TCS Spring ‘17

Assignment 2

Parsing Regular Expressions

Due: April, 19 ‘17

Problem Definition

You are asked to develop a program which is able to parse regular expressions and output whether a string is matched by the given expression. Internally, the program must construct an NDFSA that corresponds to the regular expressions. NDFSA **must be constructed according to the Thompson’s Algorithm.** The program will take a list of regular expressions and test strings in a text file, construct NDFSA and run each test string through the NDFSA. For each test string, the program should output if the string is recognised by the NDFSA that corresponds to the regular expression and the accepting trace of states from initial to the final if the string is recognized. Keep in mind that the number of accepting execution traces is not necessarily equals to one!

Example

Let us demonstrate the algorithm by creating an NDFSA for a regular expression “01*” (Figure 1).

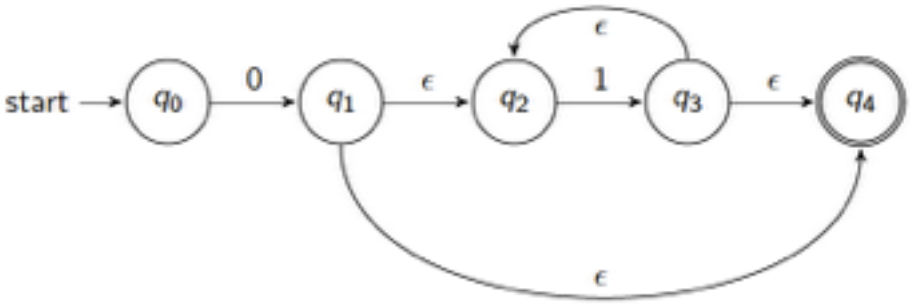

Figure 1. NDFSA for “01*”

It is a complex expression that needs to be broken down into a list of basic sub-expressions:

R = 01*
R_1 = 0
R_2 = 1
R_3 = (R_2)*

For each subexpression we construct an appropriate automaton and then combine them using composition rules from Thompson’s Algorithm. The result should be a similar automaton as shown in Figure 1.

Running “011” through this automaton will generate the following execution tree:

Figure 2. Execution tree that resulted from running “011” through NDFSA

Leaves of terminated branches are shown in red while leaves of accepting are shown in green

As you can see from Figure 2, string “011” is recognised by NDFSA and its execution trace that results in the string being accepted:

```
(q0 q1 q2 q3 q2 q3 q4)
```

Input

The first line of the input file will consist of an integer number *k* specifying the number of test cases. This is followed by the data specifying each test case. There will be no blank lines in the input file.

Each test case consists of two parts - the regular expression and a set of test strings.

The regular expression description consists of exactly 1 line and is followed by the integer number *n* specifying the number of test strings for the regular expression. The following n lines each contain a single test string.

Output

For each test string in a respective test case, print the execution trace in the form of “inital_state state_1 state_2…” if the string is recognised, else print nothing (see Sample Output for details)

NOTE 1: You only need to output the trace if the string is recognized, else output empty string
NOTE 2: The output for each test string should be on a separate line
NOTE 3: If there are more than one possible traces it is sufficient to list at least one

Development Guidelines

Your program should be developed using Python 3.4 or Python 2.7 programming language (select version from the list). Although we leave the choice of programming paradigm to you, it is desirable to use object oriented approach.

The entry point for you program should be designed in the following way:

```
if __name__ == “__main__":
      # main routines here
```

Sample Input

```
01*
5
000011111
011111
10
01
0


(0|1)01
3
001
101
010


00(0|1)*
5
00000
00
1000
0011
00110
```

Sample Output

```
q0 q1 q2 q3 q2 q3 q2 q3 q2 q3 q2 q3 q4

q0 q1 q2 q3 q4
q0 q1 q4
q0 q1 q2 q5 q6 q7
q0 q3 q4 q5 q6 q7

q0 q1 q2 q3 q6 q7 q4 q3 q6 q7 q4 q3 q6 q7 q4 q5
q0 q1 q2 q5

q0 q1 q2 q3 q8 q9 q4 q3 q8 q9 q4 q5
```

q0 q1 q2 q3 q8 q9 q4 q3 q8 q9 q4 q3 q6 q7 q4 q5