TABULAR METHOD

2020-1 논리회로설계

Tabular Method 실행_기본 예제

$$f(x1,\cdots,x4) = \Sigma(m(0,4,8,10,11,12)+d(13,15))$$

```
Number of Input Variable:
```

```
Number of Input Variable: 4
Number of Minterm, Don't care: 6 2
Minterm: 0 4 8 10 11 12
Don't Care: 13 15
# of 1s        minterm         binary
      0             0           0000
      1             4           0100
      1             8           1000
      2            10           1010
      2            12           1100
      3            11           1011
      3            13           1101
      4            15           1111
-----------------------------------------
# of 1s        minterm         binary
      0           0,4           0-00
      0           0,8           -000
      1          4,12           -100
      1          8,10           10-0
      1          8,12           1-00
      2         10,11           101-
      2         12,13           110-
      3         11,15           1-11
      3         13,15           11-1
-----------------------------------------
# of 1s        minterm         binary
      0       0,4,8,12           --00
-----------------------------------------
```

```
     PI   0   4   8  10  11  12
  10-0            V   V
  101-                V   V
  1-11                    V
  110-                        V
  11-1
  --00   V   V   V            V
---------------------------------
 >> Complete Column Dominance
     PI   0   4   8  10  11  12
  10-0            O   V
  101-                V   V
  1-11                    V
  110-                        O
  11-1
  --00   O   O   O            O
---------------------------------
 >> Complete Row Dominance
     PI   0   4   8  10  11  12
  10-0            O   O
  101-                O   O
  1-11                    O
  110-                        O
  11-1
  --00   O   O   O            O
---------------------------------
 >> Complete Petrick's Method
     PI   0   4   8  10  11  12
  10-0            O   O
  101-                O   O
  1-11                    O
  110-                        O
  11-1
  --00   O   O   O            O
```

```
 >> Final Solution:
 f(X0,X1,X2,X3) = X2'X3' + X0X1'X2
```

Tabular Method 실행_Petrick's Method

f(x1,⋯,x4) = Σ(m(0,2,3,4,5,6,7,8,9,10,11,12,13))

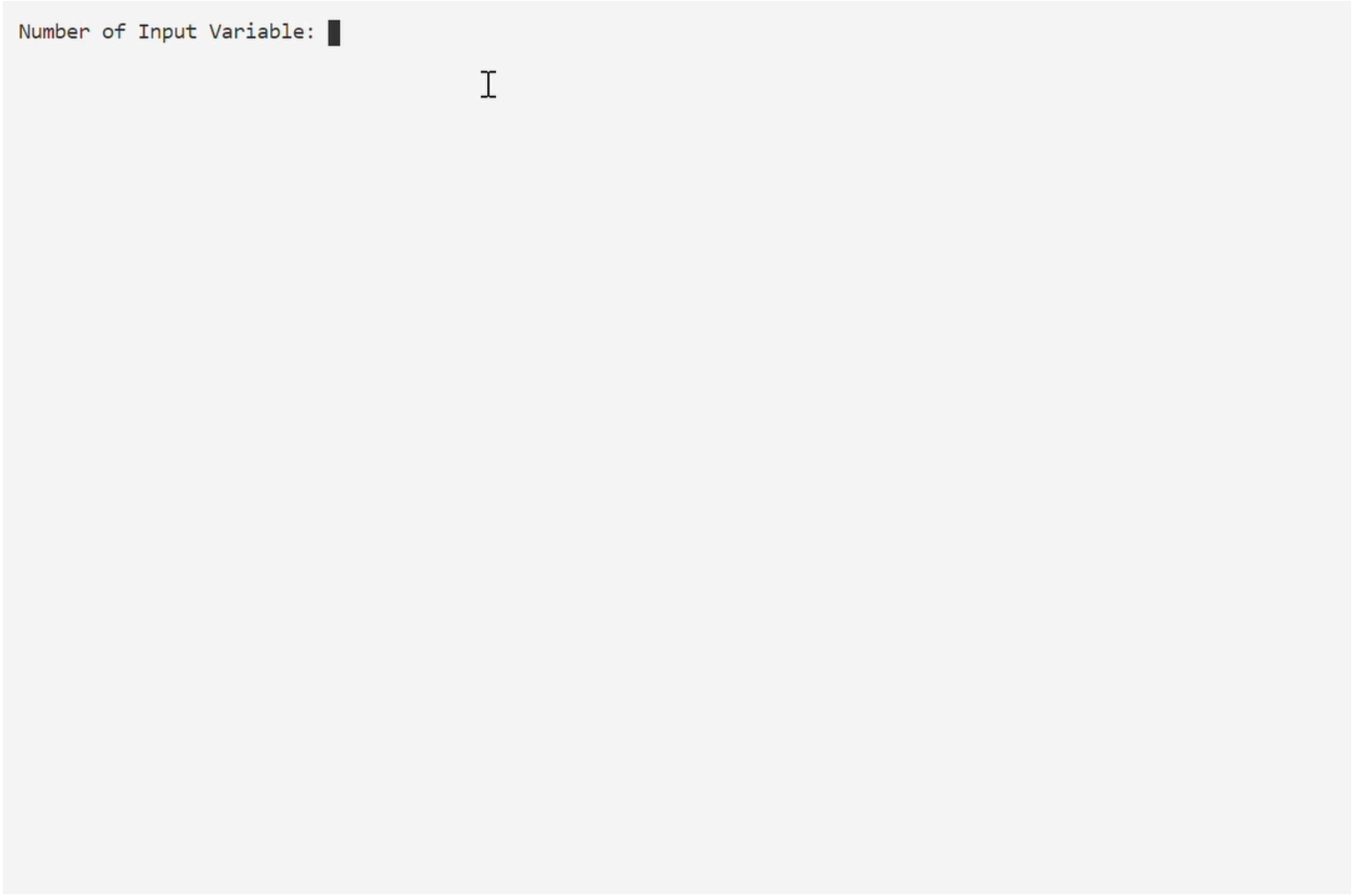
```
Number of Input Variable:
```

```
------------------------------------------------------------------------
  >> Complete Row Dominance
      PI   0   2   3   4   5   6   7   8   9  10  11  12  13
    0--0   V   V       V       V
    -0-0   V   V                           V       V
    --00   V           V                   V               V
    0-1-       V   V           V   V
    -01-       V   V                               V   V
    01--               V   V   V   V
    -10-               V   V                                   V   V
    10--                                   V   V   V   V
    1-0-                                   V   V           V   V
------------------------------------------------------------------------
------------------------------------------------------------------------
  >> Complete Petrick's Method
      PI   0   2   3   4   5   6   7   8   9  10  11  12  13
    0--0   O   O       O       O
    -0-0   O   O                           O       O
    --00   O           O                   O               O
    0-1-       O   O           O   O
    -01-       O   O                               O   O
    01--               O   O   O   O
    -10-               O   O                                   O   O
    10--                                   O   O   O   O
    1-0-                                   O   O           O   O
------------------------------------------------------------------------
  >> Final Solution:
  f(X0,X1,X2,X3) = X0'X3' + X0X1' + X1X2' + X0'X2
```

```
------------------------------------
# of 1s        minterm        binary
                                00-0
                                0-1-
                                -01-
                                01--
                                -10-
                                10--
                                1-0-
------------------------------------
    1      2,3,10,11           --00
    1       4,5,6,7            0-1-
    1      4,5,12,13           -01-
    1      8,9,10,11           01--
    1      8,9,12,13           -10-
                               10--
                               1-0-
```

Tabular Method 실행_Input Variables = 6

f(x1,⋯,x6) = Σ(m(4,12,19,23,31,37,45,49,53,61)+d(7,28,59))



Tabular Method 실행_Input Variables = 6

f(x1,⋯,x6) = Σ(m(4,12,19,23,31,37,45,49,53,61)+d(7,28,59))

```
Number of Input Variable: 6
Number of Minterm, Don't care: 10 3
Minterm: 4 12 19 23 31 37 45 49 53 61
Don't Care: 7 28 59
# of 1s        minterm         binary
      1              4         000100
      2             12         001100
      3             19         010011
      3             37         100101
      3             49         110001
      3              7         000111
      3             28         011100
      4             23         010111
      4             45         101101
      4             53         110101
      5             31         011111
      5             61         111101
      5             59         111011
-----------------------------------------
# of 1s        minterm         binary
      1            4,12         00-100
      2           12,28         0-1100
      3           19,23         010-11
      3            23,7         0-0111
      3           37,45         10-101
      3           37,53         1-0101
      3           49,53         110-01
      4           23,31         01-111
      4           45,61         1-1101
      4           53,61         11-101
-----------------------------------------
# of 1s        minterm         binary
      3    37,45,53,61         1--101
-----------------------------------------
```

```
       PI   4  12  19  23  31  37  45  49  53  61
   00-100   V   V
   0-1100       V
   010-11           V   V
   01-111               V   V
   0-0111               V
   110-01                               V   V
   1--101                       V   V       V   V
-----------------------------------------------------
   >> Complete Column Dominance
       PI   4  12  19  23  31  37  45  49  53  61
   00-100   O   O
   0-1100       O
   010-11           O   O
   01-111               O   O
   0-0111               O
   110-01                               O   O
   1--101                       O   O       O   O
-----------------------------------------------------
```

```
   >> Complete Row Dominance
       PI   4  12  19  23  31  37  45  49  53  61
   00-100   O   O
   0-1100       O
   010-11           O   O
   01-111               O   O
   0-0111               O
   110-01                               O   O
   1--101                       O   O       O   O
-----------------------------------------------------
   >> Complete Petrick's Method
       PI   4  12  19  23  31  37  45  49  53  61
   00-100   O   O
   0-1100       O
   010-11           O   O
   01-111               O   O
   0-0111               O
   110-01                               O   O
   1--101                       O   O       O   O
-----------------------------------------------------
```

```
>> Final Solution:
f(X0,X1,X2,X3,X4,X5) = X0'X1'X3X4'X5' + X0'X1X2'X4X5 + X0'X1X3X4X5 + X0X3X4'X5 + X0X1X2'X4'X5
```

Main_1

```
1   #include <iostream>
2   #include <algorithm>
3   #include <iomanip>
4   #include <vector>
5   #include <limits.h>
6   using namespace std;
```

```
23  int main(){
24      int InputVariable,nmin,ndc;
25      cout << "Number of Input Variable: ";
26      cin >> InputVariable;
27
28      cout << "Number of Minterm, Don't care: ";
29      cin >> nmin >> ndc;
30      int minterm[nmin];
31      int dontcare[ndc];
32
33      if (nmin > 0){
34          cout << "Minterm: ";
35          for (int i=0;i<nmin;i++)cin >> minterm[i];
36      }
37
38      if (ndc > 0){
39          cout  << "Don't Care: ";
40          for (int i=0;i<ndc;i++) cin >> dontcare[i];
41      }
```

#변수

InputVariable : 변수의 개수

nmin : minterm의 개수

ndc : don't care의 개수

minterm[nmin] : minterm 배열

don't care[ndc] : don't care 배열

Main_2

```
43      sort(minterm,minterm+nmin);
44      sort(dontcare,dontcare+ndc);
45
46      string PIArr[nmin+ndc][4];
47      string numOfOne, binary;
48
49      for (int i=0;i<nmin;i++){
50          binary = Binary(InputVariable,minterm[i]);
51          PIArr[i][0] = Count(binary); //# of 1s
52          PIArr[i][1] = to_string(minterm[i]); //Minterm
53          PIArr[i][2] = binary; //Binary
54          PIArr[i][3] = ""; //Combined
55      }
56      for (int i=nmin;i<nmin+ndc;i++){
57          binary = Binary(InputVariable,dontcare[i-nmin]);
58          PIArr[i][0] = Count(binary); // # of 1s
59          PIArr[i][1] = to_string(dontcare[i-nmin]); //Minterm(Dontcare)
60          PIArr[i][2] = binary; //Binary
61          PIArr[i][3] = ""; //Combined
62      }
```

```
15      string Binary(int a,int b); //minterm, dontcare -> 이진수 변환
16      string Count(string n); //변환한 이진수에서의 1의 수 (# of 1s)
```

PIArr[][4]

# of 1s	Minterm	Binary	Combined
1	4	0100	v
	8	1000	v
2	10	1010	v
	12	1100	v
3	11	1011	v
	13	1101	v
4	15	1111	v

String Binary(int a, int b);

: minterm, don't care를 이진수로 변환

```
73  // a=이진수자릿수(=InputVariable), b=이진수로 바꿀 수
74  string Binary(int a,int b){
75      char bin[a];
76      string binary = "";
77      for (int i=0;i<a;i++) bin[i] = '0';
78      int idx=0;
79      while (b>=1){
80          bin[idx++] = '0'+ b%2;
81          b/=2;
82      }
83      for (int j=a-1;j>=0;j--) binary += bin[j];
84      return binary;
85  }
```

string Count(string n);

: 이진수에서의 1의 개수(# of 1s)

```
87  //n = 이진수(=binary)
88  string Count(string n){
89      int cnt = 0;
90      for (int i=0;i<n.length();i++){
91          if (n[i] == '1') cnt++;
92      }
93      return to_string(cnt);
94  }
```

Main_3

```
64      showPI(InputVariable, nmin+ndc, PIArr); //정리한 PI table 출력
65      HammingDistance(nmin+ndc,InputVariable,PIArr); //EPI/NEPI 찾기
66
67      string solution = Dominance(nmin,minterm); //column/row dominance table
68      cout << "  >> Final Solution:" << '\n';
69      cout << "  " << showSolution(InputVariable, solution); // Final Solution 출력
70      return 0;
71  }
```

void showPI(int a, int b, string arr[][4]);

: arr배열을 표로 구성하여 보여줌

```
96   // a = # of 1s의 최대값(=InputVariable), b = arr의 row수, arr = 출력하고자 하는 arr
97   void showPI(int a, int b, string arr[][4]){
98       cout << setw(7) << "# of 1s";
99       cout << setw(15) << "minterm";
100      cout << setw(15) << "binary" << '\n';
101
102      for (int j=0;j<=a;j++){
103          for (int i=0;i<b;i++){
104              if (arr[i][0] == to_string(j)){
105                  cout << setw(7) << arr[i][0];
106                  cout << setw(15) << arr[i][1];
107                  cout << setw(15) << arr[i][2] << '\n';
108              }
109          }
110      }
111      cout << "----------------------------------------"<< '\n';
112  }
```

void showPI(int a, int b, string arr[][4]);

>> 출력

```
Number of Input Variable: 4
Number of Minterm, Don't care: 6 2
Minterm: 0 4 8 10 11 12
Don't Care: 13 15
# of 1s         minterm          binary
      0               0            0000
      1               4            0100
      1               8            1000
      2              10            1010
      2              12            1100
      3              11            1011
      3              13            1101
      4              15            1111
---------------------------------------
```

void HammingDistance(int a, int b, string arr[][4]);

: EPI와 NEPI를 찾기 위해 HD=1인 minterm, don't care 그룹화

```
132   //a=nmin+ndc(=arr의 길이), b=inputvariable
133   void HammingDistance(int a,int b,string arr[100][4]){
134       vector<int> diff;
135       string EPI[100][4];
136       int idx=0;
137       for (int i=0;i<a-1;i++){
138           for (int j=i+1;j<a;j++){
139               diff.clear();
140               if (abs(atoi(arr[i][0].c_str()) - atoi(arr[j][0].c_str())) == 1){
141                   int cnt = 0;
142                   for (int k=0;k<b;k++){
143                       if(arr[i][2][k] != arr[j][2][k]){
144                           if (arr[i][2][k] != '-' && arr[j][2][k] != '-') diff.push_back(k);
145                       }
146                       else cnt++;
147                   }
```

# of 1s의 차이가 **1**일 때

이진수 각각의 숫자가 다르면, 다른 위치를 diff에 저장

같다면, cnt+1

void HammingDistance(int a, int b, string arr[][4]);

```
149                 if (cnt == b-1){
150                     string binary = "";
151                     for (int k=0;k<b;k++){
152                         if (k == diff.front()) binary+='-';
153                         else binary += arr[i][2][k];
154                     }
155                     bool state = true;
156                     for (int k=0;k<idx;k++){
157                         if (EPI[k][2] == binary){
158                             state = false;
159                             //group과정에서 중복된 minterm에도 Combined에 V표시
160                             arr[i][3] = "V";
161                             arr[j][3] = "V";
162                         }
163                     }
164                     if (state){
165                         EPI[idx][0] = Count(binary);
166                         EPI[idx][1] = arr[i][1] +","+arr[j][1];
167                         EPI[idx][2] = binary;
168                         EPI[idx++][3] = "";
169 
170                         arr[i][3] = "V";
171                         arr[j][3] = "V";
172                     }
173                 }
174             }
175         }
176     }
```

이진수 각각의 숫자 중 다른 숫자가 하나일 때,

Binary값 재구성 : 다른 숫자 위치에는 '-', 나머지 그대로

EPI에 이미 같은 minterm, binary가 있다면,

arr[][3] (Combined)에 check

(EPI에 추가하지 않음)

EPI[][4] (새로운 PIArr) 추가,

기존 arr[][3] (Combined)에 check

(이 때, idx의 값을 통해 EPI의 길이를 알 수 있음)

void HammingDistance(int a, int b, string arr[][4]);

```
177     if (idx != 0) showPI(b,idx,EPI);
178
179     int vcnt = 0;
180     for (int i=0;i<a;i++){
181         if (arr[i][3] == "V") vcnt++;
182     }
183     if (vcnt != 0 && idx>1) HammingDistance(idx,b,EPI);
184     else EPIN
185 }
```

Size 1

1s	Minterm	Binary	
0	0	0 0 0 0	v
1	4	0 1 0 0	v
	8	1 0 0 0	v
2	10	1 0 1 0	v
	12	1 1 0 0	v
3	11	1 0 1 1	v
	13	1 1 0 1	v
4	15	1 1 1 1	v

Size 2

1s	Minterm	Binary	
0	0,4	0 - 0 0	v
	0,8	- 0 0 0	v
	8,10	1 0 - 0	
1	4,12	- 1 0 0	v
	8,12	1 - 0 0	v
2	10,11	1 0 1 -	
	12,13	1 1 0 -	
3	11,15	1 - 1 1	
	13,15	1 1 - 1	

Size 4

1s	Minterm	Binary
0	0,4,8,12	- - 0 0

Gather all PIs not checked with “v”.

>> 출력

```
# of 1s         minterm          binary
      0               0            0000
      1               4            0100
      1               8            1000
      2              10            1010
      2              12            1100
      3              11            1011
      3              13            1101
      4              15            1111
-----------------------------------------
# of 1s         minterm          binary
      0             0,4            0-00
      0             0,8            -000
      1            4,12            -100
      1            8,10            10-0
      1            8,12            1-00
      2           10,11            101-
      2           12,13            110-
      3           11,15            1-11
      3           13,15            11-1
-----------------------------------------
# of 1s         minterm          binary
      0         0,4,8,12            --00
-----------------------------------------
```

void EPINEPI(int idx1, int idx2, string arr1[][4], string arr1[][4]);

: 찾은 EPI와 NEPI 벡터에 넣기

```
 8   vector<vector<string>> vecEPI; //전체 EPI
 9   vector<string> vecEPI1; //vecEPI[0], EPI의 minterm
10   vector<string> vecEPI2; //vecEPI[1], EPI의 binary
11   vector<vector<string>> vecNEPI; //전체 NEPI
12   vector<string> vecNEPI1; //NEPI의 minterm
13   vector<string> vecNEPI2; //NEPI의 binary
```

```
114  //idx1 = arr의 row 수, idx2 = EPI의 수(=EPI 배열의 row 수), arr = EPI의 직전 배열, arr2 = EPI배열
115  void EPINEPI(int idx1, int idx2, string arr1[][4], string arr2[][4]){
116      for (int i=0;i<idx2;i++){
117          vecEPI1.push_back(arr2[i][1]);
118          vecEPI2.push_back(arr2[i][2]);
119      }
120      for (int i=0;i<idx1;i++){
121          if (arr1[i][3] == "") {
122              vecNEPI1.push_back(arr1[i][1]);
123              vecNEPI2.push_back(arr1[i][2]);
124          }
125      }
126      vecEPI.push_back(vecEPI1);
127      vecEPI.push_back(vecEPI2);
128      vecNEPI.push_back(vecNEPI1);
129      vecNEPI.push_back(vecNEPI2);
130  }
```

EPI를 벡터에 정리

- vecEPI1 : EPI의 minterm
- vecEPI2 : EPI의 binary

NEPI를 벡터에 정리

- vecNEPI1 : NEPI의 minterm
- vecNEPI2 : NEPI의 binary

Main_3

```
64      showPI(InputVariable, nmin+ndc, PIArr); //정리한 PI table 출력
65      HammingDistance(nmin+ndc,InputVariable,PIArr); //EPI/NEPI 찾기
66
67      string solution = Dominance(nmin,minterm); //column/row dominance table
68      cout << "  >> Final Solution:" << '\n';
69      cout << "  " << showSolution(InputVariable, solution); // Final Solution 출력
70      return 0;
71  }
```

string Dominance(int a, int arr[]);

: Column Dominance와 Row Dominance, Petrick's Method 구현

```
186   //a = arr 길이, arr[] = minterm;
187   string Dominance(int a, int arr[]){
188       string solution = "";
189       //vecNEPI[0] = [8,10   10,11   11,15   12,13   13,15]
190       //vecNEPI[1] = [10-0   101-    1-11    110-    11-1]
191       int row = vecNEPI[0].size()+vecEPI[0].size();
192       string table[row][a+1];
193       for (int i=0;i<row;i++){
194           for (int j=0; j<a+1; j++) table[i][j] = " ";
195       }
```

string table[row][a+1]

: Column/row Dominance를 하기 위한 기본 표

string Dominance(int a, int arr[]);

```
//vecNEPI[0] = [8,10   10,11   11,15   12,13   13,15]
//vecNEPI[1] = [10-0   101-    1-11    110-    11-1]
```

```
196     for (int i=0;i<vecNEPI[0].size();i++){
197         table[i][0] = vecNEPI[1][i];
198         istringstream aa(vecNEPI[0][i]);
199         string stringBuffer;
200         int k = 0;
201         string num[100];
202         while(getline(aa,stringBuffer,',')){
203             num[k++] = stringBuffer;
204         }
205         for (int l=0;l<k;l++){
206             for (int j=0; j<a; j++){
207                 if (num[l] == to_string(arr[j])) table[i][j+1] = "V";
208             }
209         }
210     }
```

```
211     for (int i=vecNEPI[0].size();i<row;i++){
212         table[i][0] = vecEPI[1][i-vecNEPI[0].size()];
213         istringstream aa(vecEPI[0][i-vecNEPI[0].size()]);
214         string stringBuffer;
215         int k = 0;
216         string num[100];
217         while(getline(aa,stringBuffer,',')){
218             num[k++] = stringBuffer;
219         }
220         for (int l=0;l<k;l++){
221             for (int j=0; j<a; j++){
222                 if (num[l] == to_string(arr[j])) table[i][j+1] = "V";
223             }
224         }
225     }
```

쉼표(,)를 기준으로 num배열에 분리하여 넣기

(이때, k = num배열의 길이)

해당 minterm에 check

string Dominance(int a, int arr[]);

```
227     //dominance 배열 출력
228     cout << setw(10) << "PI";
229     for (int i=0;i<a;i++) cout << setw(4) << arr[i];
230     cout << '\n';
231     for (int i=0;i<row;i++){
232         cout << setw(10) << table[i][0];
233         for (int j=1;j<a+1;j++){
234             cout << setw(4) << table[i][j];
235         }
236         cout << '\n';
237     }
238     for (int i=0;i<a+1;i++) cout << "-----";
239     cout << '\n';
```

>> 출력

```
-------------------------------------
        PI   0   4   8  10  11  12
     10-0            V   V
     101-                V   V
     1-11                    V
     110-                        V
     11-1
     --00   V   V   V            V
-------------------------------------
```

string Dominance(int a, int arr[]);

column Dominance

```
241     //column dominance
242     for (int j=1;j<a+1;j++){
243         int cnt = 0; //V의 수를 세어줌
244         int idx = 0; //유일한 V가 있는 위치 저장
245         for (int i=0;i<row;i++){
246             if (table[i][j] == "V" || table[i][j] == "O"){
247                 cnt++;
248                 idx = i;
249             }
250         }
251         if (cnt == 1 && table[idx][j]=="V"){
252             table[idx][j] = "O";
253             solution += table[idx][0]+"+";
254             for (int k=1;k<a+1;k++){
255                 if (table[idx][k] == "V") {
256                     table[idx][k] = "O";
257                     for (int l=0;l<row;l++){
258                         if (table[l][k] == "V") table[l][k] = "O";
259                     }
260                 }
261             }
262         }
263     }
```

PI	0	4	8	10	11	12
10-0			O	V		
101-				V	V	
1-11					V	
110-						O
11-1						
--00	O	O	O			O

string Dominance(int a, int arr[]);

column Dominance

```
264     cout << "   >> Complete Column Dominance" <<'\n';
265     cout << setw(10) << "PI";
266     for (int i=0;i<a;i++) cout << setw(4) << arr[i];
267     cout << '\n';
268
269     for (int i=0;i<row;i++){
270         cout << setw(10) << table[i][0];
271         for (int j=1;j<a+1;j++) cout << setw(4) << table[i][j];
272         cout << '\n';
273     }
274     for (int i=0;i<a+1;i++) cout << "-----";
275     cout << '\n';
```

>> 출력

```
-----------------------------------
   >> Complete Column Dominance
        PI   0   4   8  10  11  12
     10-0           O   V
     101-               V   V
     1-11               V
     110-                       O
     11-1
     --00   O   O   O           O
-----------------------------------
```

string Dominance(int a, int arr[]);

Row Dominance

```
277     // row dominance
278     while (true){
279         int include[row][row];
280         int includeSum[row];
281         for (int i=0;i<row;i++){
282             includeSum[i] = 0;
283             for (int j=0;j<row;j++) include[i][j] = 0;
284         }
285         bool state = false;
286         for (int i=0;i<row;i++){
287             for (int j=0;j<row;j++){
288                 if (i!=j){
289                     for (int k=1;k<a+1;k++){
290                         if (table[i][k] == "V" && table[i][k] == table[j][k]) state = true;
291                         else if (table[i][k] == "V" && table[i][k] != table[j][k]) {
292                             state = false;
293                             break;
294                         }
295                     }
296                     if (state) include[j][i] = 1;
297                     state = false;
298                 }
299             }
300         }
```

```
-----------------------------------
   >> Complete Column Dominance
       PI   0   4   8  10  11  12
     10-0           O   V
     101-               V   V
     1-11               V
     110-                       O
     11-1
     --00   O   O   O           O
-----------------------------------
```

Dominance관계 파악

Ex)P2(101-)이 (P1)10-0을 지배

string Dominance(int a, int arr[]);

Row Dominance

```
301         int max = INT_MIN;
302         int max_idx;
303         int totalSum = 0;
304         for (int i=0;i<row;i++){
305             for (int j=0;j<row;j++){
306                 includeSum[i] += include[i][j];
307             }
308             if (max < includeSum[i]){
309                 max = includeSum[i];
310                 max_idx = i;
311             }
312             totalSum += includeSum[i];
313         }
314         if (totalSum == 0) break;
315         solution += table[max_idx][0] + "+";
316         for (int i=1;i<a+1;i++){
317             if (table[max_idx][i] == "V"){
318                 for (int j=0;j<row;j++){
319                     if (table[j][i] == "V") table[j][i] = "O";
320                 }
321             }
322         }
323     }
```

PI	0	4	8	10	11	12
10-0			O	O		
101-				O	O	
1-11					O	
110-						O
11-1						
--00	O	O	O			O

string Dominance(int a, int arr[]);

Row Dominance

```
325     cout << "   >> Complete Row Dominance" <<'\n';
326     cout << setw(10) << "PI";
327     for (int i=0;i<a;i++) cout << setw(4) << arr[i];
328     cout << '\n';
329
330     for (int i=0;i<row;i++){
331         cout << setw(10) << table[i][0];
332         for (int j=1;j<a+1;j++) cout << setw(4) << table[i][j];
333         cout << '\n';
334     }
335     for (int i=0;i<a+1;i++) cout << "-----";
336     cout << '\n';
```

>> 출력

```
-----------------------------------
   >> Complete Row Dominance
         PI   0   4   8  10  11  12
      10-0            O   O
      101-                O   O
      1-11                    O
      110-                        O
      11-1
      --00   O   O   O            O
-----------------------------------
```

string Dominance(int a, int arr[]);

Petrick's Method

```
338     //Petrick's Method
339     while (true){
340         int rowSum[row];
341         int rowTotal = 0;
342         int max2_idx = 0;
343         int max2 = INT_MIN;
344         for (int i=0;i<row;i++) rowSum[i] = 0;
345         for (int i=0;i<row;i++){
346             for (int j=1;j<a+1;j++){
347                 if (table[i][j] == "V") rowSum[i]++;
348             }
349             rowTotal += rowSum[i];
350             if (max2 < rowSum[i]){
351                 max2 = rowSum[i];
352                 max2_idx = i;
353             }
354         }
355         if (rowTotal == 0) break;
356         for (int i=1;i<a+1;i++){
357             if (table[max2_idx][i] == "V"){
358                 for (int j=0;j<row;j++){
359                     if (table[j][i] == "V") table[j][i] = "O";
360                 }
361             }
362         }
363         solution += table[max2_idx][0] + "+";
364     }
```

✓ Check 개수가 가장 큰 row 위치 = max2_idx에 저장

✓ Check가 없다면 while문 break;

```
------------------------------------------------------------------------
  >> Complete Row Dominance
      PI   0   2   3   4   5   6   7   8   9  10  11  12  13
    0--0   O   O       O       O
    -0-0   O   O                               V       V
    --00   O           O                       V               V
    0-1-       O   V           O   V
    -01-       O   V                                   V   V
    01--               O   V   O   V
    -10-               O   V                                   V   V
    10--                                       V   V   V   V
    1-0-                                       V   V           V   V
------------------------------------------------------------------------
```

해당 PI를 solution에 추가

string Dominance(int a, int arr[]);

Petrick's Method

```
366     cout << "   >> Complete Petrick's Method" <<'\n';
367     cout << setw(10) << "PI";
368     for (int i=0;i<a;i++) cout << setw(4) << arr[i];
369     cout << '\n';
370
371     for (int i=0;i<row;i++){
372         cout << setw(10) << table[i][0];
373         for (int j=1;j<a+1;j++) cout << setw(4) << table[i][j
374         cout << '\n';
375     }
376     for (int i=0;i<a+1;i++) cout << "-----";
377     cout << '\n';
378
379     return solution;
380 }
```

>> 출력

```
------------------------------------------------------------------------
   >> Complete Row Dominance
       PI   0   2   3   4   5   6   7   8   9  10  11  12  13
     0--0   V   V       V       V
     -0-0   V   V                           V       V
     --00   V           V                   V               V
     0-1-       V   V           V   V
     -01-       V   V                               V   V
     01--               V   V   V   V
     -10-               V   V                                   V   V
     10--                                   V   V   V   V
     1-0-                                   V   V           V   V
------------------------------------------------------------------------
   >> Complete Petrick's Method
       PI   0   2   3   4   5   6   7   8   9  10  11  12  13
     0--0   O   O       O       O
     -0-0   O   O                           O       O
     --00   O           O                   O               O
     0-1-       O   O           O   O
     -01-       O   O                               O   O
     01--               O   O   O   O
     -10-               O   O                                   O   O
     10--                                   O   O   O   O
     1-0-                                   O   O           O   O
------------------------------------------------------------------------
```

string showSolution(int a, string str);

:Final Solution 표시

```
382  // a = InputVariable, str = PI
383  string showSolution(int a, string str){
384      //f(x0,x1,...) = 꼴로 나타내기
385      string FinalSolution = "f(";
386      for (int i=0;i<a-1;i++) FinalSolution += "X"+to_string(i)+",";
387      FinalSolution += "X"+to_string(a-1)+") = ";
388
389      //'+'를 기준으로 PI binary배열에 넣기
390      istringstream aa(str);
391      string stringBuffer;
392      int idx = 0;
393      string PI_binary[100];
394      while(getline(aa,stringBuffer,'+')){
395          PI_binary[idx++] = stringBuffer;
396      }
```

f(X1,X2,X3,X4)꼴로 나타내기

str(solution)을 기준으로 PI_binary 배열에 분리하여 넣기

(이때, idx = PI_binary배열의 길이)

string showSolution(int a, string str);

```
398     //solution을 보기 좋게 정리
399     for (int i=0;i<idx;i++){
400         for (int j=0;j<PI_binary[i].length();j++){
401             if (PI_binary[i][j] != '-') {
402                 FinalSolution += "X" + to_string(j);
403                 if (PI_binary[i][j] == '0') FinalSolution +=  "'";
404             }
405         }
406         if (i != idx-1) FinalSolution += " + ";
407     }
408     return FinalSolution;
409 }
```

>> 출력

```
>> Final Solution:
f(X0,X1,X2,X3) = X2'X3' + X0X1'X2
```

감사합니다

2020-1 논리회로설계 <Tabular Method>