**Pseudocode**

Simple enumeration algorithm

Same as Ullmann’s algorithm, but without the red texts

Ullmann’s algorithm

|  |  |
| --- | --- |
| **Input:** | query graph adjacency list |
| **Input:** | data graph of size |
| **Output:** | all subgraph isomorphisms of in |
| 1: |  |
| 2: |  |
| 3: |  |
| 4: | *:=* PRELIMINARY\_REFINEMENT*()* |
| 5: | *:=* NODES\_GENERATION*(* |
| 6: | NODES\_CHECK*()* |
| **Subroutine** ROOT\_NODE\_GENERATION | |
| 1: | **for each** in |
| 2: | **for each**  in |
| 3: | **if**  has less neighbours than |
| 4: |  |
| 5: | **else** |
| 6: |  |
| 7: | **end** |
| 8: | **end** |
| 9: | **end** |
| **Subroutine** PRELIMINARY\_REFINEMENT | |
| 1: | **for each**  in |
| 2: | **if**  has only one possible isomorphism in H |
| 3: |  |
| 4: | **end** |
| 5: | **end** |
| **Subroutine** NODES­\_GENERATION | |
| 1: | **for each**  in |
| 2: |  |
| 3: | **if** first iteration |
| 4: |  |
| 5: | **for each**  in |
| 6: | **if** |
| 7: |  |
| 8: |  |
| 9: | **for each** neighbour of |
| 10: | **if** |
| 11: |  |
| 12: | **break** |
| 13: | **end** |
| 14: | **end** |
| 15: | **if** |
| 16: |  |
| 17: | **continue** |
| 18: | **else** |
| 19: |  |
| 20: |  |
| 21: |  |
| 22: | **else** |
| 23: | **for each** inner node stored in |
| 24: | each inner node |
| 25: | **repeat** highlighted parts |
| 26: | **end** |
| 27: | **end** |
| **Subroutine** NODES\_CHECK | |
| 1: | **for each** terminal nodes in |
| 2: | **if** |
| 3: | **store** |
| 4: | **end** |
| 5: |  |

VF2

Mostly the same as Ullmann’s algorithm, except that during nodes generation, we prune out a vertex in a set of candidate vertices s.t. :

1. is not connected from already matches data vertices.
2. , where is a set of adjacent and not yet matched query vertices connected from the set of already matches query vertices, is a set of adjacent and not yet matched query vertices connected from the set of already matches data vertices, is a set of adjacent vertices to a vertex .

QuickSI

GraphQL

GADDI

SPath