**实 验 报 告**

|  |  |
| --- | --- |
| **课程名称：** | **编译技术** |
| **学生姓名：** | **葛子午** |
| **学生学号：** | **202330550381** |
| **学生专业：** | **软件工程** |
| **开课学期：** | **2024-2025第二学期** |

**软件学院**

**2025年05月**

# 实验二 正则表达式的实现（NFA to DFA）

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **地 点：** | | | B7楼 | | 331房； | | **实验台号：** | |  | |
| **实验日期与时间：** | | | 2025.5.17 | | | | **评 分：** | |  | |
| **预习检查纪录：** | | |  | | | | **实验教师：** | |  | |
| **电子文档存放位置：** | | |  | | | | | | | |
| **电子文档文件名：** | | |  | | | | | | | |
| **批改意见：** | | |  | | | | | | | |
| 出勤情况（20%） | 课堂态度（20%） | | 实验报告质量（60%） | | | | | |
| 任务完成情况 | | 源代码注释 | | 报告内容 | |
| □ 出勤  □ 缺勤  □ 早退  □ 请假 | □ 认真  □ 不认真 | | □ 完成  □ 未完成 | | □ 优秀  □ 良好  □ 通过  □ 未通过 | | □ 优秀  □ 良好  □ 通过  □ 未通过 | |

# 报告内容

1、实验目标

a) 实现NFA到DFA的转换算法（子集构造法）

b) 实现DFA最小化算法（等价类划分法）

c) 构建完整的正则表达式处理流程：正则表达式 → NFA → DFA → 最小化DFA

2、实验环境

**操作系统：Windows 11**

**开发工具：**

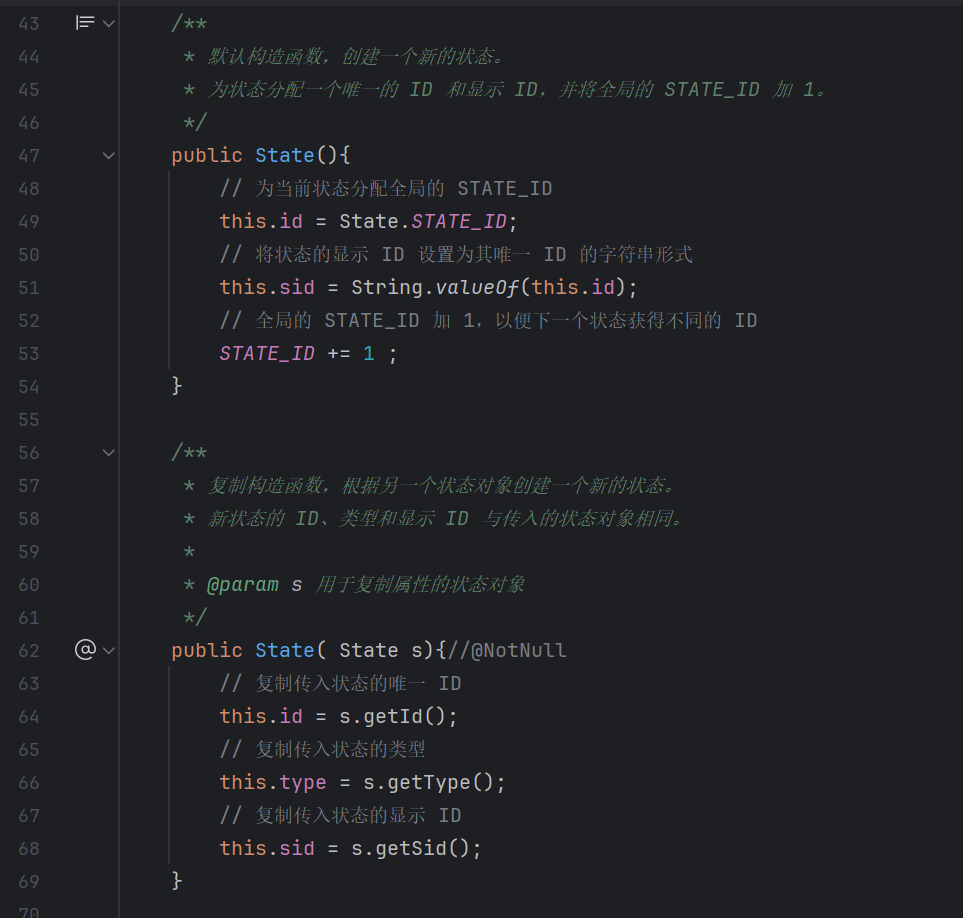
* **Java开发环境：JDK 17**
* **集成开发环境：IntelliJ IDEA**

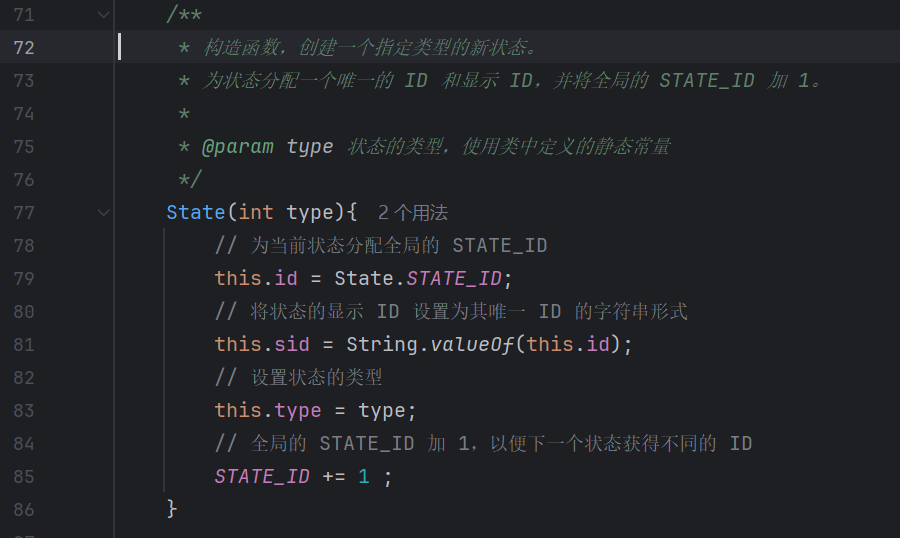
3、关键数据结构和核心算法

3.1 数据结构

（1）状态机状态 State 类

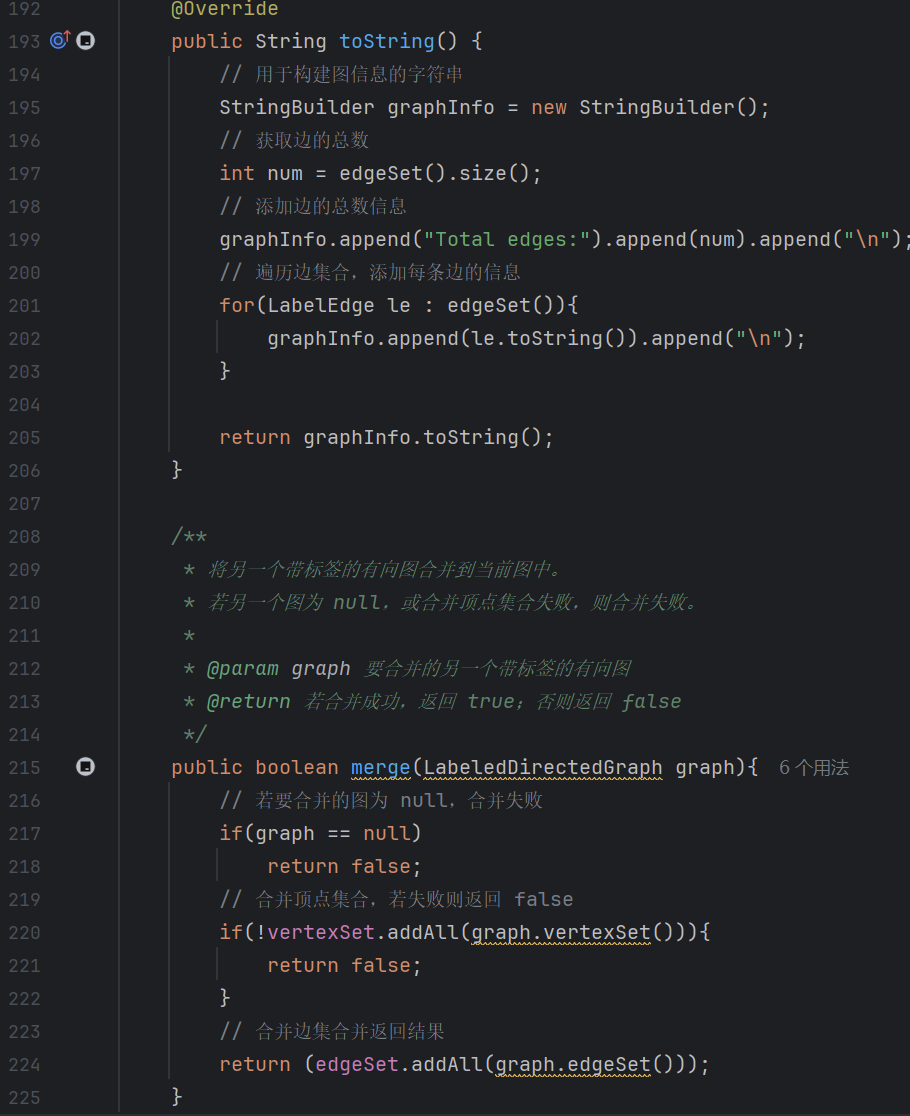






1. 转移表 LabeledDirectedGraph

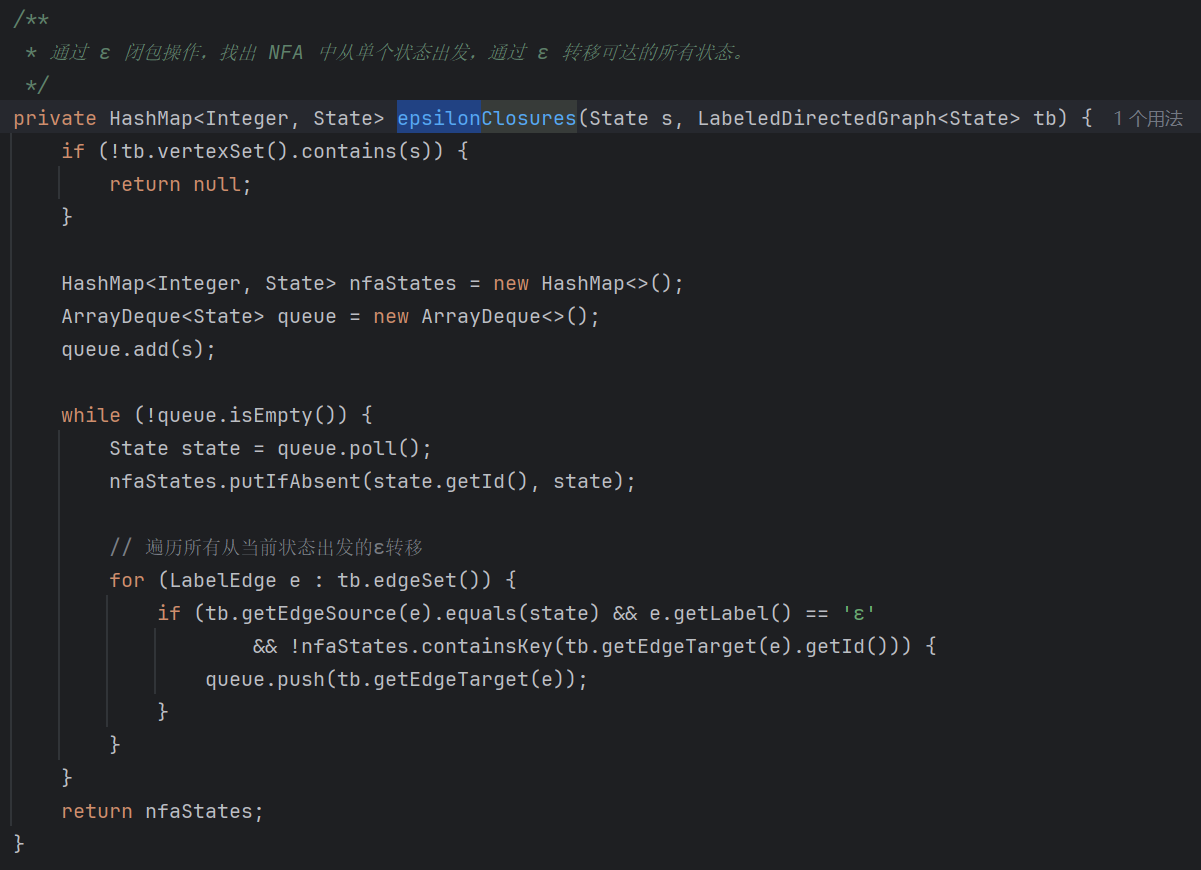




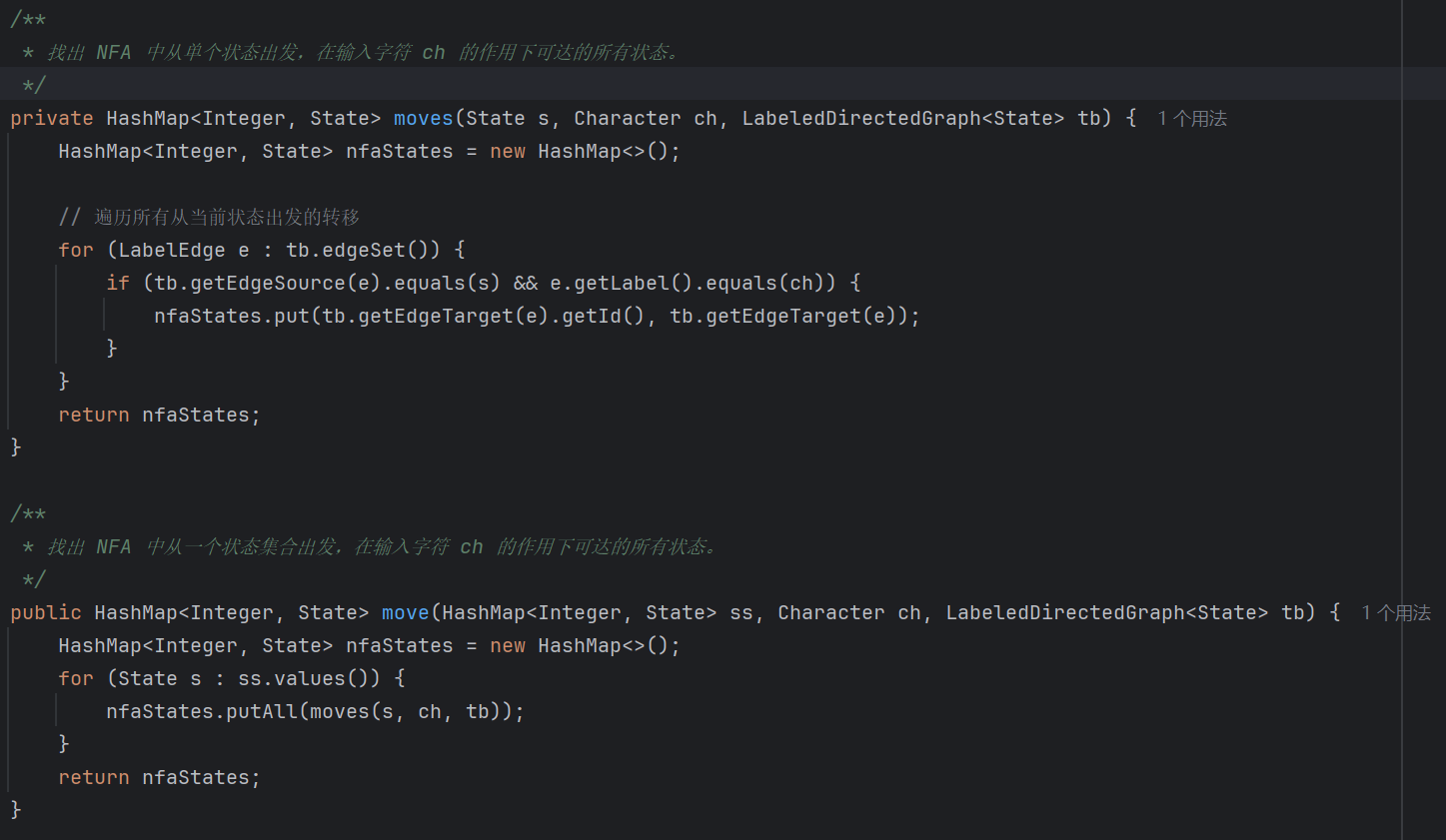
3. 2 核心算法

（1）子集构造算法（NFA→DFA）

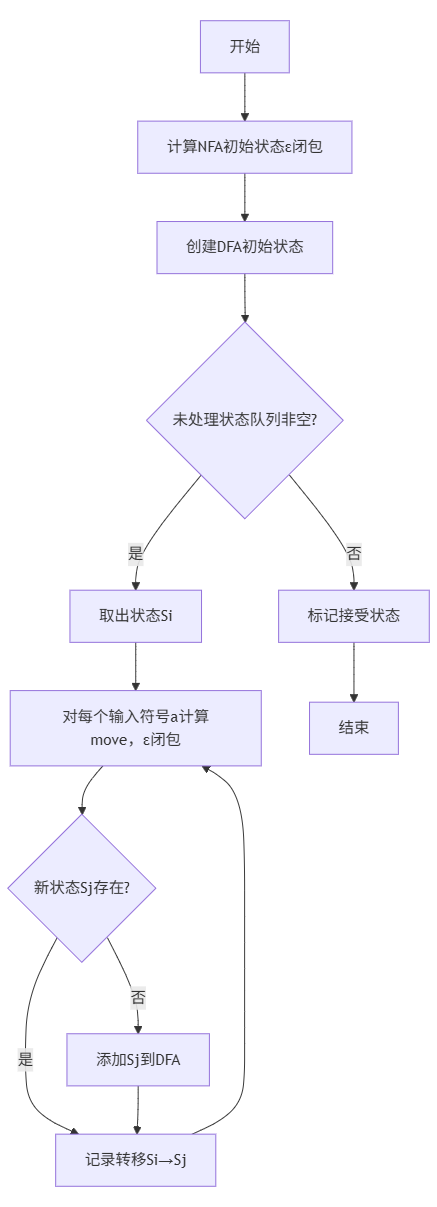
① ε闭包计算



② 状态迁移计算

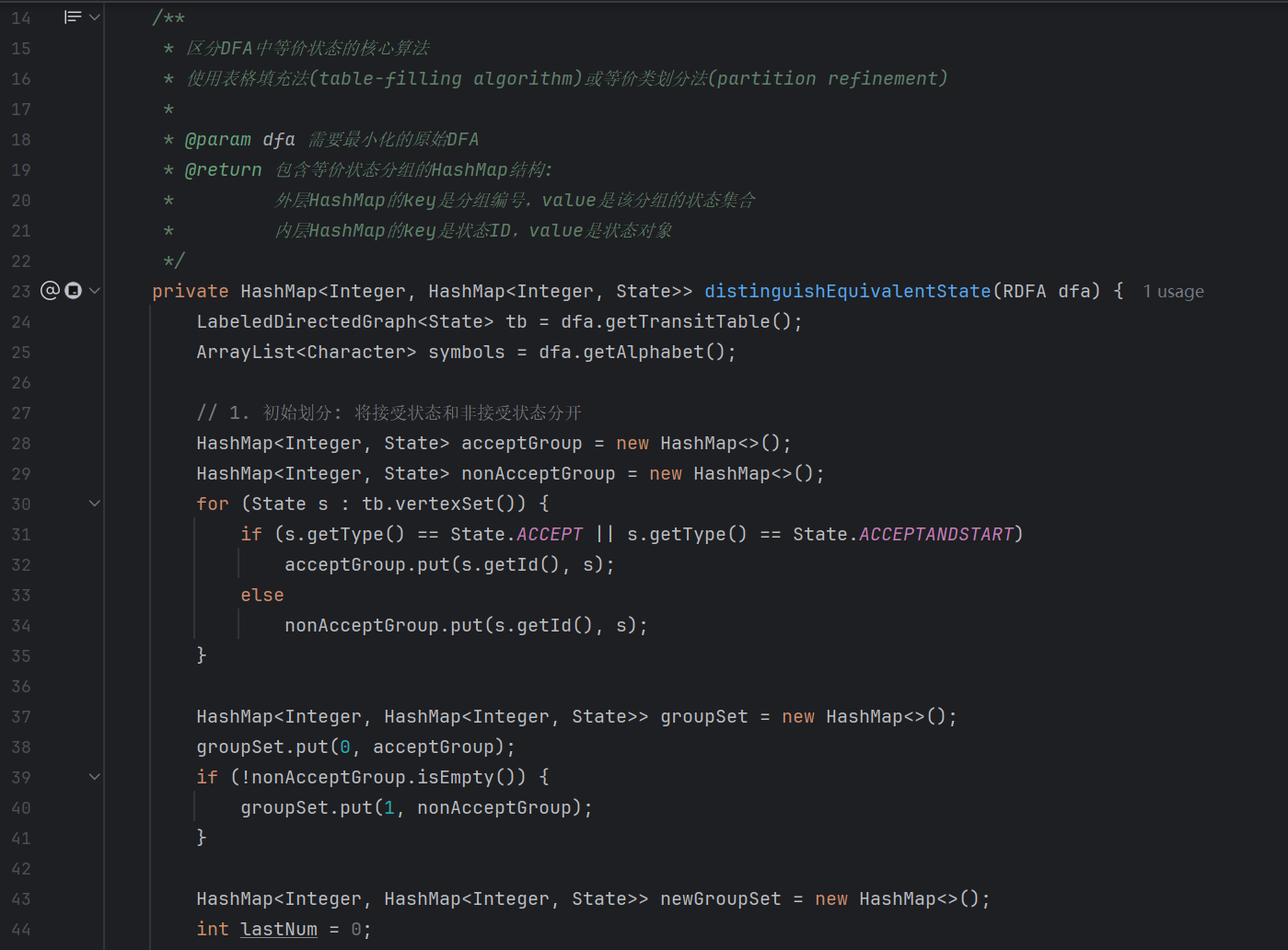


算法流程图示意：



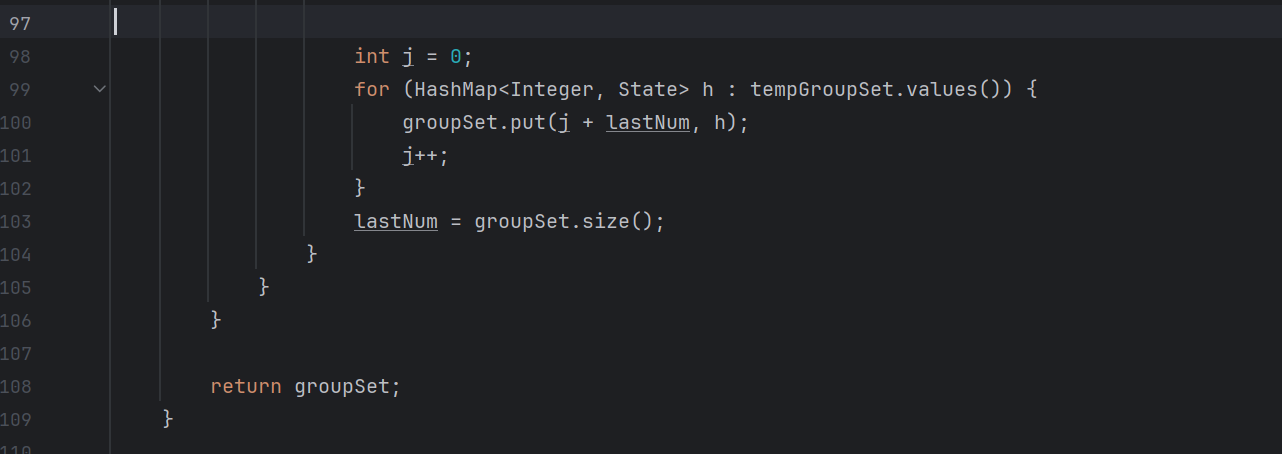
1. DFA最小化算法

① 等价类划分





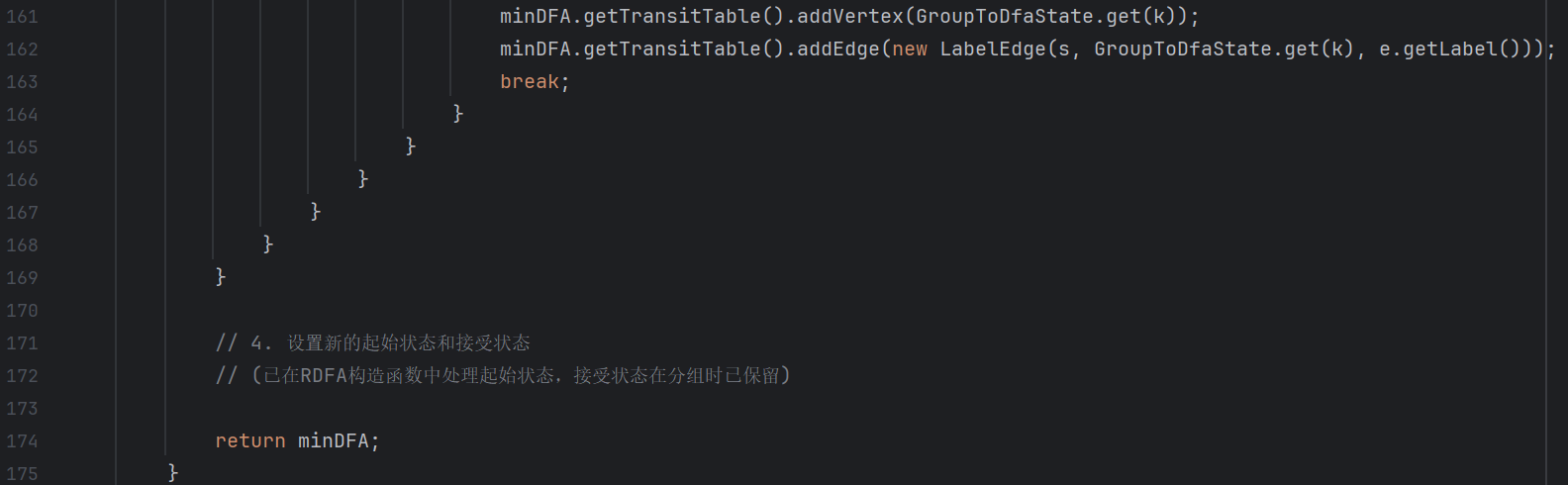




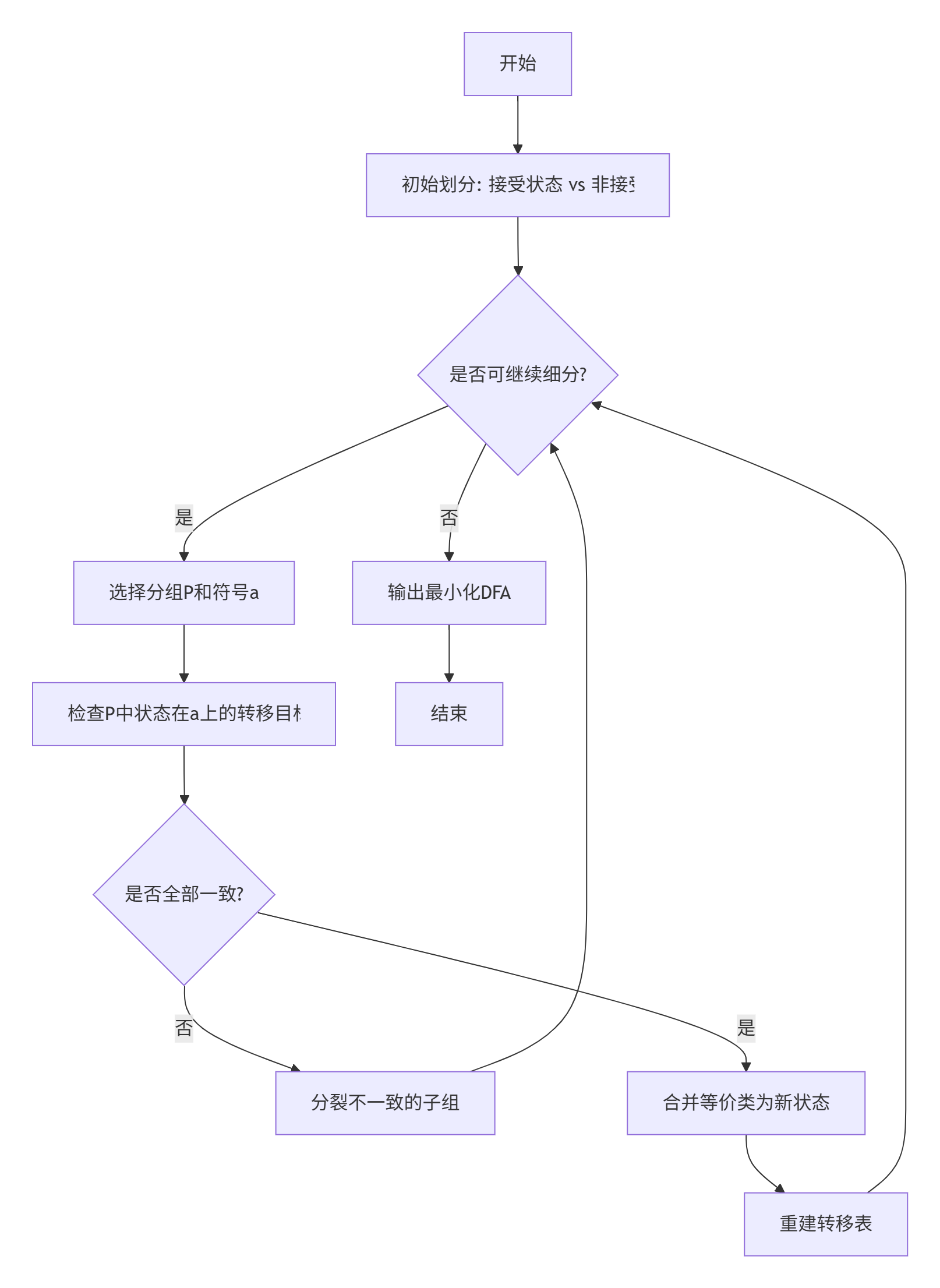
② 合并等价状态





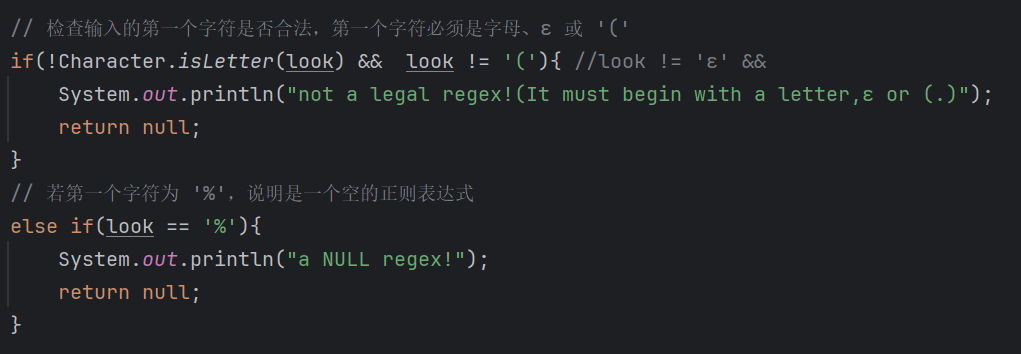


算法流程图介绍：

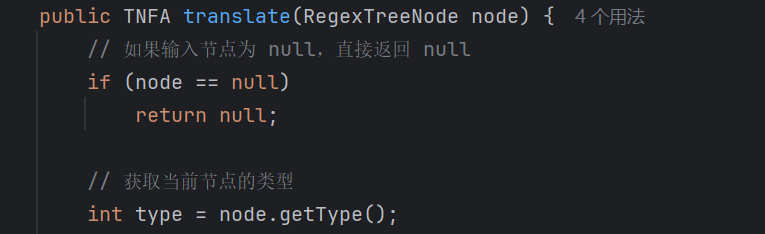


3. 3 错误处理

（1）正则表达式解析错误

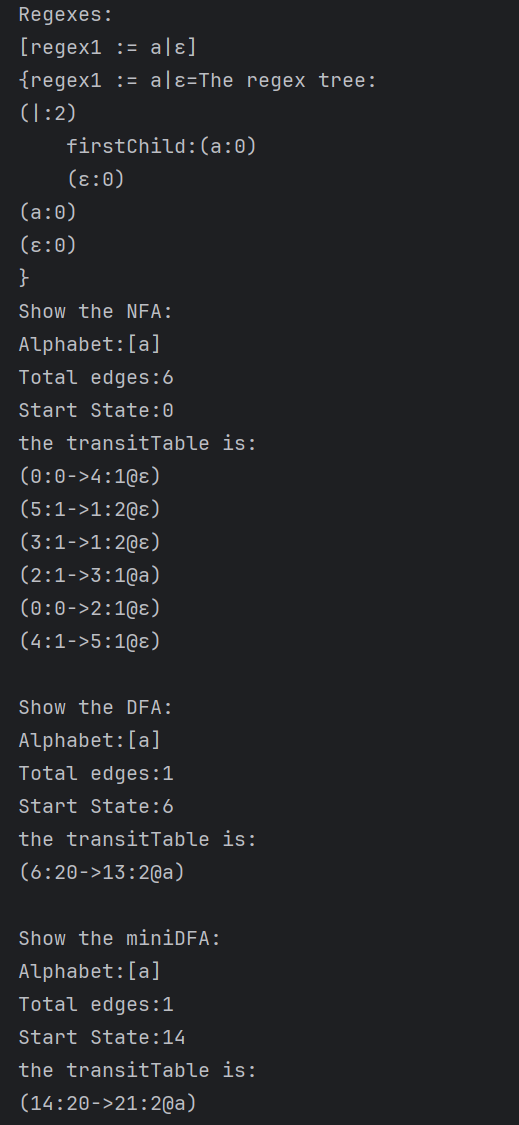


（2）NFA构建异常

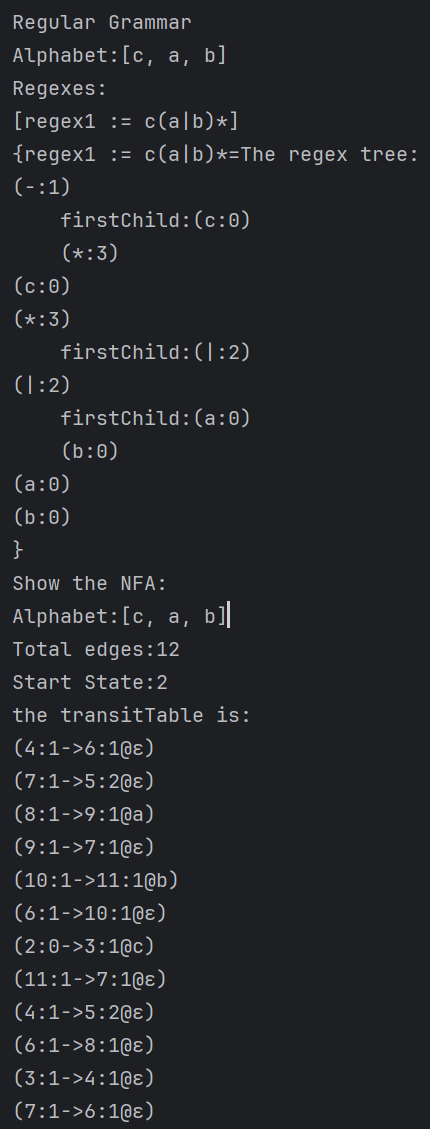


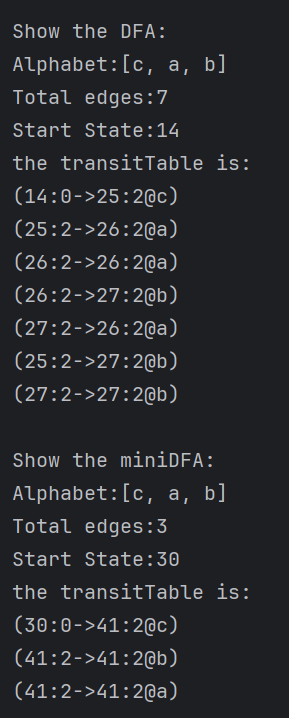
3.4 运行结果

（1）测试案例：a|ε

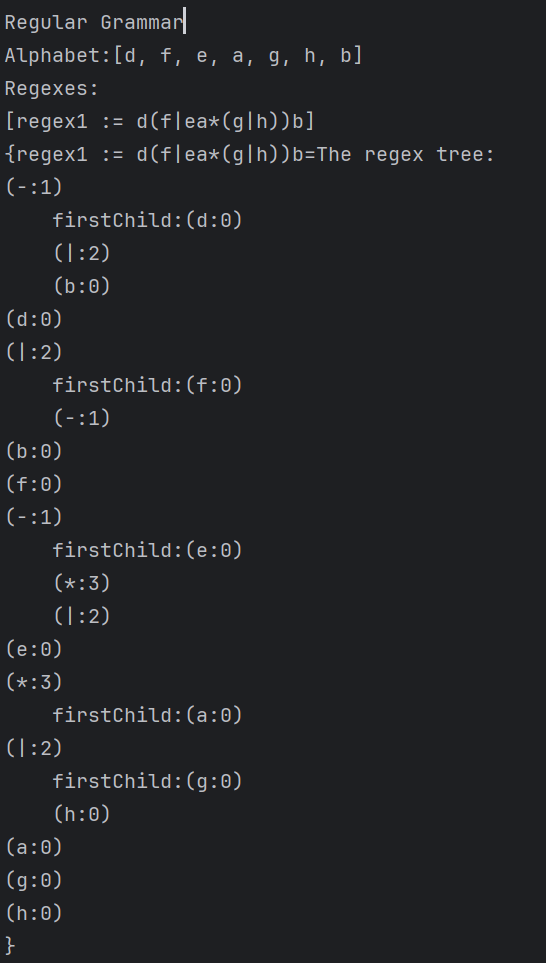


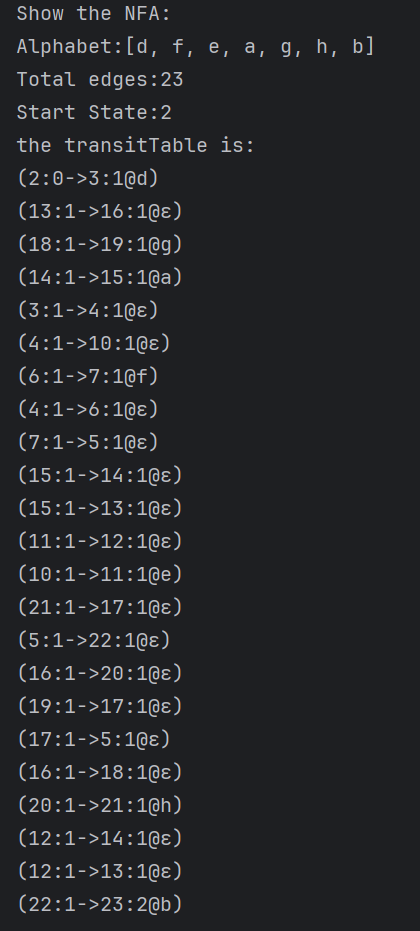
（2）测试案例：c(a|b)\*

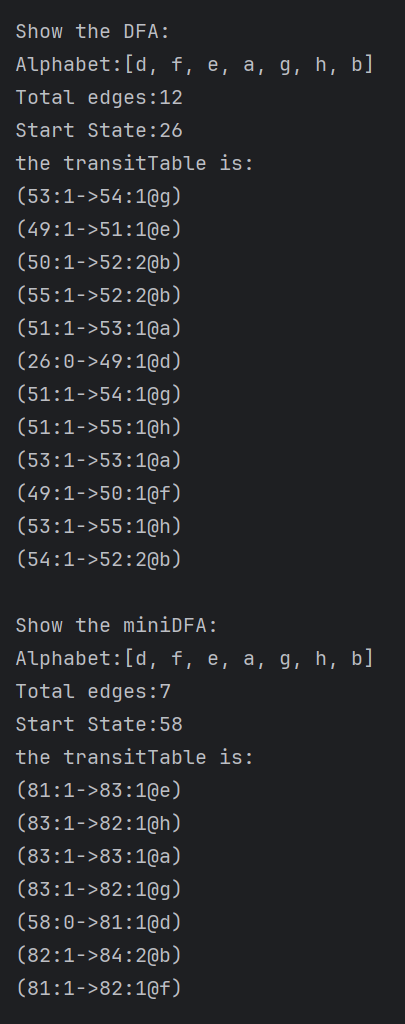




（3）测试案例：







4、总结

　这次实验让我真正理解了正则表达式引擎的工作原理，子集构造法虽然理论清晰，但实现时发现ε闭包的计算特别容易出错，虽然调试过程比较痛苦，但最终看到NFA成功转换成最小DFA的那一刻还是很有成就感的，看到复杂正则表达式生成的DFA状态数明显减少的过程。当前实现在处理更复杂的正则时会比较慢，建议后续可以优化下状态合并的逻辑，处理复杂表达式时效率会更高。