A Security Device – Memo

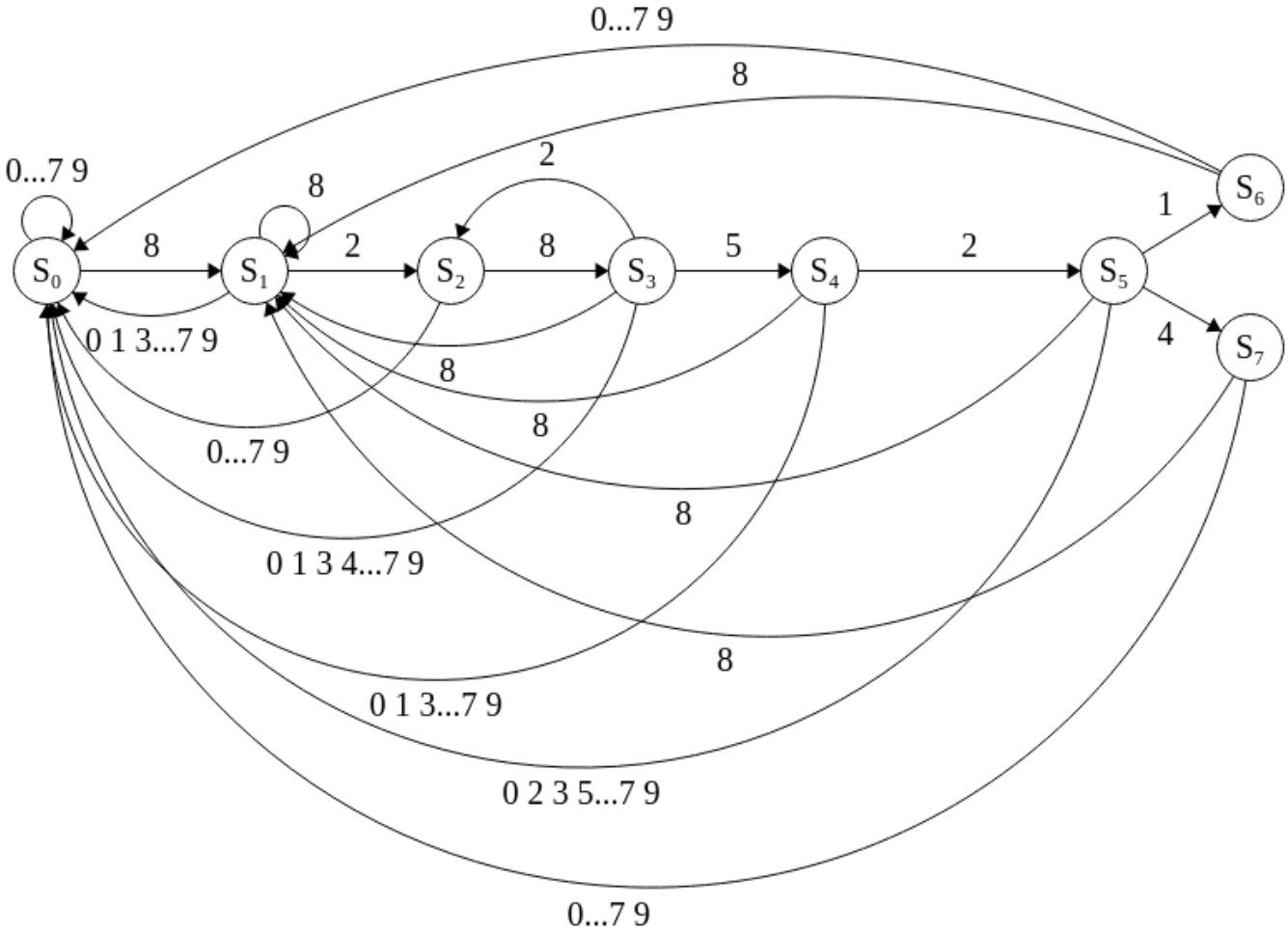
IIT Fall 2022 – CS330 Coding Assignment

Mohamad Fares (A20482852)

https://github.com/Moody162/A-Security-Device

Implementing the SecurityDevice class

The core of the project is the SecurityDevice class implemented in the “fsm.py” file. This class implements a simple security device that utilizes a Finite State Machine. The State Transition Diagram of the FSM is the one below. S_6 is the UNLOCK_STATE, and S_7 is the LOCK_STATE.



As we can see, the language accepted by the FA are all digits from 0, 1, 2, … 9. However, the code will accept anything (i.e. also strings, characters, symbols, etc.), but for anything that is not a digit, the state

of the FSM will be reset to S_0. At the same time, a negative number will be treated as if the "-" sign doesn't exist (for example, -5 will be treated the same way as 5).

The regular expression corresponding to the language of the FA is:

R = ('0' + '1' + '2' + '3' + '4' + '5' + '6' + '7' + '8' + '9')*.

As we have established, anything that is not part of the language is going to reset the FSM to S_0.

The SecurityDevice class contains one instance attribute (self.state), and the class implements two methods: output(), which returns the output of the FSM in its current state, if there is one (if not, the method returns None), and the enter(val) method, which enters input into the FSM and changes the state of the FSM accordingly.

The unittest coverage tests "fsm.py" (the file containing the FSM class) entirely.

Part 1 of project

The first part of the project is implemented in "security_device.py", which is an executable file. Running it allows the user to simulate a simple Security Device. This file imports the "fsm" module I implemented to be able to use the SecurityDevice class.

"security_device.py" reads input in an infinite loop (user has to manually kill the process to exit). It enters the input in a SecurityDevice object, and, prints messages to the screen when the output() method within the SecurityDevice class returns something that is not None.

Part 2 of project

My estimate of how long would it take for someone to randomly generate the unlock pass-code for the security device is 10^6 = 1,000,000 typed digits. My reasoning was that the probability to guess each digit on each position is 1/10, and since there are 6 digits in the pass-code, I raise the probability to the power 6.

The second part of the project is implemented in "generator.py". This python file contains two methods, guess_passcode() and guess_average(tries).

guess_passcode(0) generates random input of digits in the range 0 … 9 in a loop for a SecurityDevice object. Once the random input generated creates a sequence which changes the state of the SecurityDevice to the UNLOCK_STATE, the loop stops. The method returns the number of randomly generated input.

guess_average(tries) calls guess_passcode() tries times in a for statement. It calculates the minimum, maximum and average of the return value of each guess_passcode() call. It returns a 3-tuple containing the minimum, maximum, and average.

“generator.py” calls guess_average(20). This process takes a bit of time (maybe approximately 20 - 30 seconds or so, depending on the generated input).

I found that the average usually was a bit more than 1,000,000. It usually was around 1,200,000 – 1,400,000.