**编译原理实验二实验报告**

计科 MisakaCJQ

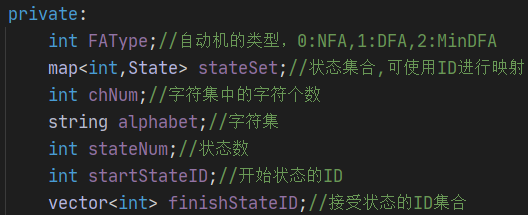
1. **实验目的**

学习和掌握将NFA转为DFA的子集构造法。

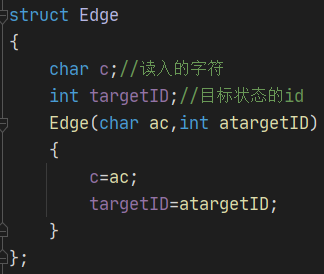
学会编程实现等价划分法最小化DFA

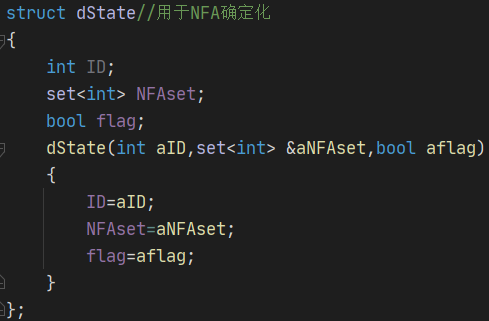
1. **实验任务**
2. 存储NFA与DFA
3. 编程实现子集构造法将NFA转换成DFA。
4. 先完善DFA，再最小化DFA。
5. **实验内容**
6. 存储NFA于DFA

定义一个自动机类FA，私有数据域：

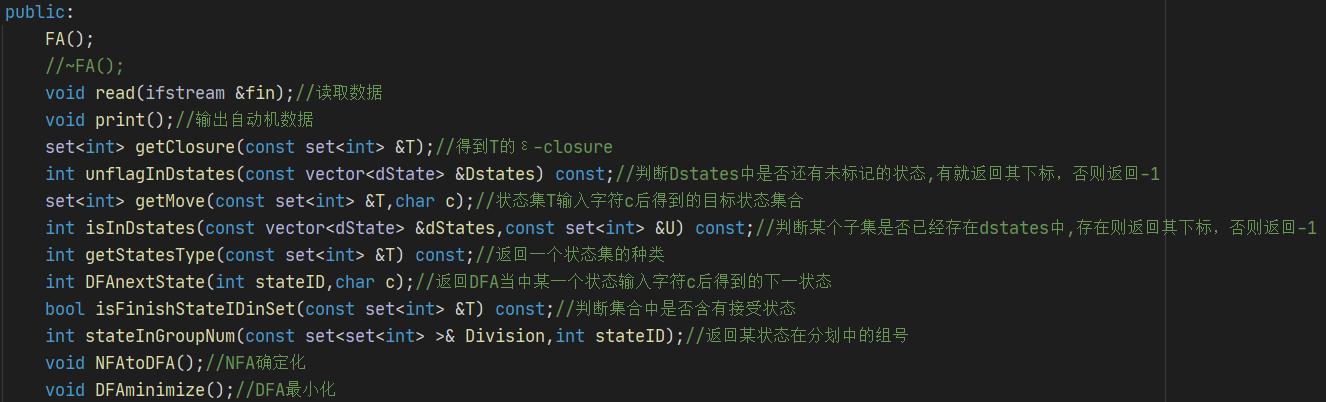


四种结构体：状态，出边，NFA确定化辅助状态，DFA最小化辅助结点

公有接口：



其中主要的函数有：

*getMove*:用以返回一个状态集合在输入一个字符后转向的下一个状态的集合

*getClosure*:计算一个状态集的ε-closure

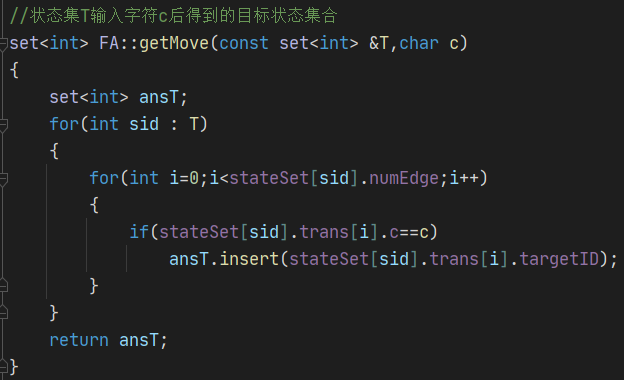
*NFAtoDFA*:执行NFA确定化操作。

*DFAminimize*:执行DFA最小化操作

其余的除了read和print的函数均为以上四个函数中部分过程的模块化。

1. 主要函数说明

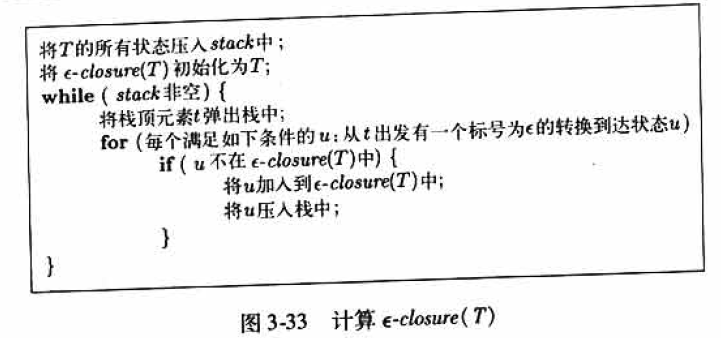
*getMove*:

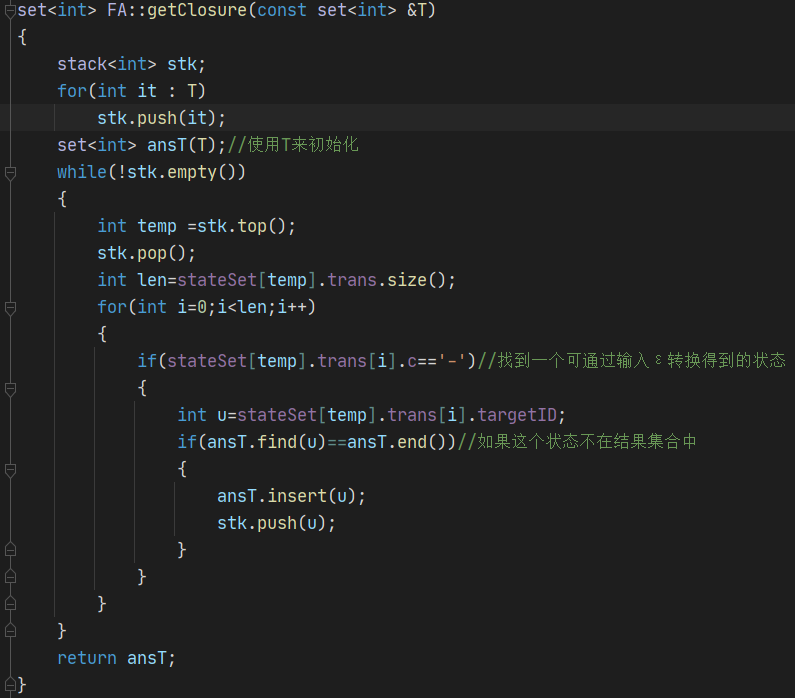


逐个遍历状态集T中的状态，对它们的出边进行遍历找到输入字符为c的状态放入结果集合ansT，最后返回。

*getClosure*:

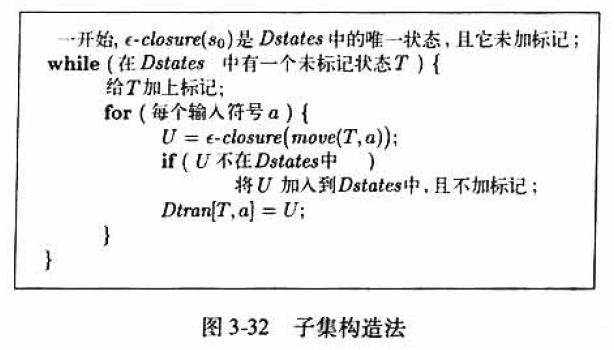
依照课本提供的如下算法实现

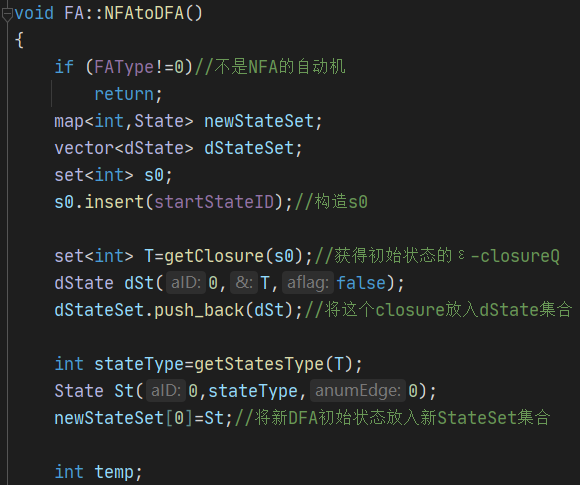




*NFAtoDFA*:

依照课本上的子集构造算法伪代码实现



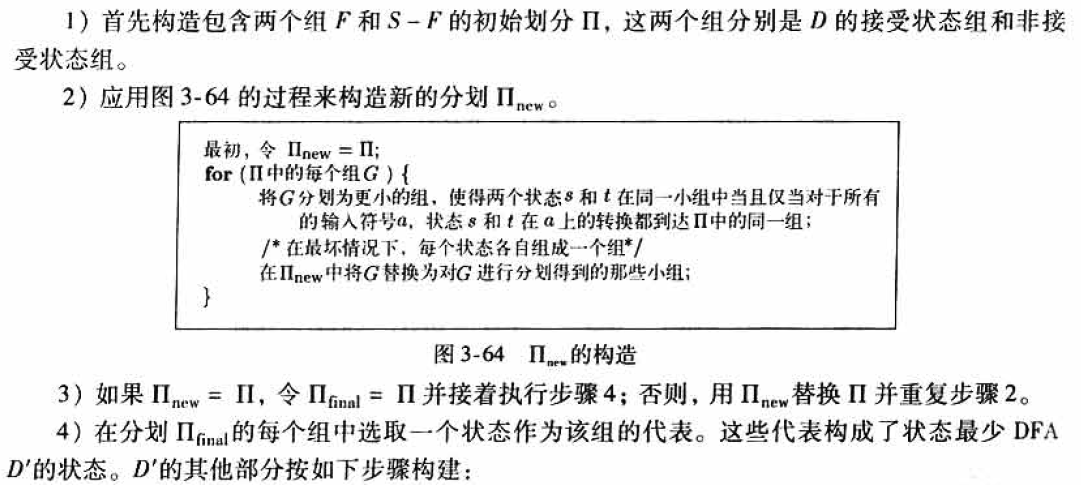


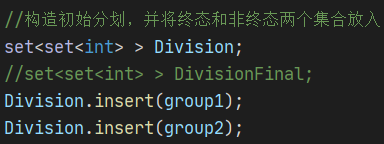


以上为这一操作的关键部分代码，其余部分为新数据的构造部分，略过。

*DFAminimize:*

依照课本上的分划构造算法伪代码实现





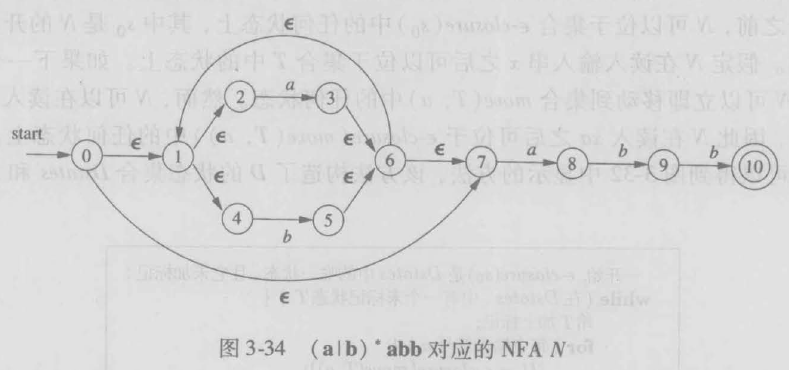




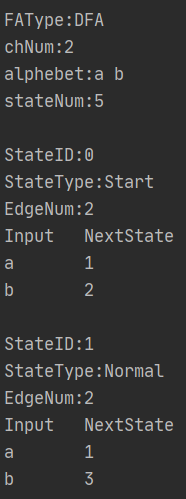
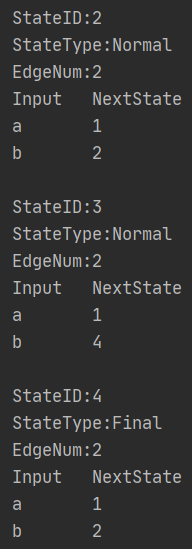
以上为关键部分代码。其中子集的递归分划使用一个队列进行BFS来实现，在每一个深度输入一个字符对子集进行分化，分划出的新子集又加入队列，在到达叶子节点时停止搜索，此时队列中剩下的为所有分划好的集合。这样编写代码使得可读性大大提高。

1. 样例测试与输出

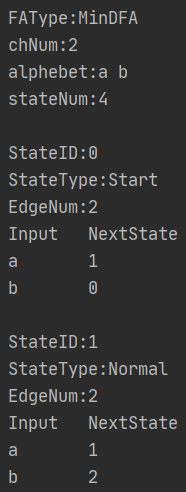
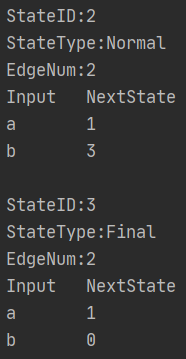
1）(a|b)\*abb



进行NFA确定化后得到一个含有5个状态的DFA：

进行DFA最小化后得到一个含有4个状态的DFA:

2) (aa|b)\*(a|bb)\*

a

a

b

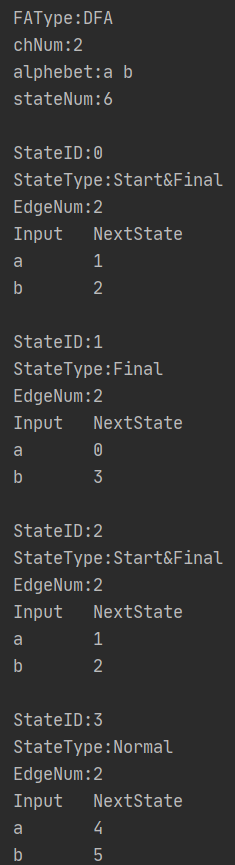
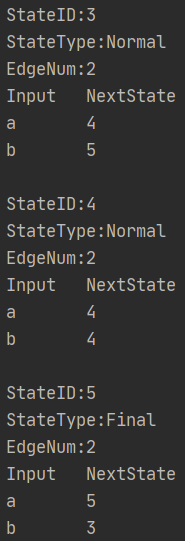
b

b

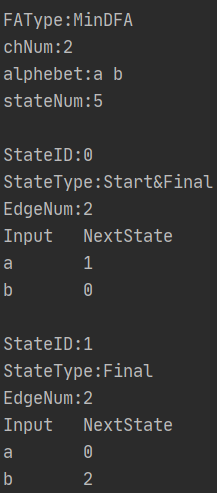
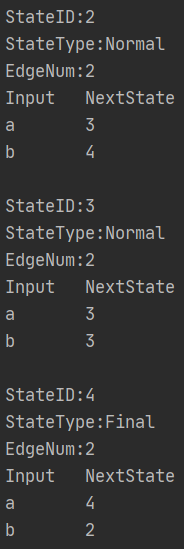
a

ε

进行NFA确定化后得到一个含有6个状态的DFA

进行DFA最小化后得到一个含有5个状态的DFA:

1. **实验总结**

这一次实验的难点主要在于子集构造法和DFA最小化当中一些对集合的操作，在这里我借助了C++set和一些函数的封装来尽量抽象和简化了对集合的操作。另一个难点在于数据结构的设计，在这里我基本上时边写边设计的。通过这一次实验我更深刻地了解了NFA确定化和DFA最小化两种算法。