Graph Matching

CSE 373
Data Structures & Algorithms
Linda Shapiro
Spring 2013

Today's Outline

- **Announcements:**
- **Today's Topics:**
  - › **Graph Matching by Backtracking Tree Search**

Graph Matching

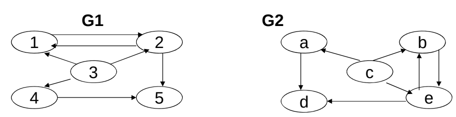
Input: 2 digraphs G1 = (V1,E1), G2 = (V2,E2)

Questions to ask:

1. Are G1 and G2 isomorphic?
2. Is G1 isomorphic to a subgraph of G2?
3. How similar is G1 to G2?
4. How similar is G1 to the most similar subgraph of G2?

Isomorphism for Digraphs

G1 is isomorphic to G2 if there is a 1-1, onto mapping h: V1 → V2 such that **(vi,vj) ∈ E1 iff (h(vi), h(vj)) ∈ E2**.

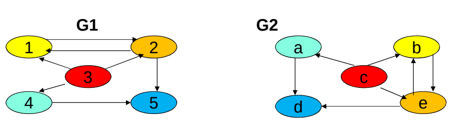


Find an isomorphism h: {1,2,3,4,5} → {a,b,c,d,e}.
Check that the condition holds for every edge.

Answer: h(1)=b, h(2)=e, h(3)=c, h(4)=a, h(5)=d

Isomorphism for Digraphs

G1 is isomorphic to G2 if there is a 1-1, onto mapping h: V1 → V2 such that (vi,vj) ∈ E1 iff (h(vi), h(vj)) ∈ E2



Answer: h(1)=b, h(2)=e, h(3)=c, h(4)=a, h(5)=d
(1,2) ∈ E1 and (h(1),h(2))=(b,e) ∈ E2.
(2,1) ∈ E1 and (e,b) ∈ E2.
(2,5) ∈ E1 and (e,d) ∈ E2.
(3,1) ∈ E1 and (c,b) ∈ E2.
(3,2) ∈ E1 and (c,e) ∈ E2.
...

Subgraph Isomorphism for Digraphs

G1 is isomorphic to a subgraph of G2 if there is a 1-1 mapping h: V1 → V2 such that **(vi,vj) ∈ E1 ⇒ (h(vi), h(vj)) ∈ E2**.

Isomorphism and subgraph isomorphism are defined similarly for undirected graphs.

In this case, when (vi,vj) ∈ E1, either (vi,vj) or (vj,vi) can be listed in E2, since they are equivalent and both mean {vi,vj}.

Subgraph Isomorphism for Graphs

G1 is isomorphic to a subgraph of G2 if there is a 1-1 mapping h: V1 → V2 such that **{vi,vj} ∈ E1 ⇒ { h(vi), h(vj) } ∈ E2**.

Because there are no directed edges, there are **more** possible mappings.

1	2	3	
c	b	d	
c	d	b	(shown on graph)
b	c	d	
b	d	c	
d	b	c	
d	c	b	

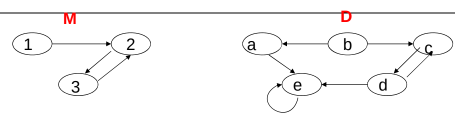
Graph Matching Algorithms: Subgraph Isomorphism for Digraph

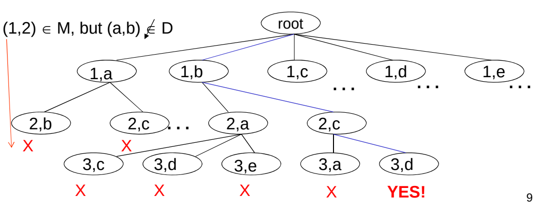
Given model graph M = (VM,EM)
data graph D = (VD,ED)

Find 1-1 mapping h:VM → VD

satisfying (vi,vj) ∈ EM ⇒ ((h(vi),h(vj)) ∈ ED.

Method: Recursive Backtracking Tree Search (Order is depth first, leftmost child first.)



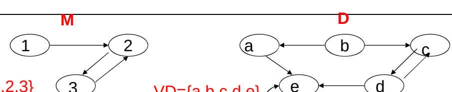


Treesearch for Subgraph Isomorphism in Digraphs

```
procedure Treesearch(VM, VD, EM, ED, h) {
  v = first(VM);
  for each w ∈ VD {
    h' = h ∪ {(v,w)};           //add to mapping
    OK = true;
    for each edge (vi,vj) in EM          (with vi < vj for
      if one of vi or vj is v and the other    undirected graphs)
         has been assigned a value in h'
          if ( (h'(vi),h'(vj)) is NOT in ED )
            {OK = false; break;};

    if OK {
          VM' = VM – v;                //remove from set
          VD' = VD – w'
           if isempty(VM') output(h');
           else Treesearch(VM',VD',EM,ED,h')
          } } }
```


Method: Recursive Backtracking Tree Search (Order is depth first, leftmost child first.)

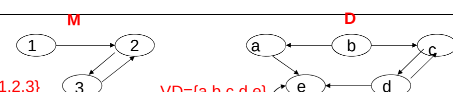


VM={1,2,3}
EM = {(1,2),(2,3),(3,2)}

VD={a,b,c,d,e}
ED={(a,e),(b,a),(b,c),(c,d),(d,c),(d,e),(e,e)}

v = 1; w = a; h' = {(1,a)}; no edges can be checked yet.
v = 2; w = b; h' = {(1,a), (2,b)}; (1,2) ∈ EM, but (a,b) not ∈ ED; OK = false.
w = c; h' = {(1,a), (2,c)}; (1,2) ∈ EM, but (a,c) not in ED; OK = false.
...
Nothing works for vertex 2 when (1,a) is in h.
The tree search comes back from the recursive call, returning back to v=1.
v=1; w = b; h' = {(1,b)}; no edges can be checked yet.
v=2; w = a; h' = {(1,b),(2,a)}; (1,2) ∈ EM and (b,a) ∈ ED, so OK stays true
v=3; w = c; h' = {(1,b),(2,a);(3,c)}; (2,3) ∈ EM but (a,c) not ∈ ED; OK = false
w = d and w = e also fail, so eventually v=3 fails, and v=2 with w=a fails

Method: Recursive Backtracking Tree Search (Order is depth first, leftmost child first.)



VM={1,2,3}
EM = {(1,2),(2,3),(3,2)}

VD={a,b,c,d,e}
ED={(a,e),(b,a),(b,c),(c,d),(d,c),(d,e),(e,e)}

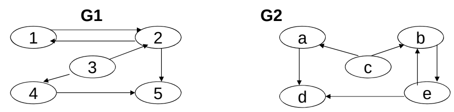
v=1; w = b; h' = {(1,b)}; no edges can be checked yet.
v=2; w = c; h' = {(1,b),(2,c)}; (1,2) ∈ EM and (b,c) ∈ ED, so OK stays true
v=3; w = a; h' = {(1,b),(2,c),(3,a)}; (2,3) ∈ EM but (c,a) not ∈ ED; OK = false
w = d; h' = {(1,b),(2,c),(3,d)}; (2,3) ∈ EM and (c,d) ∈ ED, so OK
(3,2) ∈ EM and (d,c) ∈ ED, so OK

We've used up all of VM, so output the final mapping {(1,b),(2,c),(3,d)}

Similar Digraphs

Sometimes two graphs are close to isomorphic, but have a few "errors."

Let h(1)=b, h(2)=e, h(3)=c, h(4)=a, h(5) = d.



(1,2)	(b,e)
(2,1)	(e,b)
X	(c,b)
(4,5)	(a,d)
(2,5)	(e,d)
(3,2)	X
(3,4)	(c,a)

The mapping **h** has **2 errors**.

(c,b) $\in$ G2, but (3,1) $\notin$ G1

(3,2) $\in$ G1, but (c,e) $\notin$ G2

Error of a Mapping

Intuitively, the error of mapping h tells us

- how many edges of G1 have no corresponding edge in G2 and
- how many edges of G2 have no corresponding edge in G1.

Let G1=(V1,E1) and G2=(V2,E2), and let h:V1$\rightarrow$ V2 be a 1-1, onto mapping.

forward error — EF(h) = |{(vi,vj)$\in$E1 | (h(vi),h(vj))$\notin$ E2}|
edge in E1 — corresponding edge not in E2

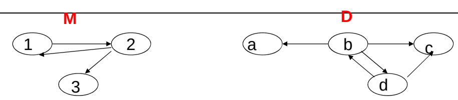
backward error — EB(h) = |{(vi,vj)$\in$E2 | (h^{-1}(vi),h^{-1}(vj))$\notin$E1}|
edge in E2 — corresponding edge not in E1

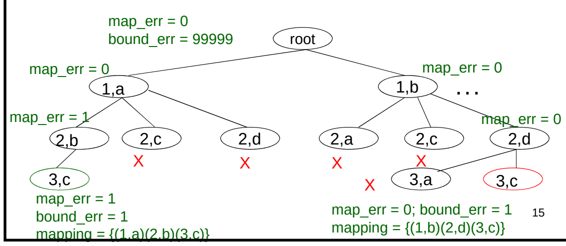
total error — Error(h) = EF(h) + EB(h)

relational distance — GD(G1,G2) = min Error(h)
for all 1-1, onto h:V1$\rightarrow$ V2

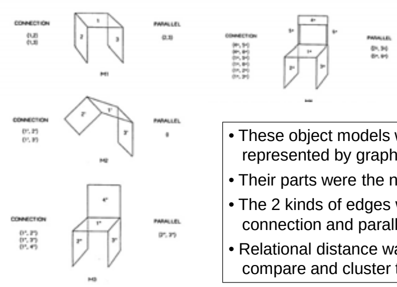
Branch-and-Bound Tree Search

Keep track of the least-error mapping.





Application of Relational Distance



- These object models were represented by graphs
- Their parts were the nodes.
- The 2 kinds of edges were for connection and parallel relations.
- Relational distance was used to compare and cluster them.