离散数学课程实验说明文档

——命题逻辑推理

作 者 姓 名： 马威

学 号： 2151294

指 导 教 师： 唐剑锋

学院、 专业： 软件学院 软件工程

****

目 录

[1 项目分析 1](#_Toc495668153)

[1.1 项目背景 1](#_Toc495668154)

[1.2 项目要求 1](#_Toc495668155)

[2 项目设计 3](#_Toc495668156)

[2.1 数据结构设计 3](#_Toc495668157)

2.2 算法设计 3

2.2.1 算法思路 3

2.2.2 性能评估 3

2.2.3 流程图表示 4

2.2.4 代码实现 4

[3 项目测试 6](#_Toc495668161)

4 心得体会 6

**1.项目分析**

1.1 项目背景

日常生活中有许多需要进行命题逻辑推理的地方，例如在破案过程中的推理。这种场合一般来说会得到许多信息，这些信息有真有假，却又环环相扣，需要缜密的推理才能辨别真假，寻得真相。

1.2 项目要求

根据下面命题，用命题逻辑推理方法确定谁是作案者，并给出推理过程，C语言源代码及演示界面。

（1）营业员A或B偷了手表；

（2）若A作案，则作案不在营业时间；

（3）若B提供的证据正确，则货柜未上锁；

（4）若B提供的证据不正确，则作案发生在营业时间；

（5）货柜上了锁。

**2.项目设计**

2.1 数据结构设计

由项目分析可以得出，该项目需要完成一次推理的模拟。在推理过程中，可以使用数组存储多个命题的真假情况。由于一般需要推理时使用到的命题数量较少，一般的数组可以做到，故本题不再额外使用新的数据结构

2.2 算法设计

**2.2.1 算法思路**

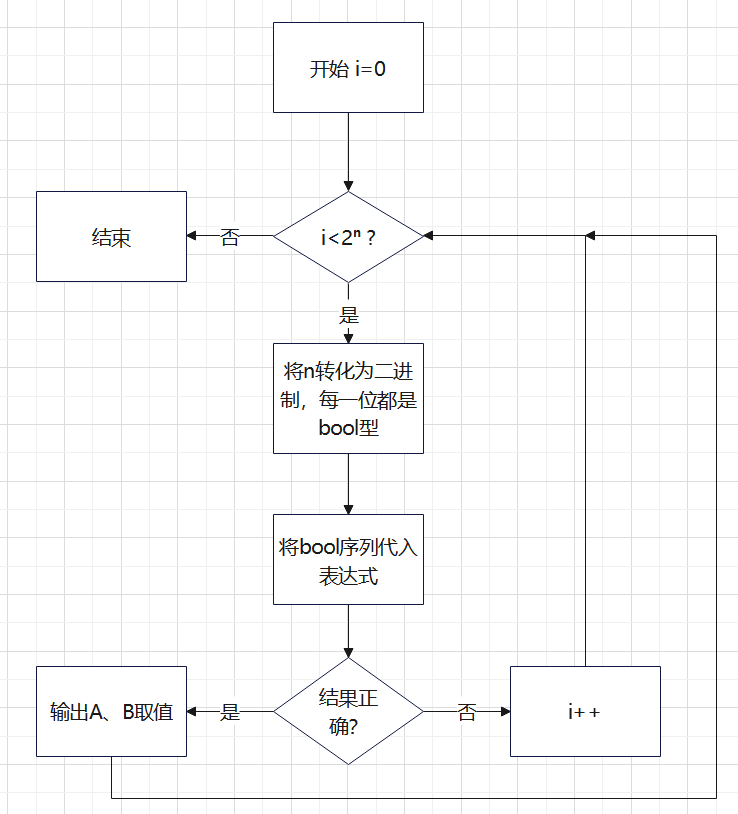
推理对于人来说可以使用复杂但快捷的推理方法，可对于计算机来说无法进行如此灵活的运算，而最合适的方法就是考虑所有的情况，再一一比对哪一种情况是对的。

拿本题中的5个命题来说，共包括了5个原子命题。抛开其内在联系不说，组合共2**5**=32种情况，又判别组合是否符合情况的表达式为(A || B) && (A -> !C) && (D -> !E) && (!D -> C) && E，故可以把每一种情况转换为一个二进制序列，代表真假的一种组合（例如第10种转换为01010，代表5个原子命题情况为“假真假真假”），然后代入到表达式判断。从0一直循环到2**n**，若结果为真，则得出一种推理结果，将其输出。

**2.2.2 性能评估**

设各个命题中总共包含n个原子命题，经过2**n**次迭代，每次循环又要产生n个布尔值，需n次操作，故时间复杂度为O(n\*2**n**)。

**2.2.3 流程图表示**

****

**2.2.4 代码实现**

#include<iostream>

#include<cmath>

using namespace std;

#define VAR\_NUM 5

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

  函数名称：trans

  功    能：将十进制数转换为二进制数，位数和命题数量对齐

  输入参数：const int number：待转换的二进制数（只读）

            bool vars[]：存储转换结果的布尔数组

  返 回 值：无

  说    明：二进制每一位是布尔类型，表示每个命题的真假

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

void trans(const int number, bool vars[])

{

    int n = number;

    int count = 0;

    do {  /\*由于除2取余法需要反转才能得到结果，故在此处直接反着存到vars数组内\*/

        const int rem = n % 2;

        vars[VAR\_NUM - count - 1] = bool(rem);

        n = (n - rem) / 2;

        count++;

    } while (n);

    for (int i = 0; i < VAR\_NUM - count; i++)  /\*多出来的空位用0（false）填补\*/

        vars[i] = false;

}

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

  函数名称：main

  功    能：主函数

  输入参数：无

  返 回 值：0

  说    明：无

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

int main()

{

    bool vars[VAR\_NUM];

    for (int i = 0; i < pow(2, VAR\_NUM); i++) {

        trans(i, vars);  /\*将十进制数转换为二进制数，位数和命题数量对齐\*/

        if (  /\*若情况符合，输出结果\*/

            (vars[0] || vars[1])

            && (!vars[0] || !vars[2])

            && (!vars[3] || !vars[4])

            && (vars[3] || vars[2])

            && vars[4]

            ) {

            cout << "A=" << vars[0] << ' ' << "B=" << vars[1] << endl;

        }

    }

    //A="营业员A偷了手表"

    //B="营业员B偷了手表"

    //C="作案在营业时间"

    //D="B提供的证据正确"

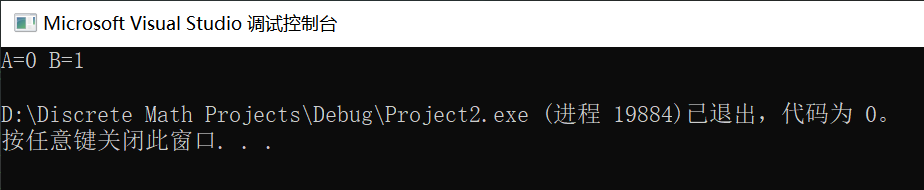
    //E="柜子上锁"

    //表达式：(A || B) && (A -> !C) && (D -> !E) && (!D -> C) && E

    return 0;

}

**3.项目测试**



由运行结果可知，是营业员B偷了手表，A没偷，且只有这一种情况。

**4.心得体会**

本项目中把数转换成二进制数（每一位是布尔型）这样的实现方法，其启发是样例代码中的多重循环。由于多重循环从代码上看形势较为复杂，编写时较容易出错，就换了一种思路实现。

然而，不曾想这样反而增加了时间复杂度，多重循环只有O(2**n**)，而二进制转换达到了O(n\*2**n**)，显然在原子命题较多的情况下这样的复杂度是难以忍受的。再考虑到一般的命题逻辑推理的n并不大，编写多重循环还是可以做到的，因此，多重循环还是可以考虑的。