

# Graphs

# Agenda –Graphs



## Graph basics and definitions

- Vertices/nodes, edges, adjacency, incidence
- Degree, in-degree, out-degree
- Degree, in-degree, out-degree
- Subgraphs, unions, isomorphism
- Adjacency matrices



## Types of Graphs

- Trees
- Undirected graphs
  - ◆ Simple graphs, Multigraphs, Pseudographs
- Digraphs, Directed multigraph
- Bipartite
- Complete graphs, cycles, wheels, cubes, complete bipartite

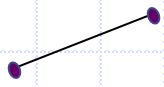
# Uses of Graph Theory in CS

- ◆ Car navigation system
- ◆ Efficient database
- ◆ Build a bot to retrieve info off WWW
- ◆ Representing computational models

Many other applications.

This course we focus more on the properties of algorithmic of graphs

# Graphs –Intuitive Notion

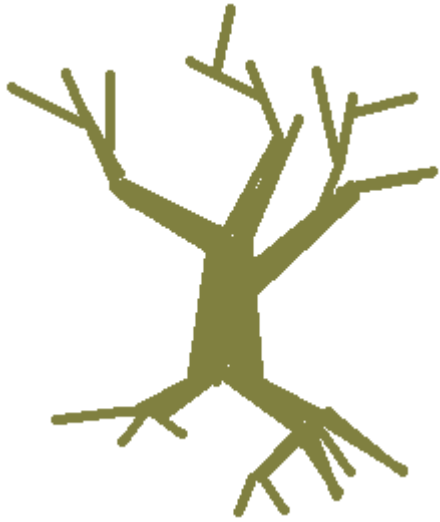
A graph is a bunch of vertices (or nodes) represented by circles • which are connected by edges, represented by line segments .

Mathematically, graphs are binary-relations on their vertex set (except for multigraphs).

In Data Structures one often starts with trees and generalizes to graphs. In this course, opposite approach: We start with graphs and restrict to get trees.

# Trees

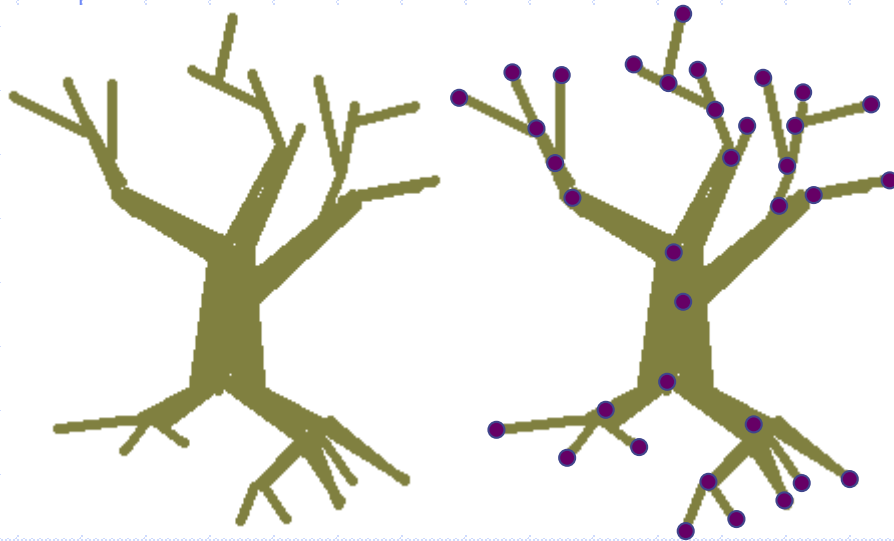
A very important type of graph in CS is called a *tree*:



Real  
Tree

# Trees

A very important type of graph in CS is called a *tree*:

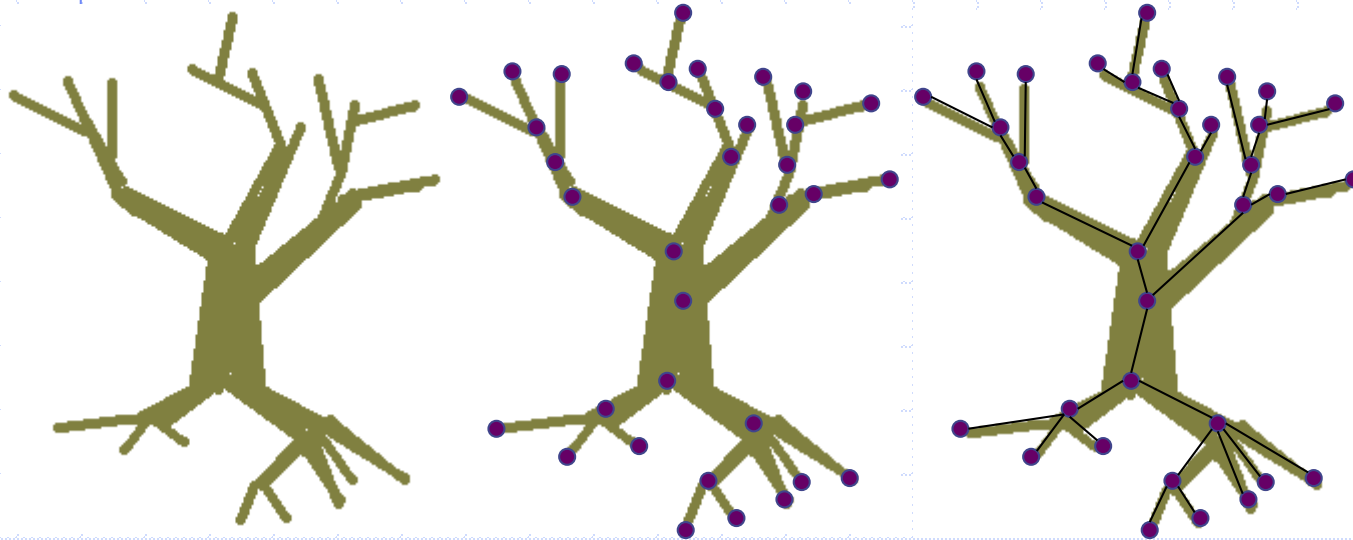


Real  
Tree

transformation

# Trees

A very important type of graph in CS is called a *tree*:



Real  
Tree

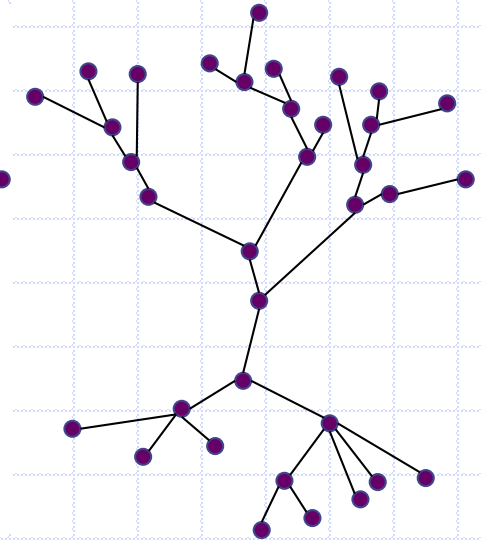
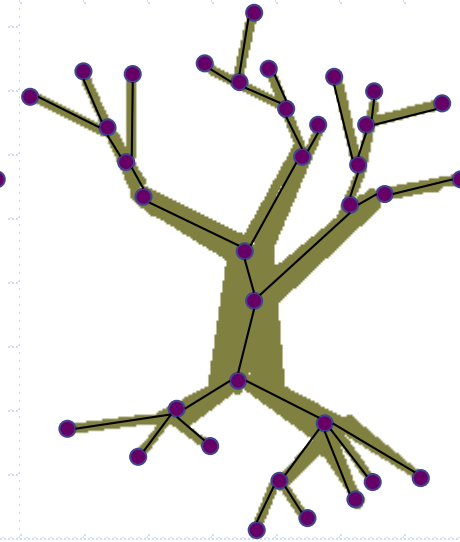
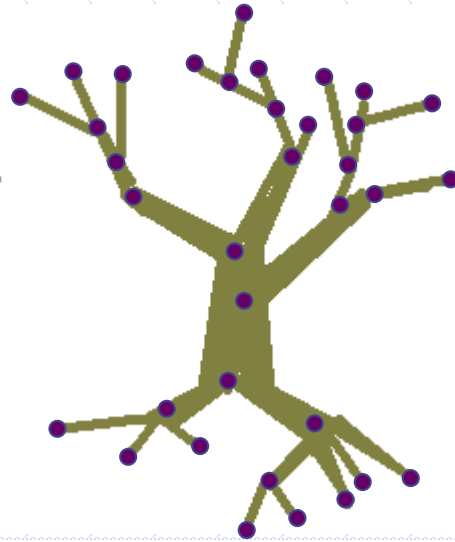
transformation

# Trees

A very important type of graph in CS is called a *tree*:



Real  
Tree



Abstract  
Tree

transformation

# Simple Graphs

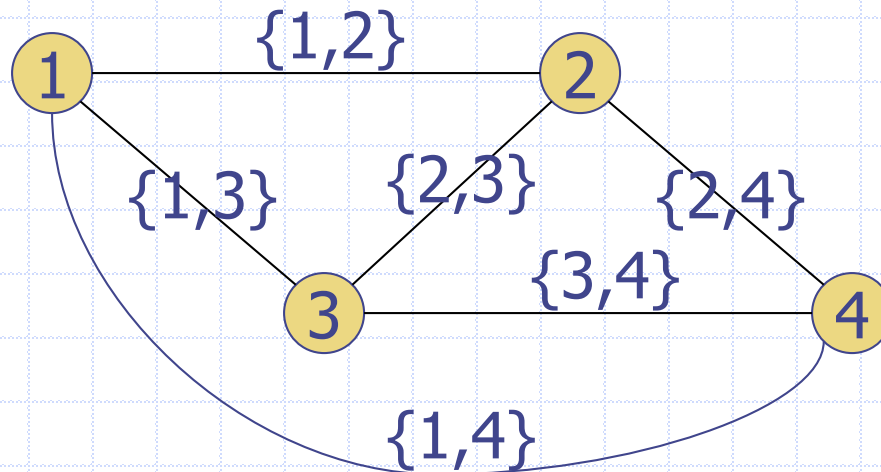
Different purposes require different types of graphs.

EG: Suppose a local computer network

- Is bidirectional (undirected)
- Has no loops (no “self-communication”)
- Has unique connections between computers

Sensible to represent as follows:

# Simple Graphs



- ◆ Vertices are labeled to associate with particular computers
- ◆ Each edge can be viewed as the set of its two endpoints

# Simple Graphs

DEF: A **simple graph**  $G = (V, E)$  consists of a non-empty set  $V$  of **vertices** (or **nodes**) and a set  $E$  (possibly empty) of **edges** where each edge is a subset of  $V$  with cardinality 2 (an unordered pair).

Q: For a set  $V$  with  $n$  elements, how many possible edges there?

# Simple Graphs

A: The number of pairs in  $V$   
 $= C(n,2) = n \cdot (n-1) / 2$

Q: How many possible graphs are there  
for the same set of vertices  $V$ ?

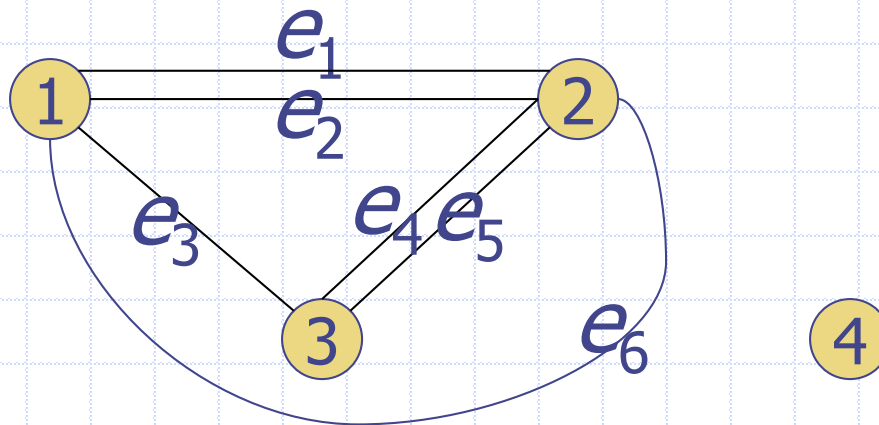
# Simple Graphs

A: The number of subsets in the set of possible edges. There are  $n \cdot (n-1) / 2$  possible edges, therefore the number of graphs on  $V$  is  $2^{n(n-1)/2}$

# Multigraphs

If computers are connected via internet instead of directly, there may be several routes to choose from for each connection. Depending on traffic, one route could be better than another. Makes sense to allow multiple edges, but still no self-loops:

# Multigraphs



Edge-labels distinguish between edges sharing same endpoints. Labeling can be thought of as function:

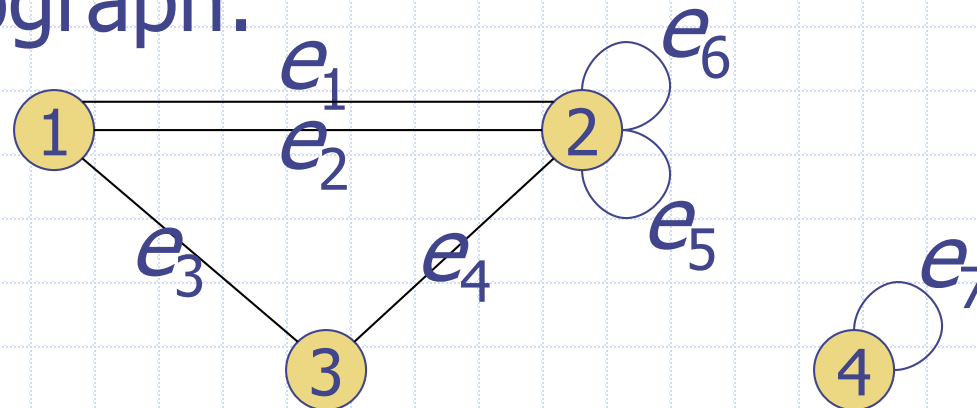
$$\begin{aligned} e_1 &\rightarrow \{1,2\}, e_2 \rightarrow \{1,2\}, e_3 \rightarrow \{1,3\}, \\ e_4 &\rightarrow \{2,3\}, e_5 \rightarrow \{2,3\}, e_6 \rightarrow \{1,2\} \end{aligned}$$

# Multigraphs

DEF: A ***multigraph***  $G = (V, E, f)$  consists of a non-empty set  $V$  of ***vertices*** (or ***nodes***), a set  $E$  (possibly empty) of ***edges*** and a function  $f$  with domain  $E$  and codomain the set of pairs in  $V$ .

# Pseudographs

If self-loops are allowed we get a pseudograph:



Now edges may be associated with a single vertex, when the edge is a loop

$$e_1 \rightarrow \{1,2\}, e_2 \rightarrow \{1,2\}, e_3 \rightarrow \{1,3\},$$

$$e_4 \rightarrow \{2,3\}, e_5 \rightarrow \{2\}, e_6 \rightarrow \{2\}, e_7 \rightarrow \{4\}$$

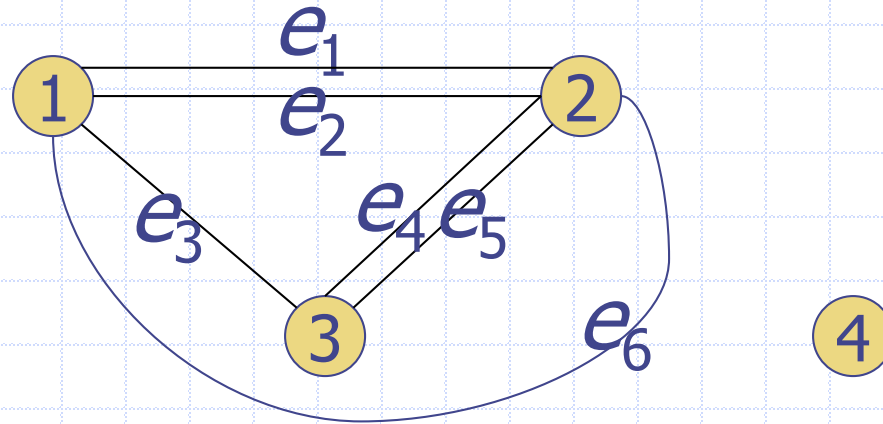
# Multigraphs

DEF: A **pseudograph**  $G = (V, E, f)$  consists of a non-empty set  $V$  of **vertices** (or **nodes**), a set  $E$  (possibly empty) of **edges** and a function  $f$  with domain  $E$  and codomain the set of pairs and singletons in  $V$ .

# Undirected Graphs

## Terminology

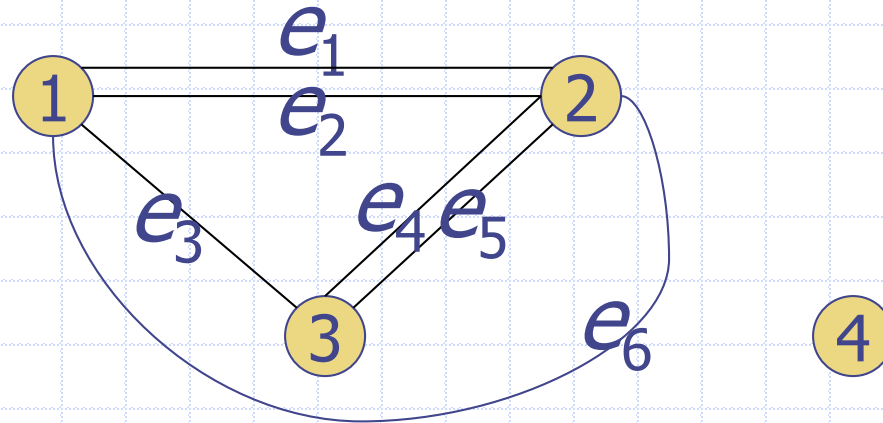
Vertices are ***adjacent*** if they are the endpoints of the same edge.



Q: Which vertices are adjacent to 1?  
How about adjacent to 2, 3, and 4?

# Undirected Graphs

## Terminology

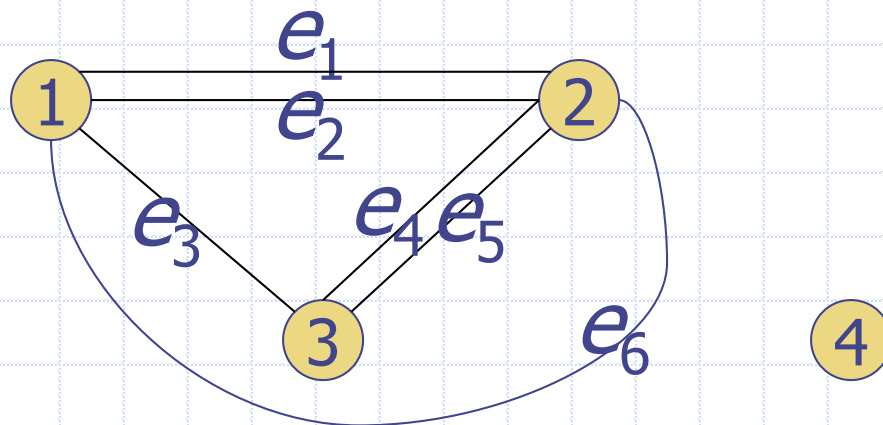


A: 1 is adjacent to 2 and 3  
2 is adjacent to 1 and 3  
3 is adjacent to 1 and 2  
4 is not adjacent to any vertex

# Undirected Graphs

## Terminology

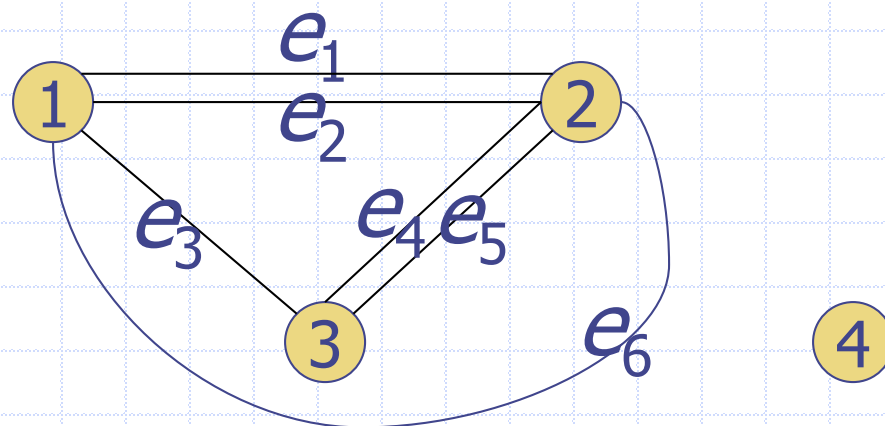
A vertex is ***incident*** with an edge (and the edge is incident with the vertex) if it is the endpoint of the edge.



Q: Which edges are incident to 1? How about incident to 2, 3, and 4?

# Undirected Graphs

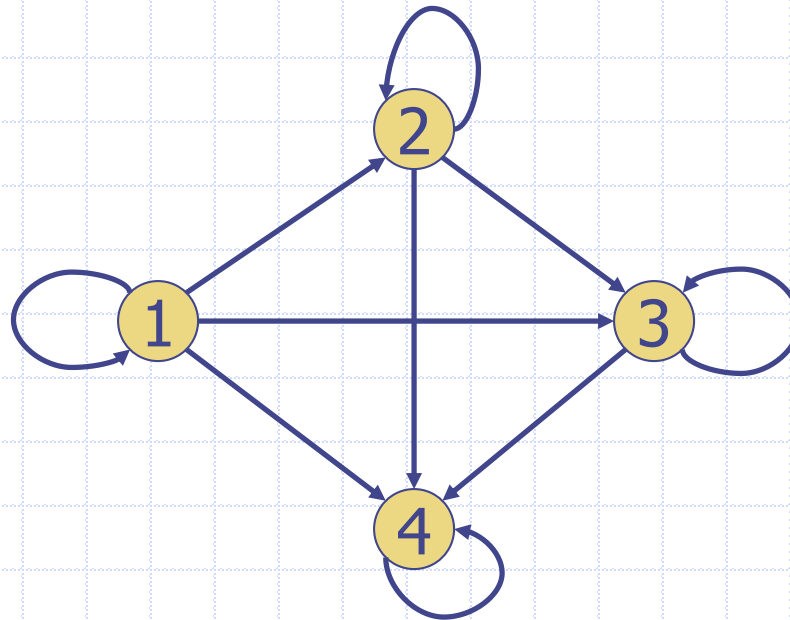
## Terminology



A:  $e_1, e_2, e_3, e_6$  are incident with 1  
1 is incident with  $e_1, e_2, e_3, e_4, e_5, e_6$   
2 is incident with  $e_3, e_4, e_5$   
3 is not incident with any edge

# Digraphs

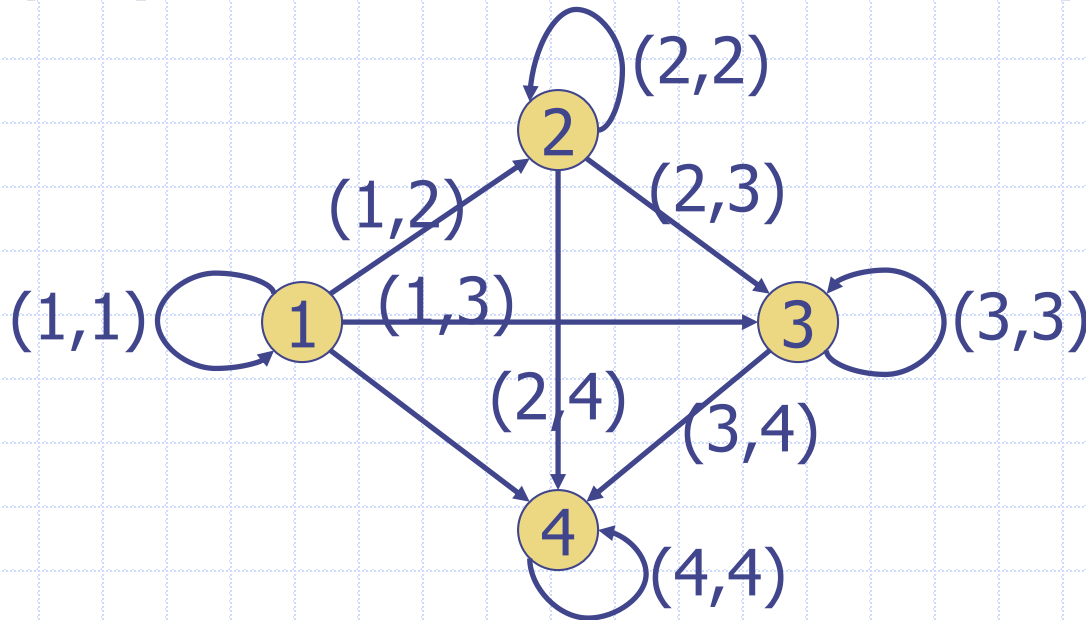
Last time introduced digraphs as a way of representing relations:



Q: What type of pair should each edge be (multiple edges not allowed)?

# Digraphs

A: Each edge is directed so an *ordered* pair (or tuple) rather than unordered pair.



Thus the set of edges  $E$  is just the represented relation on  $V$ .

# Digraphs

DEF: A **directed graph** (or **digraph**)  $G = (V, E)$  consists of a non-empty set  $V$  of **vertices** (or **nodes**) and a set  $E$  of **edges** with  $E \subseteq V \times V$ .

The edge  $(a, b)$  is also denoted by  $a \rightarrow b$  and  $a$  is called the **source** of the edge while  $b$  is called the **target** of the edge.

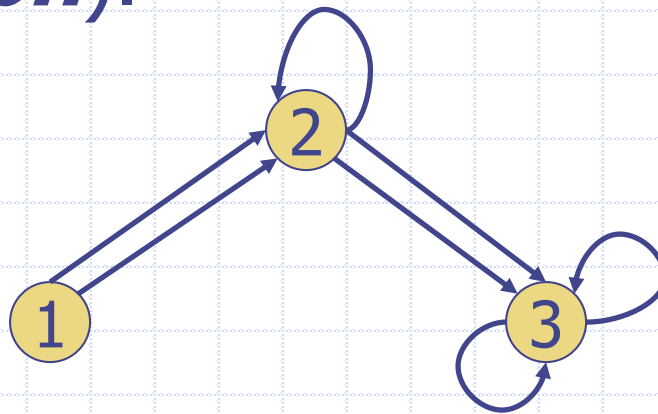
Q: For a set  $V$  with  $n$  elements, how many possible digraphs are there?

# Digraphs

A: The same as the number of relations on  $V$ , which is the number of subsets of  $V \times V$  so  $2^{n \cdot n}$ .

# Directed Multigraphs

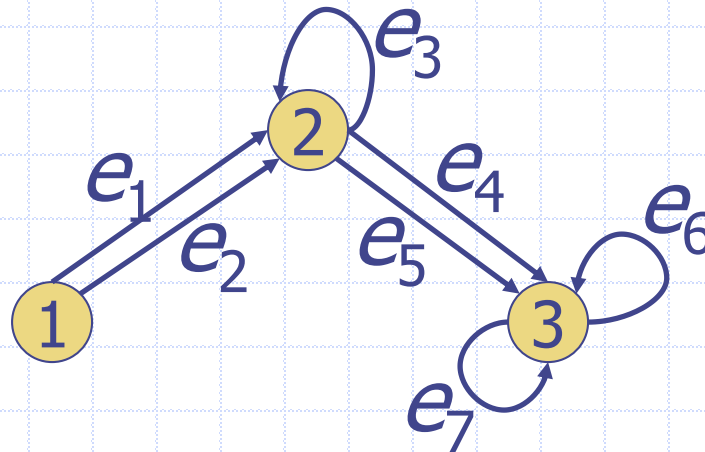
If also want to allow multiple edges in a digraph, get a ***directed multigraph*** (or ***multi-digraph***).



Q: How to use sets and functions to deal with multiple directed edges, loops?

# Directed Multigraphs

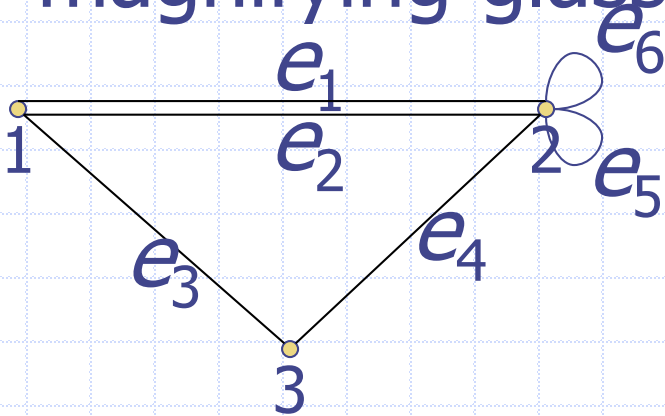
A: Have function with domain the edge set and codomain  $V \times V$ .



$e_1 \rightarrow (1,2)$ ,  $e_2 \rightarrow (1,2)$ ,  $e_3 \rightarrow (2,2)$ ,  $e_4 \rightarrow (2,3)$ ,  
 $e_5 \rightarrow (2,3)$ ,  $e_6 \rightarrow (3,3)$ ,  $e_7 \rightarrow (3,3)$

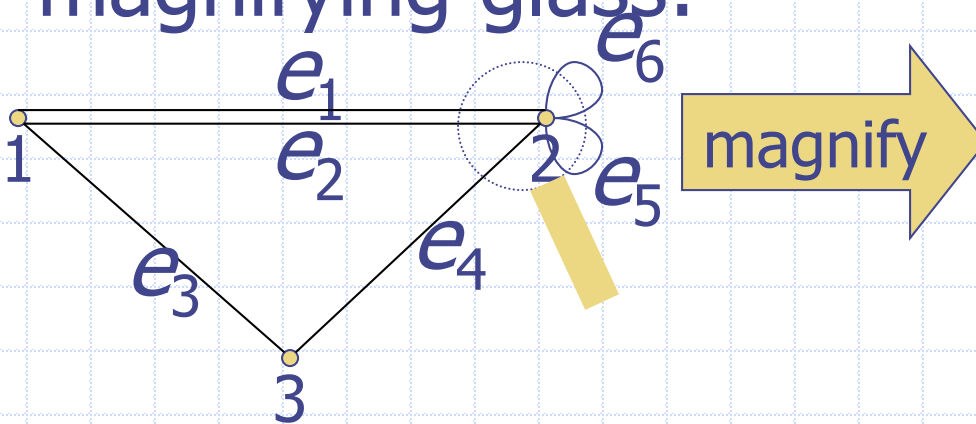
# Degree

The ***degree*** of a vertex counts the number of edges that *seem* to be sticking out if you looked under a magnifying glass:



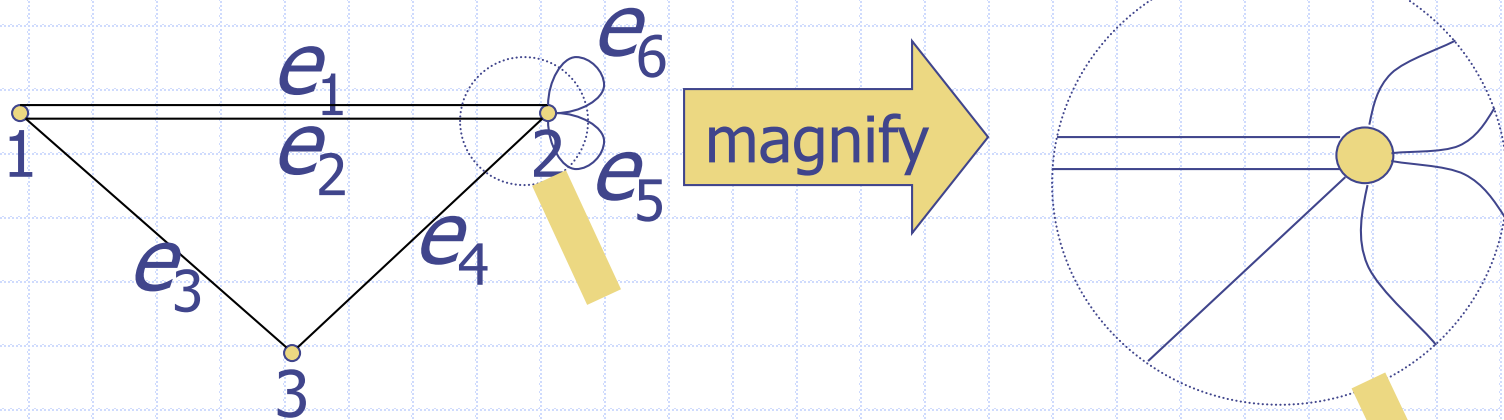
# Degree

The ***degree*** of a vertex counts the number of edges that *seem* to be sticking out if you looked under a magnifying glass:



# Degree

The ***degree*** of a vertex counts the number of edges that *seem* to be sticking out if you looked under a magnifying glass:

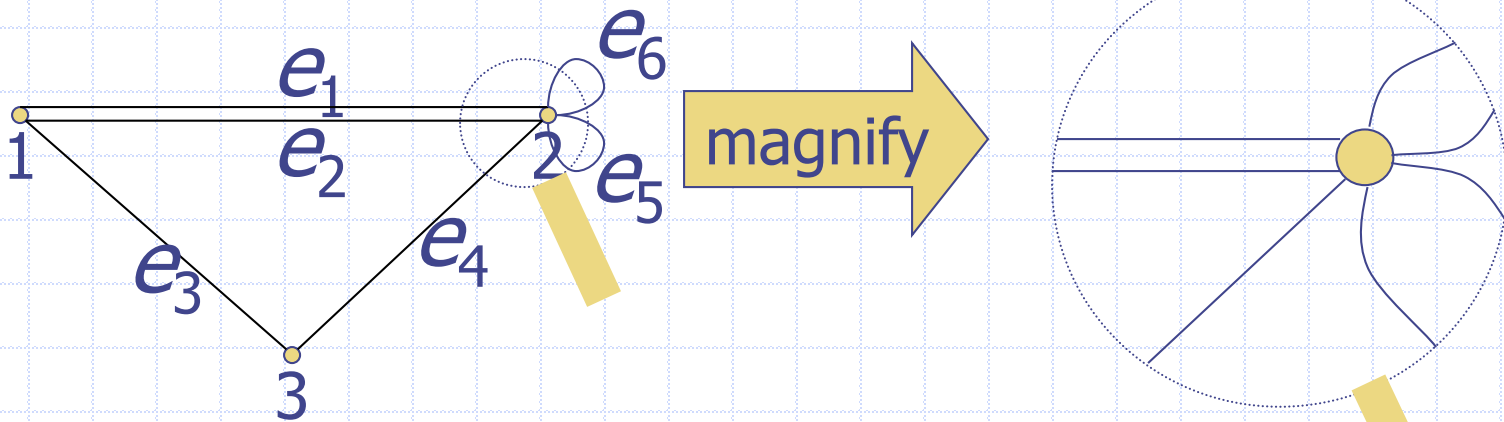


Thus  $\deg(2) = 7$  even though 2 only incident with 5 edges.

Q: How to define this formally?

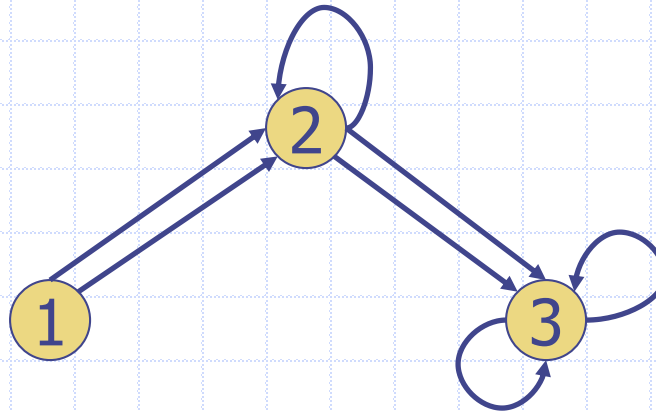
# Degree

A: Add 1 for every regular edge incident with vertex and 2 for every loop. Thus  $\deg(2) = 1 + 1 + 1 + 2 + 2 = 7$



# Oriented Degree when Edges Directed

The ***in-degree*** of a vertex ( $\deg^-$ ) counts the number of edges that stick *in* to the vertex. The ***out-degree*** ( $\deg^+$ ) counts the number sticking *out*.

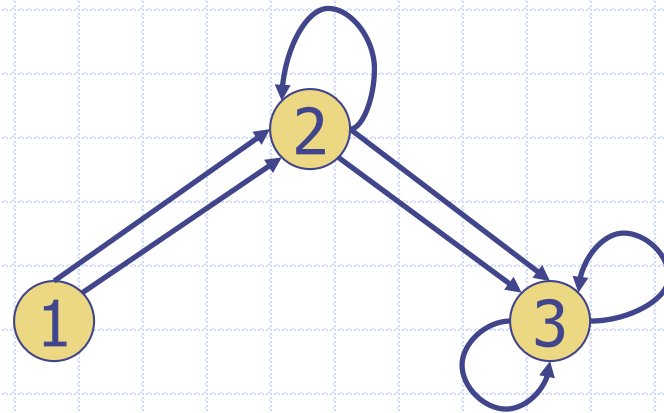


Q: What are in-degrees and out-degrees of all the vertices?

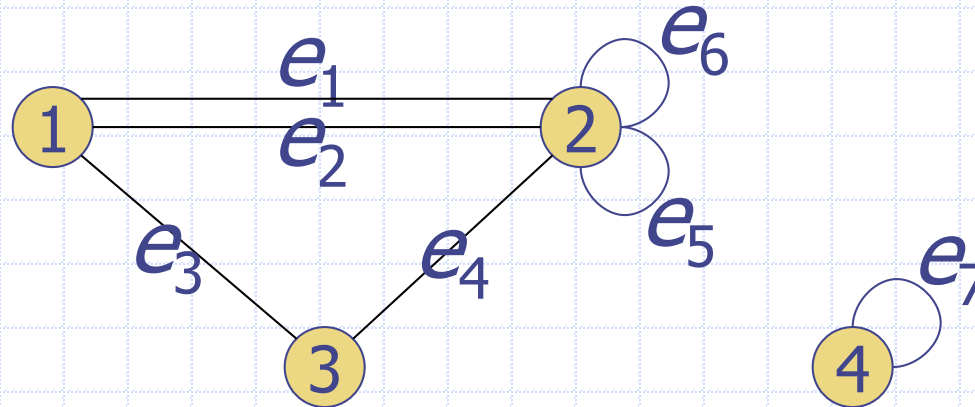
# Oriented Degree when Edges Directed

A:

- $\deg^-(1) = 0$
- $\deg^-(2) = 3$
- $\deg^-(3) = 4$
- $\deg^+(1) = 2$
- $\deg^+(2) = 3$
- $\deg^+(3) = 2$



# Handshaking Theorem



There are two ways to count the number of edges in the above graph:

1. Just count the set of edges: 7
2. Count *seeming* edges vertex by vertex and divide by 2 because double-counted edges:  
$$(\deg(1) + \deg(2) + \deg(3) + \deg(4)) / 2$$
$$= (3 + 7 + 2 + 2) / 2 = 14 / 2 = 7$$

# Handshaking Theorem

THM: In an undirected graph

$$|E| = \frac{1}{2} \sum_{e \in E} \deg(e)$$

In a directed graph

$$|E| = \sum_{e \in E} \deg^+(e) = \sum_{e \in E} \deg^-(e)$$

Q: In a party of 5 people can each person be friends with exactly three others?

# Handshaking Theorem

A: Imagine a simple graph with 5 people as *vertices* and edges being undirected edges between friends (simple graph assuming friendship is symmetric and irreflexive). Number of friends each person has is the degree of the person.

Handshaking would imply that

$$|E| = (\text{sum of degrees})/2 \quad \text{or}$$

$$2|E| = (\text{sum of degrees}) = (5 \cdot 3) = 15.$$

Impossible as 15 is not even. In general:

# Handshaking Theorem

Lemma: The number of vertices of odd degree must be even in an undirected graph.

*Proof:* Otherwise would have

$2|E| =$  Sum of even no.'s

+ an odd number of odd no.'s

→ even = even + odd

—this is impossible.  $\square$

# Graph Patterns

## Complete Graphs - $K_n$

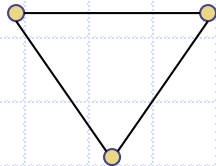
A simple graph is ***complete*** if every pair of distinct vertices share an edge. The notation  $K_n$  denotes the complete graph on  $n$  vertices.



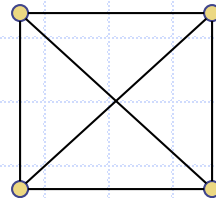
$K_1$



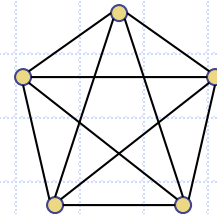
$K_2$



$K_3$



$K_4$

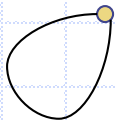


$K_5$

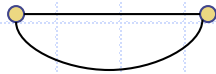
# Graph Patterns

## Cycles - $C_n$

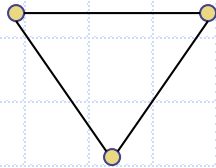
The **cycle graph**  $C_n$  is a circular graph with  $V = \{0, 1, 2, \dots, n-1\}$  where vertex  $i$  is connected to  $i+1 \bmod n$  and to  $i-1 \bmod n$ . They look like polygons:



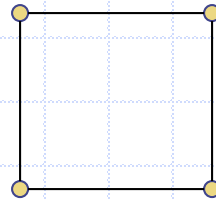
$C_1$



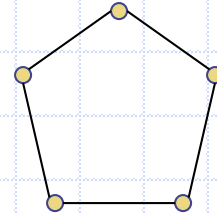
$C_2$



$C_3$



$C_4$



$C_5$

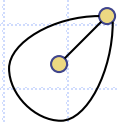
Q: What type of graph are  $C_1$  and  $C_2$  ?

# Graph Patterns

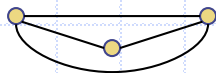
## Wheels - $W_n$

A: Pseudographs

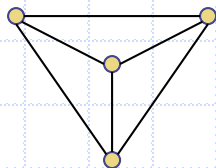
The **wheel graph**  $W_n$  is just a cycle graph with an extra vertex in the middle:



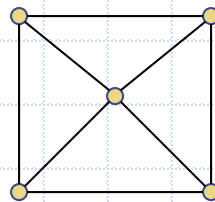
$W_1$



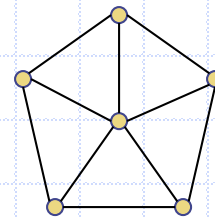
$W_2$



$W_3$



$W_4$



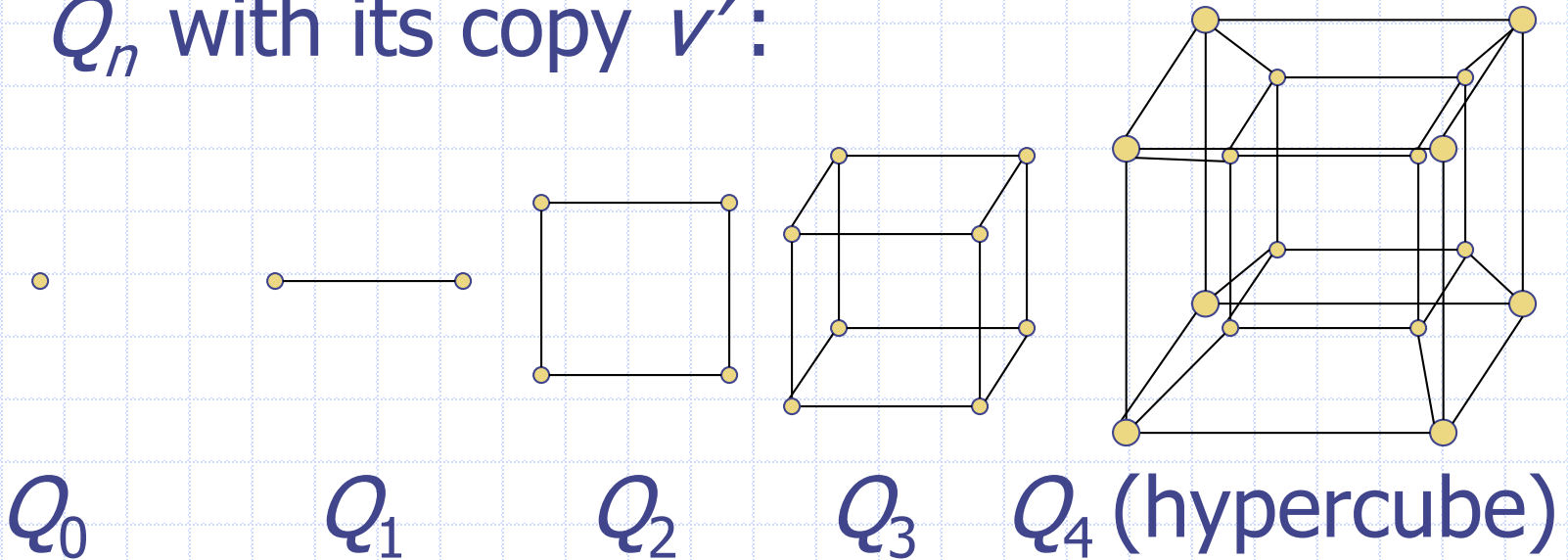
$W_5$

Usually consider wheels with 3 or more spokes only.

# Graph Patterns

## Cubes - $Q_n$

The ***n-cube***  $Q_n$  is defined recursively.  $Q_0$  is just a vertex.  $Q_{n+1}$  is gotten by taking 2 copies of  $Q_n$  and joining each vertex  $v$  of  $Q_n$  with its copy  $v'$ :

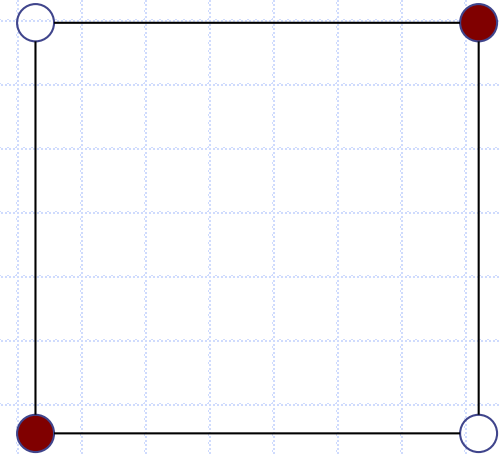


# Bipartite Graphs

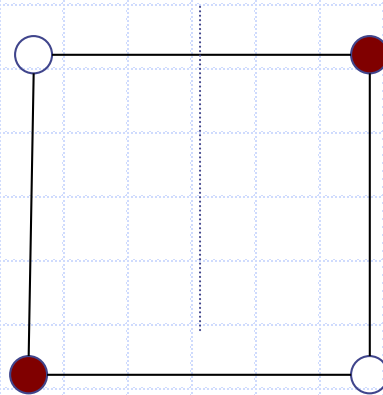
A simple graph is ***bipartite*** if  $V$  can be partitioned into  $V = V_1 \cup V_2$  so that any two adjacent vertices are in different parts of the partition. Another way of expressing the same idea is ***bichromatic***: vertices can be colored using two colors so that no two vertices of the same color are adjacent.

# Bipartite Graphs

EG:  $C_4$  is a bichromatic:

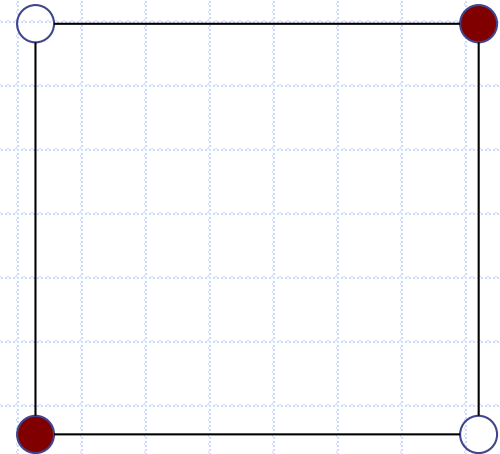


And so is bipartite, if we redraw it:

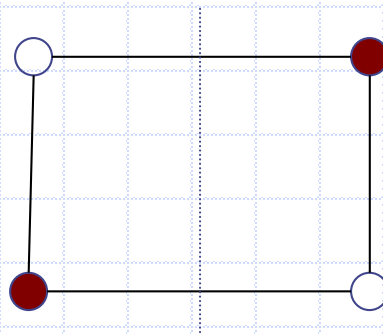


# Bipartite Graphs

EG:  $C_4$  is a bichromatic:

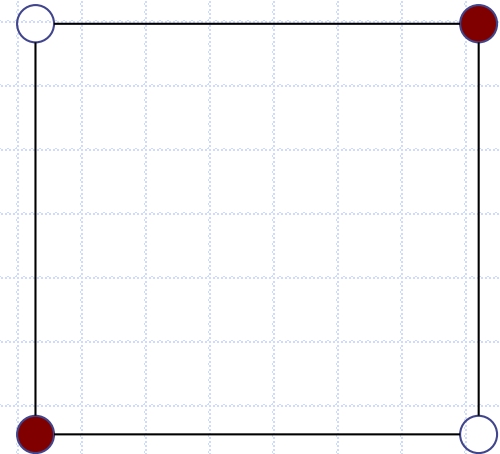


And so is bipartite, if we redraw it:

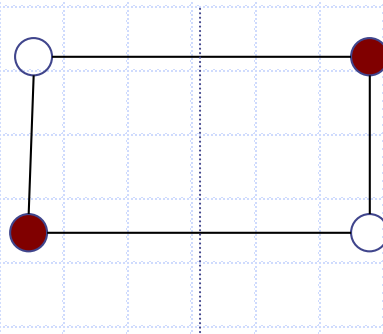


# Bipartite Graphs

EG:  $C_4$  is a bichromatic:

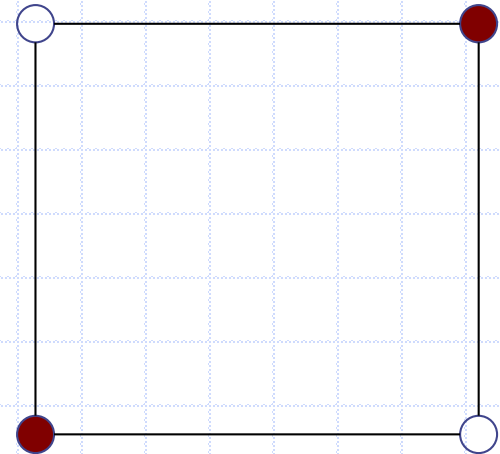


And so is bipartite, if we redraw it:

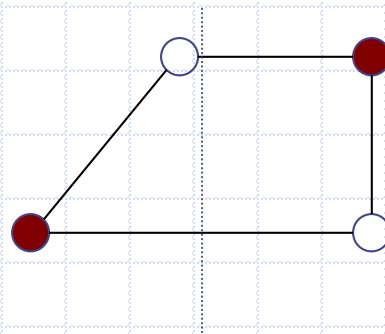


# Bipartite Graphs

EG:  $C_4$  is a bichromatic:

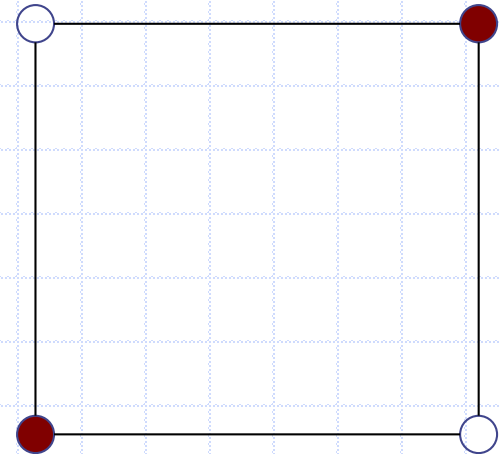


And so is bipartite, if we redraw it:

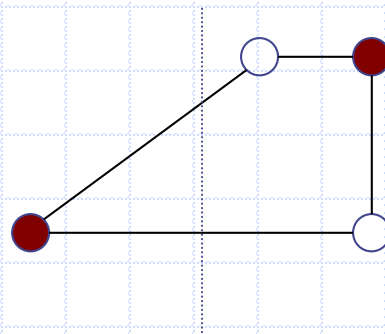


# Bipartite Graphs

EG:  $C_4$  is a bichromatic:

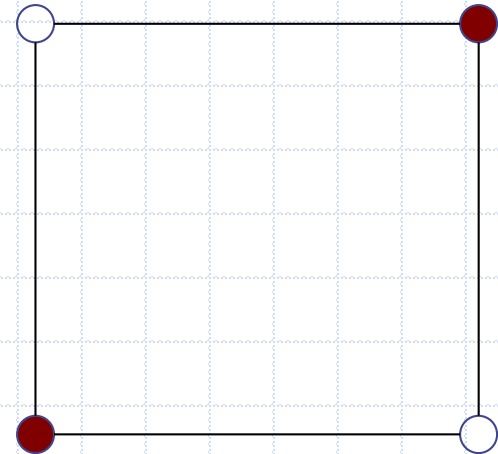


And so is bipartite, if we redraw it:

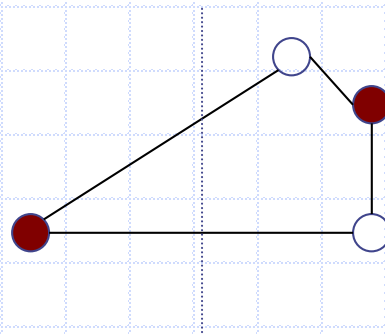


# Bipartite Graphs

EG:  $C_4$  is a bichromatic:

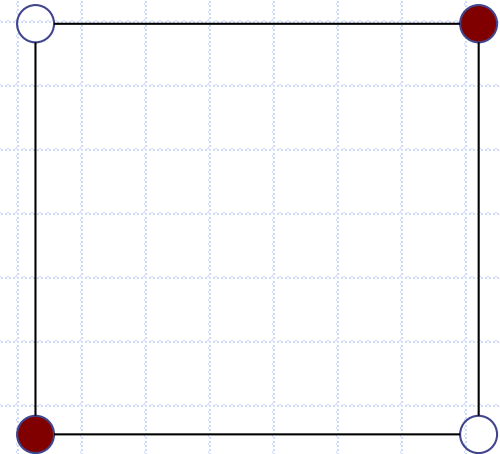


And so is bipartite, if we redraw it:

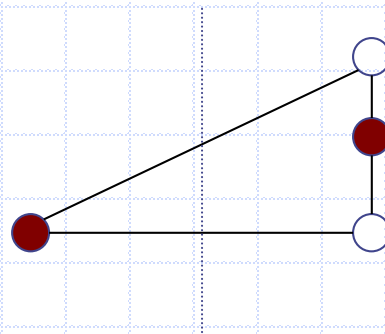


# Bipartite Graphs

EG:  $C_4$  is a bichromatic:

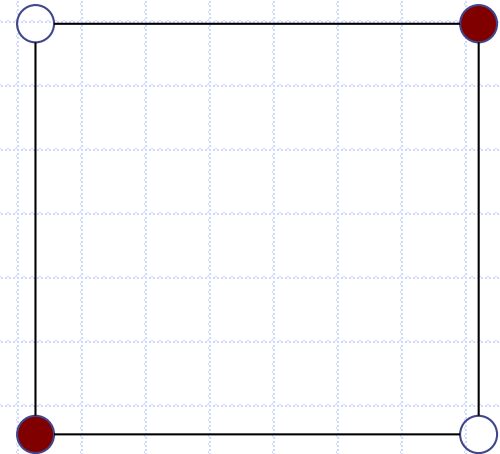


And so is bipartite, if we redraw it:

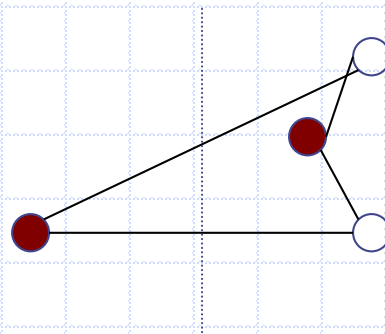


# Bipartite Graphs

EG:  $C_4$  is a bichromatic:

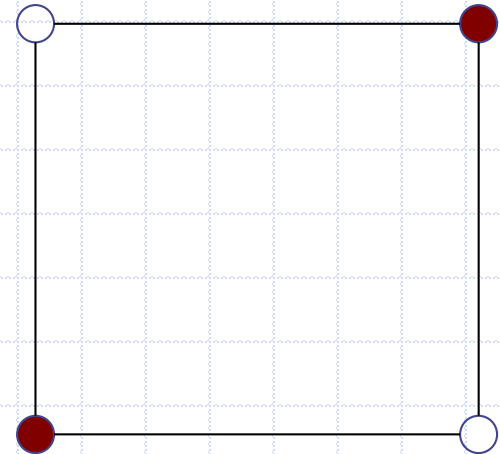


And so is bipartite, if we redraw it:

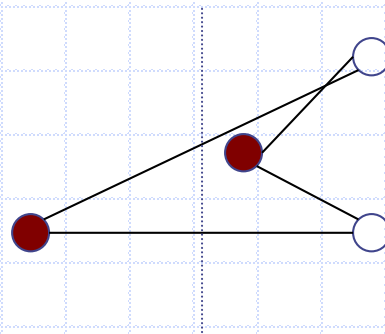


# Bipartite Graphs

EG:  $C_4$  is a bichromatic:

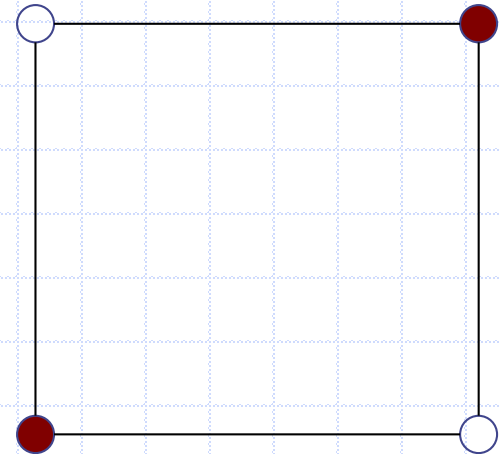


And so is bipartite, if we redraw it:

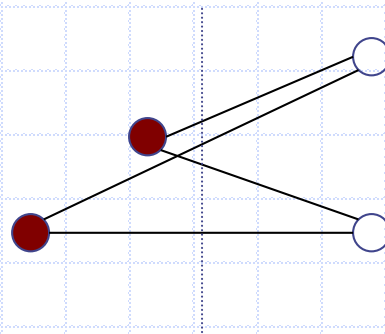


# Bipartite Graphs

EG:  $C_4$  is a bichromatic:

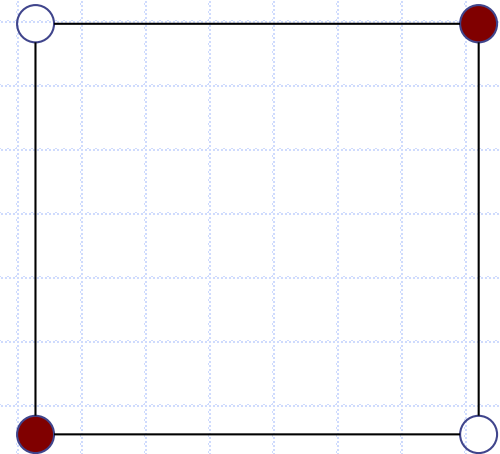


And so is bipartite, if we redraw it:

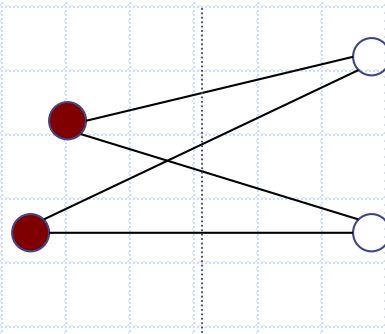


# Bipartite Graphs

EG:  $C_4$  is a bichromatic:

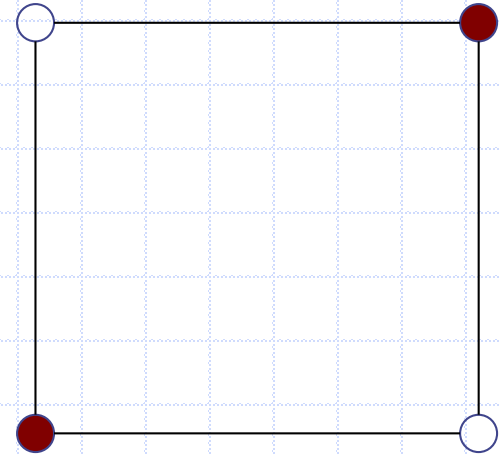


And so is bipartite, if we redraw it:

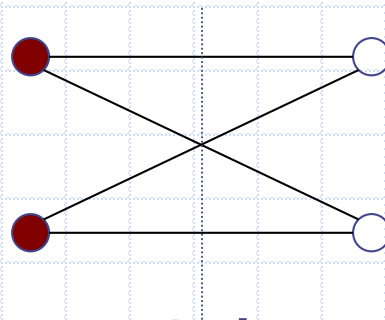


# Bipartite Graphs

EG:  $C_4$  is a bichromatic:



And so is bipartite, if we redraw it:



Q: For which  $n$  is  $C_n$  bipartite?

# Bipartite Graphs

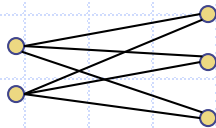
A:  $C_n$  is bipartite when  $n$  is even. For even  $n$  color all odd numbers red and all even numbers green so that vertices are only adjacent to opposite color.

If  $n$  is odd,  $C_n$  is not bipartite. If it were, color 0 red. So 1 must be green, and 2 must be red. This way, all even numbers must be red, including vertex  $n-1$ . But  $n-1$  connects to 0  $\rightarrow \leftarrow$ .

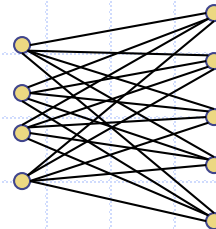
# Graph Patterns

## Complete Bipartite - $K_{m,n}$

When all possible edges exist in a simple bipartite graph with  $m$  red vertices and  $n$  green vertices, the graph is called **complete bipartite** and the notation  $K_{m,n}$  is used. EG:



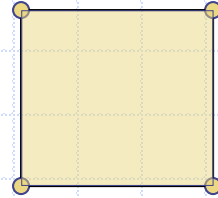
$K_{2,3}$



$K_{4,5}$

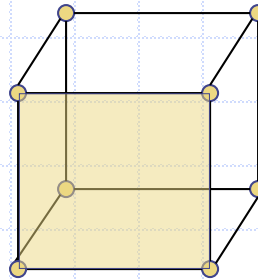
# Subgraphs

Notice that the 2-cube



occurs

inside the 3-cube



. In other

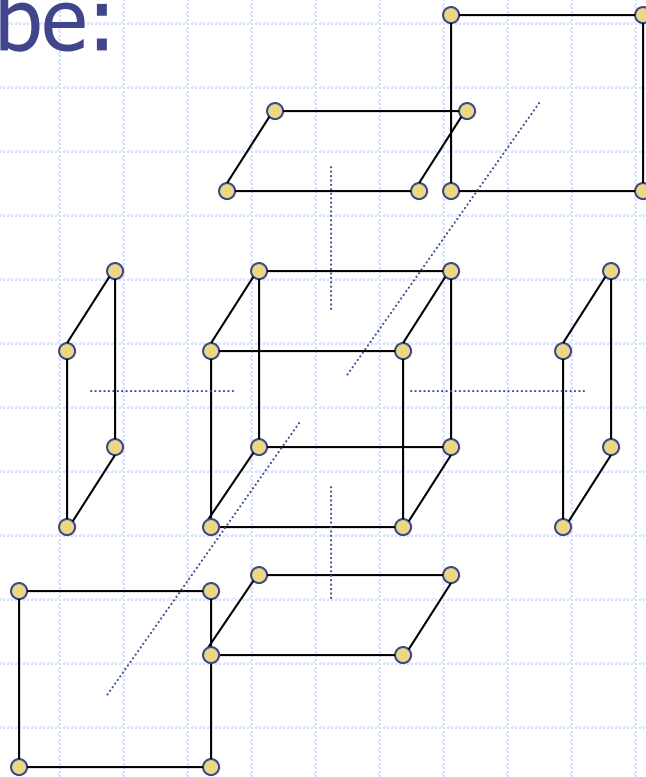
words,  $Q_2$  is a subgraph of  $Q_3$  :

DEF: Let  $G = (V, E)$  and  $H = (W, F)$  be graphs.  $H$  is said to be a **subgraph** of  $G$ , if  $W \subseteq V$  and  $F \subseteq E$ .

Q: How many  $Q_2$  subgraphs does  $Q_3$  have?

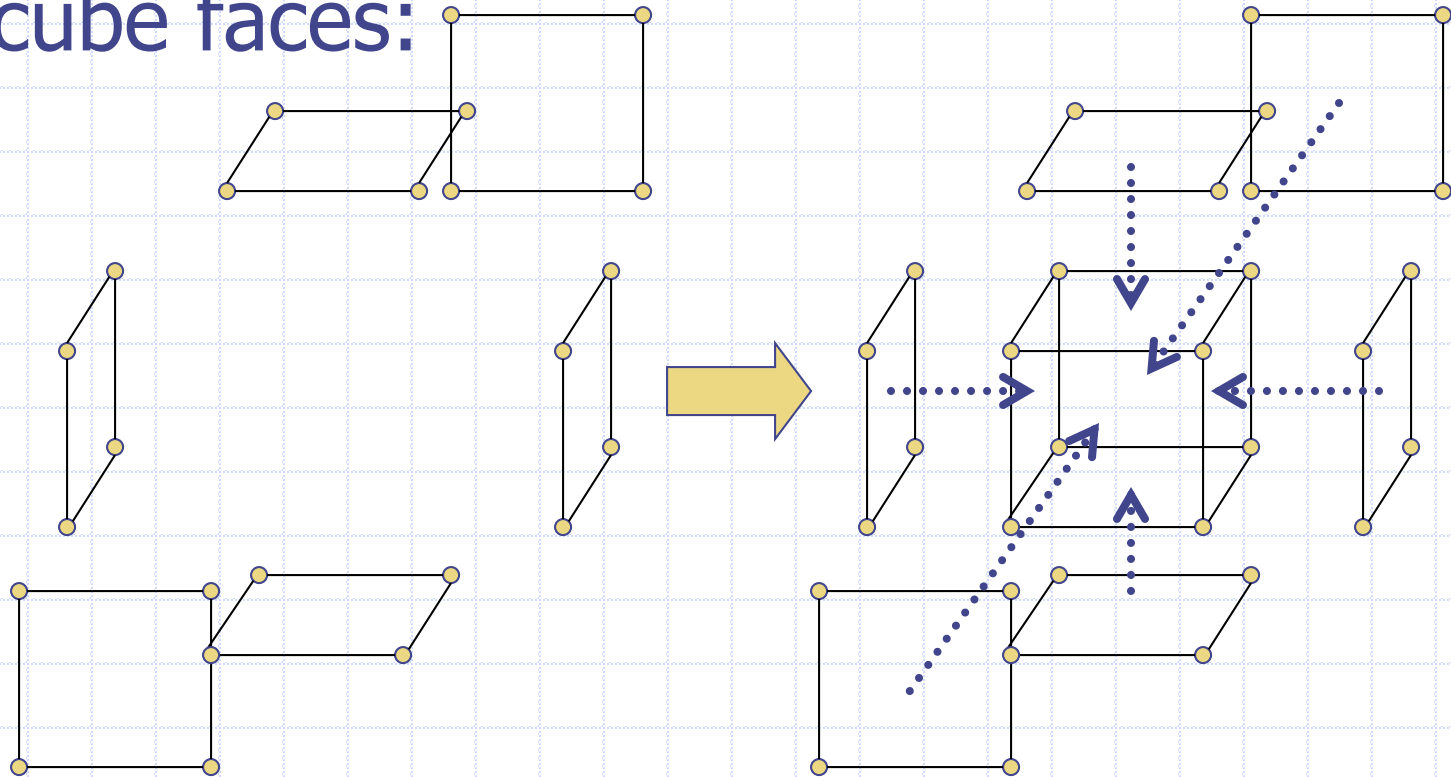
# Subgraphs

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



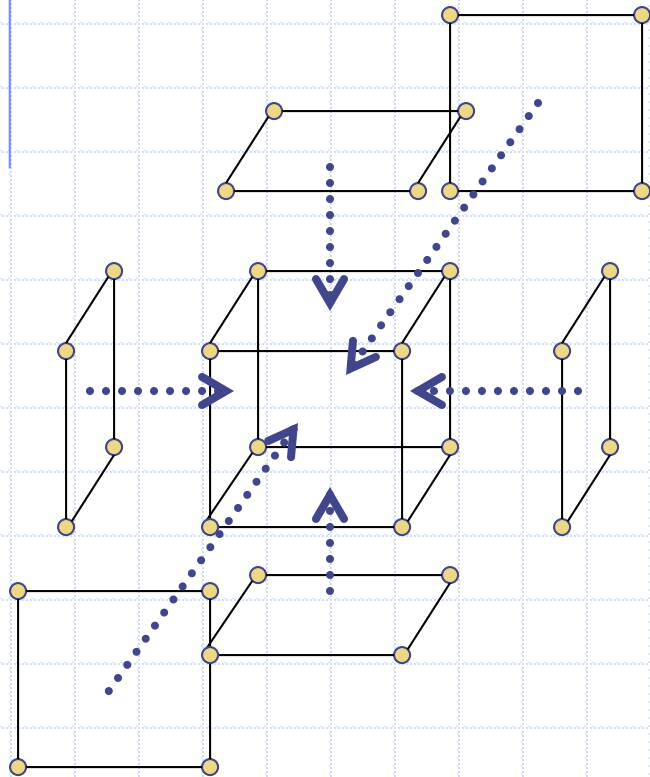
# Unions

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

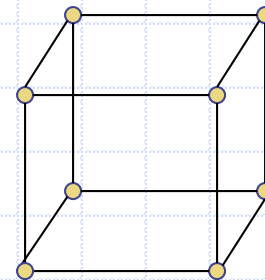
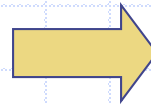


# Unions

If we assign the 2-cube faces (aka Squares) the names  $S_1, S_2, S_3, S_4, S_5, S_6$  then  $Q_3$  is the union of its faces:



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



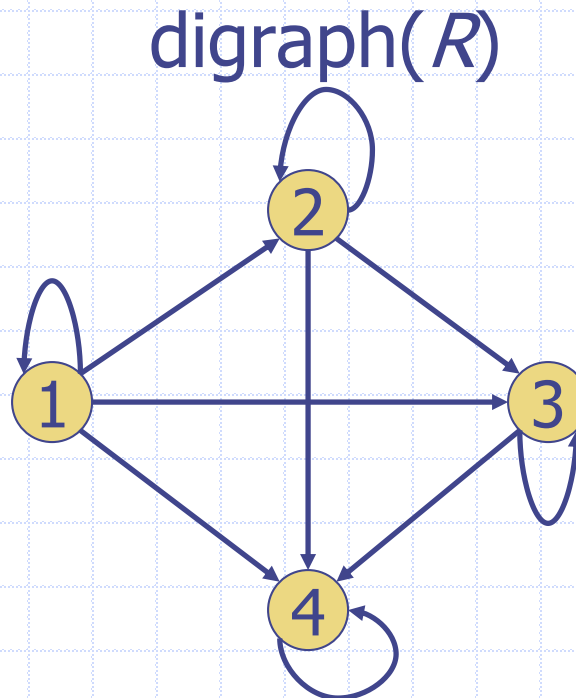
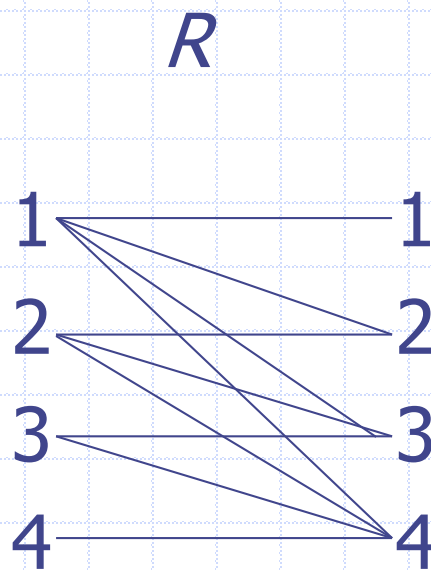
# Unions

DEF: Let  $G_1 = (V_1, E_1)$  and  $G_2 = (V_2, E_2)$  be two simple graphs (and  $V_1, V_2$  may or may not be disjoint). The ***union*** of  $G_1, G_2$  is formed by taking the union of the vertices and edges. 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.

# Adjacency Matrix

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



$M_R$

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

# Adjacency Matrix

Since digraphs *are* relations on their vertex sets, can adopt the concept to represent digraphs. In the context of graphs, we call the representation an ***adjacency matrix***:

For a digraph  $G = (V, E)$  define matrix  $A_G$  by:

- ◆ Rows, Columns –one for each vertex in  $V$
- ◆ Value at  $i^{\text{th}}$  row and  $j^{\text{th}}$  column is
  - 1 if  $i^{\text{th}}$  vertex connects to  $j^{\text{th}}$  vertex ( $i \rightarrow j$ )
  - 0 otherwise

# Adjacency Matrix -Directed Multigraphs

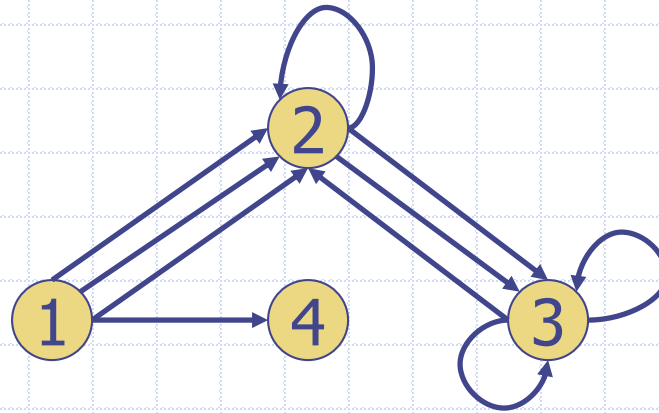
Can easily generalize to directed multigraphs by putting in the number of edges between vertices, instead of only allowing 0 and 1:

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

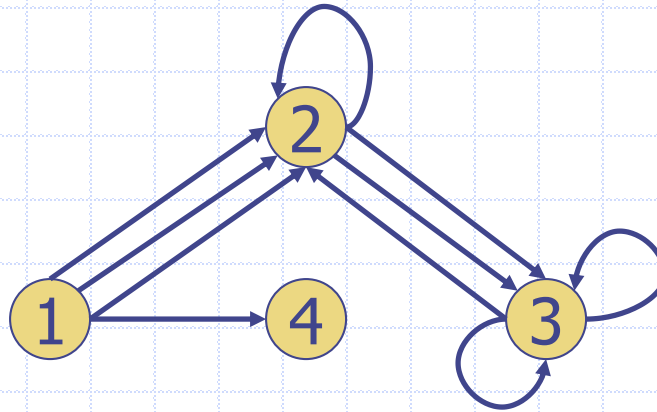
- ◆ Rows, Columns –one for each vertex in  $V$
- ◆ Value at  $i^{\text{th}}$  row and  $j^{\text{th}}$  column is
  - The number of edges with source the  $i^{\text{th}}$  vertex and target the  $j^{\text{th}}$  vertex

# Adjacency Matrix -Directed Multigraphs

Q: What is the adjacency matrix?



# Adjacency Matrix -Directed Multigraphs



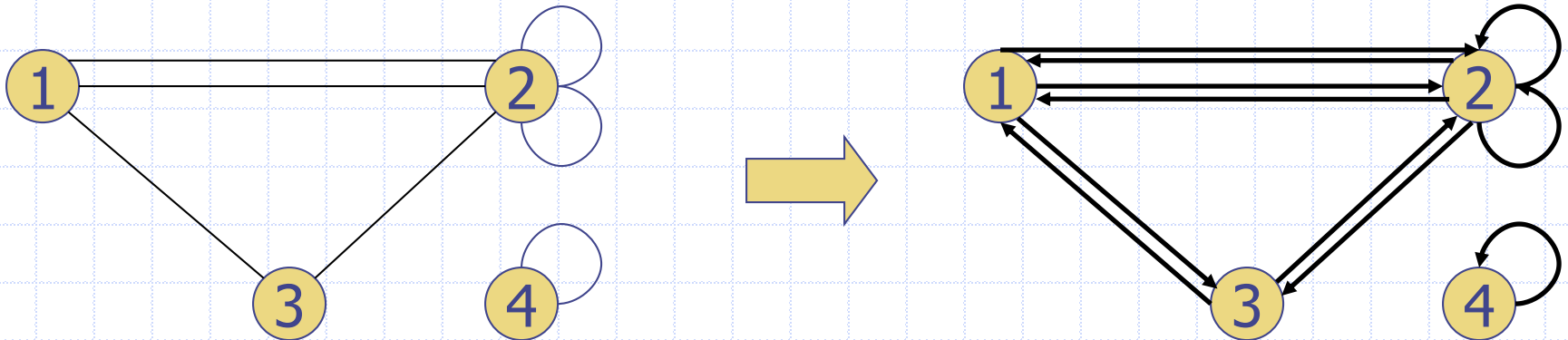
A:

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

# Adjacency Matrix

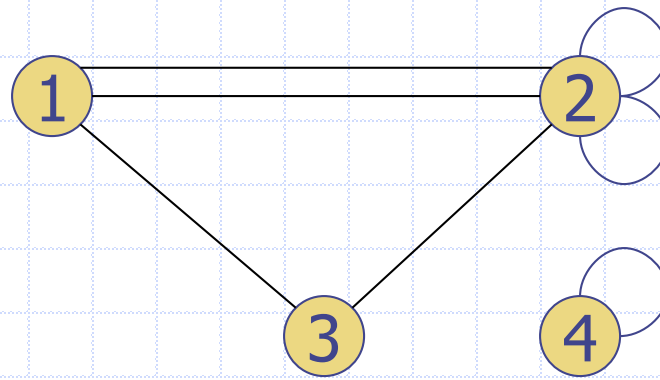
## -General

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:

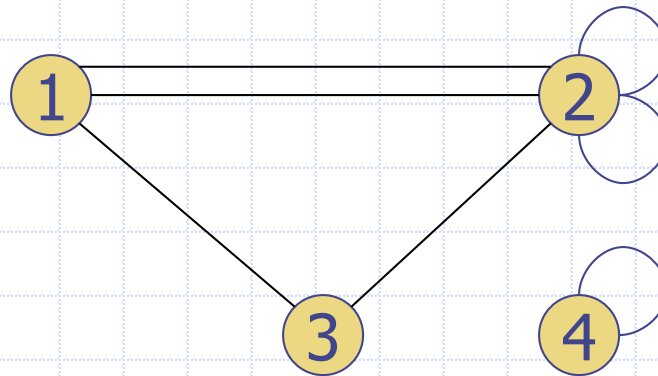


# Adjacency Matrix -General

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  $i^{\text{th}}$  row and  $j^{\text{th}}$  column is the number of edges incident with vertices  $i$  and  $j$ .

This is equivalent to converting first to a directed graph as above. Or by allowing undirected edges to take us from  $i$  to  $j$  can simply use definition for directed graphs.

# Graph Isomorphism

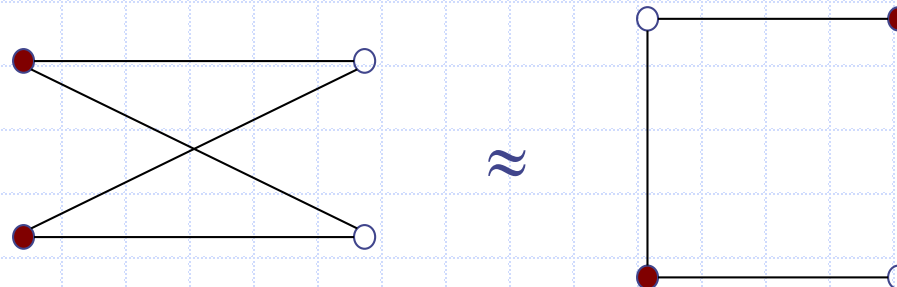
Various mathematical notions come with their own concept of *equivalence*, as opposed to equality:

◆ Equivalence for sets is bijectivity:

■ EG { 🍎 , 🍌 , 🍇 }  $\approx$  {12, 23, 43}

◆ Equivalence for graphs is isomorphism:

■ EG

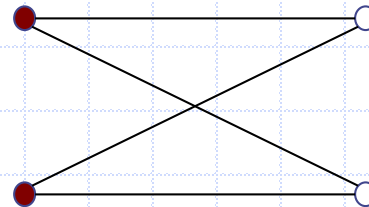
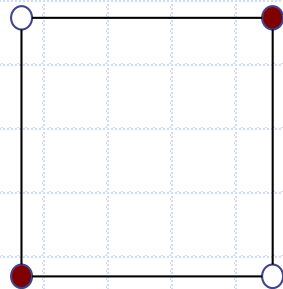


# Graph Isomorphism

Intuitively, two graphs are isomorphic if can bend, stretch and reposition vertices of the first graph, until the second graph is formed. Etymologically, *isomorphic* means "same shape".

EG: Can twist or relabel:

to obtain:



# Graph Isomorphism

## Undirected Graphs

DEF: Suppose  $G_1 = (V_1, E_1)$  and  $G_2 = (V_2, E_2)$  are pseudographs. Let  $f: V_1 \rightarrow V_2$  be a function s.t.:

- 1)  $f$  is bijective
- 2) for all vertices  $u, v$  in  $V_1$ , the number of edges between  $u$  and  $v$  in  $G_1$  is the same as the number of edges between  $f(u)$  and  $f(v)$  in  $G_2$ .

Then  $f$  is called an ***isomorphism*** and  $G_1$  is said to be ***isomorphic*** to  $G_2$ .

# Graph Isomorphism

## Digraphs

DEF: Suppose  $G_1 = (V_1, E_1)$  and  $G_2 = (V_2, E_2)$  are directed multigraphs. Let  $f: V_1 \rightarrow V_2$  be a function s.t.:

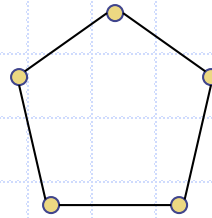
- 1)  $f$  is bijective
- 2) for all vertices  $u, v$  in  $V_1$ , the number of edges *from*  $u$  to  $v$  in  $G_1$  is the same as the number of edges between  $f(u)$  and  $f(v)$  in  $G_2$ .

Then  $f$  is called an ***isomorphism*** and  $G_1$  is said to be ***isomorphic*** to  $G_2$ .

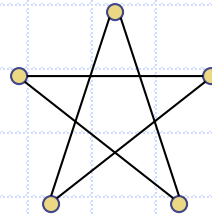
Note: Only difference between two definitions is the italicized "from" in no. 2 (was "between").

# Graph Isomorphism -Example

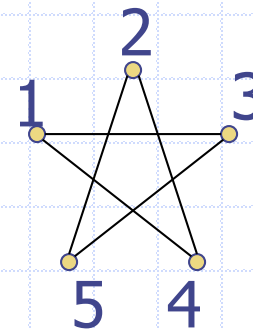
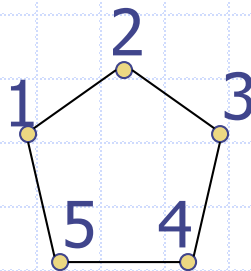
EG: Prove that



is isomorphic to



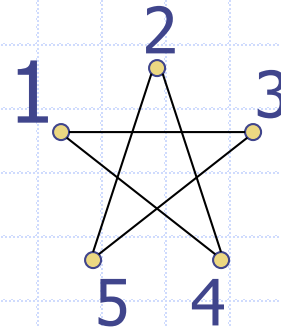
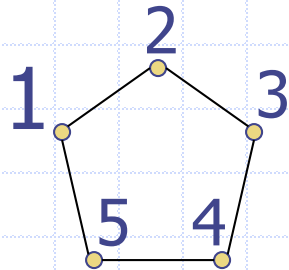
First label the vertices:



# Graph Isomorphism

## -Example

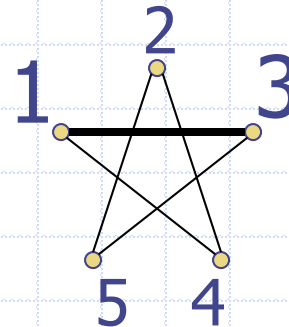
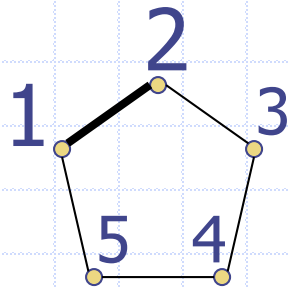
Next, set  $f(1) = 1$  and try to walk around clockwise on the star.



# Graph Isomorphism

## -Example

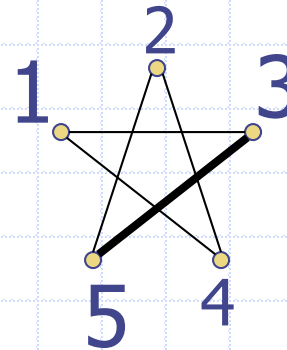
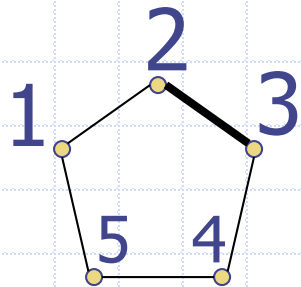
- Next, set  $f(1) = 1$  and try to walk around clockwise on the star. The next vertex seen is 3, *not* 2 so set  $f(2) = 3$ .



# Graph Isomorphism

## -Example

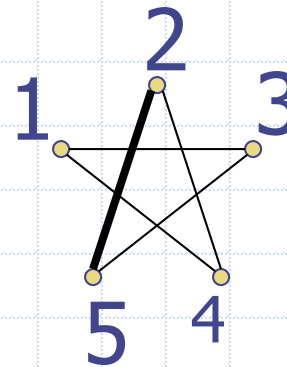
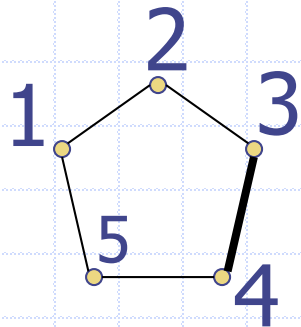
- Next, set  $f(1) = 1$  and try to walk around clockwise on the star. The next vertex seen is 3, *not* 2 so set  $f(2) = 3$ . Next vertex is 5 so set  $f(3) = 5$ .



# Graph Isomorphism

## -Example

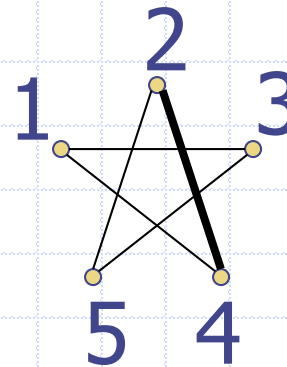
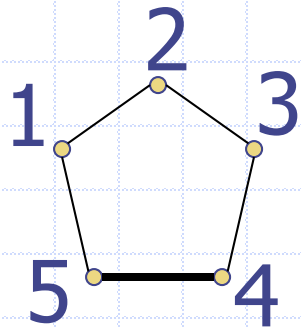
Next, set  $f(1) = 1$  and try to walk around clockwise on the star. The next vertex seen is 3, *not* 2 so set  $f(2) = 3$ . Next vertex is 5 so set  $f(3) = 5$ . In this fashion we get  $f(4) = 2$



# Graph Isomorphism

## -Example

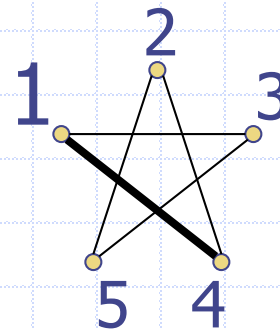
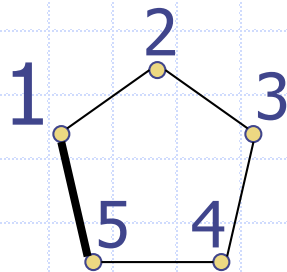
Next, set  $f(1) = 1$  and try to walk around clockwise on the star. The next vertex seen is 3, *not* 2 so set  $f(2) = 3$ . Next vertex is 5 so set  $f(3) = 5$ . In this fashion we get  $f(4) = 2$ ,  $f(5) = 4$ .



# Graph Isomorphism

## -Example

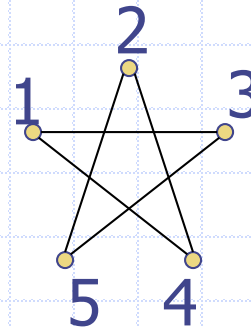
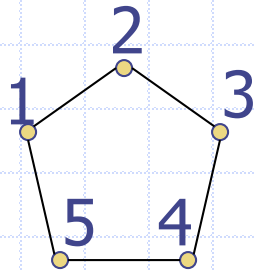
Next, set  $f(1) = 1$  and try to walk around clockwise on the star. The next vertex seen is 3, *not* 2 so set  $f(2) = 3$ . Next vertex is 5 so set  $f(3) = 5$ . In this fashion we get  $f(4) = 2$ ,  $f(5) = 4$ . If we would continue, we would get back to  $f(1) = 1$  so this process is well defined and  $f$  is a morphism.



# Graph Isomorphism

## -Example

Next, set  $f(1) = 1$  and try to walk around clockwise on the star. The next vertex seen is 3, *not* 2 so set  $f(2) = 3$ . Next vertex is 5 so set  $f(3) = 5$ . In this fashion we get  $f(4) = 2$ ,  $f(5) = 4$ . If we would continue, we would get back to  $f(1) = 1$  so this process is well defined and  $f$  is a morphism. Finally since  $f$  is bijective,  $f$  is an isomorphism.



# Properties of Isomorphisms

Since graphs are completely defined by their vertex sets and the number of edges between each pair, isomorphic graphs must have the same intrinsic properties. I.e. isomorphic graphs have the same...

- ...number of vertices and edges
- ...degrees at corresponding vertices
- ...types of possible subgraphs
- ...any other property defined in terms of the basic graph theoretic building blocks!

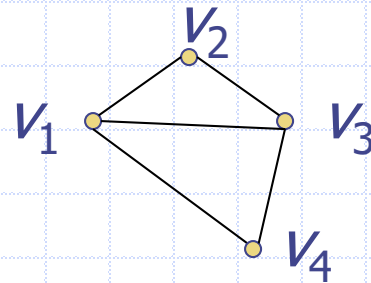
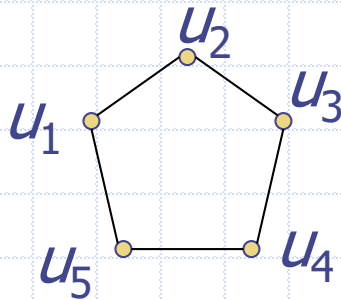
# Graph Isomorphism

## -Negative Examples

Once you see that graphs are isomorphic, easy to prove it. Proving the opposite, is usually more difficult. To show that two graphs are non-isomorphic need to show that no function can exist that satisfies defining properties of isomorphism. In practice, you try to find some intrinsic property that differs between the 2 graphs in question.

# Graph Isomorphism -Negative Examples

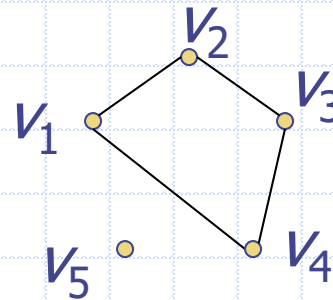
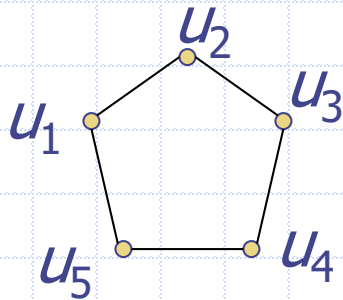
A: Why are the following non-isomorphic?



# Graph Isomorphism -Negative Examples

A: 1<sup>st</sup> graph has more vertices than 2<sup>nd</sup>.

Q: Why are the following non-isomorphic?

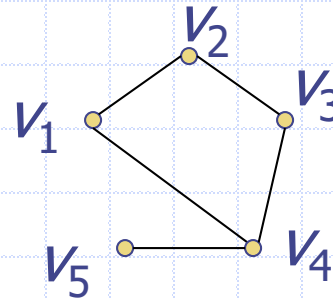
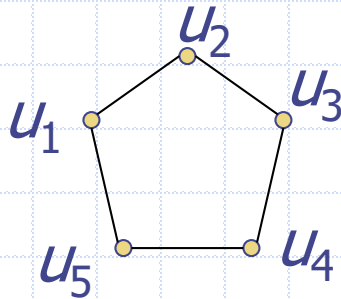


# Graph Isomorphism

## -Negative Examples

A: 1<sup>st</sup> graph has more edges than 2<sup>nd</sup>.

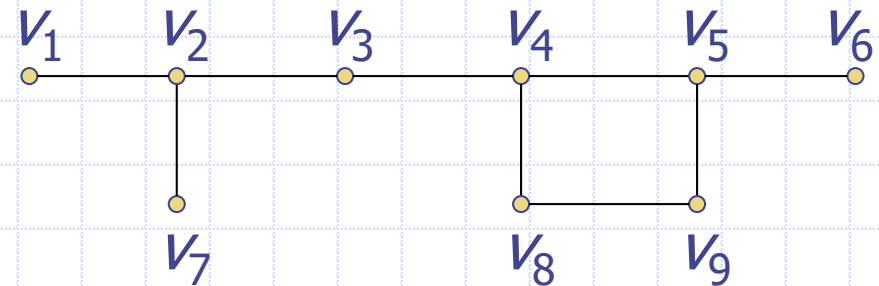
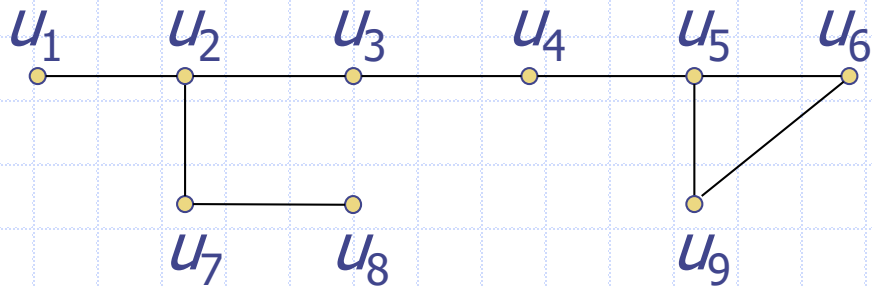
Q: Why are the following non-isomorphic?



# Graph Isomorphism -Negative Examples

A: 2<sup>nd</sup> graph has vertex of degree 1, 1<sup>st</sup> graph doesn't.

Q: Why are the following non-isomorphic?

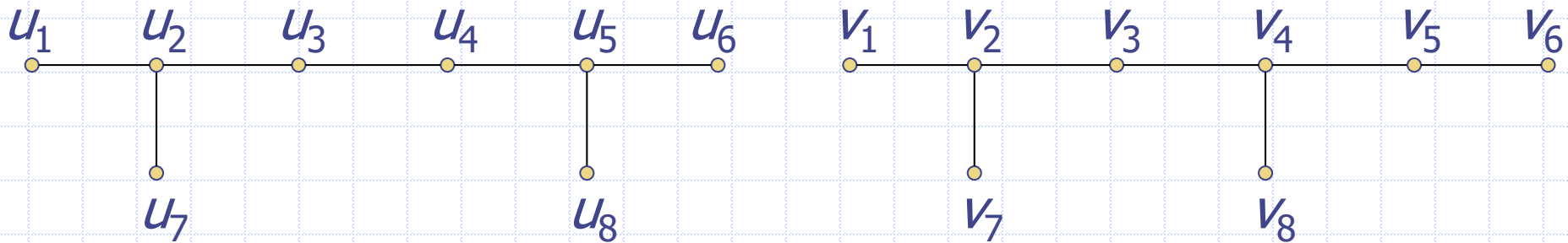


# Graph Isomorphism

## -Negative Examples

A: 1<sup>st</sup> graph has 2 degree 1 vertices, 4 degree 2 vertex and 2 degree 3 vertices.  
2<sup>nd</sup> graph has 3 degree 1 vertices, 3 degree 2 vertex and 3 degree 3 vertices.

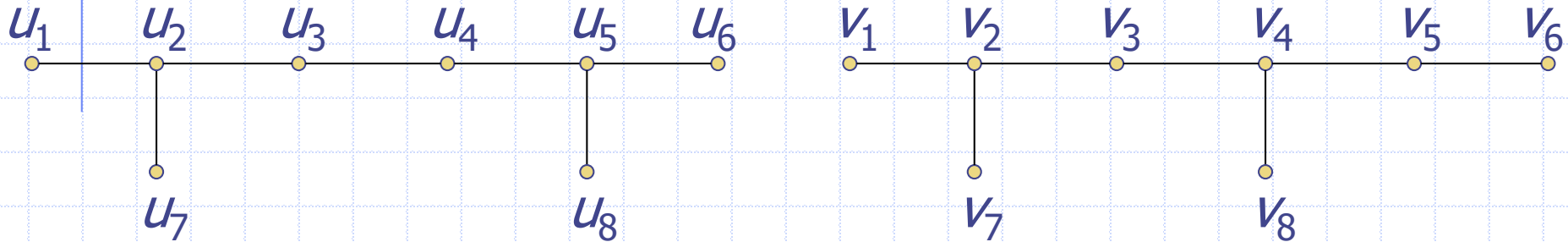
Q: Why are the following non-isomorphic?



# Graph Isomorphism

## -Negative Examples

A: None of the previous approaches work as there are the same no. of vertices, edges, and same no. of vertices per degree.



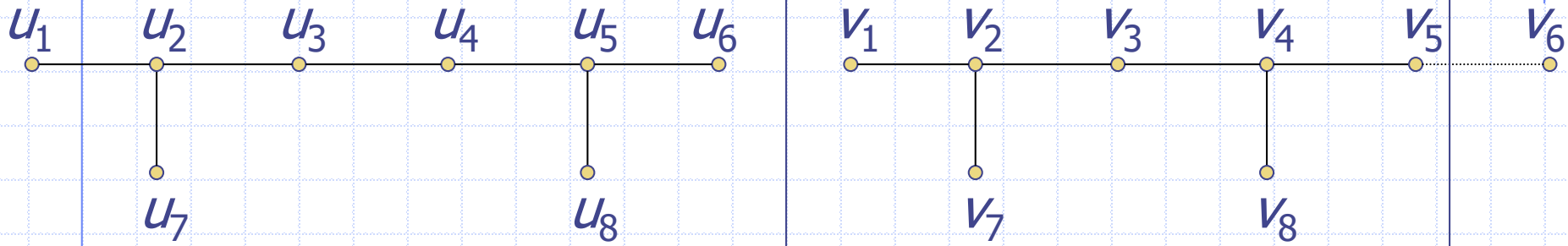
LEMMA: If  $G$  and  $H$  are isomorphic, then any subgraph of  $G$  will be isomorphic to some subgraph of  $H$ .

Q: Find a subgraph of 2<sup>nd</sup> graph which isn't a subgraph of 1<sup>st</sup> graph.

# Graph Isomorphism

## -Negative Examples

A: This subgraph is not a subgraph of the left graph.



Why not? Deg. 3 vertices must map to deg. 3 vertices. Since subgraph and left graph are symmetric, can assume  $v_2$  maps to  $u_2$ . Adjacent deg. 1 vertices to  $v_2$  must map to degree 1 vertices, forcing the deg. 2 adjacent vertex  $v_3$  to map to  $u_3$ . This forces the other vertex adjacent to  $v_3$ , namely  $v_4$  to map to  $u_4$ . But then a deg. 3 vertex has mapped to a deg. 2 vertex  $\rightarrow \leftarrow \square$  <sup>92</sup>

# Blackboard Exercise

Show that  $W_6$  is not bipartite.