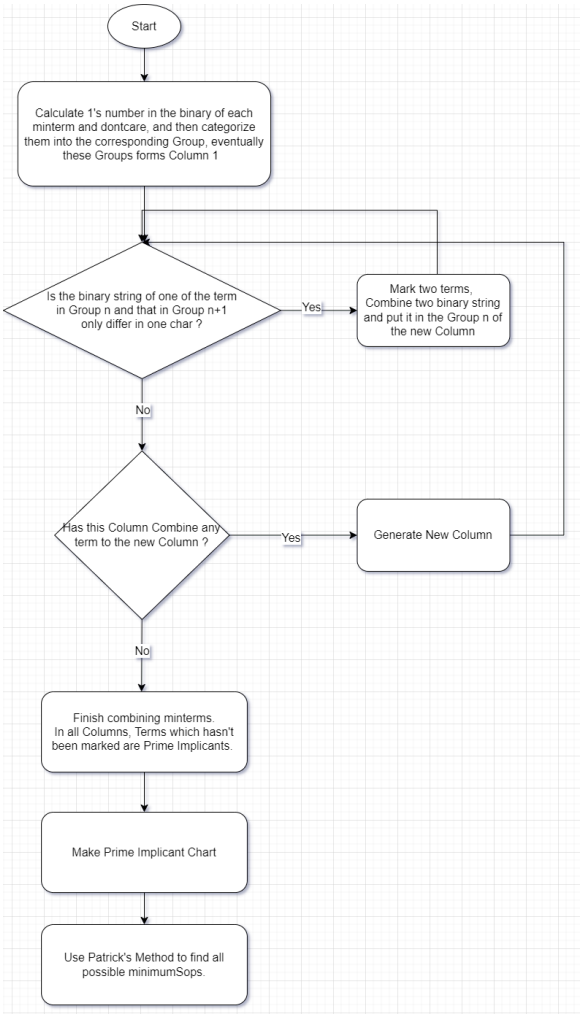
QM Method Report

E24126270 蔡承希

Flow Chart



How Program Find Primes

1. 首先，我將 `minterms` 和 `dontcares` 放入自定的類別 `Term` 當中，

`Term` 類別有五個成員變數：

1. `vector<int> minterms` ⇒ 用來存放此 Implicant 的 minterm
2. `string bin_str` ⇒ 用來存放此 Implicant 的二進位數字的字串
3. `bool is_checked` ⇒ 用來標示此 Implicant 是否已被合併
4. `bool is_dontcare` ⇒ 用來標示此 Implicant 是否為 dontcare (的組合)
5. `bool is_duplicated` ⇒ 用來標示此 Implicant 是否跟同 Group 的其他 Term 重複
   (重複指 minterms 一樣)

並將 `dontcares` 的 `Term` 的 `is_duplicated` 標記為 True，

再利用函數 `int count_bin_ones(int n)` 計算該 minterm 的二進位數字所含的 1 的數目，

`int count_bin_ones(int n)` 的原理是每次將 n 與 1 做 AND 運算，若 n 的第 0 bit 為 1 則 cnt += 1，否則就不加，最後再對 n 做 bitwise operation 將所有 bit 向右位移 1 格

```
int count_bin_ones(int n) {
    int cnt = 0;
    while(n) {
        cnt += n & 1;
        n >>= 1;
    }
    return cnt;
}
```

最後將這些 Term 以該數目為 Group (`column` 的 key)，

分類放入 `map<int, vector<Term*>> column` (Column) 當中。

Group 的資料結構以 vector<Term*> 表示

2. 接著將第 n Group (`column[n]`) 的所有 Term 與第 n+1 Group (`column[n+1]`)

   的所有 Term 去做比較 (n = 0, 1, 2, 3, ...)，利用函數 `int diff_one_bit(Term *a, Term *b)`

`int diff_one_bit(Term *a, Term *b)` 的原理單純就是把 a 跟 b 的 bin_str 的每一個字元做比較：

```
int diff_one_bit(Term *a, Term *b) {
    int cnt = 0;
    int diff_bit;
    for (int i = 0; i < _numVar; i++) {
        if (a->bin_str[i] != b->bin_str[i]) {
            diff_bit = i;
            if (cnt++) return -1;
        }
    }
    return diff_bit;
}
```

檢查兩個 Term 的 bin_str 是否只差一個字元，若是則表示可融合，接著執行以下動作：

a. 將 `bool out_flag` (用來控制是否要跳出循環) 設為 False

b. 分別將兩個 Term 的 `bool is_checked` 設為 True (表示已融合過)

c. 新增一個 Term 到 `map<int, vector<Term*>> new_column` (即下一個 Column)：

- 將原本的 bin_str 相差的字元改成 "-" 後指派給新 Term 的 `string bin_str`
- 將原本兩個 Term 的 minterms 合併到新 Term 的 `vector<int> minterms`
- 若原本兩個 Term 的 `bool is_dontcare` 為 True，則也將新 Term 的 `bool is_dontcare` 設為 True
- 利用 C++ STL 的 set 容器 (`set<set<int>> avoid_duplicate`) 不會有重複元素的特性，檢查新 Term 的 minterms 是否與同 Group 的其他新 Term 重複，若沒有才將新 Term 加入 `map<int, vector<Term*>> new_column` ，再將該 Term 的 minterms 資訊加入 `set<set<int>> avoid_duplicate`

d. 將目前的 Column 的所有 Term 進行篩選，如果有 `bool is_checked`、`bool is_duplicated`、`bool is_dontcare` 皆為 0 的 Term 的話，表示其為一個 Prime Implicant，因此將其透過函數 `string gen_literal(string bin_str)` 將 bin_str 轉換為 literals 後加入 `vector<string> primes` 中

e. 將 `map<int, vector<Term*>> new_column` 指派給 `map<int, vector<Term*>> column` (更新 Column)

接著利用 while 迴圈條件 `!out_flag` 檢查此輪是否有融合任兩個 Term 過，若有則回到 2. 的開頭繼續融合下一個 Column 中的 Term，若沒有則跳出 while 迴圈，表示已找到所有 Prime Implicants。

3. 建立 Prime Implicant Chart：

a. 建立 `map<int, vector<Term*>> cmt` 用來紀錄哪個 minterm 被哪些 Term 包含，
以 key 為 minterm，value 為 Term* 的 vector 的 STL map 容器。

b. 搜尋 Prime Implicant 中所有 Term 底下的 minterms，若某 Term 有包含到某 minterm，就將其加入 `cmt[minterm]` 底下。

How Program Find MinimumSOPs

1. 接著進行 Patrick's Method 以篩選出 minimumSOPs：

a. 建立 `vector<set<set<Term*>>> pm` 用來存放 Patrick's Method 的 POSOP

> 以 (P0)(P1+P2)(P1+P3P4) 為例，
> 第一層的 vector 存 POSOP，共有三個元素：(P0)、(P1+P2)、(P1+P3P4)
> 第二層的 set 存 SOP，如 P1+P3P4 有兩個元素：(P1)、(P3P4)
> 第三層的 set 存 P，如 P3P4 有兩個元素 (P3)、(P4)

b. 將 `map<int, vector<Term*>> cmt` 中的所有 minterm ，先將 `vector<Term*>` 中所有 Term* 轉為 {Term*} (即轉為 set)， 再加入 `set<set<Term*>>` 後 (作為一個乘積項) 加入 `vector<set<set<Term*>>> pm` 當中。

> 舉例：
> cmt[1] = { *Term0 } (表示 minterm 1 被 Term0 包含)
> cmt[3] = { *Term1, *Term2 } (表示 minterm 3 被 Term1、Term2 包含)
> cmt[7] = { *Term1, *Term3 }
> cmt[8] = { *Term1, *Term4 }
> 則
> pm = {
> set{ set{*Term0} },
> set{ set{*Term1}, set{*Term2} },
> set{ set{*Term1}, set{*Term3} },
> set{ set{*Term1}, set{*Term4} }
> }

c. 將 `vector<set<set<Term*>>> pm` 最後一個 SOP 與倒數第二個 SOP 合併，即分配律。

將倒數第二個 SOP (`pm[pm.size()-2]`) 中的每個項跟最後一個 SOP (`pm[pm.size()-2]`) 合併後取代倒數第二個 SOP，再將最後一個 SOP 刪除。

> 💡 此時使用 C++ STL 的 set 容器就會許多好處，
> 因為 set 容器有著不會有重複元素的特性，
> 對於
> `set<Term*>` 而言，XX = X 不用額外去變換，因為將兩個相同的 `Term*` 加入 set 後不會有重複的問題。
> 對於
> `set<set<Term*>>` 而言，XY+XY = XY 不用額外去變換，因為將兩個相同的 `set<Term*>` 加入 set 後不會有重複的問題。

重複進行 c. 直到 pm 只剩一項。

> 💡 舉例：
> pm = {
> set{ set{*Term0} },
> set{ set{*Term1}, set{*Term2} },
> set{ set{*Term1}, set{*Term3} },
> set{ set{*Term1}, set{*Term4} }
> }
> 第一次合併變為：
> pm = {
> set{ set{*Term0} },
> set{ set{*Term1}, set{*Term2} },
> set{ set{*Term1, *Term1}, set{*Term1, *Term3},
> set{*Term3, *Term1}, set{*Term3, *Term4} }
> }
> 但因為 set 容器有著不會重複(且會自動排序)的特性，所以會變為：
> pm = {
> set{ set{*Term0} },
> set{ set{*Term1}, set{*Term2} },
> set{ set{*Term1}, set{*Term1, *Term3}, set{*Term3, *Term4} }
> }
> 第二次合併變為(省略顯示 set 容器的特性)：
> pm = {
> set{ set{*Term0} },
> set{ set{*Term1}, set{*Term1, *Term3}, set{*Term1, *Term3, *Term4},
> set{*Term1, *Term2}, set{*Term1, *Term2, *Term3}
> }
> 第三次合併變為(省略顯示 set 容器的特性)：
> pm = {
> set{ set{*Term0, *Term1}, set{*Term0, *Term1, *Term3},
> set{*Term0, *Term1, *Term3, *Term4},
> set{*Term0, *Term1, *Term2}, set{*Term0, *Term1, *Term2, *Term3}
> }
> 此時 pm 只剩一項，停止再合併。

d. 此時挑項數最少的 `set<Term*>` 即為 minimumSop。

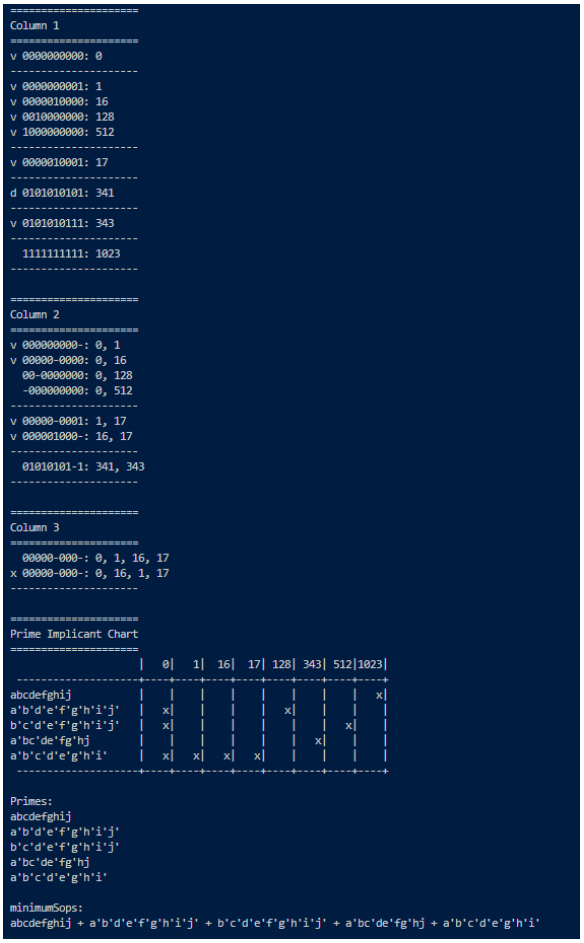
(若有多個項數最少的 `set<Term*>`，則有多個 minimumSop)

將那些 `set<Term*>` 中的 `Term*` 的 bin_str 經過函數 `gen_literal(string bin_str)` 轉換為 literals 後加入 `set<vector<string>>` `minimumSops_set` ，最後經過 set 不重複特性篩選後，再把 `minimumSops_set` 中的元素加入 `vector<vector<string>>` `minimumSops` 中。

> 💡 舉例：
> pm = {
> set{ set{*Term0, *Term1}, set{*Term0, *Term1, *Term3},
> set{*Term0, *Term1, *Term3, *Term4},
> set{*Term0, *Term1, *Term2}, set{*Term0, *Term1, *Term2, *Term3}
> }
> 檢查後得知最少的項為兩項，因此 set{*Term0, *Term1} 即為 minimumSop

Testcase 1

```
numVar = 10
minterms = [0, 1, 16, 17, 128, 343, 512, 640, 1023]
dontcares = [341]
verbose = true
```



1. 將 **minterms** (is_dontcare=False) 和 **dontcares** (is_dontcare=True) 的每個元素轉為 Term，生成 **bin_str** 後加入 **column** (1)
2. 將 **columns** (1) 中：
   - Group 0 與 Group 1 比對，
     - 發現 “0000000000”: 0 與 “0000000001”: 1 只差一位
       - 將兩者 **is_checked** 設為 True
       - 建立新 **Term**，放入 **new_column[0]**：

                - **bin_str** = “000000000-”
                - **minterm** = {0, 1}
                - 因為兩個皆非全為 dontcare，所以 **is_dontcare** = False
            - 加入 **avoid_duplicate**，以免之後造出重複的 **Term**
        - 發現 “0000000000”: 0 與 “0000010000”: 16 只差一位
            - 將兩者 **is_checked** 設為 True
            - 建立新 **Term**，放入 **new_column[0]**：
                - **bin_str** = “00000-0000”
                - **minterm** = {0, 16}
                - 因為兩個皆非全為 dontcare，所以 **is_dontcare** = False
            - 加入 **avoid_duplicate**，以免之後造出重複的 **Term**
        - (下略)
    - Group 1 與 Group 2 比對，
        - 發現 “0000000001”: 1 與 “0000010001”: 17 只差一位
            - 將兩者 **is_checked** 設為 True
            - 建立新 **Term**，放入 **new_column[1]**：
                - **bin_str** = “00000-0001”
                - **minterm** = {1, 17}
                - 因為兩個皆非全為 dontcare，所以 **is_dontcare** = False
            - 加入 **avoid_duplicate**，以免之後造出重複的 **Term**
        - 發現 “0000010000”: 16 與 “ 0000010001”: 17 只差一位
            - 將兩者 **is_checked** 設為 True
            - 建立新 **Term**，放入 **new_column[1]**：
                - **bin_str** = “000001000-”
                - **minterm** = {16, 17}
                - 因為兩個皆非全為 dontcare，所以 **is_dontcare** = False
            - 加入 **avoid_duplicate**，以免之後造出重複的 **Term**
    - 因為 Group 3 是空的，所以 Group 2 不用與其比對
    - Group 5 與 Group 6 比對，
        - 發現 “0101010101”: 341 與 “0101010111”: 343 只差一位
            - 將兩者 **is_checked** 設為 True
            - 建立新 **Term**，放入 **new_column[5]**：
                - **bin_str** = “01010101-1”
                - **minterm** = {341, 343}
                - 因為兩個皆非全為 dontcare，所以 **is_dontcare** = False
            - 加入 **avoid_duplicate**，以免之後造出重複的 **Term**
    - 因為 Group 11 是空的，所以 Group 10 不用與其比對
3. 將 new_column 指派給 column，即變成 column (2)
    - 再將 new_column 清空
4. Group 0 與 Group 1 比對，
    - 發現 “000000000-”: 0, 1 與 “000001000-”: 16, 17 只差一位

    - 將兩者 **is_checked** 設為 True
    - 建立新 **Term**，放入 **new_column[0]**：
        - **bin_str** = “00000-000-”
        - **minterm** = {0, 1, 16, 17}
        - 因為兩個皆非全為 dontcare，所以 **is_dontcare** = False
    - 加入 **avoid_duplicate**，以免之後造出重複的 **Term**
- 發現 “00000-0000”: 0, 16 與 “00000-0001”: 1, 17 只差一位
    - 將兩者 **is_checked** 設為 True
    - 建立新 **Term**，放入 **new_column[0]**：
        - **bin_str** = “00000-000-”
        - **minterm** = {0, 1, 16, 17}
        - 因為兩個皆非全為 dontcare，所以 **is_dontcare** = False
    - 加入 **avoid_duplicate**，發現此 Term 有重複，因此將其 **is_duplicated** 設為 True

5. 將 **new_column** 指派給 **column**，即變成 column (3)
6. 發先皆無法融合，因此已找到所有 Prime Implicants
7. 建立 Prime Implicant Chart (**cmt**)

Minterm	0	1	16	17	128	343	512
Prime Implicant	{0, 128} {0, 512} {0, 1, 16, 17}	{0, 1, 16, 17}	{0, 1, 16, 17}	{0, 1, 16, 17}	{0, 128}	{343}	{512

8. Patrick’s Method

   為方便描述過程
   令 {0, 128} = P1, {0, 512} = P2, {0, 1, 16, 17} = P3, {343} = P4, {512} = P5, {1023} = P6

   pm = { { {P1}, {P2}, {P3} }, { {P3} }, { {P1} }, { {P4} }, { {P5} }, { {P6} } }

   第一次合併 ⇒ pm = { {P1}, {P2}, {P3} }, { {P3} }, { {P1} }, { {P4} }, { {P5, P6} } }

   第二次合併 ⇒ pm = { {P1}, {P2}, {P3} }, { {P3} }, { {P1} }, { {P4, P5, P6} } }

   第三次合併 ⇒ pm = { {P1}, {P2}, {P3} }, { {P3} }, { {P1, P4, P5, P6} } }

   第四次合併 ⇒ pm = { {P1}, {P2}, {P3} }, { {P1, P3, P4, P5, P6} } }

   第五次合併 ⇒ pm = { { {P1, P3, P4, P5, P6} }, { {P1, P2, P3, P4, P5, P6} } }

   無法再合併 ⇒ 找 pm 中 size 最小的元素 ⇒ {P1, P3, P4, P5, P6}

   因此 minimumSop 為 P1+P3+P4+P5+P6

   經過函數 gen_literal() 轉換為 literals 後加入 minimumSops_set

   再將 minimumSops_set 轉為 vector 的 minimumSops

9. 回傳答案

Testcase 2

```
Column 1
====================
v 00000000: 0
--------------------
v 00000010: 2
v 00001000: 8
--------------------
v 00000101: 5
v 00000110: 6
v 00001010: 10
v 00001100: 12
--------------------
v 00000111: 7
v 00001101: 13
v 00001110: 14
--------------------
v 00001111: 15
--------------------
d 11111111: 255
--------------------


====================
Column 2
====================
v 000000-0: 0, 2
v 0000-000: 0, 8
--------------------
v 00000-10: 2, 6
v 0000-010: 2, 10
v 000010-0: 8, 10
v 00001-00: 8, 12
--------------------
v 000001-1: 5, 7
v 0000-101: 5, 13
v 0000011-: 6, 7
v 0000-110: 6, 14
v 00001-10: 10, 14
v 0000110-: 12, 13
v 000011-0: 12, 14
--------------------
v 0000-111: 7, 15
v 000011-1: 13, 15
v 0000111-: 14, 15
--------------------


====================
Column 3
====================
  0000-0-0: 0, 2, 8, 10
x 0000-0-0: 0, 8, 2, 10
--------------------
  0000--10: 2, 6, 10, 14
x 0000--10: 2, 10, 6, 14
  00001--0: 8, 10, 12, 14
x 00001--0: 8, 12, 10, 14
--------------------
  0000-1-1: 5, 7, 13, 15
x 0000-1-1: 5, 13, 7, 15
  0000-11-: 6, 7, 14, 15
x 0000-11-: 6, 14, 7, 15
  000011--: 12, 13, 14, 15
x 000011--: 12, 14, 13, 15
--------------------
```

```
====================
Prime Implicant Chart
====================
                    |  0|  2|  5|  6|  7|  8| 10| 12| 13| 14| 15|
 -------------------+---+---+---+---+---+---+---+---+---+---+---+
a'b'c'd'f'h'        |  x|  x|   |   |   |  x|  x|   |   |   |   |
a'b'c'd'gh'         |   |  x|   |  x|   |   |  x|   |   |  x|   |
a'b'c'd'eh'         |   |   |   |   |   |  x|  x|  x|   |  x|   |
a'b'c'd'fh          |   |   |  x|   |  x|   |   |   |  x|   |  x|
a'b'c'd'fg          |   |   |   |  x|  x|   |   |   |   |  x|  x|
a'b'c'd'ef          |   |   |   |   |   |   |   |  x|  x|  x|  x|
 -------------------+---+---+---+---+---+---+---+---+---+---+---+

Primes:
a'b'c'd'f'h'
a'b'c'd'gh'
a'b'c'd'eh'
a'b'c'd'fh
a'b'c'd'fg
a'b'c'd'ef

minimumSops:
a'b'c'd'f'h' + a'b'c'd'eh' + a'b'c'd'fh + a'b'c'd'fg
a'b'c'd'f'h' + a'b'c'd'fh + a'b'c'd'ef + a'b'c'd'fg
a'b'c'd'gh' + a'b'c'd'f'h' + a'b'c'd'eh' + a'b'c'd'fh
a'b'c'd'gh' + a'b'c'd'f'h' + a'b'c'd'fh + a'b'c'd'ef
```

1. 將 **minterms** (is_dontcare=False) 和 **dontcares** (is_dontcare=True) 的每個元素轉為 Term，生成 **bin_str** 後加入 **column** (1)
2. 將 **columns** (1) 中：
   - Group 0 與 Group 1 比對，
     - 發現 “000000000”: 0 與 “000000010”: 2 只差一位
       - 將兩者 **is_checked** 設為 True
       - 建立新 **Term**，放入 **new_column[0]**：
         - **bin_str** = “00000000-0”
         - **minterm** = {0, 2}
         - 因為兩個皆非全為 dontcare，所以 **is_dontcare** = False
       - 加入 **avoid_duplicate**，以免之後造出重複的 **Term**
     - (下略)
   - Group 1 與 Group 2 比對，
     - 發現 “00000010”: 2 與 “00000110”: 6 只差一位
       - 將兩者 **is_checked** 設為 True
       - 建立新 **Term**，放入 **new_column[1]**：
         - **bin_str** = “00000-10”
         - **minterm** = {2, 6}
         - 因為兩個皆非全為 dontcare，所以 **is_dontcare** = False
       - 加入 **avoid_duplicate**，以免之後造出重複的 **Term**
     - (下略)
   - Group 2 與 Group 3 比對，
     - 發現 “00000101”: 5與 “00000111”: 7 只差一位
       - 將兩者 **is_checked** 設為 True
       - 建立新 **Term**，放入 **new_column[2]**：
         - **bin_str** = “000001-1”
         - **minterm** = {5, 7}
         - 因為兩個皆非全為 dontcare，所以 **is_dontcare** = False
       - 加入 **avoid_duplicate**，以免之後造出重複的 **Term**
     - (下略)
   - Group 3 與 Group 4 比對，
     - 發現 “00000111”: 7 與 “00001111”: 15 只差一位
       - 將兩者 **is_checked** 設為 True
       - 建立新 **Term**，放入 **new_column[3]**：
         - **bin_str** = “0000-111”
         - **minterm** = {7, 15}
         - 因為兩個皆非全為 dontcare，所以 **is_dontcare** = False
       - 加入 **avoid_duplicate**，以免之後造出重複的 **Term**
   - 因為 Group 5 是空的，所以 Group 4 不用與其比對
   - 因為 Group 9 是空的，所以 Group 8 不用與其比對
3. 將 new_column 指派給 column，即變成 column (2)
   - 再將 new_column 清空

4. 將 **columns** (2) 中：
    - Group 0 與 Group 1 比對，
        - 發現 “000000-0”: 0, 2 與 “000010-0: 8, 10 只差一位
            - 將兩者 **is_checked** 設為 True
            - 建立新 **Term**，放入 **new_column[0]**：
                - **bin_str** = “0000-0-0”
                - **minterm** = {0, 2, 8, 10}
                - 因為兩個皆非全為 dontcare，所以 **is_dontcare** = False
            - 加入 **avoid_duplicate**，以免之後造出重複的 **Term**
        - (下略)
    - Group 1 與 Group 2 比對，
        - 發現 “0000-10”: 2, 6 與 “00001-10”: 10, 14 只差一位
            - 將兩者 **is_checked** 設為 True
            - 建立新 **Term**，放入 **new_column[1]**：
                - **bin_str** = “0000--10”
                - **minterm** = {2, 6, 10, 14}
                - 因為兩個皆非全為 dontcare，所以 **is_dontcare** = False
            - 加入 **avoid_duplicate**，以免之後造出重複的 **Term**
        - (下略)
    - Group 2 與 Group 3 比對，
        - 發現 “000001-1”: 5, 7與 “000011-1”: 13, 15 只差一位
            - 將兩者 **is_checked** 設為 True
            - 建立新 **Term**，放入 **new_column[2]**：
                - **bin_str** = “0000-1-1”
                - **minterm** = {5, 7, 13, 15}
                - 因為兩個皆非全為 dontcare，所以 **is_dontcare** = False
            - 加入 **avoid_duplicate**，以免之後造出重複的 **Term**
        - (下略)
    - 因為 Group 4 是空的，所以 Group 3 不用與其比對
5. 將 **new_column** 指派給 **column**，即變成 column (3)
    - 再將 new_column 清空
6. 發先皆無法融合，因此已找到所有 Prime Implicants
7. 建立 Prime Implicant Chart (**cmt**) (過多，不詳列出)
8. Patrick’s Method (過多，不詳列出)
    - 最終 pm[0] 中 size 最小的項有 4 個，因此 minimumSops 有四個
9. 回傳答案