Representing Graphs and Graph Isomorphism

Representing Graphs: adjacency lists

One way to represent a graph without multiple edges is to list all the edges of this graph ($G = (V, E)$).

Another way to represent a graph with no multiple edges is to use adjacency lists, which specify the vertices that are adjacent to each vertex of the graph.

EXAMPLE Use adjacency lists to describe the simple graph shown bellow.

Solution: Table 1 lists those vertices adjacent to each of the vertices of the graph.

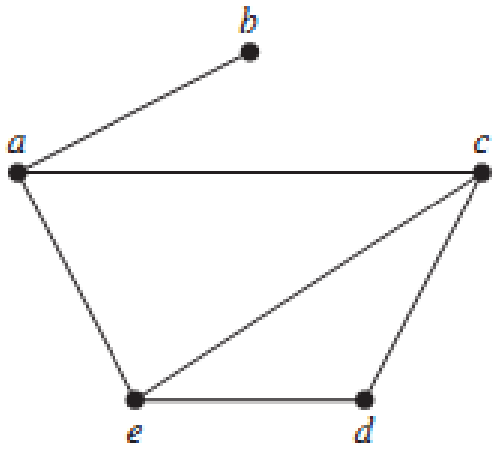


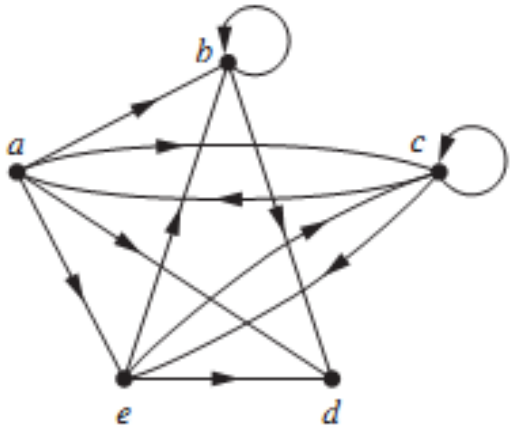
TABLE 1 An Adjacency List for a Simple Graph.

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

Adjacency lists for directed graphs

EXAMPLE Represent the directed graph *G* by listing all the vertices that are the terminal vertices of edges starting at each vertex of the graph.

Solution: Table 2 represents the directed graph *G*.



G

TABLE 2 An Adjacency List for a Directed Graph.

Initial Vertex	*Terminal Vertices*
a	b, c, d, e
b	b, d
c	a, c, e
d	
e	b, c, d

Matrices for graph representation

Carrying out graph algorithms using the representation of graphs by adjacency lists,

can be cumbersome if there are many edges in the graph.

To simplify computation, graphs can be represented using matrices.

2 types of matrices are commonly used to represent graphs.

One is based on the adjacency of vertices, and the other is based on incidence of vertices and edges.

Adjacency Matrices

Let $G = (V, E)$ a simple graph where $|V| = n$.

Suppose that the vertices of G are listed arbitrarily as $v_1, v_2, \ldots, v_n$.

The adjacency matrix A (or A_G) of G, with respect to this listing of the vertices,

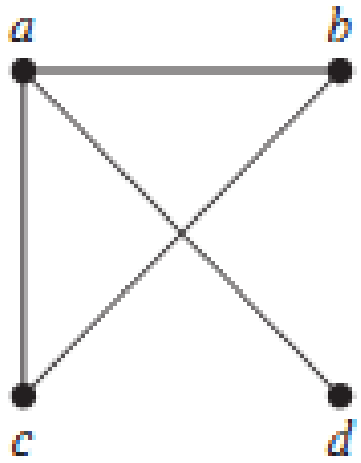
is the n x n zero–one matrix (бинарная матрица) $A=[a_{ij}]$.

$$a_{ij} = \begin{cases} 1 & \text{if } \{v_i, v_j\} \text{ is an edge of } G, \\ 0 & \text{otherwise.} \end{cases}$$

EXAMPLE Use an adjacency matrix to represent the graph G.

Solution: We order the vertices as a, b, c, d.

The matrix representing this graph is



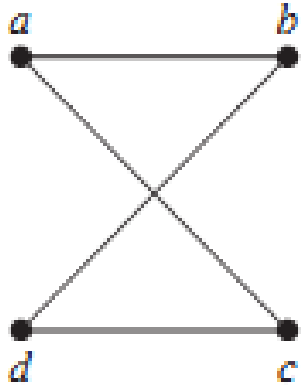
G

$$\begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}.$$

EXAMPLE Draw a graph with the adjacency matrix with respect to the ordering of vertices *a, b, c, d*.

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

Solution: A graph with this adjacency matrix.



Note that an adjacency matrix of a graph is based on the ordering chosen for the vertices.

Hence, there may be as many as $n!$ different adjacency matrices for a graph with n vertices, because there are $n!$ different orderings of n vertices.

The adjacency matrix of a simple graph is symmetric, that is, $a_{ij} = a_{ji}$, because both of

these entries are 1 when v_i and v_j are adjacent, and both are 0 otherwise.

Furthermore, because a simple graph has no loops, each entry a_{ii} , $i = 1, 2, 3,...,n$, is 0.

Adjacency matrices can also be used to represent undirected graphs with loops and with multiple edges.

A loop at the vertex v_i is represented by a 1 at the (i, i)th position of the adjacency matrix.

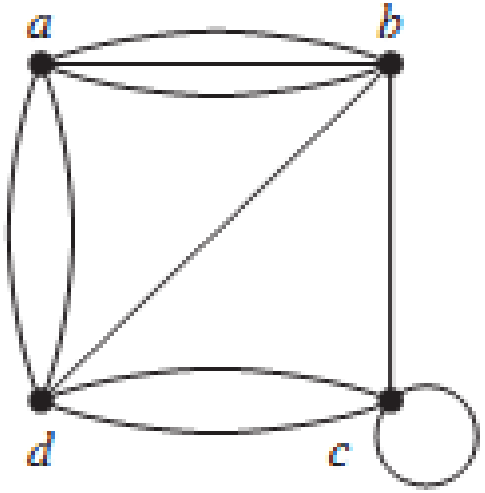
When multiple edges connecting the same pair of vertices v_i and v_j, or multiple loops at the same vertex, are present, the adjacency matrix is no longer a zero–one matrix, because the (i, j)th entry of this matrix equals the number of edges that are associated to $\{v_i, v_j\}$.

All undirected graphs, including multigraphs and pseudographs, have symmetric adjacency matrices.

EXAMPLE

Use an adjacency matrix to represent the pseudograph *G*.

Solution: The adjacency matrix using the ordering of vertices *a, b, c, d* is



G

$$\begin{bmatrix} 0 & 3 & 0 & 2 \\ 3 & 0 & 1 & 1 \\ 0 & 1 & 1 & 2 \\ 2 & 1 & 2 & 0 \end{bmatrix}.$$

Adjacency matrix for a directed graph

The matrix for a directed graph $G = (V, E)$ has a 1 in its (i, j)th position if there is an edge from v_i to v_j,

where $v_1, v_2, \ldots, v_n$ is an arbitrary listing of the vertices of the directed graph.

In other words, if $A = [a_{ij}]$ is the adjacency matrix for the directed graph with respect to this listing of the vertices, then

$$a_{ij} = \begin{cases} 1 & \text{if } (v_i, v_j) \text{ is an edge of } G, \\ 0 & \text{otherwise.} \end{cases}$$

The adjacency matrix for a directed graph does not have to be symmetric, because there may not be an edge from v_j to v_i when there is an edge from v_i to v_j.

Adjacency matrices can also be used to represent directed multigraphs.

Again, such matrices are not zero–one matrices when there are multiple edges in the same direction connecting 2 vertices.

In the adjacency matrix for a directed multigraph, a_{ij} equals the number of edges that are associated to (v_i, v_j).

TRADE-OFFS BETWEEN ADJACENCY LISTS AND ADJACENCY MATRICES

When a simple graph contains relatively few edges, that is, when it is sparse (разреженный), it is usually preferable to use adjacency lists rather than an adjacency matrix to represent the graph.

For example, if each vertex has degree not exceeding c, where $c << n$, then each adjacency list contains $\leq c$ vertices.

Hence, there are $\leq cn$ items in all these adjacency lists.

On the other hand, the adjacency matrix for the graph has n^2 entries.

Note, however, that the adjacency matrix of a sparse graph is a sparse matrix, that is, a matrix with few nonzero entries, and there are special techniques for representing, and computing with, sparse matrices.

Now suppose that a simple graph is dense (плотный), that is, suppose that it contains many edges, such as a graph that contains $> n^2/2$ edges.

In this case, using an adjacency matrix to represent the graph is usually preferable over using adjacency lists.

To see why, we compare the complexity of determining whether the possible edge $\{v_i, v_j\}$ is present.

Using an adjacency matrix, we can determine whether this edge is present by examining the (i, j)th entry in the matrix.

This entry is 1 if the graph contains this edge and is 0 otherwise.

Consequently, we need make only 1 comparison, namely, comparing this entry with 0, to determine whether this edge is present.

On the other hand, when we use adjacency lists to represent the graph, we need to search the list of vertices adjacent to either v_i or v_j to determine whether this edge is present.

This can require $\Theta(|V|)$ comparisons when many edges are present.

Incidence Matrices

Another common way to represent graphs is to use incidence matrices.

Let $G = (V, E)$ be an undirected graph.

Suppose that $v_1, v_2, \ldots, v_n$ are the vertices and $e_1, e_2, \ldots, e_m$ are the edges of G.

Then the incidence matrix with respect to this ordering of V and E is the $n \times m$ matrix

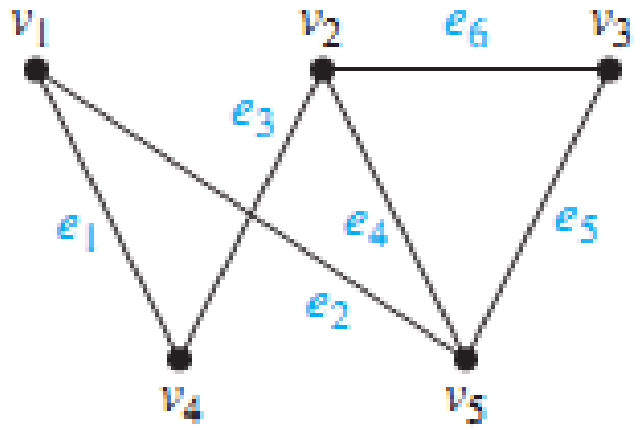
$M = [m_{ij}]$, where

$$m_{ij} = \begin{cases} 1 & \text{when edge } e_j \text{ is incident with } v_i, \\ 0 & \text{otherwise.} \end{cases}$$

EXAMPLE

Represent the graph *G* with an incidence matrix.

Solution: The incidence matrix is

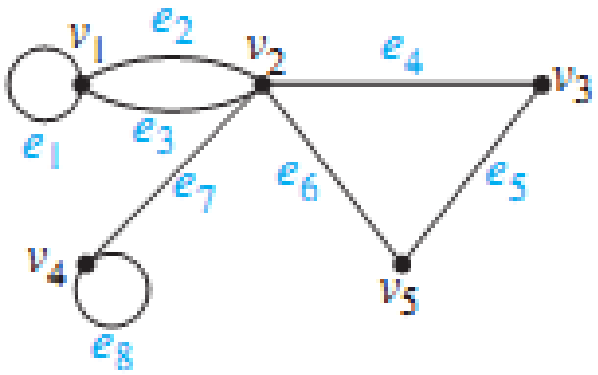


G

$$
\begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \end{array}
\begin{array}{c}
\begin{array}{cccccc} e_1 & e_2 & e_3 & e_4 & e_5 & e_6 \end{array} \\
\left[\begin{array}{cccccc}
1 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 & 0 & 1 \\
0 & 0 & 0 & 0 & 1 & 1 \\
1 & 0 & 1 & 0 & 0 & 0 \\
0 & 1 & 0 & 1 & 1 & 0
\end{array}\right]
\end{array}.
$$

EXAMPLE

Represent the pseudograph *G* using an incidence matrix.



G

$$
\begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \end{array}
\begin{array}{c}
\begin{array}{cccccccc} e_1 & e_2 & e_3 & e_4 & e_5 & e_6 & e_7 & e_8 \end{array} \\
\left[\begin{array}{cccccccc}
1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 \\
0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\
0 & 0 & 0 & 0 & 1 & 1 & 0 & 0
\end{array}\right]
\end{array}.
$$

Question: How to represent a directed graph with an incidence matrix?

Isomorphism of Graphs

We often need to know whether it is possible to draw 2 graphs in the same way.

That is, do the graphs have the same structure when we ignore the identities of their vertices?

For instance, in chemistry, graphs are used to model chemical compounds.

Different compounds can have the same molecular formula but can differ in structure.

Such compounds can be represented by graphs that cannot be drawn in the same way.

The graphs representing previously known compounds can be used to determine whether a supposedly new compound has been studied before.

DEFINITION 1 The simple graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ are isomorphic if there exists a bijection f between V_1 and V_2 with the property that a and b are adjacent in $G_1 \Leftrightarrow f(a)$ and $f(b)$ are adjacent in G_2, for all a and b in V_1.

Such a bijection f is called an isomorphism.

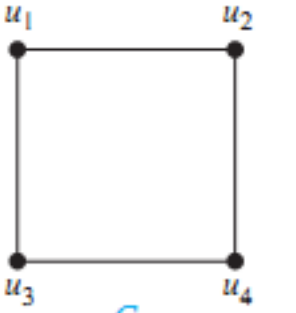
2 simple graphs that are not isomorphic are called nonisomorphic.

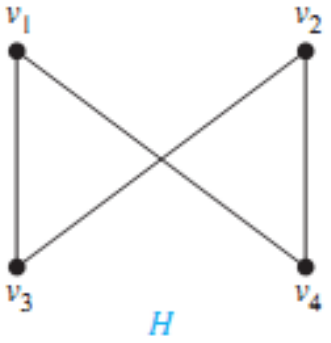
In other words, when 2 simple graphs are isomorphic, there is a bijection between vertices of the 2 graphs that preserves the adjacency relationship.

Isomorphism of simple graphs is an equivalence relation.

EXAMPLE

Show that the graphs $G = (V, E)$ and $H = (W, F)$, are isomorphic.





Solution: The function f with $f(u_1) = v_1$, $f(u_2) = v_4$, $f(u_3) = v_3$, and $f(u_4) = v_2$ is a bijection between V and W.

To see that this correspondence preserves adjacency, note that adjacent vertices in G are u_1 and u_2, u_1 and u_3, u_2 and u_4, and u_3 and u_4, and each of the pairs $f(u_1) = v_1$ and $f(u_2) = v_4$, $f(u_1) = v_1$ and $f(u_3) = v_3$, $f(u_2) = v_4$ and $f(u_4) = v_2$, and $f(u_3) = v_3$ and $f(u_4) = v_2$ consists of 2 adjacent vertices in H.

Determining whether 2 Simple Graphs are Isomorphic

It is often difficult to determine whether 2 simple graphs are isomorphic.

There are n! possible bijections between the vertex sets of 2 simple graphs with *n* vertices.

Testing each such correspondence to see whether it preserves adjacency and nonadjacency is impractical if *n* is at all large.

Sometimes it is not hard to show that 2 graphs are not isomorphic.

In particular, we can show that 2 graphs are not isomorphic if we can find a property only 1 of the 2 graphs has, but that is preserved by isomorphism.

A property preserved by isomorphism of graphs is called a graph invariant.

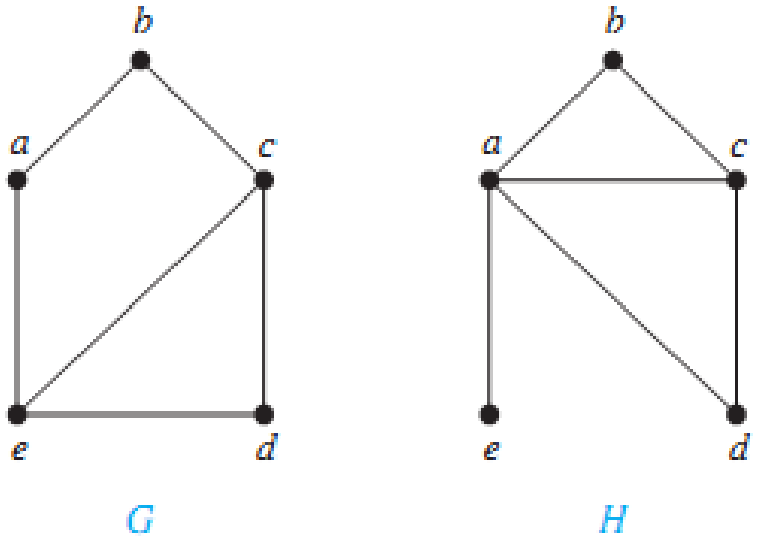
For instance, isomorphic simple graphs must have the same number of vertices, because there is a bijection between the sets of vertices of the graphs.

Isomorphic simple graphs also must have the same number of edges, because the bijection between vertices establishes a bijection between edges.

In addition, the degrees of the vertices in isomorphic simple graphs must be the same.

That is, a vertex v of degree d in G must correspond to a vertex $f(v)$ of degree d in H, because a vertex w in G is adjacent to $v \Leftrightarrow f(v)$ and $f(w)$ are adjacent in H.

EXAMPLE



Show that the graphs *G* and *H* are not isomorphic.

Solution: Both *G* and *H* have 5 vertices and 6 edges.

However, *H* has a vertex of degree 1, namely, *e*, whereas *G* has no vertices of degree 1.

It follows that *G* and *H* are not isomorphic.

The number of vertices,

the number of edges,

the number of vertices of each degree

are all invariants under isomorphism.

If any of these quantities differ in 2 simple graphs, these graphs cannot be isomorphic.

However, when these invariants are the same, it does not necessarily mean that the 2 graphs are isomorphic.

There are no useful sets of invariants currently known that can be used to determine whether simple graphs are isomorphic.

EXAMPLE

Determine whether the graphs *G* and *H* are isomorphic.

Solution: The graphs *G* and *H* both have 8 vertices and 10 edges.

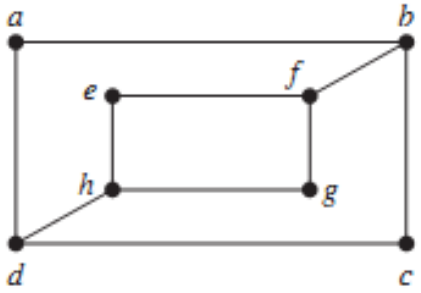
They also both have 4 vertices of degree 2 and 4 vertices of degree 3.

Because these invariants all agree, it is still conceivable that these graphs are isomorphic.

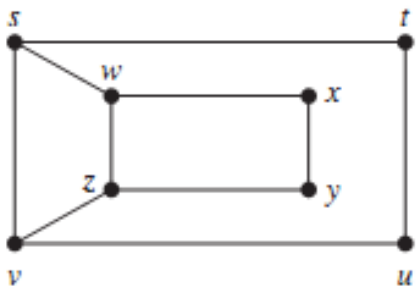
However, *G* and *H* are not isomorphic.

To see this, note that because deg(a) = 2 in *G*, a must correspond to either *t, u, x*, or *y* in *H*, because these are the vertices of degree 2 in *H*.

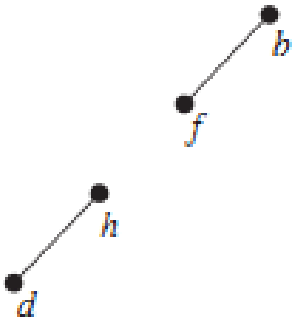
However, each of these 4 vertices in *H* is adjacent to another vertex of degree 2 in *H*, which is not true for a in *G*.

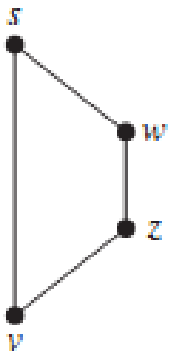


G

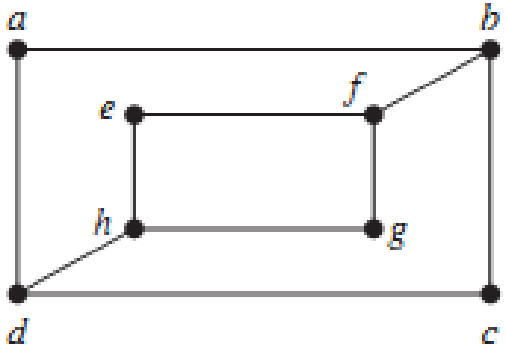


H

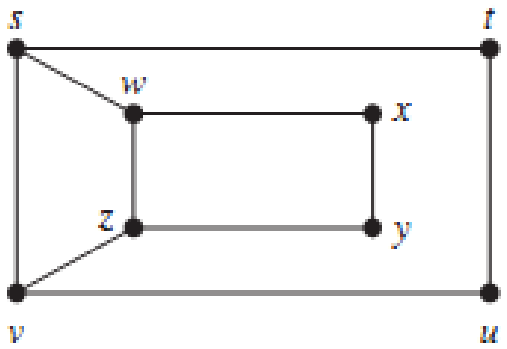




Another way to see that *G* and *H* are not isomorphic is to note that the subgraphs of *G* and *H* made up of vertices of degree 3 and the edges connecting them must be isomorphic if these 2 graphs are isomorphic (Why?). However, these subgraphs, are not isomorphic.



G



H

To show that a function *f* from the vertex set of a graph *G* to the vertex set of a graph *H* is an isomorphism, we need to show that *f* preserves the presence and absence of edges.

One helpful way to do this is to use adjacency matrices.

In particular, to show that *f* is an isomorphism, we can show that the adjacency matrix of *G* is the same as the adjacency matrix of *H*,

when rows and columns of *H* are labeled to correspond to the images under *f* of the vertices in *G* that are the

labels of these rows and columns in the adjacency matrix of *G*.

EXAMPLE

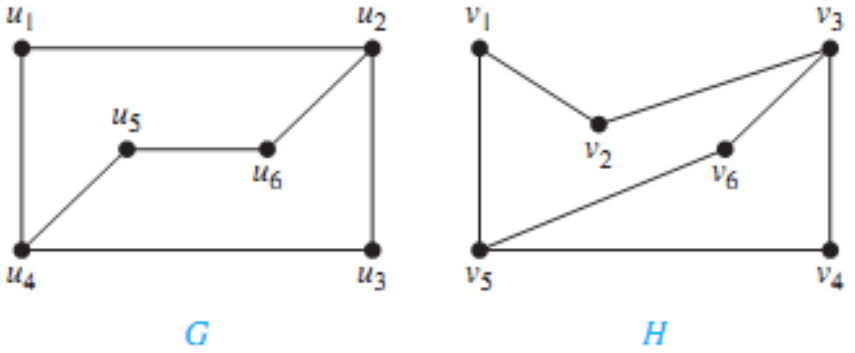
Determine whether the graphs *G* and *H* are isomorphic.

Solution: Both *G* and *H* have 6 vertices and 7 edges.

Both have 4 vertices of degree 2 and 2 vertices of degree 3.

It is also easy to see that the subgraphs of *G* and *H* consisting of all vertices of degree 2 and the edges connecting them are isomorphic.

Because *G* and *H* agree with respect to these invariants, it is reasonable to try to find an isomorphism *f* .



G *H*

We now will define a function f and then determine whether it is an isomorphism.

Because $\deg(u_1) = 2$ and because u_1 is not adjacent to any other vertex of degree 2, the image of u_1 must be either v_4 or v_6, the only vertices of degree 2 in H not adjacent to a vertex of degree 2.

We arbitrarily set $f(u_1) = v_6$. [If we found that this choice did not lead to isomorphism, we would then try $f(u_1) = v_4$.]

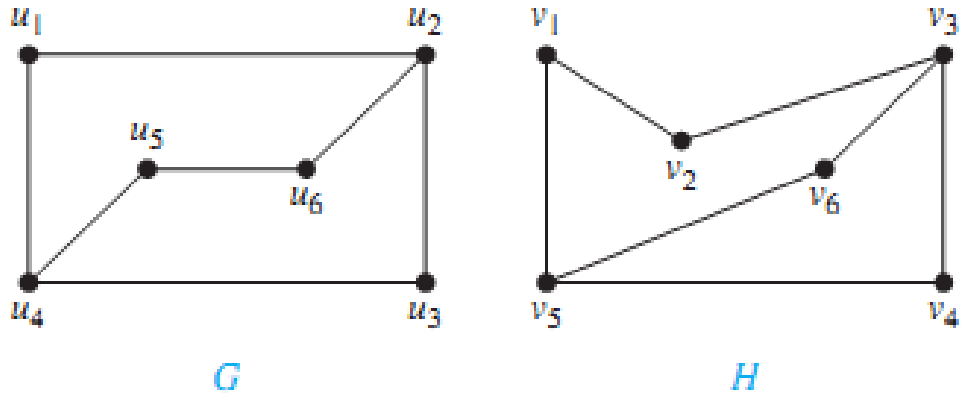
Because u_2 is adjacent to u_1, the possible images of u_2 are v_3 and v_5.

We arbitrarily set $f(u_2) = v_3$.

Continuing in this way, using adjacency of vertices and degrees as a guide, we set $f(u_3) = v_4$, $f(u_4) = v_5$, $f(u_5) = v_1$, and $f(u_6) = v_2$.

We now have a bijection between the vertex set of G and the vertex set of H, namely,

$f(u_1) = v_6$, $f(u_2) = v_3$, $f(u_3) = v_4$, $f(u_4) = v_5$, $f(u_5) = v_1$, $f(u_6) = v_2$.



G H

To see whether *f* preserves edges, we examine the adjacency matrix of *G*, and the adjacency matrix of H with the rows and columns labeled by the images of the corresponding vertices in *G*,

$$\mathbf{A}_G = \begin{array}{c} \\ u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \end{array} \begin{array}{c} \begin{array}{cccccc} u_1 & u_2 & u_3 & u_4 & u_5 & u_6 \end{array} \\ \begin{bmatrix} 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 \end{bmatrix} \end{array}, \qquad \mathbf{A}_H = \begin{array}{c} \\ v_6 \\ v_3 \\ v_4 \\ v_5 \\ v_1 \\ v_2 \end{array} \begin{array}{c} \begin{array}{cccccc} v_6 & v_3 & v_4 & v_5 & v_1 & v_2 \end{array} \\ \begin{bmatrix} 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 \end{bmatrix} \end{array}.$$

ALGORITHMS FOR GRAPH ISOMORPHISM

The best algorithms known for determining whether 2 graphs are isomorphic have exponential worst-case time complexity (in the number of vertices of the graphs).

However, linear average-case time complexity algorithms are known that solve this problem, and there is some hope, but also skepticism, that an algorithm with polynomial worst-case time complexity for determining whether two graphs are isomorphic can be found.

Practical algorithms for determining whether 2 graphs are isomorphic exist for graphs that are restricted in various ways, such as when the maximum degree of vertices is small.

The problem of determining whether any 2 graphs are isomorphic is of special interest because it is one of only a few NP problems not known to be either tractable or NP-complete.

APPLICATIONS OF GRAPH ISOMORPHISMS

Graph isomorphisms, and functions that are almost graph isomorphisms, arise in applications of graph theory to chemistry and to the design of electronic circuits, and other areas including bioinformatics and computer vision.

Chemists use multigraphs, known as molecular graphs, to model chemical compounds.

In these graphs, vertices represent atoms and edges represent chemical bonds between these atoms.

2 structural isomers, molecules with identical molecular formulas but with atoms bonded differently, have nonisomorphic molecular graphs.

When a potentially new chemical compound is synthesized, a database of molecular graphs is checked to see whether the molecular graph of the compound is the same as one already known.

Electronic circuits are modeled using graphs in which vertices represent components and edges represent connections between them.

Modern integrated circuits, known as chips, are miniaturized electronic circuits, often with millions of transistors and connections between them.

Because of the complexity of modern chips, automation tools are used to design them.

Graph isomorphism is the basis for the verification that a particular layout of a circuit produced by an automated tool corresponds to the original schematic of the design.

Graph isomorphism can also be used to determine whether a chip from one vendor includes intellectual property from a different vendor.

This can be done by looking for large isomorphic subgraphs in the graphs modeling these chips.