Graph Theory

2nd year computer science L2

Pr. Cherif Foudil
Computer Science Department
Biskra University

2025

Course program

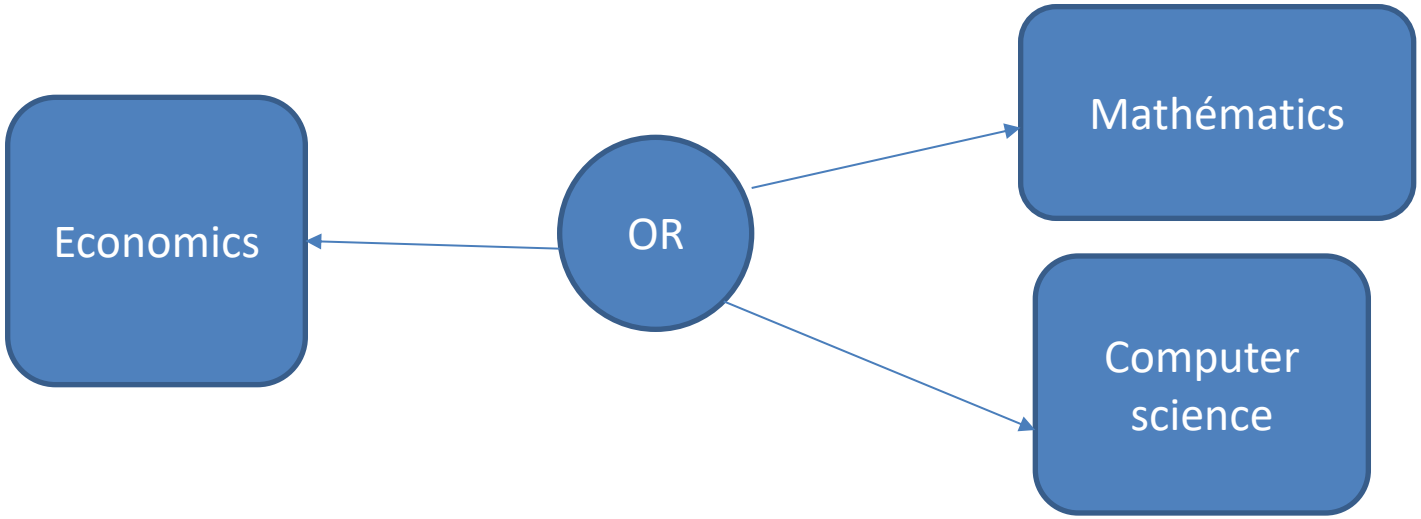
- Basic definitions
- Connectivity in a graph
- Notable paths
  - Hamiltonian
  - and Eulerian
- Trees and arborescences
- Pathfinding
- Maximum flow problem

Chapter 1

Basic definitions

1.1 Motivation

- **Graph theory** (GT) falls within the field of **operations research** (OR), which serves as a crossroads where **economics**, **mathematics**, and **computer science** intersect.

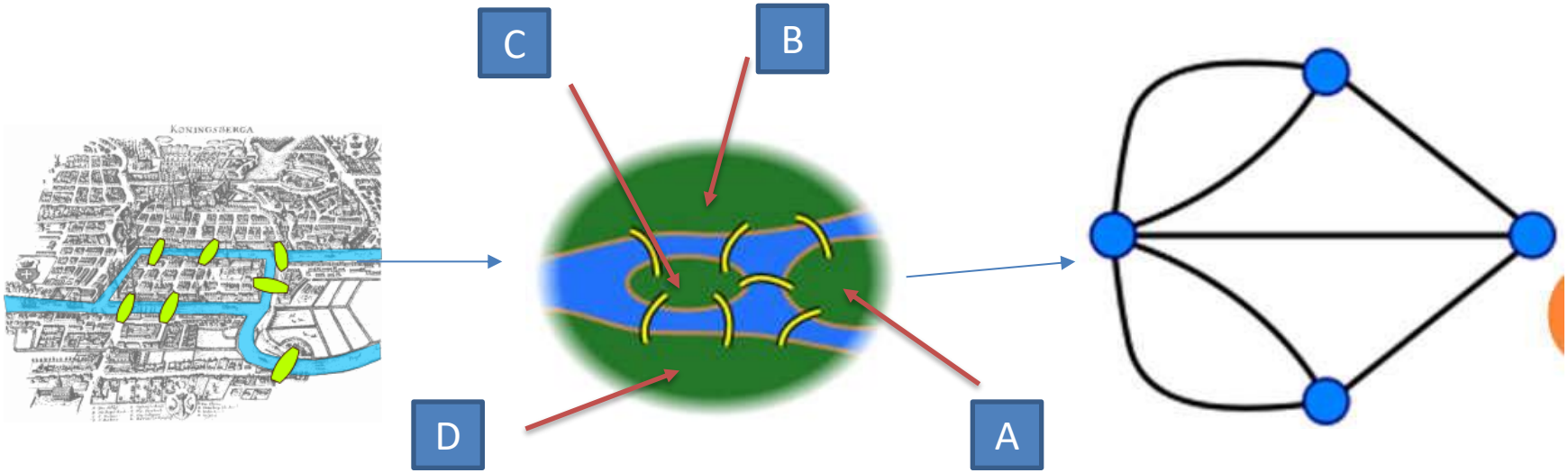


- The objective of **OR** is to find **optimal solutions** for economic problems using mathematical methods that can be programmed by computer.

1.2 History

Question:

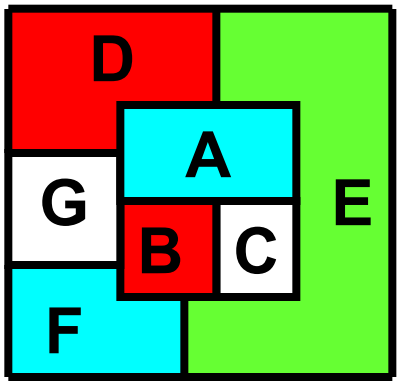
- Is it possible to walk through the city of Königsberg in such a way as to cross each bridge exactly one time and return to the starting district?

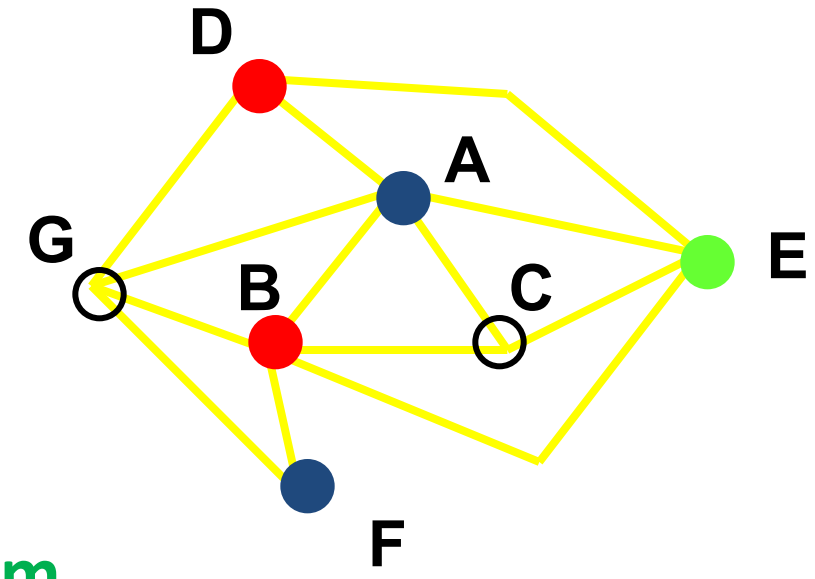


- The German mathematician **L. Euler** (1736) provided an answer to the problem faced by the residents of the city of Königsberg: how to cross the seven bridges of this city without ever crossing the same one twice. → **This marked the birth of graph theory**.

1.2 History

- In 1852, graph theory gained popularity thanks to the "**Four-Color Theorem.**"
- It was demonstrated that only four different colors are needed to color any geographical map in such a way that two adjacent regions (sharing a common boundary) always receive distinct colors.





- **The four-color problem...**

1.2 History

- Starting in 1946, GT (Graph Theory) experienced significant development thanks to researchers motivated by solving practical problems. Among them:
- **Edsger Dijkstra (1959**) for the pathfinding problem,
- **Ford and Fulkerson (1956)** for the maximum flow problem,
- **Bernard Roy (1958)** developed the MPM method for the scheduling problem.

1.2 History

- Graph theory has become essential in our daily lives for various networks:
- **Transportation networks**: road, air, rail, maritime, water, gas, electricity...
- **Data communication networks**: landline, mobile, WIFI...
- **Information networks**: databases, the web, social networks...

1.3 Graph concepts

1.3.1 definitions

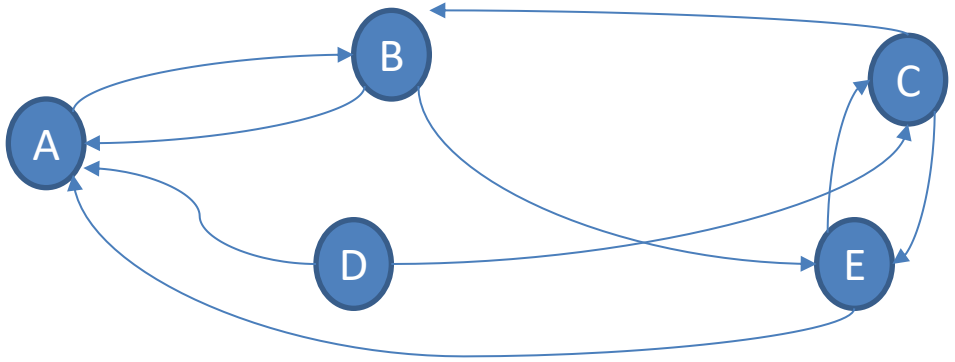
-A graph G, denoted as G=(V, E), is defined as:

- A finite set of vertices V = { v1,v2,…vn }
- A finite set of edges (or arcs) E, where an edge (or arc) connects a pair of vertices from V, E = { e_1,e_2,…e_m}

Each edge has two endpoints

It is not easy to visualize a given graph in this form → drawing it is more helpful.

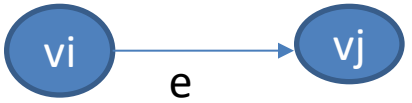
Example: A one-way traffic plan of a city, where each locality (vertex) and each road are represented by a directed arc indicating the direction of traffic.



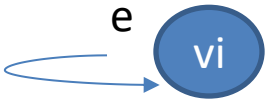
1.3 Graph concepts

1.3.2 Directed graph

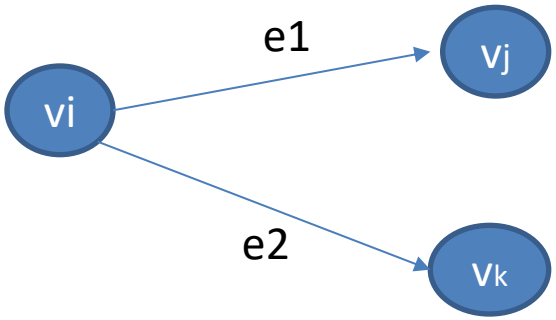
A directed graph is a graph in which if e = (v_i, v_j), v_i is the origin of the arc e, and v_j is the destination of e.



A loop, denoted as b = (v_i, v_i), is an arc where the origin coincides with the destination.



Two arcs are adjacent if they have at least one common endpoint.

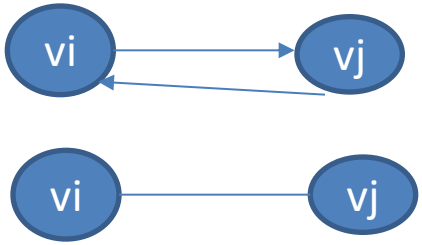


e_1 and e_2 are adjacent; they have v_i in common

1.3 Graph concepts

1.3.2 Undirected graph

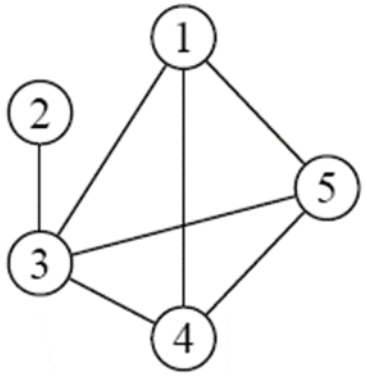
An undirected graph is a graph in which for all v_i, $v_j \in V$, if $(v_i, v_j) \in E$, then $(v_j, v_i) \in E$.



Example: G= (V, E),

V= (1, 2, 3, 4, 5) E= {(1, 5), (1, 4), (1, 3), (2, 3), (3, 4) ; (4, 5), (3, 5)}.

The graph G is represented in a schematic way as follows:



1.3 Graph concepts

1.3.2 Undirected graph

An edge e in E is defined by an unordered pair (v_1, v_2) of vertices called the endpoints of e.

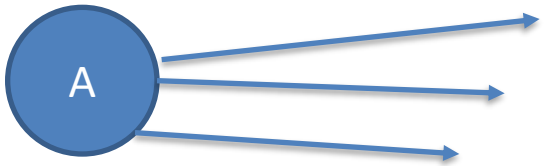
If the edge e connects vertices v_1 and v_2, we can say that these vertices are adjacent or incident to e, or that the edge e is incident to vertices v1 and v2.

The order of a graph is the number of vertices **n** in that graph.
$|V| = n$

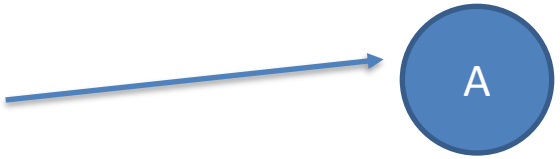
1.3 Graph concepts

1.3.3 The degree of a vertex:

- The vertex A is the initial endpoint of 3 arcs; we say that the **positive degree** of A is equal to 3, denoted by: $\mathbf{d^+_G\,(A) = 3}$



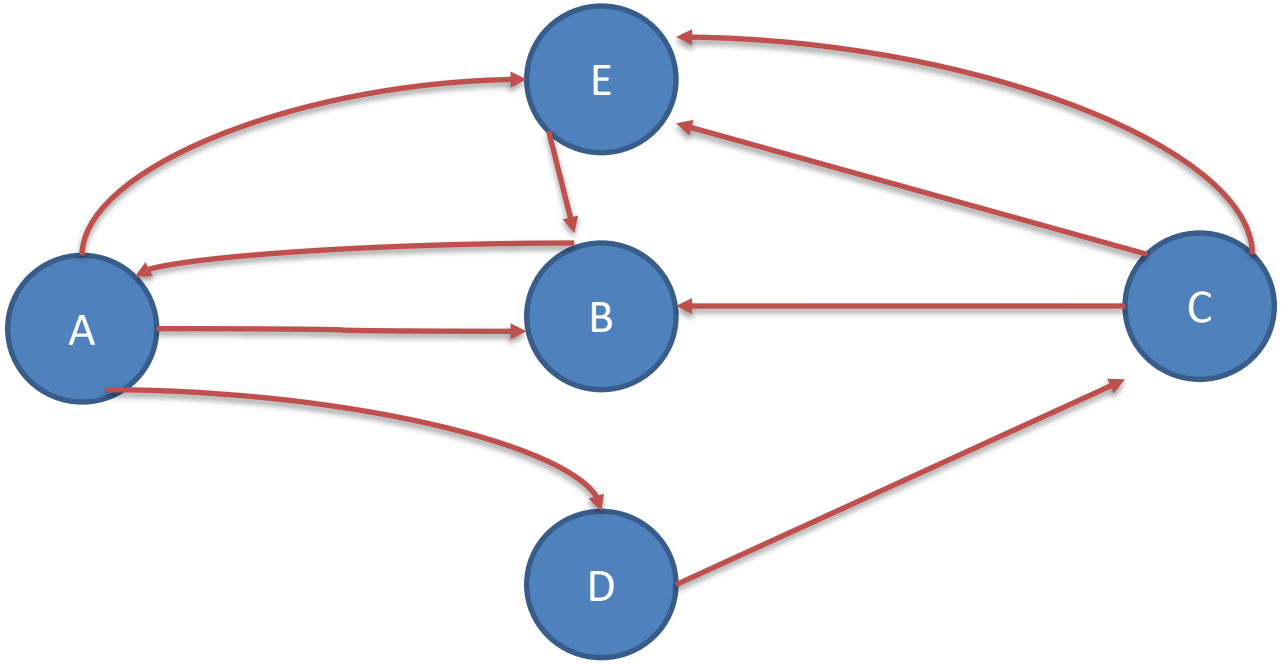
- The vertex A is the terminal endpoint of one arc; we say that **the negative degree** of A is equal to 1, denoted by: $\mathbf{d^-_G\,(A) = 1}$



- The degree $\mathbf{d_G\,(A) = d^+_G\,(A) + d^-_G\,(A)}$

1.3 Graph concepts

1.3.3 The degree of a vertex : Example



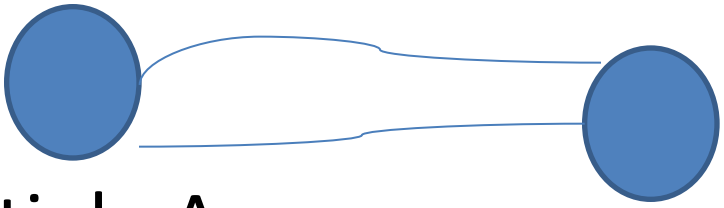
	A	B	C	D	E	
$d^+_G(x)$	3	1	3	1	1	9
$d^-_G(x)$	1	3	1	1	3	9
$d_G(x)$	4	4	4	2	4	18

1.3 Graph concepts

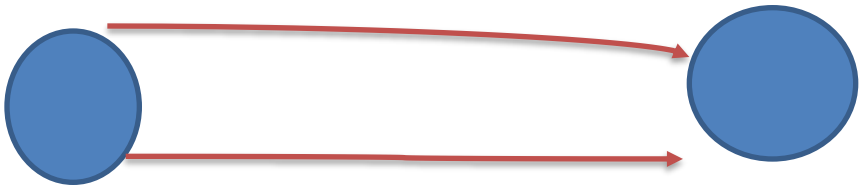
1.3.4 Types of graphs

a) Multiple Graph: G = (V, E) is a graph in which the set E of edges may contain more than one edge connecting two given vertices.

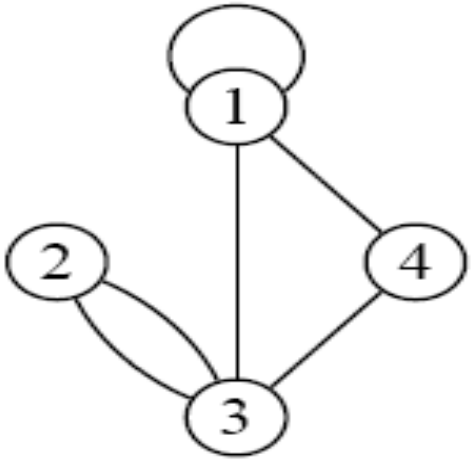
Multiple Edges



Multiple Arcs



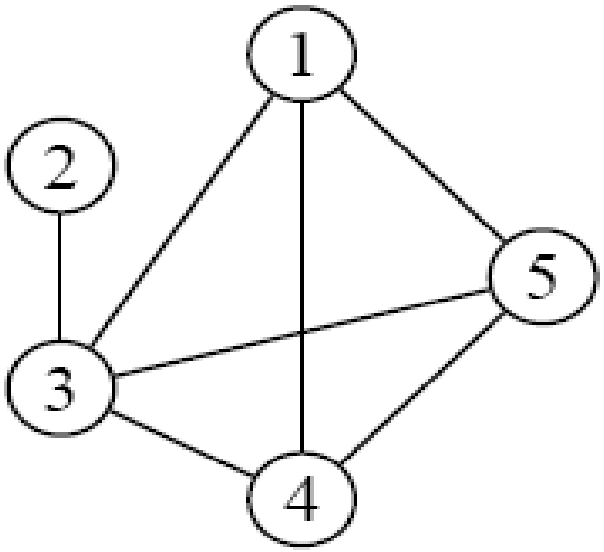
multiple graph



1.3 Graph concepts

1.3.4 Types of graphs

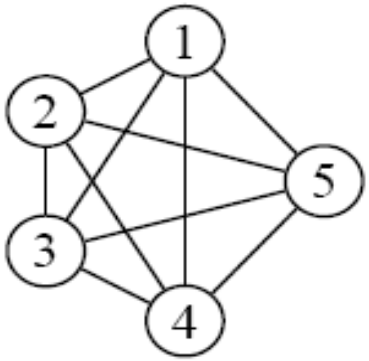
b) Simple Graph: A graph without loops or multiple edges (arcs).

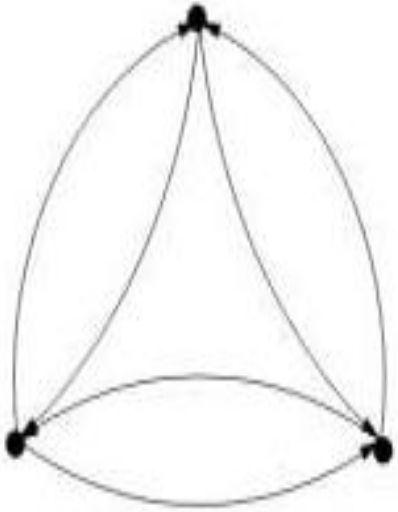


1.3 Graph concepts

1.3.4 Types of graphs

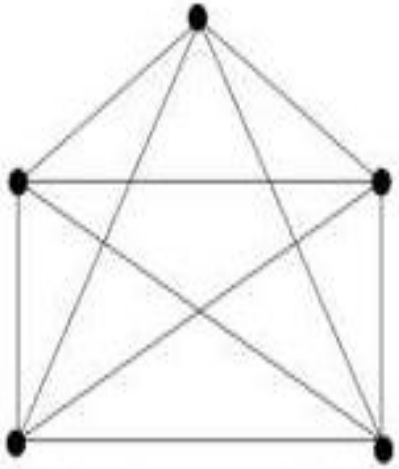
c) **Complete Graph**: A graph in which every vertex is directly connected to every other vertex.





Graphe complet (orienté) sur trois sommets

Graphe complet (non-orienté) sur trois sommets



Graphe complet (non-orienté) sur cinq sommets

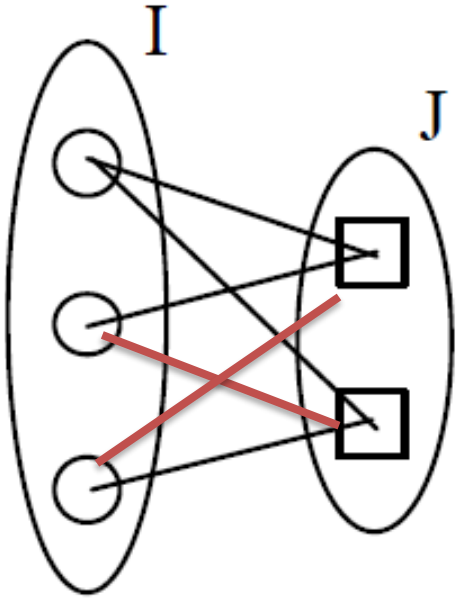
1.3 Graph concepts

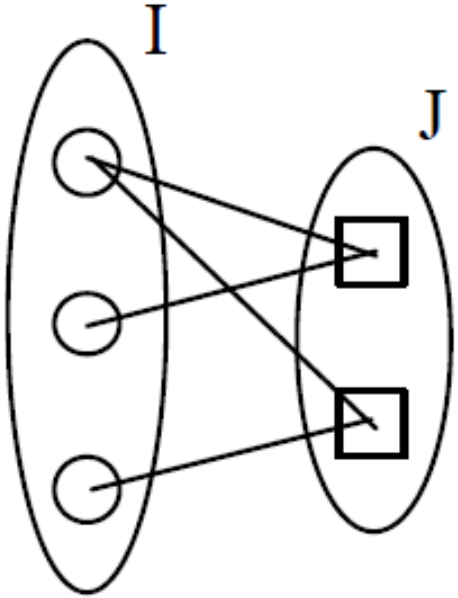
1.3.4 Types of graphs

d) Bipartite Graph: If its set of vertices can be divided into two distinct subsets I and J such that each edge has one endpoint in I and the other in J.

e) Complete Bipartite Graph :

If every vertex in set I is connected to every vertex in set J.

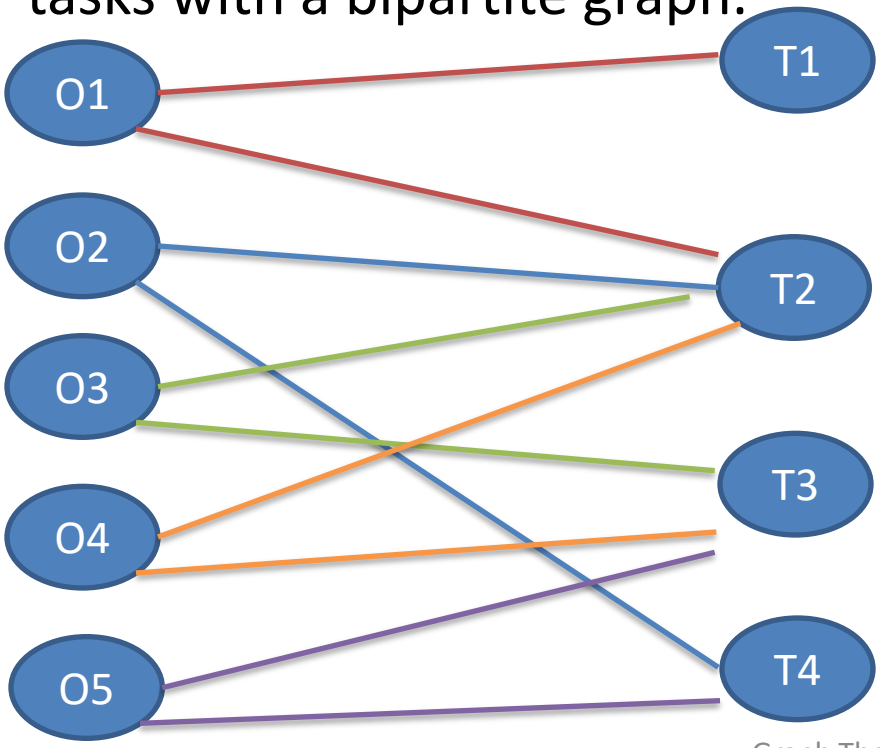




1.3 Graph concepts

1.3.4 Types of graphs

d) Example of a Bipartite Graph: In a workshop with 5 workers, each capable of performing 1 to 4 tasks, we represent the possibilities of assigning workers to different tasks with a bipartite graph.



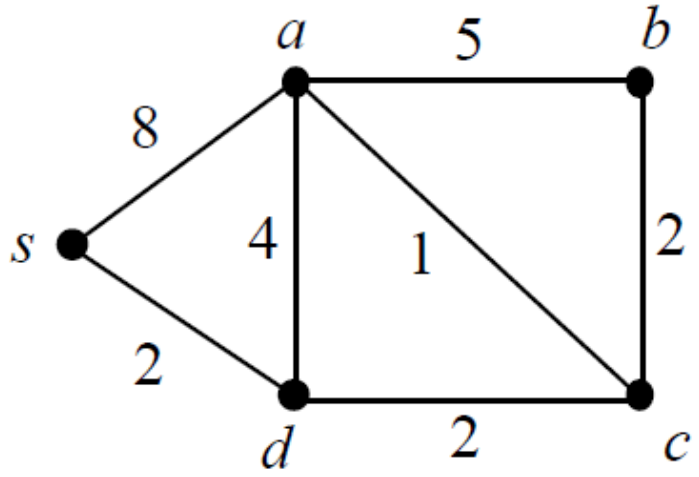
Note: If each worker can perform all tasks, a **complete bipartite graph** is obtained.

1.3 Graph concepts

1.3.4 Types of graphs

f) Weighted Graph:

When the edges represent a cost, they are assigned a number, creating a **weighted graph**.

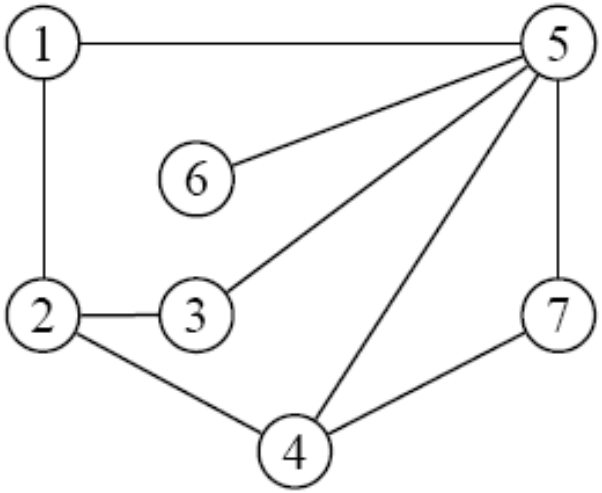


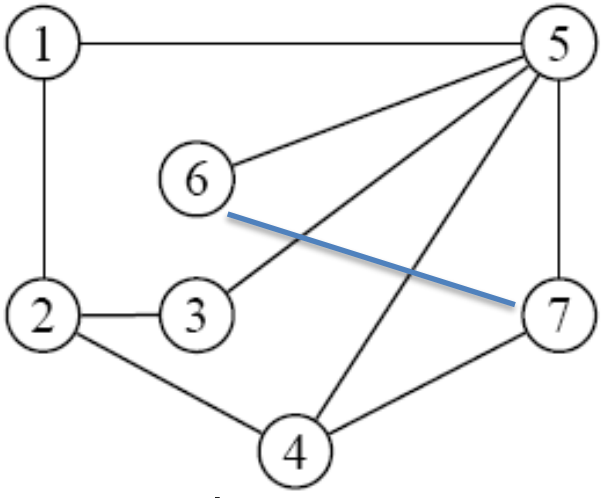
1.3 Graph concepts

1.3.4 Types of graphs

g) Planar Graph:

If its edges do not intersect, in other words, if it can be drawn in a plane in such a way that its edges do not cross.

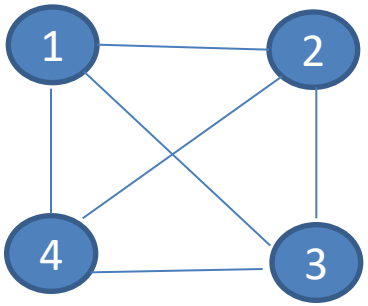


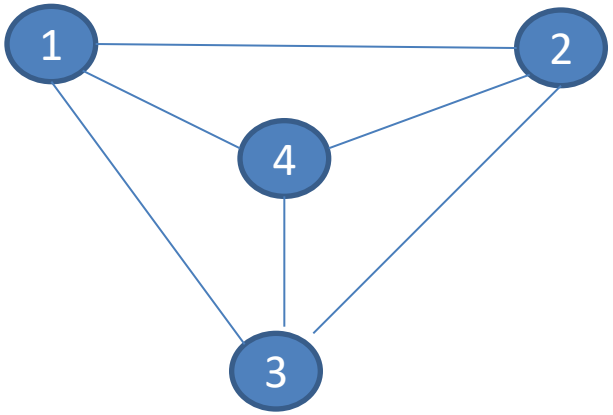


1.3 Graph concepts

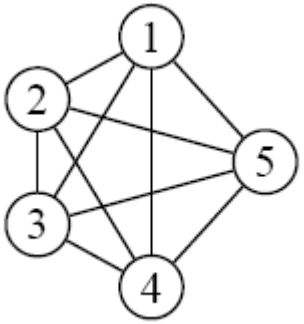
1.3.4 Types of graphs

Example 1: Complete graph with 4 vertices, it is planar if it can be transformed.



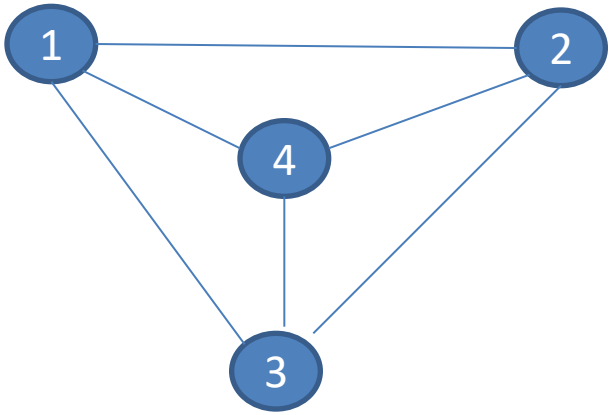


Example 2: Complete graph with 5 vertices, it is not planar, it cannot be transformed.



1.3 Graph concepts

1.3.4 Types of graphs G=(V,E)



Note: Planar graphs satisfy the formula(Euler formula):

$|V| + F = |E| + 2$ **4 + 4 = 6 + 2**

$|V|$ number of vertices

$|E|$ number of edges

F number of faces or regions

1.3 Graph concepts

1.3.4 Types of graphs

h) Graph - Isomorphism

G1 = (V1, E2) and G2 = (V2, E2) are isomorphic if:

There is a one-to-one function f from V1 to V2 with the property that a and b are adjacent in G1 if and only if f (a) and f (b) are adjacent in G2, for all a and b in V1.

Function f is called **isomorphism**

G1 = (V1, E1) , G2 = (V2, E2)

f(u1) = v1, f(u2) = v4, f(u3) = v3, f(u4) = v2,

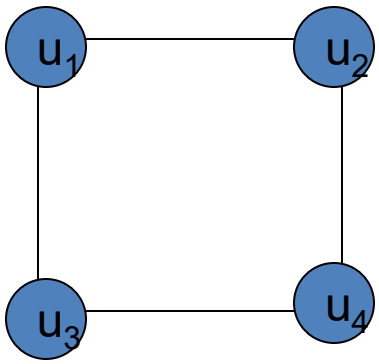
1.3 Graph concepts

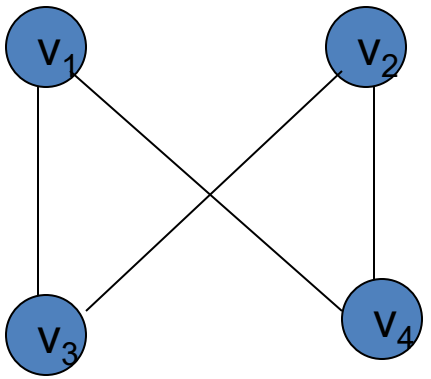
1.3.4 Types of graphs

i)Graph – Isomorphism example 1

Representation example: G1 = (V1, E1) , G2 = (V2, E2)

$f(u_1) = v_1$, $f(u_2) = v_4$, $f(u_3) = v_3$, $f(u_4) = v_2$,

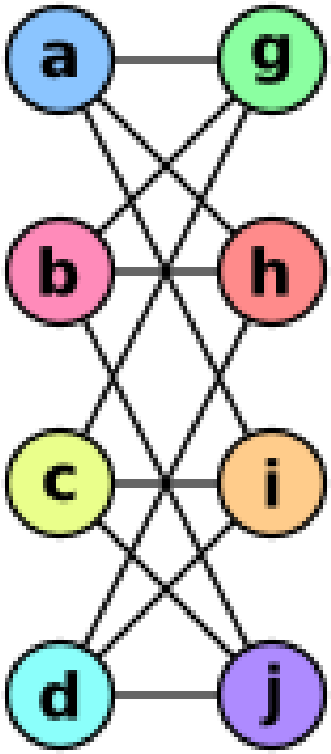


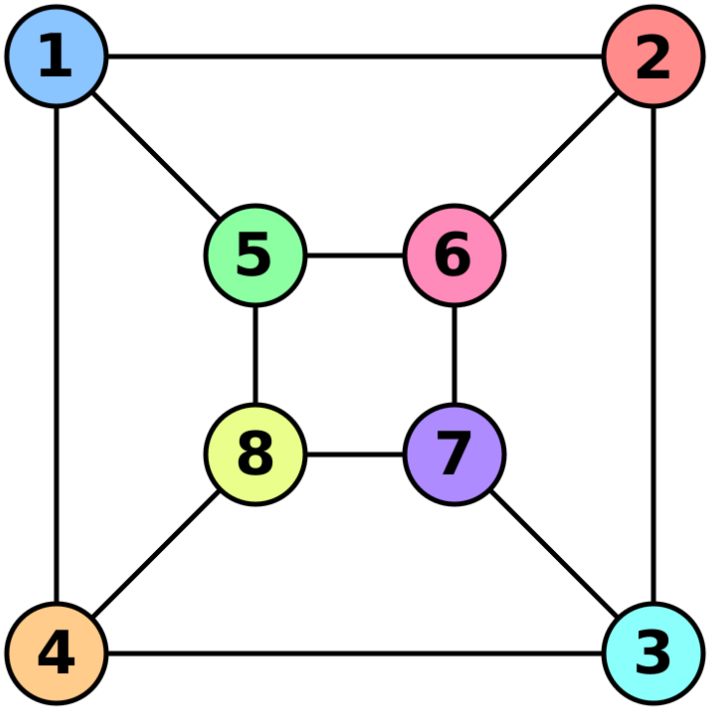


1.3 Graph concepts

i)Graph – Isomorphism (example 2)

f(a) = 1
f(b) = 6
f(c) = 8
f(d) = 3
f(g) = 5
f(h) = 2
f(i) = 4
f(j) = 7





1.3 Graph concepts

1.3.4 Types of graphs

j) Other graphs:

- A **reflexive** graph is a graph having a loop on each vertex
- A graph G = (V, E) is **symmetric** if,

  $\forall$ arc $e_1 = (v_i, v_j) \in E$, the arc $e_2 = (v_j, v_i) \in E$.

- A graph G = (V, E) is **antisymmetric** if,

  $\forall$ (For every) arc $e_1 = (v_i, v_j) \in E$, the arc $e_2 = (v_j, v_i) \notin E$.

- A graph G = (V, E) is **transitive** if,

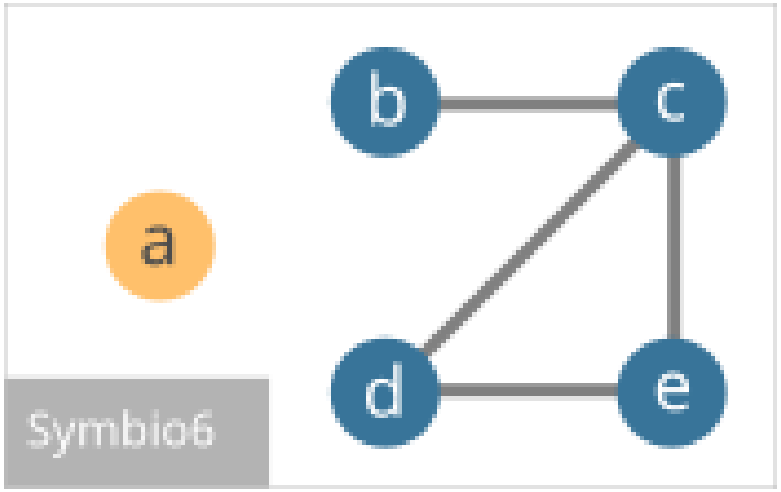
  $\forall$ arc $e_1 = (v_i, v_j) \in E$ and arc $e_2 = (v_j, v_k) \in E$ then the arc $e_3 = (v_i, v_k) \in E$.

1.3 Graph concepts

1.3.4 Types of graphs

k) Isolated vertex:

A vertex of a graph that has no edges is a vertex with degree zero.



Graph with isolated vertex a.

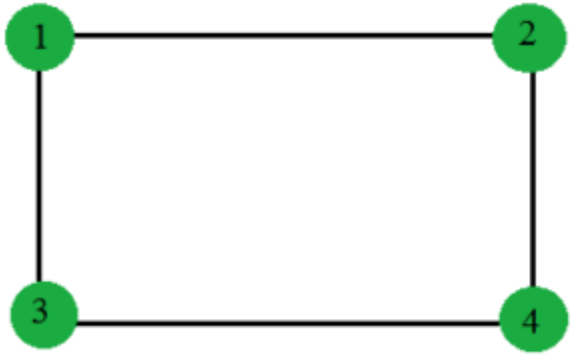
Null Graph: A null graph is defined as a graph which consists only the isolated vertices

1.3 Graph concepts

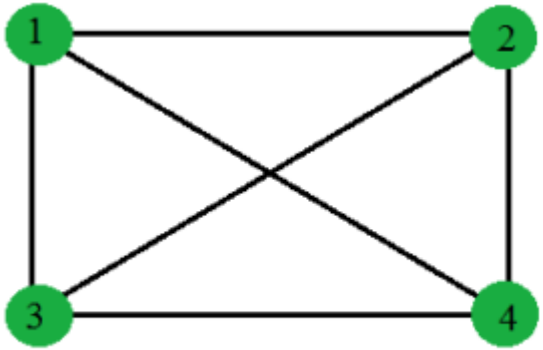
1.3.4 Types of graphs

l) Regular graph:

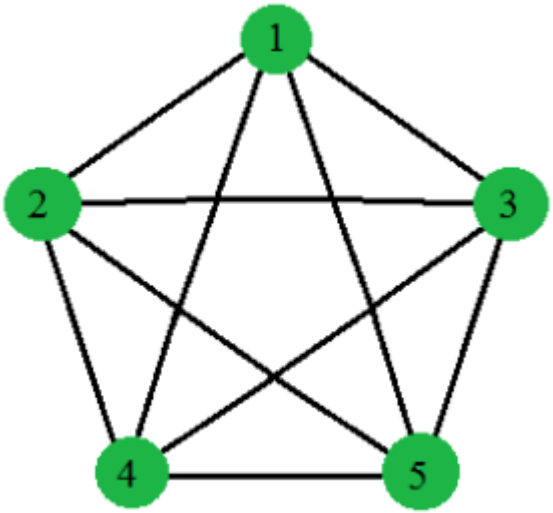
A graph is called regular graph if degree of each vertex is equal. A graph is called **K regular** if degree of each vertex in the graph is K.



2 Regular



3 Regular

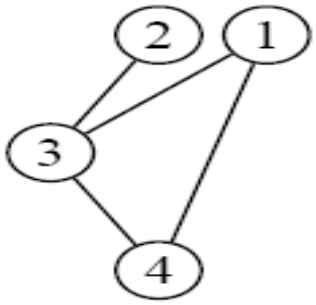


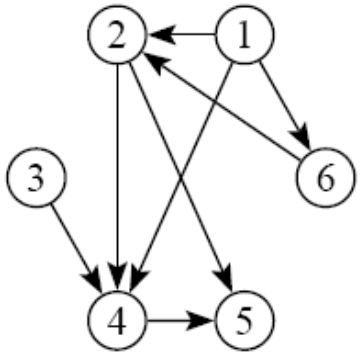
4 Regular

1.4 Representation of a graph

a) Sagittal representation (drawing):

Vertices are represented by circles, and the relationships are represented by lines or arrows,





b) Matrix representation.

For a given graph G=(V,E) with |V|=n and |E|=m (n vertices and m edges), we associate 4 types of matrices:

Adjacency matrix, Incidence matrix, Arc matrix, Associated matrix

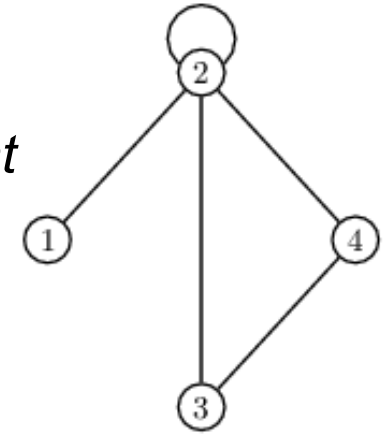
1.4 Representation of a graph

1- Adjacency Matrix: (undirected graph):

Let G be an undirected graph with n vertices numbered from 1 to n. The adjacency matrix of the graph is called matrix A=(ai,aj), where $a_{i,j}$ is the number of edges connecting vertex i to vertex j.

Example: a graph and its corresponding adjacency matrix:

Other definition: A square matrix where each row and column represent a vertex, and the entries indicate whether there is an edge between the vertices. It's often used for ***simple graphs.***



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

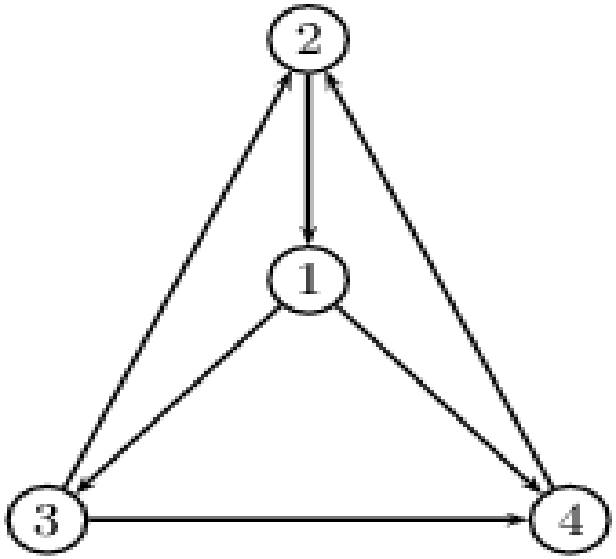
It is a symmetric matrix

1.4 Representation of a graph

Adjacency Matrix: (directed graph)

We can also define the adjacency matrix of a directed graph. This time, the coefficient $a_{i,j}$ represents the number of arcs originating from vertex i and ending at vertex j.

Example : For the following graph:



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

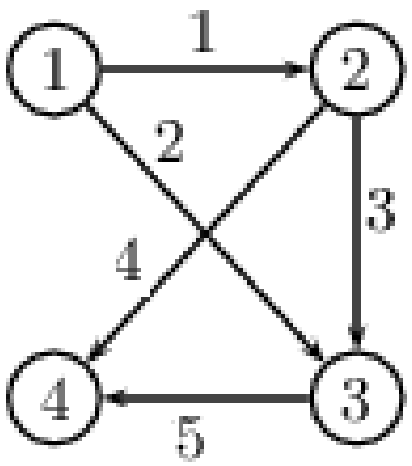
This matrix is no longer symmetric.

1.4 Representation of a graph

2- Incidence matrix: (directed graph)

Let G be a directed graph with n vertices numbered from 1 to n, and m arcs numbered from 1 to m. The **incidence matrix** of the graph is called matrix $A=(a_{i,j})$ consisting of n rows and m columns such that:

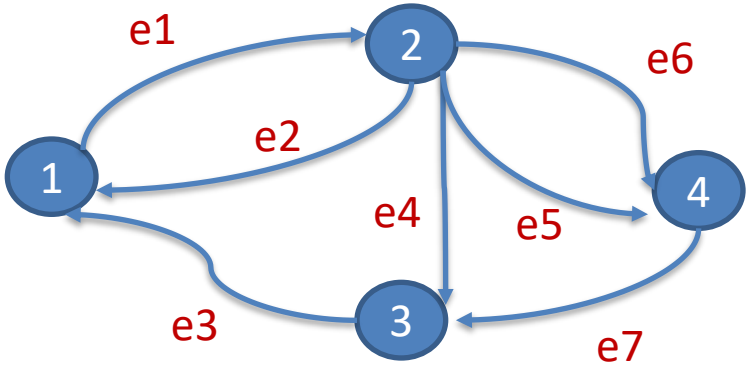
- $a_{i,j}$ equals +1 if the arc numbered j has vertex i as its origin;
- $a_{i,j}$ equals -1 if the arc numbered j has vertex i as its destination;
- $a_{i,j}$ equals 0 in all other cases. **A(vertex, arc)**



$$A = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 \\ -1 & 0 & 1 & 1 & 0 \\ 0 & -1 & -1 & 0 & 1 \\ 0 & 0 & 0 & -1 & -1 \end{pmatrix}.$$

1.4 Representation of a graph

Incidence matrix : Example: G=(V,E) of order 4 with 7 arcs.



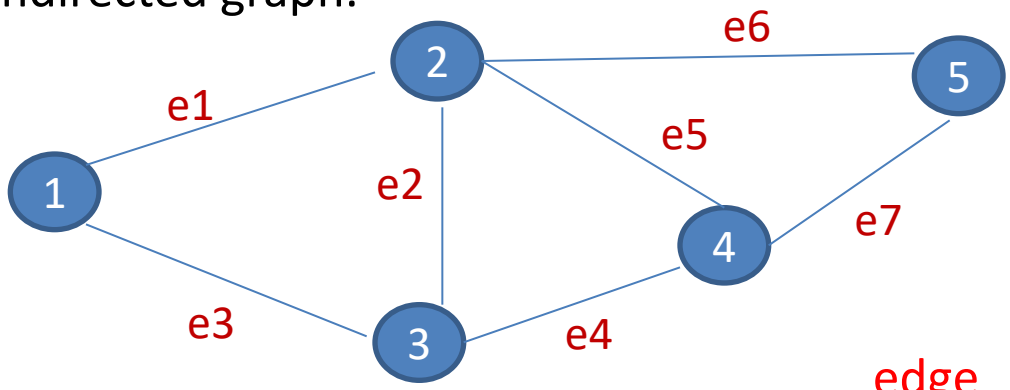
Note: *You should start filling the matrix by beginning with the arcs.*

Arc

Vertex	e1	e2	e3	e4	e5	e6	e7
1	1	-1	-1	0	0	0	0
2	-1	1	0	1	1	1	0
3	0	0	1	-1	0	0	-1
4	0	0	0	0	-1	-1	1

1.4 Representation of a graph

Incidence matrix : One can also define the incidence matrix (vertex/edge) for an undirected graph.



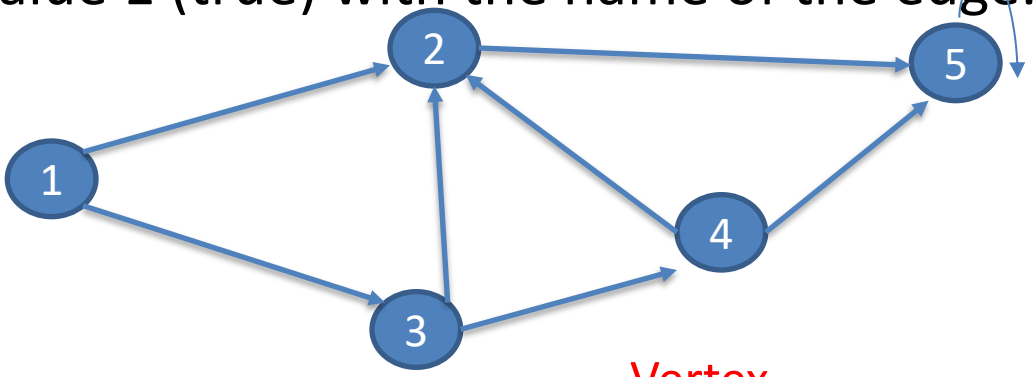
edge

Vertex

	e1	e2	e3	e4	e5	e6	e7
1	1	0	1	0	0	0	0
2	1	1	0	0	1	1	0
3	0	1	1	1	0	0	0
4	0	0	0	1	1	0	1
5	0	0	0	0	0	1	1

1.4 Representation of a graph

3- Edge Matrix: It is defined from the adjacency matrix by replacing the value 1 (true) with the name of the edge.



Vertex

Vertex

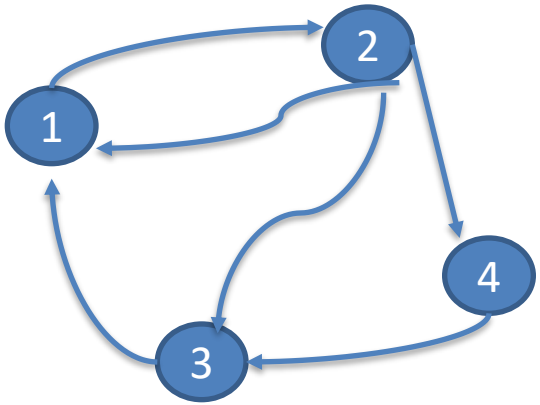
	1	**2**	**3**	**4**	**5**
1	0	12	13	0	0
2	0	0	0	0	25
3	0	32	0	34	0
4	0	42	0	0	45
5	0	0	0	0	55

1.4 Representation of a graph

4- Associated Matrix: We replace the name with the number of edges or arcs**.**

C) Representation by dictionaries:

The graph is represented by a table or dictionary of vertices. Each vertex has a list of **successor** (**predecessor**) vertices. We use a function Γ such that $\Gamma^+(x)$ is the list of successors and $\Gamma^-(x)$ is the list of predecessors.



Vertice	Successor $\Gamma^+(x)$	Predecessor $\Gamma^-(x)$
1	2	2,3
2	1,3,4	1
3	1	2,4
4	3	2

Chapter 2

Connectivity in a graph

2.1 Paths in a graph

Definitions: Let G=(V,E) be any graph, C={e1, e2, …, ep} a sequence of arcs with cardinality p such that two consecutive arcs ei and ei+1 share a common endpoint.

C is called a **chain** (walk)of cardinality p.

A **path** is a chain(walk) where the arcs are oriented in the same direction.

A **cycle** is a closed chain (the first vertex coincides with the last vertex).

A **circuit** is a closed path (the first vertex coincides with the last vertex).

2.1 Paths in a graph

A path is a **chain** (walk)or a **cycle** or a **route(way)** or a **circuit**.

A **path** is called **elementary** if it visits each of its **vertices exactly once** (except for the first vertex in the case of a cycle and circuit).

A **path** is called **simple** if it traverses each of its **edges exactly once**.

A **path** is called **Hamiltonian** if it visits each and every vertex of the graph exactly once (except for the first vertex in the case of a cycle and circuit).

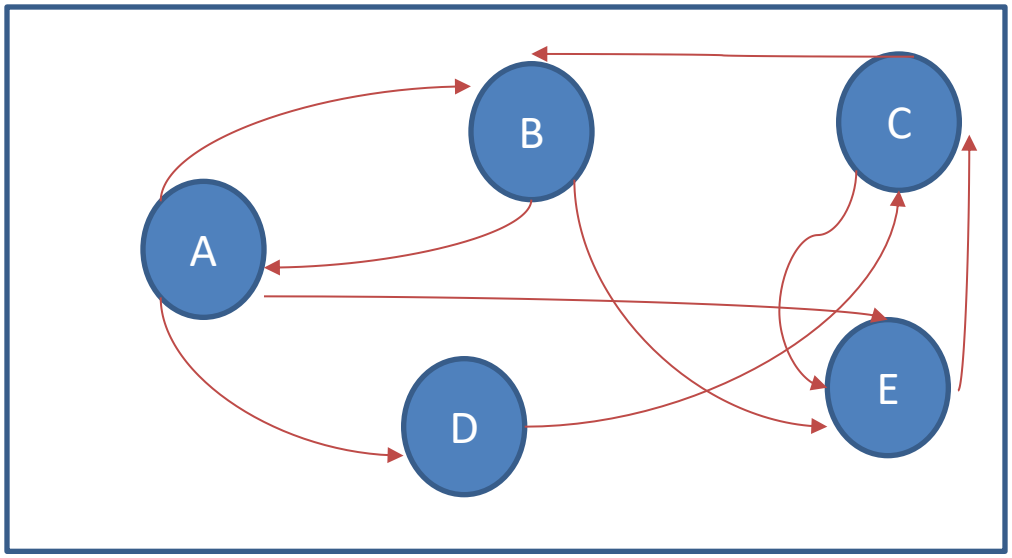
A **path** is called **Eulerian** if it traverses each and **every edge of the graph exactly once.**

2.1 Paths in a graph

2.1.1 Chain:

A chain connecting two vertices V0 and Vk in a graph G is a sequence of vertices connected by edges in such a way that two successive vertices share a common edge. It is denoted as (V0, V1, ..., Vk), and we refer to V0 and Vk as the endpoints of the chain.

Example: G=(V,E)



The sequence (A, B, C, D) is a **simple chain** connecting A to D.

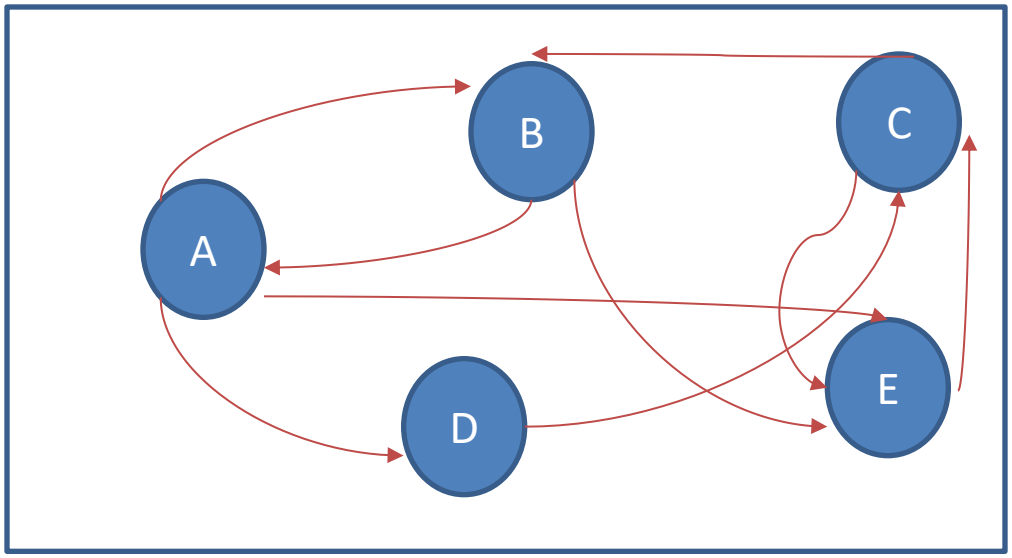
The sequence (A, E, B, A, E) is a path connecting A to E that **is not simple**: the edge (A, E) is traversed twice.

The sequence (A, B, E) is a **simple path** connecting A to E.

2.1 Paths in a graph

2.1.1 Chain:

- The length of the path is equal to the number of edges,
- A path **is elementary** if none of the **vertices** that make up the path appear more than once.
- A path is said to **be simple** if none of the **edges** appear more than once.



We call an **Eulerian path** of a graph G a path that passes **exactly once** through **each** of the edges of G.

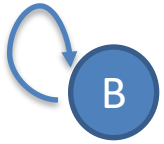
We call a **Hamiltonian path** of a graph G a path that passes **exactly once** through **each** of the vertices of G.

2.1 Paths in a graph

2.1.2 Cycle:

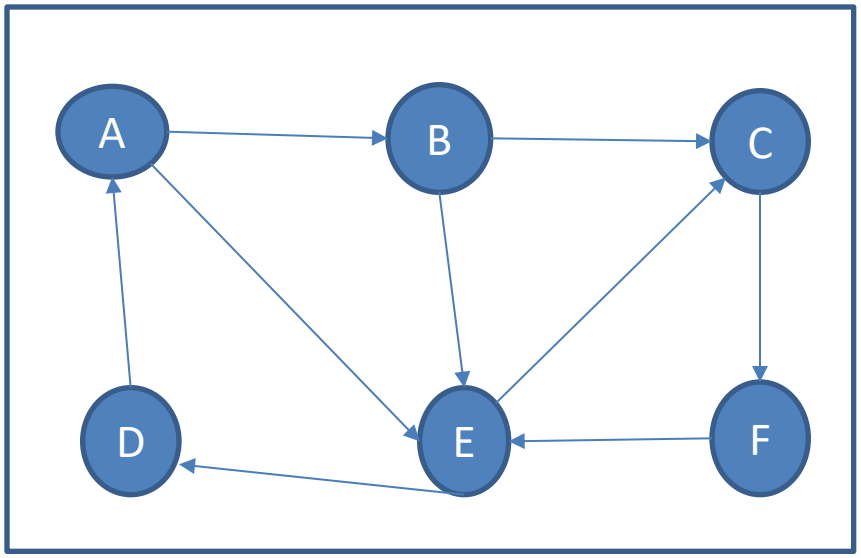
A **cycle** is a path in which the endpoints coincide (e0, e1, ..., ek = e0), and it consists of a succession of **distinct edges**.

Example: (A, B, C, E, A) is a cycle.



A **loop** is a specific type of cycle. (B, B)

A cycle is **elementary** if, while traversing it, one does not encounter the **vertices** multiple times.



C1 = (A, B, E, A) elementary cycle of length 3.

C2 = (A, B, C, F, E, D, A) elementary cycle of length 5.

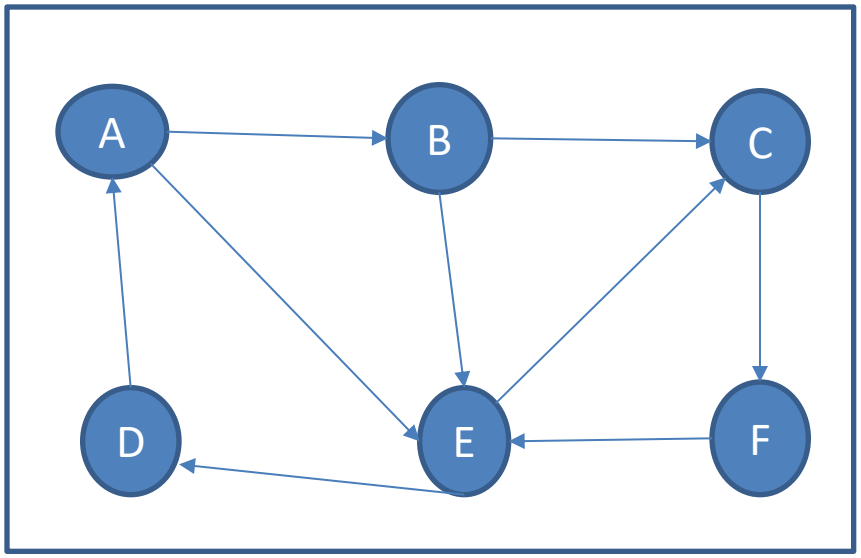
C3 = (D, A, E, F, C, E, D) cycle of length 6 that is not elementary.

C4 = (A, B, E, C, **B, E**, A) is not a cycle: the edge (B, E) is counted twice.

2.1 Paths in a graph

2.1.2 Cycle:

- We call a **Hamiltonian cycle** of a graph a cycle that passes exactly once through each of the vertices of the graph. C1 = (A, B, C, F, E, D, A) is a Hamiltonian cycle.
- A **graph is Hamiltonian** if it has a Hamiltonian cycle.
- We call an **Eulerian cycle** of a graph G a cycle that passes exactly once through each of the edges of G.
- A **graph is said to be Eulerian** if it has an Eulerian cycle.



The **distance** between two vertices in a graph G is the **length** of the **shortest** path connecting them.
Example: Distance between (A, F) = 2.

The **diameter** of a graph G is the **maximum** of the **distances** between its different vertices.
Diameter(G) = 2 ?? To be verified.

A graph without cycles is called **acyclic**.

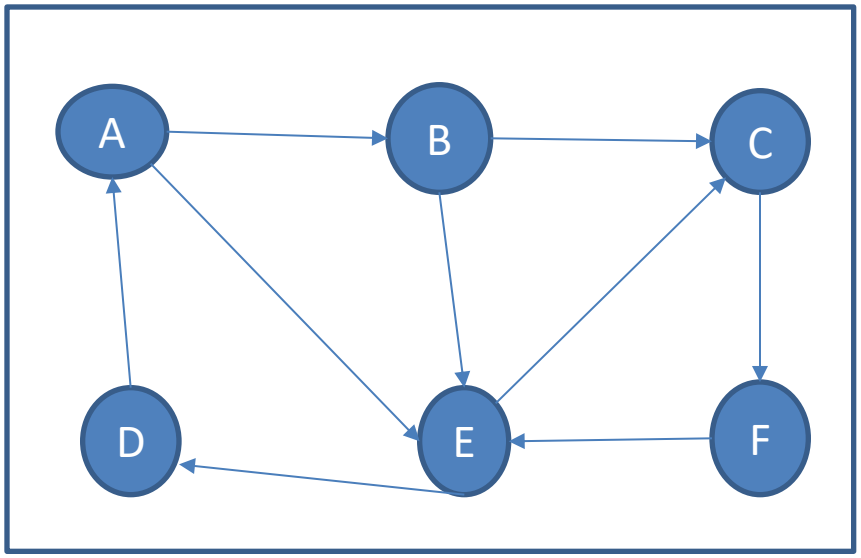
2.1 Paths in a graph

2.1.3 Path:

A path(a way) from vertex V0 to Vk in a graph G is a sequence of vertices connected successively by **directed edges** in the **same direction**.

Example: (A, E, C)

A path is called **elementary** if none of the **vertices** that make up the path appear more than once.



A path is called **simple** if none of the **edges** that make up the path appear more than once.

An **Eulerian path** is a simple path that passes exactly **once** through **each** edge of the graph.

A **Hamiltonian path** is a path that passes exactly **once** through each **vertex** of the graph.

2.1 Paths in a graph

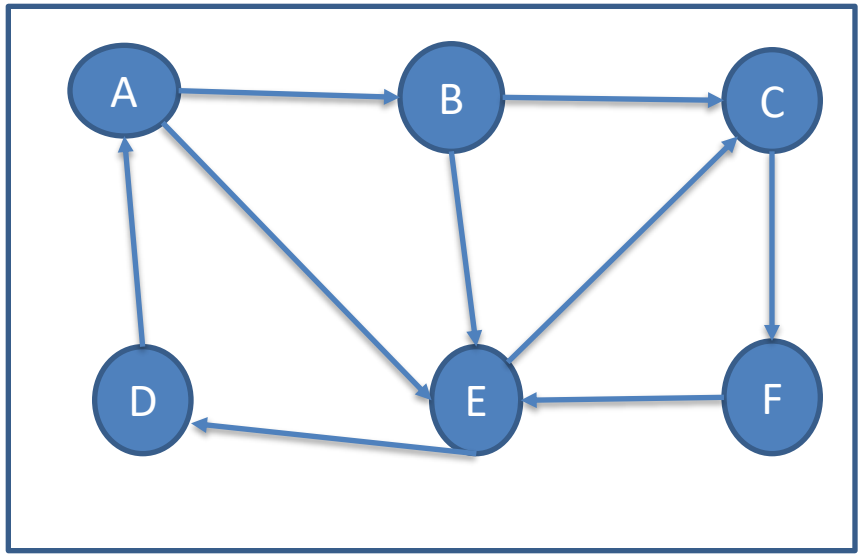
2.1.4 Circuit:

A **circuit** is a path in which the initial and final endpoints coincide.

Examples:

The sequence (A, B, C, F): elementary path.

The sequence (A, B, C, F, E): elementary path.

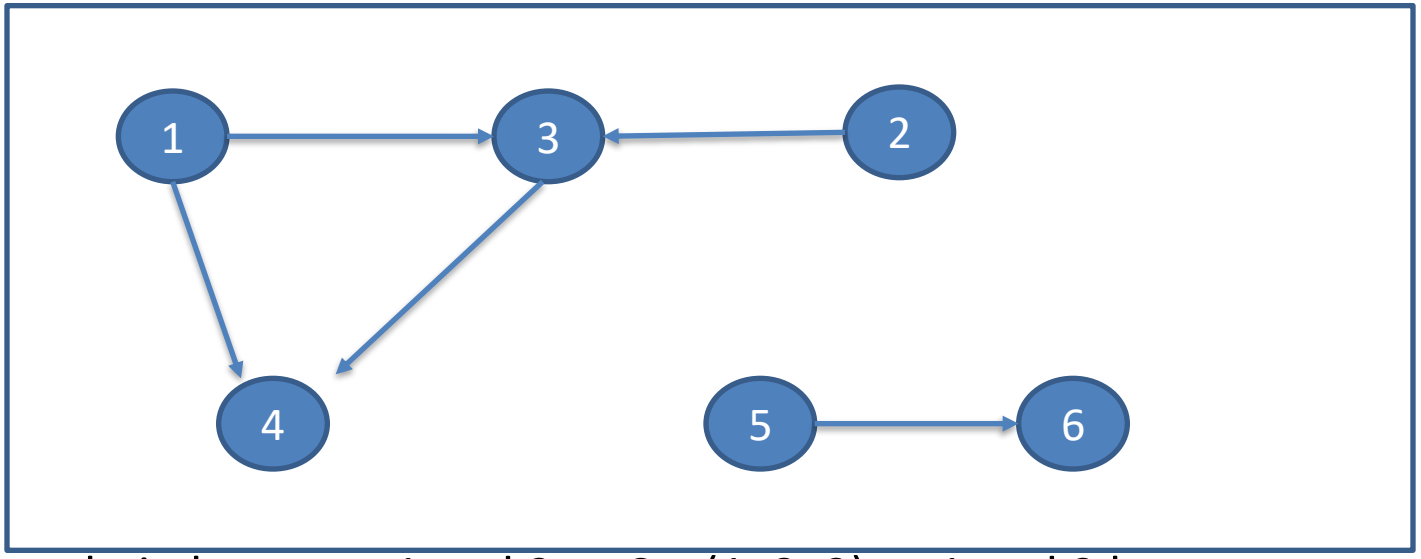


The sequence (A, B, E, A): elementary cycle.
The sequence (A, B, E, D, A): elementary circuit.
The sequence (A, B, C, F, E, D, A): Hamiltonian circuit.

2.2 Connectivity in a graph

2.2.1 The concept of connectivity:

Two vertices x and y have a connectivity relationship if and only if there exists a path(chain) between x and y



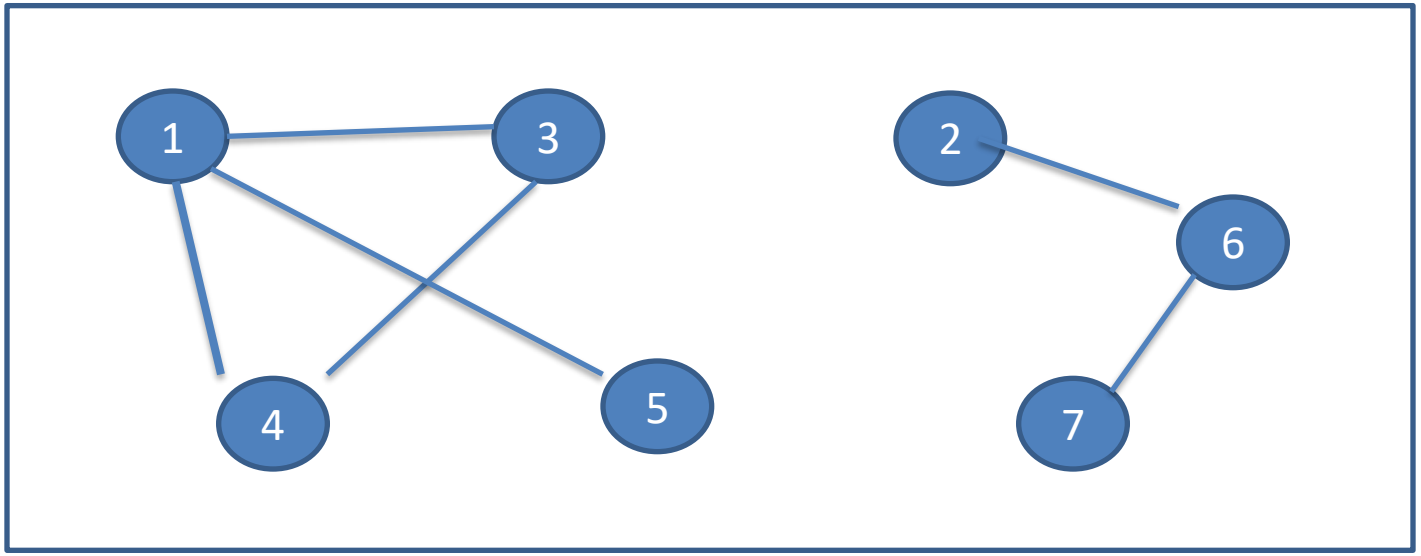
There exists a chain between 1 and 2 ⇒ C = (1, 3, 2) ⇒ 1 and 2 have a connectivity relationship.

There is no relationship between 1 and 5 ⇒ 1 and 5 do not have a connectivity relationship.

2.2 Connectivity in a graph

2.2.2 Connected graph

An undirected graph is connected if every vertex is reachable from any other vertex.



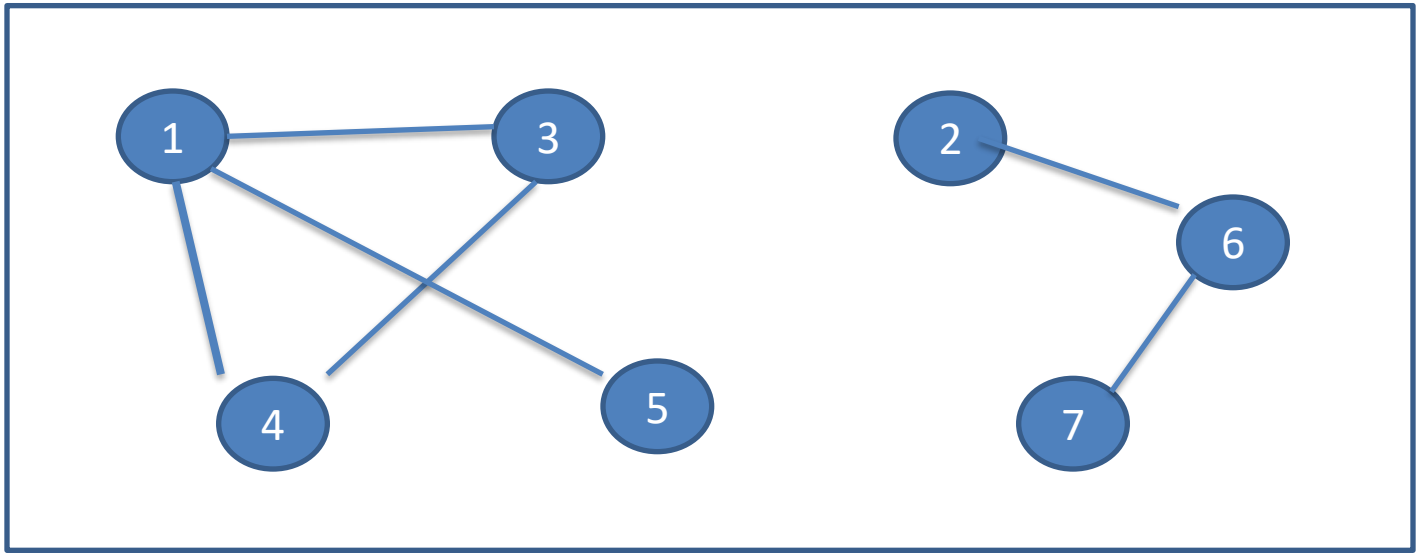
The graph is not connected.

Note: If we add an edge between 5 and 7, the graph becomes connected.

2.2 Connectivity in a graph

2.2.3 Connected component:

We call a **connected component** a set of vertices that are pairwise connected.

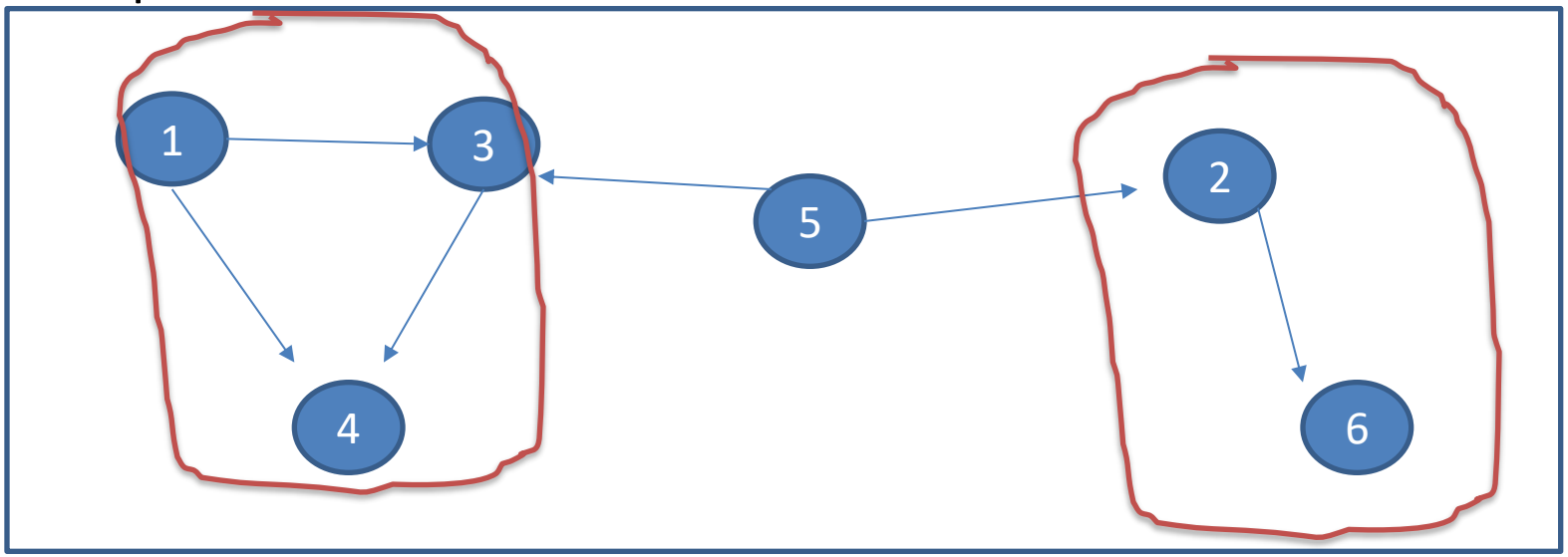


- The vertices (1, 3, 4, 5) form the first connected component.
- The vertices (2, 6, 7) form the second connected component.

2.2 Connectivity in a graph

2.2.4 Articulation point:

An **articulation point** is a vertex whose removal increases the number of connected components.



An **isthmus** is an edge whose removal has the same effect. Vertex 5 is an articulation point. Edge (5,2) is an isthmus.

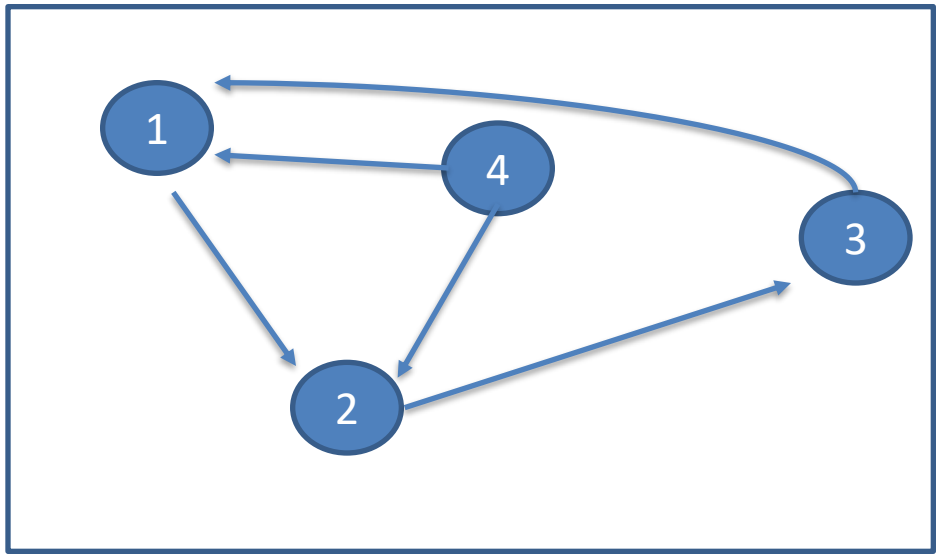
2.2 Connectivity in a graph

2.2.5 Strongly connected graph

We talk about **strong connectivity** in directed graphs.

Strong connectivity in a graph is defined as the relationship between two vertices as follows:

x and y are strongly connected ⇔ there exists a **path** from x to y **and** a **path** from y to x



Vertices 1 and 3 have a strong connectivity relationship.
We have a path from 1 to 3 and a path from 3 to 1.

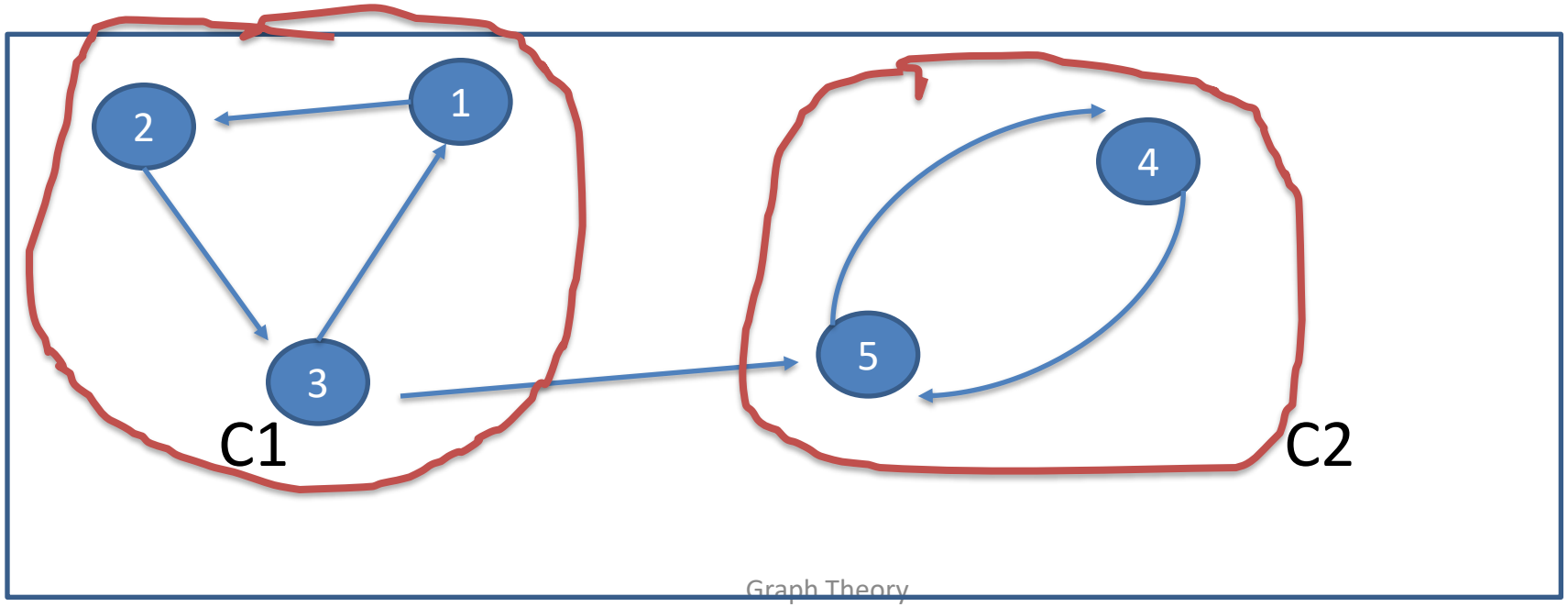
Vertices 4 and 3 do not have a strong connectivity relationship. There is a path from 4 to 3, but there is no path from 3 to 4

2.2 Connectivity in a graph

b) Strongly connected component:

We call a **strongly connected component** a set of vertices that are pairwise strongly connected.

- The set {1, 2, 3} forms a strongly connected component C1.
- The set {4, 5} forms another strongly connected component C2.
- The vertices in C1 do not have a strong connectivity relationship with the vertices in C2.



2.2 Connectivity in a graph

2.2.5 Strongly connected graph

c) Application:

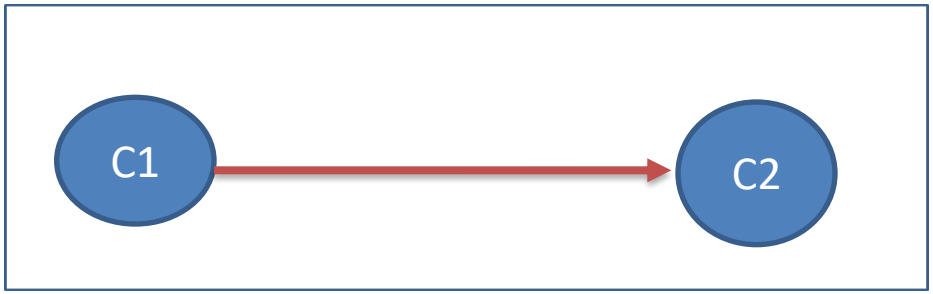
Traffic plan in a city, it is interesting to know if one can move from or to any point. One way to verify is to take the plan and check if it is strongly connected or not.

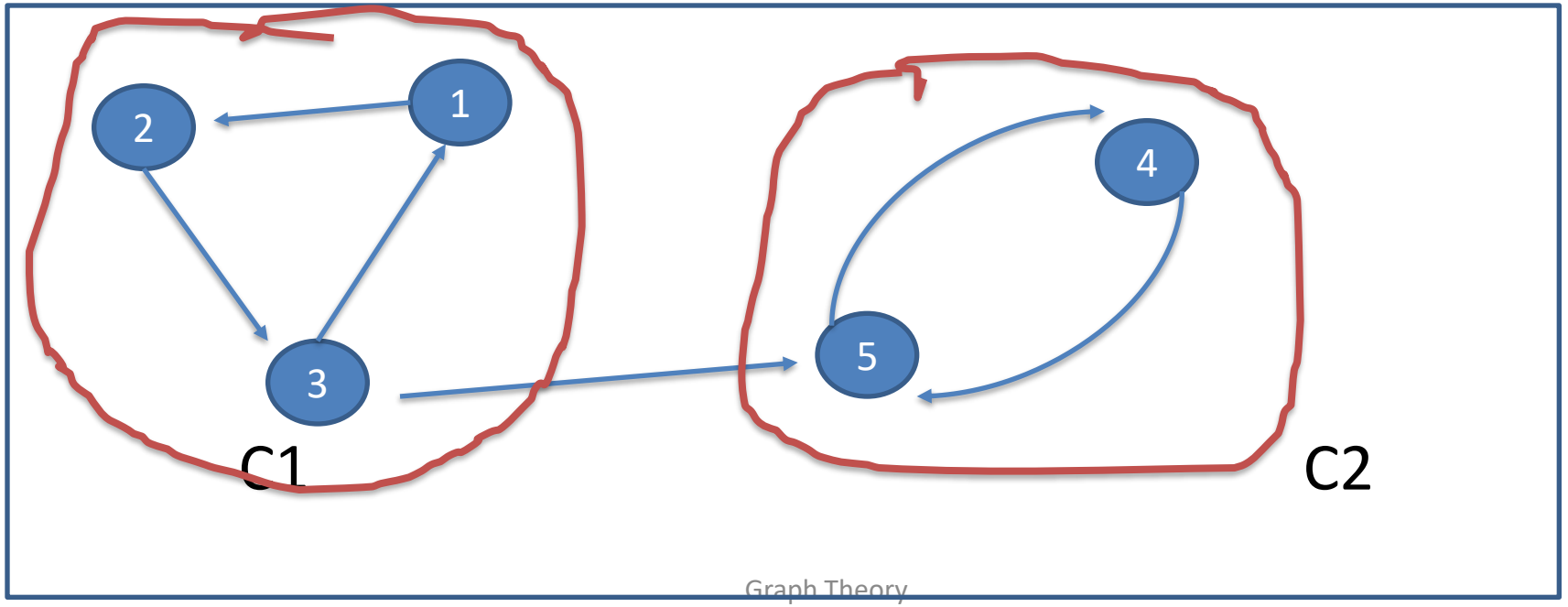
2.2 Connectivity in a graph

2.2.6 Reduced graph:

The vertices are represented by the strongly connected components.

Reduced graphe Gr





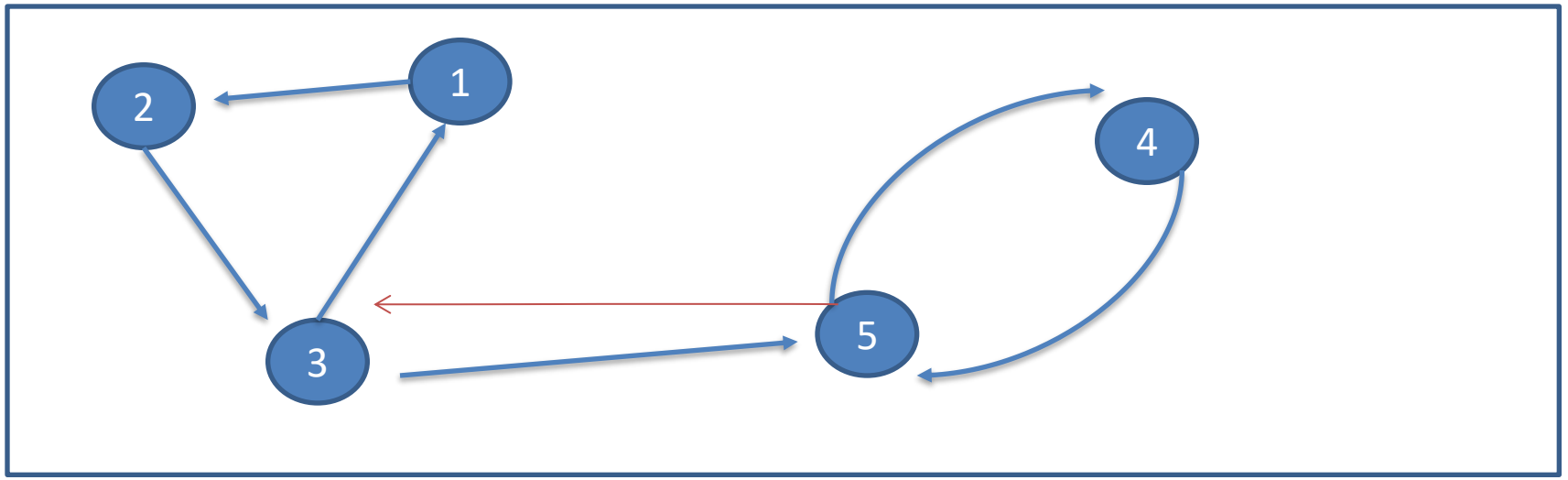
2.2 Connectivity in a graph

2.2.7 Strongly connected graph:

A graph G is said to be strongly connected if all its vertices are pairwise strongly connected, in other words, G contains a single strongly connected component.

This graph is not strongly connected; it contains 2 strongly connected components.

If we add an edge (5,3), it becomes a strongly connected graph



2.2 Connectivity in a graph

2.2.8 Search for strongly connected components:

Algorithm: Marking

Data: G=(V,E), directed graph

Results: the number K of strongly connected components and the list C1, C2, ..., Ck of strongly connected components

Step 0: Initialization K ← 0 W ← V

Step 1:

- Choose a vertex from W and mark it with a + and -
- Mark all direct and indirect successors with +
- Mark all direct and indirect predecessors with –
- Set K = K + 1 and Ck as the set of vertices marked with + and –
- Remove the vertices in Ck from W (W = W - Ck) and clear all marks
- **If** W = Ø **then** finish and proceed to **step 2**, otherwise go to **step 1**

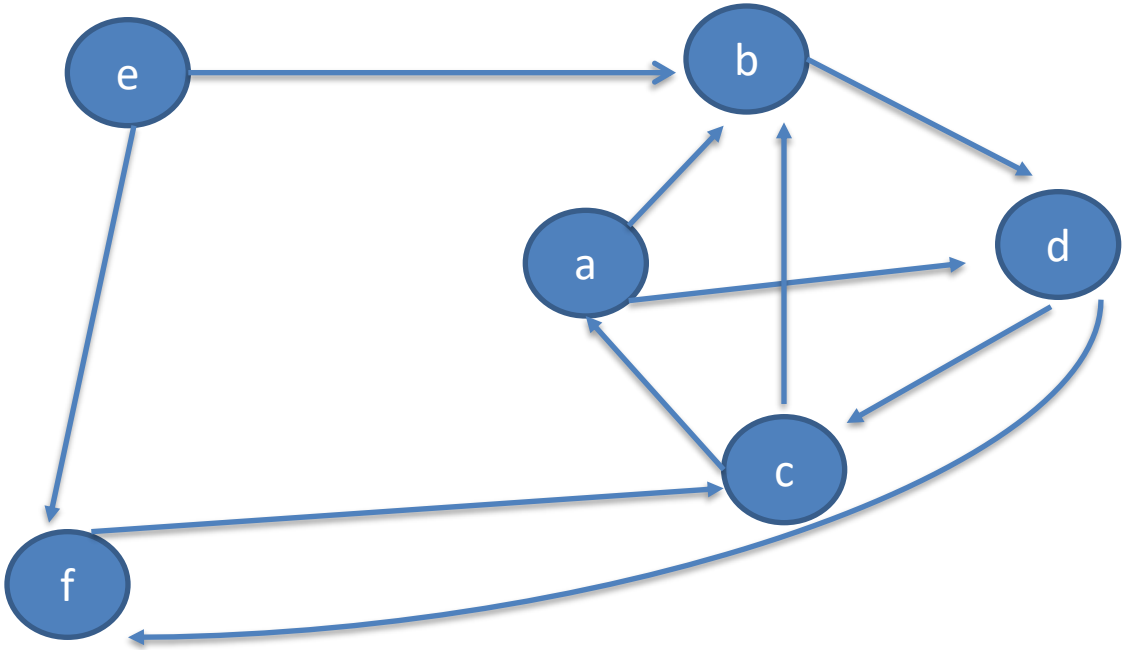
Step 2:

K is the number of strongly connected components, and each Ci is a component.

2.2 Connectivity in a graph

2.2.8 Search for strongly connected components: Example

0 : Initialisation: K = 0 W = V= {a,b,c,d,e,f}



1: We choose 'a', calculate the successors and predecessors of 'a', marked with + and -, we find C1 = {a, b, c, d, f}, and W = W - C1 = {e}.

W ≠ Ø, proceed to step 1 and continue.

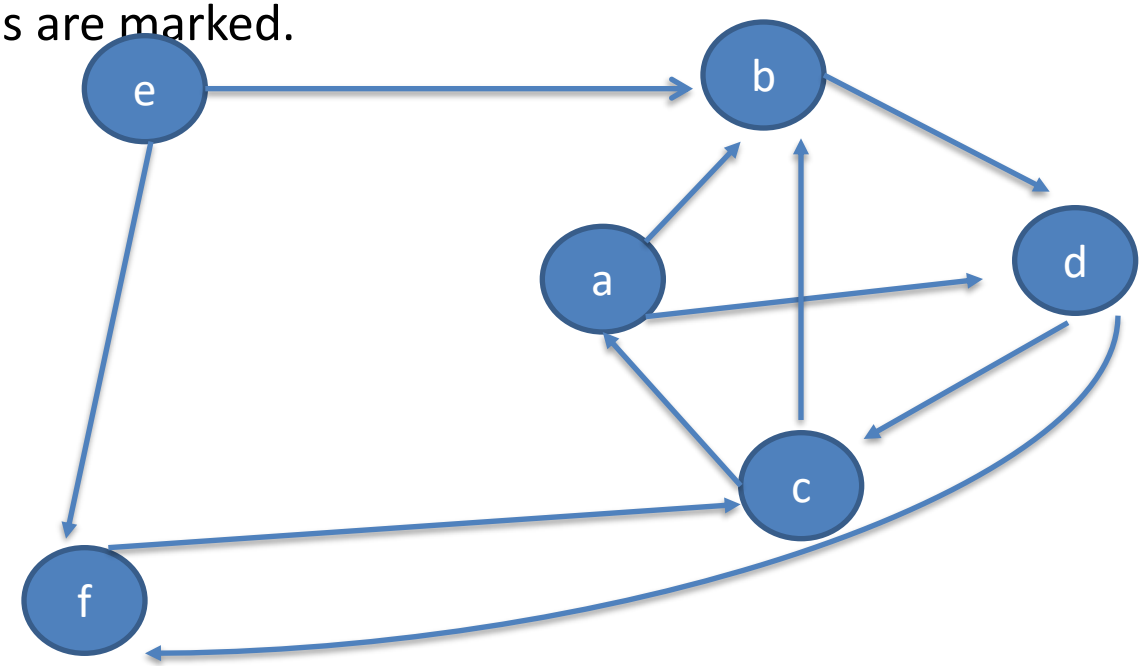
2.2 Connectivity in a graph

2.2.8 Search for strongly connected components: Example

Iteration 2: We mark vertex 'e' with + and -, so C2 = {e}.

No other vertices are marked.

G:



W = W - C2 = Ø, so finish, the number of strongly connected components = 2

C1 = {a, b, c, d, f} and C2 = {e}. The graph G is associated with a reduced graph Gr represented by:

Gr :

2.2 Connectivity in a graph

2.2.9 Graph Scheduling:

This is the arrangement of a connected graph.

Principle: Scheduling a graph involves arranging its vertices in a specific order such that the edges follow the same direction.

Different levels of vertices are defined.

Algorithm: Graph Scheduling

Data: A connected directed graph G=(V,E)

Results: The different **levels** of vertices and the **scheduled graph**

2.2 Connectivity in a graph

2.2.9 Graph Scheduling: (Algorithm Continuation)

(0) Determine the dictionary of predecessors for G formed by $(V, \Gamma^-(x))$.

(1) Identify the vertices in $\Gamma^-(x)$ that have no predecessors. $\Gamma^-(x) = \emptyset$.

(1.1) Define N_0 as the set of vertices in G with no predecessors; this set is called **level 0.**

(1.2) Cross out in the column of $\Gamma^-(x)$ all the vertices from level 0, resulting in a new column $\Gamma_1(x)$ and the subgraph G_1 with vertices V/N_0.

(2) Identify in the column Γ-(x) the vertices with no predecessors; Γ-(x) = ∅.

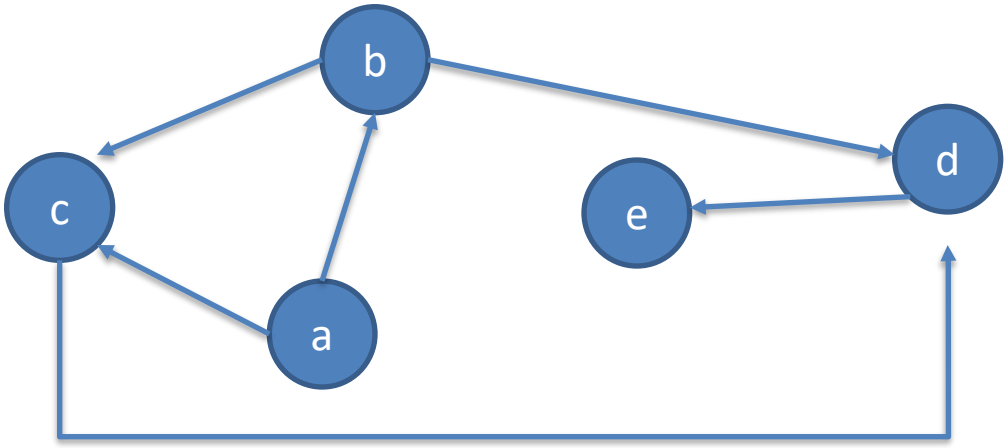
(2.1) Define N1 as the set of vertices in the graph with no predecessors.

(2.2) Cross out in the cell of Γ1(x) all the vertices at level N1, resulting in a new column Γ2(x) and the subgraph G2 generated by V/(N0 ∪ N1).

Continue this process until you complete the graph, and represent the ordered graph by levels of G.

2.2 Connectivity in a graph

2.2.9 Graph Scheduling: (example)



V	Γ- (x)
a	∅
b	a
c	a,b
d	c,b
e	d

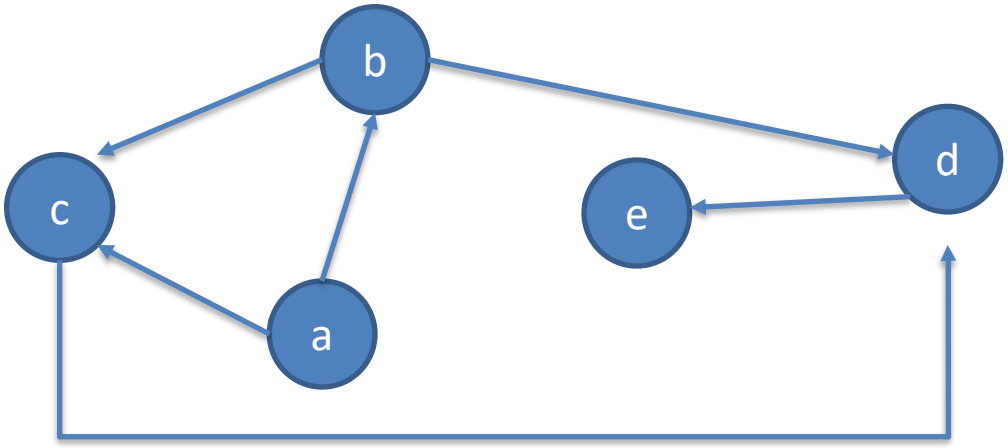
a: has no predecessor
N0={a}
Cross out vertex 'a' in Γ-(x)
And we obtain Γ-1(x)

V	Γ- 1(x)
a	/
b	∅
c	b
d	c,b
e	d

b: has no predecessor
N1={b}
Cross out vertex 'b' in Γ-1(x)
And we obtain Γ-2(x)

2.2 Connectivity in a graph

2.2.9 Graph Scheduling: (example)



V	Γ^{-} 2(x)
a	/
b	/
c	∅
d	c
e	d

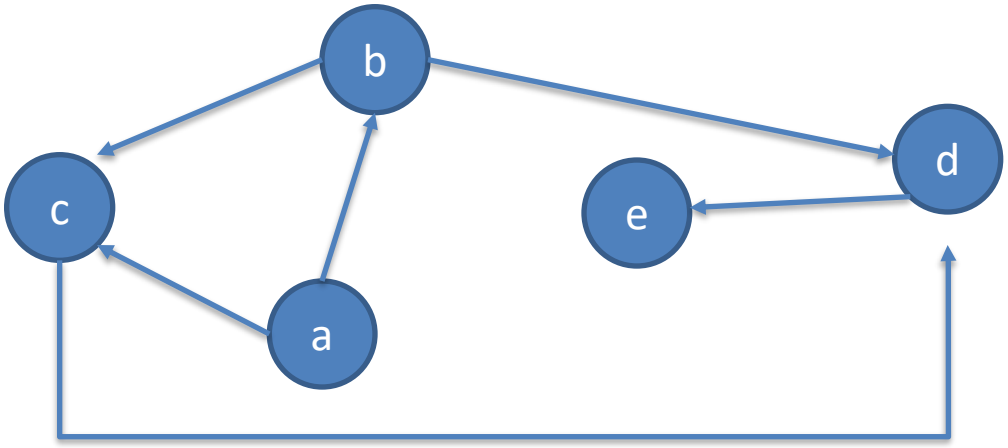
c: has no predecessor
N2={c}
Cross out vertex 'c' in Γ-2(x)
And we obtain Γ-3(x)

V	Γ^{-} 3(x)
a	/
b	/
c	/
d	∅
e	d

d: has no predecessor
N3={d}
Cross out vertex 'd' in Γ-3(x)
And we obtain Γ-4(x)

2.2 Connectivity in a graph

2.2.9 Graph Scheduling: (example)



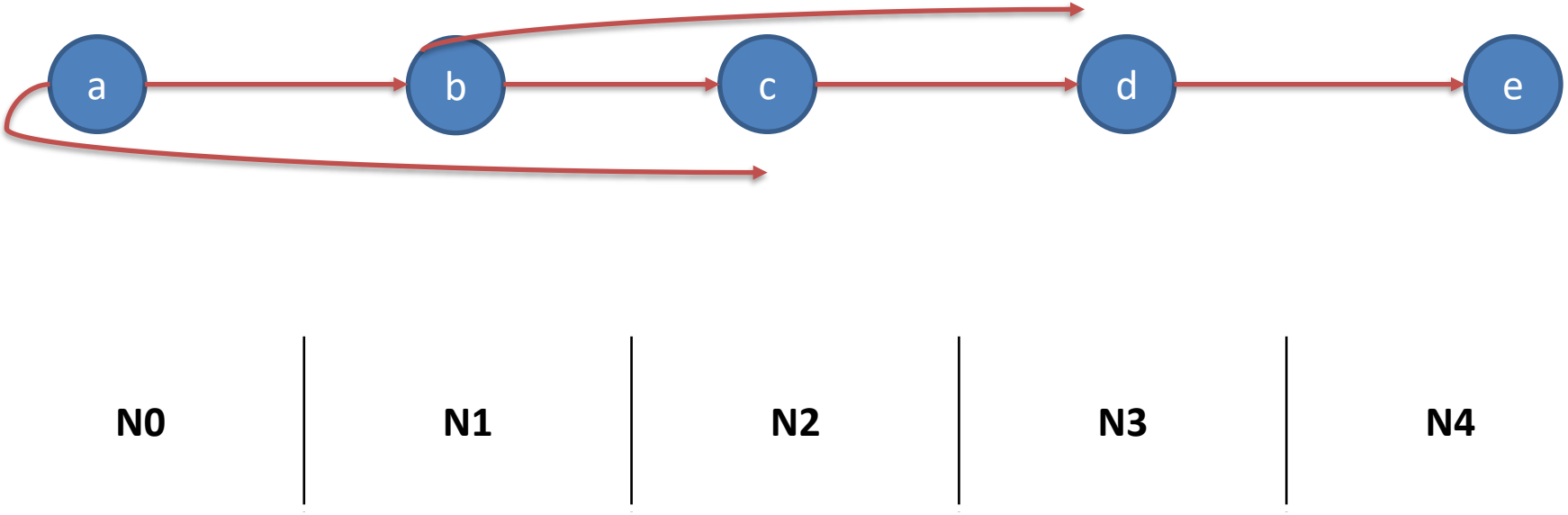
V	Γ^{-} 4(x)
a	/
b	/
c	/
d	/
e	Ø

e: has no predecessor
N4={e}

We have examined all the vertices of the graph, and thus, we have the ordered graph:
N0={a}, N1={b}, N2={c}, N3={d}, N4={e}

2.2 Connectivity in a graph

2.2.9 Graph Scheduling: (example)



This is an ordered graph

2.2 Connectivity in a graph

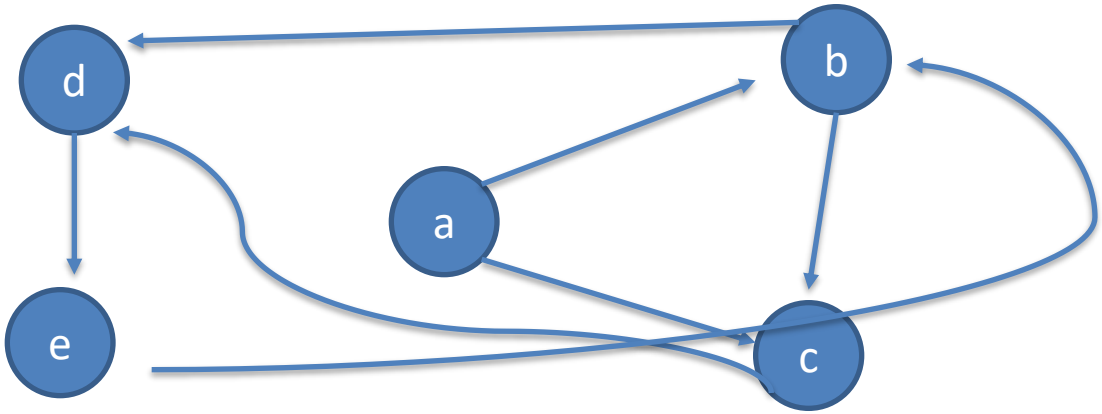
2.2.10 Searching for a circuit in a connected graph:

At a certain stage of scheduling a graph, the definition of levels becomes stuck (there are no vertices without predecessors).

➔ Consequently, ordering the graph is impossible, and we say that the graph contains a circuit.

2.2 Connectivity in a graph

2.2.10 Searching for a circuit in a connected graph: (application)



V	$\Gamma^{-}(x)$
a	∅
b	a,e
c	a,b
d	c,b
e	d

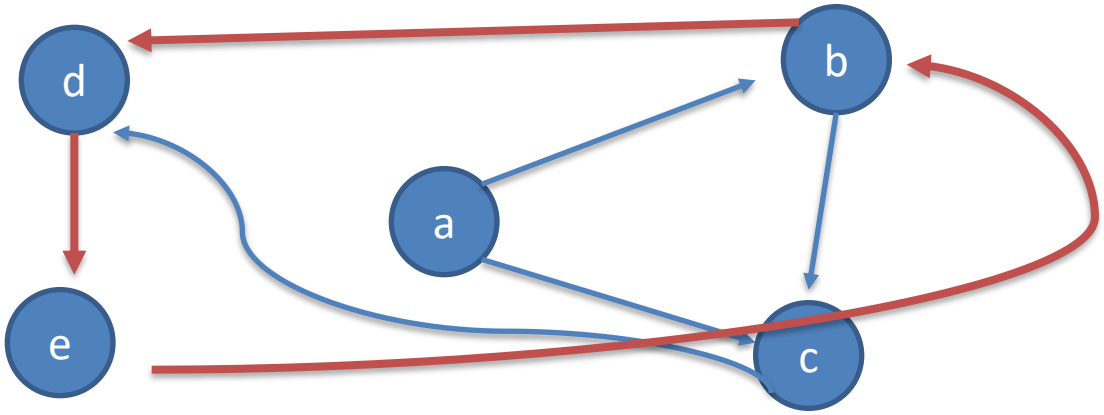
a: has no predecessor
N0={a}
We cross out vertex 'a' in Γ-(x)
And we obtain Γ1(x)

V	$\Gamma^{-}1(x)$
a	/
b	e
c	b
d	c,b
e	d

We cannot determine level N1; there are no vertices without predecessors.
⇨ G cannot be ordered.
⇨ G contains a **circuit**.

2.2 Connectivity in a graph

2.2.10 Searching for a circuit in a connected graph: (application)



V	$\Gamma^{-1}(x)$
a	/
b	e
c	b
d	c,b
e	d

To determine this circuit, we choose a vertex in Γ1(x). Let e be this vertex; it leads us back to a line in V, and we stop because the vertex e has repeated.

⇨ The sequence (e, d, b, e)

⇨ The reverse sequence forms a circuit (e, b, d, e).