1. 操作手冊

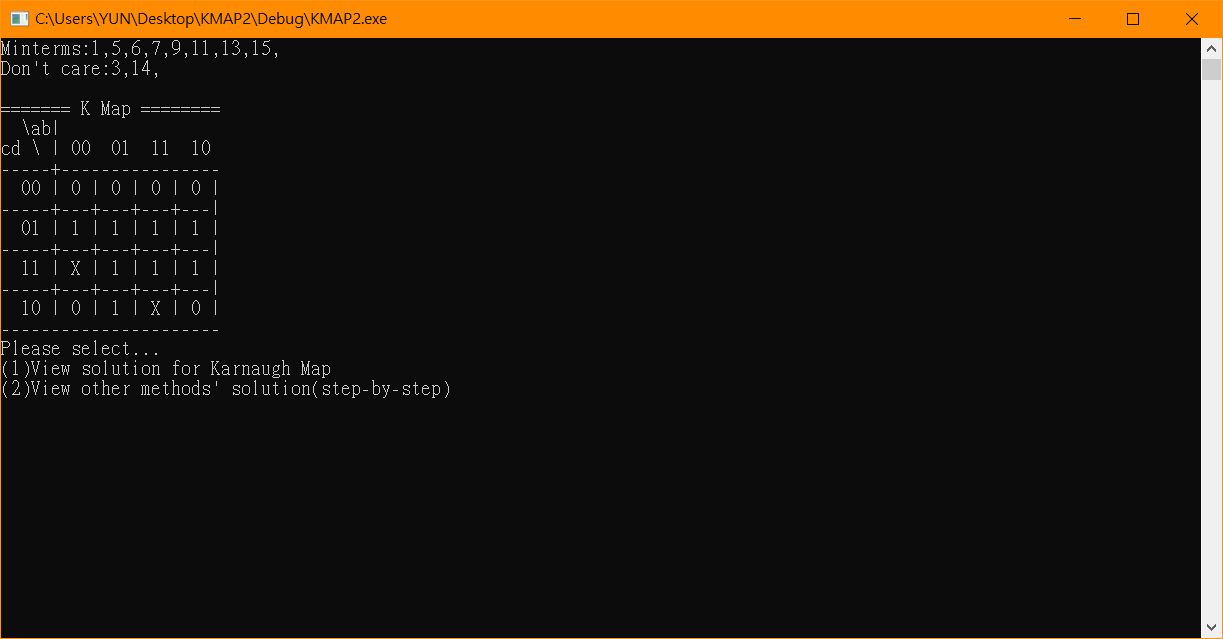
【程式功能說明】

繪製四變數布林代數式之卡諾圖，並找出Minimum SOP所有可能解。

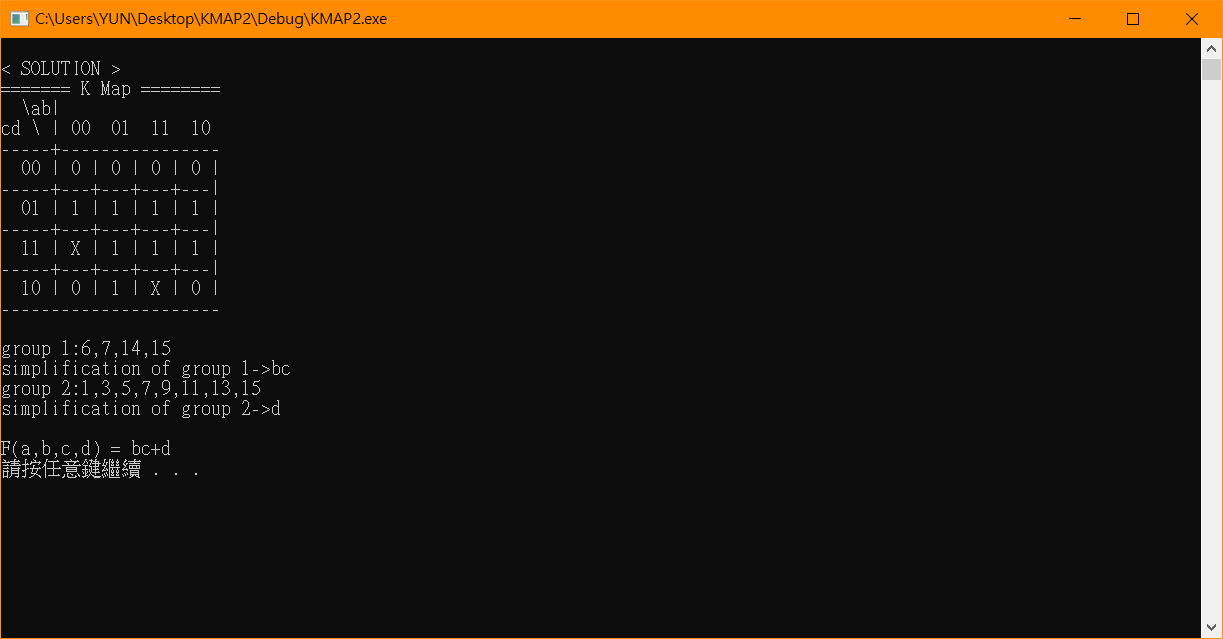
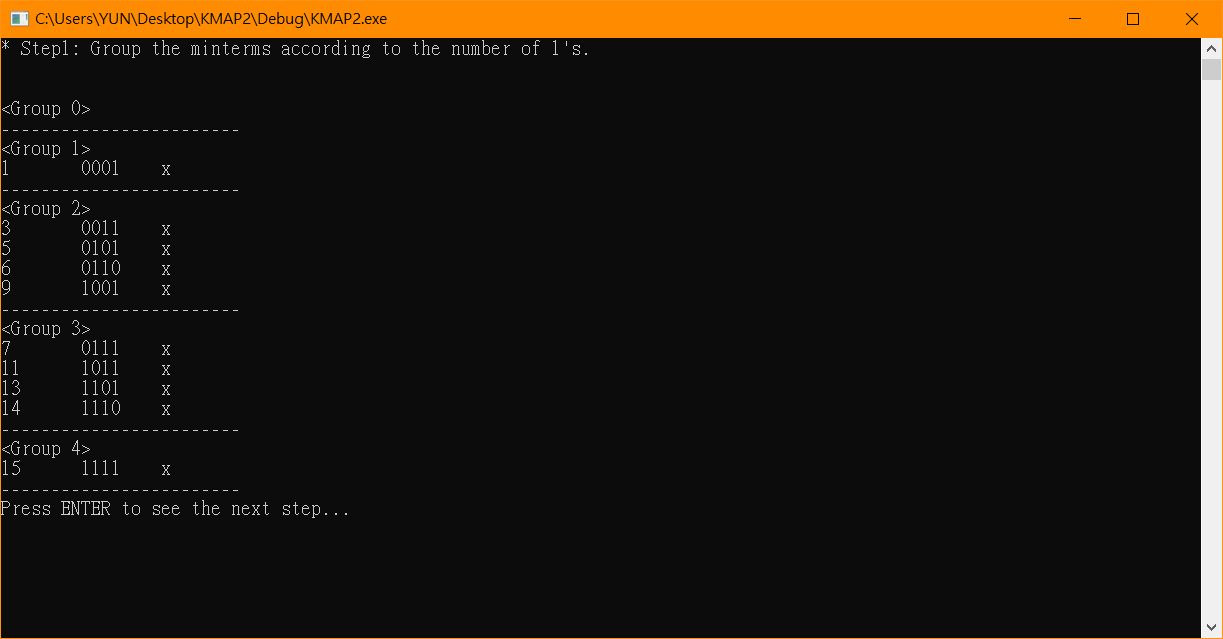
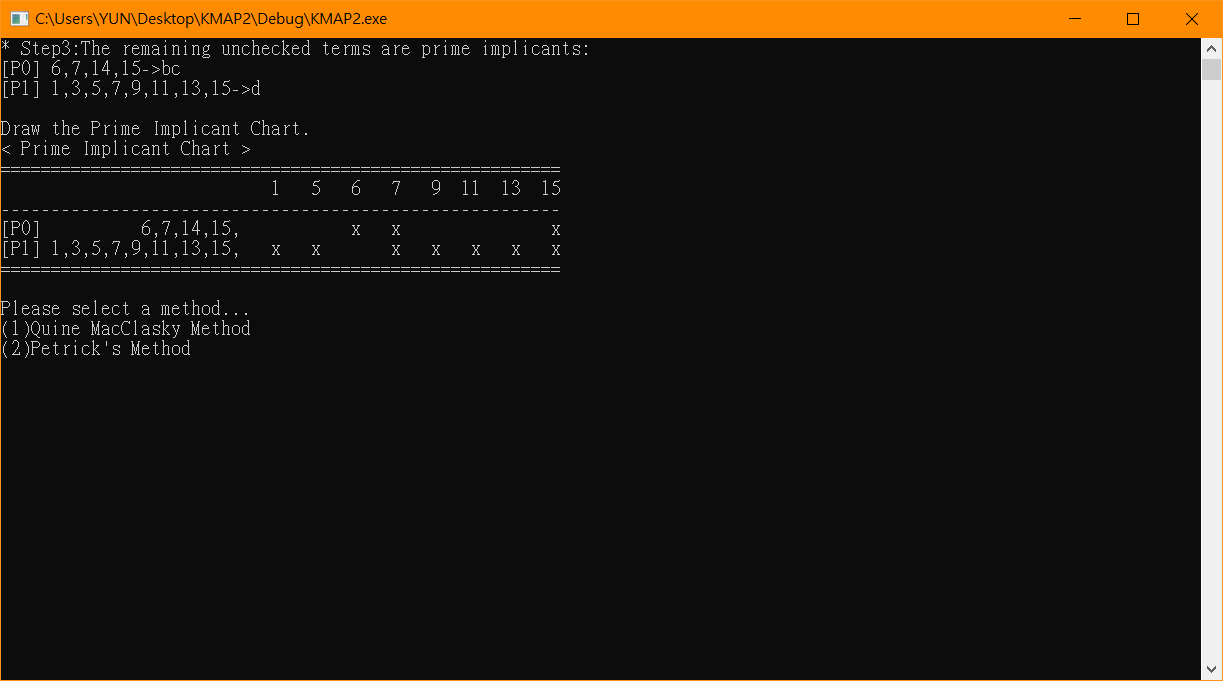
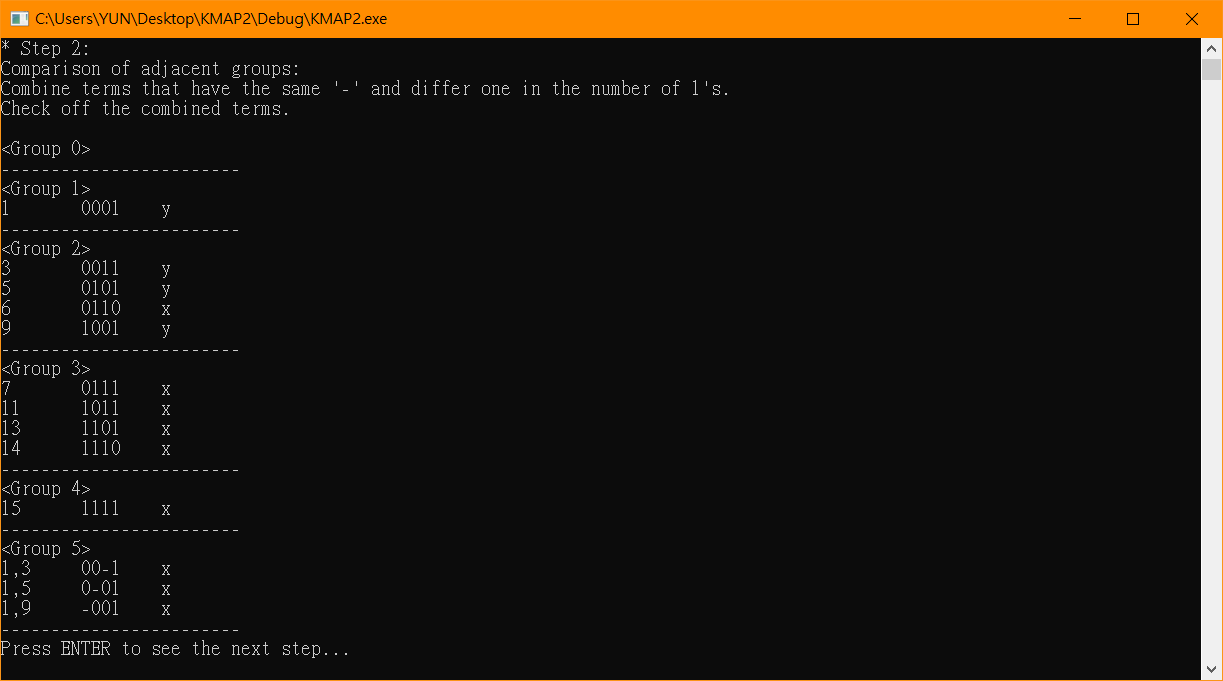
【程式使用流程】

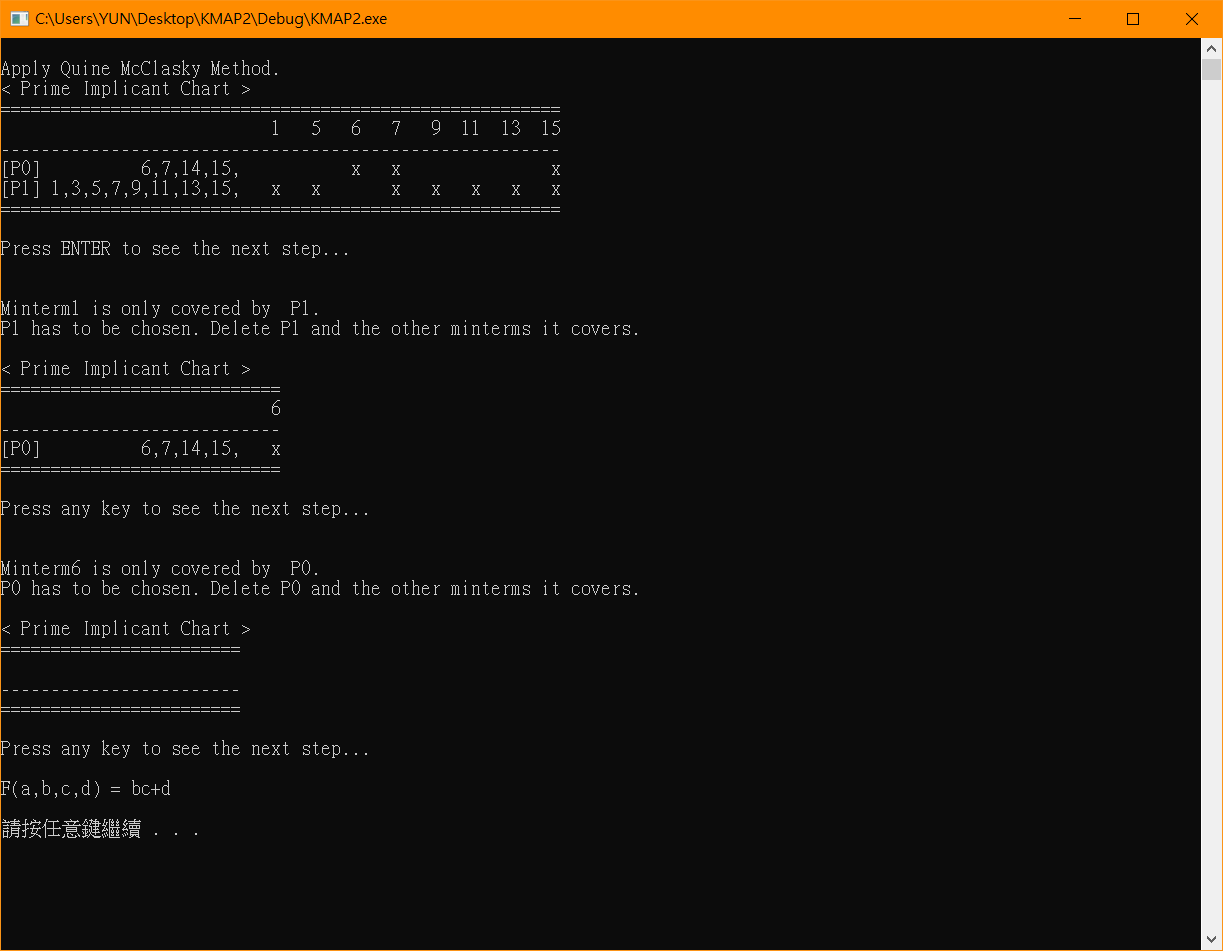
* 1. 將要讀取的檔案名稱設為input.txt，並與其他檔案放置在同一資料夾中。  
     input.txt格式：  
     將所要進行化簡的四變數布林代數式以下方範例之SOP格式表示。

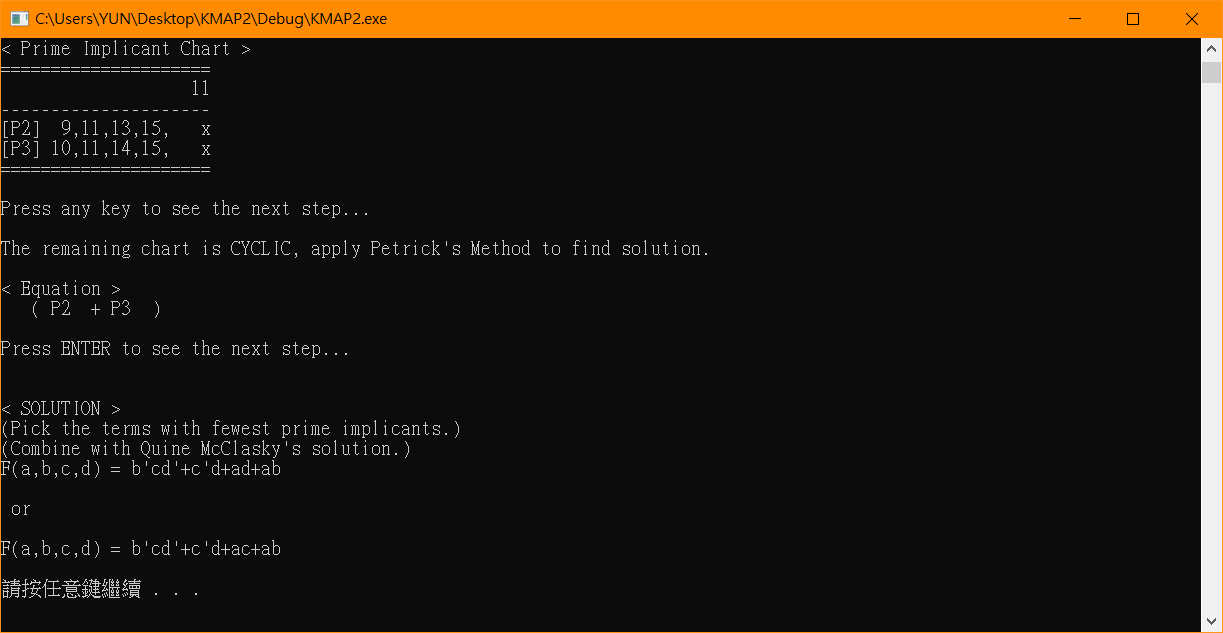
\*Don’t care 部分請寫在( )內。  
範例：ad+a'bc+c'd+(a'b'cd)+(abcd')

* 1. 開啟simulator.exe，執行畫面如下：
  2. 請依指示輸入：

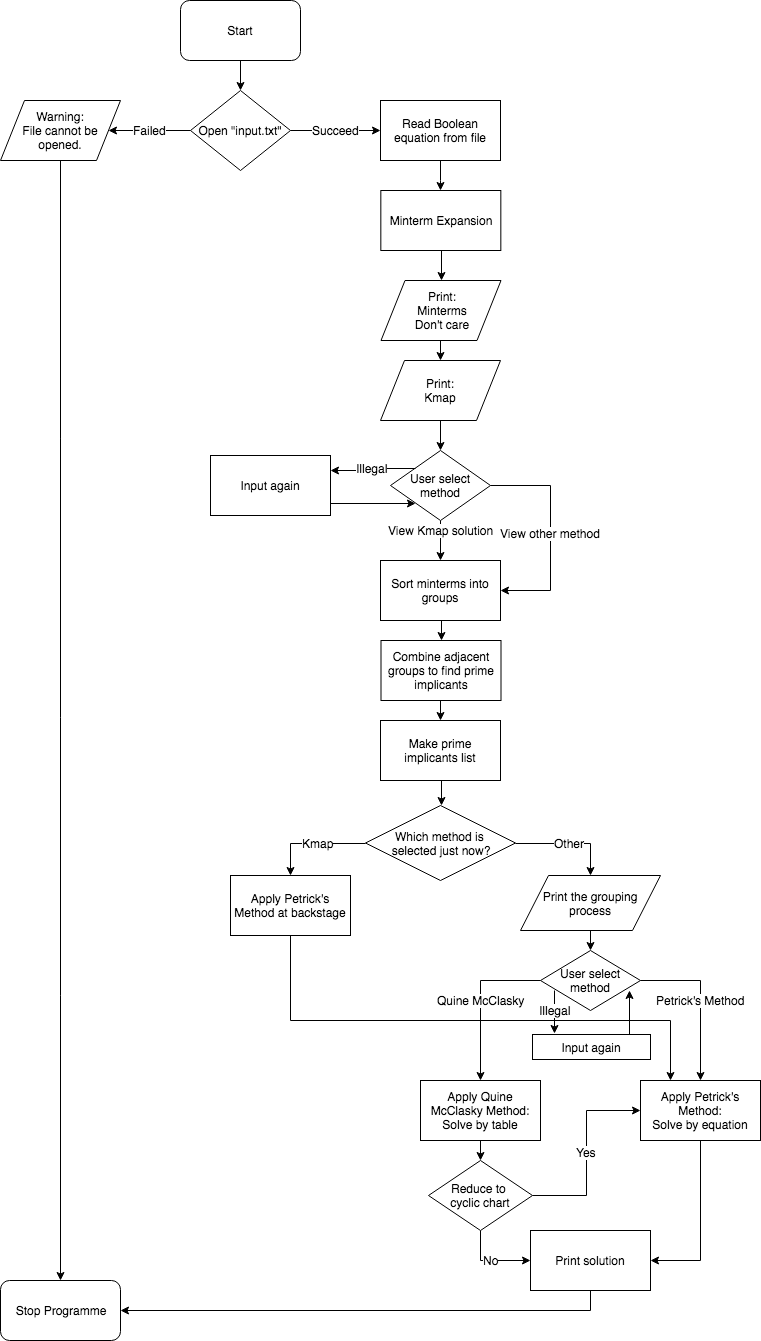
1 🡪 觀看卡諾圖之解法與式子化簡結果  
2 🡪 觀看其他解法（Quine McClasky 或 Petrick’s Method）與式子化簡結果（會顯示詳細步驟）

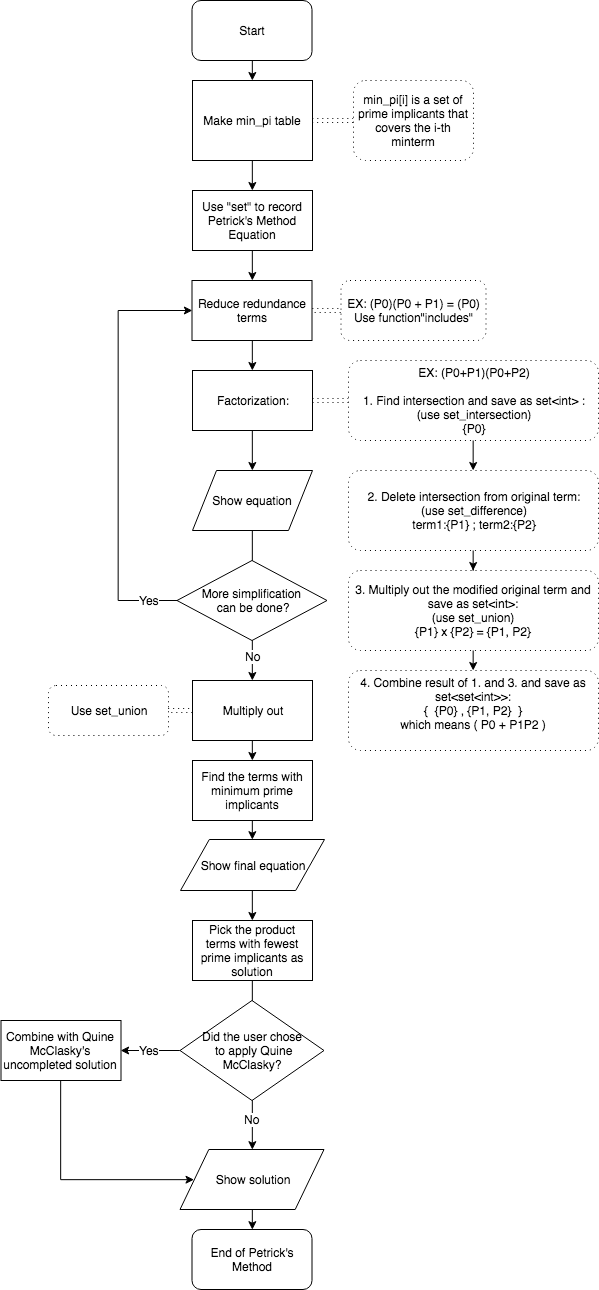
* 1. 若選擇1，執行畫面如下：
  2. 若輸入2，執行畫面如下：（請依照指示操作以繼續下一步）
  3. 執行至Step3時，請依指示輸入，選擇欲使用之解法：  
     1 🡪 使用Quine McClasky   
     2 🡪 使用Petrick’s Method
  4. 若輸入1：Quine McClasky，執行畫面如下：（請依照指示操作以繼續下一步）

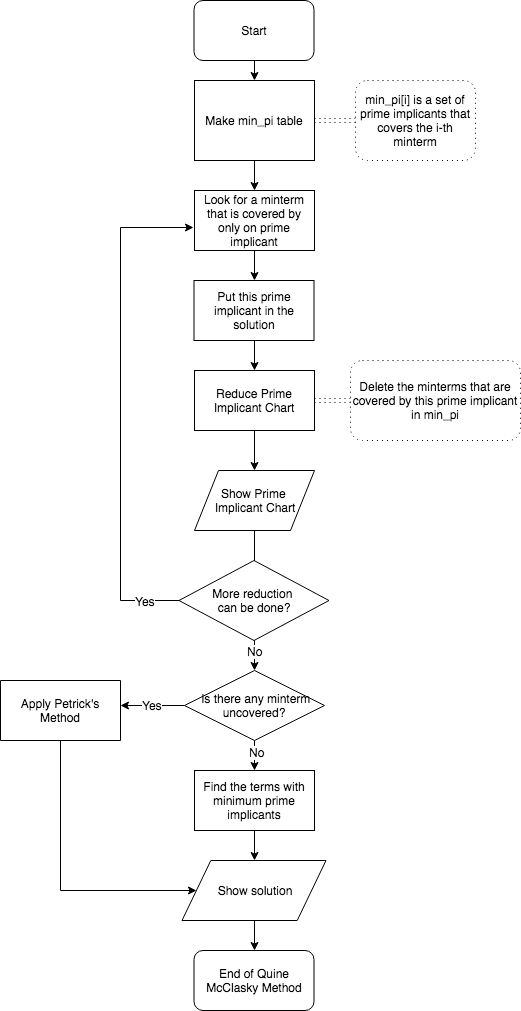


＊若有多組解，將接續使用Petrick’s Method找出所有可能解，如下：

* 1. 若輸入2，執行畫面如下：（請依照指示操作以繼續下一步）

1. 程式撰寫流程（已省略繁瑣微小的細節）
   1. 主程式Source.cpp  
      
   2. Petricks Method



* 1. Quine McClasky Method  
     

1. 作業心得

這次的作業，我選擇不使用直接在卡諾圖上畫圈，而採用了Quine McClasky 和 Petrick’s Method，一來是因為我認為卡諾圖本身是設計給人類使用的方便工具，而Quine McClasky 和 Petrick’s Method是較有系統的，只是前置作業是必須一一將Prime Implicants 做組合，複雜度對人類來說比較大，但電腦運算夠快，較適合這種系統性的方法。二來，在變數數量改變時，礙於表格設計方式迥異，我們所寫的四變數卡諾圖的程式也就失去了效用，但另外兩種方法卻不受影響，因此我決定試著寫出這兩種方法的程式，即便這次的作業沒有要求，未來我也能使用這支自己寫的程式去解多樣化的題型。

其中Petrick’s Method 是最花時間的。它並不如手寫時那麼間單，實際開始構思後我才想到，如：(P1+P2)( P1+P3) = P1 + P2P3不是只要設int變數來存1、2、3代表P1、P2、P3，然後找交集差集就能輕鬆了事，因為繼續化簡下去，X、Y、Z都可能變成Pn的組合：  
如(X+Y)(X+Z)= (X+YZ)變成  
狀況 (P1 + P2+P3 ) ( P2+P3 + P4+P5 ) = ( P2+P3 + P1 (P4+P5) )

狀況 (P1 + P2+P3 ) ( P1 + P4+P5 ) = ( P1 + (P2+P3) (P4+P5) )

遇到這樣複雜的狀況時，便會出現許多問題：

狀況 X = P2+P3，無法單純用int代表X

狀況 產生新的POS，但這個方法就是要把POS化成minimum SOP，這樣會沒完沒了。

由於種種問題，這個部分我想了非常久，最後想到用set來存，而且set的元素的唯一性也相當方便，但麻煩的是設計出來的是整個式子會變成三層的set，我在寫程式時被搞混了很多次，有大半的時間都在處理這個部分。

自己手動解卡諾圖時，可以很直接地在圖上圈出正解，但將過程用程式寫出來卻相對麻煩很多。但設計程式時，由於無法像直觀地想怎麼在紙上做記號就直接動筆，而必須一步一步地用程式碼呈現，整個解題過程中必須將邏輯徹底理清，因此設計完之後，縱然我相信效能優化還有很大的進步空間，但透過不斷地除錯修正，也能夠讓人對解題方法更加融會貫通。

1. Source Code說明

|  |  |  |
| --- | --- | --- |
| 主程式 | 內容 | |
| Source.cpp  （參考  **二、1. Source.cpp**  **流程圖**） | 使用C++ library | |
| Library | 備註 |
| #include <iostream> |  |
| #include <fstream> | 讀檔使用到ifstream |
| #include <cmath> | Minterm Expansion中計算2b 🡪 pow(2,b) |
| #include <iomanip> | 使用格式化輸出 🡪 setw(x) |
| #include <algorithm> | 取set交集 🡪 set\_intersection  取set差集 🡪 set\_difference  取set聯集 🡪 set\_union |
| #include <iterator> | 使用上方三個函式時需用到inserter |
| #include <string> |  |
| #include <vector> |  |
| #include <set> |  |
| 詳細說明 | |
| 自建函式 | |
| void mintermExpansion  (int min\_sop[16], int output, int minterm, int term[4], int b) {…} | |
| void getSolution  (vector<Cmbterms> &pi\_list, vector<int> minterms, const int min\_sop[16], const char &choice, const char &choice2) {…} | |

|  |  |  |  |
| --- | --- | --- | --- |
| 自建Class | 內容 | | |
| Cmbterms.h  Cmbterms.cpp (class Cmbterms的header file和implementation file) | 使用C++ library | | |
| Library | 備註 | |
| #include <iostream> |  | |
| #include <iomanip> | 使用格式化輸出 🡪 setw(x) | |
| #include <set> |  | |
| #include <vector> |  | |
| #include <algorithm> | 取set交集 🡪 set\_intersection  取set差集 🡪 set\_difference  取set聯集 🡪 set\_union | |
| #include <iterator> | 使用上方三個函式時需用到 inserter | |
| 詳細說明 | | |
| 為implicants的合併與分組而設計 | | |
| 主要Data members和Member functions | | |
| set<int> minterm; | | 儲存包含的minterms |
| set<int> pos\_1; | | 儲存二進位表示法中1在第幾個bit |
| set<int> pos\_x; | | 儲存二進位表示法中 – (don’t care)  在第幾個bit |
| bool checked; | | 記錄是否已被合併 |
| Bool can Combine (constCmbterms&c2) {…} | | 回傳是否能和c2合併 |
| Cmbterms operator+(Cmbterms &c2)  {…} | | 回傳和c2合併後的結果 |

|  |  |  |  |
| --- | --- | --- | --- |
| 副程式 | 內容 | | |
| print.h  print.cpp  (格式化輸出函式的header file 和 implementation file) | 使用C++ library | | |
| Library | 備註 | |
| #include <iostream> |  | |
| #include <sstream> | printPiChart()中將Cmbterms包含的minterms轉為string時使用 | |
| #include <iomanip> |  | |
| #include <string> |  | |
| #include <vector> |  | |
| #include <set> |  | |
| 詳細說明 | | |
| 將格式化輸出函式獨立以使主程式功能單純化 | | |
| void printSet(set<int> set) | | 印出Petrick’s Method式子中的Pn |
| void printSset(set< set<int> > sset) | | 印出Petrick’s Method式子中的Pn組成的product term |
| void printSum(set< set< set<int> > > sum) | | 印出Petrick’s Method完整的式子 |
| void printKmap(const int min\_sop[16]) | |  |
| void printGroup(const vector< vector<Cmbterms> > &G) | | 印出將implicants合併、分組的過程和結果 |
| void printPiChart(  const vector< Cmbterms > &pi\_list, const vector<int> &minterms,  const vector<int> &isPruned\_pi) | | 印出Quine McClasky和Petrick’s Method的Prime Implicant Chart |