Graph Theory

Dr. Lalatendu Behera

Varying Applications (examples)

- Computer networks
- Distinguish between two chemical compounds with the same molecular formula but different structures
- Solve shortest path problems between cities
- Scheduling exams and assign channels to television stations

Topics Covered

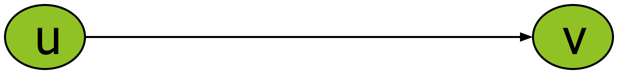
- Definitions
- Types
- Terminology
- Representation
- Sub-graphs
- Connectivity
- Hamilton and Euler definitions
- Shortest Path
- Planar Graphs
- Graph Coloring

Definitions - Graph

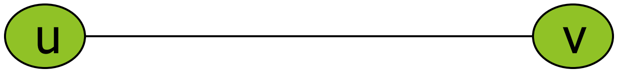
A **simple finite Graph** G = (V, E) consists of a finite nonempty set of *vertices* denoted by V = $\{v_1, v_2, \ldots\}$ and a set of edges E = $\{e_1, e_2, \ldots\}$ of 2-elements subset of V such that each edge e_k is identified with an unordered pair (v_i, v_j) of vertices, i.e., E $\subseteq V^2$, where V^2 = {A ⊆ V | $|A|$=2}

Definitions – Edge Type

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

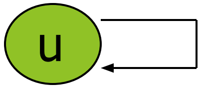


Undirected: Unordered pair of vertices. Represented as {u, v}. Disregards any sense of direction and treats both end vertices interchangeably.



Definitions – Edge Type

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

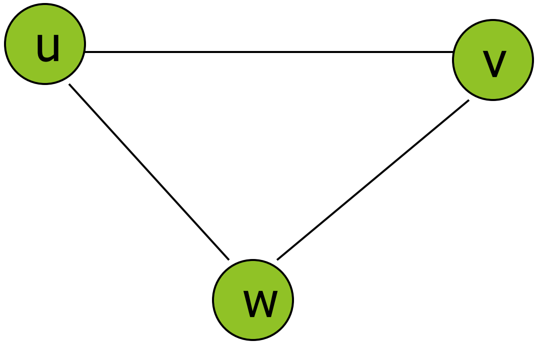


- **Multiple Edges:** Two or more edges joining the same pair of vertices.

Definitions – Graph Type

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

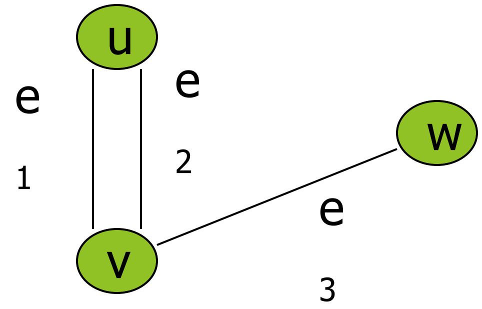
Representation Example: G(V, E), V = {u, v, w}, E = {{u, v}, {v, w}, {u, w}}



Definitions – Graph Type

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

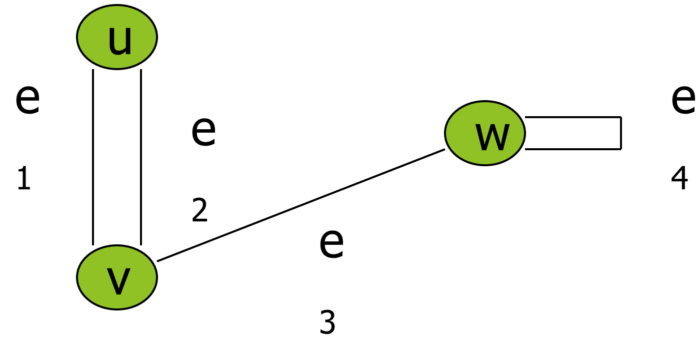
Representation Example: V = {u, v, w}, E = {e_1, e_2, e_3}



Definitions – Graph Type

Pseudograph: A pseudograph is an ordered pair G = (V, E) where V is a finite set and E is a set of edges of the form (e, {u, v}) where u and v are elements of V and no two of the edges in E have the same first coordinate. We call e an edge of G and say that e is incident to u and v. If u = v, then the pair is really just (e, {v}) and we say e as a loop at v.

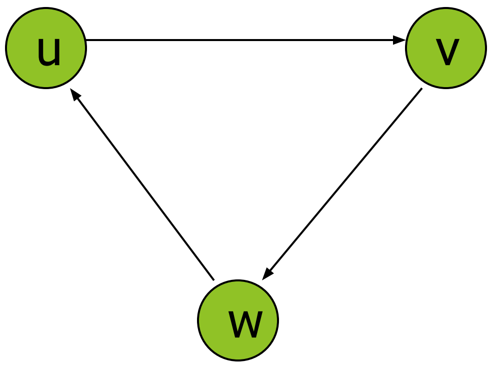
Representation Example: V = {u, v, w}, E = {e_1, e_2, e_3, e_4}



Definitions – Graph Type

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

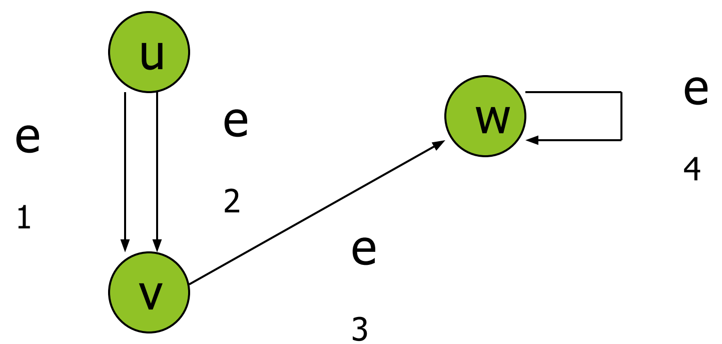
Representation Example: G(V, E), V = {u, v, w}, E = {(u, v), (v, w), (w, u)}



Definitions – Graph Type

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

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



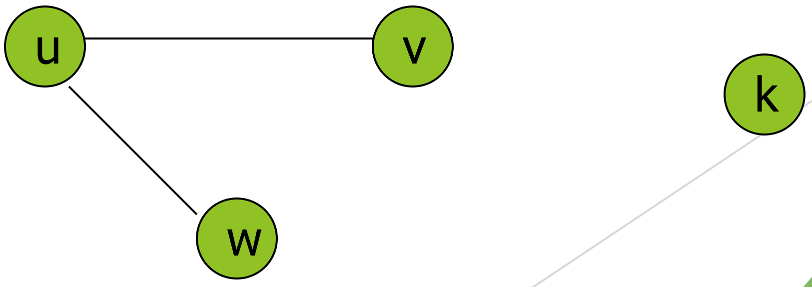
Definitions – Graph Type

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

Terminology – Undirected graphs

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

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

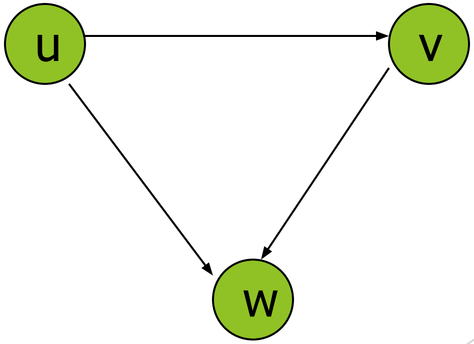


Terminology – Directed graphs

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

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

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



Theorems: Undirected Graphs

<u>Theorem 1</u>

The Handshaking theorem:

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

Every edge connects 2 vertices

You need to think why this is true? The look at the book.

Theorems: Undirected Graphs

Theorem 2

An undirected graph has even number of vertices with odd degree

Proof: V_1 is the set of even degree vertices and V_2 is the set of odd degree vertices

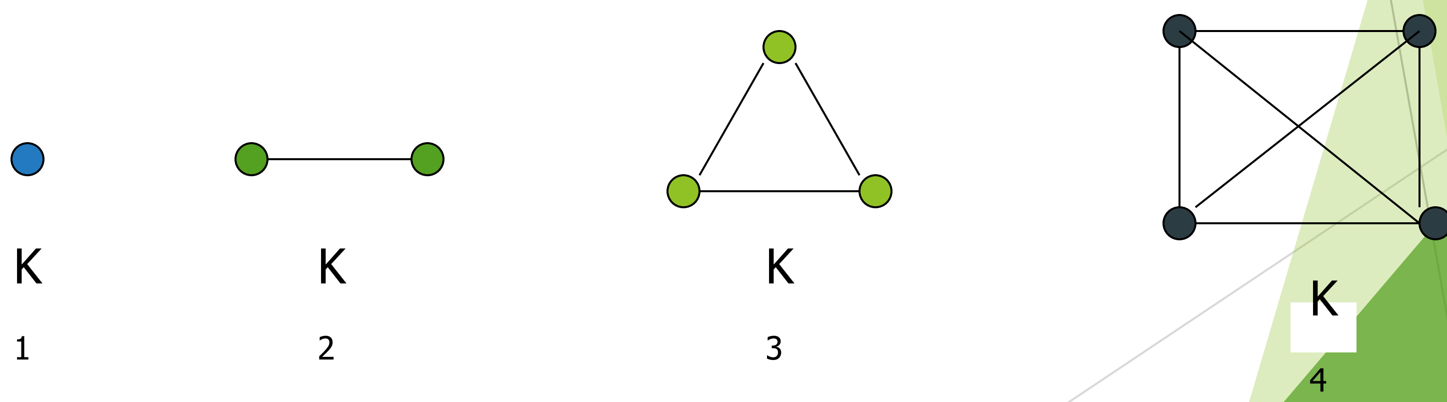
$$2e = \sum_{v \in V} \deg(V) = \sum_{u \in V_1} \deg(u) + \sum_{v \in V_2} \deg(v)$$

We know that $\sum_{u \in V_1} \deg(u)$ is even. The sum of the $\sum_{u \in V_1} \deg(u)$ and $\sum_{v \in V_2} \deg(v)$ is even because the sum is 2e. Hence the second term is also even. So $\sum_{v \in V_2} \deg(v)$ is even.

Simple graphs – special cases

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

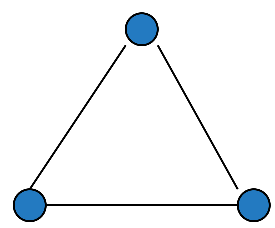
Representation Example: K_1, K_2, K_3, K_4



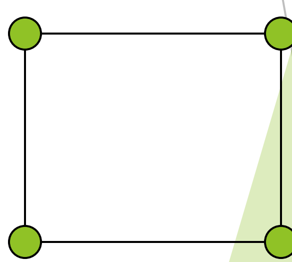
Simple graphs – special cases

- **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

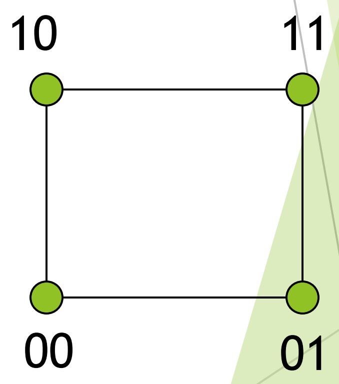
Simple graphs – special cases

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

  Examples: C_1, C_2

C

1



C

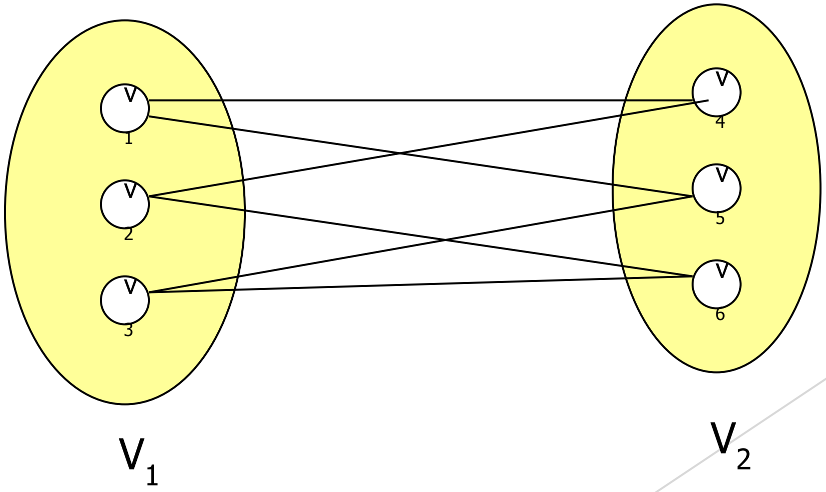
2

Bipartite graphs

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

  Application example: Representing Relations

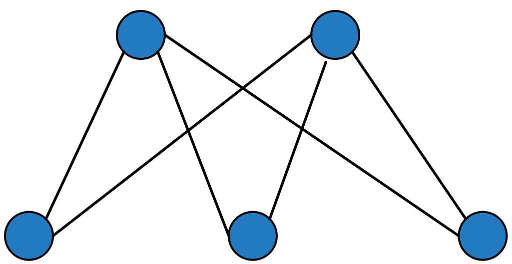
  Representation example: $V_1 = \{v_1, v_2, v_3\}$ and $V_2 = \{v_4, v_5, v_6\}$,



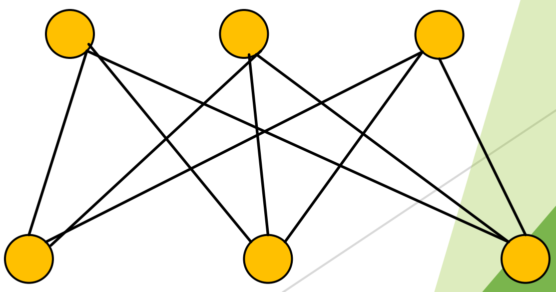
Complete Bipartite graphs

- $K_{m,n}$ is the graph that has its vertex set partitioned 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: $K_{2,3}$, $K_{3,3}$



$K_{2,3}$



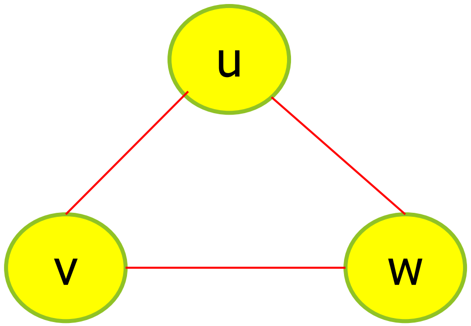
$K_{3,3}$

Subgraphs

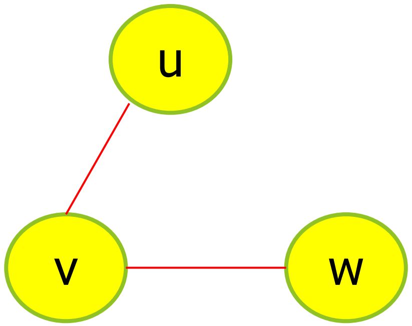
- A subgraph of a graph G = (V, E) is a graph H =(V', E') where V' is a subset of V and E' is a subset of E

Application example: solving sub-problems within a graph

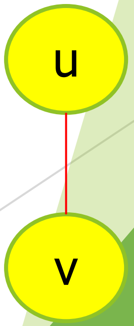
Representation example: V = {u, v, w}, E = ({u, v}, {v, w}, {w, u}}, H1, H2



G



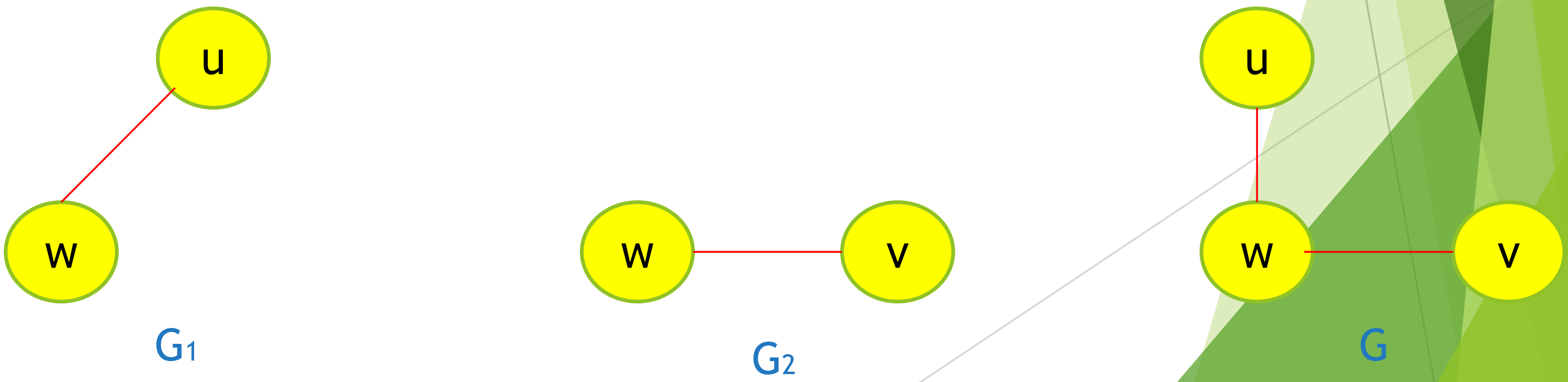
H_1



H_2

Subgraphs

- G = G1 U G2 where in 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}}



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

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

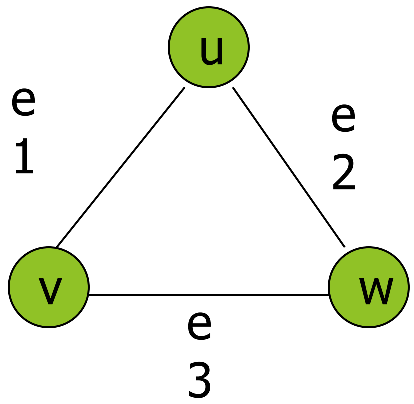
Can also be used to represent :

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 Adjacency Matrix (NxN) A = $[a_{ij}]$

  For undirected graph

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

- **For directed graph**

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

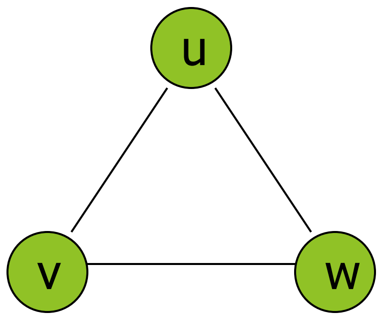
- This makes it easier to find subgraphs, and to reverse graphs if needed.

Representation- Adjacency Matrix

- Adjacency is chosen on the ordering of vertices. Hence, there as are as many as n! such matrices.
- The adjacency matrix of simple graphs are symmetric ($a_{ij} = a_{ji}$)
- 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

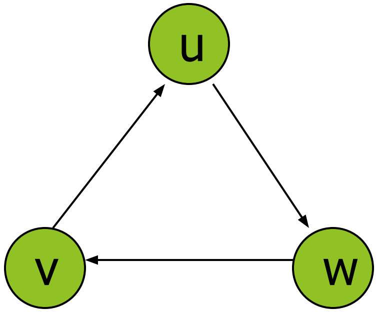
- Example: Undirected Graph G (V, E)



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

Representation- Adjacency Matrix

- Example: directed Graph G (V, E)

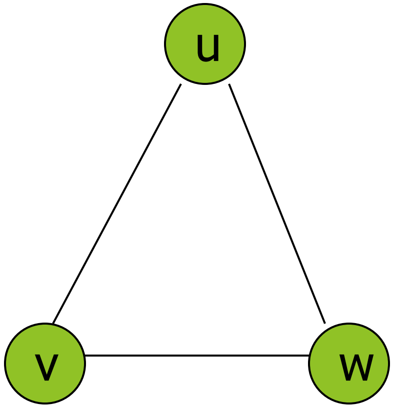


	v	u	w
v	0	1	0
u	0	0	1
w	1	0	0

Representation- Adjacency List

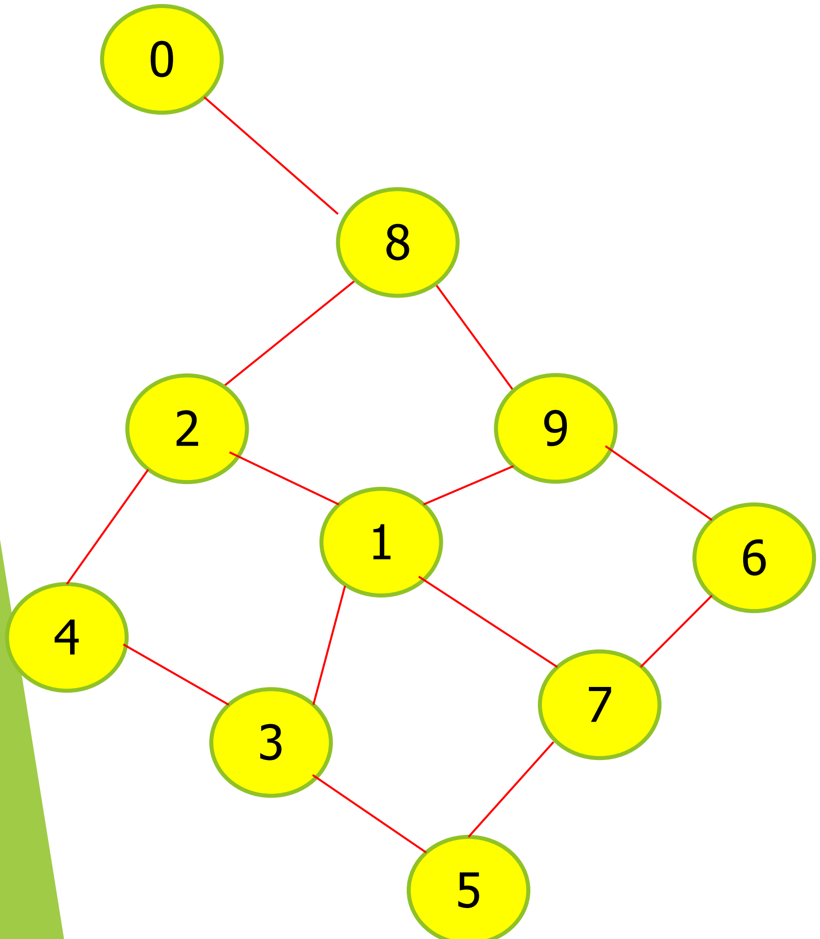
Each node (vertex) has a list of which nodes (vertex) it is adjacent

Example: undirectd graph G (V, E)



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

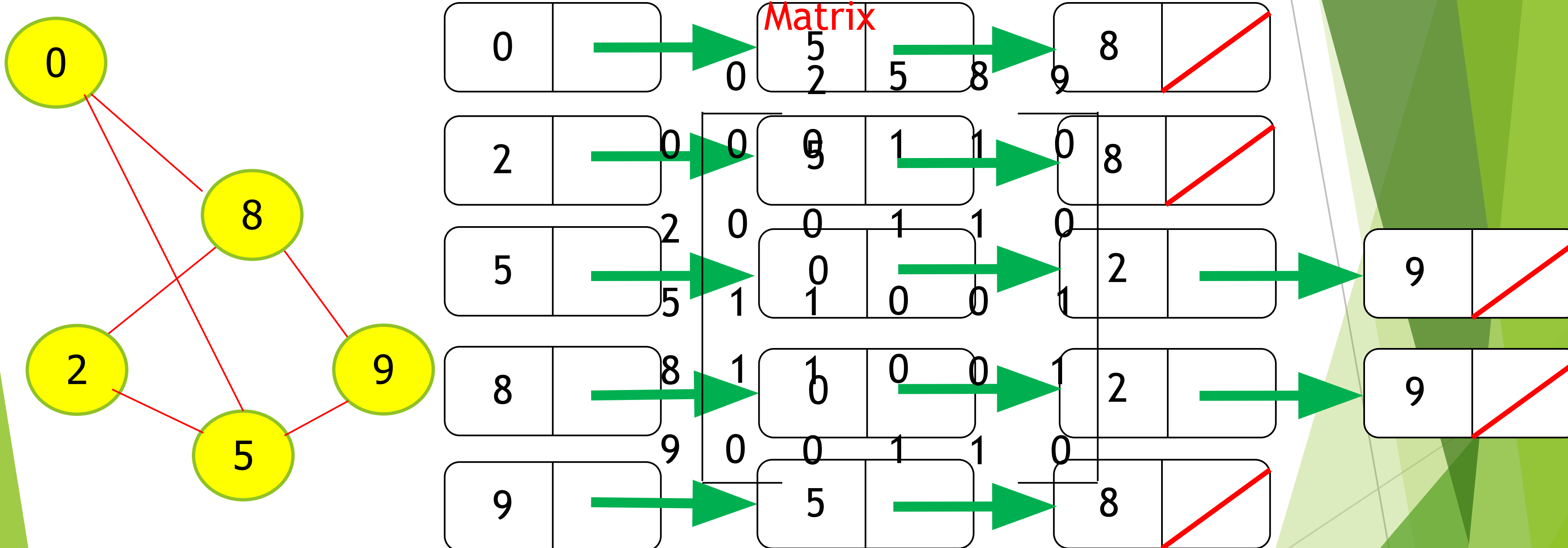
Adjacency List



Adjacency List

0	8
1	2 3 7 9
2	1 4 8
3	1 4 5
4	2 3
5	3 7
6	7 9
7	1 5 6
8	0 2 9
9	1 6 8

Adjacency List



	0	2	5	8	9
0	0	0	1	1	0
2	0	0	1	1	0
5	1	1	0	0	1
8	1	1	0	0	1
9	0	0	1	1	0

Graph -Isomorphism

- G1=(V_1, E_2) and G2=(V_2, E_2) are isomorphic if:
  - There is a one-to one and onto function F from V_1 to V_2 with the property that
    - **a** and **b** are adjacent in G_1 if and only if f(a) and f(b) are adjacent in G_2, for all **a** and **b** in V_1.
  - 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(a)=k, f(b)=l, f(c)=m, f(d)=n

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

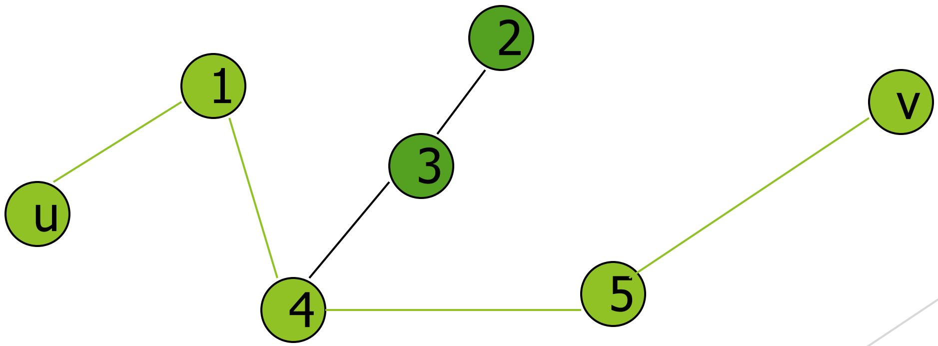
  computer using intermediate links

  - 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 k(>0) from u to v in G is a sequence of k+1 vertices x_1, x_2, … x_k+1 such that (x_i, x_i+1) is an edge for i=1,2,…,k-1.

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

SimplePath: does not contain a vertex more than once

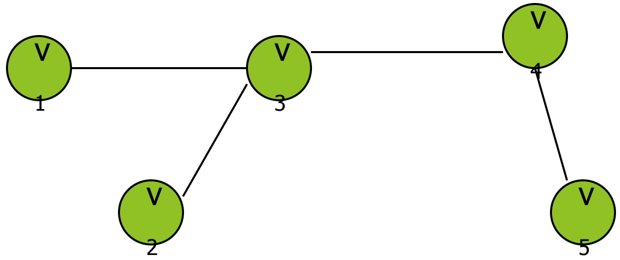
x_1

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 $\{v_i, v_j\}$, $1 \leq i, j \leq 5$

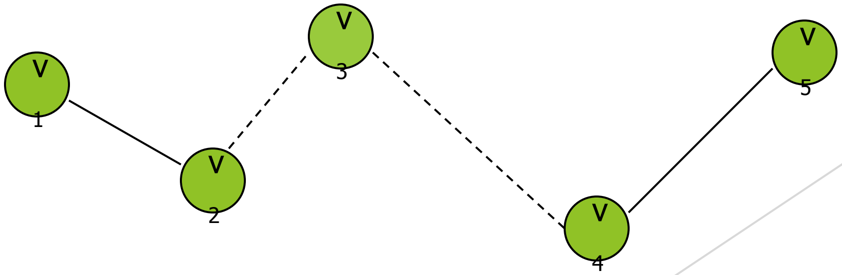


Connectivity – Connectedness

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

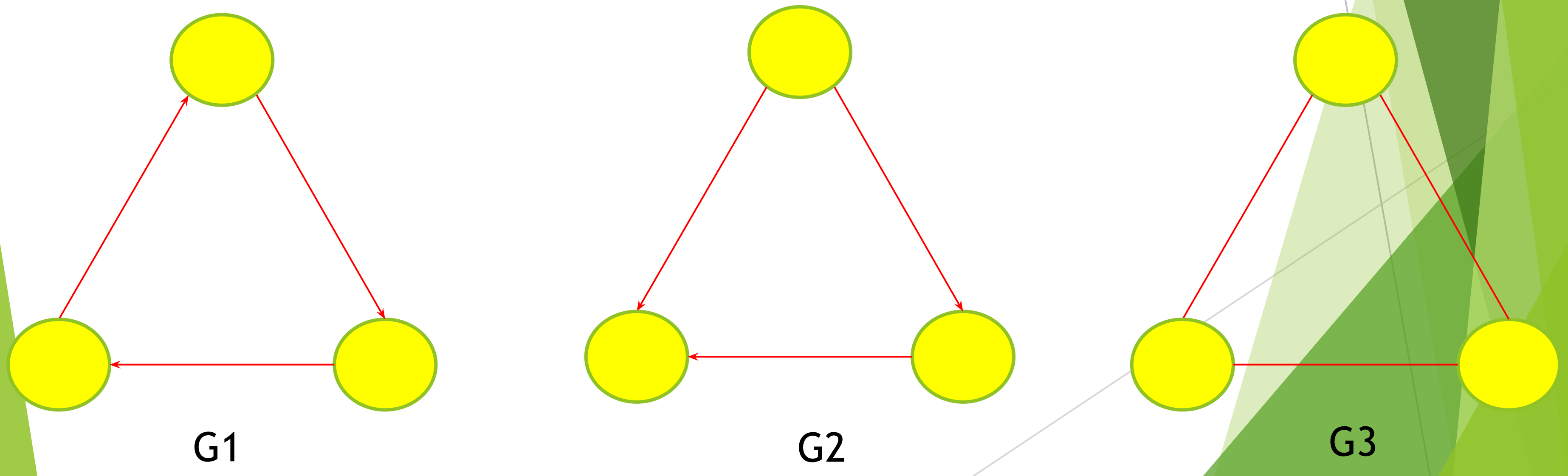
Directed Graph

- 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.

Connectivity – Connectedness

Directed Graph

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

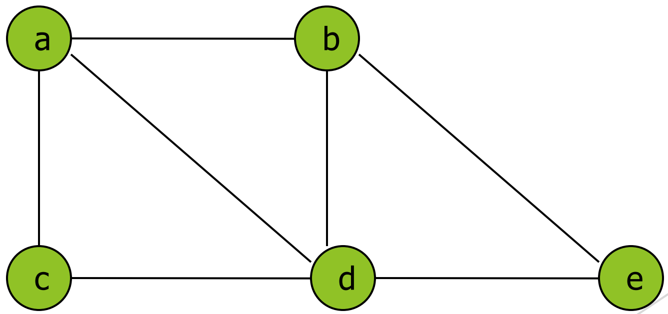


Euler - definitions

- 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

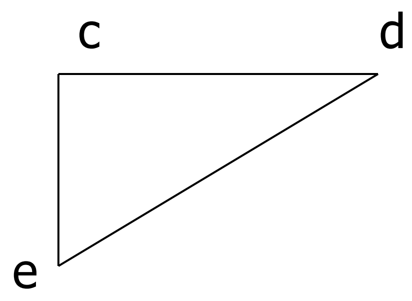


Euler - theorems

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

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.

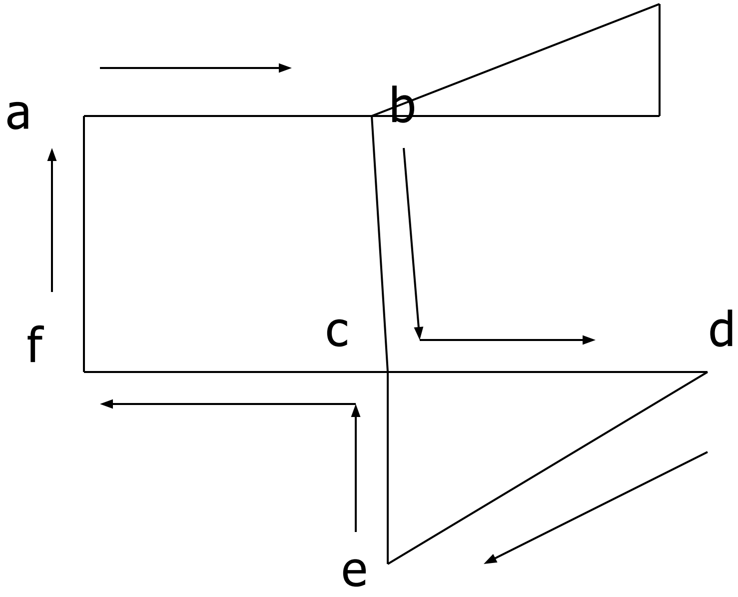
C is the common vertex for this sub-graph with its “parent”.

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



"Splice" the circuits in the 2 graphs:

{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}

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).

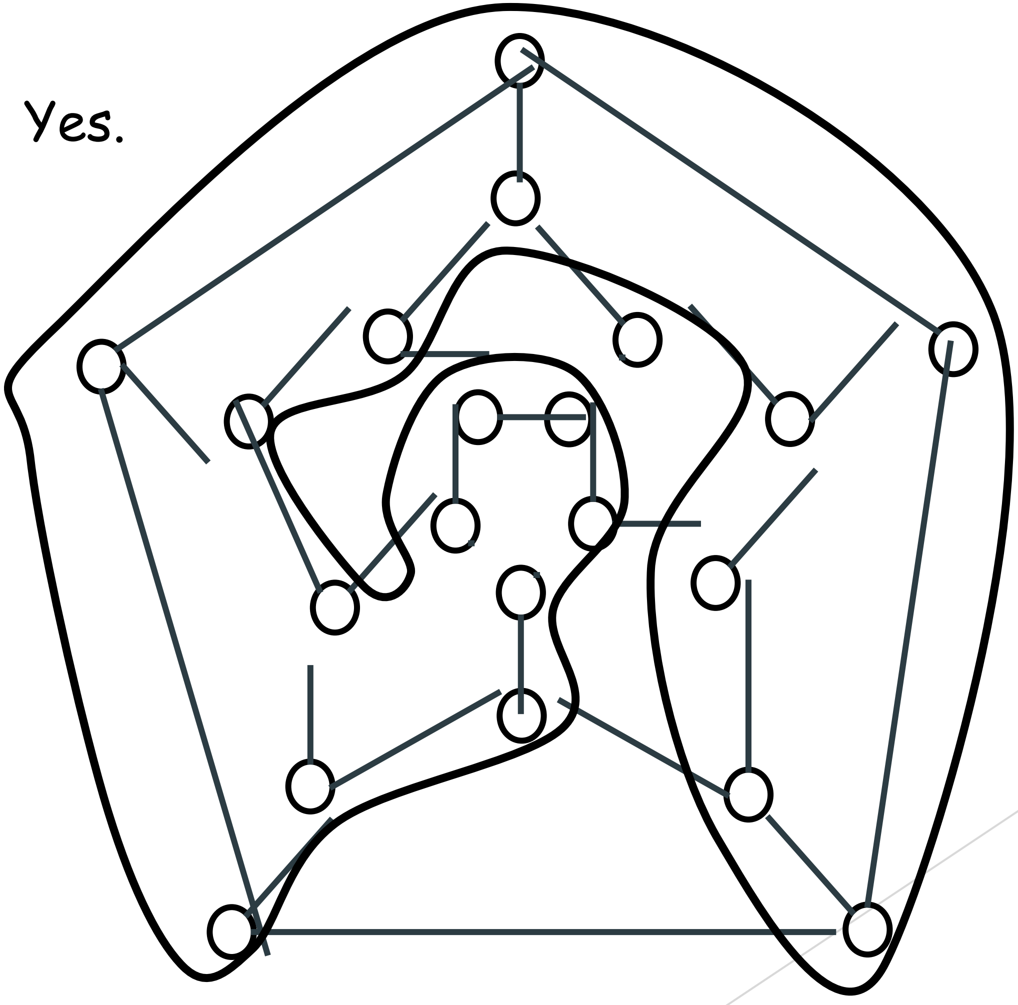
- A graph that contains a Hamiltonian path is called a **traceable graph**.

- 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.

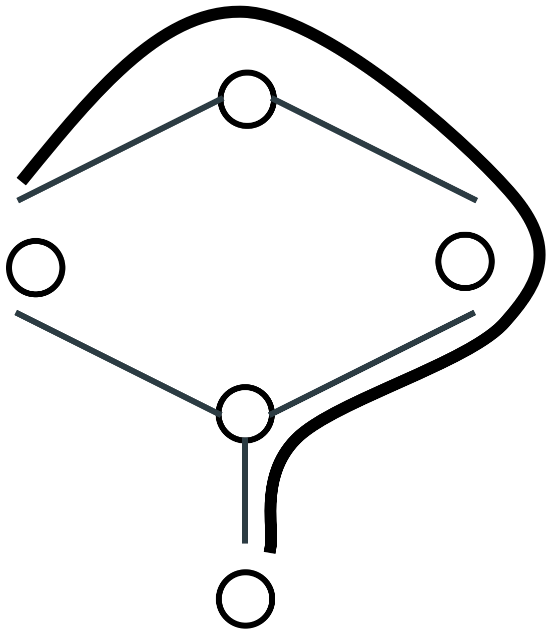
A graph of the vertices of a dodecahedron.

Is it Hamiltonian?

Yes.



Hamiltonian Graph



This one has a Hamiltonian path

Graph Applications

Shortest Path

- Generalize distance to weighted setting
- Digraph $G = (V,E)$ with weight function $W: E \rightarrow R$ (assigning real values to edges)
- Weight of path $p = v_1 \rightarrow v_2 \rightarrow \ldots \rightarrow v_k$ is

$$w(p) = \sum_{i=1}^{k-1} w(v_i, v_{i+1})$$

- Shortest path = a path of the minimum weight
- Applications
  - static/dynamic network routing
  - robot motion planning
  - map/route generation in traffic

Shortest-Path Problems

- Shortest-Path problems
  - **Single-source (single-destination).** Find a shortest path from a given source (vertex *s*) to each of the vertices.
  - **Single-pair.** Given two vertices, find a shortest path between them. Solution to single-source problem solves this problem efficiently, too.
  - **All-pairs.** Find shortest-paths for every pair of vertices. Dynamic programming algorithm.
  - Unweighted shortest-paths – BFS.

Traveling Salesman Problem

- Given a number of cities and the costs of traveling from one to the other, what is the cheapest roundtrip route that visits each city once and then returns to the starting city?

- An equivalent formulation in terms of graph theory is: Find the Hamiltonian cycle with the least weight in a weighted graph.

- It can be shown that the requirement of returning to the starting city does not change the computational complexity of the problem.

- A related problem is the (bottleneck TSP): Find the Hamiltonian cycle in a weighted graph with the minimal length of the longest edge.

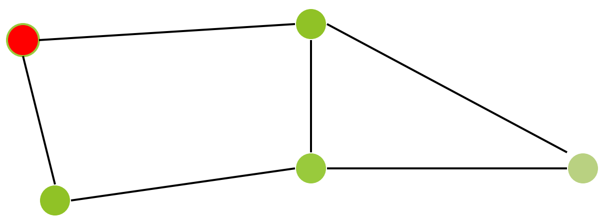
Graph Coloring Problem

- **Graph coloring** is an assignment of *"colors"*, almost always taken to be consecutive integers starting from 1 without loss of generality, to certain objects in a graph. Such objects can be vertices, edges, faces, or a mixture of the above.

- Application examples: scheduling, register allocation in a microprocessor, frequency assignment in mobile radios, and pattern matching

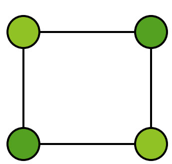
Vertex Coloring Problem

- Assignment of colors to the vertices of the graph such that proper coloring takes place (no two adjacent vertices are assigned the same color)
- **Chromatic number**: least number of colors needed to color the graph
- A graph that can be assigned a (proper) k-coloring is **k-colorable**, and it is **k-chromatic** if its chromatic number is exactly k.

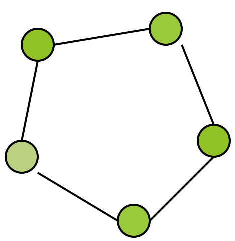


Vertex Coloring Problem

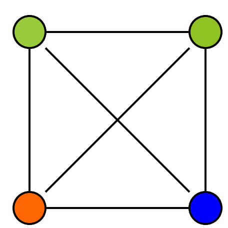
- The problem of finding a minimum coloring of a graph is NP-Hard
- The corresponding decision problem (Is there a coloring which uses at most *k* colors?) is NP-complete
- The chromatic number for C_n = 3 (n is odd) or 2 (n is even), K_n = n, $K_{m,n}$ = 2
- Cn: cycle with n vertices; Kn: fully connected graph with n vertices; Km,n: complete bipartite graph



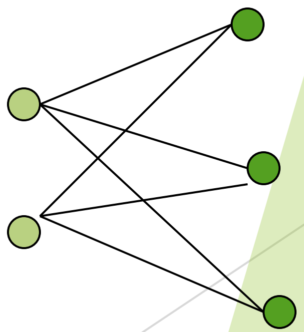
C_4



C_5



K_4



$K_{2,3}$