目录

一、Q-M 算法使用方法说明

1. 双击 QM 算法/Debug/QM.exe，打开应用程序，直接弹出黑色窗口（如图 1）；

图 1

2. 输入自变量数目

说明：使用者需要化简几个变量，输入数目即可。（以 4 个变量为例，如图 2）

注意：本实验中要求自变量数目在 1~10 之间，不在该范围内会提示输入错误。（但经过实际验证，本应用程序对 10 个以上的变同样有效，只是时间略长）。

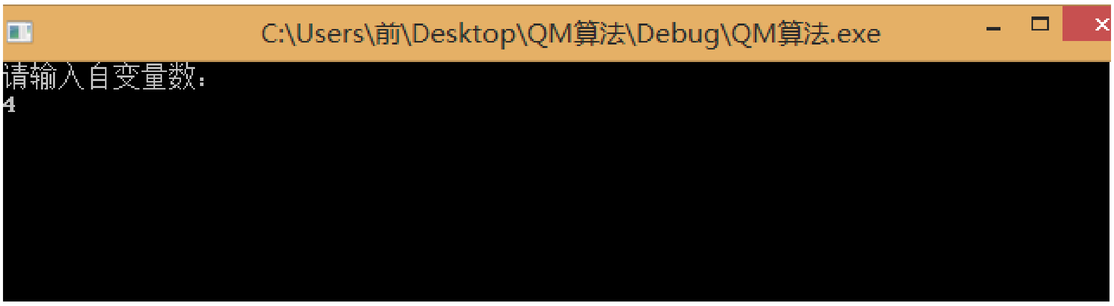


图 2

3. 输入最小项数目

说明：请输入最小项的数目（以 7 个最小项为例，如图 3）。

注意：合法的输入 N 应该满足$1 \le N \le 2^n$（其中 n 为自变量数）。当最小项数目为 0 时，研究的意义不大，因为忽略；最小项最多为2^n，表明所有项都是最小项。非法输入会提示输入错误。

图 3

4. 输入最小项编号列表

说明：请输入最小项的十进制编号（以 4 5 6 8 9 10 13 为例，如图 4）。

注意：合法的输入要求每个输入的数 N 都满足$0 \le N \le 2^n - 1$，因为最小项的编号在这个范围内才是有意义的。非法输入会提示输入错误。

图 4

5. 输入无关项数目

说明： 请输入无关项的数目（以 3 个无关项为例，如图 5）。

注意： 与 3 中输入最小项类似，合法的输入 N 应该满足$1 \le N \le 2^n - 1$（其中 n 为自变量数）。当无关项数目为 0 时，程序直接根据之前的最小项进行运算。非法输入会提示输入错误。

图 5

6. 输入无关项编号列表

说明： ①当 5 中输入的无关项数目为 0 时，跳过这一步，直接进行下一步；

②无关项数目非 0 时，请输入无关项的十进制编号（以 0 7 15 为例，如图 6）。

注意： 合法的输入要求每个输入的数 N 都满足$0 \le N \le 2^n - 1$，因为无关项的编号在这个范围内才是有意义的。非法输入会提示输入错误。

图 6

7.　**输出化简结果**

说明：应用程序会直接根据 Q-M 算法输出化简后的结果（以《现代逻辑设计（第二版）》P82-83 例 3.4 为例，结果与书本上一致，如图 7)。

注意：输出结果采用“1”“0”“-”的形式。1 表示化简结果中出现这个变量因子，0 表示出现这个变量因子取反，- 则表示不出现该变量因子。

比如，若用字母表示各变量因子，“01 - -”表示“ $\overline{A}B$ ”，“10 - 0”表示“ $A\overline{B}\overline{D}$ ”，“1 - 01” 表示 “ $A\overline{C}D$ ”。

图 7

二、设计算法的基本思路

（一）　各项的存储及预处理

1.　输入并存储项的编号

根据提示分别输入自变量的个数 n，最小项的个数 m1，无关项的个数 m2。根据 m1 和 m2 的大小申请动态内存，用两个一维整型数组分别存储最小项和无关项的十进制编号。

2.　建立项类并存储（Class Term）

创建 Term 类的动态数组，将输入的每个最小项、无关项基本信息存入数组之中，（Term 类的私有成员）包括：

*binary	用于存储编号的二进制形式的数组；
*next	指向下一项的指针；
num_of_ones	二进制中含有 1 的个数；
status	1 表示是本原蕴含项，0 表示不是。

（二）合并“二进制中只有 1 位不同”的项

1.　将“二进制表示中 1 的个数相同”的项进行分组

建立链表指针数组*t_head，每个指针 t_head[i]作为一个链表的头指针，该链表由所有“二进制形式中 1 的数目为 i”的项连接（LinkTerm 函数）而得到。

2. Q-M 算法合并“二进制中只有 1 位不同”的项

将所有的最小项和无关项按照二进制形式中所含有 1 的个数链入不同的链表之后，用 Q-M 算法进行合并。

① 对于“二进制表示中有 i 个 1”的项构成的链表，将当前指针 t_current[i]指向头指针 t_head[i]，同时将头指针 t_head[i]指向 NULL，后面进行合并之后形成的新的合并项链入该链表之中。

② 具体合并方法：

用当前指针 t_current [i]遍历链表中每一项的二进制表示，分别与 t_current [i+1]链表内每一项的二进制表示，进行逐位比较。当两项的二进制表示只有一位的数字不同时，则表示可以合并，两项均不是本原蕴含项，置状态变量 status 为“0”。同时，将不同的这一位表示为 -1，在化简的结果中表示“-”；其他相同的位保持数字不变，即可得到合并后的新项。再将合并后的项链入 t_head[i]的末端，等待继续合并。一轮合并完成之后，进行下一轮合并，直至合并全部完成。

[注意]

a. 若 t_current[i]链表内的某一项与 t_current[i+1]内的所有项都不能够合并，则这一项为本原蕴含项，将该项的 status 置为 1。更特殊地，若 t_current[i+1]指向 NULL，则说明 t_current[i]链表内的所有项都为本原蕴含项。

b. 若某一轮合并没有得到新的合并项，则合并操作结束。指针 *prime_implicant 指向所有的本原蕴含项。

（三）寻找最小覆盖

1. 创建最小覆盖的二维矩阵（实现本原蕴含图）

当上一步找到所有的本原蕴含项之后，创建最小覆盖二维矩阵**mini_cover，实际含义是《现代逻辑设计（第二版）》第 88 页的本原蕴含图。然后对最小覆盖矩阵赋值，根据每一个本原蕴含项合并的最小项，令这些最小项编号对应的列的值为 1，其余赋值为 0。对每一个本原蕴含项均进行上述操作，直至所有赋值完成。总结来说，矩阵的每一列表示覆盖的最小项的编号，而每一行表示上一步已经得到的本原蕴含项。

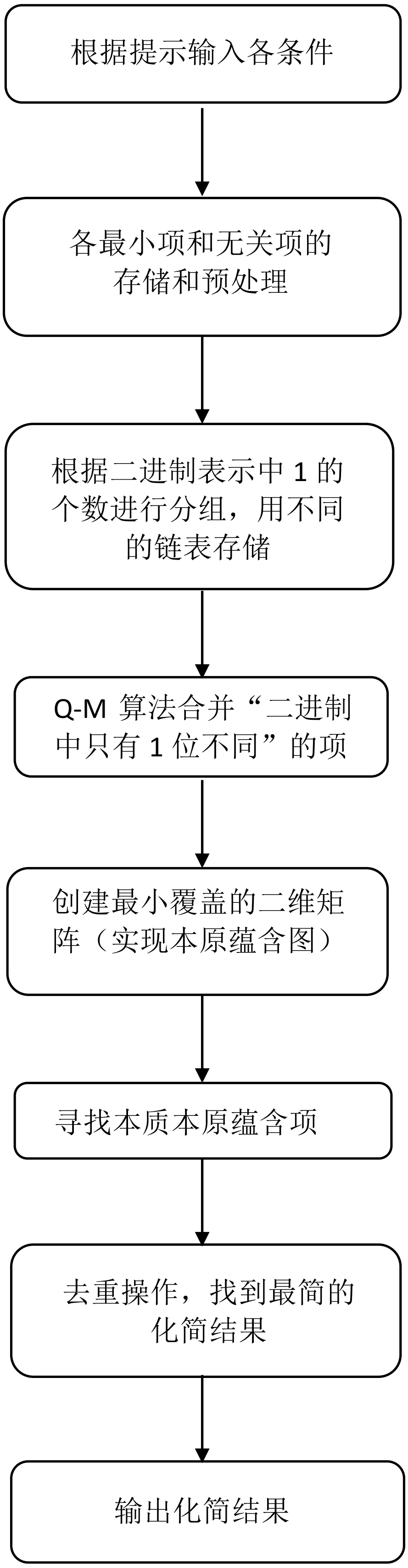
2. 寻找本质本原蕴含项

对最小覆盖矩阵赋值完成之后，按照列的次序遍矩阵。若其中一列的元素只有一个为 1，则标记该列，该列对应的本原蕴含项即为本质本原蕴含项，最终化简的表达式中必然包括这一项，用 mini_cover[i]指针指向这个本质本原蕴含项。同时遍历这一本质本原蕴含项所在的行，逢 1 就对这一列作标记，表示该列对应的最小项已经被最小覆盖所覆盖。之后再进行一次列遍历，若某列已被标记，则跳过；若某列没被标记，则遍历此列所有元素，找到最下面的 1 所对应的本原蕴含项作为最小覆盖包含的项（说明：底下的本原蕴含项由更多的最小项或者无关项合并而成）。

3. 去重操作，找到最简的化简结果

对于找到的每一个最小覆盖包含的本原蕴含项，若除去这项之外的其他项能够对矩阵的每一列都标记，则表示这一项是多余的，将其删去。多余的全部删除之后，mini-cover 指向的本原蕴含项即形成一个最小覆盖。即可输出化简得到的最终结果。

三、算法流程图


根据提示输入各条件
各最小项和无关项的存储和预处理
根据二进制表示中 1 的个数进行分组，用不同的链表存储
Q-M 算法合并“二进制中只有 1 位不同”的项
创建最小覆盖的二维矩阵（实现本原蕴含图）
寻找本质本原蕴含项
去重操作，找到最简的化简结果
输出化简结果

四、运行结果举例：

除去使用方法说明中举的例子之外，再以第 03 讲作业中 3.3(b) 为例：

3.3 (布尔简化)使用卡诺图将下列函数化简成与或形式，充分利用式中给出的无关项。

(b) $f(A,B,C,D)=\sum m(0,1,4,10,11,14)+\sum d(5,15)$

解：

首先在卡诺图中表示出各个最小项

AB\CD	00	01	11	10
00	m_0	m_1	m_3	m_2
01	m_4	m_5	m_7	m_6
11	m_{12}	m_{13}	m_{15}	m_{14}
10	m_8	m_9	m_{11}	m_{10}

则 $f(A,B,C,D)=\sum m(0,1,4,10,11,14)+\sum d(5,15)$ 在图中

AB\CD	00	01	11	10
00	m_0	m_1		
01	m_4	d_5		
11			d_{15}	m_{14}
10			m_{11}	m_{10}

则 所得卡诺图为;

AB\CD	00	01	11	10
00	1	1	0	0
01	1	×	0	0
11	0	0	×	1
10	0	0	1	1

根据卡诺图可化简得出两个本质本原蕴含项，且覆盖了所有 1 单元

所以， $f(A,B,C,D)=AC+\overline{A}\,\overline{C}$ 。

```
C:\Windows\system32\cmd.exe
请输入自变量数：
4
请输入最小项数目：
6
请输入最小项编号列表：
0 1 4 10 11 14
请输入无关项数目：
2
请输入无关项编号列表：
5 15
根据QM算法化简结果：
0-0- 1-1-
请按任意键继续. . .
```

QM 算法程序化简结果为“0 – 0 –”和“1 – 1 –”，用字母表示为 AC 和 $\overline{AC}$，与作业中使用卡诺图化简结果一致。