

**课程设计报告**

**题目：基于SAT的蜂窝数独游戏求解程序**

**课程名称：程序设计综合课程设计**

**专业班级： CS2206**

**学 号： U202215523**

**姓 名： 丁丁丁**

**指导教师： 纪俊文**

**报告日期： 2023.10.1**

**计算机科学与技术学院**

**目录**

[**任 务 书** 1](#_Toc2235596)

[**1.引 言 2**](#_Toc2235599)

1.1.SAT问题背景 [2](#_Toc2235613)

1.2.SAT问题研究意义 [2](#_Toc2235613)

[**2.系统需求分析与总体设计 4**](#_Toc2235603)

2.1.系统需求分析 [4](#_Toc2235613)

2.2.系统总体设计 [4](#_Toc2235613)

[**3.系统详细设计 6**](#_Toc2235603)

3.1.数据结构 [6](#_Toc2235613)

3.2.DPLL算法思想 [7](#_Toc2235613)

3.3.重要函数算法设计 [8](#_Toc2235613)

3.4.DPLL优化 [10](#_Toc2235613)

**4.**[**系统实现与测试 11**](#_Toc2235607)

4.1.系统实现 [11](#_Toc2235613)

4.2.系统测试 [11](#_Toc2235613)

4.3.SAT问题测试 [12](#_Toc2235613)

4.4.蜂窝数独游戏测试 [20](#_Toc2235613)

**5.**[**感悟与总结 22**](#_Toc2235610)

[**参考文献 23**](#_Toc2235612)

[**附录 24**](#_Toc2235614)

**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%)

**1引言**

**1.1 SAT问题背景**

SAT问题又称命题逻辑公式的可满足性问题（satisfiability problem），是判断对合取范式形式给出的命题逻辑公式是否存在一个真值指派使得该逻辑公式为真。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。对于SAT问题的研究从没有停止过，在1997年和2003年，H.Kautz与B.Selman两次列举出SAT搜索面临的挑战性问题，并于2011年和2007年，两度对当时的SAT问题研究现状进行了全面的综述。黄文奇提出的Solar算法在北京第三届SAT问题快速算法比赛中获得第一名。对SAT问题的求解主要有完备算法和不完备算法两大类。不完备算法主要是局部搜索算法，这种算法不能保证一定找到解，但是求解速度快，对于某些SAT问题的求解，局部搜索算法要比很多完备算法更有效。完备算法出现的时间更早，优点是可以正确判断SAT问题的可满足性，在算例无解的情况下可以给出完备的证明。对于求解SAT问题的优化算法主要有启发式算法、冲突子句学习算法、双文字监视法等。

**1.2 SAT问题研究意义**

SAT问题是第一个被证明的NP完全问题，而NP完全问题由于其极大的理论价值和困难程度，破解后将会在许多领域得到广泛应用，从而在计算复杂性理论中具有非常重要的地位。由于所有的NP完全问题都能够在多项式时间内进行转换，那么如果SAT问题能够得到高效解决，所有的NP完全问题都能够在多项式时间内得到解决。对SAT问题的求解，可用于解决计算机和人工智能领域内的CSP问题（约束满足问题）、语义信息的处理和逻辑编程等问题，也可用于解决计算机辅助设计领域中的任务规划与设计、三维物体识别等问题。SAT问题的应用领域非常广泛，还能用于解决数学研究和应用领域中的旅行商问题和逻辑算数问题。许多实际问题，例如数据库检索、积木世界规划、超大规模集成电路设计、人工智能等都可以转换成SAT问题进而进行求解。可见对SAT问题求解的研究，具有重大意义。

**2系统需求分析与总体设计**

**2.1系统需求分析**

该系统要求能实现对SAT算例cnf文件的读取，将执行结果输出并且保存到文件中，解析并遍历读取的cnf文件子句，基于DPLL算法求解SAT问题，记录DPLL执行时间，并集成一个SAT应用：蜂窝数独游戏（hanidoku）。

**2.2系统总体设计**

如图2-1为该系统设计的流程图。程序启动后，首先选择处理基础的SAT问题或者进行蜂窝数独游戏。若选择处理SAT问题，则先读取SAT算例（cnf文件），随即进行DPLL，求解出结果，展示并且保存到一个同名res文件中，最后程序结束。若选择进行蜂窝数独游戏，则首先初始化一个游戏格局，再进行蜂窝数独条规约，将该问题转化为SAT问题，随即进行DPLL求解，展示结果，最后程序结束。此外在选择进行蜂窝数独游戏时还可以输出蜂窝数独规约条件的cnf文件。



图2-1程序流程图

程序模块化

设计程序要求模块化，程序源代码进行模块化组织。主要模块包括如下：

1）主控、交互与显示模块（display）

2）CNF解析模块（cnfparser）

3）核心DPLL模块( solver)

4）蜂窝数独模块，包括游戏格局生成、归约、求解(hanidoku)

**3系统详细设计**

**3.1数据结构**

*typedef struct Node {*

*int data;*

*int negated; //1正 0负+*

*struct Node\* next;*

*}Node;*

*typedef struct clauselist {*

*int w; //子句变元数量。-1代表删除了*

*Node\* p;*

*clauselist\* next;*

*}ClauseList;*

*#define Unknow -1*

*#define False 0*

*#define True 1*

*#define None 2*

*#define MaxNumVar 1000*

*extern int Varjl[MaxNumVar + 1];*

以上为该系统中用于存放cnf文件各个子句的结构。Node用于存放变元，每个Node中，data记录变元名称，negated记录变元正负，next指向下一个变元。每个子句由一条Node链表存储。多个子句通过Clauselist构成多链表。Clauselist中，w代表该子句变元数量，若w的值为-1，则代表该子句删除。Node\* p指向子句第一个变元，next指向下一个子句。如图3-1为存储结构示意图。

此外，定义一个数组Varjl[]用于记录变元，数组下标对应变元，数组初始值都为Unknow（-1），后续可能赋值为False（0）、True（1）、None（2）。



图3-1 存储结构示意图

**3.2 DPLL算法思想**

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，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

基于单子句传播与分裂策略的DPLL算法可以描述为一个如后所示的递归过程**DPLL( *S* )**, DPLL算法也可用非递归实现。

**DPLL( *S*) :**

/\* *S*为公式对应的子句集。若其满足，返回TURE；否则返回FALSE. \*/

**{**

**while(*S*中存在单子句) {**//单子句传播

**在*S*中选一个单子句*L*；**

**依据单子句规则，利用*L*化简*S*；**

**if *S* = Φ return(TRUE);**

**else if (*S*中有空子句 ) return（FALSE）；**

**}**//while

**基于某种策略选取变元*v*； //策略对DPLL性能影响很大**

**if DPLL（*S* ∪*v* ）return(TURE); //在第一分支中搜索**

**return DPLL(*S* ∪¬*v*);//回溯到对*v*执行分支策略的初态进入另一分支**

**}**

**3.3 重要函数算法设计**

（1）函数名称：int Readcnf(char\* filename, ClauseList\*\* s, int\* var, int\* nc);

函数功能：读取cnf文件中的内容，获取变元个数，子句个数。

设计思路：找到cnf文件中“p cnf ……”这一行，读取变元个数和子句个数到var，nc中，再读取每一行的数据，存入多链表中。

（2）函数名称：void Propagation(ClauseList\*\* S, Node x);

函数功能： 对多链表S操作，进行单子句传播。

设计思路： 从头遍历，找到子句中与x变元相同的目标结点，若正负相同，则删除目标结点所在的整个子句，若正负相反，删除目标结点。

（3）函数名称：ClauseList\* Copy(ClauseList\* S);

函数功能： 复制多链表S,生成一个副本。

（4）函数名称：void ClearNodeList(Node\* node);

函数功能： 清除一整条子句。

设计思路： 单链表删除。

（5）函数名称：void ClearList(ClauseList\*\* S);

函数功能： 清除整个多链表，用于释放副本空间。

设计思路： 利用ClearList函数，删除整个多链表。

（6）函数名称：void Add(ClauseList\*\* S, Node x);

函数功能： 在多链表末端添加一个单子句，该单子句含有唯一结点x。

（7）函数名称：int Save(int a, char\* filename, double time);

函数功能： 将DPLL结果以及用时保存到filename同名的res文件中。

（8）函数名称：int IsEmpty(ClauseList\* S);

函数功能： 判断有没有空子句。

（9）函数名称：ClauseList\* IsSingleClause(ClauseList\* S);

函数功能： 在多链表中找一个单子句。

设计思路： 遍历，找到w=1的子句，返回ClauseList\*。

（10）函数名称：void PrintfCnf(ClauseList\* S);

函数功能： 打印读取的cnf文件内容。

（11）函数名称：int Sat();

函数功能： SAT求解模块函数，显示菜单界面与用户交互

设计思路： 显示一个菜单，包含完整详细的提示，结合包括DPLL在内的一众函数，实现cnf读取遍历，DPLL求解，文件保存，结果展示的功能。

（12）函数名称：int ToCnf(int a[][10]);

函数功能： 将蜂窝数独游戏的约束规约保存到cnf文件中。

（13）函数名称：int First(char\* filename);

函数功能： 初始化蜂窝数独格局。从文件读入.

（13）函数名称：void Tu();

函数功能： 显示蜂窝数独界面。

（14）函数名称：void Sudoku();

函数功能：蜂窝数独游戏模块函数，显示菜单界面与用户交互。

设计思路： 显示一个菜单，包含完整详细的提示，结合包括DPLL在内的一众函数，实现cnf读取遍历，蜂窝数独格局初始化，游戏结果求解展示，结果展示的功能。

**3.4 DPLL优化**

本次优化针对变元的选取策略。优化前，选取变元策略为：直接取第一个子句链表的第一个变元v，设为True，通过void Add(ClauseList\*\* S, Node x)函数添加单子句（v）到多链表结尾，copy后再次对副本DPLL，若不成立，释放副本空间，设v为False。该选取策略非常简单无脑，有极大优化空间。

本次实验优化后选取策略为：找到长度最短的子句，选取其第一个变元v。这种选取策略可以尽快产生更多单子句，优化效果详见4.3。

**4系统实现与测试**

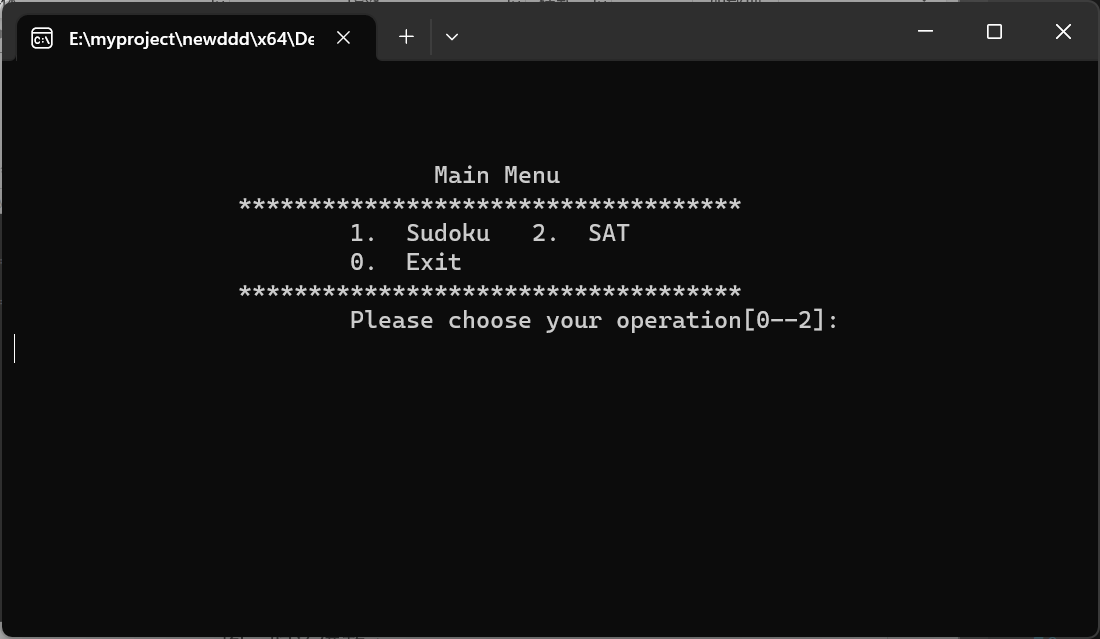
**4.1系统实现**

整个工程包含main.cpp、Sat.h、hanshu.cpp、Sudoku.cpp四个文件。main.cpp用于程序主界面的显示和主要函数的调用；Sat.h声明本程序的函数以及一些变量的声明、数据元素类型的定义；hanshu.cpp是求解SAT问题有关函数的实现；Sudoku.cpp是进行蜂窝数独游戏的函数实现。

详细代码见附录。

**4.2系统测试**

程序启动，如图4-1为菜单。1进入蜂窝数独游戏，2求解SAT问题。



4-1 菜单

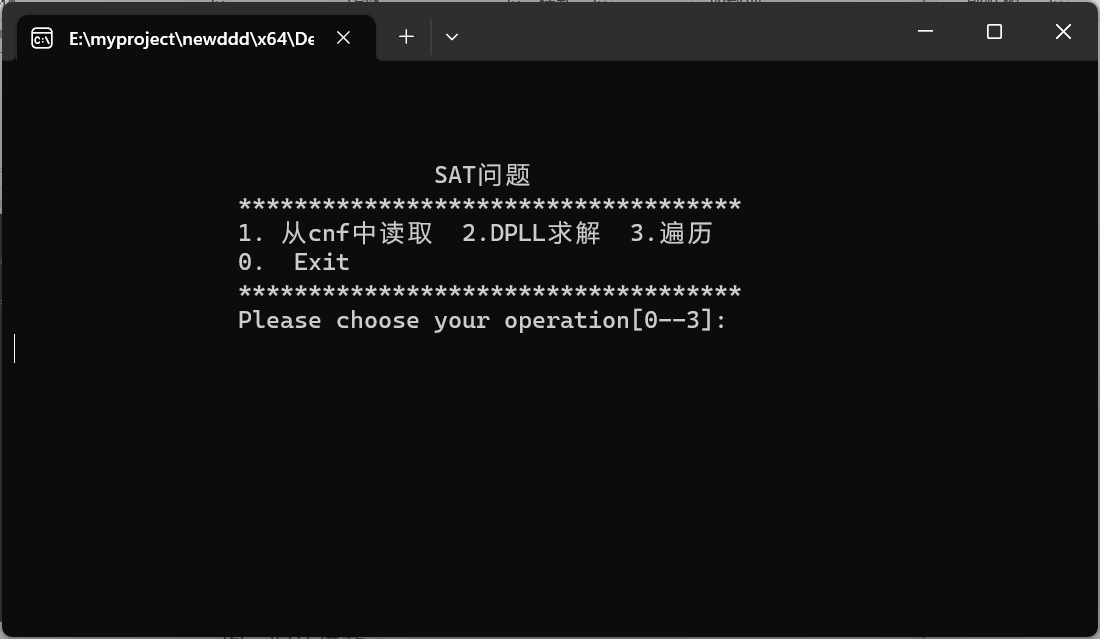


图4-2 求解SAT问题菜单

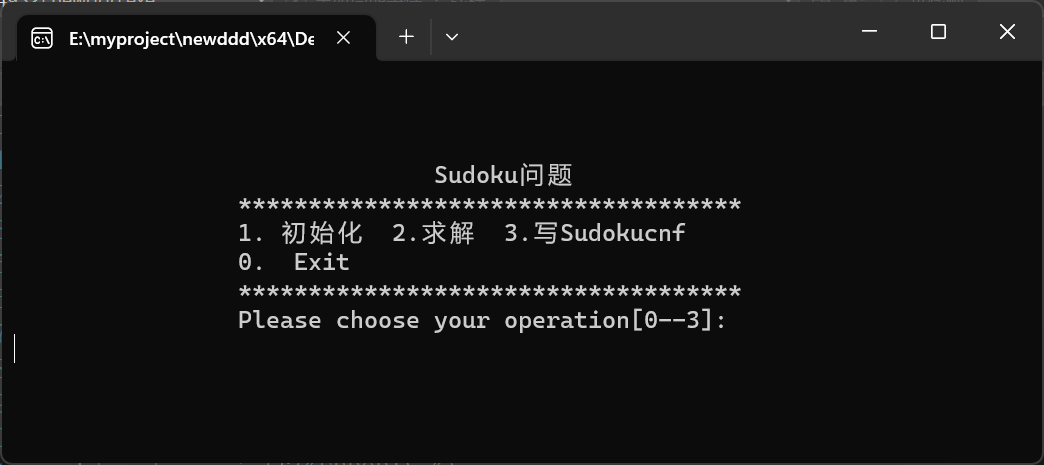


图4-3 蜂窝数独游戏菜单

**4.3 SAT问题测试**

首先测试对cnf文件的读取和遍历展示。如图4-4，图4-5。



图4-4 从cnf文件中读取

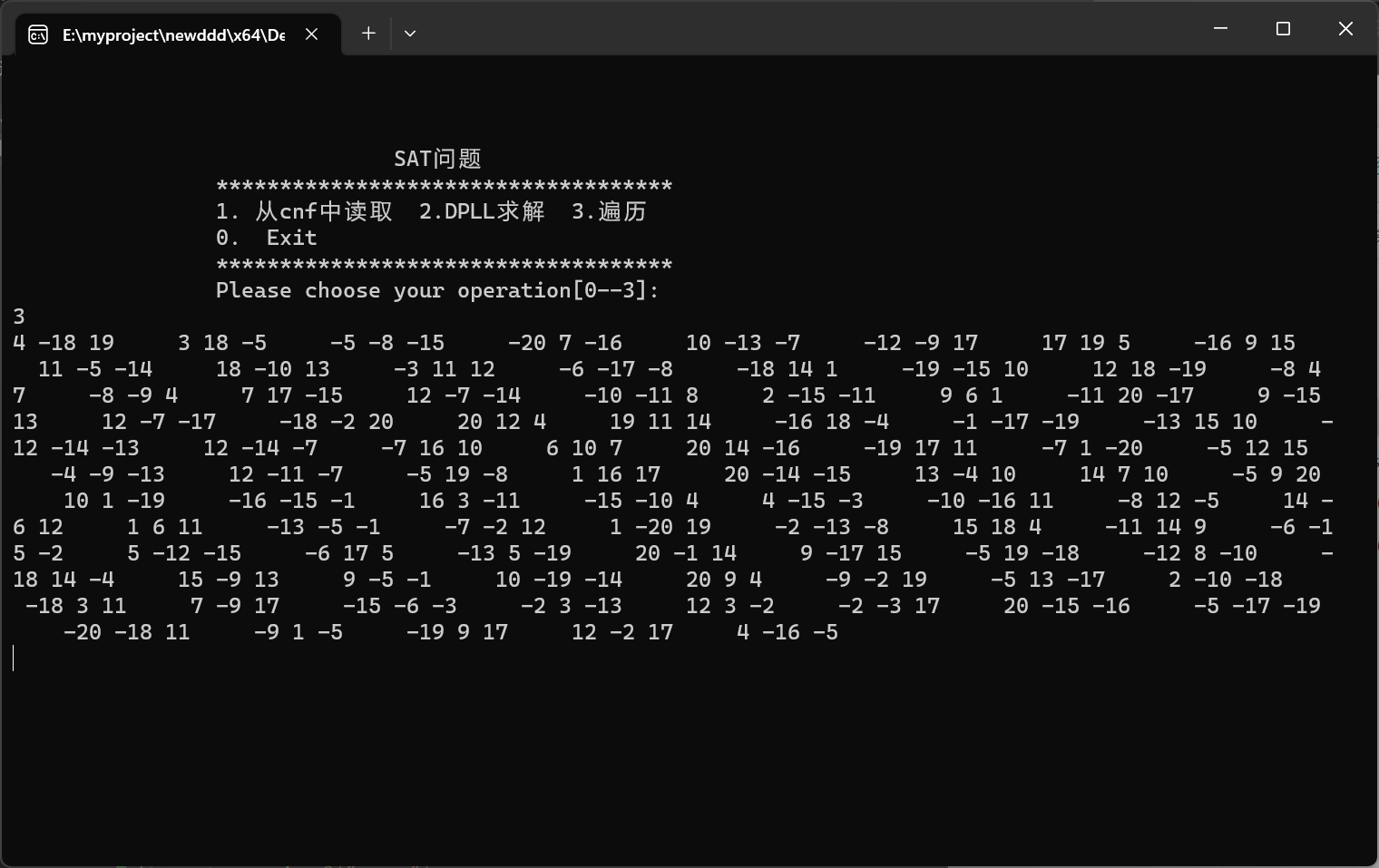


图4-5 遍历

CNF基准算例求解测试（时间单位：ms）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 变元数 | 子句数 | 算例文件夹 | 算例名称 | 优化前 | 优化后 |
| 20 | 91 | 功能测试 | sat-20.cnf | 1 | 2 |
| 30 | 420 | 功能测试 | unsat-5cnf-30.cnf | No solution | No solution |
| 181 | 3151 | 性能测试 | ais10.cnf | 8783 | 9254 |
| 303 | 2851 | 性能测试 | sud00009.cnf | 77 | 596 |

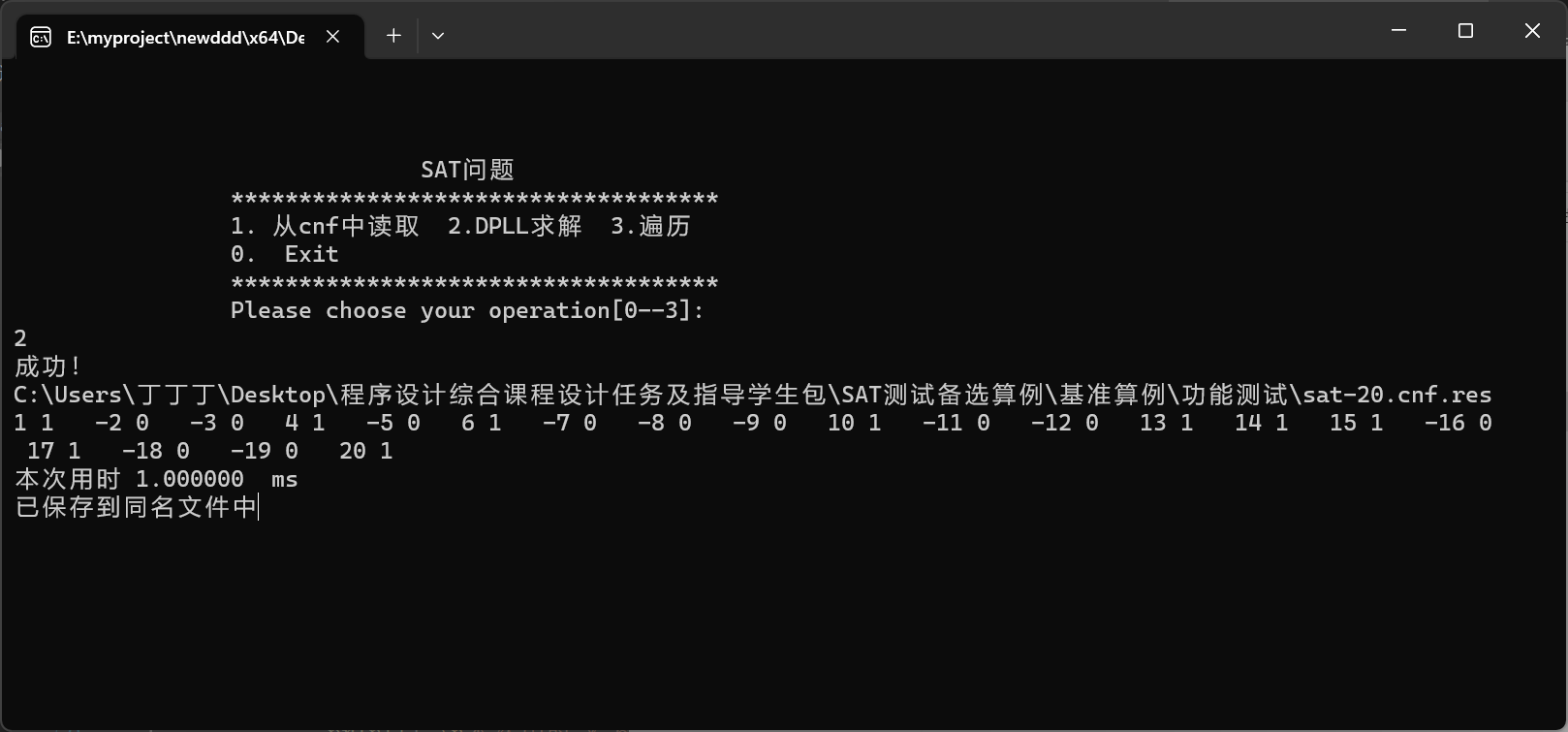


图4-6 sat-20.cnf优化前

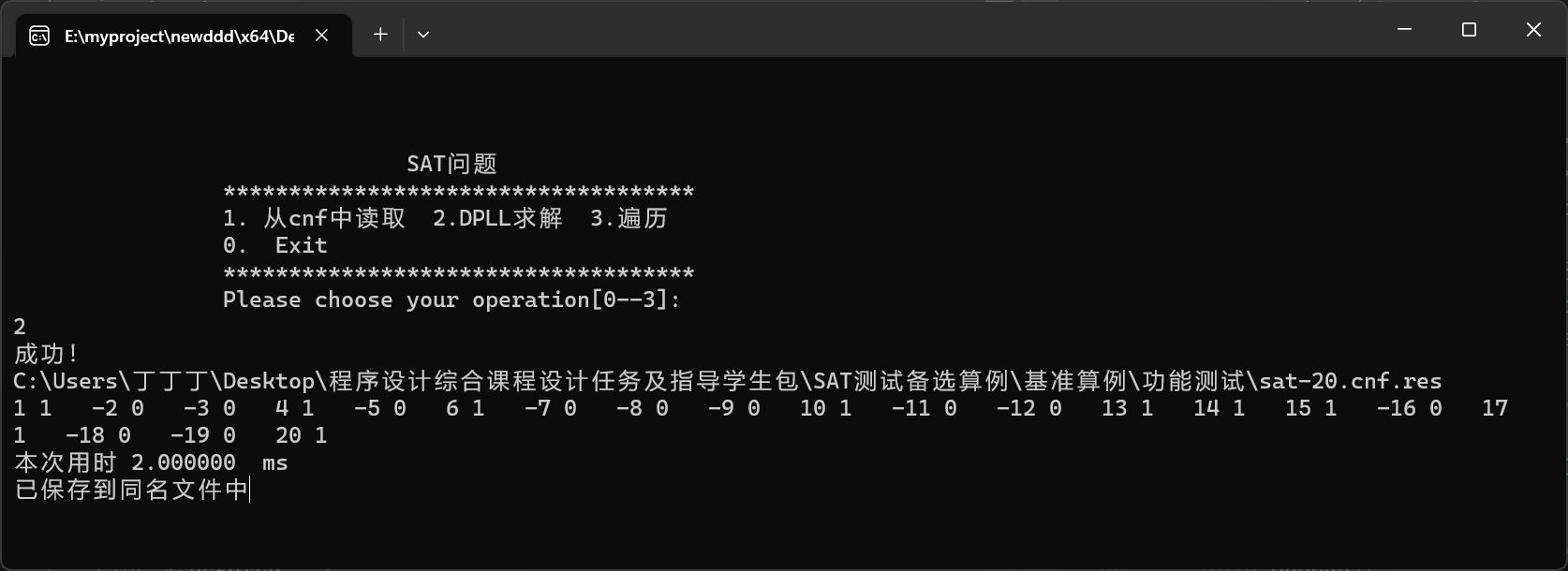


图4-7 sat-20.cnf优化后

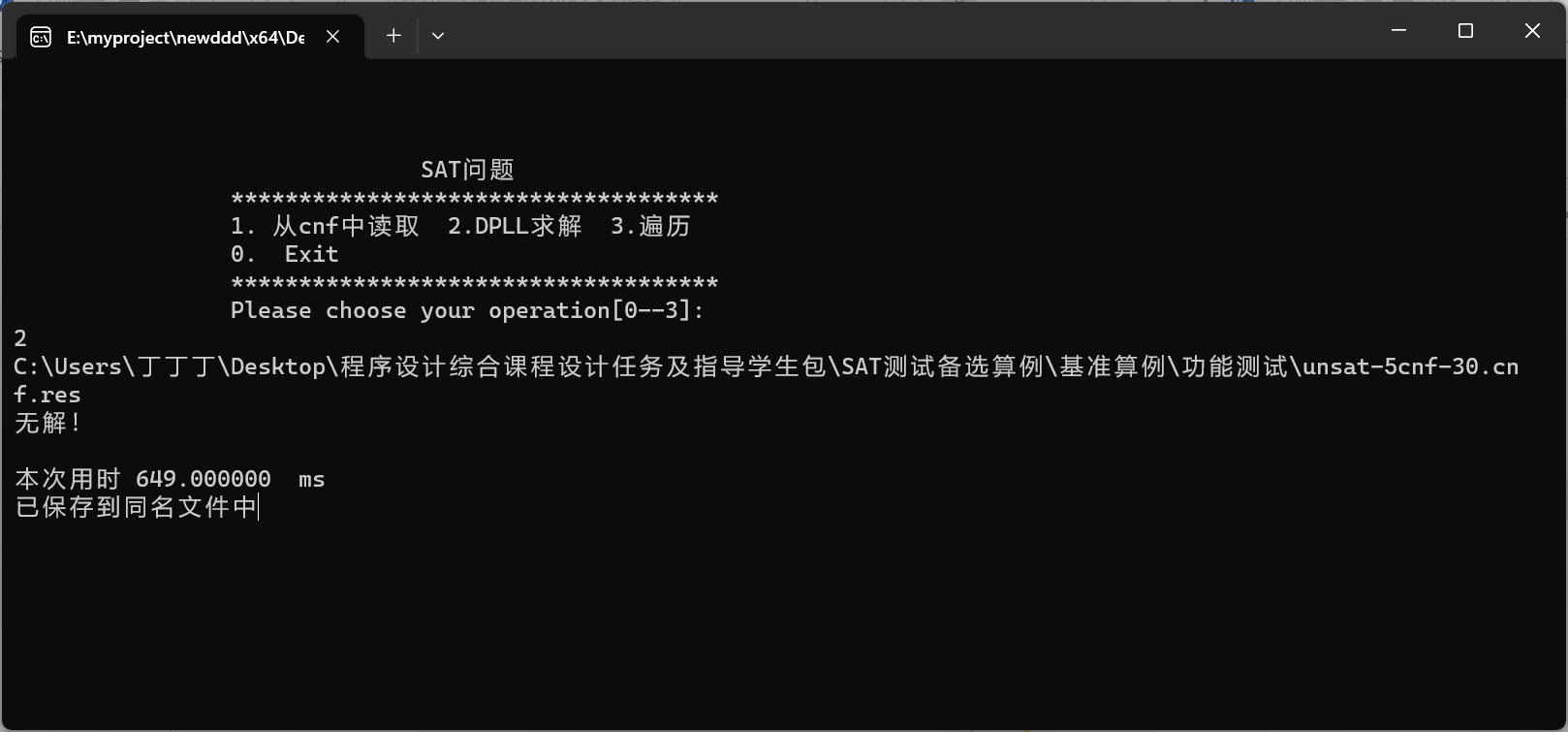


图4-8 unsat-5cnf-30.cnf算例

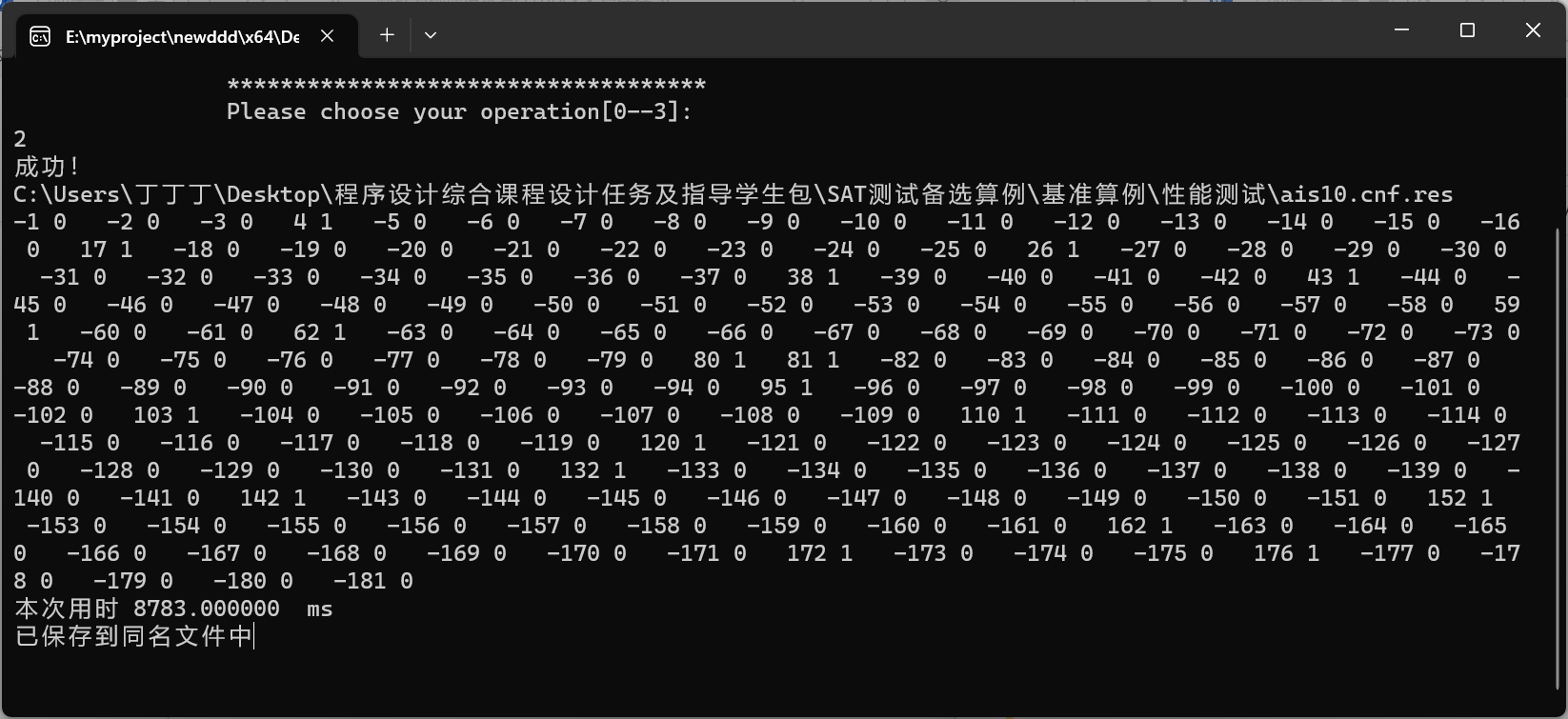


图4-9 ais10.cnf算例优化前

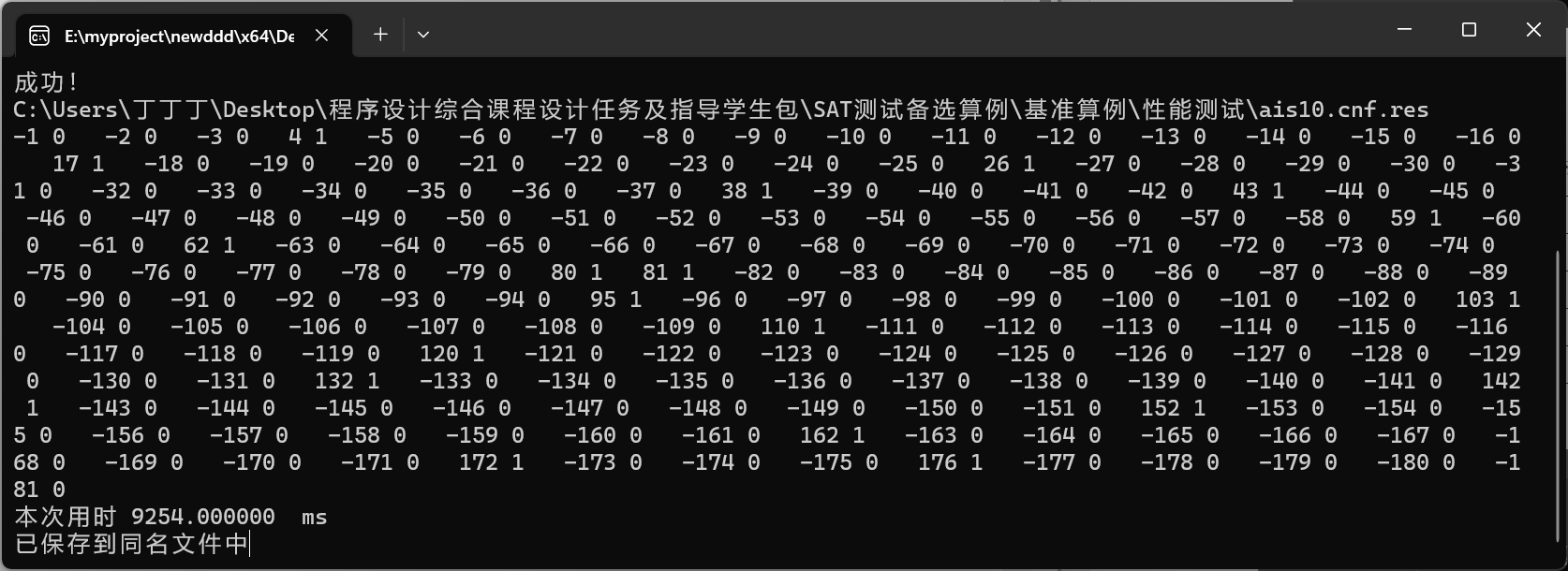


图4-10 ais10.cnf算例优化后

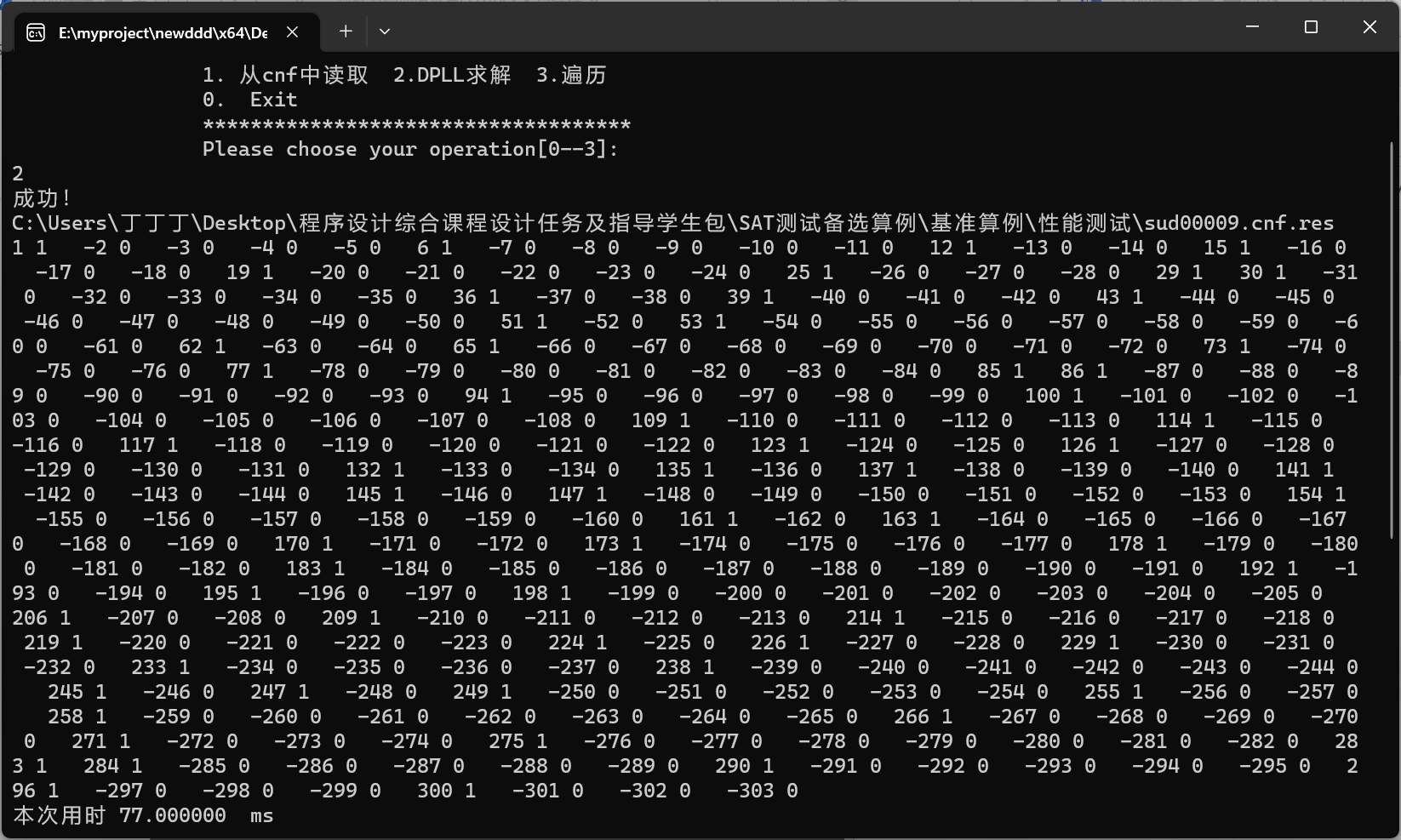


图4-11 sud00009.cnf算例优化前

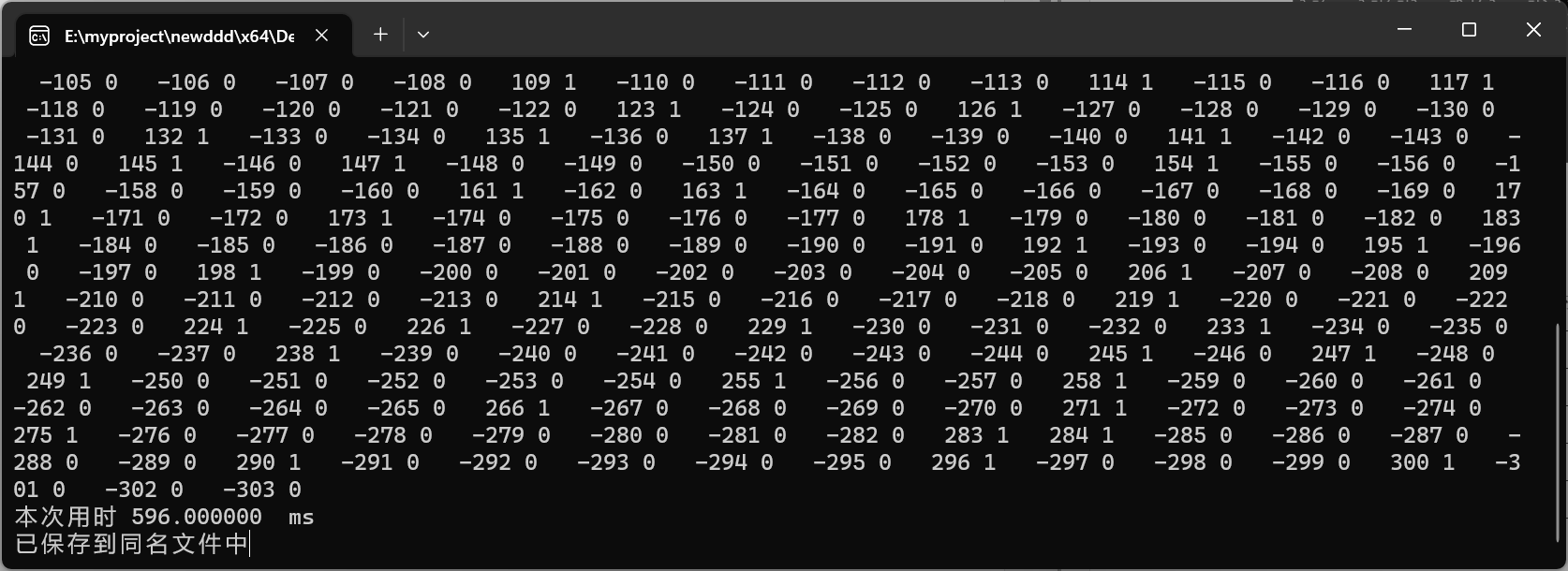


图4-12 sud00009.cnf优化后

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 变元数 | 子句数 | 算例属性 | 算例名称 | 优化前（ms） | 优化后（ms） |
| 20 | 91 | S | problem1-20.cnf | 3 | 2 |
| 50 | 80 | S | problem2-50.cnf | 20 | 16 |
| 100 | 340 | S | problem3-100.cnf | 867 | 657 |
| 50 | 100 | S | problem6-50.cnf | 55 | 517 |
| 50 | 300 | S | problem8-50.cnf | 11 | 9 |
| 100 | 600 | S | problem11-100.cnf | 152 | 175 |

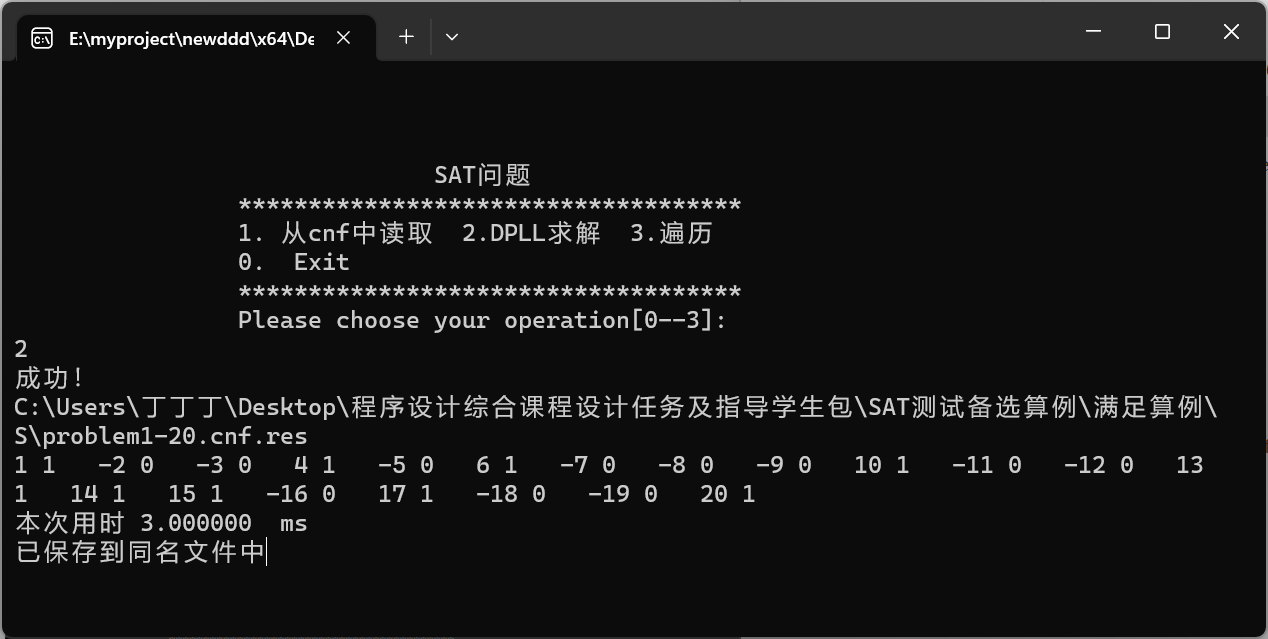


图4-13 problem1-20.cnf优化前

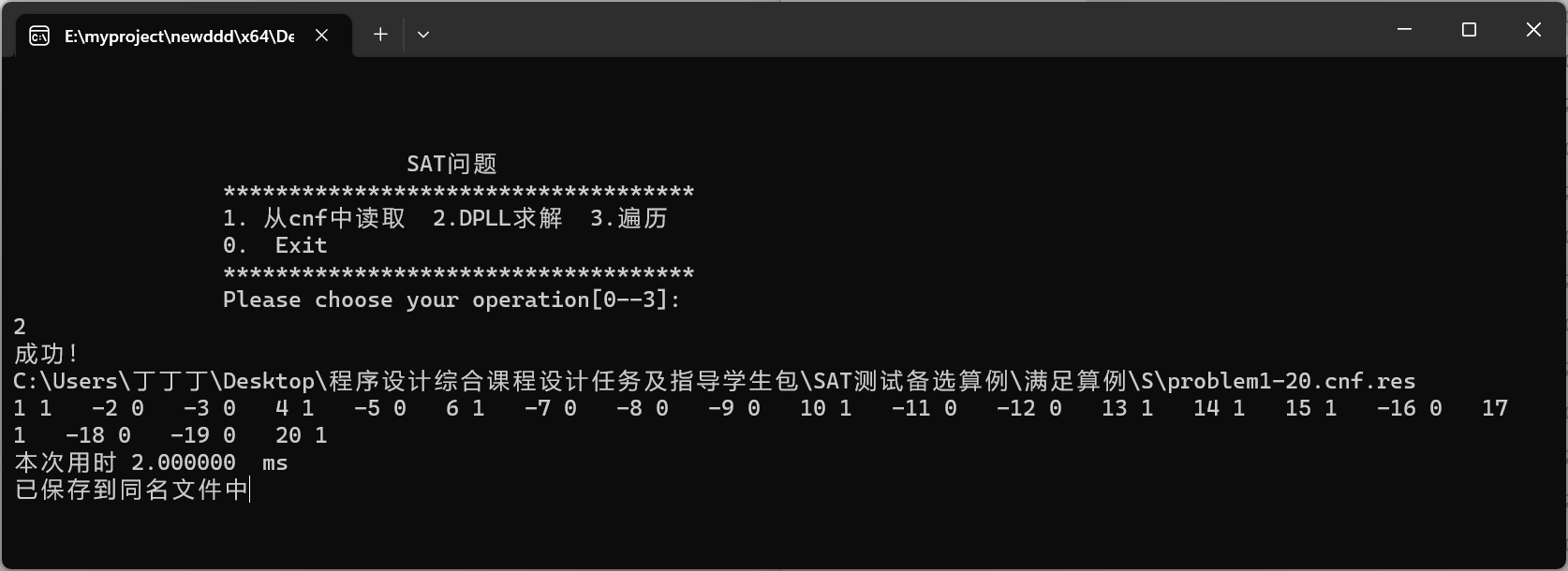


图4-14 problem1-20.cnf优化后

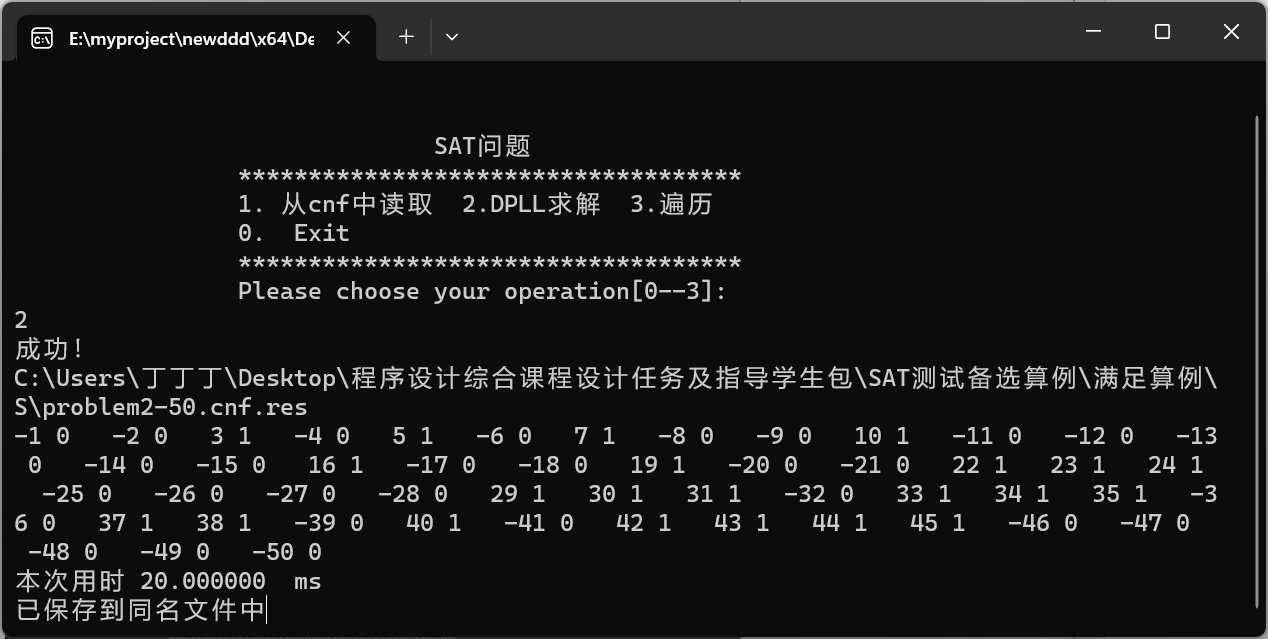


图4-15 problem2-50.cnf优化前

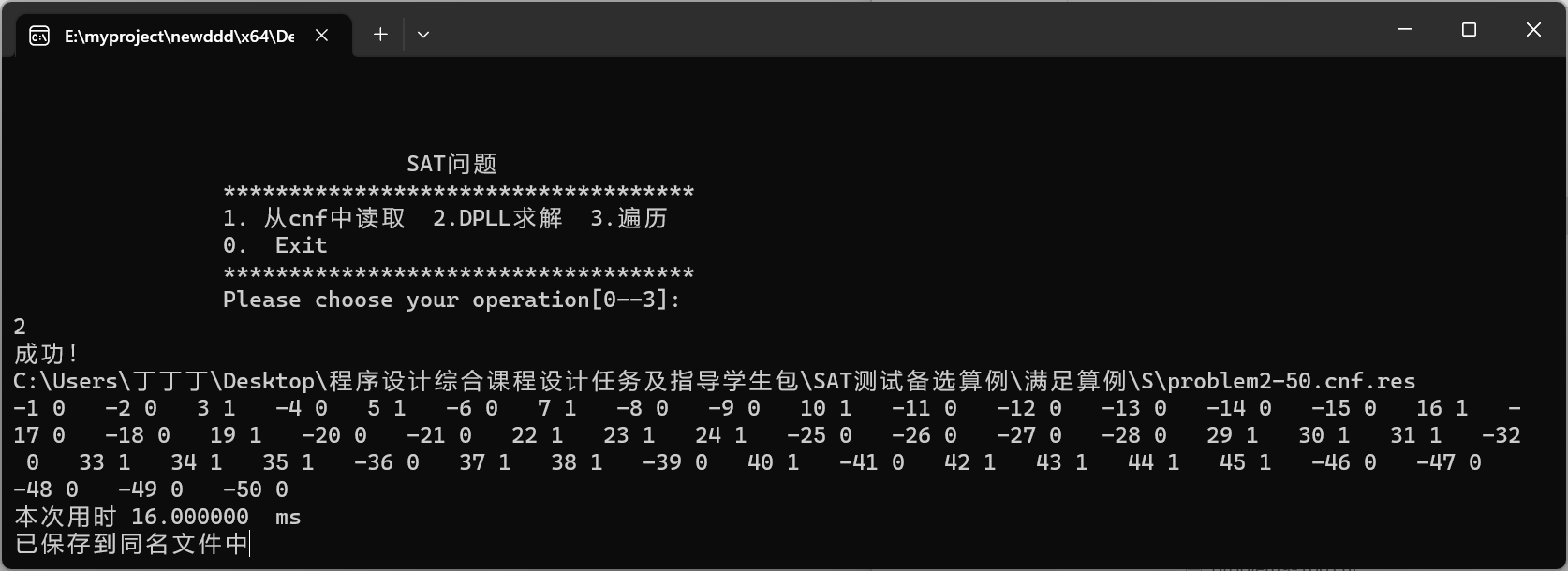


图4-16 problem2-50.cnf优化后

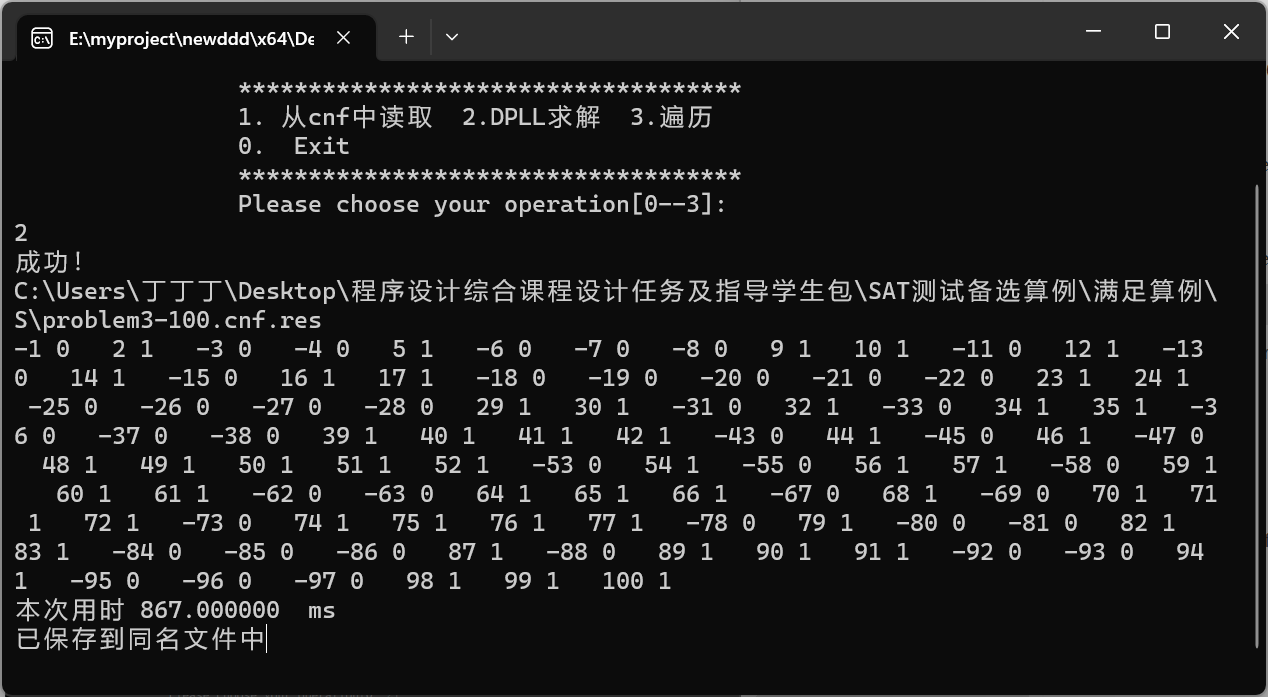


图4-17 problem3-100.cnf优化前

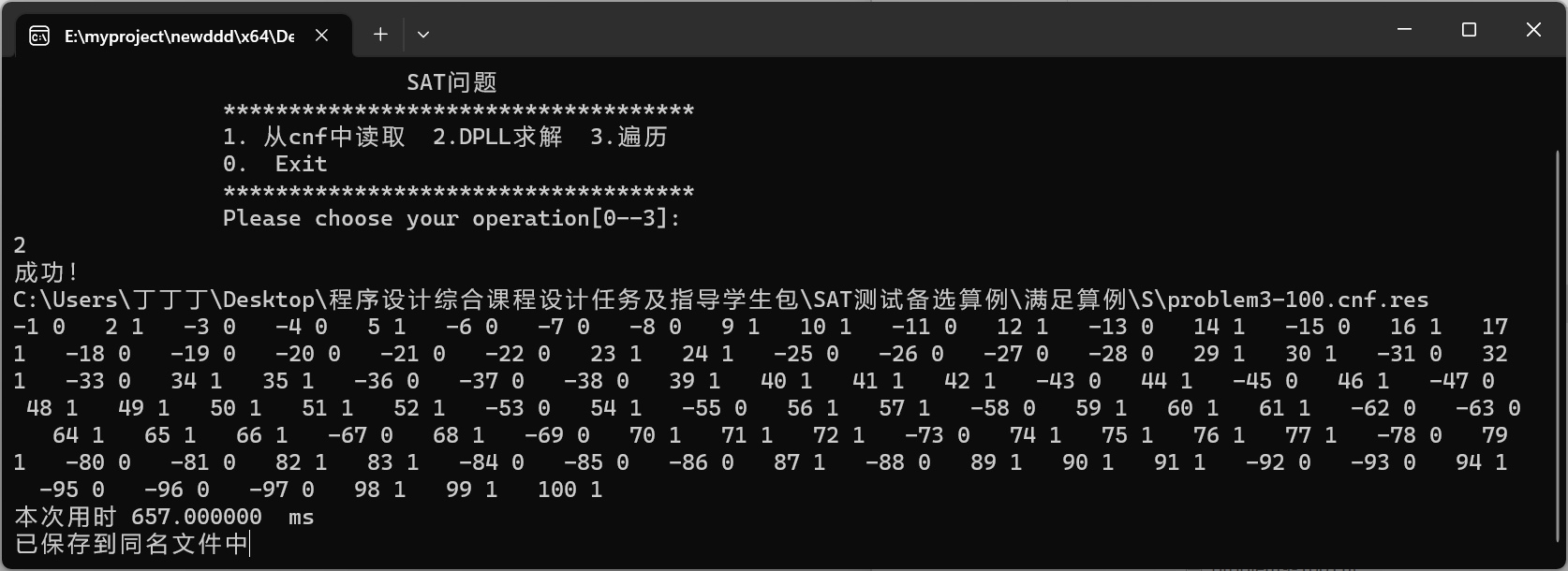


图4-18 problem3-100.cnf优化后

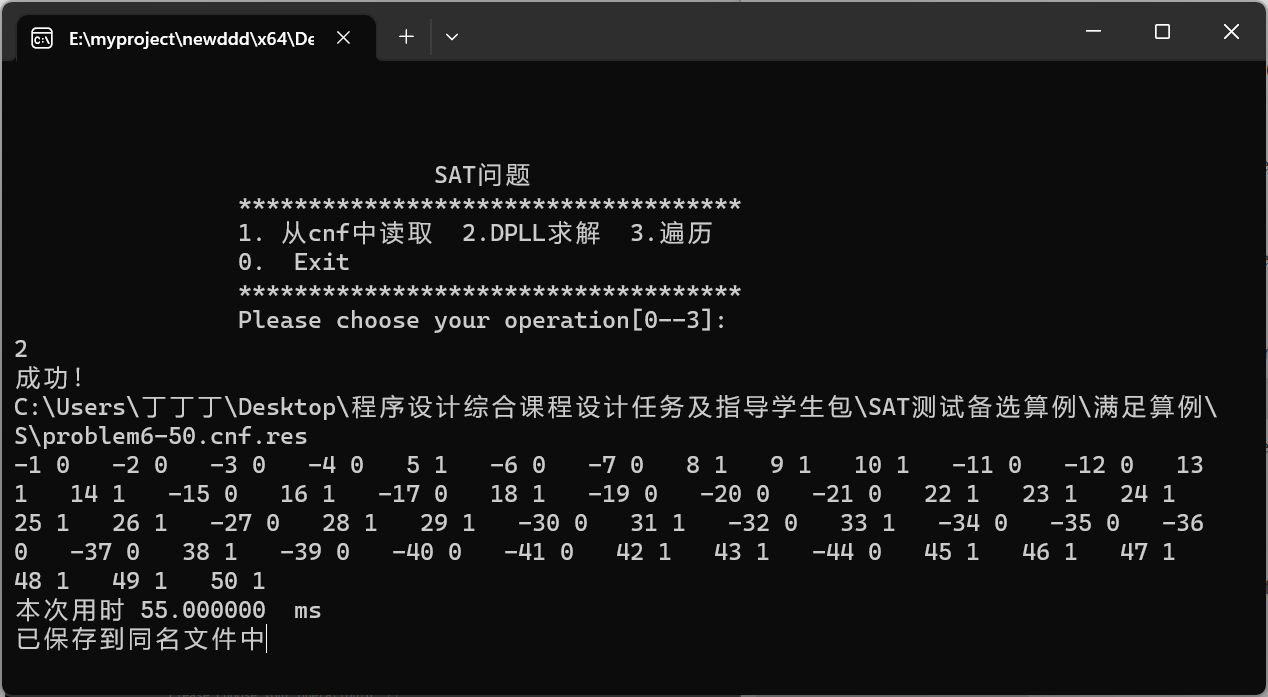


图4-19 problem6-50.cnf优化前

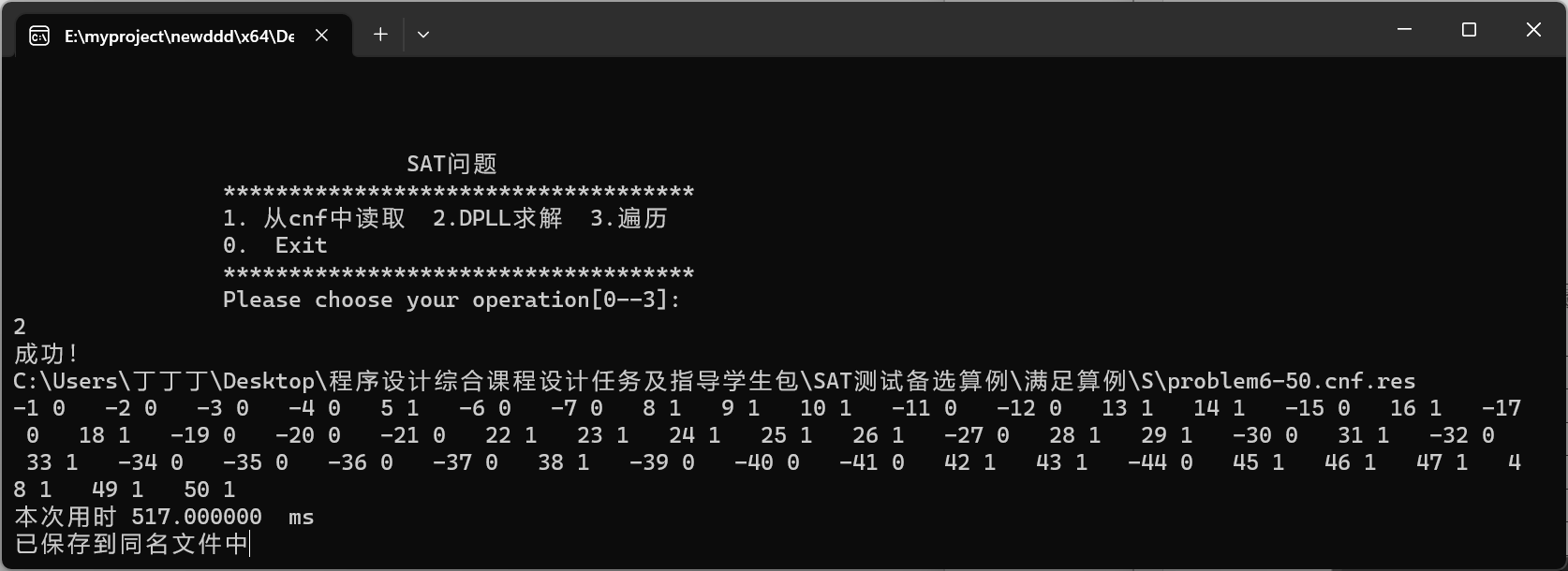


图4-20 problem6-50.cnf优化后

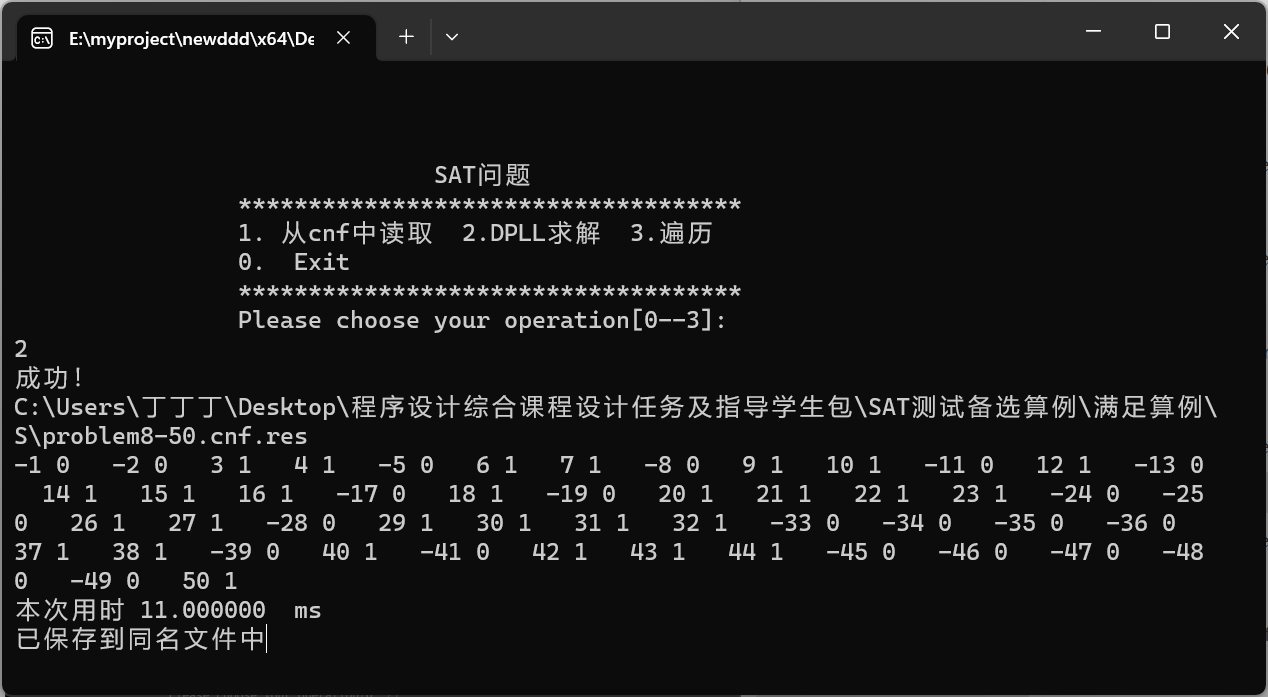


图4-21 problem8-50.cnf优化前

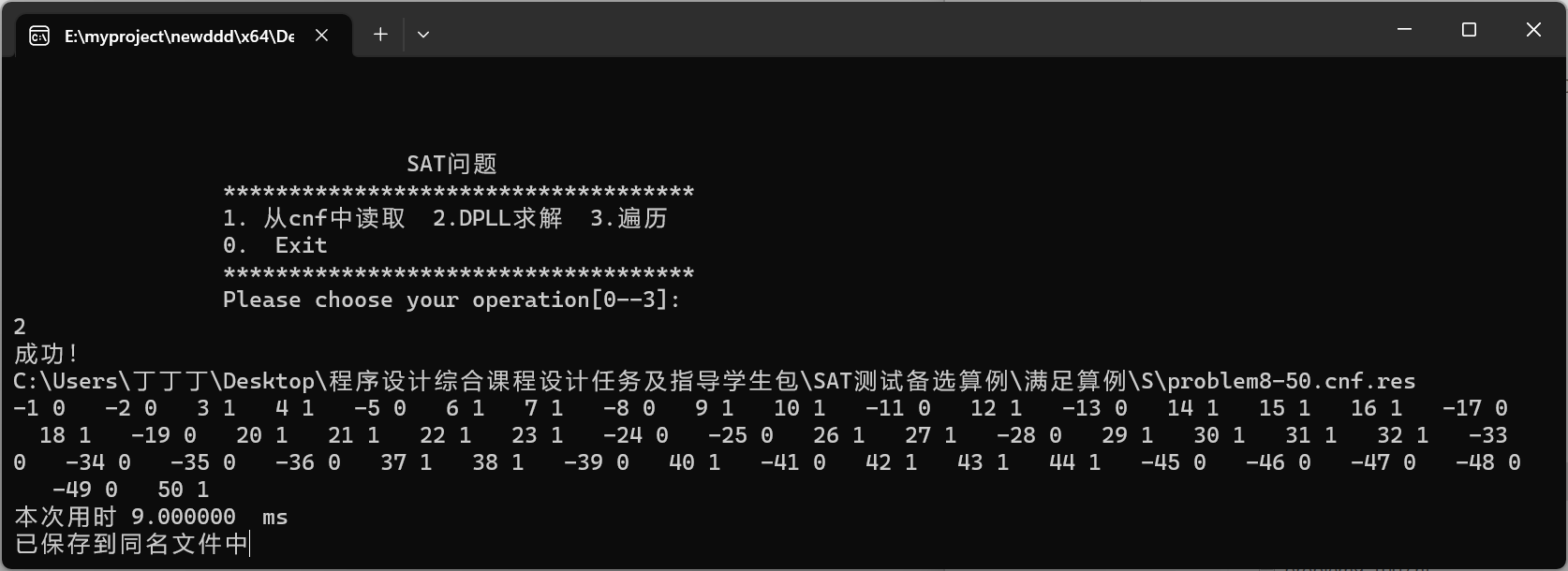


图4-22 problem8-50.cnf优化后

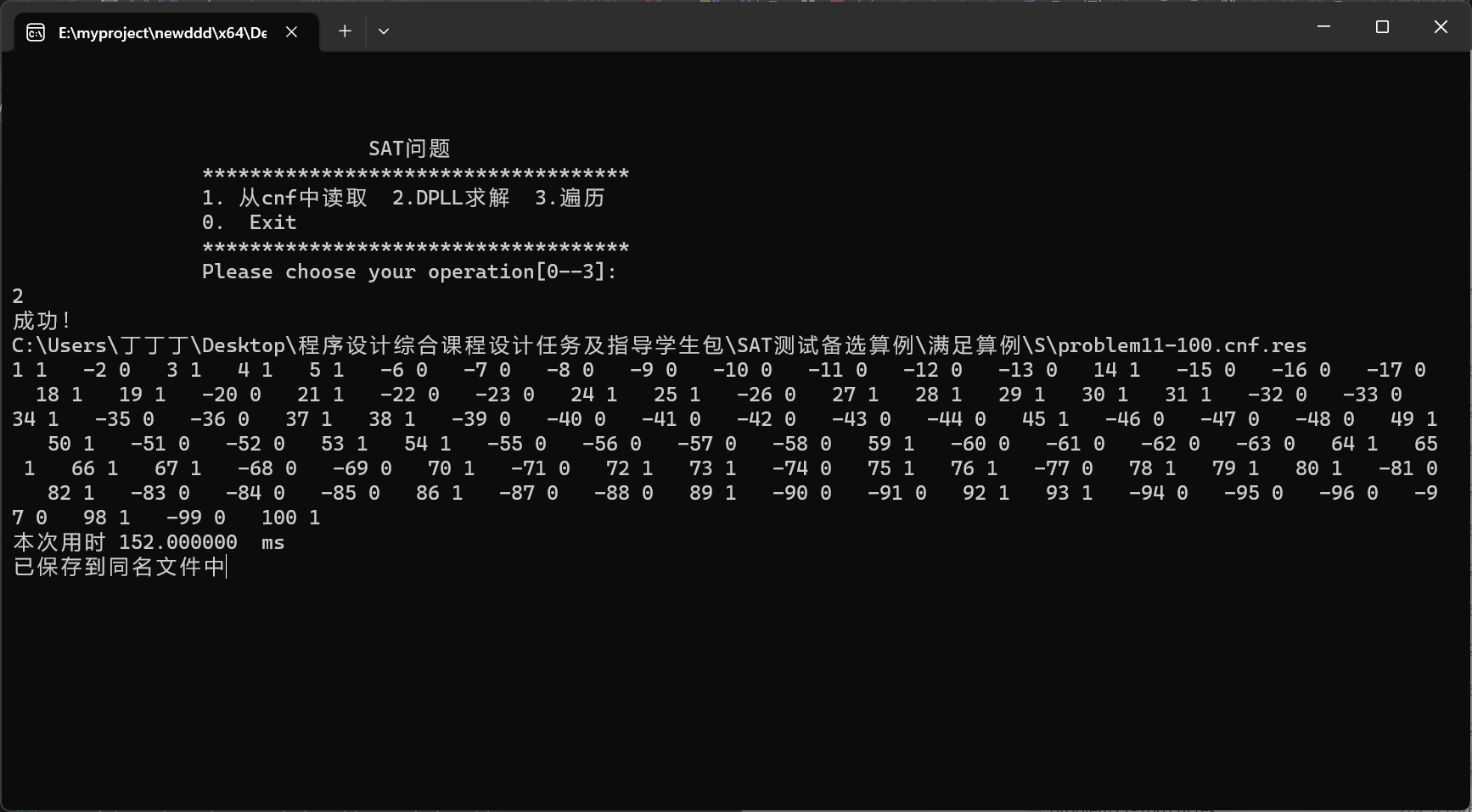


图4-23 problem11-100.cnf优化前

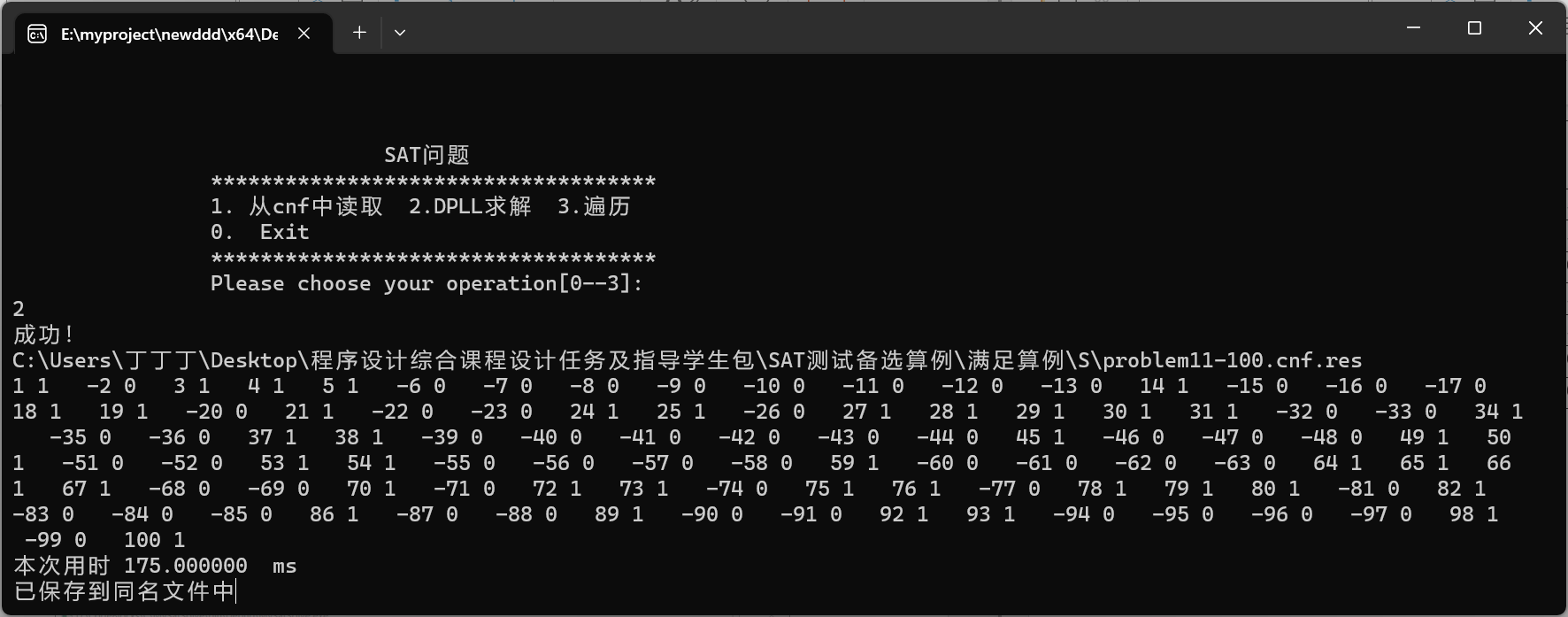


图4-24 problem11-100.cnf优化后

分析：相较于优化前，优化后的算法在部分算例上的用时甚至更长了。我认为这反映出不同策略应对不同场景的优劣是不同的。考虑到算例都是小算例，我认为优化前的算法更加适合小算例，因为小算例的子句都不长，直接选第一个省去了找最短子句的过程，而优化后的算法在面对大算例会更加快速。

**4.4 蜂窝数独游戏测试**

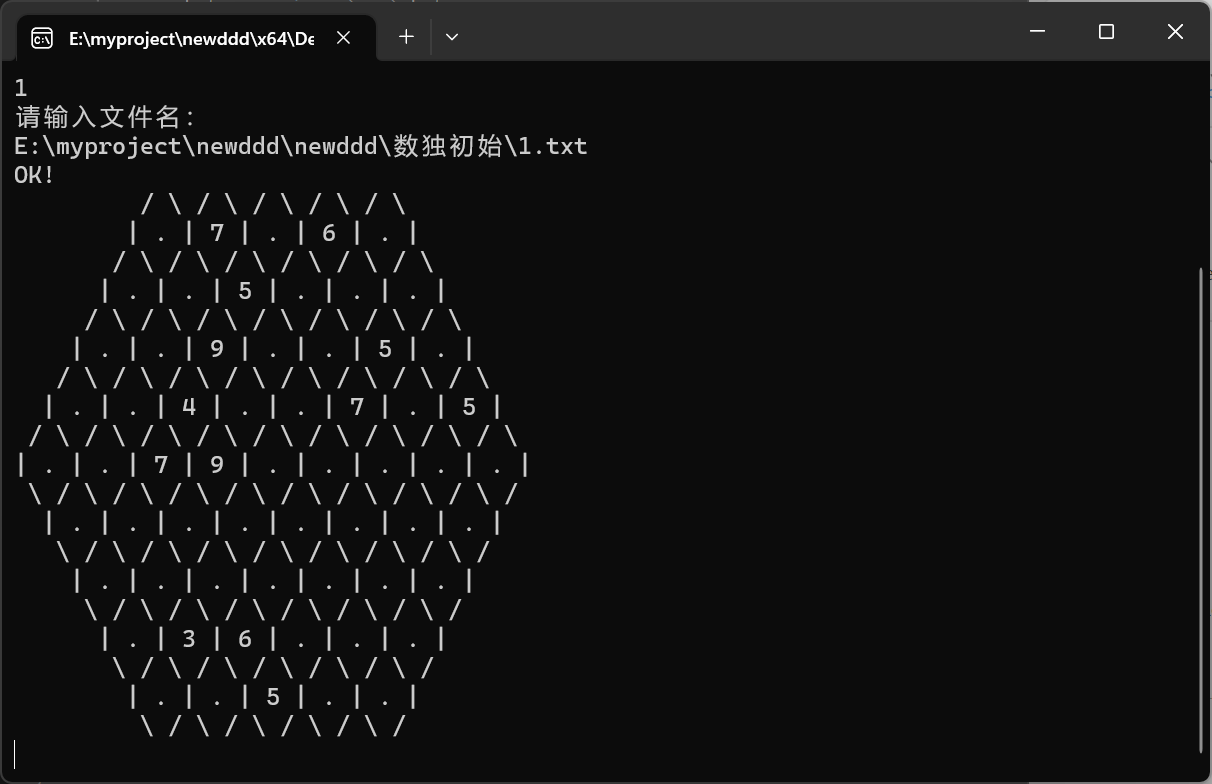


图4-25 初始化蜂窝数独



图4-26 求解游戏结果

**5总结与展望**

这次课设历时两周，对于我来说是一个巨大的挑战，不仅考验我的基础编程能力，更加要求我有统筹，整合的能力。对于整个程序的理解要很透彻。它的难度很大，结合了C语言与数据结构的知识。起初我对于DPLL算法都难以理解，经过自学，思索，请教，我彻底理解了其中的每个细节，并在优化上做了尝试。

本次课设，我完成了SAT计算器的设计，蜂窝数独游戏的搭建，以及整个程序的连结构建，确实是一次进步。但是在优化算法和蜂窝数独初始化上仍然有很大空间，我会继续思考尝试的。

通过课程设计教学与实践环节，我进一步正确理解与应用了专业知识，增强和提高了分析问题与解决问题的综合能力，加深了对于求解实际问题的基本科研步骤的体会与理解，增强和提升了信息搜索和分析技能，培养了技术总结的基本技能，锻炼了课程设计报告的撰写的能力。

**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**

附录1 ：Sat.h

#define \_CRT\_SECURE\_NO\_WARNINGS

#pragma once

#include<stdio.h>

#include <stdlib.h>

#include <math.h>

#include <time.h>

#include <string.h>

#define Unknow -1

#define False 0

#define True 1

#define None 2

#define MaxNumVar 1000

extern int Varjl[MaxNumVar + 1];

extern int var, nc;

extern int Su[10][10];//数独数组

typedef struct Node {

int data;

int negated;//1正 0负+

struct Node\* next;

}Node;

typedef struct clauselist {

int w;//子句变元数量。-1代表删除了

Node\* p;

clauselist\* next;

}ClauseList;

int Readcnf(char\* filename, ClauseList\*\* s, int\* var, int\* nc);

int Sat();

int DPLL(ClauseList\*\* S);

int DPLL2(ClauseList\*\* S);

void Propagation(ClauseList\*\* S, Node x);//单子句传播

ClauseList\* Copy(ClauseList\* S);

void ClearNodeList(Node\* node);

void ClearList(ClauseList\*\* S);

void PrintfCnf(ClauseList\* S);

void Add(ClauseList\*\* S, Node x);

int Save(int a, char\* filename, double time);

int check();

int IsEmpty(ClauseList\* S);//有没有空子句

int IsOK(ClauseList\* S);

ClauseList\* IsSingleClause(ClauseList\* S);

void Sudoku();

int ToCnf(int a[][10]);

int First(char\* filename);

void Tu();

int ReadSudoku(ClauseList\*\* s);

附录2：main.cpp

#include"Sat.h"

int Varjl[MaxNumVar + 1];

int var, nc;

int Su[10][10];

int main() {

int op = 1;

while (op) {

printf("\n\n\n");//菜单

printf("\t\t\t Main Menu \n");

printf("\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("\t\t\t1. Sudoku 2. SAT\n");

printf("\t\t\t0. Exit\n");

printf("\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("\t\t\tPlease choose your operation[0--2]:\n");

scanf("%d", &op);

system("cls");//清屏

switch (op) {

case 1:

Sudoku();

break;

case 2:

Sat();

break;

case 0:

exit(0);

default:

printf("\t\t\tPlease choose your operation again[0--2]:\n");

scanf("%d", &op);

}

}

return 0;

}

附录3：hanshu.cpp

#include"Sat.h"

int Sat() {

int op = 1,ddd, savejudge;

clock\_t start, end;

double time;

char filename[300];

Node ccs;

ClauseList\* S = NULL;

ClauseList\* cs;

while (op) {

system("cls");

printf("\n\n\n");//菜单

printf("\t\t\t SAT问题 \n");

printf("\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("\t\t1. 从cnf中读取 2.DPLL求解 3.遍历\n");

printf("\t\t0. Exit\n");

printf("\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("\t\tPlease choose your operation[0--3]:\n");

scanf("%d", &op);

//清屏

switch (op) {

case 2:

for (int i = 1; i <= MaxNumVar; i++) {

Varjl[i] = Unknow;

}

start = clock();

ddd = DPLL2(&S);

end = clock();

time = (double)(end - start)\*1000 / CLOCKS\_PER\_SEC;

if (ddd == 0) {

savejudge=Save(0,filename,time);

printf("无解！\n");

}

else {

printf("成功！\n");

savejudge=Save(1,filename,time);

for (int i = 1; i <= var; i++) {

if (Varjl[i] == True) {

printf("%d %d ", i,Varjl[i]);

}

else {

printf("-%d %d ", i,Varjl[i]);

}

}

}

printf("\n本次用时 %.6f ms\n", time);

if (savejudge == 1) {

printf("已保存到同名文件中");

}

if (savejudge == 0) {

printf("保存失败！");

}

getchar(); getchar();

break;

case 1:

printf("请输入cnf文件名：\n");

scanf("%s", filename);

ddd = Readcnf(filename, &S,&var,&nc);

if (ddd == 1) printf("读取成功！");

else {

printf("读取失败！");

S = NULL;

}

getchar(); getchar();

break;

case 3:

if (S == NULL) {

printf("请先输入cnf文件\n");

}

else {

PrintfCnf(S);

}

getchar(); getchar();

break;

case 0:

exit(0);

default:

printf("\t\t\tPlease choose your operation again[0--1]:\n");

scanf("%d", &op);

}

}

}

int DPLL(ClauseList\*\* S) {

ClauseList\* qian;

Node x;

qian = IsSingleClause(\*S);

while (qian!=NULL) {

x.data = qian->p->data;

x.negated = qian->p->negated;

x.next = NULL;

qian->w = -1;

if (x.negated == 1) {

Varjl[x.data] = True;

}

if (x.negated == 0) {

Varjl[x.data] = False;

}

Propagation(S, x);

if (IsOK(\*S)) return 1;

else if (IsEmpty(\*S)) return 0;

qian= IsSingleClause(\*S);

}

qian = \*S;

while (qian != NULL) {

if (qian->w != -1) {

x.data = qian->p->data;

x.negated = qian->p->negated;

x.next = NULL;

}

qian = qian->next;

}

ClauseList\* Sc = Copy(\*S);

//printf("选取了%d %d\n", x.data, x.negated);

Add(&Sc, x);

if (DPLL(&Sc)) {

ClearList(&Sc);

return 1;

}

else {

ClearList(&Sc);

ClauseList\* Sc2 = Copy(\*S);

if (x.negated == 1) x.negated = 0;

else x.negated = 1;

Add(&Sc2, x);

return DPLL(&Sc2);

}

}

int check() {

for (int i = 1; i <= var; i++) {

if (Varjl[i] != True && Varjl[i] != False) return 0;

}

return 1;

}

ClauseList\* IsSingleClause(ClauseList\* S) { //找单子句

ClauseList\* l = S;

while (l != NULL) {

if (l->w == 1) return l;

l = l->next;

}

return NULL;

}

ClauseList\* Copy(ClauseList\* S) {

ClauseList\* q = NULL, \* c = NULL;

ClauseList\* h = NULL;

Node\* n = NULL, \* cc = NULL;

q = S;

h = (ClauseList\*)malloc(sizeof(ClauseList));

h->next = NULL;

h->w = q->w;

if (q->p == NULL) {

h->p = NULL;

}

else {

h->p = (Node\*)malloc(sizeof(Node));

h->p->data = q->p->data;

h->p->negated = q->p->negated;

h->p->next = NULL;

n = q->p->next;

cc = h->p;

while (n != NULL) {

cc->next = (Node\*)malloc(sizeof(Node));

cc = cc->next;

cc->data = n->data;

cc->negated = n->negated;

cc->next = NULL;

n = n->next;

}

}

c = h;

q = q->next;

while (q != NULL) {

c->next=(ClauseList\*)malloc(sizeof(ClauseList));

c = c->next;

c->w = q->w;

if (q->p == NULL) {

c->p = NULL;

}

else {

c->p = (Node\*)malloc(sizeof(Node));

c->p->data = q->p->data;

c->p->negated = q->p->negated;

c->p->next = NULL;

n = q->p->next;

cc = c->p;

while (n != NULL) {

cc->next = (Node\*)malloc(sizeof(Node));

cc = cc->next;

cc->data = n->data;

cc->negated = n->negated;

cc->next = NULL;

n = n->next;

}

}

q = q->next;

}

c->next= NULL;

return h;

}

int IsOK(ClauseList\* S) {

ClauseList\* l = S;

while (l != NULL) {

if (l->w != -1) return 0;

l = l->next;

}

return 1;

}

int IsEmpty(ClauseList\* S) { //有没有空子句

ClauseList\* l = S;

while (l != NULL) {

if (l->w == 0) return 1;

l = l->next;

}

return 0;

}

void Propagation(ClauseList\*\* S, Node x) {//单子句传播

ClauseList\* current = \*S;

ClauseList\* pre = NULL;

while (current != NULL) {

Node\* prev = NULL;

Node\* current\_p = current->p;

if (current->w > 0) {

while (current\_p != NULL) {

if (current\_p->data == x.data) {

if (current\_p->negated == x.negated) {

current->w = -1;

break;

}

else {

if (prev == NULL) {

current->p = current\_p->next;

free(current\_p);

current\_p = current->p;

}

else {

prev->next = current\_p->next;

free(current\_p);

current\_p = prev->next;

}

(current->w)--;

}

}

else {

prev = current\_p;

current\_p = current\_p->next;

}

}

}

pre = current;

current = current->next;

}

}

int Readcnf(char\* filename, ClauseList\*\* s,int\* var,int\* nc) {

FILE\* fp = fopen(filename, "r");

if (fp == NULL) {

printf("ERROR!\n");

return 0;

}

int aa, bb;

int num=1,m;

char c;

//CnfSign k;

ClauseList\* h = NULL;

ClauseList\* lp = NULL;

Node\* hh = NULL;

Node\* pp = NULL;

fscanf(fp, "%c", &c);

while (c == 'c') {//注释行

while (c != '\n' && c != '\r') {

fscanf(fp, "%c", &c);

}

if (c == '\n') fscanf(fp, "%c", &c);

}

if (c == 'p') {//读p行

fscanf(fp, " cnf %d %d", &aa, &bb);

}

\*var = aa;

\*nc = bb;

//二维尾插法读入nc行数据

h = (ClauseList\*)malloc(sizeof(ClauseList));

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

if (m == 0) {

free(h);

return 0;

}

hh = (Node\*)malloc(sizeof(Node));

if (m > 0) {

hh->data = abs(m);

hh->negated = 1;

}

if (m < 0) {

hh->data = abs(m);

hh->negated = 0;

}

hh->next = NULL;

pp = hh;

while (1) {

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

if (m == 0) break;

pp->next = (Node\*)malloc(sizeof(Node));

pp = pp->next;

if (m > 0) {

pp->data = abs(m);

pp->negated = 1;

}

if (m < 0) {

pp->data = abs(m);

pp->negated = 0;

}

pp->next = NULL;

num++;

}

h->w = num;

h->p = hh;

h->next = NULL;

lp = h;

for (int i = 1; i < bb; i++) {

num = 1;

lp->next = (ClauseList\*)malloc(sizeof(ClauseList));

lp = lp->next;

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

hh = (Node\*)malloc(sizeof(Node));

if (m > 0) {

hh->data = abs(m);

hh->negated = 1;

}

if (m < 0) {

hh->data = abs(m);

hh->negated = 0;

}

hh->next = NULL;

pp = hh;

while (1) {

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

if (m == 0) break;

pp->next = (Node\*)malloc(sizeof(Node));

pp = pp->next;

if (m > 0) {

pp->data = abs(m);

pp->negated = 1;

}

if (m < 0) {

pp->data = abs(m);

pp->negated = 0;

}

pp->next = NULL;

num++;

}

lp->w = num;

lp->p = hh;

lp->next = NULL;

}

\*s = h;

fclose(fp);

return 1;

}

void PrintfCnf(ClauseList\* S) {

ClauseList\* l;

Node\* q;

l = S;

while (l != NULL) {

//printf("%d ", l->w);

q = l->p;

while (q != NULL) {

if (q->negated == 1) printf("%d ", q->data);

else printf("-%d ", q->data);

q = q->next;

}

printf(" ");

l = l->next;

}

}

//释放函数\*2

void ClearNodeList(Node\* node) {

if (node == NULL) {

return;

}

Node\* current = node;

while (current != NULL) {

Node\* temp = current;

current = current->next;

free(temp);

}

}

void ClearList(ClauseList\*\* S) {

if (S == NULL || \*S == NULL) {

return;

}

ClauseList\* current = \*S;

while (current != NULL) {

ClauseList\* temp = current;

current = current->next;

ClearNodeList(temp->p); // 释放 Node 链表内存

free(temp);

}

\*S = NULL; // 设置 S 为 NULL，确保外部指针不再引用已释放的内存

}

//加单子句函数

void Add(ClauseList\*\* S, Node x) {

ClauseList\* l;

l = \*S;

while (l->next != NULL) {

l = l->next;

}

l->next= (ClauseList\*)malloc(sizeof(ClauseList));

l = l->next;

l->p= (Node\*)malloc(sizeof(Node));

l->p->data = x.data;

l->p->negated = x.negated;

l->p->next = NULL;

l->next = NULL;

l->w = 1;

return ;

}

//保存函数

int Save(int a, char\* filename,double time) {

int l = strlen(filename);

int i = 0;

char nf[300];

strcpy(nf, filename);

strcat(nf, ".res");

FILE\* fp;

fp = fopen(nf, "w");

if (fp == NULL) {

return 0;

}

if (a == 0) {

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

fprintf(fp, "v none\n");

fprintf(fp, "t %.6f ms\n",time);

}

if (a == 1) {

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

fprintf(fp, "v ");

for (int i = 1; i <= var; i++) {

if (Varjl[i] == True) {

fprintf(fp,"%d ", i);

}

else {

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

}

}

fprintf(fp, "\n");

fprintf(fp, "t %.6f ms\n", time);

}

fclose(fp);

printf("%s\n", nf);

return 1;

}

int DPLL2(ClauseList\*\* S) {

ClauseList\* qian;

Node x;

qian = IsSingleClause(\*S);

while (qian != NULL) {

x.data = qian->p->data;

x.negated = qian->p->negated;

x.next = NULL;

qian->w = -1;

if (x.negated == 1) {

Varjl[x.data] = True;

}

if (x.negated == 0) {

Varjl[x.data] = False;

}

Propagation(S, x);

if (IsOK(\*S)) return 1;

else if (IsEmpty(\*S)) return 0;

qian = IsSingleClause(\*S);

}

qian = \*S;

int num = 1000;

while (qian != NULL) {

if (qian->w < num) {

num = qian->w;

}

qian = qian->next;

}

qian = \*S;

while (qian != NULL) {

if (qian->w == num) {

x.data = qian->p->data;

x.negated = qian->p->negated;

break;

}

qian = qian->next;

}

ClauseList\* Sc = Copy(\*S);

//printf("选取了%d %d\n", x.data, x.negated);

Add(&Sc, x);

if (DPLL(&Sc)) {

ClearList(&Sc);

return 1;

}

else {

ClearList(&Sc);

ClauseList\* Sc2 = Copy(\*S);

if (x.negated == 1) x.negated = 0;

else x.negated = 1;

Add(&Sc2, x);

return DPLL(&Sc2);

}

}

附录4：Sudoku.cpp

#include"Sat.h"

int ToCnf(int a[][10]) {

int b[10][10];

FILE\* fp;

fp = fopen("Sudoku.cnf", "w");

if (fp == NULL) {

printf("fail!\n");

return 0;

}

fprintf(fp, "p cnf 1000 7407\n");//行数

b[1][1] = 51; b[1][2] = 41; b[1][3] = 31; b[1][4] = 21; b[1][5] = 11;

b[2][1] = 61; b[2][2] = 52; b[2][3] = 42; b[2][4] = 32; b[2][5] = 22; b[2][6] = 12;

b[3][1] = 71; b[3][2] = 62; b[3][3] = 53; b[3][4] = 43; b[3][5] = 33; b[3][6] = 23; b[3][7] = 13;

b[4][1] = 81; b[4][2] = 72; b[4][3] = 63; b[4][4] = 54; b[4][5] = 44; b[4][6] = 34; b[4][7] = 24; b[4][8] = 14;

b[5][1] = 91; b[5][2] = 82; b[5][3] = 73; b[5][4] = 64; b[5][5] = 55; b[5][6] = 45; b[5][7] = 35; b[5][8] = 25; b[5][9] = 15;

b[6][1] = 92; b[6][2] = 83; b[6][3] = 74; b[6][4] = 65; b[6][5] = 56; b[6][6] = 46; b[6][7] = 36; b[6][8] = 26;

b[7][1] = 93; b[7][2] = 84; b[7][3] = 75; b[7][4] = 66; b[7][5] = 57; b[7][6] = 47; b[7][7] = 37;

b[8][1] = 94; b[8][2] = 85; b[8][3] = 76; b[8][4] = 67; b[8][5] = 58; b[8][6] = 48;

b[9][1] = 95; b[9][2] = 86; b[9][3] = 77; b[9][4] = 68; b[9][5] = 59;

//每一个格子不能填同样的数字 2196

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

for (int j = 1; j <= a[i][0]; j++) {

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

for (int q = 1; q < p; q++) {

fprintf(fp, "-%d%d%d -%d%d%d 0\n", i, j, p, i, j, q);

}

}

}

}

//1-9行,每一行上的任意两格不能为同样的。 1656

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

int j = a[i][0];

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

for (int p = 1; p <= j; p++) {

for (int q = 1; q < p; q++) {

fprintf(fp, "-%d%d%d -%d%d%d 0\n", i, p, k, i, q, k);

}

}

}

}

//第5行 一定有123456789 9

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

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

fprintf(fp, "5%d%d ", j, k);

}

fprintf(fp, "0\n");

}

//第4行 12345678 23456789

//一定有 2345678 7

for (int k = 2; k <= 8; k++) {

for (int j = 1; j <= 8; j++) {

fprintf(fp, "4%d%d ", j, k);

}

fprintf(fp, "0\n");

}

//选填（1）（9） 2

for (int j = 1; j <= 8; j++) {

fprintf(fp, "4%d1 4%d9 ", j,j);

}

fprintf(fp, "0\n");

//第6行同4 一定有 2345678 7

for (int k = 2; k <= 8; k++) {

for (int j = 1; j <= 8; j++) {

fprintf(fp, "6%d%d ", j, k);

}

fprintf(fp, "0\n");

}

//选填（1）（9） 2

for (int j = 1; j <= 8; j++) {

fprintf(fp, "6%d1 6%d9 ", j,j);

}

fprintf(fp, "0\n");

//第3行 一定有34567 5

for (int k = 3; k <= 7; k++) {

for (int j = 1; j <= 7; j++) {

fprintf(fp, "3%d%d ", j, k);

}

fprintf(fp, "0\n");

}

//选填（18） （28） （29） 3

for (int j = 1; j <= 7; j++) {

fprintf(fp, "3%d1 3%d8 ", j, j);

}

fprintf(fp, "0\n");

for (int j = 1; j <= 7; j++) {

fprintf(fp, "3%d2 3%d8 ", j, j);

}

fprintf(fp, "0\n");

for (int j = 1; j <= 7; j++) {

fprintf(fp, "3%d2 3%d9 ", j, j);

}

fprintf(fp, "0\n");

//第7行 一定有34567 5

for (int k = 3; k <= 7; k++) {

for (int j = 1; j <= 7; j++) {

fprintf(fp, "7%d%d ", j, k);

}

fprintf(fp, "0\n");

}

//选填（18） （28） （29） 3

for (int j = 1; j <= 7; j++) {

fprintf(fp, "7%d1 7%d8 ", j, j);

}

fprintf(fp, "0\n");

for (int j = 1; j <= 7; j++) {

fprintf(fp, "7%d2 7%d8 ", j, j);

}

fprintf(fp, "0\n");

for (int j = 1; j <= 7; j++) {

fprintf(fp, "7%d2 7%d9 ", j, j);

}

fprintf(fp, "0\n");

//第2行 123456 234567 345678 456789

//一定有456 3

for (int k = 4; k <= 6; k++) {

for (int j = 1; j <= 6; j++) {

fprintf(fp, "2%d%d ", j, k);

}

fprintf(fp, "0\n");

}

//选填 17 27 28 37 38 39 6

for (int j = 1; j <= 6; j++) {

fprintf(fp, "2%d1 2%d7 ", j, j);

}

fprintf(fp, "0\n");

for (int j = 1; j <= 6; j++) {

fprintf(fp, "2%d2 2%d7 ", j, j);

}

fprintf(fp, "0\n");

for (int j = 1; j <= 6; j++) {

fprintf(fp, "2%d2 2%d8 ", j, j);

}

fprintf(fp, "0\n");

for (int j = 1; j <= 6; j++) {

fprintf(fp, "2%d3 2%d7 ", j, j);

}

fprintf(fp, "0\n");

for (int j = 1; j <= 6; j++) {

fprintf(fp, "2%d3 2%d8 ", j, j);

}

fprintf(fp, "0\n");

for (int j = 1; j <= 6; j++) {

fprintf(fp, "2%d3 2%d9 ", j, j);

}

fprintf(fp, "0\n");

//第8行

//一定有456 3

for (int k = 4; k <= 6; k++) {

for (int j = 1; j <= 6; j++) {

fprintf(fp, "8%d%d ", j, k);

}

fprintf(fp, "0\n");

}

//选填 17 27 28 37 38 39 6

for (int j = 1; j <= 6; j++) {

fprintf(fp, "8%d1 8%d7 ", j, j);

}

fprintf(fp, "0\n");

for (int j = 1; j <= 6; j++) {

fprintf(fp, "8%d2 8%d7 ", j, j);

}

fprintf(fp, "0\n");

for (int j = 1; j <= 6; j++) {

fprintf(fp, "8%d2 8%d8 ", j, j);

}

fprintf(fp, "0\n");

for (int j = 1; j <= 6; j++) {

fprintf(fp, "8%d3 8%d7 ", j, j);

}

fprintf(fp, "0\n");

for (int j = 1; j <= 6; j++) {

fprintf(fp, "8%d3 8%d8 ", j, j);

}

fprintf(fp, "0\n");

for (int j = 1; j <= 6; j++) {

fprintf(fp, "8%d3 8%d9 ", j, j);

}

fprintf(fp, "0\n");

//第1行 12345 23456 34567 45678 56789

//一定有5 1

for (int j = 1; j <= 5; j++) {

fprintf(fp, "1%d5 ", j);

}

fprintf(fp, "0\n");

//选填 16 26 27 36 37 38 46 47 48 49 10

for (int j = 1; j <= 5; j++) {

fprintf(fp, "1%d1 1%d6 ", j, j);

}

fprintf(fp, "0\n");

for (int j = 1; j <= 5; j++) {

fprintf(fp, "1%d2 1%d6 ", j, j);

}

fprintf(fp, "0\n");

for (int j = 1; j <= 5; j++) {

fprintf(fp, "1%d2 1%d7 ", j, j);

}

fprintf(fp, "0\n");

for (int j = 1; j <= 5; j++) {

fprintf(fp, "1%d3 1%d6 ", j, j);

}

fprintf(fp, "0\n");

for (int j = 1; j <= 5; j++) {

fprintf(fp, "1%d3 1%d7 ", j, j);

}

fprintf(fp, "0\n");

for (int j = 1; j <= 5; j++) {

fprintf(fp, "1%d3 1%d8 ", j, j);

}

fprintf(fp, "0\n");

for (int j = 1; j <= 5; j++) {

fprintf(fp, "1%d4 1%d6 ", j, j);

}

fprintf(fp, "0\n");

for (int j = 1; j <= 5; j++) {

fprintf(fp, "1%d4 1%d7 ", j, j);

}

fprintf(fp, "0\n");

for (int j = 1; j <= 5; j++) {

fprintf(fp, "1%d4 1%d8 ", j, j);

}

fprintf(fp, "0\n");

for (int j = 1; j <= 5; j++) {

fprintf(fp, "1%d4 1%d9 ", j, j);

}

fprintf(fp, "0\n");

//第9行 12345 23456 34567 45678 56789

//一定有5 1

for (int j = 1; j <= 5; j++) {

fprintf(fp, "9%d5 ", j);

}

fprintf(fp, "0\n");

//选填 16 26 27 36 37 38 46 47 48 49 10

for (int j = 1; j <= 5; j++) {

fprintf(fp, "9%d1 9%d6 ", j, j);

}

fprintf(fp, "0\n");

for (int j = 1; j <= 5; j++) {

fprintf(fp, "9%d2 9%d6 ", j, j);

}

fprintf(fp, "0\n");

for (int j = 1; j <= 5; j++) {

fprintf(fp, "9%d2 9%d7 ", j, j);

}

fprintf(fp, "0\n");

for (int j = 1; j <= 5; j++) {

fprintf(fp, "9%d3 9%d6 ", j, j);

}

fprintf(fp, "0\n");

for (int j = 1; j <= 5; j++) {

fprintf(fp, "9%d3 9%d7 ", j, j);

}

fprintf(fp, "0\n");

for (int j = 1; j <= 5; j++) {

fprintf(fp, "9%d3 9%d8 ", j, j);

}

fprintf(fp, "0\n");

for (int j = 1; j <= 5; j++) {

fprintf(fp, "9%d4 9%d6 ", j, j);

}

fprintf(fp, "0\n");

for (int j = 1; j <= 5; j++) {

fprintf(fp, "9%d4 9%d7 ", j, j);

}

fprintf(fp, "0\n");

for (int j = 1; j <= 5; j++) {

fprintf(fp, "9%d4 9%d8 ", j, j);

}

fprintf(fp, "0\n");

for (int j = 1; j <= 5; j++) {

fprintf(fp, "9%d4 9%d9 ", j, j);

}

fprintf(fp, "0\n");

//斜1

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

int j = a[i][0];

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

for (int p = 1; p <= j; p++) {

for (int q = 1; q < p; q++) {

fprintf(fp, "-%d%d -%d%d 0\n", b[i][p], k, b[i][q], k);

}

}

}

}

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

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

fprintf(fp, "%d%d ", b[5][j], k);

}

fprintf(fp, "0\n");

}

//第4行 12345678 23456789

//一定有 2345678 7

for (int k = 2; k <= 8; k++) {

for (int j = 1; j <= 8; j++) {

fprintf(fp, "%d%d ", b[4][j], k);

}

fprintf(fp, "0\n");

}

//选填（1）（9） 2

for (int j = 1; j <= 8; j++) {

fprintf(fp, "%d1 %d9 ", b[4][j], b[4][j]);

}

fprintf(fp, "0\n");

//第6行

for (int k = 2; k <= 8; k++) {

for (int j = 1; j <= 8; j++) {

fprintf(fp, "%d%d ", b[6][j], k);

}

fprintf(fp, "0\n");

}

//选填（1）（9） 2

for (int j = 1; j <= 8; j++) {

fprintf(fp, "%d1 %d9 ", b[6][j], b[6][j]);

}

fprintf(fp, "0\n");

//第3行 一定有34567 5

for (int k = 3; k <= 7; k++) {

for (int j = 1; j <= 7; j++) {

fprintf(fp, "%d%d ", b[3][j], k);

}

fprintf(fp, "0\n");

}

//选填（18） （28） （29） 3

for (int j = 1; j <= 7; j++) {

fprintf(fp, "%d1 %d8 ", b[3][j], b[3][j]);

}

fprintf(fp, "0\n");

for (int j = 1; j <= 7; j++) {

fprintf(fp, "%d2 %d8 ", b[3][j], b[3][j]);

}

fprintf(fp, "0\n");

for (int j = 1; j <= 7; j++) {

fprintf(fp, "%d2 %d9 ", b[3][j], b[3][j]);

}

fprintf(fp, "0\n");

//第7行 一定有34567 5

for (int k = 3; k <= 7; k++) {

for (int j = 1; j <= 7; j++) {

fprintf(fp, "%d%d ", b[7][j], k);

}

fprintf(fp, "0\n");

}

//选填（18） （28） （29） 3

for (int j = 1; j <= 7; j++) {

fprintf(fp, "%d1 %d8 ", b[7][j], b[7][j]);

}

fprintf(fp, "0\n");

for (int j = 1; j <= 7; j++) {

fprintf(fp, "%d2 %d8 ", b[7][j], b[7][j]);

}

fprintf(fp, "0\n");

for (int j = 1; j <= 7; j++) {

fprintf(fp, "%d2 %d9 ", b[7][j], b[7][j]);

}

fprintf(fp, "0\n");

//第2行 123456 234567 345678 456789

//一定有456 3

for (int k = 4; k <= 6; k++) {

for (int j = 1; j <= 6; j++) {

fprintf(fp, "%d%d ", b[2][j], k);

}

fprintf(fp, "0\n");

}

//选填 17 27 28 37 38 39 6

for (int j = 1; j <= 6; j++) {

fprintf(fp, "%d1 %d7 ", b[2][j], b[2][j]);

}

fprintf(fp, "0\n");

for (int j = 1; j <= 6; j++) {

fprintf(fp, "%d2 %d7 ", b[2][j], b[2][j]);

}

fprintf(fp, "0\n");

for (int j = 1; j <= 6; j++) {

fprintf(fp, "%d2 %d8 ", b[2][j], b[2][j]);

}

fprintf(fp, "0\n");

for (int j = 1; j <= 6; j++) {

fprintf(fp, "%d3 %d7 ", b[2][j], b[2][j]);

}

fprintf(fp, "0\n");

for (int j = 1; j <= 6; j++) {

fprintf(fp, "%d3 %d8 ", b[2][j], b[2][j]);

}

fprintf(fp, "0\n");

for (int j = 1; j <= 6; j++) {

fprintf(fp, "%d3 %d9 ", b[2][j], b[2][j]);

}

fprintf(fp, "0\n");

//第8行 123456 234567 345678 456789

//一定有456 3

for (int k = 4; k <= 6; k++) {

for (int j = 1; j <= 6; j++) {

fprintf(fp, "%d%d ", b[8][j], k);

}

fprintf(fp, "0\n");

}

//选填 17 27 28 37 38 39 6

for (int j = 1; j <= 6; j++) {

fprintf(fp, "%d1 %d7 ", b[8][j], b[8][j]);

}

fprintf(fp, "0\n");

for (int j = 1; j <= 6; j++) {

fprintf(fp, "%d2 %d7 ", b[8][j], b[8][j]);

}

fprintf(fp, "0\n");

for (int j = 1; j <= 6; j++) {

fprintf(fp, "%d2 %d8 ", b[8][j], b[8][j]);

}

fprintf(fp, "0\n");

for (int j = 1; j <= 6; j++) {

fprintf(fp, "%d3 %d7 ", b[8][j], b[8][j]);

}

fprintf(fp, "0\n");

for (int j = 1; j <= 6; j++) {

fprintf(fp, "%d3 %d8 ", b[8][j], b[8][j]);

}

fprintf(fp, "0\n");

for (int j = 1; j <= 6; j++) {

fprintf(fp, "%d3 %d9 ", b[8][j], b[8][j]);

}

fprintf(fp, "0\n");

//第1行 12345 23456 34567 45678 56789

//一定有5 1

for (int j = 1; j <= 5; j++) {

fprintf(fp, "%d5 ", b[1][j]);

}

fprintf(fp, "0\n");

//选填 16 26 27 36 37 38 46 47 48 49 10

for (int j = 1; j <= 5; j++) {

fprintf(fp, "%d1 %d6 ", b[1][j], b[1][j]);

}

fprintf(fp, "0\n");

for (int j = 1; j <= 5; j++) {

fprintf(fp, "%d2 %d6 ", b[1][j], b[1][j]);

}

fprintf(fp, "0\n");

for (int j = 1; j <= 5; j++) {

fprintf(fp, "%d2 %d7 ", b[1][j], b[1][j]);

}

fprintf(fp, "0\n");

for (int j = 1; j <= 5; j++) {

fprintf(fp, "%d3 %d6 ", b[1][j], b[1][j]);

}

fprintf(fp, "0\n");

for (int j = 1; j <= 5; j++) {

fprintf(fp, "%d3 %d7 ", b[1][j], b[1][j]);

}

fprintf(fp, "0\n");

for (int j = 1; j <= 5; j++) {

fprintf(fp, "%d3 %d8 ", b[1][j], b[1][j]);

}

fprintf(fp, "0\n");

for (int j = 1; j <= 5; j++) {

fprintf(fp, "%d4 %d6 ", b[1][j], b[1][j]);

}

fprintf(fp, "0\n");

for (int j = 1; j <= 5; j++) {

fprintf(fp, "%d4 %d7 ", b[1][j], b[1][j]);

}

fprintf(fp, "0\n");

for (int j = 1; j <= 5; j++) {

fprintf(fp, "%d4 %d8 ", b[1][j], b[1][j]);

}

fprintf(fp, "0\n");

for (int j = 1; j <= 5; j++) {

fprintf(fp, "%d4 %d9 ", b[1][j], b[1][j]);

}

fprintf(fp, "0\n");

//第9行 12345 23456 34567 45678 56789

//一定有5 1

for (int j = 1; j <= 5; j++) {

fprintf(fp, "%d5 ", b[9][j]);

}

fprintf(fp, "0\n");

//选填 16 26 27 36 37 38 46 47 48 49 10

for (int j = 1; j <= 5; j++) {

fprintf(fp, "%d1 %d6 ", b[9][j], b[9][j]);

}

fprintf(fp, "0\n");

for (int j = 1; j <= 5; j++) {

fprintf(fp, "%d2 %d6 ", b[9][j], b[9][j]);

}

fprintf(fp, "0\n");

for (int j = 1; j <= 5; j++) {

fprintf(fp, "%d2 %d7 ", b[9][j], b[9][j]);

}

fprintf(fp, "0\n");

for (int j = 1; j <= 5; j++) {

fprintf(fp, "%d3 %d6 ", b[9][j], b[9][j]);

}

fprintf(fp, "0\n");

for (int j = 1; j <= 5; j++) {

fprintf(fp, "%d3 %d7 ", b[9][j], b[9][j]);

}

fprintf(fp, "0\n");

for (int j = 1; j <= 5; j++) {

fprintf(fp, "%d3 %d8 ", b[9][j], b[9][j]);

}

fprintf(fp, "0\n");

for (int j = 1; j <= 5; j++) {

fprintf(fp, "%d4 %d6 ", b[9][j], b[9][j]);

}

fprintf(fp, "0\n");

for (int j = 1; j <= 5; j++) {

fprintf(fp, "%d4 %d7 ", b[9][j], b[9][j]);

}

fprintf(fp, "0\n");

for (int j = 1; j <= 5; j++) {

fprintf(fp, "%d4 %d8 ", b[9][j], b[9][j]);

}

fprintf(fp, "0\n");

for (int j = 1; j <= 5; j++) {

fprintf(fp, "%d4 %d9 ", b[9][j], b[9][j]);

}

fprintf(fp, "0\n");

//斜2

int x = 15;

for (int j = 1; j <= 5; j++) {

b[1][j] = x;

x += 11;

}

x = 14;

for (int j = 1; j <= 5; j++) {

b[2][j] = x;

x += 11;

}

b[2][6] = 68;

x = 13;

for (int j = 1; j <= 5; j++) {

b[3][j] = x;

x += 11;

}

b[3][6] = 67; b[3][7] = 77;

x = 12;

for (int j = 1; j <= 5; j++) {

b[4][j] = x;

x += 11;

}

b[4][6] = 66; b[4][7] = 76; b[4][8] = 86;

x = 11;

for (int j = 1; j <= 5; j++) {

b[5][j] = x;

x += 11;

}

x = 65;

for (int j = 6; j <= 9; j++) {

b[5][j] = x;

x += 10;

}

x = 21;

for (int j = 1; j <= 4; j++) {

b[6][j] = x;

x += 11;

}

x = 64;

for (int j = 5; j <= 8; j++) {

b[6][j] = x;

x += 10;

}

x = 31;

for (int j = 1; j <= 3; j++) {

b[7][j] = x;

x += 11;

}

x = 63;

for (int j = 4; j <= 7; j++) {

b[7][j] = x;

x += 10;

}

x = 41;

for (int j = 1; j <= 2; j++) {

b[8][j] = x;

x += 11;

}

x = 62;

for (int j = 3; j <= 6; j++) {

b[8][j] = x;

x += 10;

}

x = 51;

for (int j = 1; j <= 5; j++) {

b[9][j] = x;

x += 10;

}

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

int j = a[i][0];

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

for (int p = 1; p <= j; p++) {

for (int q = 1; q < p; q++) {

fprintf(fp, "-%d%d -%d%d 0\n", b[i][p], k, b[i][q], k);

}

}

}

}

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

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

fprintf(fp, "%d%d ", b[5][j], k);

}

fprintf(fp, "0\n");

}

//第4行 12345678 23456789

//一定有 2345678 7

for (int k = 2; k <= 8; k++) {

for (int j = 1; j <= 8; j++) {

fprintf(fp, "%d%d ", b[4][j], k);

}

fprintf(fp, "0\n");

}

//选填（1）（9） 2

for (int j = 1; j <= 8; j++) {

fprintf(fp, "%d1 %d9 ", b[4][j], b[4][j]);

}

fprintf(fp, "0\n");

//第6行

for (int k = 2; k <= 8; k++) {

for (int j = 1; j <= 8; j++) {

fprintf(fp, "%d%d ", b[6][j], k);

}

fprintf(fp, "0\n");

}

//选填（1）（9） 2

for (int j = 1; j <= 8; j++) {

fprintf(fp, "%d1 %d9 ", b[6][j], b[6][j]);

}

fprintf(fp, "0\n");

//第3行 一定有34567 5

for (int k = 3; k <= 7; k++) {

for (int j = 1; j <= 7; j++) {

fprintf(fp, "%d%d ", b[3][j], k);

}

fprintf(fp, "0\n");

}

//选填（18） （28） （29） 3

for (int j = 1; j <= 7; j++) {

fprintf(fp, "%d1 %d8 ", b[3][j], b[3][j]);

}

fprintf(fp, "0\n");

for (int j = 1; j <= 7; j++) {

fprintf(fp, "%d2 %d8 ", b[3][j], b[3][j]);

}

fprintf(fp, "0\n");

for (int j = 1; j <= 7; j++) {

fprintf(fp, "%d2 %d9 ", b[3][j], b[3][j]);

}

fprintf(fp, "0\n");

//第7行 一定有34567 5

for (int k = 3; k <= 7; k++) {

for (int j = 1; j <= 7; j++) {

fprintf(fp, "%d%d ", b[7][j], k);

}

fprintf(fp, "0\n");

}

//选填（18） （28） （29） 3

for (int j = 1; j <= 7; j++) {

fprintf(fp, "%d1 %d8 ", b[7][j], b[7][j]);

}

fprintf(fp, "0\n");

for (int j = 1; j <= 7; j++) {

fprintf(fp, "%d2 %d8 ", b[7][j], b[7][j]);

}

fprintf(fp, "0\n");

for (int j = 1; j <= 7; j++) {

fprintf(fp, "%d2 %d9 ", b[7][j], b[7][j]);

}

fprintf(fp, "0\n");

//第2行 123456 234567 345678 456789

//一定有456 3

for (int k = 4; k <= 6; k++) {

for (int j = 1; j <= 6; j++) {

fprintf(fp, "%d%d ", b[2][j], k);

}

fprintf(fp, "0\n");

}

//选填 17 27 28 37 38 39 6

for (int j = 1; j <= 6; j++) {

fprintf(fp, "%d1 %d7 ", b[2][j], b[2][j]);

}

fprintf(fp, "0\n");

for (int j = 1; j <= 6; j++) {

fprintf(fp, "%d2 %d7 ", b[2][j], b[2][j]);

}

fprintf(fp, "0\n");

for (int j = 1; j <= 6; j++) {

fprintf(fp, "%d2 %d8 ", b[2][j], b[2][j]);

}

fprintf(fp, "0\n");

for (int j = 1; j <= 6; j++) {

fprintf(fp, "%d3 %d7 ", b[2][j], b[2][j]);

}

fprintf(fp, "0\n");

for (int j = 1; j <= 6; j++) {

fprintf(fp, "%d3 %d8 ", b[2][j], b[2][j]);

}

fprintf(fp, "0\n");

for (int j = 1; j <= 6; j++) {

fprintf(fp, "%d3 %d9 ", b[2][j], b[2][j]);

}

fprintf(fp, "0\n");

//第8行 123456 234567 345678 456789

//一定有456 3

for (int k = 4; k <= 6; k++) {

for (int j = 1; j <= 6; j++) {

fprintf(fp, "%d%d ", b[8][j], k);

}

fprintf(fp, "0\n");

}

//选填 17 27 28 37 38 39 6

for (int j = 1; j <= 6; j++) {

fprintf(fp, "%d1 %d7 ", b[8][j], b[8][j]);

}

fprintf(fp, "0\n");

for (int j = 1; j <= 6; j++) {

fprintf(fp, "%d2 %d7 ", b[8][j], b[8][j]);

}

fprintf(fp, "0\n");

for (int j = 1; j <= 6; j++) {

fprintf(fp, "%d2 %d8 ", b[8][j], b[8][j]);

}

fprintf(fp, "0\n");

for (int j = 1; j <= 6; j++) {

fprintf(fp, "%d3 %d7 ", b[8][j], b[8][j]);

}

fprintf(fp, "0\n");

for (int j = 1; j <= 6; j++) {

fprintf(fp, "%d3 %d8 ", b[8][j], b[8][j]);

}

fprintf(fp, "0\n");

for (int j = 1; j <= 6; j++) {

fprintf(fp, "%d3 %d9 ", b[8][j], b[8][j]);

}

fprintf(fp, "0\n");

//第1行 12345 23456 34567 45678 56789

//一定有5 1

for (int j = 1; j <= 5; j++) {

fprintf(fp, "%d5 ", b[1][j]);

}

fprintf(fp, "0\n");

//选填 16 26 27 36 37 38 46 47 48 49 10

for (int j = 1; j <= 5; j++) {

fprintf(fp, "%d1 %d6 ", b[1][j], b[1][j]);

}

fprintf(fp, "0\n");

for (int j = 1; j <= 5; j++) {

fprintf(fp, "%d2 %d6 ", b[1][j], b[1][j]);

}

fprintf(fp, "0\n");

for (int j = 1; j <= 5; j++) {

fprintf(fp, "%d2 %d7 ", b[1][j], b[1][j]);

}

fprintf(fp, "0\n");

for (int j = 1; j <= 5; j++) {

fprintf(fp, "%d3 %d6 ", b[1][j], b[1][j]);

}

fprintf(fp, "0\n");

for (int j = 1; j <= 5; j++) {

fprintf(fp, "%d3 %d7 ", b[1][j], b[1][j]);

}

fprintf(fp, "0\n");

for (int j = 1; j <= 5; j++) {

fprintf(fp, "%d3 %d8 ", b[1][j], b[1][j]);

}

fprintf(fp, "0\n");

for (int j = 1; j <= 5; j++) {

fprintf(fp, "%d4 %d6 ", b[1][j], b[1][j]);

}

fprintf(fp, "0\n");

for (int j = 1; j <= 5; j++) {

fprintf(fp, "%d4 %d7 ", b[1][j], b[1][j]);

}

fprintf(fp, "0\n");

for (int j = 1; j <= 5; j++) {

fprintf(fp, "%d4 %d8 ", b[1][j], b[1][j]);

}

fprintf(fp, "0\n");

for (int j = 1; j <= 5; j++) {

fprintf(fp, "%d4 %d9 ", b[1][j], b[1][j]);

}

fprintf(fp, "0\n");

//第9行 12345 23456 34567 45678 56789

//一定有5 1

for (int j = 1; j <= 5; j++) {

fprintf(fp, "%d5 ", b[9][j]);

}

fprintf(fp, "0\n");

//选填 16 26 27 36 37 38 46 47 48 49 10

for (int j = 1; j <= 5; j++) {

fprintf(fp, "%d1 %d6 ", b[9][j], b[9][j]);

}

fprintf(fp, "0\n");

for (int j = 1; j <= 5; j++) {

fprintf(fp, "%d2 %d6 ", b[9][j], b[9][j]);

}

fprintf(fp, "0\n");

for (int j = 1; j <= 5; j++) {

fprintf(fp, "%d2 %d7 ", b[9][j], b[9][j]);

}

fprintf(fp, "0\n");

for (int j = 1; j <= 5; j++) {

fprintf(fp, "%d3 %d6 ", b[9][j], b[9][j]);

}

fprintf(fp, "0\n");

for (int j = 1; j <= 5; j++) {

fprintf(fp, "%d3 %d7 ", b[9][j], b[9][j]);

}

fprintf(fp, "0\n");

for (int j = 1; j <= 5; j++) {

fprintf(fp, "%d3 %d8 ", b[9][j], b[9][j]);

}

fprintf(fp, "0\n");

for (int j = 1; j <= 5; j++) {

fprintf(fp, "%d4 %d6 ", b[9][j], b[9][j]);

}

fprintf(fp, "0\n");

for (int j = 1; j <= 5; j++) {

fprintf(fp, "%d4 %d7 ", b[9][j], b[9][j]);

}

fprintf(fp, "0\n");

for (int j = 1; j <= 5; j++) {

fprintf(fp, "%d4 %d8 ", b[9][j], b[9][j]);

}

fprintf(fp, "0\n");

for (int j = 1; j <= 5; j++) {

fprintf(fp, "%d4 %d9 ", b[9][j], b[9][j]);

}

fprintf(fp, "0\n");

fclose(fp);

return 1;

}

void Sudoku() {

ClauseList\* S = NULL;

Node x;

x.next = NULL;

int op=1,ddd=0;

char filename[300];

Su[1][0] = 5;

Su[2][0] = 6;

Su[3][0] = 7;

Su[4][0] = 8;

Su[5][0] = 9;

Su[6][0] = 8;

Su[7][0] = 7;

Su[8][0] = 6;

Su[9][0] = 5;

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

for (int j = 1; j <= Su[i][0]; j++) {

Su[i][j] = 0;

}

}

while (op) {

system("cls");

printf("\n\n\n");//菜单

printf("\t\t\t Sudoku问题 \n");

printf("\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("\t\t1. 初始化 2.求解 3.写Sudokucnf\n");

printf("\t\t0. Exit\n");

printf("\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("\t\tPlease choose your operation[0--3]:\n");

scanf("%d", &op);

switch (op) {

case 1:

printf("请输入文件名：\n");

scanf("%s", filename);

if (ddd = First(filename)) {

printf("OK!\n");

Tu();

}

else {

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

for (int j = 1; j <= Su[i][0]; j++) {

Su[i][j] = 0;

}

}

printf("Fail!");

}

getchar(); getchar();

break;

case 2:

ddd = ReadSudoku(&S);

for (int i = 1; i <= MaxNumVar; i++) {

Varjl[i] = Unknow;

}

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

for (int j = 1; j <= Su[i][0]; j++) {

if (Su[i][j] != 0) {

x.data = Su[i][j] + 10 \* j + 100 \* i;

x.negated = 1;

//printf("%d已经添加\n", x.data);

Add(&S, x);

}

}

}

ddd = DPLL(&S);

if (ddd == 0) {

printf("无解！\n");

}

else {

printf("success!\n");

for (int k = 111; k <= 1000; k++) {

if (Varjl[k] == True) {

int p, q, r;

r = k % 10;

q = (k / 10) % 10;

p = k / 100;

Su[p][q] = r;

}

}

Tu();

}

getchar(); getchar();

break;

case 3:

ddd = ToCnf(Su);

if (ddd == 1) {

printf("OK!");

}

getchar(); getchar();

break;

case 0:

exit(0);

default:

printf("\t\t\tPlease choose your operation again[0--1]:\n");

scanf("%d", &op);

}

}

return ;

}

int First(char\* filename) {

FILE\* fp = fopen(filename, "r");

if (fp == NULL) {

printf("ERROR!\n");

return 0;

}

int a, b, c;

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

while (a != 0) {

Su[a][b] = c;

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

}

fclose(fp);

return 1;

}

void Tu() {

int x;

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

x = 9 - Su[i][0];

if (i <= 5) {

for (int k = 0; k < x; k++) printf(" ");

}

else {

for (int k = 1; k < x; k++) printf(" ");

}

if (i <= 5) {

for (int k = 0; k < Su[i][0]; k++) {

printf(" / \\");

}

printf("\n");

}

else {

for (int k = 0; k <= Su[i][0]; k++) {

printf(" \\ /");

}

printf("\n");

}

for (int k = 0; k < x; k++) printf(" ");

for (int j = 1; j <= Su[i][0]; j++) {

if (Su[i][j] == 0) {

printf("| . ");

}

else {

printf("| %d ",Su[i][j]);

}

}

printf("|\n");

}

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

}

int ReadSudoku(ClauseList\*\* s) {

FILE\* fp = fopen("Sudoku.cnf", "r");

if (fp == NULL) {

printf("ERROR!\n");

return 0;

}

int aa, bb;

int num = 1, m;

char c;

//CnfSign k;

ClauseList\* h = NULL;

ClauseList\* lp = NULL;

Node\* hh = NULL;

Node\* pp = NULL;

fscanf(fp, "%c", &c);

while (c == 'c') {//注释行

while (c != '\n' && c != '\r') {

fscanf(fp, "%c", &c);

}

if (c == '\n') fscanf(fp, "%c", &c);

}

if (c == 'p') {//读p行

fscanf(fp, " cnf %d %d", &aa, &bb);

}

//二维尾插法读入nc行数据

h = (ClauseList\*)malloc(sizeof(ClauseList));

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

if (m == 0) {

free(h);

return 0;

}

hh = (Node\*)malloc(sizeof(Node));

if (m > 0) {

hh->data = abs(m);

hh->negated = 1;

}

if (m < 0) {

hh->data = abs(m);

hh->negated = 0;

}

hh->next = NULL;

pp = hh;

while (1) {

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

if (m == 0) break;

pp->next = (Node\*)malloc(sizeof(Node));

pp = pp->next;

if (m > 0) {

pp->data = abs(m);

pp->negated = 1;

}

if (m < 0) {

pp->data = abs(m);

pp->negated = 0;

}

pp->next = NULL;

num++;

}

h->w = num;

h->p = hh;

h->next = NULL;

lp = h;

for (int i = 1; i < bb; i++) {

num = 1;

lp->next = (ClauseList\*)malloc(sizeof(ClauseList));

lp = lp->next;

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

hh = (Node\*)malloc(sizeof(Node));

if (m > 0) {

hh->data = abs(m);

hh->negated = 1;

}

if (m < 0) {

hh->data = abs(m);

hh->negated = 0;

}

hh->next = NULL;

pp = hh;

while (1) {

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

if (m == 0) break;

pp->next = (Node\*)malloc(sizeof(Node));

pp = pp->next;

if (m > 0) {

pp->data = abs(m);

pp->negated = 1;

}

if (m < 0) {

pp->data = abs(m);

pp->negated = 0;

}

pp->next = NULL;

num++;

}

lp->w = num;

lp->p = hh;

lp->next = NULL;

}

\*s = h;

fclose(fp);

return 1;

}