

**课程设计报告**

**题目： 基于SAT的数独游戏求解程序**

**课程名称： 程序设计综合课程设计**

**专业班级：**

**学 号：**

**姓 名：**

**指导教师：**

**报告日期： 2019-4-3**

**计算机科学与技术学院**

**任务书**

* **设计内容**

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

* **设计要求**

要求具有如下功能：

1. **输入输出功能：**包括程序执行参数的输入，SAT算例cnf文件的读取，执行结果的输出与文件保存等。(15%)
2. **公式解析与验证：**读取cnf算例文件，解析文件，基于一定的物理结构，建立公式的内部表示；并实现对解析正确性的验证功能，即遍历内部结构逐行输出与显示每个子句，与输入算例对比可人工判断解析功能的正确性。数据结构的设计可参考文献[1-3]。(15%)
3. **DPLL过程：**基于DPLL算法框架，实现SAT算例的求解。(35%)
4. **时间性能的测量：**基于相应的时间处理函数（参考time.h），记录DPLL过程执行时间（以毫秒为单位），并作为输出信息的一部分。(5%)
5. **程序优化：**对基本DPLL的实现进行存储结构、分支变元选取策略[1-3]等某一方面进行优化设计与实现，提供较明确的性能优化率结果。优化率的计算公式为：[(t-to)/t]\*100%,其中t 为未对DPLL优化时求解基准算例的执行时间，to则为优化DPLL实现时求解同一算例的执行时间。(15%)
6. **SAT应用：**将数独游戏[5]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]。(15%)

**1．引言**

**1.1．课题背景与意义**

命题公式的可满足性属于NP问题，它一直是计算机科学的中心问题之一，很多学者在这一领域做了大量的研究。命题逻辑公式的可满足性问题(SAT)是数理逻辑、计算机科学、集成电路设计与验证和人工智能等领域中的核心问题,并且是第一个被证明出来的NP问题。 SAT问题在计算复杂性理论中具有非常重要的地位,设计并实现解决该类问题的高效算法意义重大。它对数学上的定理机器证明（ATP）也有着重要的作用。

**1.2.国内外研究现状**

SAT问题作为DPLL算法的基本应用之一，在国内外均有广泛的研究与讨论。

**1.3.课程设计的主要研究工作**

本次课程设计的主要工作是：（1）对DPLL算法进行实现，使其能够读取cnf文件中的内容，并对变量的真值进行推断，然后对其进行优化，使运行时间尽可能短。（2）运用此算法，写一个数独游戏，使其可以玩，可以自动解题，判断正误等。

**2．系统需求分析与总体设计**

**2.1．系统功能要求**

本设计要求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，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。(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%)

功能（1）至（5）为基础功能，占功能分值的85%。

1. **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]。（15%）

。

**2.2．系统总体设计**

计划设计一个集读取，分析，存储，试错，回溯，存储为一体的程序。

**3．系统详细设计**

**3.1 有关数据结构的定义**

1. 节点

typedef struct Node {

int data;

struct Node \*nextNode;

}Node;

节点是链表的组成部分，用于存放每一个数字。

1. 链表

typedef struct List {

struct List \*nextList;

Node \*head;

}List;

链表用于将各个节点串联起来，然后在各个链表之间建立连接。

**3.2 主要算法设计**

1. 加载文件模块的算法

打开cnf文件后，先判断行首第一个字符：

若是‘c’，说明此行为注释，跳过；

若是‘p’，说明此行是文件整体信息，则将之后的两个数字分别赋值给varnum（变量个数）和clsnum（子句数量）；

若是其他情况，则说明已进入正文，需要开始进行算法的试错与回溯部分。

1. DPLL判断模块

DPLL算法流程如下：

1. 先给出现次数最多的变量赋值（要求使所在子句值为真）
2. 将第一个子句的变量存在第一个链表里，然后对下一个子句重复
3. 若出现错误，则进行回溯，然后寻找下一个出现次数最多的变量进行赋值。
4. 若出现单子句，则直接赋值。

**4．系统实现与测试**

**4.1．系统实现**

1）loadfile

负责加载cnf文件并储存信息的模块

2）copysubl模块与copylist模块

复制链表

1. outputresult

输出结果

1. addvar

将一个变量加在链表的top位置，以便进行下一次赋值与回溯

1. findmaxvar

寻找出现次数（包括正负）最多的变量，这样进行赋值比较有效率。

1. removevar

移除单子句（包括原本的单子句和经过计算的单子句）

1. freelist

释放空间

1. dpll

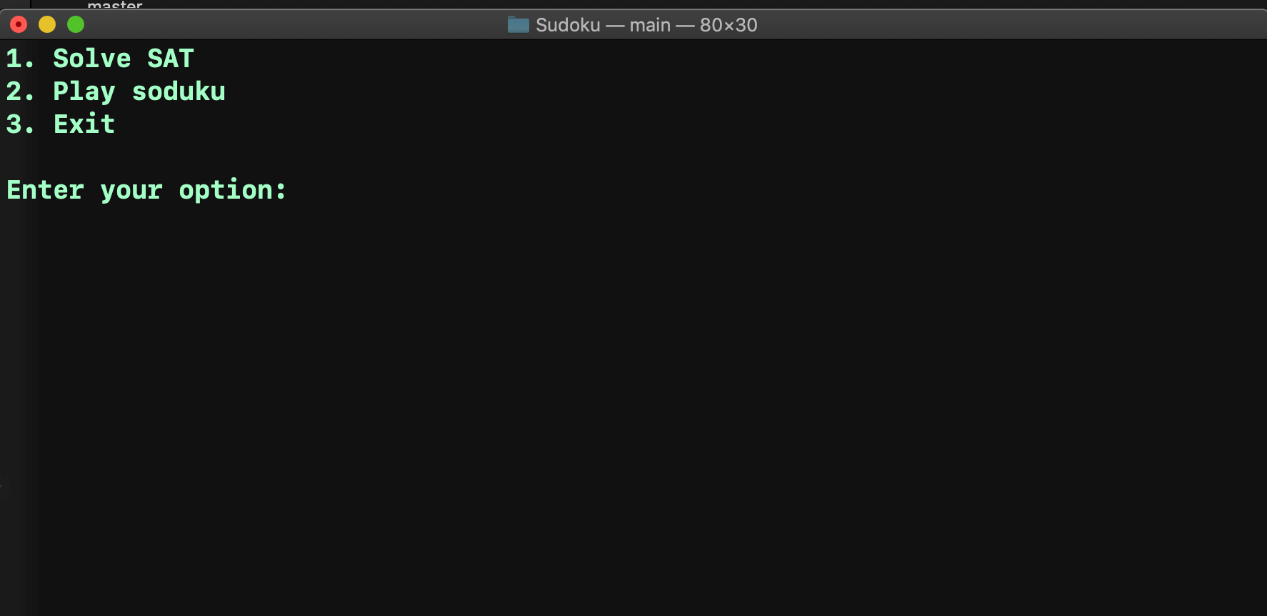
dpll算法

1. save2file

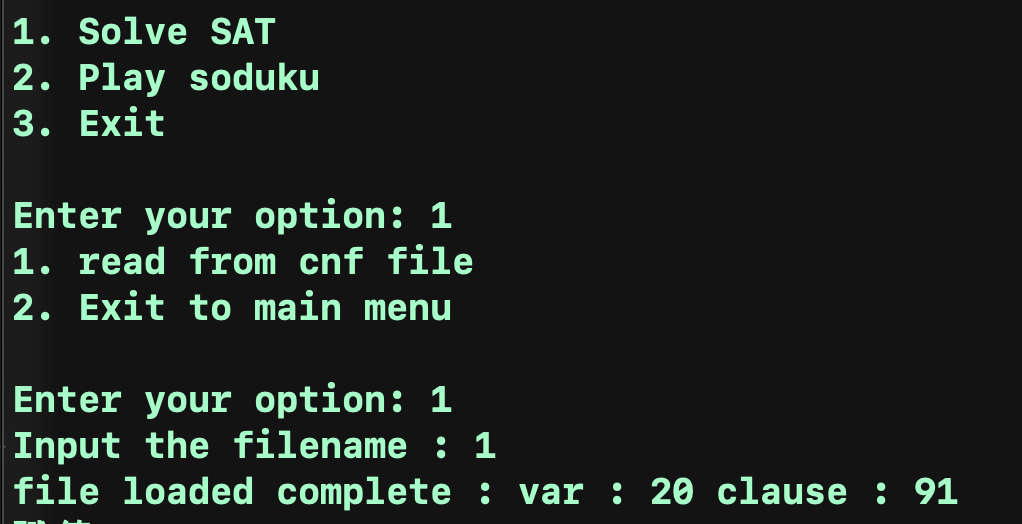
将结果输出至文件

**4.2．系统测试**

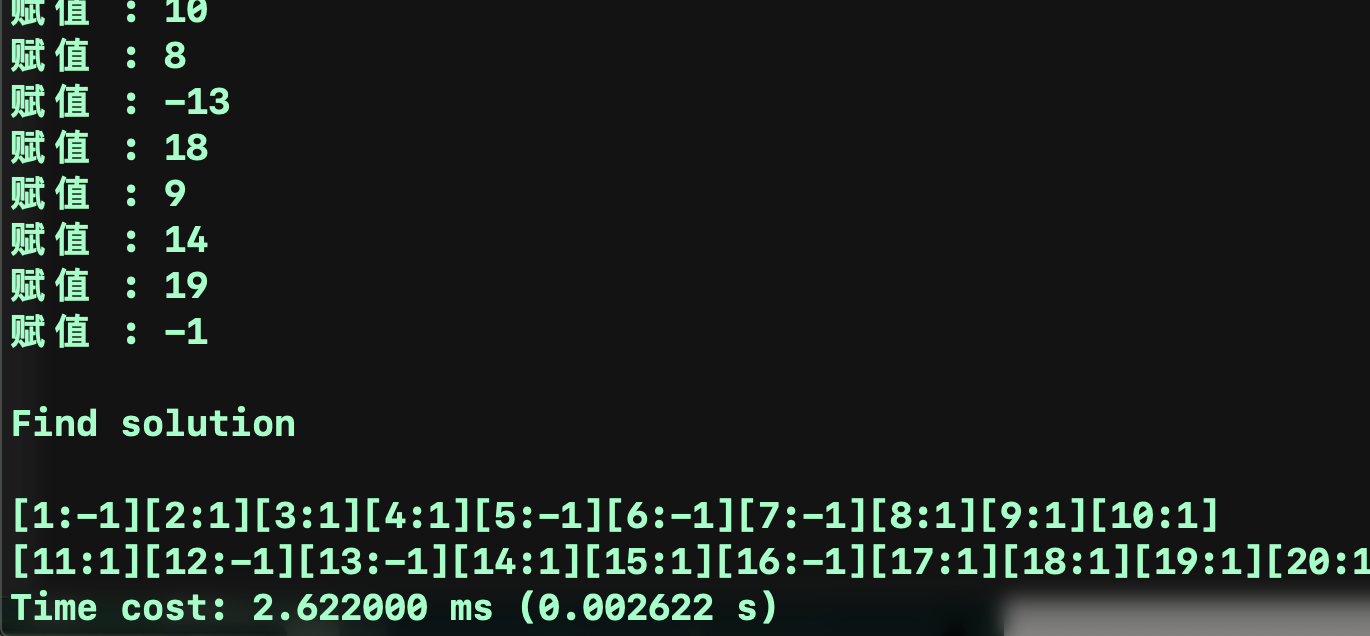
运行程序，出现选择界面：



选择1



可以看到，输入文件编号1以后，程序开始读取1号cnf文件。



读取之后，自动开始解析。并在最后一行输出最终结果。如果不存在解，则会显示错误。

本程序还有输出详细轨迹信息功能。以当前的cnf文件为例，中途生成的trace如下：

赋值 : -15

赋值 : -5

赋值 : 12

赋值 : 10

赋值 : 8

赋值 : -9

赋值 : -16

赋值 : -17

赋值 : 19

赋值 : 9

赋值 : 17

赋值 : -6

赋值 : 4

赋值 : -13

赋值 : -10

赋值 : -13

赋值 : -4

赋值 : 18

赋值 : 19

赋值 : 1

赋值 : -17

赋值 : -9

赋值 : -12

赋值 : -7

赋值 : 17

赋值 : 9

赋值 : 13

赋值 : 10

赋值 : -4

赋值 : -8

赋值 : -11

赋值 : -3

赋值 : 20

赋值 : -16

赋值 : 18

赋值 : -17

赋值 : 19

赋值 : 18

赋值 : 11

赋值 : -6

赋值 : 10

赋值 : 8

赋值 : 4

赋值 : -13

赋值 : 14

赋值 : -9

赋值 : 7

赋值 : -14

赋值 : -17

赋值 : 19

赋值 : 18

赋值 : 1

赋值 : 11

赋值 : 5

赋值 : 12

赋值 : 9

赋值 : 17

赋值 : 13

赋值 : 10

赋值 : -14

赋值 : -4

赋值 : -8

赋值 : -9

赋值 : -16

赋值 : 20

赋值 : -17

赋值 : 1

赋值 : 15

赋值 : -5

赋值 : -12

赋值 : -7

赋值 : 17

赋值 : 4

赋值 : 11

赋值 : 2

赋值 : 20

赋值 : -16

赋值 : 3

赋值 : -6

赋值 : 10

赋值 : 8

赋值 : -13

赋值 : 18

赋值 : 9

赋值 : 14

赋值 : 19

赋值 : -1

Find solution

[1:-1][2:1][3:1][4:1][5:-1][6:-1][7:-1][8:1][9:1][10:1]

[11:1][12:-1][13:-1][14:1][15:1][16:-1][17:1][18:1][19:1][20:1]

Time cost: 2.622000 ms (0.002622 s)

**5．总结与展望**

**5.1．全文总结**

对自己的工作做个总结，主要工作如下：

（1）详细地了解了DPLL算法的精髓。

（2）学会利用回溯法自动解决问题。

（3）这次的程序比较枯燥繁琐，写完这个程序，我的耐心也得到了培养。

**5.1．工作展望**

DPLL算法是国际各大科学机构都在研究的问题，需要较强的数学功底。这次的任务并不算难，但在做的过程中，我学会了从多方获取资料，与同学们探讨，研究，收获颇丰！未来，我也会努力投身于计算机科学研究中！

* **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

**附录**

//

// DPLL.h

// Sudoku

//

// Created by Kaister Wang on 2019/3/13.

// Copyright © 2019 com.KaisterWang. All rights reserved.

//

// CNF 文件用标记行“p cnf A B”来区分注释行和子句行，

// 即这行之后的每一行都表示一个布尔子句，所有的子句用 AND 连接，

// 组成一个完整的布尔表达式。这里的 A 表示变量的个数，B 表示子句的个数。

// clsNum clause number

// varNum varible number

#ifndef DPLL\_h

#define DPLL\_h

#include <stdio.h>

#include <stdlib.h>

#define PATH "./data/n.cnf"

#define RES\_PATH "./data/n.res"

//some data structers

typedef struct Node {

int data;

struct Node \*nextNode;

}Node;

typedef struct List {

struct List \*nextList;

Node \*head;

}List;

//functions

List \*loadFile(int fileNum, int \*varNum, int \*clsNum){

FILE \*fp;

int start\_read = 0, tmp = 0, over = 0;

char path[] = PATH;

List \*L = (List\*)malloc(sizeof(List));//

List \*lp = L;

List \*lp\_pre = NULL;

Node \*np = NULL;

path[7] = fileNum + 48;

fp = fopen(path, "r");

if (!fp) {

printf("cannot open file!\n");

exit(-1);

}

int i = 1;

while (true) {

i++;

if (over) {

break;

}

if (!start\_read) {

char ch;//the char read next

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

if (ch == 'p') {//ready to read cnf

start\_read = 1;

for (int i = 0; i < 4; i++) {

//read & skip "p cnf"

fgetc(fp);

}

fscanf(fp, "%d %d", varNum, clsNum);

//get the amount of var &clause

}else if(ch == 'c'){//which means comment lines

while (true) {

char c = fgetc(fp);

if (c == '\n') {

break;//next line

}

}

}else printf("This is not a standard cnf file.\n");

}else {//begin reading cnf sentence

int num = 0, status = 1;

lp->head = (Node\*)malloc(sizeof(Node));

np = lp->head;

while (true) {

status = fscanf(fp, "%d", &tmp);

//read a number & store it in tmp

if (status != 1) {

//which means no number left

over = 1;

break;

}

if (num == 0) {

np->data = tmp;

//store the number in node

np->nextNode = NULL;

num++;

continue;

}

if (tmp != 0) {

np->nextNode = (Node\*)malloc(sizeof(Node));

np->nextNode->data = tmp;

np->nextNode->nextNode = NULL;

np = np->nextNode;

num++;

} else {

//tmp == 0, another line

//lp's length = num

lp->nextList = (List\*)malloc(sizeof(List));

lp\_pre= lp;//shang yi hang

lp = lp->nextList;

lp->nextList = NULL;

break;

}

}

}

}

lp\_pre->nextList = NULL;//free the last line

fclose(fp);

printf("file loaded complete : var : %d clause : %d\n", \*varNum, \*clsNum);

return L;

}

/\*

void printList(List \*L){

printf("==========\n");

List \*lp = L;

int i = 0;

Node \*np = NULL;

while(lp){

np = lp->head;

while (np) {

printf("%3d", np->data);

np = np->nextNode;//go on

}

printf("\n");

lp = lp->nextList;

i++;

}

}

\*/

List\* copySubL(List \*L){//1 dimension

//return a copy of L

List \*L\_copy = (List\*)malloc(sizeof(List));

L\_copy->nextList = NULL;

if (L->head == NULL) {

//L is empty

L\_copy->head = NULL;

return L\_copy;

}

L\_copy->head = (Node\*)malloc(sizeof(Node));

Node \*np = L->head;

Node \*np\_copy = L\_copy->head;

while (np) {

np\_copy->data = np->data;

if (np->nextNode == NULL) {

np\_copy->nextNode = NULL;

break;

}

np\_copy->nextNode = (Node\*)malloc(sizeof(Node));

np\_copy = np\_copy->nextNode;

np = np->nextNode;

}

return L\_copy;

}

List\* copyL(List \*L){//copy list head

List\* L\_copy = copySubL(L);

List \*lp = L->nextList;

List \*lp\_copy = L\_copy;

while (lp) {

//on every sublist's head

lp\_copy->nextList = copySubL(lp);

//copy every sublist

if (lp->nextList == NULL) {

break;

}

lp\_copy = lp\_copy->nextList;

lp = lp->nextList;

}

return L\_copy;

}

void outputResult(int ans[], int varNum){

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

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

if (ans[i] == 0) {

printf("?]");

}

else printf("%d]", ans[i]);

if ((i + 1) % 10 == 0) {

printf("\n");//shi ci huan yi xia hang

}

}

}

void addVar(int var, List \*\*L){

//add a var on the top of the list

List \*lp = (List\*)malloc(sizeof(List));

lp->head = (Node\*)malloc(sizeof(Node));

lp->head->data = var;

lp->head->nextNode = NULL;

lp->nextList = \*L;

\*L = lp;

}

int findMaxVar(List \*L, int varNum){

//find next most frquently variable

int tmp = 0;

int maxfrequency = 0;

int posnum = 0;

int negnum = 0;

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

posnum = 0;

negnum = 0;

List \*lp = L;

Node \*np = NULL;

while (lp) {

np = lp->head;

while (np) {

if (np->data == (i+1)) {

posnum++;

}

else if (np->data == -(i+1)){

negnum++;

}

np = np->nextNode;

}

lp = lp->nextList;

}

if (posnum + negnum > maxfrequency) {

maxfrequency = posnum + negnum;

if (posnum > negnum) {

tmp = i + 1;

}

else tmp = - i - 1;

}

}

return tmp;

}

// return -1 : 0

//

//

int removeVar(List \*\*L, int var){

//remove clause with a var occured in a single cluse

List \*lp = \*L;

List \*lp\_pre = \*L;

while (lp) {//cycle of line

if (lp->head == NULL) {

return -1;

}

int remove\_clause = 0;

Node \*np = lp->head;

Node \*np\_pre = lp->head;

while (np) {

if (np->data == -var) {

//remove -v

if (np == lp->head) {

//which means np is the first node

lp->head = lp->head->nextNode;

free(np);

np = lp->head;

if (np == NULL) {

return -1;

//clause = 0, no zhenzhi

}

} else {

np\_pre->nextNode = np->nextNode;

free(np);

np = np\_pre->nextNode;

}

continue;

}else if (np->data == var){

Node \*head = lp->head;

if (lp == \*L) {

//shi di yi ge zi ju

\*L = (\*L)->nextList;

free(lp);

lp = (\*L);

if(\*L == NULL){

return 1;//dedaozhenzhi

}

}else{

lp\_pre->nextList = lp->nextList;

free(lp);

lp = lp\_pre->nextList;

}

Node \*bak = NULL;

while (head) {

//free every node

bak = head;

head = head->nextNode;

free(bak);

}

remove\_clause = 1;

break;

}

np\_pre = np;

np = np->nextNode;

}//end of while(np)

if (remove\_clause == 0) {

//if didn't remove clause go on

lp\_pre = lp;

lp = lp->nextList;

}

}//end of while(lp)

return 0;

}

void freeList(List \*L){

List \*lp = L;

List \*lp\_pre = NULL;

Node \*np = NULL;

while (lp) {

Node \*bak = NULL;

np = lp->head;

while (np) {

bak = np;

np = np->nextNode;

free(bak);

}

lp\_pre = lp;

lp = lp->nextList;

}

}

int dpll(List \*L, int varNum, int ans[]){

List \* new\_L = copyL(L);

//printf("Test\n");

//printList(new\_L);

//find single clause

int i = 0;

while (true) {

List \*lp = new\_L;

Node \*np = NULL;

int have\_signle = 0;

while (lp) {

np = lp->head;

while (np) {

i++;

if (lp->head && !lp->head->nextNode) {

//lp->head exists, ->next->head dont exists

//means is single clause

printf("赋值 : %d\n", np->data);

int value = (np->data) > 0 ? 1 : -1;

ans[abs(np->data) - 1] = value;

have\_signle = 1;

//

//printf("%d\n", i);

int ret = removeVar(&new\_L, np->data);

//error occurred here

//soloved

//printf("Test\n");

if (ret != 0) {

//dedaozhenzhi huo meiyouzhenzhi

return ret;

}

break;

}else if(lp->head == NULL){

freeList(new\_L);

return -1;

}

np = np->nextNode;

}//end of cycle row

if (have\_signle) {

//bu yong tiao

break;

}

lp = lp->nextList;

}//end of cycle line

if (have\_signle == 0) {

break;

}

}

if (new\_L == NULL) {

return 1;

}

int nextVar = findMaxVar(new\_L, varNum);

ans[abs(nextVar) - 1] = nextVar > 0 ? 1 : -1;

addVar(nextVar, &new\_L);

if(dpll(new\_L, varNum, ans) == 1) {

//if empty means qiu jie wan bi

freeList(new\_L);

return 1;

} else {

List \*bak = new\_L;

new\_L = new\_L->nextList;

free(bak->head);

free(bak);//delete di yi hang

addVar(-nextVar, &new\_L);

return dpll(new\_L, varNum, ans);

}

}

void save2file(int ans[], int varNum, int ret, double cost, int filename) {

char path[] = RES\_PATH;

path[7] = filename+ 48;

FILE \*fp = fopen(path, "wt");

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

if (ret == 1) {

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

if (ans[i] == 1)

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

else

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

}

fprintf(fp, "\n");

}

fprintf(fp, "%lf", cost \* 1000);

}

#endif /\* DPLL\_h \*/