数据结构与算法 project 1

1. 解题思路

思路如下：

1. 表达式联立方法：

对于参与求解过程的若干个表达式，应当设计一个方法能够统一管理它们。

1. 这若干个表达式全部都是析取式：即对于单独的一个表达式，其中任意一个boolean变量为真，这个表达式就为真。我采用结构体来实现单个析取式的表达，每一个结构体都有三个数据v1，v2，v3，它们的绝对值是对应析取式里面涉及到的变量的编号，正负表示是否为其非值。
2. 将这若干个表达式全部放进一个合取范式当中。这个合区范式为真，当且仅当所有的析取表达式为真。

这样一来，问题就转化为，如何对这m个boolean变量赋值，使得第二步中的合取范式为真。

1. 简化思路：

上述的合取范式仍然需要进一步简化，应当设计一个方法能够简化这个构造出来的合取范式。

1. 对于一个单独的boolean变量，对赋值为真，那么含有的析取表达式就都为真，在这种情况下，这些值为真的析取表达式不会阻碍合取式为真。故而可以将这些析取式从合取式当中删去。
2. 在i中，若对赋值为真则的值为假，在这种情况下，含有的析取表达式可以将从中删去，因为即使为假，它也并不会阻碍这些含有的析取表达式为真。
3. 对p赋值为假的情况同理。
4. 为了保证后续回溯的需要，同时保证运行的速度，我在simplifyCNF函数当中加入了记录操作的相关数据结构：我使用了isDeleted数组记录单个析取式是否被删除，使用了undoChanges栈来记录在每次赋值当中，有变量变成0的析取式本来的样子，使用了countHistory来记录每轮赋值当中变量删去了几次。
5. 判断策略：

综合来看，也就是说在求解的过程中：

1. 如果合取范式中的某一个析取式值为真，那么就从合取范式当中删掉这个析取式。
2. 如果某个单独的boolean变量值为假，那么就从含有这个变量的析取式当中删掉这个变量

最终如果发现合取范式被删为空，那么就说明赋值成功。如果发现存在某个析取式，其中的boolean变量被全部删掉，但这个式子本身并没有从合取范式中被删掉，那么就说明赋值失败，此时应当实行回溯。

1. 回溯策略
2. 按照3）中判断方法，设置一个功能为判断赋值是否成功的函数isCNFSatisfied。当这个函数返回真时，就说明找到了一组解，可以输出了。
3. 如果isCNFSatisfied返回值为假，那么说明这时候还有变量需要被赋值，合取范式当中的析取式没有被全部删除。
4. 选定一个没有被赋值的变量。在代码中我直接选择了第一个析取式的第一个未赋值变量。然后使用simplifyCNF函数给它赋值为真，同时实现简化合取范式的目标。另外此时在数组assignment里面记录下这个变量的取值。
5. 递归调用solveSAT函数，让代码对简化后的合取范式进行进一步的赋值简化操作。
6. 按照3）中的判断方法，对于当前尝试赋值为真的变量，如果发现赋值失败，那么就取countHistory栈的栈顶元素，弹出undoChanges栈中对应数量的元素，这样就可以撤销在刚刚进行的失败的赋值中对析取式的操作。之后改变该变量的赋值，改变数组assignment里面该变量的赋值的记录。调用simplifyCNF函数令其赋值为假，并简化合取范式，再次递归调用solveSAT函数。如果对于这个变量无论是赋值为真还是假，都失败了，那么就回溯到上一级solveSAT函数的调用，对上一级被赋值的变量进行重新赋值。
7. main函数的设计
8. 利用for循环，按照格式要求实现了数据的输入与输出。输出时按照顺序输出数组assignment。
9. 实现难点
10. 定义结构体的时候要注意不能忘记写没有参数的Disjunction函数，否则可能会出现报错，不能编译。
11. 在用于简化合取范式的simplifyCNF中，易于出错的是“i++”的时机。即，在遍历合取范式当中各个析取式，看是否有需要删去整个析取式或者删去其中某个变量的时候，应当仅在未删去整个析取式时进行“i++”操作。否则会出现某个析取式被跳过的情况，而导致求解的错误。
12. 用于简化合取范式的simplifyCNF函数和用于回溯求解的solveSAT函数当中，传入析取式个数NumDisjunction时，应当传引用。不然的话在外部调用这两个函数时，析取式的个数不能正确更新，会导致无限循环。
13. 函数solveSAT当中，需要格外小心递归调用放置的位置，以及递归调用的逻辑是否正确，不然同样也会陷入无限循环。
14. 使用的数据结构
15. 结构体：Disjunction用于表达单个的析取式，并作为数组的元素。Undo用来表示对析取式当中变量的改变。
16. 数组：在实现当中我使用到了两个数组。第一个是Disjunction \*cnf，用来存储析取式，这个数组起到了合取范式的作用，cnf[i]存储的是第i+1个析取式；第二个是int \*assignment，用来存储每个变量的具体真值，assignment[i]存储的是第i个boolean变量的取值（0或1），便于输出结果。
17. 栈：设置了countHistory栈和undoChanges栈。
18. （隐含）栈：递归调用函数solveSAT进行回溯求解时，使用到了栈。