Graph Theory

Varying Applications (examples)

- Computer networks
- Distinguish between two chemical compounds with the same molecular formula but different structures
- Solve shortest path problems between cities

Topics Covered

- Definitions
- Types
- Terminology
- Representation
- Sub-graphs
- Connectivity
- Hamilton and Euler definitions
- Shortest Path
- Planar Graphs
- Graph Coloring

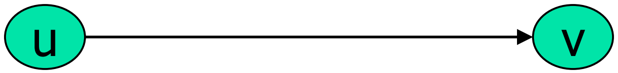
Definitions - Graph

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

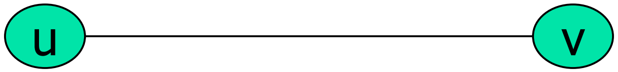
Representation: Graph G =(V, E) consists set of vertices denoted by V, or by V(G) and set of edges E, or E(G)

Definitions – Edge Type

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

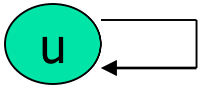


Undirected: Unordered pair of vertices. Represented as {u, v}. Disregards any sense of direction and treats both end vertices interchangeably.



Definitions – Edge Type

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

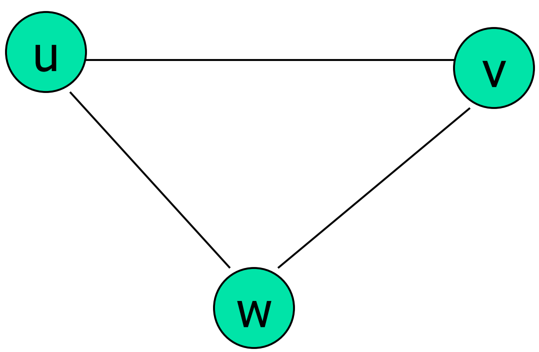


- **Multiple Edges:** Two or more edges joining the same pair of vertices.

Definitions – Graph Type

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

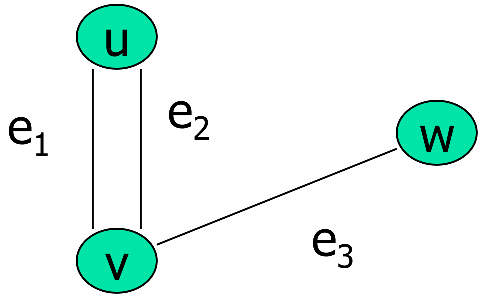
Representation Example: G(V, E), V = {u, v, w}, E = {{u, v}, {v, w}, {u, w}}



Definitions – Graph Type

Multigraph: G(V,E), consists of set of vertices V, set of Edges E and a function f from E to {{u, v}| u, v ϵ V, u ≠ v}. The edges e1 and e2 are called multiple or parallel edges if f (e1) = f (e2).

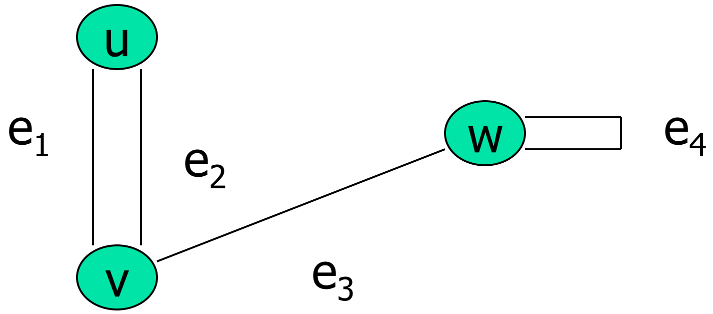
Representation Example: V = {u, v, w}, E = {e_1, e_2, e_3}



Definitions – Graph Type

Pseudograph: G(V,E), consists of set of vertices V, set of Edges E and a function F from E to {{u, v}| u, v ϵ V}. Loops allowed in such a graph.

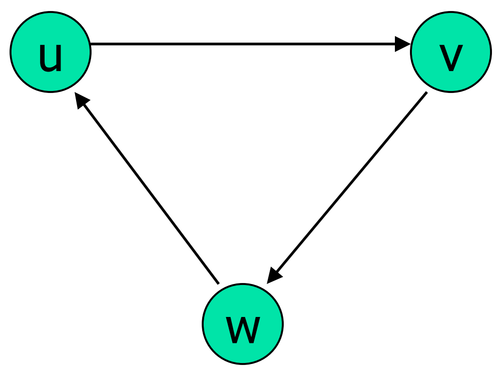
Representation Example: V = {u, v, w}, E = {e_1, e_2, e_3, e_4}



Definitions – Graph Type

Directed Graph: G(V, E), set of vertices V, and set of Edges E, that are ordered pair of elements of V (directed edges)

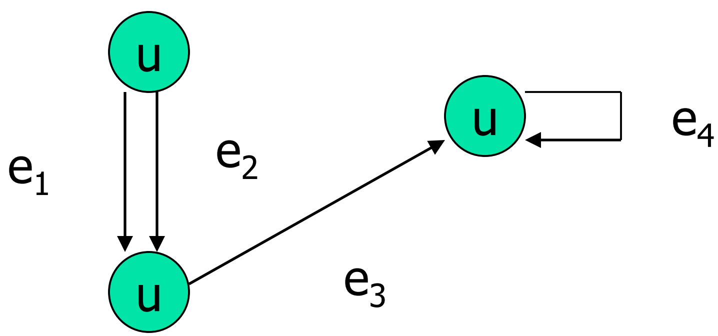
Representation Example: G(V, E), V = {u, v, w}, E = {(u, v), (v, w), (w, u)}



Definitions – Graph Type

Directed Multigraph: G(V,E), consists of set of vertices V, set of Edges E and a function f from E to {{u, v}| u, v ϵ V}. The edges e1 and e2 are multiple edges if f(e1) = f(e2)

Representation Example: V = {u, v, w}, E = {e_1, e_2, e_3, e_4}



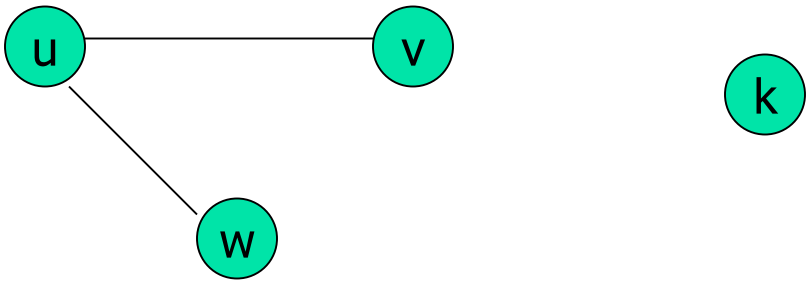
Definitions – Graph Type

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

Terminology – Undirected graphs

- u and v are **adjacent** if {u, v} is an edge, e is called **incident** with u and v. u and v are called **endpoints** of {u, v}
- **Degree of Vertex (deg (v)):** the number of edges incident on a vertex. A loop contributes twice to the degree (why?).
- **Pendant Vertex:** deg (v) =1
- **Isolated Vertex:** deg (v) = 0

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

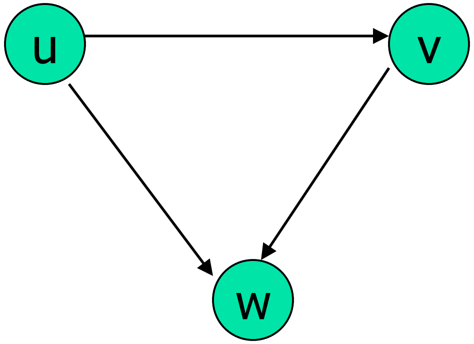


Terminology – Directed graphs

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

Note: A loop contributes 1 to both in-degree and out-degree (why?)

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



Theorems: Undirected Graphs

Theorem 1

The Handshaking theorem:

$$2e = \sum_{v \in V} \deg(v)$$

(why?) Every edge connects 2 vertices

Theorems: Undirected Graphs

Theorem 2:

An undirected graph has even number of vertices with odd degree

Pr *oof* $V1$ is the set of even degree vertices and V2 refers to odd degree vertices

$$2e = \sum_{v \in V} \deg(v) = \sum_{u \in V_1} \deg(u) + \sum_{v \in V_2} \deg(v)$$

$\Rightarrow$ deg (u) is even for $u \in V_1$,

$\Rightarrow$ The first term in the right hand side of the last equality is even.

$\Rightarrow$ The sum of the last two terms on the right hand side of

the last equality is even since sum is 2e.

Hence second term is also even

$\Rightarrow$ second term $\sum_{v \in V_2} \deg(v) = even$

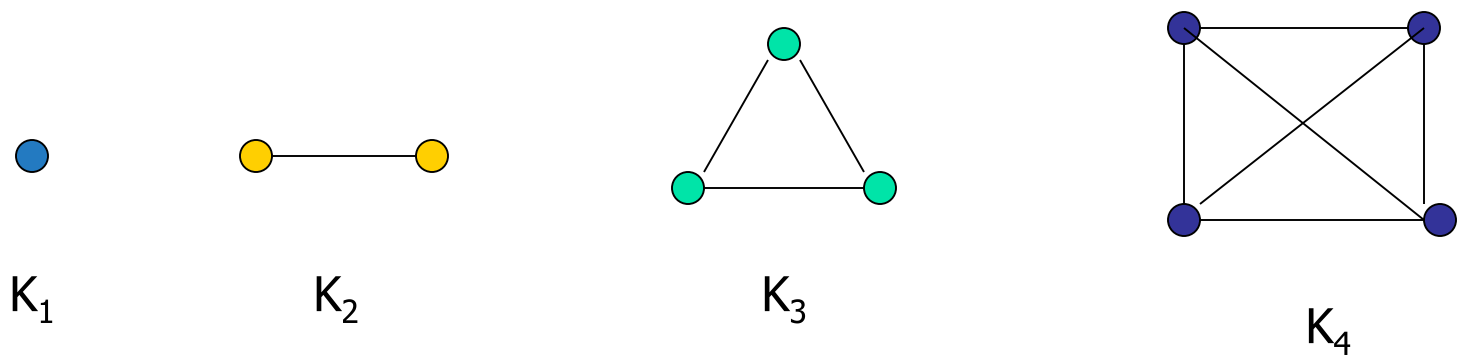
Theorems: directed Graphs

- **<u>Theorem 3:</u>** $\sum$ **deg $^{+}$ (u) =** $\sum$ **deg $^{-}$ (u) = |E|**

Simple graphs – special cases

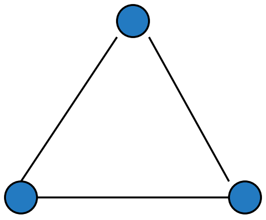
- **Complete graph:** K_n, is the simple graph that contains exactly one edge between each pair of distinct vertices.

  Representation Example: K_1, K_2, K_3, K_4

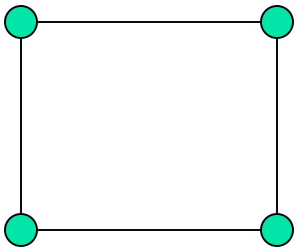


Simple graphs – special cases

- **Cycle:** C_n, $n \geq 3$ consists of n vertices v_1, v_2, v_3 … v_n and edges $\{v_1, v_2\}$, $\{v_2, v_3\}$, $\{v_3, v_4\}$ … $\{v_{n-1}, v_n\}$, $\{v_n, v_1\}$
  Representation Example: C_3, C_4



C_3

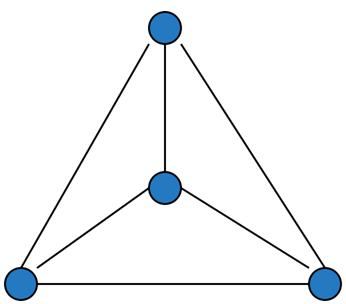


C_4

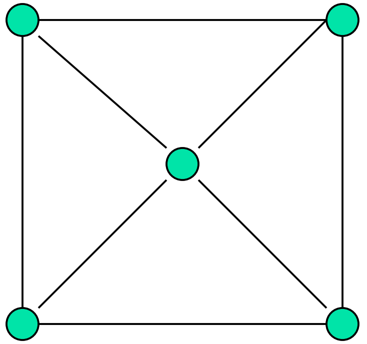
Simple graphs – special cases

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

  Representation Example: W_3, W_4



W_3

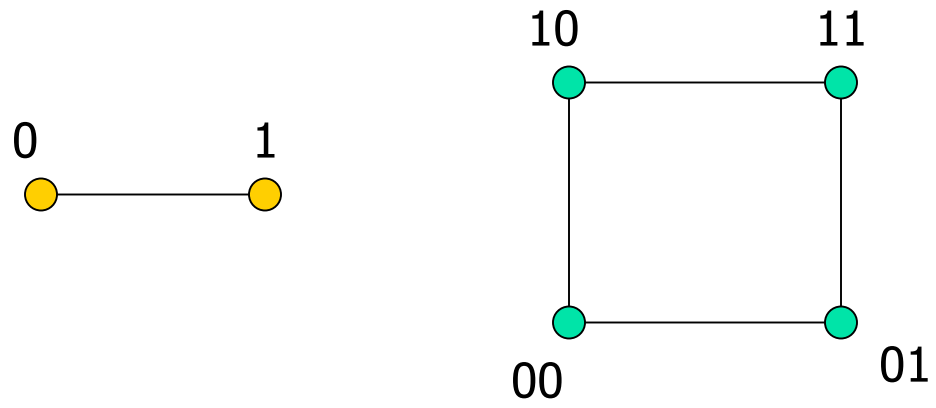


W_4

Simple graphs – special cases

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

  Representation Example: Q_1, Q_2



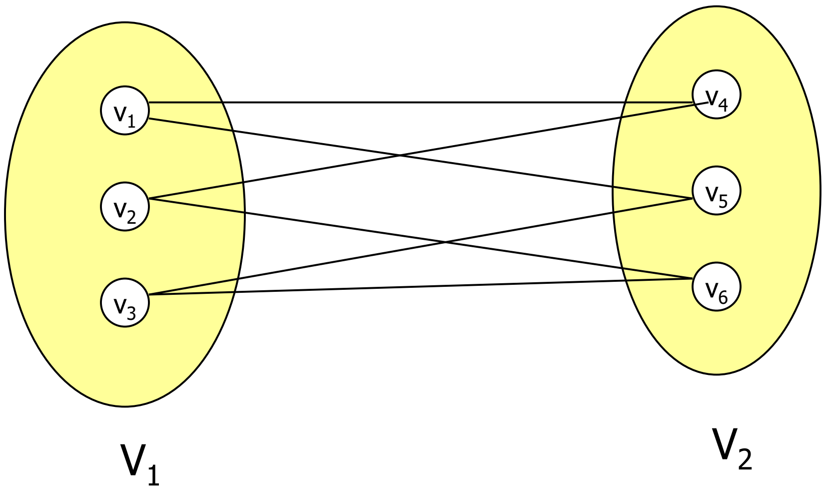
Q_1 Q_2

Bipartite graphs

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

  Application example: e-commerce

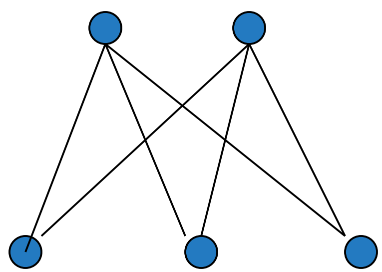
  Representation example: $V_1 = \{v_1, v_2, v_3\}$ and $V_2 = \{v_4, v_5, v_6\}$,



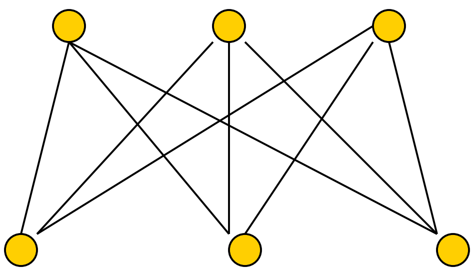
Complete Bipartite graphs

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

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



$K_{2,3}$



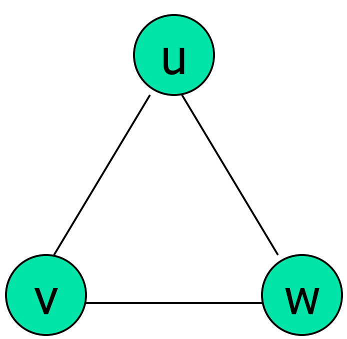
$K_{3,3}$

Subgraphs

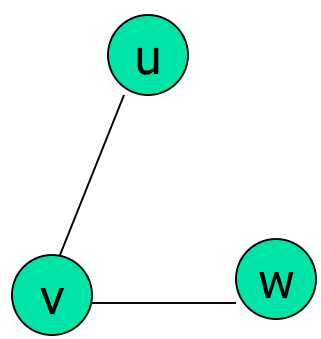
- A subgraph of a graph G = (V, E) is a graph H =(V′, E′) where V′ is a subset of V and E′ is a subset of E

  Application example: solving sub-problems within a graph

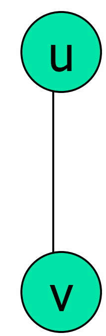
  Representation example: V = {u, v, w}, E = ({u, v}, {v, w}, {w, u}}, H_1 , H_2



G



H_1

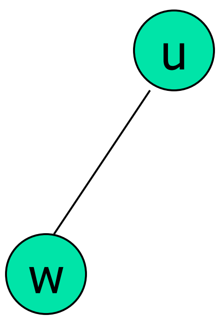


H_2

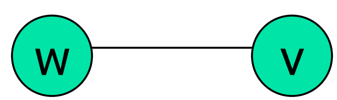
Subgraphs

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

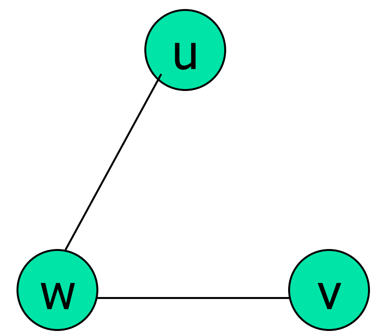
Representation example: V1 = {u, w}, E1 = {{u, w}}, V2 = {w, v}, E1 = {{w, v}}, V = {u, v ,w}, E = {{{u, w}, {{w, v}}



G1



G2



G

Representation

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

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

Representation- Incidence Matrix

- G = (V, E) be an unditected graph. Suppose that v_1, v_2, v_3, …, v_n are the vertices and e_1, e_2, …, e_m are the edges of G. Then the incidence matrix with respect to this ordering of V and E is the nx m matrix M = [m_{ij}], where

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

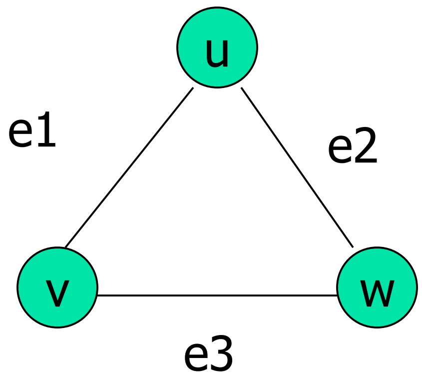
Can also be used to represent :

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

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

Representation- Incidence Matrix

- Representation Example: G = (V, E)



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

Representation- Adjacency Matrix

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

For undirected graph

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

- **For directed graph**

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

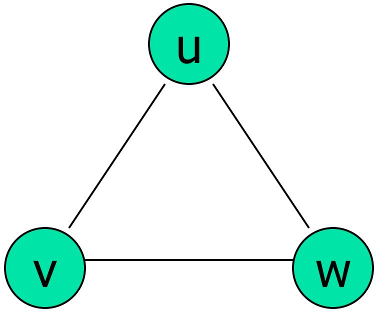
- This makes it easier to find subgraphs, and to reverse graphs if needed.

Representation- Adjacency Matrix

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

Representation- Adjacency Matrix

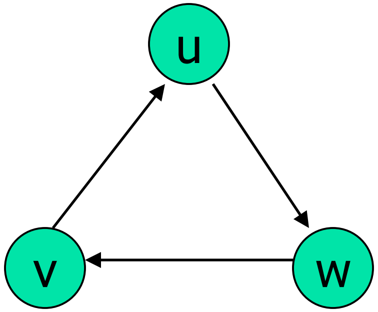
- Example: Undirected Graph G (V, E)



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

Representation- Adjacency Matrix

- Example: directed Graph G (V, E)

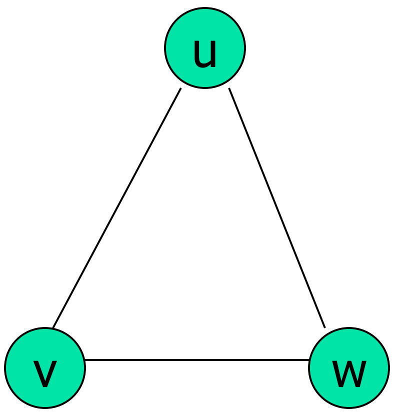


	v	u	w
v	0	1	0
u	0	0	1
w	1	0	0

Representation- Adjacency List

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

Example: undirectd graph G (V, E)



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

Graph - Isomorphism

- G1 = (V1, E2) and G2 = (V2, E2) are isomorphic if:
- There is a one-to-one and onto 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

Application Example:

In chemistry, to find if two compounds have the same structure

Graph - Isomorphism

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

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

Connectivity

- Basic Idea: In a Graph Reachability among vertices by traversing the edges

  Application Example:

  - In a city to city road-network, if one city can be reached from another city.

  - Problems if determining whether a message can be sent between two

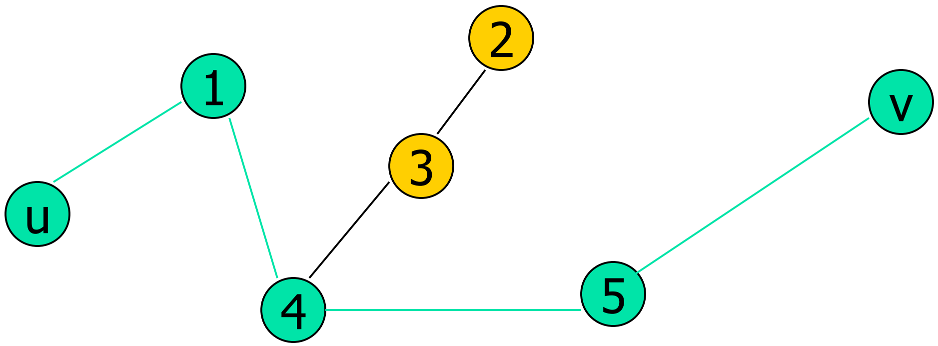
    computer using intermediate links

  - Efficiently planning routes for data delivery in the Internet

Connectivity – Path

A **Path** is a sequence of edges that begins at a vertex of a graph and travels along edges of the graph, always connecting pairs of adjacent vertices.

Representation example: G = (V, E), Path P represented, from u to v is {{u, 1}, {1, 4}, {4, 5}, {5, v}}



Connectivity – Path

Definition for Directed Graphs

A **Path** of length n (> 0) from u to v in G is a sequence of n edges e_1, e_2 , e_3, …, e_n of G such that f (e_1) = (x_o, x_1), f (e_2) = (x_1, $x_{2)}$, …, f (e_n) = (x_{n-1}, $x_{n)}$, where x_0 = u and x_n = v. A path is said to pass through x_0, x_1, …, x_n or traverse e_1, e_2 , e_3, …, e_n

For Simple Graphs, sequence is x_0, x_1, …, x_n

In directed multigraphs when it is not necessary to distinguish between their edges, we can use sequence of vertices to represent the path

Circuit/Cycle: u = v, length of path > 0

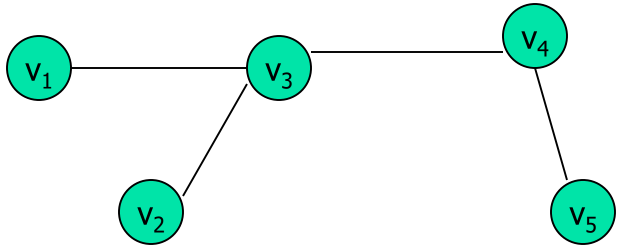
Simple Path: does not contain an edge more than once

Connectivity – Connectedness

Undirected Graph

An undirected graph is connected if there exists is a simple path between every pair of vertices

Representation Example: G (V, E) is connected since for $V = \{v_1, v_2, v_3, v_4, v_5\}$, there exists a path between $\{v_i, v_j\}$, $1 \leq i, j \leq 5$

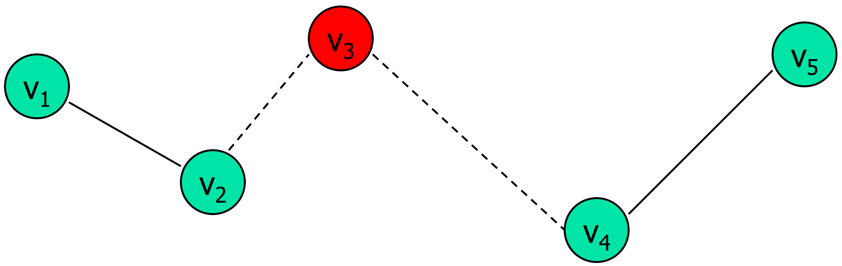


Connectivity – Connectedness

Undirected Graph

- **Articulation Point (Cut vertex):** removal of a vertex produces a subgraph with more connected components than in the original graph. The removal of a cut vertex from a connected graph produces a graph that is not connected
- **Cut Edge:** An edge whose removal produces a subgraph with more connected components than in the original graph.

  Representation example: G (V, E), v_3 is the articulation point or edge $\{v_2, v_3\}$, the number of connected components is 2 (> 1)



Connectivity – Connectedness

Directed Graph

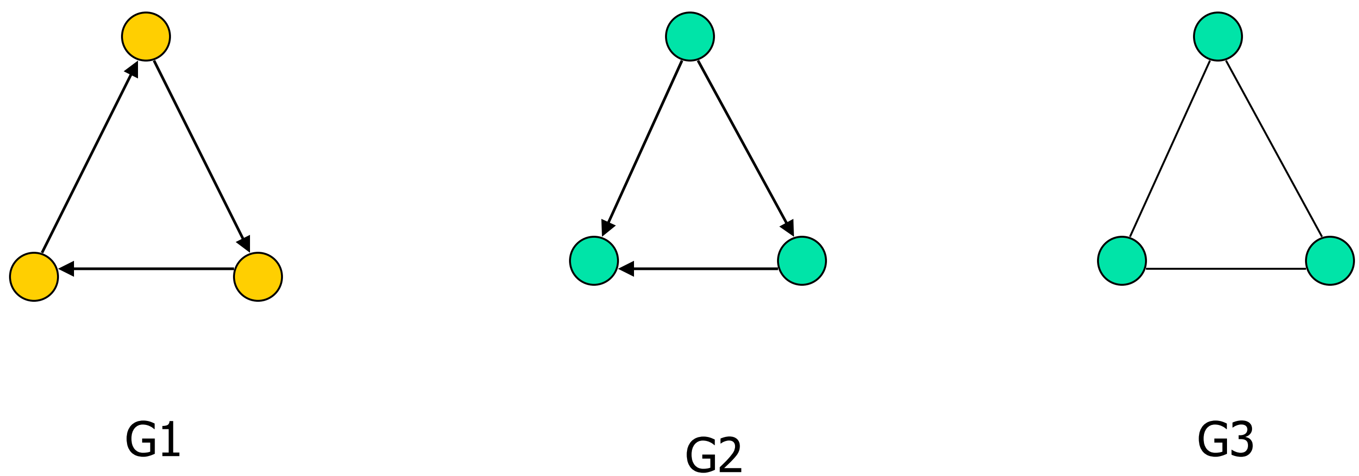
- A directed graph is **strongly connected** if there is a path from a to b and from b to a whenever a and b are vertices in the graph
- A directed graph is **weakly connected** if there is a (undirected) path between every two vertices in the underlying undirected path

A strongly connected Graph can be weakly connected but the vice-versa is not true (why?)

Connectivity – Connectedness

Directed Graph

Representation example: G1 (Strong component), G2 (Weak Component), G3 is undirected graph representation of G2 or G1

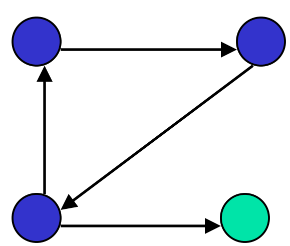


Connectivity – Connectedness

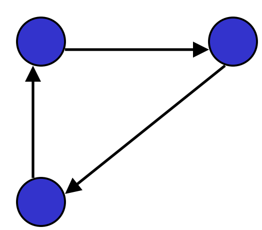
- **Directed Graph**

  Strongly connected Components: subgraphs of a Graph G that are strongly connected

  Representation example: G1 is the strongly connected component in G



G



G1

Counting Paths

- **Theorem:** Let G be a graph with adjacency matrix A with respect to the ordering v_1, v_2, …, V_n (with directed on undirected edges, with multiple edges and loops allowed). The number of different paths of length r from Vi to Vj, where r is a positive integer, equals the $(i, j)^{th}$ entry of (adjacency matrix) $A^{r.}$

  Proof: By Mathematical Induction.

  <u>Base Case:</u> For the case N = 1, a_{ij} =1 implies that there is a path of length 1. This is true since this corresponds to an edge between two vertices.

  We assume that theorem is true for N = r and prove the same for N = r +1. Assume that the $(i, j)^{th}$ entry of A^r is the number of different paths of length r from v_i to v_j. By induction hypothesis, b_{ik} is the number of paths of length r from v_i to v_k.

Counting Paths

<u>Case r +1:</u> In $A^{r+1} = A^r . A$,

The $(i, j)^{th}$ entry in A^{r+1} , $b_{i1}a_{1j} + b_{i2}\, a_{2j} + \ldots + b_{in}\, a_{nj}$

where b_{ik} is the $(i, j)^{th}$ entry of A^r.

By induction hypothesis, b_{ik} is the number of paths of length r from v_i to v_k.

The $(i, j)^{th}$ entry in A^{r+1} corresponds to the length between i and j and the length is r+1. This path is made up of length r from v_i to v_k and of length from v_k to vj. By product rule for counting, the number of such paths is $b_{ik*}\, a_{kj}$ The result is $b_{i1}a_{1j} + b_{i2}\, a_{2j} + \ldots + b_{in}\, a_{nj}$,the desired result.

Counting Paths

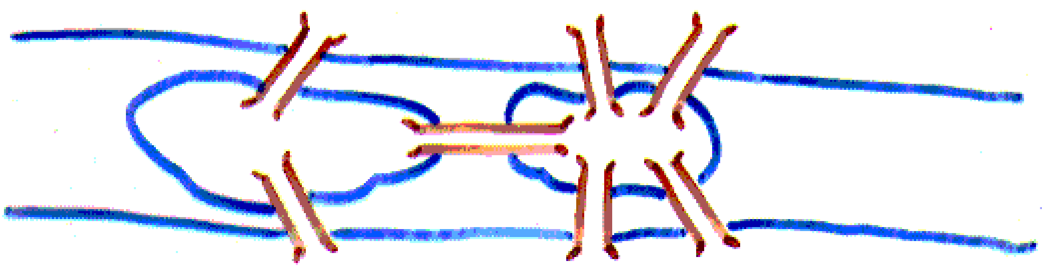
a ------- b
| |
| |
c -------d

$$A = \begin{matrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{matrix} \qquad A^4 = \begin{matrix} 8 & 0 & 0 & 8 \\ 0 & 8 & 8 & 0 \\ 0 & 8 & 8 & 0 \\ 8 & 0 & 0 & 8 \end{matrix}$$

Number of paths of length 4 from a to d is (1,4) th entry of $A^4 = 8$.

The Seven Bridges of Königsberg, Germany

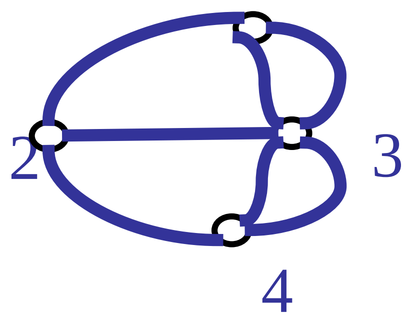
- The residents of Königsberg, Germany, wondered if it was possible to take a walking tour of the town that crossed each of the seven bridges over the Presel river exactly once. Is it possible to start at some node and take a walk that uses each edge exactly once, and ends at the starting node?



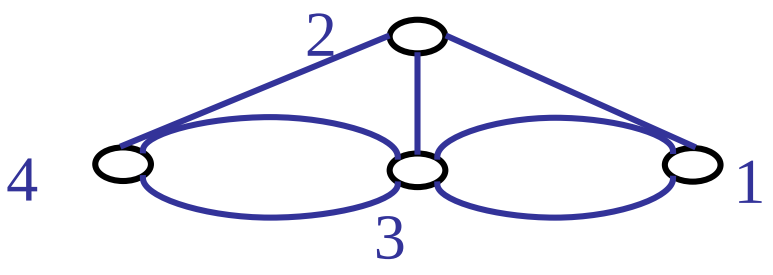
The Seven Bridges of Königsberg, Germany

You can redraw the original picture as long as for every edge between nodes *i* and *j* in the original you put an edge between nodes *i* and *j* in the redrawn version (and you put no other edges in the redrawn version).

Original:

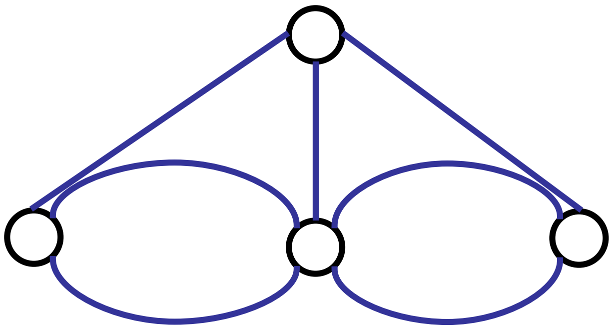


Redrawn:



The Seven Bridges of Königsberg, Germany

Euler:



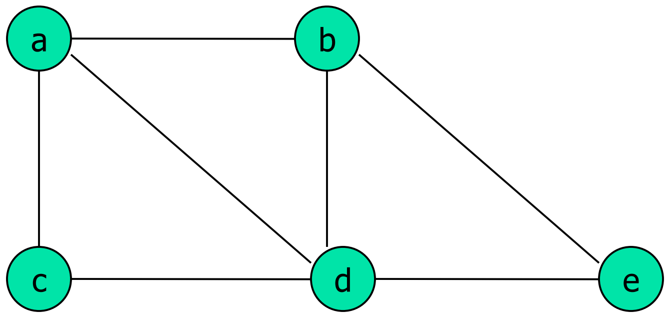
- Has no tour that uses each edge exactly once.
- (Even if we allow the walk to start and finish in different places.)
- Can you see why?

Euler - definitions

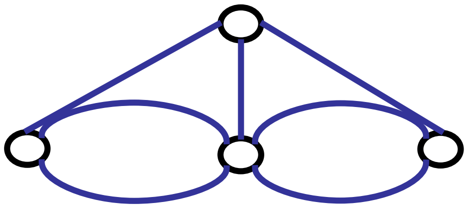
- An **Eulerian path** (**Eulerian trail**, **Euler walk**) in a graph is a path that uses each edge precisely once. If such a path exists, the graph is called **traversable**.

- An **Eulerian cycle** (**Eulerian circuit**, **Euler tour**) in a graph is a cycle that uses each edge precisely once. If such a cycle exists, the graph is called **Eulerian** (also **unicursal**).

- Representation example: G1 has Euler path a, c, d, e, b, d, a, b



The problem in our language:

Show that  is not Eulerian.

In fact, it contains no Euler trail.

Euler - theorems

1. A connected graph G is Eulerian if and only if G is connected and has no vertices of odd degree

2. A connected graph G is has an Euler trail from node *a* to some other node b if and only if G is connected and $a \neq b$ are the only two nodes of odd degree

Euler – theorems (=>)

Assume G has an Euler trail T from node a to node b (a and b not necessarily distinct).

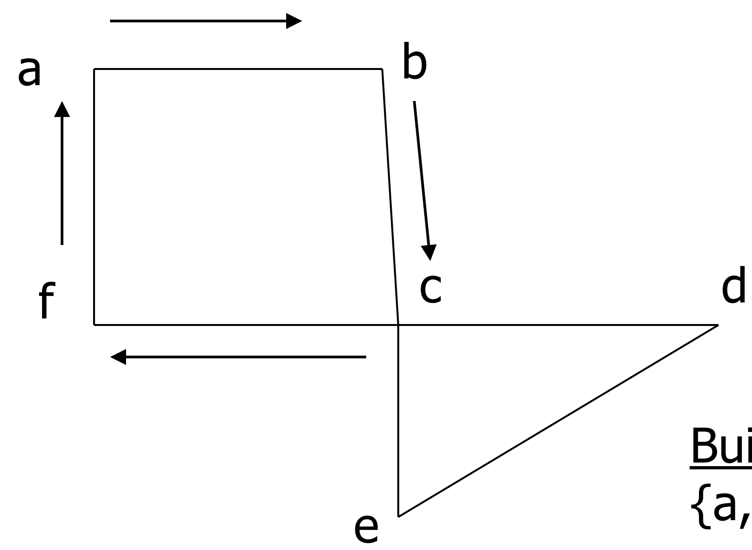
For every node besides a and b, T uses an edge to exit for each edge it uses to enter. Thus, the degree of the node is even.

1. If $a = b$, then a also has even degree. → Euler circuit

2. If $a \neq b$, then a and b both have odd degree. → Euler path

Euler - theorems

1. A connected graph G is Eulerian if and only if G is connected and has no vertices of odd degree

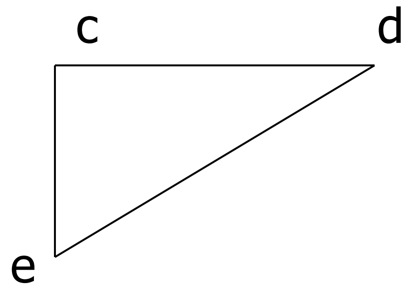


<u>Building a simple path</u>:
{a,b}, {b,c}, {c,f}, {f,a}

Euler circuit constructed if all edges are used. True here?

Euler - theorems

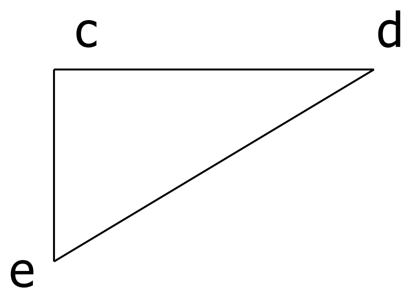
1. A connected graph G is Eulerian if and only if G is connected and has no vertices of odd degree



<u>Delete the simple path</u>:
{a,b}, {b,c}, {c,f}, {f,a}

Euler - theorems

1. A connected graph G is Eulerian if and only if G is connected and has no vertices of odd degree



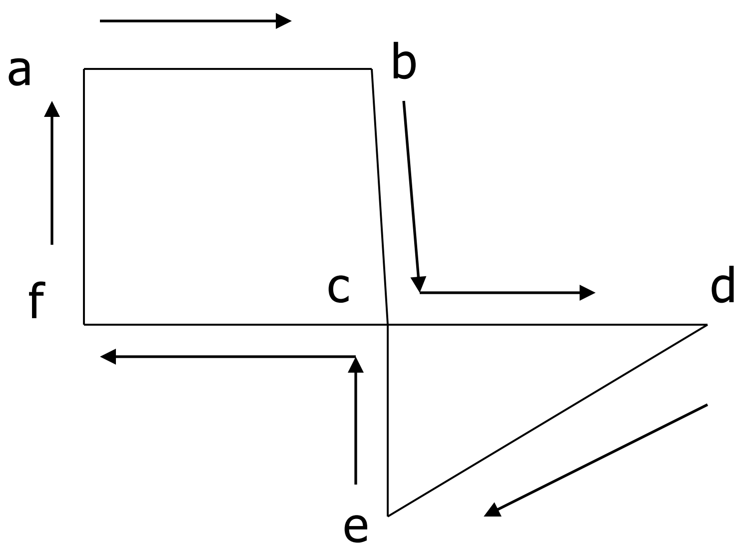
Constructed subgraph may not be connected.

C has even degree.

Start at c and take a walk:
{c,d}, {d,e}, {e,c}

Euler - theorems

1. A connected graph G is Eulerian if and only if G is connected and has no vertices of odd degree



<u>“Splice” the circuits in the 2 graphs</u>:

{a,b}, {b,c}, {c,f}, {f,a}

“+”

{c,d}, {d,e}, {e,c}

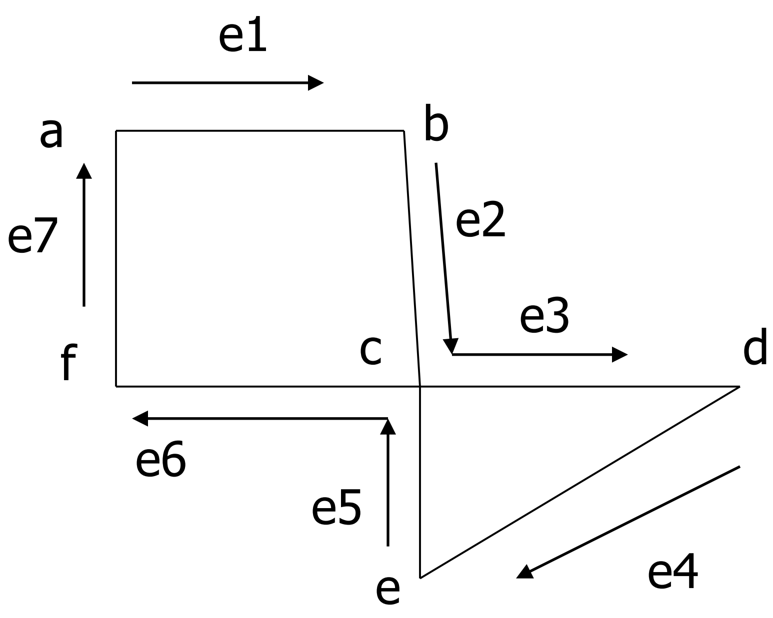
“=“

{a,b}, {b,c}, {c,d}, {d,e}, {e,c}, {c,f} {f,a}

Euler Circuit

1. Circuit C := a circuit in G beginning at an arbitrary vertex v.
    1. Add edges successively to form a path that returns to this vertex.
2. H := G – above circuit C
3. While H has edges
    1. Sub-circuit *sc* := a circuit that begins at a vertex in H that is also in C (e.g., vertex "c")
    2. H := H – *sc* (- all isolated vertices)
    3. Circuit := circuit C "spliced" with sub-circuit *sc*
4. Circuit C has the Euler circuit.

Representation- Incidence Matrix



	e_1	e_2	e_3	e4	e_5	e_6	e_7
a	1	0	0	0	0	0	1
b	1	1	0	0	0	0	0
c	0	1	1	0	1	1	0
d	0	0	1	1	0	0	0
e	0	0	0	1	1	0	0
f	0	0	0	0	0	1	1

Assignment 3

- **Write a program to obtain Euler Circuits.**
  - **Input graphs can be Eulerian, no need for checking “non” Euler graphs**
  - **Include a simple user interface to “input” the graph.**
  - **Minimum of 10 edges (no more than 15 edges needed)**
  - **Simple documentation**
  - **Include a sample graph, if needed, to test**
  - **Any programming language**

Hamiltonian Graph

- **Hamiltonian path** (also called *traceable path*) is a path that visits each vertex exactly once.

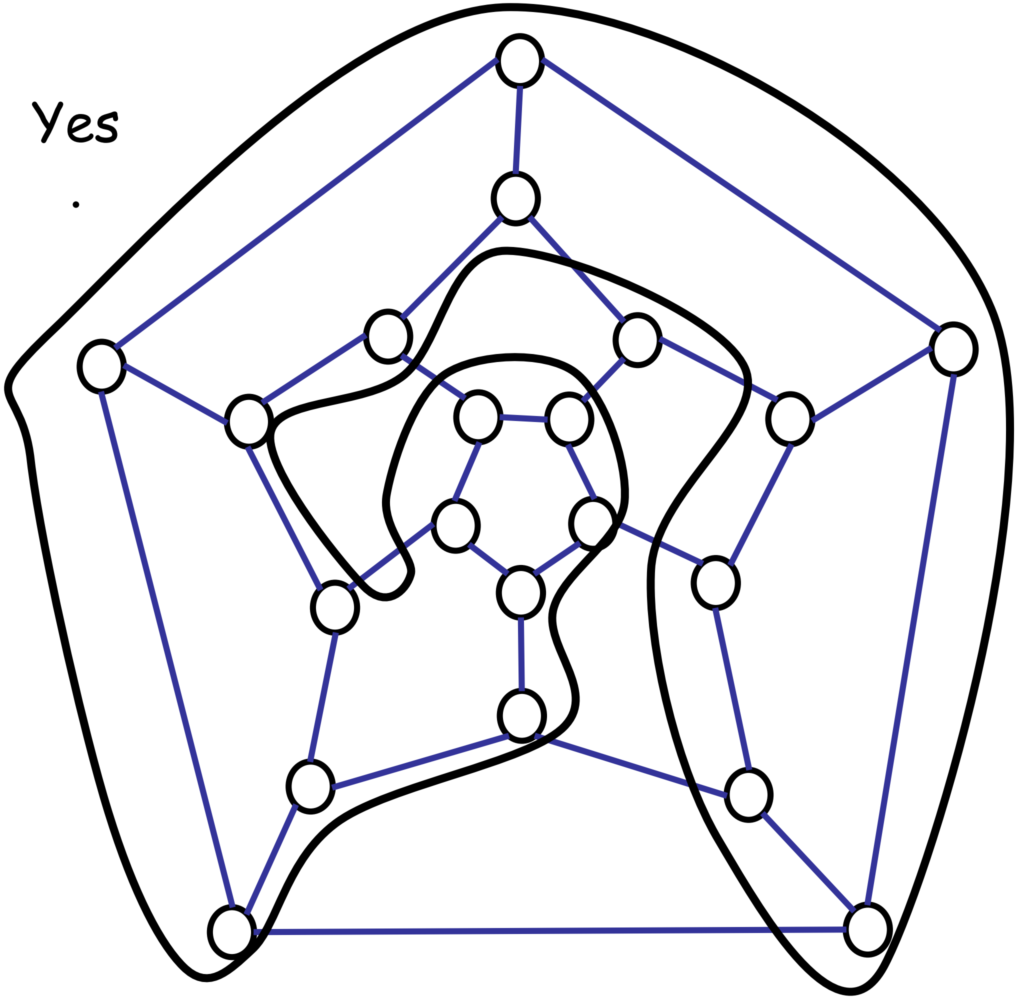
- A **Hamiltonian cycle** (also called *Hamiltonian circuit*, *vertex tour* or *graph cycle*) is a cycle that visits each vertex exactly once (except for the starting vertex, which is visited once at the start and once again at the end).

- A graph that contains a Hamiltonian path is called a **traceable graph**. A graph that contains a Hamiltonian cycle is called a **Hamiltonian graph**. Any Hamiltonian cycle can be converted to a Hamiltonian path by removing one of its edges, but a Hamiltonian path can be extended to Hamiltonian cycle only if its endpoints are adjacent.

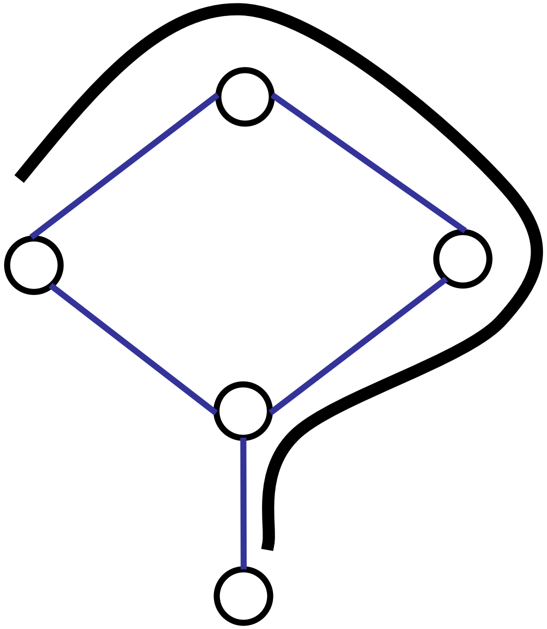
A graph of the vertices of a dodecahedron.

Is it Hamiltonian?

Yes

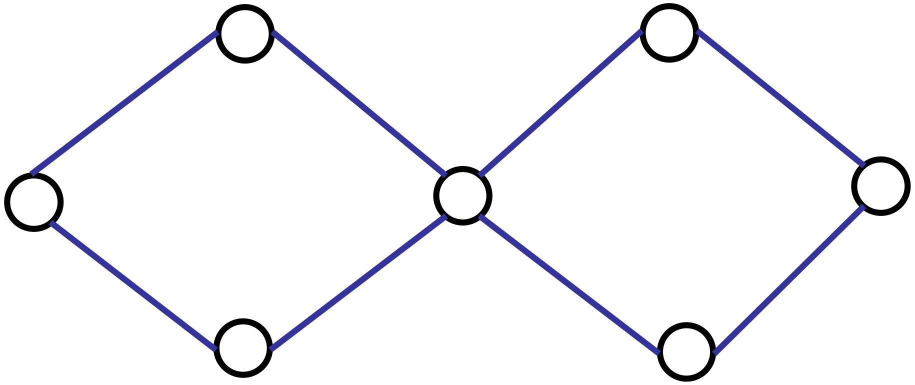


Hamiltonian Graph



This one has a Hamiltonian path, but not a Hamiltonian tour.

Hamiltonian Graph



This one has an Euler tour, but no Hamiltonian path.

Hamiltonian Graph

- Similar notions may be defined for directed graphs, where edges (arcs) of a path or a cycle are required to point in the same direction, i.e., connected tail-to-head.

- The *Hamiltonian cycle problem* or *Hamiltonian circuit problem* in graph theory is to find a Hamiltonian cycle in a given graph. The *Hamiltonian path problem* is to find a Hamiltonian path in a given graph.

- There is a simple relation between the two problems. The Hamiltonian path problem for graph **G** is equivalent to the Hamiltonian cycle problem in a graph **H** obtained from **G** by adding a new vertex and connecting it to all vertices of **G**.

- Both problems are NP-complete. However, certain classes of graphs always contain Hamiltonian paths. For example, it is known that every tournament has an odd number of Hamiltonian paths.

Hamiltonian Graph

- **DIRAC'S Theorem:** if G is a simple graph with n vertices with n ≥ 3 such that the degree of every vertex in G is at least n/2 then G has a Hamilton circuit.

- **ORE'S Theorem:** if G is a simple graph with n vertices with n ≥ 3 such that deg (u) + deg (v) ≥ n fro every pair of nonadjacent vertices u and v in G, then G has a Hamilton circuit.