華中科技大學

课程设计报告

题目：基于 SAT 的双数独游戏求解程序

课程名称：程序设计综合课程设计
专业班级：CS2108
学　　号：I202120037
姓　　名：郑澍频
指导教师：袁凌老师
报告日期：2022 年 9 月 30 日

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：$[(t-t_o)/t]*100\%$,其中 t 为未对 DPLL 优化时求解基准算例的执行时间，t_o 则为优化 DPLL 实现时求解同一算例的执行时间。(15%)

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

口 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

目录

任务书 I

1 引言 1

1.1 课题背景与意义 1

1.2 国内外研究现状 1

1.3 课程设计的主要研究工作 2

2 系统需求分析与总体设计 3

2.1 系统需求分析 3

2.2 系统总体设计 3

3 系统详细设计 4

3.1 有关数据结构的定义 4

3.2 主要算法设计 4

4 系统实现与测试 5

4.1 系统实现 5

4.2 系统测试 5

5 总结与展望 11

5.1 全文总结 11

5.2 工作展望 11

6 体会 12

参考文献 13

附录 基于 SAT 的双数独游戏求解程序的源程序 14

1 引言

1.1 课题背景与意义

在计算机科学中，布尔可满足性问题（有时称为命题可满足性问题，缩写为 SATISFIABILITY 或 SAT）是确定是否存在满足给定布尔公式的解释的问题。SAT 是第一个已知的 NP 完全问题，1971 年多伦多大学的 Stephen Cook 和 1973 年国家科学院的 Leonid Levin 独立证明了这一问题。在此之前，NP 完全问题的概念甚至不存在。SAT 问题在计算机科学、复杂性理论、密码系统、人工智能等领域发挥着至关重要的作用。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。

1.2 国内外研究现状

自 1960 年 Davis 和 Putnam 提出 DP 算法以来，SAT 求解研究逐步受到关注。然而 1971 年 Cook 证明 SAT 问题是 NPC 之后，人们对 SAT 的重视程度减弱。后来人们对 SAT 问题有了新的认识。自 1991 年起，世界各国研究机构纷纷举办 SAT 竞赛，众多学者的研究热情空前高涨，SAT 算法及其实现程序的求解效率大幅提高，SAT 问题逐渐在许多实际应用中显现出强大的作用。SAT 协会是目前推动 SAT 问题理论和应用进展的主要驱动力量，其 Satlive 网站随时更新 SAT 研究动态，发布了一系列有关会议、竞赛、技术报告、论文、图书等信息；每年举办一次 SAT 理论和应用国际学术会议，目前已召开 16 届；SAT 国际竞赛始于 2002 年，每隔两年或一年举办，2016 年成功举行了第 10 届，汇聚了大批优秀的 SAT 求解器，影响力很大。

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。

1.3 课程设计的主要研究工作

依据 Davis 和 Putnam 在 1960 年提出的 DPLL 算法来求解合取范式，并基于原始的 DPLL 算法提出优化方案，并对优化前后的 DPLL 算法进行比较，做出一些思考。将数独游戏问题转化为 SAT 问题，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。

2 系统需求分析与总体设计

2.1 系统需求分析

基于 DPLL 算法设计高效 SAT 求解器，对给定的中小规模 SAT 问题实例，理论上可判定其是否满足，满足时给出对应的一组解，并统计求解时间。要求具有输入输出功能，公式解析与验证，DPLL 过程，时间性能的测量，程序优化，SAT 应用（数独）等需求。

2.2 系统总体设计

基于 DPLL 过程实现一个高效 SAT 求解器，对于给定的中小规模算例进行求解，输出求解结果，统计求解时间。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。
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。
3. DPLL 过程：基于 DPLL 算法框架，实现 SAT 算例的求解。
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. 程序优化：对基本 DPLL 的实现进行存储结构、分支变元选取策略等某一方面进行优化设计与实现，提供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。
6. SAT 应用：将二进制数独游戏问题转化为 SAT 问题，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。

3 系统详细设计

3.1 有关数据结构的定义

十字链表结构体定义如下：

```
typedef struct SATNode {
    int data;  //数据域
    SATNode* next;
} SATNode;
typedef struct SATList {
    SATNode* head;  //表头
    SATList* next;
} SATList;
```

3.2 主要算法设计

```
status ReadFile(SATList*& cnf);
status SAT(void);
status Sudoku(void);
void destroyClause(SATList*& cnf);
status isUnitClause(SATNode* cnf);
status removeClause(SATList*& cnf, SATList*& root);
status removeNode(SATNode*& cnf, SATNode*& head);
status addClause(SATList* cnf, SATList*& root);
status isemptyClause(SATList* cnf);
status DPLL(SATList*& cnf, int value[]);
void CopyClause(SATList*& a, SATList* b);
status SaveFile(int result, double time, int value[]);
void CreateBinary(void);
void tocnf(char* fileName, int hole);
void print(int a[][9]);
int DFSCreate(int x, int y, int sudoku[][9]);
void CreateSudoku(void);
void Dighole1(int hole);
void Dighole2(int hole);
int Save_sudoku(char* filename, int* val);
void play(int flag, int* val);
status Create(struct SATList*& cnf);
int DPLL0(SATList*& cnf, int value[]);
```

4 系统实现与测试

4.1 系统实现

实验环境为 Windows 10，代码采用编辑器 Visual Studio 2022 编写。

文件说明：cxsj.cpp：整个系统构建和用户操作实现

4.2 系统测试

主控流程：

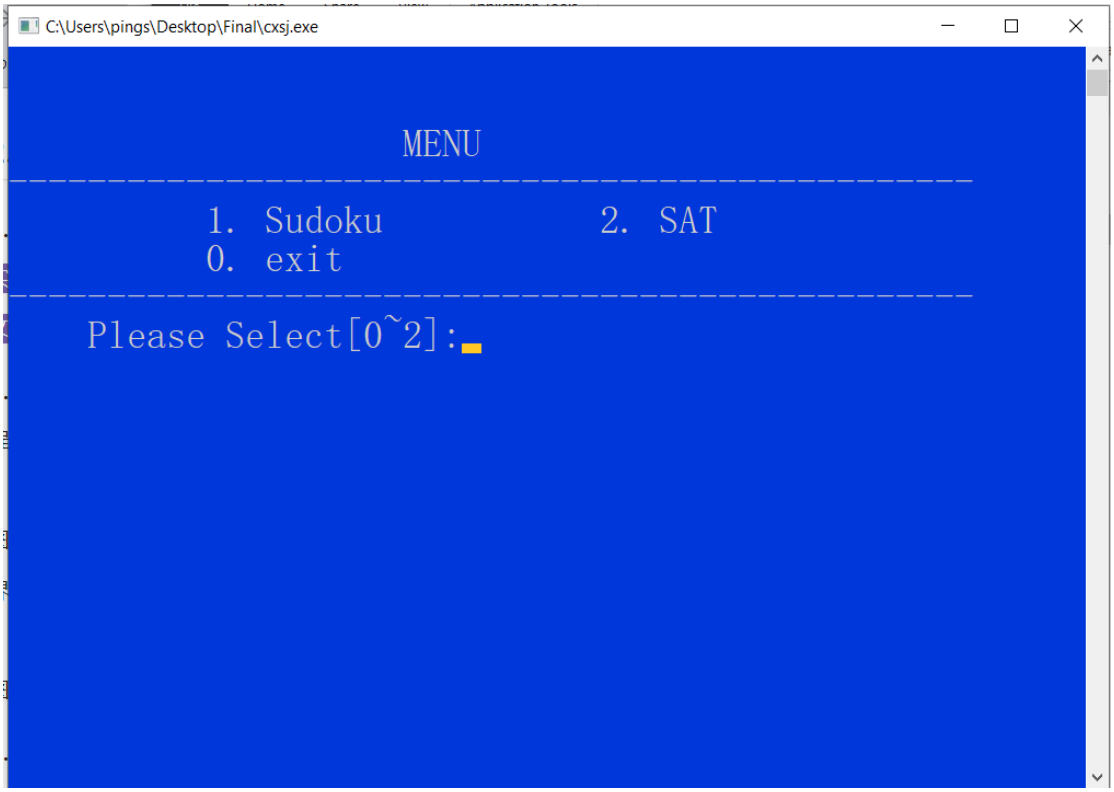

图 4.1 Main Menu

SAT 部分

```
                    SAT MENU
------------------------------------------------------------
          1.Read cnf File                       2.Transverse clause
          3.Solve(improved) and Save cnf        4.Solve(not improved)
          0.exit
------------------------------------------------
    Please Select[0~4]:4
Result: 1
1 -2 -3 -4 -5 -6 -7 -8 -9 -10 -11 -12 -13 -14 -15 -16 -17 -18 -19 20 -21
22 -23 -24 -25 -26 -27 -28 -29 -30 -31 -32 -33 -34 -35 -36 -37 -38 39 -40
 -41 -42 43 -44 -45 -46 -47 -48 -49 -50 -51 -52 -53 -54 -55 -56 -57 58 -5
9 -60 -61 -62 -63 64 -65 -66 -67 -68 -69 -70 -71 -72 -73 -74 -75 -76 77 -
78 -79 -80 -81 -82 -83 -84 85 -86 -87 -88 -89 -90 -91 -92 -93 -94 -95 96
-97 -98 -99 -100 -101 -102 -103 -104 -105 -106 -107 -108 109 -110 -111 -1
12 -113 -114 -115 -116 117 -118 -119 -120 -121 -122 -123 -124 125 -126 -1
27 -128 -129 -130 -131 -132 133 -134 -135 -136 -137 -138 -139 -140 141 -1
42 -143 -144 -145 -146 -147 -148 149 -150 -151 -152 -153 -154 -155 -156 1
57 -158 -159 -160 -161 -162 -163 -164 165 -166 -167 -168 -169 -170 -171 -
172 173 -174 -175 -176 -177 -178 -179 -180 -181
Operation Time=1630.000000ms
Results are saved to a .res file with the same name
```

图 4.2 ais10.cnf 未优化运行结果

```
                    SAT MENU
------------------------------------------------------------
          1.Read cnf File                       2.Transverse clause
          3.Solve(improved) and Save cnf        4.Solve(not improved)
          0.exit
------------------------------------------------
    Please Select[0~4]:3
Result: 1
-1 -2 -3 -4 5 -6 -7 -8 -9 -10 -11 -12 -13 -14 -15 16 -17 -18 -19 -20 -21
-22 -23 24 -25 -26 -27 -28 -29 -30 -31 -32 -33 -34 -35 -36 37 -38 -39 -40
 -41 -42 43 -44 -45 -46 -47 -48 -49 -50 -51 -52 -53 -54 -55 -56 -57 58 -5
9 -60 -61 62 -63 -64 -65 -66 -67 -68 -69 -70 -71 -72 -73 -74 -75 -76 -77
-78 79 -80 81 -82 -83 -84 -85 -86 -87 -88 -89 -90 -91 -92 -93 -94 -95 -96
 -97 -98 -99 100 101 -102 -103 -104 -105 -106 -107 -108 -109 -110 111 -11
2 -113 -114 -115 -116 -117 -118 -119 -120 121 -122 -123 -124 -125 -126 -1
27 -128 -129 -130 131 -132 -133 -134 -135 -136 -137 -138 -139 -140 141 -1
42 -143 -144 -145 -146 -147 -148 -149 -150 151 -152 -153 -154 -155 -156 -
157 -158 -159 -160 161 -162 -163 -164 -165 -166 -167 -168 -169 -170 171 -
172 -173 -174 -175 -176 -177 -178 -179 -180 181
Operation time=42.000000ms
Results are saved to a .txt file with the same namet
```

图 4.3 ais10.cnf 优化运行结果

```
    Please Select[0~4]:4
Result: 1
1 -2 -3 -4 -5 6 -7 -8 -9 -10 -11 12 -13 -14 15 -16 -17 -18 19 -20 -21 -22 -23 -2
4 25 -26 -27 -28 29 30 -31 -32 -33 -34 -35 36 -37 -38 39 -40 -41 -42 43 -44 -45
-46 -47 -48 -49 -50 51 -52 53 -54 -55 -56 -57 -58 -59 -60 -61 62 -63 -64 65 -66
-67 -68 -69 -70 -71 -72 73 -74 -75 -76 77 -78 -79 -80 -81 -82 -83 -84 85 86 -87
-88 -89 -90 -91 -92 -93 94 -95 -96 -97 -98 -99 100 -101 -102 -103 -104 -105 -106
 -107 -108 109 -110 -111 -112 -113 114 -115 -116 117 -118 -119 -120 -121 -122 12
3 -124 -125 126 -127 -128 -129 -130 -131 132 -133 -134 135 -136 137 -138 -139 -1
40 141 -142 -143 -144 145 -146 147 -148 -149 -150 -151 -152 -153 154 -155 -156 -
157 -158 -159 -160 161 -162 163 -164 -165 -166 -167 -168 -169 170 -171 -172 173
-174 -175 -176 -177 178 -179 -180 -181 -182 183 -184 -185 -186 -187 -188 -189 -1
90 -191 192 -193 -194 195 -196 -197 198 -199 -200 -201 -202 -203 -204 -205 206 -
207 -208 209 -210 -211 -212 -213 214 -215 -216 -217 -218 219 -220 -221 -222 -223
 224 -225 226 -227 -228 229 -230 -231 -232 233 -234 -235 -236 -237 238 -239 -240
 -241 -242 -243 -244 245 -246 247 -248 249 -250 -251 -252 -253 -254 255 -256 -25
7 258 -259 -260 -261 -262 -263 -264 -265 266 -267 -268 -269 -270 271 -272 -273 -
274 275 -276 -277 -278 -279 -280 -281 -282 283 284 -285 -286 -287 -288 -289 290
-291 -292 -293 -294 -295 296 -297 -298 -299 300 -301 -302 -303
Operation Time=111.000000ms
Results are saved to a .res file with the same name
```

图 4.4 sud00009.cnf 未优化运行结果

```
    Please Select[0~4]:3
Result: 1
1 -2 -3 -4 -5 6 -7 -8 -9 -10 -11 12 -13 -14 15 -16 -17 -18 19 -20 -21 -22 -23 -2
4 25 -26 -27 -28 29 30 -31 -32 -33 -34 -35 36 -37 -38 39 -40 -41 -42 43 -44 -45
-46 -47 -48 -49 -50 51 -52 53 -54 -55 -56 -57 -58 -59 -60 -61 62 -63 -64 65 -66
-67 -68 -69 -70 -71 -72 73 -74 -75 -76 77 -78 -79 -80 -81 -82 -83 -84 85 86 -87
-88 -89 -90 -91 -92 -93 94 -95 -96 -97 -98 -99 100 -101 -102 -103 -104 -105 -106
 -107 -108 109 -110 -111 -112 -113 114 -115 -116 117 -118 -119 -120 -121 -122 12
3 -124 -125 126 -127 -128 -129 -130 -131 132 -133 -134 135 -136 137 -138 -139 -1
40 141 -142 -143 -144 145 -146 147 -148 -149 -150 -151 -152 -153 154 -155 -156 -
157 -158 -159 -160 161 -162 163 -164 -165 -166 -167 -168 -169 170 -171 -172 173
-174 -175 -176 -177 178 -179 -180 -181 -182 183 -184 -185 -186 -187 -188 -189 -1
90 -191 192 -193 -194 195 -196 -197 198 -199 -200 -201 -202 -203 -204 -205 206 -
207 -208 209 -210 -211 -212 -213 214 -215 -216 -217 -218 219 -220 -221 -222 -223
 224 -225 226 -227 -228 229 -230 -231 -232 233 -234 -235 -236 -237 238 -239 -240
 -241 -242 -243 -244 245 -246 247 -248 249 -250 -251 -252 -253 -254 255 -256 -25
7 258 -259 -260 -261 -262 -263 -264 -265 266 -267 -268 -269 -270 271 -272 -273 -
274 275 -276 -277 -278 -279 -280 -281 -282 283 284 -285 -286 -287 -288 -289 290
-291 -292 -293 -294 -295 296 -297 -298 -299 300 -301 -302 -303
Operation time=93.000000ms
Results are saved to a .txt file with the same namet
```

图 4.5 sud00009.cnf 优化运行结果

运行结**果**

求解文件	求解结果	求解时间
7cnf20_90000_90000_7.shuffled-20.cnf；（可满足算例 S 级）	1	44ms
problem1-20.cnf；（可满足算例 S 级）	1	1ms
problem2-50.cnf；（可满足算例 S 级）	1	14ms
problem3-100.cnf；（可满足算例 S 级）	1	2426ms
problem6-50.cnf；（可满足算例 S 级）	1	204ms
sud00012.cnf；（可满足算例 M 级）	1	213ms
sud00021.cnf；（可满足算例 M 级）	1	1523ms
sud00079.cnf；（可满足算例 M 级）	1	91ms
sud00082.cnf；（可满足算例 M 级）	1	250.ms
sud00861.cnf；（可满足算例 M 级）	1	46ms
bart17.shuffled-231.cnf；（可满足算例 M 级）	1	50ms
sud00009.cnf；（可满足算例 M 级）	1	207ms
eh-dp04s04.shuffled-1075.cnf；（可满足算例 L 级）	1	7184ms
u-problem7-50.cnf；（不满足算例）	0	785ms
tst_v10_c100.cnf；（不满足算例）	0	0.0000ms
php-010-008.shuffled-as.sat05-1171.cnf；（不满足算例）	0	13266ms
u-5cnf_3500_3500_30f1.shuffled-30.cnf；（不满足算例）	0	295ms
ais6.cnf	1	3ms
ais8.cnf	1	23ms
ais10.cnf	1	96ms
ais12.cnf	1	372ms

数独部分

```
                    Twodoku
------------------------------------------------
          1.Twodoku
          0. exit
------------------------------------------------
    Please Select[0~1]:1
Sudoku final game is generated!
Please enter the number of spaces in the top left Sudoku
1
Please enter the number of spaces in the bottom right Sudoku
0
top left sudoku
4 1 5 9 2 6 3 8 7
2 3 6 1 7 0 4 5 9
7 8 9 3 4 5 1 2 6
1 2 3 4 5 7 6 9 8
5 4 7 6 8 9 2 1 3
6 9 8 2 1 3 5 7 4
3 5 1 7 9 4 8 6 2
8 7 4 5 6 2 9 3 1
9 6 2 8 3 1 7 4 5

bottom right sudoku
8 6 2 1 3 4 5 7 9
9 3 1 2 5 7 4 6 8
7 4 5 6 8 9 1 2 3
1 2 3 4 6 5 8 9 7
4 5 7 3 9 8 2 1 6
6 8 9 7 1 2 3 4 5
2 1 6 5 7 3 9 8 4
3 7 8 9 4 1 6 5 2
5 9 4 8 2 6 7 3 1
Please enter the required cnf file name123
Suceessfully created cnf, FileName is 123
Double Sudoku solved successfully
Operation hours1346.000000ms
The solution is saved successfully, the file name is 123
Do you want to start the game? YES or NO
```

图 4.6 TWODOKU 运行结果 1

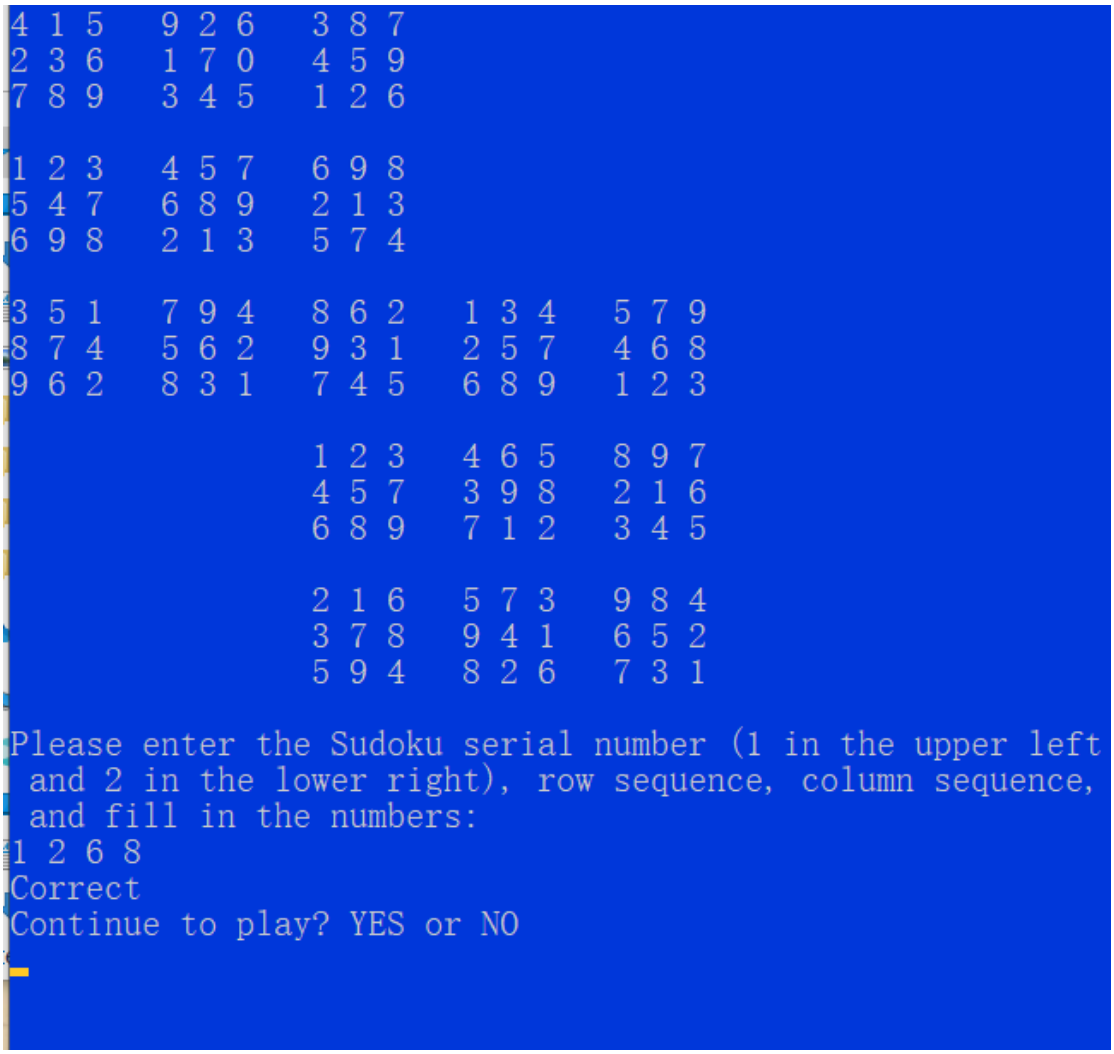

图 4.6 TWODOKU 运行结果 2

结束页面

图 4.7 结束页面

5 总结与展望

5.1 全文总结

前期因语言问题，很难完全理解题目需求，上网查资料也因为文字过多无法集中仔细的阅读整篇文章，导致之后的一切进展缓慢。在正式编写的过程中，多次向同学求助都得到了很积极友善的回答以及实用性极强的帮助，非常感谢。本次程序中和设计完成度属于中下，通过同学们的帮助完成了基本要求，但也耗费了不少时间去了解问题，也发现了自己很多的不足，日后一定努力改进。

5.2 工作展望

在今后的研究中，围绕着如下几个方面开展工作

（1）技术相关的查找与阅读

（2）学会概括，提炼精华

（3）更努力的学习，改善对算法以及编程的理解太浅和知识储备严重不足的问题

（4）规范代码编写，注释有助于对代码的理解

6 体会

在之前的学习中，我们学习了 C 语言程序设计，数据结构，以及其相关实践课程。虽然学院安排的课程为我们奠下了一定的基础，但这次的程序设计综合课程更像游戏中升了级的怪，除了基础知识，更考验学生的实践能力，搜索能力，以及自学能力。

通过这次的综合课程设计，我意识到，数据结构影响着一整个程序的稳定性。在学习上除了对语言的重视，也应该同样重视数据结构的理解。没了数据结构的支撑，程序就会像没有地基的房子一样，无法稳定的运行。

这次的课也让我明白自己的短处在哪，之后该往哪里努力，进步的空间还有很大，希望之后的自己会一次一次的变得更好。

参考文献

im8888. (2020, 08 30). *百 度* . Retrieved from 知 乎 : https://zhuanlan.zhihu.com/p/206465770

Reilly, O. (n.d.). *Chapter 18: Multi-Grid Sudokus*. Retrieved from https://learning.oreilly.com/library/view/sudoku-programming-with/9781484209950/9781484209967_Ch18.xhtml#Sec5

Wikipedia. (2022, 6 22). Retrieved from https://en.wikipedia.org/wiki/DPLL_algorithm

严蔚敏 吴伟民. (2019). *数据结构（C 语言版）*. 北京: 清华大学出版社.

附录

```
#include<stdio.h>
#include<stdlib.h>
#include <string.h>
#include <time.h>
#include <math.h>

#define TRUE 1
#define OK 1
#define FALSE 0
#define ERROR 0
#define INFEASIBLE -1
#define INCREASEMENT 100
#define CLOCKS_PRE_SEC 1000
#define row 9
#define col 9

typedef int status;
int boolCount;  //布尔变元数量
int clauseCount;  //子句数量
char fileName[100]; //文件名
int sudoku1[9][9];
int sudoku2[9][9];
int start1[9][9];
int start2[9][9];
//十字链表结构体
typedef struct SATNode {
    int data;  //数据域
    SATNode* next;
} SATNode;
typedef struct SATList {
    SATNode* head;  //表头
    SATList* next;
} SATList;
```

```
//函数声明
status ReadFile(SATList*& cnf);
status SAT(void);
status Sudoku(void);
void destroyClause(SATList*& cnf);
status isUnitClause(SATNode* cnf);
status removeClause(SATList*& cnf, SATList*& root);
status removeNode(SATNode*& cnf, SATNode*& head);
status addClause(SATList* cnf, SATList*& root);
status isemptyClause(SATList* cnf);
status DPLL(SATList*& cnf, int value[]);
void CopyClause(SATList*& a, SATList* b);
status SaveFile(int result, double time, int value[]);
void CreateBinary(void);
void tocnf(char* fileName, int hole);
void print(int a[][9]);
int DFSCreate(int x, int y, int sudoku[][9]);
void CreateSudoku(void);
void Dighole1(int hole);
void Dighole2(int hole);
int Save_sudoku(char* filename, int* val);
void play(int flag, int* val);
status Create(struct SATList*& cnf);
int DPLL0(SATList*& cnf, int value[]);

int main() {
    int op = 1;
    while (op) {
        printf("\n\n");
        printf("
Ö÷²Ëµ¥                          \n");

printf("-------------------------------------------------\n");
        printf("          1. Sudoku           2. SAT\n");
        printf("          0. exit              \n");
```

```
printf("-----------------------------------------------\n");
        printf("    Please Select[0~2]:");
        scanf("%d", &op);
        switch (op) {
        case 1:
            Sudoku();
            break;
        case 2:
            SAT();
            break;
        case 0:
            printf("Bye Bye! See you Next Time!\n");
            exit(0);
            break;
        }//end of switch
    }//end of while
}

status SAT(void) {
    int op = 1;
    int result, i;
    SATList* CNFList = NULL, * lp;
    SATNode* tp;
    int* value;
    double time;
    clock_t start, finish;
    while (op) {
        system("cls"); printf("\n\n");
        printf("                    SAT MENU
\n");

printf("-------------------------------------------------------------
\n");
        printf("          1.Read cnf File
2.Transverse clause\n");
        printf("          3.Solve(improved) and Save cnf
```

```
4.Solve(not improved)¨\n");
        printf("          0.exit¨\n");

printf("-----------------------------------------------\n");
        printf("    Please Select[0~4]:");
        scanf("%d", &op);
        switch (op) {
        case 1:
            printf("Please key in the name of cnf file to read:");
            scanf("%s", fileName);
            ReadFile(CNFList);
            break;
        case 2:
            if (CNFList == NULL) printf("File not exists\n");
            else {
                printf("cnf clause:\n");
                for (lp = CNFList; lp != NULL; lp = lp->next) {
                    for (tp = lp->head; tp != NULL; tp = tp->next) {
                        printf("%d ", tp->data);
                    }
                    printf("\n");
                }
            }
            printf("Success\n");
            getchar(); getchar();
            break;
        case 3:
            if (CNFList == NULL) printf("File not exists\n");
            else {
                value = (int*)malloc(sizeof(int) * (boolCount + 1));
                for (i = 1; i <= boolCount; i++) value[i] = 1;

                start = clock();

                result = DPLL(CNFList, value);
                finish = clock();
```

```
                printf("Result: %d\n", result);
                if (result == 1) {
                    for (i = 1; i <= boolCount; i++) {
                        if (value[i] == 1) printf("%d ", i);
                        else printf("%d ", -i);
                    }
                    printf("\n");
                }
                time = (double)(finish - start) / CLOCKS_PRE_SEC;
                printf("Operation time=%lfms\n", time * 1000);
                if (SaveFile(result, time, value) == 1)
                    printf("Results are saved to a .txt file with the
same nameþ\n");
                else printf("Failed to Save the file\n");
            }
            getchar(); getchar();
            break;
        case 4:
            value = (int*)malloc(sizeof(int) * (boolCount + 1));
            for (i = 1; i <= boolCount; i++) value[i] = 1;
            start = clock();
            result = DPLL0(CNFList, value);
            finish = clock();
            printf("Result: %d\n", result);
            if (result == 1) {
                for (i = 1; i <= boolCount; i++) {
                    if (value[i] == 1)
                        printf("%d ", i);
                    else printf("%d ", -i);
                }
                printf("\n");
            }
            time = (double)(finish - start) /
CLOCKS_PER_SEC;//¼ÆãÔËÐÐÊ±¼ä
            printf("Operation Time=%lfms\n", time *
```

```
1000);//Êä³öÔËÐÐÊ±¼ä
            if (SaveFile(result, time, value) == 1)
                printf("Results are saved to a .res file with the same
name\n");
            else printf("Failed to Save the file\n");
            getchar(); getchar();
            break;
        case 0:
            printf("Bye Bye! See you Next Time!\n");
            getchar(); getchar();
            exit(0);
            break;
        }//end of switch
    }//end of while
}

status Sudoku(void) {
    int op = 1;
    int result, i;
    SATList* CNFList = NULL, * lp;
    SATNode* tp;
    int* value;
    double time;
    char s[111111];
    clock_t start, finish;
    while (op) {
        system("cls"); printf("\n\n");
        printf("                    Twodoku
\n");

printf("-----------------------------------------------\n");
        printf("          1.Twodoku\n");
        printf("          0. exit\n");

printf("-----------------------------------------------\n");
        printf("    Please Select[0~1]:");
```

```
        scanf("%d", &op);
        switch (op) {
        case 1:
            CreateSudoku();
            printf("Sudoku final game is generated!\n");
            int hole1, hole2;
            printf("Please enter the number of spaces in the top left
Sudoku\n");
            scanf("%d", &hole1);
            Dighole1(hole1);
            printf("Please enter the number of spaces in the bottom right
Sudoku\n");
            scanf("%d", &hole2);
            Dighole2(hole2);
            printf("top left sudoku\n");
            print(start1);
            printf("\nbottom right sudoku\n");
            print(start2);
            printf("Please enter the required cnf file name");
            scanf("%s", fileName);
            tocnf(fileName, hole1 + hole2);
            printf("Suceessfully created cnf, FileName is %s\n",
fileName);
            Create(CNFList);
            value = (int*)malloc(sizeof(int) * (boolCount * 3 + 1));
            for (i = 1; i <= boolCount; i++) value[i] = 1;
            start = clock();
            if (DPLL(CNFList, value) == 1) {
                finish = clock();
                printf("Double Sudoku solved successfully\n");
                printf("Operation hours%fms\n", (double)(finish - start)
/ CLK_TCK * 1000);
                if (Save_sudoku(fileName, value)) {
                    printf("The solution is saved successfully, the file
name is %s\n", fileName);
                    printf("Do you want to start the game? YES or NO\n");
```

```
                    scanf("%s", s);
                    if (strcmp(s, "YES") == 0) play(1, value);
                }
                else printf("Failed to save\n");
            }
            else printf("solve failed\n");
            getchar(); getchar();
            break;
        case 0:
            printf("Bye Bye! See You Next Time!\n");
            getchar(); getchar();
            exit(0);
            break;
        }//end of switch
    }//end of while
}

status ReadFile(struct SATList*& cnf) {
    FILE* fp;
    char ch;
    int nodenumber, i;
    SATList* lp;
    SATNode* tp;
    fp = fopen(fileName, "r");
    if (fp == NULL) {
        printf("Error\n");
        getchar(); getchar();
        return ERROR;
    }
    while ((ch = getc(fp)) != 'p');
    getc(fp); getc(fp); getc(fp); getc(fp);
    fscanf(fp, "%d", &boolCount);
    fscanf(fp, "%d", &clauseCount);
    cnf = (SATList*)malloc(sizeof(SATList));
    cnf->next = NULL;
    cnf->head = (SATNode*)malloc(sizeof(SATNode));
```

```
    cnf->head->next = NULL;
    lp = cnf;
    tp = cnf->head;

    for (i = 0; i < clauseCount; i++, lp = lp->next, tp = lp->head) {
        fscanf(fp, "%d", &nodenumber);
        for (; nodenumber != 0; tp = tp->next) {
            tp->data = nodenumber;
            tp->next = (SATNode*)malloc(sizeof(SATNode));
            fscanf(fp, "%d", &nodenumber);
            if (nodenumber == 0) tp->next = NULL;
        }
        lp->next = (SATList*)malloc(sizeof(SATList));
        lp->next->head = (SATNode*)malloc(sizeof(SATNode));
        if (i == clauseCount - 1) {
            lp->next = NULL;
            break;
        }
    }
    fclose(fp);
    return OK;
}

status Create(struct SATList*& cnf) {
    FILE* fp;
    char ch;
    int nodenumber, i;
    SATList* lp;
    SATNode* tp;
    fp = fopen(fileName, "r");
    if (fp == NULL) {
        printf("ÎÄ¼þ´ò¿ª´íÎó\n");
        getchar(); getchar();
        return ERROR;
    }
    while ((ch = getc(fp)) != 'p');
```

```
    getc(fp); getc(fp); getc(fp); getc(fp);
    fscanf(fp, "%d", &boolCount);
    fscanf(fp, "%d", &clauseCount);
    cnf = (SATList*)malloc(sizeof(SATList));
    cnf->next = NULL;
    cnf->head = (SATNode*)malloc(sizeof(SATNode));
    cnf->head->next = NULL;
    lp = cnf;
    tp = cnf->head;

    //''½¨×Ó¾ä¼¯
    for (i = 0; i < clauseCount; i++, lp = lp->next, tp = lp->head) {
        fscanf(fp, "%d", &nodenumber);
        for (; nodenumber != 0; tp = tp->next) {
            tp->data = nodenumber;     //Êý¾Ý¸³Öµ
            tp->next = (SATNode*)malloc(sizeof(SATNode));//¿ª±ÙÐÂ½áµã
            fscanf(fp, "%d", &nodenumber);
            if (nodenumber == 0) tp->next = NULL;
        }
        lp->next = (SATList*)malloc(sizeof(SATList));//¿ª±ÙÐÂ×Ó¾ä±í
        lp->next->head =
(SATNode*)malloc(sizeof(SATNode));//¿ª±ÙÐÂ½áµã
        if (i == clauseCount - 1) {
            lp->next = NULL;
            break;
        }
    }
    return OK;
}

/* 函数名称: destroyClause*/
void destroyClause(SATList*& cnf)
{
    SATList* lp1, * lp2;
    SATNode* tp1, * tp2;
    for (lp1 = cnf; lp1 != NULL; lp1 = lp2)
```

```
    {
        lp2 = lp1->next;
        for (tp1 = lp1->head; tp1 != NULL; tp1 = tp2)
        {
            tp2 = tp1->next;
            free(tp1);
        }
        free(lp1);
    }
    cnf = NULL;
}
/*函数名称: isUnitClause*/
status isUnitClause(SATNode* cnf)
{
    if (cnf != NULL && cnf->next == NULL)
        return 1;
    else
        return 0;
}
/*函数名称: removeClause*/
int removeClause(SATList*& cnf, SATList*& root)
{
    SATList* lp = root;
    if (lp == cnf) root = root->next;  //删除为头
    else
    {
        while (lp != NULL && lp->next != cnf) lp = lp->next;
        lp->next = lp->next->next;
    }
    free(cnf);
    cnf = NULL;
    return 1;
}
/*函数名称: removeNote*/
int removeNode(SATNode*& cnf, SATNode*& head)
{
```

```
    SATNode* lp = head;
    if (lp == cnf) head = head->next;  //删除为头
    else
    {
        while (lp != NULL && lp->next != cnf) lp = lp->next;
        lp->next = lp->next->next;
    }
    free(cnf);
    cnf = NULL;
    return 1;
}
/*函数名称: addClause*/
int addClause(SATList* cnf, SATList*& root)
{
    //直接插入在表头
    if (cnf != NULL)
    {
        cnf->next = root;
        root = cnf;
        return 1;
    }
    return 0;
}
/*函数名称: emptyClause*/
int isemptyClause(SATList* cnf)
{
    SATList* lp = cnf;
    while (lp != NULL)
    {
        if (lp->head == NULL) return 1;
        lp = lp->next;
    }
    return 0;
}
/*函数名称: CopyClause*/
void CopyClause(SATList*& a, SATList* b)
```

```
{
    SATList* lpa, * lpb;
    SATNode* tpa, * tpb;
    a = (SATList*)malloc(sizeof(SATList));
    a->head = (SATNode*)malloc(sizeof(SATNode));
    a->next = NULL;
    a->head->next = NULL;
    for (lpb = b, lpa = a; lpb != NULL; lpb = lpb->next, lpa = lpa->next)
    {
        for (tpb = lpb->head, tpa = lpa->head; tpb != NULL; tpb = tpb->next,
tpa = tpa->next)
        {
            tpa->data = tpb->data;
            tpa->next = (SATNode*)malloc(sizeof(SATNode));
            tpa->next->next = NULL;
            if (tpb->next == NULL)
            {
                free(tpa->next);
                tpa->next = NULL;
            }
        }
        lpa->next = (SATList*)malloc(sizeof(SATList));
        lpa->next->head = (SATNode*)malloc(sizeof(SATNode));
        lpa->next->next = NULL;
        lpa->next->head->next = NULL;
        if (lpb->next == NULL)
        {
            free(lpa->next->head);
            free(lpa->next);
            lpa->next = NULL;
        }
    }
}
/*函数名称: DPLL*/
int DPLL(SATList*& cnf, int value[])
{
```

```
    SATList* tp = cnf, * lp = cnf, * sp;
    SATNode* dp;
    int* count, i, MaxWord, max, re; //count 记录每个文字出现次
数,MaxWord 记录出现最多次数的文字
    count = (int*)malloc(sizeof(int) * (boolCount * 2 + 1));
FIND:
    while (tp != NULL && isUnitClause(tp->head) == 0) tp = tp->next;  //
找到表中的单子句
    if (tp != NULL)
    {
        //单子句规则简化
        if (tp->head->data > 0) value[tp->head->data] = 1;
        else value[-tp->head->data] = 0;
        re = tp->head->data;
        for (lp = cnf; lp != NULL; lp = sp)
        {
            sp = lp->next;
            //查找含有核心文字的句子
            for (dp = lp->head; dp != NULL; dp = dp->next)
            {
                if (dp->data == re)
                {
                    removeClause(lp, cnf);  //删除子句，简化式子
                    break;
                }
                if (dp->data == -re)
                {
                    removeNode(dp, lp->head);  //删除文字，简化子句
                    break;
                }
            }
        }
        //极简化规则简化后
        if (cnf == NULL)
        {
            free(count);
```

```
            return 1;
        }
        else if (isemptyClause(cnf))
        {
            free(count);
            destroyClause(cnf);
            return 0;
        }
        tp = cnf;
        goto FIND;  //继续简化
    }
    for (i = 0; i <= boolCount * 2; i++) count[i] = 0;  //初始化
    //计算子句中各文字出现次数
    for (lp = cnf; lp != NULL; lp = lp->next)
    {
        for (dp = lp->head; dp != NULL; dp = dp->next)
        {
            if (dp->data > 0) count[dp->data]++;
            else count[boolCount - dp->data]++;
        }
    }
    max = 0;
    //找到出现次数最多的正文字
    for (i = 2; i <= boolCount; i++)
    {
        if (max < count[i])
        {
            max = count[i];
            MaxWord = i;
        }
    }
    if (max == 0)
    {
        //若没有出现正文字,找到出现次数最多的负文字
        for (i = boolCount + 1; i <= boolCount * 2; i++)
        {
```

```
            if (max < count[i])
            {
                max = count[i];
                MaxWord = -i;
            }
        }
    }
    free(count);
    lp = (SATList*)malloc(sizeof(SATList));
    lp->head = (SATNode*)malloc(sizeof(SATNode));
    lp->head->data = MaxWord;
    lp->head->next = NULL;
    lp->next = NULL;
    CopyClause(tp, cnf);
    addClause(lp, tp);
    if (DPLL(tp, value) == 1) return 1;  //在第一分支中搜索
    destroyClause(tp);
    lp = (SATList*)malloc(sizeof(SATList));
    lp->head = (SATNode*)malloc(sizeof(SATNode));
    lp->head->data = -MaxWord;
    lp->head->next = NULL;
    lp->next = NULL;
    addClause(lp, cnf);
    re = DPLL(cnf, value); //回溯到执行分支策略的初态进入另一分支
    destroyClause(cnf);
    return re;
}

int DPLL0(SATList*& cnf, int value[]) {
    SATList* lp = cnf, * sp = cnf, * dp;
    SATNode* tp;
    int* count, i, MaxWord, max, ret;
    //count¼ÇÂ¼Ã¿,öÎÄ×Ö³öÏÖ´ÎÊý£¬MaxWord¼ÇÂ¼³öÏÖ×î¶à´ÎÊýµÄÎÄ×Ö
    count = (int*)malloc(sizeof(int) * (boolCount * 2 + 1));
Cycle: while (lp != NULL && isUnitClause(lp->head) == 0) lp = lp->next;
       //ÕÒµ¥Ôª×Ó¾ä
```

```
if (lp != NULL) {
    if (lp->head->data > 0) value[lp->head->data] = 1;
    else value[-lp->head->data] = 0;
    ret = lp->head->data;
    for (sp = cnf; sp != NULL; sp = dp) {
        dp = sp->next;
        for (tp = sp->head; tp != NULL; tp = tp->next) {
            if (tp->data == ret) {
                removeClause(sp, cnf);
//É¾³ý×Ó¾ä±ê»¯Îª¿Õ
                break;
            }
            if (tp->data == -ret) {
                removeNode(tp, sp->head);
//É¾³ýÎÄ×Ö±ê»¯×Ó¾ä
                break;
            }
        }
    }
    if (cnf == NULL) {
        return OK;
    }
    else if (isemptyClause(cnf)) {
        destroyClause(cnf);
        return ERROR;
    }
    lp = cnf;
    goto Cycle;
//Ñ-»·»¯¾ò
}

lp = (SATList*)malloc(sizeof(SATList));
lp->head = (SATNode*)malloc(sizeof(SATNode));
lp->head->data = cnf->head->data;
lp->head->next = NULL;
lp->next = NULL;
```

```
    CopyClause(sp, cnf);
    addClause(lp, sp);
    if (DPLLO(sp, value) == OK) return OK;
          //ÔÚµÚÒ»¸öÖ§ÖÐDPLL
    destroyClause(sp);
    lp = (SATList*)malloc(sizeof(SATList));
    lp->head = (SATNode*)malloc(sizeof(SATNode));
    lp->head->data = -cnf->head->data;
    lp->head->next = NULL;
    lp->next = NULL;
    addClause(lp, cnf);
    ret = DPLLO(cnf, value);
       //»ØËÝ£¬½øÈëÁíÒ»¸öÖ§
    destroyClause(cnf);
    return ret;
}
/*函数名称: WriteFile*/
status SaveFile(int result, double time, int value[])
{
    FILE* fp;
    int i;
    for (i = 0; fileName[i] != '\0'; i++)
    {
        //修改拓展名
        if (fileName[i] == '.' && fileName[i + 4] == '\0')
        {
            fileName[i + 1] = 'r';
            fileName[i + 2] = 'e';
            fileName[i + 3] = 's';
            break;
        }
    }
    fp = fopen(fileName, "w");
    if (fp == NULL) {
        printf("Failed to open file!\n");
        return 0;
```

```
    }
    fprintf(fp, "s %d\nv ", result);  //求解结果
    if (result == 1)
    {
        //保存解值
        for (i = 1; i <= boolCount; i++)
        {
            if (value[i] == 1) fprintf(fp, "%d ", i);
            else fprintf(fp, "%d ", -i);
        }
    }
    fprintf(fp, "\nt %lf", time * 1000);  //运行时间/毫秒
    fclose(fp);
    return 1;
}

void print(int a[][9]) {
    for (int i = 0; i < 9; i++) {
        for (int j = 0; j < 9; j++) {
            printf("%d ", a[i][j]);
        }
        printf("\n");
    }
}

int DFSCreate(int x, int y, int sudoku[][9]) {
    if (x < 9 && y < 9) {
        if (sudoku[x][y] != 0) {
            return DFSCreate(x, y + 1, sudoku);
        }
        int i, j;
        char check[10];
        memset(check, 0, 10);
        for (i = 0; i < y; i++) check[sudoku[x][i]] = 1;
        for (i = 0; i < x; i++) check[sudoku[i][y]] = 1;
        for (i = x / 3 * 3; i <= x; i++) {
```

```
            if (i == x) for (j = y / 3 * 3; j < y; j++) check[sudoku[i][j]]
= 1;
            else for (j = y / 3 * 3; j < y / 3 * 3 + 3; j++)
check[sudoku[i][j]] = 1;
        }
        int flag = 0;
        for (i = 1; i <= 9 && flag == 0; i++) {
            if (check[i] == 0) {
                flag = 1;
                sudoku[x][y] = i;
                if (y == 8 && x != 8) {
                    if (DFSCreate(x + 1, 0, sudoku)) return 1;
                    else flag = 0;
                }
                else if (y != 8) {
                    if (DFSCreate(x, y + 1, sudoku)) return 1;
                    else flag = 0;
                }
            }
        }
        if (flag == 0) {
            sudoku[x][y] = 0;
            return 0;
        }
    }
    return 1;
}

void CreateSudoku(void) {
    memset(sudoku1, 0, sizeof(sudoku1));
    memset(sudoku2, 0, sizeof(sudoku2));
    int i, j;
    srand((unsigned)time(NULL));
    for (i = 0; i < 9; i++) {
        sudoku1[0][i] = rand() % 9 + 1;
        for (j = 0; j < i;) {
```

```
            if (sudoku1[0][j] == sudoku1[0][i]) {
                sudoku1[0][i] = rand() % 9 + 1;
                j = 0;
            }
            else j++;
        }
    }
    DFSCreate(1, 0, sudoku1);
    for (i = 0; i < 3; i++) {
        for (j = 0; j < 3; j++) {
            sudoku2[i][j] = sudoku1[i + 6][j + 6];
        }
    }
    DFSCreate(0, 3, sudoku2);
}

void Digholel(int hole) {
    int i, j, k, l, flag;
    for (i = 0; i < 9; i++) {
        for (j = 0; j < 9; j++) {
            start1[i][j] = sudoku1[i][j];
        }
    }
    int a[hole][2];
    srand((unsigned)time(NULL));
    for (i = 0; i < hole; i++) {
        j = rand() % 9;
        k = rand() % 9;
        flag = 0;
        for (l = 0; l < i; l++) {
            if (j == a[l][0] && k == a[l][1]) {
                flag = 1;
            }
        }
        if (flag == 0) {
            start1[j][k] = 0;
```

```
            a[i][0] = j;
            a[i][1] = k;
        }
        else i--;
    }
}

void Dighole2(int hole) {
    int i, j, k, l, flag, num = 0;
    for (i = 0; i < 9; i++) {
        for (j = 0; j < 9; j++) {
            start2[i][j] = sudoku2[i][j];
        }
    }
    for (i = 0; i < 3; i++) {
        for (j = 0; j < 3; j++) {
            start2[i][j] = start1[i + 6][j + 6];
            if (start2[i][j] == 0) num++;
        }
    }
    int a[hole - num][2];
    srand((unsigned)time(NULL));
    for (i = 0; i < hole - num; i++) {
        do {
            j = rand() % 9;
            k = rand() % 9;
        } while (j <= 2 && k <= 2);
        flag = 0;
        for (l = 0; l < i; l++) {
            if (j == a[l][0] && k == a[l][1]) {
                flag = 1;
            }
        }
        if (flag == 0) {
            start2[j][k] = 0;
            a[i][0] = j;
```

```
            a[i][1] = k;
        }
        else i--;
    }
}

void tocnf(char* fileName, int hole) {
    FILE* fp;
    fp = fopen(fileName, "w");
    int i, j, k, l, m, share = 0;
    for (i = 0; i < 3; i++) {
        for (j = 0; j < 3; j++) {
            if (start2[i][j] == 0) share++;
        }
    }
    for (i = 0; i < 9; i++) {
        for (j = 0; j < 9; j++) {
            fprintf(fp, " %d ", sudoku1[i][j]);
        }
        fprintf(fp, "        ");
        for (j = 0; j < 9; j++) {
            fprintf(fp, " %d ", sudoku2[i][j]);
        }
        fprintf(fp, "\n");
    }
    fprintf(fp, "\n");
    for (i = 0; i < 9; i++) {
        for (j = 0; j < 9; j++) {
            fprintf(fp, " %d ", start1[i][j]);
        }
        fprintf(fp, "        ");
        for (j = 0; j < 9; j++) {
            fprintf(fp, " %d ", start2[i][j]);
        }
        fprintf(fp, "\n");
    }
```

```
    fprintf(fp, "\n");
    fprintf(fp, "p cnf 2999 %d\n", 24300 - hole);
    for (i = 1; i <= 9; i++) {
        for (j = 1; j <= 9; j++) {
            if (start1[i - 1][j - 1] != 0) {
                fprintf(fp, "%d 0\n", 1000 + i * 100 + j * 10 + start1[i
- 1][j - 1]);
            }
            if (start2[i - 1][j - 1] != 0) {
                fprintf(fp, "%d 0\n", 2000 + i * 100 + j * 10 + start2[i
- 1][j - 1]);
            }
        }
    }
    for (l = 1; l <= 9; l++) {
        for (i = 1; i <= 9; i++) {
            for (k = 1; k <= 9; k++) {
                for (j = i + 1; j <= 9; j++) {
                    fprintf(fp, "-%d -%d 0\n", 1000 + ((l - 1) / 3 * 3
+ (i - 1) / 3 + 1) * 100 + ((l - 1) % 3 * 3 + (i - 1) % 3 + 1) * 10 + k,
1000 + ((l - 1) / 3 * 3 + (j - 1) / 3 + 1) * 100 + ((l - 1) % 3 * 3 + (j
- 1) % 3 + 1) * 10 + k);
                    fprintf(fp, "-%d -%d 0\n", 2000 + ((l - 1) / 3 * 3
+ (i - 1) / 3 + 1) * 100 + ((l - 1) % 3 * 3 + (i - 1) % 3 + 1) * 10 + k,
2000 + ((l - 1) / 3 * 3 + (j - 1) / 3 + 1) * 100 + ((l - 1) % 3 * 3 + (j
- 1) % 3 + 1) * 10 + k);
                }
            }
        }
    }
    for (i = 7; i <= 9; i++) {
        for (j = 7; j <= 9; j++) {
            for (k = 1; k <= 9; k++) {
                fprintf(fp, "-%d %d 0\n", 1000 + i * 100 + j * 10 + k,
2000 + (i - 6) * 100 + (j - 6) * 10 + k);
                fprintf(fp, "%d -%d 0\n", 1000 + i * 100 + j * 10 + k,
```

```
2000 + (i - 6) * 100 + (j - 6) * 10 + k);
            }
        }
    }
    for (i = 1; i <= 9; i++) {
        for (j = 1; j <= 9; j++) {
            for (k = 1; k <= 9; k++) {
                for (l = j + 1; l <= 9; l++) {
                    fprintf(fp, "-%d -%d 0\n", 1000 + i * 100 + j * 10
+ k, 1000 + i * 100 + l * 10 + k);
                    fprintf(fp, "-%d -%d 0\n", 1000 + j * 100 + i * 10
+ k, 1000 + l * 100 + i * 10 + k);
                    fprintf(fp, "-%d -%d 0\n", 2000 + i * 100 + j * 10
+ k, 2000 + i * 100 + l * 10 + k);
                    fprintf(fp, "-%d -%d 0\n", 2000 + j * 100 + i * 10
+ k, 2000 + l * 100 + i * 10 + k);
                }
            }
        }
    }
    for (i = 1; i <= 9; i++) {
        for (k = 1; k <= 9; k++) {
            for (j = 1; j <= 9; j++) {
                fprintf(fp, "%d ", 1000 + i * 100 + j * 10 + k);
            }
            fprintf(fp, "0\n");
            for (j = 1; j <= 9; j++) {
                fprintf(fp, "%d ", 1000 + j * 100 + i * 10 + k);
            }
            fprintf(fp, "0\n");
            for (j = 1; j <= 9; j++) {
                fprintf(fp, "%d ", 2000 + i * 100 + j * 10 + k);
            }
            fprintf(fp, "0\n");
            for (j = 1; j <= 9; j++) {
                fprintf(fp, "%d ", 2000 + j * 100 + i * 10 + k);
```

```
            }
            fprintf(fp, "0\n");
        }
    }
    for (l = 1; l <= 9; l++) {
        for (i = 1; i <= 3; i++) {
            for (j = 1; j <= 3; j++) {
                for (k = 1; k <= 9; k++) {
                    fprintf(fp, "%d ", 1000 + ((l - 1) / 3 * 3 + i) * 100
+ ((l - 1) % 3 * 3 + j) * 10 + k);
                }
                fprintf(fp, "0\n");
                for (k = 1; k <= 9; k++) {
                    fprintf(fp, "%d ", 2000 + ((l - 1) / 3 * 3 + i) * 100
+ ((l - 1) % 3 * 3 + j) * 10 + k);
                }
                fprintf(fp, "0\n");
            }
        }
    }
    for (i = 1; i <= 9; i++) {
        for (j = 1; j <= 9; j++) {
            for (k = 1; k <= 9; k++) {
                fprintf(fp, "%d ", 1000 + i * 100 + j * 10 + k);
            }
            fprintf(fp, "0\n");
            for (k = 1; k <= 9; k++) {
                fprintf(fp, "%d ", 2000 + i * 100 + j * 10 + k);
            }
            fprintf(fp, "0\n");
        }
    }
    for (i = 1; i <= 9; i++) {
        for (j = 1; j <= 9; j++) {
            for (k = 1; k <= 9; k++) {
                for (l = k + 1; l <= 9; l++) {
```

```
                    fprintf(fp, "-%d -%d 0\n", 1000 + i * 100 + j * 10
+ k, 1000 + i * 100 + j * 10 + 1);
                    fprintf(fp, "-%d -%d 0\n", 2000 + i * 100 + j * 10
+ k, 2000 + i * 100 + j * 10 + 1);
                }
            }
        }
    }
    fclose(fp);
}

int Save_sudoku(char* fileName, int* val) {
    FILE* fp;
    int i, j, k;
    for (i = 0; fileName[i] != '\0'; i++) {
        if (fileName[i] == '.') {
            fileName[i + 1] = 'r';
            fileName[i + 2] = 'e';
            fileName[i + 3] = 's';
            break;
        }
    }
    fp = fopen(fileName, "w");
    fprintf(fp, "Answer:\n");
    for (i = 1; i <= 9; i++) {
        if (i < 7) {
            for (j = 1; j <= 9; j++) {
                if (start1[i - 1][j - 1] != 0) fprintf(fp, "%d ", start1[i
- 1][j - 1]);
                else {
                    for (k = 1; k <= 9; k++) {
                        if (val[1000 + i * 100 + j * 10 + k] == 1)
fprintf(fp, "%d ", k);
                    }
                }
                if (j % 3 == 0) fprintf(fp, "   ");
```

```
            }
            fprintf(fp, "\n");
        }
        else {
            for (j = 1; j < 7; j++) {
                if (start1[i - 1][j - 1] != 0) fprintf(fp, "%d ", start1[i
- 1][j - 1]);
                else {
                    for (k = 1; k <= 9; k++) {
                        if (val[1000 + i * 100 + j * 10 + k] == 1)
fprintf(fp, "%d ", k);
                    }
                }
                if (j % 3 == 0) fprintf(fp, "   ");
            }
            for (j = 1; j <= 9; j++) {
                if (start2[i - 7][j - 1] != 0) fprintf(fp, "%d ", start2[i
- 7][j - 1]);
                else {
                    for (k = 1; k <= 9; k++) {
                        if (val[2000 + i * 100 + j * 10 + k] == 1)
fprintf(fp, "%d ", k);
                    }
                }
                if (j % 3 == 0) fprintf(fp, "   ");
            }
            fprintf(fp, "\n");
        }
        if (i % 3 == 0) fprintf(fp, "\n");
    }
    for (i = 4; i <= 9; i++) {
        fprintf(fp, "                          ");
        for (j = 1; j <= 9; j++) {
            if (start2[i - 1][j - 1] != 0) fprintf(fp, "%d ", start2[i
- 1][j - 1]);
            else {
```

```
                for (k = 1; k <= 9; k++) {
                    if (val[2000 + i * 100 + j * 10 + k] == 1) fprintf(fp,
"%d ", k);
                }
            }
            if (j % 3 == 0) fprintf(fp, "   ");
        }
        fprintf(fp, "\n");
        if (i % 3 == 0) fprintf(fp, "\n");
    }
    fclose(fp);
    return 1;
}

void play(int flag, int* val) {
    int i, j, k, x, num;
    char s[10];
    while (flag) {
        system("cls");
        flag = 0;
        for (i = 1; i <= 9; i++) {
            if (i < 7) {
                for (j = 1; j <= 9; j++) {
                    printf("%d ", start1[i - 1][j - 1]);
                    if (j % 3 == 0) printf("  ");
                    if (start1[i - 1][j - 1] == 0) flag = 1;
                }
                printf("\n");
            }
            else {
                for (j = 1; j < 7; j++) {
                    printf("%d ", start1[i - 1][j - 1]);
                    if (j % 3 == 0) printf("  ");
                    if (start1[i - 1][j - 1] == 0) flag = 1;
                }
                for (j = 1; j <= 9; j++) {
```

```
                printf("%d ", start2[i - 7][j - 1]);
                if (j % 3 == 0) printf("  ");
                if (start2[i - 7][j - 1] == 0) flag = 1;
            }
            printf("\n");
        }
        if (i % 3 == 0) printf("\n");
    }
    for (i = 4; i <= 9; i++) {
        printf("                ");
        for (j = 1; j <= 9; j++) {
            printf("%d ", start2[i - 1][j - 1]);
            if (j % 3 == 0) printf("  ");
            if (start2[i - 1][j - 1] == 0) flag = 1;
        }
        printf("\n");
        if (i % 3 == 0) printf("\n");
    }
    printf("Please enter the Sudoku serial number (1 in the upper left and 2 in the lower right), row sequence, column sequence, and fill in the numbers:\n");
    scanf("%d%d%d%d", &k, &i, &j, &x);
    if (k != 1 && k != 2 && (i < 1 || i>9) && (j < 1 || j>9) || (x < 1 || x>9)) printf("Error\n");
    num = k * 1000 + i * 100 + j * 10 + x;
    if (k == 1 && start1[i - 1][j - 1] != 0) {
        printf("Cannot be empty!\n");
        continue;
    }
    if (k == 2 && start2[i - 1][j - 1] != 0) {
        printf("Cannot be empty!\n");
        continue;
    }
    if (val[num] == 1) {
        printf("Correct\n");
        if (k == 1) start1[i - 1][j - 1] = x;
```

```
            else start2[i - 1][j - 1] = x;
        }
        else {
            printf("Wrong\n");
        }
        printf("Continue to play? YES or NO\n");
        scanf("%s", s);
        if (strcmp(s, "NO") == 0) return;
    }
    printf("End! Good Game!\n");
}
```