测试报告

1　分析错误原因

对于给出的正则表达式 **((+|-)?dd*)(.dd*)?((E|e)((+|-)?dd*))?** 提交上去的程序运行出错闪退，原因有两点：

- 完成了实验二的选做内容正闭包（+），导致程序无法正确区分字符'+'与正闭包运算符（+）
- 在对原正则表达式添加显式的连接符号时使用了小数点（.），导致无法正确识别题目中带小数点的正则表达式

2　修改正则表达式

根据以上两点错误原因，我将以上正则表达式进行以下两点修改：

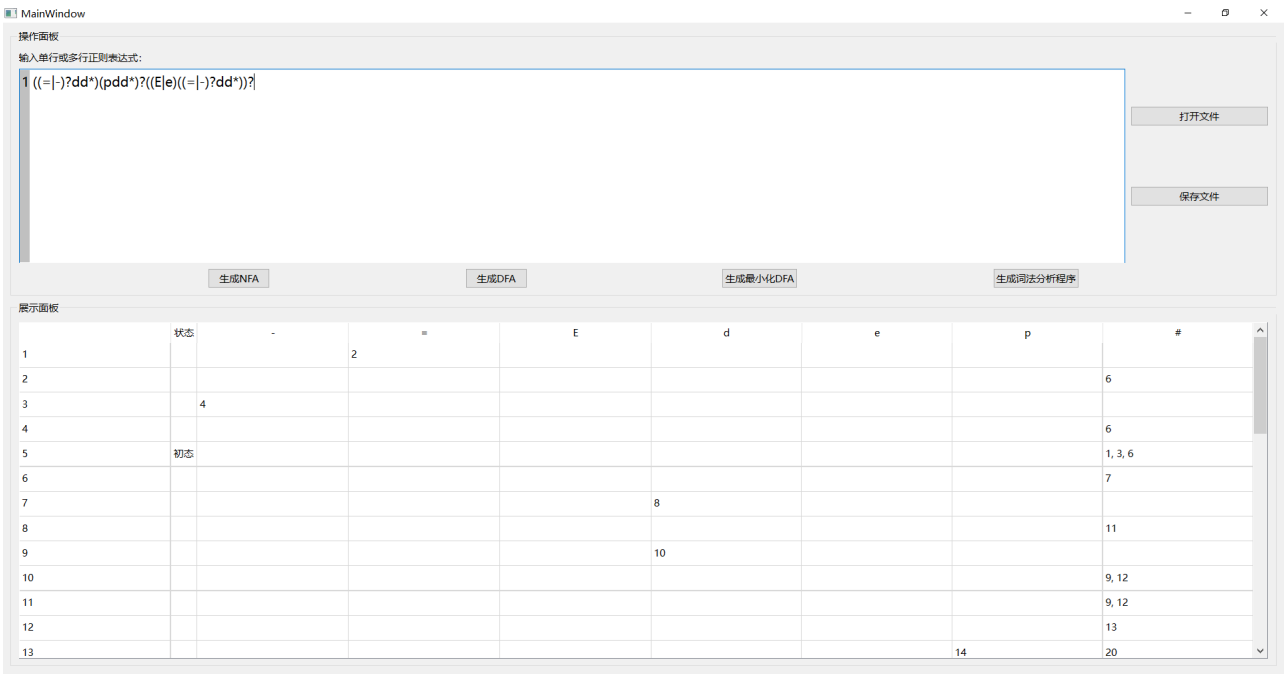
- 将原表达式中的（+）修改为（=）
- 将原表达式中的（.）修改为（**p**）

修改后的正则表达式为 **((=|-)?dd*)(pdd*)?((E|e)((=|-)?dd*))?** ，修改后程序能正常运行。

3　运行结果及分析

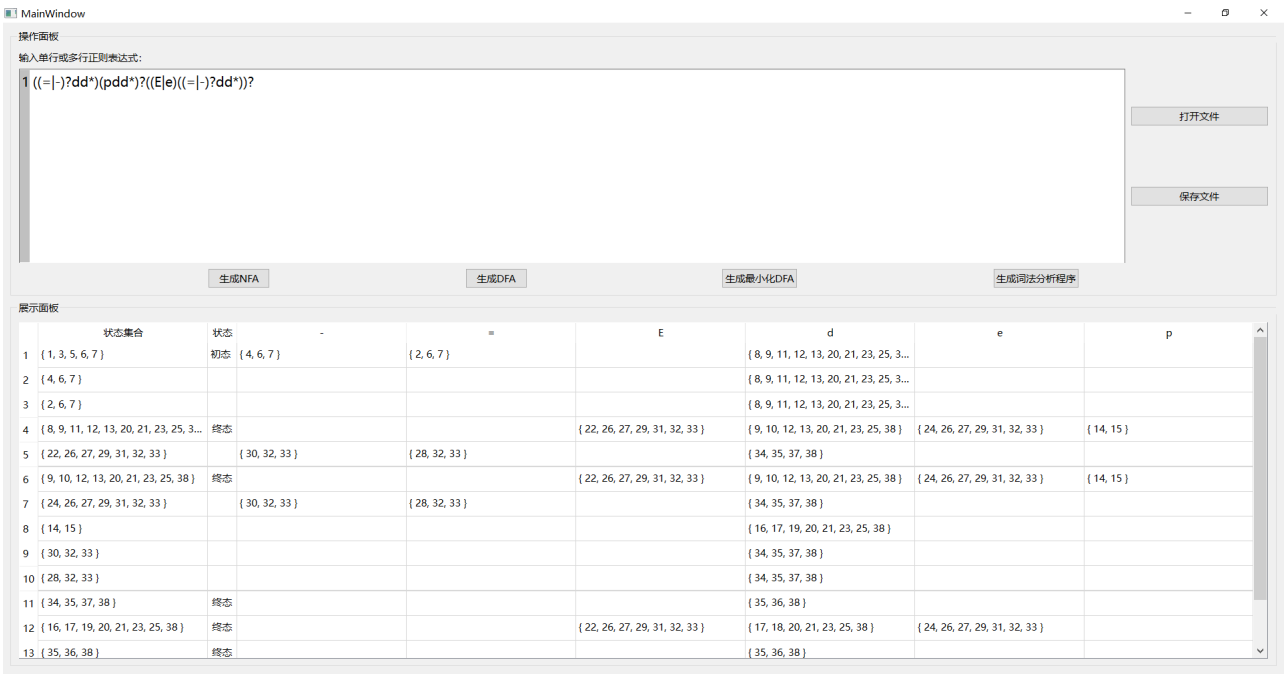
程序运行结果如下图：

3.1　生成NFA



每个字符在**NFA**中需要两个状态表示，选择运算（|）与闭包运算（*）均需要额外的两个状态来表示，可选运算（？）不需要添加额外的状态。该正则表达式总共有 **13** 个字符，**3** 个选择运算，**3** 个闭包运算，需要的状态数为 $(13+3+3)\times 2=38$ 个，而程序生成的**NFA**正好包含 **38** 个状态，可以初步判断程序生成了正确的**NFA**。

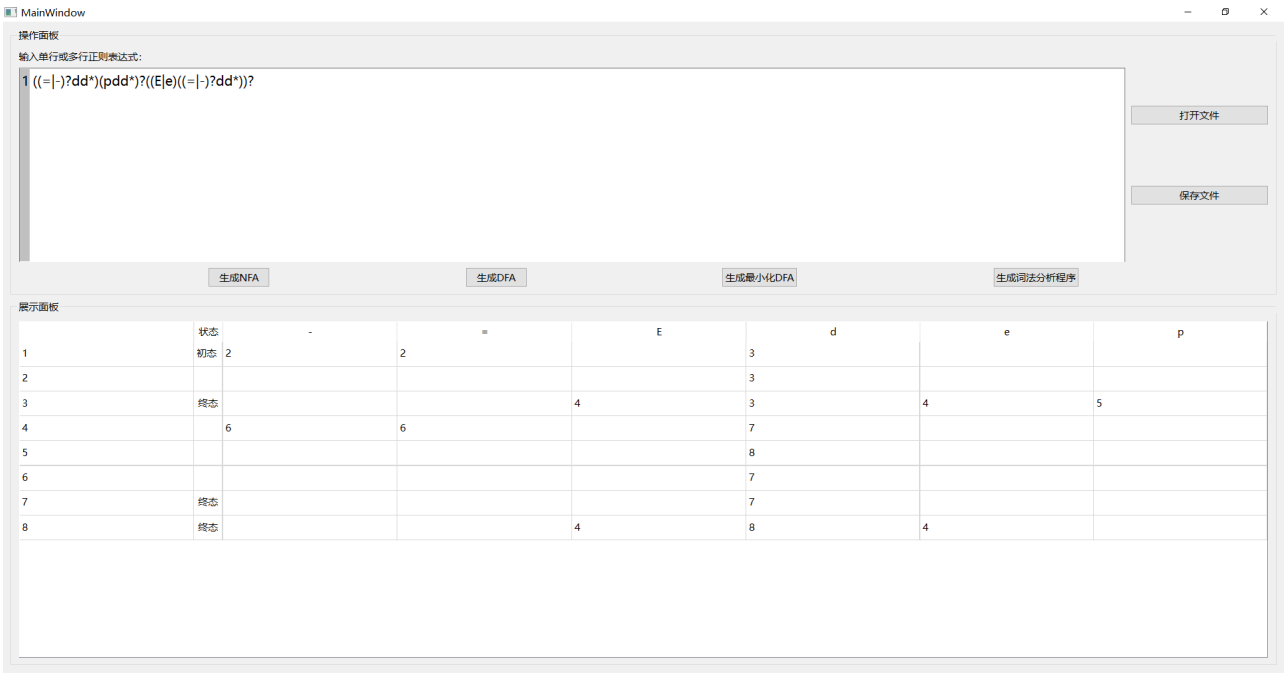
3.2 生成DFA



	状态集合	状态	-	=	E	d	e	p
1	{ 1, 3, 5, 6, 7 }	初态	{ 4, 6, 7 }	{ 2, 6, 7 }		{ 8, 9, 11, 12, 13, 20, 21, 23, 25, 3...		
2	{ 4, 6, 7 }					{ 8, 9, 11, 12, 13, 20, 21, 23, 25, 3...		
3	{ 2, 6, 7 }					{ 8, 9, 11, 12, 13, 20, 21, 23, 25, 3...		
4	{ 8, 9, 11, 12, 13, 20, 21, 23, 25, 3...	终态			{ 22, 26, 27, 29, 31, 32, 33 }	{ 9, 10, 12, 13, 20, 21, 23, 25, 38 }	{ 24, 26, 27, 29, 31, 32, 33 }	{ 14, 15 }
5	{ 22, 26, 27, 29, 31, 32, 33 }		{ 30, 32, 33 }	{ 28, 32, 33 }		{ 34, 35, 37, 38 }		
6	{ 9, 10, 12, 13, 20, 21, 23, 25, 38 }	终态			{ 22, 26, 27, 29, 31, 32, 33 }	{ 9, 10, 12, 13, 20, 21, 23, 25, 38 }	{ 24, 26, 27, 29, 31, 32, 33 }	{ 14, 15 }
7	{ 24, 26, 27, 29, 31, 32, 33 }		{ 30, 32, 33 }	{ 28, 32, 33 }		{ 34, 35, 37, 38 }		
8	{ 14, 15 }					{ 16, 17, 19, 20, 21, 23, 25, 38 }		
9	{ 30, 32, 33 }					{ 34, 35, 37, 38 }		
10	{ 28, 32, 33 }					{ 34, 35, 37, 38 }		
11	{ 34, 35, 37, 38 }	终态				{ 35, 36, 38 }		
12	{ 16, 17, 19, 20, 21, 23, 25, 38 }	终态			{ 22, 26, 27, 29, 31, 32, 33 }	{ 17, 18, 20, 21, 23, 25, 38 }	{ 24, 26, 27, 29, 31, 32, 33 }	
13	{ 35, 36, 38 }	终态				{ 35, 36, 38 }		

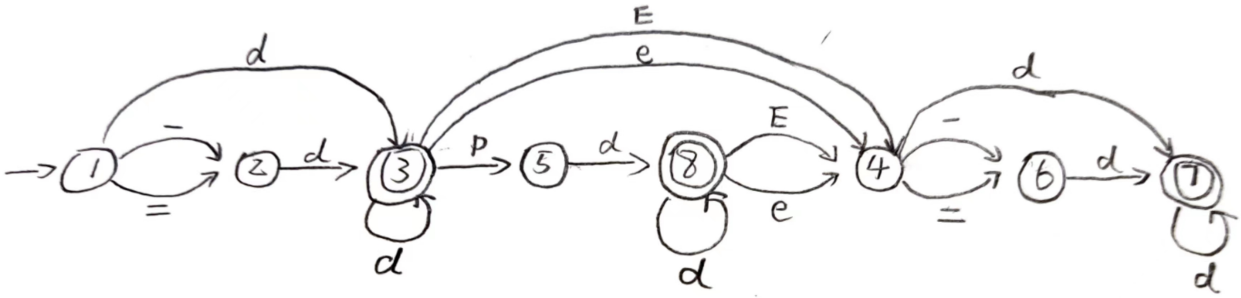
程序能将**NFA**的等价节点用集合表示并输出。

3.3 生成最小化DFA



	状态	-	=	E	d	e	p
1	初态	2	2		3		
2					3		
3	终态			4	3	4	5
4		6	6		7		
5					8		
6					7		
7	终态				7		
8	终态			4	8	4	

根据生成的最小**DFA**，画出状态转换图如下：



与浮点数的**DFA**图作对比：

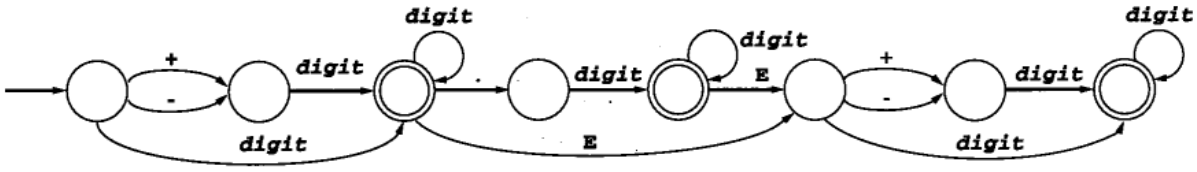


图2-3　浮点数的有穷自动机

可以看出是一致的，因此表明程序生成了正确的最小化**DFA**。

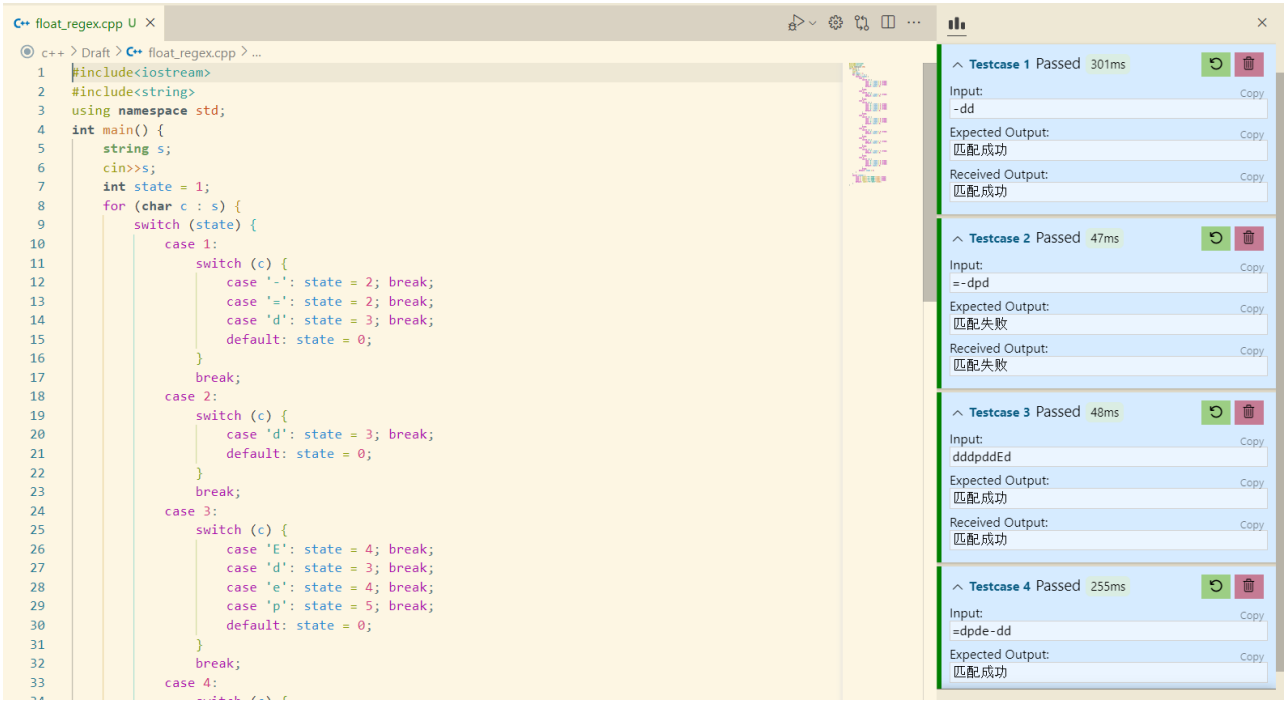
3.4　生成词法分析程序

XLEX

```
#include<iostream>
#include<string>
using namespace std;
int main() {
    string s;
    cin>>s;
    int state = 1;
    for (char c : s) {
        switch (state) {
            case 1:
                switch (c) {
                    case '-': state = 2; break;
                    case '=': state = 2; break;
                    case 'd': state = 3; break;
                    default: state = 0;
                }
                break;
            case 2:
                switch (c) {
                    case 'd': state = 3; break;
                    default: state = 0;
                }
                break;
            case 3:
                switch (c) {
                    case 'E': state = 4; break;
                    case 'd': state = 3; break;
                    case 'e': state = 4; break;
                    case 'p': state = 5; break;
                    default: state = 0;
                }
                break;
            case 4:
                switch (c) {
```

保存

将其保存为 cpp 文件，测试运行：



可以看出程序能正确地对浮点数进行识别。