Advanced Techniques for Mining Structured Data:

Graph Mining

Graph Matching

Dr C.Loglisci

PhD Course in Computer Science and Mathematics XXXII cycle

Querying Graphs

- Find the friends of a friend who are interested in pop-music
- Find all the German presidents that have been elected twice
- Find all the molecules that contain a particular compound
- Find a touristic path with more historical monuments that is closest
- to Berlin
- …

Challenges

- No fixed schema (i.e., no rule for the structure, we should change relational schemas)
- Hard to find information in a graph which meet structural specification, but also specific characteristic of the elements involved

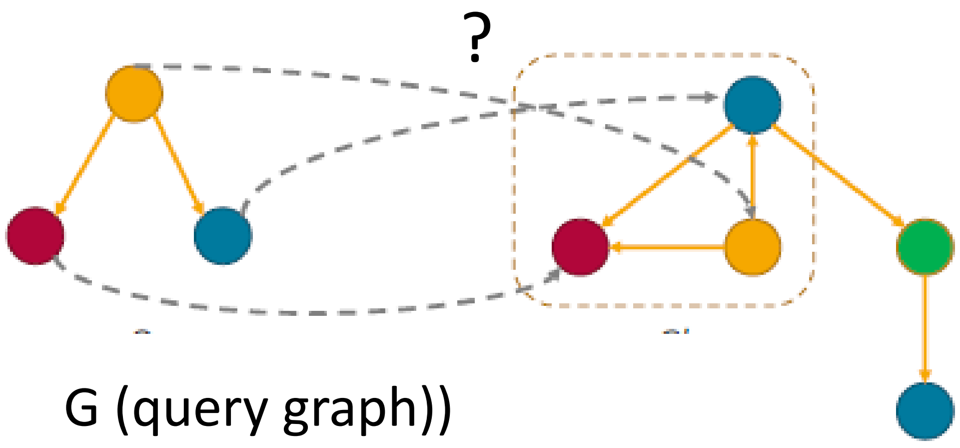
- However
- There is no a single a query language (SPARQL, Gremlin, Cypher, …) differently from ANSI SQL in RDBMS
- Many different queries (reaching a point connected to others, best neighbors, several options)

Containment Queries

- Containment queries
  - Ask if a (sub)structure/compound is contained in a graph
  - Retrieves all graphs from a graph database , such that they contain a given query graph.
  - Example is: Find all the molecules containing a specific compound

- Similarity queries
  - Retrieves all graphs from a graph database, which are similar to the query graph (exact and approximate).
  - Examples is: Find the other molecules with the same structure

Containment Queries



G (query graph))

H (database graphs)

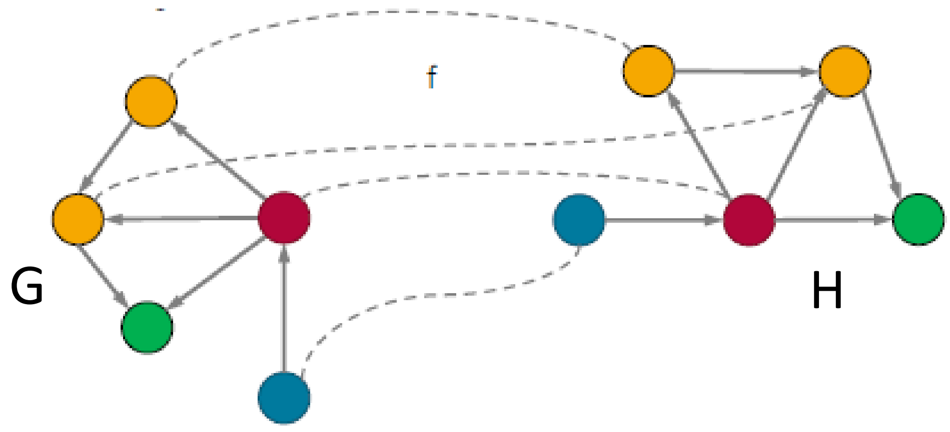
Solution:

- Recursively match structures from the query to the graph
- Return all the substructures of that kind
- Use subgraph isomorphism to find matches of the exact structure

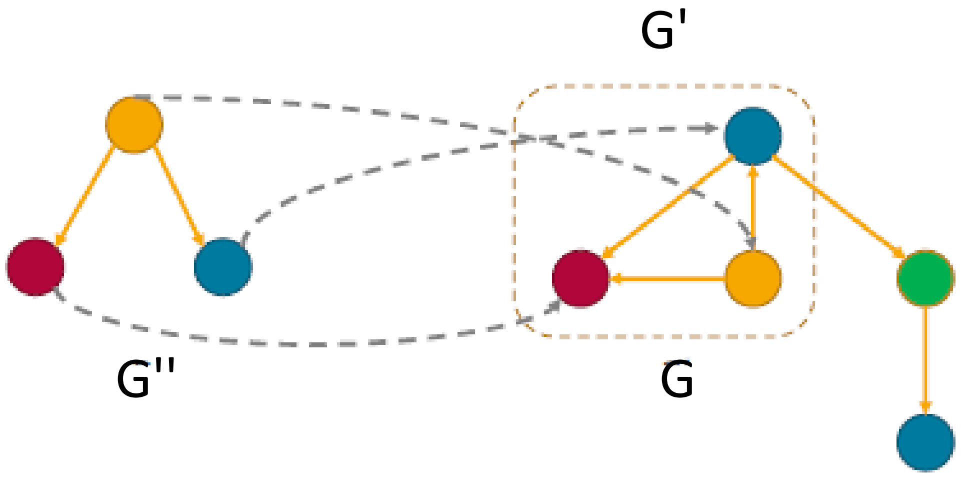
Isomorphism

Given two graphs,G: (V, E), H(V', E') G is isomorphic to H iff exists a bijective function f: V → V' s.t.:

1. For each v ∈ V, v = f (v)
2. (v, u) ∈ E iff (f (v) , (f (u)) ∈ E'



Given a G'': (V'', E'') is subgraph isomorphic to G if exists a subgraph G' ⊑ G, G'' isomorphic to G'

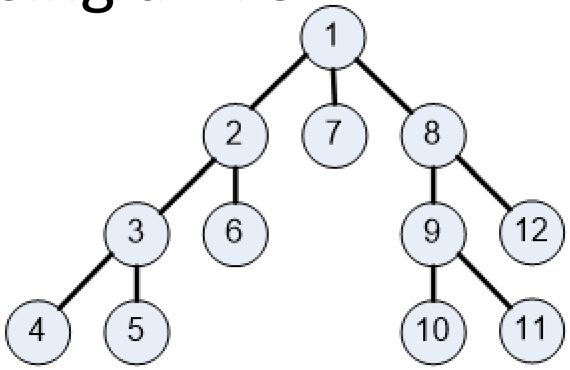


Ullmann's algorithm

- Tree-search algorithm (DFS) to generate spaces of candidate graphs

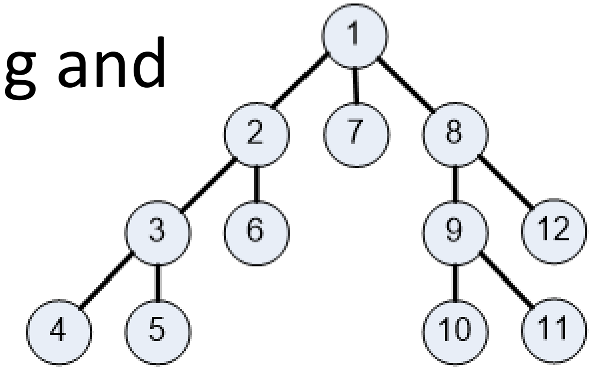
- One DFS is used to check the containment between a subgraph query and a graph database

- Check the graph isomorphism by using a DFS



Ullmann's algorithm

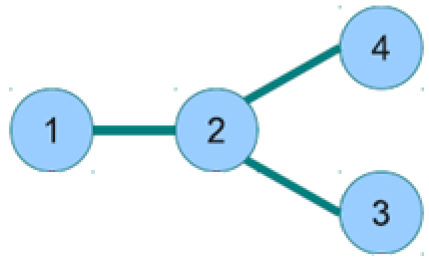
- Procedure:
  - A partial match (initially empty) is iteratively expanded by adding to it new pairs of matched nodes
  - A node pair specializes the parent node
  - The pair is chosen with the aim to satisfy some necessary conditions, usually also some heuristic condition. If this is not so, a node can be pruned
  - Finally, either the algorithm finds a complete matching, or no further node pairs may be added (backtracking)
  - Uses adjacency and permutation matrices for matching and pruning



Ullmann’s algorithm

- Basic idea:
  - the adjacency matrix AH of a graph H is:

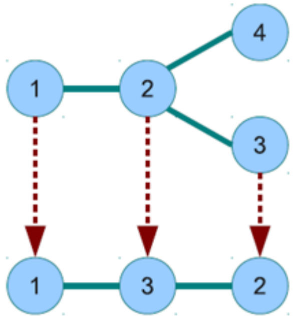
AH: $\begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$



  - the permutation matrix is equivalent to the correspondence F
  - Given: n, nodes of G, m, nodes of H
    - the permutation matrix M is nXm
    - exact one 1 in each row
    - not more than one 1 in each column

F: 1H-1G
2H-3G
3H-2G
4H-null

M: $\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$



Ullmann's algorithm

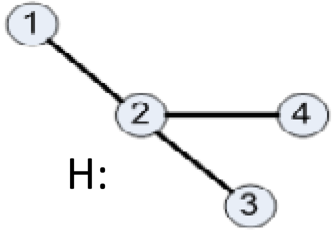
- Basic idea:
  - the contribution of the permutation matrix is to move rows and columns until to find an exact match (isomorphic subgraphs).
  - It does work also with for isomorphic graphs:

AH: $\begin{bmatrix} 0 & 1 & 0 & 0 \\ \boxed{1 \quad 0 \quad 1 \quad 1} \\ \boxed{0 \quad 1 \quad 0 \quad 0} \\ 0 & 1 & 0 & 0 \end{bmatrix}$

M AH: $\begin{vmatrix} 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 \end{vmatrix}$

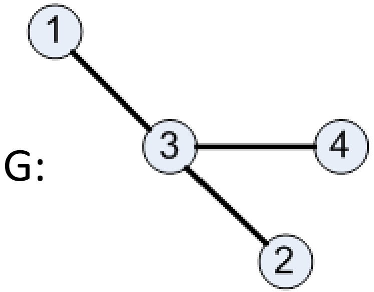
M AH M^T: $\begin{vmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{vmatrix}$

H:



M: $\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

G:



Ullmann's algorithm

- Goal: Find permutation matrices that satisfy the isomorphism criterion

  AH = M (M AG)^T

- How:
  - Enumerate, in a tree structure, candidate permutation matrices and check the criterion over each candidate
  - 1-Construction of the matrix (root) M^T

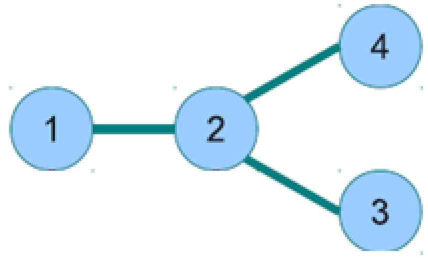
$$m_{i,j} = \begin{cases} 1 & \text{if } \deg(V_{Hi}) \geq \deg(V_{Gi}) \\ 0 & \text{otherwise} \end{cases}, m_{i,j} \in \{0,1\}$$

  - 2-Generation of all M by setting all the cells to 0 except 1 of each row of M
  - 3-Prune candidate which will not satisfy the isomorphism criterion (its children will not satisfy the criterion still)

Ullmann's algorithm

- How:
  - 1-Construction of the matrix (root) M^T

$$m_{i,j} = \begin{cases} 1 & \text{if } \deg(V_{Hi}) \geq \deg(V_{Gi}) \\ 0 & \text{otherwise} \end{cases}, m_{i,j} \in \{0,1\}$$

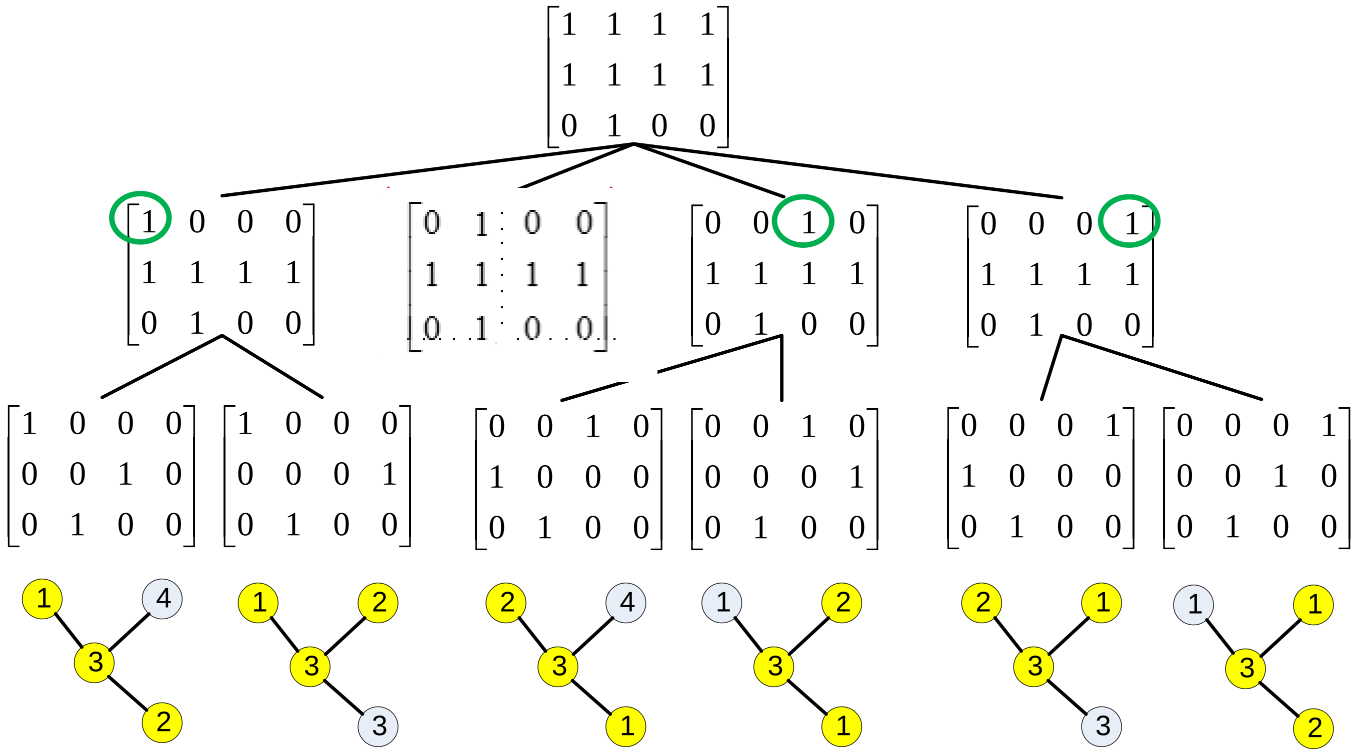


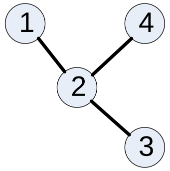
AH: $\begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$

$$\Rightarrow \quad M = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

AG: $\begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix}$

Ullmann's algorithm

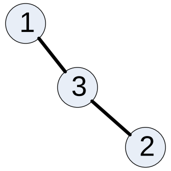




$$P = M'(M'A_H)^T$$

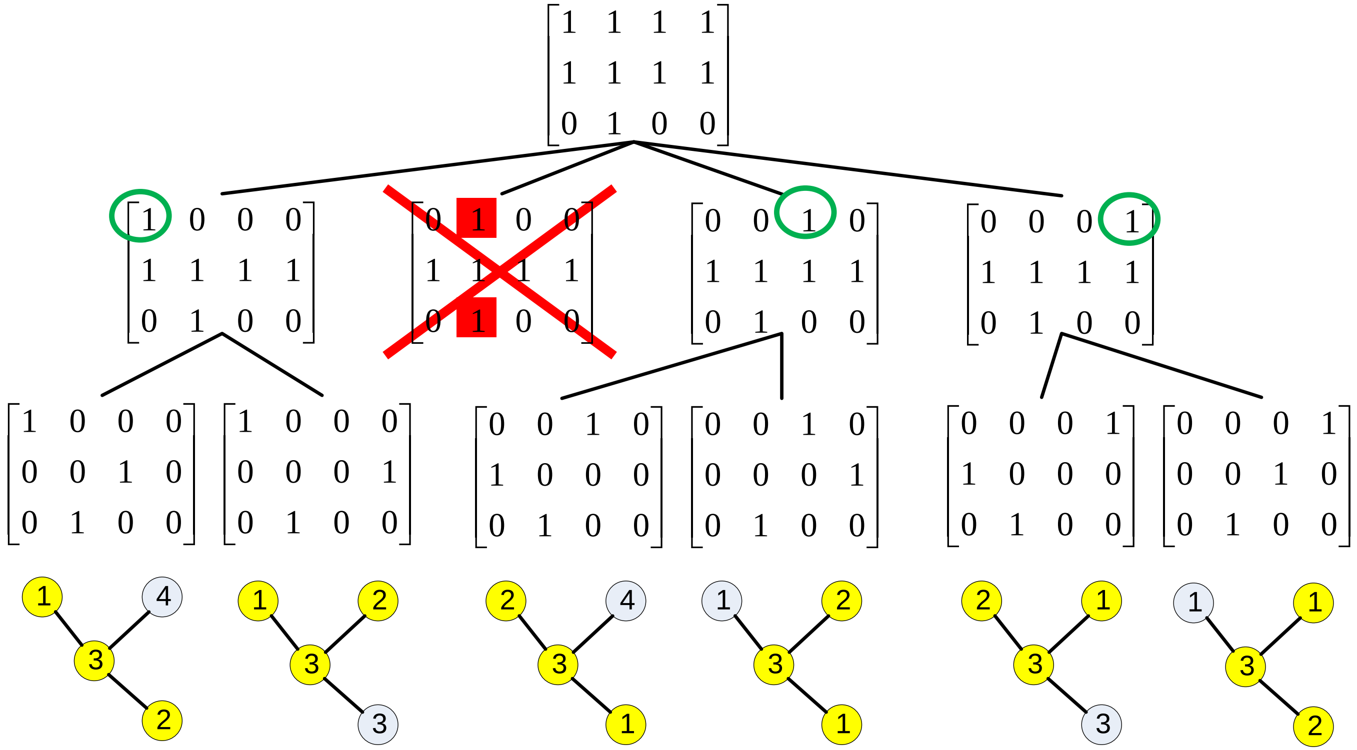
$$= \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix}$$

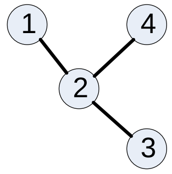
compared with $A_G = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix}$



- 2-Generation of all M by setting all the cells to 0 except 1 of each row of M, w.r.t. the parent permutation matrix

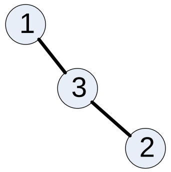
Ullmann's algorithm





$$P = M'(M'A_H)^T = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix}$$

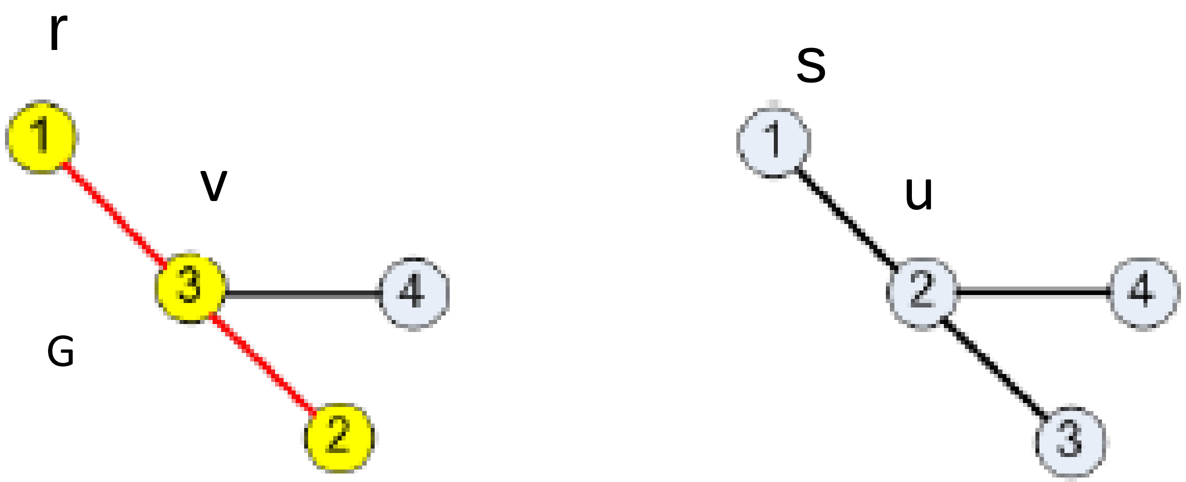
compared with $A_G = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix}$



- 2-Generation of all M by setting all the cells to 0 except 1 of each row of M, w.r.t. the parent permutation matrix

Ullmann's algorithm

- How:
  - 3-Prune candidate which will not satisfy the isomorphism criterion:
  - if a vertex v of G, v corresponds to a vertex u of H, then for each adjacent vertex of v in G, denoted as r, there must be a vertex in H, denoted as s, which holds:
  - s is adjacent u in H
  - s corresponds to r



$\forall v \in G$, $a_G(v_i)=1 \Rightarrow \exists\, m \in M$ s.t. $m_{ij} * a_H(u_j)=1$,