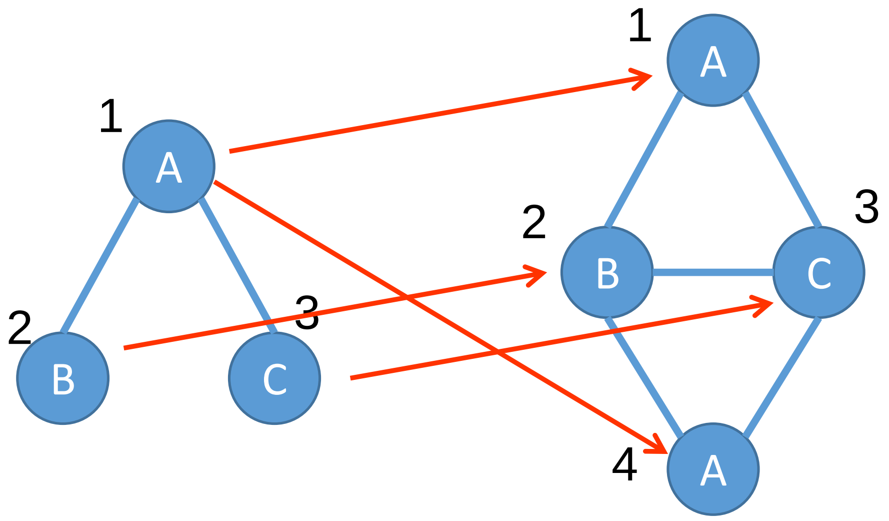
Subgraph isomorphism

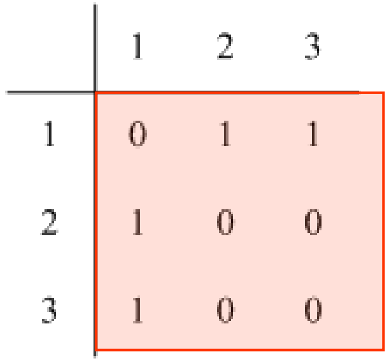
Given two graphs Q=(V(Q),E(Q), L_v, F) and G=(V(G),E(G), L_v', F′) , we say Q is subgraph isomorphism to G, if and only if there exists a function g: V(Q)→ V(G), such that

$$\forall v \in V(Q), F(v) = F'(g(v)); and$$

$$\forall v_1, v_2 \in V(Q), \overrightarrow{v_1 v_2} \in E(Q) \Rightarrow \overrightarrow{g(v_1) g(v_2)} \in E(G)$$

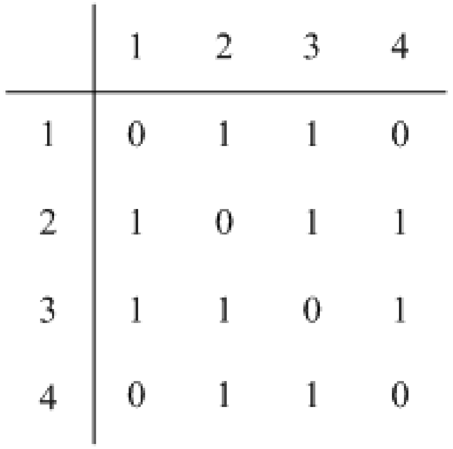


Example



	1	2	3
1	0	1	1
2	1	0	0
3	1	0	0

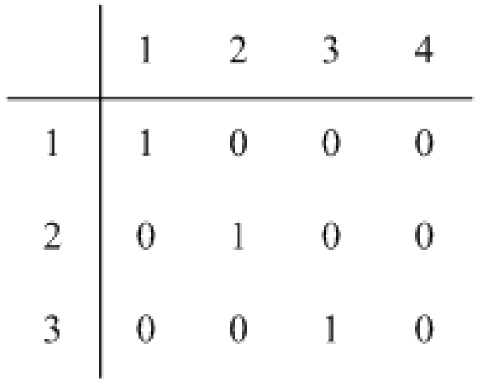
MA



	1	2	3	4
1	0	1	1	0
2	1	0	1	1
3	1	1	0	1
4	0	1	1	0

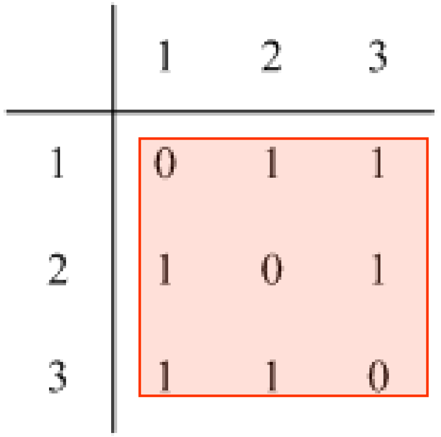
MB

$$MC = M'(M' \bullet MB)^T$$



	1	2	3	4
1	1	0	0	0
2	0	1	0	0
3	0	0	1	0

M’

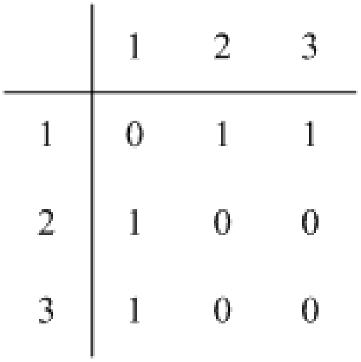


	1	2	3
1	0	1	1
2	1	0	1
3	1	1	0

$MC = M'(M' \bullet MB)^T$

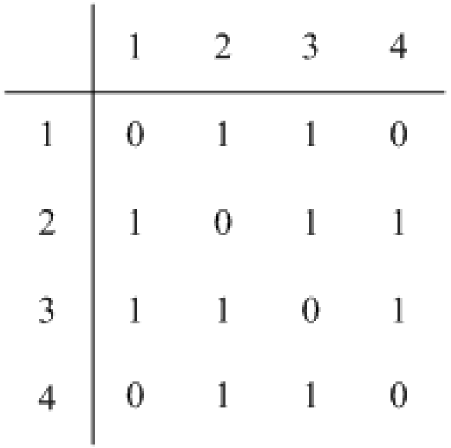
$$\forall i \forall j : (MA[i][j] = 1) \Rightarrow (MC[i][j] = 1)$$

Example



	1	2	3
1	0	1	1
2	1	0	0
3	1	0	0

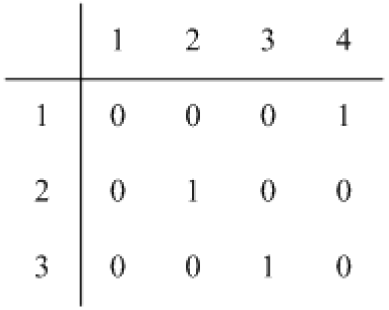
MA



	1	2	3	4
1	0	1	1	0
2	1	0	1	1
3	1	1	0	1
4	0	1	1	0

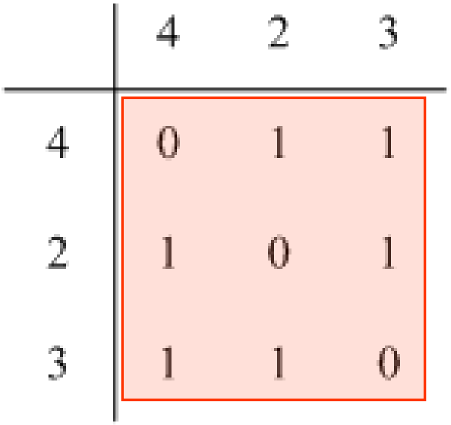
MB

$$MC = M'(M' \bullet MB)^T$$



	1	2	3	4
1	0	0	0	1
2	0	1	0	0
3	0	0	1	0

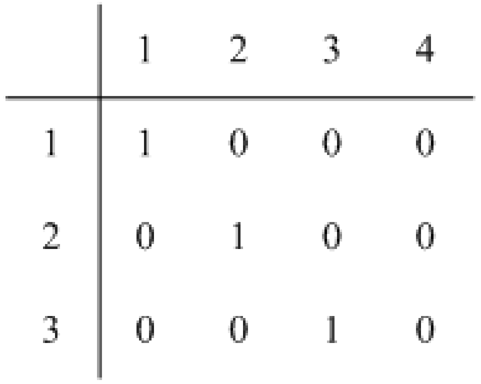
M'



	4	2	3
4	0	1	1
2	1	0	1
3	1	1	0

$$MC = M'(M' \bullet MB)^T$$

$$\forall i \forall j : (MA[i][j] = 1)$$
$$\Rightarrow (MC[i][j] = 1)$$

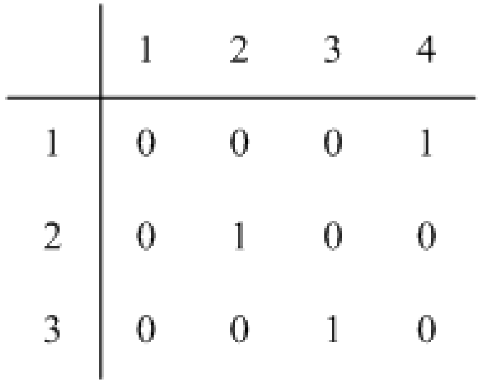


	1	2	3	4
1	1	0	0	0
2	0	1	0	0
3	0	0	1	0

M'

1) M'[i][j] = 1 means that the i-th vertex in Q corresponds to j-th vertex in query G;
2) Each row in M' contains exactly one 1;
3) No column contains more than one 1.

M' specifies an subgraph isomorphism from Q to G.



	1	2	3	4
1	0	0	0	1
2	0	1	0	0
3	0	0	1	0

M'

How to find such matrix M' ?
--- Ullmann Algorithm@76

Ullmann Algorithm (@1976)

- Given two graphs Q and G, their corresponding matrixes are $MA_{n \times n}=[a_{ij}]$ and $MB_{m \times m} = [b_{ij}]$.
- Goal: 1) Find matrix $M'_{n \times m}$ such that

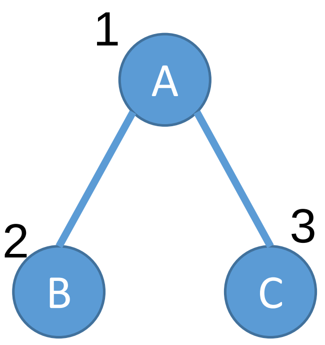
$$MC = M'(M' \bullet MB)^T$$

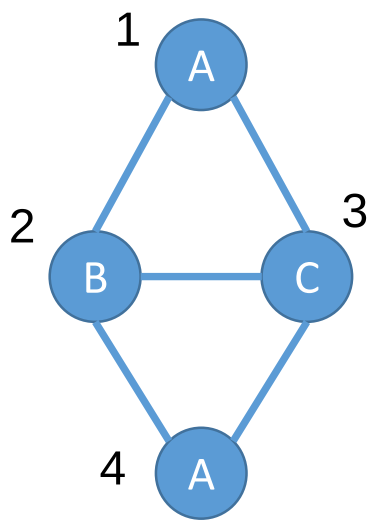
$$\forall i \forall j : (MA[i][j] = 1) \Rightarrow (MC[i][j] = 1)$$

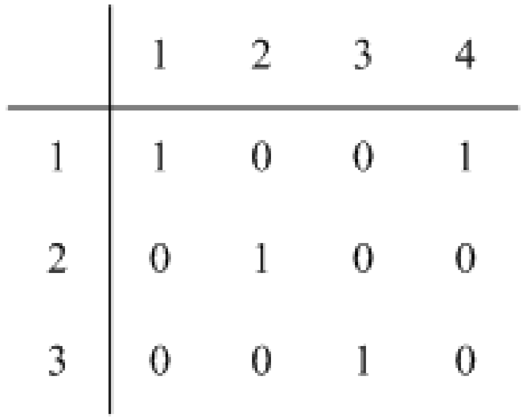
- 2) or report no such marix M'.

Basic Idea

Step 1. Set up matrix $M_{n \times m}$, such that M[i][j]=1, if 1) the i-th vertex in Q has the same label as the j-th vertex in G; and 2) the i-th vertex has smaller vertex degree than the j-th vertex in G.







	1	2	3	4
1	1	0	0	1
2	0	1	0	0
3	0	0	1	0

M

- Step 2. Matrixes M’ are generated by systematically changing to 0 all but one of the 1’s in each of the rows of M, subject to the definitory condition that no column of a matrix M’ may contain more than one 1. (the maximal depth is |MA|).

$$\begin{tikzcd}
\begin{matrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{matrix} \arrow[r] & \begin{matrix} \color{red}1 & \color{red}0 & \color{red}0 & \color{green}0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{matrix} \arrow[d] \\
\begin{matrix} \color{red}1 & \color{red}0 & \color{red}0 & \color{green}0 \\ \color{red}0 & \color{red}1 & \color{red}0 & \color{red}0 \\ \color{red}0 & \color{red}0 & \color{red}1 & \color{red}0 \end{matrix} & \begin{matrix} \color{red}1 & \color{red}0 & \color{red}0 & \color{green}0 \\ \color{red}0 & \color{red}1 & \color{red}0 & \color{red}0 \\ 0 & 0 & 1 & 0 \end{matrix} \arrow[l]
\end{tikzcd}$$

Ullmann Algorithm (@1976)

- Step 3. Verify matrix M’ by the following equation

$$MC = M'(M' \bullet MB)^T$$

$$\forall i \forall j : (MA[i][j] = 1)$$
$$\Rightarrow (MC[i][j] = 1)$$

Iterate the above steps and enumerate all possible matrixes M’ .

In the worst case, there are O(|MB|!) possible matrixes. (subgraph isomorphism is a classical NP-hard problem)

Some Optimization

- Intuitions :

1) we can reduce the search space by “neighbor connection”.

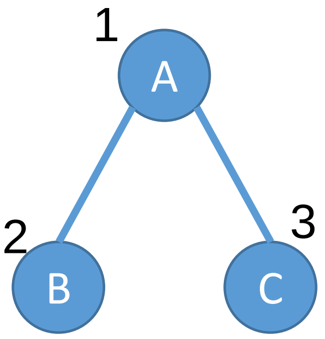
2) we can terminate some search branches as early as possible, also based on “neighbor connection”.

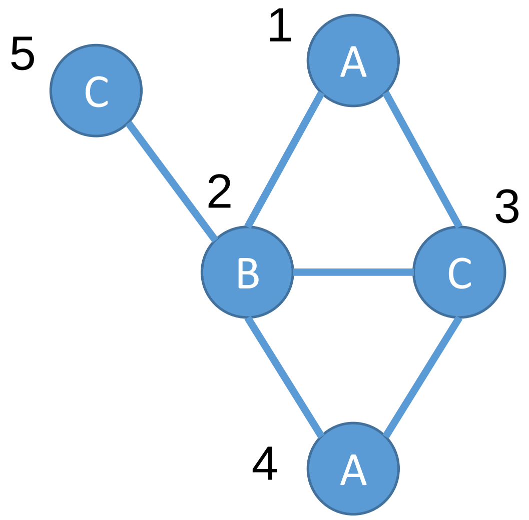
What’s neighbor connection ?

Refinement:

Let the i-th vertex *v* in Q corresponds to the j-th vertex *u* in G. Each neighbor vertex of *v* in Q must correspond to some neighbor vertex of *u* in G. Otherwise, *v* cannot correspond to *u*.

$$(\forall_{1 \le x \le N} x) \quad (MA[i][x]=1) \Rightarrow \exists_{1 \le y \le M} y \quad (M[x][y] \cdot MB[y][j]=1)$$





$$\begin{array}{c|ccccc}
\boldsymbol{M} & 1 & 2 & 3 & 4 & 5 \\
\hline
1 & 1 & 0 & 0 & 1 & 0 \\
2 & 0 & 1 & 0 & 0 & 0 \\
3 & 0 & 0 & 1 & 0 & 1
\end{array}$$

Can be removed

Ullmann Algorithm with Refinement

- Refinement Idea:

  1. Considering the matrix M, for each 1 in M, we refine it by the following equation. If fails, change 1 to 0 in M.

$$(\forall_{1 \le x \le N} x) \quad (MA[i][x] = 1) \Rightarrow \exists_{1 \le y \le M} y \quad (M[x][y] \cdot MB[y][j] = 1)$$

  2. If there exists at least one row (in M) having no 1, we report no subgraph isomorphism from Q to G.

  3. The refinement process is iterative.

Ullmann Algorithm with Refinement

- During the enumeration in the Ullmann algorithm, for some obtained matrix M’, we can also refine M’ by the same method. If fails, we can terminate the search branch as early as possible.

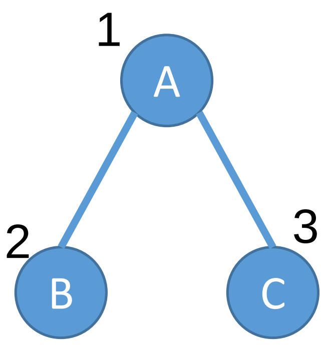
Other Algorithms

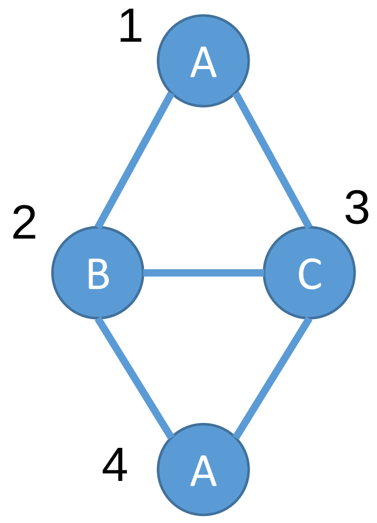
- VF2 Algorithm (@04)
- QuickSI(@08)

VF2 Algorithm @04

Considering two graph Q and G, the (sub)graph isomorphism from Q to G is expressed as the set of pairs (n,m) (with n G_1, with m G_2)

$\in$ $\in$





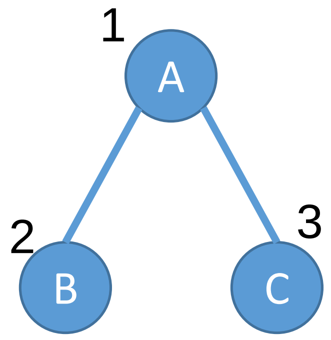
S_1	S_2
(1, 1)	(1, 4)
(2, 2)	(2, 2)
(3, 3)	(3, 3)

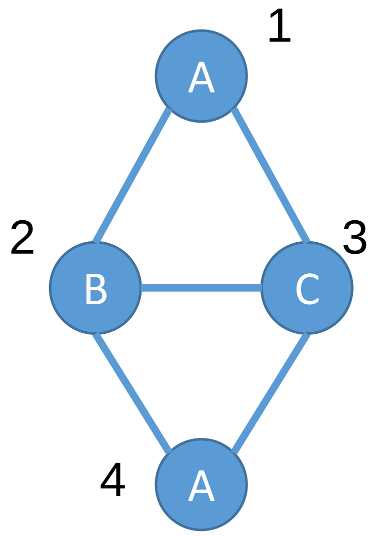
VF2 Algorithm

How to find candidate pair sets for a intermediate state ?

- Idea:

  Finding the (sub)graph isomorphism between Q and G is a sequence of state transition.





	Intermediate States
s1	(2,2)
s2	(2,2) (1,1)
s3	(2,2)(1,1)(3,3)

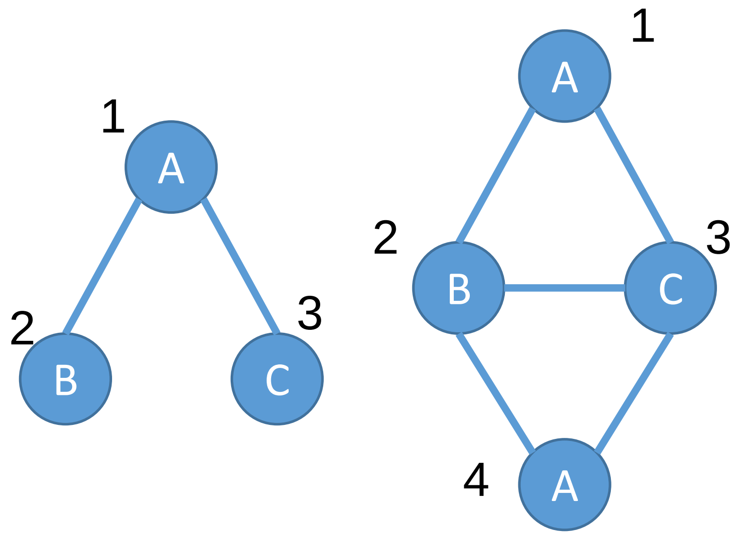
VF2 Algorithm @04

- Let *s* to be an intermediate state. Actually, s denotes a partial mapping from Q to G, namely, a mapping from a subgraph of Q to a subgraph of G. These two subgraphs are denoted as Q(s) and G(s), respectively.

- All neighbor vertices to Q(s) in graph Q are denoted as NQ(s), and all neighbor vertices to G(s) in graph G are denoted as NG(s). Candidate pair sets are a subset of NQ(s) × NG(s).

  Assume that a pair (n,m) | NQ(s) × NG(s).

VF2 Algorithm



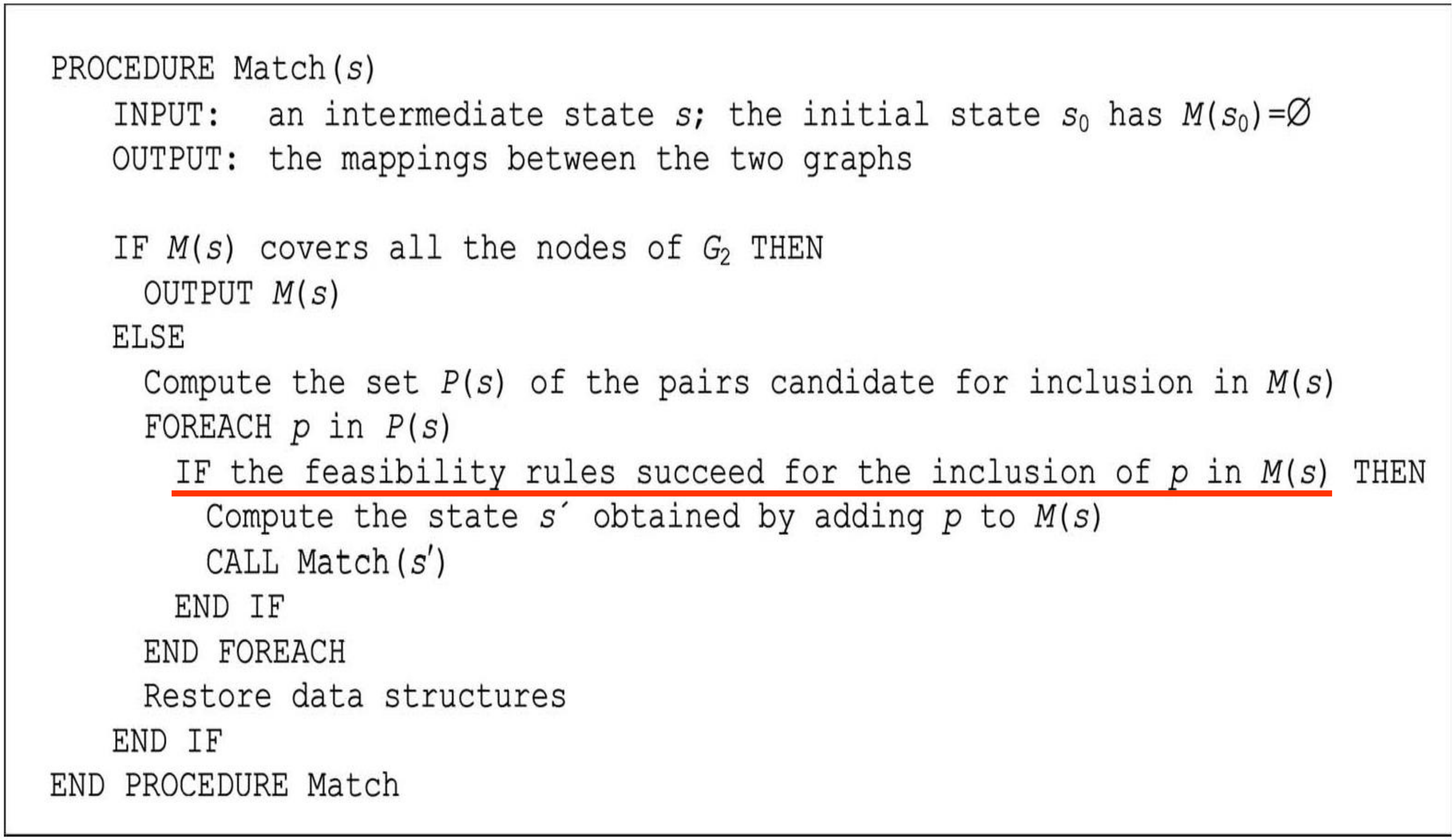
(2, 2)

Candidate Pair Sets

(1, 1) (1, 4)

(3, 3) (3,3)

VF2 Algorithm



```
PROCEDURE Match($s$)
    INPUT:   an intermediate state $s$; the initial state $s_0$ has $M(s_0)=\varnothing$
    OUTPUT:  the mappings between the two graphs

    IF $M(s)$ covers all the nodes of $G_2$ THEN
      OUTPUT $M(s)$
    ELSE
      Compute the set $P(s)$ of the pairs candidate for inclusion in $M(s)$
      FOREACH $p$ in $P(s)$
        IF the feasibility rules succeed for the inclusion of $p$ in $M(s)$ THEN
          Compute the state $s'$ obtained by adding $p$ to $M(s)$
          CALL Match($s'$)
        END IF
      END FOREACH
      Restore data structures
    END IF
END PROCEDURE Match
```

VF2 Algorithm @04

- Assume that a pair (n,m) $\in$ NQ(s) $\times$ NG(s).

$$F(s,n,m) = F_{structure}(s,n,m) \wedge F_{label}(s,n,m)$$

Some Structural Feasibility Rules

- Notations

s: An intermediate State

V1(s): The set of vertices of Q that corresponds to State s;
V2(s): The set of vertices of G that corresponds to State s;
E1(s): The set of edges of Q that corresponds to State s;
E2(s): The set of edges of Q that corresponds to State s;

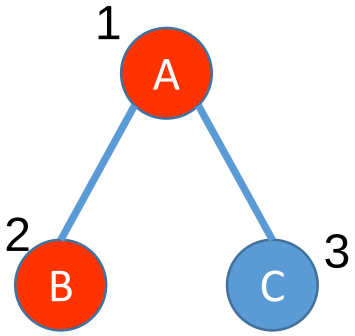
N1(n,Q): The neighbors of vertex n in graph Q;
N2(m,G): The neighbors of vertex m in graph G;

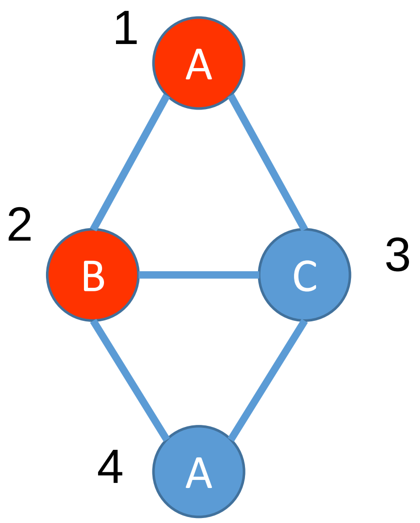
T1(s,Q): The neighbors (that are not in state s) of state s in graph Q;
T2(s,G): The neighbors (that are not in state s) of state s in graph G;

Some Structural Feasibility Rules

- Neighbor Connection

$$F(s,n,m) \Leftrightarrow (\forall n' \in (V_1(s) \cap N_1(n,Q)))$$
$$\exists m' \in (V_2(s) \cap N_2(m,G))$$





$$F_{structure}(s,n,m)$$

$s:$

$1 \leftrightarrow 1$

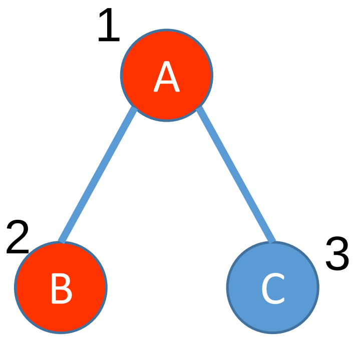
$2 \leftrightarrow 2$

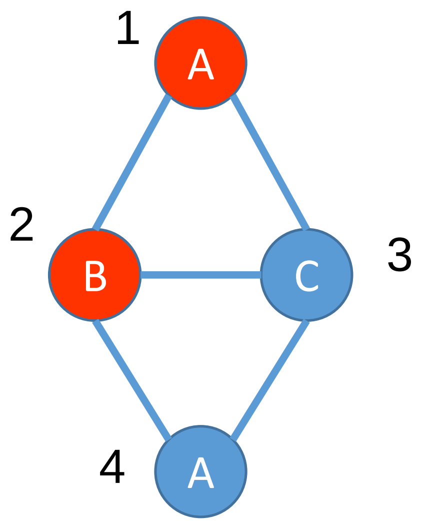
$n = 3; m = 3$

Some Structural Feasibility Rules

- Neighbor Connection

$$F(s,n,m) \Leftrightarrow |N_1(n,Q) \cap T_1(s,Q)| \le |N_2(m,G) \cap T_2(s,G)|$$





$F_{structure}(s,n,m)$

$s:$

$1 \leftrightarrow 1$

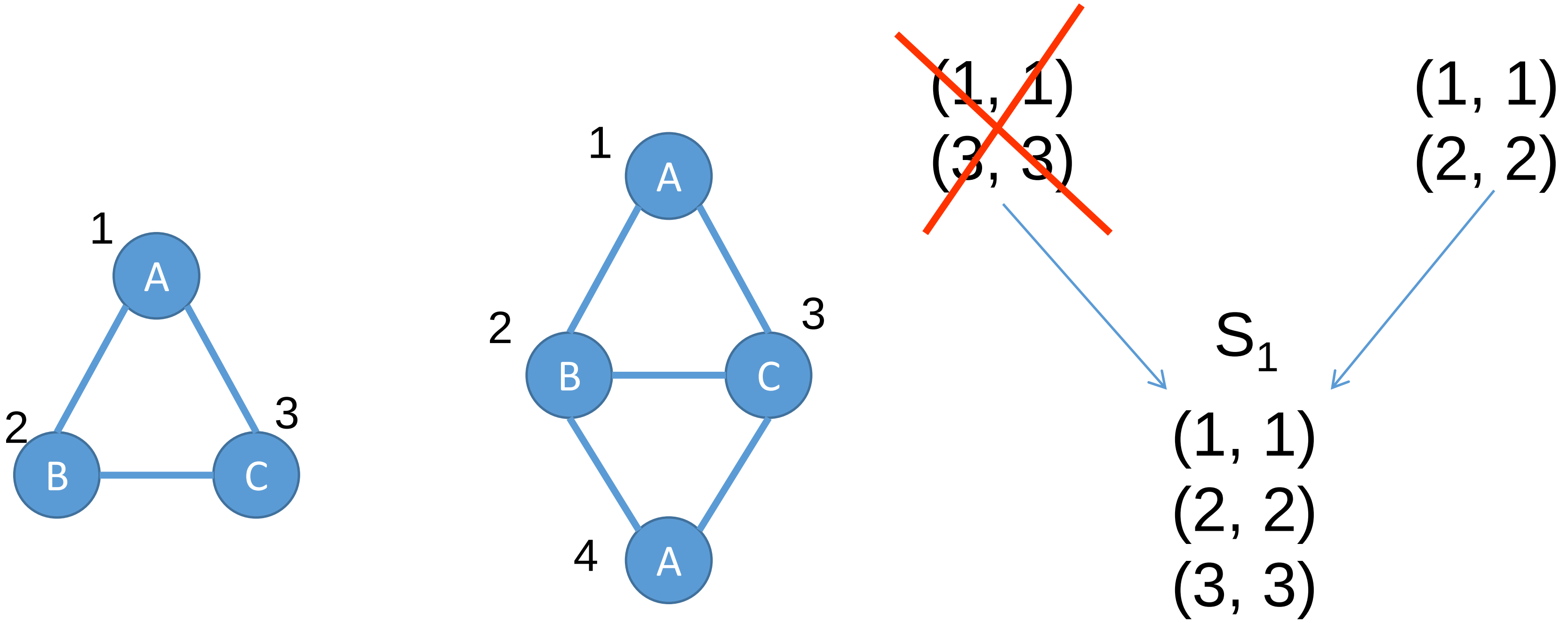
$2 \leftrightarrow 2$

$n = 3; m = 3$

VF2 Algorithm

- A state can be reached through different paths.
- An arbitrary total order is defined on the nodes of Q.

Eg. (1 < 2 < 3)



1. A good survey about graph matching algorithms:

《THIRTY YEARS OF GRAPH MATCHING IN PATTERN RECOGNITION》 @IJPR04

2. C++ library For Graph Isomorphism

VFLib library

References

1. Julian R. Ullmann: An Algorithm for Subgraph Isomorphism.J. ACM 23(1): 31-42 (1976)
2. Luigi P. Cordella,Pasquale Foggia,Carlo Sansone,Mario Vento: A (Sub)Graph Isomorphism Algorithm for Matching Large Graphs.IEEE Trans. Pattern Anal. Mach. Intell. 26(10): 1367-1372 (2004)