**编译原理实践第3次课**

**（从正则表达式到NFA）**

1. 基于MYT算法，实现将正则表达式转换为NFA。需要实现以下正则表达式符号：

a

ab

a|b

(ab)

a\*

(ab)\*

1. 输入为正则表达式，输出为一个三元组序列，每个三元组为：（起始状态，符号，结束状态）举例：

1）ab

|  |  |  |
| --- | --- | --- |
| **起始状态** | **符号** | **结束状态** |
| 0 | a | 1 |
| 1 | b | 2 |

2）ab\*

|  |  |  |
| --- | --- | --- |
| **起始状态** | **符号** | **结束状态** |
| 0 | a | 1 |
| 1 | b | 1 |

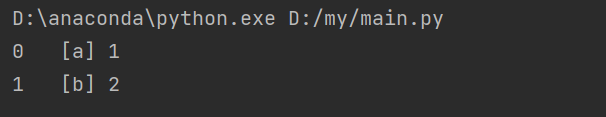
1. 示例程序在example文件夹下

核心思想是，为每种正则表达式的元字符设计一种存储三元组的方式，上述1）和2）已经给出2个例子了。

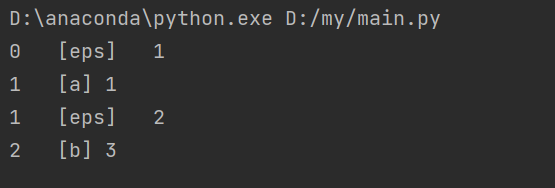
1. 最终的程序，需要能够完成以下正则表达式的解析：
2. ab
3. a\*b
4. a（a|b）c
5. a（ab）\*d

**提交代码与实验报告（实验报告里需要有对于实验过程的文字描述与运行结果截图）**

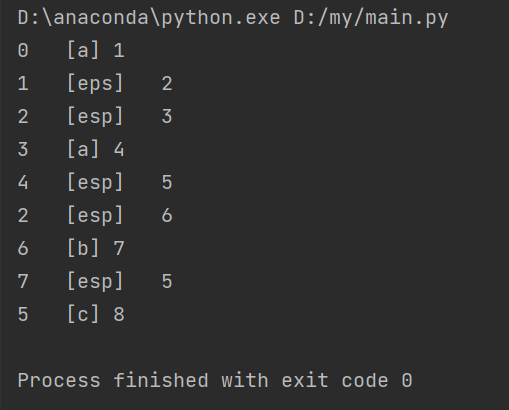
（1）ab



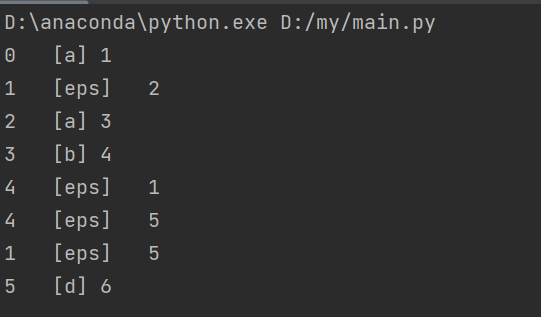
（2）a\*b



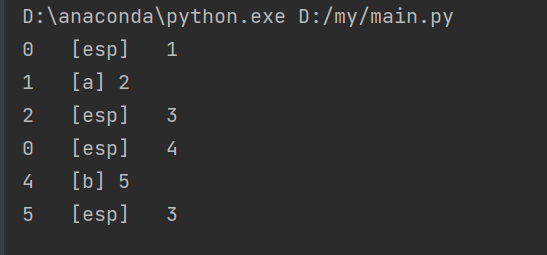
（3）a(a|b)c



（4）a(ab)\*d



1. a|b



**实验过程及感想**

**就是一开始考虑的a|b好像不是很全，后来修改了代码，改成在'|'之前和'|'之后分别处理两条路可以尽量解决这个问题，所以重新修改了代码，将在后括号处的处理调整为'|'之后处理。**

**感想：在本次实验中，实现了基于MYT算法将正则表达式转换为自动机，我做实验的时候，思路基于手工构造得出规律，可能一些复杂的式子还是难以解决。**