華中科技大學

课程设计报告

题目：基于 SAT 的数独游戏求解程序

课程名称：程序设计综合课程设计
专业班级：CS2207
学　　号：U202215554
姓　　名：付名扬
指导教师：卢萍
报告日期：2023.8.29

计算机科学与技术学院

摘要

可满足性问题(Boolean Satisfiability Problem)简称 SAT 问题，源于数理逻辑中经典命题逻辑关于公式的可满足性的概念，是理论计算机科学中一个重要的问题，也是第一个被证明的 NP-complete 问题，具有极其重大的意义。为加深对 SAT 问题的理解，强化自身代码能力与学术涵养，同时体验 NP-complete 问题应用价值，我制作了基于 DPLL 算法的 SAT 求解器系统，并根据不同的选元策略提供了多种**优化方案**，同时希望通过数独游戏对 SAT 问题进行实践。

系统包含 SAT 模块与 HANIDOKU 板块，前者通过 CNF 文件，以链表数据结构对问题变元和子句进行读取，并选择以 DPLL 为核心的 **MO(选取出现次数最多的变元)、MPO(选取出现次数最多的正元)、MOM(最短子句最多出现优先法则)与 MOM+(变元权值取平均)**四种策略对选元优化。后者采取了随机生成的数独游戏格局，设计了蜂窝的 9x17 矩阵间隔存储方式以及**映射转化公式**对数独进行归约。同时数独保证了可玩性，其中有填充数字的基础功能和获取提示、检验正确的附加功能以及利用 MPO 策略调用 SAT 求解器解决归约格局的功能。

接着对于每一个选元策略都做出了评价。首先以满分为 5 分的水平分别对上述四种策略给出了 3/4/4/4.5 的分数。之后为测试系统的**优化率**，特地选取了 S、M、L 三种规模的 30 种不同结构的算例对程序工作效率进行了测试，测试方式是统计优化前时间 t 和利用 MOM+策略优化后的时间 t1 并计算每一个算例对应的优化率。最终发现有 **80%以上**的算例成功达到了 **90%以上**的优化率，但遗憾的是并没有成功优化所有样例，有一些样例依旧无法优化。

最后在系统特色方面也设计了大量具有想象力的小功能。文本里 99%的字体是加粗蓝色大写；对数独格局打印利用了下划线实现了优化，使得其界面更加美观好看；游玩过程里系统将自动计算从游戏开始到结束的时间并在玩家胜利后打印出来；玩家胜利和失败有专属的“YOU WIN!”和“GAME OVER”全屏字符画；数独难度可分为 7 个由低到高的等级并给出自定义难度设置功能。

关键字：SAT、DPLL、优化策略、优化效率、蜂窝数独

任务书

□ 设计内容

SAT 问题即命题逻辑公式的可满足性问题（satisfiability problem），是计算机科学与人工智能基本问题，是一个典型的 NP 完全问题，可广泛应用于许多实际问题如硬件设计、安全协议验证等，具有重要理论意义与应用价值。本设计要求基于 DPLL 算法实现一个完备 SAT 求解器，对输入的 CNF 范式算例文件，解析并建立其内部表示；精心设计问题中变元、文字、子句、公式等有效的物理存储结构以及一定的分支变元处理策略，使求解器具有优化的执行性能；对一定规模的算例能有效求解，输出与文件保存求解结果，统计求解时间。

□ 设计要求

要求具有如下功能：

（1）输入输出功能：包括程序执行参数的输入，SAT 算例 cnf 文件的读取，执行结果的输出与文件保存等。(15%)

（2）公式解析与验证：读取 cnf 算例文件，解析文件，基于一定的物理结构，建立公式的内部表示；并实现对解析正确性的验证功能，即遍历内部结构逐行输出与显示每个子句，与输入算例对比可人工判断解析功能的正确性。数据结构的设计可参考文献[1-3]。(15%)

（3）DPLL 过程：基于 DPLL 算法框架，实现 SAT 算例的求解。(35%)

（4）时间性能的测量：基于相应的时间处理函数（参考 time.h），记录 DPLL 过程执行时间（以毫秒为单位），并作为输出信息的一部分。(5%)

（5）程序优化：对基本 DPLL 的实现进行存储结构、分支变元选取策略[1-3]等某一方面进行优化设计与实现，提供较明确的性能优化率结果。优化率的计算公式为：$[(t-t_o)/t]*100\%$,其中 t 为未对 DPLL 优化时求解基准算例的执行时间，t_o 则为优化 DPLL 实现时求解同一算例的执行时间。(15%)

（6）SAT 应用：将数双独游戏[5]问题转化为 SAT 问题[6-8]，并集成到上面的求解器进行数独游戏求解，游戏可玩，具有一定的/简单的交互性。应用问题归约为 SAT 问题的具体方法可参考文献[3]与[6-8]。(15%)

目录

1 引言

1.1 课题背景与意义

SAT 问题又称命题逻辑公式的可满足性问题（satisfiability problem），是对一个以合取范式（Conjunctive Normal Form，常简称 CNF）的形式给出的命题逻辑公式进行判断，找出是否存在一个真值指派使得该命题逻辑公式的值为真。SAT 问题是计算机科学与人工智能基本问题，是一个典型的 NP 完全问题。看似简单，却可广泛应用于许多实际问题如人工智能、电子设计自动化、自动化推理、硬件设计、安全协议验证等，具有重要理论意义与应用价值。

1.2 国内外研究现状

关于 SAT 问题的研究历程一直都在持续不断地发展。在 1997 年和 2003 年，H.Kautz 与 B.Selman 分别提出了两次对 SAT 搜索所面临的挑战性问题进行了详细的探讨。此外，在 2011 年和 2007 年，他们还分别撰写了两份综合性的文献，全面总结了当时 SAT 问题研究领域的现状。值得一提的是，黄文奇提出的 Solar 算法曾在北京第三届 SAT 问题快速算法比赛中荣获第一名，这进一步展示了 SAT 问题研究领域的不断进步和创新。

在 SAT 问题的解决方法方面，主要可以分为两大类：完备算法和不完备算法。不完备算法主要采用局部搜索策略，虽然不能保证一定找到解，但其解题速度较快，对于某些 SAT 问题的求解效果甚至比很多完备算法更为出色。与之相比，完备算法出现得更早。其优势在于能够正确判断 SAT 问题的可满足性，并在算例无解的情况下提供完备的证明。

在求解 SAT 问题的优化算法方面，包括启发式算法、冲突子句学习算法以及双文字监视法等多种方法，这些方法在不同情况下都发挥着重要作用。

DPLL 算法是一种高效的 SAT 问题判定算法。早在 1960 年，Davis 和 Putnam 提出了最早的 DPLL 算法，当时称为 DP 算法。这一算法的提出显著降低了 SAT 问题的复杂性和求解器所需的内存空间，为 DPLL 算法在解决 SAT 问题中的应用奠定了基础。DP 算法通过变量消解来减小搜索空间。然而，当 Logemann 和 Loveland 尝试实际实施 DP 算法时，他们发现该算法在消解过程中占用了过多的内存，受到了当时计算资源的限制。因此，在 1962 年，Logemann 和 Loveland

等人对 DP 算法进行了改进，引入了分支回溯策略，即在选择变量进行分支赋值后，若发生冲突，则进行回溯。然而，这种 DPLL 算法的求解效率较低，仅能处理最多包含 10 个变量的 SAT 问题，且应用于实际问题时表现不佳。

1996 年，Marques-Silva 和 Sakallah 提出了 GRASP 算法，该算法在实际问题求解中能够尽早剪枝不满足搜索空间，大幅缩短了搜索时间，提高了求解效率。1997 年，H. Zhang 提出了 SATO 算法，它在 DPLL 算法的基础上引入了智能回溯策略、搜索重新启动策略以及 BCP 数据结构等，有效降低了 SAT 求解器的求解时间，并解决了先前无法处理的一些 SAT 问题。2001 年，L. Zhang 提出了 zChaff 算法，通过优化改进 BCP 推导的效率和学习方式，极大提高了 SAT 问题的解决效率，成为后续 SAT 算法发展的重要基础。2005 年，Een 和 Sorensson 提出了 MiniSATI9I 算法，至今仍被认为是综合性能最出色的 SAT 求解器。该算法在 2006 年的 SAT 国际竞赛中夺冠。

华中科技大学计算机学院团队在第 20 届国际 SAT 算法竞赛中斩获了冠军。除此之外，还存在多种用于计算 SAT 问题的求解器，其中包括 GRASPI、zChaff、BerkMin 和 MiniSATI9I 等。这些求解器几乎都是基于 DPLL 算法或对该算法进行了优化而得到的。

一般而言，当代 SAT 求解器可以求解百万变量的 SAT 问题（1~2h 内）。

- 挑战 1：对于特定 SAT 问题，求解规模依旧很小。
- 挑战 2：有些现实的 SAT 问题规模远超百万变量级别！
- 挑战 3：并行算法仍然是一个难度很大的问题。

1.3 课程设计的主要研究工作

依据 Davis 和 Putnam 在 1960 年提出的 DPLL 算法来求解合取范式，并基于原始的 DPLL 算法提出优化方案，在不同优化方案中依据解决问题的时间进行对比，评估其优化效果。在搭建的 DPLL 框架上实现可游玩的蜂窝数独游戏，并巧妙利用基于 DPLL 的 SAT 求解器对其进行归约和解决，以实现 SAT 问题的简单运用。而本文则主要介绍 SAT 问题背景及意义，给出所使用数据结构和所用常量定义，介绍重要函数思路，提供 DPLL 优化方案并通过计算给出优化率，最后进行课设的总结、思考与致谢。

2 系统需求分析与总体设计

2.1 系统需求分析

系统要求分为两大模块，要求实现的是一个基于 DPLL 的 SAT 问题求解器以及一个可游玩可归约的蜂窝数独游戏。

- SAT 求解器需要先设计总体框架与数据结构便于实现。求解器的总体框架应该包括文件读取部分、文件求解部分及 res 文件保存部分。数据结构在任务书中推荐使用二维链表，但可能不是最优。
- 蜂窝数独游戏需要包括游戏格局生成部分、游玩部分、游戏格局文件保存部分（归约部分）以及求解部分。其中游玩部分需要包括填充数字、获取游戏提示、检验答案是否正确等功能。

2.2 系统总体设计

系统分为两大模块，分别为 **SAT SOLVER** 与 **HANIDOKU GAMES**.

首先，图 2.1 给出了系统主要交互的实际界面；

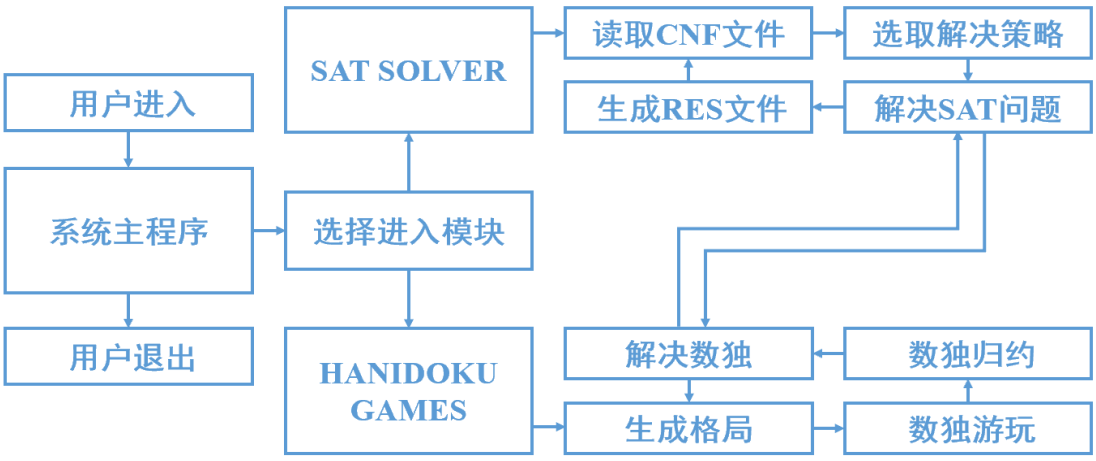


图 2.1 系统主要交互界面

用户在进入主程序后，首先会被给出两个选项，即模块 I. SAT SOLVER 与模块 II. HANIDOKU GAMES. 用户也可以不进行任何操作直接退出。

如果选取了模块 I，则自动进入 SAT SOLVER，这里有读取、打印、解决、生成结果等基本功能，但是用户必须先读取 CNF 文件才可以进行后续操作。

如果选取了模块 II，则自动进入 HANIDOKU GAMES，这里有生成、游玩、归约和解决等基本功能，此外还融入了特色鲜明的小设计，保证了趣味性。

上方图示只是简略描述。下面详细描述模块 I 和 II 的概况图。

橙色模块（SAT SOLVER）负责打印、解决和保存 CNF 文件。其中核心（用红色标出）是解决部分，这里包括多种可选的优化策略，需要用户自行选择。SAT SOLVER 的概况图可参见图 2.2；

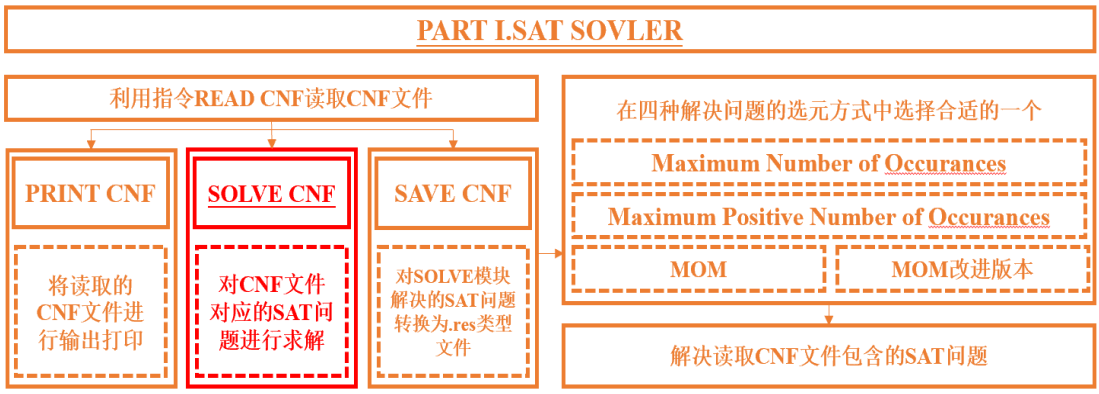


图 2.2 SAT SOLVER

绿色模块（HANIDOKU GAMES）负责生成数独格局、数独挖洞、用户交互（包括难度选择与游玩功能）、数独归约以及调用求解器解决归约 CNF 文件，HANIDOKU GAMES 的概况图可参见图 2.3；

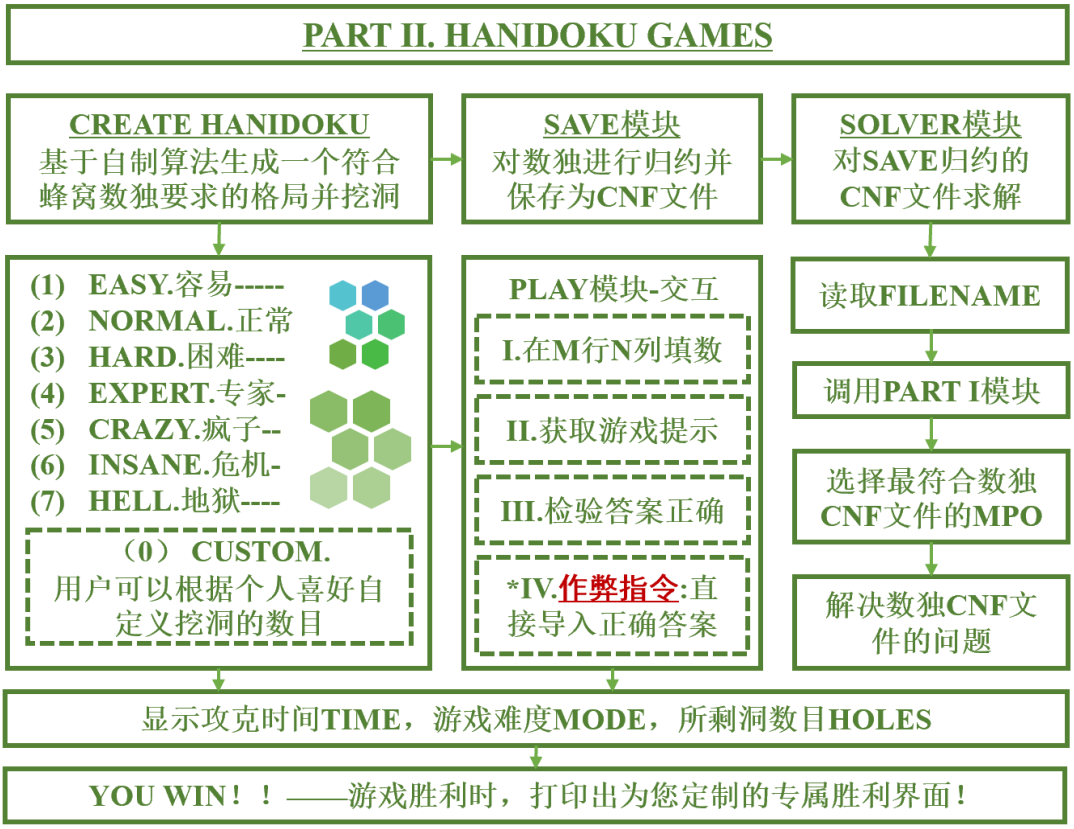


图 2.3 HANIDOKU GAMES

3 系统详细设计

3.1 有关数据结构的定义

本系统采取的数据结构是二维链表（参考任务书）。即将子句表示为由文字构成的链表（第一维）；整个公式则是由子句构成的链表（第二维）。该数据结构可能不是最优选择，但也可以形象地刻画出Bool变元、子句之间的关系。

一种优化策略是在每个**cNode结构体**内新增一个**prior指针**指向前驱，即双向链表，这样可以更加高效的回溯，但遗憾的是本系统未能及时实现。

数据结构的 typedef 定义如下。详细的结构图可参见图3.1。

```
typedef struct cNode{
    int data;//变元的序号
    cNode *next;//指向下一个变元
}cNode;//链表结点定义，即Bool变元
```

```
typedef struct clause{
    int num;//子句序号
    cNode *head;//指向该子句首元
    clause *next;//指向下一个子句
}clause;//二维链表表头，即每个子句
```

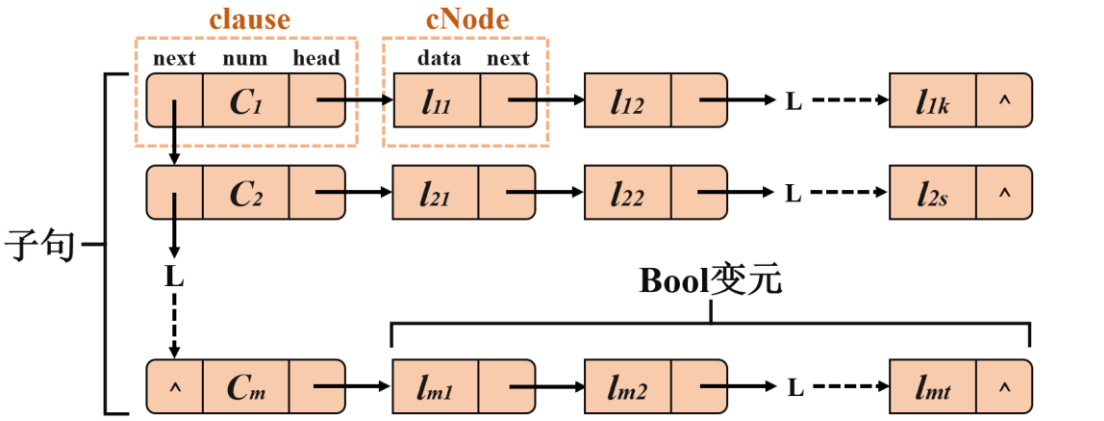


图3.1 数据结构详细定义以及具体图片

3.2 主要算法设计

3.2.1 系统核心算法-DPLL

DPLL 算法是基于树/二叉树的回溯搜索算法，主要使用两种基本处理策略：

单子句规则。如果子句集 S 中有一个单子句 L,那么 L 一定取真值，于是可以从 S 中删除所有包含 L 的子句（包括单子句本身），得到子句集 S_1，如果它是空集，则 S 可满足。否则对 S_1 中的每个子句，如果它包含文字¬L,则从该子句中去掉这个文字，这样可得到子句集合 S_2。S 可满足当且仅当 S_2 可满足。单子句传播策略就是反复利用单子句规则化简 S 的过程。

分裂策略。按某种策略选取一个文字 *L*.如果 *L* 取真值，则根据单子句传播策略，可将 *S* 化成 S_2；若 *L* 取假值（即¬*L* 成立）时，*S* 可化成 S_1.

交错使用上述两种策略可不断地对公式化简，并最终达到终止状态。

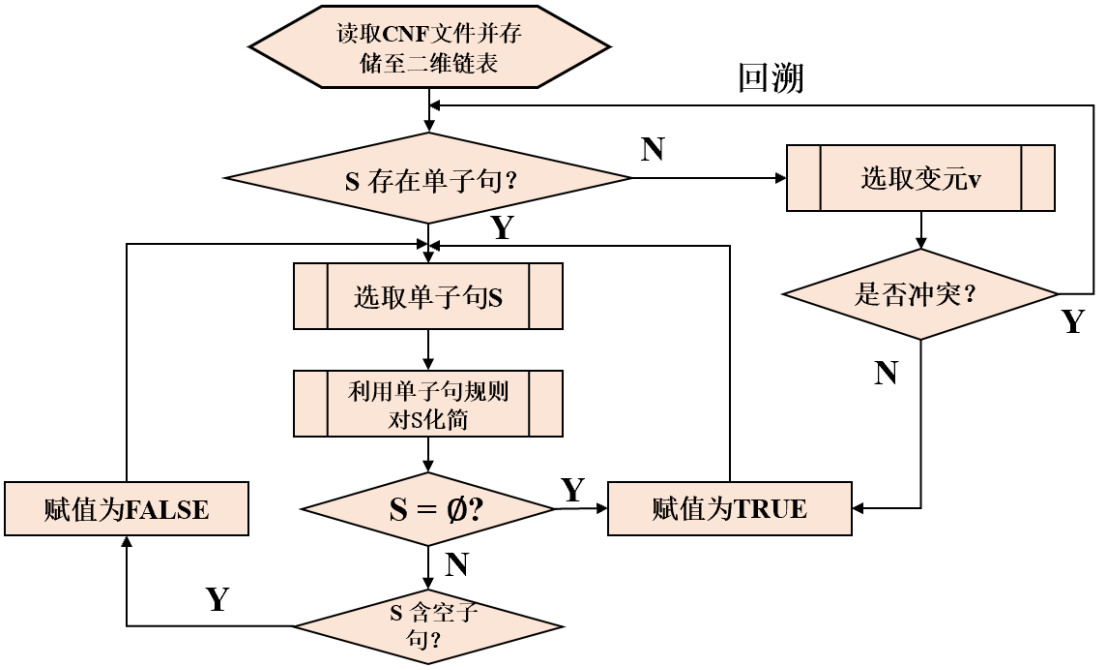


图 3.3 DPLL 流程图

3.2.2 DPLL 算法优化策略

在 DPLL 算法的流程中，有一个有趣的“**基于某种选元策略**”。笔者认为对 DPLL 算法的优化显然应该从这个方面下手。而事实上，变元的选取策略也的确对整个 SAT 求解器的工作效率有着极其重要的影响。笔者共实现了四种策略这些策略也的确优化了 DPLL 过程的效率。以下给出后四种策略的思路与分析。

这些选取策略都是启发式策略。启发式策略是指在执行 DPLL 算法时，根据一定的规则和 heuristics，选取一个最有希望的变量进行分裂。根据问题的特性和经验知识，引导搜索方向，避免了盲目搜索和无效分裂，提高了算法求解能力。优秀的启发式选取策略可以引导算法更接近解空间，从而找到更快的求解路径。。

- **Max Number of Occurance**

思路：遍历所有字句中的变元，选取当前出现次数最多的元。

分析：思路非常简单也很容易想到，同时也是很传统的方法。通过选取出现次数最多的变元，可以大大简化子句的复杂度。但是因为随机性强的文件可能大多数变元的出现次数很接近，因此表现不太突出。

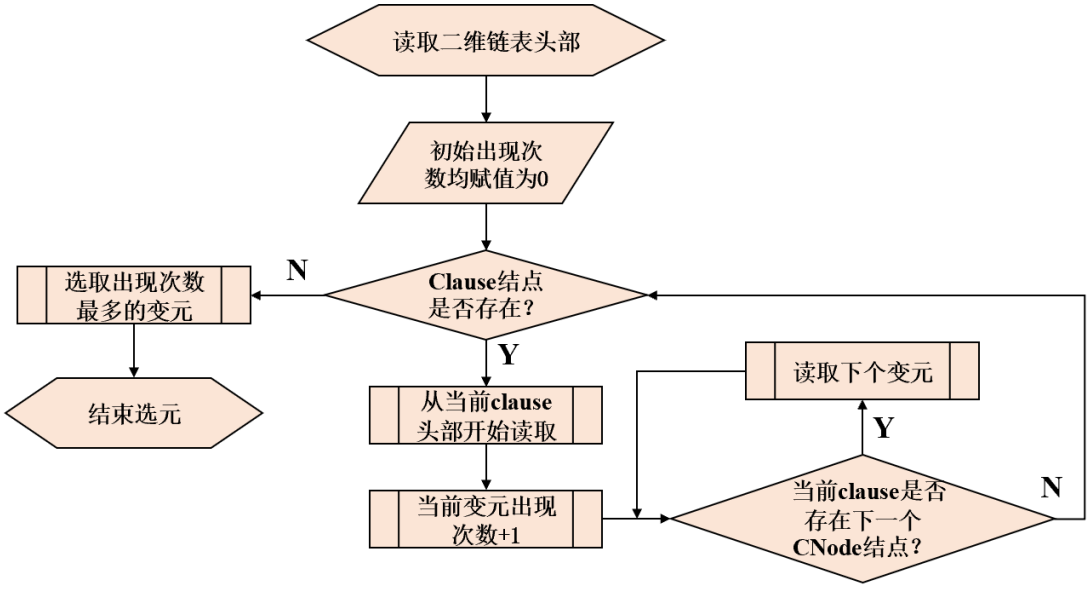


图 3.4 Max Number of Occurance 流程图

- **Maximum Positive Number of Occurances**

思路：将前者统计对象变为正元，即选取出现次数最多的正元。

分析：测试后发现该策略在数独类文件有极大的优势，而在其他文件表现平平。这是因为数独文件的变元大多为正且分布具有强规律性，通过一次选元就可以快速消除非常多的字句。这种策略也被选为解数独 CNF 文件的首选。

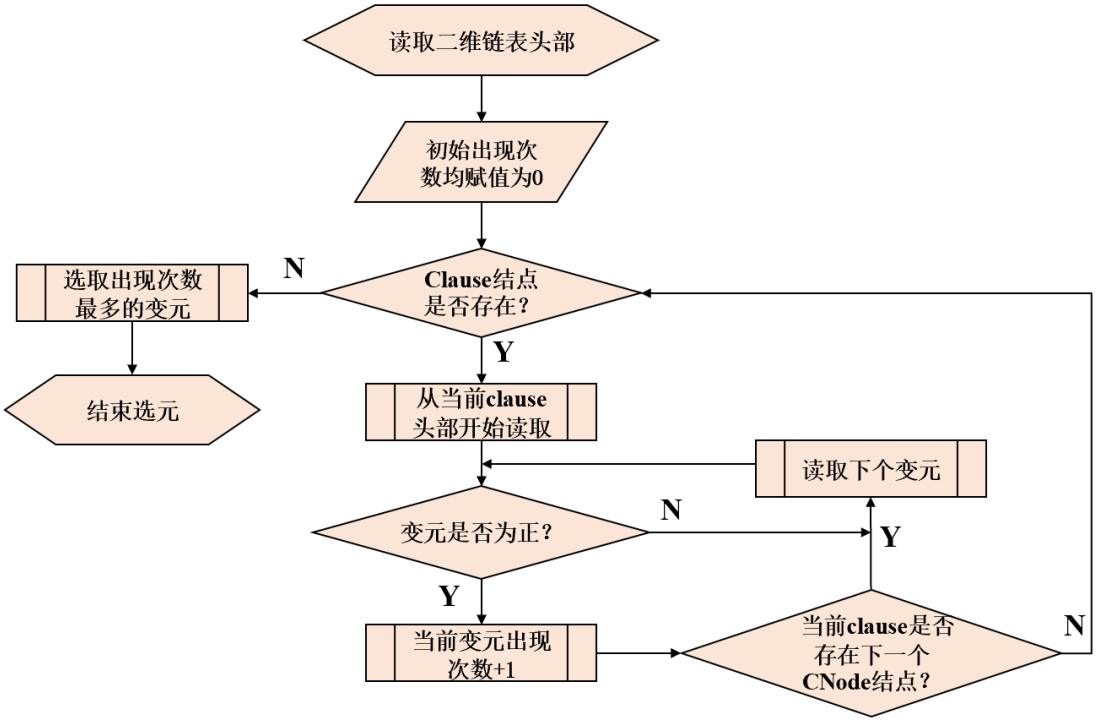


图 3.5 Max Positive Number of Occurance 流程图

- **Maximum Number of Occurances in Minimum size Clauses (MOM)**

思路：求解出和文字 *l* 相对应的 *J* 值，最终会选取使得 *J(l)*取值最大的文字 *l* 并赋值为真，求解公式如下所示：

$$J(l)=\sum_{l\in C_i}2^{-n_i}\quad（i=1,2,\cdots,m，假设共有\ m\ 个子句）$$

分析：MOM 方法又叫 **JW 方法**。公式中的n_i表示包含 *l* 的子句的长度。那么由指数函数的单调性容易看出，长度越**长**，则 J 值就越**小**。而只要子句中一个文字可以满足，那么子句就是可满足的。那么长度越长，可满足的可能性就越大，反之长度越小则越不容易满足。所以在长度较小的子句里选取出现频率较大的变元并优先满足它们，以尽快求解。

该方法也因此命名为**“最短子句出现频率最大优先”**原则。这是一种高精度的求解算法，也被许多 SAT 求解器运用。但美中不足的是由于高精度的计算导致其决策时间将会较长。同时如果子句的长度大致相同时，该算法便也失去了最大的优势。但它也不失为一种科学的算法。

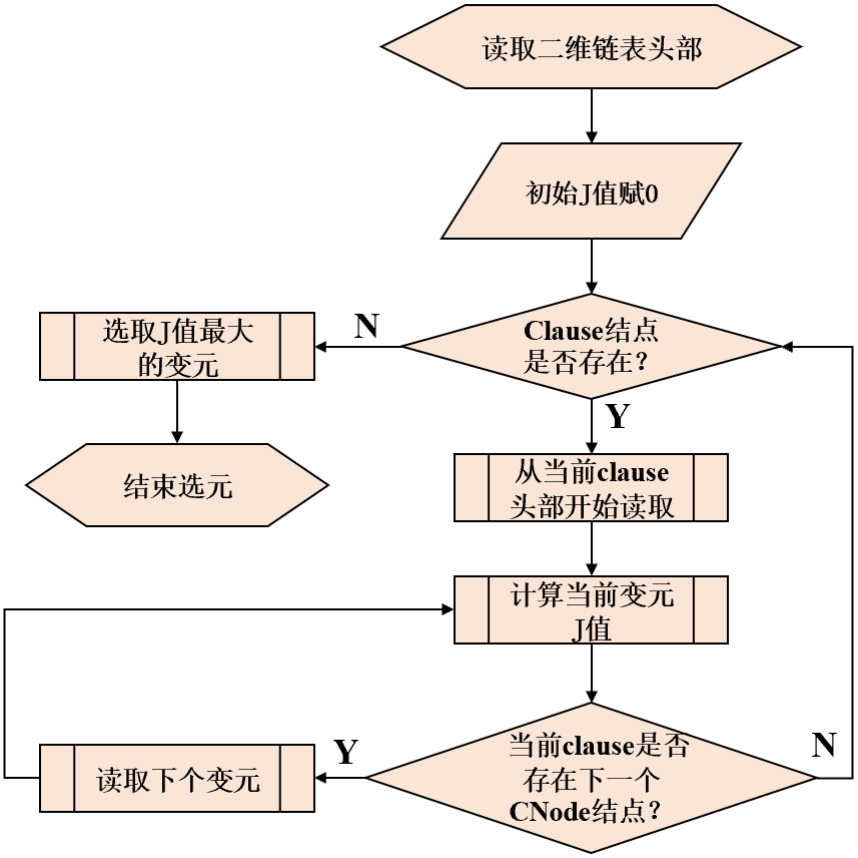


图 3.6 MOM 流程图

- **MOM-Optimized Version**

思路：大致思路与上文的 MOM 相同，但是将一个变元的正和负的 J 值进行了求平均值，即考察的是[J(+) + J(-)]/2 的值。

分析：这是一个对 MOM 的迭代版本的优化，优化方案是求平均值。由于底层逻辑相同，所以同样有着决策时间开销过大和难以应付含大量长度相同的子句的 CNF 文件，如解数独文件时由于含有上千个单子句，因此解数独并不是它的强项。该策略优势在于考察的是正负平均值，实际上就将正负变元看成一个整体，因此在消元上具有更加强大的效率，在解随机性强的文件时强于原版和另外两个策略。

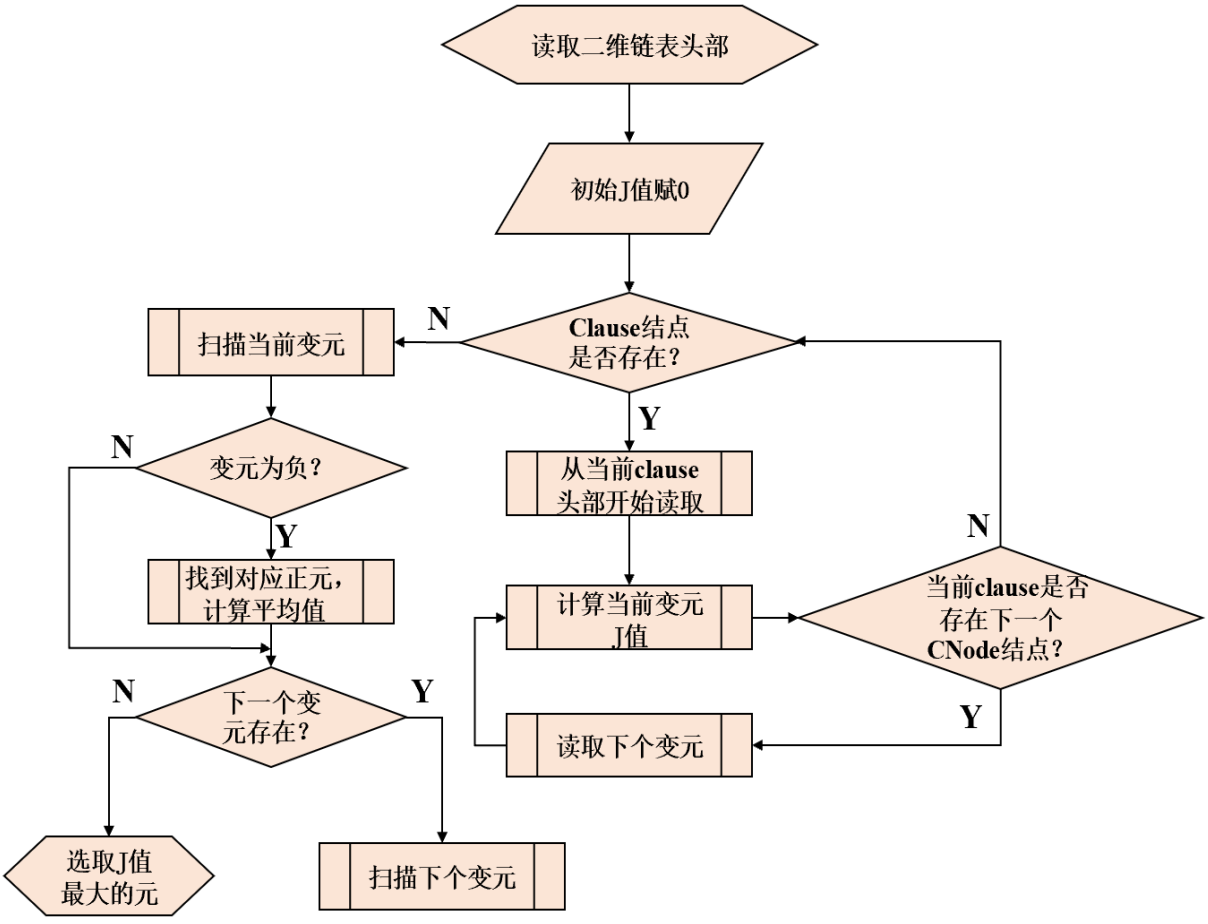


图 3.7 MOM-Optimized Version 流程图

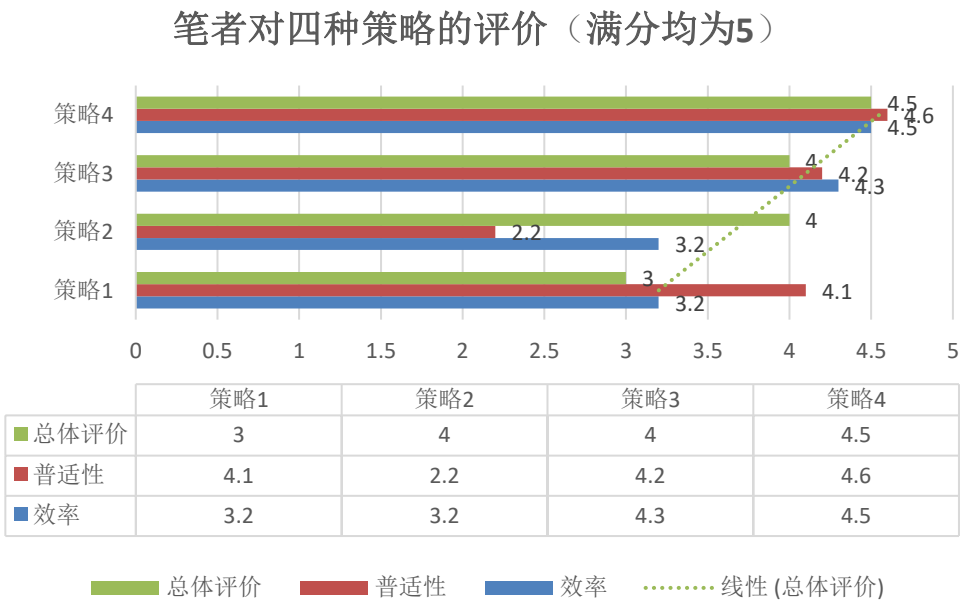
以上是四种优化选元策略(事实上笔者共实现了五种，但是第一种效率极低，故舍去了这种策略)，其实还有很多其他策略和算法，如 VSIDS 策略（变量状态独立衰减）、VIG 策略(同样类似于加权策略，但权重计算方式与 MOM 策略有所不同)等，这些也都是优化 DPLL 的广泛应用的启发式策略。

总结：策略 4 最高效。策略 2 针对性更强！

图 3.8 所包含的表格与条形图给出了笔者对这四种策略的大致评价（这里的评价较为主观粗略），具有一定参考性。

从图表可见，在策略 1~4 的迭代过程中，新版本的策略的普适性和效率有增强的趋势，并且策略 4 经测试后也的确是四种策略中最具普适性和高效的方法。而比较特殊的策略 2 则更加具有针对性，它是一个专门用来解数独文件的策略。这里图表并没有体现出来。

对于具体的针对不同算例的求解时间效率及其对比将在后文中详细给出。



	策略1	策略2	策略3	策略4
总体评价	3	4	4	4.5
普适性	4.1	2.2	4.2	4.6
效率	3.2	3.2	4.3	4.5

图 3.8 策略评价

4 系统实现与测试

4.1 系统实现

4.1.1 系统环境、常量与变量定义

实验结果常常会受到客观环境的影响。因此该部分首先有必要给出系统实现的软硬件环境，目的是提供详细的实验条件，使实验结果更加具有可信度和独特性。详细环境配置可参见表 4.1：

表 4. 1 系统实现的软硬件环境

设备名称	付名扬
处理器	13th Gen Intel(R)Core(TM)i7-1360P 2.20 GHz
机带 RAM	16.0 GB (15.7 GB 可用)
设备 ID	6BE12637-8E66-41F0-BC81-8EA961EC1936
产品 ID	00342-31428-05988-AAOEM
系统类型	Window11 64 位操作系统, 基于 x64 的处理器
笔和触控	没有可用于此显示器的笔或触控输入

接下来是系统所使用的一些重要参数、常量和变量。

- 有关系统 define 定义可参见表 4.2：

表 4.2 系统常量定义（#define）

Define 常量名	定义对象	作用
OK	**1**	**返回正常运行结果**
ERROR	**0**	**返回异常运行结果**
HANIDOKU	**1**	**运行 HANIDOKU GAME**
SAT	**0**	**运行 SAT SOLVER**

- 有关系统变量定义可参见表 4.3：

表 4.3 系统变量定义

变量类型	变量名	作用
int	**Bool_cnt**	**Bool 变元数目**
int	**Clause_cnt**	**CNF 子句个数**
int	**BranchRule**	**选择选元策略**
char*	**FileName**	**读取 CNF 文件名称**
char*	**HANIDOKUFileName**	**读取数独 CNF 文件名称**
int**	**Board**	**数独游戏格局标记**
int**	**CompletedBoard**	**完整数独格局标记**
int**	**MARK**	**对数组元素标记**
int*	**value**	**记录真值状态**

4.1.2 系统主要函数设计

1.读取 CNF 文件

函数名称：ReadCNF(clause *cnf, int sat_hanidoku)

主要参数：cnf 是操作的二维链表表头结点，sat_hanidoku 判断读取模块

运行结果：创建新的二维链表，将 CNF 文件内容保存进去

设计思路：首先判断子系统。接着读取变元和子句个数分别确立表头后 CNode 结点数目和表头结点数目，然后读取每行的数据以填入 CNode 数据域（即 Bool 变元），每读取一行则创建新的表头并重复读取数据操作，直到文件被读取完成，返回 OK.在这个过程里，最后读取的子句数目与最开始读取的给定子句数目不符，或者读取时数据不合法，则返回 ERROR.

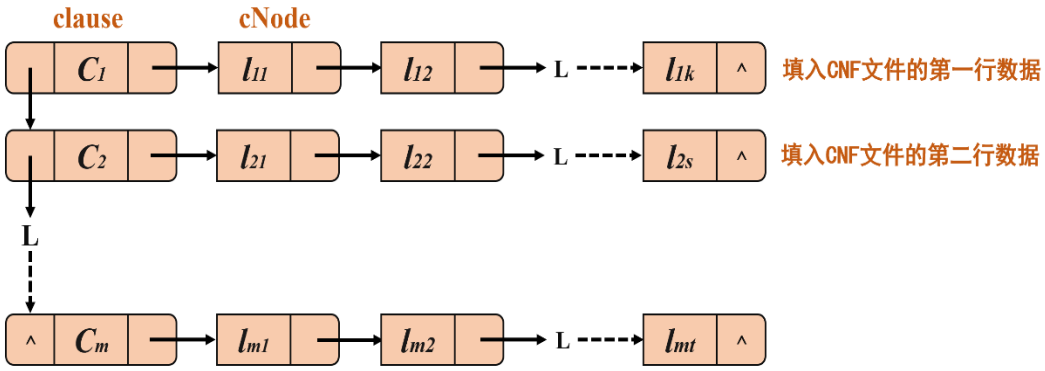


图 4.1 ReadCNF

2.销毁二维链表

函数名称：DestroyClasue(clause *cnf)

主要参数：cnf 是操作的二维链表表头结点

运行结果：销毁指定的二维链表，释放内存空间

设计思路：从第一个表头结点开始依次释放其后方的 CNode 结点，释放完成后释放此表头结点，后面的表头结点执行相同的方式，直到完全释放。

3.删除子句

函数名称：RemoveClasue(clause *cnf, clasue *cl)

主要参数：cnf 是操作的二维链表表头结点，cl 是要删除的子句

运行结果：删除指定的子句，释放内存空间

设计思路：从要删除的表头结点（即子句）开始依次释放其后方的 CNode 结点，

释放完成后释放此表头结点。如果没有该子句则报错。

4.删除子句中变元

函数名称：RemoveCNode (clause *cnf, clasue *cl, CNode *Node)

主要参数：cnf 是二维链表表头结点，cl 对应子句，Node 是待删除变元

运行结果：删除指定的子句的指定变元，释放内存空间

设计思路：查找到对应子句，之后在它内部查找对应变元，找到则删除。

5.增加子句

函数名称：AddClasue(clause *cnf, clasue *cl)

主要参数：cnf 是二维链表表头结点，cl 是待增加的子句

运行结果：在二维链表中增加新的表头节点 cl

设计思路：直接将 cl 插入到原本最后的表头结点的 next 指针即可。

6.判断是否是单子句

函数名称：IsUnitClasue(clause *cnf, clasue *cl)

主要参数：cnf 是二维链表表头结点，cl 是待判断的子句

运行结果：判断 cl 是否是单子句，给出判断结果

设计思路：直接查看第一个变元后面是否还有变元即可。

7.DPLL 过程

函数名称：DPLL(clause *cnf, int value[])

主要参数：cnf 是二维链表表头结点，value 储存变元真值

运行结果：解出读取的 CNF 文件，给出是否有解

设计思路：主要思路见 3.2.1，这里对选元策略进行了优化，优化策略见 3.2.2.

8.保存求解结果文件

函数名称：SaveRes (int res, int value[], double time)

主要参数：res 是求解结果，value 储存变元真值，time 是求解时间

运行结果：将 DPLL 过程的有解情况，变元真值以及求解时间保存

设计思路：依据 res 的值判断是否有解，通过 value 将赋值传入文件，并将所耗费时间 time 写入文件，最后生成一个 res 文件，名称与 cnf 文件相同。

9.创建数独游戏格局

函数名称：CreateGame(int **board, int row, int col)

主要参数：board 是数独游戏格局，row 表示行数，col 表示该行第几个元素

运行结果：创建一个数独游戏格局

设计思路：首先随机生成格局中的部分数字，保证其满足数独游戏格局的基本要求并标记这些已经存在的数字在格局中的位置。接着在第一个没有被标记的位置执行赋值操作，赋值操作也需要满足格局要求，每次生成都检测一遍生成结果是否合法，如果合法则通过递归继续生成，如果不合法则回溯。

最后在生成完成的游戏格局中进行挖洞处理。

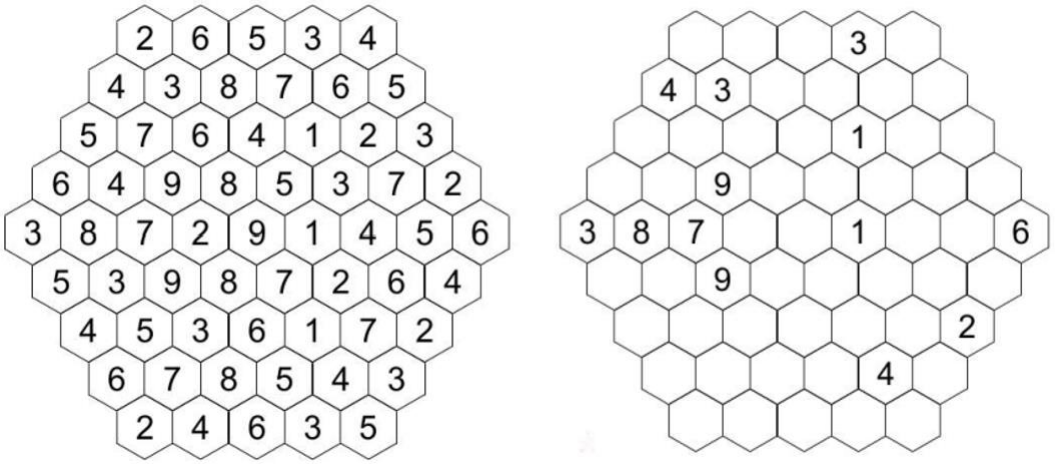


图 4.2 数独格局生成前后

10.归约数独游戏格局

函数名称：TranslateHanidokuToCNF(int **board, int holes)

主要参数：board 是数独游戏格局，holes 是挖洞数目

运行结果：将数独游戏格局归约为一个可解的 CNF 文件

设计思路：根据系统运行结果判断，没有洞的游戏格局会有 5821 个子句，因此子句个数为 5821-9*holes 个，变元个数为 61*9=549 个。根据这两个公式即可保存变元和子句个数。然后根据当前格局的已有数字读取对应约束条件。接着对行、左对角线（方向是↙）和右对角线（方向是↗）的选填约束、连续性约束分别进

行读取。这些约束的读取方式也将在下方介绍。

- **数独的约束与读取**

① 提示数字约束：

在数独游戏开始时就会给出一些提示数字。这种约束的读取非常简单，只需要遍历一次数独格局，找到第 i 行第 j 个提示数字 k，然后将 ijk 读入即可。

② 连续性约束：

由于蜂窝数独特有的连续性，即每行/对角线所填的一串数字都要满足其升序排列是一串公差为 1 的等差数列（如 1,2,3,4,5 或 2,3,4,5,6,7,8）因此不同长度的行/对角线有不同的必填数字。

表 4.4 长度与必填数字的关系

行/对角线长度	必填数字
5	5
6	4,5,6
7	3,4,5,6,7
8	2,3,4,5,6,7,8
9	1,2,3,4,5,6,7,8,9

在必填数字方面，我们以第三行进行说明。第三行的长度是 7，因此必填 3,4,5,6,7.所以我们很容易可以构建出如下的约束：

13 323 333 343 353 363 373 0　　第 3 行含有 3

314 324 334 344 354 364 374 0　　第 3 行含有 4

… …

317 327 337 347 357 367 377 0　　第 3 行含有 7

其他行/对角线的归约也可以仿照这个方式。

此外，显然数独游戏格局中，每一行/对角线不可以重复出现数字。因此也有必要在此方面进行约束条件的读取。依旧以第三行为例：

第 3 行任两格不能填同一个数字，所以有：

-311 -321 0　　前两格不同时为 1

-311 -331 0　　第 1 与第 3 格不同时为 1

… …

-311 -371 0　　第 1 与第 7 格不同时为 1

… …

③ 选填方案约束（★难点★）：

这一部分是整个约束读取部分难度最大的部分。在《2023 秋-程序设计综合课程设计任务书（2022 级）》中（以下简称《任务书》）中只介绍了第三行的填法，但最为重要的推导过程却完全没有给出来，因此该部分将重点介绍由容易得出的**析取式**推导难以分析的**最简合取式**（即最终需要归约的合法格式）的**思路**。

首先仍以第三行为例，本条线上必填 3，4，5，6，7 五个数字，1，2，8，9 为选填数字，但根据连续性约束，只有 3 个选填方案：填 1 与 2，或填 2 与 8，或填 8 与 9。选填方案用布尔公式可表示为：**(1∧2)∨(2∧8)∨(8∧9)**。

这是一个**析取式**，我们需要把它转化为**最简合取式**。《任务书》中只给出了公式 **p∨(q∧r) = (p∨q)∧(p∨r)**，但拿到这个公式，我们或许可以推出第三行的合取结果，但是在面对较难的长度为 6 的行/对角线或者难度极大的长度为 5 的行/对角线，根本让人无从下手！因此笔者将给出详细推导过程。

首先直接用公式推导第三行（对角线理论上是一致的，不再推导）。

注意：为便于描述问题，将∧省略，将∨定义为 + .

(1∧2)∨(2∧8)∨(8∧9)

= (12)+(28)+(89) //用定义简化

= (12+2)(12+8)+(89) //用公式先分解前两项

= ((1+2)(2+2)) ((1+8)(2+8))+(89) //展开前两项

= (1+2+8)(2+2+8)(1+8+8)(2+8+8)(1+2+9)(2+2+9)(1+8+9)(2+8+9) //完全展开

= (1+2+8)(2+8) (1+8)(2+8)(1+2+9)(2+9)(1+8+9)(2+8+9) //变元化简

= (1+8)(2+8)(2+9)(1+2+8) (1+2+9) (1+8+9)(2+8+9) //变元吸收

= (1+8)(2+8)(2+9) //同上

= (1∨8)∧(2∨8)∧(2∨9)

基于上述过程可以得到第二行的表达式，这个过程更麻烦但依旧可以通过暴力展开的方法解决，答案是**(1∨7)∧(2∨7)∧(3∨7)∧(2∨8)∧(3∨8)∧(3∨9).**

第二行的结果显然复杂多了，这是因为长度越短，可以选择的数字就越多。因此在只有 5 是必填数字的第一行里，通过暴力展开几乎是不可能的事情。

第一行极其复杂。所以它的选填策略要怎么解决呢？不妨观察一下不同长度对应的最简合取式，如表 4.5 所示，我们会有一个有趣的发现。

表 4.5 长度对应的合取式

行/对角线长度	简化表达之后的合取式
8	(1+9)
7	(1+8)(2+8)(2+9)
6	(1+7)(2+7)(3+7)(2+8)(3+8)(3+9)
5	???(未知

首先看每一个长度的表达式里，每个括号的起始和末尾数字：

长度为 8：起始数字是 1，末尾数字是 9

长度为 7：起始数字有 1,2，末尾数字有 8,9

长度为 6：起始数字有 1,2,3，末尾数字有 7,8,9.

所以我们不妨归纳：长度为 5，起始数字有 1,2,3,4，末尾数字有 6,7,8,9.

在长度为 7 的情况下，9 出现 1 次，8 出现 2 次。长度为 6 的情况下，9 出现 1 次，8 出现 2 次，7 出现 3 次。所以我们可以归纳出，在长度为 5 的情况下，9 出现 1 次，8 出现 2 次，7 出现 3 次，**6 出现 4 次**！

那么结果应该是(?+6)(?+6)(?+6)(?+6)(?+7)(?+7)(?+7)(?+8)(?+8)(?+9).

而每个问号代指的数字也是由规律进行归纳的。

我们直接观察长度为 6 的情况，末尾数字 7 出现了 3 次，它对应的起始数字是 1,2,3.而末尾数字 8 出现了 2 次，它对应的起始数字是 2,3.最后的末尾数字 9 只出现了 1 次，它对应的起始数字是 3.

所以可以大胆猜想：长度为 5 的情况，末尾数字 6 应该对应了 1,2,3,4；末尾数字 7 对应了 2,3,4；末尾数字 8 对应了 3,4；末尾数字 9 对应了 4。

基于以上分析、猜想和归纳，我们可以得出长度为 5 时，表达式应该是：

(1+6)(2+6)(3+6)(4+6)(2+7)(3+7)(4+7)(3+8)(4+8)(4+9)

把定义符号取消后得出原始最简合取式，即

(1∨6)∧(2∨6)∧(3∨6)∧(4∨6)∧(2∨7)∧(3∨7)∧(4∨7)∧(3∨8)∧(4∨8)∧(4∨9)

这样就成功建立了最为困难的一个最简合取式。虽然它是通过不完全归纳法得出的，但是结果经过验证是正确的。蜂窝数独游戏格局的归约的最难的部分，笔者认为非此莫属。从该式的复杂度就可以看出，仅仅依靠一个布尔恒等式将会涉及到非常复杂的计算（仅仅是长度为 6 的情况，最后完全展开后就有数十个二元项，那么长度为 5 的情况的复杂度将难以想象），这是难以办到的。

- **映射的思路与建立**

值得一提的是，由于这里的数独游戏格局并不是一个矩形，而笔者用于实现游戏格局的方式为二维数组，首先有必要对建立二维数组格局进行说明。

这里采取用矩阵（二维数组）**规模为 9x17，行列下标均为从 1 开始**。

由于蜂窝数独中相邻行的数字是参差的，所以在矩阵中采取了**间隔存储**。如图 4.3 所示即是图 4.2 里蜂窝数独格局的矩阵表示。左图是原始矩阵（原二维数组），六边形内部已经可以看出蜂窝数独格局的雏形。0 元在这里无意义，只是作为间隔使用，以此实现相邻行的参差效果。右图是左图矩阵中省略非必要的 0 元后得到的精炼图示。此时已经很容易看出蜂窝的形状了。

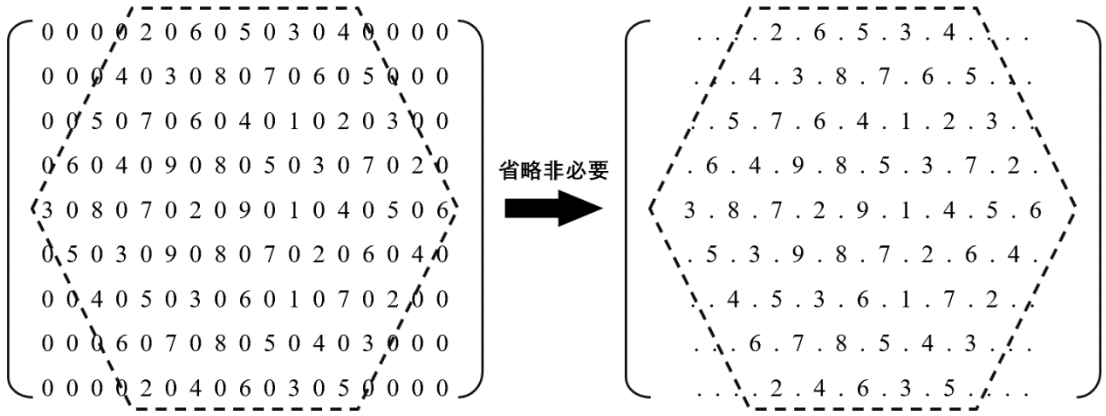


图 4.3 矩阵—蜂窝数独

由于蜂窝数独格局和矩形二维数组格局的巨大差异，我们难以直接归约。因此必须建立一个**映射关系**，即通过数组里的下标 i，j 映射到蜂窝数独格局里的第 p 行第 q 个，否则想要直接进行归约是非常困难的。

其中 i→p 的映射 ***f***：***f***（***i***）= ***i.***这是显而易见的。

同时由于蜂窝数独游戏里，不同的行有不同的数字个数，因此需要构建从 i，j→q 的二元映射。公式如下：

$$g: g(i,j) = (j + 1 - |5 - i|) / 2.$$

该公式的推导也很有意思。笔者在推导公式时发现难以直接得出该公式，但意外发现该公式的逆向是很容易的。即通过映射 p,q→j 来表示，然后以映射 ***f*** 把 p 转为 i，最后从方程 h（p, q）= j 与 p = i 中解出 q 即可。

基于以上思考，我们可以尝试列出方程 h = j. 观察蜂窝格局里第 p 行的第一个数字在矩阵的下标位置，我们不难发现，它在矩阵格局中的下标为 （p，| 5 – p | + 1）.由于矩阵的间隔存储，第 p 行第 q 个数字的下标就为（p，2 *（q-1）+

| 5 – p | + 1）.所以有 j = 2 *（q-1）+ | 5 – p | + 1，解出 q，代入 p = i，即可得到映射 g：$g(i,j) = (j + 1 - |5 - i|) / 2$.

有了映射的构建和上述的归约方法规律，就可以把数独归约到 CNF 文件了。

4.2 系统测试

4.2.1 SAT SOLVER 模块功能测试

①READ CNF

读取正确格式的CNF文件后，显示读取成功；如果读取的文件名称不合法，或者是文件内容不合法，则显示读取错误。图4.4给出了读取成功的样例。图4.5给出了读取名称不合法的样例（名称是111，而没有.cnf后缀），图4.6给出了文件内容不合法的样例（乱码）。

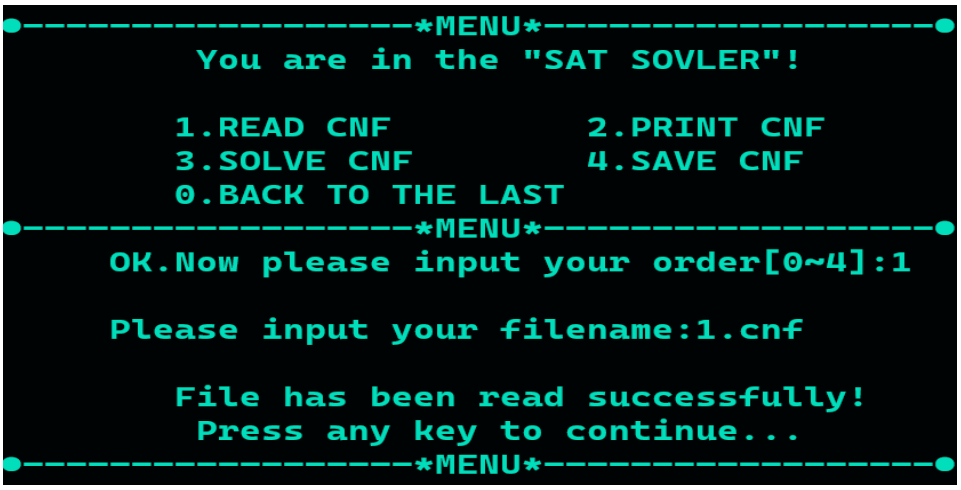


图 4.4 读取成功

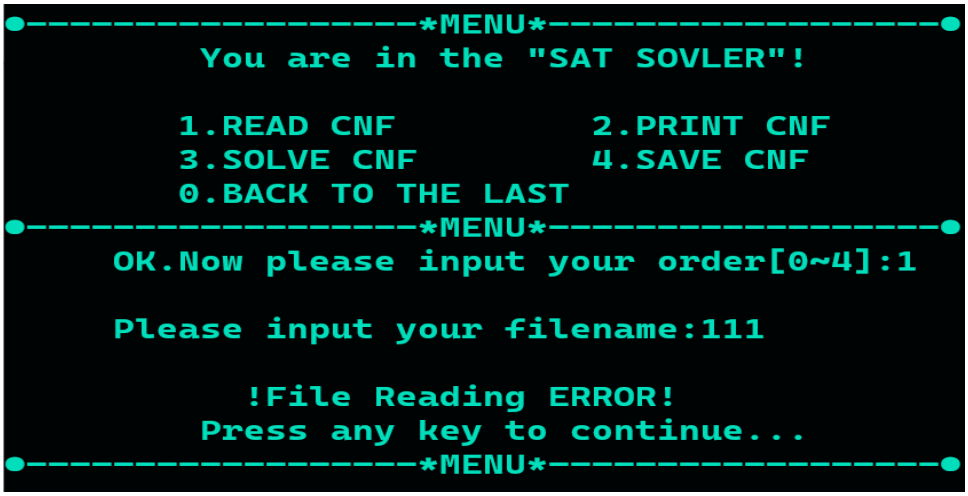


图 4.5 名称不合法

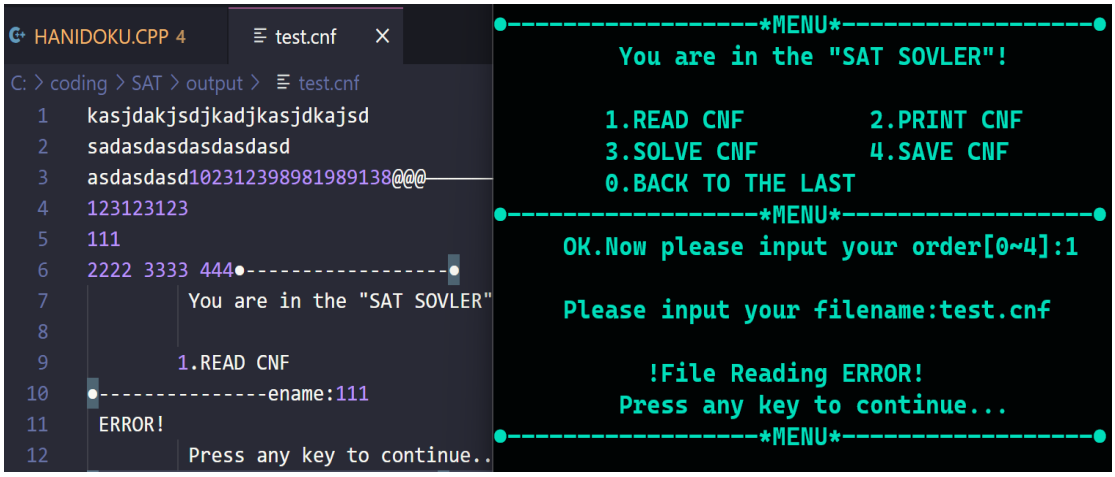


图 4.6 乱码文件

②PRINT CNF

该功能首先需要 READ CNF 成功，前文已经叙述过，这里默认成功。图 4.7 是打印测试，对测试 5 的文件内容进行了简化修改后进行测试：

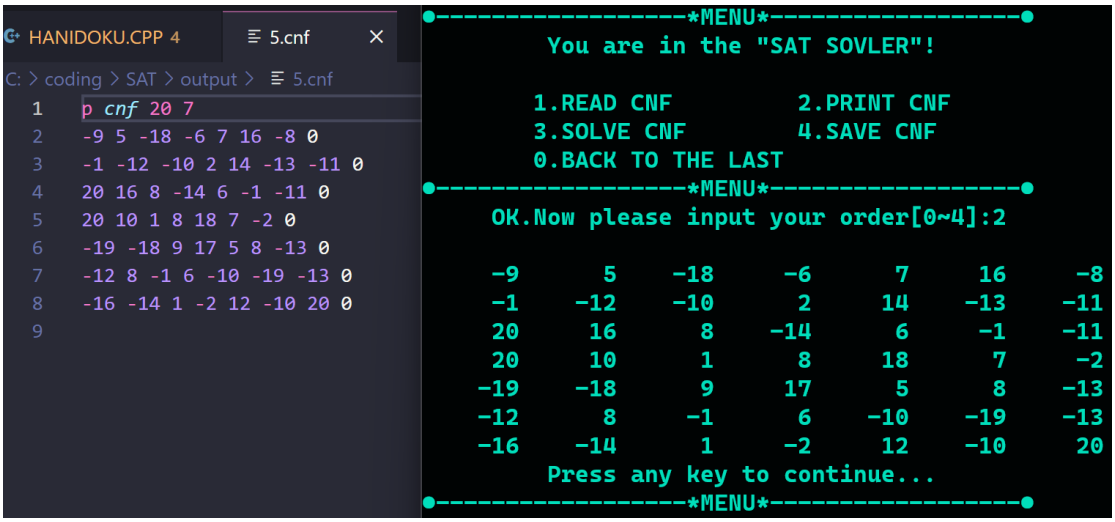


图 4.7 打印文件

③SOLVE CNF

在测试 CNF 文件运行时间时，选取的优化策略是 **MOM-Optimized Version**，这是因为第一个策略并不成熟、第二个不具有普适性、第三个是该策略的弱化版。

(1)功能和性能测试

表 4.6 功能和性能测试

变元数	子句数	算例名称	优化前(ms)	优化后(ms)	优化率
20	91	sat-20.cnf	1	0	100%
30	420	unsat-5cnf-30.cnf	63	15	76%
181	3151	ais10.cnf	1499	49	97%
303	2851	sud00009.cnf	4207	32	99%

(2)S 型算例测试

表 4.7 S 型算例测试

变元数	子句数	算例名称	优化前(ms)	优化后(ms)	优化率
20	1532	7cnf20_90000_90000_7.shuffled-20.cnf	31	16	48%
20	91	problem1-20.cnf	1	0	100%
50	80	problem2-50.cnf	16	0	100%
100	340	problem3-100.cnf	738	31	96%
50	100	problem6-50.cnf	127	0	100%
50	300	problem8-50.cnf	16	0	100%
100	200	problem9-100.cnf	78233	0	100%
100	600	problem11-100.cnf	188	16	91%
25	100	tst_v25_c100.cnf	1	0	100%

(3)M 型算例测试

表 4.8 M 型算例测试

变元数	子句数	算例名称	优化前(ms)	优化后(ms)	优化率
200	320	problem5-200.cnf	Inf	63	100%
200	1200	problem12-200.cnf	Inf	423	100%
301	2780	sud00001.cnf	36227	31	100%
303	2851	sud00009.cnf	2158	16	99%
232	1901	sud00012.cnf	492	157	68%
308	2911	sud00021.cnf	31427	206	99%
301	2810	sud00079.cnf	5076	47	99%
224	1672	sud00082.cnf	987	95	90%
297	2721	sud00861.cnf	63	0	100%

(4)L 型算例测试

表 4.9 L 型算例测试

变元数	子句数	算例名称	优化前(ms)	优化后(ms)	优化率
6498	130997	ec-vda_gr_rcs_w9.shuffled-6498.cnf	Inf	Inf	-
1075	3152	eh-dp04s04.shuffled-1075.cnf	3093	280	91%
625	76775	eh-vmpc_25.renamed-as.sat05-1913-625.cnf	Inf	Inf	-
841	120147	eh-vmpc_29.renamed-as.sat05-1916-841.cnf	Inf	38170	100%
3176	10297	e-par32-3.shuffled-3176.cnf	Inf	2992	100%
50073	210223	mc-sha0_36_5-50073.cnf	Inf	Inf	-
1024	161664	m-vmpc_32.renamed-as.sat05-1919-1024.cnf	Inf	Inf	-

***在针对大型算例，优化后的策略依旧显得有些无力。**

4.2.2 HANIDOKU GAMES 模块功能测试

这一部分主要介绍数独游戏模块。数独实际界面如图 4.8 所示：

左图是游戏前，右图是游戏完成之后。在挖洞处，其字符会被亮绿色标出。

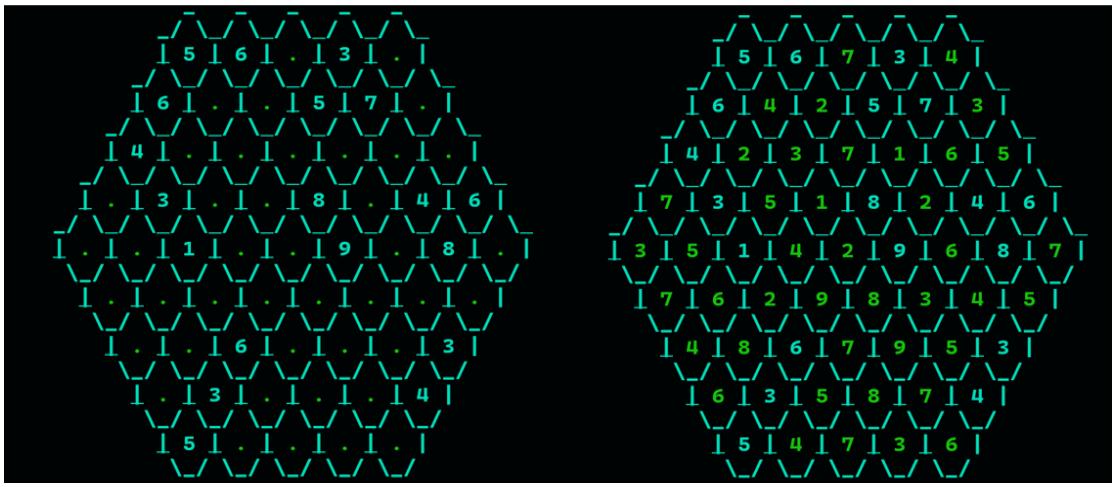


图 4.8 数独游戏界面

① 游戏游玩测试

(1) 填入数字：

只有在空缺的地方可以填上数字。填数字的方法是输入一个有序三元组（i,j,k），代表的是在第 i 行第 j 个填上数字 k. 整数 k≤0 或 k≥10 都不行；填在未挖洞的地方也不行。报错时会警告并要求用户重新填充。

图 4.9 是测试结果：左图是填入成功的样例；中间是 k 值不合法的样例；右图是位置不合法的样例。

图 4.9 填充测试

(2) 获取提示：

在游玩过程，输入 11，i，j 可以获取第 i 行第 j 个数字的值。

如图 4.10，左图是获取第 1 行第 1 个数字，答案是 6；中间是获取第 1 行第 2 个数字，答案是 7；右图是获取已知数字，虽然没有用，但是也保证了严谨性。

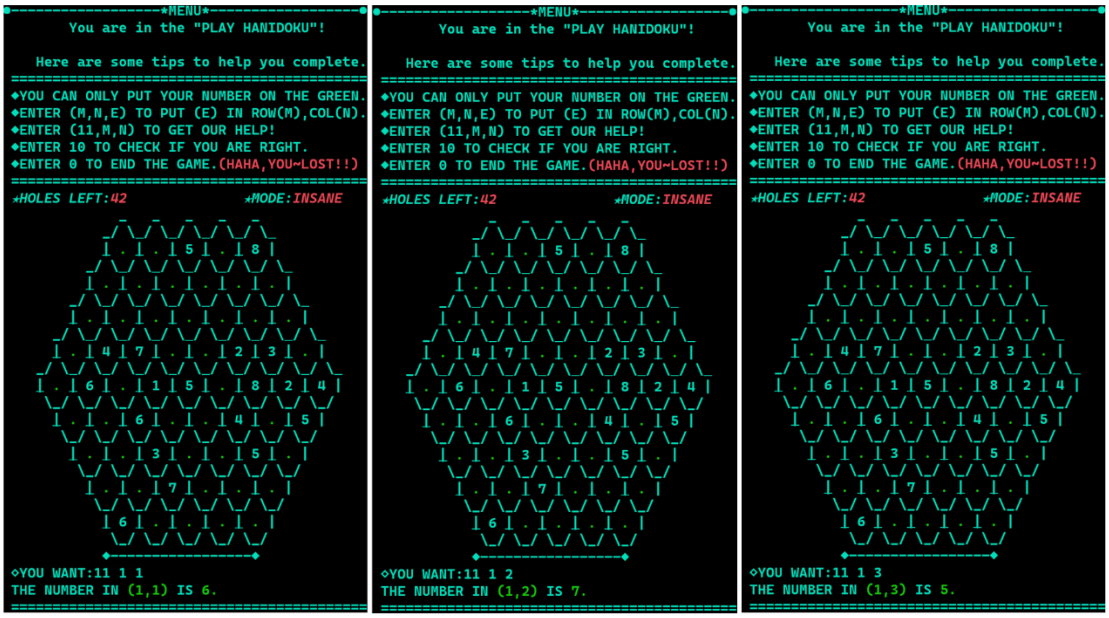


图 4.10 获取提示测试

(3) 检验正确性：

在游玩过程，数入10，i，j可以查看第i行第j个数字的填充是否正确。

如果没有填充则显示答案错误；如果填充错误数字也显示错误；只有填充了正确的数字，才表示答案是正确的。图4.11左一是没有填充的样例；左二是正确样例；右一是填充错误的样例；右二是数独游戏的正确解。

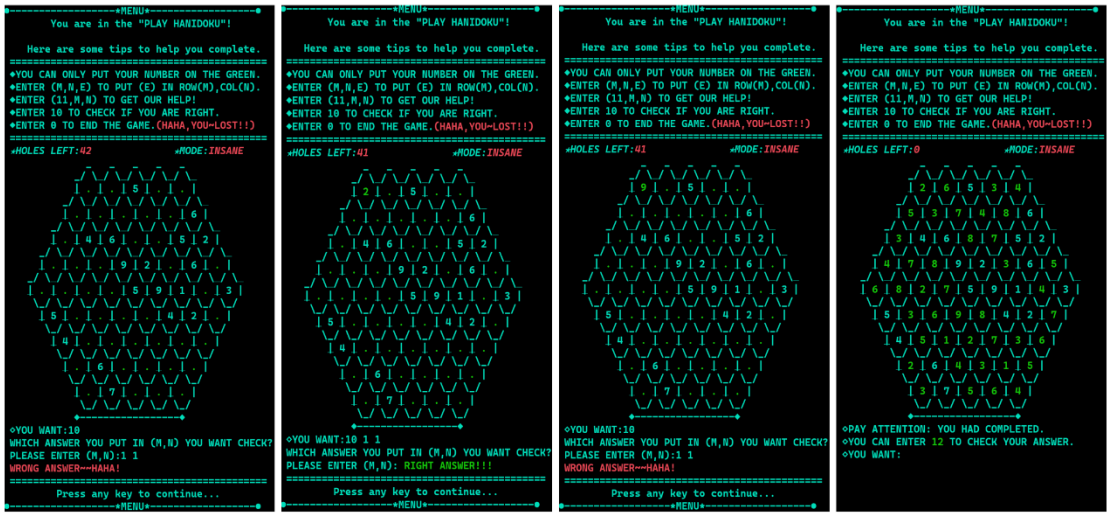


图 4. 11 检验正确性测试

② 保存数独文件

在 HANIDOKU GAMES 模块中也可以把当前游玩的数独游戏归约为 CNF 文件（且必须是 CNF 文件，否则无法读取）并解决该文件。本部分主要介绍保存部分，原理已经在 4.1.2 节的第 10 部分给出。这里不再赘述。

该部分需要在 HANIDOKU GAMES 的界面输入指令 4 进入，然后系统要求输入文件名，这里输入任何文件名都可以保存。图 4.12 是保存成功的示意图。

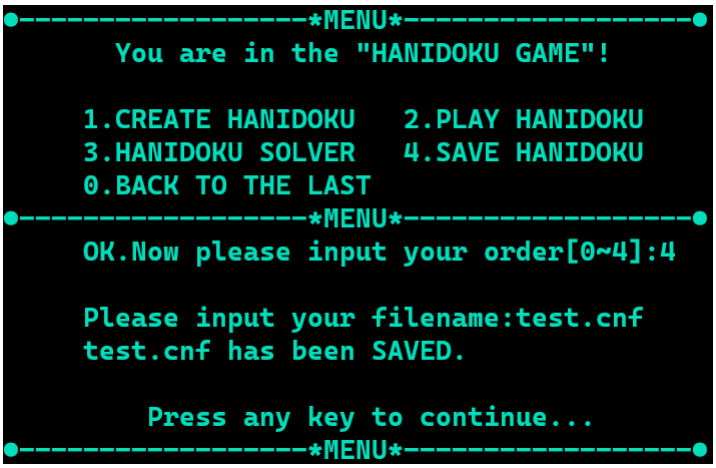


图 4.12 保存文件成功

可以从图 4.13 看到，刚刚创建的文件“test.cnf”已经被保存下来了。

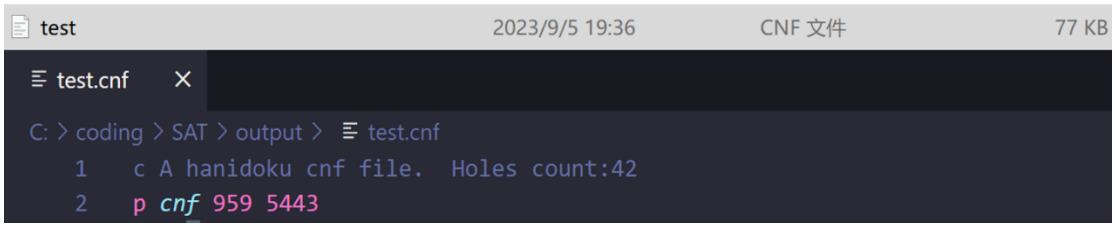


图 4. 13 查看文件是否被保存

③ 解数独文件

这一部分其实和前方 SAT SOLVER 模块的 SOLVE SAT 基本一致，只是这里不再需要选取解决方案，系统会自动选择最适合解数独的 **Maximum Positive Number of Occurances** 方案来运行。这里也需要先读取文件，读取失败不行。只有输入了正确的文件格式才可以运行。

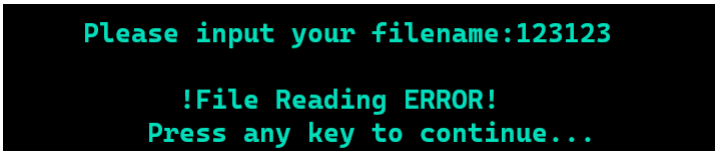


图 4.14 打开文件失败

如图 4.15，以 test.cnf 为例，文件显示读取成功并且在 63ms 内给出了解。

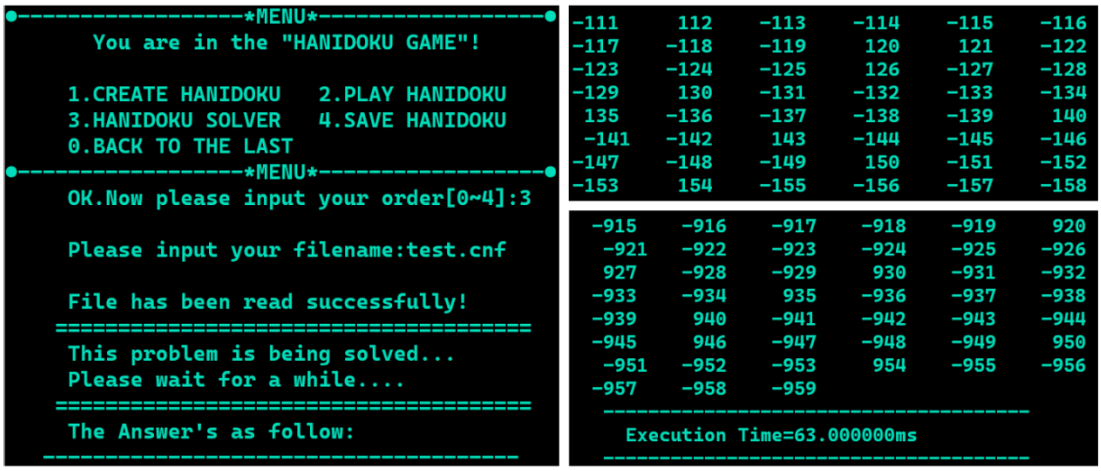


图 4.15 打开文件成功

5 系统特色分析

笔者在实现程序的过程中也加入了大量全新设计（并非优化设计），这些设计使得可以使界面更独特，也可以使用户的体验更加完善。不过遗憾的是笔者并没有学习JavaScript，因此无法实现更加高级的 js 界面。

特色1. 字体颜色调整

在整个系统中，在cmd窗口出现的99%字体都是加粗的蓝色字体，字体大小为12号。相比于普通的白色字体，这种字体更醒目，可以使用户更快地捕捉到屏幕上呈现的内容，同时蓝色字体配上黑色的背景，个人认为有一种科技感，可以带来良好的审美体验（**个人看法，审美是具有很大的个体差异的！**）。

cmd窗口的配色方案如图5.1所示，其实只更改了字体和选中背景。

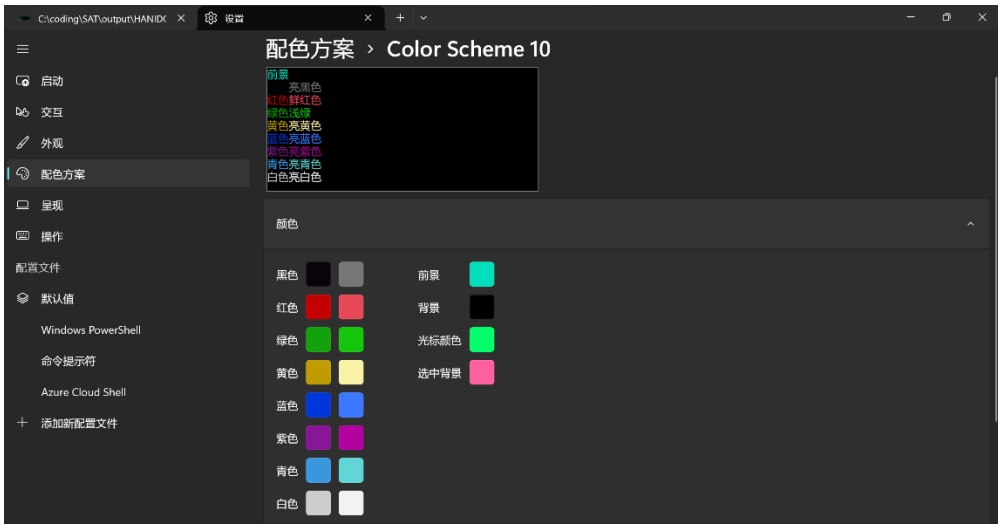


图 5.1 配色方案

特色2. 蜂窝数独格局优化

在《任务书》中，其实出示了一种蜂窝数独格局的样例，笔者在其他同学的程序里也看到了一些不同的格局呈现方式。但笔者设计了更加精细的格局。

如图5.2，左图是《任务书》设计的格局。右图是笔者的优化格局。在可以填充数字的地方用亮绿色标出（这一点在4.2.2已经提过），此外还对边框进行了优化。笔者在蜂窝的每一个“洞”都设计了上下界限，利用的是下划线。这样就可以使得整个格局更加像一个自然界意义上的蜂巢，而且也更美观。

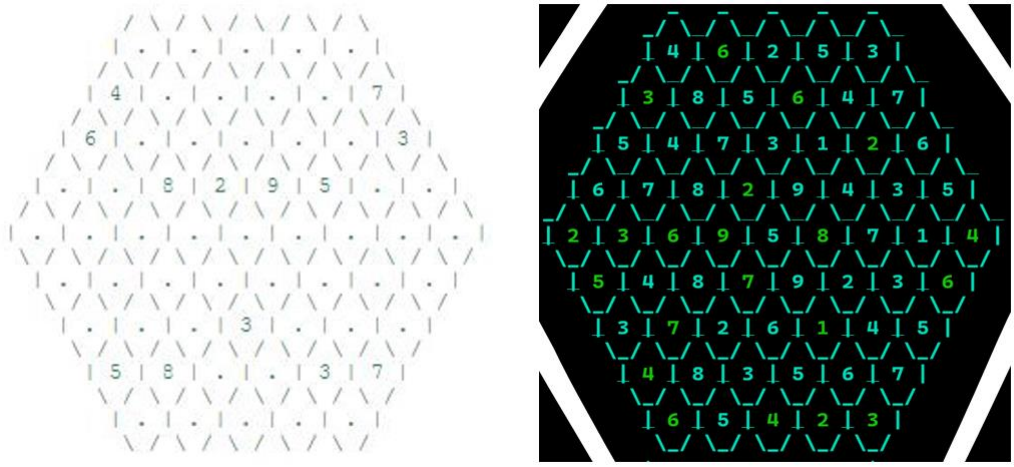


图 5.2 蜂窝数独格局优化

特色3. 游戏“作弊”指令

在制作HANIDOKU GAMES模块程序调试过程中，为了便于调试，笔者在PLAY HANIDOKU功能区域增加了一项指令，输入该指令可以直接读入游戏的正确答案。在完成整个程序的设计后，笔者并没有把这条指令删除，而是把它放进了switch语句里的一个非常隐秘的开关里。

该指令因为能够一次性读取正确答案且没有任何限制，所以被命名为“作弊”指令。**触发方式**是在数独游玩界面里输入“114514”（没有任何提示）。

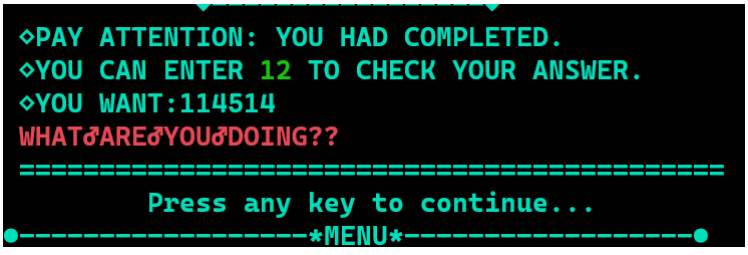


图 5.3 作弊指令

效果见图 5.4。

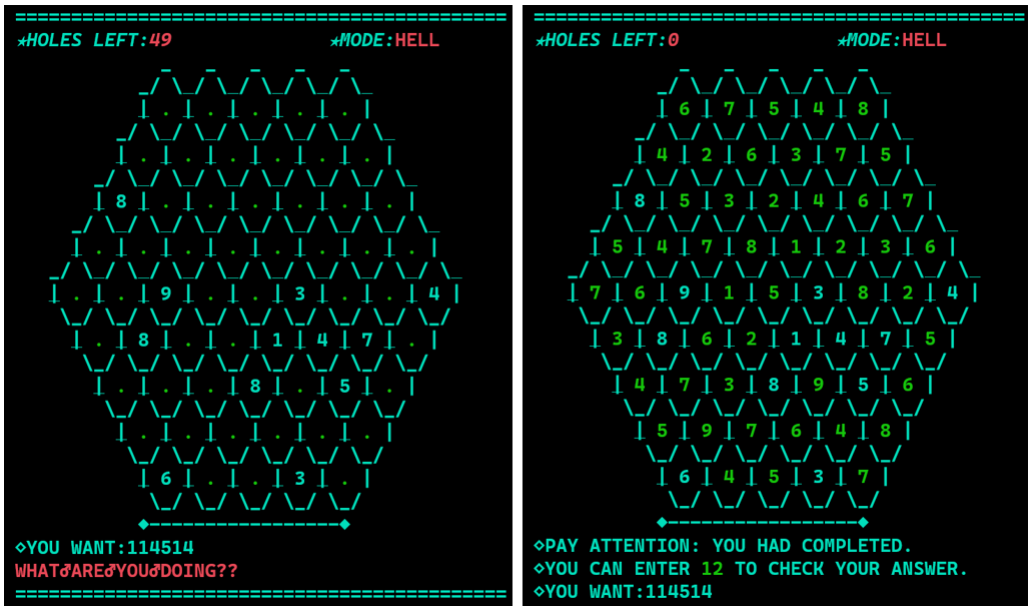


图 5.4 作弊效果图

特色4. “YOU WIN! ” 与 “GAME OVER”

YOU WIN! : 在游玩过程中，系统会实时判断整个格局剩余的空缺数目。如果没有空缺数目，这说明玩家已经基本上完成了游戏。系统会提示输入 12 来查看数独终局是否正确，如果正确则进行下一步操作。

如果玩家的终局的确是正确的，首先将输出“NO!!YOU WIN!!!”（玩笑话），之后输出玩家从开始到输入 12 检查正确后所历经的时间，如图 5.5 所示。

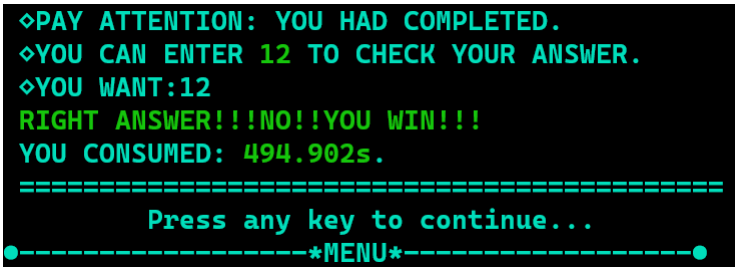


图 5.5 游戏胜利

之后系统自动跳转至专属胜利界面“YOU WIN!”，如图 5.6 所示。

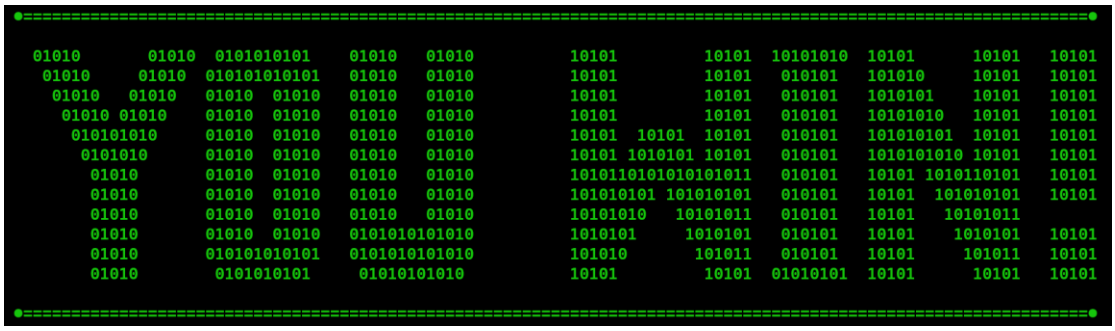


图 5.6 YOU WIN!

GAME OVER: 游戏有胜就有负。如果玩家在游戏途中想要退出（即输入 0），那么将被认为是“出局”。这时系统将在玩家输入 0 后自动打印出立体字符“GAME OVER”。字体和“YOU WIN!”不同，是红色闪烁的，如图 5.7 所示。

该字符采取了全新的设计，和上方的绿色 01 代码不同，这里的字符更像是三维的，这是利用了“_/”的特点，在二维屏幕上呈现 3D 效果。

图 5.7 GAME OVER

特色5. “骷髅头”

如果用户在进入系统主界面时什么都不做就退出系统，系统会在退出之前打印出一个蓝色的骷髅头字符画，配文是“Thanks for your use!”和“宇宙很大，生活更大，也许以后还有缘再见.”（后者语出《三体 • 死神永生》，当时笔者在暑假里阅读了《三体》系列）。

骷髅头的形象设计是原创的，颜色是蓝色的，也采用了01代码背景模式，见图5.8.

图 5.8 骷髅头

该部分小结：

这一章并非任务要求，而是在课设任务书要求之外自行加入的。这些小设计并不是硬件意义上的优化，但也可以丰富用户的交互体验。更重要的是可以体现笔者在制作程序时的一些奇特的灵感和想象，笔者认为这些东西应该保留下来并且专门开辟一章的篇幅来对它们进行一一介绍。而从另外的意义上讲，在丰富用户体验的同时，这些设计也让笔者在设计程序的过程中体会到了极大的成就感和快乐。

6 总结与展望

6.1 全文总结

在整个程序设计中，笔者执行了以下工作内容：

1. 搭建系统框架

在本次程序设计中，笔者按《任务书》要求构建了符合要求的系统框架，将系统分为了两个大板块，分别是 SAT SOLVER 与 HANIDOKU GAMES。两大板块其中又分为了多个子板块，这些子板块已经在前文进行了非常详细的介绍。每个板块之间都需要用户的输入作为桥梁去沟通。

2. 设计基于 DPLL 的 SAT 求解器

这部分属于系统第一大板块的主功能。笔者在熟悉了《任务书》要求之后，查阅资料和阅读文献，最终成功还原了 DPLL 算法并且利用 DPLL 算法解决了大量测试算例以及检验算例，成功搭建了基于 DPLL 的 SAT 求解器。

3. 对 DPLL 算法进行优化

DPLL 原始算法在解决中大型算例时非常吃力，其原因很大程度上在于选取变元策略。因此笔者特别设计了四种不同的优化选元策略，这里面有一种策略是专门针对数独文件进行选元的，同时其他三种普适性策略也的确大大提升了 DPLL 算法的解决速度和工作效率。

4. 分析优化策略的可行性

在报告中对大量的 CNF 文件进行了测试，并将优化前和优化后的时间进行了对比并且给出了每一个算例的优化率，经过测试后，优化率大多在 90%以上，但仍然有部分算例在优化后也难以或根本无法成功解决。

5. 设计数独并归约

这部分属于 HANIDOKU GAMES，笔者实现了数独格局的随机生成，并且赋予了数独游戏基本的游玩功能、全新的操作功能以及更加美观的界面设计。其中，数独格局也可以通过算法来归约为一个 CNF 文件，在此之后也可以调用 SAT 求解器对其进行求解。而关于归约功能，笔者专门设计了一种映射方式来存储数独，并且利用数学计算和归纳法成功推出了每一行的合取式。

6.2 工作展望

在今后的研究中，围绕着如下几个方面开展工作：

1. 加强程序的交互功能

在本次程序设计中，笔者按《任务书》要求实现了用户与系统之间的交互。但是在完成后发现还是有很多地方有一些缺憾。比如通篇都采用了大写字母导致新用户难以直接理解，对于打印字体的居中等格式仍然不够完美。今后笔者会在这个方面进行改进。

2. 学习 JavaScript

笔者在向助教检查程序时发现有一个“ j s 界面”的相关评分栏目。最开始根本不知道这是什么，最后在网上搜索后，笔者发现自己的眼界实在是太低了。笔者认为一旦实现了 JavaScript 界面，那么整个程序的美观程度将会得到一个极其巨大的飞跃。可惜由于才疏学浅，未能及时实现。今后笔者在其他课设作品中一定会争取拿下这一块内容。

3. 对 DPLL 的理解

笔者在进行课设之前根本不知道 SAT 问题和 DPLL 算法，最开始也无从下手。但是在参考《任务书》和查阅资料后对其有了初步的了解。更进一步的，在进行整个程序设计时尤其是设计优化策略时，笔者对于 DPLL 算法有了更加深入的理解。这激发了笔者探索未知领域的兴趣。

4. 探究 SAT 可满足性问题

笔者最开始看到 SAT 问题的形式，认为只是一个穷举就可以解决的问题，但是在亲自尝试了之后才发现，穷举根本无法满足如此巨大的时间开销。通过这次程序设计笔者也明白了 NP 完全问题的极大困难性和其重要的应用领域，以及看到了世界各国在攻克这种问题所付出的巨大努力。

5. 数独的设计与归约

这次课设其实是笔者第一次接触数独游戏，而且第一次接触就涉及到了难度较大的非常规的蜂窝数独游戏。这对笔者而言无疑是一个艰巨的任务和挑战。但是在最后克服了一切困难后，笔者收获了成就感，更激发了对数独游戏的兴趣。笔者认为这种设计精妙的益智小游戏非常适合自己。

7 体会

这项程序设计开始于 2023 年 7 月 29 日，结束于 2023 年 8 月 29 日，刚好历经了一个月的时间，前前后后更新了十几次版本才呈现出最后的结果。

最开始了解到这个任务其实是在 2023 年五月左右，当时数据结构老师就已经向我们透露，大二开学时将会有一个两周时长的程序设计，需要我们暑假就进行准备，当时我并没有太在意。

当我着手开始准备的时候，两个任务设计对我而言都感觉像是难以逾越的大山。我在这两个选项中徘徊了很久，最后还是决定进行数独方面的程序设计。因为另一个对我而言实在难以看懂，而在设计数独游戏这一块我认为比较有趣味性。

在制作课设的期间，还是遇到了许多问题和困难。如最开始设计的递归算法耗时非常大，根本无法等到跑完出结果；制作的二维链表忘记释放内存导致经常程序运行到一半就爆掉；函数参数引用经常出错…由于程序体量较大，因此一旦出现问题就会涉及到十分复杂的调试，这一点真的让人头疼。

但是设计也是很快乐的。我尽量用自己的方式把程序的界面和交互制作得更加漂亮，虽然没能实现 js 界面等更加高级的操作，但是我的确尽力了。我还是很喜欢自己设计的数独格局界面的。另外附加的一些实用功能和恶趣味功能也让我在设计时非常愉快，在逐行打印字符画时更是妙趣横生。

对 DPLL 算法进行优化策略时我也查阅了大量资料和文献。通过文献我知道了启发式策略，并根据这种策略，顺着设计出了一系列选元算法，最后看到代码运行效率成功提高时也收获了大量的喜悦。

暑假里除了完成程序设计，我也自学了其他知识：如逻辑表达式化简（这对我进行数独归约具有极大的帮助）、卡诺图化简、其他数据结构的实际应用（刷题网站）。而这些收获终于会成为我前进的动力和能源，让我飞得更高。

感谢您的阅读！再见！

2023 年 9 月 6 日

参考文献

[1] 张健著. 逻辑公式的可满足性判定—方法、工具及应用. 科学出版社，2000

[2] Tanbir Ahmed. An Implementation of the DPLL Algorithm. Master thesis, Concordia University,Canada,2009

[3] 陈稳. 基于 DPLL 的 SAT 算法的研究与应用.硕士学位论文，电子科技大学，2011

[4] Carsten Sinz. Visualizing SAT Instances and Runs of the DPLL Algorithm. J Autom Reasoning (2007) 39:219–243

[5] 360 百科：数独游戏 https://baike.so.com/doc/3390505-3569059.html

Twodoku： https://en.grandgames.net/multisudoku/twodoku

[6] Tjark Weber. A sat-based sudoku solver. In 12th International Conference on Logic for Programming, Artificial Intelligence and Reasoning, LPAR 2005, pages 11–15, 2005.

[7] Ins Lynce and Jol Ouaknine. Sudoku as a sat problem. In Proceedings of the 9th International Symposium on Artificial Intelligence and Mathematics, AIMATH 2006, Fort Lauderdale. Springer, 2006.

[8] Uwe Pfeiffer, Tomas Karnagel and Guido Scheffler. A Sudoku-Solver for Large Puzzles using SAT. LPAR-17-short (EPiC Series, vol. 13), 52–57

[9] Sudoku Puzzles Generating: from Easy to Evil.

http://zhangroup.aporc.org/images/files/Paper_3485.pdf

[10] 薛源海，蒋彪彬，李永卓. 基于“挖洞”思想的数独游戏生成算法. 数学的实践与认识,2009,39(21):1-7

[11] 黄祖贤. 数独游戏的问题生成及求解算法优化. 安徽工业大学学报(自然科学版), 2015,32(2):187-191

附录

```
/****程序文件导入****/
#include <math.h>
#include <stdbool.h>
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
/********************/

/****程序常量定义****/
#define OK 1
#define ERROR 0
#define HANIDOKU 1//表示选择数独游戏
#define SAT 0//表示选择解决 SAT 问题
/********************/

/****程序变量定义****/
typedef int status;
//返回 int 函数错误参数用数
const int inf = 2147328174;
//分别为 cnf 文件中 bool 变元数，子句数，数独解数，选元方式
int Bool_cnt,Clause_cnt,Ans_num,BranchRule;
char fileName[2222],HANIDOKUFileName[2222];//文件名
int MARK[9+1][17+1]={0};//标记数独
int fullBoard[9+1][17+1];
int board[9+1][17+1];
/********************/

/****数据结构定义****/
typedef struct cNode{
    int data;
    cNode *next;
}cNode;//链表结点定义
typedef struct clause{
    int num;
    cNode *head;
    clause *next;
}clause;//二维链表表头
/********************/

/****程序函数声明****/
int read(FILE *fp);
```

```
int translaterow(int a);
int getMaxBool(clause *cnf);
int translatecol(int i,int j);
int getNextBool_MOM(clause *cnf);
int getMaxBool_verhanidoku(clause *cnf);
int getNextBool_Positive_MOM(clause *cnf);
int getNextBool_MOM_optimized(clause *cnf);
//////////////////////////////////////////////////////
bool isUnitClause(clause *cl);
bool emptyClause(clause *cnf);
bool DPLL(clause *&cnf, int *v, int satORhanidoku);
//////////////////////////////////////////////////////
void HELLO();
void REFLECT();
void GAMEOVER();
void WIN();void GHOST();
void TEMP(int &a,int &b);
void FLIP2();void FLIP3();
void FLIP4();void FLIP5();
void FLIP6();void FLIP1();
void CREATE();void ROTATE();
void printHANIDOKU(int a[][18]);
void destroyClause(clause *&cnf);
void createFullBoard(int a[][18]);
void removeNode(clause *cl, cNode *&nd);
void dig(int a[][18],int b[][18],int holes);
void removeClause(clause *&cnf, clause *&cl);
void getForgery(clause *&forgery, clause *cnf);
void deleltеSingleClause(clause *s, clause *&cnf);
void game(int fullBoard[][18], int board[][18], int holes);
void gameStart(int fullBoard[][18], int board[][18], int modes);
//////////////////////////////////////////////////////
status SaveRes(int s, int *v, double t);
status addClause(clause *cl, clause *&cnf);
status ReadCNF(clause *&cnf, int satORhanidoku);
status transhanidokuToCNF(int a[][18], int holes);
status CreateGame(int a[][18], int row, int col);
status checkmark(int test,int m1[10],int m2[10],int m3[10],int x1,int x2,int x3);

int main()
{
    clause *CNF;//二维链表表头
    int order = -1;//总指令
    int order1 = -1;//SAT 问题指令
```

```
int order2 = -1;//HANIDOKU 指令
int flag_cnf = 1;
int * value;
int flag_hdk;
int flag_game;
int modes;
double TIME;//解决问题所用时间
clock_t start, finish;
bool res;
flag_game=1;

while (order)
{
    system("cls");//清屏
    printf("        \033[1;36;5m★SAT-HANIDOKU~v1.18 FMY★\033[0;31;0m\n");
    //HELLO();
    printf("\n●-----------------*MENU*-----------------●\n");
    printf("       ^_^Welcome to this program !^_^\n");
    //GHOST();
    printf("   ====================================\n");
    printf("      ↓↓↓Please select the orders:↓↓↓\n\n");
    printf("     1.SAT SOVLER    2.HANIDOKU GAME\n");
    printf("       \033[0;31;1m0.EXIT(DON'T YOU LIKE IT?)\033[0;3;0m\n");
    printf("●-----------------*MENU*-----------------●\n");
    printf("    OK.Now please input your order[0~2]:");
    scanf("%d",&order);
    printf("\n");
    switch (order)
    {
    case 0:{
        GHOST();
        printf("           Thanks for your use!\n\n    宇宙很大,生活更大,也许以后
还有缘相见.\n");
        printf("●-----------------*MENU*-----------------●\n");
        return 0;
    }//case0

    case 1:{
        while (order1)
        {
            system("cls");//清屏
            printf("\n\n●-----------------*MENU*-----------------●\n");
            printf("        You are in the \"SAT SOVLER\"!\n\n");
```

```
printf("       1.READ CNF        2.PRINT CNF\n");
printf("       3.SOLVE CNF       4.SAVE CNF\n");
printf("       0.BACK TO THE LAST\n");
printf("●------------------*MENU*------------------●\n");
printf("    OK.Now please input your order[0~4]:");
scanf("%d",&order1);
printf("\n");
if(order1==0){order1=-1;break;}

switch (order1){
   case 1:{
      printf("    Please input your filename:");
      scanf("%s", fileName);
      if (ReadCNF(CNF, SAT) == OK){
         flag_cnf = 0;
         printf("\n       File has been read successfully!\n");
      }//if ok
      else{
         flag_cnf = 1;
         printf("\n          !File Reading ERROR!\n");
      }//else error
      printf("        Press any key to continue...\n");
      printf("●------------------*MENU*------------------●\n");
      getchar(),getchar();//按任意键以继续
      break;
   }//case1 1

   case 2:{
      if (flag_cnf)
      printf("          CNF File Not Input Yet!\n ");
      else{
         for (clause *lp = CNF; lp; lp = lp->next){
         for (cNode *tp = lp->head; tp; tp = tp->next)
            printf("%7d", tp->data);
            printf("\n");
         }
      }//else
      printf("        Press any key to continue...\n");
      printf("●------------------*MENU*------------------●\n");
      getchar(),getchar();//按任意键以继续
      break;
   }//case1 2

   case 3:{
```

```
                    if (flag_cnf)
                    printf("          CNF File Not Input Yet!\n ");
                    else{
                        printf("   ↓There are 6 choices for you to select↓\n");
                        printf("   ---------------------------------------\n");
                        printf("   1.Maximum Number of Occurances\n");
                        printf("   2.Maximum Positive Number of Occurances\n");
                        printf("   3.MOM                      4.MOM+\n");
                        printf("   5.Randomly Select(NOT RECOMMENDED!)\n");
                        printf("   ---------------------------------------\n");
                        printf("    Please input your choice[1~4]:");
                        scanf("%d",&BranchRule);
                        printf("\n        This  problem  is  being  solved...\n
Please wait for a while....\n");
                    value = (int *)malloc(sizeof(int) * (Bool_cnt + 1));
                    for (int i = 0; i <= Bool_cnt; i++)
                        value[i] = 1;
                    start = clock();
                    res = DPLL(CNF, value, SAT);
                    finish = clock();
                    if (!res)
                    {
                        printf("\n    The Answer's as follow:\n");
                        printf("   ---------------------------------------\n");
                        printf("               No Answer!\n");
                    }

                    else
                    {
                        printf("\n    The Answer's as follow:\n");
                        printf("   ---------------------------------------\n");
                        for (int i = 1; i <= Bool_cnt; i++)
                        {
                            if (value[i])
                            {
                                if(i%6!=0)
                                printf("%6d\t", i);
                                else
                                printf("%6d\n ", i);
                            }
                            else
                            {
                                if(i%6!=0)
                                printf("%6d\t", -i);
```

```
                        else printf("%6d\n", -i);
                    }
                }
                printf("\n");
            }//else
            TIME = (double)(finish - start) / CLOCKS_PER_SEC;
            printf("   ----------------------------------------\n");
            printf("    Execution Time=%lfms\n", TIME * 1000);
            }//else

            printf("        Press any key to continue...\n");
            printf("●------------------*MENU*-----------------●\n");
            getchar(),getchar();//按任意键以继续
            break;
        }//case1 3
        case 4:{
            if (flag_cnf)
            printf("          CNF File Not Input Yet!\n ");
            else{
                if (SaveRes(res, value, TIME))
                    printf("    Save files to *.res of the same name.\n");
                else printf("          Files saving ERROR!\n");
            }//else

            printf("        Press any key to continue...\n");
            printf("●------------------*MENU*-----------------●\n");
            getchar(),getchar();//按任意键以继续
            break;

        }//case1 4

    }//switch1

}//while1

}//case1

case 2:{
    flag_hdk = 1;
    while (order2)
    {
        system("cls");//清屏
        printf("\n\n●------------------*MENU*-----------------●\n");
        printf("      You are in the \"HANIDOKU GAME\"!\n\n");
```

```
            printf("    1.CREATE HANIDOKU   2.PLAY HANIDOKU\n");
            printf("    3.HANIDOKU SOLVER   4.SAVE HANIDOKU\n");
            printf("    0.BACK TO THE LAST\n");
            printf("●-----------------*MENU*-----------------●\n");
            printf("    OK.Now please input your order[0~4]:");
            scanf("%d",&order2);
            printf("\n");
            if(order2==0){order2=-1;break;}
            ////////////////////////////
            switch (order2)
            {
                case 1:{
                    flag_game=0;
                    for(int i=0;i<=9;i++)
                        for(int j=0;j<=17;j++)
                            {board[i][j]=0;MARK[i][j]=0;}//置0

                    printf(" ↓There are some different kinds of modes:↓\n");
                    printf("  ---------------------------------------\n");
                    printf("          1.EASY          2.NORMAL\n");
                    printf("          3.HARD          4.EXPERT\n");
                    printf("          5.CRAZY         6.INSANE\n");
                    printf("          7.HELL          0.CUSTOM\n");
                    printf("  ---------------------------------------\n");
                    printf("Select these modes according to preference[0~7]:");
                    scanf("%d",&modes);
                    if (modes==0){
                    printf("\n      ?HOW MANY HOLES DO YOU WANT[1~50]:");
                    scanf("%d", &modes);
                    }//if dif=0
                    else modes *= 7;
                    ////////////////////////////////
                    printf("  ---------------------------------------\n");
                    printf("           YOU WILL CHALLENGE:\n");
//                    printf("          ◆-----------------◆\n");
                    gameStart(fullBoard, board, modes);
                    printf("          ◆-----------------◆\n");
                    printf("       ARE YOU READY? PRESS 2 TO GAME!\n");
                    printf("  ---------------------------------------\n");
                    flag_hdk = 0;
                    printf("        Press any key to continue...\n");
                    printf("●-----------------*MENU*-----------------●\n");
                    getchar(),getchar();//按任意键以继续
                    break;
```

```
                }//case2 1
                case 2:{
                    if(flag_game!=0)
                    {
                        printf("         YOU HADN'T CHOOSE MODES YET!\n ");
                        printf("        Press any key to continue...\n");
                    printf("●------------------*MENU*-----------------●\n");
                    getchar(),getchar();//按任意键以继续
                        break;
                    }
                    else{
                    game(fullBoard,board,modes);
                    break;}
                }//case2 2
                case 3:{
                    printf("    Please input your filename:");
                    scanf("%s", HANIDOKUFileName);
                    if (ReadCNF(CNF, HANIDOKU) == OK){
                        flag_cnf = 0;
                        printf("\n    File has been read successfully!\n");
                    }//if ok
                    else{
                        flag_cnf = 1;
                        printf("\n           !File Reading ERROR!\n");break;
                    }//else error
                        printf("    ====================================\n");
                        printf("    This problem is being solved...\n    Please
wait for a while....\n");
                        printf("    ====================================");
                    value = (int *)malloc(sizeof(int) * (Bool_cnt + 1));
                    for (int i = 0; i <= Bool_cnt; i++)
                        value[i] = 1;
                    start = clock();
                    res = DPLL(CNF, value, HANIDOKU);
                    finish = clock();
                    if (!res)
                    {
                        printf("\n    The Answer's as follow:\n");
                        printf("   ------------------------------------\n");
                        printf("               No Answer!\n");
                    }

                    else
                    {
```

```
                    printf("\n    The Answer's as follow:\n");
                    printf("   ------------------------------------\n");
                    for (int i = 111; i <= Bool_cnt; i++)
                    {
                        if (value[i])
                        {
                            if((i-2)%6!=0)
                            printf("%6d\t", i);
                            else
                            printf("%6d\n ", i);
                        }
                        else
                        {
                            if((i-2)%6!=0)
                            printf("%6d\t", -i);
                            else printf("%6d\n", -i);
                        }
                    }
                    printf("\n");
                }//else
                TIME = (double)(finish - start) / CLOCKS_PER_SEC;
                printf("   ------------------------------------\n");
                printf("    Execution Time=%lfms\n", TIME * 1000);
                printf("   ------------------------------------\n");
                printf("        Press any key to continue...\n");
                printf("●------------------*MENU*------------------●\n");
                getchar(),getchar();//按任意键以继续
                break;
            }//case2 3
            case 4:{
                if(flag_game!=0)
                {
                    printf("         YOU HADN'T CHOOSE MODES YET!\n ");
                    printf("         Press any key to continue...\n");

printf("●------------------*MENU*------------------●\n");
                    getchar(),getchar();//按任意键以继续
                    break;
                }

                printf("    Please input your filename:");
                scanf("%s",HANIDOKUFileName);
                transhanidokuToCNF(board,modes);
                printf("    %s has been SAVED.\n\n",HANIDOKUFileName);
```

```
                    printf("       Press any key to continue...\n");
                    printf("●------------------*MENU*-----------------●\n");
                    getchar(),getchar();//按任意键以继续
                    break;
                }//case2 4
            }//switch2
        }//while2
    }//case2

    }//switch

}//while

    return 0;
}
void TEMP(int &a,int &b)
{
    int t;
    t=a;
    a=b;
    b=t;
}
int ABS(int a){return a > 0 ? a : -a;}
int read(FILE *fp){
    int flag = 0;
    int x = 0, f = 1;
    char c;
    c = getc(fp);
    while (c < '0' || c > '9'){
        if (c == '-')
            f = -1, c = getc(fp);
        else return inf;}

    while (c >= '0' && c <= '9'){
        x = x * 10 + c - '0';
        c = getc(fp);
        if (c == -1)
            flag = 1;}

    if (!flag && c != ' ' && c != '\n')
        return inf;
    return x * f;
}
status ReadCNF(clause *&cnf, int satORhanidoku){
```

```
FILE *fp;
char ch;
cNode *pn;
clause *pc;
char check[5] = {' ', 'c', 'n', 'f', ' '};
if (satORhanidoku == SAT)
    fp = fopen(fileName, "r");
else
    fp = fopen(HANIDOKUFileName, "r");
if (!fp)
    return ERROR;
while ((ch = getc(fp)) == 'c')
    while ((ch = getc(fp)) != '\n')
        continue;//跳过所有注释行
if (ch != 'p')
    return ERROR;
for (int i = 0; i < 5; i++)
    if ((ch = getc(fp)) != check[i])
        return ERROR;

if ((Bool_cnt = read(fp)) == inf || (Clause_cnt = read(fp)) == inf)
    return ERROR;//p后面都是int变量!!

//构建二维链表
cnf = (clause *)malloc(sizeof(clause));
cnf->next = NULL;
cnf->head = (cNode *)malloc(sizeof(cNode));
cnf->head->next = NULL;
cnf->num = 0;
pc = cnf;
pn = cnf->head;

//^ init part ^
for (int i = 1; i <= Clause_cnt; i++)
{
    int dat;
    if ((dat = read(fp)) == inf)
        return ERROR;
    while (dat)
    {
        pc->num++;
        pn->data = dat;
        pn->next = (cNode *)malloc(sizeof(cNode));
        if ((dat = read(fp)) == inf)
```

```
            return ERROR;
        if (!dat)
            pn->next = NULL;
        pn = pn->next;
    }
    pc->next = (clause *)malloc(sizeof(clause));
    pc->next->num = 0;
    pc->next->head = (cNode *)malloc(sizeof(cNode));
    if (i == Clause_cnt){
        pc->next = NULL;
        break;}
    pc = pc->next;
    pn = pc->head;
}
//^ read part ^
fclose(fp);
return OK;
}
/*销毁二维链表
函数名称：destroyClause(clause *&cnf)
函数参数说明：cnf 表示整个二维链表头。
操作结果：销毁整个二维链表。
设计思想：基本同读取 CNF 文件，但执行的是释放空间。*/
void destroyClause(clause *&cnf)
{
    clause *ppc, *pc2;
    cNode *pn1, *pn2;
    ppc = cnf;
    while (ppc)
    {
        pn1 = ppc->head;
        while (pn1)
            pn2 = pn1->next, free(pn1), pn1 = pn2;
        pc2 = ppc->next, free(ppc), ppc = pc2;
    }
    cnf = NULL;
}
/*删除子句
函数名称：removeClause(clause *&cnf, clause *&cl)
函数参数说明：cnf 表示整个二维链表头，cl 表示删除子句头结点。
操作结果：在二维链表中删除 cl 子句。
设计思想：搜索表头，找到 cl 并释放其与其子节点。*/
void removeClause(clause *&cnf, clause *&cl)
```

```
{
    if (cl == cnf)
        cnf = cnf->next;
    else{
        clause *ppc = cnf;
        while (ppc && ppc->next != cl)
            ppc = ppc->next;
        ppc->next = ppc->next->next;
    }
    cNode *pn1, *pn2;
    for (pn1 = cl->head; pn1;)
        pn2 = pn1->next, free(pn1), pn1 = pn2;
    free(cl);
    cl = NULL;
}
/*删除句元
函数名称：removeNode(clause *cl, cNode *&nd)
函数参数说明：nd 表示要删除的句元，cl 表示所在的子句。
操作结果：在子句 cl 中删除 nd 句元。
设计思想：搜索子句，找到 nd 句元并释放结点。*/
void removeNode(cNode *&head, cNode *&nd)
{
    cNode *ppn = head;
    if (ppn == nd)
        head = head->next;
    else{
        while (ppn && ppn->next != nd)
            ppn = ppn->next;
        ppn->next = ppn->next->next;}
    free(nd);
    nd = NULL;
}
/*增加子句
函数名称：addClause(clause *cl, clause *&cnf)
函数参数说明：cl 表示增添的子句，cnf 表示整个二维链表表头。
操作结果：在二维链表中插入 cl 子句。
设计思想：直接将 cnf 子句接到 cl 后，并使 cl 成为新的二维链表头。*/
status addClause(clause *cl, clause *&cnf)
{
    if (cl){
        cl->next = cnf;
        cnf = cl;
        return OK;}return ERROR;
```

```
}
/*判空子句
函数名称：emptyClause(clause *cnf)
函数参数说明：cl 表示需要判断的子句的表头。
操作结果：判断 cl 是否为空并返回判断结果。
设计思想：直接判断 cl 子句是否有句元结点。*/
bool emptyClause(clause *cnf)
{
    for (clause *p = cnf; p; p = p->next)
        if (!p->head)
            return true;
    return false;
}
/*删除单子句
函数名称：deleltеSingleClause(clause *s, clause *&cnf)
函数参数说明：s 表示单子句，cnf 表示整个二维链表。
操作结果：对二维链表的所有子句中包含单子句句元的，
          若句元真值相同，直接删除该子句，否则删除该子句中
          的单子句句元结点。
设计思想：遍历整个二维链表，符合条件则直接调用相应函数。*/
void deleltеSingleClause(clause *s, clause *&cnf)
{
    clause *tmp;
    int n = s->head->data;
    for (clause *ppc = cnf; ppc; ppc = tmp)
    {
        tmp = ppc->next;
        for (cNode *ppn = ppc->head; ppn; ppn = ppn->next)
        {
            if (ppn->data == n)
            {
                removeClause(cnf, ppc);
                break;
            }
            if (ppn->data == -n)
            {
                removeNode(ppc->head, ppn);
                ppc->num--;
                break;
            }
        }
    }
}
```

```
/*判断是否为单子句
函数名称：isUnitClause(clause *cl)
函数参数说明：cl 表示子句。
操作结果：判断 cl 是否为单子句并返回判断结果。
设计思想：直接找出子句第一个句元结点后是否还有句元结点。*/
bool isUnitClause(clause *cl)
{

    if (cl->head != NULL && cl->head->next == NULL)
        return true;
    return false;
}
/*复制二维链表
函数名称：getForgery(clause *&forgery, clause *cnf)
函数参数说明：forgery 表示复制的二维链表，cnf 表示原链表。
操作结果：得到原二维链表的复制品。
设计思想：基本同读取 CNF 文件，但执行的是复制结点。*/
void getForgery(clause *&forgery, clause *cnf) // A great forgery is just the same.
{
    clause *ppc, *pc;
    cNode *ppn, *pn;
    forgery = (clause *)malloc(sizeof(clause));
    forgery->head = (cNode *)malloc(sizeof(cNode));
    forgery->next = NULL;
    forgery->head->next = NULL;
    forgery->num = 0;
    for (pc = cnf, ppc = forgery; pc != NULL; pc = pc->next, ppc = ppc->next)
    {
        for (pn = pc->head, ppn = ppc->head; pn != NULL; pn = pn->next, ppn = ppn->next)
        {
            ppc->num++;
            ppn->data = pn->data;
            ppn->next = (cNode *)malloc(sizeof(cNode));
            ppn->next->next = NULL;
            if (pn->next == NULL)
                free(ppn->next), ppn->next = NULL;
        }
        ppc->next = (clause *)malloc(sizeof(clause));
        ppc->next->head = (cNode *)malloc(sizeof(cNode));
        ppc->next->next = NULL;
        ppc->next->head->next = NULL;
        ppc->next->num = 0;
        if (pc->next == NULL)
```

```
            free(ppc->next->head), free(ppc->next), ppc->next = NULL;
        }
    }
    /*选元
    函数名称：getNextBool(clause *&cnf)
    函数参数说明：cnf 表示整个二维链表头。
    操作结果：通过某种方式选择一个较为优越的变元进行下一分支判断。*/
    int getMaxBool(clause *cnf)
    {
        int *cnt = (int *)malloc(sizeof(int) * (Bool_cnt * 2 + 1));
        for (int i = 0; i <= Bool_cnt * 2; i++)
            cnt[i] = 0;
        for (clause *pc = cnf; pc; pc = pc->next)
            for (cNode *pn = pc->head; pn; pn = pn->next)
            {
                if (pn->data > 0)
                    cnt[pn->data]++;
                else
                    cnt[Bool_cnt - pn->data]++;
            }
        int maxBool, maxTimes = 0;
        for (int i = 1; i <= Bool_cnt; i++)
            if (cnt[i] > maxTimes)
                maxTimes = cnt[i], maxBool = i;
        if (maxTimes == 0)
        {
            for (int i = Bool_cnt + 1; i <= Bool_cnt * 2; i++)
                if (cnt[i] > maxTimes)
                    maxTimes = cnt[i], maxBool = i - Bool_cnt;
        }
        free(cnt);
        return maxBool;
    }
    int getMaxBool_verhanidoku(clause *cnf)
    {
        int cnt[2222] = {0};
        for (clause *pc = cnf; pc; pc = pc->next)
            for (cNode *pn = pc->head; pn; pn = pn->next)
            {
                if (pn->data > 0)
                    cnt[pn->data]++;
                else
                    cnt[1000 - pn->data]++;
            }
```

```
    int maxBool, maxTimes = 0;
    for (int i = 1; i <= 1000; i++)
        if (cnt[i + 1000] > maxTimes)
            maxTimes = cnt[i], maxBool = -i;
    if (maxTimes == 0)
    {
        for (int i = 1; i <= 1000; i++)
            if (cnt[i] > maxTimes)
                maxTimes = cnt[i], maxBool = i;
    }
    return maxBool;
}
int getNextBool_MOMS(clause *cnf)
{
    int minNode = Bool_cnt;
    int *cnt = (int *)malloc((Bool_cnt + 1) * sizeof(int));
    for (int i = 0; i < Bool_cnt; i++)
        cnt[i] = 0;
    for (clause *pc = cnf; pc; pc = pc->next)
        if (pc->num < minNode)
            minNode = pc->num;
    for (clause *pc = cnf; pc; pc = pc->next)
        if (pc->num == minNode)
            for (cNode *pn = pc->head; pn; pn = pn->next)
                cnt[ABS(pn->data)]++;
    int maxAppear = 0, maxBool;
    for (int i = 1; i <= Bool_cnt; i++)
        if (cnt[i] > maxAppear)
            maxAppear = cnt[i], maxBool = i;
    free(cnt);
    return maxBool;
}
double J(int n){return pow(2.0, (double)(-n));}
int getNextBool_MOM(clause *cnf)
{
    double *weight = (double *)malloc(sizeof(double) * (Bool_cnt + 1));
    for (int i = 0; i <= Bool_cnt; i++)
        weight[i] = 0.0;
    for (clause *pc = cnf; pc; pc = pc->next)
        for (cNode *pn = pc->head; pn; pn = pn->next)
            weight[ABS(pn->data)] += J(pc->num);
    double maxw = 0;
    int maxBool;
    for (int i = 1; i <= Bool_cnt; i++)
```

```
        if (weight[i] > maxw)
            maxw = weight[i], maxBool = i;
    free(weight);
    return maxBool;
}
int getNextBool_MOM_optimized(clause *cnf)
{
    double *weight = (double *)malloc(sizeof(double) * (Bool_cnt * 2 + 1));
    for (int i = 0; i <= Bool_cnt * 2; i++)
        weight[i] = 0.0;
    for (clause *pc = cnf; pc; pc = pc->next)
        for (cNode *pn = pc->head; pn; pn = pn->next)
        {
            if (pn->data > 0)
                weight[pn->data] += J(pc->num);
            else
                weight[Bool_cnt - pn->data] += J(pc->num);
        }
    double maxw = 0.0;
    int maxBool;
    for (int i = 1; i <= Bool_cnt; i++)
        if (weight[i] + weight[i + Bool_cnt] > maxw)
        {
            maxw = weight[i] + weight[i + Bool_cnt], maxBool = i;
        }
    if (weight[maxBool] < weight[maxBool + Bool_cnt])
        maxBool = -maxBool;
    free(weight);
    return maxBool;
}
int getNextBool_Positive_MOM(clause *cnf)
{
    int *mark = (int *)malloc(sizeof(int) * (Bool_cnt + 1));
    double *weight = (double *)malloc(sizeof(double) * (Bool_cnt * 2 + 1));
    for (int i = 0; i <= Bool_cnt; i++)
        mark[i] = 0;
    for (int i = 0; i <= Bool_cnt * 2; i++)
        weight[i] = 0.0;
    for (clause *pc = cnf; pc; pc = pc->next)
        for (cNode *pn = pc->head; pn; pn = pn->next)
        {
            if (pn->data < 0)
                break;
            else if (pn->next == NULL)
```

```
            {
                for (cNode *tmp = pc->head; tmp; tmp = tmp->next)
                    mark[tmp->data] = 1;
                break;
            }
        }
    for (clause *pc = cnf; pc; pc = pc->next)
        for (cNode *pn = pc->head; pn; pn = pn->next)
        {
            if (mark[ABS(pn->data)])
            {
                if (pn->data > 0)
                    weight[pn->data] += J(pc->num);
                else
                    weight[Bool_cnt - pn->data] += J(pc->num);
            }
        }
    free(mark);
    double maxw = 0.0;
    int maxBool = 0;
    for (int i = 1; i <= Bool_cnt; i++)
        if (weight[i] + weight[i + Bool_cnt] > maxw)
        {
            maxw = weight[i] + weight[i + Bool_cnt], maxBool = i;
        }
    if (weight[maxBool] < weight[maxBool + Bool_cnt])
        maxBool = -maxBool;
    free(weight);
    if (maxBool == 0)
        maxBool = getNextBool_MOM_optimized(cnf);
    return maxBool;
}
/*DPLL
函数名称: DPLL(clause *&cnf, int *v, int satORhanidoku)
函数参数说明: cnf 表示整个二维链表头, v 是储存变元真值的数组, satORhanidoku 用于判断当前子系统。
操作结果: 解 CNF 文件并将结果保存到 v 中。*/
bool DPLL(clause *&cnf, int *v, int satORhanidoku)
{
    int flag = 1;
    clause *pc;
    while (flag)
    {
        flag = 0;
```

```
        pc = cnf;
        while (pc && !isUnitClause(pc))
            pc = pc->next;
        if (pc != NULL)
        {
            if (pc->head->data > 0)
                v[pc->head->data] = 1;
            else
                v[-pc->head->data] = 0;
            deleleteSingleClause(pc, cnf);
            if (cnf == NULL)
                return true;
            else if (emptyClause(cnf))
                return false;
            flag = 1;
        }
    }
    //^ delete single clause ^
    int maxBool;
    if (satORhanidoku == HANIDOKU)
        maxBool = getMaxBool_verhanidoku(cnf);
    else
    {
        if (BranchRule == 1)
            maxBool = getMaxBool(cnf);//getMaxBool_verhanidoku(cnf);
        //MO
        else if (BranchRule == 2)
            maxBool = getMaxBool_verhanidoku(cnf);//getNextBool_MOMS(cnf);
        //MPO
        else if (BranchRule == 3)
            maxBool = getNextBool_MOM(cnf);//getNextBool_MOM(cnf);
        //MOM
        else if (BranchRule == 4)
            maxBool                                                                    =
getNextBool_Positive_MOM(cnf);//getNextBool_MOM_optimized(cnf);
        // pos MOM
        //else if (BranchRule == 5)
            //maxBool = getNextBool_Positive_MOM(cnf);
        else maxBool = getMaxBool(cnf);

    }
    //printf("**%d**\n",maxBool);
    //^ pick the best bool. ^
    clause *newSingleClause = (clause *)malloc(sizeof(clause)), *forgery;
```

```
    newSingleClause->head = (cNode *)malloc(sizeof(cNode));
    newSingleClause->next = NULL;
    newSingleClause->head->data = maxBool;
    newSingleClause->head->next = NULL;
    newSingleClause->num = 1;
    getForgery(forgery, cnf);
    addClause(newSingleClause, forgery);
    if (DPLL(forgery, v, satORhanidoku) == true)
        return true; // The first Branch in which the new Single Clause is true;
    destroyClause(forgery);
    newSingleClause = (clause *)malloc(sizeof(clause));
    newSingleClause->head = (cNode *)malloc(sizeof(cNode));
    newSingleClause->next = NULL;
    newSingleClause->head->data = -maxBool;
    newSingleClause->head->next = NULL;
    newSingleClause->num = 1;
    addClause(newSingleClause, cnf);
    bool ans = DPLL(cnf, v, satORhanidoku); // The second Branch in which the new
Single Clause is false.
    destroyClause(cnf);
    return ans;
}
/*保存 res 文件
函数名称: SaveRes(int s, int *v, double t)
函数参数说明: s 是解 cnf 的结果(是否可满足), v 是记录变元最终结果的数组, t 记录运行时间。
操作结果: 根据要求保存解 cnf 最后的结果。*/
status SaveRes(int s, int *v, double t)
{
    FILE *fp;
    for (int i = 0; fileName[i] != '\0'; i++)
        if (fileName[i] == '.' && fileName[i + 4] == '\0')
        {
            fileName[i + 1] = 'r', fileName[i + 2] = 'e', fileName[i + 3] = 's';
            break;
        }
    if (!(fp = fopen(fileName, "w")))
    {
        printf("File Creating ERROR!\n");
        return ERROR;
    }
    fprintf(fp, "s %d\nv ", s);
    if (s)
        for (int i = 1; i <= Bool_cnt; i++)
        {
```

```
            if (v[i])
                fprintf(fp, "%d ", i);
            else
                fprintf(fp, "%d ", -i);
        }
    fprintf(fp, "\nt %lf", t * 1000);
    fclose(fp);
    return OK;
}

//////////////////////////数独游戏//////////////////////////////
void gameStart(int fullBoard[][18], int board[][18], int modes){
    //createFullBoard(fullBoard);
    CREATE();
    dig(fullBoard, board, modes);
   // printHANIDOKU(fullBoard);
    printHANIDOKU(board);
}
void CREATE()
{
    int a1[10][18]=
    {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
     0,0,0,0,0,5,0,4,0,7,0,3,0,6,0,0,0,0,
     0,0,0,0,6,0,3,0,5,0,8,0,7,0,4,0,0,0,
     0,0,0,4,0,8,0,6,0,7,0,9,0,5,0,3,0,0,
     0,0,7,0,6,0,2,0,9,0,8,0,3,0,4,0,5,0,
     0,3,0,5,0,1,0,4,0,2,0,9,0,6,0,8,0,7,
     0,0,7,0,3,0,5,0,1,0,8,0,2,0,4,0,6,0,
     0,0,0,4,0,2,0,3,0,7,0,1,0,6,0,5,0,0,
     0,0,0,0,6,0,4,0,2,0,5,0,7,0,3,0,0,0,
     0,0,0,0,0,5,0,6,0,7,0,3,0,4,0,0,0,0};
    int a2[10][18]=
    {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
     0,0,0,0,0,6,0,4,0,5,0,3,0,7,0,0,0,0,
     0,0,0,0,5,0,9,0,7,0,6,0,4,0,8,0,0,0,
     0,0,0,4,0,7,0,3,0,8,0,9,0,5,0,6,0,0,
     0,0,3,0,8,0,6,0,2,0,1,0,4,0,7,0,5,0,
     0,7,0,6,0,9,0,1,0,5,0,3,0,8,0,2,0,4,
     0,0,5,0,4,0,7,0,8,0,1,0,2,0,3,0,6,0,
     0,0,0,8,0,5,0,3,0,2,0,4,0,6,0,7,0,0,
     0,0,0,0,4,0,2,0,6,0,3,0,7,0,5,0,0,0,
     0,0,0,0,0,6,0,7,0,5,0,4,0,8,0,0,0,0};

     int arr[21];
```

```
    srand(time(0));
    arr[0]=rand()%2;

    if(arr[0]==0)
    for(int i=1;i<=9;i++)
    for(int j=1;j<=17;j++)
    fullBoard[i][j]=a1[i][j];

    if(arr[0]==1)
    for(int i=1;i<=9;i++)
    for(int j=1;j<=17;j++)
    fullBoard[i][j]=a2[i][j];

    for(int i=1;i<=150;i++)
    {
       arr[i]=1+rand()%8;//取 12345678
       switch (arr[i])
       {
       case 1:
       FLIP1();break;
       case 2:
       FLIP2();break;
       case 3:
       FLIP4();break;
       case 4:
       FLIP4();break;
       case 5:
       FLIP5();break;
       case 6:
       FLIP6();break;
       case 7:
       ROTATE();break;
       case 8:
       REFLECT();break;
       }
    }

}
void FLIP1()
{
   for(int i=1;i<=9;i++)
   {
       for(int j=1;j<=8;j++)
       {
```

```
            int t;
            t=fullBoard[i][j];
            fullBoard[i][j]=fullBoard[i][18-j];
            fullBoard[i][18-j]=t;
        }
    }
}
void FLIP2()
{
    for(int i=1;i<=5;i++)
    {//i+4,i -> 1,3+2i
        for(int j=1;j<=(i+4)/2;j++)
        {
            TEMP(fullBoard[i+4+1-j][i+j-1],fullBoard[j][3+2*i+1-j]);
        }
    }
    for(int i=1;i<=4;i++)
    {//9,5+2i -> i+1,i+13
        for(int j=1;j<=(9-i)/2;j++)
        {
            TEMP(fullBoard[9+1-j][j-1+5+2*i],fullBoard[i+1+j-1][i+13+1-j]);
        }
    }
}
void FLIP3()
{
    FLIP1();FLIP2();FLIP1();
}
void FLIP4()
{
    for(int i=1;i<=17;i++)
    for(int j=1;j<=4;j++)
    TEMP(fullBoard[j][i],fullBoard[10-j][i]);
}
void FLIP5()
{
    for(int i=1;i<=5;i++)
    {//i+4,i -> 9,15-2i
        for(int j=1;j<=(6-i)/2;j++)
        {

TEMP(fullBoard[i+4+j-1][i+3*(j-1)],fullBoard[9-j+1][15-2*i-3*(j-1)]);
        }
    }
```

```
    for(int i=1;i<=4;i++)
    {//5-i,i+1 -> 9-i,13+i
        for(int j=1;j<=2;j++)
        {

TEMP(fullBoard[5-i+j-1][i+1+3*(j-1)],fullBoard[9-i-j+1][13+i-3*(j-1)]);
        }
    }
    for(int i=1;i<=4;i++)
    {//6-i,i+2 -> 9-i,11+i
        for(int j=1;j<=2;j++)
        {

TEMP(fullBoard[6-i+j-1][i+2+3*(j-1)],fullBoard[9-i-j+1][11+i-3*(j-1)]);
        }
    }
    for(int i=1;i<=4;i++)
    {//1,2i+5 -> 5-i,17-i
        for(int j=1;j<=(5-i)/2;j++)
        {

TEMP(fullBoard[1+j-1][2*i+5+3*(j-1)],fullBoard[5-i-j+1][17-i-3*(j-1)]);
        }
    }
}
void FLIP6()
{
    FLIP4();
    FLIP5();
    FLIP4();
}
void ROTATE()
{
    int arr[10][18]={0};
    for(int i=1;i<=9;i++)
     for(int j=1;j<=17;j++)
     arr[i][j]=fullBoard[i][j];

    for(int i=1;i<=5;i++)
    {//4+i,i -> i,6-i
        for(int j=1;j<=4+i;j++)
        {
            fullBoard[i][6-i+2*(j-1)]=arr[4+i-j+1][i+j-1];
        }
```

```
    }
    for(int i=1;i<=4;i++)
    {//9,2i+5 -> 5+i,1+i
        for(int j=1;j<=9-i;j++)
        {
            fullBoard[5+i][1+i+2*(j-1)]=arr[9-j+1][2*i+5+j-1];
        }

    }
}
void REFLECT()
{
    for(int i=1;i<=9;i++)
    for(int j=1;j<=17;j++)
    if(1<=fullBoard[i][j]&&fullBoard[i][j]<=9)
    fullBoard[i][j]=10-fullBoard[i][j];
}
void createFullBoard(int a[][18]){
    srand(time(0));
    //第一行要连续!
    int x=rand()%5+5;//x=5,6,7,8,9,取值范围 x-4 ≤ a ≤ x.
    for(int i=5;i<=13;i=i+2){
        a[1][i]=rand() % 9 + 1;
        while(a[1][i]>x||a[1][i]<x-4)
        {
            a[1][i]=rand() % 9 + 1;
        }
        int j=5;
        while(j<i){
            if(a[1][i]==a[1][j]&&a[1][i]!=0)
             {a[1][i] = rand() % 9 + 1, j = 5;
             while(a[1][i]>x||a[1][i]<x-4)
        {
            a[1][i]=rand() % 9 + 1;
        }}
            else j=j+2;
        }
    }//随机生成第一行的数字.
    CreateGame(a, 2, 4);
}
status checkmark(int test,int m1[10],int m2[10],int m3[10],int x1,int x2,int x3)
{
    int max=9,min=1;
    int MAX[4]={0,9,9,9};
```

```
    int MIN[4]={0,1,1,1};
for(int i=1;i<=3;i++)
{
    int x;
    int *p=NULL;
    switch (i)
    {
    case 1:
    x=x1;p=m1;
    break;
    case 2:
    x=x2;p=m2;
    break;
    case 3:
    x=x3;p=m3;
    }
    switch (x)
    {
    case 5:{
        for(int j=1;j<=9;j++)
        {
            if(p[j]==0)continue;
            int t=j+4;
            if(t>9)t=9;
            if(t<MAX[i])MAX[i]=t;

            int tt=j-4;
            if(tt<1)tt=1;
            if(tt>MIN[i])MIN[i]=tt;
        }break;
    }
    case 6:{
        for(int j=1;j<=9;j++)
        {
            if(p[j]==0)continue;
            int t=j+5;
            if(t>9)t=9;
            if(t<MAX[i])MAX[i]=t;

            int tt=j-5;
            if(tt<1)tt=1;
            if(tt>MIN[i])MIN[i]=tt;
        }break;
    }
```

```
    case 7:{
        for(int j=1;j<=9;j++)
        {
            if(p[j]==0)continue;
            int t=j+6;
            if(t>9)t=9;
            if(t<MAX[i])MAX[i]=t;

            int tt=j-6;
            if(tt<1)tt=1;
            if(tt>MIN[i])MIN[i]=tt;
        }break;
    }
    case 8:{
        for(int j=1;j<=9;j++)
        {
            if(p[j]==0)continue;
            int t=j+7;
            if(t>9)t=9;
            if(t<MAX[i])MAX[i]=t;

            int tt=j-7;
            if(tt<1)tt=1;
            if(tt>MIN[i])MIN[i]=tt;
        }break;
    }
    case 9:{
        MAX[i]=9;
        MIN[i]=1;
        break;
    }
    }
}
max=MAX[1]>MAX[2]?MAX[2]:MAX[1];
int e=MAX[3]>max?max:MAX[3];

min=MIN[1]>MIN[2]?MIN[1]:MIN[2];
int ee=MIN[3]>min?MAX[3]:min;

if(test>=ee&&test<=e)return 1;
else return 0;

}
status CreateGame(int a[][18], int row, int col){
```

```
   if(row<=9&& ( col>=1+ABS(5-row) && col<=17-ABS(5-row) )&&row%2==col%2 )//行列取值范围,只有行列合法才可以进去
   {
       int mark[10]={0};
       int flag=1;
       int x1,x2=1,x3=1;
       //x1:直接取这一行全部的有效元素个数.
       x1=9-ABS(5-row);
       //x2:每次变换后,row-i,col+i 与 row-i,col+i
       for(int                 i=1;                 row-i>=1                 &&(col+i>=1+ABS(5-row+i)&&col+i<=17-ABS(5-row+i)) ;i++)x2++;
       for(int                 i=1;                 row+i<=9                 &&(col-i>=1+ABS(5-row-i)&&col-i<=17-ABS(5-row-i)) ;i++)x2++;
       //x3:每次变换后,row-i,col-i 与 row+i,col+i
       for(int                 i=1;                 row-i>=1                 &&(col-i>=1+ABS(5-row+i)&&col-i<=17-ABS(5-row+i)) ;i++)x3++;
       for(int                 i=1;                 row+i<=9                 &&(col+i>=1+ABS(5-row-i)&&col+i<=17-ABS(5-row-i)) ;i++)x3++;

       int mark1[10]={0},mark2[10]={0},mark3[10]={0};
       for(int                                      i=1+ABS(5-row);i<=col;i++){mark[a[row][i]]=1;mark1[a[row][i]]=1;}//对行做标记
       for(int                                                  i=0;i<=row-1;i++){mark[a[row-i][col+i]]=1;mark2[a[row-i][col+i]]=1;}//对↗做标记
       for(int                                                  i=0;i<=row-1;i++){mark[a[row-i][col-i]]=1;mark3[a[row-i][col-i]]=1;}//对↖做标记

       for(int i=1;i<=9&&flag;i++)
       {
          if((!mark[i])&&checkmark(i,mark1,mark2,mark3,x1,x2,x3))//这个数字没被用过，并且符合 check
          {
             flag=0;
             a[row][col]=i;
             if(col==17-ABS(5-row)&&row!=9)
             {
                if(CreateGame(a,row+1,1+ABS(4-row))==OK)
                return OK;
                else flag=1;
             }
             else if(col!=17-ABS(5-row))
             {
                if (CreateGame(a, row, col + 2) == OK)
```

```
                return OK;
                else flag = 1;
            }
        }

    }
    if (flag)
    {
        a[row][col] = 0; // Proper Number Not Found -> delete and travel back.
        return ERROR;
    }
}
//printhanidoku(a);
return OK;
}
/*void printHANIDOKU(int a[][18])
{
    printf("            _   _   _   _   _\n");
    for(int i=1;i<=19;i++)
    {
        int x=ABS(10-i)+4;
        if(i%2==1&&i<=9)
        {
            for(int j=1;j<=x-1;j++)printf(" ");
            printf("_");
        }
        else
        {
            for(int j=1;j<=x;j++)printf(" ");
        }
        if(i%2==1&&i<=9)
        {
            int n1=19-ABS(10-i);
            int n2=n1/2;
            for(int j=1;j<=n2;j++)printf("/ \\\033[4;3;1m \033[0;3;0m");
            printf("\n");
            continue;
        }
        if(i%2==1&&i>9)
        {
            int n1=19-ABS(10-i);
            int n2=n1/2;
            for(int j=1;j<=n2;j++)printf("\\_/ ");
            printf("\n");
```

```
            continue;
        }
        if(i%2==0)
        {
            int n1=19-ABS(10-i);
            int t=i/2;
            int k=1+ABS(5-t);
            for(int j=1;j<=n1;j++)
            {
                if(j%2==1&&j<n1)printf("\033[4;3;1m|\033[0;3;0m");
                if(j==n1){printf("\033[4;3;1m|\033[0;3;0m");
                printf("  ");}
                if(j%2==0)
                {
                    if(MARK[i/2][k]==1)
                    {
                    if(a[i/2][k]==0) printf("\033[1;32;1m . \033[0;3;0m");
                    else printf("\033[1;32;1m %d \033[0;3;0m",a[i/2][k]);
                    }
                    if(MARK[i/2][k]==0)
                    {
                    if(a[i/2][k]==0) printf(" . ");
                    else printf(" %d ",a[i/2][k]);
                    }
                    k=k+2;
                }
            }
            printf("\n");
            continue;
        }
    }
}*/
void printHANIDOKU(int a[][18])
{
    printf("            _   _   _   _   _\n");
    for(int i=1;i<=19;i++)
    {
        int x=ABS(10-i)+4;
        if(i%2==1&&i<=9)
        {
            for(int j=1;j<=x-1;j++)printf(" ");
            printf("_");
        }
        else
```

```
{
    for(int j=1;j<=x;j++)printf(" ");
}
if(i%2==1&&i<=9)
{
    int n1=19-ABS(10-i);
    int n2=n1/2;
    for(int j=1;j<=n2;j++)printf("/ \\\033[4;3;1m \033[0;3;0m");
    printf("\n");
    continue;
}
if(i%2==1&&i>9)
{
    int n1=19-ABS(10-i);
    int n2=n1/2;
    for(int j=1;j<=n2;j++)printf("\\_/ ");
    printf("\n");
    continue;
}
if(i%2==0)
{
    int n1=19-ABS(10-i);
    int t=i/2;
    int k=1+ABS(5-t);
    for(int j=1;j<=n1;j++)
    {
        if(j%2==1&&j<n1)printf("\033[4;3;1m|\033[0;3;0m");
        if(j==n1){printf("\033[0;3;0m|\033[0;3;0m");
        printf("  ");}
        if(j%2==0)
        {
            if(MARK[i/2][k]==1)
            {
            if(a[i/2][k]==0) printf("\033[1;32;1m . \033[0;3;0m");
            else printf("\033[1;32;1m %d \033[0;3;0m",a[i/2][k]);
            }
            if(MARK[i/2][k]==0)
            {
            if(a[i/2][k]==0) printf(" . ");
            else printf(" %d ",a[i/2][k]);
            }
            k=k+2;
        }
    }
```

```
            printf("\n");
            continue;
        }
    }
}
void dig(int a[][18],int b[][18],int holes)
{
    srand(time(0));
    for(int i=1;i<=9;i++)
        for(int j=1;j<=18;j++)
            b[i][j]=a[i][j];//复制数据
    int i=1;
    while(i<=holes)
    {
        int m=rand()%9+1;
        int n=rand()%17+1;
        while(!(n>=1+ABS(5-m)&&n<=17-ABS(5-m)&&(m%2==n%2)))
            n=rand()%17+1;
        if(b[m][n]!=0)
        {
            b[m][n]=0;
            MARK[m][n]=1;
            i++;
            continue;
        }
        if(b[m][n]==0)
            continue;
    }
}
void game(int fullBoard[][18], int board[][18], int holes)
{
    int x=holes;
    int M=-1,N=-1;
    int order=-1;
    double start=clock();
    while(1){
    system("cls");//清屏
    printf("●------------------*MENU*------------------●\n");
    printf("       You are in the \"PLAY HANIDOKU\"!\n\n");
    printf("   Here are some tips to help you complete.\n");
    printf(" ============================================\n");
    printf(" ◆YOU CAN ONLY PUT YOUR NUMBER ON THE GREEN.\n");
    printf(" ◆ENTER (M,N,E) TO PUT (E) IN ROW(M),COL(N).\n");
    printf(" ◆ENTER (11,M,N) TO GET OUR HELP!\n");
```

```
    printf(" ◆ENTER 10 TO CHECK IF YOU ARE RIGHT.\n");
    printf("          ◆ENTER          0          TO          END          THE
GAME.\033[1;31;1m(HAHA,YOU~LOST!!)\033[0;3;0m\n");
    printf(" ========================================\n");
    printf("          \033[1;3;1m★HOLES          LEFT:\033[1;31;1m%d\033[0;3;1m
★",holes);
    switch (x)
    {
    case 7:
        printf("MODE:\033[1;32;1mEASY\033[0;3;0m\n");
        break;
    case 14:
        printf("MODE:\033[1;36;1mNORMAL\033[0;3;0m\n");
        break;
    case 21:
        printf("MODE:\033[1;34;1mHARD\033[0;3;0m\n");
        break;
    case 28:
        printf("MODE:\033[1;33;1mEXPERT\033[0;3;0m\n");
        break;
    case 35:
        printf("MODE:\033[1;35;1mCRAZY\033[0;3;0m\n");
        break;
    case 42:
        printf("MODE:\033[1;31;1mINSANE\033[0;3;0m\n");
        break;
    case 49:
        printf("MODE:\033[0;31;1mHELL\033[0;3;0m\n");
        break;
    default:
        printf("MODE:\033[1;30;1mCUSTOM\033[0;3;0m\n");
        break;
    }
    printHANIDOKU(board);
    printf("          ◆-----------------◆\n");
        if(holes==0)
    {
        printf(" ◇PAY ATTENTION: YOU HAD COMPLETED.\n");
        printf(" ◇YOU CAN ENTER \033[1;32;1m12\033[0;3;0m TO CHECK YOUR ANSWER.\n");
    }
        printf(" ◇YOU WANT:");
    scanf("%d",&order);
        if(order>=1&&order<=9)
        {
```

```
    int E=0;
    M=order;
    scanf("%d %d",&N,&E);
    if (N>9-ABS(5-M)||N<=0||MARK[M][ABS(5-M)+2*N-1]==0||E<=0||E>9)
    {
        printf(" PLEASE BE CAREFUL!!!!!\n");
        printf(" ============================================\n");
        printf("       Press any key to continue...\n");
               printf("●------------------*MENU*-----------------●\n");
               getchar(),getchar();//按任意键以继续
        continue;
    }
    if (MARK[M][ABS(5-M)+2*N-1]==1&&board[M][ABS(5-M)+2*N-1]==0)
    {holes--;}
    board[M][ABS(5-M)+2*N-1]=E;
    printf(" \033[1;32;1mPUT SUCCESSFULLY!!\033[0;3;0m\n");
    printf(" ============================================\n");
    printf("       Press any key to continue...\n");
              printf("●------------------*MENU*-----------------●\n");
              getchar(),getchar();//按任意键以继续
    continue;
}
switch (order)
{
    case 0:{
        for(int i=1;i<=9;i++)
        {
            for(int j=1;j<=17;j++)
            {
                if(MARK[i][j]==1)
                board[i][j]=0;
            }
        }
        system("cls");GAMEOVER();getchar();getchar();
        return;
    }//case 0
    case 114514:{
        holes=0;
        for(int i=1;i<=9;i++)
        for(int j=1;j<=17;j++)
        board[i][j]=fullBoard[i][j];
        printf("\033[1;31;1m WHAT♂ARE♂YOU♂DOING??\033[0;3;0m\n");
        printf(" ============================================\n");
        printf("       Press any key to continue...\n");
```

```
                printf("●------------------*MENU*-----------------●\n");
                getchar(),getchar();//按任意键以继续
            break;
        }//case 114514
        case 12:{
            if(holes!=0)break;
            int judge=1;
            for(int i=1;i<=9;i++)
            for(int j=1;j<=17;j++)
            if(board[i][j]!=fullBoard[i][j])judge=0;
            if(judge==0)
            {
                printf("\033[1;31;1m WRONG ANSWER~~HAHA!\033[0;3;0m\n");
                printf(" ============================================\n");
                printf("        Press any key to continue...\n");
                    printf("●------------------*MENU*-----------------●\n");
                    getchar(),getchar();//按任意键以继续
                break;
            }
            else
            {
               printf("\033[1;32;1m           RIGHT           ANSWER!!!NO!!YOU
WIN!!!\033[0;3;0m\n");
               double finish=clock();
               printf("                                                    YOU
CONSUMED:\033[1;32;1m %.3fs\033[0;3;0m.\n",(finish-start)/1000);
                printf(" ============================================\n");
                printf("        Press any key to continue...\n");
                    printf("●------------------*MENU*-----------------●\n");
                    getchar(),getchar();//按任意键以继续
                system("cls");//清屏
                WIN();
                for(int i=1;i<=9;i++)
                for(int j=1;j<=17;j++)
                    if(MARK[i][j]==1)
                    board[i][j]=0;
                getchar(),getchar();//按任意键以继续
                return;
                break;
            }
        }//case 12
        case 11:{
            scanf("%d %d",&M,&N);
            if(M<=0||N<=0||N>9-ABS(5-M)||M>9)
```

```
            {
                printf(" PLEASE BE CAREFUL!!!!!\n");
                printf(" ==========================================\n");
                printf("       Press any key to continue...\n");
                    printf("●------------------*MENU*-----------------●\n");
                    getchar(),getchar();//按任意键以继续
                break;
            }
            int x=fullBoard[M][ABS(5-M)+2*N-1];
            printf("   THE   NUMBER   IN   \033[1;32;1m(%d,%d)\033[0;3;0m   IS
\033[1;32;1m%d.\033[0;3;0m\n",M,N,x);
            printf(" ==========================================\n");
            printf("       Press any key to continue...\n");
                   printf("●------------------*MENU*-----------------●\n");
                   getchar(),getchar();//按任意键以继续
            break;
          }//case 11
          case 10:{
            printf(" WHICH ANSWER YOU PUT IN (M,N) YOU WANT CHECK?\n");
            printf(" PLEASE ENTER (M,N):");
            scanf("%d %d",&M,&N);
            if(M<=0||N<=0||N>9-ABS(5-M)||M>9)
            {
                printf(" YOU CAN'T PUT IT IN WALLS!!\n");
                printf(" ==========================================\n");
                printf("       Press any key to continue...\n");
                    printf("●------------------*MENU*-----------------●\n");
                    getchar(),getchar();//按任意键以继续
                break;
            }
            else{
                int x=board[M][ABS(5-M)+2*N-1];
                if(x==fullBoard[M][ABS(5-M)+2*N-1])
                {
                    printf("\033[1;32;1m RIGHT ANSWER!!!\033[0;3;0m\n");
                    printf(" ==========================================\n");
                    printf("       Press any key to continue...\n");
                    printf("●------------------*MENU*-----------------●\n");
                    getchar(),getchar();//按任意键以继续
                    break;
                }
                else
                {
                    printf("\033[1;31;1m WRONG ANSWER~~HAHA!\033[0;3;0m\n");
```

```
                    printf(" ========================================\n");
                    printf("       Press any key to continue...\n");
                    printf("●------------------*MENU*-----------------●\n");
                    getchar(),getchar();//按任意键以继续
                    break;
                }
            }
         }//case 10
      }
   }

   printf("       Press any key to continue...\n");
                    printf("●------------------*MENU*-----------------●\n");
                    getchar(),getchar();//按任意键以继续
}
void WIN()
{
printf("\033[1;32;1m\n\n\n\n\n\n\n\n");
printf("
●=============================================================================
==================================●\n\n");
printf("    01010         01010   0101010101     01010    01010            10101
10101  10101010  10101     10101   10101\n");
printf("     01010       01010   010101010101    01010    01010            10101
10101   010101   101010    10101   10101\n");
printf("      01010    01010    01010   01010    01010    01010            10101
10101   010101   1010101    10101   10101\n");
printf("       01010 01010     01010   01010    01010    01010            10101
10101   010101   10101010   10101   10101\n");
printf("       010101010     01010  01010   01010   01010          10101  10101
10101   010101   101010101  10101   10101\n");
printf("        0101010      01010  01010   01010   01010          10101 1010101
10101   010101   1010101010 10101   10101\n");
printf("                 01010             01010    01010      01010      01010
10101101010101010101   010101   10101 1010110101   10101\n");
printf("          01010        01010  01010   01010   01010           101010101
101010101   010101   10101  101010101   10101\n");
printf("          01010        01010  01010   01010   01010           10101010
10101011   010101   10101   10101011  \n");
printf("          01010        01010  01010   0101010101010            1010101
1010101   010101   10101    1010101   10101\n");
printf("          01010        010101010101    0101010101010            101010
101011   010101   10101    101011   10101\n");
printf("           01010         0101010101      01010101010             10101
```

```
10101  01010101  10101     10101   10101\n");
printf("\n
●=================================================================================
==================================●\n");
printf("\033[0;3;0m");
}
void GHOST()
{
   // printf("\033[1;31;5m");
    printf("                 01010101\n");
    printf("              01010101010101\n");
    printf("            0101010101010101\n");
    printf("           0101010101010101\n");
    printf("           0101010101010101\n");
    printf("           010    0101    010\n");
    printf("           01    010101    01\n");
    printf("           010   01010101   101\n");
    printf("            01010101  01010101\n");
    printf("             0101010101010101\n");
    printf("             0 01 010101 01 0\n");
    printf("              01 01 01 01 01\n");
    printf("               01       01\n");
    printf("                1010  1010\n");
    printf("                 01010101 \n\n");
   // printf("\033[0;3;5m");
}
//
status transhanidokuToCNF(int a[][18],int holes)
{//自 3 开始到 5823 ---> 最多有 5821 个子句,实际上有 5821-holes 个
    FILE *fp;
    if (!(fp = fopen(HANIDOKUFileName, "w")))
    {
        fprintf(fp, "File creating error!\n");
        return ERROR;
    }
    fprintf(fp, "c A hanidoku cnf file.  Holes count:%d\n", holes);
    char s[5];//开辟四位数长度空间
    int num=5821-9*holes;
    for(int i=0;i<4;i++)
    {
        int x=num%10;
        num=num/10;
        s[3-i]=x+48;
    }s[4]='\0';
```

```
    fprintf(fp, "p cnf 959 ");//全部是要用字符写.
for(int i=0;i<4;i++)
fprintf(fp,"%c",s[i]);
fprintf(fp,"\n");
////////////////////////////////////////////////
for(int i=1;i<=9;i++)//已结束
{//读取提示数字
    for(int j=1+ABS(5-i);j>=1+ABS(5-i)&&j<=17-ABS(5-i);j=j+2)//行列数字符合取值范围
    {
        int row=i;//行是一样的.
        int col=(j+1-ABS(5-row))/2;
        if(board[i][j]!=0)
        {
            fprintf(fp,"%c%c%c 0\n",row+48,col+48,board[i][j]+48);
            for(int k=1;k<=9;k++)
            {
                if(k!=board[i][j])fprintf(fp,"-%c%c%c 0\n",row+48,col+48,k+48);
            }

        }

    }
}
/**/
for(int i=1;i<=9;i++)
{
    for(int j=1+ABS(5-i);j<=17-ABS(5-i);j+=2)
    {
        for(int k=1;k<=9;k++)
        {
            fprintf(fp,"%c%c%c ",i+48,translatecol(i,j)+48,k+48);
        }fprintf(fp,"0\n");
    }
}
//读取行连续性约束
for(int i=1;i<=9;i++)
{//代表数字1~9
    for(int j=1;j<=9;j++)
    {//代表行1~9
        int col=9-ABS(5-j);//一行有几个数字
        for(int k=1;k<=col-1;k++)//连续性约束的第一个数字
```

```
            {
                for(int l=k+1;l<=col;l++)
                {
                    fprintf(fp,"-%c%c%c                                    -%c%c%c
0\n",j+48,k+48,i+48,j+48,l+48,i+48);
                }
            }
        }
    }
    //读取↗行连续性约束
    for(int i=1;i<=9;i++)
    {//代表数字1~9
        for(int j=5;j<=13;j=j+2)//列的下标
        {
            int x=translatecol(1,j)+4;//对应12345+4,每列有几个数字
            for(int k=1;k<=x-1;k++)//每列大循环次数是x-1
            {
                for(int l=1;l<=x-k;l++)//大循环的小循环次数是x-k
                {
                    //k,j+1-k -> k+l,j+1-k-l
                    int r1=k,r2=k+l;
                    int c1=translatecol(r1,j+1-k);
                    int c2=translatecol(r2,j+1-k-l);
                    fprintf(fp,"-%c%c%c                                    -%c%c%c
0\n",r1+48,c1+48,i+48,r2+48,c2+48,i+48);
                }
            }
        }
        for(int j=7;j<=13;j=j+2)//另一部分的列下标
        {
            int x=10-translatecol(9,j);//对应10-2345,每列有几个数字
            for(int k=1;k<=x-1;k++)//每列大循环次数是x-1
            {
                for(int l=1;l<=x-k;l++)//大循环的小循环次数是x-k
                {
                    //9+1-k,j+k-1 -> 9+1-k-l,j+k-1+l
                    int r1=10-k,r2=10-k-l;
                    int c1=translatecol(r1,j+k-1);
                    int c2=translatecol(r2,j+k-1+l);
                    fprintf(fp,"-%c%c%c                                    -%c%c%c
0\n",r1+48,c1+48,i+48,r2+48,c2+48,i+48);
                }
            }
```

```
        }
    }
    //读取↖行连续性约束
    for(int i=1;i<=9;i++)
    {//代表数字1~9
        for(int j=5;j<=13;j=j+2)
        {
            int x=translatecol(9,j)+4;//对应12345+4,每列有几个数字
            for(int k=1;k<=x-1;k++)
            {
                for(int l=1;l<=x-k;l++)
                {
                    //9+1-k,j+1-k -> 9+1-k-l,j+1-k-l
                    int r1=10-k,r2=10-k-l;
                    int c1=translatecol(r1,j+1-k);
                    int c2=translatecol(r2,j+1-k-l);
                    fprintf(fp,"-%c%c%c                                    -%c%c%c
0\n",r1+48,c1+48,i+48,r2+48,c2+48,i+48);
                }
            }
        }
        for(int j=7;j<=13;j=j+2)
        {
            int x=10-translatecol(1,j);//对应10-2345,每列有几个数字
            for(int k=1;k<=x-1;k++)
            {
                for(int l=1;l<=x-k;l++)
                {
                    //k,j+k-1 -> k+l,j+k-1+l
                    int r1=k,r2=k+l;
                    int c1=translatecol(r1,j+k-1);
                    int c2=translatecol(r2,j+k-1+l);
                    fprintf(fp,"-%c%c%c                                    -%c%c%c
0\n",r1+48,c1+48,i+48,r2+48,c2+48,i+48);
                }
            }

        }
    }
    /**/
    for(int i=1;i<=9;i++)
    {//行选填约束
        if(i==5)continue;//这一行有9个,不需要选填约束.
        switch (i)
```

```
{
    case 1:{
        for(int j=1;j<=10;j++)
        {
            for(int k=1;k<=5;k++)
            {
                if(j==1)fprintf(fp,"1%c1 1%c6 ",k+48,k+48);
                if(j==2)fprintf(fp,"1%c2 1%c6 ",k+48,k+48);
                if(j==3)fprintf(fp,"1%c3 1%c6 ",k+48,k+48);
                if(j==4)fprintf(fp,"1%c4 1%c6 ",k+48,k+48);
                if(j==5)fprintf(fp,"1%c2 1%c7 ",k+48,k+48);
                if(j==6)fprintf(fp,"1%c3 1%c7 ",k+48,k+48);
                if(j==7)fprintf(fp,"1%c4 1%c7 ",k+48,k+48);
                if(j==8)fprintf(fp,"1%c3 1%c8 ",k+48,k+48);
                if(j==9)fprintf(fp,"1%c4 1%c8 ",k+48,k+48);
                if(j==10)fprintf(fp,"1%c4 1%c9 ",k+48,k+48);
            }fprintf(fp,"0\n");
        }break;
    }
    case 2:{
        for(int j=1;j<=6;j++)
        {
            for(int k=1;k<=6;k++)
            {
                if(j==1)fprintf(fp,"2%c1 2%c7 ",k+48,k+48);
                if(j==2)fprintf(fp,"2%c2 2%c7 ",k+48,k+48);
                if(j==3)fprintf(fp,"2%c3 2%c7 ",k+48,k+48);
                if(j==4)fprintf(fp,"2%c2 2%c8 ",k+48,k+48);
                if(j==5)fprintf(fp,"2%c3 2%c8 ",k+48,k+48);
                if(j==6)fprintf(fp,"2%c3 2%c9 ",k+48,k+48);
            }fprintf(fp,"0\n");
        }break;
    }
    case 3:{
        for(int j=1;j<=3;j++)
        {
            for(int k=1;k<=7;k++)
            {
                if(j==1)fprintf(fp,"3%c1 3%c8 ",k+48,k+48);
                if(j==2)fprintf(fp,"3%c2 3%c8 ",k+48,k+48);
                if(j==3)fprintf(fp,"3%c2 3%c9 ",k+48,k+48);
            }fprintf(fp,"0\n");
        }break;
    }
```

```
case 4:{
    for(int j=1;j<=8;j++)
    fprintf(fp,"4%c1 4%c9 ",j+48,j+48);
    fprintf(fp,"0\n");break;
}
case 6:{
    for(int j=1;j<=8;j++)
    fprintf(fp,"6%c1 6%c9 ",j+48,j+48);
    fprintf(fp,"0\n");break;
}
case 7:{
    for(int j=1;j<=3;j++)
    {
        for(int k=1;k<=7;k++)
        {
            if(j==1)fprintf(fp,"7%c1 7%c8 ",k+48,k+48);
            if(j==2)fprintf(fp,"7%c2 7%c8 ",k+48,k+48);
            if(j==3)fprintf(fp,"7%c2 7%c9 ",k+48,k+48);
        }fprintf(fp,"0\n");
    }break;
}
case 8:{
    for(int j=1;j<=6;j++)
    {
        for(int k=1;k<=6;k++)
        {
            if(j==1)fprintf(fp,"8%c1 8%c7 ",k+48,k+48);
            if(j==2)fprintf(fp,"8%c2 8%c7 ",k+48,k+48);
            if(j==3)fprintf(fp,"8%c3 8%c7 ",k+48,k+48);
            if(j==4)fprintf(fp,"8%c2 8%c8 ",k+48,k+48);
            if(j==5)fprintf(fp,"8%c3 8%c8 ",k+48,k+48);
            if(j==6)fprintf(fp,"8%c3 8%c9 ",k+48,k+48);
        }fprintf(fp,"0\n");
    }break;
}
case 9:{
    for(int j=1;j<=10;j++)
    {
        for(int k=1;k<=5;k++)
        {
            if(j==1)fprintf(fp,"9%c1 9%c6 ",k+48,k+48);
            if(j==2)fprintf(fp,"9%c2 9%c6 ",k+48,k+48);
            if(j==3)fprintf(fp,"9%c3 9%c6 ",k+48,k+48);
            if(j==4)fprintf(fp,"9%c4 9%c6 ",k+48,k+48);
```

```
                    if(j==5)fprintf(fp,"9%c2 9%c7 ",k+48,k+48);
                    if(j==6)fprintf(fp,"9%c3 9%c7 ",k+48,k+48);
                    if(j==7)fprintf(fp,"9%c4 9%c7 ",k+48,k+48);
                    if(j==8)fprintf(fp,"9%c3 9%c8 ",k+48,k+48);
                    if(j==9)fprintf(fp,"9%c4 9%c8 ",k+48,k+48);
                    if(j==10)fprintf(fp,"9%c4 9%c9 ",k+48,k+48);
                }fprintf(fp,"0\n");
            }break;
        }
    }

}

//↗选填约束
for(int i=1;i<=9;i++)
{
    switch (i)
    {
    case 1:{
        for(int j=1;j<=10;j++)
        {
            for(int k=1;k<=5;k++)
            {
                int row=k,col=translatecol(k,2*i+3-k+1);
                if(j==1)fprintf(fp,"%c%c1                                %c%c6
",row+48,col+48,row+48,col+48);
                if(j==2)fprintf(fp,"%c%c2                                %c%c6
",row+48,col+48,row+48,col+48);
                if(j==3)fprintf(fp,"%c%c3                                %c%c6
",row+48,col+48,row+48,col+48);
                if(j==4)fprintf(fp,"%c%c4                                %c%c6
",row+48,col+48,row+48,col+48);
                if(j==5)fprintf(fp,"%c%c2                                %c%c7
",row+48,col+48,row+48,col+48);
                if(j==6)fprintf(fp,"%c%c3                                %c%c7
",row+48,col+48,row+48,col+48);
                if(j==7)fprintf(fp,"%c%c4                                %c%c7
",row+48,col+48,row+48,col+48);
                if(j==8)fprintf(fp,"%c%c3                                %c%c8
",row+48,col+48,row+48,col+48);
                if(j==9)fprintf(fp,"%c%c4                                %c%c8
",row+48,col+48,row+48,col+48);
                if(j==10)fprintf(fp,"%c%c4                               %c%c9
",row+48,col+48,row+48,col+48);
```

```
            }fprintf(fp,"0\n");
        }break;
    }
    case 2:{
        for(int j=1;j<=6;j++)
        {
            for(int k=1;k<=6;k++)
            {
                int row=k,col=translatecol(k,2*i+3-k+1);
                if(j==1)fprintf(fp,"%c%c1                                %c%c7
",row+48,col+48,row+48,col+48);
                if(j==2)fprintf(fp,"%c%c2                                %c%c7
",row+48,col+48,row+48,col+48);
                if(j==3)fprintf(fp,"%c%c3                                %c%c7
",row+48,col+48,row+48,col+48);
                if(j==4)fprintf(fp,"%c%c2                                %c%c8
",row+48,col+48,row+48,col+48);
                if(j==5)fprintf(fp,"%c%c3                                %c%c8
",row+48,col+48,row+48,col+48);
                if(j==6)fprintf(fp,"%c%c3                                %c%c9
",row+48,col+48,row+48,col+48);
            }fprintf(fp,"0\n");
        }break;
    }
    case 3:{
        for(int j=1;j<=3;j++)
        {
            for(int k=1;k<=7;k++)
            {
                int row=k,col=translatecol(k,2*i+3-k+1);
                if(j==1)fprintf(fp,"%c%c1                                %c%c8
",row+48,col+48,row+48,col+48);
                if(j==2)fprintf(fp,"%c%c2                                %c%c8
",row+48,col+48,row+48,col+48);
                if(j==3)fprintf(fp,"%c%c2                                %c%c9
",row+48,col+48,row+48,col+48);
            }fprintf(fp,"0\n");
        }break;
    }
    case 4:{
        for(int k=1;k<=8;k++)
        {
            int row=k,col=translatecol(k,2*i+3-k+1);
            fprintf(fp,"%c%c1 %c%c9 ",row+48,col+48,row+48,col+48);
```

```
            }fprintf(fp,"0\n");break;
        }
        case 6:{
            for(int k=1;k<=8;k++)
            {
                int row=k+i-5,col=translatecol(row,14+i-6-k+1);
                fprintf(fp,"%c%c1 %c%c9 ",row+48,col+48,row+48,col+48);
            }fprintf(fp,"0\n");break;
        }
        case 7:{
            for(int j=1;j<=3;j++)
            {
                for(int k=1;k<=7;k++)
                {
                    int row=k+i-5,col=translatecol(row,14+i-6-k+1);
                    if(j==1)fprintf(fp,"%c%c1                                    %c%c8
",row+48,col+48,row+48,col+48);
                    if(j==2)fprintf(fp,"%c%c2                                    %c%c8
",row+48,col+48,row+48,col+48);
                    if(j==3)fprintf(fp,"%c%c2                                    %c%c9
",row+48,col+48,row+48,col+48);
                }fprintf(fp,"0\n");
            }break;
        }
        case 8:{
            for(int j=1;j<=6;j++)
            {
                for(int k=1;k<=6;k++)
                {
                    int row=k+i-5,col=translatecol(row,14+i-6-k+1);
                    if(j==1)fprintf(fp,"%c%c1                                    %c%c7
",row+48,col+48,row+48,col+48);
                    if(j==2)fprintf(fp,"%c%c2                                    %c%c7
",row+48,col+48,row+48,col+48);
                    if(j==3)fprintf(fp,"%c%c3                                    %c%c7
",row+48,col+48,row+48,col+48);
                    if(j==4)fprintf(fp,"%c%c2                                    %c%c8
",row+48,col+48,row+48,col+48);
                    if(j==5)fprintf(fp,"%c%c3                                    %c%c8
",row+48,col+48,row+48,col+48);
                    if(j==6)fprintf(fp,"%c%c3                                    %c%c9
",row+48,col+48,row+48,col+48);
                }fprintf(fp,"0\n");
            }break;
```

```
        }
        case 9:{
            for(int j=1;j<=10;j++)
            {
                for(int k=1;k<=5;k++)
                {
                    int row=k+i-5,col=translatecol(row,14+i-6-k+1);
                    if(j==1)fprintf(fp,"%c%c1                                %c%c6
",row+48,col+48,row+48,col+48);
                    if(j==2)fprintf(fp,"%c%c2                                %c%c6
",row+48,col+48,row+48,col+48);
                    if(j==3)fprintf(fp,"%c%c3                                %c%c6
",row+48,col+48,row+48,col+48);
                    if(j==4)fprintf(fp,"%c%c4                                %c%c6
",row+48,col+48,row+48,col+48);
                    if(j==5)fprintf(fp,"%c%c2                                %c%c7
",row+48,col+48,row+48,col+48);
                    if(j==6)fprintf(fp,"%c%c3                                %c%c7
",row+48,col+48,row+48,col+48);
                    if(j==7)fprintf(fp,"%c%c4                                %c%c7
",row+48,col+48,row+48,col+48);
                    if(j==8)fprintf(fp,"%c%c3                                %c%c8
",row+48,col+48,row+48,col+48);
                    if(j==9)fprintf(fp,"%c%c4                                %c%c8
",row+48,col+48,row+48,col+48);
                    if(j==10)fprintf(fp,"%c%c4                               %c%c9
",row+48,col+48,row+48,col+48);
                }fprintf(fp,"0\n");
            }break;
        }
        }
    }
    //↖选填约束
    for(int i=1;i<=9;i++)
    {
        switch (i)
        {
        case 1:{
            for(int j=1;j<=10;j++)
            {
                for(int k=1;k<=5;k++)
                {
                    int row=5+k-i,col=translatecol(row,k+i-1);
                    if(j==1)fprintf(fp,"%c%c1                                %c%c6
```

```
",row+48,col+48,row+48,col+48);
                if(j==2)fprintf(fp,"%c%c2                                %c%c6
",row+48,col+48,row+48,col+48);
                if(j==3)fprintf(fp,"%c%c3                                %c%c6
",row+48,col+48,row+48,col+48);
                if(j==4)fprintf(fp,"%c%c4                                %c%c6
",row+48,col+48,row+48,col+48);
                if(j==5)fprintf(fp,"%c%c2                                %c%c7
",row+48,col+48,row+48,col+48);
                if(j==6)fprintf(fp,"%c%c3                                %c%c7
",row+48,col+48,row+48,col+48);
                if(j==7)fprintf(fp,"%c%c4                                %c%c7
",row+48,col+48,row+48,col+48);
                if(j==8)fprintf(fp,"%c%c3                                %c%c8
",row+48,col+48,row+48,col+48);
                if(j==9)fprintf(fp,"%c%c4                                %c%c8
",row+48,col+48,row+48,col+48);
                if(j==10)fprintf(fp,"%c%c4                               %c%c9
",row+48,col+48,row+48,col+48);
            }fprintf(fp,"0\n");
        }break;
    }
    case 2:{
        for(int j=1;j<=6;j++)
        {
            for(int k=1;k<=6;k++)
            {
                int row=5+k-i,col=translatecol(row,k+i-1);
                if(j==1)fprintf(fp,"%c%c1                                %c%c7
",row+48,col+48,row+48,col+48);
                if(j==2)fprintf(fp,"%c%c2                                %c%c7
",row+48,col+48,row+48,col+48);
                if(j==3)fprintf(fp,"%c%c3                                %c%c7
",row+48,col+48,row+48,col+48);
                if(j==4)fprintf(fp,"%c%c2                                %c%c8
",row+48,col+48,row+48,col+48);
                if(j==5)fprintf(fp,"%c%c3                                %c%c8
",row+48,col+48,row+48,col+48);
                if(j==6)fprintf(fp,"%c%c3                                %c%c9
",row+48,col+48,row+48,col+48);
            }fprintf(fp,"0\n");
        }break;
    }
    case 3:{
```

```
            for(int j=1;j<=3;j++)
            {
                for(int k=1;k<=7;k++)
                {
                    int row=5+k-i,col=translatecol(row,k+i-1);
                    if(j==1)fprintf(fp,"%c%c1                                %c%c8
",row+48,col+48,row+48,col+48);
                    if(j==2)fprintf(fp,"%c%c2                                %c%c8
",row+48,col+48,row+48,col+48);
                    if(j==3)fprintf(fp,"%c%c2                                %c%c9
",row+48,col+48,row+48,col+48);
                }fprintf(fp,"0\n");
            }break;
        }
        case 4:{
            for(int k=1;k<=8;k++)
            {
                int row=5+k-i,col=translatecol(row,k+i-1);
                fprintf(fp,"%c%c1 %c%c9 ",row+48,col+48,row+48,col+48);
            }fprintf(fp,"0\n");break;
        }
        case 6:{
            for(int k=1;k<=8;k++)
            {
                int row=k,col=translatecol(k,2*i-5+k-1);
                fprintf(fp,"%c%c1 %c%c9 ",row+48,col+48,row+48,col+48);
            }fprintf(fp,"0\n");break;
        }
        case 7:{
            for(int j=1;j<=3;j++)
            {
                for(int k=1;k<=7;k++)
                {
                    int row=k,col=translatecol(k,2*i-5+k-1);
                    if(j==1)fprintf(fp,"%c%c1                                %c%c8
",row+48,col+48,row+48,col+48);
                    if(j==2)fprintf(fp,"%c%c2                                %c%c8
",row+48,col+48,row+48,col+48);
                    if(j==3)fprintf(fp,"%c%c2                                %c%c9
",row+48,col+48,row+48,col+48);
                }fprintf(fp,"0\n");
            }break;
        }
        case 8:{
```

```
            for(int j=1;j<=6;j++)
            {
                for(int k=1;k<=6;k++)
                {
                    int row=k,col=translatecol(k,2*i-5+k-1);
                    if(j==1)fprintf(fp,"%c%c1                                %c%c7
",row+48,col+48,row+48,col+48);
                    if(j==2)fprintf(fp,"%c%c2                                %c%c7
",row+48,col+48,row+48,col+48);
                    if(j==3)fprintf(fp,"%c%c3                                %c%c7
",row+48,col+48,row+48,col+48);
                    if(j==4)fprintf(fp,"%c%c2                                %c%c8
",row+48,col+48,row+48,col+48);
                    if(j==5)fprintf(fp,"%c%c3                                %c%c8
",row+48,col+48,row+48,col+48);
                    if(j==6)fprintf(fp,"%c%c3                                %c%c9
",row+48,col+48,row+48,col+48);
                }fprintf(fp,"0\n");
            }break;
        }
        case 9:{
            for(int j=1;j<=10;j++)
            {
                for(int k=1;k<=5;k++)
                {
                    int row=k,col=translatecol(k,2*i-5+k-1);
                    if(j==1)fprintf(fp,"%c%c1                                %c%c6
",row+48,col+48,row+48,col+48);
                    if(j==2)fprintf(fp,"%c%c2                                %c%c6
",row+48,col+48,row+48,col+48);
                    if(j==3)fprintf(fp,"%c%c3                                %c%c6
",row+48,col+48,row+48,col+48);
                    if(j==4)fprintf(fp,"%c%c4                                %c%c6
",row+48,col+48,row+48,col+48);
                    if(j==5)fprintf(fp,"%c%c2                                %c%c7
",row+48,col+48,row+48,col+48);
                    if(j==6)fprintf(fp,"%c%c3                                %c%c7
",row+48,col+48,row+48,col+48);
                    if(j==7)fprintf(fp,"%c%c4                                %c%c7
",row+48,col+48,row+48,col+48);
                    if(j==8)fprintf(fp,"%c%c3                                %c%c8
",row+48,col+48,row+48,col+48);
                    if(j==9)fprintf(fp,"%c%c4                                %c%c8
",row+48,col+48,row+48,col+48);
```

```
                if(j==10)fprintf(fp,"%c%c4                                %c%c9
",row+48,col+48,row+48,col+48);
            }fprintf(fp,"0\n");
        }break;
    }
    }
}
//1,9 行-5  2,8 行-456  3,7 行-34567  4,6 行-2345678 5,5 行-123456789------含有性
约束
//数目:9-2*ABS(5-row),取值范围:1+ABS(5-row)~9-ABS(5-row)
for(int i=1;i<=9;i++)
{
    for(int k=1+ABS(5-i);k<=9-ABS(5-i);k++){
    for(int j=1+ABS(5-i);j<=17-ABS(5-i);j+=2)
    {
        fprintf(fp,"%c%c%c ",i+48,translatecol(i,j)+48,k+48);
    }fprintf(fp,"0\n");}//fprintf(fp,"0\n");
}
for(int i=5;i<=13;i+=2)
{
    int t=translatecol(1,i);
    for(int k=1+ABS(5-t);k<=9-ABS(5-t);k++)
    {
        for(int j=1;j<=t+4;j++)
        {
            fprintf(fp,"%c%c%c ",j+48,translatecol(j,i+1-j)+48,k+48);
        }fprintf(fp,"0\n");
    }
}
for(int i=14;i<=17;i++)
{
    int t=i-8;
    int row=i-12;
    for(int k=1+ABS(5-t);k<=9-ABS(5-t);k++)
    {
        for(int j=1;j<=14-t;j++)
        {
            fprintf(fp,"%c%c%c
",row+48+j-1,translatecol(row+j-1,i-j+1)+48,k+48);
        }fprintf(fp,"0\n");
    }
}
for(int i=1;i<=5;i++)
{
```

```
       int t=i;
       for(int k=1+ABS(5-t);k<=9-ABS(5-t);k++)
       {
          for(int j=1;j<=i+4;j++)
          {
             fprintf(fp,"%c%c%c
",6-i+j-1+48,translatecol(6-i+j-1,i+j-1)+48,k+48);
          }fprintf(fp,"0\n");
       }
   }
   for(int i=1;i<=4;i++)
   {
       int t=i+5;
       for(int k=1+ABS(5-t);k<=9-ABS(5-t);k++)
       {
          for(int j=1;j<=9-i;j++)
          {
             fprintf(fp,"%c%c%c ",j+48,translatecol(j,2*i+j-1+5)+48,k+48);
          }fprintf(fp,"0\n");
       }
   }
   ////////////////////////////////////////////////
   fclose(fp);
   return OK;
}
int translaterow(int a)
{return a;}
int translatecol(int i,int j)
{return (j+1-ABS(5-i))/2;}
/*
          _/    _/          _/    _/
         _/    _/    _/_/    _/    _/    _/_/
        _/_/_/_/  _/_/_/_/  _/    _/    _/   _/
       _/    _/  _/       _/    _/    _/   _/
      _/    _/    _/_/_/  _/_/  _/_/   _/_/

*/
void HELLO()
{
   printf("          _/    _/          _/    _/\n");
   printf("         _/    _/    _/_/    _/    _/    _/_/\n");
   printf("        _/_/_/_/  _/_/_/_/  _/    _/    _/   _/\n");
   printf("       _/    _/  _/       _/    _/    _/   _/\n");
   printf("      _/    _/    _/_/_/  _/_/  _/_/   _/_/\n");
```

```
}
/*
          _/_/_/_/  _/_/_/_/  _/       _/  _/_/_/_/         _/_/_/_/  _/    _/
_/_/_/_/  _/_/_/_/
         _/     _/  _/    _/  _/_/   _/_/  _/              _/    _/  _/    _/  _/
_/    _/
        _/         _/_/_/_/  _/  _/_/ _/  _/_/_/_/         _/    _/  _/    _/
_/_/_/_/  _/_/_/
       _/   _/_/  _/    _/  _/  _/   _/  _/              _/    _/  _/   _/   _/
_/   _/
        _/_/_/    _/     _/  _/   _/   _/  _/_/_/_/          _/_/_/_/  _/_/_/
_/_/_/_/   _/    _/
*/
void GAMEOVER()
{
    printf("\n\n\n\n\033[5;31;1m");
    printf("SUCH A DISAPPOINTMENT~~YOU LOST~~\n");

printf("=====================================================================
=====================================\n\n");
    printf("           _/_/_/_/  _/_/_/_/  _/      _/  _/_/_/_/        _/_/_/_/
_/    _/  _/_/_/_/  _/_/_/_/\n");
    printf("         _/    _/  _/    _/  _/_/   _/_/  _/             _/    _/  _/
_/   _/       _/    _/\n");
    printf("        _/        _/_/_/_/  _/  _/_/ _/  _/_/_/_/        _/    _/  _/
_/    _/_/_/_/  _/_/_/\n");
    printf("       _/   _/_/  _/    _/  _/  _/   _/  _/             _/    _/  _/ _/
_/       _/   _/\n");
    printf("        _/_/_/   _/    _/  _/  _/   _/  _/_/_/_/       _/_/_/_/  _/_/
_/_/_/_/  _/    _/\n");

printf("\n=====================================================================
=====================================\n");
    printf("TRY AGAIN.\n");
    printf("\033[0;3;0m");
}。
```