编译原理实践第 1 次课
正则表达式到 NFA 和 DFA 的转换
(2 周实践课内容)

1. 编程环境 (不限)

Python (2.7, 3.6)
C++
Java

2. 实现正则表达式至最简 DFA 的转换，包括:

1) MYT 算法的实现 (正则表达式转 NFA)

要求(以 Python 为例):

python ***.py input.txt output.txt
其中:
1) ***.py: 您的 MYT 算法主程序
2) input.txt: 需要待转换的正则表达式. 为方便起见, 该文件只包含一行, 即只有一个正则表达式. 如:

a(b|c)*d

3) output.txt: 存放生成的 NFA，其中以三元组形式存放，每行一个表示 NFA 的一条边，格式为 1 a 2, 表示状态 1 经过符号 a 到达状态 2. 最后 start state: 1 表示状态 1 为开始状态; accepting states: 2 3 表示状态 2 和 3 为接受状态.

1 a 2
….
start state: 1 accepting states: 2 3

2) 子集构造算法的实现 (NFA 转 DFA)

要求(以 Python 为例):

python ***.py input.txt output.txt
其中:
1) ***.py: 您的子集构造算法主程序
2) input.txt: 需要待转换的 NFA (格式见上面 output.txt).
3) output.txt: 存放生成的 DFA (格式与 input.txt 格式一致)

3) DFA 最小化

要求(以 Python 为例):

python ***.py input.txt　output.txt
其中:
1) ***.py: 您的 DFA 最小化算法主程序
2) input.txt: 需要待最小化的 DFA (格式见上面 output.txt).
3) output.txt: 存放最小化的 DFA (格式与 input.txt 一致)

注意: 为方便起见，用字符 e 表示空串 ε，并且假设输入的正则表达式不涉及符号 e 。

3. 提交源码，以及运行说明 readme.pdf。readme.pdf 中说明如何编译及运行，以及一个例子 (a|b)*abb 的例子运行结果。