Homework II

火柴棒游戏

(总体实现报告)

2018 年 10 月 10 日

沈子衿 3160104734

一、概述

本次 Java 作业火柴棒游戏的步骤如下：首先，用户从命令行输入最大数字的位数（如 1 位数、2 位数、3 位数），然后，用户从命令行输入提示数（2 或 3），表示等号左式数字的个数，再输入题目类型编号（移动、移除、添加），以及要进行操作的火柴棒根数。在获取这些信息后，系统随机自动生成火柴棒游戏并展示，同时提示用户输入答案。用户输入答案后，系统将验证是否正确；若正确，则提示正确；若错误，则让用户继续输入，若用户直接回车，则显示正确答案。

根据题目要求，结合作者自身编程能力和客观需求，本题支持高位数和多左式数字，并支持多种移动情况。同时，本题利用字符集支持在命令行下实现了模拟 LED 数字的可视化表达式，有助于用户获得更好的游戏体验。

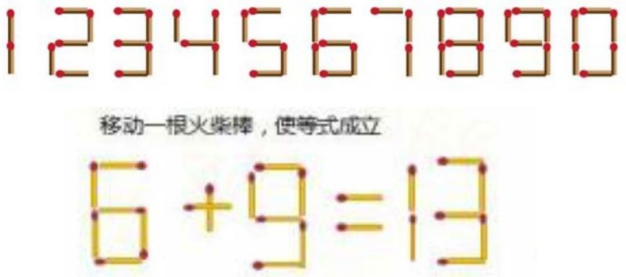


图 1-火柴棒摆放规则

二、实验环境

操作系统: Microsoft Windows 10
Java 版本: JDK 1.8.0_181;
开发环境: Visual Studio Code, Windows PowerShell, Java Compiler;
测试环境: Java Debugger for Visual Studio Code, JetBrains IntelliJ IDEA;
运行环境: Java Virtual Machine, Windows PowerShell;

三、设计思路

3.1 总体设计思路

火柴棒游戏设计的核心是生成可解且有一定挑战性的试题。为此，我们的方案是：在接收用户输入的约束条件后，首先利用随机算法，生成一个已经成立的复杂等式，然后依照火柴棒的摆放规则考察该等式各数字的每一位及各运算符（除等号外）是否具备在特定条件下转化为其他数字/运算符的条件，并对每一种可能条件予以穷举，生成题库；最后从穷举生成的题库中随机抽取一题，作为候选试题显示在交互界面上。

在这一过程中，有四个主要任务值得注意。一是，如何建立数字与数字、数字与运算符之间的转化关系；二是，用何种策略对所有增、删、移的情况进行穷举；三是，如何实现同

用户的交互（包括接收并处理用户的输入以及格式化输出结果），四是，如何实现对答案正确性的验证。接下来，我将重点围绕这三点对设计思路进行阐述。

3.2 模块 1-转化关系的建立

由于本题对可移动火柴棒数量作出了限制（一次不超过 2 个），因此对 0-9 这 10 个数字之间的相互转化关系进行手动穷举，并直接将对应的临接表静态存储于全局的策略是完全可行的。但是，这种方法容易出错，且不具备可拓展性。因此，我决定使用动态生成图的方式来建立它们之间的关系。

```
class Node{
    int number;
    int[] previous;
    int[] next;
    /*--构造函数已省略--*/
}
```

图 2-图节点的数据结构

和转化关系相关的数据结构定义和如左图所示。每一个数字对应一个 Node 节点。该节点包含该数字的整型和两个长为 10 的 int 数组。Previous 表征该数字的火柴棒形式是否可以通过去除一根火柴棒转化为对应下标的火柴棒形式。如果可以，对应下标位置的值置为 1，否则置为 0。Next 同理，只不过由去除一个火柴棒变成了增添一个火柴棒。值得注意的是，在将一个数字的某个 next 值置为 1 后，其对应位置上的 Previous 值也应当置为 1。

为了方便判断数字之间的转化关系，我对每一根火柴棒进行编号，并依据其有无为每一个数字提供了一个七位二进制值。这不仅直观，且易于比较。

由需求知，我们需要支持移动两根火柴棒的情况。在图结构下，这种移动可以使用两次单独的火柴棒移动来等效，但需要注意中间体特例，即第一次转换后生成的并非有效数码，只有进行第二次相同转换时才能形成有效数码。中间体特例可以通过将非有效数码纳入节点的方式解决，但由于本题规模较小，没有采取前者的解决方案，而是建立一个值为 2 的 Next-Previous 关系，以表征这两个数码可以通过增/删 2 根火柴棒实现相互转换。

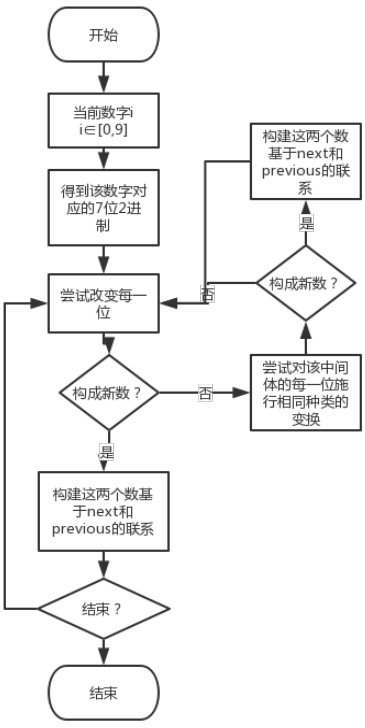


图 3-第一次建图时的流程图

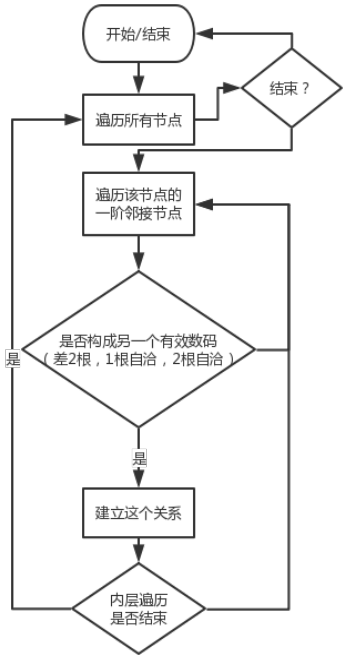


图 4-第二次建图时的流程图

后来，为了让两次单独的火柴棒移动同有中间体的情况统一，并考虑内部变动（不增不删，内部移动一根变为另一个数）的情况，我们对已经完成构建的图进行了二次构建。本次构建令所有值为 2 的 Next-Previous 关系全部直接相连，并增添了内部移动一根（记为 3）和内部移动二根（记为 4）的情况。这样，数码和数码之间的对应关系就完全建立起来了。

3.3 模块 2-对增、删、移情况的穷举

在建立了数码之间的对应关系后，我们有必要对所有增、删、移的情况进行穷举。

增删情况比较单一，几乎只涉及 1 个数码/运算符，只需遍历生成算式的数码表，按照对应关系对数码进行变动，并把所有情况存入题库中即可。唯一值得注意的是减号和加号自身火柴数的增减，以及增/删两根火柴可以通过分别在两个数码上增/删一根实现。这两种特殊情况所涉代码量不多，可以分开讨论。

较为复杂的是移动的情况。对于移动一根火柴，有以下情况：

1. 数字之间移动；
2. 加号移向数字；
3. 数字移向减号；

对于移动两根火柴，有以下情况：

1. 两个数字间移动（+2、-2）；
2. 三个数字间移动（+2/-1/-1、-2/+1/+1）；
3. 数字与操作符间移动（数字-2，操作符+1/+1 或 操作符 -1/-1，数字+2 或 任意操作符与任意数字-1/-1，任意操作符与任意数字+1/+1 或 数字±2， 一个数字和一个操作符±1）；

有必要对这些情况进行初步的分类讨论。由于数码只有固定的 10 个，因此，虽然这一过程会导致大量循环嵌套，但运行时间还是较短的。

值得注意的是，由于移动两根火柴的情况会出现大量可能性，在样本量较大的情况下，可取前 200-300 份样本存入题库。

每一个随机生成的正确等式都对应多个问题。

3.4 模块 3-与用户的交互

与用户的交互主要涉及三个层面：接收信息，处理信息，反馈信息。

本游戏的交互流程如下图：

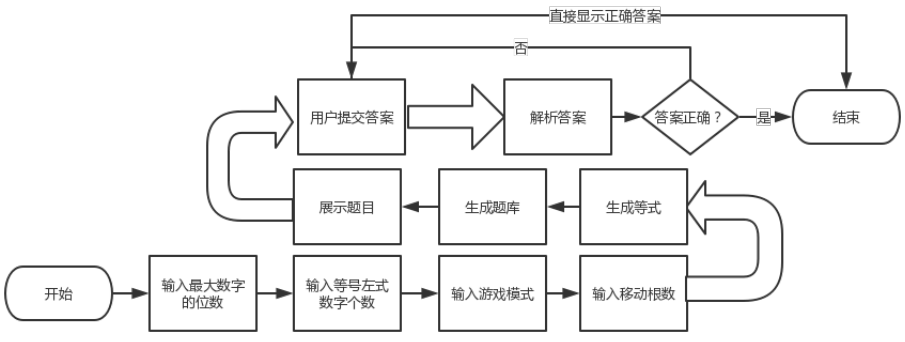


图 5-交互流程示意图

值得注意的是展示题目的过程。如前所述，我们使用制表符字符集实现了在命令行上的“拟图形化“显示。

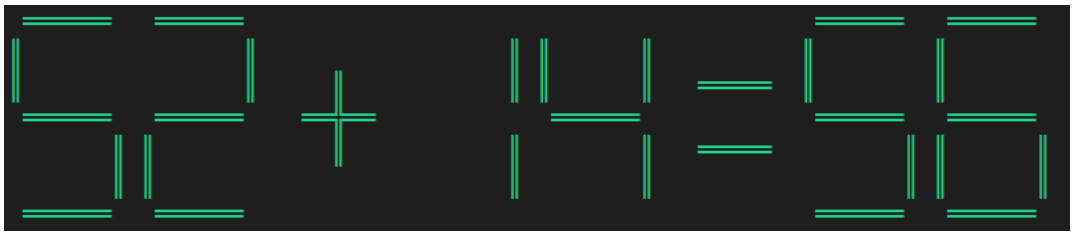


图 6-拟图形化显示算式

这种拟图形化显示方法实际上并不复杂，本质上就是预先准备好各数码和操作符对应的字符数组，然后按照算式顺序逐行连接、打印的结果。在我的代码中，图形高度被设置为 7 行。

3.5 模块 4-答案验证

答案验证是最后，也是最关键的一环。答案验证的流程大致如下：

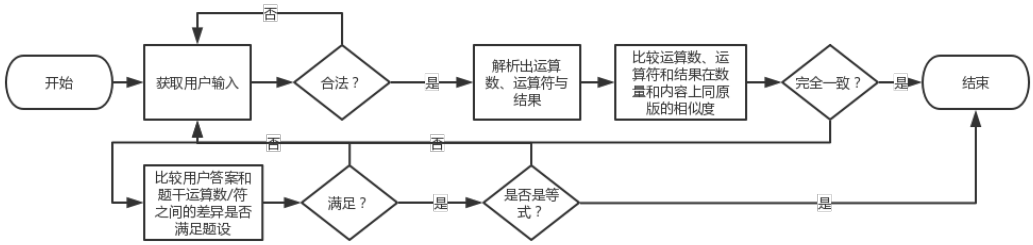
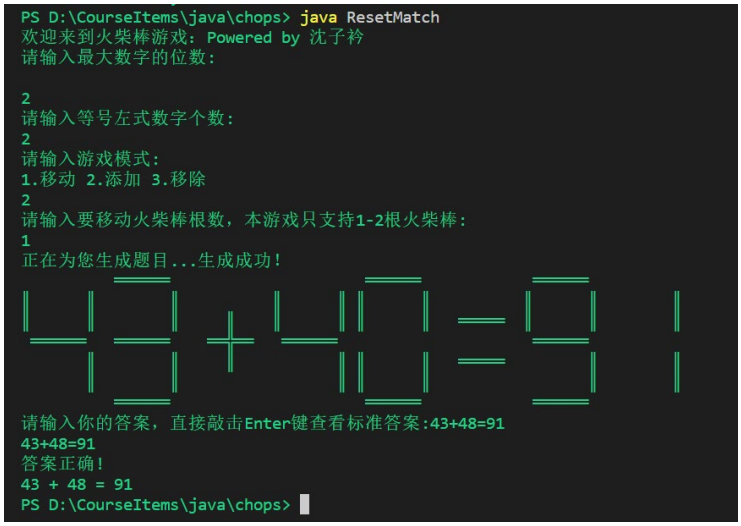


图 7-答案验证流程

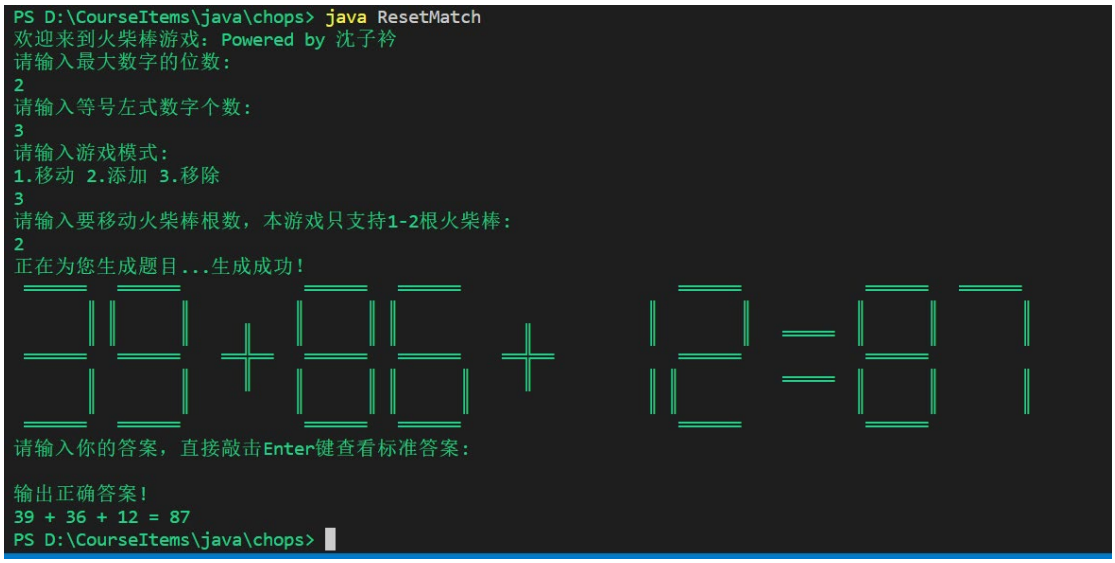
之所以要引入这一套答案验证流程，是因为可能存在多解，尽管这种情况发生的概率比较小。事实上，只要满足等式且按照题干要求变换的答案都算正确答案。

四、程序测试样例

4.1 添加（1 根）

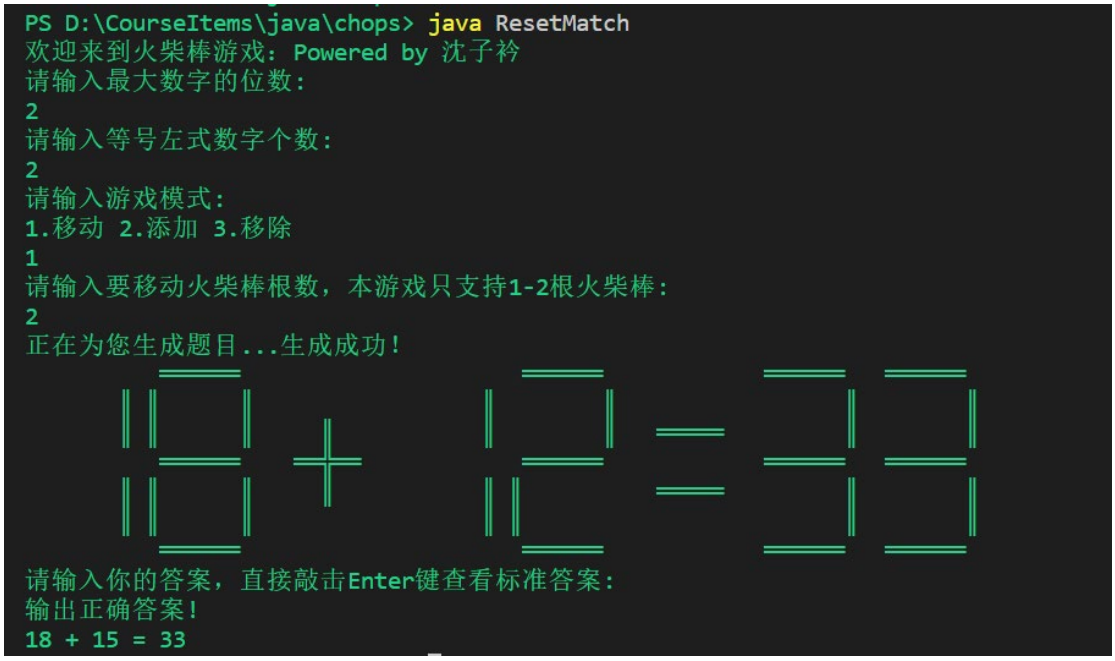


4.2 移除（2 根）



4.3 移动（1 根）

4.4 移动（2 根）



五、心得体会

本次实验代码量较大，细节较多，对于我们这些易受 C++特性影响的 Java 的初学者来说确实存在着一定的挑战。但是我们克服了重重阻力，完成了这一任务，也加深了对概念地理解。可以说，本次作业是具有深远且重大的意义的。在未来的学习工作中，我会再接再厉，争取更大的胜利。