Logic System Assignment 2 – A simple CAD tool based on Quine-McCluskey (QM) Method					
系級	電機 115	學號	E24116071	姓名	劉又慈
Language : C++					

- Code explanation

一、 合併階段

我使用 vector<vector<pair<set<int>, string>>>的結構來實現 QM 法中的每個 column，另外宣告了一個全域 set<int> nothing = { -1 }，用來放置於僅需要 string 資料的 pair 當中，以下圖示說明該結構:

	0	1	2
0	<minterm 編號, binary>	<nothing, 是否使用過 [1]>	<nothing, group[2]>
…	…	…	…

註、1:用"x"表示該項未被合併，"v"表示該項已被合併。

2:用 string 型態的數字紀錄這一項有幾個 1。

每一次的合併使用雙重迴圈遍歷整個二維 vector，在遍歷的過程中依照以下判斷順序來決定兩項是否合併:

1. 比對 group 的是否相鄰，此處使用 stoi()將 string 型態的 group 轉成 int，然後將迴圈當前指到的兩項 group 相減，若為 1，即表示兩 group 相鄰。
2. 比對兩項的 binary 相異位數是否為 1，此處使用 function NumOfDiff 做判斷，如果該 function 回傳值為 1，代表 binary 相異位數為 1，即兩項可以合併。

如果可以合併，就會使用 set.insert()的功能將 minterm 編號合併，binary 相異的位置改成'-'，然後預設此新合併出的項使用狀態為"x"，反之，進行合併的兩項使用狀態改為"v"，代表已經被使用，重新計算新合併的 group 後，存入下一個二維 vector。當雙重迴圈結束遍歷，代表下一個 column 已經建立完成，此時會回去看上一個 column 是否有未被合併的項(即使用狀態為"x")，若有該項就是 prime implicant，會將他存入一個名為 PI 的 vector<pair<set<int>, string>>結構，以上將重複操作直到合併次數等於 numVar。

二、 找 minimum SOP 階段

使用 removeSamePI 的 function 首先去除都是 don't care 組成的 prime implicant，此處使用使用三層迴圈，計算這個 prime implicant 裡面的包住的有幾個 don't care，如果計算而得的這個數量等於 prime implicant 所包含的項的數量，就表示整個 prime implicant 都是 don't care 組成的，要將其去除。接著使用

set 結構內資料不重複的性質去除重複的 prime implicant，由此我們會得到一個精簡過的 vector<pair<set<int>, string>>叫做 newPI，用於存放所有 prime implicant。

下一步要從 newPI 中找到 essential prime implicant，作法是先用雙重迴圈尋找哪個 minterm 只有被一個 prime implicant 包含，記錄這個 minterm 後再用雙重迴圈找出有這個 minterm 的 prime implicant，把這個 prime implicant 存入 esPI，是一個 vector<pair<set<int>, string>>的結構，然後就可以把這個 prime implicant 從 newPI 當中刪除，完成以上操作後可以得到 essential prime implicant 和剩下的 prime implicant，分別儲存於兩個二維 vector 中。

接著我使用 Patrick Method 來尋找 minimum SOP:

1. 把已經被 esPI 包含的 minterm 刪去，找到還沒有被涵括的 minterm 有哪些。
2. 使用雙重迴圈尋找有這幾個 minterm 的 prime implicant，將這個 prime implicant，在 newPI 的 index 存入 pat，是一個 vector<set<int>>的結構，一格裡存一個 minterm 找到的結果，以下圖示說明:

0	1	2	…
<newPI 的 index>	<newPI 的 index>	<newPI 的 index>	…

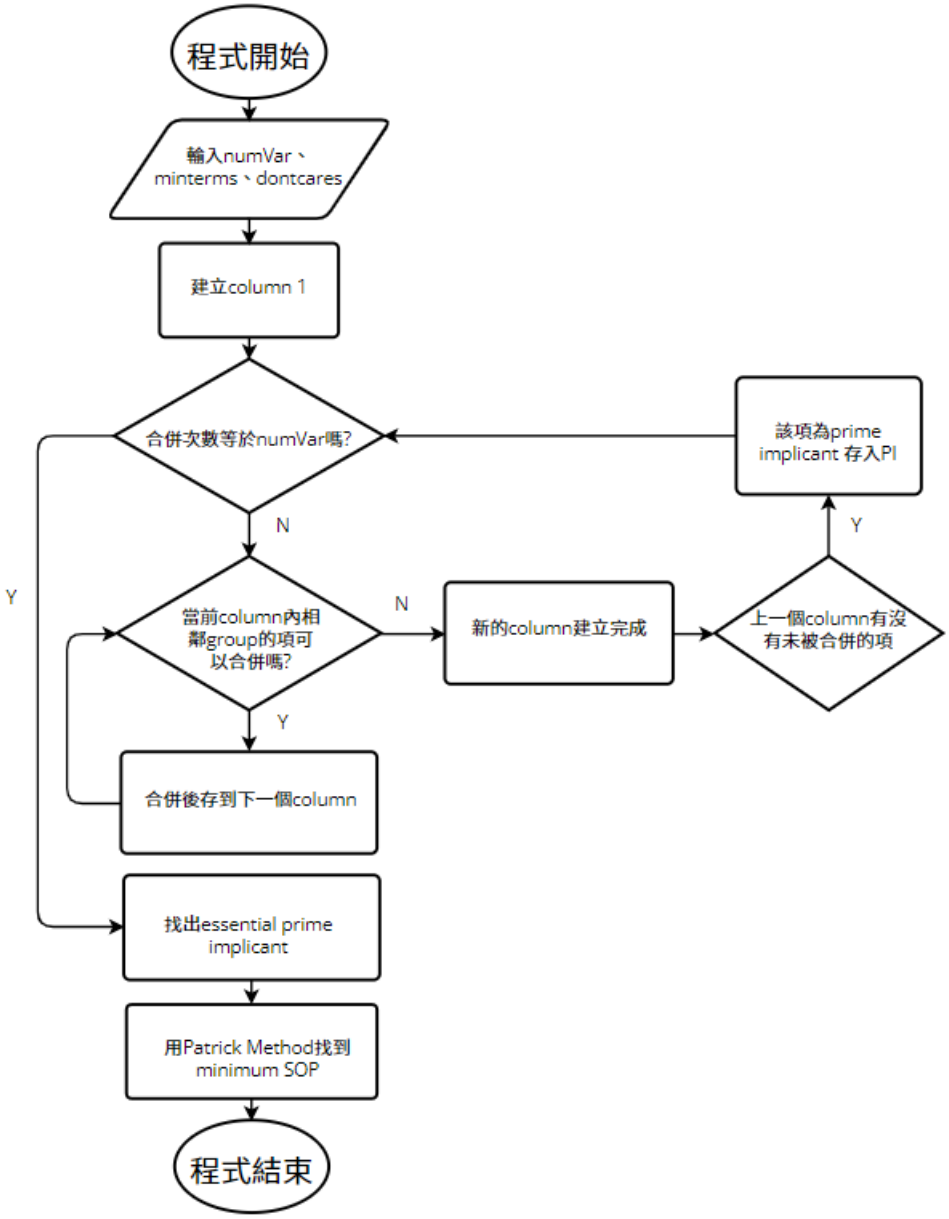
舉例來說，假如 5、7、112 這幾個 minterm 沒有被 essential prime implicant 包含，那上圖 0 這一格就會存有包含 5 這個 minterm 的 prime implicant 在 newPI 中的 index，包含 7 這個 minterm 的 prime implicant 在 newPI 中的 index 就會被放到 1 這一格，以此類推。

3. 用遞迴的的方式生成 pat 集合中元素的所有可能组合，並把他們存在 allSOP 這個 set<set<int>>中。遞迴的邏輯如下:
   i. Function ‘expand’接受四个參數：
      - pat：儲存每個尚未被包含的 minterm 有出現的 prime implicant。
      - level：是 int，表示現在在 pat 的哪一格當中。
      - product：一個 set<int>，表示到目前為止元素的乘積。
      - allSOP：儲存結果的二維 set。
   ii. 遞迴中，當 level 等於 pat 的大小時，當下 product 中存的資料表示一個完整的組合，也就是一種 SOP，因此插入到 allSOP 中。
   iii. 遞迴中會遍歷 pat 中每個 level 中的每個元素。把元素插入到 product 後再對 level + 1 進行遞迴。

iv. 在進行遞迴之前，會檢查當前元素是否已在 product 中。如果已有，那麼在該次遞迴結束後，要把它從 product 中 erase，以恢復 product 的原狀態。

4. 找到所有有可能的 SOP 之後，用互相比較的方式找到最小的那些 set，因為這代表他們將剩下的 minterm 都含括時所使用的 prime implicant 數最少，再加上 essential prime implicant 就找到 minimum SOP。

- Flow Chart



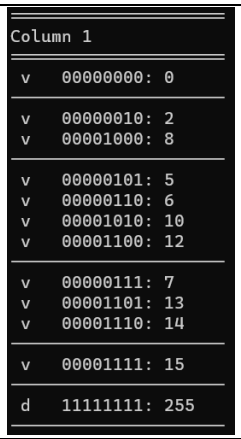
- Some test results

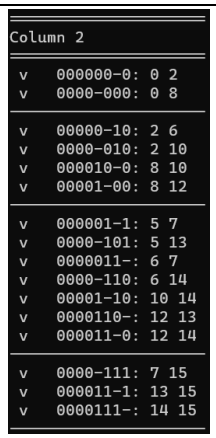
測資一

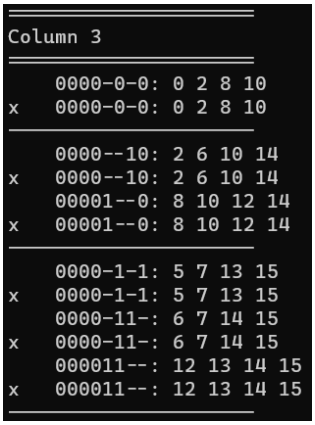
numVar = 8

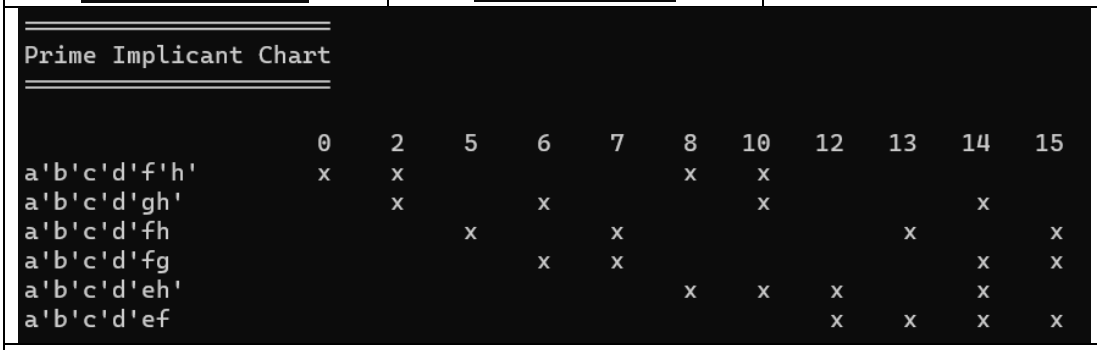
minterms = [0, 2, 5, 6, 7, 8, 10, 12, 13, 14, 15]

dontcares = [255]







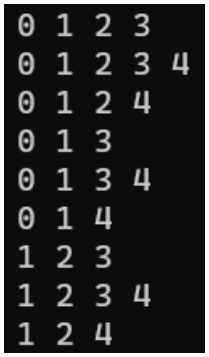


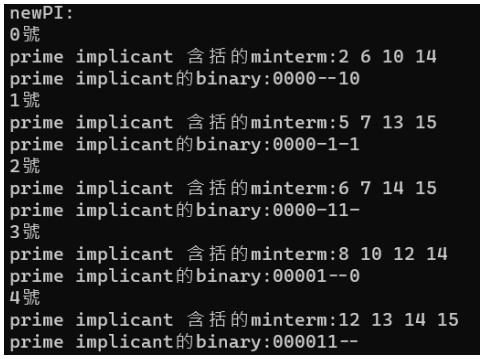

```
prime implicants:
a'b'c'd'f'h' a'b'c'd'gh' a'b'c'd'fh a'b'c'd'fg a'b'c'd'eh' a'b'c'd'ef
minimun SOPs:
a'b'c'd'gh'  a'b'c'd'fh  a'b'c'd'eh'  a'b'c'd'f'h'
a'b'c'd'gh'  a'b'c'd'fh  a'b'c'd'ef  a'b'c'd'f'h'
a'b'c'd'fh  a'b'c'd'fg  a'b'c'd'eh'  a'b'c'd'f'h'
a'b'c'd'fh  a'b'c'd'fg  a'b'c'd'ef  a'b'c'd'f'h'
```

說明

1. 首先會先建立儲存 column 1 資料的二維 vector，並判斷哪幾項可以合併(原理已在上個部分說明)，合併後的項會放入儲存 column 2 資料的二維 vector，重複此步驟直到合併次數等於 numVar，測資一最多合併兩次，所以第三到第八次的合併會因為程式碼中 if 的判斷條件沒有符合，所以沒有資料被放入後面的 column。
2. 圖片中沒有打勾的項就是 prime implicant，將這些項目儲存於一維 vector。

3. 找出 essential prime implicant，由 prime implicant chart 可以看出這筆測資的 essential prime implicant 是 a’b’c’d’fh 和 a’b’c’d’f’h’，因為 minterm 2 和 minterm 5 分別只有被這兩個 prime implicant 包含。
4. 用 Patrick Method 找出剩下的 prime implicant 可包含所有 minterm 的所有組合。左圖是遞迴找到的所有組合，數字對應 prime implicant 在 newPI 裡的 index，右圖是 newPI 裡存的資料:





5. 找出使用最少 prime implicant 的組合，也就是上面左圖的第四、六、七、九行，把這些編號轉換回對應的 prime implicant 再加上 essential prime implicant 就找到所有 minimum SOP 了。

測資二

numVar = 9

minterms = [24 37 83 138 217 228 269 354 368 376 415 476 508]

dontcares = [54 175 214 301 316 332 336 358 398 412 428 473]

```
Column 1
    000011000: 24
    000100101: 37
    010001010: 138
 d  101010000: 336
    001010011: 83
    011100100: 228
 v  100001101: 269
 v  101100010: 354
 v  101110000: 368
 d  000110110: 54
 d  101001100: 332
 v  011011001: 217
 v  101111000: 376
 d  011010110: 214
 d  100101101: 301
 d  100111100: 316
 d  101100110: 358
 d  110001110: 398
 d  110011100: 412
 d  110101100: 428
 v  111011100: 476
 d  010101111: 175
 d  111011001: 473
    110011111: 415
 v  111111100: 508
```

```
Column 2
   101-10000: 336 368
   100-01101: 269 301
   101100-10: 354 358
   10111-000: 368 376
   -11011001: 217 473
   11-011100: 412 476
   111-11100: 476 508
```

```
Prime Implicant Chart

                  24   37   83  138  217  228  269  354  368  376  415  476  508
a'b'c'd'efg'h'i'    x
a'b'c'de'f'gh'i          x
a'b'cd'ef'g'hi                x
a'bc'd'e'fg'hi'                    x
bcd'efg'h'i                             x
a'bcde'f'gh'i'                               x
ab'c'e'fgh'i                                      x
ab'cef'g'h'i'                                          x
ab'cde'f'hi'                                      x
ab'cdeg'h'i'                                           x    x
abd'efgh'i'                                                      x
abc'd'efghi                                                 x
abcefgh'i'                                                       x    x
```

```
prime implicants:
a'b'c'd'efg'h'i' a'b'c'de'f'gh'i a'b'cd'ef'g'hi a'bc'd'e'fg'hi' bcd'efg'h'i a'bcde'f'gh'i' ab'c'e'f
h'i ab'cef'g'h'i' ab'cde'f'hi' ab'cdeg'h'i' abd'efgh'i' abc'd'efghi abcefgh'i'
minimun SOPs:
a'b'c'de'f'gh'i a'b'cd'ef'g'hi a'bc'd'e'fg'hi' bcd'efg'h'i a'bcde'f'gh'i' ab'c'e'fgh'i ab'cde'f'hi'
ab'cdeg'h'i' abc'd'efghi abcefgh'i' a'b'c'd'efg'h'i'
```

測資三

numVar = 10

minterms = [0, 1, 16, 17, 128, 343, 512, 640, 1023]

dontcares = [341]

```
Column 1
 v 0000000000: 0
 v 0000000001: 1
 v 0000010000: 16
 v 0010000000: 128
 v 1000000000: 512
 v 0000010001: 17
 v 1010000000: 640
 d 0101010101: 341
 v 0101010111: 343
   1111111111: 1023
```

```
Column 2
 v 000000000-: 0 1
 v 00000-0000: 0 16
 v 00-0000000: 0 128
 v -000000000: 0 512
 v 00000-0001: 1 17
 v 000001000-: 16 17
 v -010000000: 128 640
 v 10-0000000: 512 640
   01010101-1: 341 343
```

```
Column 3
  00000-000-: 0 1 16 17
x 00000-000-: 0 1 16 17
  -0-0000000: 0 128 512 640
x -0-0000000: 0 128 512 640
```

```
Prime Implicant Chart

                    0    1   16   17  128  343  512  640 1023
a'b'c'd'e'g'h'i'    x    x    x    x
b'd'e'f'g'h'i'j'    x                   x         x    x
a'bc'de'fg'hj                                x
abcdefghij                                                x
```

```
prime implicants:
a'b'c'd'e'g'h'i' b'd'e'f'g'h'i'j' a'bc'de'fg'hj abcdefghij
minimun SOPs:
a'b'c'd'e'g'h'i' b'd'e'f'g'h'i'j' a'bc'de'fg'hj abcdefghij
```