GRAPH THEORY

Module 2

Rijin IK

Assistant Professor
Department of Computer Science and Engineering
Vimal Jyothi Engineering College
Chemperi

March 18, 2023

Outline

Euler graph

Euler graph

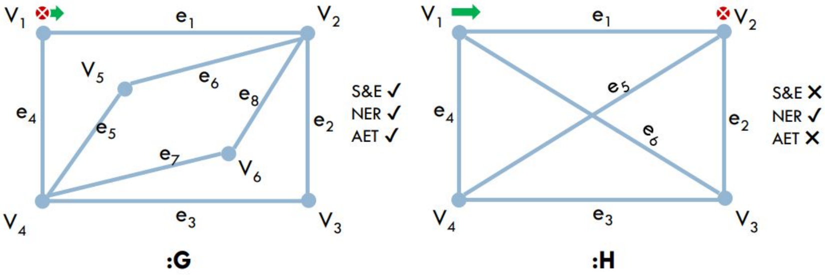
A connected graph G is called an Euler graph, if there is a closed trail/walk which includes every edge of the graph G exactly once.

OR

A graph that consists of an Euler line

- **An Euler line:** A closed walk that contains all the edges of the graph exactly once
  - **Closed** - starting and ending at same vertex (S&E)
  - **Walk** - no edge repetition, vertex may repeat (NER)
  - **Euler** - all edges traversed exactly once (AET)

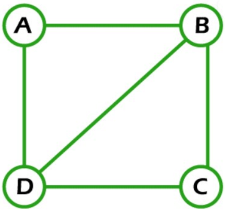
Euler graph



- Graph G is an Euler graph as we can get an Euler line,
- $V_1 e_1 V_2 e_2 V_3 e_3 V_4 e_7 V_6 e_8 V_2 e_6 V_5 e_5 V_4 e_4 V_1$
- Note that starting from any vertex, we can get an Euler line
- But, Graph H is not an Euler graph, as from none of the vertices, we are able to get an Euler line

Euler Path

- An Euler path is a path that uses every edge of a graph exactly once. An Euler path starts and ends at different vertices.



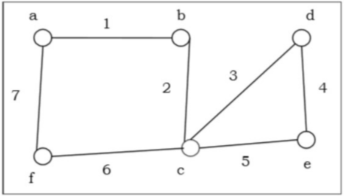
$$B \to C \to D \to B \to A \to D$$

Non-Euler Graph

Euler graph

Euler Circuit

- An Euler circuit is a circuit that uses every edge of a graph exactly once.
- An Euler circuit always starts and ends at the same vertex.
- A connected graph G is an Euler graph if and only if all vertices of G are of even degree, and a connected graph G is Eulerian if and only if its edge set can be decomposed into cycles.



The above graph is an Euler graph as a 1 b 2 c 3 d 4 e 5 c 6 f 7 g covers all the edges of the graph.

Euler's Theorem

Suppose we have a connected graph.

- If the graph has an Euler circuit, then each vertex of the graph has even degree.
- If each vertex of the graph has even degree, then the graph has an Euler circuit.

Euler graph

Theorem

A connected graph G is Euler if and only if all of its vertices are of even degree

Proof

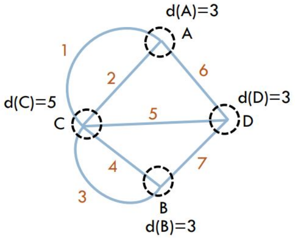
- **Suppose G is an Euler graph and connected**
- Then G contains an Euler line, which is a closed walk, containing all the edges of G exactly once
- Then, as we trace the walk, we find that the walk meets a vertex v along one edge and leaves that vertex along another edge
- This is true for all vertices of the walk, including the terminal vertex
- Hence we can say that every vertex has to be of even degree

Proof Cont..

Conversely:

- **Suppose that all vertices of G are of even degree and G is connected**
- Then start a walk from an arbitrary vertex v and go through the edges of G, one by one, without repetition and finally come back to the vertex v
- It is possible as every vertex is of even degree
- And since G is connected, we can trace through all the edges of G
- Then G is an Euler graph
- Hence the theorem

Konigsberg bridge problem



- The graph representing the Konigsberg bridge problem has got all its vertices with odd degree
- Hence it is not an Euler graph
  - As a result we cannot trace a closed walk in it
  - That is how Euler proved that it is not possible to walk over each of the seven bridges exactly once and return to the starting point

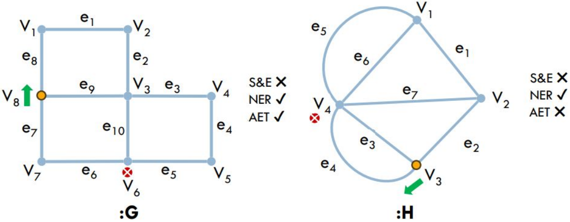
Unicursal Graph

Unicursal Graph

An open walk(or traces) that includes (or traces) all edges of a graph without retracing any edge is called a unicursal line or open Euler line. **A connected graph that has a unicursal line is called a unicursal graph.**

- It is the same as an Euler line except that the starting and the ending vertex is not the same or it is an Open Euler Line
  - Starting and ending on different vertices (S&E)
  - No edge repetition, vertex may repeat (NER)
  - All edges traversed exactly once (AET)

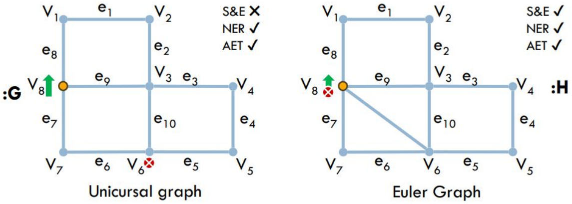
Unicursal Graph



- Graph G is a Unicursal graph as we can get a Unicursal line,
- V8 e8 V1 e1 V2 e2 V3 e3 V4 e4 V5 e5 V6 e6 V7 e7 V8 e9 V3 e10 V6
- Note that we need to start from one of the odd-degree vertices and end at the other odd-degree vertex in order to get the unicursal line
- Whereas, Graph H is not a Unicursal graph, as from none of the vertices, we are able to get a Unicursal line

Unicursal → Euler

- Just by adding an edge between the starting and ending vertex of a unicursal line, we can get an Euler line
- Hence the unicursal graph becomes an Euler graph
  - A connected graph is **unicursal if and only if it has exactly 2 vertices of odd degree**
  - Adding an edge between the two odd degree vertices makes the graph Eulerian



  - Drawing an edge between the odd-degree vertex pair makes it possible to take a closed walk, tracing all edges of G exactly once, there by making it an Euler graph
  - Hence graph H is an Euler graph

Theorem

In a connected graph G with exactly 2k odd vertices, there exists k edge disjoint subgraphs such that they together contain all edges of G and that, each is a unicursal graph

Proof:

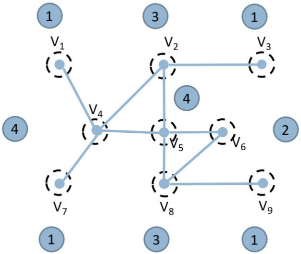
- Let the odd vertices of the given graph be named $x_1, x_2, x_3, ...x_k, w_1, w_2, w_3, ...w_k$ in some arbitrary order
- Add k edges to G between the vertex pairs, $(x_1, w_1)(x_2, w_2)............(x_k, w_k)$ to form a new graph G'
- Now since every vertex of G' is of even degree, G' consists of an Euler line p

Proof Cont..:

- Now if we remove from p the k edges we just added then it will split the graph into k walks, each of which is a unicursal line
- The first removal will result in a single unicursal line, the second removal will split that into 2 unicursal lines, and each successive removal will split a unicursal lines into 2 unicursal lines, until there are k of them
- Thus the theorem.

Unicursal Graph

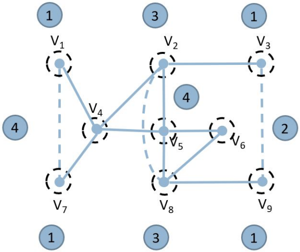
Example



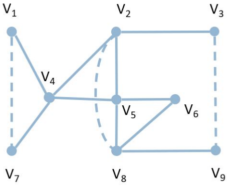
- No. of vertices, n=9
- No. of odd vertices$= 6 = 2x3; (k = 3)$
- Then there must be 3 edge disjoint subgraphs, each of which is a unicursal graph as well

Example Cont..

Add edge between every pair of odd vertices



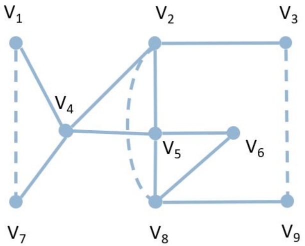
Now all vertices are even



Unicursal Graph

Example Cont..

Trace an euler Line P

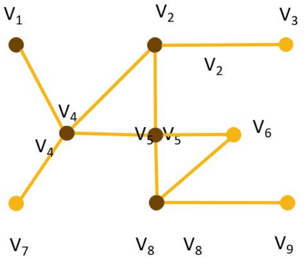


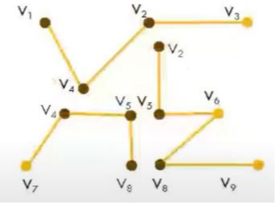
ρ : v1- v4 – v2 – v3	–	v9 – v8 – v6 – v5 – v2	–	v8 – v5 – v4 – v7	–	v1

Unicursal Graph

Example Cont..

Now remove the added edges





- The Euler line will break into 3 pieces;
- each piece is a unicursal line; and each graph is an edge disjoint subgraph of the initial graph

p : v1- v4 – v2 – v3

v9 – v8 – v6 – v5 – v2

v8 – v5 – v4 – v7

Operations on Graph

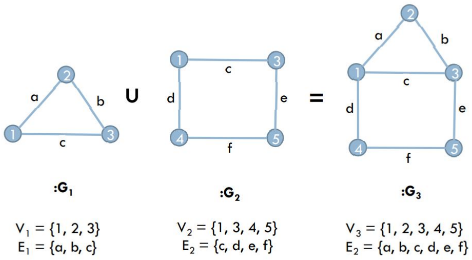
1. Union
2. Intersection
3. Ring Sum
4. Decomposition
5. Deletion
6. Fusion

Union

Let G1(V1,E1) and G2(V2,E2) be two graphs, then union of G1 and G2 is a graph G3

- Where $G3 = G1 \cup G2$
- Vertex set $V3 = V1 \cup V2$
- Edge Set $E3 = E1 \cup E2$

Example

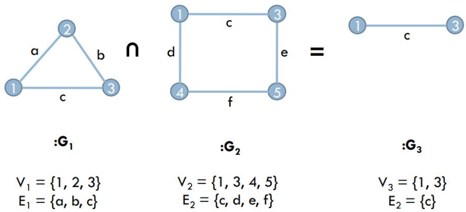


Intersection

Let G1(V1,E1) and G2(V2,E2) be two graphs, then the intersection of G1 and G2 is a graph G3

- Where $G3 = G1 \cap G2$
- Vertex set $V3 = V1 \cap V2$
- Edge Set $E3 = E1 \cap E2$

Example



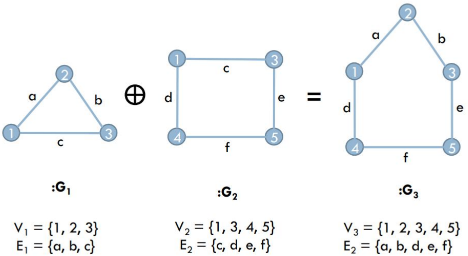
Operations on Graph

Ring sum

Let G1(V1,E1) and G2(V2,E2) be two graphs, then the ring sum of G1 and G2 is a graph G3

- Where $G3 = G1 \oplus G2$
- Vertex set $V3 = V1 \cup V2$
- Edge Set $E3 = (E1 \cup E2) - (E1 \cap E2)$

Example



Ring sum

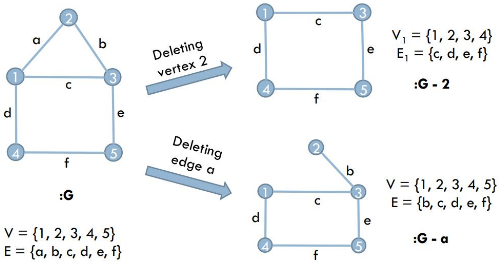
- If G1 & G2 are edge disjoint graphs, (no common edge) then G3 will be $G1 \cup G2$
- For any graph G, $G \oplus G$ is a null graph (only vertices, no edges)
- For any subgraph g of G, $G \oplus g$ is $G - g$; the compliment of g in G

Operations on Graph

DELETION

- **Deleting a vertex:** If V_i is a vertex of graph G the $G - V_i$ denotes a sub graph of G obtained by deleting vertex V_i . Deleting a vertex implies, deleting all edges incident on it.
- **Deleting an Edge:** If e_j is an edge of G then $G - e_j$ denotes a sub graph of G obtained by deleting edge e_j . However deleting an edge does not imply deleting its end vertices.

Example

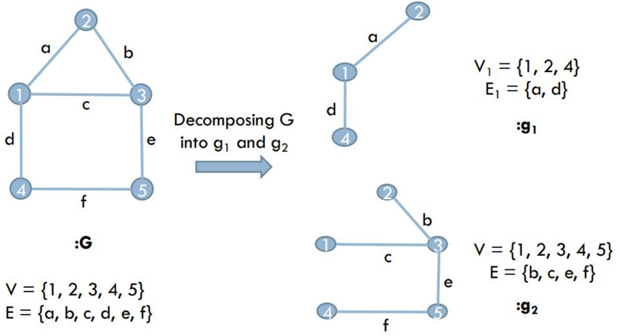


Operations on Graph

Decomposition

- A graph G is said to have decomposed into two graphs g1 and g2 if
  - $g1 \cup g2 = G$
  - $g1 \cap g2 = G=$ null graph
- All edges of G must be present either in g1 or g2 but no edge can be common for g1 and g2.
- Vertices may however be common, but has to be there either in g1 and g2
- A graph containing 'm' edges can be decomposed in $2^m - 1$ ways into pairs of sub graphs

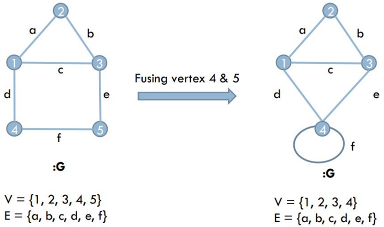
Decomposition Example



FUSION

A pair of vertices $v_i \& v_j$ are said to be fused, if the two vertices are replaced by a single vertex vk , such that all edges incident on both $v_i \& v_j$ are now incident on the new vertex or fused vertex vk

- Note
  -
    - Fusion reduces the no. of vertices in the graph by 1
    - Fusion does not alter the no. of edges
    - Degree of the new vertex will be the sum of the degrees of the vertices fused
    - Any edge between $v_i \& v_j$ shall become a self loop



Euler graph

Therom

A connected graph G is an Euler graph if and only if it can be decomposed into circuit

Proof:

- **Suppose G can be decomposed into circuits**
- Then G is a union of edge disjoint circuits
- We know that in any circuit, the degree of each vertex is 2
- Then, when we join all these circuits together to form graph G, G will have all its vertices with even degree
- Since G has all even degree vertices, G is an Euler graph

Proof Cont..:

Conversely:

- **Suppose G is an Euler graph**
- Then G has all its vertices with even degree
- Consider a vertex v1. Degree of v1 is at least 2. ie 2 edges are incident on v1.
  - Let one of these edges be connecting v1 & v2 . Since v2 is also of even degree, it must have at least 2 edges incident on it.
  - Let one of them be connecting v2 & v3. Proceeding in this fashion we eventually arrive at a vertex that has been previously traversed, there by forming a circuit p1
- Remove the circuit p1 from G. Then all the remaining vertices in the graph are again of even degree. From this remove another circuit p2 in the same way
- Continue this process of removing circuits from the graph, until no edges are left
- Thus we have decomposed the graph into many circuits

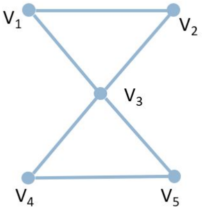
Arbitrarily traceable graphs

- Usually in an Euler graph, we can get an Euler line starting from any vertex irrespective of the order in which we traverse the vertices
- In some Euler graphs, we may not get an Euler line if we start from a certain vertex, due to the order in which we choose the intermediate vertices
- Such graphs are known as Arbitrarily Traceable Graphs

Arbitrarily traceable graphs

A graph is arbitrarily traceable graph from vertex v if an Euler line is always obtained when we start from v; but an Euler line may or may not be obtained if we start from other vertices

Arbitrarily traceable graphs

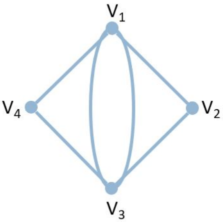


:G

- Starting from V1 may/may not get an Euler line
- Starting from V3 definitely get an Euler line

- Suppose we start from vertex V1, move to V2, and then to V3
  - At V3 , we have 3 choices of new edges. If we go back to V1 choosing e2 , we would end up without getting an Euler line
  - Situation is the same if we start with V2 , V4 or V5
  - In all these cases, there is an ambiguity at V3
  - Even though we take an untraversed edge, we may not get an Euler line
- Whereas if we start with vertex V3 , we would surely get an Euler line
- Hence the name ‘arbitrarily traceable from V3

Arbitrarily traceable graphs

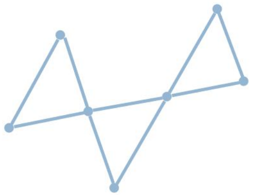


:H

- Starting from V4 may/may not get an Euler line
- Starting from V1 definitely get an Euler line

- H is an arbitrarily traceable graph from two of its vertices V1 and V3
- Because if we start with V1 or V3 , we would surely get an Euler line irrespective of the order in which we choose the intermediate edges
- Whereas, if we start with V2 or V4 we may or may not end up with an Euler line
- Hence we say that graph H is an arbitrarily traceable graph from V1 and V3

Arbitrarily traceable graphs

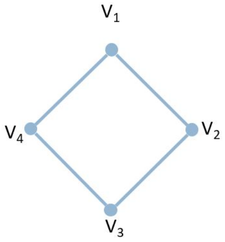


:J

- Starting from V1 may/may not get an Euler line

- J is not arbitrarily traceable graph from any of the vertices
- Because whichever vertex we start, there is an ambiguity at every intermediate vertex & hence we may or may not end up with an Euler line
- However J is an Euler graph

Arbitrarily traceable graphs

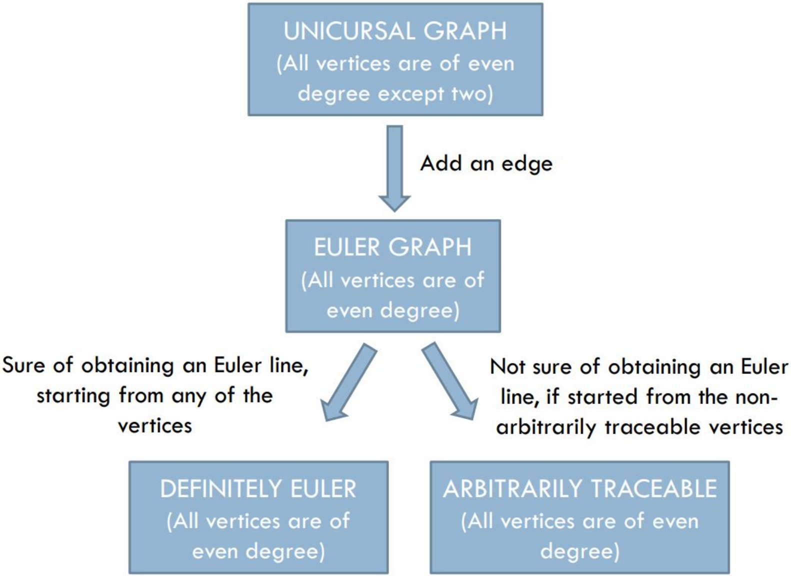


:K

- Starting from any vertex Definitely get an Euler line

- Graph K is an arbitrarily traceable graph from all its vertices
- Because whichever vertex we start with, we are able to get an Euler line irrespective of the order in which we choose the intermediate vertices
- They are normal Euler graphs (definitely Euler)

Arbitrarily traceable graphs



Arbitrarily traceable graphs

Theorem

An Euler graph G is arbitrarily traceable from vertex v in G if and only if every circuit in G contains v

Proof:

- **Suppose G is an Euler graph that is arbitrarily traceable from vertex v**
  - Assume that there is a cycle C not passing through v
  - Remove the cycle C from G to get H i.e, H = G-C
  - Since a cycle has all its vertices with degree 2, taking out a cycle from an Euler graph leaves the remaining graph, Euler
- Now in H trace an Euler line p starting & ending at v; obviously p contains all edges of H
- Bring back the removed cycle C to H and try to extend the v to v walk so as to include all edges of C
  - It would not be possible without repeating at least one edge, contradicting that G is an ATG from v (starting a walk from v we should always get an Euler line)

Proof Cont...:
Conversely

- **Let every circuit of G contain v**
- Assume on the contrary that G is not arbitrarily traceable from v
- Then there must be a v to v closed walk, W in G that contains all edges of G. Let one such edge be incident at a vertex u on W
- So, every vertex of H=G-W is of even degree & v is an isolated vertex of H & u is not
- Then, component of H containing u is therefore Eulerian sub graph of G not passing through v contradicting the assumption
- Hence the proof

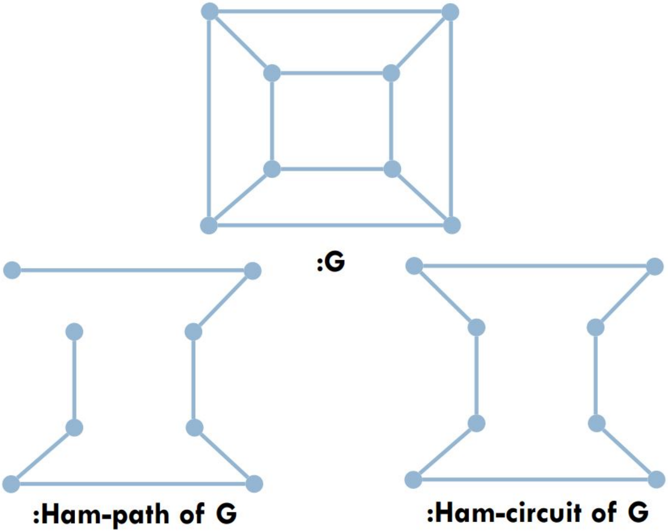
Hamiltonian Graph

A graph that contains a Hamiltonian circuit

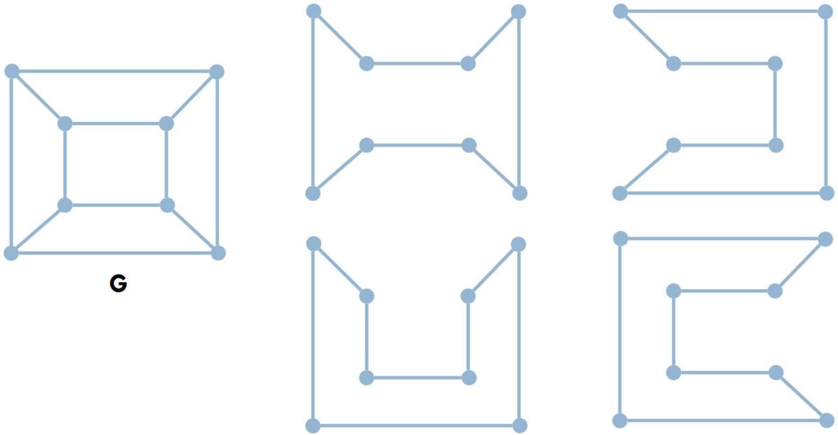
- **A Hamiltonian circuit:** A circuit in a connected graph that traverses all its vertices exactly once

- **A Hamiltonian Path:** A path in a connected graph that traverses all its vertices exactly once
- Hamiltonians graphs do not require to contain all edges of the graph; it may or may not contain all the edges;
- constrain on vertex repetition automatically controls edge repetition

Hamiltonian Graph



More Ham-circuits of G



G

Hamiltonian Graph

Hamiltonian Path

- Edges & Vertices cannot repeat
- Must contain all vertices of the graph
- May drop out some of the edges
- Will contain exactly (n-1) edges
- Hence length of Ham-path is always (n-1)

Hamiltonian circuit

- Edges &Vertices cannot repeat
- Starts & ends at the same vertex
- Must contain all vertices of the graph
- It is not necessary to contain all edges
- Will contain exactly n edges,Hence length of Ham-circuit is always n
- If we remove one edge from Ham-circuit, we get a Ham-path
- Since Ham-path is a sub graph of Ham-circuit, every graph that has a Ham-circuit will have a Ham-path; However, the reverse may not be true

Necessary & Sufficient condition

- How do we know if a graph is Euler?
  - All vertices will be of even degree
- How do we know if a graph is Hamiltonian?
  - No conditions found yet, whereby we can identify if a graph is Hamiltonian or not
  - However we can assure that all complete graphs with 3 or more vertices are Hamiltonian

Hamiltonian Graph

Ham-circuits in Complete graphs

- In a complete graph with n vertices,
  - Maximum number of Ham-circuits possible is

$$\frac{(n-1)!}{2}$$

  - Maximum number of edge disjoint Ham-circuits is

$$\frac{(n-1)}{2} \quad \textit{if } 'n' \textit{ is odd}$$

$$\frac{(n-2)}{2} \quad \textit{if } 'n' \textit{ is even}$$

THEOREM

In a complete graph with n vertices there are $\frac{(n-1)}{2}$ edge-disjoint Hamiltonian circuits,if n is an odd number ≥ 3

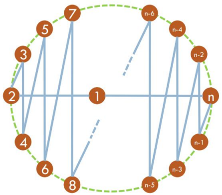
Proof:

- A complete graph with n vertices has $\frac{n(n-1)}{2}$ edges
- Any Ham-circuit will have n edges
- Therefore, the no. of edge disjoint Ham-circuits cannot exceed $\frac{(n-1)}{2}$
- That means the maximum no. of edge disjoint Ham-circuits is $\frac{(n-1)}{2}$

Proof cont..:

Obtaining the edge disjoint Ham-circuits

- The subgraph given below is a Ham-circuit of a complete graph with n vertices
- Starting from vertex 1 and tracing along the solid lines, we get the first circuit which is 1,2,3,4........... (n-1), n,1



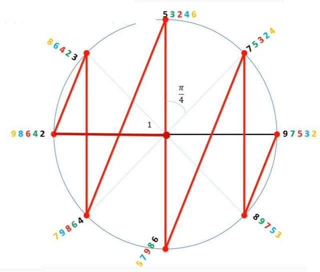
Proof cont..:

- Now, imagine the vertices (except 1) to be lying on the circumference of a circle
- Rotate the circle by $(\frac{360}{n-1})^0, (\frac{2x360}{n-1})^0, (\frac{3x360}{n-1})^0 \ldots . (\frac{n-3}{2} \frac{x360}{n-1})^0$
- Each rotation produces a new Ham-circuit that has no edge in common with the previous ones
- There for the number of edge disjoint Ham-circuits$=\frac{n-3}{2} + 1 = \frac{n-1}{2}$
- The 1st solution is obtained in the default position, without any rotation
- Hence the theorm

Hamiltonian Graph

Example:(Recalling seating arrangement problem)

- Keeping the vertices fixed on a circle, rotate the polygonal pattern clockwise by $\frac{1x2\pi}{8}, \frac{2x2\pi}{8}, \frac{3x2\pi}{8}$.produces a Hamiltonian circuit that has no edge in common with any previous one.
- The four Hamiltonian circuits are
  - 1234567891
  - 1426385971
  - 1648293751
  - 1869472531
  - 1987654321(Returns back tosame sequence)

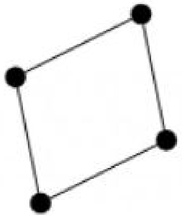


Hamiltonian Graph

Dirac's theorem

Let G = (V, E) be a graph with n vertices ($n \geq 3$) in which each vertex has degree at least $\frac{n}{2}$. Then G has a Hamiltonian cycle.

Example:

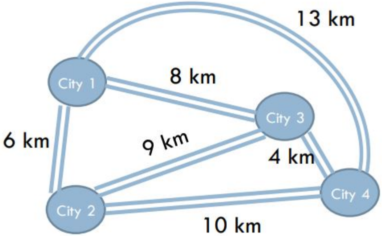


- Here n is 4 ie ≥ 3
- $d(a) = d(b) = d(c) = d(d) = 2$ ie $\geq \frac{n}{2}$
- So the graph is Hamiltonian

Travelling Salesman Problem

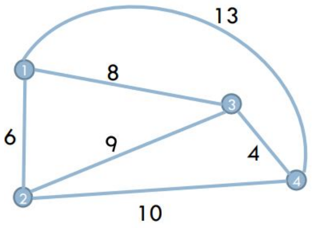
The real life problem

A salesman is required to visit a number of cities during a trip. Given the distance between the cities, in what order should he travel so as to visit every city exactly once & return at the starting point, with minimum distance travelled?



Travelling Salesman Problem

- Representing cities as vertices & roads between them as weighted edges
- The weight of each edge represents the distance between the connecting cities



Number of solutions

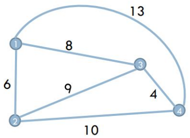
- If each of the cities has a road to every other city, we get a complete weighted graph
- The graph has numerous Ham-circuits; the salesman has to choose the one with the smallest sum of distances
- The no. of Ham-circuits in a complete graph with n vertices $= \frac{(n-1)!}{2}$
  - At the starting vertex, we have (n-1) edges to choose from
  - At the next vertex, we have (n-2) edges to choose from and so on
  - At the second last vertex we will have (n-(n-2)) $=$ 2 edges to choose from
  - Finally at the last vertex we will have just 1 edge to move on
  - Hence we have (n-1)! possible no. of choices
  - However this has to be divided by 2 as each Ham-circuit has been counted twice
  - ie $= \frac{(n-1)!}{2}$ Ham-circuit

Travelling Salesman Problem

Finding the solutions

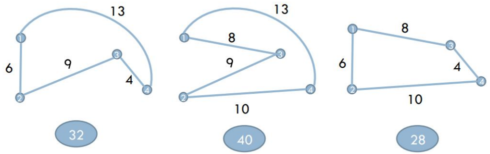
- Enumerating all $= \frac{(n-1)!}{2}$ Ham-circuits & calculating the sum weight of each circuit, and choosing the one with the lowest sum-weight is the solution for the salesman
- However, It is not practical
- Neither can we employ some computer algorithm for the job as no such algorithm has proved to be efficient enough

Example:



In the example $= \frac{(4-1)!}{2} = 3$ Ham-circuits

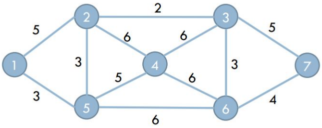
If the salesman was to start from city 1, then possible routes are



He would choose the one with lowest weight, 28

Example:

Q : Trace a travelling salesman tour on the graph given below

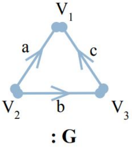


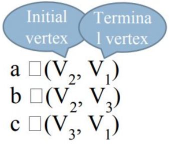
Directed Graph

A directed graph (or a digraph, or an oriented graph) G consists of a set of vertices V = { v1, v2, . . . }, a set of edges E = { e1, e2, . . . }, Such that $e_k \in E$ is identified by an ordered pair (v_i, v_j) where $v_i \& v_j \in V$ and e_k directed from v_i *to* v_j

- The vertex V_i which is the starting vertex of e_k is known as the initial vertex & V_j, which is the ending vertex of e_k is known as the terminal vertex

Example





- G = (V, E)
- V = {V1,V2,V3}
- E = {a, b, c}

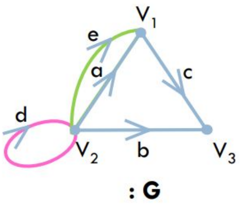
Directed Graph

Self loop

- An edge having the same vertex as its initial vertex and terminal vertex is called a self loop

Parallel edges

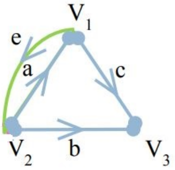
- Two edges of a digraph are said to be parallel edges if they have the same initial vertex and terminal vertex; i.e, they have the same direction



- edge d is a self loop with V_2 as its initial and terminal vertex
- edge e and a are together known as parallel edges

Symmetric edges

- Two edges of a digraph are said to be symmetric edges if the initial vertex of one edge is the terminal vertex of the other and the terminal vertex of the former is the initial vertex of the latter & hence they will be opposite in direction



: G

edge e and a are together known as symmetric edges

Directed Graph

Degree of a vertex (Valency)

- **Indegree (invalency):** The number of edges incident into a vertex;
  - It is denoted by d^{-}(V)
- **Outdegree (outvalency):** The number of edges incident out from a vertex;
  - It is denoted by d^{+}(V)

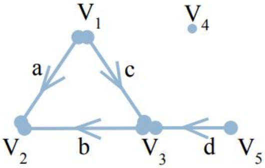
for any vertex V, $d(V) = d^{-} + d^{+}(V)$

Isolated vertex

- A vertex having no incident edge is called an isolated vertex
- The indegree & outdegree of an isolated vertex is zero

Pendant vertex

- A vertex that has only 1 incident edge is called a pendant vertex; direction can be either 'into the vertex' or 'away from the vertex'
- The total degree of a pendant vertex is 1



- vertex V_4 is an isolated vertex
- V_5 is a pendant vertex; $d^-(V_5) = 0$ & $d^+(V_5) = 1$

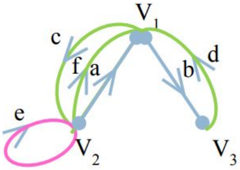
Directed Graph

Symmetric digraphs

- A digraph that has a symmetric pair for all its edges

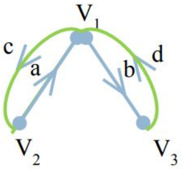
Simple symmetric digraph

- A digraph that has a symmetric pair for all its edges as well as free of self loops and parallel edges



: **G**

Symmetric digraph

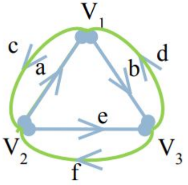


: **H**

Simple symmetric digraph

Complete symmetric digraph

- A simple digraph that has a symmetric pair of edges between every pair of vertices
- It will contain n(n-1) edges



Complete symmetric digraph

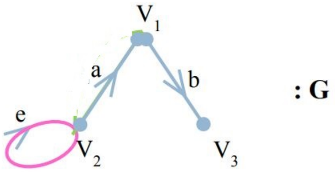
Directed Graph

Asymmetric digraph

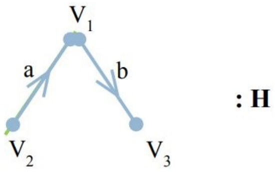
- A digraph that do not have a symmetric pair for any of its edges(can have self loop)

Simple asymmetric digraph

- A digraph that do not have a symmetric pair for any of its edges as well as free of self loops and parallel edges



Asymmetric digraph

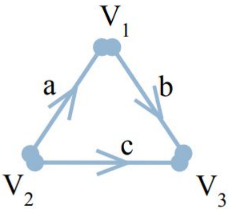


Simple asymmetric digraph

Directed Graph

Complete asymmetric digraph

- A simple digraph that has an edge between every pair of its vertices and none of its edges has a symmetric pair
- It will contain $\frac{n(n-1)}{2}$ edges
- Also known as complete tournament



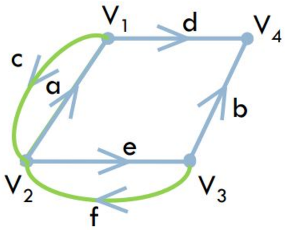
Complete asymmetric digraph

Directed Graph

Note:

- For a graph to be symmetric, all its edges must have a symmetric pair
- For a graph to be asymmetric, none of its edges must have a symmetric pair

The graph G is neither symmetric nor asymmetric



: G

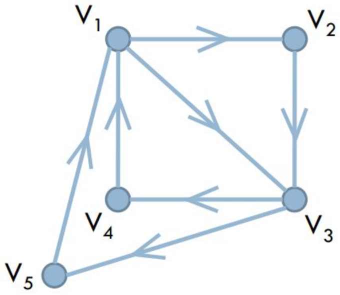
- Two of its edges have a symmetric pair
- Two edges do not have a symmetric pair

Directed Graph

Balanced digraph (Isograph)

- A digraph that has all its vertices, balanced
  - i.e, at every vertex indegree = outdegree
- Eg: G is a balanced digraph since

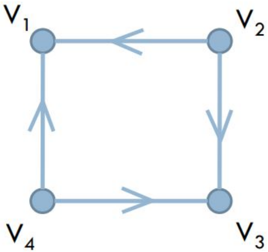
- $d^-(V_1) = d^+(V_1) = 2$
- $d^-(V_2) = d^+(V_2) = 1$
- $d^-(V_3) = d^+(V_3) = 2$
- $d^-(V_4) = d^+(V_4) = 1$
- $d^-(V_5) = d^+(V_5) = 1$



: G

Regular digraph

- A digraph in which all vertices have the same degree (total degree)
- Eg: G is a regular digraph since
- $d(V1) = d(V2) = d(V3) = d(V4) = 2$

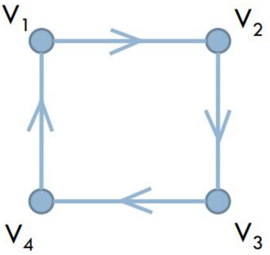


: G

Balanced regular digraph

- A digraph in which all vertices have the same indegree as well as outdegree
- Eg: G is a balanced regular digraph since

- $d^-(V_1) = d^+(V_1) = 1$
- $d^-(V_2) = d^+(V_2) = 1$
- $d^-(V_3) = d^+(V_3) = 1$
- $d^-(V_4) = d^+(V_4) = 1$



: G

Directed Graph

Theorem

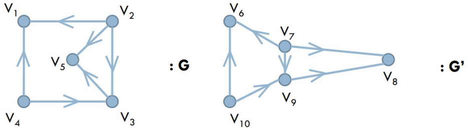
In any digraph, sum of all indegrees = sum of all outdegrees = no. of edges
i.e $\sum_{i=1}^{n} d^{+}(v_i) = \sum_{i=1}^{n} d^{-}(v_i) = e$

Proof:

- Since each directed edge of any digraph, contributes '1' towards the total indegree & '1' towards the total outdegree of the graph,
- no. of edges must give the total indegree, which is the same as the total outdegree of the graph

Isomorphic digraphs

- Two digraphs are said to be isomorphic if
  - Their undirected versions are isomorphic
  - Corresponding edges are in the same direction
- Hence for 2 digraphs to be isomorphic they must have
  - Same no. of vertices
  - Same no. of edges
  - Corresponding vertices with same indegree and out degree



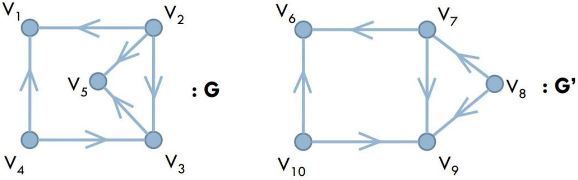
Isomorphic digraphs

	G	G'
No. of vertices	5	5
No. od edges	6	6

G	Indeg	Outdeg	G'	Indeg	Outdeg
V_1	2	0	V_6	2	0
V_2	0	3	V_7	0	3
V_3	2	1	V_8	2	0
V_4	0	2	V_9	2	1
V_5	2	0	V_{10}	0	2

Isomorphic digraphs

Example



Non-isomorphic digraphs

	G	G'
No. of vertices	5	5
No. od edges	6	6

G	Indeg	Outdeg	G'	Indeg	Outdeg
V_1	2	0	V_6	2	0
V_2	0	3	V_7	1	2
V_3	2	1	V_8	0	2
V_4	0	2	V_9	3	0
V_5	2	0	V_{10}	0	2

Directed Graph

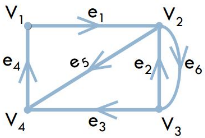
Directed Walk

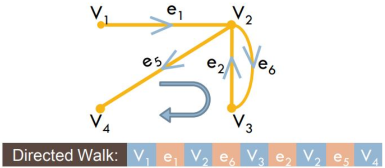
A finite alternating sequence of vertices and edges, beginning and ending at vertices, such that

- each edge is directed from the vertex preceding it and towards the vertex following it and no edge appears more than once
  - **All edges must be orienting in one direction**

In a directed walk

- No edge can appear more than once
- **All edges must be orienting in one direction**
- Vertex may appear more than once
- Starts and ends at different vertices





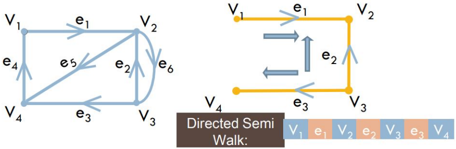
Directed Graph

Directed Semi-Walk

A finite alternating sequence of vertices and edges, beginning and ending at vertices, such that

- each edge is incident with the vertex preceding it and following it and no edge appears more than once
  - **Edges may have any direction**

- In a directed semi-walk
  - No edge can appear more than once
  - **Edges may have any direction**
  - Vertex may appear more than once
  - Starts and ends at different vertices
    - when the direction is taken off, walk and semi-walk is the same



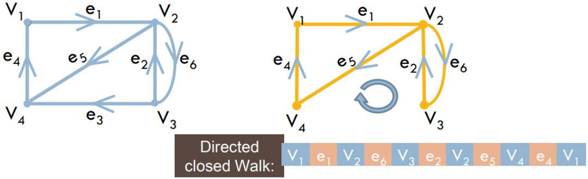
Directed Graph

Directed Closed-Walk

A finite alternating sequence of vertices and edges, beginning and ending at the same vertex, such that that

- each edge is directed from the vertex preceding it and towards the vertex following it and no edge appears more than once

- In a directed Closed-walk
  - No edge can appear more than once
  - **All edges must be orienting in one direction**
  - Vertex may appear more than once
  - Walk is a sub graph of G
  - Starts and ends at the same vertex



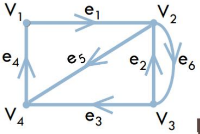
Directed Graph

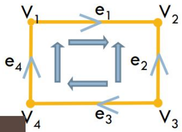
Directed Closed Semi-Walk

A finite alternating sequence of vertices and edges, beginning and ending at the same vertex, such that that

- each edge is incident with the vertex preceding it and following it and no edge appears more than once

- In a directed Closed-walk
  - No edge can appear more than once
  - **Edges may have any direction**
  - Vertex may appear more than once
  - Walk is a sub graph of G
  - Starts and ends at the same vertex





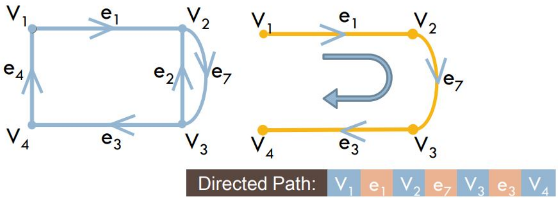
Directed Closed Semi Walk: V_1 e_1 V_2 e_2 V_3 e_3 V_4 e_4 V_1

Directed Graph

Directed path

An Open directed walk in which no vertex appears more than once is called a directed path.

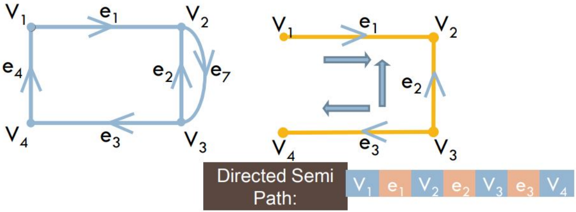
- No vertex or edge repetition
- Starting and ending at different vertices
- All edges orienting towards same direction



Directed semi-path

An Open directed walk in which

- No vertex or edge repetition
- Starting and ending at different vertices
- Edges may have any direction

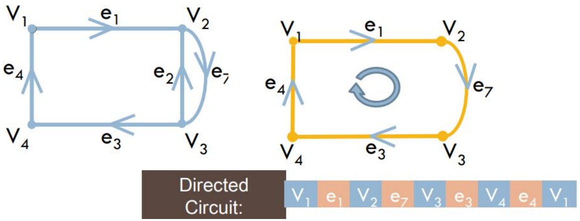


Directed Graph

Directed circuit

Closed directed walk in which

- No vertex or edge repetition
- Starting and ending at the same vertex
- All edges orienting towards same direction

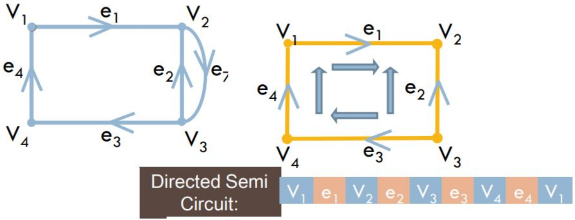


Directed Graph

Directed semi-circuit

Closed directed walk in which

- No vertex or edge repetition
- Starting and ending at the same vertex
- Edges may have any direction



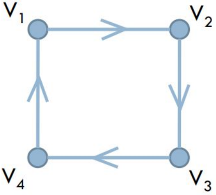
Connected digraphs

Strongly connected:

- A digraph is strongly connected if there is at least one directed path between every pair of its vertices

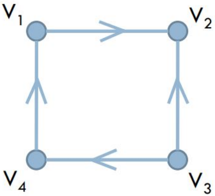
Weakly connected:

- A digraph is weakly connected if there is at least one path between every pair of its vertices, irrespective of the directions. Hence the undirected version of a weakly directed graph is always connected



Strongly connected graph

: **G**



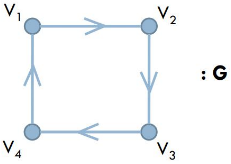
Weakly connected graph

: **H**

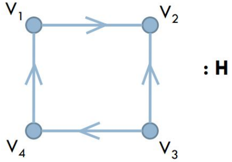
Directed Graph

Accessibility / Reachability

In a digraph G, a vertex vj is said to be accessible from vi if there is a directed path from vi to vj



Strongly connected graph



Weakly connected graph

Accessibility from the vertices

G	V_1	V_2	V_3	V_4
V_1	-	✓	✓	✓
V_2	✓	-	✓	✓
V_3	✓	✓	-	✓
V_4	✓	✓	✓	-

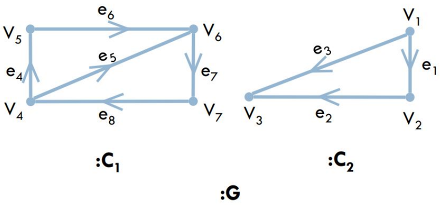
H	V_1	V_2	V_3	V_4
V_1	-	✓	×	×
V_2	×	-	×	×
V_3	✓	✓	-	✓
V_4	✓	✓	×	-

Directed Graph

Disconnected digraph

A digraph is disconnected if its corresponding undirected version is disconnected

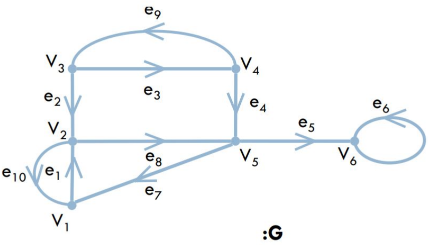
- Each component of the disconnected digraph is a digraph itself



- Graph G has 2 components, c1 and c2
- c1 is strongly connected
- c2 is weakly connected

Fragments

The maximal strongly connected subgraphs of a digraph or of a component are called fragments



In graph G, the fragments are

- {e3,e9}
- {e8,e7,e1,e10}
- {e6}

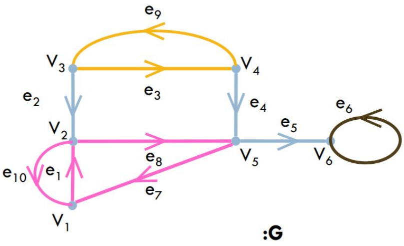
Directed Graph

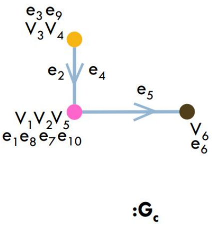
Condensation

The condensation G_C of a digraph G is obtained by replacing each strongly connected fragment by a single vertex & all directed edges from one fragment to another are replaced by a single directed edge

- Condensation of a strongly connected digraph is a single vertex
- Condensation of a digraph has no directed circuit

G_C is the condensation of graph G





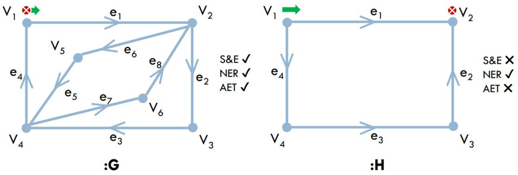
Directed Graph

Euler digraph

A digraph that contains a directed Euler line

- **A directed Euler line:** A closed directed walk which traverses every edge of the graph exactly once

Every Euler digraph would be a strongly connected digraph; However the reverse may not be true



- Graph G is an Euler digraph as we can get an directed Euler line,
  - $V_1\ e_1\ V_2\ e_2\ V_3\ e_3\ V_4\ e_7\ V_6\ e_8\ V_2\ e_6\ V_5\ e_5\ V_4\ e_4\ V_1$
- Graph H is not an Euler graph, as from none of the vertices, we are able to get a directed Euler line

Directed Graph

Theorem

A digraph G is an Euler digraph if & only if G is connected & every vertex of G is balanced

Proof:

- **Suppose G is an Euler digraph**
- Then G must contain a directed Euler line - a closed directed walk, W
- W must contain all the edges of G
- So as to visit a vertex, we need to traverse an edge directed towards the vertex & one that is directed away from that vertex
- Hence for each visit, a vertex needs an in degree and a corresponding out degree
- This is true for all the vertices of W, including the initial vertex as the walk finally ends there
- Hence every vertex need to be balanced

Directed Graph

Proof Cont..

- **Suppose every vertex of G is balanced**
- Start a walk W from an arbitrary vertex v. Visit all possible vertices. At each vertex we are able to leave along another edge, not yet traversed, as indegree = outdegree. Continue the process until back at v. If W contains all edges of G, then G is an Euler digraph
- If not, remove all edges of W from G
- The remaining graph G' has again, all its vertices, balanced. Now, since G is connected, there would be a vertex u, common to W & G'. Starting from u, start a walk W' in G'. If W' covers all the edges of G' then then G is an Euler digraph
- If not, remove all edges of W' from G'
- Call the remaining graph as G".Continue the process, until we get a closed walk, that traverse all edges of G exactly once
- Hence G is an Euler digraph

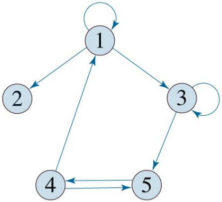
Digraphs and binary relation

- A binary relation on a finite set can also be represented using a directed graph (a digraph for short).
- If a binary relation R is defined on a set A then the elements of the set A are represented by vertices, and the ordered pairs of the relation R are represented by the directed edges.
- Suppose we are given a set A and a binary relation R

$$A = \{1, 2, 3, 4, 5\}$$

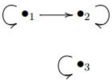
$$A = \{(1,1),(1,2),(1,3),(3,3),(3,5),(4,1),(4,5),(5,4)\}$$

The relation R on A is represented by its digraph as follows:



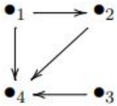
Reflexive relation:

- A relation R on a set X that satisfies $x_i R x_i$ for every $x_i \in X$ is called a reflexive relation.
- The digraph of a reflexive relation will have a self loop at every vertex.
- Such a digraph representing a reflexive binary relation on its vertex set may be called a **reflexive digraph**
- Example: $A = \{1, 2, 3\}$ and R is the relation $R = \{(1, 1), (1, 2), (2, 2), (3, 3)\}$. This is a reflexive relation, and the associated diagram is



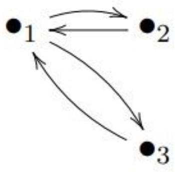
Digraphs and binary relation

- A digraph in which no vertex has a self loop is called an **irreflexive digraph**
- Example:$A = \{1, 2, 3, 4\}$ and R is the relation $R = \{(1, 4), (1, 2), (2, 4), (3, 4)\}$. This is a irreflexive relation, and the associated diagram is



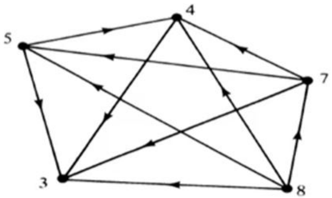
Symmetric Relation:

- For some relation R it may happen that for all x_i and x_j, if $x_i R x_j$ holds then $x_j R x_i$ also holds. Such a relation is called a symmetric relation.
- The digraph of a symmetric relation is a **symmetric digraph** because for every directed edge from vertex x_i *to* x_j there is a directed edge from x_j *to* x_i
- Example: $A = \{1, 2, 3\}$ and R is the relation $R = \{(1, 2), (2, 1), (3, 1), (1, 3)\}$. This is a symmetric relation, and the associated diagram is



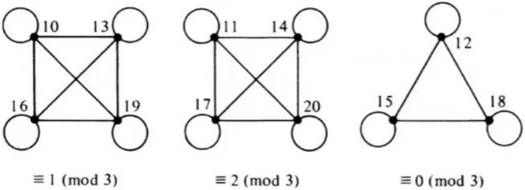
Transitive Relation:

- A relation R is said to be transitive if for any three elements $x_{,i}$, x_j *and* x_k in the set , $x_i R x_j$ and $x_j R x_k$ always imply $x_i R x_k$.
- A digraph representing a transitive relation is called a **transitive directed graph.**
- Example: $A = \{3, 4, 5, 7, 8\}$ and R is the relation(is less than) $R = \{(7, 4), (4, 3), (7, 3)\}$. This is a transitive relation, and the associated diagram is



Equivalence Relation:

- A binary relation is called an equivalence relation if it is reflexive,symmetric and transitive.
- The graph representing equivalence relation is called equivalence graph
- Example:
  - A set of 11 integers(10 through 20) consisting of equivalence relation "is congruent to modulo 3"
  - $A = \{10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20\}$



FLEURY'S ALGORITHM

FLEURY'S ALGORITHM

It is an algorithm used to find an Euler path or circuit in a graph, provided it exists.

1. Start with any vertex of non-zero degree.
2. Choose any edge leaving this vertex, which is not a bridge (cut edges).
3. If there is no such edge, stop.
4. Otherwise, append the edge to the Euler tour, remove it from the graph, and repeat the process starting with the other endpoint of this edge.

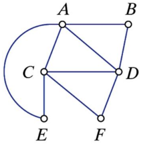
FLEURY'S ALGORITHM

It is an algorithm used to find an Euler path or circuit in a graph, provided it exists.

1. **Preliminaries:** Make sure that the graph is connected and either
   - Case (1) has no odd vertices (circuit)
   - Case (2) has just two odd vertices (path).
2. **Start:** Choose a starting vertex.
   - In case (1) this can be any vertex;
   - case (2) It must be one of the two odd vertices.
3. **Intermediate steps:** At each step, if you have a choice, don't choose a bridge of the yet-to-be-traveled part of the graph.However, if you have only one choice, take it.
4. **End:** When you can't travel any more, the circuit (path) is complete.
   - In case (1) you will be back at the starting vertex; in case (2) you will end at the other odd vertex.

Example

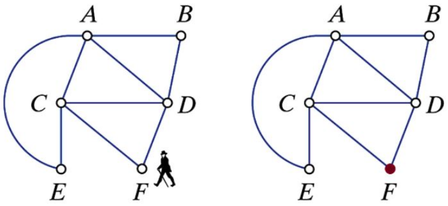
- To find an Euler circuit in the graph shown, we will start with two copies of the graph.
- Copy 1 is to keep track of the "future"
- Copy 2 is to keep track of the "past."
- Every time you travel along an edge, erase that edge from copy 1, but mark it (say in red) and label it with the appropriate number on copy2
- As you move forward copy 1 disappears and copy 2 gets redder, showing the actual Euler path or circuit.



FLEURY’S ALGORITHM

Example Cont...

Start: We can pick any starting point we want. Let’s say we start at *F*.



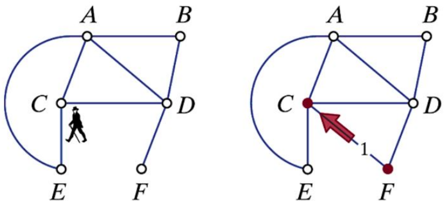
Copy 1 Copy 2

Current Path	Next edge
W=F	{F,C}

FLEURY'S ALGORITHM

Example Cont...

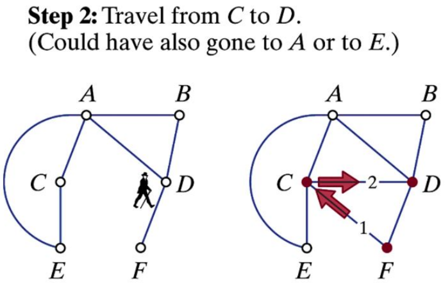
Step 1: Travel from F to C.
(Could have also gone from F to D.)



Copy 1 Copy 2

Current Path	Next edge
W=F	{F,C}
W=F,C	{C,D}

Example Cont...

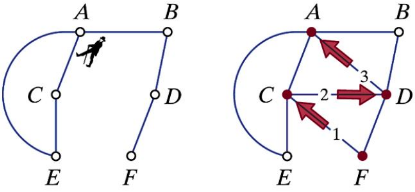


Copy 1 Copy 2

Current Path	Next edge
W=F	{F,C}
W=F,C	{C,D}
W=F,C,D	{D,A}

Example Cont...

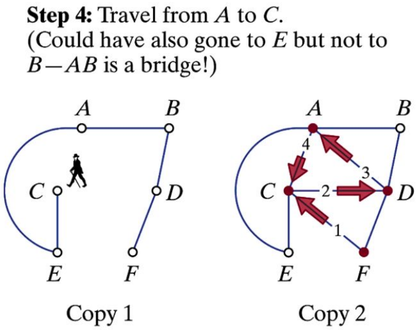
Step 3: Travel from D to A.
(Could have also gone to B but not to F—DF is a bridge!)



Copy 1 Copy 2

Current Path	Next edge
W=F	{F,C}
W=F,C	{C,D}
W=F,C,D	{D,A}
W=F,C,D,A	{A,C}

Example Cont...

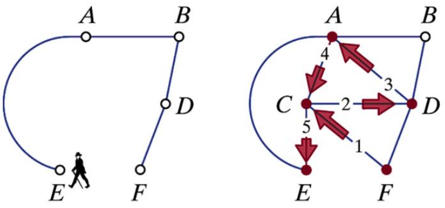


Current Path	Next edge
W=F	{F,C}
W=F,C	{C,D}
W=F,C,D	{D,A}
W=F,C,D,A	{A,C}
W=F,C,D,A,C	{C,E}

FLEURY'S ALGORITHM

Example Cont...

Step 5: Travel from C to E.
(There is no choice!)

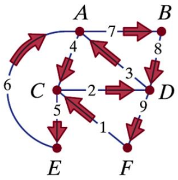


Copy 1 Copy 2

Current Path	Next edge
W=F	{F,C}
W=F,C	{C,D}
W=F,C,D	{D,A}
W=F,C,D,C	{A,C}
W=F,C,D,A,C	{C,E}
W=F,C,D,A,C,E	{E,A}

Example Cont...

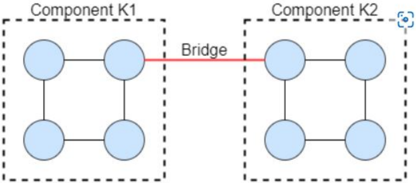
Steps 6, 7, 8, and 9: Only one way to go at each step.



Copy 1 Copy 2

Current Path	Next edge
W=F	{F,C}
W=F,C	{C,D}
W=F,C,D	{D,A}
W=F,C,D,C	{A,C}
W=F,C,D,A,C	{C,E}
W=F,C,D,A,C,E	{E,A}
W=F,C,D,A,C,E,A	{A,B}
W=F,C,D,A,C,E,A,B	{B,D}
W=F,C,D,A,C,E,A,B,D	{D,F}
W=F,C,D,A,C,E,A,B,D,F	–

Q:Perform Fleury's algorithm on the following graph



Q:Perform Fleury's algorithm on the following graph