VCU, Department of Computer Science

CMSC 302

Graphs

Vojislav Kecman

A good picture is worth a thousand words

- **Expressive power** is **the first explanation** for a **success of graphs**
- More claims for graphs come later
- Example for a title above follows!

Greek pre-Socratic Philosophers

- Thales of Miletus influenced Anaximander, Pythagoras, Heraclitus and Anaximenses of Miletus
- Anaximander infl. Pythagoras
- Pherecides of Syros infl. Pythagoras
- Anaximander infl. Heraclitus
- Pythagoras infl. Heraclitus
- Pythagoras infl. Empedocles
- Pythagoras infl. Philolaus
- Pythagoras infl. Archytas
- Pythagoras infl. Alcmaeon of Croton
- Philolaus infl. Archytas
- Heraclitus infl. Parmenides
- Parmenides infl. Democritus
- Democritus infl. Philolaus
- Parmenides infl. Melissus of Samos
- Parmenides infl. Socrates
- Leucippus infl. Democritus, (and this is about 60% of the story)

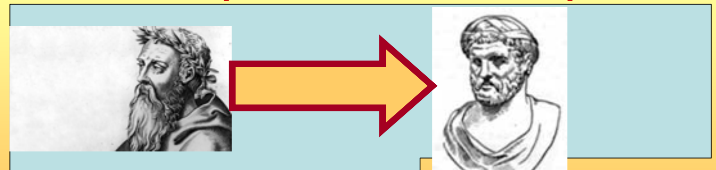
Question for you:

Did Heraclitus infl. Archytas?

Next question:

Did Pythagoras infl. Melissus of Samos?

Greek pre-Socratic Philosophers

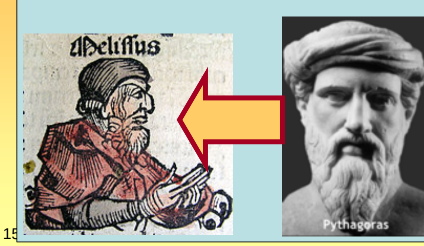


Question for you:

Did Heraclitus infl. Archytas?

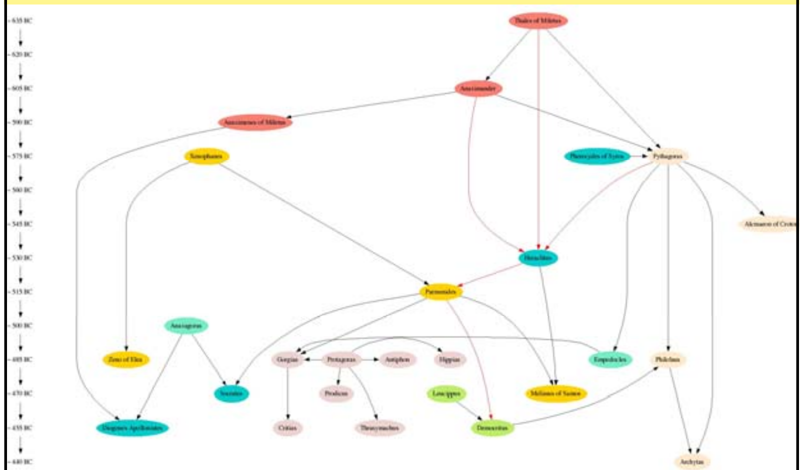
Next question:

Did Pythagoras infl. Melissus of Samos?



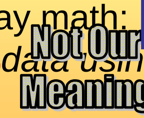
The whole Greek Pre-Socratic Philosopher in GRAPH, and same questions for you now:

Did Heraclitus infl. Archytas? Did Pythagoras infl. Melissus of Samos?



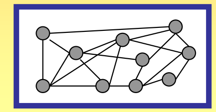
§ 9.1 What are Graphs?

- ~~General meaning in everyday math: *A plot or chart of numerical data using a coordinate system.*~~



- Technical meaning in **discrete mathematics**:

  A particular class of discrete structures (to be defined) that is useful for representing relations and has a convenient webby-looking graphical representation.

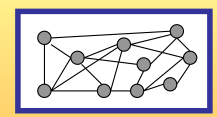


Applications of Graphs

- Potentially anything (graphs can represent relations, relations can describe the extension of any predicate).
- Apps in networking, scheduling, flow optimization, circuit design, path planning.
- Geneology analysis, computer game-playing, program compilation, object-oriented design, …

Simple Graphs

- Correspond to symmetric binary relations *R*.
- A *simple graph* *G*=(*V*,*E*) consists of:
  - **a set** *V* of *vertices* or *nodes* (*V* corresponds to the universe of the relation *R*),
  - **a set** *E* of *edges* (*arcs*, *links*): unordered pairs of (distinct) elements ***u*,*v*** $\in$ ***V***, such that ***uRv***.

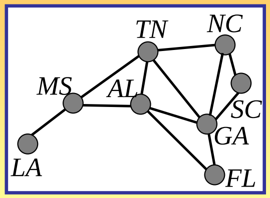


Visual Representation of a Simple Graph

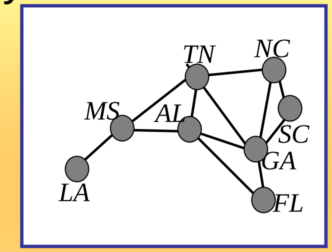
Note, in a simple graph there is only ONE EDGE between vertices & no ARROWS & no LOOPS

Example of a *Simple Graph*

- Let *V* be the set of states in the southeastern U.S.:
  - –*V*={FL, GA, AL, MS, LA, SC, TN, NC}
- Let *E*={{*u*,*v*}|*u* adjoins *v*}
  - ={{FL,GA},{FL,AL}, {GA,AL}, {GA,SC},{GA,TN}, {GA,NC}, {AL,MS}, {AL,TN}, {MS,LA}, {MS,TN}, {TN,NC},{NC,SC}}



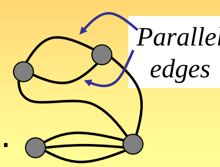
Well, what's simpler and clearer in representing adjacency of SE states ?



Sure, this is not bad either

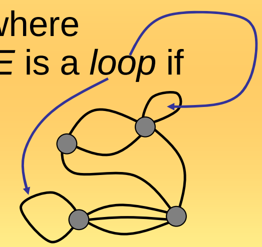
Multigraphs

- Like simple graphs, but there may be ***more than one* edge** connecting two given nodes.
- A *multigraph G*=(*V*, *E*, *f*) consists of a set *V* of vertices, a set *E* of edges (as primitive objects), and a function *f*: *E*→{{*u*,*v*}|*u*,*v*∈*V* ∧ *u*≠*v*}.
- e.g., nodes are cities, edges are segments of major highways.



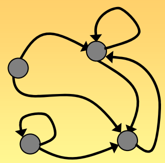
Pseudographs

- Like a multigraph, but **edges connecting a node to itself are allowed**.
- A *pseudograph G*=(*V*, *E*, *f*) where *f*:*E*→{{*u*,*v*}|*u*,*v*∈*V*}. Edge *e*∈*E* is a *loop* if *f*(*e*)={*u*,*u*}={*u*}.
- *e.g.*, nodes are campsites in a state park, edges are hiking trails through the woods.



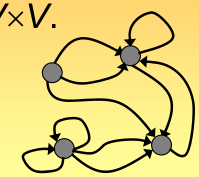
Directed Graphs

- Correspond to arbitrary binary relations *R*, which need not be symmetric.
- A *directed graph* (*V*,*E*) consists of a set of vertices *V* and a binary relation *E* on *V*.
- *E.g.*: *V* = people,
  E={(*x*,*y*) | *x* loves *y*}



Directed Multigraphs

- Like directed graphs, but there may be more than one arc from a node to another.
- A *directed multigraph G*=(*V*, *E*, *f*) consists of a set *V* of vertices, a set *E* of edges, and a function *f*:*E*→*V*×*V*.
- E.g., *V*=web pages, *E*=hyperlinks. *The WWW is a directed multigraph...*



Types of Graphs: Summary

- Summary of the book's definitions.
- Keep in mind this terminology is not fully standardized...

Term	Edge type	Multiple edges ok?	Self-loops ok?
Simple graph	Undir.	No	No
Multigraph	Undir.	Yes	No
Pseudograph	Undir.	Yes	Yes
Directed simple graph	Directed	No	Yes
Directed multigraph	Directed	Yes	Yes

§ 9.2 Graph Terminology

- *adjacent, or neighboring*
- *degree,*
- *connects,*
- *endpoints,*
- *initial,*
- *terminal,*
- *in-degree,*
- *out-degree,*

- *complete,*
- *cycles,*
- *wheels,*
- *n-cubes,*
- *bipartite,*
- *subgraph,*
- *union.*

Adjacency

Let *G* be an undirected graph with **edge set *E***. Let *e*∈*E* be (or map to) the pair {*u*,*v*}.

(Note that *u* and *v* are vertices!)

Then we say:

- *u*, *v* are *adjacent* / *neighbors* / *connected*.
- Edge *e* is *incident with* vertices *u* and *v*.
- Edge *e connects u* and *v*.
- Vertices *u* and *v* are *endpoints* of edge *e*.

Degree of a Vertex

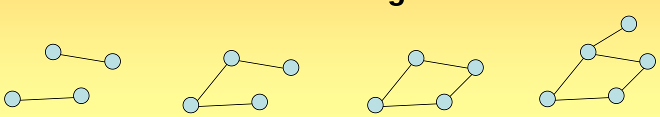
- Let *G* be an undirected graph, *v*∈*V* a vertex.
- The ***degree of v***, deg(*v*), is its number of incident edges. (*Except that any self-loops are counted twice*.)
- A vertex with degree 0 is *isolated*.
- A vertex of degree 1 is *pendant*.

Handshaking Theorem

- Let *G* be an undirected (simple, multi-, or pseudo-) graph with vertex set *V* and edge set *E*. Then

$$\sum_{v \in V} \deg(v) = 2|E|$$

- Proof: Each edge contributes twice to the degree count of all vertices
- Corollary: Any **undirected** graph has an **even** number of vertices of **odd degree**.



Example:

If a graph has 5 vertices, can each vertex have degree 3? 4?

Solution:

- The sum is 3•5 = 15 which is an odd number. Not possible.
- The sum is 20 = 2 | *E* | and 20/2 = 10. May be possible.

Question for a class: Is it possible to have a graph of 5 vertices each having degree 1?

Answer: It is not!!! (Sum of the degrees of graph is then five, and we know that it must be EVEN)

Directed Adjacency

- Let G be a directed (possibly multi-) graph, and let e be an edge of G that is (or maps to) (u,v). Then we say:
  - u is *adjacent to* v, v is *adjacent from* u
  - *e comes from* u, e *goes to* v.
  - *e connects u to v*, *e goes from u to v*
  - the *initial vertex* of e is u
  - the *terminal vertex* of e is v

Directed Degree

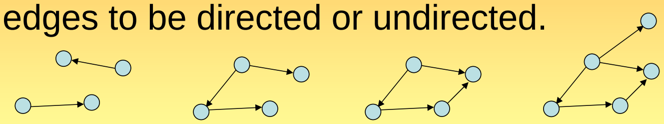
- Let G be a directed graph, v a vertex of G.
  - The *in-degree* of v, $\deg^{-}(v)$, is the number of edges going to v.
  - The *out-degree* of v, $\deg^{+}(v)$, is the number of edges coming from v.
  - The *degree* of v, $\deg(v) \equiv \deg^{-}(v) + \deg^{+}(v)$, is the sum of v's in-degree and out-degree.

Directed Handshaking Theorem

- Let G be a directed (possibly multi-) graph with vertex set V and edge set E. Then:

$$\sum_{v \in V} \deg^{-}(v) = \sum_{v \in V} \deg^{+}(v) = \frac{1}{2} \sum_{v \in V} \deg(v) = |E|$$

- Note that the **degree of a node is unchanged** by whether we consider its edges to be directed or undirected.



The visual **counting trick** here is – count the **heads** of arrows as **deg**⁻ and count the **ends** of the tails as **deg**⁺

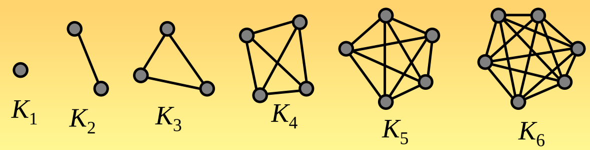
Special Graph Structures

Special cases of undirected graph structures:

- Complete graphs K_n
- Cycles C_n
- Wheels W_n
- n-Cubes Q_n
- Bipartite graphs
- Complete bipartite graphs $K_{m,n}$

Complete Graphs

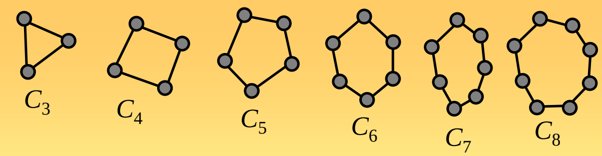
- For any $n \in \mathbf{N}$, a *complete graph* on n vertices, K_n, is a simple graph with n nodes in which every node is adjacent (connected) to every other node: $\forall u,v \in V$: $u \neq v \leftrightarrow \{u,v\} \in E$.



Note that K_n *has* $\sum_{i=1}^{n} i = \frac{n(n-1)}{2}$ *edges.*

Cycles

- For any $n \geq 3$, a *cycle* on n vertices, C_n, is a simple graph where $V=\{v_1,v_2,\ldots,v_n\}$ and $E=\{\{v_1,v_2\},\{v_2,v_3\},\ldots,\{v_{n-1},v_n\},\{v_n,v_1\}\}$.

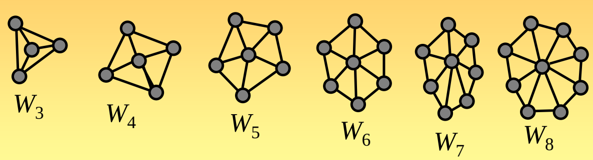


How many edges are there in C_n?

Note that in order to get a visually looking cycle, vertices should be sorted around a 'circle'

Wheels

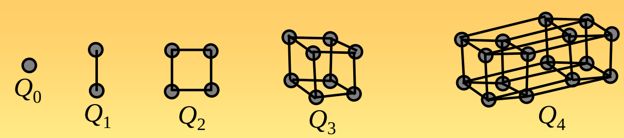
- For any $n \geq 3$, a *wheel* W_n, is a simple graph obtained by taking the cycle C_n and adding one extra vertex v_{hub} and n extra edges $\{\{v_{hub},v_1\}, \{v_{hub},v_2\},\ldots,\{v_{hub},v_n\}\}$.



How many edges are there in W_n?

n-cubes (hypercubes)

- For any $n \in \mathbf{N}$, the hypercube Q_n is a simple graph consisting of two copies of Q_{n-1} connected together at **corresponding** nodes. Q_0 has 1 node.

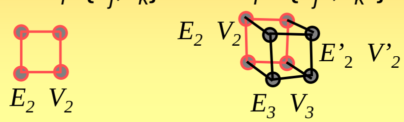


Number of vertices: 2^n. ***Number of edges***: *An exercise to try in class!*

$|E_n| = 2*|E_{n-1}|+|V_{n-1}|$

n-cubes (hypercubes)

- For any $n \in \mathbf{N}$, the hypercube Q_n can be defined recursively as follows:
  - $Q_0=\{\{v_0\},\varnothing\}$ (one node and no edges)
  - For any $n \in \mathbf{N}$, if $Q_n=(V,E)$, where $V=\{v_1,\ldots,v_a\}$ and $E_n=\{e_1,\ldots,e_b\}$, then $Q_{n+1}=(V\cup\{v_1',\ldots,v_a'\}$, $E_{n+1} = E_n\cup\{e_1',\ldots,e_b'\}\cup\{\{v_1,v_1'\},\{v_2,v_2'\},\ldots,\{v_a,v_a'\}\})$ where $v_1',\ldots,v_a'$ are new vertices, and where if $e_i=\{v_j,v_k\}$ then $e_i'=\{v_j',v_k'\}$.



Bipartite Graphs

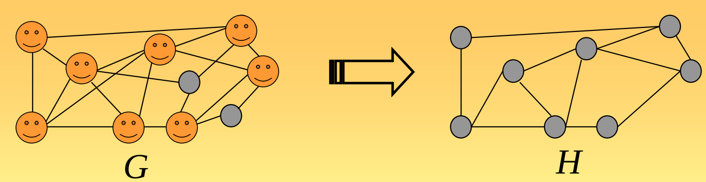
- Skipping this topic for this semester…

Complete Bipartite Graphs

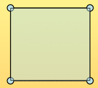
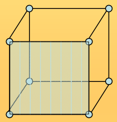
- Skip...

Subgraphs

- A subgraph of a graph $G=(V,E)$ is a graph $H=(W,F)$ where $W\subseteq V$ and $F\subseteq E$.



Subgraphs

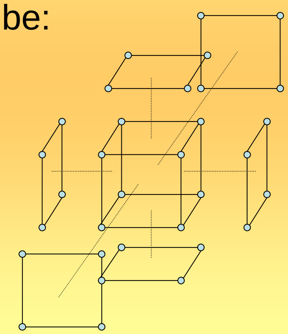
Notice that the 2-cube  occurs inside the 3-cube . In other words, Q_2 is a subgraph of Q_3:

Q: How many Q_2 subgraphs does Q_3 have?

6, see next slide!

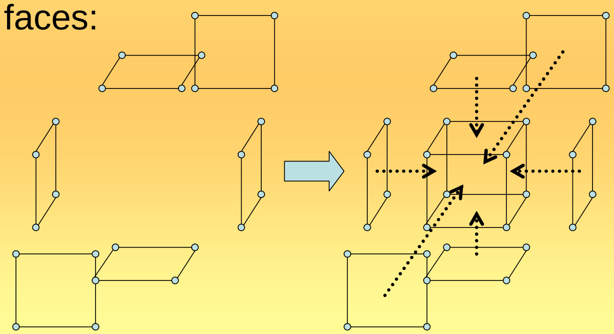
Subgraphs

A: Each face of Q_3 is a Q_2 subgraph so the answer is 6, as this is the number of sides of a 3-cube:



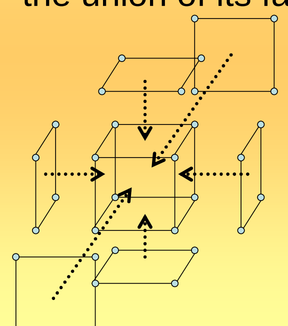
Graph Unions

In previous example one can actually reconstruct the 3-cube from its 6 2-cube faces:

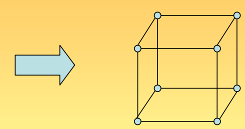


Unions

If we assign the 2-cube sides (i.e., squares) the names S_1, S_2, S_3, S_4, S_5, S_6 then Q_3 is the union of its faces:



$$Q_3 = S_1 \cup S_2 \cup S_3 \cup S_4 \cup S_5 \cup S_6$$



Graph Unions

- The *union* $G_1 \cup G_2$ of two simple graphs $G_1=(V_1, E_1)$ and $G_2=(V_2,E_2)$ (where V_1,V_2 may or may not be disjoint) is the simple graph $(V_1 \cup V_2, E_1 \cup E_2)$, i.e.,
  - $G_1 \cup G_2 = (V_1 \cup V_2, E_1 \cup E_2)$

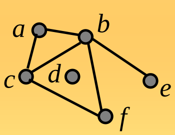
A similar definitions can be created for unions of digraphs, multigraphs, pseudographs, etc.

§ 9.3 Graph Representations & Isomorphism

- Graph representations:
  - Adjacency lists.
  - Adjacency matrices.
  - Incidence matrices.
- Graph isomorphism:
  - Two graphs are isomorphic iff they are identical except for their node names.

Adjacency Lists

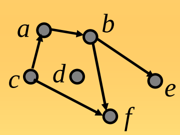
- A table with **1 row per vertex**, listing its adjacent vertices.



Vertex	*Adjacent Vertices*
a	*b, c*
b	*a, c, e, f*
c	*a, b, f*
d	
e	*b*
f	*c, b*

Directed Adjacency Lists

- **1 row per node**, listing the terminal nodes of each edge **incident** from that node.

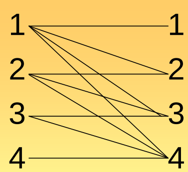


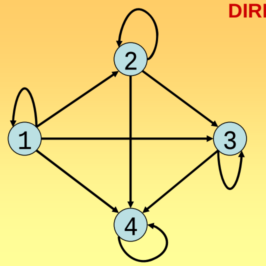
Vertex	*Adjacent Vertices*
a	*b,* [illegible]
b	[illegible], *e, f*
c	*a,* [illegible] *f*
d	
e	[illegible]
f	[illegible]

Adjacency Matrix

We already saw a way of representing relations on a set with a Boolean matrix:

R digraph(*R*) M_R



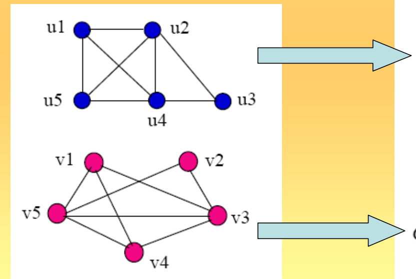


Note, the matrix is for a DIRECTED GRAPH on the left

$$\begin{pmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Adjacency Matrices

- Matrix $\mathbf{A}=[a_{ij}]$, where a_{ij} is 1 if $\{v_i, v_j\}$ is an edge of G, 0 otherwise.



$$G1 = \begin{bmatrix} 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 \end{bmatrix}$$

$$G2 = \begin{bmatrix} 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 \end{bmatrix}$$

Adjacency Matrix (Directed Multigraphs)

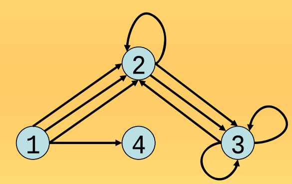
Can easily generalize to **directed** multigraphs by putting in the number of edges between vertices, instead of only allowing 0 and 1:

For a directed multigraph $G = (V,E)$ define the matrix A_G by:

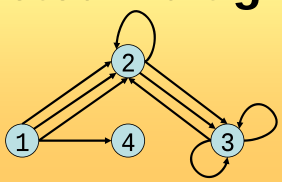
- Rows, Columns –one for each vertex in V
- Value at i th row and j th column is
  - **The number of edges with source the i th vertex and target the j th vertex**

Adjacency Matrix -Directed Multigraphs

Q: What is the adjacency matrix?



Adjacency Matrix -Directed Multigraphs

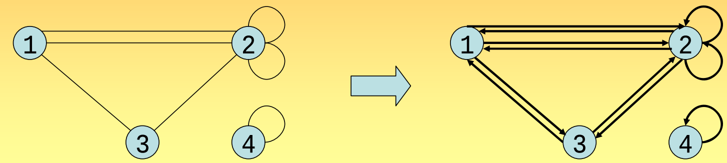


A:

$$\begin{pmatrix} 0 & 3 & 0 & 1 \\ 0 & 1 & 2 & 0 \\ 0 & 1 & 2 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

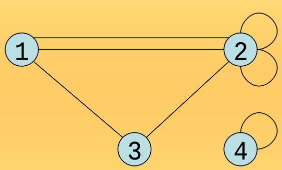
Adjacency Matrix-General

Undirected graphs can be viewed as directed graphs by turning each undirected edge into two oppositely oriented directed edges, *except when the edge is a self-loop in which case only 1 directed edge is introduced.* EG:

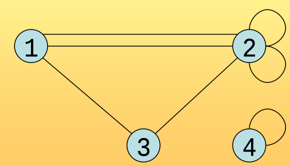


Adjacency Matrix-General

Q: What's the adjacency matrix?



Adjacency Matrix-General



A:

$$\begin{pmatrix} 0 & 2 & 1 & 0 \\ 2 & 2 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Notice that answer is ***symmetric***.

Adjacency Matrix-General

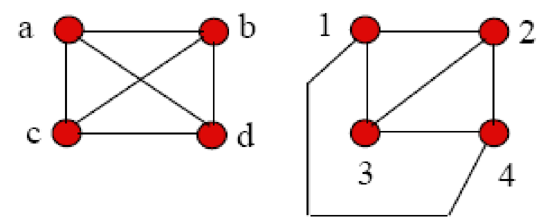
For an undirected graph $G = (V,E)$ define the matrix A_G by:

- Rows, Columns –one for each element of V
- **Value at i^{th} row and j^{th} column is the number of edges incidents with vertices i and j.**

Isomorphism

The two graphs below are really the same graph.

One is drawn so that no edges intersect (planar).



We say these graphs are *isomorphic*.

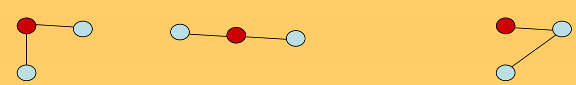
Graph Isomorphism

- Formal definition:
  - Simple graphs $G_1=(V_1, E_1)$ and $G_2=(V_2, E_2)$ are *isomorphic* iff $\exists$ a **bijection *f***: $V_1 \rightarrow V_2$ such that $\forall\, a,b \in V_1$, a and b are adjacent in G_1 iff $f(a)$ and $f(b)$ are adjacent in G_2.
  - ***f*** is the "renaming" function that makes the two graphs identical.
  - Definition can easily be extended to other types of graphs.

Graph Invariants under Isomorphism

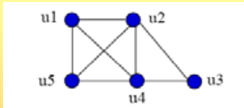
Necessary but not *sufficient* conditions for $G_1=(V_1, E_1)$ to be isomorphic to $G_2=(V_2, E_2)$:

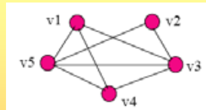
$|V1|=|V2|$, $|E1|=|E2|$.



- The number of vertices with degree n is the same in both graphs.
- For every proper subgraph g of one graph, there is a proper subgraph of the other graph that is isomorphic to g.

Are the following 2 graphs isomorphic?





Note! Proving isomorphism is a very hard problem. Doing it by hand is a bummer!!! **Why?**

Invariants - things that $G1$ and $G2$ must have in common to be isomorphic:

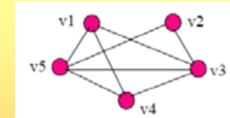
- the same number of vertices
- the same number of edges
- degrees of corresponding vertices are the same.
- if one is bipartite, the other must be
- if one is complete, the other must be
- if one is a wheel, the other must be

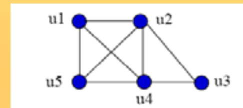
etc. ____________

Solution

Check . . .

- They have the same number of vertices = 5
- They have the same number of edges = 8
- They have the same number of vertices with the same degrees: 2, 3, 3, 4, 4.





- Now we try to construct the isomorphism f using the degrees of vertices to help us.
  - deg(u3) = deg(v2) = 2 so

    f(u3) = v2 [1]

is our only choice.

- deg(u1) = deg(u5) = deg(v1) = deg(v4) = 3 so we must have either [2] [3]

  i) f(u1) = v1 and f(u5) = v4

  or

  ii) f(u1) = v4 and f(u5) = v1

Perhaps either choice will work.

- Finally since deg(u2) = deg(u4) = deg(v3) = deg(v5) = 4 we must have either

  [4] i) f(u2) = v3 and f(u4) = v5 [5]

  or

  ii) f(u2) = v5 and f(u4) = v3.

We first try the relabeling using i) in each case to get the function [1] [2] [3] [4]

$3 \to 2, 1 \to 1, 5 \to 4, 2 \to 3, 4 \to 5$ [5]

- permute the rows and columns of the adjacency matrix of $G1$ using the above map to see if we get the adjacency matrix of $G2$.

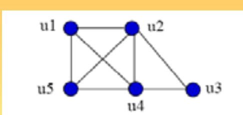
or

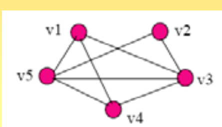
- change the labels of the graph $G2$ to produce the graph $G2^*$ according to the above permutation and recalculate the adjacency matrix.

Permutation Matrix

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

Recall:





Adj. matrices **without** relabeling

$$G1 = \begin{bmatrix} 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 \end{bmatrix}$$

$$G2 = \begin{bmatrix} 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 \end{bmatrix}$$

MATLAB CODE relabeling *i*

Instructor only: Run the code isomorphism_graphs.m

```
G1  =  [0 1 0 1 1
        1 0 1 1 1
        0 1 0 1 0
        1 1 1 0 1
        1 1 0 1 0]
P   =  [1 0 0 0 0
        0 0 1 0 0
        0 1 0 0 0
        0 0 0 0 1
        0 0 0 1 0]
% Multiply from LEFT to permute the ROWS
G1r = P*G1

% Multiply from RIGHT to permute the COLUMNS
G2_star = G1r*P
```

G1r:

$$\begin{matrix} 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 1 \end{matrix}$$

G2_star:

$$\begin{matrix} 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 \end{matrix}$$

MATLAB CODE relabeling *ii*

```
G1 =  [0 1 0 1 1
       1 0 1 1 1
       0 1 0 1 0
       1 1 1 0 1
       1 1 0 1 0]
P  =  [0 0 0 1 0
       0 0 0 0 1
       0 1 0 0 0
       0 0 1 0 0
       1 0 0 0 0]
% Multiply from LEFT to permute the ROWS
G1r = P*G1
```

0	1	0	1	1
0	1	0	1	0
1	0	1	1	1
1	1	0	1	0
1	1	1	0	1

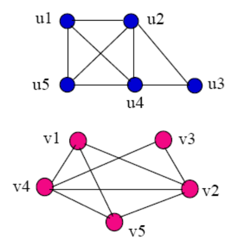
```
% Multiply from RIGHT to permute the COLUMNS
G2_star = G1r*P
```

0	0	1	1	1
0	0	1	0	1
1	1	0	1	1
1	0	1	0	1
1	1	1	1	0

15/10/2015 57/72

Solution, cont.

The new labeling of *G2*, *G2**, becomes



The new adjacency matrix becomes:

If we did the same steps by multiplying G2 by P from left and right we would have got

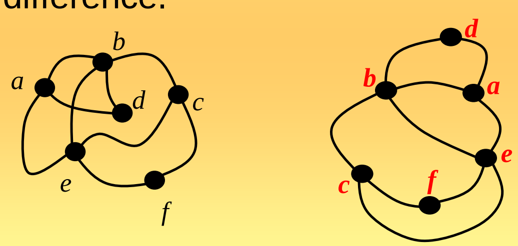
$$G2^* = \begin{bmatrix} 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 \end{bmatrix}$$

which is the same adjacency matrix as for G1. Hence we have found an isomorphism!

15/10/2015 58/72

Isomorphism Example

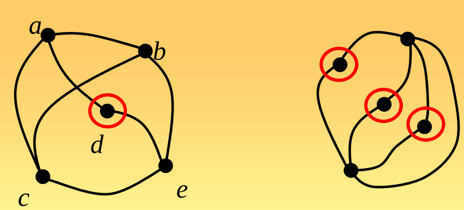
- If isomorphic, label the 2nd graph to show the isomorphism, else identify difference.



15/10/2015 59/72

Are These Isomorphic?

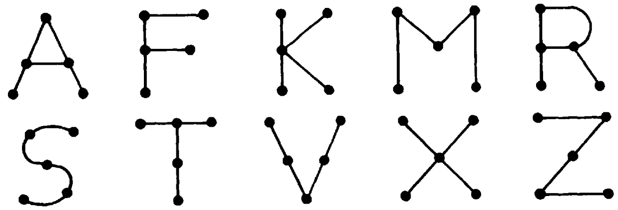
- If isomorphic, label the 2nd graph to show the isomorphism, else identify difference.



** Same # of vertices*
** Same # of edges*
** Different # of verts of degree 2! (1 vs 3)*

15/10/2015 60/72

Which of the graphs below are isomorphic?



A & R,

F & T,

K & X,

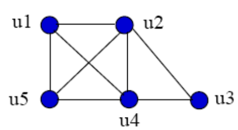
M, S, V & Z

§ 9.4 Connectivity

- In an undirected graph, a *path of length n from u to v* is a sequence of adjacent edges going from vertex u to vertex v.
- A path is a ***circuit*** if *u*=*v*, i.e., if it ends at *u*
- A path *traverses* the vertices along it.
- A path is ***simple*** if it contains no **edge more than once**.

Note: There is nothing to prevent traversing an edge back and forth to produce arbitrarily long paths. This is usually not interesting which is why we define a simple path.

Example:



There are many paths from u1 to u3 in G1:

1) u1, u4, u2, u3; length = 3, the path is simple

2) u1, u5, u4, u1, u2, u3; length = 5, the path is simple and it contains a circuit u1, u5, u4, u1.

3) u1, u2, u5, u4, u3; length = 4, the path is simple

How many simple paths are there?

5, 8, 9, 10, or more ?

Count!!!

It's tricky, isn't it?

Paths in Directed Graphs

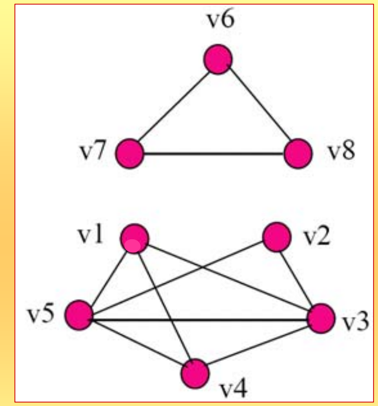
- Same as in undirected graphs, but the path must go in the direction of the arrows.

Connectedness

- An **undirected** graph is ***connected*** iff there is a path between **every pair of distinct vertices** in the graph.
- Theorem: There is a *simple* path between any pair of vertices in a connected undirected graph.
- *Connected component*: connected subgraph
- A *cut vertex* or *cut edge* separates 1 connected component into 2 if removed.

Example

Which of these 2 graphs is *a connected graph?*



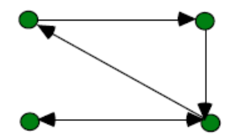
Both are!!!

Directed Connectedness

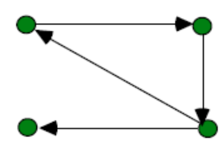
- A **directed** graph is ***strongly connected*** iff there is a directed path from *a* to *b* for any two vertices *a* and *b*.
- It is ***weakly connected*** iff the **underlying *undirected* graph** (*i.e.*, with edge directions removed) is connected.
- Note *strongly* implies *weakly* but not vice-versa.

Examples:

- strongly connected (hence weakly connected)



- not strongly connected but weakly connected.



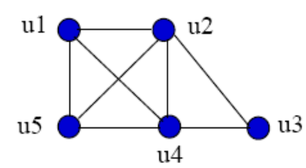
Paths & Isomorphism

- Note that connectedness, and the existence of a circuit or simple circuit of length *k* are graph invariants with respect to isomorphism.

Counting Paths w Adjacency Matrices

- Let **A** be the adjacency matrix of graph *G*.
- The number of paths of **length *k*** from v_i to v_j is equal to $(\mathbf{A}^k)_{i,j}$. (The notation $(\mathbf{M})_{i,j}$ denotes $m_{i,j}$ where $[m_{i,j}] = \mathbf{M}$.)

Example:



Caution!!!

- We are analyzing **undirected** graphs here
- So, there will be differences in respect to math we used in Transitive Closures
- There, we used Boolean Product
- Here, we'll use a classic/standard matrix product
- Hence, the matrices we'll get will tell us some new stories. They will give us some novel and different insights.

Number of 1-pathses are in **M**

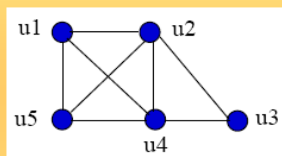
$$M = \begin{bmatrix} 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 \end{bmatrix}$$

How many **2-pathses** are there between vertices

1-1

1-2

2-2 ?



$$M^2 = \begin{bmatrix} 3 & 2 & 2 & 2 & 2 \\ 2 & 4 & 1 & 3 & 2 \\ 2 & 1 & 2 & 1 & 2 \\ 2 & 3 & 1 & 4 & 2 \\ 2 & 2 & 2 & 2 & 3 \end{bmatrix}$$

How many **3-pathse**s are there between vertices

1-1

1-2

2-2 ?

$$M^3 = \begin{bmatrix} 6 & 9 & 4 & 9 & 7 \\ 9 & 8 & 7 & 9 & 9 \\ 4 & 7 & 2 & 7 & 4 \\ 9 & 9 & 7 & 8 & 9 \\ 7 & 9 & 4 & 9 & 6 \end{bmatrix}$$

Reminder for lecturer only! RUN GRAPHS.M, now!!!

Here, the **Graphs** stories end, and the **Chapter 9** on **Trees** start.

As it may be suspected,
Trees are just special subgroups of **Graphs** **but,**
due to their importance and overall usefulness Trees are treated separately and in details!!!