Algorithms and Complexity

Graph Theory

Lecturer: Dr. Alaa Ahmed Abbood
Lecture 9.
Class 2nd .
Time: 8:30-10:30
Department: Businesses Information Technology (BIT)

Topics covered

- Definitions
- Types
- Terminology
- Representation
- Sub-graphs

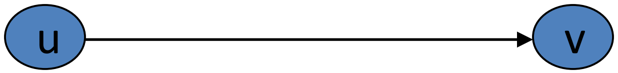
Definitions - Graph

A generalization of the simple concept of a set of dots, links, edges or arcs.

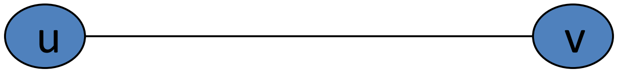
Representation: Graph G =(V, E) consists set of vertices denoted by V, or by V(G) and set of edges E, or E(G)

Definitions – Edge Type

Directed: Ordered pair of vertices. Represented as (u, v) directed from vertex u to v.

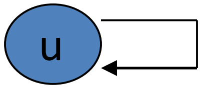


Undirected: Unordered pair of vertices. Represented as {u, v}. Disregards any sense of direction and treats both end vertices interchangeably.



Definitions – Edge Type

Loop: A loop is an edge whose endpoints are equal i.e., an edge joining a vertex to it self is called a loop. Represented as {u, u} = {u}

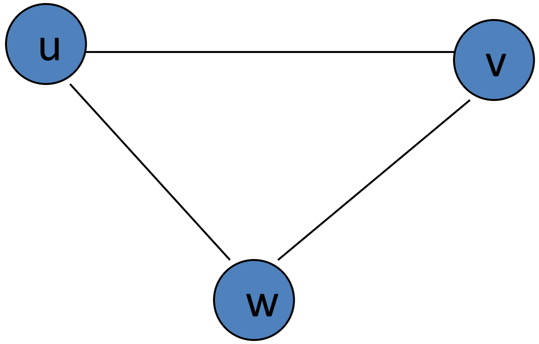


Multiple Edges: Two or more edges joining the same pair of vertices.

Definitions – Graph Type

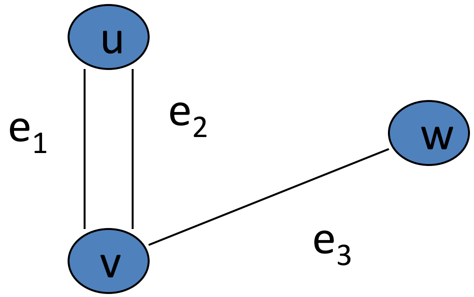
Simple (Undirected) Graph: consists of V, a nonempty set of vertices, and E, a set of unordered pairs of distinct elements of V called edges (undirected)

Representation Example: G(V, E), V = {u, v, w}, E = {{u, v}, {v, w}, {u, w}}



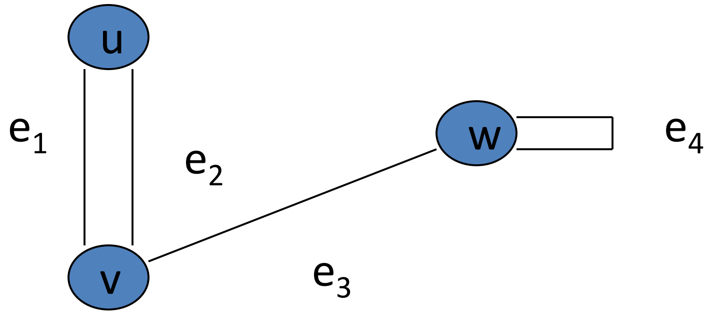
Definitions – Graph Type

- **Multigraph:** G(V,E), consists of set of vertices V, set of Edges E and a function f from E to {{u, v}| u, v V, u $\neq$ v}. The edges e1 and e2 are called multiple or parallel edges if f (e1) = f (e2).
- Representation Example: V = {u, v, w}, E = {e_1, e_2, e_3}



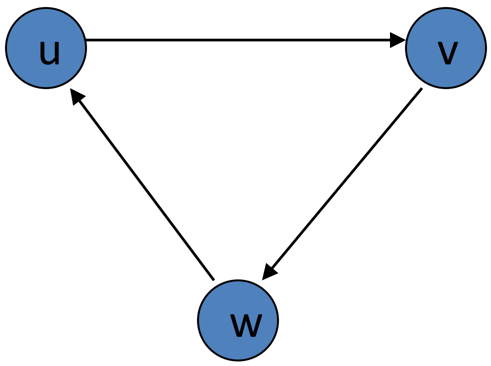
Definitions – Graph Type

- **Pseudograph:** G(V,E), consists of set of vertices V, set of Edges E and a function F from E to {{u, v}| u, v Î V}. Loops allowed in such a graph.
- Representation Example: V = {u, v, w}, E = {e_1, e_2, e_3, e_4}



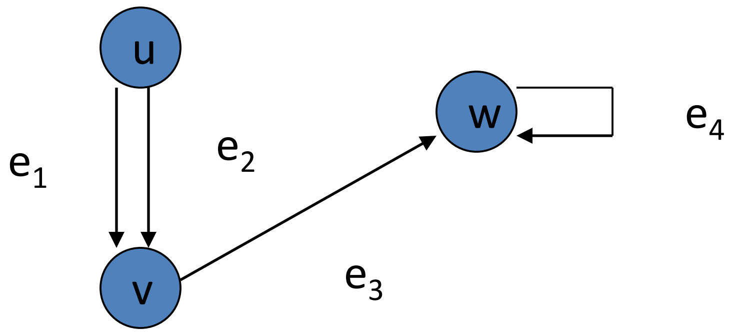
Definitions – Graph Type

- **Directed Graph:** G(V, E), set of vertices V, and set of Edges E, that are ordered pair of elements of V (directed edges)
- Representation Example: G(V, E), V = {u, v, w}, E = {(u, v), (v, w), (w, u)}



Definitions – Graph Type

- **Directed Multigraph:** G(V,E), consists of set of vertices V, set of Edges E and a function f from E to {{u, v}| u, v V}. The edges e1 and e2 are multiple edges if f(e1) = f(e2)
- Representation Example: V = {u, v, w}, E = {e_1, e_2, e_3, e_4}



Definitions – Graph Type

Type	Edges	Multiple Edges Allowed ?	Loops Allowed ?
Simple Graph	undirected	No	No
Multigraph	undirected	Yes	No
Pseudograph	undirected	Yes	Yes
Directed Graph	directed	No	Yes
Directed Multigraph	directed	Yes	Yes

Terminology – Undirected graphs

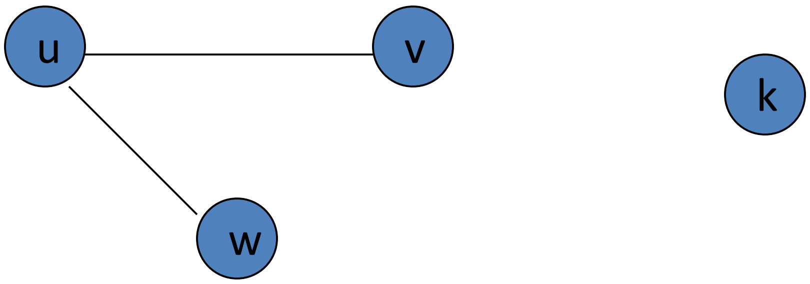
- u and v are **adjacent** if {u, v} is an edge, e is called **incident** with u and v. u and v are called **endpoints** of {u, v}

- **Degree of Vertex (deg (v)):** the number of edges incident on a vertex. A loop contributes twice to the degree.

- **Pendant Vertex:** deg (v) =1

- **Isolated Vertex:** deg (k) = 0

Representation Example: For V = {u, v, w} , E = { {u, w}, {u, v} }, deg (u) = 2, deg (v) = 1, deg (w) = 1, deg (k) = 0, w and v are pendant , k is isolated

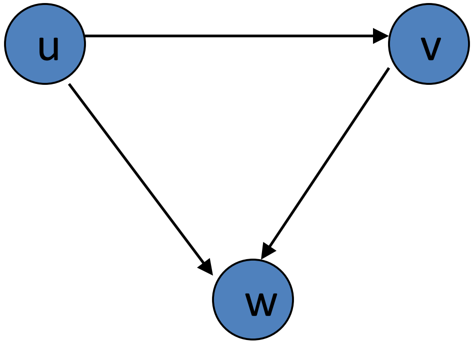


Terminology – Directed graphs

- For the edge (u, v), u is **adjacent to** v OR v is **adjacent from** u, u – **Initial vertex**, v – **Terminal vertex**
- **In-degree (deg^- (u)):** number of edges for which u is terminal vertex
- **Out-degree (deg^+ (u)):** number of edges for which u is initial vertex

Note: A loop contributes 1 to both in-degree and out-degree

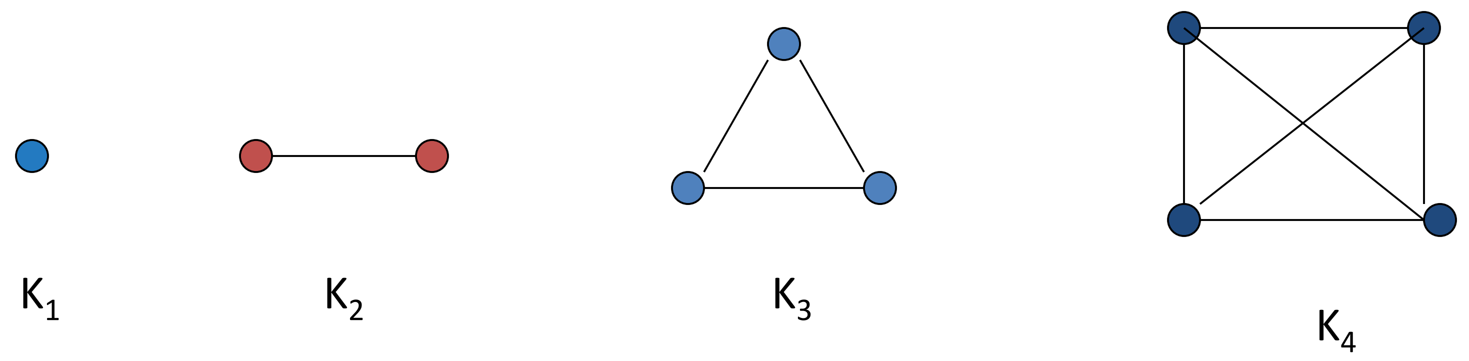
Representation Example: For V = {u, v, w} , E = { (u, w), (v, w), (u, v) }, deg^- (u) = 0, deg^+ (u) = 2, deg^- (v) = 1,
deg^+ (v) = 1, and deg^- (w) = 2, deg^+ (w) = 0



Simple graphs – special cases

- **Complete graph:** K_n, is the simple graph that contains exactly one edge between each pair of distinct vertices.

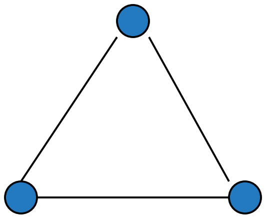
  Representation Example: K_1, K_2, K_3, K_4



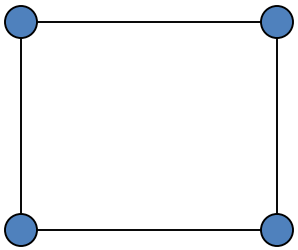
Simple graphs – special cases

- **Cycle:** C_n, $n \geq 3$ consists of n vertices $v_1, v_2, v_3 \ldots v_n$ and edges $\{v_1, v_2\}, \{v_2, v_3\}, \{v_3, v_4\} \ldots \{v_{n-1}, v_n\}, \{v_n, v_1\}$

Representation Example: C_3, C_4



C_3

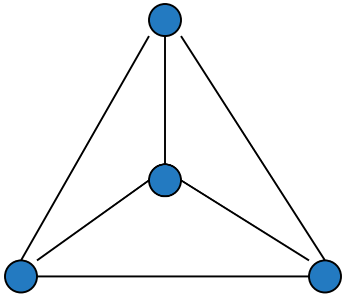


C_4

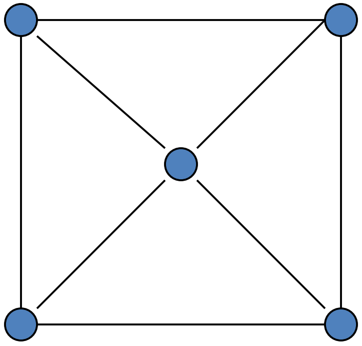
Simple graphs – special cases

- **Wheels:** W_n, obtained by adding additional vertex to Cn and connecting all vertices to this new vertex by new edges.

Representation Example: W_3, W_4



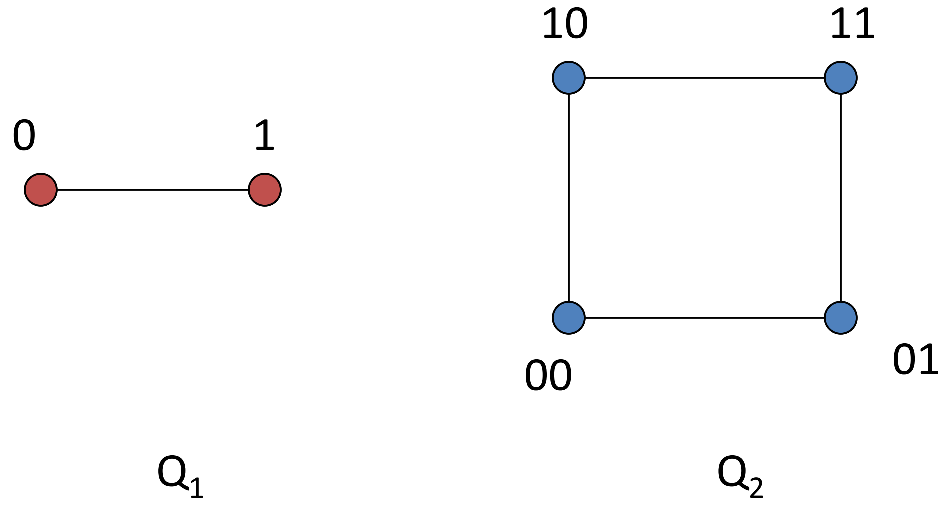
W_3



W_4

Simple graphs – special cases

- **N-cubes:** Q_n, vertices represented by $2n$ bit strings of length n. Two vertices are adjacent if and only if the bit strings that they represent differ by exactly one bit positions
- Representation Example: Q_1, Q_2

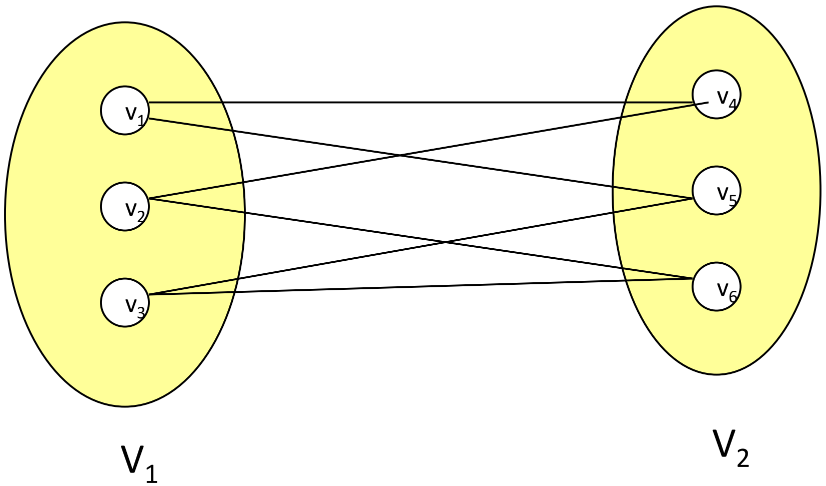


Bipartite graphs

- In a simple graph G, if V can be partitioned into two disjoint sets V_1 and V_2 such that every edge in the graph connects a vertex in V_1 and a vertex V_2 (so that no edge in G connects either two vertices in V_1 or two vertices in V_2)

  Application example: Representing Relations

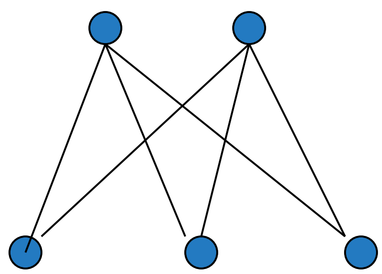
  Representation example: $V_1 = \{v_1, v_2, v_3\}$ and $V_2 = \{v_4, v_5, v_6\}$,



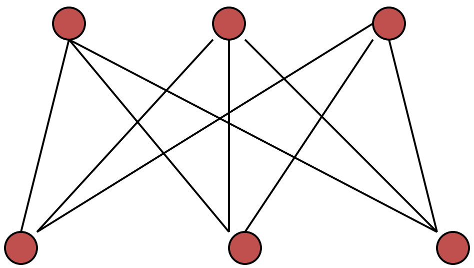
Complete Bipartite graphs

- $K_{m,n}$ is the graph that has its vertex set portioned into two subsets of m and n vertices, respectively There is an edge between two vertices if and only if one vertex is in the first subset and the other vertex is in the second subset.

Representation example: $K_{2,3,}$ $K_{3,3}$



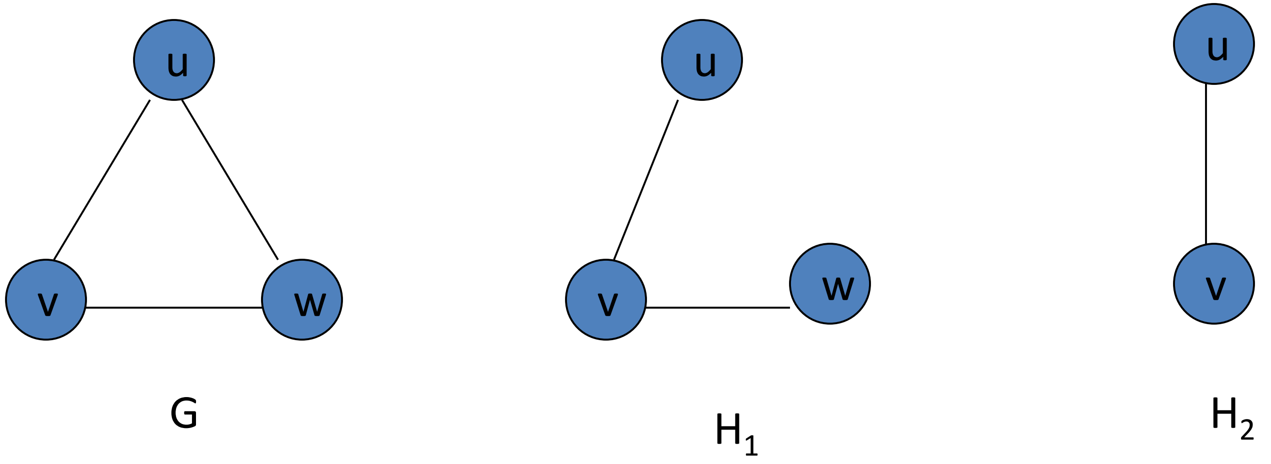
$K_{2,3}$



$K_{3,3}$

Subgraphs

- A subgraph of a graph G = (V, E) is a graph H =(V', E') where V' is a subset of V and E' is a subset of E
- Application example: solving sub-problems within a graph
- Representation example: V = {u, v, w}, E = ({u, v}, {v, w}, {w, u}}, H_1 , H_2

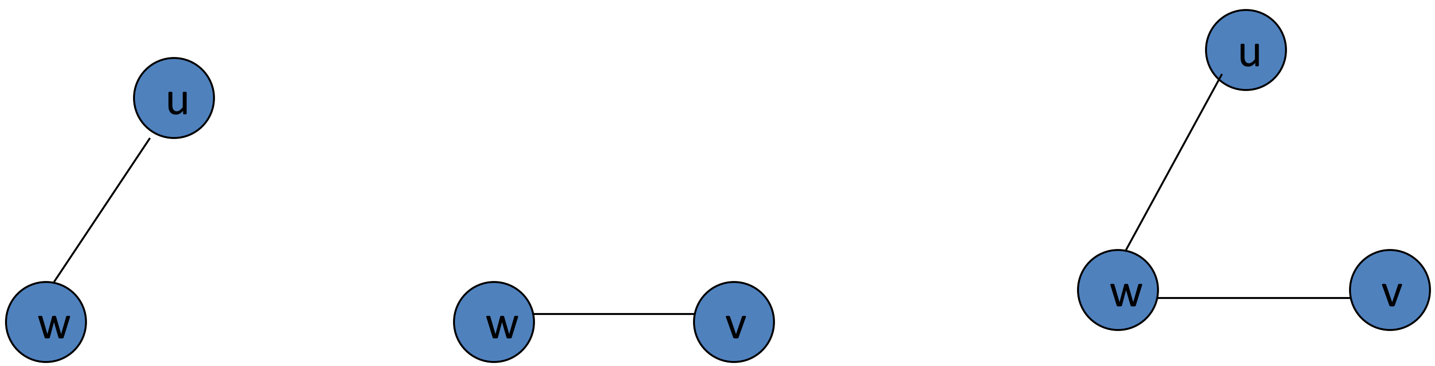


Subgraphs

- G = G1 U G2 wherein E = E1 U E2 and V = V1 U V2, G, G1 and G2 are simple graphs of G

Representation example: V1 = {u, w}, E1 = {{u, w}}, V2 = {w, v},

E2= {{w, v}}, V = {u, v ,w}, E = {{{u, w}, {{w, v}}



G1 G2 G

Representation

- **Incidence (Matrix):** Most useful when information about edges is more desirable than information about vertices.

- **Adjacency (Matrix/List):** Most useful when information about the vertices is more desirable than information about the edges. These two representations are also most popular since information about the vertices is often more desirable than edges in most applications

Representation- Incidence Matrix

- G = (V, E) be an unditected graph. Suppose that $v_1, v_2, v_3, \ldots, v_n$ 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 nx m matrix M = [m $_{ij}$], where

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

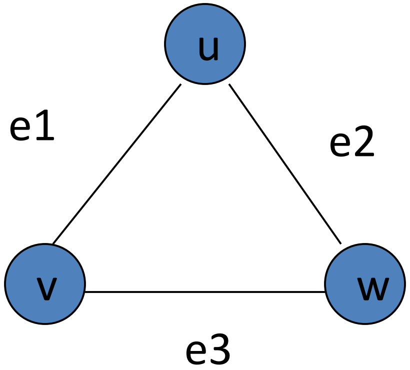
Can also be used to represent :

Multiple edges: by using columns with identical entries, since these edges are incident with the same pair of vertices

Loops: by using a column with exactly one entry equal to 1, corresponding to the vertex that is incident with the loop

Representation- Incidence Matrix

- Representation Example: G = (V, E)



	e_1	e_2	e_3
v	1	0	1
u	1	1	0
w	0	1	1

Representation- Adjacency Matrix

- There is an N x N matrix, where |V| = N , the Adjacenct Matrix (NxN) A = $[a_{ij}]$

For undirected graph

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

- **For directed graph**

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

- This makes it easier to find subgraphs, and to reverse graphs if needed.

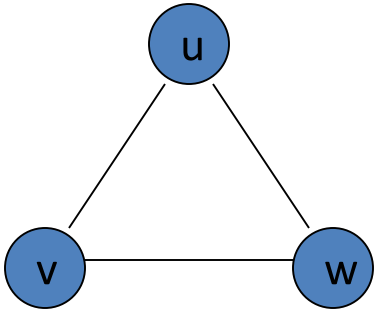
Representation- Adjacency Matrix

- Adjacency is chosen on the ordering of vertices. Hence, there as are as many as n! such matrices.
- The adjacency matrix of simple graphs are symmetric ($a_{ij} = a_{ji}$) (why?)
- When there are relatively few edges in the graph the adjacency matrix is a **sparse matrix**
- Directed Multigraphs can be represented by using aij = number of edges from v_i to v_j

Note: Sparse matrix is a matrix that most values are zero.

Representation- Adjacency Matrix

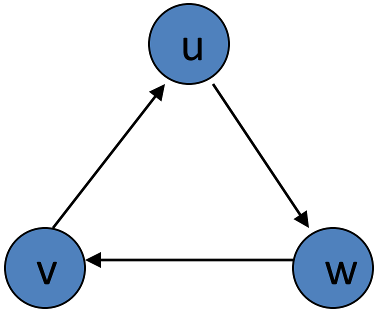
- Example: Undirected Graph G (V, E)



	v	u	w
v	0	1	1
u	1	0	1
w	1	1	0

Representation- Adjacency Matrix

- Example: directed Graph G (V, E)

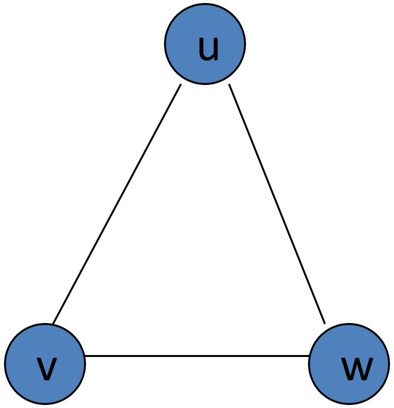


	v	u	w
v	0	1	0
u	0	0	1
w	1	0	0

Representation- Adjacency List

Each node (vertex) has a list of which nodes (vertex) it is adjacent

Example: undirectd graph G (V, E)



node	Adjacency List
u	v , w
v	w, u
w	u , v

كلية معلوماتية الاعمال

THANK YOU