Chap 2 Structure and Representation

Yih-Lang Li (李毅郎)
Computer Science Department
National Chiao Tung University, Taiwan

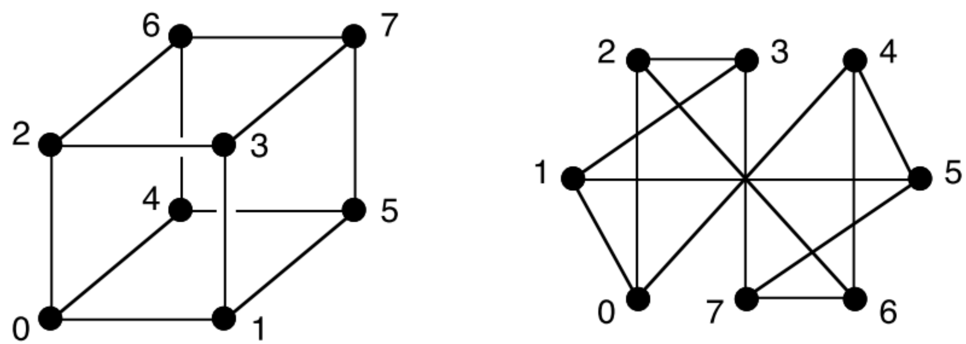
The sources of most figure images are from the course slides (Graph Theory) of Prof. Gross

Outline

- Graphs Isomorphism
- Automorphisms and Symmetry
- Subgraphs
- Some Graph Operations
- Tests for Non-Isomorphism
- Matrix Representaiton
- More Graph Operations

2.1 Graph Isomorphism – Structurally Equivalent Graphs

- **EXAMPLE 2.1.1.**
  - ✓ The same vertex sets
  - ✓ The same adjacency table

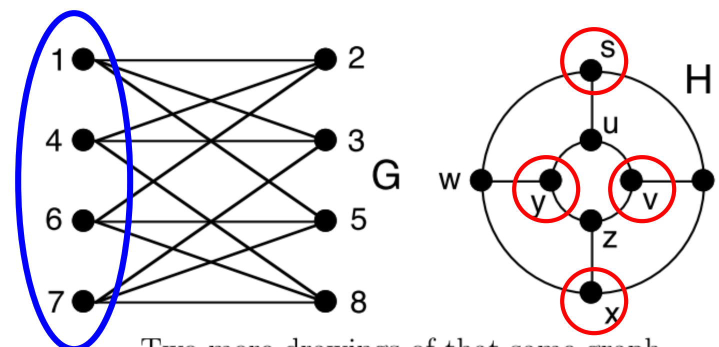


0.	1	2	4
1.	0	3	5
2.	0	4	6
3.	1	2	7
4.	0	5	6
5.	1	4	6
6.	2	4	7
7.	3	5	6

Two different drawings of the same graph.

- **EXAMPLE 2.1.2.**
  - ✓ Different vertex sets
  - ✓ A bijection function f maps V_G to V_H
  - ✓ $1 \to s$, $2 \to t$, $3 \to u$, $4 \to v$, $5 \to w$, $6 \to x$, $7 \to y$, $8 \to z$
  - ✓ Neighborhood also bijectively maps to neighborhood
  - ✓ $N(1) \mapsto N(f(1)) = N(s)$

    $\{2, 3, 5\} \mapsto \{t, u, w\}$



Two more drawings of that same graph.

Formalizing Structural Equivalence for Simple Graphs

- **DEFINITION**: Let G and H be two simple graphs. A vertex bijection $f : V_G \to V_H$ ***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.

- **DEFINITION**: A vertex bijection $f : V_G \to V_H$ 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 pair of vertices in G,

  u and v are adjacent in $G \Leftrightarrow f(u)$ and $f(v)$ are adjacent in H.

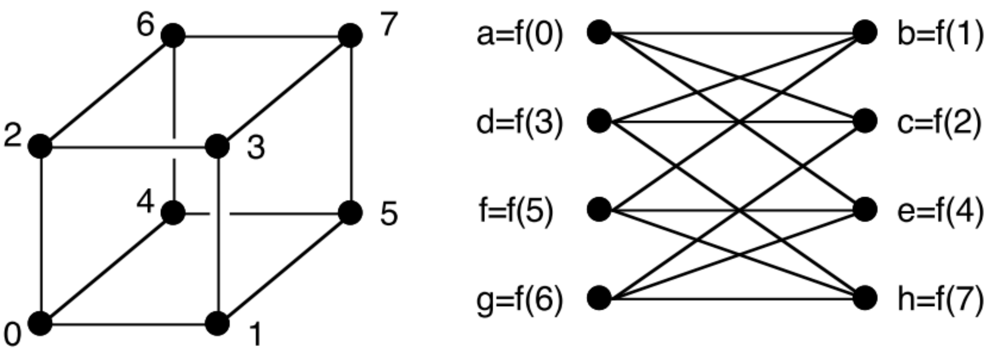
- **DEFINITION**: Two simple graphs G and H are isomorphic, denoted $G \cong H$, if

  $\exists$ a structure-preserving bijection $f : V_G \to V_H$. Such a function f between (the vertex-sets of) G and H is called an ***isomorphism*** from G to H.

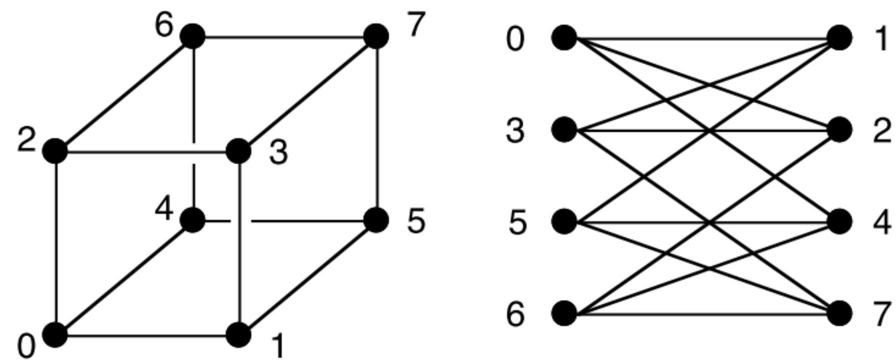
- **NOTATION**: When we think of a vertex function $f : V_G \to V_H$ as a mapping from one graph to another, we may write $f : G \to H$.

Formalizing Structural Equivalence for Simple Graphs

- Two ways to depict an isomorphism



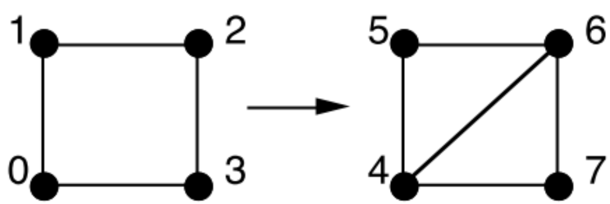
Specifying an isom betw two simple graphs.



Another way of depicting an isomorphism.

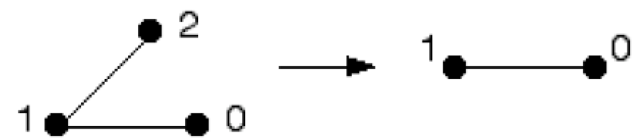
- Linear graph mapping – not required to preserve non-adjacency & not necessarily bijection
- Two non-isomorphism examples

$j \mapsto j+4$



Bijective and adj-preserving, but not an isom.

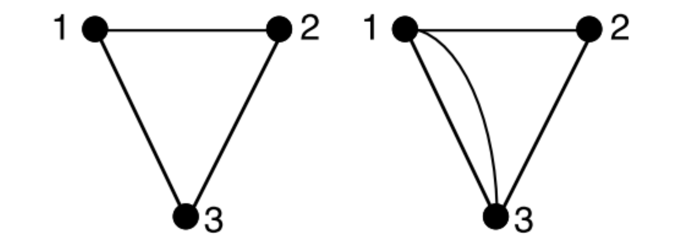
$j \mapsto j \bmod 2$



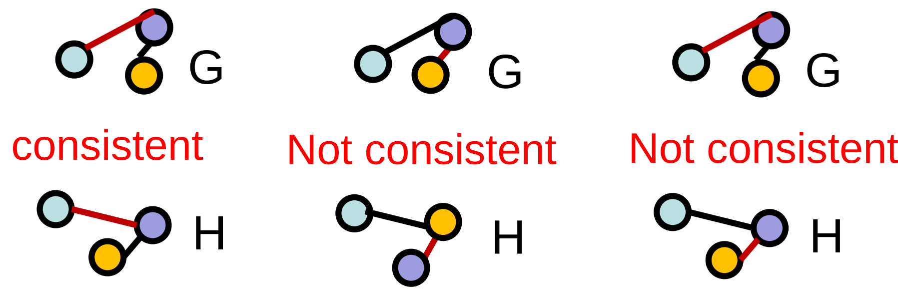
Preserves adj and non-adj, but not bijective.

Extending the Definition of Isomorphism to General Graphs

$f(i) = i$



These graphs are not structurally equivalent.

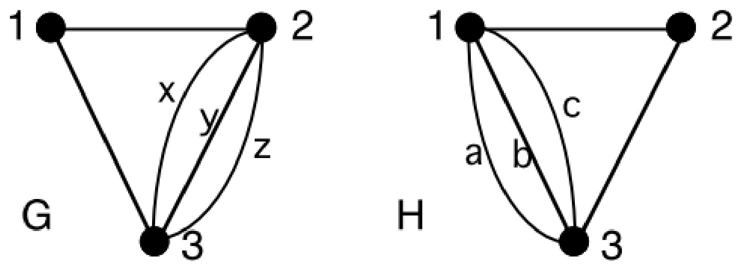


- **DEFINITION**: A vertex bijection $f : V_G \to V_H$ between two graphs G and H, simple or general, is structure-preserving if
  - (1) the # of edges (even if 0) between every pair of distinct vertices u and v in graph G equals the # of edges between their images $f(u)$ and $f(v)$ in graph H, and
  - (2) the # of self-loops at each vertex x in G equals the # of self-loops at the vertex $f(x)$ in H.
- **DEFINITION**: Two graphs G and H (simple or general) are isomorphic graphs if
  $\exists$ structure-preserving vertex bijection $f : V_G \to V_H$. This relationship is denoted $G \cong H$.
- **DEFINITION**: For isomorphic graphs G and H, a pair of bijections $f_V : V_G \to V_H$ and $f_E : E_G \to E_H$ is ***consistent***
  if for every edge $e \in E_G$, the function f_V maps the endpoints of e to the endpoints of the edge $f_E(e)$.

  $(f_V : V_G \to V_H, f_E : E_G \to E_H)$ is often written as $f : G \to H$

Isomorphism for Graphs with Multi-Edges

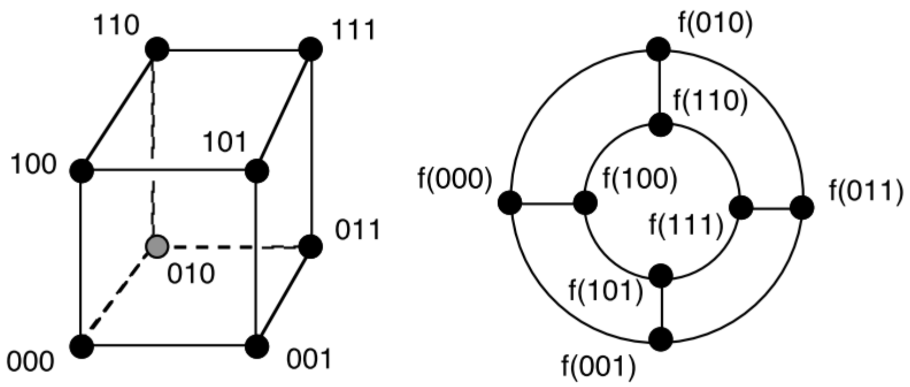
- **Proposition 2.1.1.** *Let G and H be any two graphs. Then $G \cong H$ if and only if there is a vertex bijection $f_V : V_G \to V_H$ and an edge bijection $f_E : E_G \to E_H$ that are consistent.*
- **Remark.** If G and H are isomorphic simple graphs, then every structure-preserving vertex bijection $f : V_G \to V_H$ induces a unique consistent edge bijection, implicitly given by the rule: $uv \mapsto f(u)f(v)$.
- **DEFINITION**: If G and H are graphs with multi-edges, then an ***isomorphism*** from G to H is specified by giving a vertex bijection $f_V : V_G \to V_H$ and an edge bijection $f_E : E_G \to E_H$ that are consistent.
- **Example.**
  - ✓ **What are 12 distinct isomorphisms?**



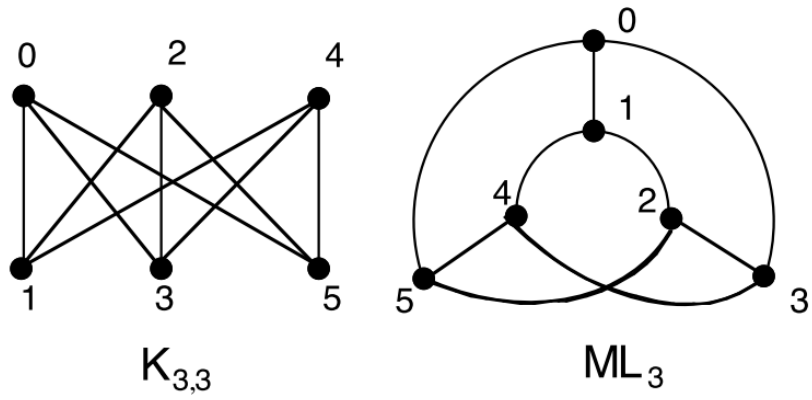
There are 12 distinct isoms from G to H.

Isomorphic Graph Pairs

- **Theorem 2.1.2.** *Let G and H be isomorphic graphs. Then they must have the same number of vertices and the same number of edges.*
- **Theorem 2.1.3.** *Let $f : G \to H$ be a graph isomorphism and let $v \in V_G$. Then $deg(f(v)) = deg(v)$.*
- **Corollary 2.1.4.** *Let G and H be isomorphic graphs. Then they have the same degree sequence.*
- **Corollary 2.1.5.** *Let $f : G \to H$ be a graph isomorphism and let $e \in E_G$. Then the endpoints of edge $f(e)$* have the same degrees of the endpoints of *e*.
- **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.



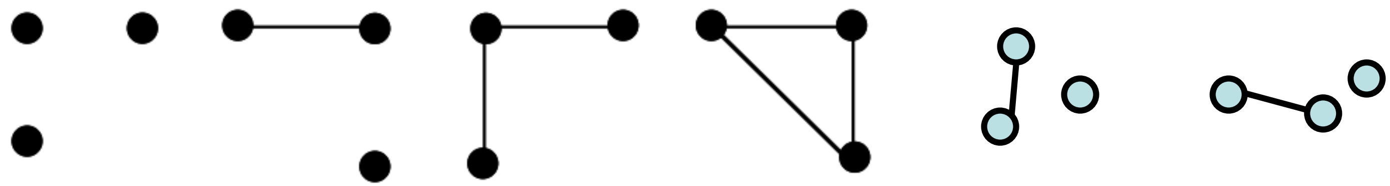
Hypercube Q_3 and circ ladder CL_4 are isom.



$K_{3,3}$ and the Möbius ladder ML_3 are isom.

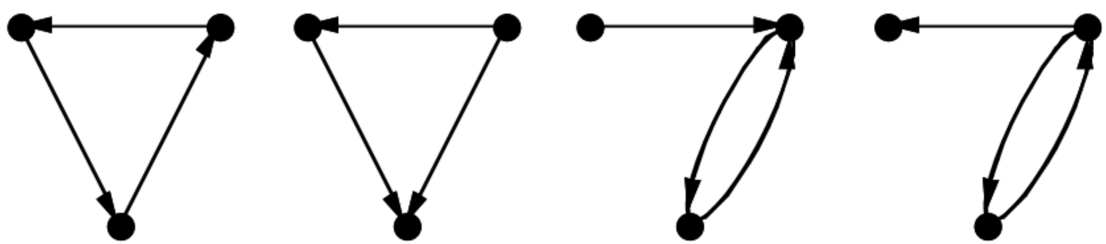
Isomorphism Type of a graph & Isomorphism Digraphs

- If $f = (f_V, f_E)$ is an isomorphsim from G to H, then $f^{-1} = (f_V^{-1}, f_E^{-1})$ is an isomorphism from H to G.
  - ✓ the relation "isomorphic to" is an equivalence relation. (*symmetric*, *reflexive*, *transitive*)
- **DEFINITION**: Each equivalent class under $\cong$ is called an ***isomorphism type***.



The 4 isom types for a simple 3-vertex graph.

- **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 $f(v)$.



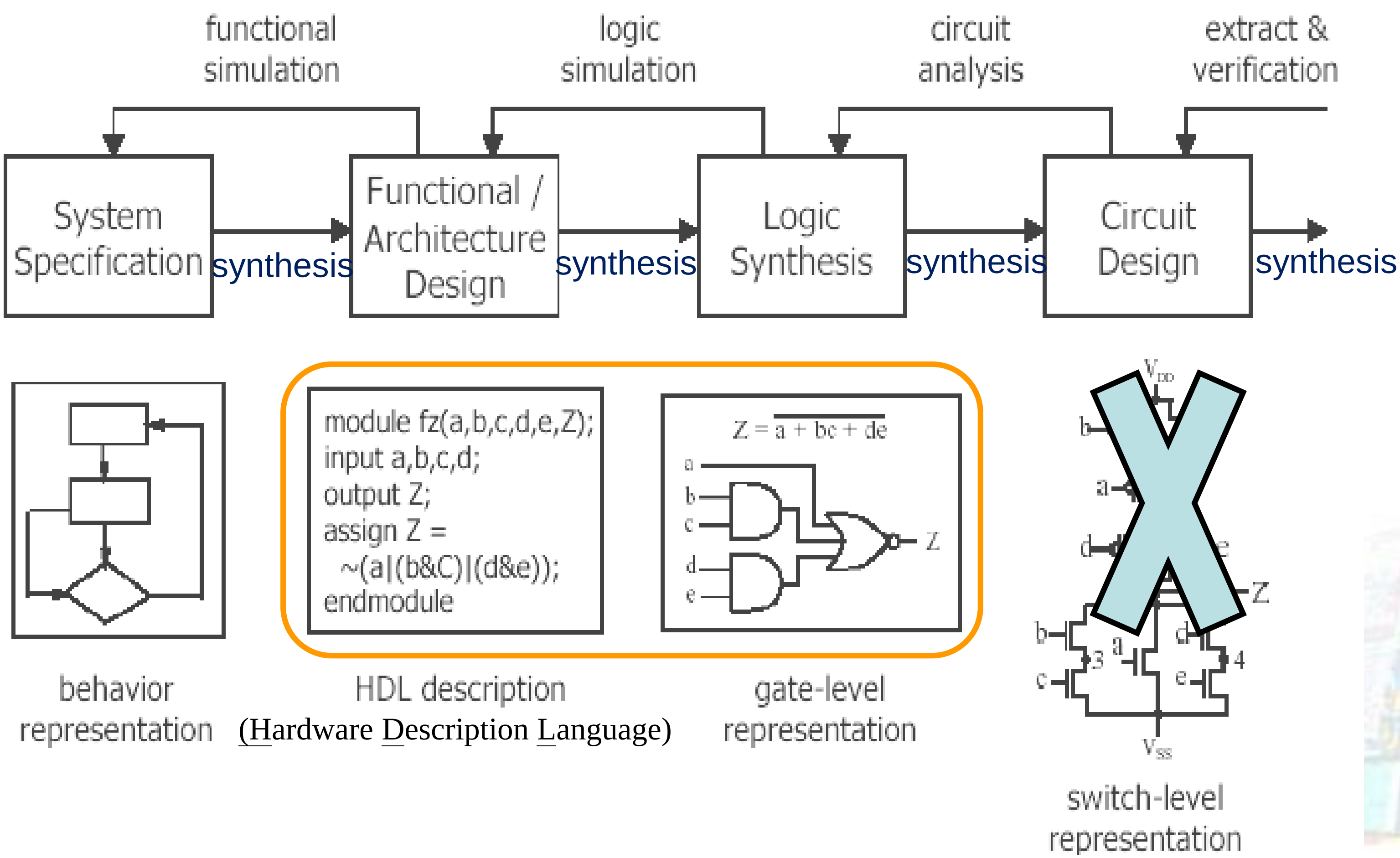
Four non-isomorphic digraphs.

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

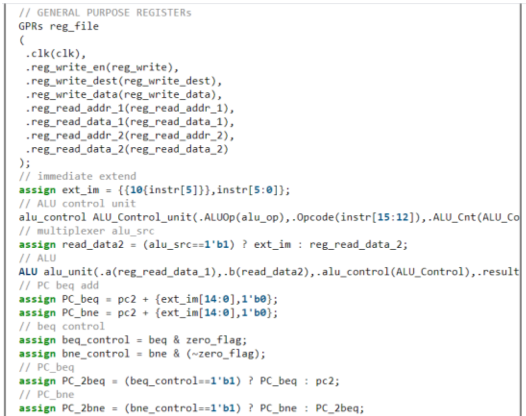
VLSI Design Flow

- The application of graph isomorphism to VLSI design (chip design)

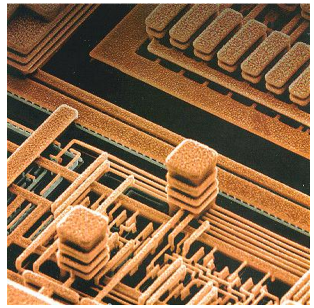
  Very **L**arge **S**caled **I**ntegrated Circuit



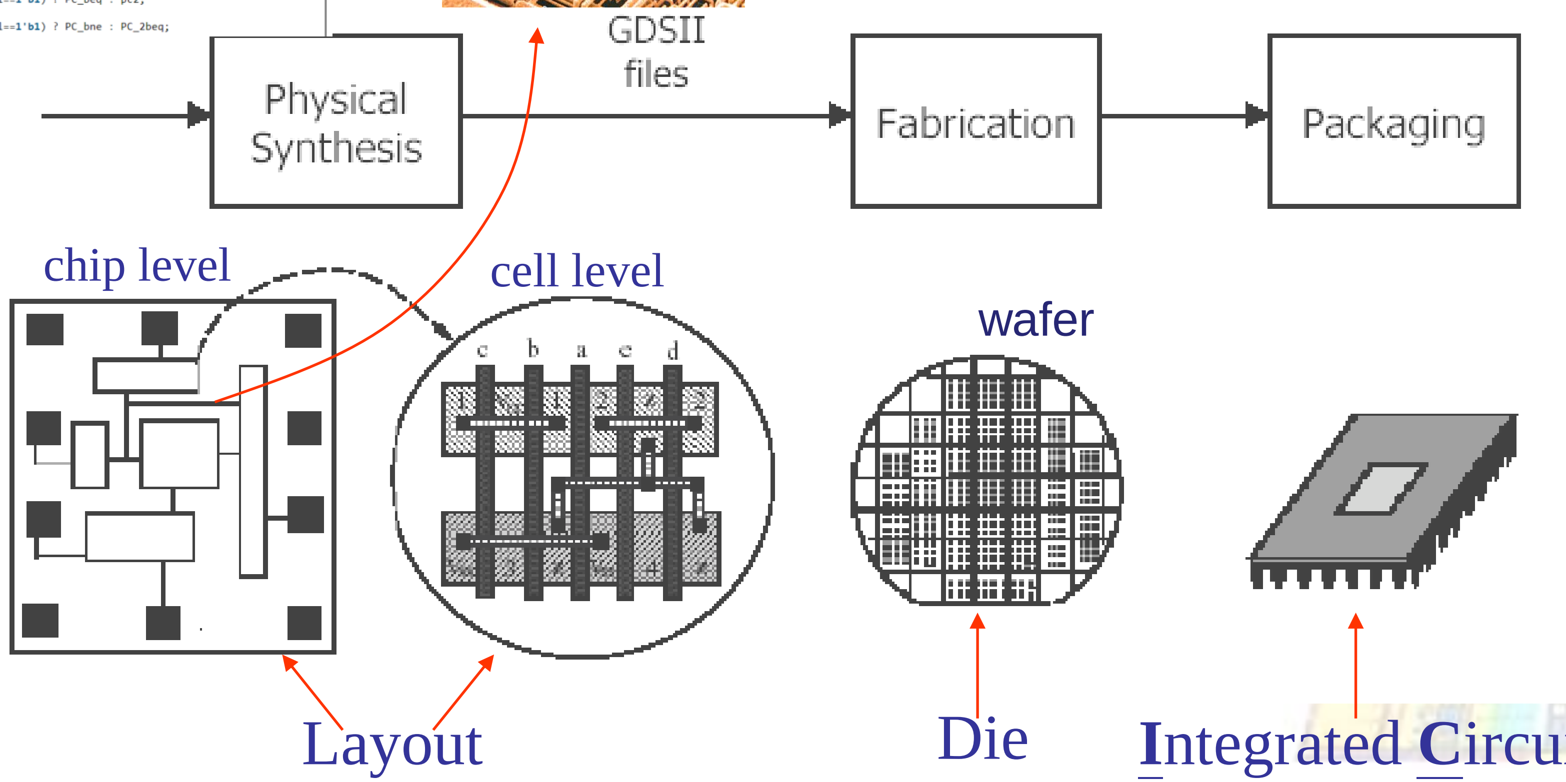
VLSI Circuit Design Flow



```
// GENERAL PURPOSE REGISTERs
GPRs reg_file
(
 .clk(clk),
 .reg_write_en(reg_write),
 .reg_write_dest(reg_write_dest),
 .reg_write_data(reg_write_data),
 .reg_read_addr_1(reg_read_addr_1),
 .reg_read_data_1(reg_read_data_1),
 .reg_read_addr_2(reg_read_addr_2),
 .reg_read_data_2(reg_read_data_2)
);
// immediate extend
assign ext_im = {{10{instr[5]}},instr[5:0]};
// ALU control unit
alu_control ALU_Control_unit(.ALUOp(alu_op),.Opcode(instr[15:12]),.ALU_Cnt(ALU_Co
// multiplexer alu_src
assign read_data2 = (alu_src==1'b1) ? ext_im : reg_read_data_2;
// ALU
ALU alu_unit(.a(reg_read_data_1),.b(read_data2),.alu_control(ALU_Control),.result
// PC beq add
assign PC_beq = pc2 + {ext_im[14:0],1'b0};
assign PC_bne = pc2 + {ext_im[14:0],1'b0};
// beq control
assign beq_control = beq & zero_flag;
assign bne_control = bne & (~zero_flag);
// PC_beq
assign PC_2beq = (beq_control==1'b1) ? PC_beq : pc2;
// PC_bne
assign PC_2bne = (bne_control==1'b1) ? PC_bne : PC_2beq;
```



Describe the geometries in each layer and in a hierarchical way.
Chip (program) → top modules (main functions in ***main***)
→ bottom modules (functions) → cells → transistors
→ Geometrical shapes (most in Manhattan shapes)



Chip Layout Views

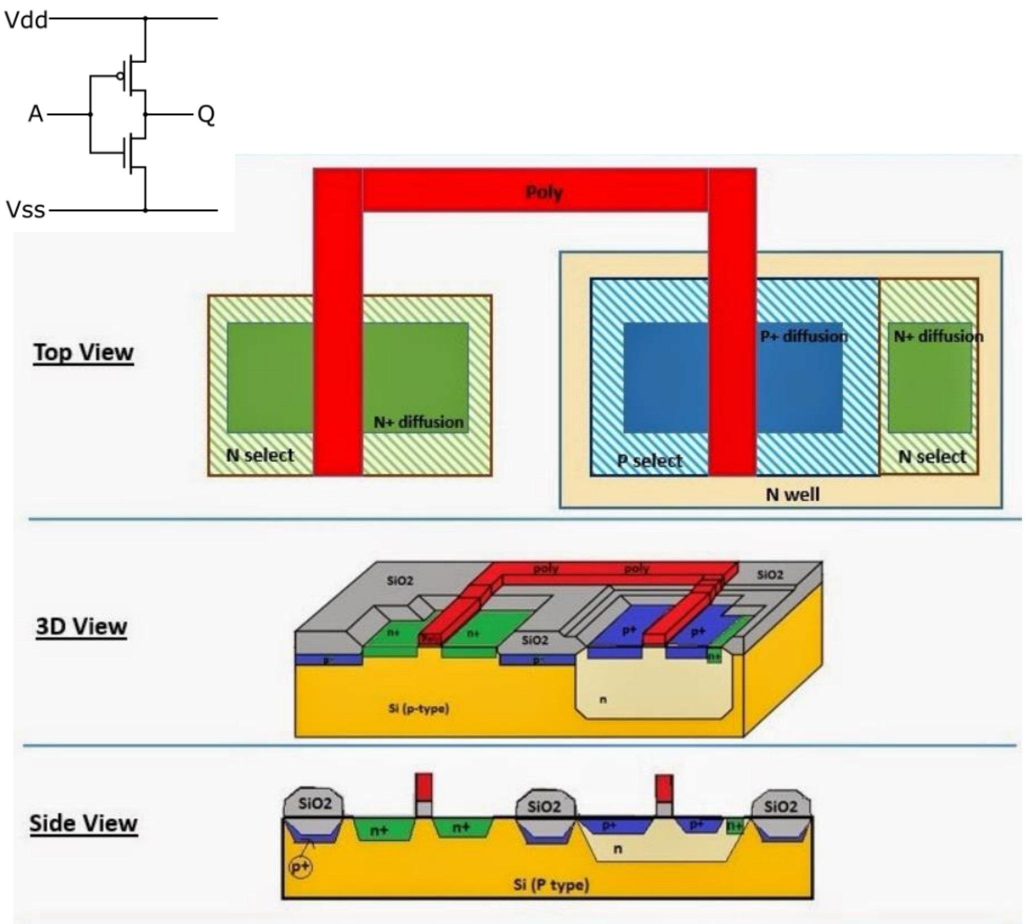
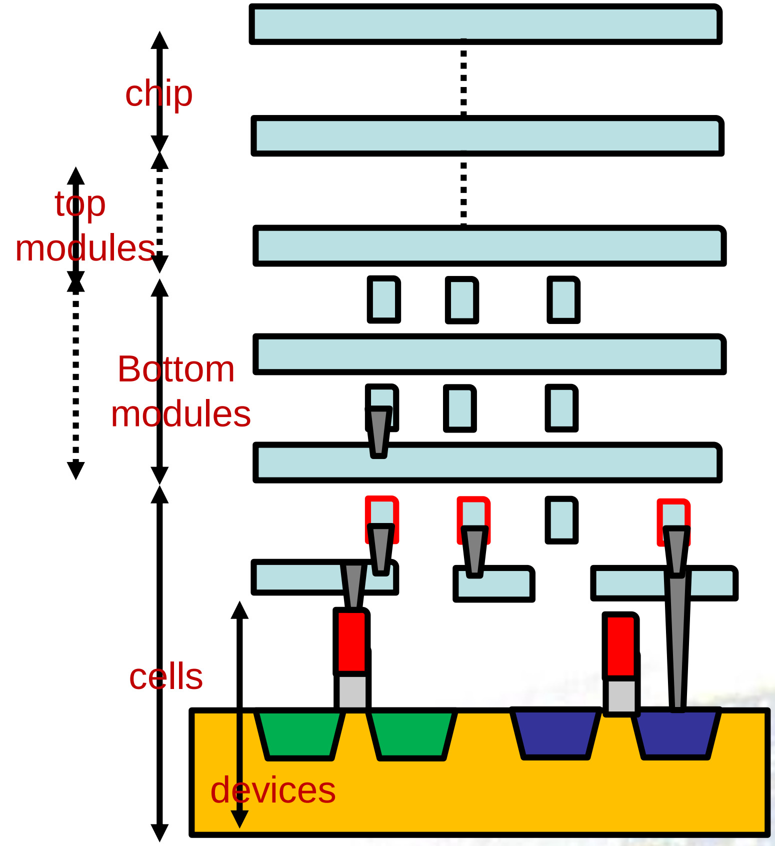
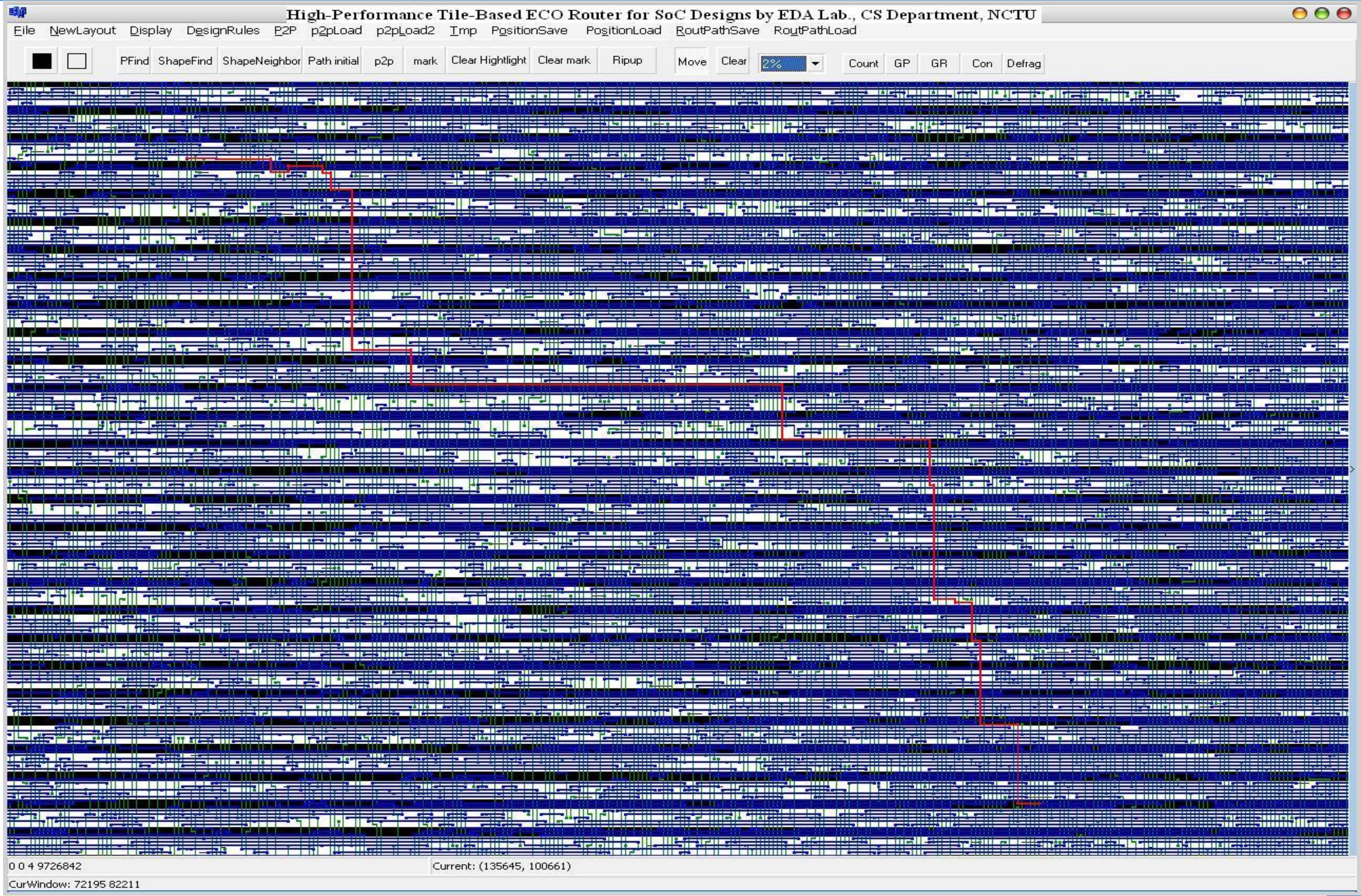


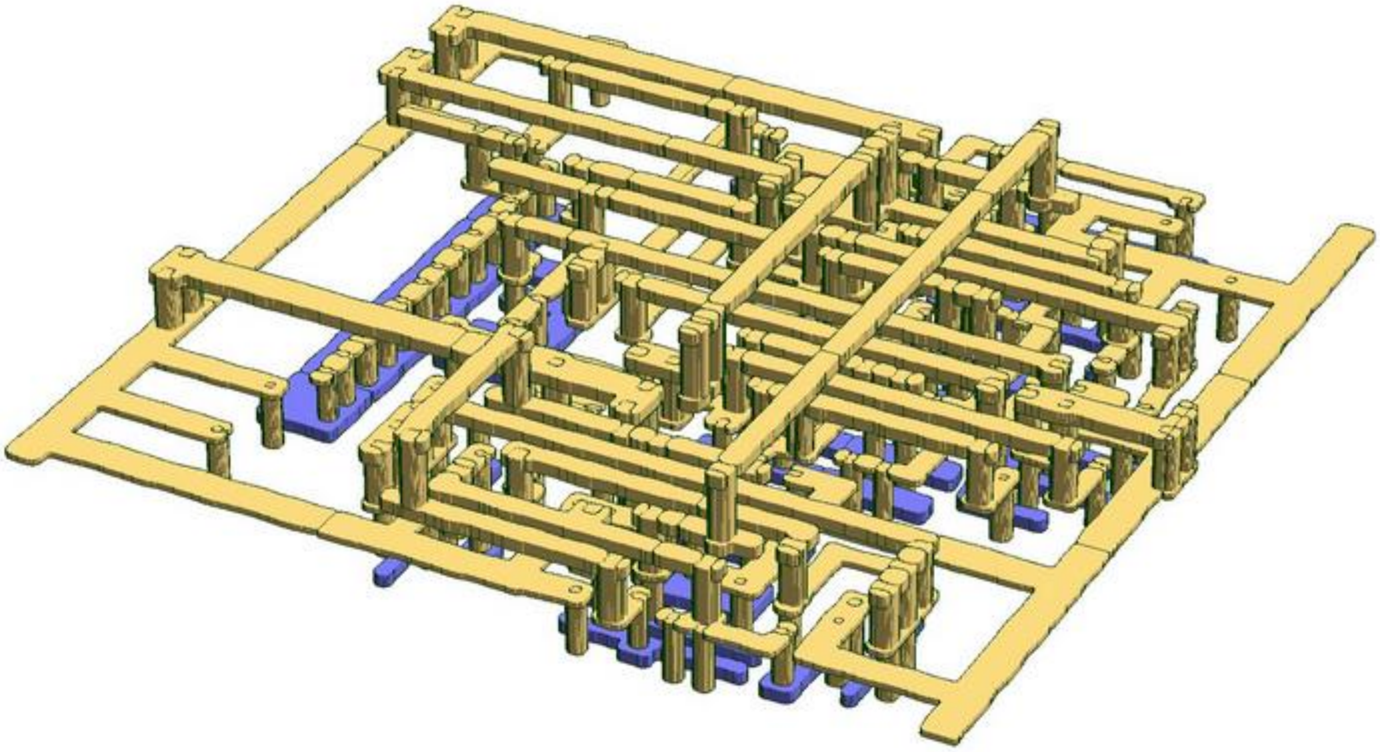
Figure source: http://www.vlsi-expert.com/2014/11/cmos-layout-design.html



NCTU CS-EDA Lab Router

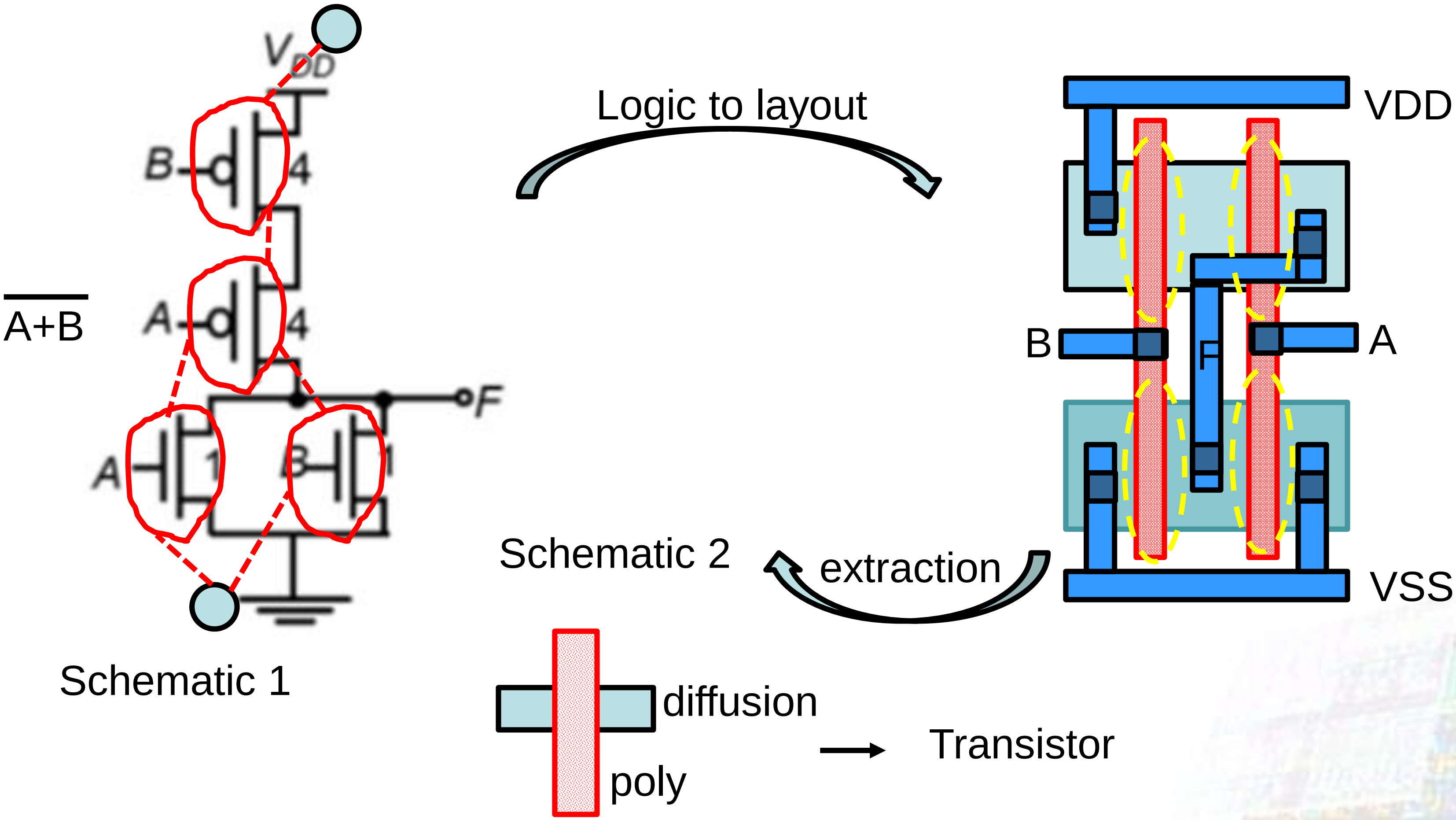


Layout vs. Schematic



Verification between Two Design Levels

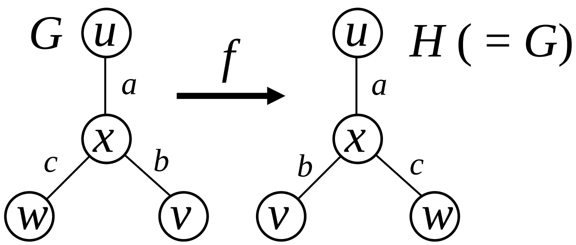
- Layout versus Schematic (LVS) checking – backward verify the correctness of synthesis from schematic to layout



Is schematic 1 equivalent to schematic 2? Graph isomorphism problem

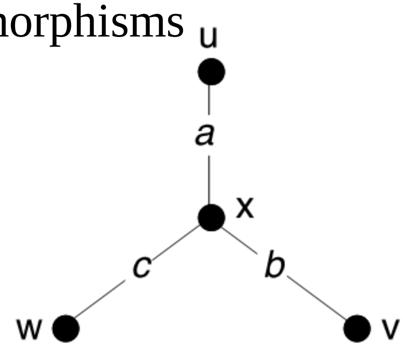
2.2 Automorphisms and Symmetry

- **DEFINITION**: An isomorphism from a graph G to itself is called an ***automorphism***.
- **Remark:** Any structure-preserving vertex-permutaion is associated with one (if simple) or more (if there are any multi-edges) automorphism of G. The proportion of vertex-permutation of V_G that are structure-preserving is a measure of the *symmetry* of G.
- The most convenient representation of a permutation is as a *product of disjoint cycles*.
- **NOTATION**: (x), $(x_i\ x_j)$, $(x_0\ x_1\ \ldots\ x_{n-1})$
- **Example:** The permutation $\pi = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\ 7 & 4 & 1 & 8 & 5 & 2 & 9 & 6 & 3 \end{pmatrix}$



has the ***disjoint cycle form*** $\pi = (1\ 7\ 9\ 3)(2\ 4\ 8\ 6)(5)$

- **Geometric symmetry** – A geometric symmetry on a graph drawing can be used to represent an automorphism on the graph.
- **Example.** $K_{1,3}$ has six automorphisms
  - ✓ Why not 4!=24?

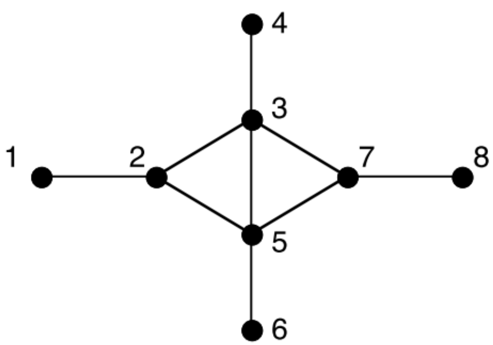


The graph $K_{1,3}$.

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	$(x)\ (w)\ (u\ v)$	$(c)\ (a\ b)$

Limitations of Geometric Symmetry

- When a graph is highly symmetric, automorphisms can be obtained by reflecting and rotating graph.



Automorphisms

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

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

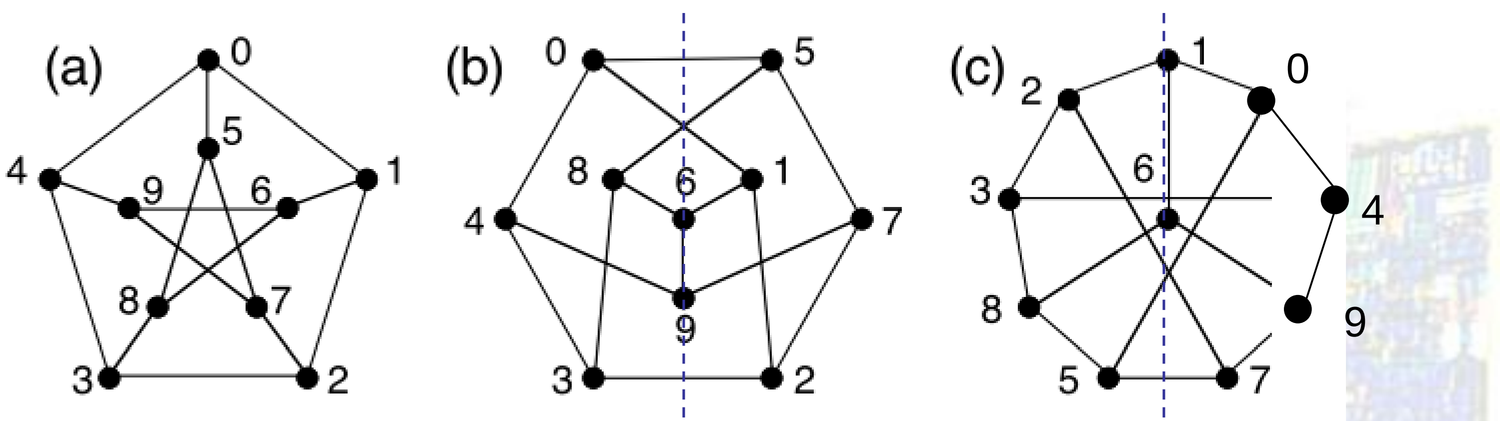
$\lambda_2 = (1)(2)(3\ 5)(4\ 6)(7)(8)$

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

A graph with four automorphisms.

Limitations of Geometric Symmetric

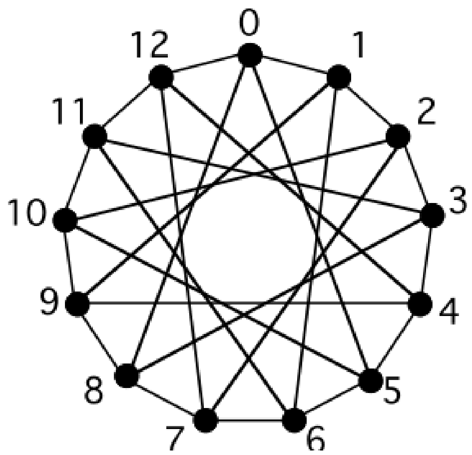
- Three drawings of the Petersen graph. Symmetry property : (a) (5-fold) > (b) and (c) (2-fold)
  - ✓ (0 1 2 3 4) (5 6 7 8 9) for (a) , (0 5) (1 8) (4 7) (2 3) (6) (9) for (b), similar mapping for (c)



Three drawings of the Petersen graph.

Vertex- and Edge-Transitive Graphs

- **DEFINITION**: 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.
- **DEFINITION**: 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.
- **Example.** $K_{1,3}$ is edge-transitive, but not vertex-transitive, since every automorphism must map the 3-valent vertex to itself.
- **Example.** The complete graph K_n is edge-transitive and vertex-transitive for every n.
- **Example.** The hypercube graph Q_n is vertex-transitive and edge-transitive for every n.
- **Example.** The Petersen graph is vertex-transitive and edge-transitive.
- **Example.** Every circulant graph $circ(n; S)$ is vertex-transitive.
  - ✓ vertex function $i \mapsto i + k \mod n$ is an automorphism.
  - ✓ map i to j : $k=abs(j-i)$.

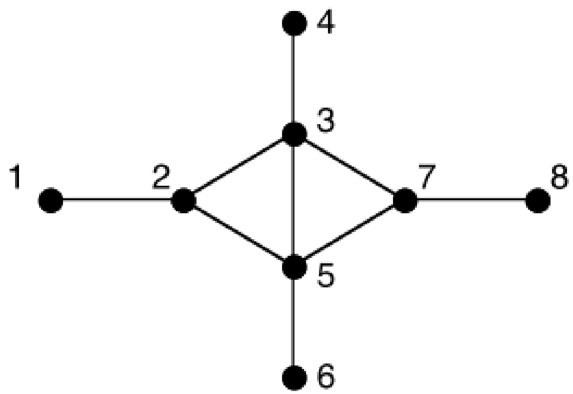


The circulant graph $circ(13 : 1, 5)$.

Vertex Orbits and Edge Orbits

- **DEFINITION**: 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

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

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

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

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

Graph of Example 2.3.

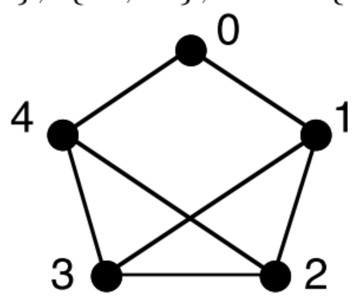
- **Theorem 2.2.1.** *All vertices in the same orbit have the same degree.*
- **Theorem 2.2.2.** *All edges in the same orbit have the same pair of degrees at their endpoints.*
- **Remark:** A vertex-transitive graph is a graph with only one vertex orbit, and an edge-transitive graph is a graph with only one edge orbit.
- **Example.** Complete graph K_n has only one vertex orbit and one edge orbit.
- **Example.** Each of the two partite sets of the complete bipartite graph $K_{m,n}$ is a vertex orbit. The graph is vertex-transitive only if $m=n$. However, $K_{m,n}$ is always edge-transitive.

How to Find an Orbit

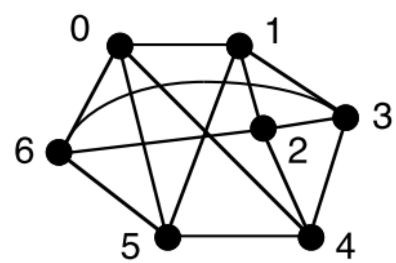
- It is not known whether there exists a polynomial-time algorithm for finding orbits.
- We can use Theorems 2.2.1. & 2.2.2. plus the following observation – if an automorphism maps vertex *u* to vertex *v*, then it maps the neighbors of *u* to the neighbors of *v*.
- Middle figure: We can simply see (0 5) (1 4) (2) (3) (6) is an automorphism from the symmetry.
  - ✓ **Further observation** – vertices 0, 2, 3 and 5 have common three neighbors (1, 4, 6) that are independent, which means vertices 0, 2, 3, and 5 are in an orbit.
  - ✓ **Further observation** – vertices 1, 4, and 6 each have two pairs of adjacent vertices (2, 3) and (0, 5), which means we can redraw the middle figure as the right one.
  - ✓ Final vertex orbit: {0, 2, 3, 5} and {1, 4, 6}. Edge orbit: {05, 23}, {the other edges}.

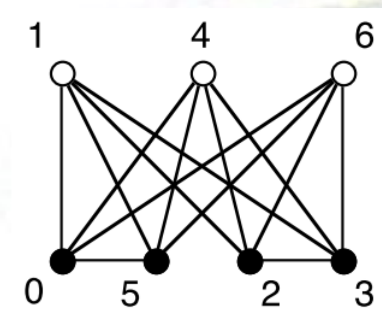
vertex orbits $\{0\}$, $\{1, 4\}$, and $\{2, 3\}$

edge orbits $\{23\}$, $\{01, 04\}$, and $\{12, 13, 24, 34\}$



Find the vertex orbits and the edge orbits.

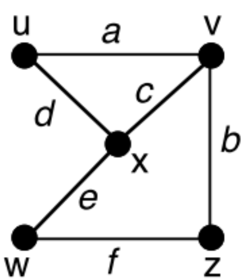




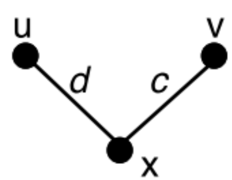
2.3 Subgraphs

- Properties of a given graph are often determined by the existence or non-existence of certain types of smaller graphs inside it.
  - ✓ Theorem 1.5.3 asserts that a graph is bipartite if and only if it contains no odd cycle.
- **DEFINITION**: A ***subgraph*** of a graph G is a graph H whose vertices and edges are all in G. If H is subgraph of G, we may also say that G is a ***supergraph*** of H.
- **DEFINITION**: A ***subdigraph*** of a digraph G is a digraph H whose vertices and arcs are all in G.
- The incident table for a subgraph H can be obtained simply by deleting from the incident table of G each column that does not correspond to an edge in H.
- **DEFINITION**: A ***proper*** subgraph 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.
  - ✓ "*proper*" means non-empty as well as not total set.

G

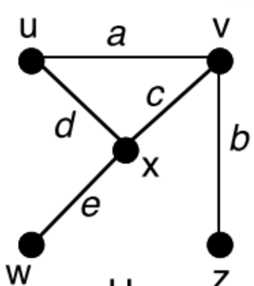


edge	a	b	c	d	e	f
endpts	u v	v z	v x	u x	x w	w z



H_1

edge	c	d
endpts	v x	u x

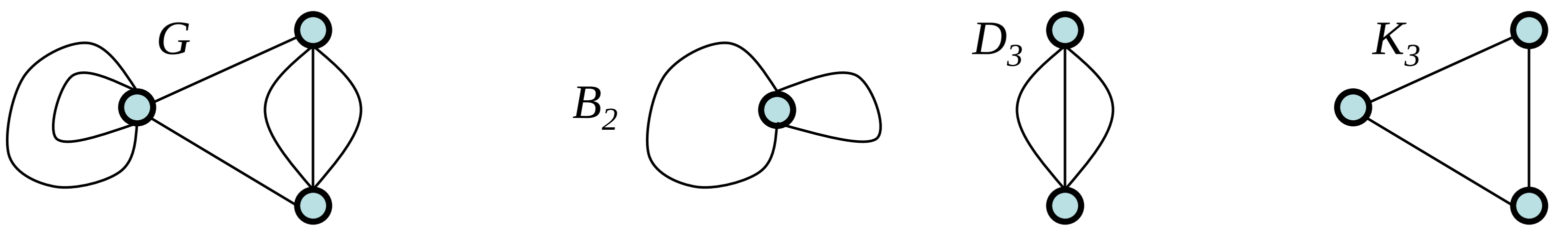


H_2

edge	a	b	c	d	e
endpts	u v	v z	v x	u x	x w

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.
- **Example.** The graph G of Figure 2.3.1 contains as subgraphs exactly one copy each of C_3, C_4, and C_5.
- **Example.** A graph with n vertices is hamiltonian if and only if it contains a graph isomorphic to the *n-cycle* C_n.
- **Example.** The graph G shown in the left has among its subgraphs a B_2 (bouquet), a D_3 (dipole), and three copies of K_3.

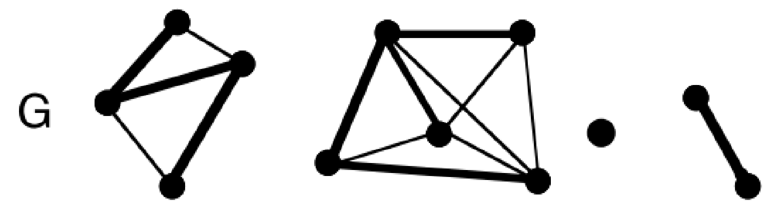


- **DEFINITION**: A subgraph H is said to ***span*** a graph G if $V_H = V_G$.
- **DEFINITION**: A ***spanning tree*** is a spanning subgraph that is a tree.

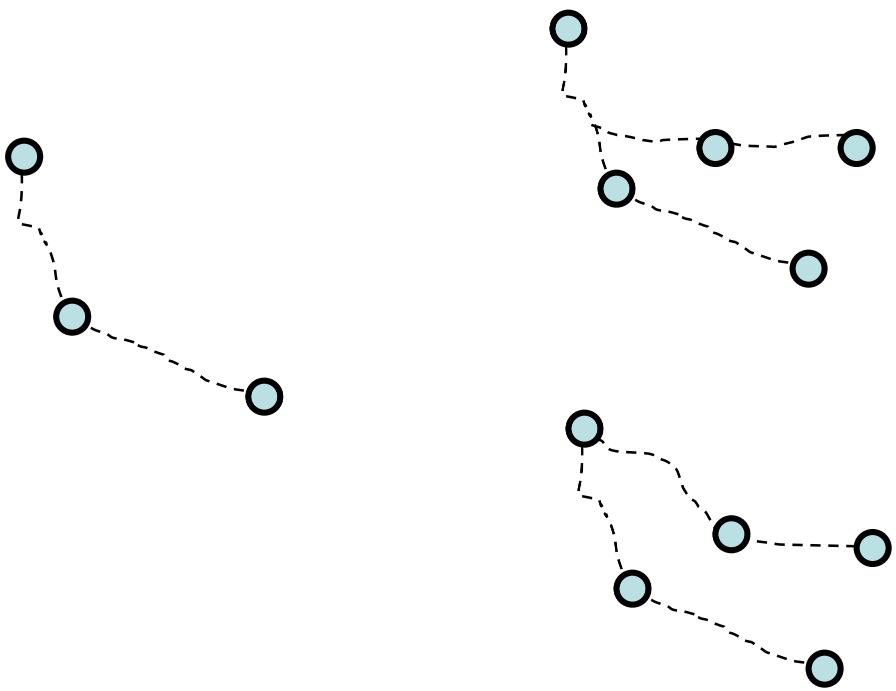
A spanning tree.

Spanning Subgraph, Cliques

- **Proposition 2.3.1.** A graph is connected if and only if it contains a spanning tree.
- **Proposition 2.3.2.** Every acyclic subgraph of a connected graph G is contained in at least one spanning tree of G.
- **DEFINITION**: An acyclic graph is called a ***forest***.

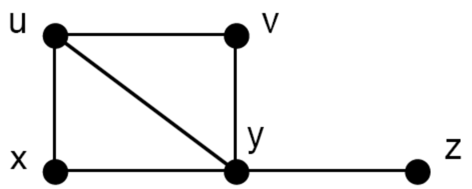


A spanning forest H of graph G.



Independent Sets

- **DEFINITION:** 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.
  - ✓ A clique of a graph G is a maximal subset of mutually adjacent vertices in G.
  - ✓ In other books, clique may not be required to be a maximal subset.
- **DEFINITION**: The ***clique number*** of a graph G is the number $\omega(G)$ of vertices in a largest clique in G.
- **Example.** $\{u, v, y\}$, $\{u, x, y\}$, $\{y, z\}$ but not $\{u, y\}$



A graph with three cliques.

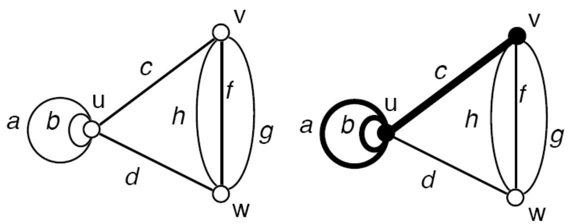
- **DEFINITION**: A subset S of V_G is said to be an ***independent set*** if no pair of vertices in S is joined by an edge.
- **DEFINITION**: The ***independence number*** of a graph G is the number $\alpha(G)$ of vertices in a largest independent set in G.
- **Remark:** the concepts of clique and independent set is complementary.

Induced Subgraphs

- **DEFINITION**: For a given graph G, the ***subgraph induced on a vertex subset*** U of V_G, denoted $G(U)$, is the 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,

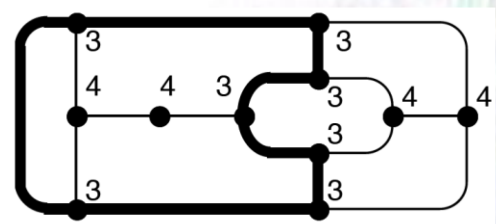
$$V_{G(U)} = U \qquad and \qquad E_{G(U)} = \{e \in E_G \mid endpts(e) \subseteq U\}$$

- **Example 2.3.9.** The subgraph induced on a vertex set U of a clique is a complete graph.
- **Example 2.3.10.**
- **DEFINITION**: For a given graph G, the ***subgraph induced on an edge subset*** D of E_G, denoted $G(D)$ is $V_{G(D)} = \{v \in V_G \mid v \in endpts(e), \, for\ some\ e \in D\} \qquad and \qquad E_{G(D)} = D$
- **DEFINITION:** The center of a graph G, denoted $Z(G)$, is the subgraph induced on the set of central vertices of G (see Sec.1.4).
- **Example.** The right figure with minimum eccentricity of 3



induced on $\{u, v\}$ (vertices) and $\{a, c\}$ (edges)

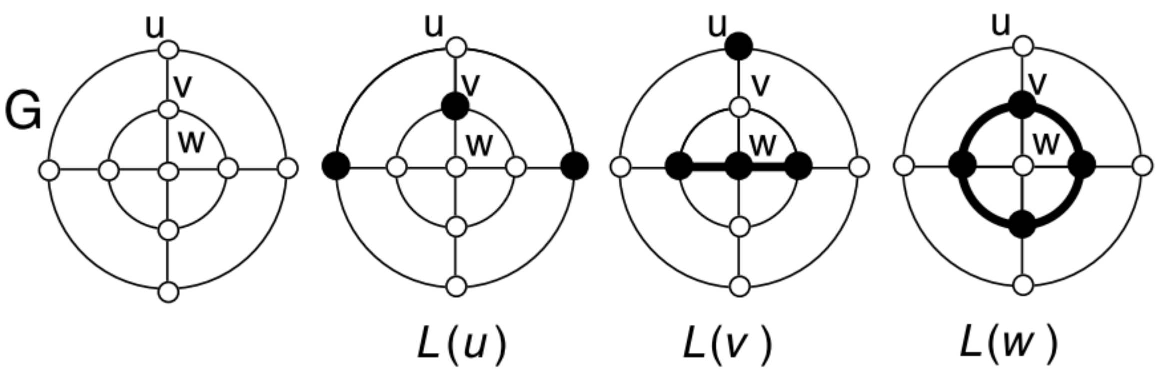
A subgraph induced on a subset of vertices and edges



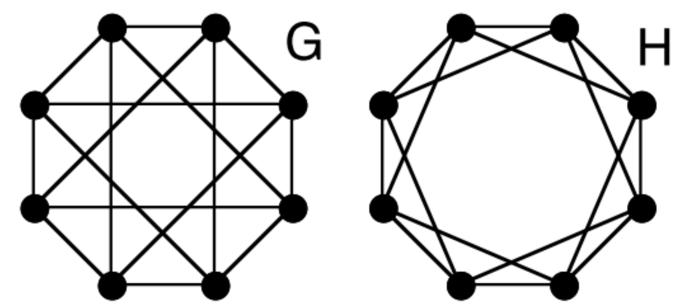
A graph whose center is a 7-cycle.

Local Subgraphs

- **DEFINITION**: The ***(open) local subgraph*** (or ***(open) neighborhood subgraph***) of a vertex v is the subgraph $L(v)$ induced on the neighbors of v.
- **Theorem 2.3.3.** Let $f : G \to H$ be a graph isom and $u \in V_G$. Then f maps the local subgraph $L(u)$ of G isomorphically to the local subgraph $L(f(u))$ of H.
- **Example.** Thus G and H are not isomorphic.
  - ✓ All local subgraphs of G are all isomorphic to $4K_1$. All local subgraphs of H are isomorphic to P_4.
  - ✓ $\alpha(G) = 4$ but $\alpha(H) = 2$, $\omega(G) = 2$ but $\omega(H) = 3$.



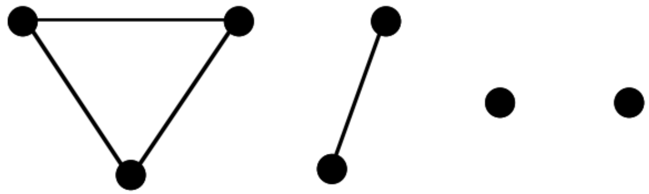
A graph G and three of its local subgraphs.



Two 4-regular 8-vertex graphs.

Components

- **DEFINITION**: A ***component*** of a graph G is a maximal connected subgraph of G.
  - ✓ The only component of a connected graph is the entire graph.
  - ✓ Membership in the same component is an *equivalence relation* on V_G, called the *reachability relation*.
- **Review §1.4.** A vertex v is said to be reachable from vertex u if there is a walk from u to v.
- **DEFINITION**: In a graph G, the ***component of a vertex*** v, denoted $C(v)$, is the subgraph induced by the subset of all vertices reachable from v.
- **ALTERNATIVE DEFINITION**: A ***component*** of a graph G is a subgraph induced by an equivalence class of the reachability relation on V_G.
- **Notation**: The # of components of a graph G is denoted $c(G)$.
- **COMPUTATIONAL NOTE**: Partitioning a small graph into components is trivial. But larger graphs that are specified by some computer representation require a computer algorithm. These "component-finding" algorithms are developed in Chapter 4 as straightforward applications of our graph-traversal procedures.

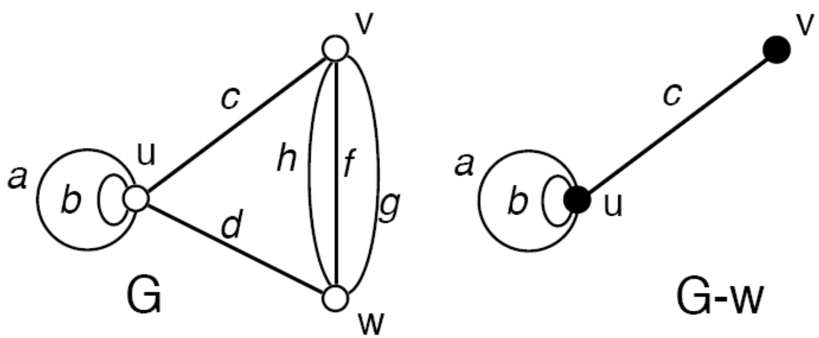


A graph with four components.

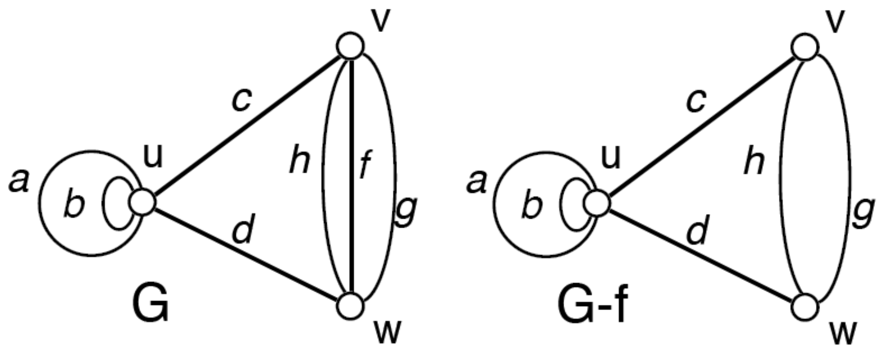
2.4 Some Graph Operations

- **DEFINITION**: The ***vertex-deletion subgraph*** $G - v$ is the subgraph induced by the vertex-set $V_G - \{v\}$. That is, $V_{G-v} = V_G - \{v\}$ *and* $E_{G-v} = \{e \in E_G \mid v \notin endpts(e)\}$
- **DEFINITION**: The ***edge-deletion subgraph*** $G - e$ is similar to the subgraph induced by the edge-set $E_G - \{e\}$ except that V_{G-e} is the same as V_G. That is,

$V_{G-e} = V_G$ *and* $E_{G-e} = E_G - \{e\}$



The result of deleting vertex w from graph G.

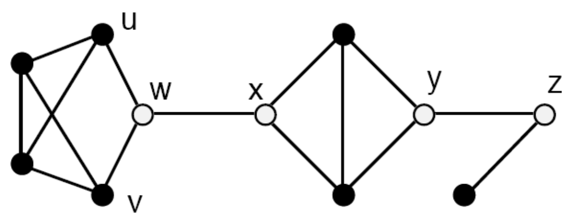


The result of deleting edge f from graph G.

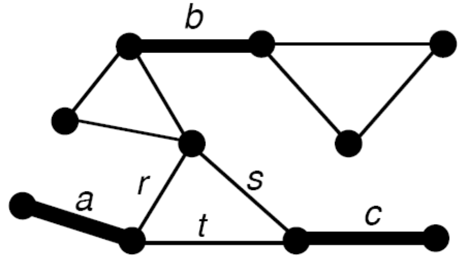
Network Vulnerability

- **DEFINITION**: A ***vertex-cut*** in a graph G is a vertex-set U such that $G - U$ has more components than G.
- **DEFINITION**: A ***cut-vertex*** (or ***cutpoint***) is a vertex-cut consisting of a single vertex.
- **DEFINITION**: An ***edge-cut*** in a graph G is a set of edges D such that $G - D$ has more components than G.
- **DEFINITION**: A ***cut-edge*** (or ***bridge***) is an edge-cut consisting of a single edge.
- **DEFINITION**: An edge e of a graph is called a ***cycle-edge*** if e lies in some cycle of that graph.
- **Proposition 2.4.1.** *Let e be an edge of a connected graph G. Then G – e is connected if and only if e is a cycle-edge of G.*
- **Corollary 2.4.2.** *An edge of a graph is a cut-edge if and only if it is not a cycle-edge.*
- **Corollary 2.4.3.** *Let e be any edge of a graph G. Then*

$$c(G-e)=\begin{cases} c(G), & \text{if } e \text{ is a cycle-edge} \\ c(G)+1, & \text{otherwise} \end{cases}$$



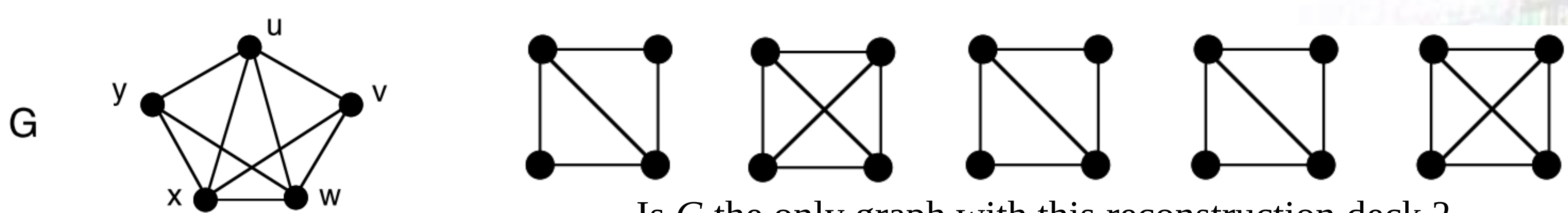
A graph with four cut-vertices.



A graph with three cut-edges.

The Graph-Reconstruction Problem

- **DEFINITION**: 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 $G - v_1, \ldots, G - v_n$
- **DEFINITION**: The ***reconstruction deck*** of G 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.
- **DEFINITION**: The ***graph-reconstruction problem*** is to decide whether two non-isomorphic simple graphs with three or more vertices can have the same reconstruction deck.
- **Remark:** The graph-reconstruction problem would be easy to solve if the vertex-deletion subgraphs included the vertex and edge names.
- ***Reconstruction Conjecture:*** *Let G and H be two graphs with $V_G = \{v_1, v_2, \ldots, v_n\}$ and $V_H = \{w_1, w_2, \ldots, w_n\}$ with $n \geq 3$, and with the same reconstruction deck, i.e., such that $G - v_i \cong H - w_i$, for each $i = 1, \ldots, n$. Then $G \cong H$.*



Is G the only graph with this reconstruction deck ?

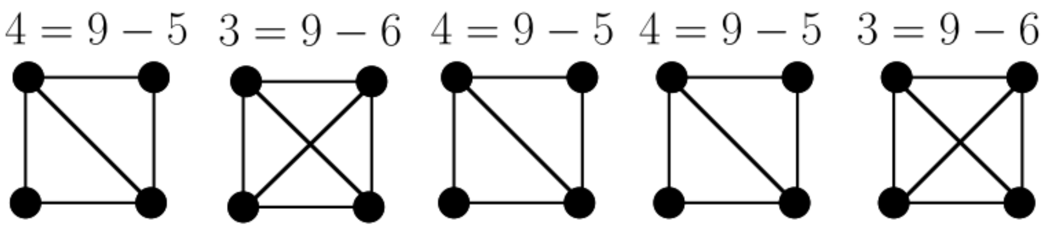
A graph and its vertex-deletion subgraph list.

The Graph-Reconstruction Problem

- **Theorem 2.4.4.** The number of vertices and the number of edges of a graph G can be calculated from its vertex-deletion subgraph list.
  - ✓ One edge connects two vertices, so each edge is deleted twice and appears in $n - 2$ subgraphs, so $|E_G| = \dfrac{1}{n-2}\sum_{v}|E_{G-v}|$

- **Corollary 2.4.5.** The degree sequence of a graph G can be calculated from its reconstruction deck.
  - ✓ First compute $|E_G|$

$$\frac{1}{3}(5+6+5+5+6) = \frac{27}{3} = 9$$

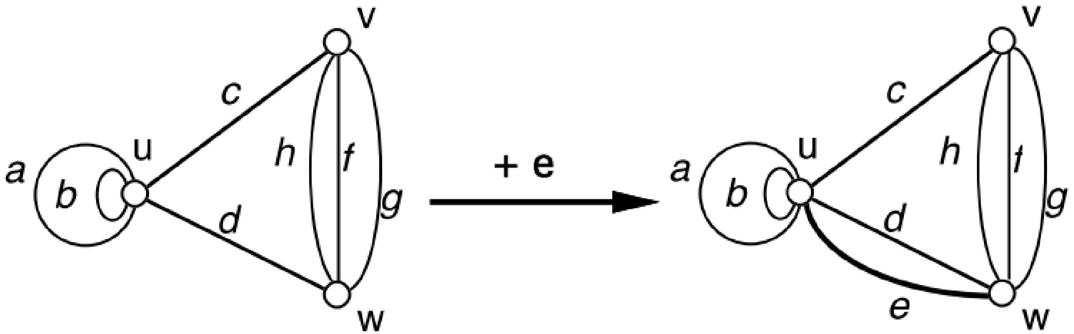


- **Corollary 2.4.6.** Any regular graph can be reconstructed from its reconstruction deck.
  - ✓ For k-regular graph, each subgraph contains n-1 vertices of degree (k-1).

- **Remark:** B. McKay [Mc77] and A. Nijenhuis [Ni77] have shown, with the aid of computers, that a counterexample to the reconstruction conjecture would have to have at least 10 vertices.

Adding Edges or Vertices

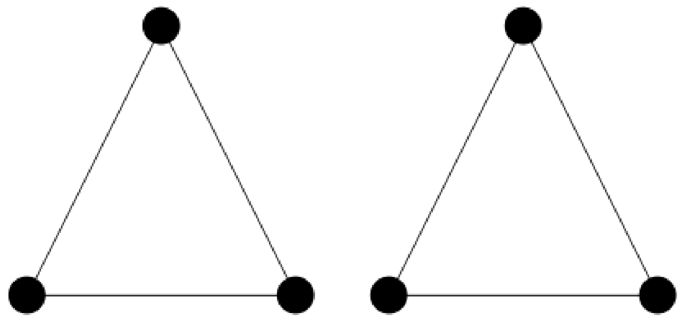
- **DEFINITION**: ***Adding an edge*** between two vertices u and w of a graph G means creating a supergraph, denoted $G \cup \{e\}$, with vertex-set V_G and edge-set $E_G \cup \{e\}$, where e is a new edge with endpoints u and w.
- **DEFINITION**: ***Adding a vertex*** v to a graph G, where v is a new 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.
- **DEFINITION**: Any one of the operations of adding or deleting a vertex or adding or deleting an edge is called a ***primary maintenance operation***.
- **Remark:** *Secondary operations* on a graph are those that are realizable by a combination and/or repetition of one or more of the primary graph operations.



Adding an edge e with endpoints u and w.

Graph Union

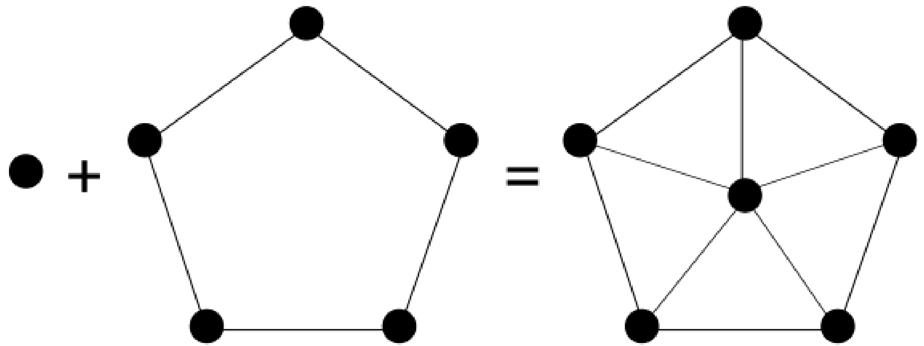
- **DEFINITION**: The (***graph***) ***union*** of two graphs $G = (V, E)$ and $G' = (V', E')$ is the graph $G \cup G'$, whose vertex-set and edge-set are the disjoint unions, respectively, of the vertex-sets and edge-sets of G and G'.
  - ✓ Union is a secondary graph operation and is not the set-theoretic union operation.
- **Example 2.4.7.** Graph union of two K_3 is not a single K_3 (set union will yield a K_3).
- **DEFINITION**: The n-***fold self-union*** nG is the iterated disjoint union $G \cup \cdots \cup G$ of n copies of the graph G. (e.g. example is 2-fold self-union $2K_3$.)



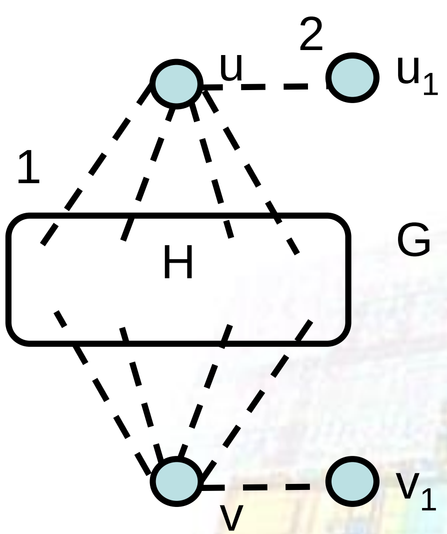
The graph union $K_3 \cup K_3$ of two copies of K_3.

Joining a Vertex to a Graph

- **DEFINITION**: If a new vertex v is joined to each vertex 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 $G + v$.
  - ✓ Joining a vertex to a graph is a secondary operation using primary operations.
- **DEFINITION**: The ***n-wheel*** W_n is the join $K_1 + C_n$ of a single vertex and an n-cycle. (The n-cycle forms the rim of the wheel, and the additional vertex is its hub.) If n is even, then W_n is called an ***even wheel***; if odd, then W_n is called an ***odd wheel***.
- **Proposition 2.4.7.** *Let H be a graph of diameter d. Then there is a graph G of radius d of which H is the center.*
  - ✓ $ecc(w) = 2$ for every $w \in H$, $ecc(u_1) = 4$.

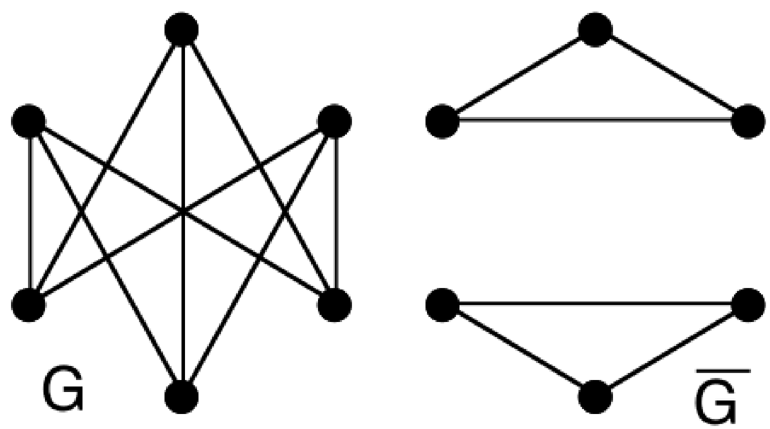


The 5-wheel $W_5 = K_1 + C_5$.



Edge-Complementation

- **DEFINITION**: Let G be a simple graph. Its ***edge-complement*** (or ***complement***) $\overline{G}$ has the same vertex-set, but two vertices are adjacent in $\overline{G}$ if and only if they are not adjacent in G.
- **Remark:** The edge-complement of the edge-complement is the original graph, i.e., $\overline{\overline{G}} = G$.
- **Theorem 2.4.8.** *Let G be a simple graph. Then* $\omega(G) = \alpha(\overline{G})$ *and* $\alpha(G) = \omega(\overline{G})$
- **Remark:** Other graph operations are defined in §2.7.



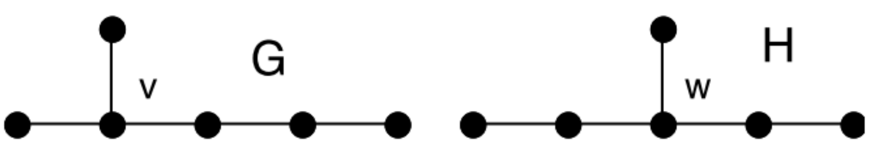
A graph and its complement.

2.5 Tests for Non-Isomorphism

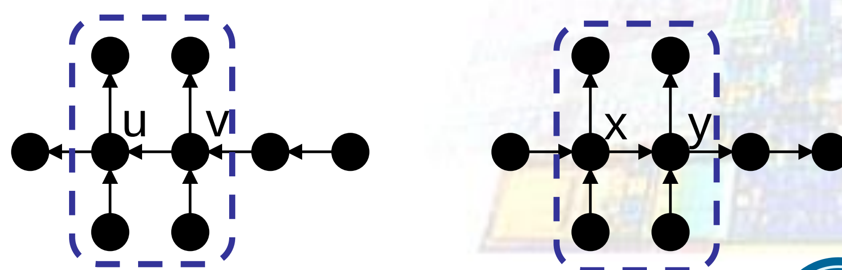
- Although there is no method that can be easily applied to decide whether two graphs are isomorphic or not, we can simplify this problem by some invariant factors.
- **DEFINITION**: A ***graph invariant*** (or ***digraph invariant***) is a property of graphs (digraphs) that is preserved by isomorphisms.
- **Example:** the number of vertices/edges and the degree sequence.

Local Invariant

- **Theorem 2.5.1.** Let $f : G \to H$ be a graph isomorphism, and let $v \in V_G$. Then the multiset of degrees of the neighbors of v equals the multiset of degrees of the neighbors of $f(v)$.
  - ✓ Corollary 2.1.5. *Let $f : G \to H$ be a graph isomorphism and let $e \in E_G$. Then the endpoints of edge* $f(e)$ have the same degrees of the endpoints of e.
- **Example:** (a). Same degree sequence: <1,1,1,2,2,3>; (b) one 3-degree vertex; (c). different multiset of degrees of two 3-degree vertices' neighbors. <1,1,2> and <1,2,2>
- **Example:** (a) v maps to x; (b) u must map to y

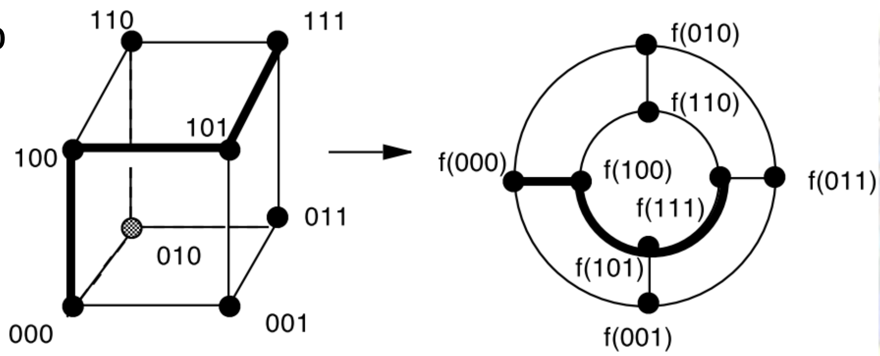


Non-isom graphs with the same degree seq.



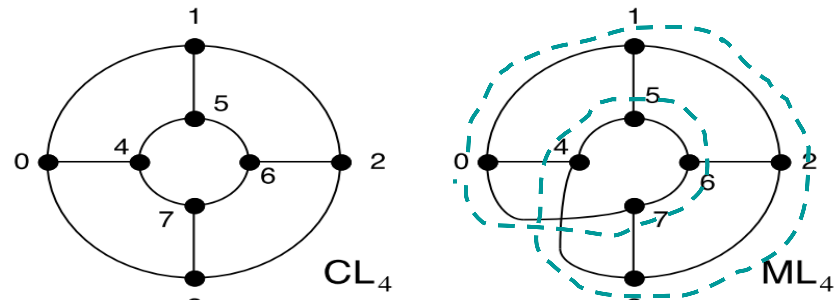
Distance Invariants

- **DEFINITION**: Let $W = \langle v_0, e_1, v_1, \ldots, e_n, v_n \rangle$ be a walk in the domain G of a graph isomorphism $f: G \to H$. Then the ***image of walk*** W is the walk $f(W) = \langle f(v_0), f(e_1), f(v_1), \ldots, f(e_n), f(v_n) \rangle$ in graph H.
- **Theorem 2.5.2.** The isomorphic image of a graph walk W is a walk of the same length.
  - ✓ Isomorphism perserves edge-multiplicity
- **Corollary 2.5.3.** The isomorphic image of a trail, path, or cycle is a trail, path, or cycle, respectively, of the same length.
  - ✓ By Theorem 2.5.2. & the bijectivity of the isomorphism.
- **Corollary 2.5.4.** For each integer l, two isomorphic graphs must have the same number of trails (paths) (cycles) of length l.
- **Corollary 2.5.5.** The diameter, the radius, and the girth are graph invariants.
  - ✓ By Corollary 2.5.3. & the bijectivity of iso



Distance Invariants & Subgraph Presence

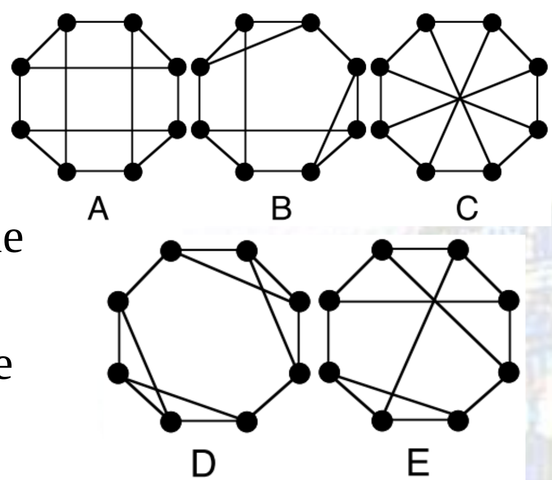
- **Example:** CL_4 : (a) rotational symmetry; (b) isomorphism to swap inner and outer cycles. ML_4 : $j \mapsto j+1$ mod 8 is an automorphism. CL_4 and ML_4 are vertex-transitive.
  - ✓ $\alpha(CL_4)$ =4 but $\alpha(ML_4)$=3



Non-isom 3-regular graphs.

Subgraph Presence

- **Theorem 2.5.6.** *For each graph-isomorphism type, the number of distinct subgraphs in a graph having that isomorphism type is a graph invariant.*
  - ✓ Bijectivity of isomorphism establishes this invariant.
- **Example:** A and C have no K_3 subgraphs, B has two, D has four, and E has one. Thus, Thm 2.5.6 implies that the only possible isomorphic pair is A and C. However, graph C has a 5-cycle, but graph A (bipartite) does not. Alternatively, we see that 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 $\alpha(C) = 3$.



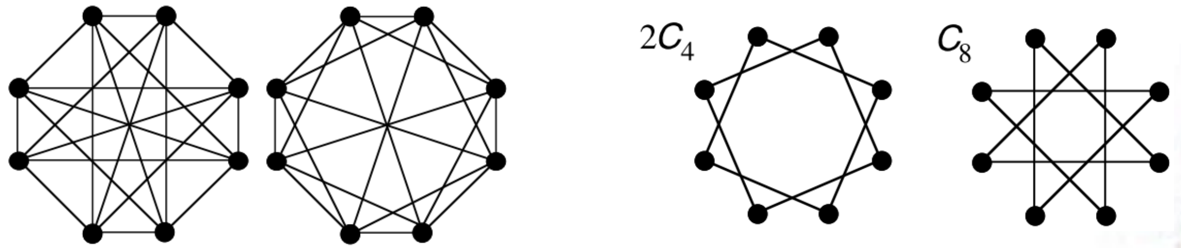
Five mutually non-isom, 8-vertex, 3-reg graphs.

Edge Complementation

- **Theorem 2.5.7.** *Let G and H both be simple graphs. They are isomorphic if and only if their edge-complements are isomorphic.*
  - ✓ *Graph isomorphism necessarily preserves non-adjacency as well as adjacency.*
- **Example:**
- **Remark.** Edge complementation is useful for simple dense graph.

Summary

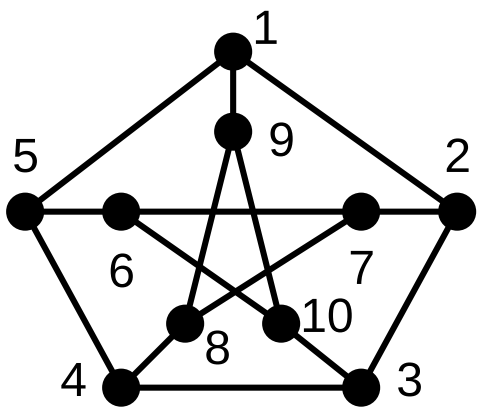
- Some graph invariants for graph isomorphism – (1) & (2) the number of vertices and edges; (3) degree sequence; (4) multiset of local graphs; (5) degrees of neighbors of a forced match; (6) diameter, radius, girth; (7) independence number, clique number; (8) For any possible subgraph, the number of distinct copies; (9) For a simple graph, the edge-complement.

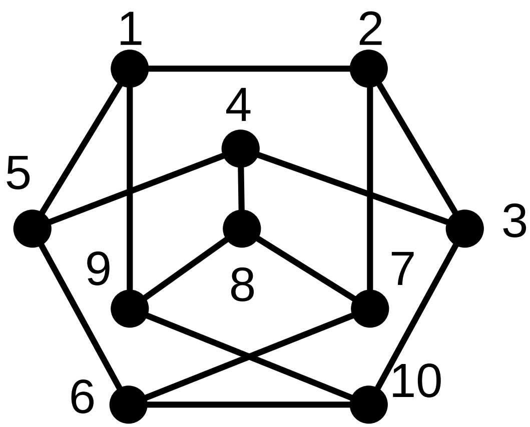


Two relatively dense, non-isomorphic 5-regular graphs and their edge-complements.

Using Invariants to Construct an Isomorphism

- **Example:** Two graphs have one P_9 (1,2,3,4,5,6,7,8,9,10). According to Corollary 2.5.3., vertices along these two paths can have a isomorphism mapping.





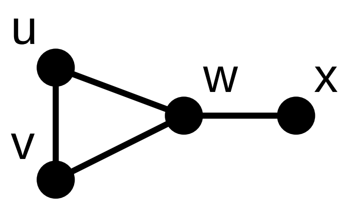
Two copies of the Petersen graph.

2.6 Matrix Representations

- **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.**



uu × uv + uv × vv + uw × wv + ux × xv

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

- **Proposition 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.*
  - ✓ **Proof:** The assertion holds for $r = 1$. The inductive step follows from the definition of matrix multiplication.

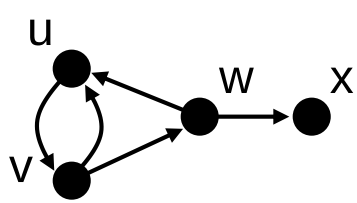
$$A_G^2 = \begin{pmatrix} 2 & 1 & 1 & 1 \\ 1 & 2 & 1 & 1 \\ 1 & 1 & 3 & 0 \\ 1 & 1 & 0 & 1 \end{pmatrix} \qquad A_G^3 = \begin{pmatrix} 2 & 3 & 4 & 1 \\ 3 & 2 & 4 & 1 \\ 4 & 4 & 2 & 3 \\ 1 & 1 & 3 & 0 \end{pmatrix} \qquad A_G^3[u,w] = 4 \begin{cases} u-w-u-w \\ u-w-x-w \\ u-w-v-w \\ u-v-u-w \end{cases}$$

Adjacency Matrices

- **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

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

- **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 element of column j equals the indegree of vertex v_j.*
- **Example 2.6.2.**



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

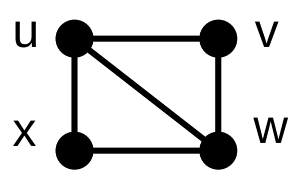
- **Proposition 2.6.3.** *Let D be a digraph with adjacency matrix A_D. Then the value of the entry* A^r_D[u,v] *of the r^{th} power of matrix A_D equals the number of directed u-v walks of length r.*

- **Remark:** To extend the definition of adjacent matrix to a general graph (or digraph), one lets $A_G[u,v]$ equals the number of edges between vertex u and vertex v (or from vertex u to vertex v).

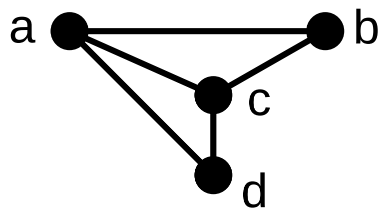
Brute–Force Graph–Isomorphism Testing

- Isomorphism checking can be realized by finding two vertex orderings such that their adjacency matrices are the same.

- **Example 2.6.3.**



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



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

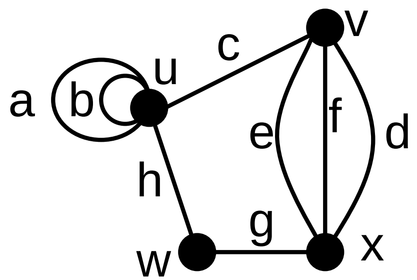
- **COMPUTATIONAL NOTE**: For all but the smallest graphs, exhaustive method is hopelessly inefficient.

Incidence Matrices for Undirected Graphs

- **DEFINITION**: The ***incidence matrix*** of a graph G is the matrix I_G whose rows and columns are indexed by some orderings of V_G *and* E_G, respectively , such that

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

- **Example 2.6.4.**



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

- **Proposition 2.6.4.** *The sum of the entries in any row of an incidence matrix is the degree of the corresponding vertex.*

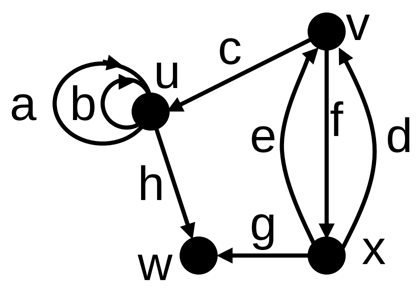
- **Proposition 2.6.5.** *The sum of the entries in any column of an incidence matrix is equal to 2.*

Incidence Matrices for Digraphs

- **DEFINITION**: The ***incidence matrix*** of a digraph D is the matrix whose rows and columns are indexed by some orderings of V_D *and* E_D, respectively, such that

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

- **Example 2.6.5.**

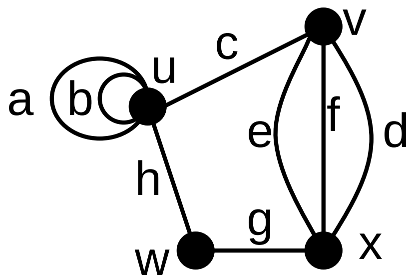


$$I_D = \begin{array}{c} \\ u \\ v \\ w \\ x \end{array} \begin{array}{c} \begin{array}{cccccccc} a & b & c & d & e & f & g & h \end{array} \\ \begin{pmatrix} 2 & 2 & 1 & 0 & 0 & 0 & 0 & -1 \\ 0 & 0 & -1 & 1 & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & -1 & -1 & 1 & -1 & 0 \end{pmatrix}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

Using Incidence Tables to Save Space

- **DEFINITION**: The ***table of incident edges*** for a graph G is an incidence table that lists , for each vertex v , all the edges incident on v . This table is denoted $I_{V:E}(G)$.

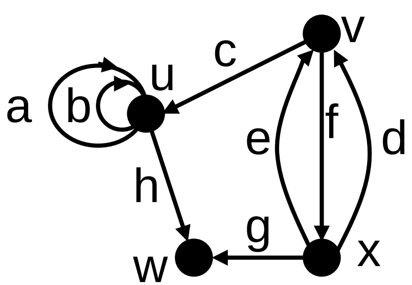
- **Example 2.6.6.**



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

- **DEFINITION**: For a digraph D , the ***table of outgoing arcs*** , denoted $out_{V:E}(D)$, lists , for each vertex v , all arcs that are directed form v. The ***table of incoming arcs*** , denoted $in_{V:E}(D)$, is defined similarly.

- **Example 2.6.7.**



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

2.7 More Graph Operations

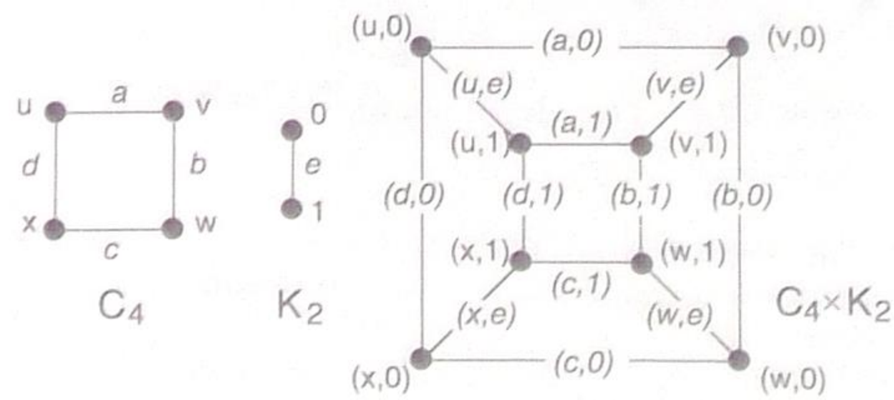
- **DEFINITION**: The ***(cartesian) product*** $G \times H$ of the graphs G and H has as its vertex-set the cartesian product

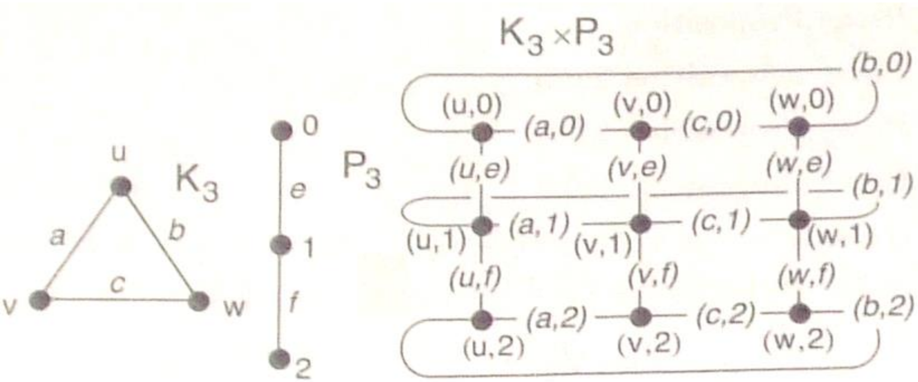
$$V_{G\times H} = V_G \times V_H$$

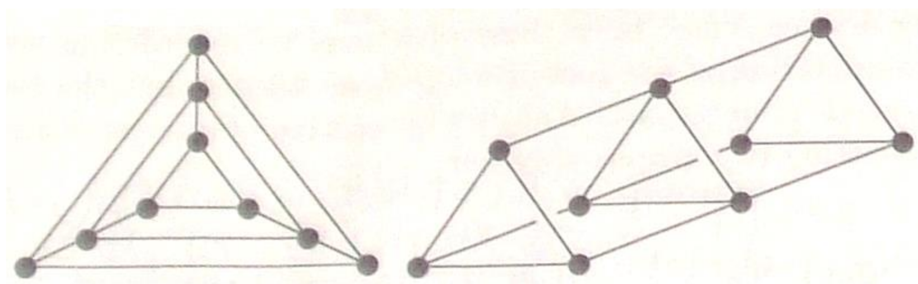
and as its edges a union of two products:

$$E_{G\times H} = (V_G \times E_H) \cup (E_G \times V_H)$$

- **Example 2.7.1.** Cartesian product $C_4 \times K_2$
- **Example 2.7.2.** Cartesian product $K_3 \times P_3$





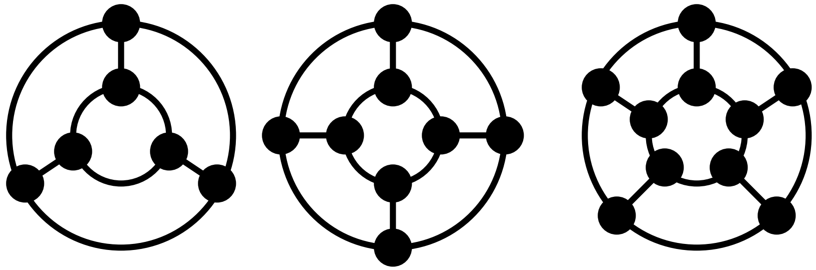


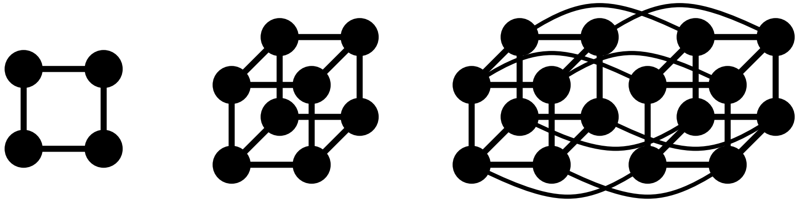
Two alternative views of $K_3 \times P_3$

Cartesian Product

- **DEFINITION**: The product $K_2 \times C_n$ is called a ***circular ladder with*** *n* ***rungs*** and often denoted CL_n.

CL_3, CL_4, and CL_5

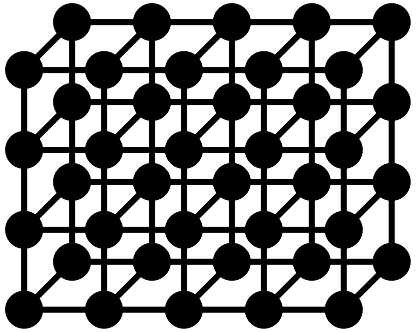


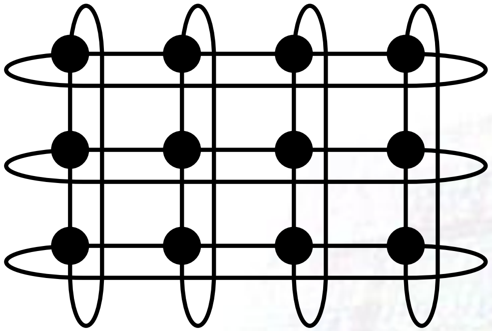


Q_2, Q_3, and Q_4

- **DEFINITION**: The iterated product $K_2 \times \ldots \times K_2$ of n copies of K_2 is called either the ***hypercube graph of dimension*** *n* or the *n*-***hypercube graph***. It is denoted Q_n.

- **DEFINITION**: The $m_1 \times m_2 \times \ldots \times m_n$-***mesh*** is the iterated product $P_{m_1} \times P_{m_2} \times \cdots \times P_{m_n}$ of paths.

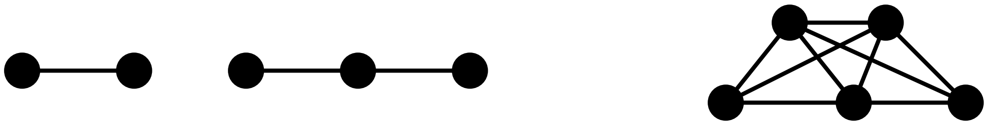




- **DEFINITION**: The ***wraparound*** $m_1 \times m_2 \times \ldots \times m_n$-***mesh*** is the iterated product $C_{m_1} \times C_{m_2} \times \cdots \times C_{m_n}$ of cycles.

Join

- **Remark:** The product operation is both commutative and associative.
- **DEFINITION**: The ***join*** $G + H$ of the graphs G and H is obtained from the graph union $G \cup H$ by adding an edge between each vertex of G and each vertex of H.

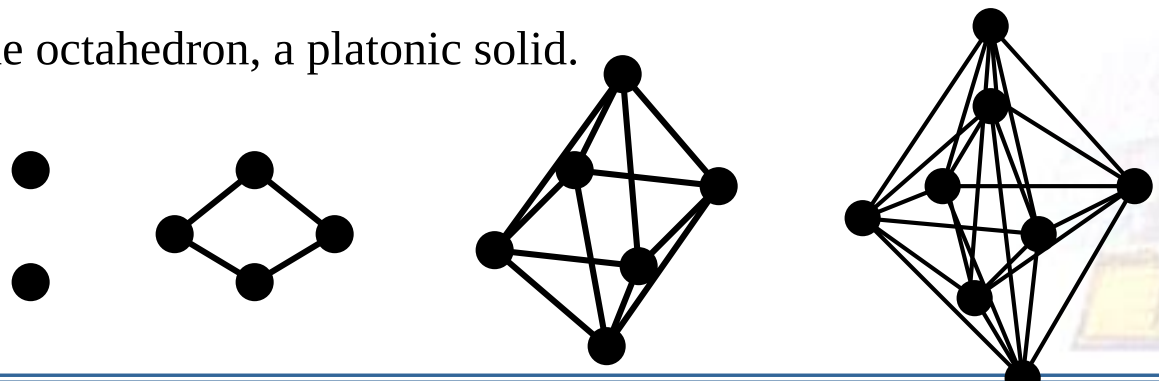


- **Example 2.7.3.** The join $K_m + K_n$ is isomorphic to the complete graph K_{m+n}.
- **Example 2.7.4.** The join $mK_1 + nK_1$ is isomorphic to the complete bipartite graph $K_{m,n}$.
- **DEFINITION**: The ***n-dimensional octahedral graph*** O_n, is defined recursively, using the join operation.

$$O_n = \begin{cases} 2K_1 & \text{if } n = 1 \\ 2K_1 + O_{n-1} & \text{if } n > 1 \end{cases}$$

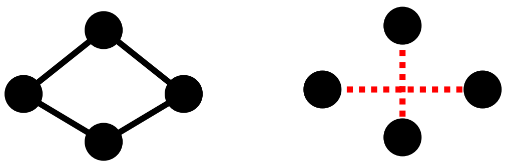
It is also called the ***n-octahedral graph*** or, when $n = 3$, the ***octahedral graph***, because it is the 1-skeleton of the octahedron, a platonic solid.

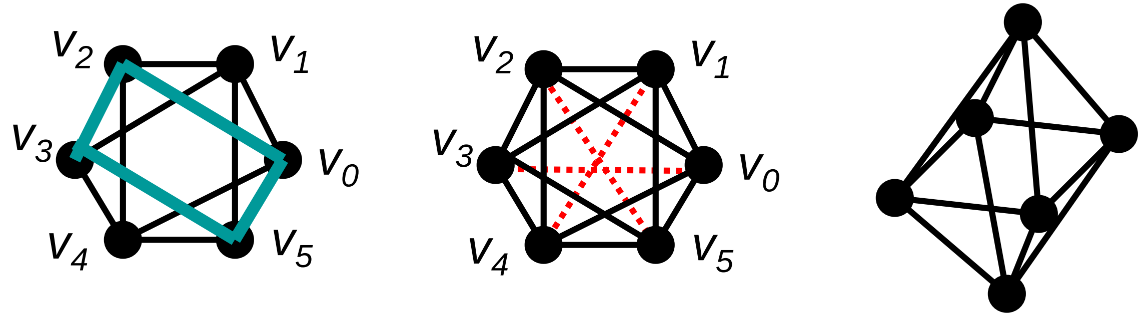
- **Example 2.7.5.**



Join & Amalgamations

- **Example 2.7.6.** The edge complement of the graph nK_2 (in K_{2n}) is isomorphic to O_n.



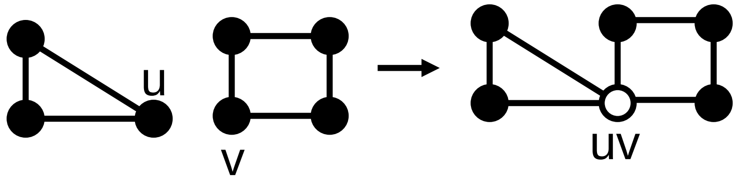


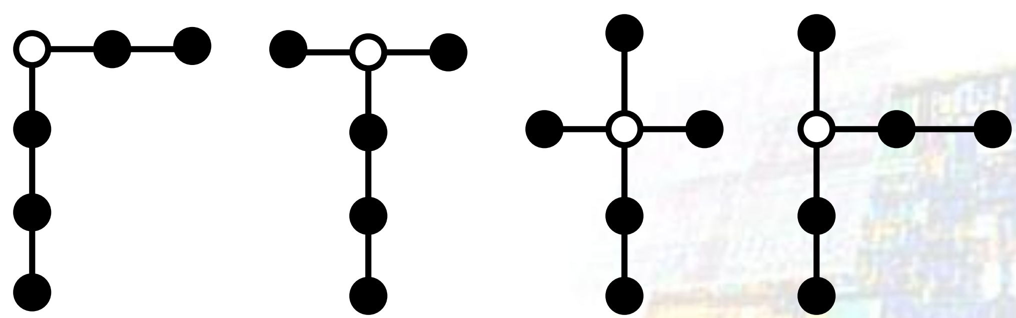
Amalgamation

- **DEFINITION**: Let G and H be disjoint graphs, with $u \in V_G$ and $v \in V_H$. The ***vertex amalgamation*** $(G \cup H) / \{ u = v\}$ is the graph obtained from the union $G \cup H$ by merging (or amalgamating) vertex u of graph G and vertex v of graph H into a single vertex, called uv. The vertex-set of this new graph is $(V_G - \{u\}) \cup (V_H - \{v\}) \cup \{uv\}$, and the edge-set is $E_G \cup E_H$, except that any edge that had u or v as an endpoint now has the amalgamated vertex uv as an endpoint instead.

- **Example 2.7.7.**

- **Example 2.7.8.**

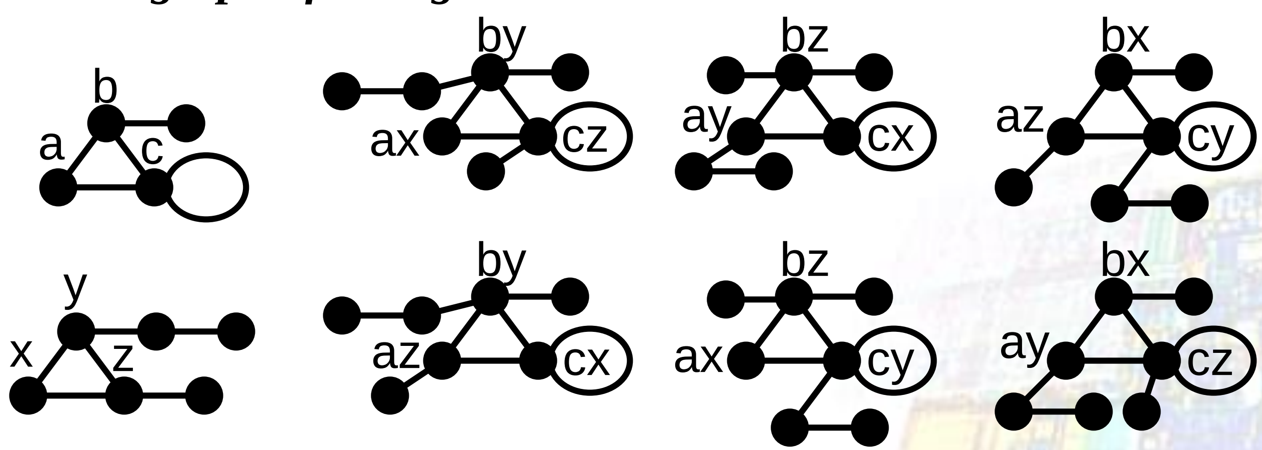




Four different vertex amalgamations of P_3 and P_4

Amalgamations

- **DEFINITION**: Let G and H be disjoint graphs, with X a subgraph of G and Y a subgraph of H. Let f: $X \rightarrow Y$ be an isomorphism between these subgraphs. The ***amalgamation of*** G ***and*** H ***modulo the isomorphism*** f: $X \rightarrow Y$ is the graph obtained from the union $G \cup H$ by merging each vertex u and each edge e of subgraph X with their images $f(u)$ and $f(e)$ in subgraph Y. The amalgamated vertex is generically denoted $uf(u)$, and the amalgamated edge is generically denoted $ef(e)$. The vertex-set of this new graph is $(V_G - V_X) \cup (V_H - V_Y) \cup \{uf(u) \mid u \in V_X\}$, and the edge-set is $(E_G - E_X) \cup (E_H - E_Y) \cup \{ef(e) \mid e \in E_X\}$, except that any edge that had $u \in V_X$ or $f(u) \in V_Y$ as an endpoint now has the amalgamated vertex $uf(u)$ as an endpoint instead. This general amalgamation is denoted $(G \cup H) / f$: $X \rightarrow Y$.
- **DEFINITION**: In an amalgamated graph $(G \cup H) / f$: $X \rightarrow Y$, the image of the pasted subgraphs X and Y is called the ***subgraph of amalgamation***.
- **Example 2.7.9.**



six different possible amalgamated graphs.