基于深度优先搜索实现火柴数学题自动解题系统

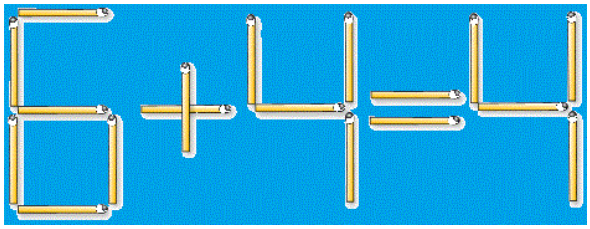
班级：自 73 班

学号：2017011507

姓名：陈昱宏

一、任务描述：

火柴棒数学题是一个非常经典的问题，在小学的奥数中经常出现，如下图所示，该问题是给一个不合理的数学等式或已经成立的数学等式，要求在移动一根或其他限定根数的条件下，使数学表达式的等式成立或转换成新的成立表达式。



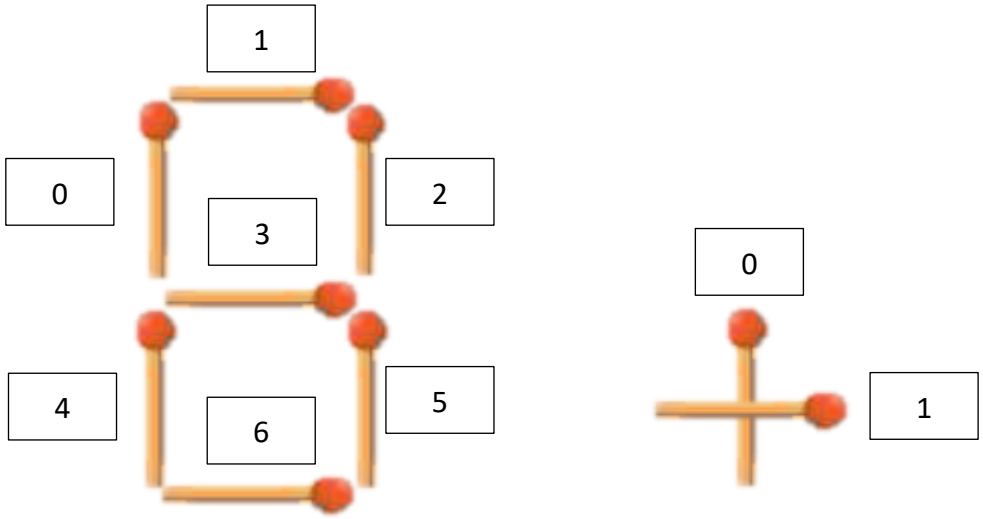
二、问题建模：

我们可以把一个数字分成七段，由七根火柴组成，即电子电路元件—七段数码管，符号则用两根火柴表示（依然使用七个数字表示，但有效位只有两位），如此一来我们可以将一个数学表达式，转换成一个N * 7的数组，可以透过改变相对应火柴的值来控制是否存在这根火柴。

以下为我对于本问题的状态表示：

（1）状态表示：一个长度为 13 的数组，数组内每个元素为长度 7 的字符串，用 1 表示该火柴存在，0 表示该火柴不存在。对于符号位相应的冗余项，第三个字符表示符号的值（2 代表+，3 代表-，4 代表*，5 代表=），第四个字符用来表示转换操作中的目标符号（2 代表+，4 代表*），其余项为 6。火柴顺序如下

图所示：



（2）转换操作：ADD(i, j, sign)代表在第i位的第j个火柴的位置添加火柴，如果符号位添加时有多种变换可能，用 sign 来控制变换后的符号；SUB(i, j, sign)代表在第i位的第j个火柴的位置添加火柴，如果符号位减少时有多种变换可能，用 sign 来控制变换后的符号。

根据上述的建模，以下将利用建模来示范深度搜索的状态转换过程（以$6+4=4$且只能移动一根为例）：

{"0000000", "0000000", "1101111", "1126666",
"0000000", "0000000", "1011010", "1156666",
"0000000", "0000000", "0000000", "0000000", "1011010"}

$\xrightarrow{SUB(2,3,no\ sign)}$ {"0000000", "0000000", "1100111", "1126666",
"0000000", "0000000", "1011010", "1156666",
"0000000", "0000000", "0000000", "0000000", "1011010"}

$\xrightarrow{ADD(2,2,no\ sign)}$ {"0000000", "0000000", "1110111", "1126666",
"0000000", "0000000", "1011010", "1156666",
"0000000", "0000000", "0000000", "0000000", "1011010"}

如上步骤就可以搜索出其中一个答案$0+4=4$。

三、编程语言与环境设置：

本项目以 C#语言进行编写，使用 Visual Studio 2019 进行编译，.Net Framework 版本为 4.7.2，使用函数皆为标准库，无需安装特定库函数。

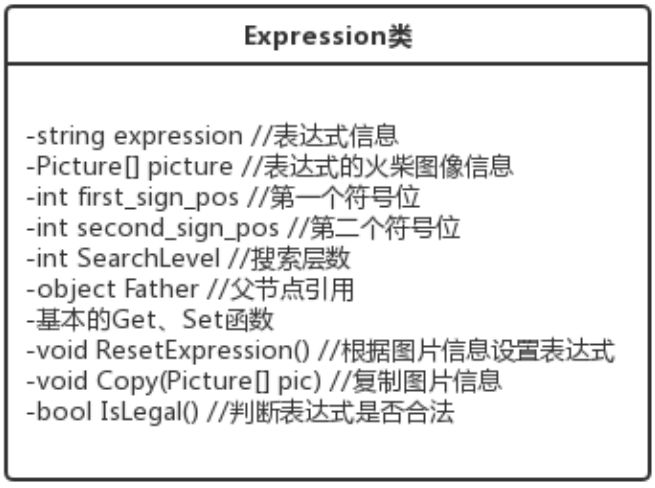
四、算法设计与实现：

（一）数据结构介绍：

本项目采用面向对象的方法实现，除了基本的窗体类之外，自己还设计了两个类，分别为：Picture 类和 Expression 类。

1. Picture 类：

2. Expression 类：



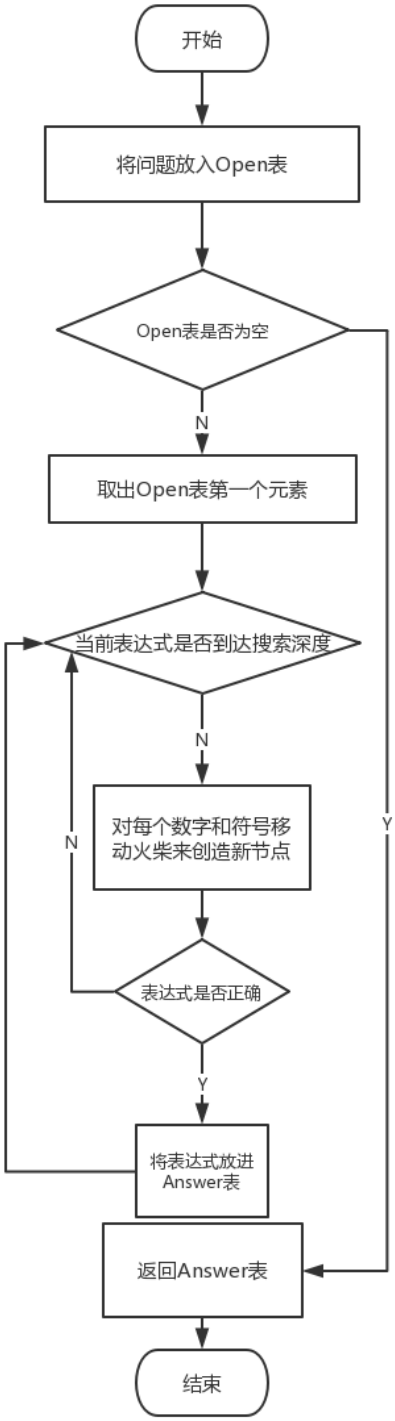
（二）算法介绍:

本项目采用深度优先搜索作为主要的搜索算法，搜索每层的设计顺序是先减后加的顺序，是为了保证火柴数目不变，以下为核心算法的伪代码:

Stack DFS(deapth, Question, NumToPicDict, PicToNum, Open, Close, Answer)

Open. Push(Question)
while(! Open. IsEmpty):
  Expression = Open. Pop()
  if Expression. SearchLevel ! = deapth:
  for i = 0, … , Expression. Length − 1:
  Open. Push(CreateNewNode(i, Expression))
  if Expression. IsCorrect:
  Answer. Push(Expression)
  Close. Push(Expression)
return Answer

流程图见下页:



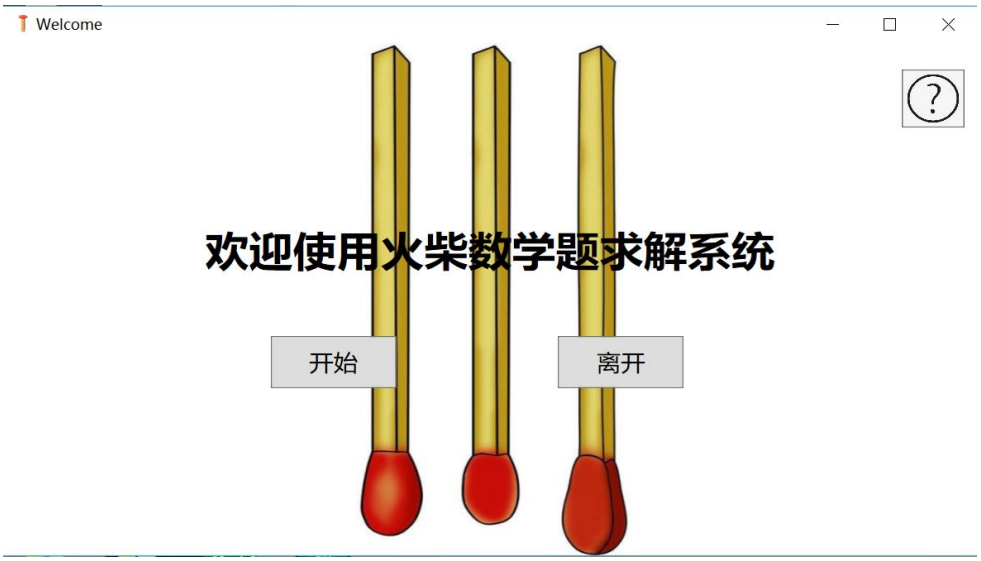
五、UI 设计和使用说明:

（一）UI 设计:

本项目主要有三个界面，分别为开始界面、主要运算界面和帮助界面，以下将对各个界面展开介绍:

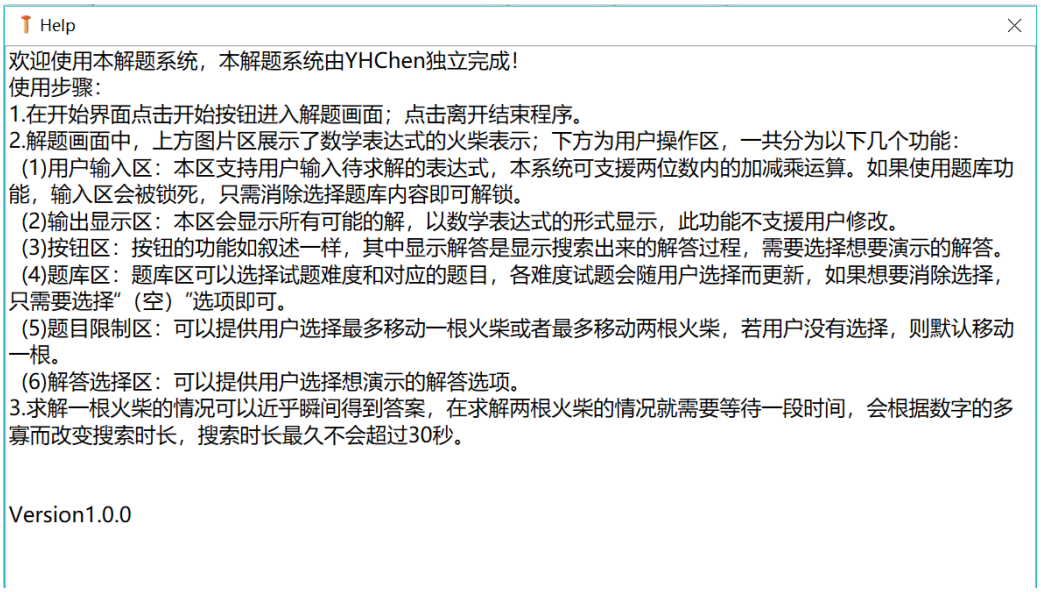
1. 开始界面：

开始界面有三个按钮，分别为“开始”、“离开”和“？”（帮助按钮），点击“开始”按钮进入主要运算界面；点击“离开”按钮结束程序；点击“？”按钮进入帮助界面。



2. 帮助界面：

帮助界面如下图，会显示简单的使用步骤和版本信息。



3. 主要运算界面：

主要运算界面包含六大板块，分别为：题目显示区、用户输入区、结果输出区、题库选择区、题目限制区、解答演示选择区。以下将分开进行介绍：

（1）题目显示区：此区功能主要是将数学表达式转换成火柴形式显示出来，其中数字的显示会靠右对齐。

（2）用户输入区：此区支持用户自行输入想要求解的表达式。

（3）结果输出区：此区功能主要是将搜索结果显示出来。

（4）题库选择区：此区可以提供用户选择内建的题库，可以选择题库难度，每个难度都有对应的题库。

（5）题目限制区：此区可以提供用户选择可以移动的火柴根数。

（6）解答演示选择区：此区可以让用户选择演示的解答。

（二）使用说明：

正常操作使用步骤：

1.在开始界面点击开始按钮进入解题画面。

2.解题画面中，在用户输入框输入待求解的题目或从题库选择区选

择题目，需要注意的是，使用题库功能后，输入框会被锁死，如果要解锁输入框，只需在题库选择区中，选择“（空）”选项，即可清空选择并解锁输入框。

3.点击显示题目，可以显示题目的火柴表达式，如果输入表达式不合法，会有提示框提示。

4.可以选择移动的火柴根数，也可以不选（默认为一根），点击搜索即可开始搜索过程。求解一根火柴的情况可以近乎瞬间得到答案，在求解两根火柴的情况就需要等待一段时间，会根据数字的多寡而改变搜索时长，搜索时长最久不会超过30秒。

5.若搜索完无解，会有提示框显示；若有解，会有提示框显示“搜索成功”，并在解答显示区显示所有解答。

6.可以在解答选择的ComboBox中选择要演示的解答，点击显示解答即可开始演示火柴移动过程。

六、实验总结：

在本次实验中，我结合了课程中学习的状态空间法，将较为复杂的问题，简化为计算机能够识别的状态，这里利用数字电路的知识，将数字和符号变成七段数码管，再设置合理的转换操作设置，利用深度优先搜索，完成了此次的实验项目。

此外本次实验使用的是C#变成，在此之前我并没有学习过C#，

也是借由这次的实验，自己又学习了一个新的编程技能。当然这次虽然使用的是面向对象编程，但为了调用方便，我并没有很严格的封装，这也是我之后仍需要改进的地方。