实例化如图 1 所示的 DFA，测试代码如代码 1 所示

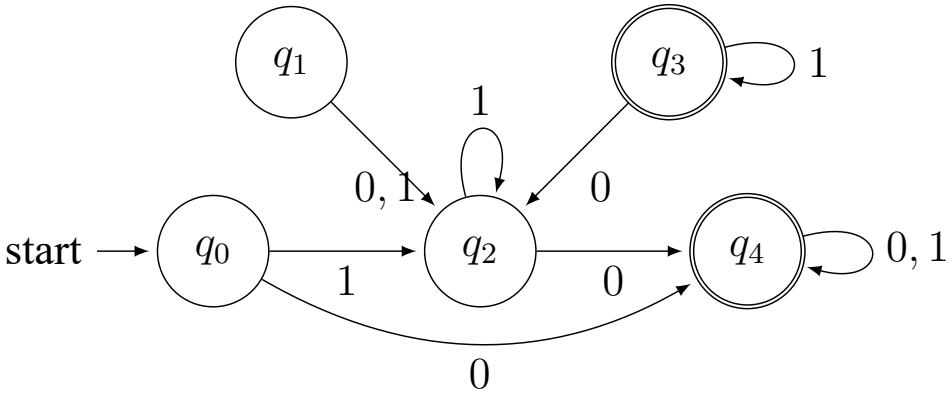

图 1

```
void minDFATest3()
{
  DFA_components dfa_com1;

  // StateSet S  开始状态集
  dfa_com1.S.set_domain(5);
  dfa_com1.S.add(0);

  // StateSet F  结束状态集
  dfa_com1.F.set_domain(5);
  dfa_com1.F.add(3);
  dfa_com1.F.add(4);

  int i = 5;
  while (i--)
  {
    dfa_com1.Q.allocate();
  }

  dfa_com1.T.set_domain(5);
  dfa_com1.T.add_transition(0, '0', 4);
  dfa_com1.T.add_transition(0, '1', 2);
  dfa_com1.T.add_transition(1, '0', 2);
  dfa_com1.T.add_transition(1, '1', 2);
  dfa_com1.T.add_transition(2, '0', 4);
  dfa_com1.T.add_transition(2, '1', 2);
  dfa_com1.T.add_transition(3, '0', 2);
  dfa_com1.T.add_transition(3, '1', 3);
  dfa_com1.T.add_transition(4, '0', 4);
  dfa_com1.T.add_transition(4, '1', 4);

  //实例化一个DFA对象
  DFA dfa1(dfa_com1);
  cout << "\n************ DFA\n" << std::flush;
```

```
35   cout << dfa1 << endl;
36
37   dfa1.usefulf();  // 没有删除1,3 ？
38   cout << dfa1 << endl;
39   cout << " is the DFA Usefulf ?: " << dfa1.Usefulf() << endl;
40
41   dfa1.min_Hopcroft();
42   cout << "\n************ minDFA\n" << std::flush;
43   cout << dfa1 << endl;
44 }
```

代码 1: 图 1 中的 DFA

代码 1 将输出如下信息

```
1  ************ DFA
2
3  DFA
4  Q = [0,5)
5  S = { 0 }
6  F = { 3  4 }
7  Transitions =
8  0->{ '0'->4  '1'->2 }
9  1->{ ['0','1']->2 }
10 2->{ '0'->4  '1'->2 }
11 3->{ '0'->2  '1'->3 }
12 4->{ ['0','1']->4 }
13
14 current = -1
15
16
17 DFA
18 Q = [0,5)
19 S = { 0 }
20 F = { 3  4 }
21 Transitions =
22 0->{ '0'->4  '1'->2 }
23 1->{ ['0','1']->2 }
24 2->{ '0'->4  '1'->2 }
25 3->{ '0'->2  '1'->3 }
26 4->{ ['0','1']->4 }
27
28 current = -1
29
30  is the DFA Usefulf ?: 1
31
32 ************ minDFA
33
34 DFA
35 Q = [0,4)
36 S = { 0 }
```

```
F = { 2  3 }
Transitions =
0->{ '0'->3  '1'->0 }
1->{ ['0','1']->0 }
2->{ '0'->0  '1'->2 }
3->{ ['0','1']->3 }
current = -1
```

代码 2: 图 1 中的 DFA 在算法 Hopcroft 算法中的输出

本例中，q_1 和 q_3 都不是 `final-unreachable` （陷阱）状态，所以不会在执行函数 `DFA::usefulf()` 后去除。

执行最小化算法 `DFA::min_Hopcroft` 后的 DFA 如图 2 所示。

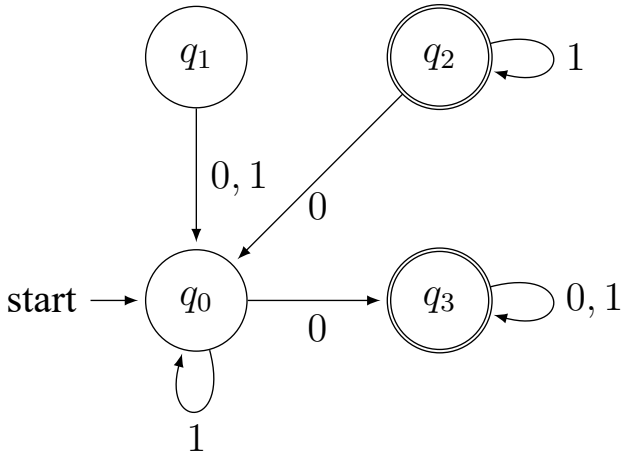

图 2

经过测试，算法`DFA::min_dragon`, `DFA::min_Watson`, `DFA::min_HopcroftUllman`的输出与算法`DFA::min_Hopcroft`相同。而算法`DFA::min_Brzozowski`的输出为

```
************ DFA
DFA
Q = [0,2)
S = { 0 }
F = { 1 }
Transitions =
0->{ '0'->1  '1'->0 }
1->{ ['0','1']->1 }

current = -1
```

代码 3: 图 1 中的 DFA 在算法 Hopcroft 算法中的输出

对应的状态转移图为图 3

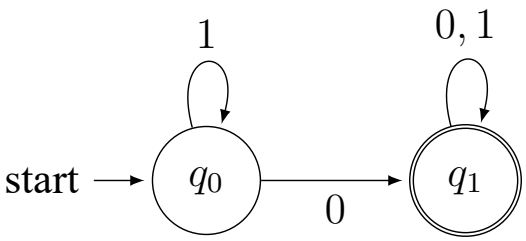

图 3

结论

对于图 1 中的自动机，仅有算法 DFA::min_Brzozowski输出了正确的结果，DFA::min_Hopcroft无论是否经过修改，均输出错误结果。

对于图 1 中的自动机, 移除状态 q_3 之后，之前错误的算法也能输出正确结果。

注 0.1. 除了算法DFA::min_Brzozowski外，其他算法均不能移除自动机中的开始不可达状态。

增加对其他算法的测试后的代码如下

```
void minDFATest3()
{
  DFA_components dfa_com1;

  // StateSet S  开始状态集
  dfa_com1.S.set_domain(5);
  dfa_com1.S.add(0);

  // StateSet F  结束状态集
  dfa_com1.F.set_domain(5);
  dfa_com1.F.add(3);
  dfa_com1.F.add(4);

  int i = 5;
  while (i--)
  {
    dfa_com1.Q.allocate();
  }

  dfa_com1.T.set_domain(5);
  dfa_com1.T.add_transition(0, '0', 4);
  dfa_com1.T.add_transition(0, '1', 2);
  dfa_com1.T.add_transition(1, '0', 2);
  dfa_com1.T.add_transition(1, '1', 2);
  dfa_com1.T.add_transition(2, '0', 4);
  dfa_com1.T.add_transition(2, '1', 2);
  dfa_com1.T.add_transition(3, '0', 2);
  dfa_com1.T.add_transition(3, '1', 3);
```

```
29    dfa_com1.T.add_transition(4, '0', 4);
30    dfa_com1.T.add_transition(4, '1', 4);
31
32    //实例化一个DFA对象
33    DFA dfa1(dfa_com1);
34    cout << "\n************ DFA\n" << std::flush;
35    cout << dfa1 << endl;
36
37    cout << " is the DFA Usefulf ?: " << dfa1.Usefulf() << endl;
38    dfa1.usefulf();  // 没有删除1,3 ？
39    cout << dfa1 << endl;
40    cout << " is the DFA Usefulf ?: " << dfa1.Usefulf() << endl;
41
42    dfa1.min_Hopcroft();
43    cout << "\n************ minDFA (Hopcroft) \n" << std::flush;
44    cout << dfa1 << endl;
45
46
47    //增加对其他最小化算法的测试
48    DFA dfa2(dfa_com1);
49    dfa2.min_Brzozowski();
50    cout << "\n************ minDFA (Brzozowski)\n" << std::flush;
51    cout << dfa2 << endl;
52
53    DFA dfa3(dfa_com1);
54    dfa3.usefulf();
55    dfa3.min_dragon();
56    cout << "\n************ minDFA (dragon) \n" << std::flush;
57    cout << dfa3 << endl;
58
59    DFA dfa4(dfa_com1);
60    dfa4.usefulf();
61    dfa4.min_HopcroftUllman();
62    cout << "\n************ minDFA (HopcroftUllman) \n" << std::flush;
63    cout << dfa4 << endl;
64
65    DFA dfa5(dfa_com1);
66    dfa5.usefulf();
67    dfa5.min_Watson();
68    cout << "\n************ minDFA (Watson)\n" << std::flush;
69    cout << dfa5 << endl;
70 }
```