Administrative Matters

- Course: Graph Theory
- Time/Location: M56-EC016
- Instructor: 李毅郎
- E-mail: ylli@cs.nctu.edu.tw
- URL: https://people.cs.nctu.edu.tw/~ylli/
- Office: 工三441
- Office Hours: T56 or make an appointment by @
- Teaching Assistants:
  - ✓ 張竣翔： lemonade.cs07@nycu.edu.tw
  - ✓ 石孟祐： yoyochang24.c@nycu.edu.tw
  - ✓ 倪子傑： nibilly118@gmail.com
  - ✓ 耿鈺展： kch772018@gmail.com
  - ✓ 黃晨凱： huangkevin88.ee07@nctu.edu.tw
  - ✓ 李承翰： darrenleeleelee1@gmail.com
  - ✓ 白育濬： a0935796620@gmail.com
- Prerequisites: none
- Textbook: "Graph Theory and Its Applications" by Jonathan L. Gross and Jay Yellen, Second Edition. Publisher: Chapman & Hall/CRC

Administrative Matters

- Course goal: This course introduces well-known graph models and their applications, and also offers deeper discussions in various well-known graph models and their deduction processes, which are helpful preliminary practices to develop your own solid research model.
- Course contents:
  - ✓ Introduction to Graph Models
  - ✓ Structures and Representation
  - ✓ Trees
  - ✓ Spanning Trees
  - ✓ Connectivity
  - ✓ Optimal Graph Traversals
  - ✓ Planarity and Kuratowski‘s Theorem
  - ✓ Drawing Graphs and Maps
  - ✓ Graph Colorings
  - ✓ Network Flows and Applications

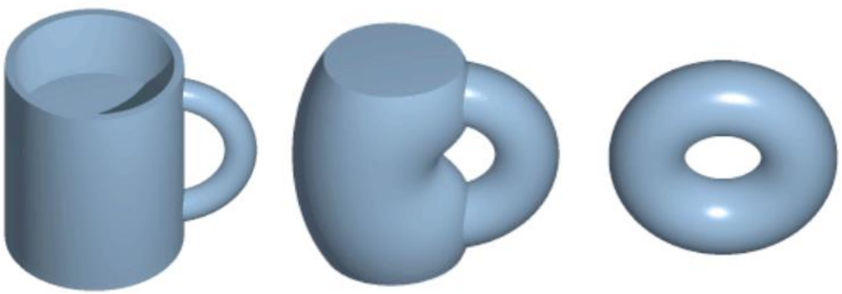
Administrative Matters

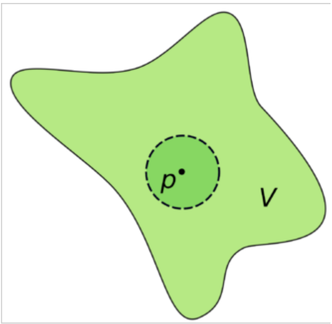
- Grading
  - ✓ assignments: 35%,
  - ✓ Two tests: 65%, (The better one : 35%, the worse one 30%)
  - ✓ Class participation: bonus

- Course Web Site: eCampus
- Academic Honesty**:** Avoiding *cheating* at all cost.

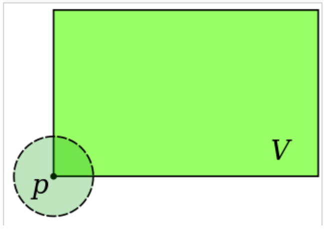
Preliminary

- Geometria (Euclidean geometry, Solid geometry, non-Euclidean geometry , Riemannian geometry, Algebraic geometry, Differential geometry, Topology) & Algebra
- Manifold: Each point of an *n*-dimensional manifold has a neighborhood that is homeomorphic (topological isomorphical) to the Euclidean space of dimension *n*.
- 1-D manifold: line and circle. 2-D manifold: surface including plane, sphere and torus.
- Graph theory can be regarded as 1-D topology.





柏拉圖的形上學將世界切割為兩個不同的區塊：「形式的」智慧世界、以及我們所感覺到的世界。我們所感覺到的世界是從有智慧的形式或理想裡所複製的，但這些複製版本並不完美。那些真正的形式是完美的而且無法改變的，而且只有使用智力加以理解才能實現之，這也表示了人的智力並不包含知覺能力或想像力。



source: wiki

Preliminary

Preliminary

- **The problem of Seven Bridges of Königsberg** solved by Leonhard Euler in 1735

Figure source: wiki

Chap 1. Introduction to Graph Models

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

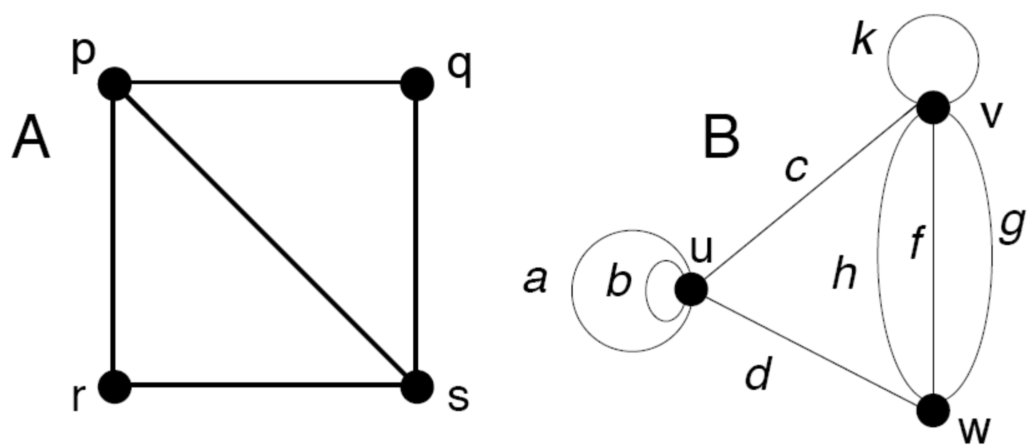
The sources of most figure images are from the course slides (Graph Theory) of Prof. Gross

Outline

- Graph Theory Introduction
- Graphs and Digraphs
- Common Families of Graphs
- Graph Modeling Applications
- Walks and Distance
- Paths, Cycles, and Trees
- Vertex and Edge Attributes: More Applications

1.1 Graphs and Digraphs

- **DEFINITION**: A ***graph*** $G = (V,E)$ is a mathematical structure consisting of two finite sets V and E. The elements of V are called ***vertices*** (or ***nodes***), and the elements of E are called ***edges***. Each edge has a set of one or two vertices associated to it, which are called its ***endpoints***.
- **TERMINOLOGY**: An edge is said to ***join*** its endpoints. A vertex joined by an edge to a vertex v is said to be a ***neighbor*** of v.
- **DEFINITION**: The (***open***) ***neighborhood*** of a vertex v in a graph G, denoted $N(v)$, is the set of all the neighbors of v. The ***closed neighborhood*** of v is given by $N[v] = N(v) \cup \{v\}$.
- **DEFINITION**: A ***simple graph*** has neither self-loops nor multi-edges. A (***general***) ***graph*** may have self-loops and/or multi-edges.



$E_A=\{pq, pr, ps, rs, qs\}$

$E_B=\{a, b, c, d, f, g, h, k\}$ can use vertex to represent edge like E_A?

What's the difference between E_A and E_B?

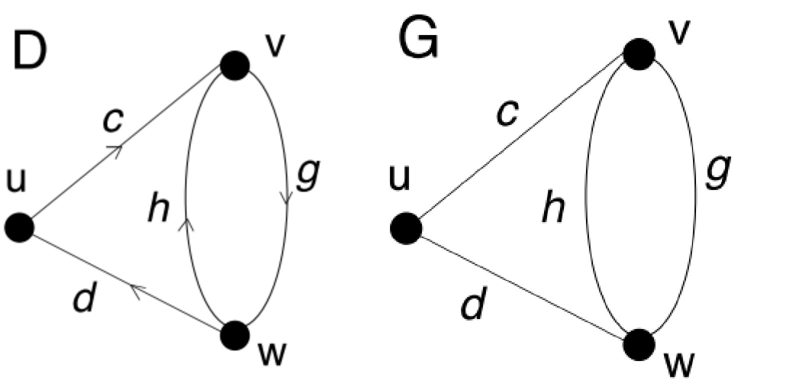
Line drawings of a graph *A* and a graph *B*.

Simple, General, Null, and Trivial Graphs and Edge Directions

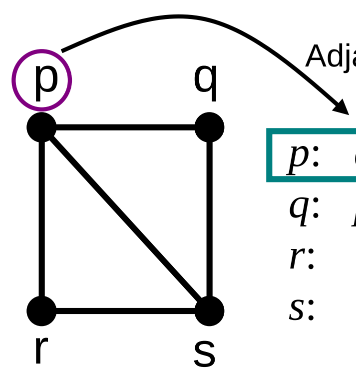
- **DEFINITION**: A ***proper edge*** is an edge that joins two distinct vertices.
- **DEFINITION**: A ***self-loop*** is an edge that joins a single endpoint to itself.
- **DEFINITION**: A ***multi-edge*** is a collection of two or more edges having identical endpoints. The ***edge multiplicity*** is the number of edges within the multi-edge.
- **DEFINITION**: A ***simple graph*** has neither self-loops nor multi-edges.
- **DEFINITION**: A ***loopless graph*** (or ***multi-graph***) may have multi-edges but no self-loops.
- **DEFINITION**: A ***(general) graph*** may have self-loops and/or multi-edges.
- **DEFINITION**: A ***null graph*** is a graph whose vertex- and edge-sets are empty.
- **DEFINITION**: A ***trivial graph*** is a graph consisting of one vertex and no edges.
- **DEFINITION**: ***A directed edge*** (or ***arc*, from tail to head**)**,** ***oppositely directed, multi-arc, arc multiplicity, directed graph*** (or ***digraph***)**,** ***simple digraph, mixed graph*** (or ***partially directed graph***)
- ***Forward*:** along the direction; ***backward*:** anti the direction.

Formal Specification of Graphs and Digraphs

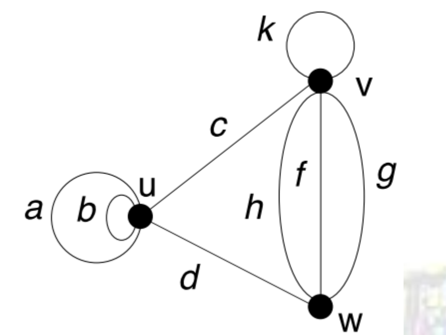
- **DEFINITION**: The ***underlying graph*** of a directed or mixed graph *G* is the graph that results from removing all the designations of head and tail from the directed edges of *G* (i.e., deleting all the edge directions).
- **DEFINITION**: A ***formal specification of a simple graph*** is given by an ***adjacency table*** with a row for each vertex, containing the list of neighbors of that vertex.
- A ***formal specification of a general graph*** *G* = (*V*, *E*, *endpts*) consists of a list of its vertices, a list of its edges, and a two-row ***incidence table*** (specifying the *endpts* function) whose columns are indexed by the edges. The entries in the column corresponding to edge *e* are the endpoints of *e*.



A digraph and its underlying graph



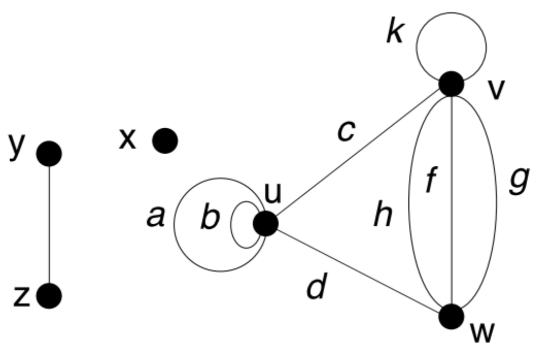
p:	*q*	*r*	*s*
q:	*p*	*s*	
r:	*p*	*s*	
s:	*p*	*q*	*r*



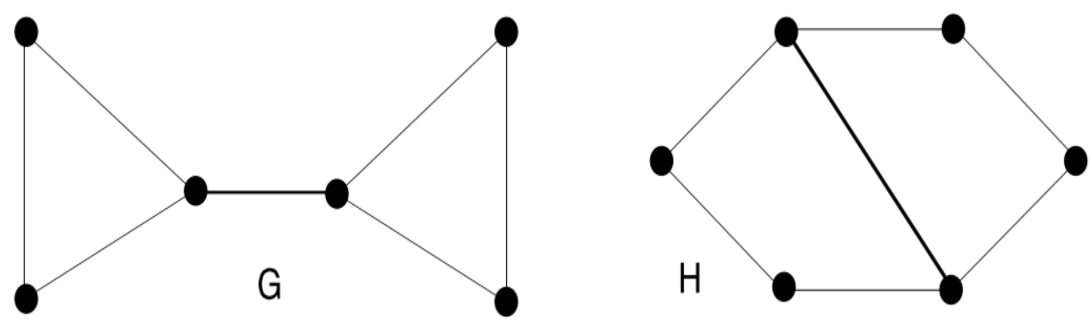
edge	a	b	c	d	f	g	h	k
endpts	u	u	u	w	v	v	w	v
	u	u	v	u	w	w	v	v

Degree of a Vertex

- **DEFINITION**: A ***formal specification of a general digraph or a mixed graph*** $D = (V, E, endpts, head, tail)$ is obtained from the formal specification of the underlying graph by adding the functions $head : E_G \rightarrow V_G$ and $tail : E_G \rightarrow V_G$, which designate the *head* vertex and *tail* vertex of each arc.
  - ✓ How to add *head* and *tail* function – ordered-pair, mark head or fixed positioning
- **DEFINITION**: ***Adjacent vertices, adjacent edges, incident, degree/valence, d-valent vertex***
- **DEFINITION**: The ***degree sequence*** of a graph is the sequence formed by arranging the vertex degrees in non-increasing order.
- Two graphs with different structures may have the same degree sequence.



< 6, 6, 4, 1, 1, 0 >
v u w z y x

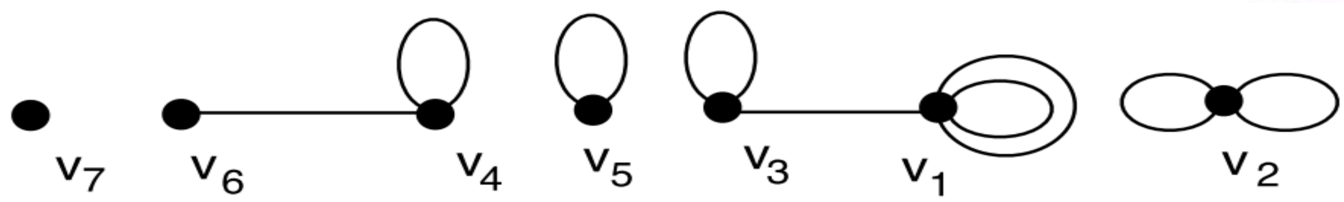


Two graphs with the same degree sequence
< 3, 3, 2, 2, 2, 2>

Degree of a Vertex

- **Proposition 1.1.1.** *A non-trivial simple graph G must have at least one pair of vertices whose degrees are equal.*
  - ✓ **Proof:** pigeonhole principle.
- **Theorem 1.1.2 [Euler's Degree-Sum Theorem].** *The sum of the degrees of the vertices of a graph is twice the number of edges.*
- **Corollary 1.1.3.** *In a graph, there is an even number of vertices having odd degree.*
- **Corollary 1.1.4.** *The degree sequence of a graph is a finite, non-increasing sequence of nonnegative integers whose sum is even.*
- **Question:** For a non-increasing nonnegative sequence of integers, is this sequence the degree sequence of some graph if the sum of all integers is even?
- **Example:** To construct a graph whose degree sequence is < 5, 4, 3, 3, 2, 1, 0 >

$v_1\ v_2\ \ v_3\ v_4\ \ v_5\ \ v_6\ v_7$



Graphic Sequences

- **Theorem 1.1.5.** *Suppose that $<d_1, d_2, \ldots, d_n>$ is a sequence of nonnegative integers whose sum is even. Then there exists a graph with vertices $v_1, v_2, \ldots, v_n$ such that $deg(v_i)=d_i$, for $i = 1, \ldots, n$.*

- **DEFINITION:** A sequence $<d_1, d_2, \ldots, d_n>$ is said to be graphic if there is a permutation of it that is the degree sequence of some simple graph. Such a simple graph is said to ***realize*** the sequence.

- **Theorem 1.1.6.** *Let $<d_1, d_2, \ldots, d_n>$ be a graphic sequence, with $d_1 \geq d_2 \geq \ldots \geq d_n$. Then there is a simple graph with vertex-set $\{v_1, \ldots, v_n\}$ satisfying $deg(v_i)=d_i$ for $i = 1, 2, \ldots, n$,* ***such that v_1 is adjacent to vertices $v_2, \ldots, v_{d_1+1}$.***

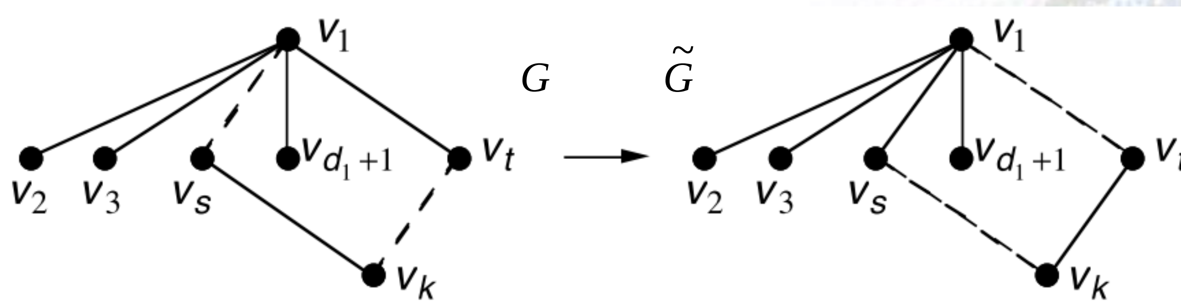
  - ✓ **Proof:**

    $r = |N_G(v_1) \cap \{v_2, \ldots, v_{d_1+1}\}|$ *Select G such that r is maximum.*

    Assume there is a vertex v_s, $2 \leq s \leq d_1 + 1 < t$, $deg(v_1) \geq deg(v_s) \geq deg(v_{d_1+1}) \geq deg(v_t)$,

    In case $deg(v_s) > deg(v_t)$, $\exists v_k$. $G \rightarrow \tilde{G}$, $|N_{\tilde{G}}(v_1) \cap \{v_2, \ldots, v_{d_1+1}\}| = r + 1 \rightarrow$ contradiction to the assumption.

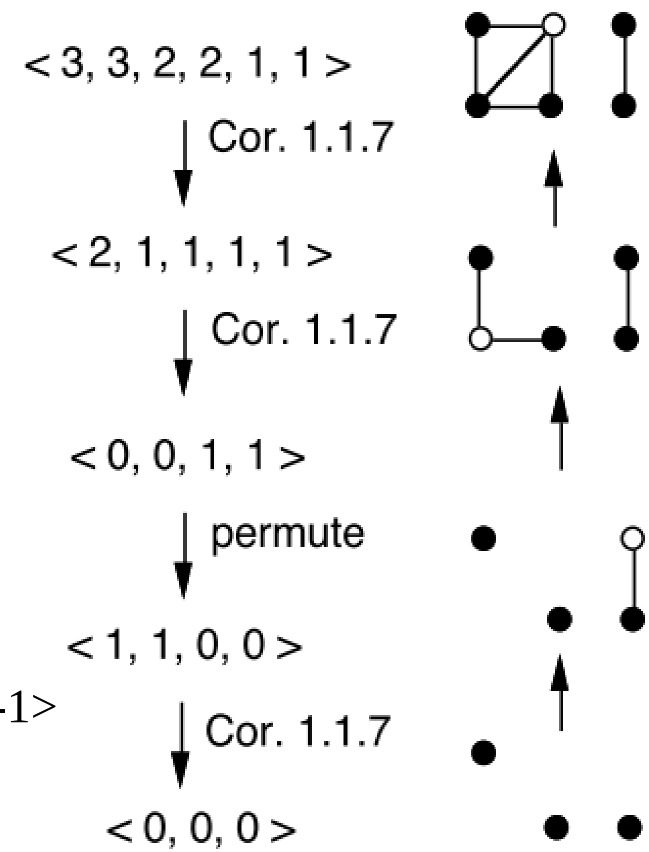
    In case $deg(v_s) = deg(v_t)$, swap v_t with v_s.



Graphic Sequences

- **Corollary 1.1.7 [Havel and Hakimi].** *A sequence* $< d_1, d_2, \ldots, d_n >$ *of nonnegative integers such that* $d_1 \geq d_2 \geq \ldots \geq d_n$ *is graphic if and only if the sequence* $< d_2 - 1, \ldots, d_{d_1+1} - 1, d_{d_1+2}, \ldots, d_n >$ *is graphic.*

Algorithm: Recursive GraphicSequence($\langle d_1, d_2, \ldots, d_n \rangle$)
Input: a non-increasing sequence $\langle d_1, d_2, \ldots, d_n \rangle$.
Output: TRUE if the sequence is graphic; FALSE if it is not.
If $d_1 = 0$ && $d_n = 0$
    Return TRUE
Else
    If $d_n < 0$
        Return FALSE
    Else
        Let $\langle a_1, a_2, \ldots, a_{n-1} \rangle$ be a non-incr permutation
          of $\langle d_2 - 1, \ldots, d_{d_1+1} - 1, d_{d_1+2}, \ldots, d_n \rangle$.
        Return GraphicSequence($\langle a_1, a_2, \ldots, a_{n-1} \rangle$)



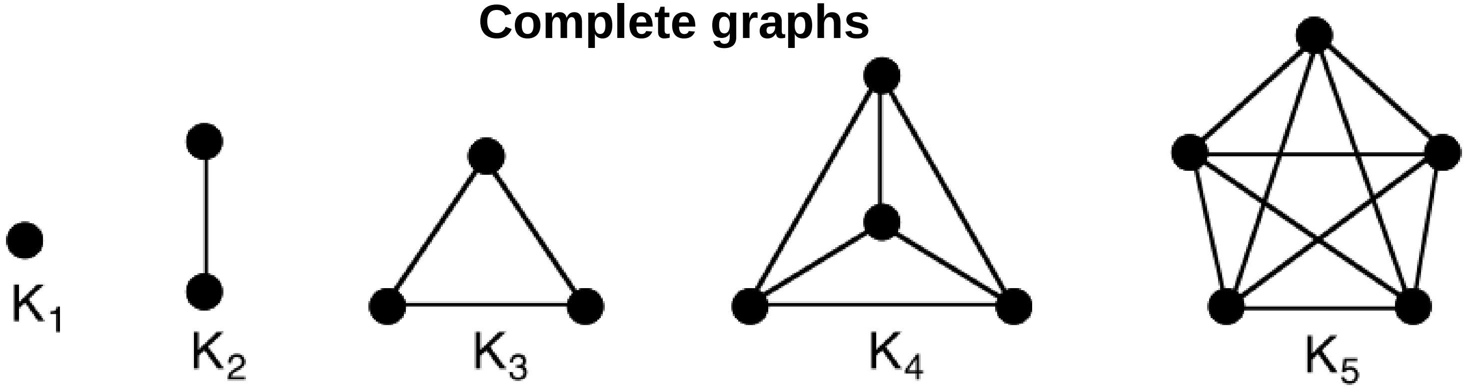
- **Example.** $<5,4,3,2,2,1> \rightarrow <3,2,1,1,0> \rightarrow <1,0,0,0> \rightarrow <0,0,-1>$

Indegree and Outdegree in a Digraph

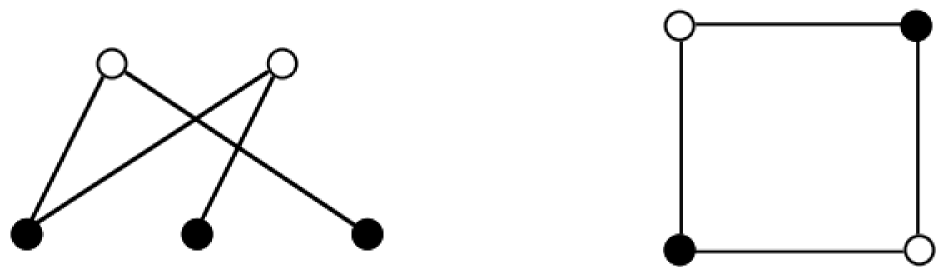
- **DEFINITION**: The ***indegree*** of a vertex v in a digraph is the number of arcs directed to v; the ***outdegree*** of vertex v is the number of arcs directed from v. Each self-loop at v counts one toward the indegree of v and one toward the outdegree.
- **Theorem 1.1.8.** *In a digraph, the sum of the indegrees and the sum of outdegrees both equal the number of edges.*

1.2 Common Families of Graphs

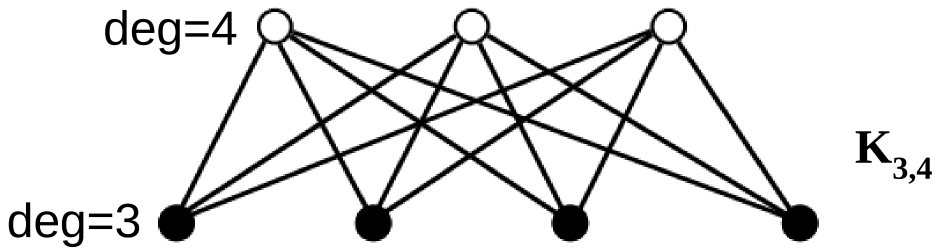
Complete graphs



Bipartite graphs

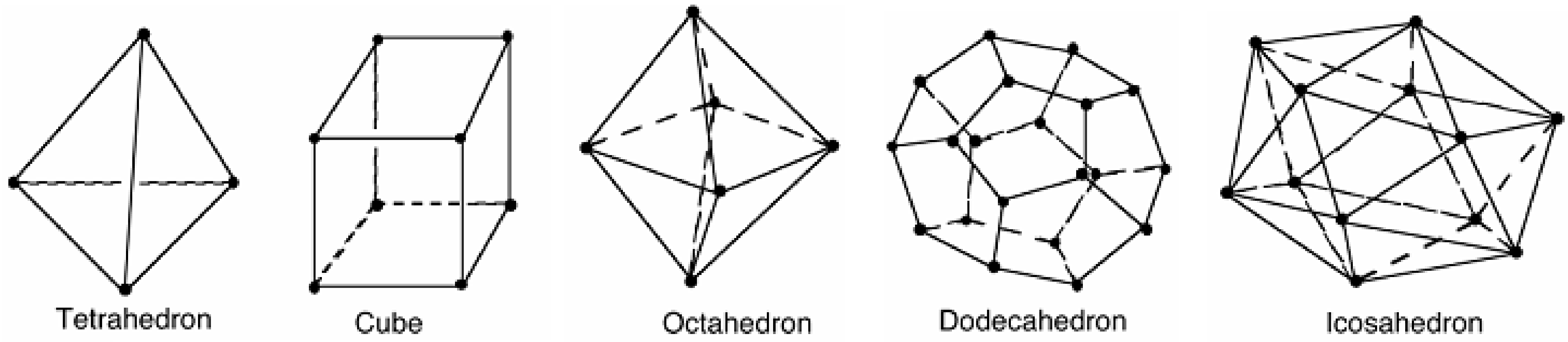


Complete bipartite graph



Regular Graphs

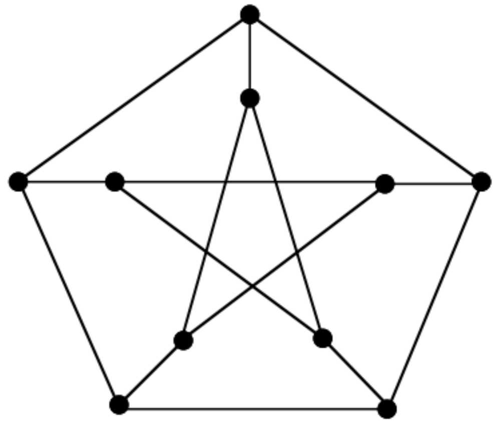
- **DEFINITION**: A regular graph is a graph whose vertices have common degree *k* and is denoted ***k-regular*** graph.



(fire) (earth) (air) (Ether) (water)

- ***Platonic graph***: Only five types of this kind of graphs (in "Timaeus dialogue" & "The elements") and each of them consists of regular polygons
  - ✓ ***Tetrahedron***: 4 regular triangles; ***cube***: 6 squares; ***Octahedron***: 8 regular triangles; ***Dodecahedron***: 12 regular pentagons; ***Icosahedron***: 20 regular triangles
  - ✓ **Property 1**: The number of polygons adjacent to each vertex is the same.
  - ✓ **Property 2**: There exists a sphere such that all vertices are on its surface.
  - ✓ **Property 3**: The sum of vertex number and face number equals to the edge number plus 2.

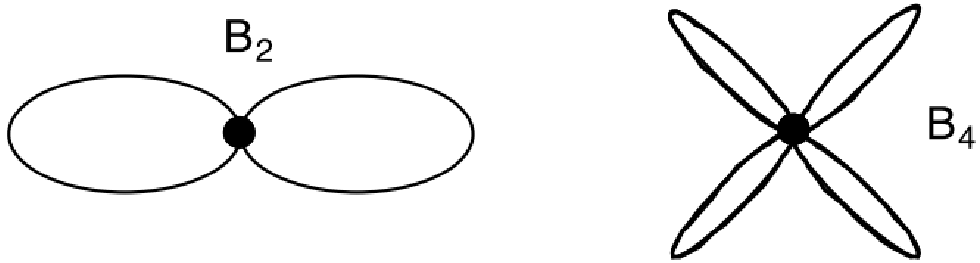
Regular Graphs



Peterson graph

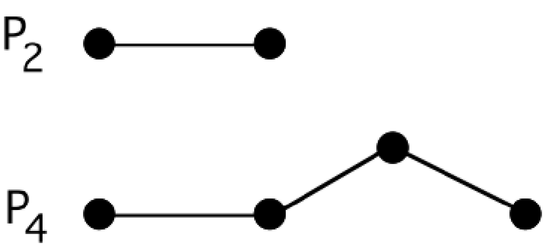
Bouquets, Dipoles, Path Graphs, and Cycle Graphs

- **DEFINITION**: A graph consisting of a single vertex with n self-loops is called a ***bouquet*** and is denoted B_n.



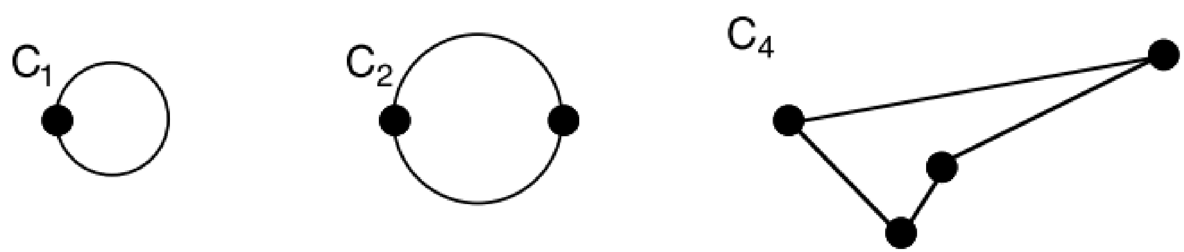
- **DEFINITION**: A graph consisting of two vertices and n edges joining them is called a ***dipole*** and is denoted D_n.

$|V_P|=|E_P|+1$



Path graphs P_2 and P_4

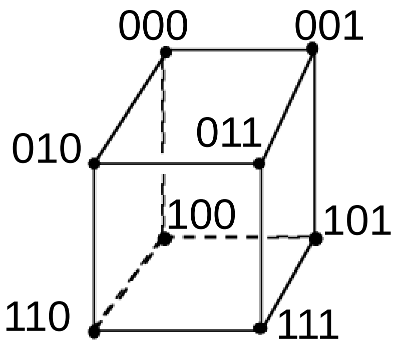
$|V_P|=|E_P|$



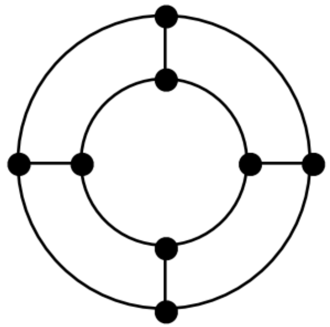
Cycle graph C_1, C_2, and C_4

Hypercubes, Circular Ladders, and Circulant Graphs

- Hypercube: Q_3

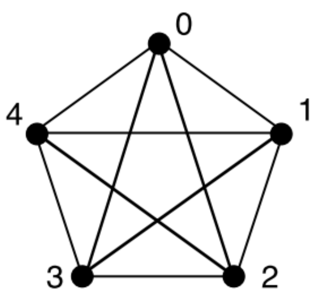


- Circular ladder graph: CL_4

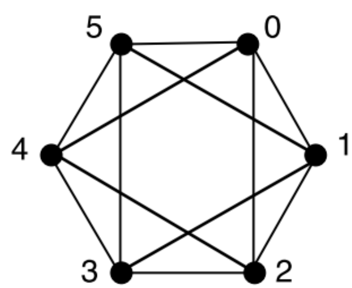


- **DEFINITION**: To the group of integers $Z_n=\{0,1,\ldots,n\text{-}1\}$ under addition modulo n and a set $S\subseteq\{1,2,\ldots,n\text{-}1\}$, the ***circulant graph*** $circ(n : S)$ has a vertex set Z_n, and two vertices i and j are adjacent if and only if there is a number $s\in S$ such that $i+s = j \bmod n$ or $j+s = i \bmod n$.
  - ✓ The elements of the set S are called ***connections***.

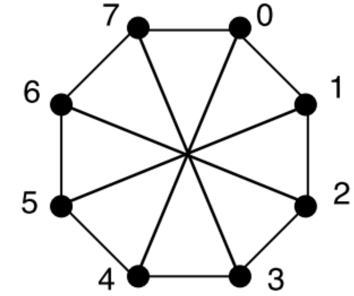
$\mathbf{circ}(\mathbf{5} : \mathbf{1}, \mathbf{2})$



$\mathbf{circ}(\mathbf{6} : \mathbf{1}, \mathbf{2})$



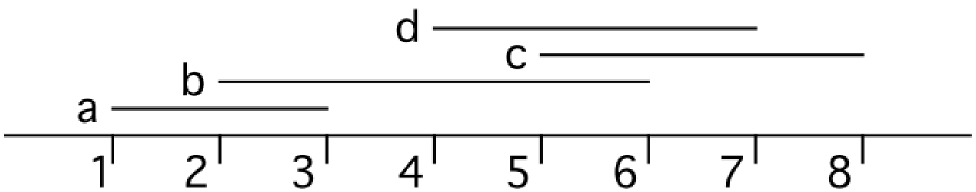
$\mathbf{circ}(\mathbf{8} : \mathbf{1}, \mathbf{4})$

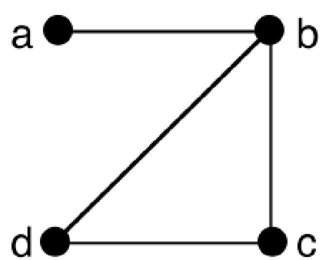


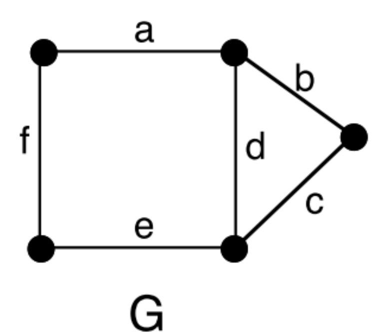
Intersection, Interval Graphs, and Line Graphs

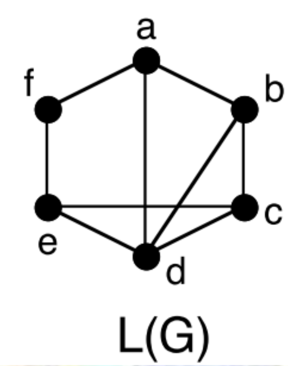
- **DEFINITION**: A simple graph G with vertex-set $V_G=\{v_1, v_2, \ldots, v_n\}$ is an ***intersection graph*** if there exists a family of sets $F=\{S_1, S_2, \ldots, S_n\}$ such that vertex v_i is adjacent to v_j if and only if $i \neq j$ and $S_i \cap S_j \neq \emptyset$.
- **DEFINITION**: A simple graph is an ***interval graph*** if it is an intersection graph corresponding to a family of intervals on the real line.
- **DEFINITION**: The ***line graph*** $L(G)$ of a graph G has a vertex for each edge of G, and two vertices in $L(G)$ are adjacent if and only if the corresponding edges in G have a vertex in common.

$a \leftrightarrow (1,3) \qquad b \leftrightarrow (2,6) \qquad c \leftrightarrow (5,8) \qquad d \leftrightarrow (4,7)$



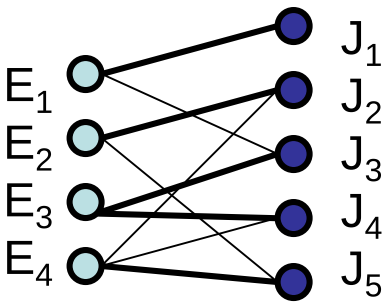


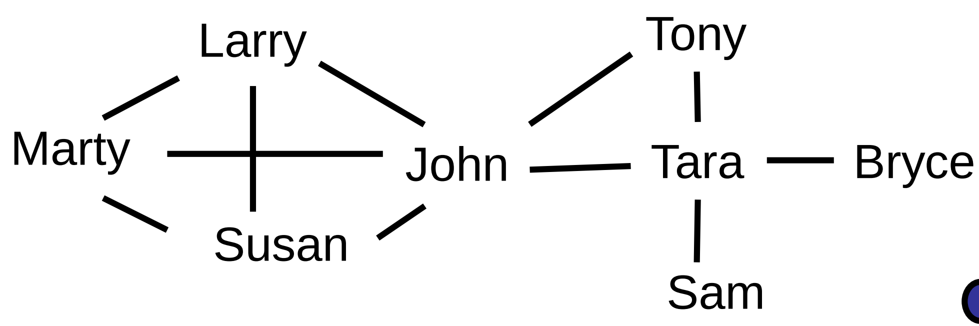


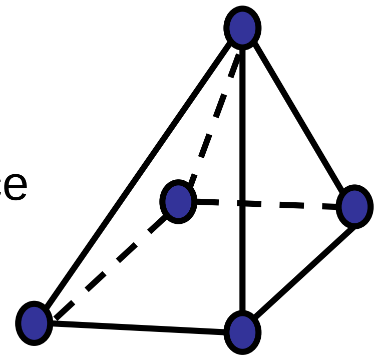


1.3 Graph Modeling Applications

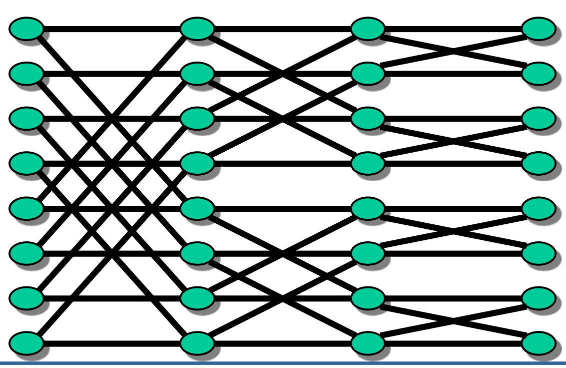
- An optimal assignment of employees to jobs
- An acquaintance network
- Geometric Polyhedra – A non-regular 1-skeleton of a polyhedron



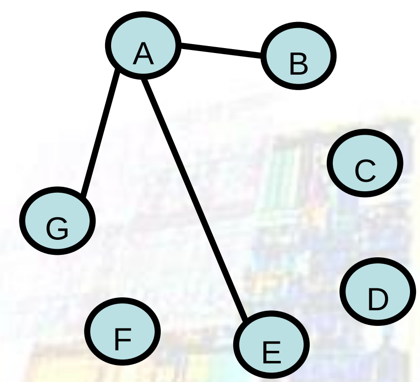




- Butterfly interconnection
- Assigning broadcasting frequencies

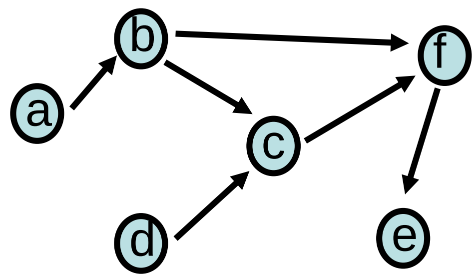


	B	C	D	E	F	G
A	55	110	108	60	150	88
B		87	142	133	98	139
C			77	91	85	93
D				75	114	82
E					107	41
F						123



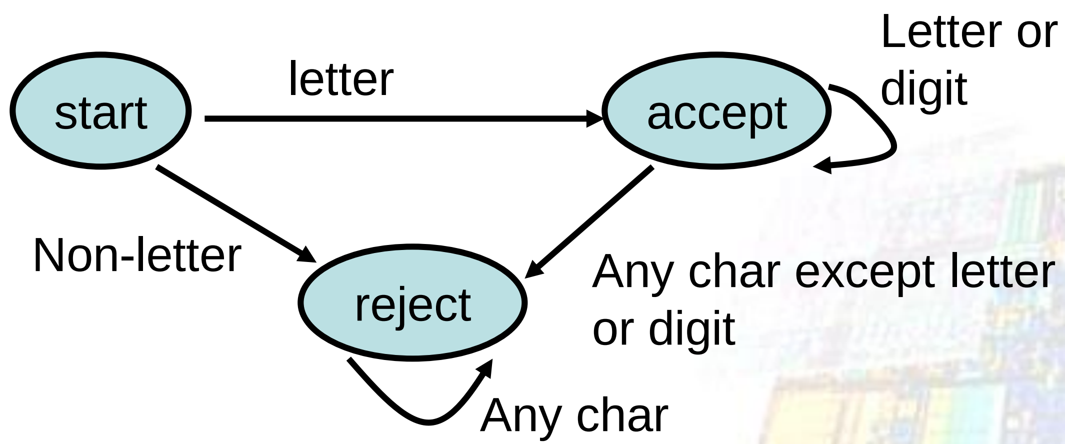
1.3 Graph Modeling Applications

- Activity-scheduling networks



- Flow diagrams for computer programs (auto-tracing & auto-verification)

- Lexical Scanners
  - ✓ Ex. **if** and **ifabc**

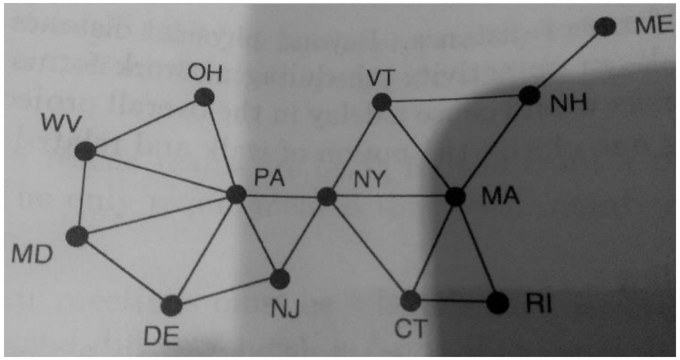


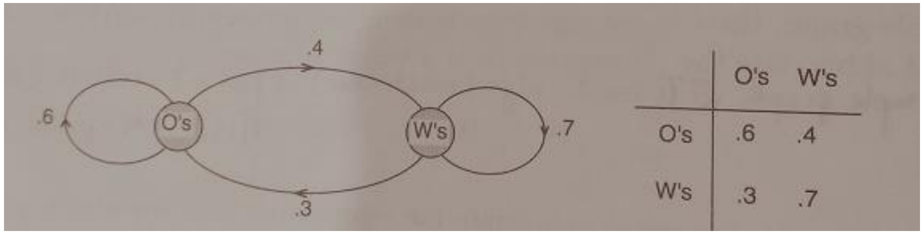
1.4 Walks and Distance – Walks and Directed Walks

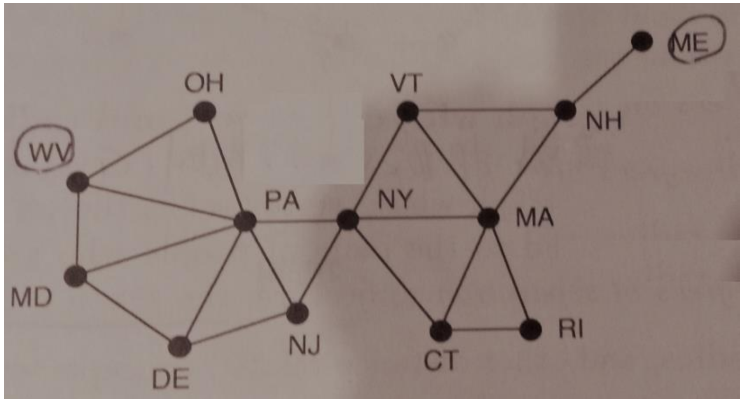
- **DEFINITION**: In a graph G, a walk from vertex v_0 to vertex v_n is an alternating sequence $W = < v_0, e_1, v_1, e_2, \ldots, v_{n-1}, e_n, v_n >$ of vertices and edges, such that endpts(e_i) = $\{v_{i-1}, v_i\}$, for $i = 1, \ldots, n$.
  - ✓ For directed graph, W is a ***directed walk*** if each edge e_i is directed from vertex v_{i-1} to vertex v_i. ($tail(e_i) = v_{i-1}$ and $head(e_i) = v_i$)
- For a simple graph, $W = < v_0, v_1, \ldots, v_n>$
- For a general graph, $W = < v_0, e_1, e_2, \ldots, e_n, v_n>$

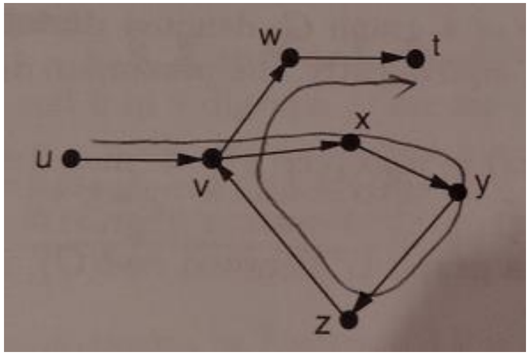
- **DEFINITION**: The ***length*** of a walk or directed walk is the number of edge-steps in the walk sequence.
- **DEFINITION**: A ***close walk*** (or ***closed directed walk***) is a nontrivial walk (or directed walk) that begins and ends at the same vertex. An ***open walk*** (or ***open directed walk***) begins and ends at different vertices.
- **DEFINITION**: The ***concatenation*** of two walks $W_1 = < v_0, e_1, \ldots, v_{k-1}, e_k, v_k>$ and $W_2 = < v_k, e_{k+1}, v_{k+1}, e_{k+2}, \ldots, v_{n-1}, e_n, v_n>$ such that walk W_2 begins where walk W_1 ends, is the walk $W_1 \circ W_2 = < v_0, e_1, \ldots, v_{k-1}, e_k, v_k, e_{k+1}, \ldots, v_{n-1}, e_n, v_n >$
- ***Sub-walk***

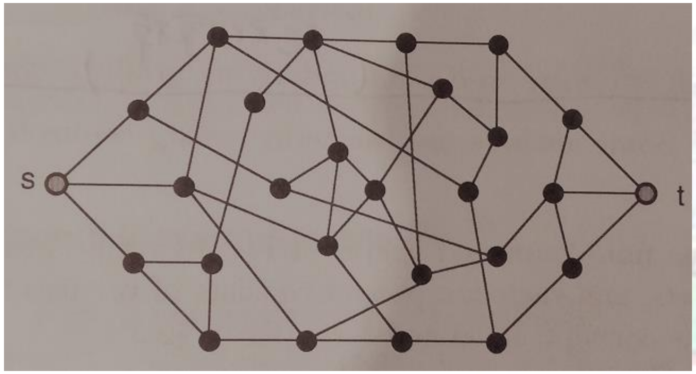
1.4 Walks and Distance – Walks and Directed Walks









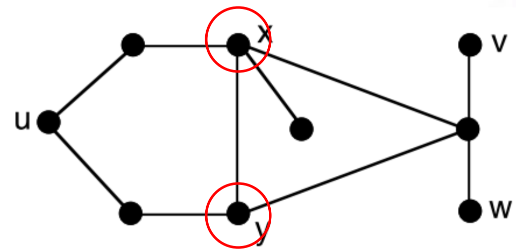


Distance, Eccentricity, Diameter, and Radius

- **DEFINITION**: The ***distance*** $d(s, t)$ from a vertex s to a vertex t in a graph G is the length of a shortest s-t walk if one exists; otherwise, $d(s, t) = \infty$
  - ✓ ***Directed distance***
- **DEFINITION**: The ***eccentricity*** of a vertex v in a graph G, denoted $ecc(v)$, is the distance from v to a vertex farthest from v. That is, $ecc(v) = \max_{x \in V_G}\{d(v, x)\}$
- **DEFINITION**: The ***diameter*** of a graph G, denoted $diam(G)$, is the maximum of the vertex eccentricities in G or, equivalently, the maximum distance between two vertices in G. That is

$$diam(G) = \max_{x \in V_G}\{ecc(x)\} = \max_{x, y \in V_G}\{d(x, y)\}$$

- **DEFINITION**: The ***radius*** of a graph G, denoted $rad(G)$, is the mnimum of the vertex eccentricities. That is, $rad(G) = \min_{x \in V_G}\{ecc(x)\}$
- **DEFINITION**: A ***central vertex*** v of a graph G is a vertex with minimum eccentricity. Thus, $ecc(v) = rad(G)$.
- **Example.** $diam(G) = 4$, $rad(G) = 2$

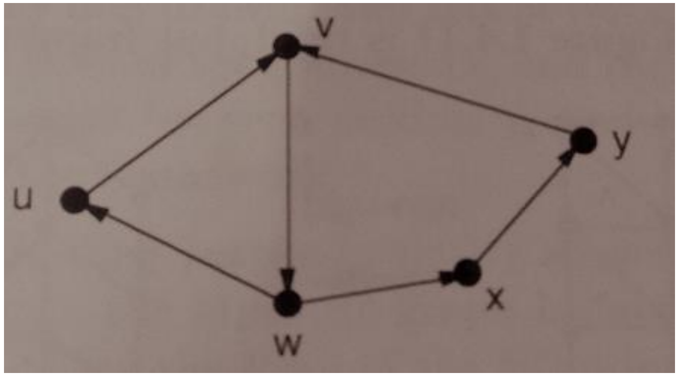


Connectedness, Strongly Connected Digraph

- **DEFINITIONS**: ***reachable from; connected; connected* (digraph);**

- **DEFINITIONS**: ***mutually reachable; strongly connected***

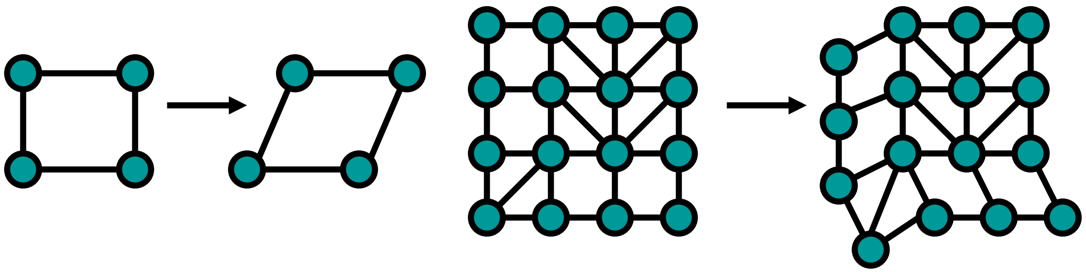
- **DEFINITIONS**: ***strongly orientable; strong orientation***



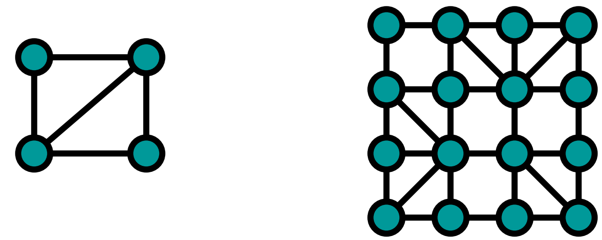
Application of Connectedness to Rigidity

- *Rigidity of Rectangular Fameworks*: Consider a 2-dimensional framework of steel beams connected by joints that allow each beam to swivel in the plane.
- **DEFINITION**: the framework is said to be ***rigid*** if none of its beams can swivel.

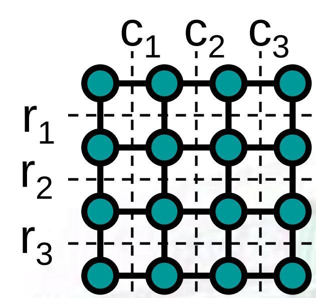
Non-rigid

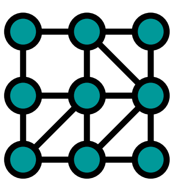


rigid



- Checking rule: Rigidity can be judged by checking if every rectangle must keep their center lines in perpendicular
  - ✓ A diagonal brace in a rectangle at row i and column $j \rightarrow\ r_i \perp c_j$
  - ✓ $\perp$ is not completely transitive but can be formed as a sequence
    - ➢ $r_i \perp c_j$ & $c_j \perp r_k \rightarrow r_i /\!/ r_k$ and $r_i \perp c_j \perp r_k$ but not $r_i \perp r_k$





$r_1 \perp c_2$ & $c_2 \perp r_2$ & $c_1 \perp r_2$

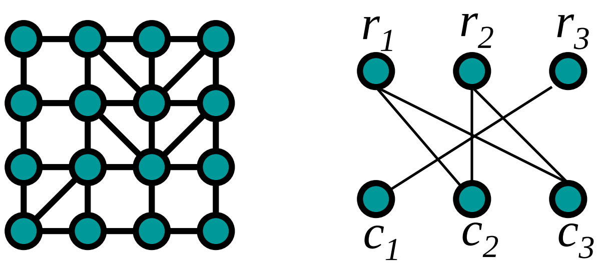
$r_1 \perp c_2 \perp r_2 \perp c_1 \rightarrow r_1 \perp c_1$

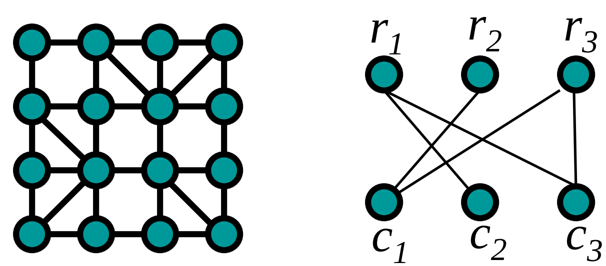
$r_1 \perp c_2$ & $r_1 \perp c_3$ & $r_2 \perp c_2$ & $r_2 \perp c_3$ & $r_3 \perp c_1$

$r_1 \perp c_2 \perp r_2 \perp c_3 \perp r_1$ & $r_3 \perp c_1$

Application of Connectedness to Rigidity

- **OBSERVATION**: If every pair row i and column j has a related perpendicularity sequence starting at r_i and ending at c_j then the framework is rigid.
- We can build a bipartite graph for perpendicularity sequence by forming a row node set and a column node set and inserting an edge between a row node and a column node if they have perpendicularity relation.





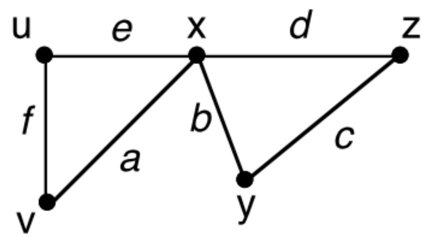
$r_1 \perp c_2$ & $r_1 \perp c_3$ & $r_2 \perp c_1$ & $r_3 \perp c_1$ & $r_3 \perp c_3$

$c_2 \perp r_1 \perp c_3 \perp r_3 \perp c_1 \perp r_2$

- **Theorem 1.4.1.** *A rectangular framework is rigid if only if its associated bipartite graph is connected.*

1.5 Paths, Cycles, and Trees

- **DEFINITION**: ***trail*** (no repeated edges)***; path*** (no repeated vertices and edges)***; trivial***
- **TERMINOLOGY**: no universally agreed-upon terminology for walks, trail, and paths.



W=<*v*, *a*, *e*, *f*, *a*, *d*, *z*>

T = <v, a, b, c, d, e, u>

- **DEFINITION**: Given a walk $W = < v_0, e_1, v_1, \ldots, v_{n-1}, e_n, v_n >$ that contains a nontrivial closed subwalk $W' = < v_k, e_{k+1}, v_{k+1}, \ldots, v_{m-1}, e_m, v_k >$, the ***reduction of walk** W **by subwalk** W'*, denoted $W - W'$, is the walk

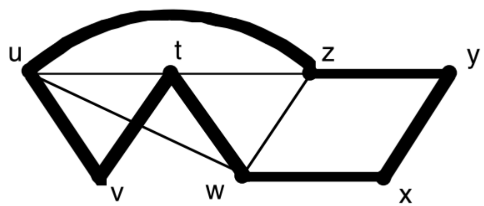
$$W - W' = < v_0, e_1, \ldots, v_{k-1}, e_k, v_k, e_{m+1}, v_{m+1}, \ldots, v_{n-1}, e_n, v_n >$$

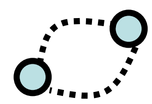
Thus, $W - W'$ is obtained by deleting from W all of the vertices and edges of W' except v_k.

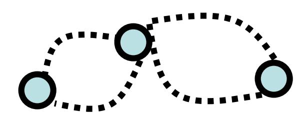
- **DEFINITION**: Given two walks A and B, the walk B is said to be a ***reduced walk of** A* if there exists a sequence of walks $A = W_1, W_2, \ldots, W_r = B$ such that for each $i = 1, \ldots, r - 1$, walk W_{i+1} is the reduction of W_i by some closed subwalk of W_i.

Deleting Closed Subwalks from Walks and Cycles

- **Lemma 1.5.1.** Every open *x*-*y* walk *W* is either an *x*-*y* path or contains a closed subwalk.
- **Theorem 1.5.2.** Let *W* be an open *x*-*y* walk. Then either *W* is an *x*-*y* path or there is an *x*-*y* path that is a reduced walk of *W*.
- **Corollary 1.5.3.** The distance from a vertex *x* to a reachable vertex *y* is always realizable by an *x*-*y* path.







- **DEFINITION**: ***cycle; acyclic graph; hamiltonian cycle; hamiltonian graph***
- **Theorem 1.5.4.** A graph *G* is bipartite if and only if it has no cycles of odd length.

Necessity ⇒ In a bipartite, a cycle must return to original set

Sufficiency ⇐ Assume *G* is connected. Get a partition (*X*,*Y*) of *V* as follows

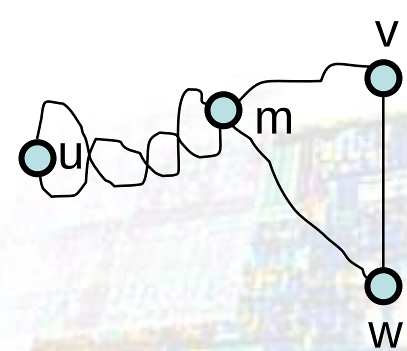
pick any vertex *u*, $X = \{x \mid d(u,x) \text{ is even}\}$; $Y = \{y \mid d(u,y) \text{ is odd}\}$;

assume (*X*,*Y*) is not a bipartition of *G* → $e=(v,w)$, *v* and *w* are in *X* or *Y*

Let P_1 and P_2 be the shortest path from *u* to *v* and *w*. The lengths of P_1 and P_2 are both even or odd. There is at least one common point between P_1 and P_2, say *m*.

P_1 and P_2 have the same length from *u* to *m*.

Cycle (*m*, *v*, *w*, *m*) is odd-length.



Deleting Closed Subwalks from Walks and Cycles

- **Proposition 1.5.5.** Every non-trivial, closed trail T contains a subwalk that is a cycle.

  Let T' be a minimum length, nontrivial, closed subwalk of $T \rightarrow$ no proper closed subwalk.

- **Remark.** The assertion of Proposition 1.5.5 is no longer true if T is merely a closed walk.

- **DEFINITION**: A collection of edge-disjoint cycles, C_1, C_2, …, C_m, is called a ***decomposition*** of a closed trail T if each cycle C_i is either a subwalk or a reduced walk of T and the edge-sets of the cycles ***partition*** the edge-set of trail T.

- **Theorem 1.5.6.** A closed trail can be decomposed into edge-disjoint cycles.

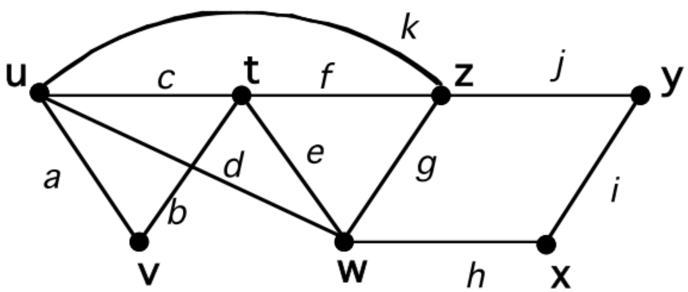
  prove by induction:

  a closed trail with only one edge is itself a cycle. Assume that the theorem holds for all closed trails with m or fewer edges.

  for the edges > m, By the proposition 1.5.5, the trail must contain a cycle C. T-C can be decomposed into edge-disjoint cycles => T also can be decomposed into edge-disjoint cycles.

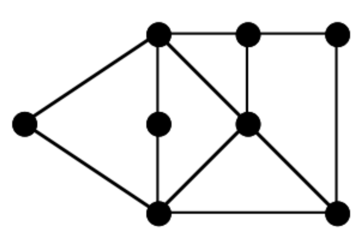
Cycles and Eulerian Trails

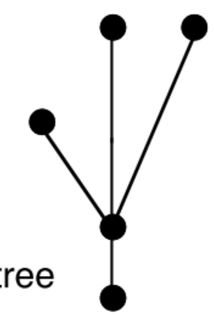
- Edge-disjoint cycles



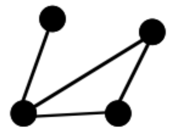
T = <*u, a, b, c, d, e, f, g, h, i, j, k, u*>
C_1= <*z, k, c, f, z*>, <*u, a, b, c, u*>
C_2 = <*u, a, b, e, d, u*>, <*w, e, f, g, w*>
C_3 = <*w, h, i, j, g, w*>, <*u, d, h, i, j, k, u*>

- **DEFINITION**: ***Eulerian trail, eulerian tour, eulerian graph***
- **DEFINITION**: The ***girth*** of a graph *G* with at least one cycle is the length of a shortest cycle in *G*.
- **DEFINITION**: A ***tree*** is a connected graph that has no cycles.

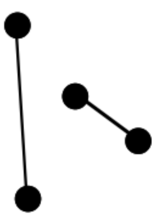




tree



non-tree



non-tree