

**课程设计报告**

**题目：基于SAT的对角线数独游戏求解程序**

**课程名称：程序设计综合课程设计**

**专业班级： 计科2303**

**学 号： U202310917**

**姓 名： 张佳艺**

**指导教师： 卢萍**

**报告日期： 2024-9-12**

**计算机科学与技术学院**

**任务书**

* **设计内容**

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

* **设计要求**

要求具有如下功能：

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

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

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

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

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

**（6）SAT应用：**将数双独游戏[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%)

* **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

1.1.1课题背景 1

1.1.2课堂意义 1

1.2国内外研究现状 2

1.3课程设计的主要研究工作 2

1.3.1基于DPLL算法的SAT求解器 2

1.3.2基于SAT求解器的对角线数独的生成，归约和求解 3

**2系统需求分析与总体设计** 4

2.1系统需求分析 4

2.1.1SAT求解器 4

2.1.2对角线数独的生成，归约和求解 4

2.2系统总体设计 4

**3系统详细设计** 6

3.1有关数据结构的定义 6

3.2主要算法设计 7

3.2.1cnf文件读取及存储 7

3.2.2DPLL核心算法 8

3.2.3res输出和保存 14

3.2.4对角线数独应用 15

**4系统实现与测试** 18

4.1系统实现 18

4.2系统测试 19

4.2.1solver 模块测试 19

4.2.2X-Sudoku模块测试 22

**5总结与展望** 27

5.1全文总结 27

5.2工作展望 27

**6体会** 28

**参考文献** 29

**附录** 30

**1引言**

**1.1课题背景与意义**

**1.1.1课题背景**

对角线数独是普通数独的一种衍生变体。一般的数独是用1~9的填充一个9\*9的数独残局，使数独的每一行，每一列和每个3\*3的宫格中都包含1~9这九个数，且每行每列和3\*3宫格中的数字没有重复。而对角线数独在此基础上有要求两条对角线上也要包含1~9这九个数且不出现重复。这使解对角线数独变得更加复杂也更加有趣。

而求解数独的难度会随着数独的规模和约束条件增加，而为了高效地解决数独难题，引入SAT问题求解器是一个有效的算法。

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。本课题主要基于DPLL算法实现一个高效的SAT问题求解器，进而应用于对角线数独游戏求解。

**1.1.2课堂意义**

对角线数独游戏的求解会随着规模和约束条件的增加变得复杂，人工求解低效且困难，引进SAT问题求解器，将数独约束条件转化为布尔公式进行求解可以大幅提高问题求解效率。对算法进一步改进应用，可以实现对角线数独的生成和验证，进一步加强数独游戏的互动性和可玩性。

同时，在将SAT问题求解器应用于数独游戏的过程中，有助于我们进一步理解并改进DPLL的算法，不断优化，为SAT求解器在其他领域的应用提供启发。

**1.2国内外研究现状**

国内外对基于SAT的对角线数独游戏求解程序都进行了大量且深入的研究，并取得了一定的成果。

国外在算法优化、SAT求解器性能提升和应用领域拓展方面都研究出了显著成果，如在DPLL的基础上发展而来CDCL算法，加入了冲突学习和子句传播的技术，显著提升了求解效率，其在求解大规模问题方面表现出色。而针对对角线数独，研究者们提出了多种优化算法。例如，结合启发式方法（如最小剩余值启发式）和局部搜索优化技术来改进SAT求解器在处理对角线数独时的性能。同时，有效的约束传播技术在对角线数独中也被广泛应用。通过提前传播约束，可以减少搜索空间，提高求解效率。此外，研究者们还在继续探索如何更好地利用对角线约束进行剪枝和冲突学习

而国内研究也在逐步跟进，致力于让其在实际问题中发挥作用，包括硬件验证、优化问题等领域，希望其应用于实际工业问题中。在算法优化方面，大量研究者对对角线数独的SAT建模技术进行了深入探索，在传统数独建模的基础上，研究了如何高效地处理对角线约束，并提出了相应的改进策略。此外，研究者们还尝试通过多种技术手段提高对角线数独的求解效率，例如改进启发式策略、优化冲突学习机制等。

随着研究的不断深入和技术的不断进步，数独问题的建模和求解将不断优化， SAT求解器也将有望应用在更多实际问题和其他领域。

**1.3****课程设计的主要研究工作**

**1.3.1基于DPLL算法的SAT求解器**

设计SAT问题中变元、文字、子句、公式等有效的物理存储结构，并基于DPLL过程实现一个高效SAT求解器，实现对给定的ｃｎｆ文件正确的读取，存储，求解。同时基于相应的时间处理函数，记录DPLL过程执行时间（以毫秒为单位），并在SAT求解器的基础上对DPLL算法进行优化，通过对存储结构、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。

**1.3. 2基于SAT求解器的对角线数独的生成，归约和求解**

随机生成对角线数独终盘，并采用挖洞法实现对角线数独的游戏盘和提示数，将对角线数独游戏规则约束转化为SAT问题，并把它表示为CNF公式的形式，在规则约束下，用SAT求解器对归约的ｃｎｆ范式进行求解。

**2****系统需求分析与总体设计**

**2.1****系统需求分析**

**2.1.1 SAT求解器**

　　实现对给定的ｃｎｆ文件正确的读取，存储，求解。同时基于相应的时间处理函数，记录DPLL过程执行时间（以毫秒为单位），并在SAT求解器的基础上对DPLL算法进行优化，通过对存储结构、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。

**2.1.2对角线数独的生成，归约和求解**

　　基于SAT求解器实现对角线数独的生成，规约，求解和验证，并实现一定的互动性和可玩性。

**2.2系统总体设计**

系统主要有两大功能模块：SAT求解器模块和对角线数独游戏模块。

SAT求解器模块主要包括五个子功能模块：读取cnf文件并存储，输出cnf文件内容，DPLL求解SAT问题，输出求解答案和运行时间，保存答案至res文件。

对角线数独游戏模块主要包括四个子功能模块：数独游戏生成，数独求解，DPLL答案验证，DPLL求解参考答案，各模块之间可以随时切换。

图示, 示意图

描述已自动生成

图1-1 系统模块结构图

**3系统详细设计**

**3.1有关数据结构的定义**

系统中主要要处理两大部分的数据；

第一部分是cnf 文件中的数据，包括子句集，构成子句集的子句（Clause）和构成子句的变元（Literal），变元的值为int 型，采用二维链表存储；

第二部分是数独部分的数据，包括数据的行，列，值，为int 型，采用二维数组存储；

表1-1 数据类型表

|  |  |  |  |
| --- | --- | --- | --- |
| 数据结构名 | 数据项 | 数据类型 | 数据名 |
| Literal | value | int | 变元值 |
|  | next | Literal \* | 指向同子句下一个 |
|  | pre | Literal \* | 指向同子句上一个 |
| Clause | num | int | 子句中变元数 |
|  | head | Literal \* | 指向子句中第一个变元 |
|  | next | Clause \* | 指向下一子句 |
| Shudu | i | int | 数独行 |
|  | j | int | 数独列 |
|  | k | int | 数独值 |

数独数据与SAT问题变元关联如下：第i行第j列值为k转换为变元值公式为（(i-1)\*81+(j-1)\*9+k）;若不为k，则变元值取负数。

变元与子句通过指向变元集第一个变元的头指针关联，子句是变元的一维链表；子句与cnf范式通过指向第一个子句的指针关联，cnf范式是关于变元的二维链表。

图表, 箱线图

描述已自动生成

图1-2 数据结构关联图

**3.2 主要算法设计**

**3.2.1 cnf文件读取及存储**

算法名：ReadCNF

算法实现：

1. 传入cnf文件地址，打开文件;
2. 核对检查cnf文件格式，跳过注释行;
3. 读入变元数量ltrnum和子句数量clsnum；
4. 为指向二维链表的指针CNF分配空间，准备存储cnf文件内容，定义指向当前读取子句的指针pc和指向当前读取变元的指针pl;

5.通过ReadLiteral函数依次读取变元值并存储pl->value中，下一个值不为0时，pl指向下一变元，继续读取，否则pc指向下一子句，pl指向下一子句的头结点；

图示

描述已自动生成6.读入正确子句数且读到0表示cnf文件读取完毕，关闭文件，返回OK，否则返回FALSE；

图1-3-1 cnf文件读取流程图

算法名：ReadLiteral

算法实现：依次读入数字，遇到-号标记，返回值时\*（-1），读到空格返回结果

**3.2.2 DPLL核心算法**

算法名：DPLL

算法实现：该算法主要采用单子句规则化简和分裂策略两种化简交叉进行，通过回溯搜索求解SAT问题；

1. 循环遍历子句集，寻找单子句，若找到根据单子句规则对子句集化简，继续循环遍历，若找不到单子句，跳出循环；
2. 根据分裂变元选择策略选择相应变元，复制cnf子句集为回溯做准备；
3. 取opv为真作为单子句变元值加入子句集Tcnf,对Tcnf递归调用DPLL算法检测其可满足性，若可满足，则原子句集也可满足，释放Tcnf空间，返回TRUE；
4. 若不满足，取opv为假作为单子句变元值加入子句集Tcnf，对Tcnf递归调用DPLL算法检测其可满足性，记录结果，释放Tcnf空间后返回记录结果；

图示

描述已自动生成

图1-3-2 DPLL流程图

DPLL功能函数：

findUnitClause：寻找单子句

算法实现：遍历cnf，若子句变元数为1，返回变元值，否则遍历结束返回0；

IsEmpty：判断是否存在空子句

算法实现：遍历cnf，若子句变元数为1，返回1，否则遍历结束返回0；

Simplification：单子句规则化简,根据传入的opv对cnf化简

算法实现：依次遍历cnf变元值，若pl->value=opv，删除当前子句，若pl->value=-opv，删除当前节点；

dltClause算法实现：

图示

描述已自动生成

图1-3-3 删除子句流程图

dltLiteral算法实现：

图示

描述已自动生成

图1-3-4 删除节点流程图

getTcnf 算法实现：

1. 传入原子句集cnf和复制子句集Tcnf的地址；
2. Tcnf初始化，分配其指向第一个子句的空间；
3. 定义指针pc,Tpc分别指向cnf头节点和Tcnf头结点，定义指针pl和Tpl分别指向pc->head和Tpc->head;
4. 双重循环遍历cnf的子句和变元，将pc的数据赋给Tpc,将pl的数据赋给Tpl,当pl->next!=NULL时，为Tpl->next分配空间，并将Tpl指向Tpl->next,当pl->next==NULL时，退出内层循环，当pc->next!=NULL时为Tpc->next分配空间，并将Tpc指向Tpc->next,tpl指向Tpc->head,当pc->next==NULL时，退出外层循环；
5. 结束循环，退出函数；

addClause：在cnf子句集用头插法加入一个变元值为val的单子句

算法实现：

1. 定义一个指向新子句的指针newcls;
2. 为newcls分配空间，newcls->num赋值为1，newcls->next指向cnf;
3. 为newcls->head分配空间，newcls->head->value赋值为val，newcls->head->next/pre赋值为NULL；
4. 将cnf指向newcls,返回；

FreeTcnf：释放复制体Tcnf空间

算法实现:

图示

描述已自动生成

图1-3-5 释放Tcnf空间流程图

分裂变元选择策略：主要考虑了五种策略差异，最终选择了策略(1)(4)(5)

1. getFirstLtr 取子句集第一个子句的第一个变元值为opv

算法实现：如果cnf->head->value>0，返回cnf->head->value，否则返回-cnf->head->value;

1. getMaxLtr 选取出现次数最多的变元，且考虑正负

图示

描述已自动生成算法实现：

图1-3-6 getMaxLtr流程图

1. getMaxLtr\_improved1 选取出现次数最多的变元，且不考虑正负

算法实现：

图示

描述已自动生成

图1-3-7 getMaxLtr1流程图

1. getMaxLtr\_improved2 选取在较短子句中出现次数最多的变元，且考虑正负

算法实现：

图示

描述已自动生成

图1-3-8 getMaxLt12流程图

1. getMaxLtr\_improved3 选取在较短子句中出现次数最多的变元，且不考虑正负

算法实现：

图示

描述已自动生成

图1-3-9 getMaxLtr3流程图

**3.2.3 res输出和保存**

算法名：saveAns

算法实现：

1.传入求解满足性结果和DPLL执行时间；

2.打开res文件，若打开失败，返回ERROR；

3.写入求解满足性；

4.若有满足性答案，依次读入ans[i]，若ans[i]大于0，写入i，否则写入-i,i==ltrnum时，结束遍历；

5.写入DPLL算法执行时间；

6.关闭文件，返回OK；

**3.2.4** **对角线数独应用**

生成对角线数独游戏，采用挖洞法生成：

CreatXSudoku：随机生成符合对角线规则约束的9\*9数独

算法实现：

1. 首先随机生成中间3\*3格子内的数，采用rand()随机生成1~9的数，若与已填入的数不重复的话，填入数值，若重复，重新生成随机数继续验证是否冲突，直到九个格子都填入不重复的九个数；
2. 采用轮换法根据中间已填入的3\*3格子填入四边3\*3的格子，且使每行，每列的数字都不重复；
3. 图示

   描述已自动生成根据对角线约束依次填入四角3\*3的格子，使两条对角线上的数各不相同；

图1-3-10 数独生成流程图

creatHoles：根据选择难度随机挖去指定数目的洞

算法实现：

1. 循环随机数生成挖洞位置，若该位置已经挖去，则继续随机生成挖洞位置；
2. 检验挖去该位置是否影响对角线数独唯一解，若产生多解，则继续随机生成挖洞位置；
3. 否则挖去该位置数，需挖洞数减1，继续重复1~3过程；
4. 当需挖洞数减为0时，退出循环，返回OK；

图示

描述已自动生成

图1-3-11 挖洞流程图

ShuduToCnf：将生成的对角线数独和对角线数独规则约束归约为cnf文件

算法实现：

1. 打开保存文件地址，若打开失败，返回ERROR；
2. 将注释行和变元数量，子句数量等写入文件；
3. 遍历数独数组，若shudu[i][j]非0，将提示数写入文件，变元值为（i-1）\*81+(j-1)\*9+shudu[i][j];
4. 将格约束写入文件，遍历数独格子，每个格子都需填入1~9这九个数；
5. 将行约束写入文件，遍历数独每行，每行中任两个格子不可填入同一个数；
6. 将列约束写入文件，遍历数独每列，每列中任两个格子不可填入同一个数；
7. 将3\*3约束写入文件，每个数在3\*3的格子中不可重复；
8. 将对角线约束写入文件，两条对角线上不可出现重复的数字；
9. 关闭文件，返回OK；

图示, 示意图

描述已自动生成

图1-3-12 数独归约流程图

**4系统实现与测试**

**4.1系统实现**

本系统实现的硬件环境主要是包括处理器（CPU），8G以上的内存，存储设备，输入输出等外设的计算机环境，软件环境主要是在Windows操作系统及Dev C++上完成调试校验，由C语言实现；

有关数据结构部分的C语言定义如下：

typedef struct literal{

int value;

struct literal \*pre;

struct literal \*next;

}Literal; //文字

typedef struct clause{

int num; //子句中文字数目

struct literal \*head; //指向子句中第一个文字

struct clause \*next; //指向下一个子句

}Clause;

系统中主要包括四大模块的函数：主界面的函数主要实现界面的显示，跳转和链接调用各功能函数的功能；cnfparser的函数主要实现cnf文件的读取，存储和res文件的保存输出；solver的函数主要实现基于DPLL的SAT求解器功能；X-Sudoku的函数主要实现对角线数独的生成，归约，校验和求解功能。函数具体功能和调用关系详见下图：

图示

描述已自动生成

图1-4函数关系示意图

**4.2****系统测试**

**4.2.1 solver 模块测试**

（1）本模块主要实现基于DPLL算法实现SAT求解器，即对读取的cnf文件正确求解其可满足性的功能。设计目标是可以根据不同的SAT问题选取不同的分裂变元政策，以达到优化算法时间，求解更多规模文件的目的。

（2）测试数据主要分为以下选取方向：功能测试算例（包括满足算例和不满足算例，规模较小，主要验证算法可行性），性能测试算例（包括不同特规模的满足算例和不满足算例），特殊算例（包括现有算法不能求解的算例，用以优化测试）。

（3）测试大纲如下：

函数功能测试：主要采用功能测试算例，测试模块各函数是否实现预期功能；

DPLL算法测试：主要采用功能测试算例，测试满足数据和不满足数据求解结果是否正确，对DPLL算法进行基本实现；

SAT求解性能测试：主要采用性能测试算例数据和特殊数据，测试模块对大型数据的处理能力，并作出相应优化，测试优化效果；

（4）测试运行结果：

模块各函数基本实现，DPLL算法功能基本实现，SAT求解器基本实现：

图表, 散点图

描述已自动生成

图形用户界面

低可信度描述已自动生成图1-5-1 基准满足算例读取结果图

图1-5-2基准满足算例求解图

日历

描述已自动生成

图形用户界面, 文本, 应用程序

中度可信度描述已自动生成图1-5-3基准不满足算例读取结果图

图1-5-4基准不满足算例求解图

模块具有基本的性能处理能力，经优化可以有效节省算法求解时间，处理规模较大的cnf文件：

图形用户界面, 应用程序

描述已自动生成

图1-5-5性能测试算例求解图

（5）经运行结果确认，该模块基本实现预定目标和功能，可以实现一定规模的SAT问题求解。

**4.2.2 X-Sudoku模块测试**

（1）本模块主要实现基于SAT求解器对角线数独游戏，即根据难度随机生成对应的对角线数独游戏，求解数独，验证答案等功能。设计目标是可以根据用户需求生成对应难度的对角线数独游戏，为用户求解数独提供一定的交互性，用户求解时可以查看答案，求解后可以基于DPLL算法验证自己答案，或用SAT求解器直接求解答案，为用户提供多种选择，增加游戏趣味性。

（2）测试数据主要分为以下选取方向：选取不同难度的对角线数独，分别求解正确，错误，系统自身生成的答案，验证对角线数独的各项功能。

（3）测试大纲如下：

选取Simple难度的对角线数独游戏，用户解决对角线数独游戏并验证答案正确性；选取Normal难度的对角线数独游戏，填入错误的答案，验证其正确性；选取Difficult难度的对角线数独游戏，用SAT求解器求解其答案。

图形用户界面, 文本, 应用程序, 聊天或短信

描述已自动生成（4）测试运行结果：

图形用户界面, 文本, 应用程序, 聊天或短信

描述已自动生成图1-6-1对角线数独界面图

图1-6-2对角线数独难度选择界面图

表格

描述已自动生成

图1-6-3 Simple数独生成图

图形用户界面

低可信度描述已自动生成

图1-6-4数独求解验证成功图

表格

描述已自动生成

图1-6-5 Normal数独生成图

图形用户界面

中度可信度描述已自动生成

图1-6-6数独求解验证失败图

表格

中度可信度描述已自动生成

图1-6-7 Difficult数独生成图

表格

描述已自动生成

图1-6-8 DPLL求解数独图

（5）经运行结果确认，该模块基本实现预定目标和功能，具有一定界面交互性和游玩趣味性。

**5 总结与展望**

**5.1****全文总结**

在这次课设作业中，我的主要工作如下：

（1）研读任务书，查阅相关论文资料，了解基于DPLL算法的SAT求解器，对算法实现的原理和相关功能由一定认识，并对程序实现需要的模块规划预期目标；

（2）根据预期目标实现程序的基本功能——即SAT求解器的算法实现；

（3）对实现的DPLL算法测试数据，验证程序正确性并作出调整完善；

（4）基于实现的SAT求解器完成对角线数独游戏的功能模块，实现对角线数独的生成，归约，求解和验证，并测试验证其正确性；

（5）基于特殊或较大规模的测试数据对程序作出进一步完善优化，减少求解时间，增强程序性能；

（6）在原有的程序基础上，采用easyX制作简易界面，增强各功能模块联系，增加界面与用户的互动性和游戏趣味性。

**5.1工作展望**

在今后的研究中，我将主要围绕着如下几个方面开展工作:

（1）进一步优化程序数据结构，优化SAT求解器算法实现，以实现更多数据的求解；

（2）拓展对角线数独游戏的功能，如增加多人对局，难度闯关等模式，进一步增强对角线数独游戏的可玩性；

（3）完善界面，使其更美观，更具互动性和趣味性；

# 6 体会

完成一项项目设计和实现需要花费很多精力和功夫，但在这一过程中我也收获良多。

首先就是项目开始实现之前，一定要明确项目需求和实现目标，这次项目实现过程中，我就因为对角线数独游戏没有充分应用SAT求解器而中途推翻重做。

其次就是要模块化设计，项目的需求越大，要实现的功能就越多，如果将功能都集成在一个文件，不仅会让代码过于冗长，还会增加程序测试完善的困难度。而模块化设计可以让程序的功能分工更明确，便于我们编写实现和测试优化。

最后尤为重要的是对实现的函数功能进行测试，采取尽量全面的测试数据，逐一验证各函数的功能性，对于规模较大的测试数据和特殊数据，针对性优化，增加代码处理性能，确保程序功能完善，满足项目需求。

我相信，这次项目实现的过程体会会对我未来工作奠定宝贵的基础经验。

**参考文献**

[1] 华中科技大学综合课程设计，华中科技大学计算机科学与技术学院程序设计综合课程设计课程组，2024年5月

[2] 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

[3] 360百科：数独游戏<https://baike.so.com/doc/3390505-3569059.html>

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

[4] 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

[5] 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**

**1界面.cpp**

#include <stdio.h>

#include <easyx.h>

#include "数独\_main.h"

#include <graphics.h>

ExMessage msg={ 0 }; //获取鼠标信息

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*函数名称：drawshape，drawtext，drawtable

\*函数功能：绘制数独棋盘

\*条件：无

\*输入：无 输出：数独棋盘

\*返回值类型：void类型

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void drawshape(){

setlinecolor(RGB(70,130,180));

setlinestyle(PS\_SOLID,3);

setfillcolor(RGB(176,196,222));

rectangle(100,100,370,370);

setlinestyle(PS\_SOLID,2);

line(190,100,190,370); line(280,100,280,370);

line(100,190,370,190); line(100,280,370,280);

setlinestyle(PS\_SOLID,1);

line(130,100,130,370); line(160,100,160,370);

line(220,100,220,370); line(250,100,250,370);

line(310,100,310,370); line(340,100,340,370);

line(100,130,370,130); line(100,160,370,160);

line(100,220,370,220); line(100,250,370,250);

line(100,310,370,310); line(100,340,370,340);

}

void drawtext(){

settextstyle(25,0,"微软雅黑");

settextcolor(RGB(72,61,139));

outtextxy(110,70,"1"); outtextxy(80,100,"1");

outtextxy(140,70,"2"); outtextxy(80,130,"2");

outtextxy(170,70,"3"); outtextxy(80,160,"3");

outtextxy(200,70,"4"); outtextxy(80,190,"4");

outtextxy(230,70,"5"); outtextxy(80,220,"5");

outtextxy(260,70,"6"); outtextxy(80,250,"6");

outtextxy(290,70,"7"); outtextxy(80,280,"7");

outtextxy(320,70,"8"); outtextxy(80,310,"8");

outtextxy(350,70,"9"); outtextxy(80,340,"9");

}

void drawtable(){

drawshape();

drawtext();

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*函数名称：InArea

\*函数功能：判断鼠标位置

\*条件：获得鼠标信息

\*输入：鼠标位置和按钮位置 输出：是or否

\*返回值类型：int类型

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int InArea(int x,int y,int w,int h,int mx, int my){

if((mx>=x&&mx<=x+w)&&(my>=y&&my<=y+h)){

return 1;

}

return 0;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*函数名称：button

\*函数功能：绘制按钮并判断是否按下

\*条件：获得鼠标信息

\*输入：按钮坐标，长宽和按钮文字 输出：是or否

\*返回值类型：int类型

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int button(int x,int y,int w,int h,const char \*text){

settextstyle(20,0,"微软雅黑");

setlinecolor(RGB(70,130,180));

setlinestyle(PS\_SOLID,2);

if(InArea(x,y,w,h,msg.x,msg.y)){

setfillcolor(RGB(238,221,130));

}

else{

setfillcolor(RGB(250,250,210));

}

fillroundrect(x,y,x+w,y+h,20,20);

int hSpace=(w-textwidth(text))/2;

int vSpace=(h-textheight(text))/2;

outtextxy(x+hSpace,y+vSpace,text);

if(msg.message==WM\_LBUTTONDOWN&&InArea(x,y,w,h,msg.x,msg.y)){

return 1;

}

return 0;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*函数名称：Filltable

\*函数功能：填充数独棋盘完成解题

\*条件：数独生成成功

\*输入：鼠标位置 输出：数独棋盘现状

\*返回值类型：void类型

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Filltable(){

char num[20];

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

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

if(shudu[i][j]){

settextstyle(20,0,"微软雅黑");

sprintf(num,"%d",shudu[i][j]);

outtextxy(80+30\*i,70+30\*j,num);

}

else{

if(button(500,70,40,40,"1")){

n=1;

}

if(button(500,120,40,40,"2")){

n=2;

}

if(button(500,170,40,40,"3")){

n=3;

}

if(button(500,220,40,40,"4")){

n=4;

}

if(button(500,270,40,40,"5")){

n=5;

}

if(button(500,320,40,40,"6")){

n=6;

}

if(button(500,370,40,40,"7")){

n=7;

}

if(button(500,420,40,40,"8")){

n=8;

}

if(button(500,470,40,40,"9")){

n=9;

}

if(n!=0){

if(button(70+i\*30,70+j\*30,30,30,"")){

shudu[i][j]=n;

n=0;

}

}

}

}

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*函数名称：Fillres

\*函数功能：填充数独显示参考答案

\*条件：数独生成成功

\*输入：无 输出：数独答案

\*返回值类型：void类型

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Fillres(){

char num[20];

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

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

sprintf(num,"%d",table[i][j]);

outtextxy(80+30\*i,70+30\*j,num);

}

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*界面\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Interface2();

void Interface3();

void Interreadcnf(){

int flag=1;

while(flag){

peekmessage(&msg,EX\_MOUSE);

BeginBatchDraw();

cleardevice();

settextstyle(20,0,"微软雅黑");

if(button(10,10,160,40,"请输入cnf文件地址：")){

InputBox(filename,200,NULL,NULL,NULL,250,40,false);

}

if(ReadCNF(CNF,filename)==OK){

flag1=0;

outtextxy(10,60,"cnf文件读取成功");

}

else {

outtextxy(10,60,"ERROR！cnf文件读取出错");

}

if(button(760,550,100,40,"<<-return!")){

msg.message=0;

return;

}

EndBatchDraw ();

msg.message=0;

}

}

void Interprintnext(Clause \*cls){

int flag=1;

Clause \*pc;

while(flag){

msg.message=0;

peekmessage(&msg,EX\_MOUSE);

BeginBatchDraw();

cleardevice();

settextstyle(20,0,"微软雅黑");

outtextxy(10,10,"文件读取结果如下：");

int m=10,n=40,t=10;

char num[20];

for(Clause \*c=cls;c;c=c->next){

for(Literal \*ltr=c->head;ltr;ltr=ltr->next){

sprintf(num,"%d",ltr->value);

outtextxy(m,n,num);

m+=50;

}

n+=20;

if(n>=580){

n=40;

t+=450;

}

if(button(800,200,40,40,"->")){

cleardevice();

Interprintnext(pc->next);

}

if(t>600){

pc=c;

break;

}

m=t;

}

if(button(760,550,100,40,"<<-return!")){

msg.message=0;

flag=0;

}

EndBatchDraw ();

}

}

void Interprintcnf(){

int flag=1;

Clause \*c;

while(flag){

peekmessage(&msg,EX\_MOUSE);

BeginBatchDraw();

cleardevice();

settextstyle(20,0,"微软雅黑");

if(flag1){

outtextxy(10,10,"请先读取文件");

}

else{

outtextxy(10,10,"文件读取结果如下：");

int m=10,n=40,t=10;

char num[20];

for(Clause \*cls=CNF;cls;cls=cls->next){

for(Literal \*ltr=cls->head;ltr;ltr=ltr->next){

sprintf(num,"%d",ltr->value);

outtextxy(m,n,num);

m+=50;

}

n+=20;

if(n>=580){

n=40;

t+=450;

}

if(button(800,200,40,40,"->")){

cleardevice();

Interprintnext(c->next);

}

if(t>600){

c=cls;

break;

}

m=t;

}

}

if(button(760,550,100,40,"<<-return!")){

msg.message=0;

return;

}

EndBatchDraw ();

msg.message=0;

}

}

void Intersolvesat(){

int flag=1,f=0,k=0,s=0,z=0;

char num[50];

while(flag){

peekmessage(&msg,EX\_MOUSE);

BeginBatchDraw();

cleardevice();

settextstyle(20,0,"微软雅黑");

if(flag1){

outtextxy(10,10,"请先读取文件");

}

else{

if(f==0&&k==0&&z==0){

ans=(int \*)malloc(sizeof(int)\*(ltrnum+1));

for(int i=0;i<=ltrnum;i++)

ans[i]=1;

}

outtextxy(10,10,"请选择变元选择策略：");

if(f||button(10,50,60,40,"方案a") ){

if(!f){

start=clock();

t=DPLL(CNF,4);

finish=clock();

t2=(double)(finish-start)/CLOCKS\_PER\_SEC;

}

f=1;

if(t==true){

sprintf(num,"%lf",t2\*1000);

outtextxy(10,130,"该问题成功找到满足条件,用时"); outtextxy(380,130,"ms");

outtextxy(220,130,num);

flag2=1;

}

else{

outtextxy(10,100,"该问题未能找到满足条件");

}

}

if(k||button(110,50,60,40,"方案b")) {

if(!k){

start=clock();

t=DPLL(CNF,3);

finish=clock();

t1=(double)(finish-start)/CLOCKS\_PER\_SEC;

}

k=1;

if(t==true){

sprintf(num,"%lf",t1\*1000);

outtextxy(10,100,"该问题成功找到满足条件,用时"); outtextxy(380,100,"ms");

outtextxy(220,100,num);

flag2=1;

}

else{

outtextxy(10,100,"该问题未能找到满足条件");

}

}

if(z||button(210,50,60,40,"方案c")) {

if(!z){

start=clock();

t=DPLL(CNF,0);

finish=clock();

t2=(double)(finish-start)/CLOCKS\_PER\_SEC;

}

z=1;

if(t==true){

sprintf(num,"%lf",t2\*1000);

outtextxy(10,160,"该问题成功找到满足条件,用时"); outtextxy(380,160,"ms");

outtextxy(220,160,num);

flag2=1;

}

else{

outtextxy(10,100,"该问题未能找到满足条件");

}

}

if((f||z)&&k){

if(s||button(10,190,60,40,"优化率")){

if(t1<t2){

double v=t1;

t1=t2;

t2=v;

}

sprintf(num,"%lf",(t1-t2)/t1\*100);

outtextxy(10,250,"优化率"); outtextxy(180,250,"%");

outtextxy(90,250,num);

s=1;

}

}

}

if(button(760,550,100,40,"<<-return!")){

msg.message=0;

return;

}

EndBatchDraw ();

msg.message=0;

}

}

void Internextans(int l){

int flag=1,j;

while(flag){

peekmessage(&msg,EX\_MOUSE);

BeginBatchDraw();

cleardevice();

settextstyle(20,0,"微软雅黑");

char num[20];

int m=10,n=10;

for(int i=l+1;i<=ltrnum;i++){

if(ans[i])

sprintf(num,"%d",i);

else

sprintf(num,"%d",-i);

outtextxy(m,n,num);

n+=20;

if(n>=540){

n=10;

m+=50;

}

if(j<ltrnum&&button(800,200,40,40,"->")){

msg.message=0;

cleardevice();

Internextans(j);

}

if(m>=740){

j=i;

break;

}

}

if(button(760,550,100,40,"<<-return!")){

msg.message=0;

return;

}

EndBatchDraw ();

msg.message=0;

}

}

void Interprintans(){

int flag=1,j;

while(flag){

peekmessage(&msg,EX\_MOUSE);

BeginBatchDraw();

cleardevice();

settextstyle(20,0,"微软雅黑");

if(flag1){

outtextxy(10,10,"请先读取文件");

}

else if(!t){

outtextxy(10,10,"该问题未能找到满足条件");

}

else{

char num[20];

int m=10,n=10;

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

if(ans[i])

sprintf(num,"%d",i);

else

sprintf(num,"%d",-i);

outtextxy(m,n,num);

n+=20;

if(n>=540){

n=10;

m+=50;

}

if(j<ltrnum&&button(800,200,40,40,"->")){

msg.message=0;

cleardevice();

Internextans(j);

}

if(m>=740){

j=i;

break;

}

}

}

if(button(760,550,100,40,"<<-return!")){

msg.message=0;

return;

}

EndBatchDraw ();

msg.message=0;

}

}

void Intersave(){

int flag=1;

while(flag){

peekmessage(&msg,EX\_MOUSE);

BeginBatchDraw();

cleardevice();

settextstyle(20,0,"微软雅黑");

if(button(10,10,160,40,"=>请输入保存文件地址：")){

InputBox(filename2,200,NULL,NULL,NULL,250,40,false);

}

if(saveAns(flag2,t1)==OK){

flag1=0;

outtextxy(10,60,"答案已保存");

}

else {

outtextxy(10,60,"ERROR!答案保存失败");

}

if(button(760,550,100,40,"<<-return!")){

msg.message=0;

return;

}

EndBatchDraw ();

msg.message=0;

}

}

void Interprintres();

void Interverifyres();

void Intersolvexsudoku();

void Intercreatxsudoku(){

int flag=1,f1=1,f2=1,f3=1;

while(flag){

msg.message=0;

peekmessage(&msg,EX\_MOUSE);

BeginBatchDraw();

cleardevice();

settextstyle(20,0,"微软雅黑");

if(f1){

f1=0;

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

for(int j=1;j<=N;j++)

table[i][j]=0;

}

CreatXSudoku();

}

if(f2){

outtextxy(10,10,"=》请选择难度");

if(f2&&button(10,50,200,40,"->>op.1 level 1.(Simple)")){

f2=0;

holesnum=12;

}

if(f2&&button(10,100,200,40,"->>op.2 level 2.(Normal)")){

f2=0;

holesnum=28;

}

if(f2&&button(10,150,200,40,"->>op.3 level 3.(Difficult)")){

f2=0;

holesnum=42;

}

}

if(f1==0&&f2==0){

if(f3){

creatHoles(holesnum);

f3=0;

}

outtextxy(10,10,"X-Sudoku生成完毕！");

}

if(f3==0&&button(760,500,100,40,"->>solve!")){

msg.message=0;

Intersolvexsudoku();

}

if(button(760,550,100,40,"<<-return!")){

msg.message=0;

return;

}

EndBatchDraw ();

}

}

void Intersolvexsudoku(){

int flag=1;

while(flag){

msg.message=0;

peekmessage(&msg,EX\_MOUSE);

BeginBatchDraw();

cleardevice();

settextstyle(20,0,"微软雅黑");

drawtable();

Filltable();

if(button(760,450,100,40,"->>vertify!")){

msg.message=0;

Interverifyres();

}

if(button(760,500,100,40,"->>answer!")){

msg.message=0;

Interprintres();

}

if(button(760,550,100,40,"<<-return!")){

msg.message=0;

return;

}

EndBatchDraw ();

}

}

void Interverifyres(){

int flag=1,f1=DPLLVerification();

while(flag){

msg.message=0;

peekmessage(&msg,EX\_MOUSE);

BeginBatchDraw();

cleardevice();

settextstyle(20,0,"微软雅黑");

if(f1){

f1=1;

outtextxy(10,10,"GREAT!恭喜你成功解出答案！");

}

else{

outtextxy(10,10,"很遗憾，答案错误，再试试吧！");

outtextxy(10,50,"参考答案如下：");

if(button(30,100,100,40,"->>answer!")){

msg.message=0;

Interprintres();

}

}

if(button(760,550,100,40,"<<-return!")){

msg.message=0;

return;

}

EndBatchDraw ();

}

}

void Interprintres(){

int flag=1;

while(flag){

msg.message=0;

peekmessage(&msg,EX\_MOUSE);

BeginBatchDraw();

cleardevice();

settextstyle(20,0,"微软雅黑");

outtextxy(10,10,"参考答案如下：");

drawtable();

Fillres();

if(button(760,550,100,40,"<<-return!")){

msg.message=0;

return;

}

EndBatchDraw ();

}

}

void Fillans(){

char num[20];

int x=0,y=0,z=0;

for(int i=1;i<=729;i++){

if(ans[i]>0){

z=i%9;

if(z==0)

z=9;

y=(i-z)/9%9+1;

x=(i-z)/81+1;

sprintf(num,"%d",z);

outtextxy(80+30\*x,70+30\*y,num);

}

}

}

void InterDPLLsolver(){

ShuduToCnf();

ReadCNF(XSudoku,"D:\\桌面\\CS202303-U202310917-张佳艺\\X-Sudoku\\save\\shudu.cnf");

ans=(int \*)malloc(sizeof(int)\*(ltrnum+1));

for(int i=0;i<=729;i++)

ans[i]=1;

DPLL(XSudoku,0);

int flag=1;

while(flag){

msg.message=0;

peekmessage(&msg,EX\_MOUSE);

BeginBatchDraw();

cleardevice();

settextstyle(20,0,"微软雅黑");

drawtable();

Fillans();

if(button(760,550,100,40,"<<-return!")){

msg.message=0;

return;

}

EndBatchDraw ();

}

}

//总菜单

void Interface1(){

initgraph(880,600) ;

HWND hwnd = GetHWnd();

SetWindowPos(hwnd,NULL,100,100,0,0,SWP\_NOSIZE|SWP\_NOZORDER);

setbkcolor(RGB(230,195,195));

setbkmode(TRANSPARENT);

int flag=1;

while(flag){

peekmessage(&msg,EX\_MOUSE);

BeginBatchDraw();

cleardevice();

settextstyle(24,0,"微软雅黑");

settextcolor(RGB(72,61,139));

outtextxy(10,10,"=>欢迎使用本系统，一起来玩对角线数独吧！");

settextstyle(20,0,"微软雅黑");

setlinecolor(RGB(70,130,180));

setlinestyle(PS\_SOLID,2);

outtextxy(230,40,"MENUE");

line(10,50,228,50); line(280,50,510,50);

setfillcolor(RGB(176,196,222));

fillrectangle(10,70,510,360);

if(button(40,90,160,40,"->>op.1 SAT"))

Interface2();

if(button(40,180,160,40,"->>op.2 X-Sudoku"))

Interface3();

if(button(380,310,120,40,"<<-op.0 exit!"))

flag=0;

EndBatchDraw ();

msg.message=0;

}

}

//SAT菜单

void Interface2(){

int flag=1;

while(flag){

msg.message=0;

peekmessage(&msg,EX\_MOUSE);

BeginBatchDraw();

cleardevice();

settextstyle(24,0,"微软雅黑");

settextcolor(RGB(72,61,139));

outtextxy(10,10,"=》op.1 SAT");

settextstyle(20,0,"微软雅黑");

setlinecolor(RGB(70,130,180));

setlinestyle(PS\_SOLID,2);

outtextxy(230,40,"MENUE");

line(10,50,228,50); line(280,50,510,50);

setfillcolor(RGB(176,196,222));

fillrectangle(10,70,510,360);

if(button(40,90,160,40,"->>op.1 read cnf"))

Interreadcnf();

if(button(280,90,160,40,"->>op.2 print cnf"))

Interprintcnf();

if(button(40,180,160,40,"->>op.3 solve SAT"))

Intersolvesat();

if(button(280,180,160,40,"->>op.4 print answer"))

Interprintans();

if(button(40,270,200,40,"->>op.5 save answer to file"))

Intersave();

if(button(380,310,120,40,"<<-op.0 return!"))

flag=0;

EndBatchDraw ();

msg.message=0;

}

}

//X-Sudoku菜单

void Interface3(){

int flag=1;

while(flag){

msg.message=0;

peekmessage(&msg,EX\_MOUSE);

BeginBatchDraw();

cleardevice();

settextstyle(24,0,"微软雅黑");

settextcolor(RGB(72,61,139));

outtextxy(10,10,"=》op.2 X-Sudoku");

settextstyle(20,0,"微软雅黑");

setlinecolor(RGB(70,130,180));

setlinestyle(PS\_SOLID,2);

outtextxy(230,40,"MENUE");

line(10,50,228,50); line(280,50,510,50);

setfillcolor(RGB(176,196,222));

fillrectangle(10,70,510,360);

if(button(40,90,210,40,"->>op.1 creat a X-Sudoku"))

Intercreatxsudoku();

if(button(280,90,210,40,"->>op.2 solve the X-Sudoku"))

Intersolvexsudoku();

if(button(40,180,210,40,"->>op.3 verify the result"))

Interverifyres();

if(button(280,180,210,40,"->>op.4 print the answer"))

Interprintres();

if(button(40,270,210,40,"->>op.5 DPLL solver"))

InterDPLLsolver();

if(button(380,310,120,40,"<<-op.0 return!"))

flag=0;

EndBatchDraw ();

msg.message=0;

}

}

int main(){

Interface1(); //游戏开始，进入总菜单

return 0;

}

**2 main.h**

#include "define.h" // 数据结构和变量定义

#include "cnfparser.h" // 读取cnf文件，打印cnf文件内容和保存文件

#include "solver.h" // DPLL求解器

#include "X-Sudoku.h" // 数独生成和DPLL验证

**3 define.h**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <math.h>

#include <time.h>

#define TRUE 1

#define FALSE 0

#define OK 1

#define ERROR 0

#define N 9

typedef struct literal{

int value;

struct literal \*pre;

struct literal \*next;

}Literal; //文字

typedef struct clause{

int num; //子句中文字数目

struct literal \*head; //指向子句中第一个文字

struct clause \*next; //指向下一个子句

}Clause;

int cx=GetSystemMetrics (SM\_CXFULLSCREEN);

int cy=GetSystemMetrics (SM\_CYFULLSCREEN);

int op=1,op1,op2,flag3,op3;

int flag1=1; //文件读取标志

int flag2=0; //DPLL求解结果

int holesnum=0; //挖洞数目

clock\_t start,finish; //算法开始时间和结束时间

double t1,t2; //优化前后求解时间

typedef int status;

char filename[188],filename2[188]; //cnf文件地址和保存文件地址

int ltrnum,clsnum; //文字数量，子句数量

int inf=0x2fffffff;

Clause \*CNF=NULL; // cnf文件读取结果

int \*ans; // DPLL求解答案

int table[10][10],shudu[10][10],answer[10][10];

//数独盘 ，答案和解题记录

int l;

Clause \*XSudoku=NULL; //数独转化为SAT

status t; //DPLL函数返回值

int n;

**4 cnfparser.h**

//读入文字

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*函数名称：readLit

\*函数功能：读取literal

\*条件：文件成功打开

\*输入：指向文件的指针 输出：literal的值

\*返回值类型：int类型

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int readLit(FILE\* fp)

{

int flag=0;

int x=0, f=1;

char ch;

ch=getc(fp);

while(ch<'0'||ch>'9')

{

if (ch=='-'){

f=-1;

ch=getc(fp);

}

else

return inf;

}

while(ch>='0'&&ch<='9')

{

x=x\*10+ch-'0';

ch=getc(fp);

if (ch==-1) //文件结束

flag=1;

}

if (!flag&&ch!=' '&&ch!='\n')

return inf;

return x\*f;

}

//读取cnf文件

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*函数名称：ReadCNF

\*函数功能：读取cnf文件

\*条件：文件成功打开

\*输入：存储文件读取结果的结构地址 输出：文件读取状态

\*返回值类型：status类型

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

status ReadCNF(Clause\* &cnf,const char \*filename)

{

FILE\* fp;

fp = fopen(filename, "r");

if (!fp)

return ERROR;

char ch;

char comment[5] = { ' ', 'c', 'n', 'f', ' ' };

while ((ch = getc(fp)) == 'c') //跳出注释行

while ((ch = getc(fp)) != '\n')

continue;

if (ch != 'p')

return ERROR;

for (int i = 0; i < 5; i++) //读入p cnf

if ((ch = getc(fp)) != comment[i])

{

return ERROR;

}

if ((ltrnum = readLit(fp)) == inf || (clsnum = readLit(fp)) == inf) //读入文字数目和子句数目

return ERROR;

cnf=(Clause\*)malloc(sizeof(Clause)); //初始化

cnf->next = NULL;

cnf->head = (Literal\*)malloc(sizeof(Literal));

cnf->head->pre = NULL;

cnf->head->next = NULL;

cnf->num = 0;

Literal\* pltr;

Clause\* pcls;

pcls=cnf;

pltr=cnf->head;

for (int i = 1; i <= clsnum; i++) //读入子句

{

int value;

if ((value = readLit(fp)) == inf)

return ERROR;

while (value)

{

pcls->num++;

pltr->value = value;

pltr->next = NULL;

if ((value = readLit(fp)) == inf)

return ERROR;

if (value){

pltr->next = (Literal\*)malloc(sizeof(Literal));

pltr->next->pre=pltr;

}

pltr = pltr->next;

}

if (i==clsnum)

{

pcls->next = NULL;

break;

}

pcls->next = (Clause\*)malloc(sizeof(Clause));

pcls->next->num = 0;

pcls->next->head = (Literal\*)malloc(sizeof(Literal));

pcls=pcls->next;

pltr=pcls->head;

}

fclose(fp);

return OK;

}

//保存cnf算例结果

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*函数名称：saveAns

\*函数功能：保存SAT求解结果

\*条件：文件成功打开

\*输入：DPLL求解结果和时间 输出：文件存储状态

\*返回值类型：status类型

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

status saveAns(int flag,double time)

{

FILE\* fp;

if (!(fp = fopen(filename2, "w+")))

{

return ERROR;

}

fprintf(fp, "s %d\nv ", flag);

//求解结果/1表示满足，0表示不满足，-1表示在限定时间内未完成求解

if (flag) //满足时写入结果

for (int i = 1; i <= ltrnum; i++)

{

if (ans[i])

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

else

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

}

fprintf(fp, "\nt %lf ms", t2\*1000);

fclose(fp);

return OK;

}

**5 solver.h**

//DPLL

status findUnitClause(Clause \*&cnf) { //寻找单子句 ，找到返回单子句的文字值，否则返回0

Clause\* pc;

Literal\* pl;

for (pc = cnf; pc; pc=pc->next)

{

pl=pc->head;

if (pl->next==NULL)

{

return pl->value;

}

}

return 0;

}

status IsEmpty(Clause \*cnf){ //判断是否存在空子句 ，存在返回TRUE，否则返回FALSE

Clause \*p=cnf;

while(p){

if(p->head==NULL)

return TRUE;

p=p->next;

}

return FALSE;

}

void dltClause(Clause \*&cnf, Clause \*&opc) //在cnf子句集中子句 opc

{

if (opc == cnf) //如果opc是头子句

cnf = cnf->next;

else //否则

{

Clause\* pc = cnf;

while (pc && pc->next != opc)

pc = pc->next;

pc->next = opc->next;

}

Literal \*pl1, \*pl2; //先删除opc里的文字

for (pl1 = opc->head; pl1; pl1=pl2){

pl2 = pl1->next;

free(pl1);

}

free(opc);

opc = NULL;

}

void dltLiteral(Literal \*&head, Literal \*&opl) //传入子句的头结点，在子句中删除结点opl

{

Literal \*pl=head;

if (opl == head) //opl是头节点

head = head->next;

else //否则

{

while (pl && pl->next != opl)

pl = pl->next;

pl->next = pl->next->next;

}

free(opl);

opl = NULL;

}

void Simplification(Clause \*&cnf,int val){ //传入单子句的文字值，根据单子句规则化简cnf子句集

Clause \*pc2;

for(Clause \*pc1=cnf;pc1;pc1=pc2)

{

pc2=pc1->next;

for(Literal \*pl = pc1->head; pl ; pl = pl->next)

{

if (pl->value == val) //删除所有包含val的子句

{

dltClause(cnf,pc1);

break;

}

if (pl->value == -val) //删除所有包含-val的节点

{

dltLiteral(pc1->head,pl);

pc1->num--;

break;

}

}

}

}

void getTcnf(Clause \*&Tcnf,Clause \*cnf){ //复制cnf子句集，便于分裂后回溯

Clause \*Tpc, \*pc;

Literal \*Tpl, \*pl;

Tcnf = (Clause\*)malloc(sizeof(Clause));

Tcnf->head = (Literal \*)malloc(sizeof(Literal));

Tcnf->head->pre = NULL;

Tcnf->head->next = NULL;

Tcnf->next = NULL;

Tcnf->num = 0;

for (pc=cnf,Tpc=Tcnf;pc;pc=pc->next,Tpc=Tpc->next)

{

for (pl= pc->head,Tpl=Tpc->head;pl;pl=pl->next,Tpl=Tpl->next)

{

Tpc->num++;

Tpl->value = pl->value;

if (pl->next != NULL){

Tpl->next = (Literal \*)malloc(sizeof(Literal));

Tpl->next->next = NULL;

Tpl->next->pre = Tpl;

}

else

Tpl->next=NULL;

}

if (pc->next != NULL){

Tpc->next=(Clause \*)malloc(sizeof(Clause));

Tpc->next->head = (Literal \*)malloc(sizeof(Literal));

Tpc->next->head->pre = NULL;

Tpc->next->head->next = NULL;

Tpc->next->next = NULL;

Tpc->next->num = 0;

}

else

Tpc->next=NULL;

}

}

void addClause(int opv, Clause\*& cnf) //在cnf子句集中加入文字值为opv的单子句

{

Clause \*newcls=(Clause\*)malloc(sizeof(Clause));

newcls->head = (Literal\*)malloc(sizeof(Literal));

newcls->head->value=opv;

newcls->head->pre=NULL;

newcls->head->next = NULL;

newcls->next = cnf;

newcls->num = 1;

cnf=newcls;

}

void FreeTcnf(Clause \*&Tcnf){

Clause\* pc1, \* pc2;

Literal\* pl1, \* pl2;

pc1 = Tcnf;

while (pc1)

{

pl1 = pc1->head;

while (pl1){

pl2=pl1->next;

free(pl1);

pl1=pl2;

}

pc2 = pc1->next;

free(pc1);

pc1 = pc2;

}

Tcnf = NULL;

}

//分裂变元选择 /\*经比较最终选择getFirstLtr，getMaxLtr\_improved3，getMaxLtr\_improved4作为选择策略\*/

int getFirstLtr(Clause\* cnf)

//选择子句集第一个子句的头文字值

{

Clause \* pc = cnf;

if(pc->head->value>0)

return pc->head->value;

return -pc->head->value;

}

IntgetMaxLtr(Clause\*cnf)

//选择出现次数最多的变元

{

int\* time = (int\*)malloc(sizeof(int) \* (ltrnum \* 2 + 1));

for (int i = 0; i <= ltrnum \* 2; i++)

time[i] = 0;

for (Clause\* pc = cnf; pc; pc = pc->next)

for (Literal\* pl = pc->head; pl; pl = pl->next)

{

if (pl->value > 0)

time[pl->value]++;

else

time[ltrnum - pl->value]++;

}

int maxltr, maxTimes = 0;

for (int i = 1; i <= ltrnum; i++)

if (time[i] > maxTimes)

maxTimes = time[i], maxltr = i;

for (int i = ltrnum + 1; i <= ltrnum \* 2; i++)

if (time[i] > maxTimes)

maxTimes = time[i], maxltr = ltrnum - i;

free(time);

return maxltr;

}

int getMaxLtr\_improved1(Clause\* cnf)

//选取总出现次数最多的变元

{

int\* time = (int\*)malloc(sizeof(int) \* (ltrnum \* 2 + 1));

for (int i = 0; i <= ltrnum \* 2; i++)

time[i] = 0;

for (Clause\* pc = cnf; pc; pc = pc->next)

for (Literal\* pl = pc->head; pl; pl = pl->next)

{

if (pl->value > 0)

time[pl->value]++;

else

time[ltrnum - pl->value]++;

}

int maxltr, maxTimes = 0;

for (int i = 1; i <= ltrnum; i++)

if (time[i]+time[ltrnum+i] > maxTimes)

maxTimes = time[i]+time[ltrnum+i] , maxltr = i;

free(time);

return maxltr;

}

int getMaxLtr\_improved2(Clause\* cnf)

//选取在短子句总出现次数最多的变元

{

double\* time = (double\*)malloc(sizeof(double) \* (ltrnum + 1));

for (int i = 0; i <= ltrnum; i++)

time[i] = 0.0;

for (Clause\* pc = cnf; pc; pc = pc->next)

for (Literal\* pl = pc->head; pl; pl = pl->next)

if(pl->value>0)

time[pl->value] += pow(2.0, (double)(-(pc->num)));

else

time[-pl->value] += pow(2.0, (double)(-(pc->num)));

double maxTime = 0 ;

int maxltr=0 ;

for (int i = 1; i <= ltrnum; i++)

if (time[i] > maxTime)

maxTime = time[i], maxltr = i;

free(time);

return maxltr;

}

int getMaxLtr\_improved3(Clause\* cnf)

//选取在短子句出现次数最多的文字

{

double\* time = (double\*)malloc(sizeof(double) \* (ltrnum \* 2 + 1));

for (int i = 0; i <= ltrnum \* 2; i++)

time[i] = 0.0;

for (Clause\* pc = cnf; pc; pc = pc->next)

for (Literal\* pl = pc->head; pl; pl = pl->next)

{

if (pl->value > 0)

time[pl->value] += pow(2.0, (double)(-(pc->num)));

else

time[ltrnum - pl->value] += pow(2.0, (double)(-(pc->num)));

}

double maxTime = 0.0;

int maxl=0;

for (int i = 1; i <= ltrnum; i++)

if (time[i] + time[i + ltrnum] > maxTime)

{

maxTime = time[i] + time[i + ltrnum], maxl = i;

}

if (time[maxl] < time[maxl + ltrnum])

maxl = -maxl;

free(time);

return maxl;

}

status DPLL(Clause \*&cnf,int flag3){

//传入cnf子句集和变元选择策略，DPLL求解SAT

int opv;

opv = findUnitClause(cnf);

while(opv){ //单子句规则化简

if(opv>0){

ans[opv]=1;

}

else{

ans[-opv]=0;

}

Simplification(cnf,opv);

if(cnf==NULL) //化简为空子集 ，可满足

return TRUE;

else if(IsEmpty(cnf)==TRUE) //出现空子句，不满足

return FALSE;

opv = findUnitClause(cnf);

}

//分裂策略化简

switch(flag3){ //选择变元

case 0:

opv=getFirstLtr(cnf);

break;

case 1:

opv=getMaxLtr(cnf);

break;

case 2:

opv=getMaxLtr\_improved1(cnf);

break;

case 3:

opv=getMaxLtr\_improved2(cnf);

break;

case 4:

opv=getMaxLtr\_improved3(cnf);

}

Clause \*Tcnf;

getTcnf(Tcnf,cnf); //取opv为真

addClause(opv,Tcnf);

if(DPLL(Tcnf,flag3)==TRUE)

return TRUE;

FreeTcnf(Tcnf);

getTcnf(Tcnf,cnf); //取opv为假

addClause(-opv,Tcnf);

status res=DPLL(Tcnf,flag3);

FreeTcnf(Tcnf);

return res;

}

**6 X-Sudoku.h**

status IsConflict(int k){

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

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

if(k==table[i][j])

return 0;

}

}

return 1;

}

void PrintXSudoku(void);

void CreatXSudoku(void){ //生成一个N\*N的完整对角线数独

srand(time(NULL));

for(int i=4;i<=6;i++){ //随机填充中间宫格

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

while(table[i][j]==0){

l=rand()%9+1;

if(IsConflict(l)){

table[i][j]=l;

}

}

}

}

for(int i=4;i<=6;i++){ //填充四边宫格

int t=1;

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

if(i==4){

table[i+1][t]=table[i][j];

table[i+2][t+6]=table[i][j];

t++;

}

else if(i==5){

table[i+1][t]=table[i][j];

table[i-1][t+6]=table[i][j];

t++;

}

else{

table[i-2][t]=table[i][j];

table[i-1][t+6]=table[i][j];

t++;

}

}

}

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

int t=1;

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

if(j==4){

table[t][j+1]=table[i][j];

table[t+6][j+2]=table[i][j];

t++;

}

else if(j==5){

table[t][j+1]=table[i][j];

table[t+6][j-1]=table[i][j];

t++;

}

else{

table[t][j-2]=table[i][j];

table[t+6][j-1]=table[i][j];

t++;

}

}

}

table[1][1]=table[3][7]=table[9][2]=table[7][8]=table[5][6];

table[1][2]=table[2][8]=table[7][1]=table[8][7]=table[6][6];

table[1][3]=table[2][9]=table[7][2]=table[8][8]=table[6][4];

table[2][1]=table[1][9]=table[8][3]=table[7][7]=table[6][5];

table[2][2]=table[3][8]=table[8][1]=table[9][7]=table[4][6];

table[2][3]=table[3][9]=table[8][2]=table[9][8]=table[4][4];

table[3][1]=table[1][7]=table[9][3]=table[7][9]=table[5][5];

table[3][2]=table[1][8]=table[7][3]=table[9][9]=table[5][4];

table[3][3]=table[2][7]=table[9][1]=table[8][9]=table[4][5];

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

for(int j=0;j<=N;j++){

shudu[i][j]=table[i][j];

}

}

}

status IsTrue(int m,int n,int num){ //填入是否正确

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

if(i!=m&&num==shudu[i][n])

return 0;

}

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

if(j!=n&&num==shudu[m][j])

return 0;

}

int x=3\*((m-1)/3)+1;

int y=3\*((n-1)/3)+1;

for(int i=0;i<3;i++){

for(int j=0;j<3;j++){

if(x+i!=m&&y+j!=n&&num==shudu[x+i][y+j])

return 0;

}

}

if(m==n){

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

if(i!=m&&i!=n&&num==shudu[i][i])

return 0;

}

}

if(m+n==10){

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

if(i!=m&&10-i!=n&&num==shudu[i][10-i])

return 0;

}

}

return 1;

}

status solve\_XSudoku(){ //验证挖洞后可解性

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

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

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

for(int num=1;num<=N;num++){

if(IsTrue(i,j,num)){

shudu[i][j]=num;

if(solve\_XSudoku()){

shudu[i][j]=0;

return 1;

}

}

}

shudu[i][j]=0;

return 0;

}

}

}

}

void creatHoles(int holes){ //挖洞法生成数独棋盘

int m,n;

for(;1;){

do{

m=rand()%9+1;

n=rand()%9+1;

if(shudu[m][n])

break;

}while(1);

int tmp=shudu[m][n];

shudu[m][n]=0;

if(solve\_XSudoku()){

holes--;

if(holes==0)

break;

}

else{

shudu[m][n]=tmp;

}

}

printf("X-Sudoku已成功生成：\n");

}

void ShuduToCnf(){

FILE\* fp = fopen("D:\\桌面\\CS202303-U202310917-张佳艺\\X-Sudoku\\save\\shudu.cnf", "w+");

if (!fp) exit(ERROR);

fprintf(fp, "c\np cnf 729 %d\n", 6642 + 81 - holesnum);

int i,j,k;

for (i = 1; i <= N; i++) {

for (j = 1; j <= N; j++) {

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

fprintf(fp, "%d 0\n", (i-1) \* 81 + (j-1) \* 9 + shudu[i][j]);

}

}

}

for (i = 1; i <= 9; i++) { //格约束

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

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

fprintf(fp, "%d ", (i-1) \* 81 + (j-1) \* 9 + k);

}

fprintf(fp, "0\n");

}

}

for (j = 1; j <= 9; j++) { //列约束

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

for (i = 1; i <= 8; i++) {

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

fprintf(fp, "%d ", (-1)\*((i-1) \* 81 + (j-1) \* 9 + k));

fprintf(fp, "%d 0\n", (-1) \* ((l-1) \* 81 + (j-1) \* 9 + k));

}

}

}

}

for (i = 1; i <= 9; i++) { //行约束

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

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

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

fprintf(fp, "%d ", (-1) \* ((i-1) \* 81 + (j-1) \* 9 + k));

fprintf(fp, "%d 0\n", (-1) \* ((i-1) \* 81 + (l-1) \* 9 + k));

}

}

}

}

for (k = 1; k <= 9; k++) { //3\*3约束

for (i = 0; i <= 2; i++) {

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

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

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

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

fprintf(fp, "%d ", (-1) \* ((3 \* i + x - 1) \* 81 + (3 \* j + y - 1) \* 9 + k));

fprintf(fp, "%d 0\n", (-1) \* ((3 \* i + x - 1) \* 81 + (3 \* j + z - 1) \* 9 + k));

}

}

}

}

}

}

int m;

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

for (i = 0; i <= 2; i++) {

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

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

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

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

for (m = 1; m <= 3; m++) {

fprintf(fp, "%d ", (-1) \* ((3 \* i + x - 1) \* 81 + (3 \* j + y - 1) \* 9 + k));

fprintf(fp, "%d 0\n", (-1) \* ((3 \* i + z - 1) \* 81 + (3 \* j + m - 1) \* 9 + k));

}

}

}

}

}

}

}

for (i = 1; i <= 9; i++) { //对角线约束

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

if (i + j <= 9) {

for (k = i + j + 1; k <= 9; k++) {

fprintf(fp, "%d ", (-1) \* ((i-1) \* 81 + (j-1) \* 9 + k));

fprintf(fp, "%d 0\n", (-1) \* ((i - 1 + k - 9 - 1) \* 81 + (j + 1 + k - 9 - 1) \* 9 + k));

}

}

}

}

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

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

if (i + (9 - j) <= 9) {

for (k = i + (9 - j) + 1; k <= 9; k++) {

fprintf(fp, "%d ", (-1) \* ((i-1) \* 81 + (j-1) \* 9 + k));

fprintf(fp, "%d 0\n", (-1) \* ((i - 1 + k - 10 - 1) \* 81 + (j + 1 - k + 10 - 1) \* 9 + k));

}

}

}

}

fclose(fp);

return;

}

status DPLLVerification(){ //DPLL验证答案可满足性

ShuduToCnf();

ReadCNF(XSudoku,"D:\\桌面\\CS202303-U202310917-张佳艺\\X-Sudoku\\save\\shudu.cnf");

int num;

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

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

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

if(k==shudu[i][j])

num=i\*100+j\*10+shudu[i][j];

else

num=0-(i\*100+j\*10+k);

Simplification(XSudoku,num);

if(XSudoku==NULL)

return TRUE;

else if(IsEmpty(XSudoku)==TRUE)

return FALSE;

}

}

}

}