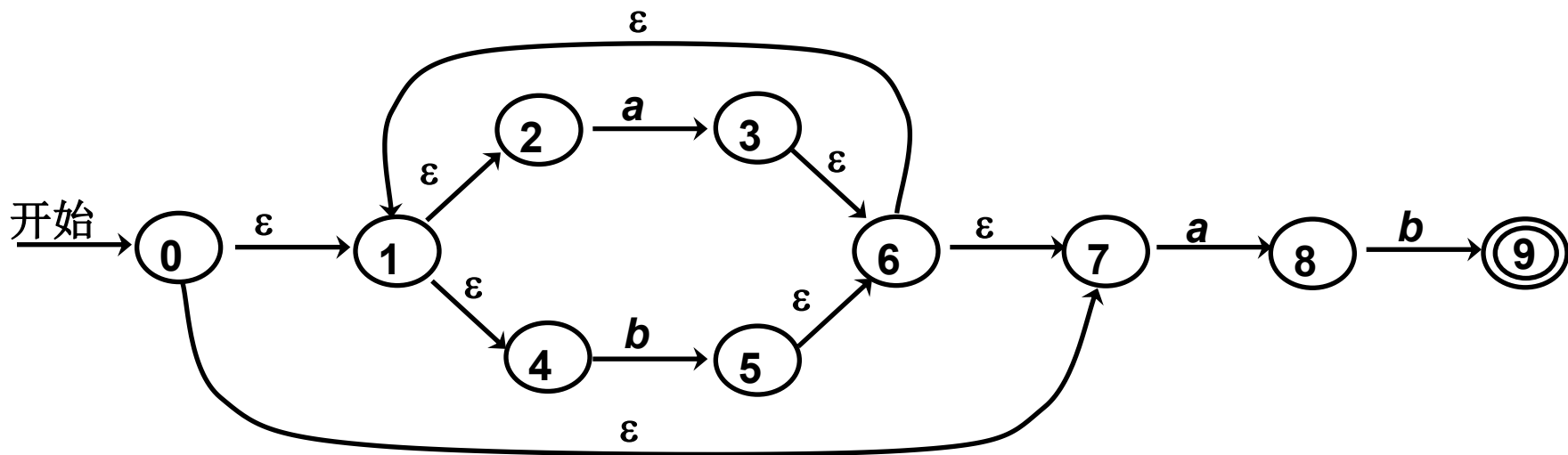
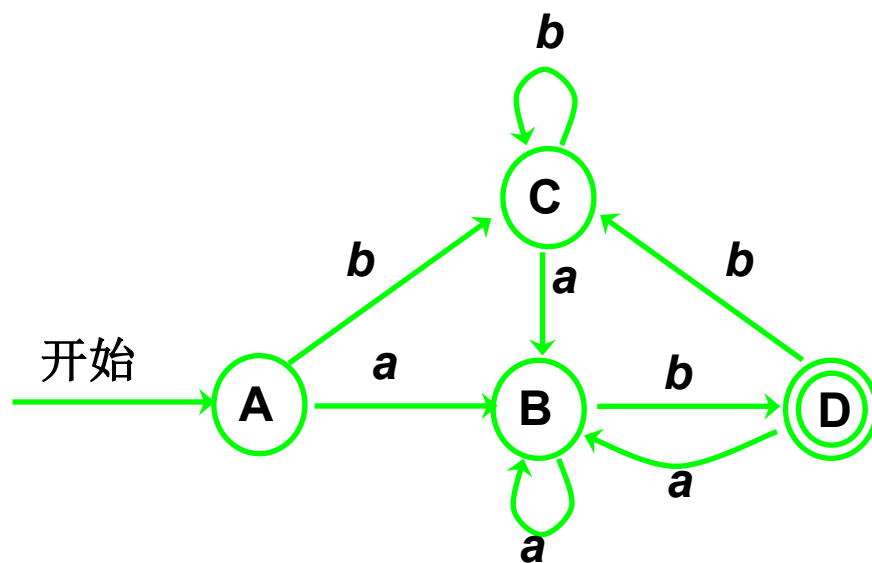


C语言实现NFA \rightarrow DFA



输入： NFA

| 状态 | a | b | ϵ |
|----|---|---|------------|
| 0 | | | 1,7 |
| 1 | | | 2,4 |
| 2 | 3 | | |
| 3 | | | 6 |
| 4 | | 5 | |
| 5 | | | 6 |
| 6 | | | 1,7 |
| 7 | 8 | | |
| 8 | | 9 | |
| 9 | | | |



输出: **DFA**

| 状态 | a | b |
|----|---|---|
| A | B | C |
| B | B | D |
| C | B | C |
| D | B | C |

NFA \rightarrow DFA

- 输入配置文件：NFA状态转换表
- 输出：DFA状态转换表
- 状态转换表的格式
 - 状态数（状态以数字表示，如0,1,2...）
 - 输入字母表
 - 状态转换二维表
 - 接受状态集合
 - 例：
 - 10, a, b, ϵ
 - void; void; 1,7
 - void; void; 2,4
 - 3; void; void
 - ...
 - 9

NFA \rightarrow DFA

- 方法
 - ϵ -closure集合的构造
 - 状态转换表的构造

NFA \rightarrow DFA

- ϵ -closure集合的构造

把 T 的所有状态压入栈;

ϵ -closure(T) 的初值置为 T ;

while 栈非空 do begin

 把栈顶元素 t 弹出栈;

 for 每个状态 u (条件是从 t 到 u 的边上的标记为 ϵ) do

 if u 不在 ϵ -closure(T) 中 do begin

 把 u 加入 ϵ -closure(T);

 把 u 压入栈

 end

end

NFA \rightarrow DFA

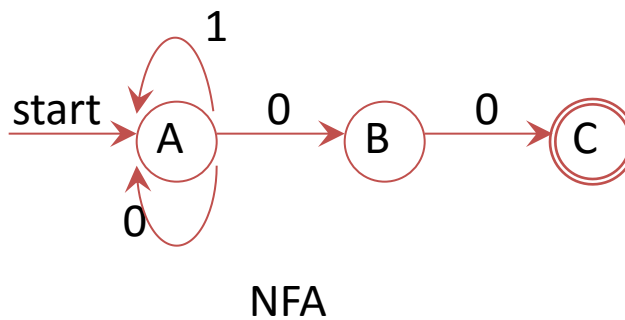
- 状态转化表的构造

```
初始,  $\epsilon - \text{closure}(s)$  是  $Dstates$  仅有的状态, 并且尚未标记;  
while  $Dstates$  有尚未标记的状态  $T$  do begin  
    标记  $T$ ;  
    for 每个输入符号  $a$  do begin  
         $U := \epsilon - \text{closure}(\text{move}(T, a))$ ;  
        if  $U$  不在  $Dstates$  中 then  
            把  $U$  作为尚未标记的状态加入  $Dstates$ ;  
         $Dtran[T, a] := U$   
    end  
end  
end
```

NFA \rightarrow DFA

- 测试1

— 输入: NFA

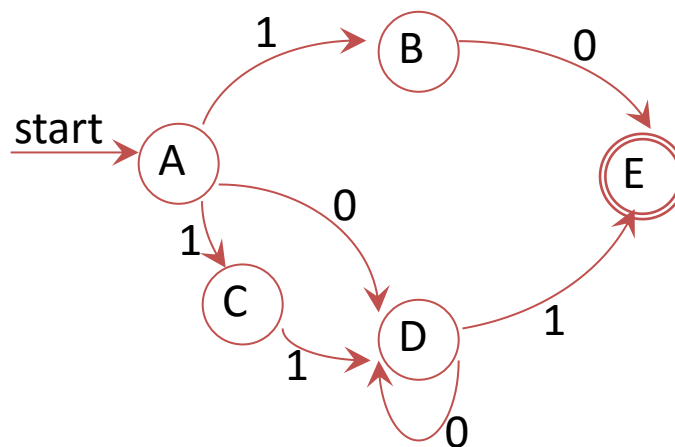


— 输出DFA

?

NFA \rightarrow DFA

- 测试2
 - 输入: NFA



NFA

— 输出DFA

?