

**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：**  CS2002

**学 号：** U202015324

**姓 名：** 屈绍博

**指导教师：**  李丹

**报告日期：** 2021/10/8

**计算机科学与技术学院**

**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-to)/t]\*100%,其中t 为未对DPLL优化时求解基准算例的执行时间，to则为优化DPLL实现时求解同一算例的执行时间。(15%)
6. **SAT应用：**将数独游戏[5，6]问题转化为SAT问题[6]，并集成到上面的求解器进行问题求解，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[6-9]。(15%)

* **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]TanbirAhmed. An Implementation of the DPLL Algorithm.Masterthesis,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]CarstenSinz. Visualizing SAT Instances and Runsof the DPLL Algorithm.JAutom Reasoning (2007) 39:219–243

[5] Binary Puzzle：<http://www.binarypuzzle.com/>

[6] Putranto H. Utomo and Rusydi H. Makarim. Solving a Binary Puzzle. Mathematics in Computer Science,(2017) 11:515–526

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

[8]InsLynce and JolOuaknine. Sudoku as a sat problem.In Proceedings of the 9th InternationalSymposium on Artificial Intelligence and Mathematics, AIMATH 2006, Fort Lauderdale.Springer,2006.

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

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

http://zhangroup.aporc.org/images/files/Paper\_3485.pdf

**目录**

[1引言 1](#_Toc85126098)

[1.1课题背景与意义 1](#_Toc85126099)

[1.1.1 SAT问题简介 1](#_Toc85126100)

[1.1.2数独问题简介 2](#_Toc85126101)

[1.2国内外研究现状 2](#_Toc85126102)

[1.3课程设计的主要研究工作 3](#_Toc85126103)

[2系统需求分析与总体设计 4](#_Toc85126104)

[2.1系统需求分析 4](#_Toc85126105)

[2.2系统总体设计 4](#_Toc85126106)

[3.1有关数据结构的定义 7](#_Toc85126107)

[3.2主要算法设计 7](#_Toc85126108)

[3.2.1读取cnf文件，创建cnf公式 7](#_Toc85126109)

[3.2.1dpll算法 8](#_Toc85126110)

[3.2.1 将数独转换成sat问题 11](#_Toc85126111)

[3.2.1 生成数独和数独游戏 11](#_Toc85126112)

[4系统实现与测试 14](#_Toc85126113)

[4.1系统实现 14](#_Toc85126114)

[4.1.1 系统功能说明 14](#_Toc85126115)

[4.1.2 测试环境说明 17](#_Toc85126116)

[4.1.3 数据结构定义 18](#_Toc85126117)

[4.1.4 函数说明 18](#_Toc85126118)

[5总结与展望 29](#_Toc85126119)

[5.1全文总结 29](#_Toc85126120)

[5.2工作展望 30](#_Toc85126121)

[6体会 31](#_Toc85126122)

# 1引言

## 1.1课题背景与意义

### 1.1.1 SAT问题简介

布尔表达式是由布尔变量和运算符（NOT , AND ,OR）所构成的表达式。

  如果对于变量的某个true,false赋值，使得一个布尔表达式的值为true，则该布尔表达式是可满足的。例如布尔公式 A = ((NOT x) AND y) OR ( x AND (NOT z))，当 x = false, y = true, z = false时，该布尔表达式值为true，则表达式A就是可满足的。可满足性问题就是判定一个给定的合取范式的布尔公式是否是可满足的。

  已知的NP-complete问题多达几百个，但作为这些问题的“祖先”，历史上第一个被证明的NP-complete问题是来自于布尔逻辑的可满足性问题（SATISFIABLITY problem），简称为SAT。

  SAT问题是逻辑学的一个基本问题，也是当今计算机科学和人工智能研究的核心问题。工程技术、军事、工商管理、交通运输及自然科学研究中的许多重要问题，如程控电话的自动交换、大型数据库的维护、大规模集成电路的自动布线、软件自动开发、机器人动作规划等，都可转化成SAT问题。因此致力于寻找求解SAT问题的快速而有效的算法，不仅在理论研究上而且在许多应用领域都具有极其重要的意义。

  SAT的问题被证明是NP难解的问题。目前解决该问题的方法主要有完备的方法和不完备的方法两大类。完备的方法优点是保证能正确地判断SAT问题的可满足性，但其计算效率很低，平均的计算时间为多项式阶，最差的情况计算时间为指数阶，不适用于求解大规模的SAT问题。不完备的方法的优点是求解的时间比完备的方法快得多，但在很少数的情况下不能正确地判断SAT问题的可满足性。传统的方法有：枚举法、局部搜索法和贪婪算法等，但由于搜索空间大，问题一般难以求解。对于像SAT一类的NP难问题，采用一些现代启发式方法如演化算法往往较为有效。

### 1.1.2数独问题简介

数独是源自18世纪瑞士的一种数学游戏。是一种运用纸、笔进行演算的逻辑游戏。玩家需要根据9×9盘面上的已知数字，推理出所有剩余空格的数字，并满足每一行、每一列、每一个粗线宫（3\*3）内的数字均含1-9，不重复。

数独盘面是个九宫，每一宫又分为九个小格。在这八十一格中给出一定的已知数字和解题条件，利用逻辑和推理，在其他的空格上填入1-9的数字。使1-9每个数字在每一行、每一列和每一宫中都只出现一次，所以又称“九宫格”。

## 1.2国内外研究现状

布尔可满足性问题(即sat问题)是第一个被证明属于NP完全的问题。SAT问题的基本形式指给定一个命题变量集合X和一个X上的合取范式，判 断 是 否存在一个真值赋值，使为真。如果能找到，则称是可满足的(satisfiable)，否则称是不可满足的(unsatisfiable)。SAT问题的模型发现形式指当可满足时，给出使公式可满足的一组赋值。

根据SAT国际竞赛按照问题产生来源将SAT问题实例划分为应用类(Application)、组合类(Hardcombination)和随机类(Random)。

目前典型的SAT求解算法包括确定性算法和随机搜索算法两大类。确定性算法采取穷举和回溯思想，从理论上保证给定命题公式的可满足性，并在实例无解的情况下给出完备证明，但不适用于求解大规模的SAT问题。随机搜索算法主要基于局部搜索思想，绝大多数随机搜索算法不能判断SAT问题的不可满足性，但由于采用了启发式策略来指导搜索，在处理可满足的大规模随机类问题时，往往能比确定性算法更快得到一个解。

DPLL算法是一种判定SAT问题的高效算法。早在1960年，Davis 和Putnam就提出了最早的DPLL算法，当时称为DP算法，该算法的提出有效地降低了SAT问题的复杂性和求解器的空间限制这些棘手的情况，奠定了DPLL算法在判定SAT问题时的地位。DP算法通过在给定的表达式中进行变量消解，达到了降低搜索空间大小的目的。但当Logemann和Loveland尝试实现DP算法时，发现该算法在消解时占用过多的内存空间,这在当时的条件下是受限的。于是在1962年，Logemann和Loveland等人对DP算法进行了改进，改变了变量消解的方式，形成了最初的DPLL算法。此时的DPLL算法采用的是分支回溯策略，即不断地选择变量进行分支赋值，当发生冲突时再进行回溯。这种DPLL算法求解效率较低，最多只能解决10变量的SAT问题，且求解范围受限，对随机生成的实例求解效果较好，对实际应用转化而来的实例求解效果不佳。

随着研究的深入，优化技术的不断改进，DPLL算法亦趋于完善。但无论DPLL算法结合何种优化技术，在判定SAT问题时，总会有约80%的时间耗费在BCP过程中,而判定变量的选取结果直接影响该过程的运行时间，所以判定变量选取策略在DPLL算法的优化技术研究中占有重要的地位，这也就成为了历年来DPLL算法优化领域的热门课题之一。

## 1.3课程设计的主要研究工作

本次实验主要是研究sat问题的求解方法，了解各种dpll算法的改进，并且了解如何将实际问题建模转换为sat问题求解，同时熟悉文件操作，与实际精密结合，具体来说，主要工作如下：

1. 了解cnf文件的格式，如注释格式和正文格式，结合sat问题的研究，学习CNF范式的基本知识，知道每一行的含义，了解里面内容的结构
2. 设计一个合的数据结构，要求能够存储.cnf文件中的所有的有效信息，要求对子句以及变元的插入和删除尽量高效。
3. 编写dpll算法，能够较为高效地求解sat问题。然后查阅相关资料，对dpll算法进行优化，并且求解优化率，对相应优化方法进行相应分析。
4. 将数独问题转换为sat问题，写入.cnf 文件，然后利用求解器求解。利用挖洞法生成具有唯一解的数独，并且简单设计一个交互界面可供用户游玩数独游戏。

# 2系统需求分析与总体设计

## 2.1系统需求分析

本系统应当有两个主要功能，一是sat问题求解，二是数独的转换以及利用dpll求解数独。

第一个问题：设计一个数据结构，存储从.cnf文件中读入的数据。并编写读入函数跳过注释将.cnf文件转化为对应的格式存储在该数据结构中，如果发现.cnf文件格式错误要提醒用户并终止读入。读入结束后利用dpll算法，求解sat问题，将问题的解存在文件中。

第二个问题：按照一定的规则将数独转换为sat问题，将得到的数据存在sudoku.cnf文件中，然后在利用dpll算法，求解该sat问题。除此之外，还需要通过挖洞法生成一个具有唯一解的数独，然后编写一个简单的交互函数实现数独游戏的游玩，交互函数需要实时提示用户输入答案的正误以及要有显示正确答案的功能。还需要编写读入数独格局的函数并对现有数度格局进行求解。

## 2.2系统总体设计

这个系统主要有两个部分，一个是sat问题求解，二是数独游戏。如图2.1

Sat部分主要功能为.cnf格式文件的读入以及求解。

数独部分的主要功能为数独的随机生成、挖洞构建初盘，或是从现有数独格局中读入数独初盘，以及数独初盘向.cnf格式文件的转换，利用dpll求解转化为sat问题的数度问题，数独的简单游玩。



图2.1系统初始界面

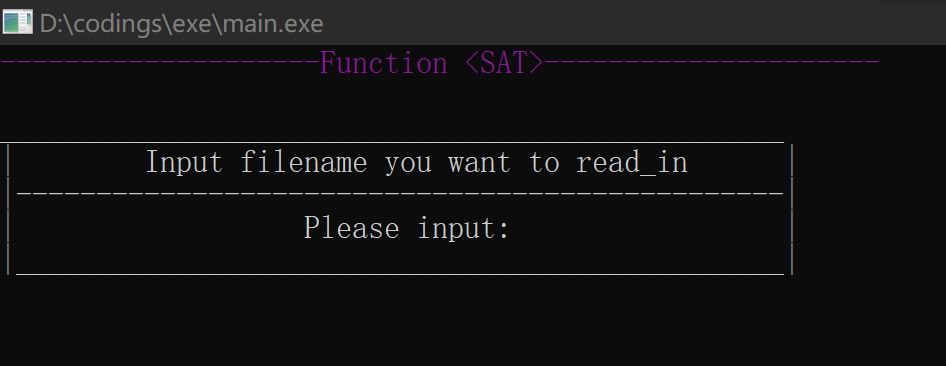


图2.2 sat界面

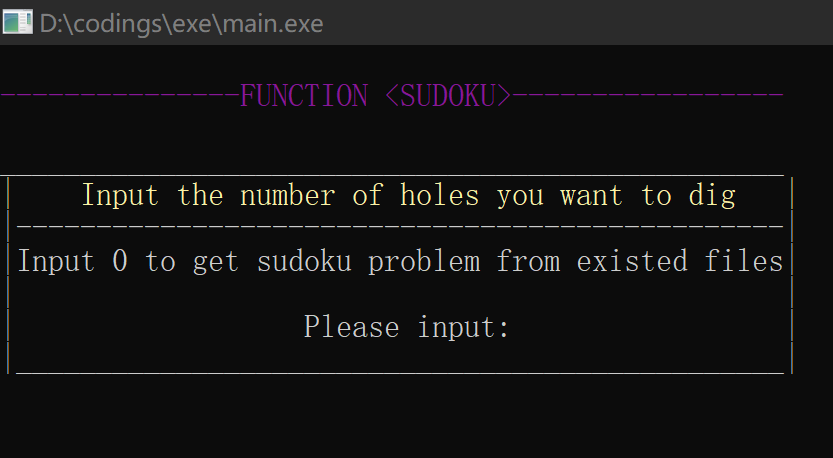


图2.3 数独界面

整个系统的流程图如图2.4：

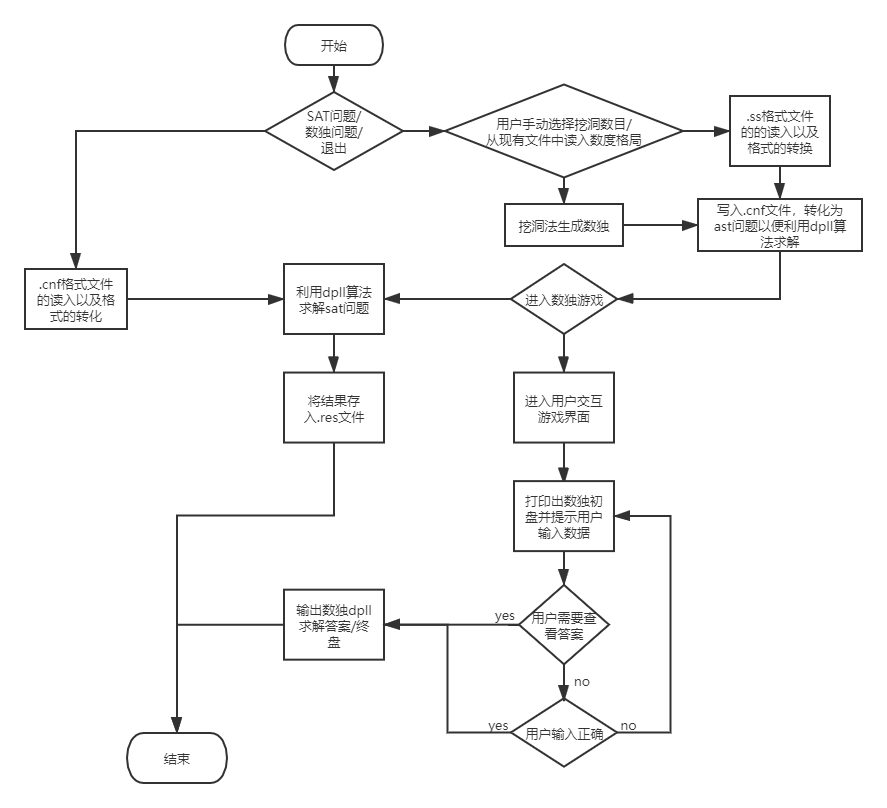


图2.4 系统流程图

**3系统详细设计**

## 3.1有关数据结构的定义

1.数据结构采用广义表，共有个两部分嵌套而成，一个是索引表，一个是子句链表。索引表部分存储了每个子句的头结点，构成链表，子句部分存储了该子句所有的文字，构成链表。

2．索引表结点由三部分组成，一个是对应子句的头结点，一个是对应子句的长度，一个是指向下一个索引表结点的指针。字句部分由两部分组成，一个是当前结点的数据，另一个是指向当前子句下一个变元结点的指针。

数据结构图示如下：

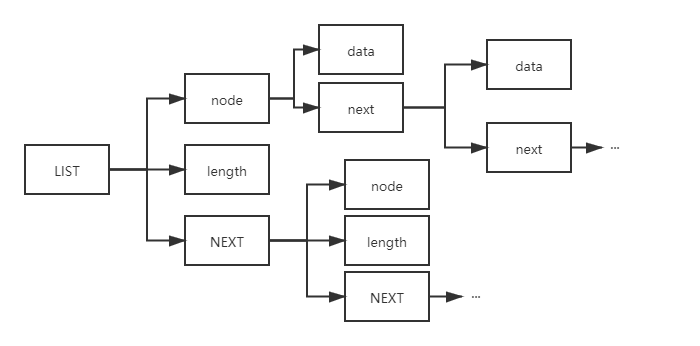


图3.1 数据结构关系图

## 3.2主要算法设计

### 3.2.1读取cnf文件，创建cnf公式

读取.cnf文件之前首先需要深刻理解.cnf文件的格式，.cnf文件主要是每一行的开头，是c，p或者数字，分这四类，c代表注释，可能有多行，p只有一行，该行内有关于此.cnf文件的基础信息，其他的直接读到文件末尾就可以了所以分三种情况

c开头：直接跳过

p开头：读取空格直到遇见第一个数字，第一个数字是文字的个数，第二个数字是子句的个数，将相应的数字存在相应的内容里面。注意要进行字符串到整形数的转换。

数字：一直读取直到结束，注意分为正和负两种情况，也进行字符串到整形数的转换。读到终止符0为止，将数据依次存入一个字句结构中。

如果均不属于上述情况，则说明该.cnf文件有格式错误，应报错提示用户并停止读入。

### 3.2.1dpll算法

Dpll算法的核心是回溯，重点是分支选取和剪枝。按照单子句规则以及分裂策略来选取变元分支，一步步减少字句数量以及字句长度直至求出最终结果。若对某个变元的两个分支均不能求出正确解则继续向上回溯，单子句以及分裂策略的解释如下。

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。

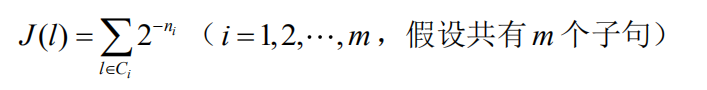
分裂策略。按某种策略选取一个文字*L*.如果*L*取真值，则根据单子句传播策略，可将*S*化成*S*2；若*L*取假值（即*¬L*成立）时，*S*可化成*S*1.

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，其执行过程可表示为一棵二叉搜索树,如下图2.2所示。



图3.2 DPLL算法搜索树

分支选择未优化之前是按照顺序选择变元，优化后是通过一个函数的大小来选择变元，由大到小选择，该函数的值为，其中n为子句中文字的个数。当然最值得注意的是如果一个子句只有一个文字则该文字的值已经确定，就可以减少很多计算



回溯主要是要恢复之前的答案，以及撤销对子句的删除操作

流程图如下

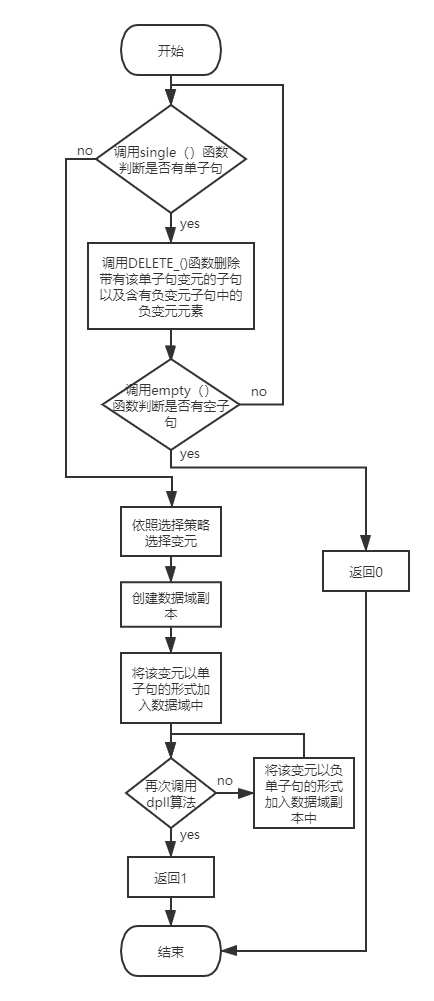


图3.3：dpll算法流程图

### 3.2.1 将数独转换成sat问题

将数独转换成sat问题需要对数独进行分析，数独游戏要求在9×9的网格中每个单元填入1至9的一个数字，必须满足三种约束：每一行、每一列及9个3×3的盒子中的数字都不重复。

我们利用三位整数ijk（i, j, k∈{1,2,…,9}）来代表位置是（i，j）的单元是否能填入数字k，如163变元表示第1行6列填入3；负文字 -452表示第4行5列不填入2。这样编码一共有999个变元。

对于第一个以及第二个约束条件，以行约束为例，要求每行需要填入1～9中的每个数字，且每个数字只出现一次。以第1行为例可表示为：

111 121 131 141 151 161 171 181 191 0 //第1行含有1

112 122 132 142 152 162 172 182 192 0 //第1行含有2

… …

119 129 139 149 159 169 179 189 199 0 //第1行含有9

-111 -121 0 //前两格不同时为1

-111 -131 0 //第1与第3格不同时为1

… …

-111 -191 0 //第1与第9格不同时为1

… …

列约束与行约束类似，也按照上方规则写入.cnf文件中。

不仅如此，我们还要约束每个单元中只能填入一个数字，除上述数独基础规则需要写入.cnf文件中意外，我们还要将所给数独初盘的提示数据按上述格式转换为变元后写入.cnf文件，例如初盘中位置（3，6）上的数据是9，我们就需要将变元369以单子句格式写入.cnf文件中。

然后通过计算，我们可以得到总的子句的个数写入文件，就可以通过前面已经解决的sat问题利用dpll算法求出答案。

### 3.2.1 生成数独和数独游戏

通过挖洞法生成具有唯一解的数独，挖洞法是分步进行的，利用dfs以及回溯的形式确保该数独有唯一解：

生成数独：首先随机生成一个未被挖掘的位置，判断如果该位置被挖掘，数独是否还有唯一解，如果有则挖去，如果没有唯一解则重新随机选择位置并进行上述步骤，以此类推直至挖的洞数已经达到了用户输入的指定值。

数独游戏：首先打印出数独初盘，给玩家需要填入的位置，玩家填入数据时，实时判断填入的数据是否正确，如果错误提示玩家重新填写。以此类推直至玩家解出该数独，如果玩家不能解出数独需要查看答案，则输入0显示答案。

利用dfs求解数独解的个数的流程图如下

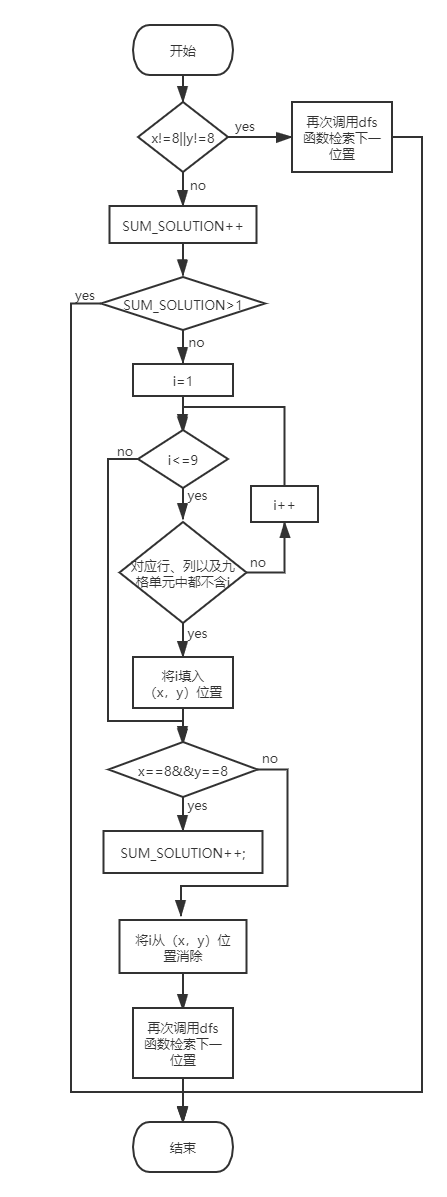


图3.4：挖洞法流程图

# 4系统实现与测试

## 4.1系统实现

### 4.1.1 系统功能说明

用户需求为较高效率地求解sat问题，并且通过这个技术解决一个实际问题，在这里是一个数独的问题，在数独问题中，我们还需要随机生成数独初盘以及通过挖洞法生成有唯一解的数度初盘供用户游玩。

简单的流程说明，或者说是程序操作说明书如下：

1. 完成sat问题的求解

首先选择功能1 SAT，如图

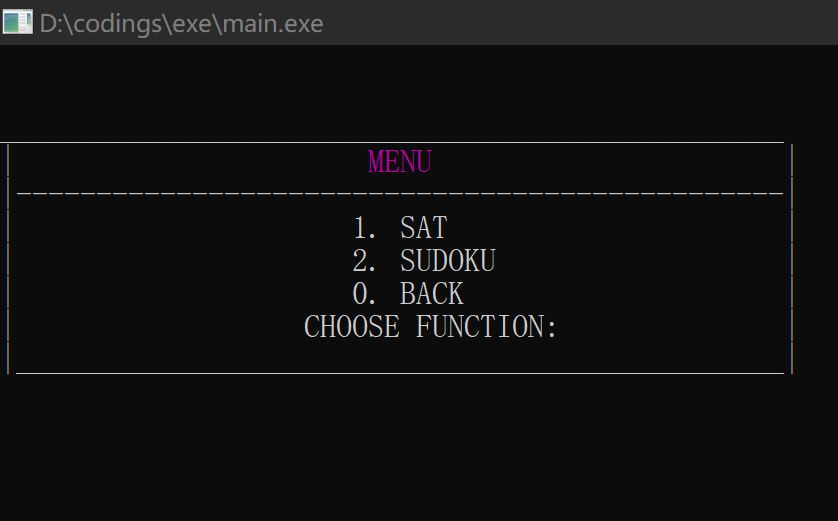


图4.1：选择sat问题求解

然后选择问题所在的.cnf文件读入，如图：

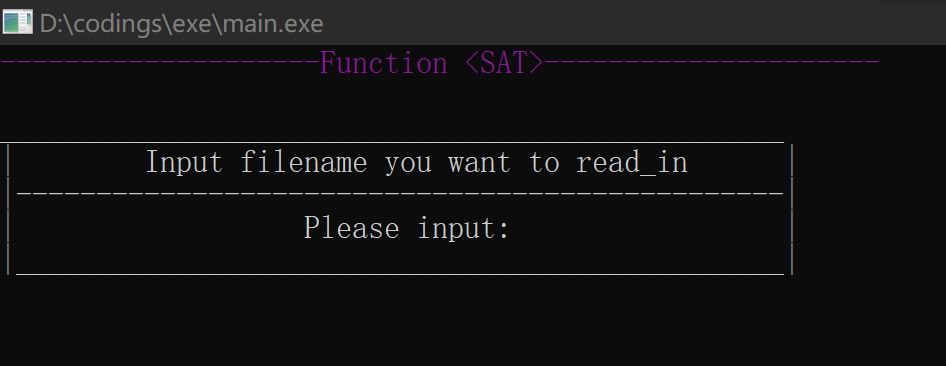


图4.2：选择读入文件

然后选择希望使用的选择策略进行求解，

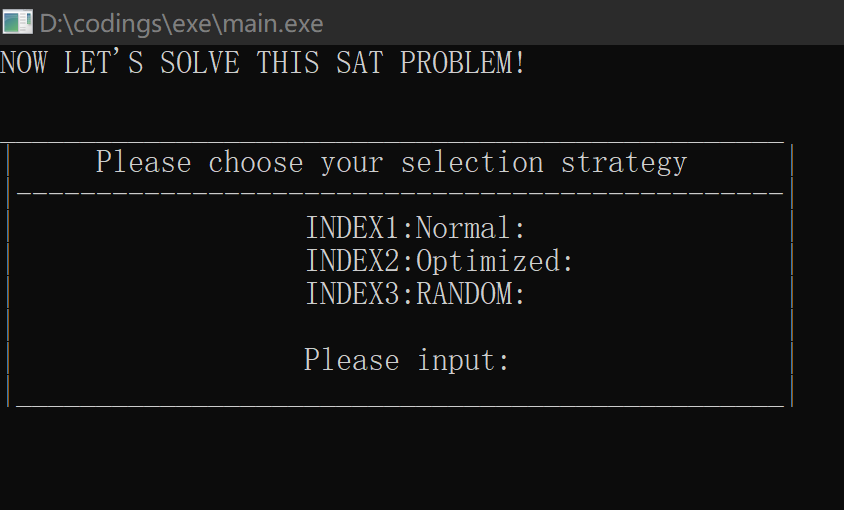


图4.3：选择选择策略

最终得出答案。

1. 求出数独并且求解

如图4.1所示，选择功能2 SUDOKU，如图：

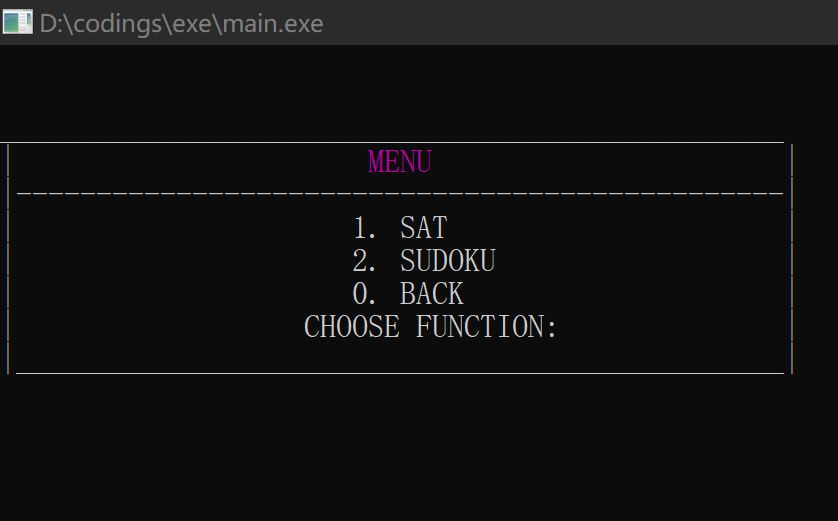


图4.4：选择生成数独

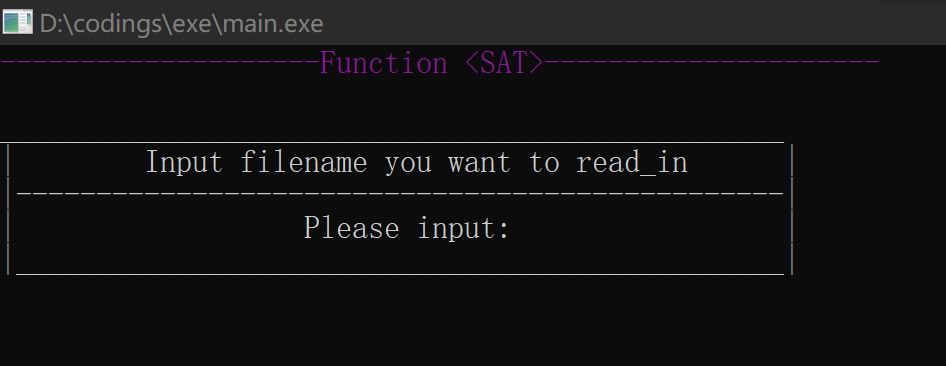


图4.5：选择从现有格局中读入或使用挖洞法构建初盘

然后你将看见本次数独游戏的初盘

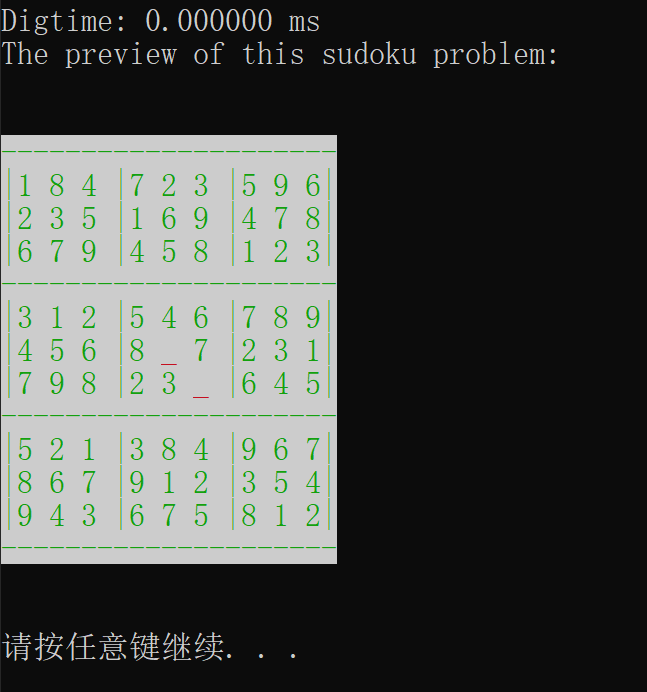


图4.6：显示数独初盘

接着数独初盘将会转化为.cnf格式文件存储，而你选择希望使用的选择策略对其进行求解

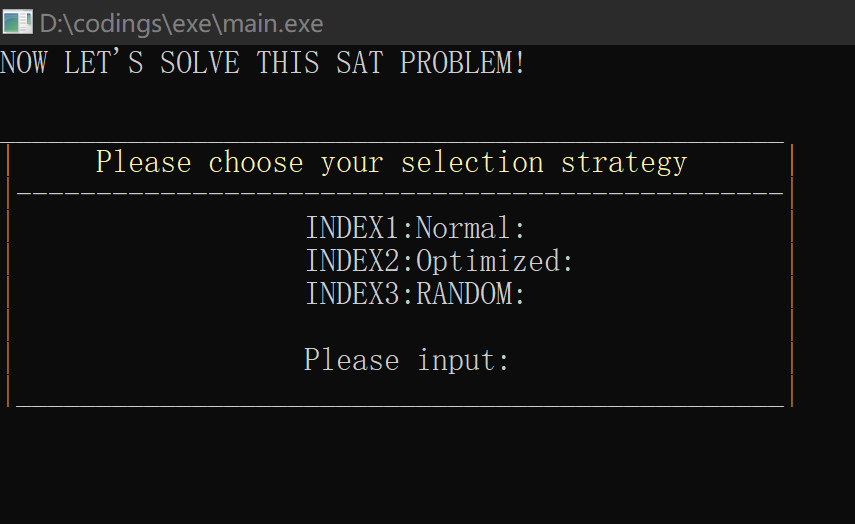


图4.7：选择选择策略

接下来是数独游玩部分，你要按照系统提示的顺序来填入你的答案，如果答案正确，系统会提示Right Answer！如果某步填入错误，系统会提示Wrong Answer！你将从这步开始重新填入答案。如果无法解出答案，可以在任一步输入0来查看答案。

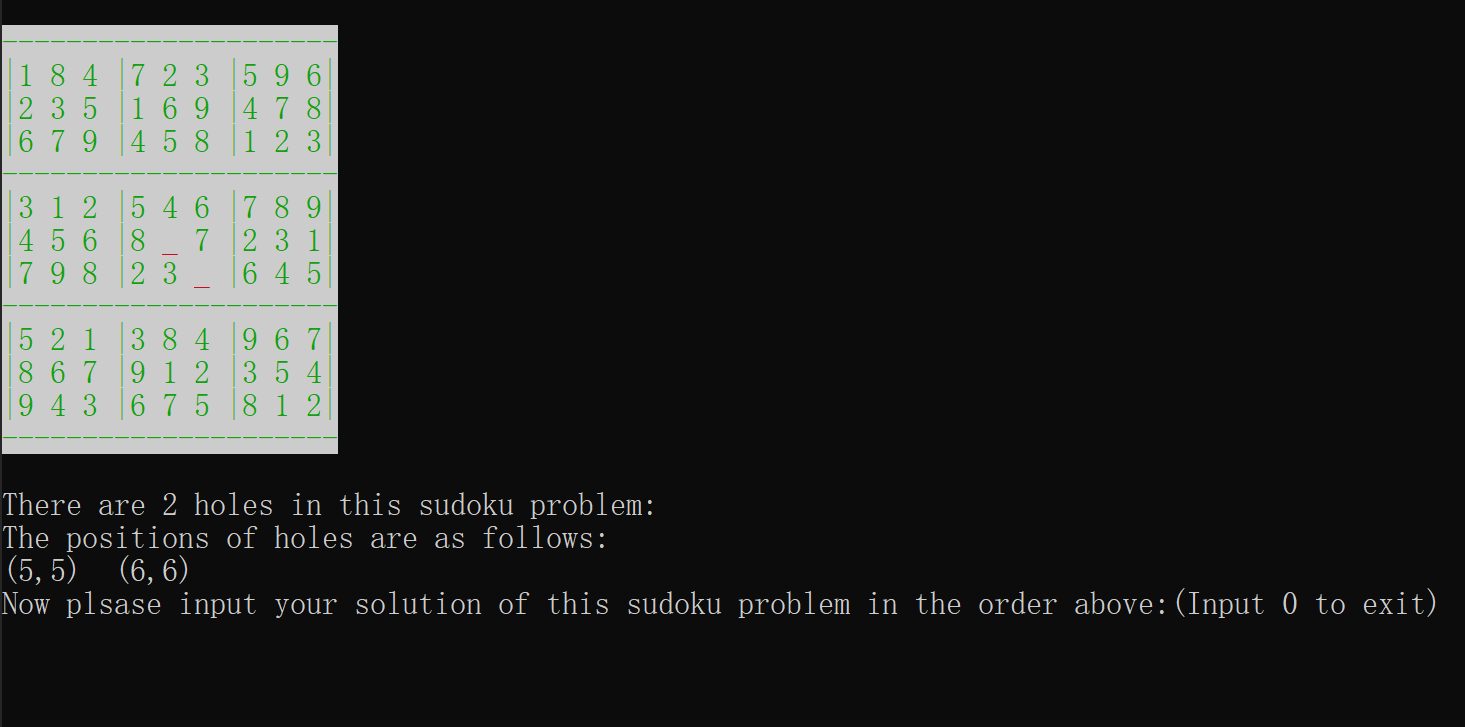


图4.8：按顺序输入答案

以上就是该程序使用的主要步骤。

### 4.1.2 测试环境说明

设备名称 LAPTOP-AR2NOGT7

处理器 Intel(R) Core(TM) i7-10710U CPU @ 1.10GHz 1.61 GHz

机带 RAM 16.0 GB (15.8 GB 可用)

产品 ID 00342-35858-58333-AAOEM

系统版本 Windows 10 家庭中文版

版本号 20H2

安装日期 ‎2021/‎4/‎20

操作系统内部版本 19042.1237

软件环境 Visual Studio Code （1.61.0）

### 4.1.3 数据结构定义

1.字节节点

typedef struct node

{

int data;//结点变元数据

struct node\* next;//指向下一个结点的指针

} node;

2.子句节点

typedef struct LIST

{

struct node\* head;//对应子句的头节点

int length;//子句的长度

struct LIST\* NEXT;//指向下一个子句的指针

} LIST;

### 4.1.4 函数说明

LIST\* single(LIST\* HEAD);//判断该子句是否为单子句

LIST \*Copy(LIST \*L);//构造一个数据副本，方便回溯使用

LIST\* add(LIST\* HEAD, int value); //在数据中加入一个值为value的单子句

LIST\* deletelist(LIST\* HEAD) //删除HEAD所指的子句

LIST\* deletenode(LIST\* HEAD, int value); //删除HEAD所指的子句中的变元值为value的结点

int DELETE(LIST\* HEAD, int value); //删除带有value值结点的子句，以及子句中含有-value值的子句中的-value结点

int empty(LIST\* HEAD); //判断数据域中是否有空子句

int DPLL(LIST\* HEAD, int\* RESULT,int maxtimes[]);//DPLL算法主体

int dig\_mark(int x,int y,int value); //在x行y列以及九格单元的标记中除去value的值的标记，表示value已经从（x，y）位置被删除了

int re\_dig\_mark(int x,int y,int value); //在x行y列以及九格单元的标记中加入value的值的标记，表示value在（x，y）位置上

int show();//展示数独初盘

int ass(int x,int y,int value,int index); //处理数独初盘，如果index为1将初盘（x，y）位置设定为value，如果index为0将初盘（x，y）位置设定为0，表示已经被挖去

void dfs(int x,int y); //利用dfs算法求解数独解的个数

int singlesolution();//判断数独是否有唯一解

void createStartinggrid(int num); //挖掉num个位置，创造初盘

int Digit(int a[][9], int x, int y); //在第一行的基础上随机生成一个9\*9的数独终盘，利用行标记，列标记以及九格单元标记判断该位置是否能填入当前随机到的数字

void randomFirstRow(int a[], int n); //随机生成终盘第一行

void createSudoku(int a[][9]); //随机生成一个9\*9的数独终盘

void print();//展示数独终盘

int ToCnf(int a[][9],int holes); //将数独初盘转换为sudoku.cnf文件

int CNF\_TRANSLATER\_SUDOKU();//处理sudoku.cnf文件的函数，将sudoku.cnf文件的数据写入广义表数据域中

int find\_solution();//利用dpll算法求解sudoku.cnf

int CNF\_TRANSLATER();//处理.cnf文件的函数，将.cnf文件的数据写入广义表数据域中

int SAT\_FUNCTION()//sat求解问题函数;

int SUDOKU\_FUNCTION();//数独游戏功能函数

函数调用关系如下：

首先是第一部分的调用情况

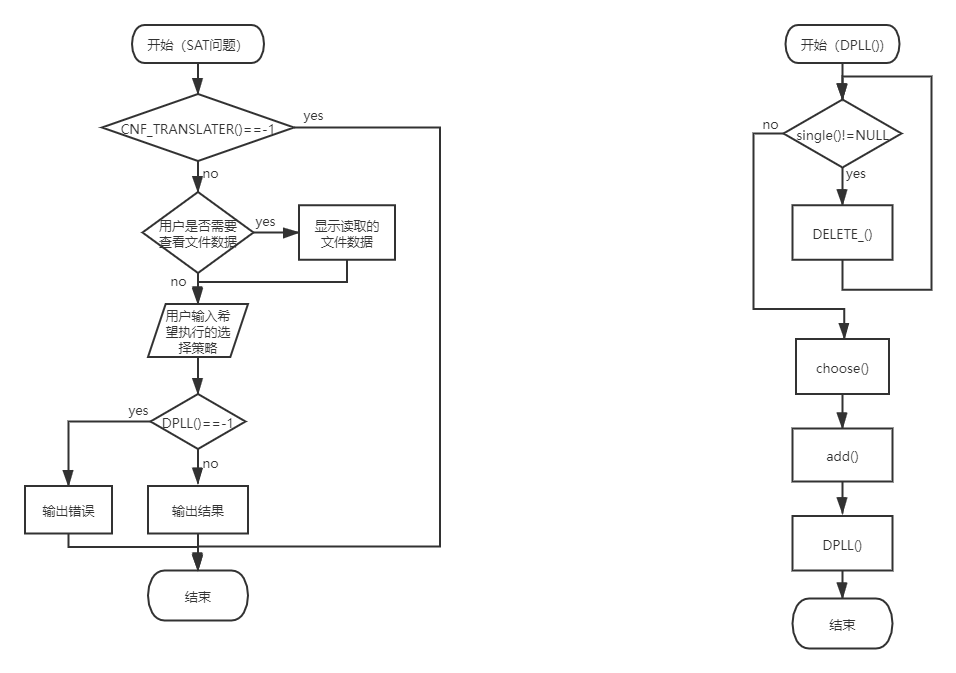


图4.1：sat问题求解调用情况以及DPLL算法调用函数情况

其次是第二部分的调用情况：

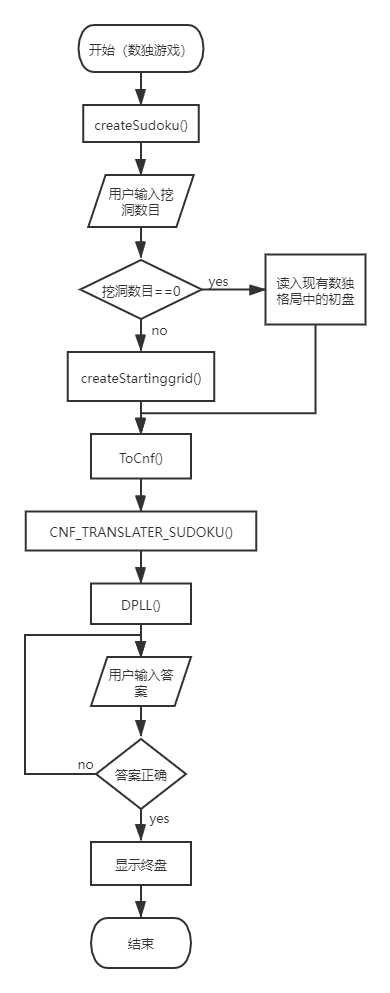


图4.2：数独问题求解调用情况

**4.2系统测试**

系统功能主要分为两部分，1：求解sat问题，2：求解数独问题

而对于第一个功能，分为两个部分，1：cnf文件读入 2：sat问题求解

对于第二个功能，第一部分是生成数独问题，有两种方法，一：从现有的数独算例文件中读取 二：挖洞法生成一个具有唯一解的数独供操作

然后生成数独以后我们需要求解数独，或者是玩数独游戏，还需要测试是否正确。

先测试第一个模块，cnf文件的读入和sat问题求解以及答案是否正确放在一起测试。直接测试给定的12个文件，由于分为优化前和优化后，把相应的时间以及优化率放在一个表格之中

表4.2.1：sat问题求解测试表

|  |  |  |
| --- | --- | --- |
| 输入 | 输出 | 结果分析 |
| 1（错误格式） | ERROR FORMAT! | 正常 |
| 2（满足算例） | 输出运算结果 | 正常 |
| 4（不满足算例） | 输出运算结果 | 正常 |
| 其他测试集（计算优化率） | 输出运算结果 | 正常 |

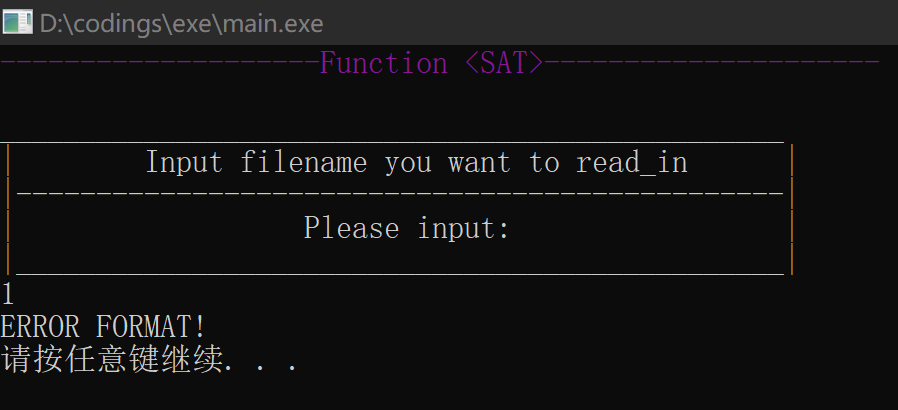


图4.3：格式错误文件的测试结果

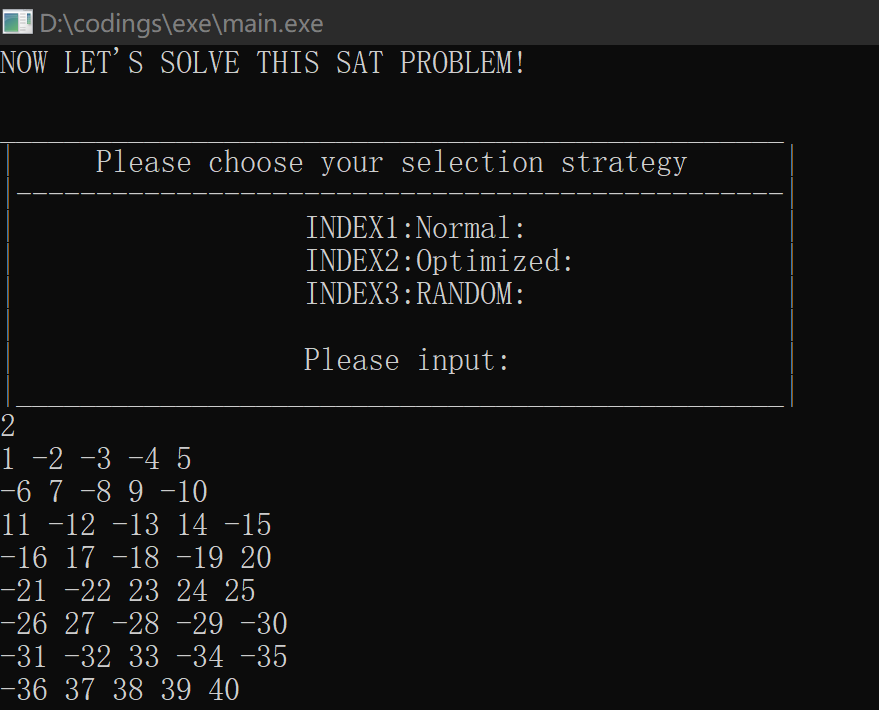


图4.4：满足算例的测试结果

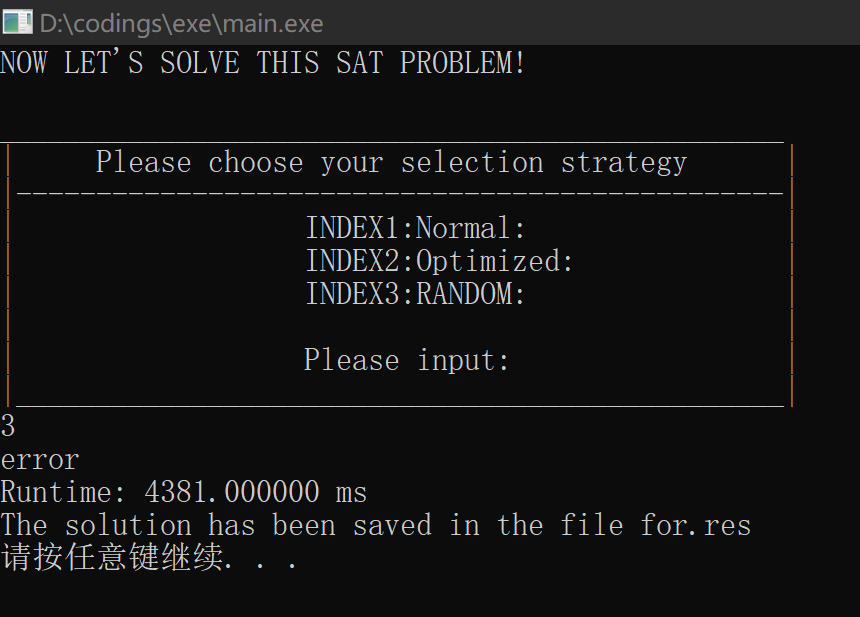


图4.5：不满足算例的测试结果

表4.2.2：优化率表

|  |  |  |  |
| --- | --- | --- | --- |
| 文件序号 | 优化前 | 优化后 | 优化率 |
| 1 |  |  | 9.2% |
| 2 | 无法运行 |  | 1 |
| 3 |  |  | -49.7% |
| 4 |  |  | 15.9% |
| 5 |  |  | 44.1% |
| 6 | 无法运行 |  | 1 |

然后是测试第二部分的功能

表4.2.3：数独求解测试表

|  |  |  |
| --- | --- | --- |
| 输入 | 目的 | 结果分析 |
| 2 | 挖洞法生成两个空的数独初盘 | 正常 |
| 0 12 | 从现有数独格局中读入初盘 | 正常 |
| / | 进行数独游戏（答案正确） | 正常 |
| / | 进行数独游戏（答案错误） | 正常 |
| / | 进行数独游戏（输出答案） | 正常 |

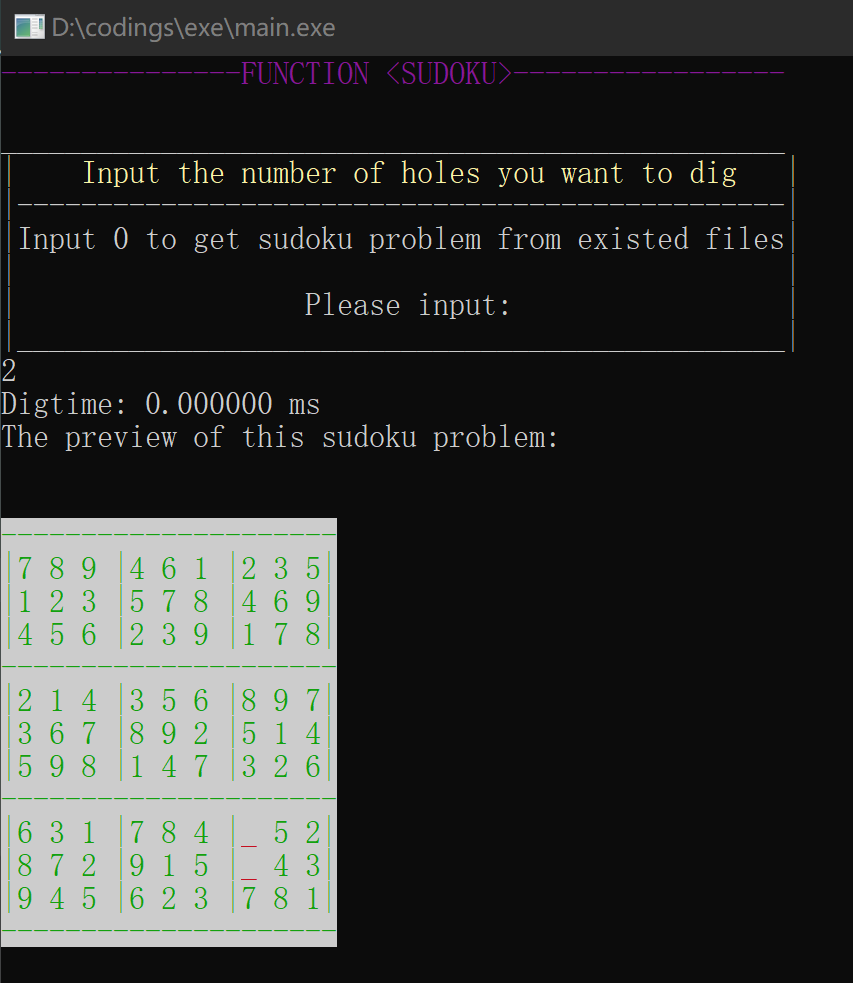


图4.6：挖洞法生成数独

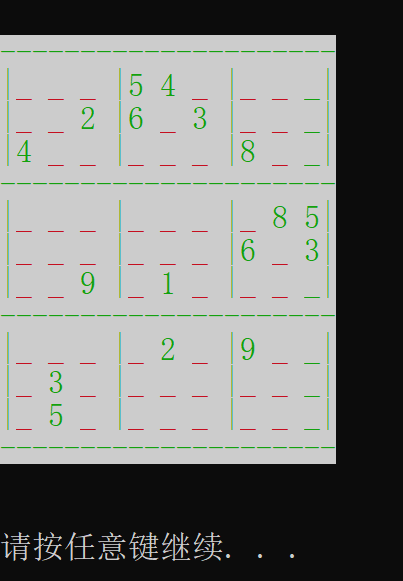


图4.7：从现有数独格局中读入数独初盘

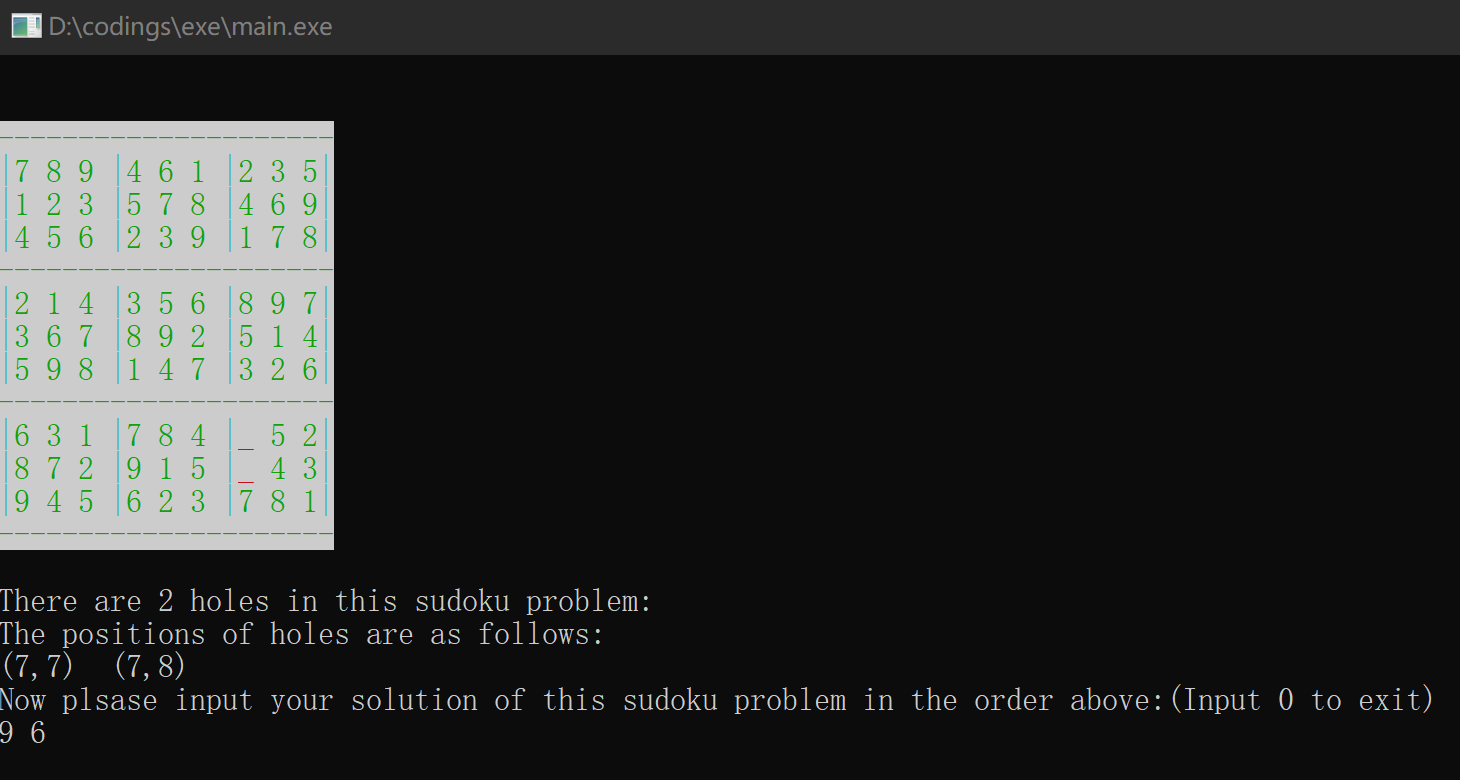


图4.8：进行数独游戏（输入正确答案）

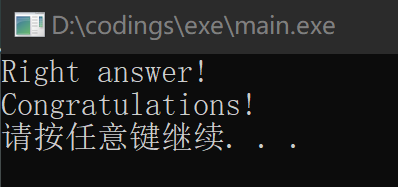


图4.9：验证答案是否正确

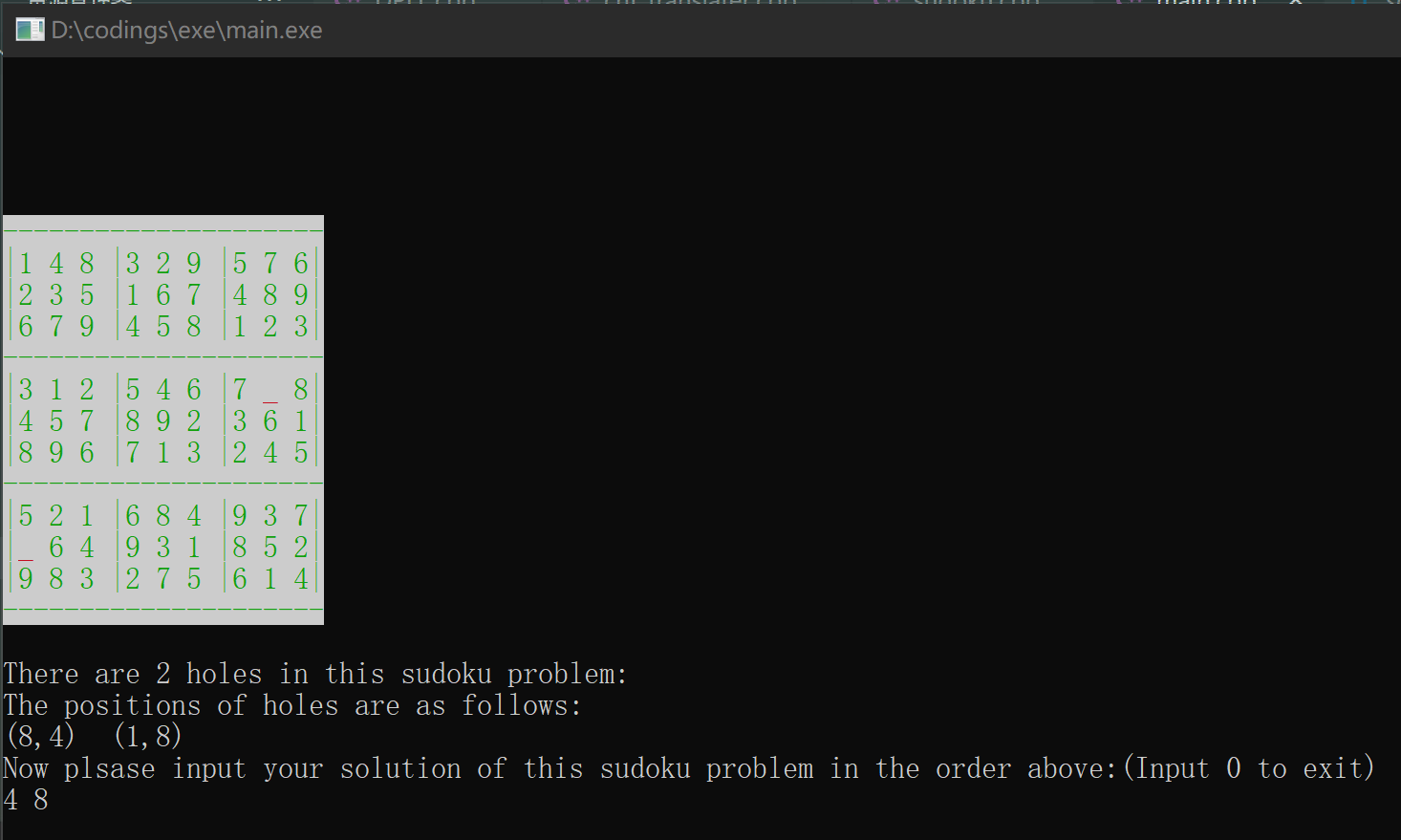


图4.10：进行数独游戏（输入错误答案）

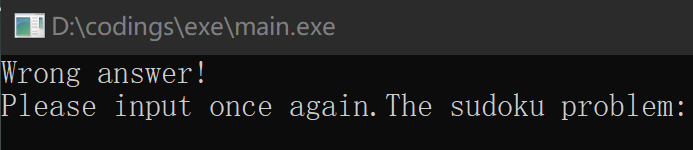


图4.11：验证答案是否正确

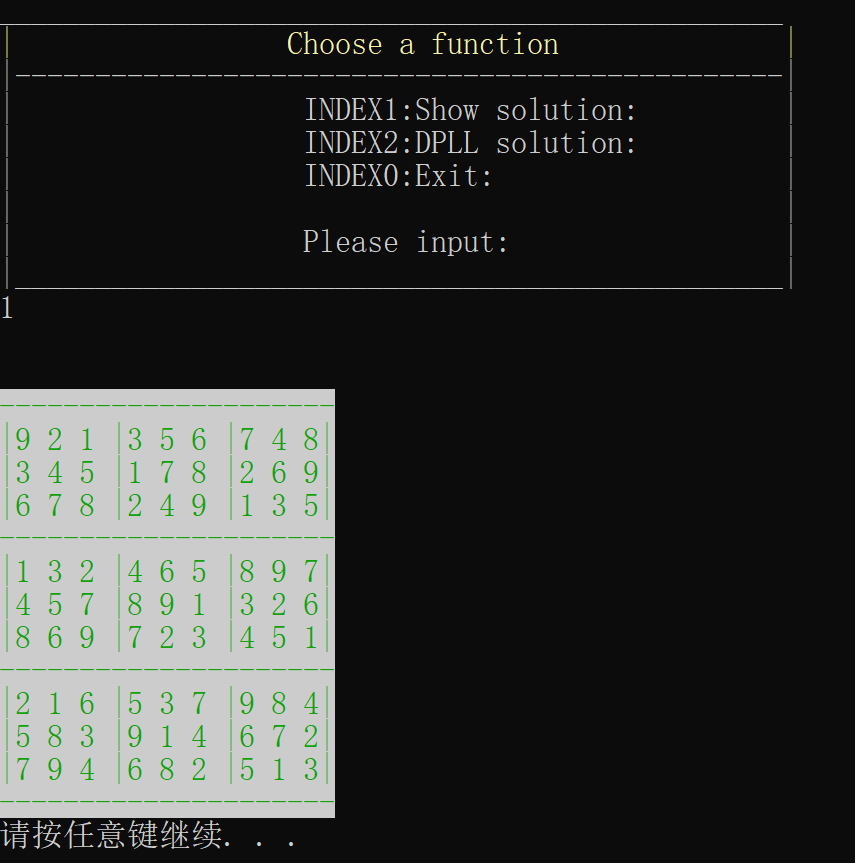


图4.12：进行数独游戏（查看答案）

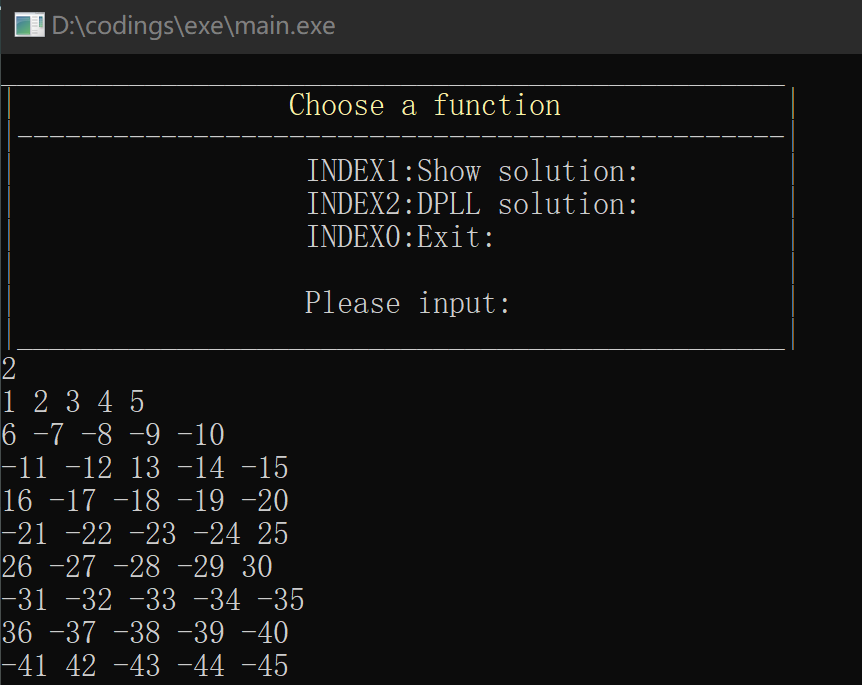


图4.13：进行数独游戏（查看sat问题的解）

# 5总结与展望

## 5.1全文总结

对自己的工作做个总结，主要工作如下：

1. 通过阅读任务书以及网上搜集资料对dpll算法有了初步的理解，也对程序的整体结构以及需要应用的数据结构有了初步的构思。

2. 设计数据结构，我选择了广义表的数据结构，因为在dpll算法运行过程中会涉及到大量的删除插入以及查找操作，广义表能高效地进行数据的删除以及插入，但内存开销较大，在权衡过后，我选择了广义表作为程序基础的数据结构。

3. 文件读入部分的编写较为简单，主要为注释的判断及跳过，正文数据内容的写入，内存空间的开辟，在.cnf文件的格式转换过程中我没有遇到困难。

4. dpll算法编写的过程中我遇到了一些困难，在刚开始编写dpll算法的过程中我没有释放内存空间，导致小型算例可以成功运算，而几乎所有中大型算例都会内存溢出的问题，而且程序编写初期我用的IDE是code blocks，他会限制单个程序的内存开销不能超过2GB，所以在内存管理方面我遇到了不小的阻力，在进行了部分内存的释放以及更换了IDE后，这个问题得到了初步的解决。

5. 数独游戏初看起来无从下手，但在阅读了任务书以及相关资料后解决方案也就变得清晰了起来，数独的随机终盘利用数个标记数组以及随机数的调用即可解决，而挖洞法构造初盘也不是十分困难，我利用了dfs算法，在每次挖洞之后判断数独解的个数，如果超过1个就更换位置挖洞，以此类推即可得到一个答案唯一的数独初盘。游戏交互仅需要一些简单的数据判断即可解决。

6. 利用dpll求解数独问题初看来也十分棘手，这个问题也在阅读了相关资料后便得到了解决，变元定义，子句写入等方法均在资料中有所提示，转化为.cnf文件后便可调用dpll算法进行求解，这个问题也得到了解决。

7. dpll算法的优化，从分支选取和数据结构两个方面进行优化，使得程序在面对更加复杂的数据能够加快计算速度。

8. 程序仍需继续优化，还存在着许多冗杂时间以及空间开销，尤其是内存开销，由于数据结构以及算法设计等问题，我很难在整体思路不进行改变的情况下对内存进行完全的释放，每次运算或者回溯后还有部分空间尚未释放，这也是这个程序存在的最主要的问题。

## 5.2工作展望

对于这个实验来说，我认为有三个方面的问题可以进行研究，也比较值得研究的是：

（1）优化分支选取策略，当前优化策略（按照出现次数选择）还是太过低效，在变元选择方面尚有很大的优化空间，选择策略对整体的运行时间开销和空间开销都有着很大的影响。

（2）优化数据结构，广义表虽然能提供高效的删除和插入操作，但内存开销过大，尤其是在进行每一步分支选取时都要进行副本的复制以备回溯使用，而这些副本的绝大部分在答案求解出之前是无法被释放的，广义表的数据结构过于简单，如果能加入更多的元素，重新设计一个数据结构或许能解决这个问题。

（3）实际问题转化为sat问题的研究，sat问题不仅仅能引用到数独问题上，许多现实中的实际问题也能映射成为sat问题，利用dpll问题进行高效的求解，数独只是其中很简单的一个，如何将许多问题转化为sat问题也是需要继续研究的方向之一。

# 6体会

这是我第一次进行从数据结构到算法设计，从数据读入到数据处理完全独立设计的实验，实验共计千余行代码，对于我来说还是有一定的难度的，颇具挑战性，也令我印象和体会颇深。对于这次课程设计实验，我有如下几点体会以及教训。

1. 做好这次实验不仅要有扎实的语言功底，也要有纵观整个程序设计的宽阔视野，二者缺一不可。在用惯了C++的一系列类库之后，我的C语言明显有些生疏，在编写数据结构以及对数据的处理过程中犯了不少低级错误，这也告诫我们不能过于依赖封装库，首先要清楚其底层逻辑。说起纵观整个程序设计的宽广视野，虽然dpll算法的大意已经给明，但却有多种实现方法，这就要求我们在设计数据结构时便要有对dpll算法设计的一个总体的认识。
2. 要巧妙地进行调试，调试不仅仅是加几个断点就盯着监视框看，巧妙地设计调试部分能让我们很快地找出程序的问题所在并加以解决。
3. 变量名函数名不能随心所欲地起，程序中也要适当加入注释，在构建这种包含很多函数的项目时，很容易混淆各种变量以及函数，所以要对易混淆的部分加以注释，并且使用清晰、意义明确的名称。

当然，困难的东西总能给人更多的收获，在这次实验中，我不仅温习了C语言的各种知识，加强了我的C语言代码能力。也增强了我对广义表数据结构的理解，增强了我的通观全局的能力，也让我对dpll算法有了一定的了解。

不仅如此，这次课程设计实验还锻炼了我的阅读和分析能力，锻炼了我将纸面上的理论转换为代码的能力，这是十分可贵的。

-

**参考文献**

[1]王静康,张凤宝,夏淑倩等.论化工本科专业国际认证与国内认证的“实质性”.高等工程教育研究,2014,5:1-4

[2]Stone J A, Howard L P. A simple technique for observing periodic nonlinearities in Michelson interferometers. Precision Engineering,1998,22(4):220-232

[3]朱印红,袁衍明.Dreamweaver完美网页设计——技术入门篇.(第一版).北京:中国电力出版社,2006:19～20

[4]Lewis S L. Physics and chemistry of the solar system.北京:北京大学出版社,2014.1～2

[5]陈剑.上博简《民之父母》“而得既塞於四海矣”句解释[EB/OL］.简帛研究网站，http://www.bamboosilk.org/Wssf/2003/chenjian03.htm．2003-01-18

**附录**

//hellodpll.h

#ifndef HELLODPLL\_H\_INCLUDED

#define HELLODPLL\_H\_INCLUDED

#include <stdio.h>

#include <malloc.h>

#include <stdlib.h>

#include <string.h>

#include <windows.h>

#include <iostream>

#include <ctime>

#include <algorithm>

#include <fstream>

#define OK 1

#define ERROR 0

using namespace std;

int sudoku\_begin[9][9];

int sudoku\_dig[9][9];

int line\_mark[9][10];

int row\_mark[9][10];

int unit\_mark[3][3][10];

int sudoku\_solution[730];

int holes\_mark[82][2];

int RESULT[5000];

int times[5000],maxtimes[5000],pos[5000],neg[5000];

int SUM\_SOLUTION;

int INDEX;

int m,n;

typedef struct node

{

    int data;

    struct node\* next;

} node;

typedef struct LIST

{

    struct node\* head;

    int length;

    struct LIST\* NEXT;

} LIST;

LIST\* HEAD;

FILE\* fp;

clock\_t Start1,End1;

LIST\* single(LIST\* HEAD);

LIST \*Copy(LIST \*L);

LIST\* add(LIST\* HEAD, int value);

LIST\* deletelist(LIST\* HEAD);

LIST\* deletenode(LIST\* HEAD, int value);

int DELETE\_(LIST\* HEAD, int value);

int empty(LIST\* HEAD);

int DPLL(LIST\* HEAD, int\* RESULT,int maxtimes[]);

int dig\_mark(int x,int y,int value);

int re\_dig\_mark(int x,int y,int value);

int show();

int ass(int x,int y,int value,int index);

void dfs(int x,int y);

int singlesolution();

void createStartinggrid(int num);

int Digit(int a[][9], int x, int y);

void randomFirstRow(int a[], int n);

void createSudoku(int a[][9]);

void print();

int ToCnf(int a[][9],int holes);

int CNF\_TRANSLATER\_SUDOKU();

int find\_solution();

int CNF\_TRANSLATER();

int SAT\_FUNCTION();

int SUDOKU\_FUNCTION();

void setColor(unsigned short ForeColor=7,unsigned short BackGroundColor=0)

{

HANDLE handle=GetStdHandle(STD\_OUTPUT\_HANDLE);//鑾峰彇褰撳墠绐楀彛鍙ユ焺

SetConsoleTextAttribute(handle,ForeColor+BackGroundColor\*0x10);//璁剧疆棰滆壊

}

void SetPos(int x,int y)

{

    COORD pos;

    HANDLE handle;

    pos.X=x;

    pos.Y=y;

    handle=GetStdHandle(STD\_OUTPUT\_HANDLE);

    SetConsoleCursorPosition(handle,pos);

}

#endif // HELLODPLL\_H\_INCLUDED

//main.cpp

#include "hellodpll.h"

#include "sudoku.cpp"

#include "cnf\_translater.cpp"

#include "DPLL.cpp"

int main()

{

    int x = 1, result, i;

    // LIST\* HEAD;

    // FILE\* fp;

    // clock\_t Start1,End1;

    while(x)

    {

        system("cls");

        printf("\n\n");

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

        printf("|");

        setColor(13,0);

        printf("                      MENU                      ");

        setColor(7,0);

        printf("|\n");

        printf("|------------------------------------------------|\n");

        printf("|                     1. SAT                     |\n");

        printf("|                     2. SUDOKU                  |\n");

        printf("|                     0. BACK                    |\n");

        printf("|                  CHOOSE FUNCTION:              |\n");

        printf("|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\n");

        scanf("%d", &x);

        if(x==1)

        {

            SAT\_FUNCTION();

        }

        else if(x==0) {

            printf("\nBye bye~;");

            system("Pause");

            return 0;

        }

        else if(x==2){

            if(SUDOKU\_FUNCTION()==-1);

        }

        else

        {

            printf("\nILLIGAL FUNCTION!\n");

            getchar();

            getchar();

        }

    }

    return OK;

}

//sudoku.cpp

#include "hellodpll.h"

int find\_solution(){

    // CNF\_TRANSLATER\_SUDOKU();

    LIST \*POSITION,\*HEAD;

    int i,k,b,c;

    memset(sudoku\_solution,0,sizeof(sudoku\_solution));

    for(i=0;i<1000;i++){

        sudoku\_solution[i]=RESULT[i];

    }

    b=0;

    // show();

    // printf("\n");

    // print();

    for(i=0;i<9;i++){

        for(k=0;k<9;k++){

            if(sudoku\_dig[i][k]==0){

                holes\_mark[b][0]=i;

                holes\_mark[b][1]=k;

                // printf("%d ",sudoku\_begin[i][k]);

                // holes\_mark[b][2]=sudoku\_begin[i][k];

                for(c=(i+1)\*100+(k+1)\*10;c<(i+1)\*100+(k+1)\*10+9;c++){

                    if(RESULT[c])holes\_mark[b][2]=c-((i+1)\*100+(k+1)\*10-1);

                }

                //printf("%d ",holes\_mark[b][2]);

                b++;

            }

        }

    }

    printf("The sudoku problem:\n\n");

    show();

    printf("\nThere are %d holes in this sudoku problem:\nThe positions of holes are as follows:\n",b);

    for(i=0;i<b;i++){

        printf("(%d,%d)  ",holes\_mark[i][1]+1,holes\_mark[i][0]+1);

        if((i+1)%5==0)printf("\n");

    }

    printf("\nNow plsase input your solution of this sudoku problem in the order above:(Input 0 to exit)\n");

    for(i=0;i<b;i++){

        scanf("%d",&c);

        if(c==sudoku\_begin[holes\_mark[i][0]][holes\_mark[i][1]])continue;

        //if(c==holes\_mark[b][2])continue;

        else if(c==0)break;

        else {

            system("cls");

            printf("Wrong answer!\nPlease input once again.");

            i=-1;

            printf("The sudoku problem:\n\n");

            show();

            printf("\nThere are %d holes in this sudoku problem:\nThe position of holes are as follows:\n",b);

            for(k=0;k<b;k++){

                printf("(%d,%d)  ",holes\_mark[k][1]+1,holes\_mark[k][0]+1);

                if((k+1)%5==0)printf("\n");

            }

            printf("\nNow plsase input your solution of this sudoku problem in the order above:(Input 0 to exit)\n");

        }

    }

    system("cls");

    printf("Right answer!\nCongratulations!\n");

    system("Pause");

    return 0;

}

int SUDOKU\_FUNCTION(){

    int i,k,b,c,flag=0;

            system("cls");

            setColor(5,0);

            printf("\n---------------FUNCTION <SUDOKU>-----------------\n\n");

            setColor(7,0);

            createSudoku(sudoku\_begin);

            for(i=0;i<9;i++){

                for(k=0;k<=9;k++){

                    line\_mark[i][k]=1;

                    row\_mark[i][k]=1;

                }

            }

            for(i=0;i<3;i++){

                for(k=0;k<3;k++){

                    for(b=0;b<10;b++){

                        unit\_mark[i][k][b]=1;

                    }

                }

            }

            int hole\_num;

            // printf("Input the number of holes you want to dig\n(Input 0 to get sudoku problem from existed files):");

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

            setColor(14,0);

            printf("|    Input the number of holes you want to dig   |\n");

            setColor(7,0);

            printf("|------------------------------------------------|\n");

            printf("|Input 0 to get sudoku problem from existed files|\n");

            printf("|                                                |\n");

            printf("|                  Please input:                 |\n");

            printf("|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\n");

            scanf("%d",&hole\_num);

            if(hole\_num!=0)

            {        Start1=clock();

                    createStartinggrid(hole\_num);

                    End1=clock();}

            else {

                    flag=1;

                    char c,tmp[10480],name[100],filename[100];

                    system("cls");

                    // printf("                        Input filename:");

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

                    setColor(14,0);

                    printf("|        Input filename you want to read\_in      |\n");

                    setColor(7,0);

                    printf("|------------------------------------------------|\n");

                    printf("|                  Please input:                 |\n");

                    printf("|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\n");

                    scanf("%s",filename);

                    strcpy(name,"C:\\Users\\13226\\Desktop\\SAT\\");

                    strcat(name,filename);

                    strcat(name,".ss");

                    fp=fopen(name, "r+");

                    for(i=0,k=0;;){

                        c=getc(fp);

                        if(c=='!')continue;

                        else if(c=='.'){

                            sudoku\_dig[i][k]=0;

                            hole\_num++;

                        }

                        else if(c=='-'){

                            char tmp[10471];

                            fgets(tmp,10470,fp);

                            continue;

                        }

                        else {

                            sudoku\_dig[i][k]=c-'0';

                        }

                        // x+(y+1)/9,(y+1)%9+1

                        if(i==8&&k==8)break;

                        if(k==8)getc(fp);

                        i=i+(k+1)/9;

                        k=(k+1)%9;

                    }

            }

            ToCnf(sudoku\_dig,hole\_num);

            printf("Digtime: %lf ms\n",(double)(End1-Start1)/CLOCKS\_PER\_SEC\*1000.0);

            //printf("The sudoku problem has been translated to .cnf foramt and been saved in the file ""sudoku.cnf"" \n");

            printf("The preview of this sudoku problem:\n\n");

            show();

            printf("\n\n");

            int sh;

            // printf("Input 1 to show the translated data of the sudoku problem\nInput 0 to skip\n");

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

            // printf("|                 Choose a function              |\n");

            // printf("|------------------------------------------------|\n");

            // printf("|                  INDEX1:Show solution:         |\n");

            // printf("|                  INDEX2:DPLL solution:         |\n");

            // printf("|                  INDEX3:Exit:                  |\n");

            // printf("|                                                |\n");

            // printf("|                  Please input:                 |\n");

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

            // scanf("%d",&sh);

            sh=0;

            system("Pause");

            LIST \*POSITION;

            node \*p;

            CNF\_TRANSLATER\_SUDOKU();

            char name[100];

            if(sh==1){

                POSITION=HEAD->NEXT;

                for(;POSITION!=NULL;POSITION=POSITION->NEXT){

                    p=POSITION->head;

                    for(;p!=NULL;p=p->next){

                        printf("%d ",p->data);

                    }

                    printf("  length=%d\n",POSITION->length);

                }

            }

            // system("Pause");

            system("cls");

            printf("NOW LET'S SOLVE THIS SAT PROBLEM!\n\n");

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

            setColor(14,0);

            printf("|     Please choose your selection strategy      |\n");

            setColor(7,0);

            printf("|------------------------------------------------|\n");

            printf("|                  INDEX1:Normal:                |\n");

            printf("|                  INDEX2:Optimized:             |\n");

            printf("|                  INDEX3:RANDOM:                |\n");

            printf("|                                                |\n");

            printf("|                  Please input:                 |\n");

            printf("|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\n");

            scanf("%d",&INDEX);

            for(i=0;i<m;i++){

                int position;

                position=max\_element(times+1,times+m+1)-times;

                maxtimes[i]=position;

                // printf("%d ",position);

                times[position]=-1;

            }

            // printf("\n");

            // for(i=0;i<m;i++){

            //     printf("%d ",maxtimes[i]);

            // }

            // system("Pause");

            // int RESULT[m];                             // 鐪熷€艰〃

            memset(RESULT,-1,sizeof(RESULT));

            clock\_t Start,End;

            Start=clock();

            b=DPLL(HEAD, RESULT,maxtimes);

                End=clock();

                printf("Runtime: %lf ms\n",(double)(End-Start)/CLOCKS\_PER\_SEC\*1000.0);

                printf("The solution has been saved in the file ""for.res""\n");

                // scanf("%s",filename);

                strcpy(name,"C:\\Users\\13226\\Desktop\\SAT\\for.res");

                // strcat(name,filename);

                // strcat(name,".res");

                // fp=fopen(name, "w");

                if(b) {

                    fprintf(fp,"s 1\nv \n");

                    for(i=0;i<m;i++){

                        if(RESULT[i]!=0){

                            fprintf(fp,"%6d ",i+1);

                        }

                        else {

                            fprintf(fp,"%6d ",-i-1);

                        }

                    }

                }

                else fprintf(fp,"s 0\n");

                fclose(fp);

                system("Pause");

            system("cls");

            find\_solution();

            for(i=0,k=0;i<9;i++){

                for(;k<9;k++){

                    sudoku\_begin[i][k]=sudoku\_dig[i][k];

                }

            }

            for(i=0,k=0;i<9;i++){

                for(;k<9;k++){

                    if(sudoku\_begin[i][k])continue;

                    for(c=(i+1)\*100+(k+1)\*10;c<(i+1)\*100+(k+1)\*10+9;c++){

                    if(RESULT[c])sudoku\_begin[i][k]=c-((i+1)\*100+(k+1)\*10-1);

                }

                }

            }

            // printf("\n\nInput 1 to show solution\nInput 2 to show the DPLL solution\nInput 0 to exit:");

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

            setColor(14,0);

            printf("|                 Choose a function              |\n");

            setColor(7,0);

            printf("|------------------------------------------------|\n");

            printf("|                  INDEX1:Show solution:         |\n");

            printf("|                  INDEX2:DPLL solution:         |\n");

            printf("|                  INDEX0:Exit:                  |\n");

            printf("|                                                |\n");

            printf("|                  Please input:                 |\n");

            printf("|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\n");

            scanf("%d",&b);

            while(b!=0){

                if(b==1){

                    print();

                }

                else if(b==2){

                    for(i=0;i<999;i++){

                        if(RESULT[i]!=0){

                        printf("%d ",i+1);

                    }

                    else {

                        printf("-%d ",i+1);

                        }

                    if((i+1)%5==0)printf("\n");

                        }

                }

                else printf("Illigal index!");

                system("Pause");

                system("cls");

                // getchar();

                getchar();

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

            setColor(14,0);

            printf("|                 Choose a function              |\n");

            setColor(7,0);

            printf("|------------------------------------------------|\n");

            printf("|                  INDEX1:Show solution:         |\n");

            printf("|                  INDEX2:DPLL solution:         |\n");

            printf("|                  INDEX0:Exit:                  |\n");

            printf("|                                                |\n");

            printf("|                  Please input:                 |\n");

            printf("|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\n");

                scanf("%d",&b);

            }

            printf("Bye bye~\n");

            system("Pause");

            return 0;

}

//dpll.cpp

#include "hellodpll.h"

LIST\* single(LIST\* HEAD)

{

    LIST\* POSITION = HEAD;

    while(POSITION)

    {

        if(POSITION->head && !POSITION->head->next)    return POSITION;

        POSITION = POSITION->NEXT;

    }

    return NULL;

}

LIST \*Copy(LIST \*L){

    LIST \*NEWHEAD;

    LIST \*POSITION;

    LIST \*COPY=L->NEXT;

    NEWHEAD=(LIST\*)malloc(sizeof(LIST));

    NEWHEAD->NEXT=NULL;

    NEWHEAD->head=NULL;

    POSITION=NEWHEAD;

    for(;COPY!=NULL;COPY=COPY->NEXT){

        POSITION->NEXT=(LIST\*)malloc(sizeof(LIST));

        POSITION=POSITION->NEXT;

        POSITION->NEXT=NULL;

        POSITION->length=COPY->length;

        node \*p;

        node \*copyp=COPY->head;

        if(copyp!=NULL){

            POSITION->head=(node\*)malloc(sizeof(node));

            p=POSITION->head;

            p->data=copyp->data;

            p->next=NULL;

            copyp=copyp->next;

            //POSITION->head=p;

            //2霉???猫掳?

            for(;copyp!=NULL;copyp=copyp->next){

                p->next=(node\*)malloc(sizeof(node));

                p=p->next;

                p->next=NULL;

                p->data=copyp->data;

            }

        //p->next=NULL;

        }

        else {

            POSITION->head=NULL;

        }

    }

    return NEWHEAD;

}

LIST\* add(LIST\* HEAD, int value)

{

    LIST\* POSITION;

    POSITION = (LIST \*)malloc(sizeof(LIST));

    POSITION->head = (node \*)malloc(sizeof(node));

    POSITION->head->data = value;

    POSITION->head->next = NULL;

    POSITION->NEXT = HEAD->NEXT;

    HEAD->NEXT = POSITION;

    return HEAD;

}

int choose(LIST\* HEAD,int index,int RESULT[],int maxtimes[])

{

    if(index==2){

        int i;

        for(i=0;i<m;i++){

            if(RESULT[maxtimes[i]-1]==-1){

                int p=maxtimes[i];

                RESULT[maxtimes[i]-1]=1;

                return p;

                // if(pos[maxtimes[i]]>=neg[maxtimes[i]])return p;

                // else return -p;

            }

        }

        int value = HEAD->NEXT->head->data;

        return value;

    }

    else if(index==1){

        int i;

        for(i=m-1;i>=0;i--){

            if(RESULT[maxtimes[i]-1]==-1){

                int p=maxtimes[i];

                RESULT[maxtimes[i]-1]=1;

                return p;

                // if(pos[maxtimes[i]]>=neg[maxtimes[i]])return p;

                // else return -p;

            }

        }

        int value = HEAD->NEXT->head->data;

        return value;

    }

    else {

        int value = HEAD->NEXT->head->data;

        return value;

    }

}

LIST\* deletelist(LIST\* HEAD)

{

    LIST\* POSITION = HEAD;

    node\* p;

    for(;HEAD->head;)

    {

        p = HEAD->head;

        HEAD->head = HEAD->head->next;

        free(p);

    }

    HEAD = HEAD->NEXT;

    free(POSITION);

    return HEAD;

}

LIST\* deletenode(LIST\* HEAD, int value)

{

    node \*p = HEAD->head, \*tmp;

        for(;HEAD->head;){

            p=HEAD->head;

            if(p->data!=value&&p->data!=-value)break;

            else {

                if(p->data==value)return NULL;

                else {

                    tmp=HEAD->head->next;

                    free(p);

                    HEAD->head=tmp;

                }

            }

    }

    p=HEAD->head;

    while(p)

    {

        if(p->next && p->next->data == -value)

        {

            tmp = p->next;

            p->next = p->next->next;

            free(tmp);

        }

        else if(p->next && p->next->data == value)  return NULL;

        p = p->next;

    }

    return HEAD;

}

int DELETE\_(LIST\* HEAD, int value)

{

    LIST \*POSITION = HEAD->NEXT, \*POSITION\_ = HEAD;

    for(;POSITION!=NULL;)

    {

        POSITION = deletenode(POSITION, value);

        if(POSITION == NULL)

        {

            POSITION = deletelist(POSITION\_->NEXT);

            POSITION\_->NEXT = POSITION;

            continue;

        }

        if(POSITION == NULL)   break;

        POSITION\_ = POSITION;

        POSITION = POSITION->NEXT;

    }

    return OK;

}

int empty(LIST\* HEAD)

{

    LIST\* POSITION = HEAD->NEXT;

    while(POSITION)

    {

        if(!POSITION->head) return 1;

        POSITION = POSITION->NEXT;

    }

    return 0;

}

int DPLL(LIST\* HEAD, int\* RESULT,int maxtimes[])

{

    int i,k,b,c,value;

    LIST\* POSITION = HEAD;

    LIST\* SINGLE = single(POSITION);

    for(;SINGLE!=NULL;)

    {

        if(SINGLE->head->data>0)RESULT[SINGLE->head->data-1]=1;

        else RESULT[-(SINGLE->head->data)-1]=0;

        value=SINGLE->head->data;

        DELETE\_(HEAD, value);

        if(HEAD->NEXT == NULL) return 1;

        else if(empty(HEAD))   return 0;

        POSITION = HEAD;

        SINGLE = single(POSITION);

    }

    value = choose(HEAD,INDEX,RESULT,maxtimes);

    if(DPLL(add(Copy(HEAD), value),RESULT,maxtimes))  return 1;

    return DPLL(add(HEAD, -value),RESULT,maxtimes);

}

int dig\_mark(int x,int y,int value){

    int i,k,b,c;

    line\_mark[x][value]=0;

    row\_mark[y][value]=0;

    unit\_mark[x/3][y/3][value]=0;

    return 0;

}

int re\_dig\_mark(int x,int y,int value){

    int i,k,b,c;

    line\_mark[x][value]=1;

    row\_mark[y][value]=1;

    unit\_mark[x/3][y/3][value]=1;

    return 0;

}

// int show(){

//     int i, j;

//     SetPos(0,16);

//     setColor(10,8);

//     printf("      ---------------------\n");

//      for (i = 0; i < 9; i++)

//      {

//           printf("      ");

//      for (j = 0; j < 9; j++)

//      {

//              if (j % 3 == 0) printf("|");

//             if(j==8) {

//                 if (sudoku\_dig[i][j]!=0)printf("%d|\n",sudoku\_dig[i][j]);

//                 else printf("\_|\n");

//                 continue;}

//              if (sudoku\_dig[i][j]!=0) printf("%d ",sudoku\_dig[i][j]);

//              else printf("\_ ");

//      }

//      if ((i + 1) % 3 == 0) printf("      ---------------------\n");

//      }

//     setColor(7,0);

//       return 0;

// }

int show(){

    int i, j;

    SetPos(0,15);

    setColor(2,7);

    printf("---------------------\n");

    for (i = 0; i < 9; i++)

    {

        for (j = 0; j < 9; j++)

        {

            if (j % 3 == 0) printf("|");

            if(j==8) {

                if (sudoku\_dig[i][j]!=0)printf("%d|\n",sudoku\_dig[i][j]);

                else printf("\_|\n");

                continue;}

            if (sudoku\_dig[i][j]!=0) printf("%d ",sudoku\_dig[i][j]);

            else {

                  setColor(4,7);

                  printf("\_ ");

                  setColor(2,7);

              }

        }

        if ((i + 1) % 3 == 0) printf("---------------------\n");

    }

    setColor(7,0);

    return 0;

}

int ass(int x,int y,int value,int index){

    int i,k,b,c;

    if(index==1){

        sudoku\_dig[x][y]=value;

        return 0;

    }

    else sudoku\_dig[x][y]=0;

    return 0;

}

void dfs(int x,int y) {

    if(SUM\_SOLUTION>1)return;

    if(sudoku\_dig[x][y]) {

        if(x!=8||y!=8) {

            dfs(x+(y+1)/9,(y+1)%9);

            return;

            }

        else {

            SUM\_SOLUTION++;

            if(SUM\_SOLUTION>1)return;

            //show();

            // printf("\n");

            return;

            }

    }

    for(int i=1;i<=9;i++) {

        if(SUM\_SOLUTION>1)return;

        if(line\_mark[x][i]||row\_mark[y][i]||unit\_mark[x/3][y/3][i]) continue;

        ass(x,y,i,1);

        if(x==8&&y==8) {

            SUM\_SOLUTION++;

            //show();

            // printf("\n");

            ass(x,y,i,-1);

            //break;

            }

        else dfs(x+(y+1)/9,(y+1)%9);

        ass(x,y,i,-1);

    }

}

int singlesolution(){

    SUM\_SOLUTION=0;

    dfs(0,0);

    // printf("%d\n",SUM\_SOLUTION);

    // if(SUM\_SOLUTION==1)printf("YES\n");

    // if(SUM\_SOLUTION==0)show();

    if(SUM\_SOLUTION==1)return 1;

    else return 0;

}

// void createStartinggrid(int num) {

//     int i,k,b,j,m,flag=0;

//     int c[num][2];

//     for(i=0;i<num;i++){

//         for(k=0;k<2;k++){

//             c[i][k]=0;

//         }

//     }

//     srand((unsigned)time(NULL));

//     for( i = 0; i < 9 ; i ++)

//         for( j = 0; j < 9 ; j++)

//             sudoku\_dig[i][j] = sudoku\_begin[i][j];

//     for( i = 0; i < num ; i++) {

//         j = rand()%9;

//         k = rand()%9;

//         flag = 0;

//         for(m = 0; m < i ; m++)

//             if( j == c[m][0] && k == c[m][1]){

//                 // i--;

//                 // continue;

//                 flag=1;

//             }

//             if(flag==1){

//                 i--;

//                 continue;

//             }

//             if(flag==0)

//             {

//                 b=sudoku\_begin[j][k];

//                 dig\_mark(j,k,b);

//                 sudoku\_dig[j][k]=0;

//                 if(!singlesolution()){

//                     i--;

//                     sudoku\_dig[j][k]=b;

//                     re\_dig\_mark(j,k,b);

//                     continue;

//                 }

//                 else{

//                     c[i][0] = j;

//                     c[i][1] = k;

//                 }

//             }

//     }

//     // show();

// }

void createStartinggrid(int num) {

    int i,k,b,j,m,flag=0;

    int c[10][10];

    for(i=0;i<10;i++){

        for(k=0;k<10;k++){

            c[i][k]=0;

        }

    }

    srand((unsigned)time(NULL));

    for( i = 0; i < 9 ; i ++)

        for( j = 0; j < 9 ; j++)

            sudoku\_dig[i][j] = sudoku\_begin[i][j];

    for( i = 0; i < num ; i++) {

        j = rand()%9;

        k = rand()%9;

            if(c[j][k]==1){

                i--;

                continue;

            }

            else if(num<40)

            {

                b=sudoku\_begin[j][k];

                dig\_mark(j,k,b);

                sudoku\_dig[j][k]=0;

                c[j][k]=1;

                if(!singlesolution()){

                    i--;

                    sudoku\_dig[j][k]=b;

                    re\_dig\_mark(j,k,b);

                    continue;

                }

                else{

                    c[j][k]=1;

                }

            }

            else {

                b=sudoku\_begin[j][k];

                dig\_mark(j,k,b);

                sudoku\_dig[j][k]=0;

                c[j][k]=1;

            }

    }

    // show();

}

int Digit(int a[][9], int x, int y) {

    int i,k,b,c,j,flag;

    int check[10];

    for(i=0;i<10;i++){

        check[i]=1;

    }

    if (x < 9 && y < 9) {

        for(i = 0 ; i < x ; i++)

            check[a[i][y]] = 0;

        for(i = 0 ; i < y ; i++)

            check[a[x][i]] = 0;

        for(i = x/3\*3 ; i <= x; i++) {

            if(i == x)

                for(j = y/3\*3 ; j < y; j++)

                    check[a[i][j]] = 0;

            else

                for(j = y/3\*3 ; j < y/3\*3 + 3; j++)

                    check[a[i][j]] = 0;

        }

        flag=0;

        for(k = 1; k <= 9 && flag == 0 ; k++){

            // for(i=0;i<10;i++)printf("%d ",check[i]);

            // printf("\n");

            if(check[k]==1){

                flag = 1;

                a[x][y] = k;

                if(y == 9-1 && x != 9-1){

                    if(Digit(a,x+1,0) == 1) return 1;

                    else flag = 0;

                }

                else if(y != 9-1){

                    if(Digit(a,x,y+1) == 1) return 1;

                    else flag = 0;

                }

            }

        }

        if( flag == 0 ) {

            a[x][y] = 0;

            return 0;

        }

    }

    return 1;

}

// int Digit(int a[][9], int i, int j) {//閫掑綊濉?鍏呮暟鐙?鍏冪礌

//     if (i < 9 && j < 9) {

//         int x,y,k;

//         int check[9+1]={1};//鐢ㄤ簬鎺掗櫎宸茬粡浣跨敤杩囩殑鐨勬暟瀛?

//         for(x = 0 ; x < i ; x++)

//             check[a[x][j]] = 0;//鍒楀凡浣跨敤鐨勬暟瀛楃疆涓篧RONG

//         for(x = 0 ; x < j ; x++)

//             check[a[i][x]] = 0;//琛屼娇鐢ㄨ繃鐨勬暟瀛楃疆涓篧RONG

//         for(x = i/3\*3 ; x <= i; x++) {

//             if(x == i)

//                 for(y = j/3\*3 ; y < j; y++)

//                     check[a[x][y]] = 0;

//             else

//                 for(y = j/3\*3 ; y < j/3\*3 + 3; y++)

//                     check[a[x][y]] = 0;

//         }

//         int flag = 0;

//         for(k = 1; k <= 9 && flag == 0 ; k++){//浠巆heck鏁扮粍涓?鏌ユ壘瀹夊叏鐨勬暟瀛?

//             if(check[k] == 1){

//                 flag = 1;

//                 a[i][j] = k;

//                 if(j == 9-1 && i != 9-1){

//                     if(Digit(a,i+1,0) == 1) return 1;

//                     else flag = 0;

//                 }

//                 else if(j != 9-1){

//                     if(Digit(a,i,j+1) == 1) return 1;

//                     else flag = 0;

//                 }

//             }

//         }

//         if( flag == 0 ) {

//             a[i][j] = 0;

//             return 0;

//         }

//     }

//     return 1;

// }

void randomFirstRow(int a[], int n) {//闅忔満鐢熸垚绗?涓€琛?

    int i,k,b,c,j;

    srand((unsigned)time(NULL));

    for( i = 0 ; i < n ; i++){

        a[i] = rand()%9 + 1;

        j = 0 ;

        while(j < i){

            if(a[i] == a[j]){

                a[i] = rand()%9 + 1;

                j = 0;

            }

            else j++;

        }

    }

}

void createSudoku(int a[][9]){

    randomFirstRow(a[0],9);

    Digit(a,1,0);

}

// void createStartinggrid(const int a[][9], int b[][9], int num) {//闅忔満鐢熸垚鍒濈洏

//     int i,k,b,c,j,m,flag=0;

//     int c[num][2];

//     srand((unsigned)time(NULL));

//     for( i = 0; i < 9 ; i ++)

//         for( j = 0; j < 9 ; j++)

//             b[i][j] = a[i][j];

//     for( i = 0; i < num ; i++) {

//         j = rand()%9;

//         k = rand()%9;

//         flag = 0;

//         for(m = 0; m < i ; m++)

//             if( j == c[m][0] && k == c[m][1]){

//                 i--;

//                 continue;

//             }

//             else if(singlesolution()){

//                 i--;

//                 continue;

//             }

//             else{

//             b[j][k] = 0;

//             c[i][0] = j;

//             c[i][1] = k;

//         }

//         else

//             i--;

//     }

// }

//

void print(){//鎵撳嵃鏁扮嫭鏁扮粍

    int i, j;

    SetPos(0,15);

    setColor(2,7);

    printf("---------------------\n");

    for (i = 0; i < 9; i++)

    {

        for (j = 0; j < 9; j++)

        {

            if (j % 3 == 0) printf("|");

            if(j==8) {

                if (sudoku\_begin[i][j]!=0)printf("%d|\n",sudoku\_begin[i][j]);

                else printf("\_|\n");

                continue;}

            if (sudoku\_begin[i][j]!=0) printf("%d ",sudoku\_begin[i][j]);

            else {

                  setColor(4,7);

                  printf("\_ ");

                  setColor(2,7);

              }

        }

        if ((i + 1) % 3 == 0) printf("---------------------\n");

    }

    setColor(7,0);

}

int SAT\_FUNCTION(){

        system("cls");

            setColor(5,0);

            printf("--------------------Function <SAT>---------------------\n\n");

            setColor(7,0);

            if(CNF\_TRANSLATER()==-1)return 0;

            // char c,tmp[10480],name[100],filename[100];

            // printf("                Input filename:\n");

            // scanf("%s",filename);

            // strcpy(name,"C:\\Users\\13226\\Desktop\\SAT\\");

            // strcat(name,filename);

            // strcat(name,".cnf");

            // fp=fopen(name, "r+");

            // int i,k,b;

            // while(c=getc(fp)){

            //     if(c=='c'){

            //         fgets(tmp,10470,fp);

            //     }

            //     else if(c=='p'){

            //         break;

            //     }

            // }

            // for(;;){

            //     c=getc(fp);

            //     if(c=='f')break;

            // }

            // getc(fp);

            // fscanf(fp,"%d %d",&m,&n);

            // int times[m+1],maxtimes[m+1],pos[m+1],neg[m+1];

            // memset(times,0,sizeof(times));

            // memset(maxtimes,0,sizeof(maxtimes));

            // memset(pos,0,sizeof(pos));

            // memset(neg,0,sizeof(neg));

            // Start1=clock();

            // node \*p;

            // LIST \*HEAD,\*POSITION;

            // HEAD=(LIST\*)malloc(sizeof(LIST));

            // HEAD->NEXT=NULL;

            // POSITION=HEAD;

            // POSITION->NEXT=NULL;

            // POSITION->head=NULL;

            // for(i=1;i<n;i++){

            //     POSITION->NEXT=(LIST\*)malloc(sizeof(LIST));

            //     POSITION=POSITION->NEXT;

            //     POSITION->NEXT=NULL;

            //     POSITION->head=(node\*)malloc(sizeof(node));

            //     POSITION->head->next=NULL;

            //     p=POSITION->head;

            //     fscanf(fp,"%d ",&b);

            //     times[abs(b)]++;

            //     // if(b>0)pos[b]++;

            //     // else neg[-b]++;

            //     p->data=b;

            //     p->next=NULL;

            //     for(k=1;;k++){

            //         fscanf(fp,"%d ",&b);

            //         if(b==0)break;

            //         times[abs(b)]++;

            //         // if(b>0)pos[b]++;

            //         // else neg[-b]++;

            //         p->next=(node\*)malloc(sizeof(node));

            //         p=p->next;

            //         p->data=b;

            //         p->next=NULL;

            //     }

            //     POSITION->length=k;

            // }

            // fclose(fp);

            // // for(POSITION=HEAD;POSITION!=NULL;POSITION=POSITION->NEXT){

            // //     for(p=POSITION->head;p!=NULL;p=p->next){

            // //         printf("%d ",p->data);

            // //     }

            // //     printf("haha\n");

            // // }

            // // for(i=1;i<=m;i++){

            // //     printf("%d %d %d\n",times[i],pos[i],neg[i]);

            // // }

            // // printf("\n");

            // End1=clock();

            // system("cls");

            // printf("Read-in\_time: %lf ms\n\n",(double)(End1-Start1)/CLOCKS\_PER\_SEC\*1000.0);

            int sh;

            printf("        Input 1 to show the data of this file\n        Input 0 to skip\n");

            scanf("%d",&sh);

            LIST \*POSITION;

            node \*p;

            int b;

            char name[100];

            if(sh==1){

                POSITION=HEAD->NEXT;

                for(;POSITION!=NULL;POSITION=POSITION->NEXT){

                    p=POSITION->head;

                    for(;p!=NULL;p=p->next){

                        printf("%d ",p->data);

                    }

                    printf("  length=%d\n",POSITION->length);

                }

            }

            system("Pause");

            system("cls");

            printf("NOW LET'S SOLVE THIS SAT PROBLEM!\n\n");

            // printf("-------------------INDEX1:Normal  INDEX2:Optimized  INDEX3:RANDOM--------------------\n");

            // printf("                                    Input INDEX:");

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

            printf("|     Please choose your selection strategy      |\n");

            printf("|------------------------------------------------|\n");

            printf("|                  INDEX1:Normal:                |\n");

            printf("|                  INDEX2:Optimized:             |\n");

            printf("|                  INDEX3:RANDOM:                |\n");

            printf("|                                                |\n");

            printf("|                  Please input:                 |\n");

            printf("|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\n");

            scanf("%d",&INDEX);

            int i;

            for(i=0;i<m;i++){

                int position;

                position=max\_element(times+1,times+m+1)-times;

                maxtimes[i]=position;

                // printf("%d ",position);

                times[position]=-1;

            }

            // printf("\n");

            // for(i=0;i<m;i++){

            //     printf("%d ",maxtimes[i]);

            // }

            // system("Pause");

            // int RESULT[m];                             // 鐪熷€艰〃

            memset(RESULT,-1,sizeof(RESULT));

            clock\_t Start,End;

            Start=clock();

            b=DPLL(HEAD, RESULT,maxtimes);

            if(b)for(i=0;i<m;i++){

                if(RESULT[i]!=0){

                  printf("%d ",i+1);

               }

              else {

                  printf("-%d ",i+1);

                }

               if((i+1)%5==0)printf("\n");

                  }

            else printf("error");

                printf("\n");

                End=clock();

                printf("Runtime: %lf ms\n",(double)(End-Start)/CLOCKS\_PER\_SEC\*1000.0);

                printf("The solution has been saved in the file ""for.res""\n");

                // scanf("%s",filename);

                strcpy(name,"C:\\Users\\13226\\Desktop\\SAT\\for.res");

                // strcat(name,filename);

                // strcat(name,".res");

                fp=fopen(name, "w");

                if(b) {

                    fprintf(fp,"s 1\nv ");

                    for(i=0;i<m;i++){

                        if(RESULT[i]!=0){

                            fprintf(fp,"%6d ",i+1);

                        }

                        else {

                            fprintf(fp,"%6d ",-i-1);

                        }

                    }

                }

                else fprintf(fp,"s 0\n");

                fprintf(fp,"\nt %6d ",(End-Start)/CLOCKS\_PER\_SEC\*1000);

                fclose(fp);

                system("Pause");

        return 0;

}

//cnf\_translater.cpp

#include "hellodpll.h"

int ToCnf(int a[][9],int holes) {

    FILE \*fp;

    char c,tmp[10480],name[100];

    strcpy(name,"C:\\Users\\13226\\Desktop\\SAT\\sudoku.cnf");

    fp=fopen(name, "w+");

    int i,k,b,m,n;

    fprintf(fp,"p cnf 999 %d\n",8910-holes);

    for (int x = 0; x < 9; ++x) {

        for (int y = 0; y < 9; ++y)

            if(a[x][y] != 0)

                fprintf(fp,"%d 0\n",(x+1)\*100 + (y+1)\*10 + a[x][y]);

    }

    //entry

    for (int x = 1; x <= 9; ++x) {

        for (int y = 1; y <= 9; ++y) {

            for (int z = 1; z <= 9; ++z)

                fprintf(fp,"%d ",x \* 100 + y \* 10 + z);

                fprintf(fp,"%d \n",0);

        }

    }

    //row

    for (int y = 1; y <= 9; ++y) {

        for (int z = 1; z <= 9; ++z)

            for (int x = 1; x <= 8; ++x)

                for (int i = x+1; i <= 9; ++i)

                    fprintf(fp,"%d %d 0\n",0 - (x\*100 + y\*10 + z),0 - (i\*100 + y\*10 + z));

    }

    //column

    for (int x = 1; x <= 9; ++x) {

        for (int z = 1; z <=9 ; ++z)

            for (int y = 1; y <= 8; ++y)

                for (int i = y+1; i <= 9; ++i)

                    fprintf(fp,"%d %d 0\n",0-(x\*100 + y\*10 + z),0-(x\*100 + i\*10 + z));

    }

    //3\*3 sub-grids

    for (int z = 1; z <= 9 ; ++z) {

        for (int i = 0; i <=2 ; ++i)

            for (int j = 0; j <=2 ; ++j)

                for (int x = 1; x <= 3 ; ++x)

                    for (int y = 1; y <= 3; ++y)

                        for (int k = y+1; k <= 3; ++k)

                            fprintf(fp,"%d %d 0\n",0 - ((3\*i+x)\*100 + (3\*j+y)\*10 + z),0-((3\*i+x)\*100 + (3\*j+k)\*10 + z));

    }

    for (int z = 1; z <= 9; z++) {

        for (int i = 0; i <= 2; i++)

            for (int j = 0; j <= 2; j++)

                for (int x = 1; x <= 3; x++)

                    for (int y = 1; y <= 3; y++)

                        for (int k = x + 1; k <= 3; k++)

                            for (int l = 1; l <= 3; l++)

                                fprintf(fp,"%d %d 0\n", 0 - ((3\*i+x)\*100 + (3\*j+y)\*10 + z),0 - ((3\*i+k)\*100 + (3\*j+l)\*10 + z));

    }

    fclose(fp);

    return 0;

}

int CNF\_TRANSLATER\_SUDOKU(){

            char c,tmp[10480],name[100],filename[100];

            strcpy(name,"C:\\Users\\13226\\Desktop\\SAT\\sudoku.cnf");

            fp=fopen(name, "r+");

            int i,k,b;

            while(c=getc(fp)){

                if(c=='c'){

                    fgets(tmp,10470,fp);

                }

                else if(c=='p'){

                    break;

                }

                else {

                    printf("ERROR FORMAT!\n");

                    return 0;

                }

            }

            for(;;){

                c=getc(fp);

                if(c=='f')break;

            }

            getc(fp);

            fscanf(fp,"%d %d",&m,&n);

            // int times[m+1],maxtimes[m+1],pos[m+1],neg[m+1];

            memset(times,0,sizeof(times));

            memset(maxtimes,0,sizeof(maxtimes));

            // memset(pos,0,sizeof(pos));

            // memset(neg,0,sizeof(neg));

            Start1=clock();

            node \*p;

            LIST \*POSITION;

            HEAD=(LIST\*)malloc(sizeof(LIST));

            HEAD->NEXT=NULL;

            POSITION=HEAD;

            POSITION->NEXT=NULL;

            POSITION->head=NULL;

            for(i=1;i<n;i++){

                POSITION->NEXT=(LIST\*)malloc(sizeof(LIST));

                POSITION=POSITION->NEXT;

                POSITION->NEXT=NULL;

                POSITION->head=(node\*)malloc(sizeof(node));

                POSITION->head->next=NULL;

                p=POSITION->head;

                fscanf(fp,"%d ",&b);

                times[abs(b)]++;

                // if(b>0)pos[b]++;

                // else neg[-b]++;

                p->data=b;

                p->next=NULL;

                for(k=1;;k++){

                    fscanf(fp,"%d ",&b);

                    if(b==0)break;

                    times[abs(b)]++;

                    // if(b>0)pos[b]++;

                    // else neg[-b]++;

                    p->next=(node\*)malloc(sizeof(node));

                    p=p->next;

                    p->data=b;

                    p->next=NULL;

                }

                POSITION->length=k;

            }

            fclose(fp);

            // for(POSITION=HEAD;POSITION!=NULL;POSITION=POSITION->NEXT){

            //     for(p=POSITION->head;p!=NULL;p=p->next){

            //         printf("%d ",p->data);

            //     }

            //     printf("haha\n");

            // }

            // for(i=1;i<=m;i++){

            //     printf("%d %d %d\n",times[i],pos[i],neg[i]);

            // }

            // printf("\n");

            End1=clock();

            system("cls");

            printf("Read-in\_time of sudoku: %lf ms\n\n",(double)(End1-Start1)/CLOCKS\_PER\_SEC\*1000.0);

            return 0;

}

int CNF\_TRANSLATER(){

            char c,tmp[10480],name[100],filename[100];

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

            printf("|        Input filename you want to read\_in      |\n");

            printf("|------------------------------------------------|\n");

            printf("|                  Please input:                 |\n");

            printf("|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\n");

            scanf("%s",filename);

            strcpy(name,"C:\\Users\\13226\\Desktop\\SAT\\");

            strcat(name,filename);

            strcat(name,".cnf");

            fp=fopen(name, "r+");

            int i,k,b;

            // while(c=getc(fp)){

            //     if(c=='c'){

            //         fgets(tmp,10470,fp);

            //     }

            //     else if(c=='p'){

            //         break;

            //     }

            // }

            while(c=getc(fp)){

                if(c=='c'){

                    fgets(tmp,10470,fp);

                }

                else if(c=='p'){

                    break;

                }

                else {

                    printf("ERROR FORMAT!\n");

                    system("Pause");

                    return -1;

                }

            }

            for(;;){

                c=getc(fp);

                if(c=='f')break;

            }

            getc(fp);

            fscanf(fp,"%d %d",&m,&n);

            // int times[m+1],maxtimes[m+1],pos[m+1],neg[m+1];

            memset(times,0,sizeof(times));

            memset(maxtimes,0,sizeof(maxtimes));

            // memset(pos,0,sizeof(pos));

            // memset(neg,0,sizeof(neg));

            Start1=clock();

            node \*p;

            LIST \*POSITION;

            HEAD=(LIST\*)malloc(sizeof(LIST));

            HEAD->NEXT=NULL;

            POSITION=HEAD;

            POSITION->NEXT=NULL;

            POSITION->head=NULL;

            for(i=1;i<n;i++){

                POSITION->NEXT=(LIST\*)malloc(sizeof(LIST));

                POSITION=POSITION->NEXT;

                POSITION->NEXT=NULL;

                POSITION->head=(node\*)malloc(sizeof(node));

                POSITION->head->next=NULL;

                p=POSITION->head;

                fscanf(fp,"%d ",&b);

                times[abs(b)]++;

                // if(b>0)pos[b]++;

                // else neg[-b]++;

                p->data=b;

                p->next=NULL;

                for(k=1;;k++){

                    fscanf(fp,"%d ",&b);

                    if(b==0)break;

                    times[abs(b)]++;

                    // if(b>0)pos[b]++;

                    // else neg[-b]++;

                    p->next=(node\*)malloc(sizeof(node));

                    p=p->next;

                    p->data=b;

                    p->next=NULL;

                }

                POSITION->length=k;

            }

            fclose(fp);

            // for(POSITION=HEAD;POSITION!=NULL;POSITION=POSITION->NEXT){

            //     for(p=POSITION->head;p!=NULL;p=p->next){

            //         printf("%d ",p->data);

            //     }

            //     printf("haha\n");

            // }

            // for(i=1;i<=m;i++){

            //     printf("%d %d %d\n",times[i],pos[i],neg[i]);

            // }

            // printf("\n");

            End1=clock();

            system("cls");

            printf("Read-in\_time: %lf ms\n\n",(double)(End1-Start1)/CLOCKS\_PER\_SEC\*1000.0);

            return 0;

}