Structure and Representation

BINF739 SPRING2007
Jeff Solka and Jennifer Weller

Acknowledgement

- Unless otherwise noted all figures in this lecture have been adapted from Gross and Yellen, *Graph Theory and Its Application*, Chapman and Hall/CRC Press, 2006.

Introduction

- The structure of a graph is what characterizes a graph that is independent of its representation
- A graph can have many representations
  - Incidence table
  - Drawings
- When are two graphs structurally equivalent (isomorphism problem)
- Incidence matrix and adjacency matrix allow use to utilize vector space mathematics
- Addition and deletion of nodes and edges on graphs become similar in scope to the elementary row operations that we find in linear algebra

2.1 Graph Isomorphism

- Structurally Equivalent Graphs
  - How do we know that these two graphs are the same?

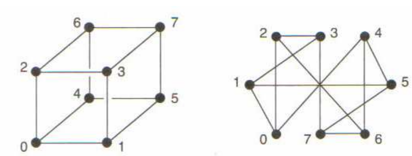


Figure 2.1.1 Two different drawings of the same graph.

2.1 Graph Isomorphism

- Structurally Equivalent Graphs
  - What if the edges are labeled differently from one graph to another or not at all.
    - There in lies the rub.

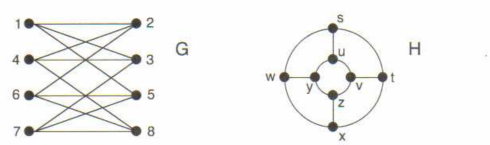


Figure 2.1.2 Two drawings of essentially the same graph.

2.1 Graph Isomorphism

Figure 2.1.2 Two drawings of essentially the same graph.

$1 \to s$	$5 \to w$
$2 \to t$	$6 \to x$
$3 \to u$	$7 \to y$
$4 \to v$	$8 \to z$

2.1 Graph Isomorphism

- Formalizing Structural Equivalence for Simple Graphs
  - Def. – Leg *G* and *H* be two simple graphs. A vertex bijection *f*: $V_G \rightarrow V_H$ **preserves adjacency** if for every pair of adjacent vertices *u* and *v* in graph *G*, the vertices *f(u)* and *f(v)* are adjacent in graph *H*. Similarly, *f* **preserves non-adjacency** if *f(u)* and *f(v)* are non-adjacent whenever *u* and *v* are non-adjacent.
  - Def. – A vertex bijection : $V_G \rightarrow V_H$ between the vertex-sets of two simple graphs G and H is structure-preserving if it preserves adjacency and non-adjacency. That is, for every paif of vertices in G,

$$u \text{ and } v \text{ are adjacent in G} \Leftrightarrow f(u) \text{ and } f(v) \text{ are adjacent in } H$$

2.1 Graph Isomorphism

- Formalizing Structural Equivalence for Simple Graphs

Def. - Two simple graphs G and H are isomorphic, denoted $G \cong H$, if there exists a structure-preserving vertex bijection $f : V_G \rightarrow V_H$. Such a function f between the vertex-sets of G and H is called an isomorphism from G to H.

2.1 Graph Isomorphism

- Formalizing Structural Equivalence for Simple Graphs

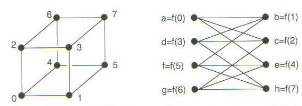


Figure 2.1.3 Specifying an isomorphism between two simple graphs.

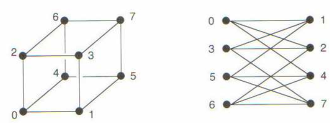


Figure 2.1.4 Another way of depicting an isomorphism.

2.1 Graph Isomorphism

- Formalizing Structural Equivalence for Simple Graphs

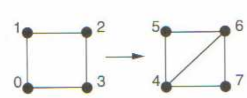


Figure 2.1.5 Bijective and adjacency-preserving, but not an isomorphism.

Figure 2.1.6 Preserves adjacency and non-adjacency, but is not bijective.

2.1 Graph Isomorphism

- Extending the Definition of Isomorphism to General Graphs
  - Does this mapping preserve adjacency and non-adjacency?
  - Are these two graphs structurally equivalent?

Figure 2.1.7 Two graphs that are not structurally equivalent.

2.1 Graph Isomorphism

- Extending the Definition of Isomorphism to General Graphs
  - Def. – A vertex bijection *f:* $V_g \rightarrow V_H$ between the vertex-sets of two graphs *G* and *H*, simple or general, is structure-preserving if
    1. The number of (edges) (even if 0) between every pair of distinct vertices *u* and *v* in graph *G* equals the number of edges between their images *f(u)* and *f(v)* in graph *H*, and
    2. The number of self-loops at each vertex *x* in *G* equals the number of self-loops at the vertex *f(x)* in *H*.

2.1 Graph Isomorphism

- Specifying an Isomorphism Between Graphs Having Multi-Edges
  - Def. – Let *G* and *H* be two isomorphic graphs. A vertex bijection f_V: $V_G \rightarrow V_H$ and an edge bijection f_E: $E_G \rightarrow E_H$ are consistent if to every edge *e* in E_G, the function f_V maps the endpoints of *e* to the endpoints of edge $f_E(e)$. A consistent mapping pair (f_V: $V_G \rightarrow V_H$, f_E: $E_G \rightarrow E_H$) is often written shorthand as $f: G \rightarrow H$.
  - Prop. 2.1.1 – Let *G* and *H* be any two graphs. Then *G* is isomorphic iff there is a vertex bijection f_V: $V_G \rightarrow V_H$ and an edge bijection f_E: $E_G \rightarrow E_H$ that are consistent.

2.1 Graph Isomorphism

DEFINITION: If G and H are graphs *with multi-edges*, then an **isomorphism** from G to H is specified by giving a vertex bijection $f_V : V_G \rightarrow V_H$ *and* an edge bijection $f_E : E_G \rightarrow E_H$ that are consistent.

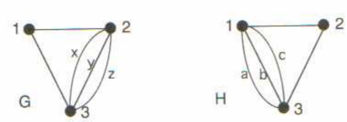


Figure 2.1.8 There are 12 distinct isomorphisms from G to H.

2.1 Graph Isomorphism

- Isomorphic Graph Pairs
  - Thm. 2.1.2 – Let *G* and *H* be isomorphic graphs. The they have the same number of vertices and the same number of edges.
  - Thm. 2.1.3 – Let *f: G → H* be a graph isomorphism and let *v* be in V_G then *deg(f(v)) = deg(v).*
  - Cor. 2.1.4 – Let *G* and *H* be isomorphic graphs. Then they have the same degree sequence.
  - Cor. 2.1.5 – Let *f: G → H* ne a graph isomorphism and let *e* be in E_G. Then the endpoints of edge *f(e)* have the same degrees as the endpoints of *e*.

2.1 Graph Isomorphism

- Isomorphic Graph Pairs

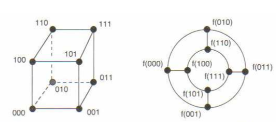


Figure 2.1.9 Hypercube graph Q_3 and circular ladder CL_4 are isomorphic.

DEFINITION: The **Möbius ladder** ML_n is a graph obtained from the circular ladder CL_n by deleting from the circular ladder two of its parallel curved edges and replacing them with two edges that cross-match their endpoints.

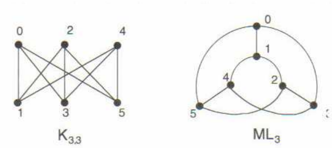


Figure 2.1.10 Bipartite graph $K_{3,3}$ and Möbius ladder ML_3 are isomorphic.

2.1 Graph Isomorphism

- Isomorphism Type of a Graph

Def. - Each equivalence class under $\cong$ (is isomorphic to) is called an isomorphism type.

Figure 2.1.11 The four isomorphism types for a simple 3-vertex graph.

2.1 Graph Isomorphism

- Isomorphism of Digraphs

DEFINITION: Two digraphs are ***isomorphic*** if there is an isomorphism f between their underlying graphs that preserves the direction of each edge. That is, e is directed from u to v if and only if $f(e)$ is directed from $f(u)$ to $f(v)$.

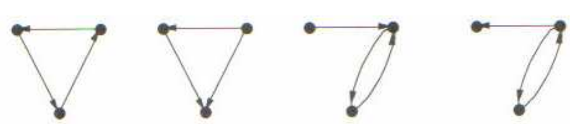


Figure 2.1.12 Four non-isomorphic digraphs.

2.1 Graph Isomorphism

- The Isomorphism Problem

DEFINITION: The ***graph-isomorphism problem*** is to devise a practical general algorithm to decide graph isomorphism, or, alternatively, to prove that no such algorithm exists.

Application 2.1.1 *Computer Chip Intellectual Property Rights:* Suppose that not long after ABC Corporation develops and markets a computer chip, it happens that the DEF Corporation markets a chip with striking operational similarities. If ABC could prove that DEF's circuitry is merely a rearrangement of the ABC circuitry (i.e., that the circuitries are isomorphic), they might have the basis for a patent-infringement suit. If ABC had to check structure preservation for each of the permutations of the nodes of the DEF chip, the task would take prohibitively long. However, knowledge of the organization of the chips might enable the ABC engineers to take a shortcut.

2.2 Automorphisms and Symmetry

- Def. – An isomorphism from a graph *G* to itself is called an **automorphism**.

- Thus an automorphism p of a graph *G* is a structure-preserving permutation on the vertex set of *G*, along with a (consistent) permutation π_E on the edge-set of of *G*. We may write $\pi = (\pi_V, \pi_E)$.

2.2 Automorphisms and Symmetry

- Permutations and Cycle Notation

Example 2.2.1: The permutation

$$\pi = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\ 7 & 4 & 1 & 8 & 5 & 2 & 9 & 6 & 3 \end{pmatrix}$$

which maps 1 to 7, 2 to 4, and so on, has the ***disjoint cycle form***

$$\pi = (1\ 7\ 9\ 3)(2\ 4\ 8\ 6)(5)$$

2.2 Automorphisms and Symmetry

- Geometric Symmetry

Example 2.2.2: The graph $K_{1,3}$ has six automorphisms. Each of them is realizable by a rotation or reflection of the drawing in Figure 2.2.1.

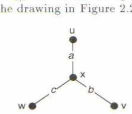


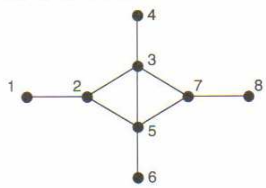
Figure 2.2.1 The graph $K_{1,3}$.

Symmetry	Vertex permutation	Edge permutation
identity	$(u)(v)(w)(x)$	$(a)(b)(c)$
120° rotation	$(x)(u\ v\ w)$	$(a\ b\ c)$
240° rotation	$(x)(u\ w\ v)$	$(a\ c\ b)$
refl. thru a	$(x)(u)(v\ w)$	$(a)(b\ c)$
refl. thru b	$(x)(v)(u\ w)$	$(b)(a\ c)$
refl. thru c	$(x)(w)(u\ v)$	$(c)(a\ b)$

$K_{1,4}$ has 4 vertices hence there should be 4! = 24 permutations but node x must be fixed (why??). Hence there are 3! Or 6 permutations

2.2 Automorphisms and Symmetry

- Geometric Symmetry



Automorphisms

$\lambda_0 = (1)(2)(3)(4)(5)(6)(7)(8)$

$\lambda_1 = (1\ 8)(2\ 7)(3)(4)(5)(6)$

$\lambda_2 = (1)(2)(3\ 5)(4\ 6)(7)(8)$

$\lambda_3 = (1\ 8)(2\ 7)(3\ 5)(4\ 6)$

Figure 2.2.2 A graph with four automorphisms.

2.2 Automorphisms and Symmetry

- Limitations of Geometric Symmetry

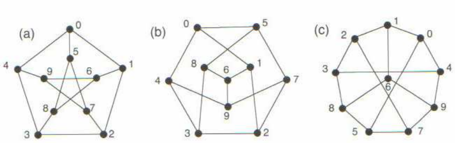
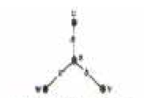


Figure 2.2.3 Three drawings of the Petersen graph.

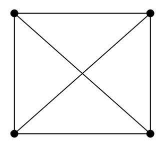
Example 2.2.4: Figure 2.2.3 shows three different drawings of the same-labeled Petersen graph. The leftmost drawing has 5-fold rotational symmetry that corresponds to the automorphism (0 1 2 3 4)(5 6 7 8 9), but this automorphism does not correspond to any geometric symmetry of either of the other two drawings. The automorphism (0 5)(1 8)(4 7)(2 3)(6)(9) is realized by 2-fold reflectional symmetry in the middle and rightmost drawings (about the axis through vertices 6 and 9) but is not realizable by any geometric symmetry in the leftmost drawing. There are several other automorphisms that are realizable by geometric symmetry in at least one of the drawings but not realizable in at least one of the other ones. (See Exercises.)

2.2 Automorphisms and Symmetry

- Limitations of Geometric Symmetry
  - Def. – A graph G is **vertex-transitive** if for every vertex pair *u*, *v* in V_G, there is an automorphism that maps *u* to *v*.
  - Def. A graph *G* is **edge-transitive** if for every edge pair *d, e* in E_G, there is an automorphism that maps *d* to *e*.



edge-transitive
not vertex transitive (why)



edge-transitive
and vertex-transitive (LTR)

2.2 Automorphisms and Symmetry

- Limitations of Geometric Symmetry

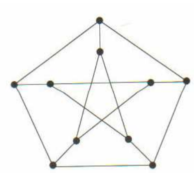


Figure 1.2.6 The Petersen graph.

VT and ET

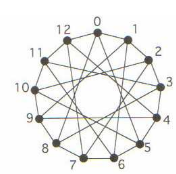


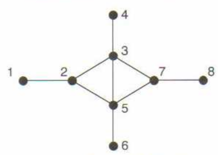
Figure 2.2.4 The circulant graph $circ(13 : 1, 5)$.

VT and ET

2.2 Automorphisms and Symmetry

- Vertex Orbits and Edge Orbits
  - Def. – The equivalence classes of the vertices of a graph under the action of the automorphisms are called **vertex orbits**. The equivalence classes of the edges are called **edge orbits**.

vertex orbits : $\{1,8\}$, $\{4,6\}$, $\{2,7\}$, $\{3,5\}$
edge orbits : $\{12,78\}$, $\{34,56\}$, $\{23,25,37,57\}$, $\{35\}$



Automorphisms

$\lambda_0=(1)(2)(3)(4)(5)(6)(7)(8)$
$\lambda_1=(1\ 8)(2\ 7)(3)(4)(5)(6)$
$\lambda_2=(1)(2)(3\ 5)(4\ 6)(7)(8)$
$\lambda_3=(1\ 8)(2\ 7)(3\ 5)(4\ 6)$

Figure 2.2.5 Graph of Example 2.2.3.

Thm. 2.2.1 – All vertices in the same orbit have the same degree.

Thm. 2.2.2 – Alledges in the same orbit have the same pair of degrees at their endpoints.

Automorphism theory lies at the intersection of graph theory and group theory (chpt. 14 and chpt. 15)

2.2 Automorphisms and Symmetry

- How to Find Orbits
  - It is not known whether there is a polynomial-time algorithm for finding orbits.

Example 2.2.13: In the graph of Figure 2.2.6, vertex 0 is the only vertex of degree 2, so it is in an orbit by itself. The vertical reflection (0)(1 4)(2 3) establishes that vertices 1 and 4 are co-orbital and that 2 and 3 are co-orbital. Moreover, since vertices 1 and 4 have a 2-valent neighbor but vertices 2 and 3 do not, the orbit of vertices 1 and 4 must be different from that of 2 and 3. Thus, the vertex orbits are

$$\{0\}, \{1,4\}, \text{ and } \{2,3\}$$

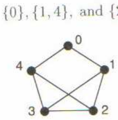


Figure 2.2.6 Find the vertex orbits and the edge orbits.

2.3 Subgraphs

- Def. – A **subgraph** of a graph *G* is a graph *H* whose vertices and edges are all in *G*. If *H* is a subgraph of *G*, we may also say that *G* is a **supergraph** of *H*.

- Def. – A subdigraph of a digraph *G* is a digraph *H* whose vertices and arcs are all in *G*.

- Def. – A proper subset *H* of *G* is a subgraph such that V_H is a proper subset of V_G or E_H is a proper subset of E_G.

2.3 Subgraphs

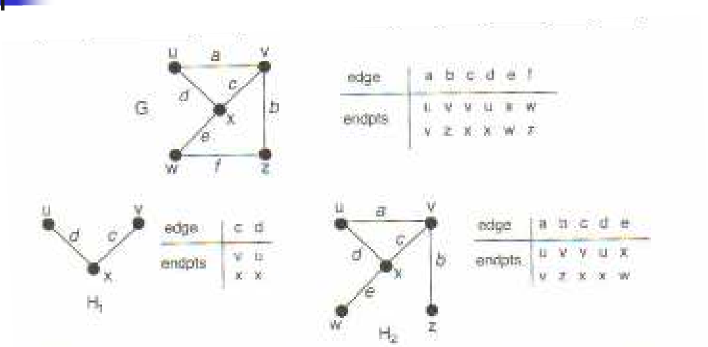


Figure 2.3.1 A graph G and two (proper) subgraphs H_1 and H_2.

Structure and Representation
BINF739 Solka/Weller BINF739

2.3 Subgraphs

- A Broader Use of the Term "Subgraph"
  - The usual meaning of the phrase "*H* is a subgraph of *G*" is that *H* is merely isomorphic to a subgraph of *G*."

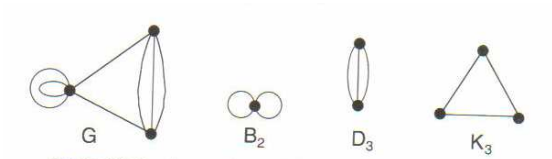
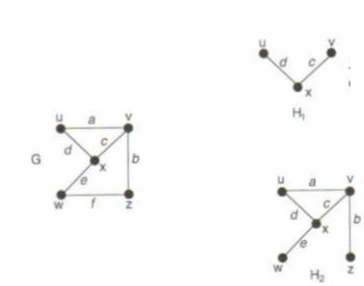


Figure 2.3.2 A graph and three of its subgraphs.

2.3 Subgraphs

- Spanning Subgraphs
  - Def. – A subgraph *H* is said to **span** a graph *G* is $V_H = V_G$



2.3 Subgraphs

- Spanning Subgraphs
  - Def. – A **spanning tree** of a graph is a spanning subgraph that is a tree.

Figure 2.3.3 A spanning tree.

2.3 Subgraphs

- Spanning Subgraphs
  - Prop. 2.3.1 – A graph *G* is **connected** iff it contains a spanning tree.
  - Prop. 2.3.2 – Every acyclic subgraph of a connected graph *G* is contained in at least one spanning tree of *G*.
  - Def. – An acyclic graph is called a **forest**.

2.3 Subgraphs

• Spanning Subgraphs

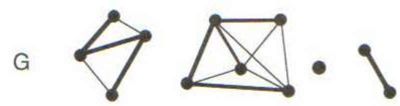


Figure 2.3.4 A spanning forest H of graph G.

Structure and Representation
BINF739 Solka/Weller BINF739

2.3 Subgraphs

- Cliques and Independent Sets
  - Def. – A subset S of V_G is called a clique if every pair of vertices in S is joined by at least one edge, and no proper superset of S has this property.
  - N.B. – Sometimes this property is defined without the maximality condition.
  - Def. – The clique number of a graph G is the number $\omega(G)$ of vertices in the largest clique in G.

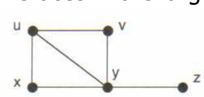


Figure 2.3.5 A graph with three cliques.

Structure and Representation
BINF739 Solka/Weller BINF739

2.3 Subgraphs

- Cliques and Independent Sets
  - Def. – A subset S of V_G is said to be an independent set if no pairs of vertices in S is joined by an edge. That is, S is a subset of mutually non-adjacent vertices of G.
  - Def. – The independence number of a graph G is the number $\alpha(G)$ of vertices in a largest independent set in G.

2.3 Subgraphs

- Induced Subgraphs
  - Def. – For a given graph G, the subgraph induced on a vertex subset U of V_G, denoted $G(U)$, is a subgraph of G whose vertex-set is U and whose edge-set consists of all edges in G that have both endpoints in U. That is,

$$V_{G(U)} = U \text{ and } E_{G(U)} = \{ e \in E_G \mid \text{endpts}(e) \subseteq U \}$$

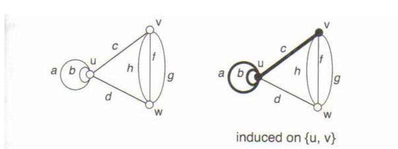


Figure 2.3.6 A subgraph induced on a subset of vertices.

2.3 Subgraphs

- Def. – For a given graph *G*, the subgraph induced on an edge subset *D* of E_G, denoted *G(D),* is the subgraph of *G* whose edge-set is *D* and whose vertex-set consists of all vertices that are incident with at least one edge in *D*. That is

$$V_{G(D)} = \{ v \in V_G \mid v \in \text{endpts}(e),\ \text{for some } e \in D \} \text{ and } E_{G(D)} = D$$

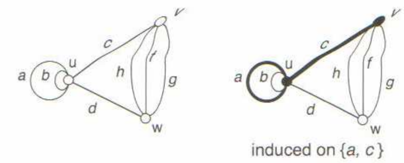


Figure 2.3.7 A subgraph induced on a subset of edges.

Structure and Representation
BINF739 Solka/Weller BINF739

2.3 Subgraphs

- Induced Subgraphs
  - Def. – The **center** of a graph G, denoted Z(G), is the subgraph induced on the set of central vertices of G.

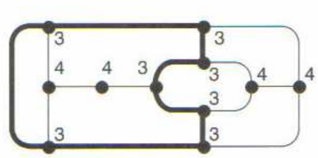


Figure 2.3.8 A graph whose center is a 7-cycle.

Structure and Representation
BINF739 Solka/Weller BINF739

2.3 Subgraphs

- Local Subgraphs
  - Def. – The **(open) local subgraph or (open) neighborhood subgraph** of a vertex v is the subgraph *L(v)* induced on the neighbors of *v*.

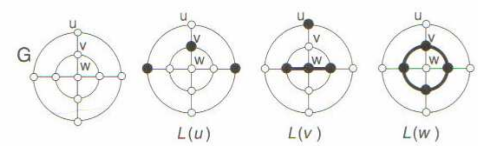


Figure 2.3.9 A graph G and three of its local subgraphs.

2.3 Subgraphs

- Local Subgraphs
  - Def. – Thm. 2.3.3 – Let $f : G \rightarrow H$ be a graph isomorphism and u be in V_G. Then f maps the local subgraph *L(u)* of G isomorphically to the local subgraph *L(f(u))* of *H*.

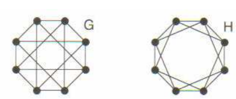


Figure 2.3.10 Two 4-regular 8-vertex graphs.

- The local subgraphs for graph G are all isomorphic to $4K_1$.The local subgraphs for graph H are all isomorphic to P_3. Hence the two graphs are not isomorphic. Also we note $\alpha(G) = 4$ but $\alpha(H) = 2$ and $\omega(G) = 2$ but $\omega(H) = 3$

2.3 Subgraphs

- Def. – A **component** of a graph *G* is a maximal connected subgraph of *G*. In other words, a connected subgraph *H* is a component of *G* if *H* is not a proper subgraph of any connected subgraph of *G*.

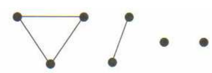


Figure 2.3.11 A graph with four components.

- Def. – In a graph , the component of a vertex *v*, denoted *C(v),* is the subgraph induced by the subset of all vertices reachable from *v*.

2.4 Some Graph Operations

- Deleting Vertices or Edges
  - Def. – If *v* is a vertex of a graph *G*, then the vertex-deletion subgraph G – v is the subgraph induced by the vertex set $V_G - \{v\}$.
  - Def. – If *e* is an edge of a graph *G*, then the edge-deletion subgraph *G – e* is the subgraph induced by the edge-set $E_G - \{e\}$.

2.4 Some Graph Operations

- Deleting Vertices or Edges

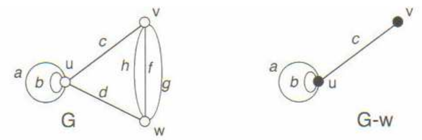


Figure 2.4.1 The result of deleting the vertex w from graph G.

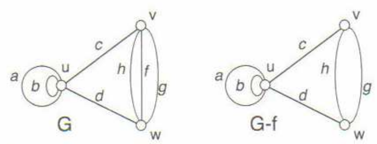


Figure 2.4.2 The result of deleting the edge f from graph G.

2.4 Some Graph Operations

- Network Vulnerability
  - Def. – A **vertex-cut** in a graph *G* is a vertex-set *U* such that *G-U* has more components than *G*.
  - A cut-vertex (or cutpoint) is a vertex-cut consisting of a single vertex.

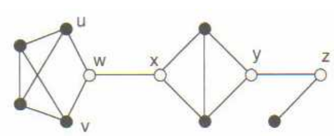


Figure 2.4.3 A graph with four cut-vertices.

Structure and Representation
BINF739 Solka/Weller BINF739

2.4 Some Graph Operations

- Network Vulnerability
  - Def. – An **edge-cut** in a graph *G* is a set of edges D such that *G-D* has more component than G.
  - Def. – A **cut-edge** (or bridge) is an edge-cut consisting of a single edge.

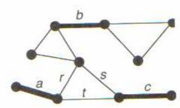


Figure 2.4.4 A graph with three cut-edges.

2.4 Some Graph Operations

- Network Vulnerability
  - Def. – An edge of a graph is called a **cycle-edge** if e lies in some cycle or the graph.
  - Prop. 2.4.1 – Let *e* be an edge of a connected graph *G*. Then *G-e* is connected iff e is a cycle-edge of *G*.
  - Prop. 2.4.2 – An edge of a graph is a cut-edge iff it is not a cycle-edge.

2.4 Some Graph Operations

- The Graph-Reconstruction Problem
  - Def. – Let *G* be a graph with $V_G = \{v_1, v_2, \ldots, v_n\}$. Then the **vertex-deletion subgraph** list of *G* is the list of the subgraphs $G\text{-}v_1, \ldots, G\text{-}v_n$
  - The reconstruction deck of a graph is its vertex-deletion subgraph list, with no labels on the vertices. We regard each individual vertex-deletion subgraph as being a card in the deck.

2.4 Some Graph Operations

- The Graph-Reconstruction Problem

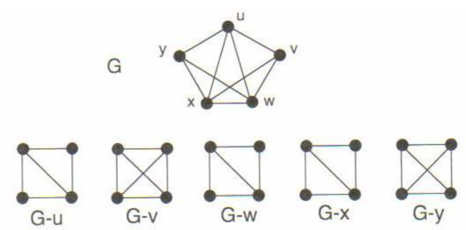


Figure 2.4.6 A graph and its vertex-deletion subgraph list.

2.4 Some Graph Operations

- The Graph-Reconstruction Problem
- Def. – The graph-reconstruction problem is to decide whether two non-isomorphic graphs with three or more vertices can have the same reconstruction deck.
- N. B. – For unlabeled graphs this is one of th foremost unsolved problems in graph theory.

Figure 2.4.7 Is G the only graph with this reconstruction deck?

2.4 Some Graph Operations

- Adding Edges or Vertices
  - Def. – Adding an edge between two vertices *u* and *w* of a graph *G*, means creating a supergraph, denoted *G U {e}*, with vertex-set V_G and edge-set E_G *U {e}* where *e* is a new edge with endpoints *u* and *w*.

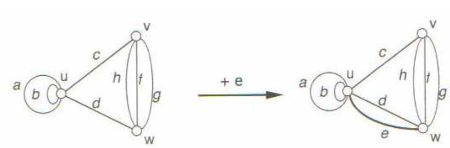


Figure 2.4.8 Adding an edge e with endpoints u and w.

Structure and Representation
BINF739 Solka/Weller BINF739

2.4 Some Graph Operations

- Adding Edges or Vertices
  - Def. – Adding a vertex to a graph *G,* where *v* is a new vertex not already in V_G, means creating a supergraph, denoted $G \cup \{v\}$, with vertex-set

    $V_G \cup \{v\}$ and edge-set E_G.

2.4 Some Graph Operations

- Graph Union
  - Def. – The graph union of two graphs $G = (V, E)$ and $G' = (V', E')$ is the graph $G \cup G'$ whose vertex-set and edge-set are the disjoint unions, respectively, of the vertex sets and edge-sets of G and G'.

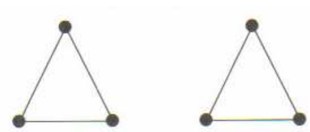


Figure 2.4.9 The graph union $K_3 \cup K_3$ of two copies of K_3.

2.4 Some Graph Operations

- Joining a Vertex to a Graph
  - Def. – If a new vertex *v* is joined to each of the pre-existing vertices of a graph *G*, then the resulting graph is called the join of *v* to *G* or the suspension of *G* from *v*, and is denoted *G + v*.

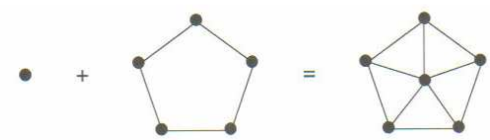


Figure 2.4.10 The 5-wheel $W_5 = K_1 + C_5$.

2.4 Some Graph Operations

- Edge- Complementation
  - Der. – Let G be a simple graph. Its edge-complement (or complement) $\overline{G}$ is the graph on the same vertex set , such that two vertices are adjacent in $\overline{G}$ iff the are not adjacent in G.

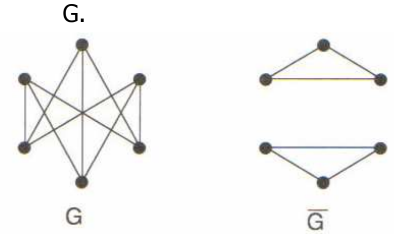


Figure 2.4.12 A graph and its complement.

2.5 Tests for Non-Isomorphisms

- Def. – A **graph invariant** (or digraph invariant) is a property of graphs (digraphs) that is preserved by isomorphism.
  - Some examples of such properties include number of vertices, the number of edges, and the degree sequence
  - Thm. 2.5.1 – Let *f : G → H* be a graph isomorphism, and let *v* be in V_G. Then the multiset of degrees of the neighbors of *v* equals the multiset of degrees of the neighbors of *f(v).*

2.5 Tests for Non-Isomorphisms

- A Local Invariant

Figure 2.5.1 Non-isomorphic graphs with the same degree sequence.

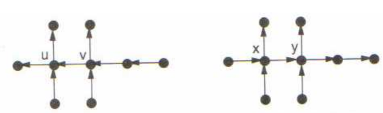


Figure 2.5.2 Non-isomorphic digraphs with identical degree sequences.

2.5 Tests for Non-Isomorphisms

- Distance Invariants
  - Thm. 2.5.2 – The isomorphic image of a graph walk is a walk of the same length.

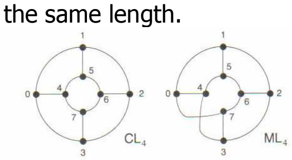


Figure 2.5.3 Non-isomorphic graphs with the same degree sequence.

To reduce the calculation of diameter from checking all $\binom{8}{2} = 28$ vertex pairs to checking the maximum distance from any one vertex, we first establish that both graphs are vertex-transitive. For the circular ladder we observe the symmetries of rotation and the isomorphism that swaps the inner cycle with the outer cycle. For the Möbius ladder, we observe that $j \mapsto j + 1 \mod 8$ is an automorphism, whose iteration establishes vertex-transitivity.

The maximum distance from vertex 0 in CL_4 is 3, for vertex 6. The maximum distance from vertex 0 in ML_4 is 2. Thus, they are not isomorphic. We may also observe that $\alpha(CL_4) = 4$, but $\alpha(ML_4) = 3$.

2.5 Tests for Non-Isomorphisms

- Subgraph Presence
  - Thm. 2.5.6 – For each graph-isomorphic type, the number of distinct subgraphs in a graph having that isomorphism type is a graph invariant.

Example 2.5.5: The five graphs in Figure 2.5.4 are all 3-regular, even though they have the same degree sequence. Whereas A and C have no K_3 subgraphs, B has two, D has four, and E has one. Thus, Theorem 2.5.6 implies that the only possible isomorphic pair is A and C. However, graph C has a 5-cycle, but graph A does not, because it is bipartite.

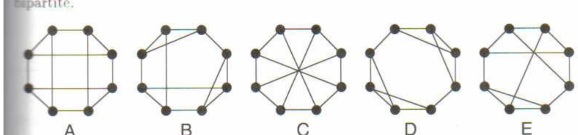


Figure 2.5.4 Five mutually non-isomorphic, 8-vertex, 3-regular graphs.

Alternatively, we may observe that graphs A and C are the only pair with the same multiset of local subgraphs. We could distinguish this pair by observing that $\alpha(A) = 4$ and $\alpha(C) = 3$.

2.5 Tests for Non-Isomorphisms

- Edge- Complementation
  - Thm. 2.5.7 – Let G and H both be simple graphs. They are isomorphic iff their edge-complements are isomorphic.

Example 2.5.6: The two graphs in Figure 2.5.5 are relatively dense, simple graphs (both with 20 out of 28 possible edges). The edge-complement of the left graph consists of two disjoint 4-cycles, and the edge-complement of the right graph is an 8-cycle. Since these edge-complements are non-isomorphic, the original graphs must be non-isomorphic.

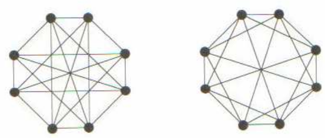


Figure 2.5.5 Two relatively dense, non-isomorphic 5-regular graphs.

Structure and Representation
BINF739 Solka/Weller BINF739

2.5 Tests for Non-Isomorphisms

Table 2.5.1 Some graph invariants.

1. The number of vertices
2. The number of edges
3. The degree sequence
4. The multiset of local graphs
5. Degrees of neighbors of a forced match
6. Diameter, radius, girth
7. Independence number, clique number
8. For any possible subgraph, the number of distinct copies
9. For a simple graph, the edge-complement

Structure and Representation
BINF739 Solka/Weller BINF739

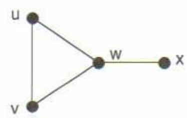
2.6 Matrix Representations

Adjacency Matrices

DEFINITION: The ***adjacency matrix of a simple graph*** G, denoted A_G, is the symmetric matrix whose rows and columns are both indexed by identical orderings of V_G, such that

$$A_G[u,v] = \begin{cases} 1 & \text{if } u \text{ and } v \text{ are adjacent} \\ 0 & \text{otherwise} \end{cases}$$

Example 2.6.1: Figure 2.6.1 shows the adjacency matrix of a graph, with respect to the vertex ordering u, v, w, x.



$$A_G = \begin{array}{c} \\ u \\ v \\ w \\ x \end{array} \begin{array}{c} \begin{array}{cccc} u & v & w & x \end{array} \\ \begin{pmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \end{array}$$

Figure 2.6.1 A graph and its adjacency matrix.

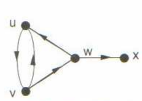
2.6 Matrix Representations

- Adjacency Matrices
  - Prop. 2.6.1 - Let. G be a graph with adjacency matrix A_G. Then the value of element $A^r_G[u,v]$ of the r-th power of matrix A_G equals the number of u-v walks of length r.

DEFINITION: The ***adjacency matrix of a simple digraph*** D, denoted A_D, is the matrix whose rows and columns are both indexed by identical orderings of V_G, such that

$$A_D[u,v] = \begin{cases} 1 & \text{if there is a edge from } u \text{ to } v \\ 0 & \text{otherwise} \end{cases}$$

Example 2.6.2: The adjacency matrix of the digraph in Figure 2.6.2 uses the vertex ordering u, v, w, x.



$$A_D = \begin{array}{c} \\ u \\ v \\ w \\ x \end{array} \begin{array}{c} \begin{array}{cccc} u & v & w & x \end{array} \\ \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix} \end{array}$$

Figure 2.6.2 A digraph and its adjacency matrix.

2.6 Matrix Representations

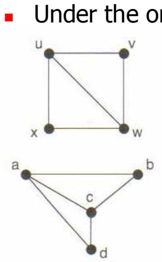
- Adjacency Matrices

Proposition 2.6.2. *Let D be a digraph with $V_D = v_1, v_2, \ldots, v_n$. Then the sum of the elements of row i of the adjacency matrix A_D equals the outdegree of vertex v_i, and the sum of the elements of column j equals the indegree of vertex v_j.* ◇ (Exercises)

Proposition 2.6.3. *Let D be a digraph with adjacency matrix A_D. Then the value of the entry $A_D^r[u,v]$ of the r^{th} power of matrix A_D equals the number of directed u-v walks of length r.* ◇ (Exercises)

2.6 Matrix Representations

- Brute-Force-Graph-Isomorphism-Testing
  - Under the orderings *u, v, w, x* and *a, d, c, b*



$$A_G = \begin{array}{c} \\ u \\ v \\ w \\ x \end{array} \begin{array}{c} \begin{array}{cccc} u & v & w & x \end{array} \\ \begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{pmatrix} \end{array}$$

$$A_H = \begin{array}{c} \\ a \\ d \\ c \\ b \end{array} \begin{array}{c} \begin{array}{cccc} a & d & c & b \end{array} \\ \begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{pmatrix} \end{array}$$

Figure 2.6.3 Establishing isomorphism using adjacency matrices.

2.6 Matrix Representations

- Brute-Force-Graph-Isomorphism-Testing

Algorithm 2.6.1: Brute-force test for graph isomorphism

Input: graphs G and H.
Output: Return YES or NO, according to whether G is isomorphic to H.

```
If |V_G| ≠ |V_H|
    Return NO.
If degree sequences are not equal
    Return NO.
Fix a vertex ordering for graph G.
Write the adjacency matrix A_G with respect to that ordering.
For each vertex ordering τ of graph H
    Write A_H with respect to ordering τ
    If A_H(w.r.t. τ) = A_G
        Return YES.
Return NO.
```

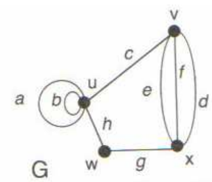

2.6 Matrix Representations

- Incidence Matrices for Undirected Graphs
  - Def. – The incidence matrix for a graph G is the matrix I_G whose rows and columns are indexed by some ordering of V_G and E_G respectively such that

$$I_G[v,e] = \begin{cases} 0 \text{ if } v \text{ is not an endpoint of } e \\ 1 \text{ if } v \text{ is an endpoint of } e \\ 2 \text{ if } e \text{ is a self-loop at } v\ ^{\dagger} \end{cases}$$

2.6 Matrix Representations

- Incidence Matrices for Undirected Graphs



$$I_G = \begin{array}{c} \\ u \\ v \\ w \\ x \end{array} \begin{array}{c} \begin{array}{cccccccc} a & b & c & d & e & f & g & h \end{array} \\ \begin{pmatrix} 2 & 2 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \end{pmatrix} \end{array}$$

Figure 2.6.4 A graph and its incidence matrix.

2.6 Matrix Representations

- Incidence Matrices for Undirected Graphs
  - Prop. 2.6.4 – The sum of the entries in the row of an incidence matrix is the degree of the corresponding vertex.
  - Prop. 2.6.5 – The sum of the entries in any column of an incidence matrix is equal to 2.

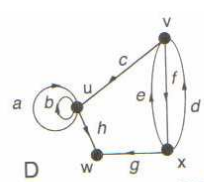
2.6 Matrix Representations

- Incidence Matrix for Digraphs
  - Def. – The incidence matrix for a digraph D is the matrix IG whose rows and columns are indexed by some ordering of V_D and E_D respectively such that

$$I_D[v,e] = \begin{cases} 0 \text{ if } v \text{ is not an endpoint of } e \\ 1 \text{ if } v \text{ is the head of } e \\ -1 \text{ if } v \text{ is the tail of } e \\ 2 \text{ if } e \text{ is a self-loop at } v \end{cases}$$

2.6 Matrix Representations

- Incidence Matrix for Digraphs

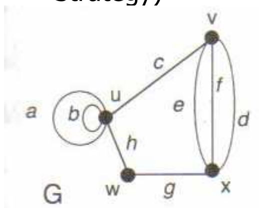


$$I_D = \begin{array}{c} \\ u \\ v \\ w \\ x \end{array} \begin{array}{c} \begin{array}{cccccccc} a & b & c & d & e & f & g & h \end{array} \\ \begin{pmatrix} 2 & 2 & 1 & 0 & 0 & 0 & 0 & -1 \\ 0 & 0 & -1 & 1 & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & -1 & -1 & 1 & -1 & 0 \end{pmatrix} \end{array}$$

Figure 2.6.5 A digraph and its incidence matrix.

2.6 Matrix Representations

- Table of Incident Edges (a More Efficient Storage Strategy)



$$I_{V:E}(G) = \begin{array}{lllll} \underline{u}: & a & b & c & h \\ \underline{v}: & c & d & e & f \\ \underline{w}: & g & h & & \\ \underline{x}: & d & e & f & g \end{array}$$

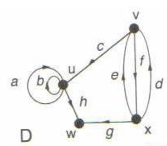
2.6 Matrix Representations

- Comparing the storage requirements of the three storage strategies
  - Incidence matrix O(|V| - |E|)
  - Adjacency matrix O($|V|^2$)
  - Table of incident edges O(|E|)

2.6 Matrix Representations

- Table of outgoing and incoming arcs

$$in_{V:E}(G) = \begin{array}{llll} \underline{u}: & a & b & c \\ \underline{v}: & d & e & \\ \underline{w}: & g & h & \\ \underline{x}: & f & & \end{array} \qquad out_{V:E}(G) = \begin{array}{llll} \underline{u}: & a & b & h \\ \underline{v}: & c & f & \\ \underline{w}: & & & \\ \underline{x}: & d & e & g \end{array}$$



$$I_D = \begin{array}{c} \\ u \\ v \\ w \\ x \end{array} \begin{array}{c} \begin{array}{cccccccc} a & b & c & d & e & f & g & h \end{array} \\ \begin{pmatrix} 2 & 2 & 1 & 0 & 0 & 0 & 0 & -1 \\ 0 & 0 & -1 & 1 & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & -1 & -1 & 1 & -1 & 0 \end{pmatrix} \end{array}$$

Figure 2.6.5 A digraph and its incidence matrix.

2.7 More Graph Operations

- Left to the Reader