks if the state we are going to visit has already been visited before. What this does is save massive amounts of computational time as it can cut out many possibilities that an algorithm such as brute force would not.. This data structure is the reason we are able to process such large numbers so quickly,

#### 5. Summary of Results Obtained

- The results we obtained from this project are quite stunning. At the start of this project we were doing a lot of research on how to generate the next smallest palindrome given the constraints. We soon realized that this approach to the project was not going to work. The reason being is it is basically a faster brute force approach in which we only look at palindromes but we look at all of them. That discovery is what led us to the current solution we have now. Instead of generating each and every palindrome we ourselves need to generate the palindrome we are looking for. What this enables us to do is throw out so many wrong answers just from the get go. This method cuts down on the time complexity of the problem and essentially is the solution to our problem. This project was by no means easy, in fact it quite possibly could have been the hardest project we

have ever jumped into. However the hard work and results we obtained were well worth it. As everyone in the group agrees the learning experience that came along with this behemoth of a problem was unmatched as we all feel much more knowledgeable about the application and of DFA driven programs.

## 6. Sample Program Run



```
C:\WINDOWS\py.exe

We chose the smallest palindrome option of this problem.
Enter a positive integer for the value of N.
4
Enter a positive integer for the length of the palindrome you want to find.
6
Would you like the DFA table? (y or n): y
X | 0 1 2 3 4 5 6 7 8 9
-----
0 | [0, 1, 2, 3, 0, 1, 2, 3, 0, 1]
1 | [2, 3, 0, 1, 2, 3, 0, 1, 2, 3]
2 | [0, 1, 2, 3, 0, 1, 2, 3, 0, 1]
3 | [2, 3, 0, 1, 2, 3, 0, 1, 2, 3]

The smallest palindrome divisible by 4 of length 6 is: 210012
Do you want to run the program again?(yes or no) yes

We chose the smallest palindrome option of this problem.
Enter a positive integer for the value of N.
3
Enter a positive integer for the length of the palindrome you want to find.
6
Would you like the DFA table? (y or n): y
X | 0 1 2 3 4 5 6 7 8 9
-----
0 | [0, 1, 2, 0, 1, 2, 0, 1, 2, 0]
1 | [1, 2, 0, 1, 2, 0, 1, 2, 0, 1]
2 | [2, 0, 1, 2, 0, 1, 2, 0, 1, 2]

The smallest palindrome divisible by 3 of length 6 is: 102201
Do you want to run the program again?(yes or no) _
```

## 7. References

<https://stackoverflow.com/questions/35272592/how-are-finite-automata-implemented-in-code>

<https://www.hackerearth.com/practice/algorithms/graphs/breadth-first-search/tutorial/>

[https://en.wikipedia.org/wiki/Deterministic\\_finite\\_automaton](https://en.wikipedia.org/wiki/Deterministic_finite_automaton)