文 华 学 院

0学号 200501050026 专业 软件工程 班级 软件2001 学生姓名 梁艺潇 **0**

2021～2022学年度第二学期《数据结构课程设计》考查试卷（A卷）

装订线内请勿答题

课程性质：必修 使用范围：本科

考核时间：2022年 月 日

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 考试类型 | 论文 | 大作业 | 调研报告 | 其他 | 总分 |  |
|  |  |  | √ | 阅卷老师签名 |  |

试 题：

主要功能

1.交通咨询系统的设计并实现

2.平衡二叉排序树的设计实现

3. 内部排序算法研究

4. 数据结构实验系统

5. 自拟题目

要 求：

1.每人任选一题，独立完成；

1. 各题须说明课程设计目的、内容和要求
2. 实验数据和实验结果均以计算机截图方式表示；
3. 按设计报告格式完成课程设计报告。



**数据结构**

**课程设计报告**

**课题名称： 基于SAT的数独游戏求解程序**

**姓 名： 梁艺潇**

**学 号： 200501050026**

**院 系： 信息科学与技术**

**专业班级： 软件 2001**

**指导教师： 祝建华**

**完成日期： 2022 年 月 日**

**目 录**

### 课程设计目的

### 课程设计内容和要求

### 课程设计总体方案及分析

### 结束语

### 附录（源代码）

### 参考文献

# 1.课程设计目的

基于“程序设计”综合课程设计实践课程规划原则及其在计算机相关专业人才培养中的地位，其应该体现与达到如下目标：

⑴综合性训练目标：在该课程中涉及C语言的主要编程要素，如典型的数据类型与控制结构；覆盖多种典型的数据结构如线性结构、二叉树与树结构、图结构及查找表结构等。从先前实验课的单要素或单一结构训练向多要素，多结构综合应用训练转变。

⑵培养应用问题的求解能力：程序设计是为问题求解服务的，提高对应用问题进行分析，数据抽象与建模，及问题定义与功能划分等综合分析与表示能力。

⑶程序编写向程序设计转化：在实验课程中，老师基本描述了相关数据结构，程序框架及主要算法，基于此进行程序编写训练，其属于验证与复现性编程实践。综合程序设计要求同学们基于对应用问题的分析，建立求解模型，设计数据结构与主要算法，从而进行程序设计，更多地体现“设计”的内涵与份量。

⑷进一步培养编程规范性与工程化素养：通过“程序设计”综合课程设计实践进一步培养良好的规范性编程习惯，以及一定的程序设计与软件开发的工程化素养，按照问题定义、必要的需求分析、系统设计、编程实现、程序测试分析及编制程序设计综合课程设计报告的流程组织本实践课程的开展与进行，形成初步的工程化程序设计素养。

# 2．课程设计内容和要求

## 2.1问题描述

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。

## 2.2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。
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]。
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[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。
6. **SAT应用：**将数独游戏[5]问题转化为SAT问题[6-8]，并集成到上面的求解器进行问题求解，游戏可玩，具有一定的/简单的交互性。应用问题归约为SAT问题的具体方法可参考文献[3]与[6-8]。

# 3.课程设计总体方案及分析

## 3.1 问题分析

本次课程设计的实现功能重点在于解决SAT问题，其中DPLL算法是经典的SAT完备型求解算法，对给定的一个SAT问题实例，理论上可判定其是否满足，满足时可给出对应的一组解。所以本程序设计的算法与框架基于DPLL算法进行设计与改进。

##### DPLL算法思想：

DPLL算法是一种基于树的回溯算法，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.

根据上述规则可不断对公式化简，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3.1所示。



图 3.1 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* ),为了优化执行效率，一般用非递归实现。

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);

return DPLL(*S* ∪¬*v*);

}

##### DPLL改进思路：

首先假定在初始情况下，不存在已经确认真假的变元，我们将其称为未知变元。算法的主循环则通过对选取的未知变元进行赋值操作，从而进入一个分支，这个过程我们称为判断。

进入分支后，则开始对子句集进行分析，分析当前分支状态下子句集是否会出现不满足情况，如果未出现不满足情况则继续选取未知变元进行判断。如果出现不满足情况则进行回溯操作。回溯到0判定级则表示即使未进入任何一个判定分支，该输入实例仍然是不可满足的，在这种情况下，求解器将给出该被求解实例不可满足的结论。其中回溯操作为回退到上一判定级，如果所选定的未知变量已经判断过两次，既选定变量所赋值的两种状态已经分析完毕，则继续退回上一判断级。

在进入分支后的子句分析过程中，由与所选变元被赋值，可能会使子句集的某些子句变为只包含一个文字的子句，那么可以将这个子句中的文字所对应变量进行直接判断，既可以确定下一判断级分支，从而减少分支选择提高效率。

对应改进的DPLL算法中，每次选择的未知变元是从所有未知变元中所选取影响力最大的变元。影响力为在未被消除子句中变元所出的子句数量。对于影响力由读取cnf文件创建十字链表时进行初始化统计。

##### 物理结构设计：

十字链表节点：

struct crossListNode

{

int id //文字（子句被消除的原因）

bool falseTrue //节点是否消除(子句是否消除)

crossListNode\* up //上节点

crossListNode\* down //下节点

crossListNode\* left //左节点

crossListNode\* right //右节点

};

变元头节点：

struct nodeUphead

{

int leng //影响力

bool falseTrue //变元真假

bool type //是否处理

bool danzj //是否作为单子句处理

crossListNode\* begin //变元列的起始地址

crossListNode\* fail //变元列的结束地址

};

DPLL类：

class CnfParser

{

int boolsize //变元数量

int clausesize //子句数量

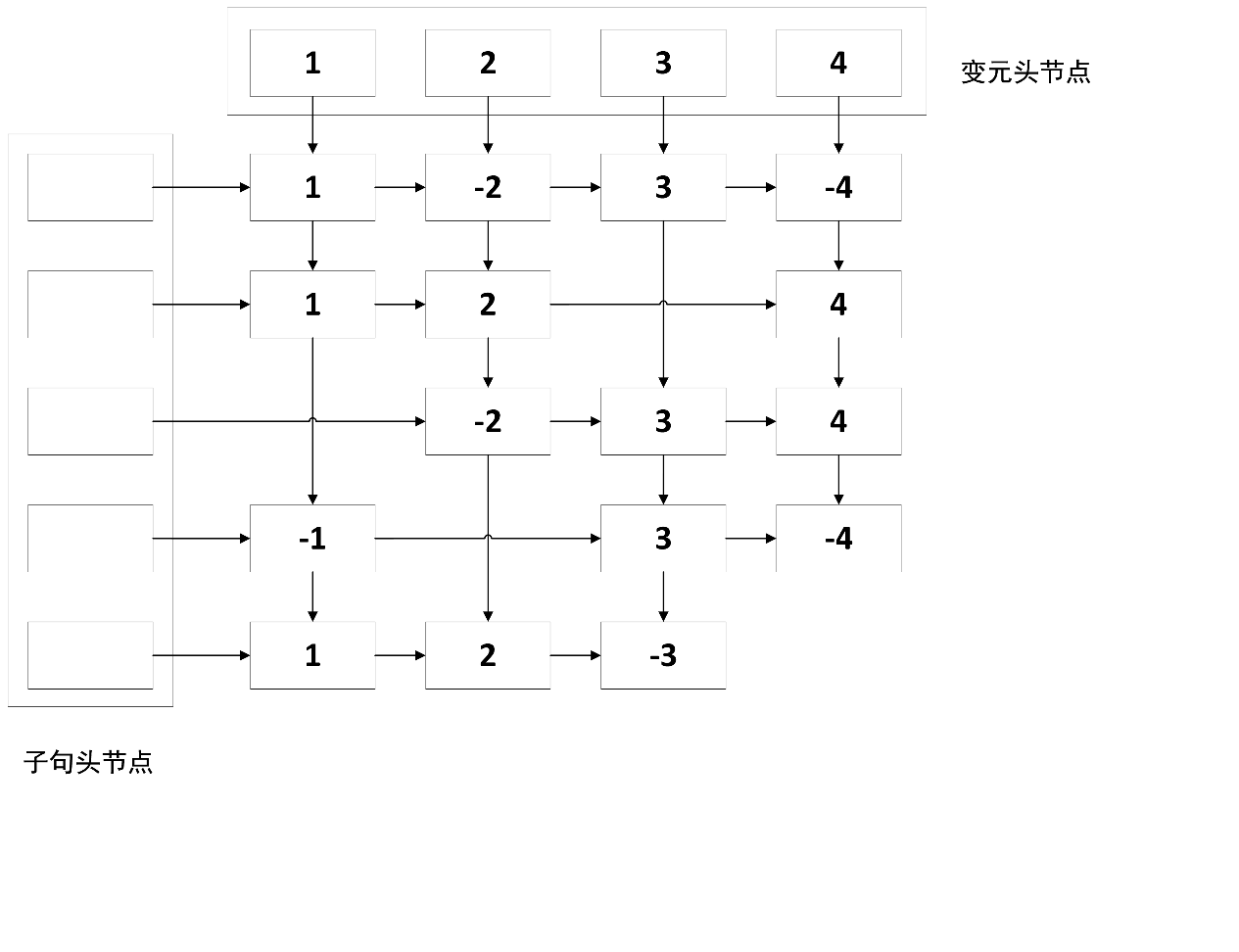
crossListNode \*root //子句起始节点

unordered\_map<int, nodeUphead \*> vmap //变元编号头节点映射表

Deque<int> \*clause //待处理单子句队列

Deque<int> \*back //栈(保存状态)

};



## 3.2 系统功能模块设计

**…**

## 3.3 测试数据与结果

# 4．结束语

# …

# 5． 附录（源代码）

**…**

## 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>

[6] Tjark Weber. A sat-based sudoku solver. In 12th International Conference on Logic forProgramming, 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 InternationalSymposium 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] Robert Ganian and Stefan Szeider. Community Structure Inspired Algorithms for SAT and #SAT. International Conference on Theory and Applications of Satisfiability Testing(SAT 2015),223-237360

[11] 赵伟楠. 对可满足性（SAT）问题求全解的算法研究及实现. 硕士学位论文. 北京交通大学