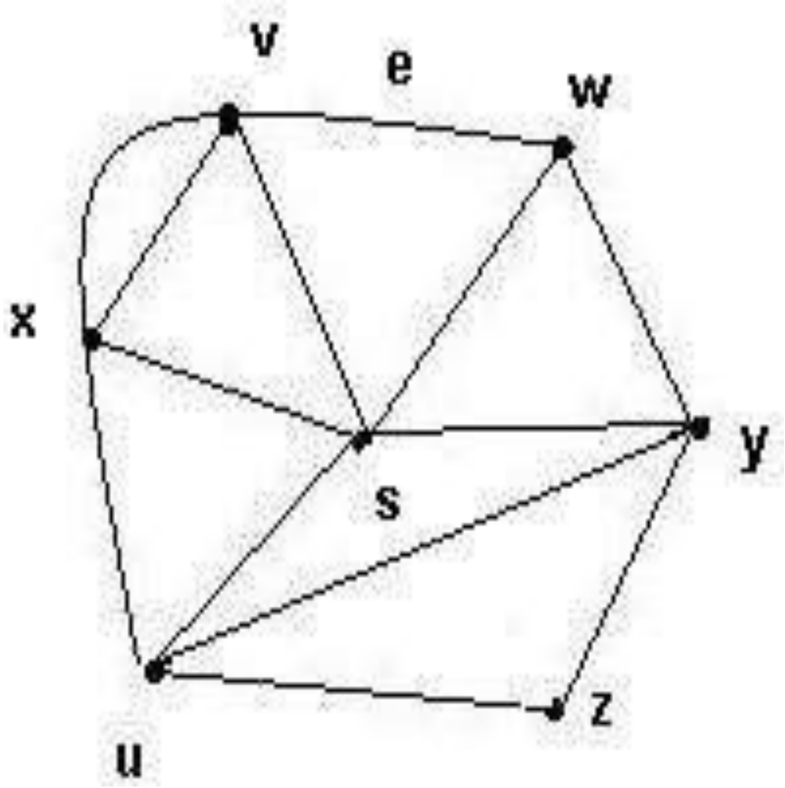
Concepts of Graph Theory

Introduction

What is a graph ?

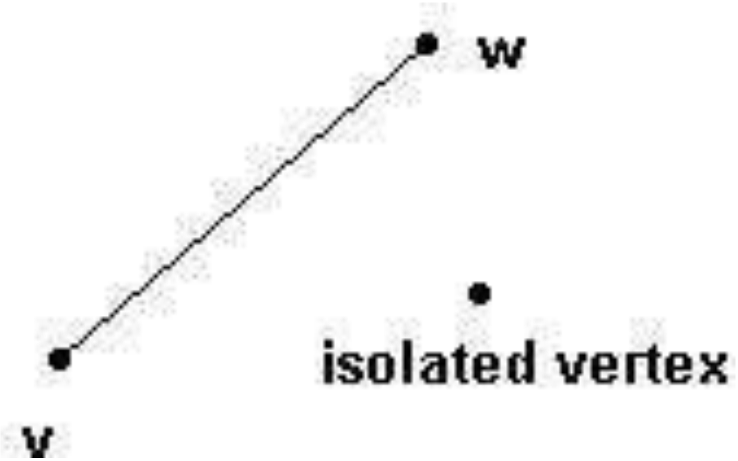
- **It is a pair G = (V, E), where**
  - V = V(G) = set of vertices
  - E = E(G) = set of edges

- **Example:**
  - V = {s, u, v, w, x, y, z}
  - E = {(x,s), (x,v), (x,u), (v,w), (s,v), (s,u), (s,w), (s,y), (w,y), (u,y), (u,z),(y,z)}



Edges

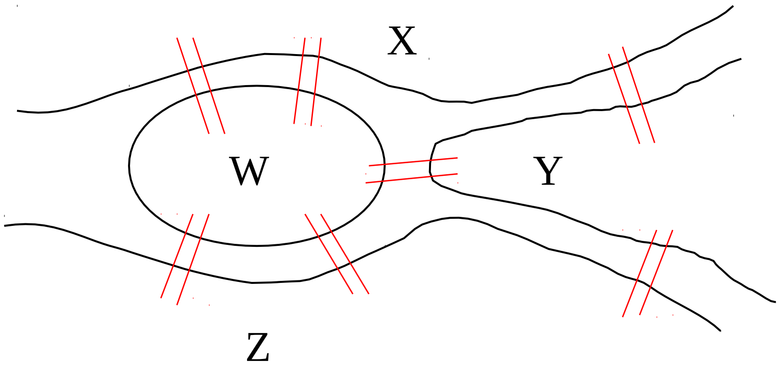
- An edge may be labeled by a pair of vertices, for instance e = (v,w).
- e is said to be *incident* on v and w.
- Isolated vertex = a vertex without incident edges.



The Königsberg Bridge Problem

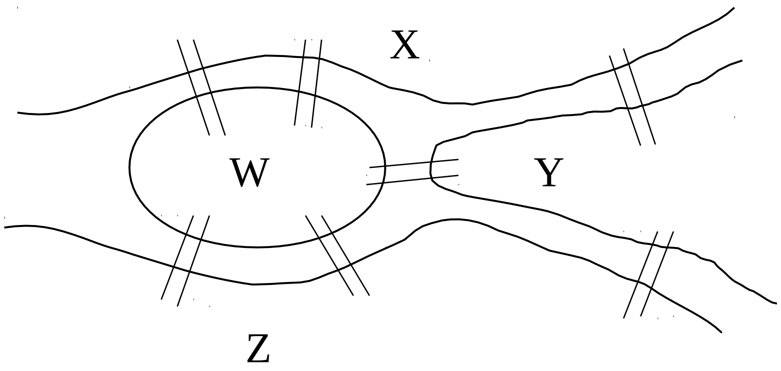
- Königsber is a city on the Pregel river in Prussia
- The city occupied two islands plus areas on both banks
- Problem:

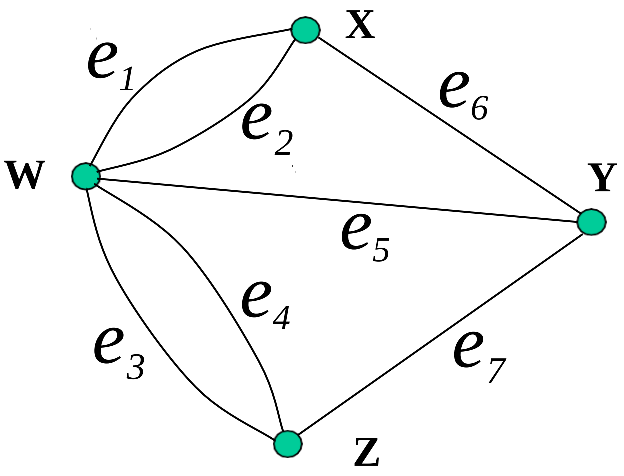
  Whether they could leave home, cross every bridge exactly once, and return home.



A Model

- A *vertex* : a region
- An *edge* : a path(bridge) between two regions





Order and Size

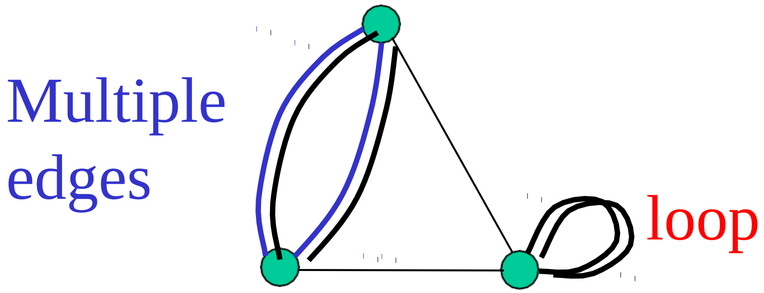
- **The *order* of a graph G, written $n(G)$, is the number of vertices in G.**

- **An *n-vertex graph* is a graph of order n.**
- **The *size* of a graph G, written $e(G)$, is the number of edges in G.**

- **For $n \in N$, the notation $[n]$ indicates the set $\{1,\ldots, n\}$.**

Loop, Multiple edges

- *Loop* : **An edge whose endpoints are equal**
- *Multiple edges* : **Edges have the same pair of endpoints**



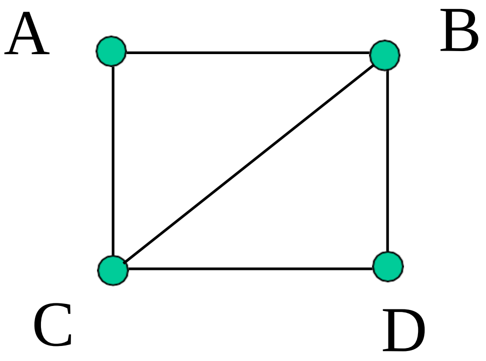
Multiple edges

loop

Adjacent, neighbors

- **Two vertices are** *adjacent* **and are** *neighbors* **if they are the endpoints of an edge**

- **Example:**
  - *A* **and** *B* **are adjacent**
  - *A* **and** *D* **are not adjacent**

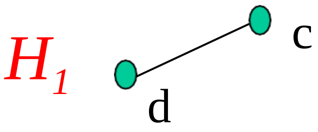


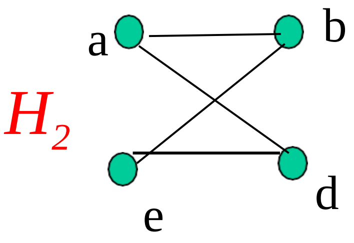
Finite Graph, Null Graph

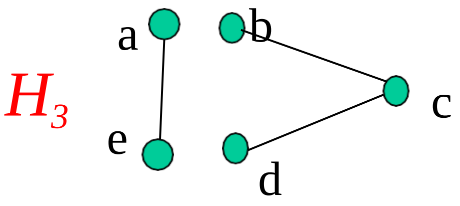
- *Finite graph* **: an graph whose vertex set and edge set are finite**
- *Null graph* **: the graph whose vertex set and edges are empty**

Connected and Disconnected

- ***Connected*** : There exists at least one path between two vertices
- ***Disconnected*** : Otherwise
- Example:
  - H_1 **and** H_2 **are connected**
  - H_3 **is disconnected**





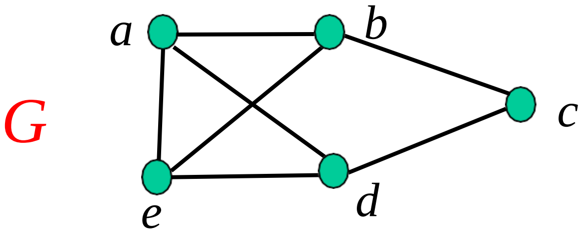


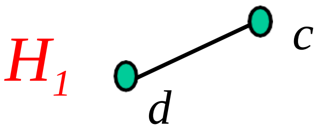
Subgraphs

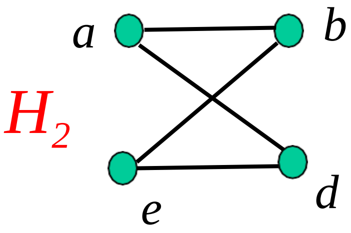
- A *subgraph* of a graph G is a graph H such that:
  - $V(H) \subseteq V(G)$ and $E(H) \subseteq E(G)$ and
  - The assignment of endpoints to edges in H is the same as in G.

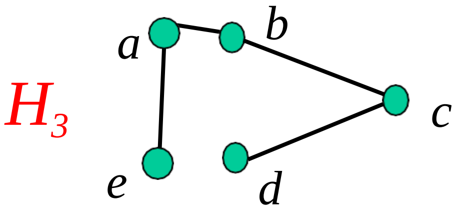
Subgraphs

- **Example:** H_1, H_2, **and** H_3 **are subgraphs of** G









Walks, Trails

- A *walk* : a list of vertices and edges $v_0, e_1, v_1,$ …., e_k, v_k such that, for $1 \le i \le k$, the edge e_i has endpoints v_{i-1} and v_i .

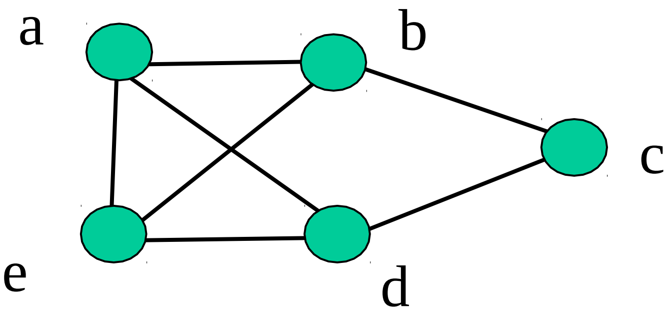
- A *trail* : a walk with no repeated edge.

Paths

- A u,v-walk or u,v-trail has first vertex u and last vertex v; these are its endpoints.
- A u,v-*path* : a u,v-trail with no repeated vertex.
- The *length* of a walk, trail, path, or cycle is its number of edges.
- A walk or trail is *closed* if its endpoints are the same.

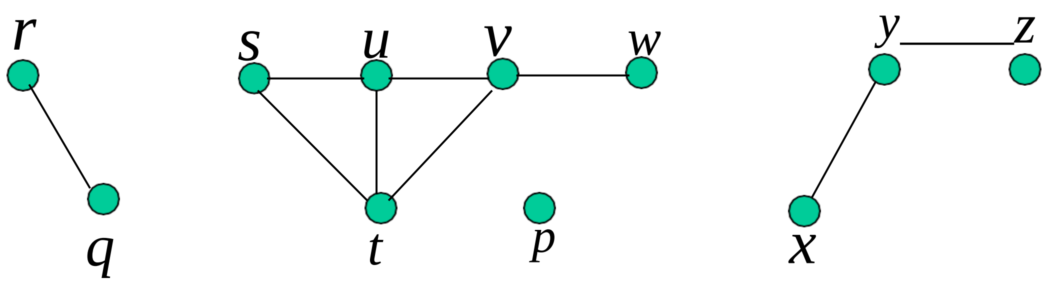
Path and Cycle

- ***Path*** **: a sequence of distinct vertices such that two consecutive vertices are adjacent**
  - **Example:** (*a*, *d*, *c*, *b*, *e*) **is a path**
  - (*a*, *b*, *e*, *d*, *c*, *b*, *e*, *d*) **is not a path; it is a walk**
- ***Cycle*** **: a closed Path**
  - **Example:** (*a*, *d*, *c*, *b*, *e*, *a*) **is a cycle**



Components

- The *components* of a graph G are its **maximal** connected subgraphs
- A component (or graph) is *trivial* if it has no edges; otherwise it is nontrivial
- An *isolated vertex* is a vertex of degree 0



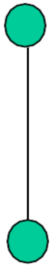
Theorem: Every graph with n vertices and k edges has at least n-k components.

Proof**:**

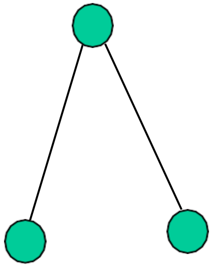
- An n-vertex graph with no edges has n components
- Each edge added reduces this by at most 1
- If k edges are added, then the number of components is at least $n - k$

Theorem: Every graph with *n* vertices and *k* edges has at least *n-k* components.

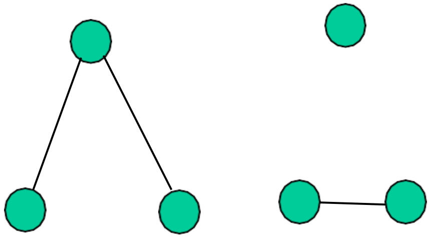
- **Examples:**



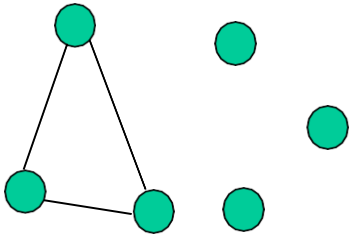
n =2, *k* =1,
1 component



n =3, *k* =2,
1 component



n =6, *k* =3,
3 components

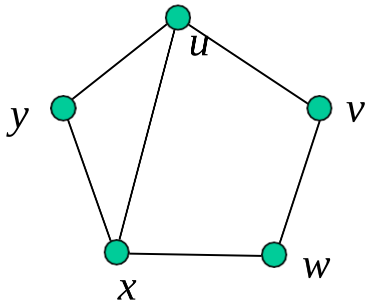


n =6, *k* =3,
4 components

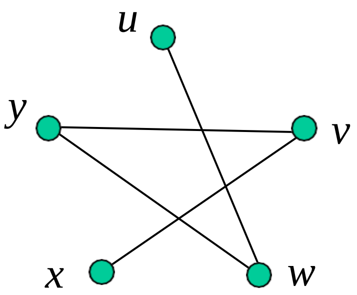
Complement

- ***Complement of*** G: **The complement** G' **of a simple graph** G :
  - **A simple graph**
  - $V(G') = V(G)$
  - $E(G') = \{ uv \mid uv \notin E(G) \}$

G

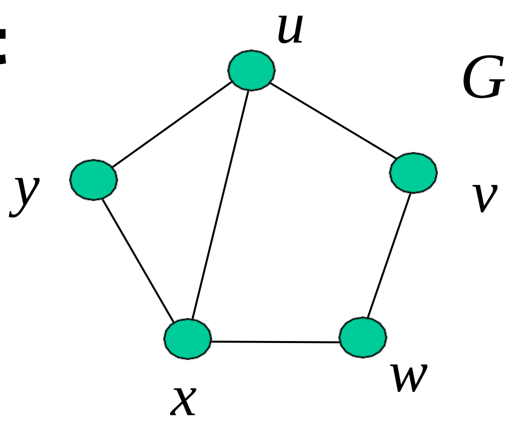


G’



Clique and Independent set

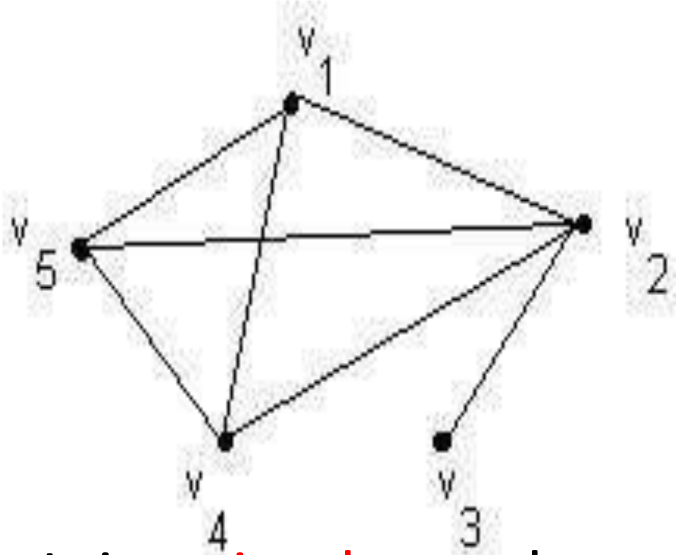
- **A *Clique* in a graph: a set of pairwise adjacent vertices (a complete subgraph)**
- **An *independent set* in a graph: a set of pairwise nonadjacent vertices**
- **Example:**
  - $\{x, y, u\}$ **is a clique in** G
  - $\{u, w\}$ **is an independent set**



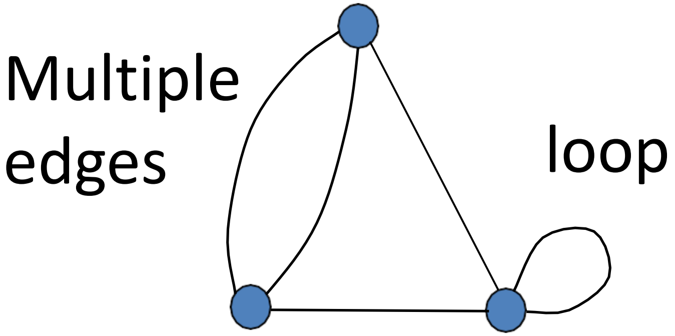
Simple graph

A simple graph ***G (V , E)*** consists of a non-empty set of vertices ‘***V***’ and a set ‘***E***’ of edges,

such that each edge ***‘e’*** belongs to ***E*** is associated with an unordered pair of distinct vertices, called its endpoints.



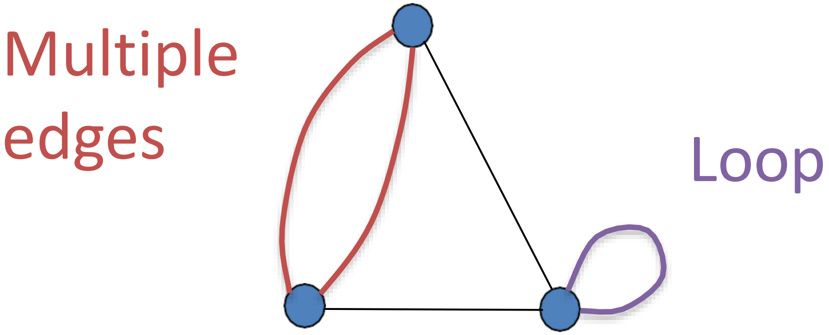
It is a simple graph.



It is not simple graph.

Multigraphs (or Pseudo-graphs)

- *Loop* : An edge whose endpoints are same
- *Multiple edges* : Edges have the same pair of endpoints



Multiple edges

Loop

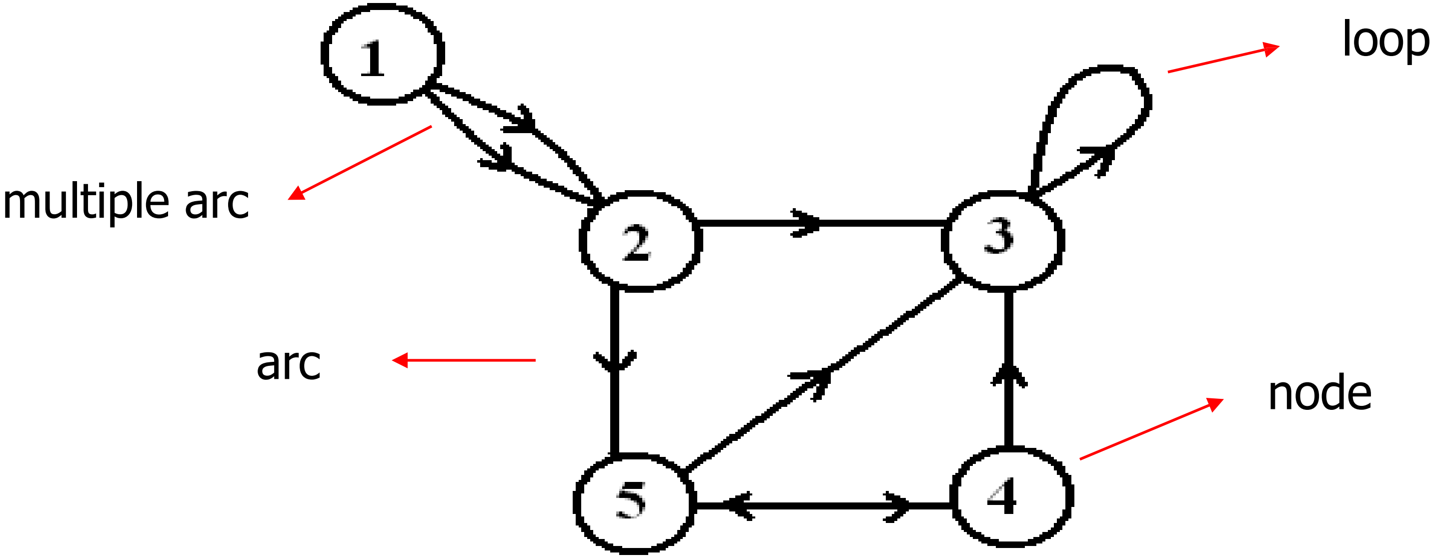
Unlike simple graphs, it contains the edges with same endpoints.

Note: Multigraphs doesn’t contain loops.

Digraphs

A digraph is a pair ***G* = (*V* , *A*)** of a set ‘*V*’, whose elements are called *vertices* or *nodes*,

With a set ‘*A*’ of ordered pairs of vertices, called *arcs*, *directed edges*, or *arrows* (and sometimes simply *edges* with the corresponding set named *E* instead of *A*).



loop

multiple arc

arc

node

Graph Terminology

Type	Edges	Multiple Edges Allowed?	Loops Allowed?
Simple graph	Undirected	No	No
Multigraph	Undirected	Yes	No
Pseudo graph	Undirected	Yes	Yes
Simple Directed graph	Directed	No	No
Directed Multigraph	Directed	Yes	Yes
Mixed graph	Undirected & Directed	Yes	Yes

Graph Models

1. *NICHE OVERLAP* Graphs in Ecosystem

Graphs are used in many models involving the interaction of different species of animals. For instance, the competition b/w species in an ecosystem can be modeled using a niche overlap graph.

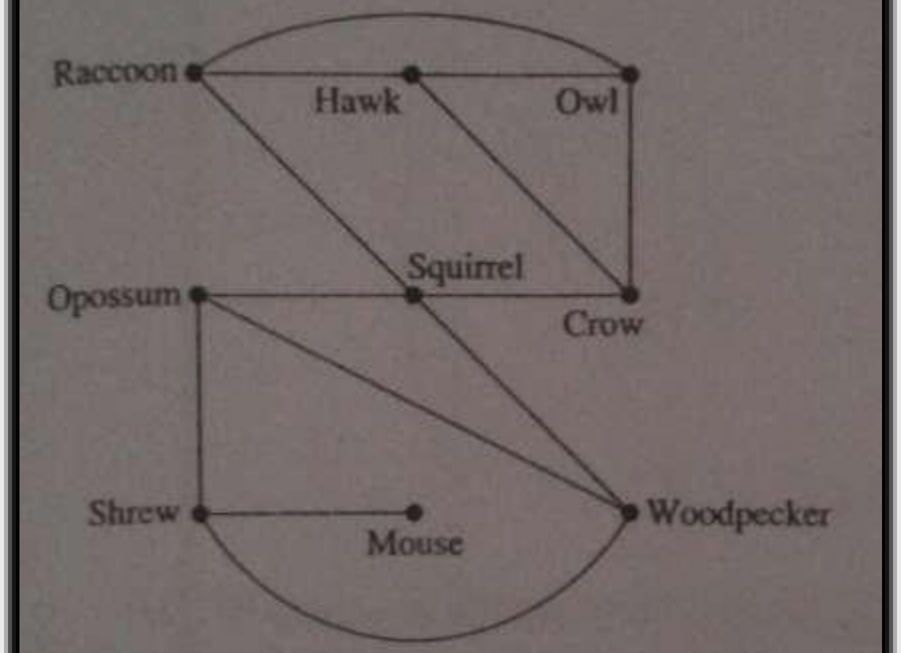


FIGURE 6 A Niche Overlap Graph.

2. **Round-Robin Tournaments**

A tournament where each team plays each other team exactly once is called a round-robin tournament. Such tournaments can be modeled using directed graphs where each team is represented by a vertex.

Note: (a , b) is an edge if team *a* beats team *b* and if the edge is directed in the vertex then that team is defeated by the other one and vice-versa.

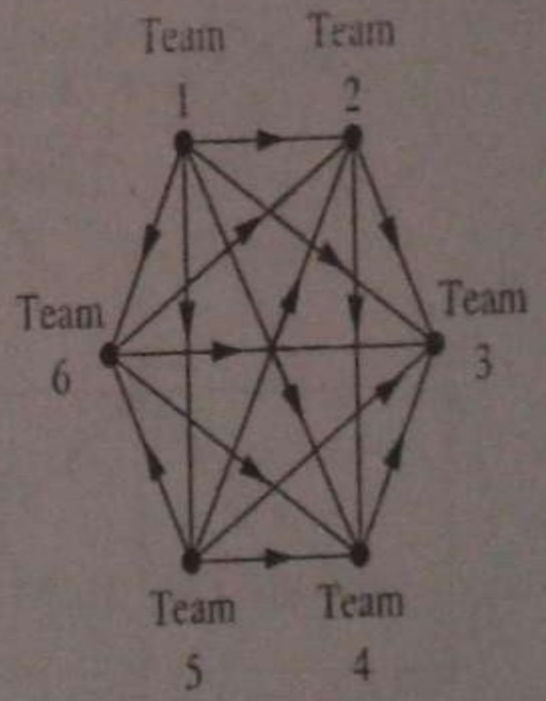


FIGURE 9 A Graph Model of a Round-Robin Tournament.

Degree of graph

The degree of a vertex in a simple graph, denoted by deg(v), is the number of edges incident on it.

Degree also means number of adjacent vertices.

For a Node,

- The number of head endpoints adjacent to a node is called the ***Indegree*** and it is denoted by ***deg^{-} (v).***
- The number of tail endpoints adjacent to a node is called ***Outdegree*** and it is denoted by ***deg^{+} (v).***

For example in the digraph,

Out-deg(1) = 2

In-deg(1) = 0

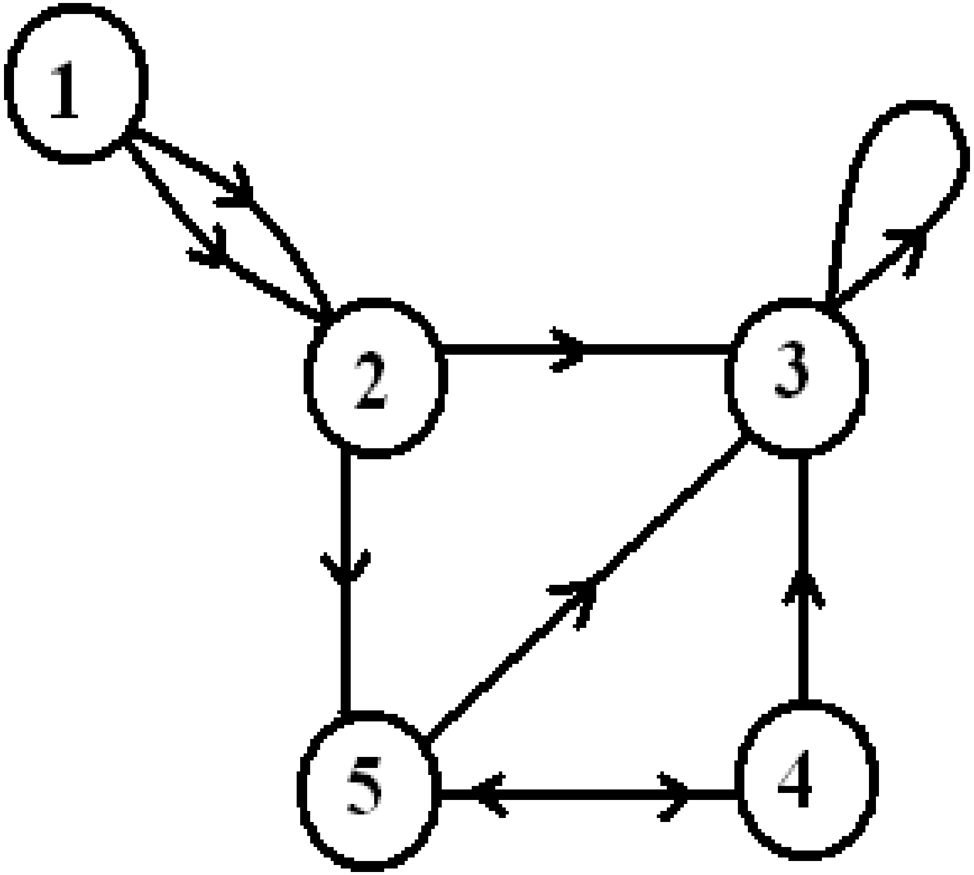
Out-deg(2) = 2

In-deg(2) = 2

Out-deg(3) = 1

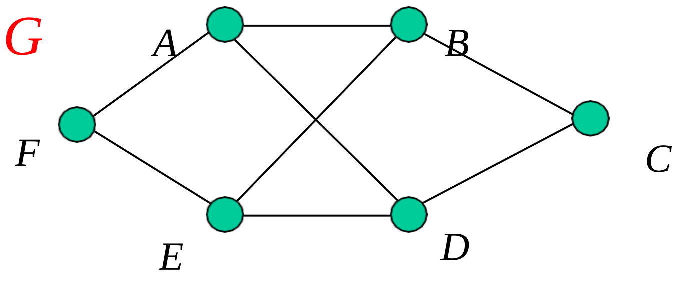
In-deg(3) = 4

And so on…….



Degree of Graph

- The degree of vertex v in a graph G, written or $d(v)$, is the number of edges incident to v, except that each loop at v counts twice
- The maximal degree is $\Delta(G)$
- The minimum degree is $\delta(G)$



$d(B) = 3, \ d(C) = 2$

$\Delta(G) = 3, \delta(G) = 2$

Handshaking Lemma

Let ***G* = (*V* , *E*)** be an undirected graph with ***e*** edges. Then,

$$\mathbf{2}\boldsymbol{e} = \quad {}_{v\in V}\, \boldsymbol{deg}(\ ).$$

For example,

How many edges are there in a graph with 10 vertices each of degree six?

Solution:

$$\text{deg}(\ ) = 6*10 = 60$$

which follows 2e = 60 ,i.e., e = 30

Corollary of Handshaking theorem

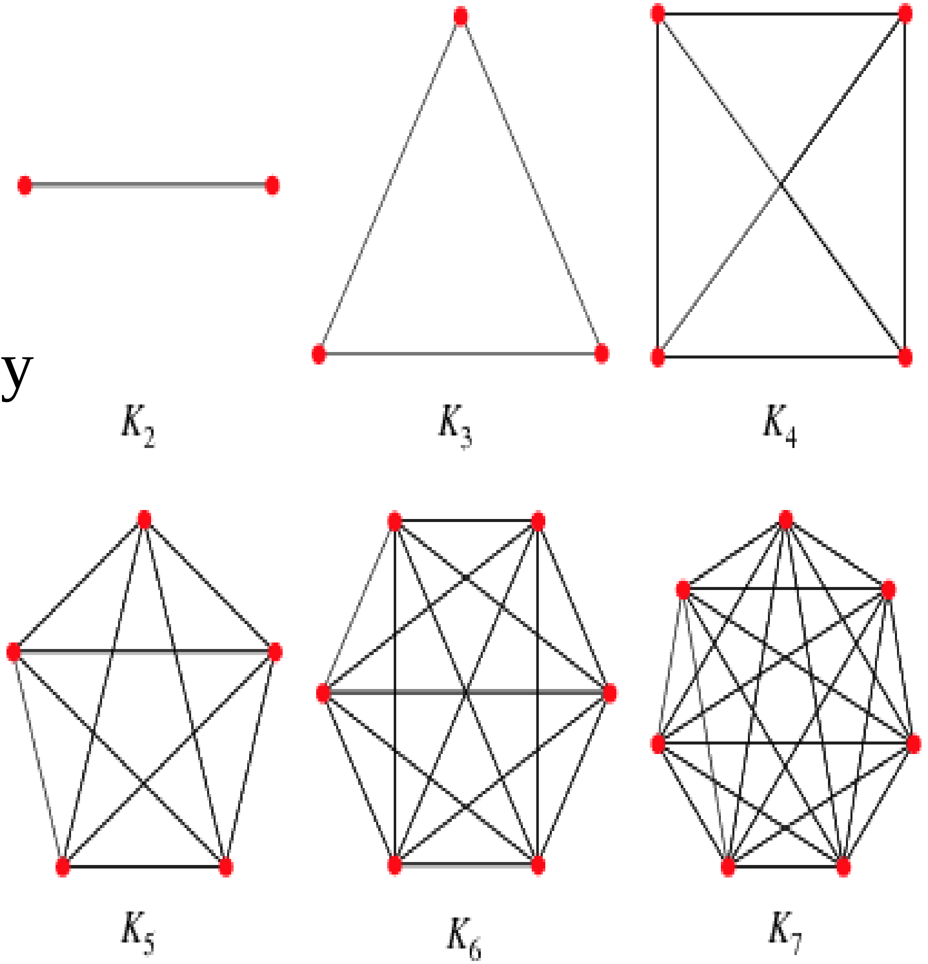
Theorem:

In any simple graph, there are an even number of vertices of odd degree.

Some Special Graphs

1. **Complete graph K_n :**

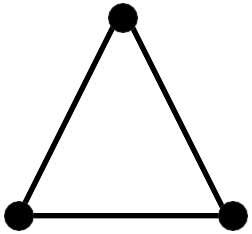
The *complete graph* K_n is the graph with ‘*n*’ vertices and every pair of vertices is joined by an edge , like in Mesh topology.



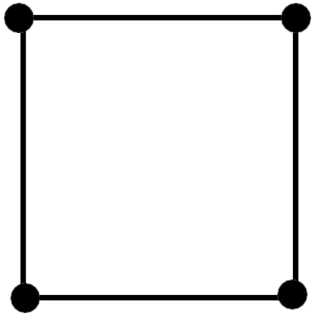
The figure represents K_2,K_3,………,K_7.

2. Cycle Graph :

A cycle graph C_n , where $n >= 3$, sometimes simply known as an *n*- Cycle, is a graph on *n* nodes , 1,2,…….,*n* , and edges {1,2} , {2,3} , …. , {n-1,n} , {n,1}.

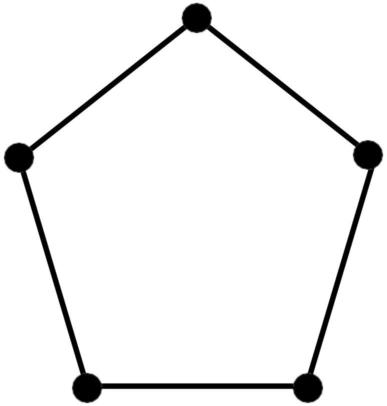


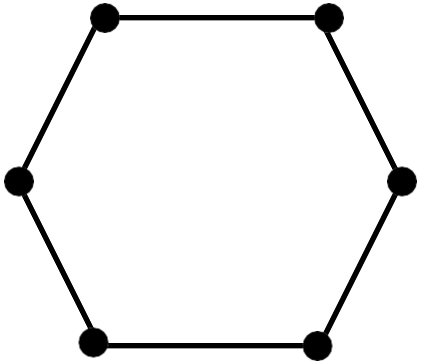
C_3



C_4

C_5

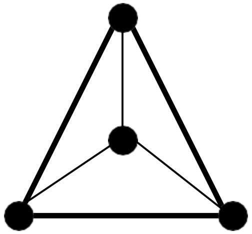




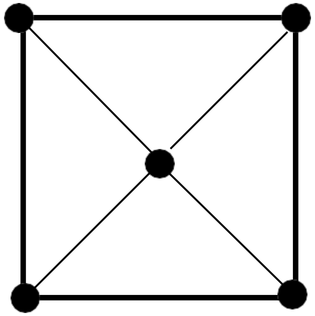
C_6

3. **Wheel Graph :**

A Wheel graph W_n contain an additional vertex to the cycle C_n, for n>=3 , and connect this new vertex to each of the n vertices in C_n, by new edges. The wheels W_3 , W_4 , W_5 , W_6 are displayed below.

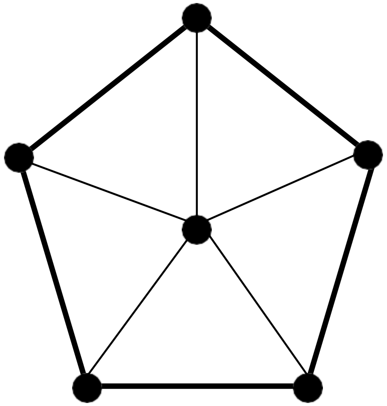


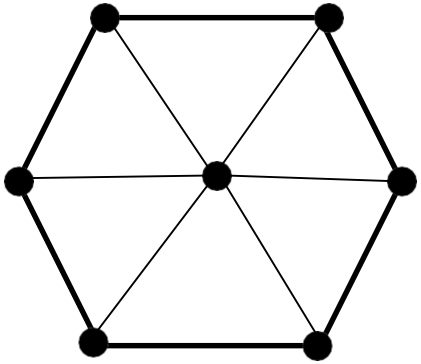
W_3



W_4

W_5





W_6

4. N-Cube :

The n-cube (hypercube) Q_n is the graph whose vertices represent 2^n bit strings of length n. Two vertices are adjacent if and only if the bit strings differ in exactly one position.

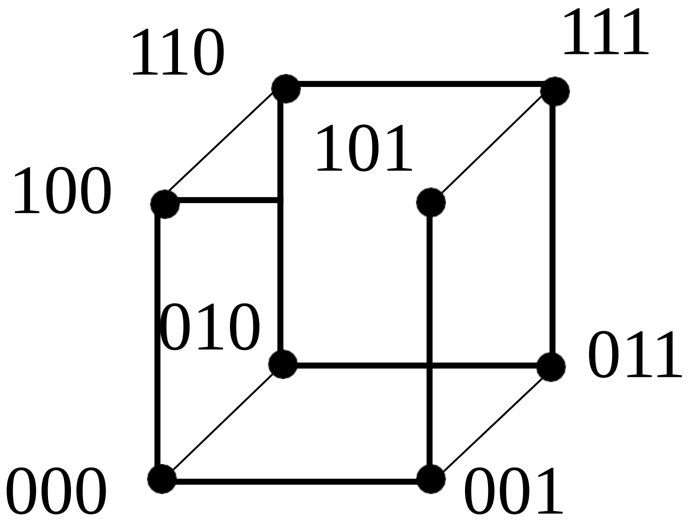
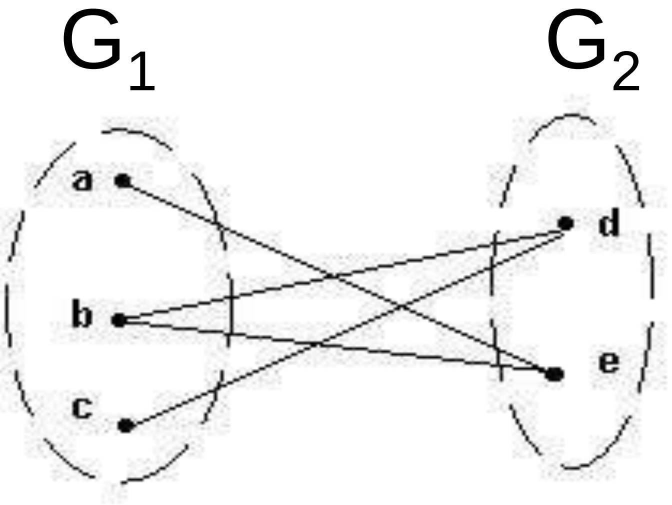


Figure represents Q_3

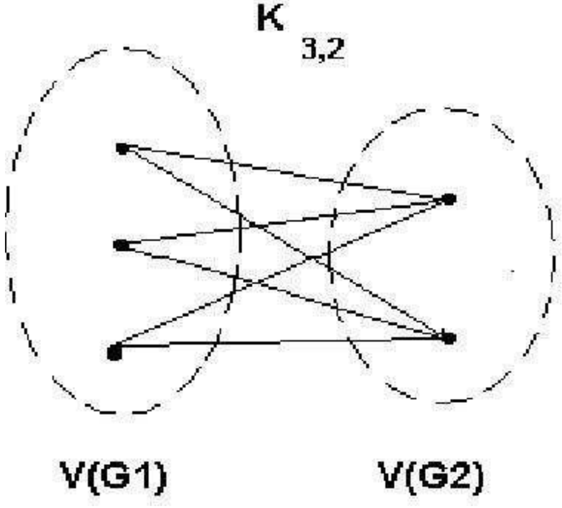
Bipartite Graphs

- A bipartite graph G is a graph such that
  - $V(G) = V(G_1) \cup V(G_2)$
  - $|V(G_1)| = m$, $|V(G_2)| = n$
  - $V(G_1) \cap V(G_2) = \emptyset$

  No edges exist between any two vertices in the same subset $V(G_k)$, $k = 1,2$



Complete Bipartite graph $K_{m,n}$



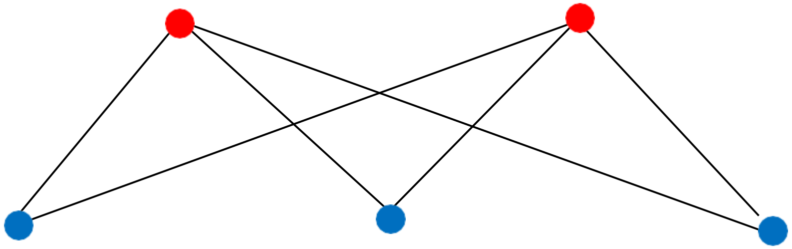
A bipartite graph is the *complete* bipartite graph $K_{m,n}$ if every vertex in $V(G_1)$ is joined to a vertex in $V(G_2)$ and conversely,

$|V(G_1)| = m$, $|V(G_2)| = n$

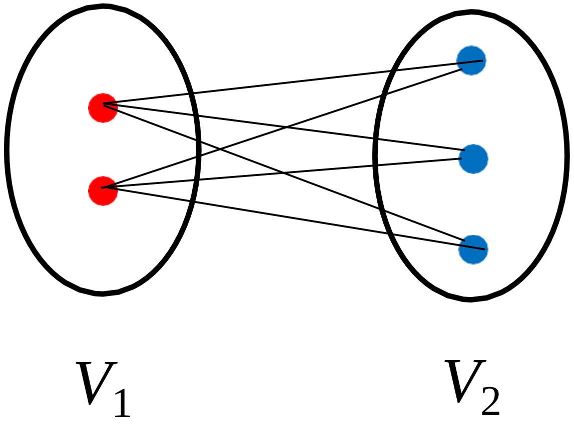
Bipartite Graphs in terms of Graph Coloring

Theorem:

A simple graph is bipartite if and only if it is possible to assign one of two different colors to each vertex of the graph so that no two adjacent vertices are assigned the same color.



$K_{2\,,\,3}$



Proof:

First, suppose that $G = (V , E)$ is a simple graph. Then $V = V_1 U V_2$, where V_1 and V_2 are disjoint sets and every edge in E connects a vertex in V_1 and a vertex in V_2. If we assign one color to each vertex in V_1 and a second color to each vertex in V_2, then no two adjacent vertices are assigned the same color.

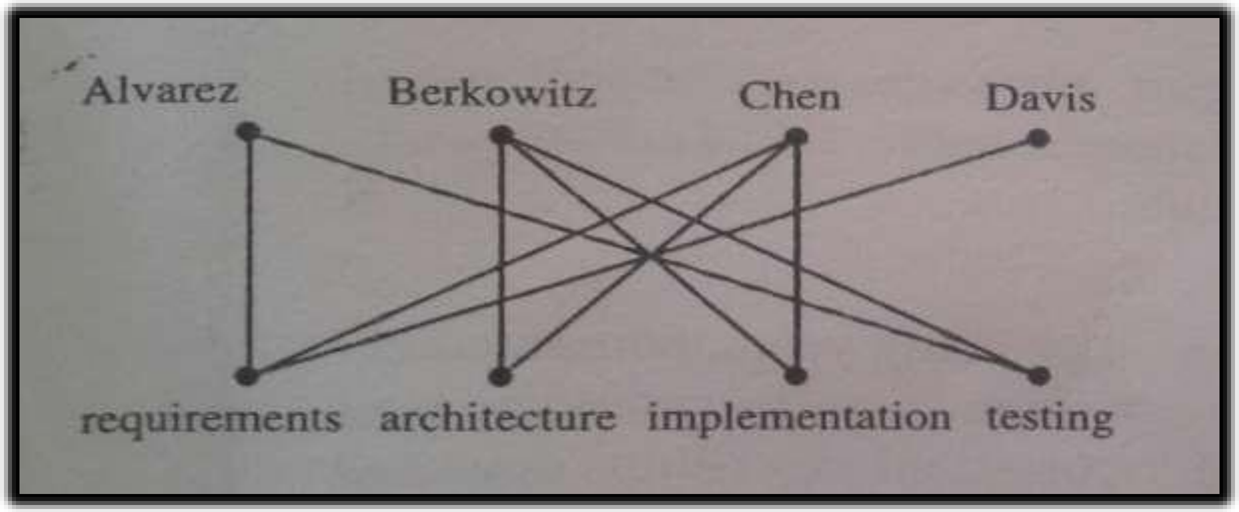
Now suppose that it is possible to assign colors to the vertices of the graph using just two colors so that no two adjacent vertices are assigned the same color. Let V_1 be the set of vertices assigned one color and V_2 be the set of vertices assigned the other color. Then, V_1 and V_2 are disjoint and $V = V_1 U V_2$. Furthermore, every edge connects a vertex in V_1 and a vertex in V_2 because no two adjacent vertices are either both in V_1 or both in V_2.

Consequently, G is bipartite.

Applications of Special Types of Graphs

1. **Job Assignments :**

Suppose that there are ***m*** employees in a group and ***j*** different jobs that need to be done where ***m***<=***j***. Each employee is trained to do one or more of these ***j*** jobs. We can use a graph to model employee capabilities. We represent each employee by a vertex and each job by a vertex. For each employee, we include an edge from the vertex representing that employee to the vertices representing all jobs the employee has been trained to do.



2. Local Area Network :

Various computers in a building, such as minicomputers and personal computers, as well as peripheral devices such as printers and plotters, can be where all devices are connected to a central control device. A local network can be represented using a complete bipartite graph $\boldsymbol{K_{1,n}}$. Messages are sent from device to device through the central control device.

Other local area networks are based on a ring topology, where each device is connected to exactly two others. Local area networks with a ring topology are modeled using *n*-cycles, $\boldsymbol{C_n}$. Messages are sent from device to device around the cycle until the intended recipient of a message is reached.