Graph Isomorphism Impossibility

Ganesh Arun Masurkar

Introduction:

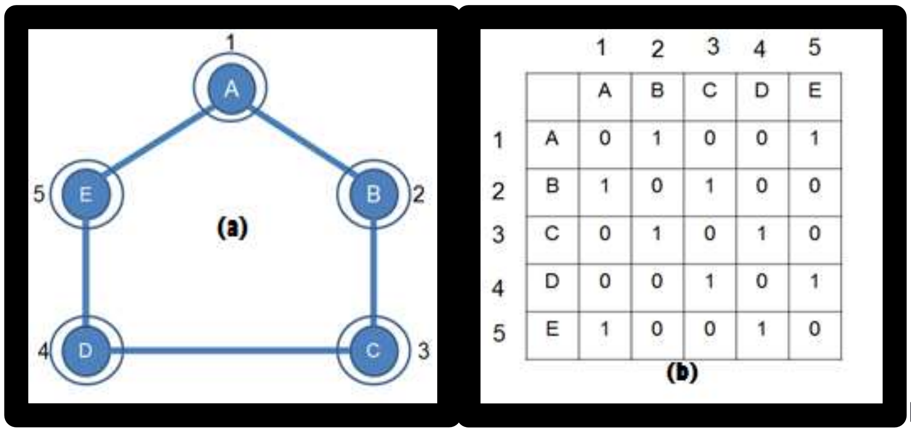
Graph isomorphism is the property where two graphs have same number of vertices connected in the same way. There are many classes of graphs with a polynomial time algorithm to find whether graphs are isomorphic or not. For my project, I will be considering *Simple Connected Un-directional* graphs. The most basic properties that two isomorphic graphs should have are; same number of vertices, same number of edges and same set of degrees (if a graph G have exactly one vertex with degree 4 then the other graph G' should also have exactly one vertex with degree 4). For my project, I will be comparing graphs that satisfies these basic conditions.

Background:

There are n! ways to represent a graph with n vertices [1]. An exhaustive approach is to find all the n! representation of a graph (G), if any of the n! graphs matches the other graph G' then G and G' are isomorphic. The corollary dictates, for graph G to be isomorphic to G' then G' should be one of the n! representation of G and vice versa. My approach is based on this corollary.

Approach and Proof:

Assume there are *n* slots, where each slot can only hold a vertex. Assign a distinct logical number to each slot at random while generating an adjacency matrix for a graph. If you have *n* distinct slots and *n* distinct vertices, then you can place the vertices in n! ways (basic knowledge of permutation can prove that). These n! configuration/representation is the number of ways to represent a graph.



		1	2	3	4	5
		A	B	C	D	E
1	A	0	1	0	0	1
2	B	1	0	1	0	0
3	C	0	1	0	1	0
4	D	0	0	1	0	1
5	E	1	0	0	1	0

Fig. 1.

Fig 1.a. is a graph with 5 vertices labeled from A to E and 5 slots numbered from 1 to 5 (at random). Fig 1.b. is the corresponding adjacency matrix with vertices in the first row and first column, the 1's and 0's represents the connectivity of a vertex as seen in the graph. The numbers 1 to 5 represent the slot number; vertex A is at row 1 since it occupies slot 1 in the graph, vertex B is at row 2 since it occupies slot 2 in the graph and so on. It is evident from the approach that if you change the slot number the adjacency matrix changes as well.

Since, we are representing a graph using an adjacency matrix many of these n! representations are duplicated more than once. Consider a cyclic graph were *n* vertices are connected in a cycle. There (n-1)! ways to connect *n* vertices in a cycle (basic knowledge of permutation can prove that). Out of these (n-1)! representations, $[\frac{(n-1)!}{2}]$ representations are inverse/opposite of the other $[\frac{(n-1)!}{2}]$ representations. So, there are only $[\frac{(n-1)!}{2}]$ distinct adjacency matrix for a cyclic graph with *n* vertices (distinct adjacency matrix means the inner connectivity part with 1's and 0's is distinct). For example, for a cyclic graph with 5 vertices there are only 12 distinct adjacency matrices. The corresponding 12 graph are shown in fig 2. For each of these 12 graphs, interchange vertex in slot 2 with vertex in slot 5 and interchange vertex in 3 with vertex in slot 4 and the resulting configurations will be one of the 12 configurations.

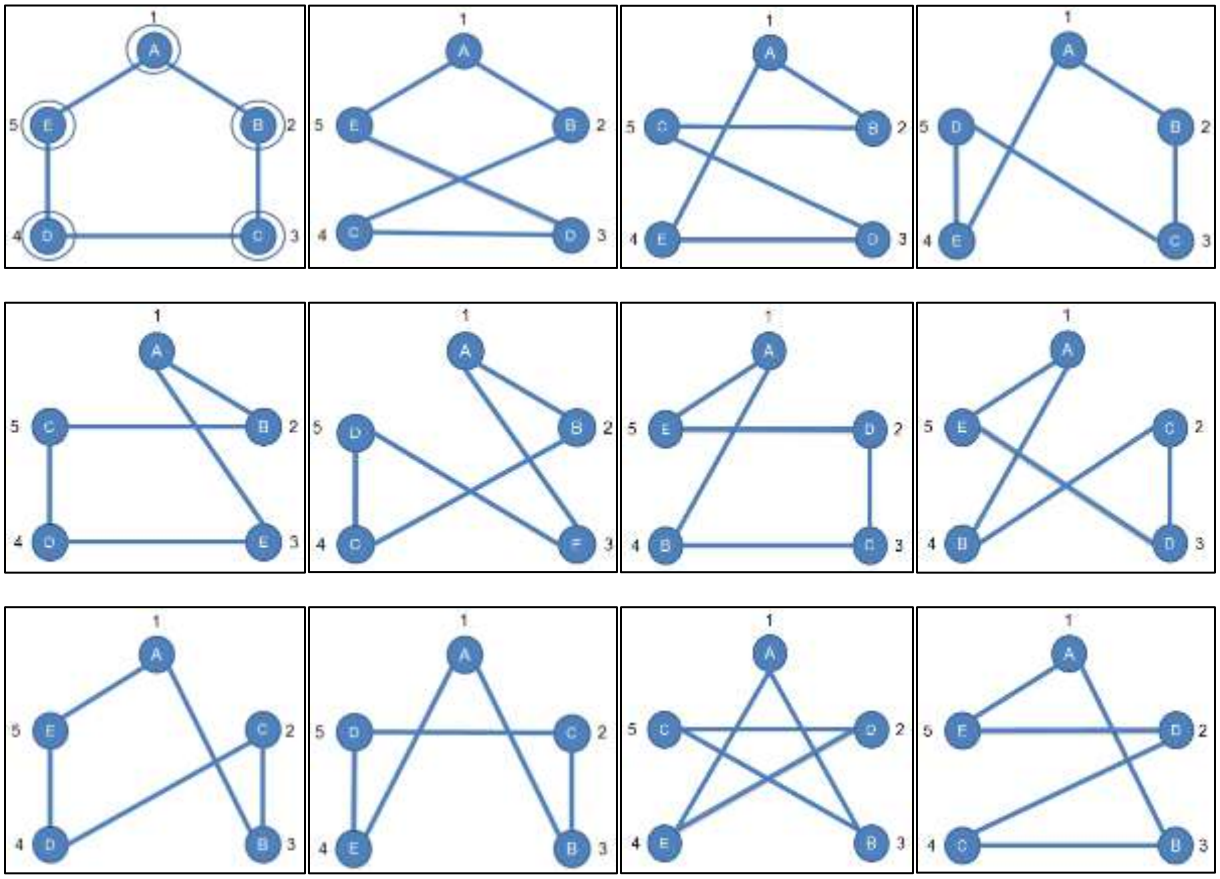


Fig. 2.

So, for cyclic graphs the complexity of the algorithm is reduced to number of distinct adjacency matrices for a graph i.e.: $[\frac{(n-1)!}{2}]$. Different graphs will have different number of distinct adjacency matrices (based on the permutation for that graph), so the complexity changes as well.

To move from one configuration to another we would have to perform a series of rotation/swapping of vertices. Rotation(x,y) refers to swapping the vertices i.e.: interchanging the slots x and y occupies. If we move a vertex to a slot, we would have to move some other vertex to the currently emptied slot (maybe the vertex that was evicted or some other vertex). For example, in fig 2 we can move from one configuration to another with at-least one rotation.

Pseudo-code for finding new distinct adjacency matrix:

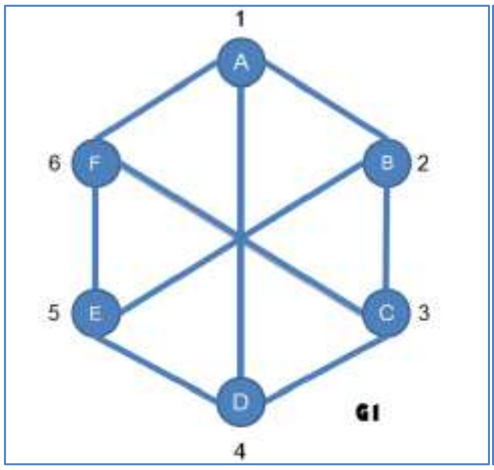
```
Initialize the two-adjacency matrix [reference matrix (RM) and combinational matrix (CM)]

List-of-config

for each row i from 1 to n

    if RM[i] does not match CM[i]

        //-- indexes range is 1 to n

        A:

        for j from i+1 to n

            find row CM[j] that closely matches row RM[i] then Rotation(i, j)

            //-- the row[j] in $M_C$ that has the least number of mismatches in 1's with
            //-- row[i] of $M_{R;}$

        end

        [x,y]=the two indexes in rows CM[i] and RM[i] that do not match

        Rotation(x,y)

        If configuration is present in List-of-config

            if i is NOT equal to n then

                Rotation(x,y) //-- to go back to the previous configuration

                i++ //-- to move to the next row

                go to A:
```

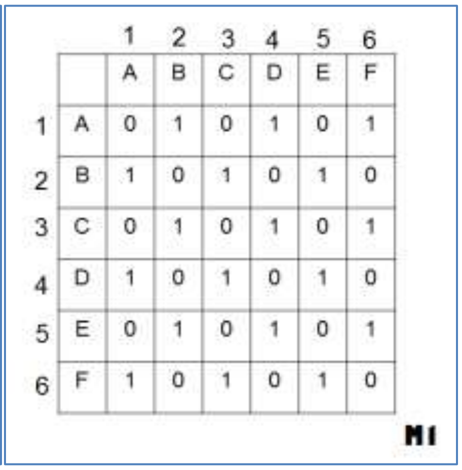
```
            else
                the graphs are not isomorphic then break
            end
        else
            add the new config to List-of-config
        end
        i=1
    end
    if i is equal to n
        the graphs are isomorphic;
        first column CM[0] is the mapping
    end
end
```

Invariant:

Based on the representations for n! permutations, the mismatches (the mismatch in ones for each row in the two matrix) should be rectified/reversed/tallied and the matches should be retained/conserved.

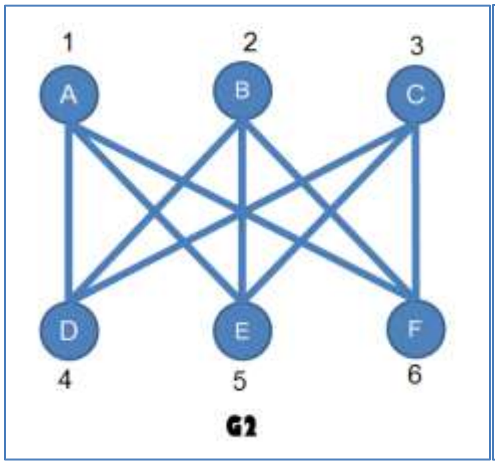
Example 1: Consider two representations of a $K_{3,3}$ graph as below.

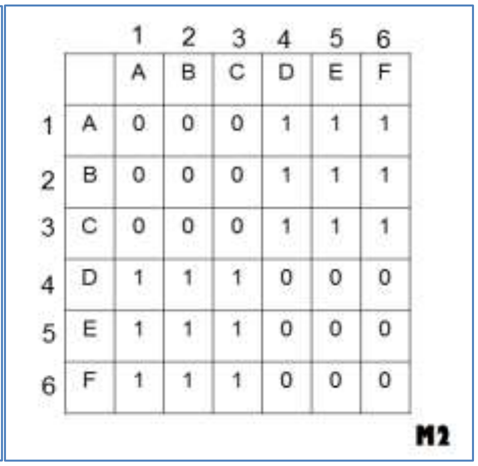




		1	2	3	4	5	6
		A	B	C	D	E	F
1	A	0	1	0	1	0	1
2	B	1	0	1	0	1	0
3	C	0	1	0	1	0	1
4	D	1	0	1	0	1	0
5	E	0	1	0	1	0	1
6	F	1	0	1	0	1	0

M1

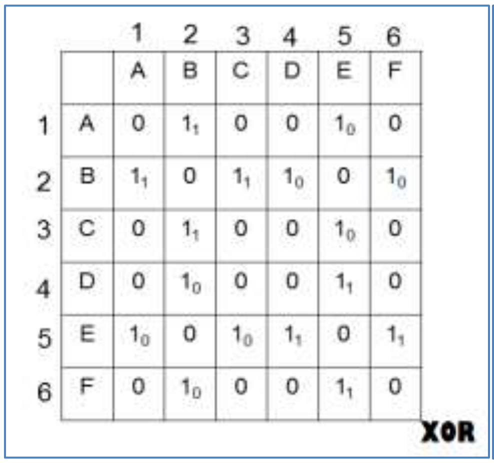




		1	2	3	4	5	6
		A	B	C	D	E	F
1	A	0	0	0	1	1	1
2	B	0	0	0	1	1	1
3	C	0	0	0	1	1	1
4	D	1	1	1	0	0	0
5	E	1	1	1	0	0	0
6	F	1	1	1	0	0	0

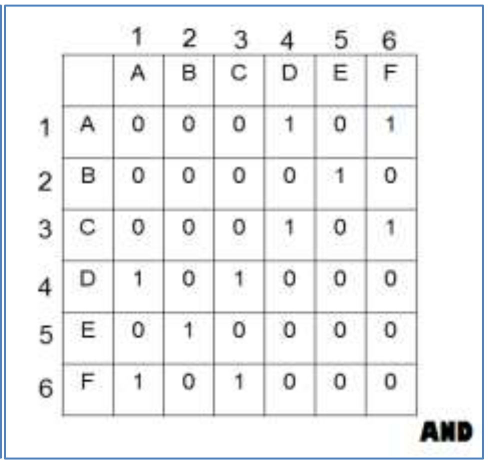
M2

To find the invariant find the XOR (I_{MM}) and AND (I_M) element-wise of M1 and M2. The I_{MM} represents the mismatches of M1 and M2 and I_M represents the mismatches of M1 and M2. The XOR (I_{MM}) and AND (I_M) is as follows:



		1	2	3	4	5	6
		A	B	C	D	E	F
1	A	0	1_1	0	0	1_0	0
2	B	1_1	0	1_1	1_0	0	1_0
3	C	0	1_1	0	0	1_0	0
4	D	0	1_0	0	0	1_1	0
5	E	1_0	0	1_0	1_1	0	1_1
6	F	0	1_0	0	0	1_1	0

XOR



		1	2	3	4	5	6
		A	B	C	D	E	F
1	A	0	0	0	1	0	1
2	B	0	0	0	0	1	0
3	C	0	0	0	1	0	1
4	D	1	0	1	0	0	0
5	E	0	1	0	0	0	0
6	F	1	0	1	0	0	0

AND

First, we define a computational matrix (CM) and a Reference Matrix (RM) at random. In this example, M1 is computational matrix and M2 is Reference Matrix. 1_1 in I_{MM} represents that there is a 1 in CM and 0 in RM and 1_0 in I_{MM} represents that there is a 0 in CM and 1 in RM. Intuitively, all 1_0 should be replaced/reverted by 1_1 so that M1 transform to M2.

Now, for each row in I_{MM} find the mismatch and rotate/swap those indexes. When swapping x and y, value in cell(x,x) should be equal to cell(y,y) (both 0 or both 1 when comparing rows in I_M and both 0 or one 1_0 and the other 1_1 when comparing rows in I_{MM}) and value in cell (x,y) should be equal to cell(y,x). All the rest of the cells in row x should be equal to cells in row y and the cells in column x should be equal to cells in column y. This comparison should be done for both I_{MM} and I_M.

Possible cases when comparing cells in I_{MM} and I_M:

When 1_0 is compared with 1_0. No changes in I_{MM}.

When 1_0 is compared with 1_1. The mismatches are reverted, so both position can be switched to 0.

When 1_0 is compared with 0. If there is a 1 in I_M at the same position as that of 0 in I_{MM} then the value at position of 1_0 is switched to 0 in I_{MM}, value at position of 0 is switched to 1_0 in I_{MM}, value at position of 1 is switched to 0 in I_M and value at position of 1_0 is switched to 1 in I_M. If there is a 0 in I_M at the same position as that of 0 in I_{MM} then no changes in I_{MM}.

When 1_1 is compared with 1_1. No changes in I_{MM}.

When 1_1 is compared with 0. If there is a 0 in I_M at the same position as that of 0 in I_{MM} then the value at position of 1_1 is switched to 0 in I_{MM}, value at position of 0 is switched to 1_1 in I_{MM}, value at position of 0 is switched to 0 in I_M and value at position of 1_1 is switched to 1 in I_M. If there is a 1 in I_M at the same position as that of 0 in I_{MM} then no changes in I_{MM}.

For slot/row 1:

There is mismatch at index $2(1_1)$ and $5(1_0)$, so rotate 2 and 5.

In I_{MM} : (2,2) = (5,5) = 0 and (2,5) = (5,2) = 0. The rest of the cells in row and column 2 (1,3,4,6) is also equal to cells in row 5 (1,3,4,6) i.e.: $(1_{1,}1_{1,}1_{0,}1_0)$ and $(1_{0,}1_{0,}1_{1,}1_1)$ respectively. So, values in rows 1, 2 and 5 and columns 1,2 and 5 are turned to 0. In other words, rotating 2 and 5 transforms rows 1, 2 and 5 of M1 to that of M2.

In I_M : (2,2) = (5,5) = 0 and (2,5) = (5,2) = 1. The rest of the cells in row and column 2 (1,3,4,6) is also equal to cells in row 5 (1,3,4,6) i.e.: (0,0,0,0). So, after rotating 2 and 5 no other mismatches are added all the ones are retained and some new matches are added which results in M2.

For slot/row 2:

There are mismatches at index 1,3,4 and 6. But, those mismatches were reverted when we rotated 2 and 5 for row 1.

For slot/row 3:

There is mismatch at index 2 and 5. But, those mismatches were reverted when we rotated 2 and 5 for row 1.

For slot/row 4:

There is mismatch at index 2 and 5. But, those mismatches were reverted when we rotated 2 and 5 for row 1.

For slot/row 5:

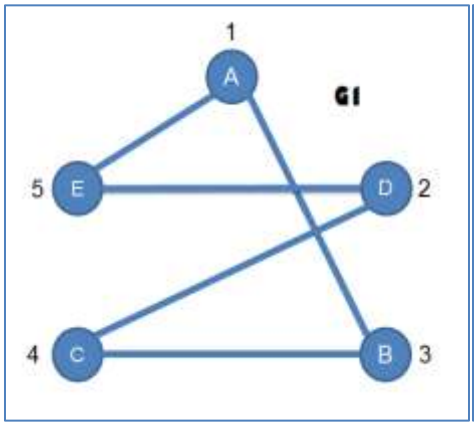
There are mismatches at index 1,3,4 and 6. But, those mismatches were reverted when we rotated 2 and 5 for row 1.

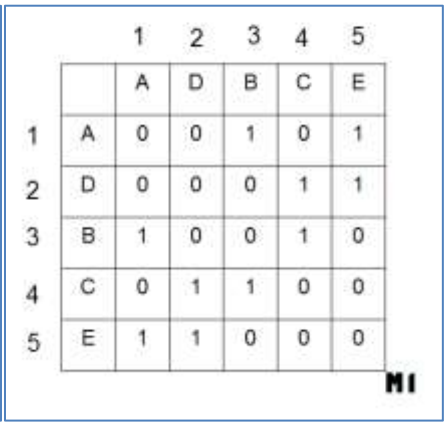
<u>For slot/row 6:</u>

There is mismatch at index 2 and 5. But, those mismatches were reverted when we rotated 2 and 5 for row 1.

After rotating 2 and 5, M1 is transformed to M2. M1 is now equal to M2, which proves that the two graphs are isomorphic.

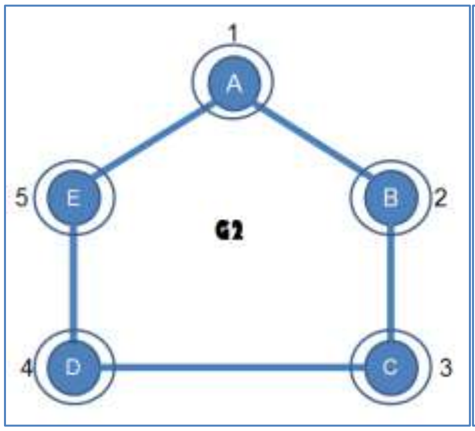
<u>Example 2</u>: Consider two representations of a cyclic graph with 5 vertices as below.

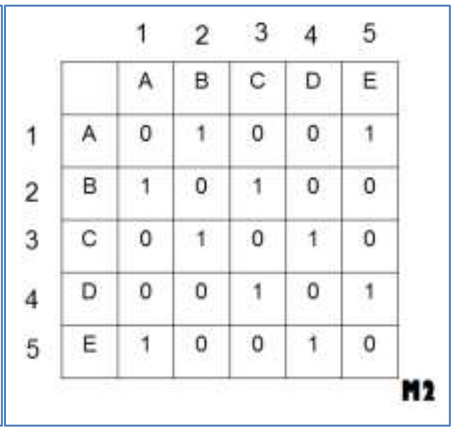




		1	2	3	4	5
		A	D	B	C	E
1	A	0	0	1	0	1
2	D	0	0	0	1	1
3	B	1	0	0	1	0
4	C	0	1	1	0	0
5	E	1	1	0	0	0

M1

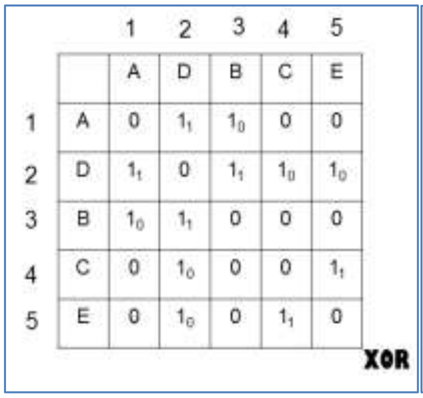




		1	2	3	4	5
		A	B	C	D	E
1	A	0	1	0	0	1
2	B	1	0	1	0	0
3	C	0	1	0	1	0
4	D	0	0	1	0	1
5	E	1	0	0	1	0

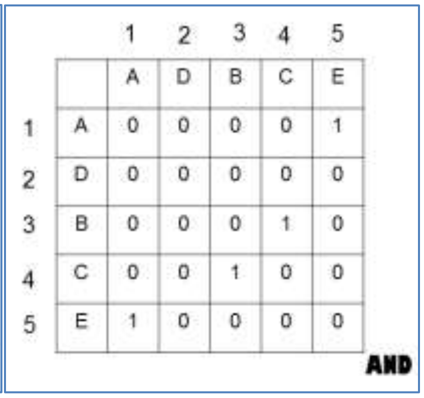
M2

The XOR (I_{MM}) and AND (I_M) element-wise of M1 and M2 is as follows.



		1	2	3	4	5
		A	D	B	C	E
1	A	0	1_1	1_0	0	0
2	D	1_1	0	1_1	1_0	1_0
3	B	1_0	1_1	0	0	0
4	C	0	1_0	0	0	1_1
5	E	0	1_0	0	1_1	0

XOR



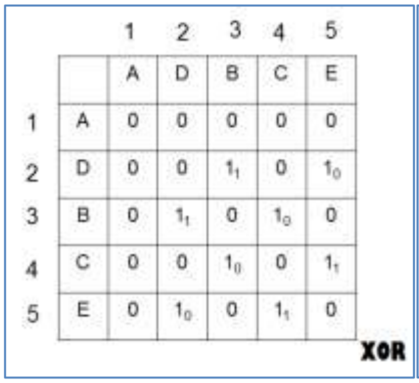
		1	2	3	4	5
		A	D	B	C	E
1	A	0	0	0	0	1
2	D	0	0	0	0	0
3	B	0	0	0	1	0
4	C	0	0	1	0	0
5	E	1	0	0	0	0

AND

For slot/row 1:

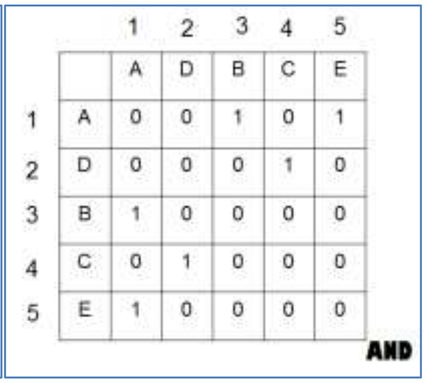
There is mismatch at index 2(1_1) and 3(1_0), so rotate 2 and 3.

After comparing row and column 2 and 3 in I_{MM} and I_M based on the possible cases as described in Example 1 the new/updated I_{MM} and I_M are as follows.



		1	2	3	4	5
		A	D	B	C	E
1	A	0	0	0	0	0
2	D	0	0	1_1	0	1_0
3	B	0	1_1	0	1_0	0
4	C	0	0	1_0	0	1_1
5	E	0	1_0	0	1_1	0

XOR



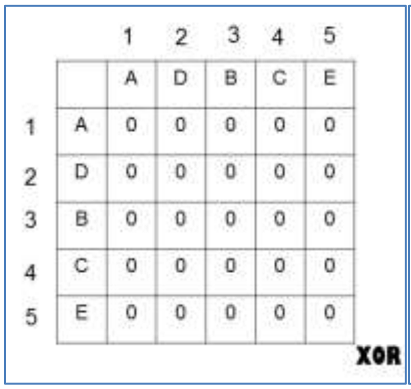
		1	2	3	4	5
		A	D	B	C	E
1	A	0	0	1	0	1
2	D	0	0	0	1	0
3	B	1	0	0	0	0
4	C	0	1	0	0	0
5	E	1	0	0	0	0

AND

For slot/row 2:

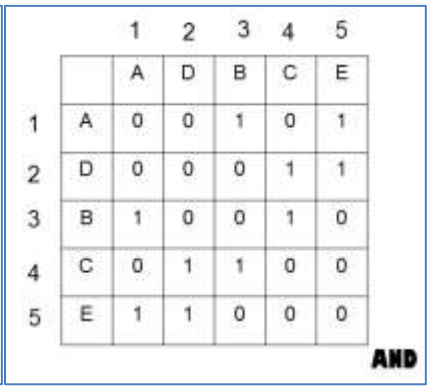
There is mismatch at index 3(1_1) and 5(1_0), so rotate 3 and 5.

After comparing row and column 3 and 5 in I_{MM} and I_M based on the possible cases as described in Example 1 the new/updated I_{MM} and I_M are as follows.



		1	2	3	4	5
		A	D	B	C	E
1	A	0	0	0	0	0
2	D	0	0	0	0	0
3	B	0	0	0	0	0
4	C	0	0	0	0	0
5	E	0	0	0	0	0

XOR

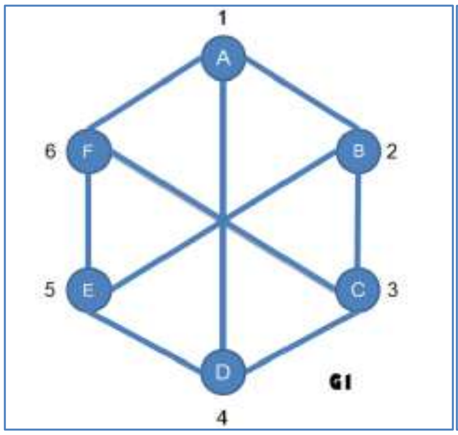


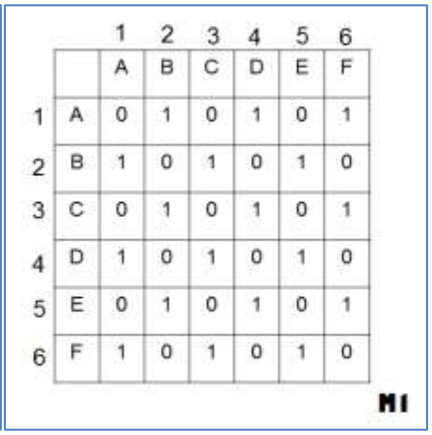
		1	2	3	4	5
		A	D	B	C	E
1	A	0	0	1	0	1
2	D	0	0	0	1	1
3	B	1	0	0	1	0
4	C	0	1	1	0	0
5	E	1	1	0	0	0

AND

There are no more mismatches and I_M is equal to M2. So, the graphs are Isomorphic.

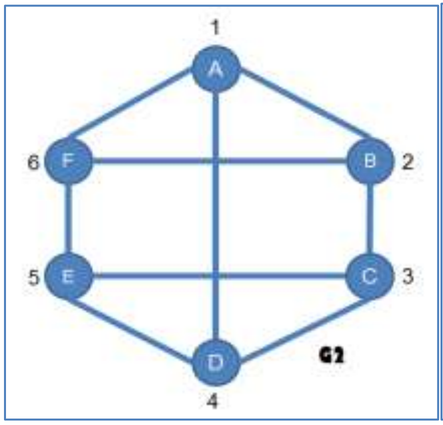
Example 3: Consider two graphs $K_{3,3}$ and another non-isomorphic graph as below.

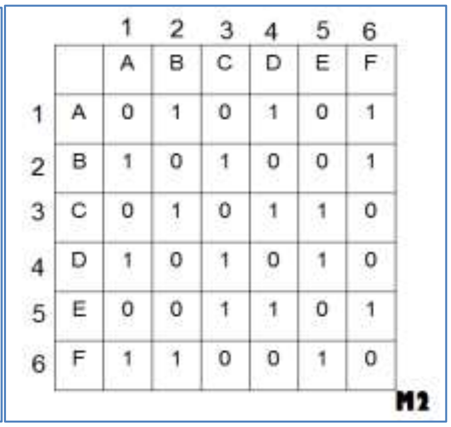




		1	2	3	4	5	6
		A	B	C	D	E	F
1	A	0	1	0	1	0	1
2	B	1	0	1	0	1	0
3	C	0	1	0	1	0	1
4	D	1	0	1	0	1	0
5	E	0	1	0	1	0	1
6	F	1	0	1	0	1	0

M1

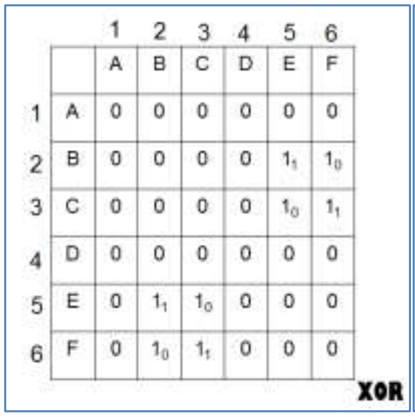




		1	2	3	4	5	6
		A	B	C	D	E	F
1	A	0	1	0	1	0	1
2	B	1	0	1	0	0	1
3	C	0	1	0	1	1	0
4	D	1	0	1	0	1	0
5	E	0	0	1	1	0	1
6	F	1	1	0	0	1	0

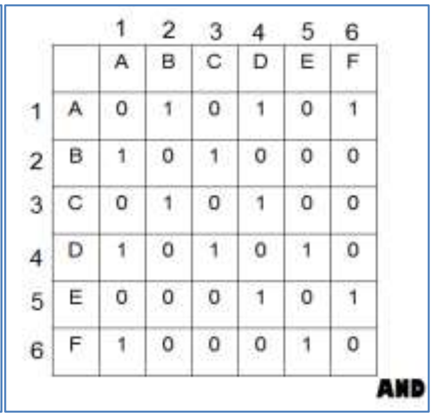
M2

The XOR (I_{MM}) and AND (I_M) element-wise of M1 and M2 is as follows.



		1	2	3	4	5	6
		A	B	C	D	E	F
1	A	0	0	0	0	0	0
2	B	0	0	0	0	1_1	1_0
3	C	0	0	0	0	1_0	1_1
4	D	0	0	0	0	0	0
5	E	0	1_1	1_0	0	0	0
6	F	0	1_0	1_1	0	0	0

XOR



		1	2	3	4	5	6
		A	B	C	D	E	F
1	A	0	1	0	1	0	1
2	B	1	0	1	0	0	0
3	C	0	1	0	1	0	0
4	D	1	0	1	0	1	0
5	E	0	0	0	1	0	1
6	F	1	0	0	0	1	0

AND

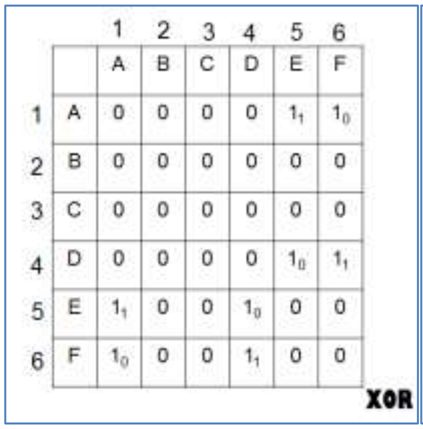
For slot/row 1:

No mismatches.

For slot/row 2:

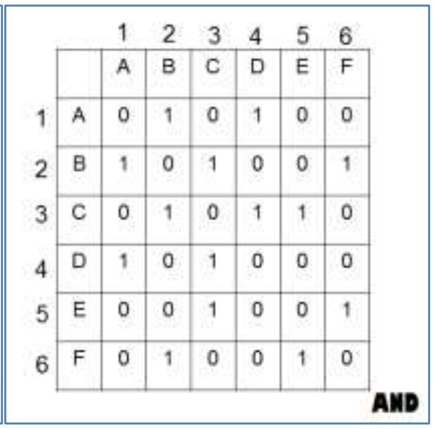
There is mismatch at index 5(1_1) and 6(1_0), so rotate 5 and 6.

After comparing row and column 5 and 6 in I_{MM} and I_M based on the possible cases as described in Example 1 the new/updated I_{MM} and I_M are as follows.



		1	2	3	4	5	6
		A	B	C	D	E	F
1	A	0	0	0	0	1_1	1_0
2	B	0	0	0	0	0	0
3	C	0	0	0	0	0	0
4	D	0	0	0	0	1_0	1_1
5	E	1_1	0	0	1_0	0	0
6	F	1_0	0	0	1_1	0	0

XOR



		1	2	3	4	5	6
		A	B	C	D	E	F
1	A	0	1	0	1	0	0
2	B	1	0	1	0	0	1
3	C	0	1	0	1	1	0
4	D	1	0	1	0	0	0
5	E	0	0	1	0	0	1
6	F	0	1	0	0	1	0

AND

<u>For slot/row 1:</u>

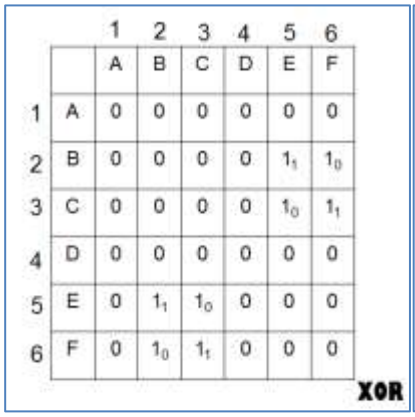
There is mismatch at index 5(1_1) and 6(1_0). But rotating 5 and 6 would lead to the previous configuration.

<u>For slot/row 5:</u>

There is mismatch at index 1(1_1) and 4(1_0). But rotating 1 and 4 and rotating 5 and 6 are interchangeable i.e.: if we rotate 5 and 6 the mismatch at index 1 and 4 is reverted as well, they are the same. So, the graphs are not Isomorphic.

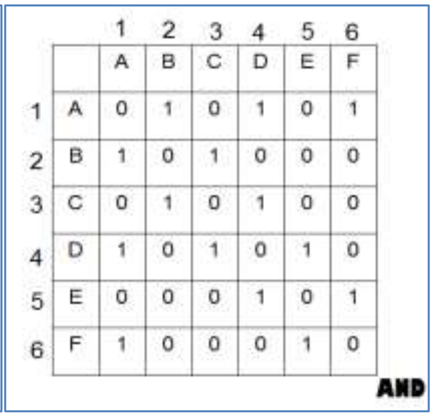
Proof that rotating 1 and 4 and rotating 5 and 6 are interchangeable:

Case 1: If we rotate 5 and 6, we get back to the initial configuration of I_{MM} and I_M.



		1	2	3	4	5	6
		A	B	C	D	E	F
1	A	0	0	0	0	0	0
2	B	0	0	0	0	1_1	1_0
3	C	0	0	0	0	1_0	1_1
4	D	0	0	0	0	0	0
5	E	0	1_1	1_0	0	0	0
6	F	0	1_0	1_1	0	0	0

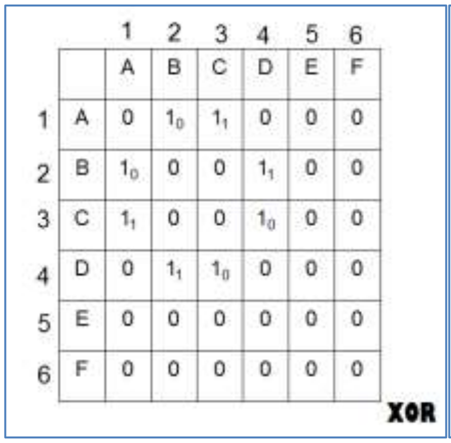
XOR



		1	2	3	4	5	6
		A	B	C	D	E	F
1	A	0	1	0	1	0	1
2	B	1	0	1	0	0	0
3	C	0	1	0	1	0	0
4	D	1	0	1	0	1	0
5	E	0	0	0	1	0	1
6	F	1	0	0	0	1	0

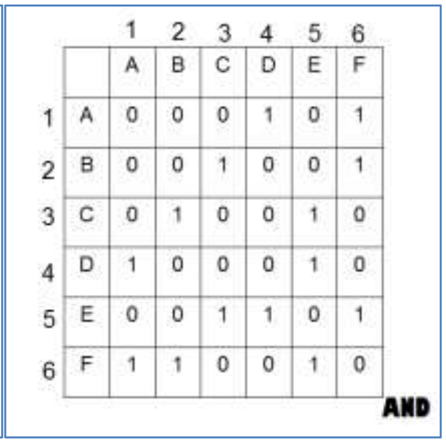
AND

Case 2: After rotating 1 and 4, I_{MM} and I_M are as follows.



		1	2	3	4	5	6
		A	B	C	D	E	F
1	A	0	1_0	1_1	0	0	0
2	B	1_0	0	0	1_1	0	0
3	C	1_1	0	0	1_0	0	0
4	D	0	1_1	1_0	0	0	0
5	E	0	0	0	0	0	0
6	F	0	0	0	0	0	0

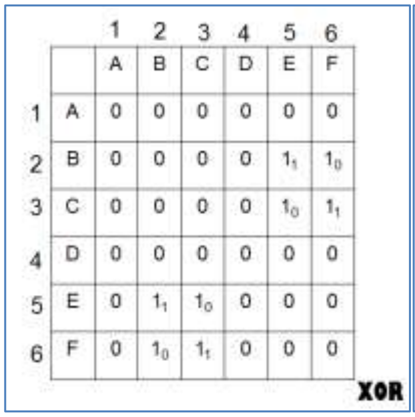
XOR



		1	2	3	4	5	6
		A	B	C	D	E	F
1	A	0	0	0	1	0	1
2	B	0	0	1	0	0	1
3	C	0	1	0	0	1	0
4	D	1	0	0	0	1	0
5	E	0	0	1	1	0	1
6	F	1	1	0	0	1	0

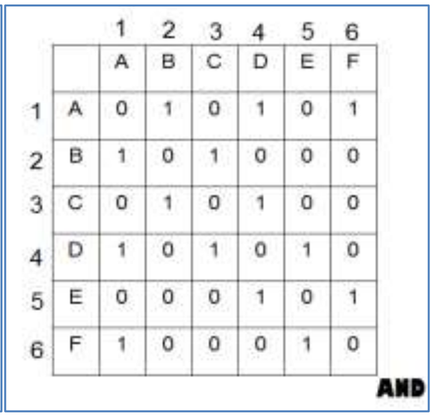
AND

From this configuration, we cannot rotate 1 and 4 again because it will lead to the previous configuration. So, the only other choice is rotating 2 and 3. After rotating 2 and 3, I_{MM} and I_M are as follows.



		1	2	3	4	5	6
		A	B	C	D	E	F
1	A	0	0	0	0	0	0
2	B	0	0	0	0	1_1	1_0
3	C	0	0	0	0	1_0	1_1
4	D	0	0	0	0	0	0
5	E	0	1_1	1_0	0	0	0
6	F	0	1_0	1_1	0	0	0

XOR



		1	2	3	4	5	6
		A	B	C	D	E	F
1	A	0	1	0	1	0	1
2	B	1	0	1	0	0	0
3	C	0	1	0	1	0	0
4	D	1	0	1	0	1	0
5	E	0	0	0	1	0	1
6	F	1	0	0	0	1	0

AND

This is the initial configuration of I_{MM} and I_M. Irrespective of the rotation (either 1 and 4 or 5 and 6) we end up in the same configuration. Hence, rotating 1 and 4 and rotating 5 and 6 are interchangeable.

Complexity:

The complexity of this algorithm is dependent on the number of distinct adjacency matrices for a graph. Different graphs will have different number of distinct adjacency matrices; the number depends on the topology/connectivity of the network. Assume, a graph has α distinct adjacency matrices. The complexity of the algorithm will be O(α).

References:

[1] Kocay, William and Kreher, Donald L, Graphs, algorithms, and optimization, CRC Press, 2016.