ATC-Assignment 2

In this assignment you need to implement a tool to build a NFA from a regular expression.

Problem Given a regular expression r, the tool should implement a NFA $\mathcal{A}$ which accepts languge L recognized by r. We will address the tool as **NFA-Builder**.

NFA-Builder should build on the tree representation for the regular expression. For example, in the figure 1 is an expression tree for regular expression $(a + (b \cdot (c^*)))$.

The tool should build and output NFA at the root node after visiting all other nodes in the tree.

NFA-Builder assumes alphabet as $\Sigma = \{a, b, c\}$.

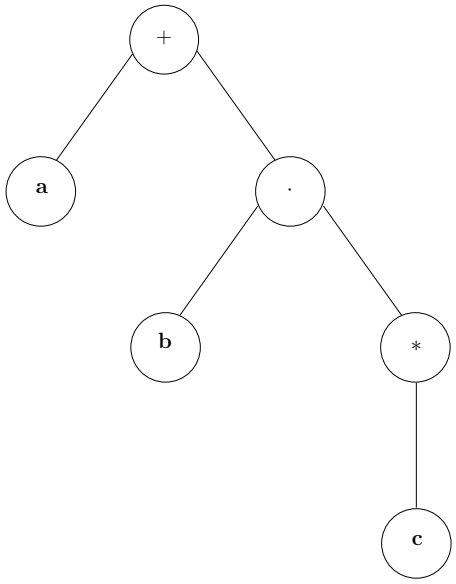

Figure 1: Expression tree for $(a + (b \cdot (c^*)))$

Code and Implementation Details To enable uniform implementation, this repository `https://github.com/AbhishekIIScPhD/Regex2Automaton` contains code that has place holders to implement required functions.

Note: You are required to use only these files or create new files of your own without importing modules of any sort.

Please look into each file in the repository before you begin implementation. The implementation is spread across two files

* ***main.py*** contains the entry point to the tool. The tool is implemented as a unit-test, which means if the tool implementation does not produce the expected output at any of the input tests then the tool terminates with an assertion failure.
* ***utils.py*** contains classes **class ETree, class ENode, class FSA, class NFA**.
    * **class ETree** models a expression tree.
    * **class ENode** models a node in an expression tree.
    * **class FSA** models a finite state automaton.
    * **class NFA** models a non-deterministic finite state automaton.

Your implementation goes into ***utils.py***, specifically as methods of **class ETree**. You are required to visit and process each node **ENode** in the expression tree **ETree** . Processing a node in the tree means to build an automaton at each node. Once an automaton is built at the children of a node you should propagate them to build an automaton at the parent node. Details on automaton representation are given in the subsequent sections.

Input Details The input is read from "input.txt". A sample input file is,

```
3         // number of regexes
(a*)      // regex_1
1         // number of tests
aaaaaaaa  // test1
(b*)      // regex_2
1         // number of tests
bbbbbbbb  // test1
(a+b)     // regex_3
2         // number of tests
a         // test1
b         // test2
```

We will use e to represent an epsilon in a regular expression.

Output Details You should output the final automaton at the root node on the terminal/ stdout.

Below is the representation of the automaton due to the regular expression tree in figure 1 at its root.

Note: This may not be the only generated NFA, the number of states may differ.

```
3       // Number of states
{0}     // Initial State is a set in python
{1,2}   // Final States is set in python

// transition matrix from a state in row to a state
// in column on each alphabet is a dictionary in python where
// keys are alphabet and values are transtions for each
// symbol in the alphabet.
{
'a' : [[0,1,0],
       [0,0,0],
       [0,0,0]],
'b' : [[0,0,1]
       [0,0,0],
       [0,0,0]],
'c' : [[0,0,0],
       [0,0,0],
       [0,0,1]]
}
```

Evaluation The repository has a file named "expOutput.txt" this file will contain the expected decisions on input test strings in "input.txt".

Evaluation is based on the number of matches with the decisions in "expOutput.txt".

Evaluation also considers the quality of the submitted code in terms of readability of the code with proper variable names and comments as needed.

What to submit? Submit only the following files.

* main.py
* utils.py
* other files related to the implementation.
* empty input.txt and expOuput.txt files.

We will create an assignment on the Teams channel. You are expected to zip the above mentioned files and submit it as <**name**>_<**reg_no**>_**assig2.zip**.

Setup You are required to install **Python3** and **Git** to work with the assignment.

1 Clone the repository by visiting the given link.

2 Run the main.py file as "python3 main.py"

Please let us know if you have any issues with the implementation.